# PIC18F27/47/57Q43

## 28/40/44/48-Pin, Low-Power, High-Performance Microcontroller with XLP Technology

## Introduction

The PIC18-Q43 microcontroller family is available in 28/40/44/48-pin devices for real-time control applications. This family features a 12-bit Analog-to-Digital Converter with Computation (ADCC) automating Capacitive Voltage Divider (CVD) techniques for advanced capacitive touch sensing, averaging, filtering, oversampling and threshold comparison. This family showcases a new 16-bit Pulse-Width Modulator (PWM) module which provides dual independent outputs on the same time base. Additional features include vectored interrupt controller with fixed latency for handling interrupts, system bus arbiter, Direct Memory Access (DMA) capabilities, Universal Asynchronous Receiver-Transmitter (UART) with support for asynchronous, Digital Multiplex (DMX), Digital Addressable Lighting Interface (DALI) and Local Interconnect Network (LIN) protocols, Serial Peripheral Interface (SPI), $I^2C$, memory features like Memory Access Partition (MAP) to support users in data protection and bootloader applications, and Device Information Area (DIA), which stores factory calibration values to help improve temperature sensor accuracy.

## PIC18-Q43 Family Types

**Table 1. Devices Included in This Data Sheet**

| Device | Program Memory Flash (bytes) | Data SRAM (bytes) | Data EEPROM (bytes) | Memory Access Partition/ Device Information Area | I/O Pins/ Peripheral Pin Select | 8-Bit Timer with HLT/ 16-Bit Timers | 16-Bit Dual PWM/ CCP | Complementary Waveform Generator | Signal Measurement Timer | Numerically Controlled Oscillator | Configurable Logic Cell | 12-Bit ADCC (channels) | 8-Bit DAC | Comparator/ Zero-Cross Detect | High-Low Voltage Detect | SPI/$I^2C$ | UART/ UART with Protocol Support | Direct Memory Access (DMA) | Windowed Watchdog Timer | 16-Bit CRC with Scanner | Vectored Interrupts | Peripheral Module Disable | Temperature Indicator |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PIC18F27Q43 | 128k | 8192 | 1024 | Y/Y | 25/Y | 3/4 | 3/3 | 3 | 1 | 3 | 8 | 24 | 1 | 2/1 | 1 | 2/1 | 4/1 | 6 | Y | Y | Y | Y | Y |
| PIC18F47Q43 | 128k | 8192 | 1024 | Y/Y | 36/Y | 3/4 | 3/3 | 3 | 1 | 3 | 8 | 35 | 1 | 2/1 | 1 | 2/1 | 4/1 | 6 | Y | Y | Y | Y | Y |
| PIC18F57Q43 | 128k | 8192 | 1024 | Y/Y | 44/Y | 3/4 | 3/3 | 3 | 1 | 3 | 8 | 43 | 1 | 2/1 | 1 | 2/1 | 4/1 | 6 | Y | Y | Y | Y | Y |

© 2019-2023 Microchip Technology Inc. and its subsidiaries

**Table 2. Devices not Included in This Data Sheet**

| Device | Program Memory Flash (bytes) | Data SRAM (bytes) | Data EEPROM (bytes) | Memory Access Partition/ Device Information Area | I/O Pins/ Peripheral Pin Select | 8-Bit Timer with HLT/ 16-Bit Timers | 16-Bit Dual PWM/ CCP | Complementary Waveform Generator | Signal Measurement Timer | Numerically Controlled Oscillator | Configurable Logic Cell | 12-Bit ADCC (channels) | 8-Bit DAC | Comparator/ Zero-Cross Detect | High-Low Voltage Detect | SPI/$I^2C$ | UART/ UART with Protocol Support | Direct Memory Access (DMA) | Windowed Watchdog Timer | 16-Bit CRC with Scanner | Vectored Interrupts | Peripheral Module Disable | Temperature Indicator |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| PIC18F25Q43 | 32k | 2048 | 1024 | Y/Y | 25/Y | 3/4 | 3/3 | 3 | 1 | 3 | 8 | 24 | 1 | 2/1 | 1 | 2/1 | 4/1 | 6 | Y | Y | Y | Y | Y |
| PIC18F26Q43 | 64k | 4096 | 1024 | Y/Y | 25/Y | 3/4 | 3/3 | 3 | 1 | 3 | 8 | 24 | 1 | 2/1 | 1 | 2/1 | 4/1 | 6 | Y | Y | Y | Y | Y |
| PIC18F45Q43 | 32k | 2048 | 1024 | Y/Y | 36/Y | 3/4 | 3/3 | 3 | 1 | 3 | 8 | 35 | 1 | 2/1 | 1 | 2/1 | 4/1 | 6 | Y | Y | Y | Y | Y |
| PIC18F46Q43 | 64k | 4096 | 1024 | Y/Y | 36/Y | 3/4 | 3/3 | 3 | 1 | 3 | 8 | 35 | 1 | 2/1 | 1 | 2/1 | 4/1 | 6 | Y | Y | Y | Y | Y |
| PIC18F55Q43 | 32k | 2048 | 1024 | Y/Y | 44/Y | 3/4 | 3/3 | 3 | 1 | 3 | 8 | 43 | 1 | 2/1 | 1 | 2/1 | 4/1 | 6 | Y | Y | Y | Y | Y |
| PIC18F56Q43 | 64k | 4096 | 1024 | Y/Y | 44/Y | 3/4 | 3/3 | 3 | 1 | 3 | 8 | 43 | 1 | 2/1 | 1 | 2/1 | 4/1 | 6 | Y | Y | Y | Y | Y |

## Features

- C Compiler Optimized RISC Architecture
- Operating Speed:
  - DC – 64 MHz clock input
  - 62.5 ns minimum instruction cycle
- Six Direct Memory Access (DMA) Controllers:
  - Data transfers to SFR/GPR spaces from either Program Flash Memory, Data EEPROM or SFR/GPR spaces
  - User programmable source and destination sizes
  - Hardware and software triggered data transfers
- Vectored Interrupt Capability:
  - Selectable high/low priority
  - Fixed interrupt latency of three instruction cycles
  - Programmable vector table base address
  - Backwards compatible with previous interrupt capabilities
- 127-Level Deep Hardware Stack
- Low-Current Power-on Reset (POR)
- Configurable Power-up Timer (PWRT)
- Brown-out Reset (BOR)
- Low-Power BOR (LPBOR) Option
- Windowed Watchdog Timer (WWDT):
  - Watchdog Reset on too long or too short interval between watchdog clear events
  - Variable prescaler selection
  - Variable window size selection

## Memory

- Up to 128 KB of Program Flash Memory
- Up to 8 KB of Data SRAM Memory
- 1024 Bytes Data EEPROM
- Memory Access Partition: The Program Flash Memory can be partitioned into:
  - Application Block
  - Boot Block
  - Storage Area Flash (SAF) Block

© 2019-2023 Microchip Technology Inc. and its subsidiaries

- Programmable Code Protection and Write Protection
- Device Information Area (DIA) Stores:
  - Temperature indicator factory calibrated data
  - Fixed Voltage Reference measurement data
  - Microchip unique identifier
- Device Characteristics Information (DCI) Area Stores:
  - Program/erase row sizes
  - Pin count details
  - EEPROM size
- Direct, Indirect and Relative Addressing modes

## Operating Characteristics

- Operating Voltage Range:
  - 1.8V to 5.5V
- Temperature Range:
  - Industrial: -40°C to 85°C
  - Extended: -40°C to 125°C

## Power-Saving Functionality

- Doze: CPU and Peripherals Running at Different Cycle Rates (typically CPU is lower)
- Idle: CPU Halted While Peripherals Operate
- Sleep: Lowest Power Consumption
- Peripheral Module Disable (PMD):
  - Ability to selectively disable hardware module to minimize active power consumption of unused peripherals
- Low-Power Mode Features:
  - Sleep: < 800 nA typical @ 1.8V
  - Operating Current:
    - 48 µA @ 32 kHz, 3V, typical

## Digital Peripherals

- Three 16-Bit Pulse-Width Modulators (PWM):
  - Dual outputs for each PWM module
  - Integrated 16-bit timer/counter
  - Double-buffered user registers for duty cycles
  - Right/Left/Center/Variable aligned modes of operation
  - Multiple clock and Reset signal selections
- Four 16-Bit Timers (TMR0/1/3/5)
- Three 8-Bit Timers (TMR2/4/6) with Hardware Limit Timer (HLT)
- Eight Configurable Logic Cell (CLC):
  - Integrated combinational and sequential logic
- Three Complementary Waveform Generators (CWG):
  - Rising and falling edge dead-band control
  - Full-bridge, half-bridge, 1-channel drive
  - Multiple signal sources
  - Programmable dead band
  - Fault-shutdown input

© 2019-2023 Microchip Technology Inc. and its subsidiaries

- Three Capture/Compare/PWM (CCP) modules:
  - 16-bit resolution for Capture/Compare modes
  - 10-bit resolution for PWM mode
- Three Numerically Controlled Oscillators (NCO):
  - Generates true linear frequency control and increased frequency resolution
  - Input clock up to 64 MHz
- Signal Measurement Timer (SMT):
  - 24-bit timer/counter with prescaler
  - Several modes of operation like Time-of-Flight, Period and Duty Cycle measurement etc.
- Data Signal Modulator (DSM):
  - Multiplex two carrier clocks, with glitch prevention feature
  - Multiple sources for each carrier
- Programmable CRC with Memory Scan:
  - Reliable data/program memory monitoring for Fail-Safe operation (e.g., Class B)
  - Calculate 16-bit CRC over any portion of Program Flash Memory
- Five UART modules:
  - One module (UART1) supports LIN host and client, DMX mode, DALI gear and device protocols
  - Asynchronous UART, RS-232, RS-485 compatible
  - Automatic and user timed BREAK period generation
  - Automatic checksums
  - Programmable 1, 1.5, and two Stop bits
  - Wake-up on BREAK reception
  - DMA compatible
- Two SPI modules:
  - Configurable length bytes
  - Arbitrary length data packets
  - Transmit-without-receive and receive-without-transmit option
  - Transfer byte counter
  - Separate transmit and receive buffers with 2-byte FIFO and DMA capabilities
- One $I^2C$ module, SMBus, PMBus™ Compatible:
  - Supports Standard-mode (100 kHz), Fast-mode (400 kHz) and Fast-mode plus (1 MHz) modes of operation
  - 7-bit and 10-bit addressing modes with address masking modes
  - Dedicated address, transmit and receive buffers and DMA capabilities
  - Bus collision detection with arbitration
  - Bus time-out detection and handling
  - $I^2C$, SMBus 2.0 and SMBus 3.0, and 1.8V input level selections
  - Separate Transmit and Receive Buffers with 2-byte FIFO and DMA capabilities
  - Multi-Host mode, including self-addressing
- Device I/O Port Features:
  - 25 I/O pins (PIC18F25/26/27Q43)
  - 36 I/O pins (PIC18F45/46/47Q43)
  - 44 I/O pins (PIC18F55/56/57Q43)
  - Individually programmable I/O direction, open-drain, slew rate and weak pull-up control
  - Interrupt-on-change on most pins
  - Three programmable external interrupt pins
- Peripheral Pin Select (PPS):
  - Enables pin mapping of digital I/O

© 2019-2023 Microchip Technology Inc. and its subsidiaries

## Analog Peripherals

- Analog-to-Digital Converter with Computation (ADCC):
  - Up to 140 ksps
  - Up to 43 external channels
  - Automated math functions on input signals:
    - Averaging, filter calculations, oversampling and threshold comparison
  - Operates in Sleep
  - Five internal analog channels
  - Hardware Capacitive Voltage Divider (CVD) Support:
    - Adjustable sample-and-hold capacitor array
    - Guard ring digital output drive
    - Automates touch sampling and reduces software size and CPU usage when touch or proximity sensing is required
- 8-Bit Digital-to-Analog Converter (DAC):
  - Buffered output available on two I/O pins
  - Internal connections to Analog-to-Digital (ADC) and Comparators
- Two Comparators (CMP):
  - Four external inputs
  - Configurable output polarity
  - External output via Peripheral Pin Select
- Zero-Cross Detect (ZCD):
  - Detect when AC signal on pin crosses ground
- Voltage Reference:
  - Fixed Voltage Reference with 1.024V, 2.048V and 4.096V output levels
  - Internal connections to ADC, Comparator and DAC

## Clocking Structure

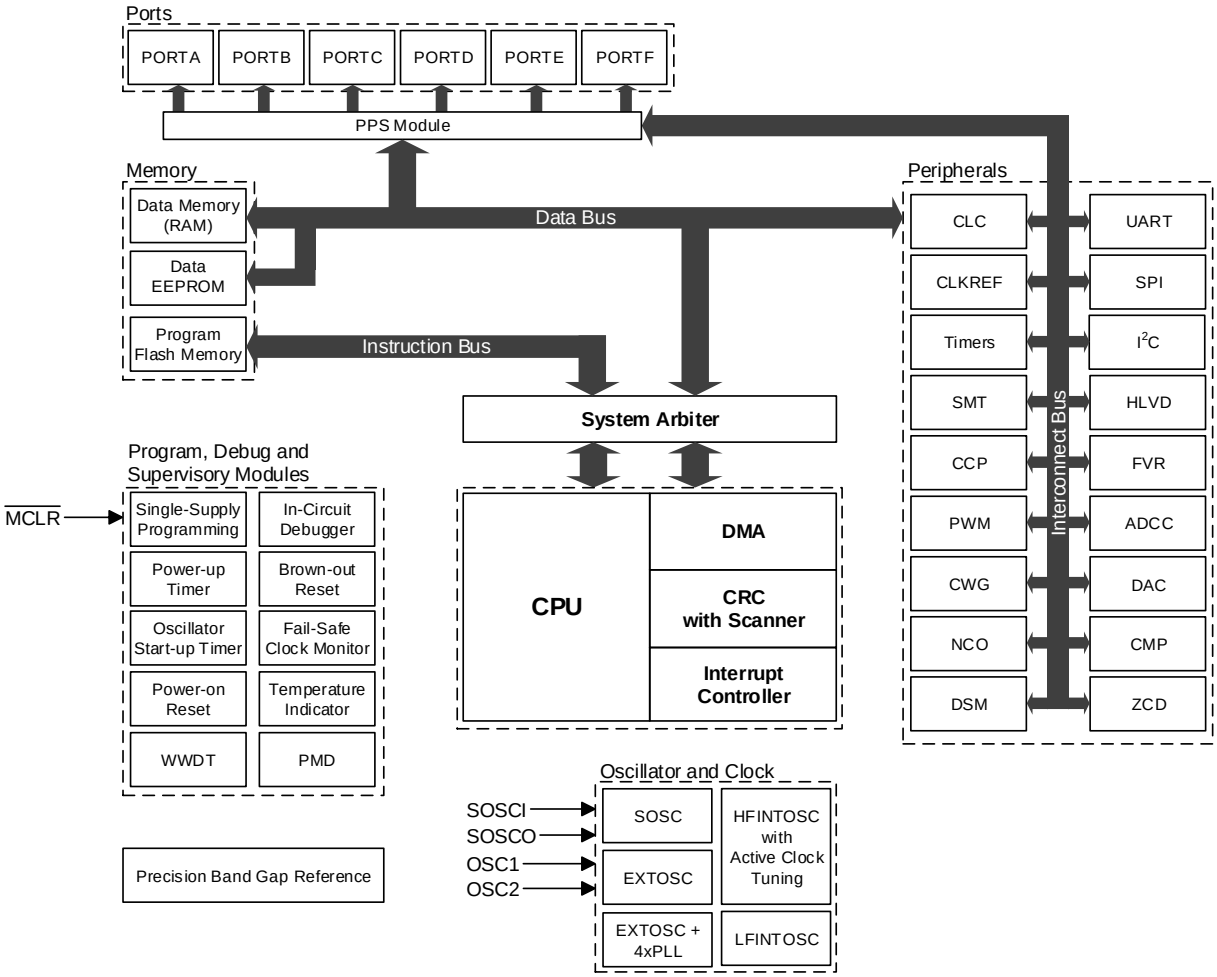
- High-Precision Internal Oscillator Block (HFINTOSC):
  - Selectable frequencies up to 64 MHz
  - ±1% at calibration
  - Active Clock Tuning of HFINTOSC for better accuracy
- 31 kHz Low-Power Internal Oscillator (LFINTOSC)
- External 32 kHz Crystal Oscillator (SOSC)
- External High-frequency Oscillator Block:
  - Three crystal/resonator modes
  - Digital Clock Input mode
  - 4x PLL with external sources
- Fail-Safe Clock Monitor:
  - Allows for operational recovery if external clock stops
- Oscillator Start-up Timer (OST):
  - Ensures stability of crystal oscillator sources

## Programming/Debug Features

- In-Circuit Serial Programming™ (ICSP™) via Two Pins
- In-Circuit Debug (ICD) with Three Breakpoints via Two Pins
- Debug Integrated On-Chip

© 2019-2023 Microchip Technology Inc. and its subsidiaries

## Q43 Block Diagram



# Table of Contents

Introduction....1

PIC18-Q43 Family Types....1

Features....2

1. Packages....9
2. Pin Diagrams....10
3. Pin Allocation Tables....14
4. Guidelines for Getting Started with PIC18-Q43 Microcontrollers....20
5. Register and Bit Naming Conventions....25
6. Register Legend....27
7. PIC18 CPU....28
8. Device Configuration....46
9. Memory Organization....61
10. NVM - Nonvolatile Memory Module....92
11. VIC - Vectored Interrupt Controller Module....117
12. OSC - Oscillator Module (With Fail-Safe Clock Monitor)....193
13. CRC - Cyclic Redundancy Check Module with Memory Scanner....218
14. Resets....235
15. WWDT - Windowed Watchdog Timer....249
16. DMA - Direct Memory Access....260
17. Power-Saving Modes....297
18. PMD - Peripheral Module Disable....305
19. I/O Ports....315
20. IOC - Interrupt-on-Change....331
21. PPS - Peripheral Pin Select Module....337
22. CLC - Configurable Logic Cell....349
23. CLKREF - Reference Clock Output Module....370
24. TMR0 - Timer0 Module....375
25. TMR1 - Timer1 Module with Gate Control....383
26. TMR2 - Timer2 Module....400

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

27. SMT - Signal Measurement Timer ..... 422
28. CCP - Capture/Compare/PWM Module ..... 447
29. Capture, Compare, and PWM Timers Selection ..... 460
30. PWM - Pulse-Width Modulator with Compare ..... 463
31. CWG - Complementary Waveform Generator Module ..... 489
32. NCO - Numerically Controlled Oscillator Module ..... 516
33. DSM - Data Signal Modulator Module ..... 526
34. UART - Universal Asynchronous Receiver Transmitter with Protocol Support ..... 537
35. SPI - Serial Peripheral Interface Module ..... 583
36. $I^2C$ - Inter-Integrated Circuit Module ..... 616
37. HLVD - High/Low-Voltage Detect ..... 698
38. FVR - Fixed Voltage Reference ..... 706
39. Temperature Indicator Module ..... 710
40. ADCC - Analog-to-Digital Converter with Computation Module ..... 715
41. DAC - Digital-to-Analog Converter Module - 8-Bit ..... 761
42. CMP - Comparator Module ..... 766
43. ZCD - Zero-Cross Detection Module ..... 777
44. Instruction Set Summary ..... 784
45. ICSP™ - In-Circuit Serial Programming™ ..... 869
46. Register Summary ..... 872
47. Electrical Specifications ..... 887
48. DC and AC Characteristics Graphs and Tables ..... 918
49. Packaging Information ..... 937
50. Appendix A: Revision History ..... 964

Microchip Information ..... 965

# 1. Packages

**Table 1-1. Packages**

| Device | 28-pin SPDIP | 28-pin SOIC | 28-pin SSOP | 28-pin VQFN 4x4x1 | 40-pin PDIP | 40-pin QFN 5x5x0.9 | 44-pin TQFP | 48-pin TQFP 7x7x1 | 48-pin VQFN 6x6x0.9 |
|---|---|---|---|---|---|---|---|---|---|
| PIC18F25Q43 | ● | ● | ● | ● | | | | | |
| PIC18F26Q43 | ● | ● | ● | ● | | | | | |
| PIC18F27Q43 | ● | ● | ● | ● | | | | | |
| PIC18F45Q43 | | | | | ● | ● | ● | | |
| PIC18F46Q43 | | | | | ● | ● | ● | | |
| PIC18F47Q43 | | | | | ● | ● | ● | | |
| PIC18F55Q43 | | | | | | | | ● | ● |
| PIC18F56Q43 | | | | | | | | ● | ● |
| PIC18F57Q43 | | | | | | | | ● | ● |

# 2. Pin Diagrams

**Figure 2-1.**
**28-Pin SPDIP**

**28-Pin SSOP**

**28-Pin SOIC**

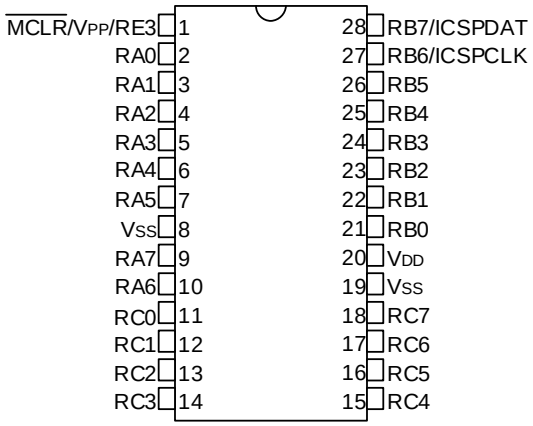
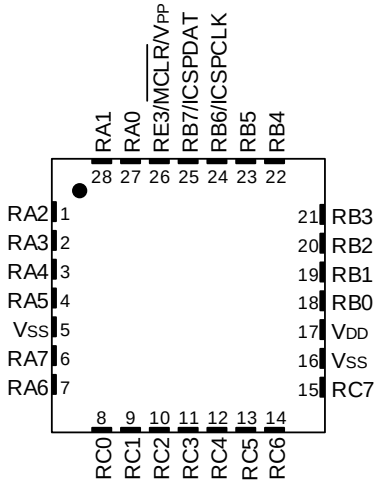



**Figure 2-2.**
**28-Pin VQFN**




**Note:** It is recommended that the exposed bottom pad be connected to $V_{SS}$; however, it must not be the only $V_{SS}$ connection to the device.

**Figure 2-3.**
**40-Pin PDIP**

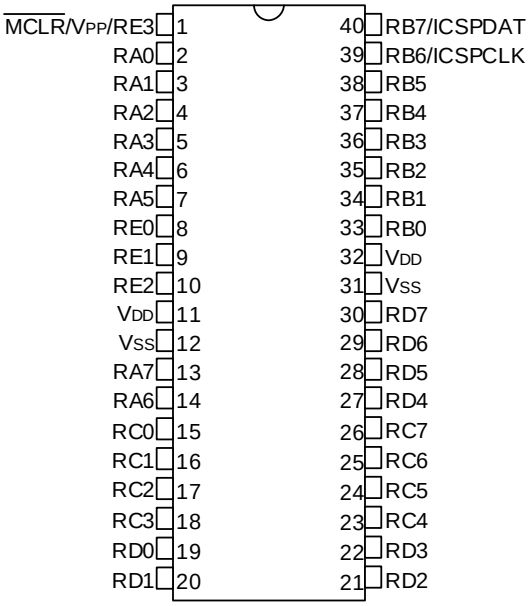
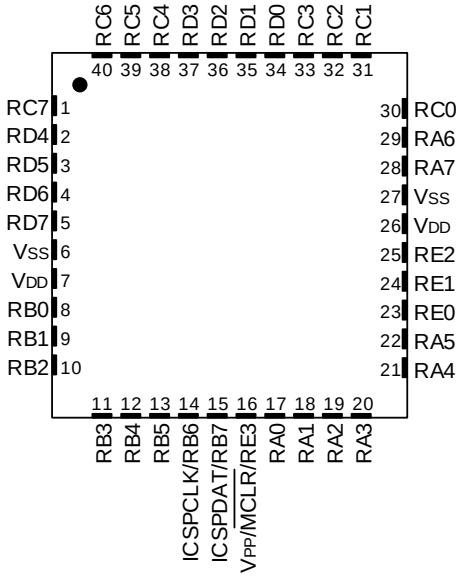



**Figure 2-4.**
**40-Pin QFN**




**Note:** It is recommended that the exposed bottom pad be connected to $V_{SS}$; however, it must not be the only $V_{SS}$ connection to the device.

**Figure 2-5.**
**44-Pin TQFP**

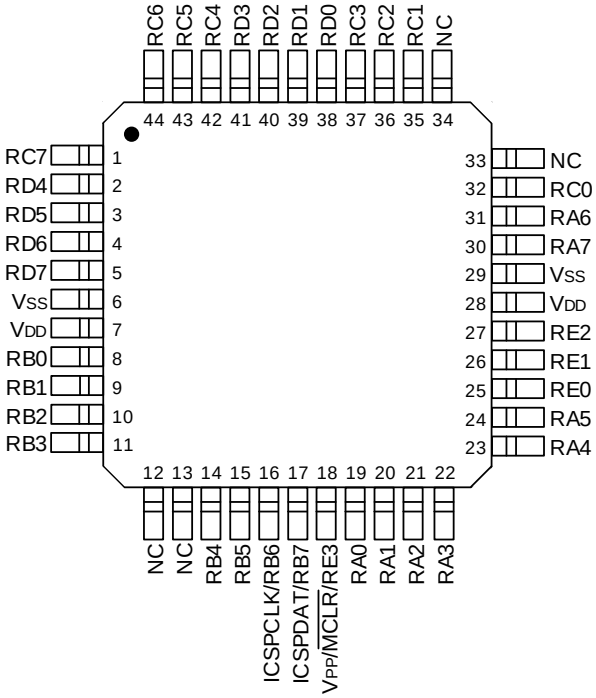
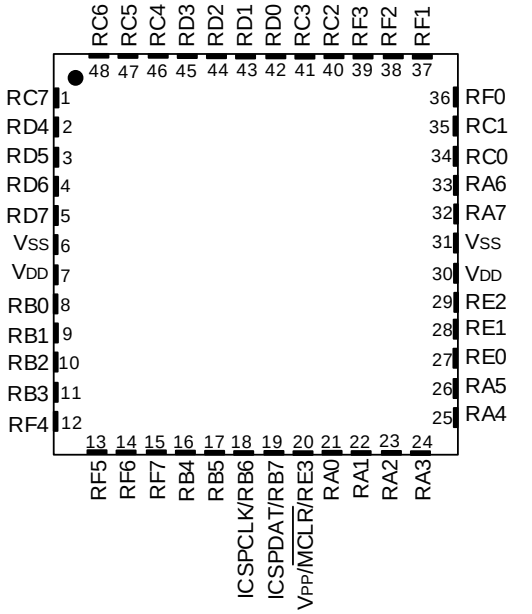
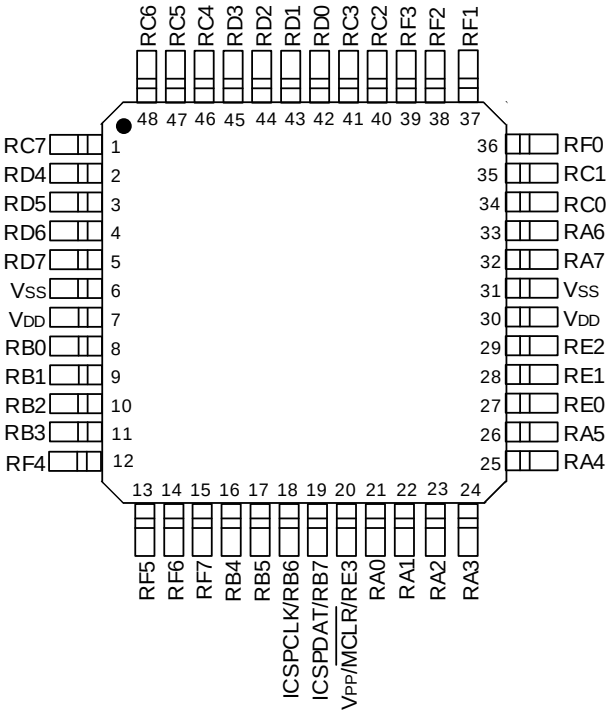



**Figure 2-6.**
**48-Pin VQFN**




**Note:** It is recommended that the exposed bottom pad be connected to $V_{SS}$; however, it must not be the only $V_{SS}$ connection to the device.

**Figure 2-7.**
**48-Pin TQFP**


RC6 RC5 RC4 RD3 RD2 RD1 RD0 RC3 RC2 RF3 RF2 RF1
48 47 46 45 44 43 42 41 40 39 38 37
RC7 1
RD4 2
RD5 3
RD6 4
RD7 5
VSS 6
VDD 7
RB0 8
RB1 9
RB2 10
RB3 11
RF4 12
36 RF0
35 RC1
34 RC0
33 RA6
32 RA7
31 VSS
30 VDD
29 RE2
28 RE1
27 RE0
26 RA5
25 RA4
13 14 15 16 17 18 19 20 21 22 23 24
RF5 RF6 RF7 RB4 RB5 ICSPCLK/RB6 ICSPDAT/RB7 VPP/MCLR/RE3 RA0 RA1 RA2 RA3

# 3. Pin Allocation Tables

**Table 3-1. 28-Pin Allocation Table**

| I/O(2) | 28-Pin SPDIP, SOIC, SSOP | 28-Pin VQFN | A/D | Reference | Comparator | ZCD | Timers/SMT | 16-bit PWM/ CCP | CWG | CLC | SPI | $I^2C$ | UART | DSM | IOC | Interrupt | Basic |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RA0 | 2 | 27 | ANA0 | — | C1IN0-<br>C2IN0- | — | — | — | — | CLCIN0(1)<br>CLCIN4(1) | — | — | — | — | IOCA0 | — | — |
| RA1 | 3 | 28 | ANA1 | — | C1IN1-<br>C2IN1- | — | — | — | — | CLCIN1(1)<br>CLCIN5(1) | — | — | — | — | IOCA1 | — | — |
| RA2 | 4 | 1 | ANA2 | DAC1OUT1<br>VREF- (DAC)<br>VREF- (ADC) | C1IN0+<br>C2IN0+ | — | — | — | — | — | — | — | — | — | IOCA2 | — | — |
| RA3 | 5 | 2 | ANA3 | VREF+ (DAC)<br>VREF+ (ADC) | C1IN1+ | — | — | — | — | — | — | — | — | MDCARL(1) | IOCA3 | — | — |
| RA4 | 6 | 3 | ANA4 | — | — | — | T0CKI(1) | — | — | — | SS2(1) | — | CTS5(1) | MDCARH(1) | IOCA4 | — | — |
| RA5 | 7 | 4 | ANA5 | — | — | — | — | — | — | — | SS1(1) | — | RX5(1) | MDSRC(1) | IOCA5 | — | — |
| RA6 | 10 | 7 | ANA6 | — | — | — | — | — | — | — | — | — | CTS3(1) | — | IOCA6 | — | CLKOUT<br>OSC2 |
| RA7 | 9 | 6 | ANA7 | — | — | — | — | — | — | — | — | — | RX3(1) | — | IOCA7 | — | OSC1<br>CLKIN |
| RB0 | 21 | 18 | ANB0 | — | C2IN1+ | ZCDIN | — | — | CWG1(1) | — | — | — | — | — | IOCB0 | INT0(1) | — |
| RB1 | 22 | 19 | ANB1 | — | C1IN3-<br>C2IN3- | — | — | — | CWG2(1) | — | — | —(4) | — | — | IOCB1 | INT1(1) | — |
| RB2 | 23 | 20 | ANB2 | — | — | — | — | — | CWG3(1) | — | SDI2(1) | —(4) | — | — | IOCB2 | INT2(1) | — |
| RB3 | 24 | 21 | ANB3 | — | C1IN2-<br>C2IN2- | — | — | — | — | — | SCK2(1) | — | — | — | IOCB3 | — | — |
| RB4 | 25 | 22 | ANB4<br>ADACT(1) | — | — | — | T5G(1) | — | — | — | — | — | CTS4(1) | — | IOCB4 | — | — |
| RB5 | 26 | 23 | ANB5 | — | — | — | T1G(1) | CCP3(1) | — | — | — | — | RX4(1) | — | IOCB5 | — | — |
| RB6 | 27 | 24 | ANB6 | — | — | — | — | — | — | CLCIN2(1)<br>CLCIN6(1) | | — | CTS2(1) | — | IOCB6 | — | ICSPCLK |
| RB7 | 28 | 25 | ANB7 | DAC1OUT2 | — | — | T6IN(1) | PWM3ERS(1) | — | CLCIN3(1)<br>CLCIN7(1) | — | — | RX2(1) | — | IOCB7 | — | ICSPDAT |

.....continued

| I/O(2) | 28-Pin SPDIP, SOIC, SSOP | 28-Pin VQFN | A/D | Reference | Comparator | ZCD | Timers/SMT | 16-bit PWM/ CCP | CWG | CLC | SPI | $I^2C$ | UART | DSM | IOC | Interrupt | Basic |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RC0 | 11 | 8 | ANC0 | — | — | — | T1CKI(1)<br>T3CKI(1)<br>T3G(1)<br>SMT1WIN(1) | — | — | — | — | — | — | — | IOCC0 | — | SOSCO |
| RC1 | 12 | 9 | ANC1 | — | — | — | SMT1SIG(1) | CCP2(1) | — | — | — | — | — | — | IOCC1 | — | SOSCIN<br>SOSCI |
| RC2 | 13 | 10 | ANC2 | — | — | — | T5CKI(1) | PWMIN0(1)<br>CCP1(1) | — | — | — | — | — | — | IOCC2 | — | — |
| RC3 | 14 | 11 | ANC3 | — | — | — | T2IN(1) | PWM1ERS(1) | — | — | SCK1(1) | SCL1(3,4) | — | — | IOCC3 | — | — |
| RC4 | 15 | 12 | ANC4 | — | — | — | — | — | — | — | SDI1(1) | SDA(3,4) | — | — | IOCC4 | — | — |
| RC5 | 16 | 13 | ANC5 | — | — | — | T4IN(1) | PWM2ERS(1) | — | — | — | — | — | — | IOCC5 | — | — |
| RC6 | 17 | 14 | ANC6 | — | — | — | — | PWMIN1(1) | — | — | — | — | CTS1(1) | — | IOCC6 | — | — |
| RC7 | 18 | 15 | ANC7 | — | — | — | — | — | — | — | — | — | RX1(1) | — | IOCC7 | — | — |
| RE3 | 1 | 26 | — | — | — | — | — | — | — | — | — | — | — | — | IOCE3 | — | Vpp/$\overline{MCLR}$ |
| VSS | 19 | 16 | — | — | — | — | — | — | — | — | — | — | — | — | — | — | VSS |
| VDD(5) | 20 | 17 | — | — | — | — | — | — | — | — | — | — | — | — | — | — | VDD(5) |
| VSS | 8 | 5 | — | — | — | — | — | — | — | — | — | — | — | — | — | — | VSS |
| OUT(2) | — | — | ADGRDA<br>ADGRDB | — | C1OUT<br>C2OUT | — | TMR0 | PWM11<br>PWM12<br>PWM21<br>PWM22<br>PWM31<br>PWM32<br>CCP1<br>CCP2<br>CCP3 | CWG1A<br>CWG1B<br>CWG1C<br>CWG1D<br>CWG2A<br>CWG2B<br>CWG2C<br>CWG2D<br>CWG3A<br>CWG3B<br>CWG3C<br>CWG3D | CLC1OUT<br>CLC2OUT<br>CLC3OUT<br>CLC4OUT<br>CLC5OUT<br>CLC6OUT<br>CLC7OUT<br>CLC8OUT | SS1<br>SCK1<br>SDO1<br>SS2<br>SCK2<br>SDO2 | SDA1<br>SCL1 | DTR1<br>RTS1<br>TX1<br>DTR2<br>RTS2<br>TX2<br>DTR3<br>RTS3<br>TX3<br>DTR4<br>RTS4<br>TX4<br>DTR5<br>RTS5<br>TX5 | DSM1 | — | — | — |

© 2019-2023 Microchip Technology Inc. and its subsidiaries

**Notes:**

1. This is a PPS remappable input signal. The input function may be moved from the default location shown to one of several other PORTx pins. Refer to the peripheral input selection table for details on which PORT pins may be used for this signal.
2. All output signals shown in this row are PPS remappable. These signals may be mapped to output onto one of several PORTx pin options as described in the peripheral output selection table.
3. This is a bidirectional signal. For normal module operation, the firmware will map this signal to the same pin in both the PPS input and PPS output registers.
4. These pins are configured for $I^2C$ logic levels; The SCLx/SDAx signals may be assigned to any of these pins. PPS assignments to the other pins will operate, but input logic levels will be standard TTL/ST as selected by the INLVL register, instead of the $I^2C$ specific or SMBus input buffer thresholds.
5. A 0.1 uF bypass capacitor to $V_{SS}$ is required on the $V_{DD}$ pin.

© 2019-2023 Microchip Technology Inc. and its subsidiaries

# 3. Pin Allocation Tables

**Table 3-2. 40/44/48-Pin Allocation Table**

| I/O(2) | 40 Pin PDIP | 40 Pin QFN | 44 Pin TQFP | 48 Pin TQFP / VQFN | A/D | Reference | Comparator | ZCD | Timers/SMT | 16-bit PWM/ CCP | CWG | CLC | SPI | I2C | UART | DSM | IOC | Interrupt | Basic |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RA0 | 2 | 17 | 19 | 21 | ANA0 | — | C1IN0-<br>C2IN0- | — | — | — | — | CLCIN0(1)<br>CLCIN4(1) | — | — | — | — | IOCA0 | — | — |
| RA1 | 3 | 18 | 20 | 22 | ANA1 | — | C1IN1-<br>C2IN1- | — | — | — | — | CLCIN1(1)<br>CLCIN5(1) | — | — | — | — | IOCA1 | — | — |
| RA2 | 4 | 19 | 21 | 23 | ANA2 | DAC1OUT1<br>VREF- (DAC)<br>VREF- (ADC) | C1IN0+<br>C2IN0+ | — | — | — | — | — | — | — | — | — | IOCA2 | — | — |
| RA3 | 5 | 20 | 22 | 24 | ANA3 | VREF+ (DAC)<br>VREF+ (ADC) | C1IN1+ | — | — | — | — | — | — | — | — | MDCARL(1) | IOCA3 | — | — |
| RA4 | 6 | 21 | 23 | 25 | ANA4 | — | — | — | T0CKI(1) | — | — | — | SS2(1) | — | CTS5(1) | MDCARH(1) | IOCA4 | — | — |
| RA5 | 7 | 22 | 24 | 26 | ANA5 | — | — | — | — | — | — | — | SS1(1) | — | RX5(1) | MDSRC(1) | IOCA5 | — | — |
| RA6 | 14 | 29 | 31 | 33 | ANA6 | — | — | — | — | — | — | — | — | — | CTS3(1) | — | IOCA6 | — | CLKOUT<br>OSC2 |
| RA7 | 13 | 28 | 30 | 32 | ANA7 | — | — | — | — | — | — | — | — | — | RX3(1) | — | IOCA7 | — | OSC1<br>CLKIN |
| RB0 | 33 | 8 | 8 | 8 | ANB0 | — | C2IN1+ | ZCDIN | — | — | CWG1(1) | — | — | — | — | — | IOCB0 | INT0(1) | — |
| RB1 | 34 | 9 | 9 | 9 | ANB1 | — | C1IN3-<br>C2IN3- | — | — | — | CWG2(1) | — | — | —(4) | — | — | IOCB1 | INT1(1) | — |
| RB2 | 35 | 10 | 10 | 10 | ANB2 | — | — | — | — | — | CWG3(1) | — | SDI2(1) | —(4) | — | — | IOCB2 | INT2(1) | — |
| RB3 | 36 | 11 | 11 | 11 | ANB3 | — | C1IN2-<br>C2IN2- | — | — | — | — | — | SCK2(1) | — | — | — | IOCB3 | — | — |
| RB4 | 37 | 12 | 14 | 16 | ANB4<br>ADACT(1) | — | — | — | T5G(1) | — | — | — | — | — | CTS4(1) | — | IOCB4 | — | — |
| RB5 | 38 | 13 | 15 | 17 | ANB5 | — | — | — | T1G(1) | CCP3(1) | — | — | — | — | RX4(1) | — | IOCB5 | — | — |
| RB6 | 39 | 14 | 16 | 18 | ANB6 | — | — | — | — | — | — | CLCIN2(1)<br>CLCIN6(1) | | — | CTS2(1) | — | IOCB6 | — | ICSPCLK |

© 2019-2023 Microchip Technology Inc. and its subsidiaries

.....continued

| I/O(2) | 40 Pin PDIP | 40 Pin QFN | 44 Pin TQFP | 48 Pin TQFP / VQFN | A/D | Reference | Comparator | ZCD | Timers/SMT | 16-bit PWM/ CCP | CWG | CLC | SPI | I2C | UART | DSM | IOC | Interrupt | Basic |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RB7 | 40 | 15 | 17 | 19 | ANB7 | DAC1OUT2 | — | — | T6IN(1) | PWM3ERS(1) | — | CLCIN3(1) CLCIN7(1) | — | — | RX2(1) | — | IOCB7 | — | ICSPDAT |
| RC0 | 15 | 30 | 32 | 34 | ANC0 | — | — | — | T1CKI(1) T3CKI(1) T3G(1) SMT1WIN(1) | — | — | — | — | — | — | — | IOCC0 | — | SOSCO |
| RC1 | 16 | 31 | 35 | 35 | ANC1 | — | — | — | SMT1SIG(1) | CCP2(1) | — | — | — | — | — | — | IOCC1 | — | SOSCIN SOSCI |
| RC2 | 17 | 32 | 36 | 40 | ANC2 | — | — | — | T5CKI(1) | PWMIN0(1) CCP1(1) | — | — | — | — | — | — | IOCC2 | — | — |
| RC3 | 18 | 33 | 37 | 41 | ANC3 | — | — | — | T2IN(1) | PWM1ERS(1) | — | — | SCK1(1) | SCL1(3,4) | — | — | IOCC3 | — | — |
| RC4 | 23 | 38 | 42 | 46 | ANC4 | — | — | — | — | — | — | — | SDI1(1) | SDA(3,4) | — | — | IOCC4 | — | — |
| RC5 | 24 | 39 | 43 | 47 | ANC5 | — | — | — | T4IN(1) | PWM2ERS(1) | — | — | — | — | — | — | IOCC5 | — | — |
| RC6 | 25 | 40 | 44 | 48 | ANC6 | — | — | — | — | PWMIN1(1) | — | — | — | — | CTS1(1) | — | IOCC6 | — | — |
| RC7 | 26 | 1 | 1 | 1 | ANC7 | — | — | — | — | — | — | — | — | — | RX1(1) | — | IOCC7 | — | — |
| RD0 | 19 | 34 | 38 | 42 | AND0 | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| RD1 | 20 | 35 | 39 | 43 | AND1 | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| RD2 | 21 | 36 | 40 | 44 | AND2 | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| RD3 | 22 | 37 | 41 | 45 | AND3 | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| RD4 | 27 | 2 | 2 | 2 | AND4 | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| RD5 | 28 | 3 | 3 | 3 | AND5 | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| RD6 | 29 | 4 | 4 | 4 | AND6 | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| RD7 | 30 | 5 | 5 | 5 | AND7 | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| RE0 | 8 | 23 | 25 | 27 | ANE0 | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| RE1 | 9 | 24 | 26 | 28 | ANE1 | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| RE2 | 10 | 25 | 27 | 29 | ANE2 | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| RE3 | 1 | 16 | 18 | 20 | — | — | — | — | — | — | — | — | — | — | — | — | IOCE3 | — | Vpp/$\overline{\text{MCLR}}$ |
| RF0 | — | — | — | 36 | ANF0 | — | — | — | — | — | — | — | — | — | — | — | | — | — |
| RF1 | — | — | — | 37 | ANF1 | — | — | — | — | — | — | — | — | — | — | — | | — | — |
| RF2 | — | — | — | 38 | ANF2 | — | — | — | — | — | — | — | — | — | — | — | | — | |
| RF3 | — | — | — | 39 | ANF3 | — | — | — | — | — | — | — | — | — | — | — | | — | |
| RF4 | — | — | — | 12 | ANF4 | — | — | — | — | — | — | — | — | — | — | — | | — | |
| RF5 | — | — | — | 13 | ANF5 | — | — | — | — | — | — | — | — | — | — | — | | — | |

© 2019-2023 Microchip Technology Inc. and its subsidiaries

.....continued

| I/O(2) | 40 Pin PDIP | 40 Pin QFN | 44 Pin TQFP | 48 Pin TQFP / VQFN | A/D | Reference | Comparator | ZCD | Timers/SMT | 16-bit PWM/ CCP | CWG | CLC | SPI | I2C | UART | DSM | IOC | Interrupt | Basic |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RF6 | — | — | — | 14 | ANF6 | — | — | — | — | — | — | — | — | — | — | — | | — | |
| RF7 | — | — | — | 15 | ANF7 | — | — | — | — | — | — | — | — | — | — | — | | — | |
| VSS | 12, 31 | 6, 27 | 6, 29 | 6,31 | — | — | — | — | — | — | — | — | — | — | — | — | — | — | VSS |
| VDD(5) | 11, 32 | 7, 26 | 7, 28 | 7, 30 | — | — | — | — | — | — | — | — | — | — | — | — | — | — | VDD(5) |
| OUT(2) | | | | | ADGRDA<br>ADGRDB | — | C1OUT<br>C2OUT | — | TMR0 | PWM11<br>PWM12<br>PWM21<br>PWM22<br>PWM31<br>PWM32<br>CCP1<br>CCP2<br>CCP3 | CWG1A<br>CWG1B<br>CWG1C<br>CWG1D<br>CWG2A<br>CWG2B<br>CWG2C<br>CWG2D<br>CWG3A<br>CWG3B<br>CWG3C<br>CWG3D | CLC1OUT<br>CLC2OUT<br>CLC3OUT<br>CLC4OUT<br>CLC5OUT<br>CLC6OUT<br>CLC7OUT<br>CLC8OUT | SS1<br>SCK1<br>SDO1<br>SS2<br>SCK2<br>SDO2 | SDA1<br>SCL1 | DTR1<br>RTS1<br>TX1<br>DTR2<br>RTS2<br>TX2<br>DTR3<br>RTS3<br>TX3<br>DTR4<br>RTS4<br>TX4<br>DTR5<br>RTS5<br>TX5 | DSM1 | — | — | — |

**Notes:**

1. This is a PPS remappable input signal. The input function may be moved from the default location shown to one of several other PORTx pins. Refer to the peripheral input selection table for details on which PORT pins may be used for this signal.
2. All output signals shown in this row are PPS remappable. These signals may be mapped to output onto one of several PORTx pin options as described in the peripheral output selection table.
3. This is a bidirectional signal. For normal module operation, the firmware will map this signal to the same pin in both the PPS input and PPS output registers.
4. These pins are configured for $I^2C$ logic levels; The SCLx/SDAx signals may be assigned to any of these pins. PPS assignments to the other pins will operate, but input logic levels will be standard TTL/ST as selected by the INLVL register, instead of the $I^2C$ specific or SMBus input buffer thresholds.
5. A 0.1 uF bypass capacitor to $V_{SS}$ is required on all $V_{DD}$ pins.

© 2019-2023 Microchip Technology Inc. and its subsidiaries

# 4. Guidelines for Getting Started with PIC18-Q43 Microcontrollers

## 4.1 Basic Connection Requirements

Getting started with the PIC18-Q43 family of 8-bit microcontrollers requires attention to a minimal set of device pin connections before proceeding with development.

The following pins must always be connected:

- All $V_{DD}$ and $V_{SS}$ pins (see the Power Supply Pins section)
- $\overline{MCLR}$ pin (see the Master Clear ($\overline{MCLR}$) Pin section)

These pins must also be connected if they are being used in the end application:

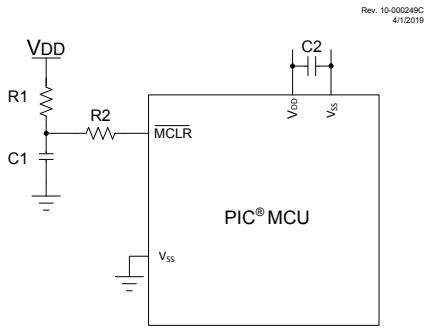
- ICSPCLK/ICSPDAT pins used for In-Circuit Serial Programming™ (ICSP™) and debugging purposes (see the In-Circuit Serial Programming (ICSP) Pins section)
- OSCI and OSCO pins when an external oscillator source is used (see the External Oscillator Pins section)

Additionally, the following pins may be required:

- $V_{REF}+/V_{REF}-$ pins are used when external voltage reference for analog modules is implemented

The minimum mandatory connections are shown in the figure below.

**Figure 4-1. Recommended Minimum Connections**




**Key:**
C1: 0.1 µF, 20V ceramic (recommended)
R1: 10 kΩ (recommended)
R2: 100Ω to 470Ω (recommended)
C2: 0.1 µF, 20V ceramic (required)

## 4.2 Power Supply Pins

### 4.2.1 Decoupling Capacitors

The use of decoupling capacitors on every pair of power supply pins ($V_{DD}$ and $V_{SS}$) is required.

Consider the following criteria when using decoupling capacitors:

- Value and type of capacitor: A 0.1 µF (100 nF), 10-20V capacitor is recommended. The capacitor needs to be a low-ESR device, with a resonance frequency in the range of 200 MHz and higher. Ceramic capacitors are recommended.
- Placement on the printed circuit board: The decoupling capacitors need to be placed as close to the pins as possible. It is recommended to place the capacitors on the same side of the board as the device. If space is constricted, the capacitor can be placed on another layer on the PCB using a via; however, ensure that the trace length from the pin to the capacitor is no greater than 0.25 inch (6 mm).

© 2019-2023 Microchip Technology Inc. and its subsidiaries

- Handling high-frequency noise: If the board is experiencing high-frequency noise (upward of tens of MHz), add a second ceramic type capacitor in parallel to the above described decoupling capacitor. The value of the second capacitor can be in the range of 0.01 µF to 0.001 µF. Place this second capacitor next to each primary decoupling capacitor. In high-speed circuit designs, consider implementing a decade pair of capacitances as close to the power and ground pins as possible (e.g., 0.1 µF in parallel with 0.001 µF).
- Maximizing performance: On the board layout from the power supply circuit, run the power and return traces to the decoupling capacitors first, and then to the device pins. This ensures that the decoupling capacitors are first in the power chain. Equally important is to keep the trace length between the capacitor and the power pins to a minimum, thereby reducing PCB trace inductance.

### 4.2.2 Tank Capacitors

On boards with power traces running longer than six inches in length, it is suggested to use a tank capacitor for integrated circuits, including microcontrollers, to supply a local power source. The value of the tank capacitor will be determined based on the trace resistance that connects the power supply source to the device, and the maximum current drawn by the device in the application. In other words, select the tank capacitor that meets the acceptable voltage sag at the device. Typical values range from 4.7 µF to 47 µF.

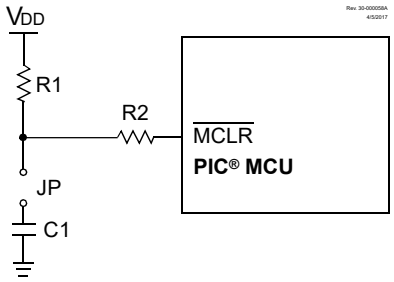
## 4.3 Master Clear ($\overline{MCLR}$) Pin

The $\overline{MCLR}$ pin provides two specific device functions: Device Reset, and Device Programming and Debugging. If programming and debugging are not required in the end application, a direct connection to $V_{DD}$ may be all that is required. The addition of other components, to help increase the application's resistance to spurious Resets from voltage sags, may be beneficial. A typical configuration is shown in Figure 4-1. Other circuit designs may be implemented, depending on the application's requirements.

During programming and debugging, the resistance and capacitance that can be added to the pin must be considered. Device programmers and debuggers drive the $\overline{MCLR}$ pin. Consequently, specific voltage levels ($V_{IH}$ and $V_{IL}$) and fast signal transitions must not be adversely affected. Therefore, specific values of R1 and C1 will need to be adjusted based on the application and PCB requirements. For example, it is recommended that the capacitor, C1, be isolated from the $\overline{MCLR}$ pin during programming and debugging operations by using a jumper (Figure 4-2). The jumper is replaced for normal run-time operations.

Any components associated with the $\overline{MCLR}$ pin need to be placed within 0.25 inch (6 mm) of the pin.

**Figure 4-2. Example of $\overline{MCLR}$ Pin Connections**




**Notes:**

1. R1 ≤ 10 kΩ is recommended. A suggested starting value is 10 kΩ. Ensure that the $\overline{MCLR}$ pin $V_{IH}$ and $V_{IL}$ specifications are met.
2. R2 ≤ 470Ω will limit any current flowing into $\overline{MCLR}$ from the extended capacitor, C1, in the event of $\overline{MCLR}$ pin breakdown, due to Electrostatic Discharge (ESD) or Electrical Overstress (EOS). Ensure that the $\overline{MCLR}$ pin $V_{IH}$ and $V_{IL}$ specifications are met.

## 4.4 In-Circuit Serial Programming™ (ICSP™) Pins

The ICSPCLK and ICSPDAT pins are used for ICSP and debugging purposes. It is recommended to keep the trace length between the ICSP connector and the ICSP pins on the device as short as possible. If the ICSP connector is expected to experience an ESD event, a series resistor is recommended, with the value in the range of a few tens of ohms, not to exceed 100Ω.

© 2019-2023 Microchip Technology Inc. and its subsidiaries

Pull-up resistors, series diodes and capacitors on the ICSPCLK and ICSPDAT pins are not recommended as they can interfere with the programmer/debugger communications to the device. If such discrete components are an application requirement, they need to be removed from the circuit during programming and debugging. Alternatively, refer to the AC/DC characteristics and timing requirements information in the respective device Flash programming specification for information on capacitive loading limits, and pin input voltage high ($V_{IH}$) and input low ($V_{IL}$) requirements.

For device emulation, ensure that the “Communication Channel Select” (i.e., ICSPCLK/ICSPDAT pins), programmed into the device, matches the physical connections for the ICSP to the Microchip debugger/emulator tool.

## 4.5 External Oscillator Pins

Many microcontrollers have options for at least two oscillators: A high-frequency primary oscillator and a low-frequency secondary oscillator.

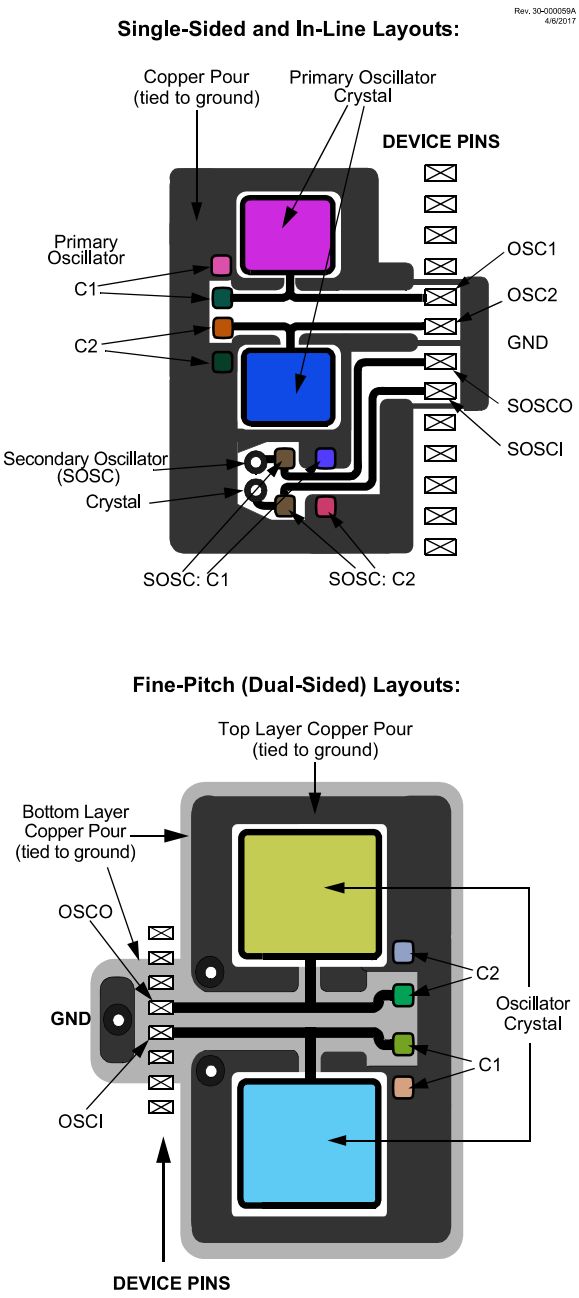
The oscillator circuit needs to be placed on the same side of the board as the device. Place the oscillator circuit close to the respective oscillator pins with no more than 0.5 inch (12 mm) between the circuit components and the pins. The load capacitors have to be placed next to the oscillator itself, on the same side of the board.

Use a grounded copper pour around the oscillator circuit to isolate it from surrounding circuits. The grounded copper pour needs to be routed directly to the MCU ground. Do not run any signal traces or power traces inside the ground pour. Also, if using a two-sided board, avoid any traces on the other side of the board where the crystal is placed.

Layout suggestions are shown in the following figure. In-line packages may be handled with a single-sided layout that completely encompasses the oscillator pins. With fine-pitch packages, it is not always possible to completely surround the pins and components. A suitable solution is to tie the broken guard sections to a mirrored ground layer. In all cases, the guard trace(s) must be returned to ground.

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

**Figure 4-3. Suggested Placement of the Oscillator Circuit**




In planning the application's routing and I/O assignments, ensure that adjacent PORT pins, and other signals in close proximity to the oscillator, are benign (i.e., free of high frequencies, short rise and fall times, and other similar noise).

For additional information and design guidance on oscillator circuits, refer to these Microchip application notes, available at the corporate website (www.microchip.com):

- AN826, *"Crystal Oscillator Basics and Crystal Selection for rfPIC™ and PICmicro® Devices"*
- AN849, *"Basic PICmicro® Oscillator Design"*
- AN943, *"Practical PICmicro® Oscillator Analysis and Design"*
- AN949, *"Making Your Oscillator Work"*

## 4.6 Unused I/Os

Unused I/O pins need to be configured as outputs and driven to a Logic Low state. Alternatively, connect a 1 kΩ to 10 kΩ resistor to $V_{SS}$ on unused pins to drive the output to logic low.

# 5. Register and Bit Naming Conventions

## 5.1 Register Names

When there are multiple instances of the same peripheral in a device, the Peripheral Control registers will be depicted as the concatenation of a peripheral identifier, peripheral instance, and control identifier. The Control registers section will show just one instance of all the register names with an ‘x’ in the place of the peripheral instance number. This naming convention may also be applied to peripherals when there is only one instance of that peripheral in the device to maintain compatibility with other devices in the family that contain more than one.

## 5.2 Bit Names

There are two variants for bit names:

- Short name: Bit function abbreviation
- Long name: Peripheral abbreviation + short name

### 5.2.1 Short Bit Names

Short bit names are an abbreviation for the bit function. For example, some peripherals are enabled with the EN bit. The bit names shown in the registers are the short name variant.

Short bit names are useful when accessing bits in C programs. The general format for accessing bits by the short name is RegisterNamebits.ShortName. For example, the enable bit, ON, in the ADCON0 register can be set in C programs with the instruction `ADCON0bits.ON = 1`.

Short names are not useful in assembly programs because the same name may be used by different peripherals in different bit positions. When it occurs, during the include file generation, the short bit name instances are appended with an underscore plus the name of the register where the bit resides, to avoid naming contentions.

### 5.2.2 Long Bit Names

Long bit names are constructed by adding a peripheral abbreviation prefix to the short name. The prefix is unique to the peripheral, thereby making every long bit name unique. The long bit name for the ADC enable bit is the ADC prefix, AD, appended with the enable bit short name, ON, resulting in the unique bit name ADON.

Long bit names are useful in both C and assembly programs. For example, in C the ADCON0 enable bit can be set with the `ADON = 1` instruction. In assembly, this bit can be set with the `BSF ADCON0,ADON` instruction.

### 5.2.3 Bit Fields

Bit fields are two or more adjacent bits in the same register. Bit fields adhere only to the short bit naming convention. For example, the three Least Significant bits of the ADCON2 register contain the ADC Operating Mode Selection bit. The short name for this field is MD and the long name is ADMD. Bit field access is only possible in C programs. The following example demonstrates a C program instruction for setting the ADC to operate in Accumulate mode:

```
ADCON2bits.MD = 0b001;
```

Individual bits in a bit field can also be accessed with long and short bit names. Each bit is the field name appended with the number of the bit position within the field. For example, the Most Significant MODE bit has the short bit name MD2 and the long bit name is ADMD2. The following two examples demonstrate assembly program sequences for setting the ADC to operate in Accumulate mode:

```
MOVLW  ~(1<<MD2 | 1<<MD1)
ANDWF  ADCON2,F
MOVLW  1<<MD0
IORWF  ADCON2,F
```

```
BCF    ADCON2,ADMD2
BCF    ADCON2,ADMD1
BSF    ADCON2,ADMD0
```

## 5.3 Register and Bit Naming Exceptions

### 5.3.1 Status, Interrupt and Mirror Bits

Status, Interrupt enables, Interrupt flags and Mirror bits are contained in registers that span more than one peripheral. In these cases, the bit name shown is unique so there is no prefix or short name variant.

# 6. Register Legend

**Table 6-1. Register Legend**

| Symbol | Definition |
|---|---|
| R | Readable bit |
| W | Writable bit |
| HS | Hardware settable bit |
| HC | Hardware clearable bit |
| S | Set only bit |
| C | Clear only bit |
| U | Unimplemented bit, read as '0' |
| '1' | Bit value is set |
| '0' | Bit value is cleared |
| x | Bit value is unknown |
| u | Bit value is unchanged |
| q | Bit value depends on condition |
| m | Bit value is predefined |

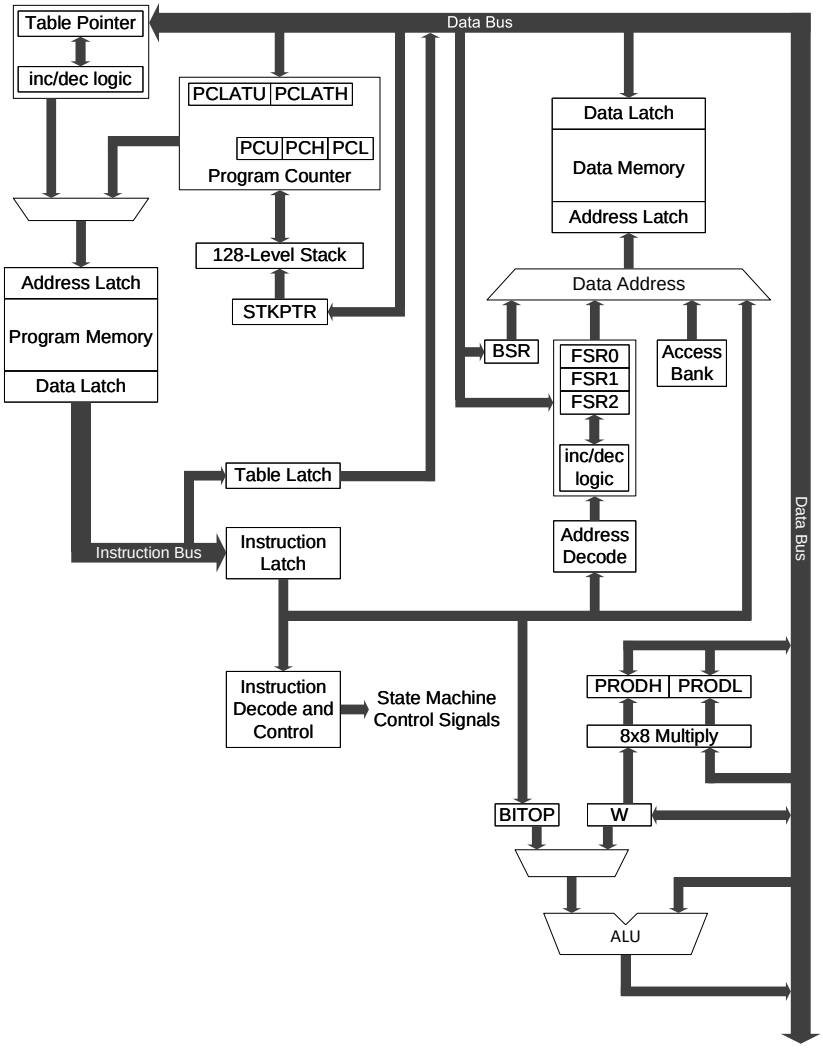
© 2019-2023 Microchip Technology Inc.
and its subsidiaries

# 7. PIC18 CPU

This family of devices contains a PIC18 8-bit CPU core based on the modified Harvard architecture. The PIC18 CPU supports:

- System arbitration which decides memory access allocation depending on user priorities
- Vectored interrupt capability with automatic two-level deep context saving
- 127-level deep hardware stack with overflow and underflow Reset capabilities
- Support Direct, Indirect, and Relative Addressing modes
- 8x8 hardware multiplier

**Figure 7-1. Family Block Diagram**




## 7.1 System Arbitration

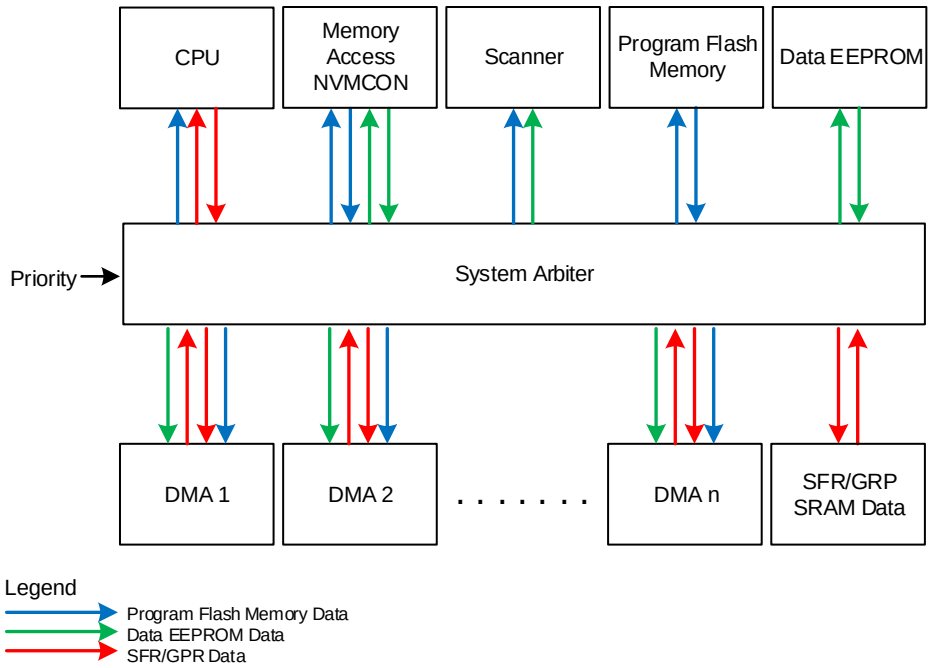
The system arbiter resolves memory access between the system level selections (i.e., Main, Interrupt Service Routine) and peripheral selection (e.g., DMA and Scanner) based on user-assigned priorities. A block diagram of the system arbiter can be found below. Each of the system level and peripheral selections has its own priority selection registers. Memory access priority is resolved using the number written to the corresponding Priority registers, 0 being the highest priority selection and the maximum value being the lowest priority. All system level and peripheral level

selections default to the lowest priority configuration. If the same value is in two or more Priority registers, priority is given to the higher-listed selection according to the following table.

**Table 7-1. Default Priorities**

| Selection | | Priority Register Reset Value |
|---|---|---|
| System Level | ISR | 7 |
| | MAIN | 7 |
| Peripheral | DMA1 | 7 |
| | DMA2 | 7 |
| | DMA3 | 7 |
| | DMA4 | 7 |
| | DMA5 | 7 |
| | DMA6 | 7 |
| | SCANNER | 7 |

**Figure 7-2. System Arbiter Block Diagram**




### 7.1.1 Priority Lock

The system arbiter grants memory access to the peripheral selections (DMAx, Scanner) as long as the PRLOCKED bit is set. Priority selections are locked by setting the PRLOCKED bit. Setting and clearing this bit requires a special sequence as an extra precaution against inadvertent changes. The following code examples demonstrate the Priority Lock and Priority Unlock sequences.

**Example 7-1. Priority Lock Sequence**

```
INTCON0bits.GIE = 0;          // Disable Interrupts;
PRLOCK = 0x55;
PRLOCK = 0xAA;
```

```
PRLOCKbits.PRLOCKED = 1;     // Grant memory access to peripherals;
INTCON0bits.GIE = 1;         // Enable Interrupts;
```

**Example 7-2. Priority Unlock Sequence**

```
INTCON0bits.GIE = 0;         // Disable Interrupts;
PRLOCK = 0x55;
PRLOCK = 0xAA;
PRLOCKbits.PRLOCKED = 0;     // Allow changing priority settings;
INTCON0bits.GIE = 1;         // Enable Interrupts;
```

## 7.2 Memory Access Scheme

The user can assign priorities to both system level and peripheral selections based on which the system arbiter grants memory access. Consider the following priority scenarios between ISR, MAIN and peripherals.

### 7.2.1 ISR Priority > Main Priority > Peripheral Priority

When the peripheral priority (e.g., DMA, Scanner) is lower than ISR and MAIN priority, and the peripheral requires:

1. Access to the Program Flash Memory, then the peripheral waits for an instruction cycle in which the CPU does not need to access the PFM (such as a branch instruction) and uses that cycle to do its own Program Flash Memory access, unless a PFM Read/Write operation is in progress.
2. Access to the SFR/GPR, then the peripheral waits for an instruction cycle in which the CPU does not need to access the SFR/GPR (such as `MOVLW`, `CALL`, `NOP`) and uses that cycle to do its own SFR/GPR access.
3. Access to the Data EEPROM, then the peripheral has access to Data EEPROM unless a Data EEPROM Read/Write operation is being performed.

This results in the lowest throughput for the peripheral to access the memory, and does so without any impact on execution times.

### 7.2.2 Peripheral Priority > ISR Priority > Main Priority

When the peripheral priority (DMA, Scanner) is higher than ISR and MAIN priority, the CPU operation is stalled when the peripheral requests memory. The CPU is held in its current state until the peripheral completes its operation. This results in the highest throughput for the peripheral to access the memory, but has the cost of stalling other execution while it occurs.

### 7.2.3 ISR Priority > Peripheral Priority > Main Priority

In this case, interrupt routines and peripheral operation (DMAx, Scanner) will stall the Main loop. Interrupt will preempt peripheral operation, which results in lowest interrupt latency.

### 7.2.4 Peripheral 1 Priority > ISR Priority > Main Priority > Peripheral 2 Priority

In this case, the Peripheral 1 will stall the execution of the CPU. However, Peripheral 2 can access the memory in cycles unused by Peripheral 1, ISR and the Main Routine.

## 7.3 8x8 Hardware Multiplier

This device includes an 8x8 hardware multiplier as part of the ALU within the CPU. The multiplier performs an unsigned operation and yields a 16-bit result that is stored in the product register, PROD. The multiplier's operation does not affect any flags in the STATUS register.

Making multiplication a hardware operation allows it to be completed in a single instruction cycle. This has the advantages of higher computational throughput and reduced code size for multiplication algorithms and allows the device to be used in many applications previously reserved for digital signal processors. A comparison of various hardware and software multiply operations, along with the savings in memory and execution time, is shown in Table 7-2.

**Table 7-2. Performance Comparison for Various Multiply Operations**

| Routine | Multiply Method | Program Memory (Words) | Cycles (Max) | Time | | | |
|---|---|---|---|---|---|---|---|
| | | | | @ 64 MHz | @ 40 MHz | @ 10 MHz | @ 4 MHz |
| 8x8 unsigned | Without hardware multiply | 13 | 69 | 4.3 µs | 6.9 µs | 27.6 µs | 69 µs |
| | Hardware multiply | 1 | 1 | 62.5 ns | 100 ns | 400 ns | 1 µs |
| 8x8 signed | Without hardware multiply | 33 | 91 | 5.7 µs | 9.1 µs | 36.4 µs | 91 µs |
| | Hardware multiply | 6 | 6 | 375 ns | 600 ns | 2.4 µs | 6 µs |
| 16x16 unsigned | Without hardware multiply | 21 | 242 | 15.1 µs | 24.2 µs | 96.8 µs | 242 µs |
| | Hardware multiply | 28 | 28 | 1.8 µs | 2.8 µs | 11.2 µs | 28 µs |
| 16x16 signed | Without hardware multiply | 52 | 254 | 15.9 µs | 25.4 µs | 102.6 µs | 254 µs |
| | Hardware multiply | 35 | 40 | 2.5 µs | 4.0 µs | 16.0 µs | 40 µs |

### 7.3.1 Operation

Example 7-3 shows the instruction sequence for an 8x8 unsigned multiplication. Only one instruction is required when one of the arguments is already loaded in the WREG register. Example 7-4 shows the sequence to do an 8x8 signed multiplication. To account for the sign bits of the arguments, each argument's Most Significant bit (MSb) is tested and the appropriate subtractions are done.

**Example 7-3. 8x8 Unsigned Multiply Routine**

```
MOVF    ARG1, W   ;

MULWF   ARG2      ; ARG1 * ARG2 -> PRODH:PRODL
```

**Example 7-4. 8x8 Signed Multiply Routine**

```
MOVF    ARG1, W
MULWF   ARG2       ; ARG1 * ARG2 -> PRODH:PRODL
BTFSC   ARG2, SB   ; Test Sign Bit
SUBWF   PRODH, F   ; PRODH = PRODH - ARG1
MOVF    ARG2, W
BTFSC   ARG1, SB   ; Test Sign Bit
SUBWF   PRODH, F   ; PRODH = PRODH - ARG2
```

### 7.3.2 16x16 Unsigned Multiplication Algorithm

Example 7-6 shows the sequence to do a 16x16 unsigned multiplication. Example 7-5 shows the algorithm that is used. The 32-bit result is stored in four registers.

**Example 7-5. 16x16 Unsigned Multiply Algorithm**

$$RES3:RES0 = ARG1H:ARG1L \bullet ARG2H:ARG2L = \left(ARG1H \bullet ARG2H \bullet 2^{16}\right) + \left(ARG1H \bullet ARG2L \bullet 2^{8}\right) + \left(ARG1L \bullet ARG2H \bullet 2^{8}\right) + \left(ARG1L \bullet ARG2L\right)$$

**Example 7-6. 16x16 Unsigned Multiply Routine**

```
        MOVF    ARG1L, W
        MULWF   ARG2L           ; ARG1L * ARG2L → PRODH:PRODL
        MOVFF   PRODH, RES1     ;
        MOVFF   PRODL, RES0     ;
;
        MOVF    ARG1H, W        ;
        MULWF   ARG2H           ; ARG1H * ARG2H → PRODH:PRODL
        MOVFF   PRODH, RES3     ;
        MOVFF   PRODL, RES2     ;
;
        MOVF    ARG1L, W
        MULWF   ARG2H           ; ARG1L * ARG2H → PRODH:PRODL
        MOVF    PRODL, W        ;
        ADDWF   RES1, F         ; Add cross products
        MOVF    PRODH, W        ;
        ADDWFC  RES2, F         ;
        CLRF    WREG            ;
        ADDWFC  RES3, F         ;
;
        MOVF    ARG1H, W        ;
        MULWF   ARG2L           ; ARG1H * ARG2L → PRODH:PRODL
        MOVF    PRODL, W        ;
        ADDWF   RES1, F         ; Add cross products
        MOVF    PRODH, W        ;
        ADDWFC  RES2, F         ;
        CLRF    WREG            ;
        ADDWFC  RES3, F         ;
```

### 7.3.3 16x16 Signed Multiplication Algorithm

Example 7-8 shows the sequence to do a 16x16 signed multiply. Example 7-7 shows the algorithm used. The 32-bit result is stored in four registers. To account for the sign bits of the arguments, the MSb for each argument pair is tested and the appropriate subtractions are done.

**Example 7-7. 16x16 Signed Multiply Algorithm**

$$RES3:RES0 = ARG1H:ARG1L \bullet ARG2H:ARG2L = \left(ARG1H \bullet ARG2H \bullet 2^{16}\right) + \left(ARG1H \bullet ARG2L \bullet 2^{8}\right) + \left(ARG1L \bullet ARG2H \bullet 2^{8}\right) + \left(ARG1L \bullet ARG2L\right) + \left(-1 \bullet ARG2H<7> \bullet ARG1H:ARG1L \bullet 2^{16}\right) + \left(-1 \bullet ARG1H<7> \bullet ARG2H:ARG2L \bullet 2^{16}\right)$$

**Example 7-8. 16x16 Signed Multiply Routine**

```
        MOVF    ARG1L, W
        MULW    ARG2L           ; ARG1L * ARG2L → PRODH:PRODL
        MOVF    PRODH, RES1     ;
        MOVFF   PRODL, RES0     ;
;
        MOVF    ARG1H, W
        MULWF   ARG2H           ; ARG1H * ARG2H → PRODH:PRODL
        MOVFF   PRODH, RES3     ;
        MOVFF   PRODL, RES2     ;
;
        MOVF    ARG1L, W
        MULWF   ARG2H           ; ARG1L * ARG2H → PRODH:PRODL
        MOVF    PRODL, W        ;
        ADDWF   RES1, F         ; Add cross products
        MOVF    PRODH, W        ;
        ADDWFC  RES2, F         ;
        CLRF    WREG            ;
        ADDWFC  RES3, F         ;
;
        MOVF    ARG1H, W        ;
        MULWF   ARG2L           ; ARG1H * ARG2L → PRODH:PRODL
```

```
        MOVF    PRODL, W        ;
        ADDWF   RES1, F         ; Add cross products
        MOVF    PRODH, W        ;
        ADDWFC  RES2, F         ;
        CLRF    WREG            ;
        ADDWFC  RES3, F         ;
;
        BTFSS   ARG2H, 7        ; ARG2H:ARG2L neg?
        BRA     SIGN_ARG1       ; no, check ARG1
        MOVF    ARG1L, W        ;
        SUBWF   RES2            ;
        MOVF    ARG1H, W        ;
        SUBWFB  RES3
;

SIGN_ARG1:
        BTFSS   ARG1H, 7        ; ARG1H:ARG1L neg?
        BRA     CONT_CODE       ; no, done
        MOVF    ARG2L, W        ;
        SUBWF   RES2            ;
        MOVF    ARG2H, W        ;
        SUBWFB  RES3
;
CONT_CODE:
          :
```

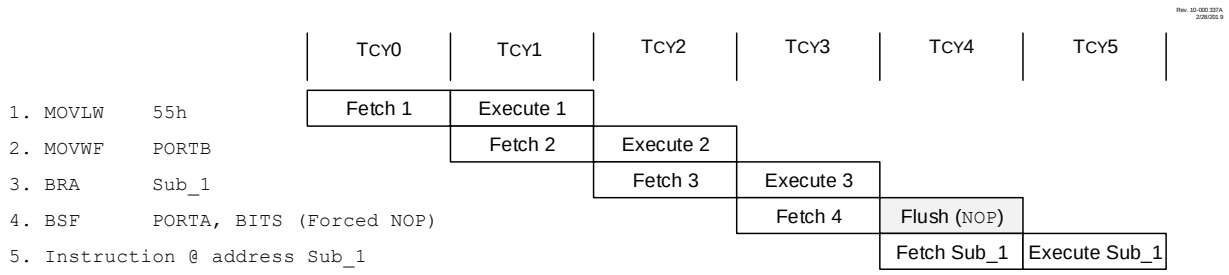
## 7.4 PIC18 Instruction Cycle

### 7.4.1 Instruction Flow/Pipelining

An “Instruction Cycle” consists of four cycles of the oscillator clock. The instruction fetch and execute are pipelined in such a manner that a fetch takes one instruction cycle, while the decode and execute take another instruction cycle. However, due to the pipelining, each instruction effectively executes in one cycle. If an instruction causes the Program Counter (PC) to change (e.g., `GOTO`), then two cycles are required to complete the instruction (Figure 7-3).

A fetch cycle begins with the Program Counter (PC) incrementing followed by the execution cycle. In the execution cycle, the fetched instruction is latched onto the Instruction Register (IR). This instruction is then decoded and executed during the next few oscillator clock cycles. Data memory is read (operand read) and written (destination write) during the execution cycle as well.

**Figure 7-3. Instruction Pipeline Flow**



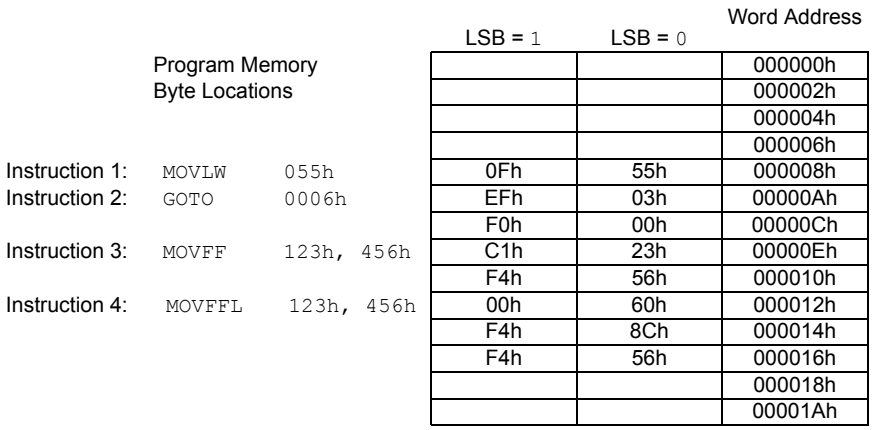

**Note:** There are some instructions that take multiple cycles to execute. Refer to the **“Instruction Set Summary”** section for details.

### 7.4.2 Instructions in Program Memory

The program memory is addressed in bytes. Instructions are stored as either two bytes, four bytes, or six bytes in program memory. The Least Significant Byte of an instruction word is always stored in a program memory location with an even address (LSb = `0`). To maintain alignment with instruction boundaries, the PC increments in steps of two and the LSb will always read ‘`0`’. See the **“Program Counter”** section in the **“Memory Organization”** chapter for more details. The instructions in the Program Memory figure below shows how instruction words are stored in the program memory.

The CALL and GOTO instructions have the absolute program memory address embedded into the instruction. Since instructions are always stored on word boundaries, the data contained in the instruction is a word address. The word address is written to the corresponding bits of the Program Counter register, which accesses the desired byte address in program memory. Instruction #2 in the example shows how the instruction GOTO 0006h is encoded in the program memory. Program branch instructions, which encode a relative address offset, operate in the same manner. The offset value stored in a branch instruction represents the number of single-word instructions that the PC will be offset by.

**Figure 7-4. Instructions in Program Memory**




### 7.4.3 Multi-Word Instructions

The standard PIC18 instruction set has six two-word instructions: CALL, MOVFF, GOTO, LFSR, MOVSF and MOVSS and two three-word instructions: MOVFFL and MOVSFL. In all cases, the second and the third word of the instruction always has 1111 as its four Most Significant bits; the other 12 bits are literal data, usually a data memory address.

The use of 1111 in the four MSbs of an instruction specifies a special form of NOP. If the instruction is executed in proper sequence, immediately after the first word, the data in the second word is accessed and used by the instruction sequence. If the first word is skipped for some reason and the second word is executed by itself, a NOP is executed instead. This is necessary for cases when the two-word instruction is preceded by a conditional instruction that changes the PC.

Table 7-3 and Table 7-4 show more details of how two-word instructions work. Table 7-5 and Table 7-6 show more details of how three-word instructions work.

**Important:** See the **"PIC18 Instruction Execution and the Extended Instruction Set"** section for information on two-word instructions in the extended instruction set.

**Table 7-3. Two-Word Instructions (Case 1)**

| Object Code | Source Code | Comment |
|---|---|---|
| 0110 0110 0000 0000 | TSTFSZ REG1 | ; is RAM location 0? |
| 1100 0001 0101 0011 | MOVFF REG1,REG2 | ; No, skip this word |
| 1111 0100 0101 0110 | | ; Execute this word as NOP |
| 0010 0100 0000 0000 | ADDWF REG3 | ; continue code |

**Table 7-4. Two-Word Instructions (Case 2)**

| Object Code | Source Code | Comment |
|---|---|---|
| 0110 0110 0000 0000 | TSTFSZ REG1 | ; is RAM location 0? |
| 1100 0001 0101 0011 | MOVFF REG1,REG2 | ; Yes, execute this word |
| 1111 0100 0101 0110 | | ; 2nd word of instruction |
| 0010 0100 0000 0000 | ADDWF REG3 | ; continue code |

**Table 7-5. Three-Word Instructions (Case 1)**

| Object Code | Source Code | Comment |
|---|---|---|
| 0110 0110 0000 0000 | TSTFSZ REG1 | ; is RAM location 0? |
| 0000 0000 0110 0000 | MOVFFL REG1,REG2 | ; Yes, skip this word |
| 1111 0100 1000 1100 | | ; Execute this word as NOP |
| 1111 0100 0101 0110 | | ; Execute this word as NOP |
| 0010 0100 0000 0000 | ADDWF REG3 | ; continue code |

**Table 7-6. Three-Word Instructions (Case 2)**

| Object Code | Source Code | Comment |
|---|---|---|
| 0110 0110 0000 0000 | TSTFSZ REG1 | ; is RAM location 0? |
| 0000 0000 0110 0000 | MOVFFL REG1,REG2 | ; No, execute this word |
| 1111 0100 1000 1100 | | ; 2nd word of instruction |
| 1111 0100 0101 0110 | | ; 3rd word of instruction |
| 0010 0100 0000 0000 | ADDWF REG3 | ; continue code |

## 7.5 STATUS Register

The STATUS register contains the arithmetic status of the ALU. As with any other SFR, it can be the operand for any instruction. If the STATUS register is the destination for an instruction that affects the Z, DC, C, OV or N bits, the results of the instruction are not written; instead, the STATUS register is updated according to the instruction performed. Therefore, the result of an instruction with the STATUS register as its destination may be different than intended. As an example, `CLRF STATUS` will set the Z bit and leave the remaining Status bits unchanged (‘`000u u1uu`’).

It is recommended that only `BCF`, `BSF`, `SWAPF`, `MOVFF` and `MOVWF` instructions are used to alter the STATUS register, because these instructions do not affect the Z, C, DC, OV or N bits in the STATUS register. For other instructions that do not affect Status bits, see the instruction set summaries.

**Important:** The C and DC bits operate as the $\overline{\text{Borrow}}$ and $\overline{\text{Digit Borrow}}$ bits, respectively, in subtraction.

## 7.6 Call Shadow Register

When `CALL` instruction is used, the WREG, BSR and STATUS are automatically saved in hardware and can be accessed using the WREG_CSHAD, BSR_CSHAD and STATUS_CSHAD registers.

**Important:**
The contents of these registers need to be handled correctly to avoid erroneous code execution.

## 7.7 Register Definitions: System Arbiter

### 7.7.1 ISRPR

**Name:** ISRPR
**Address:** 0x0BF

Interrupt Service Routine Priority Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | PR[2:0] | | |
| Access | | | | | | R/W | R/W | R/W |
| Reset | | | | | | 1 | 1 | 1 |

**Bits 2:0 – PR[2:0]** Interrupt Service Routine Priority Selection

| Value | Description |
|---|---|
| 111 | System Arbiter Priority Level: 7 (Lowest Priority) |
| 110 | System Arbiter Priority Level: 6 |
| 101 | System Arbiter Priority Level: 5 |
| 100 | System Arbiter Priority Level: 4 |
| 011 | System Arbiter Priority Level: 3 |
| 010 | System Arbiter Priority Level: 2 |
| 001 | System Arbiter Priority Level: 1 |
| 000 | System Arbiter Priority Level: 0 (Highest Priority) |

### 7.7.2 MAINPR

**Name:** MAINPR
**Address:** 0x0BE

Main Routine Priority Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | PR[2:0] | | |
| Access | | | | | | R/W | R/W | R/W |
| Reset | | | | | | 1 | 1 | 1 |

**Bits 2:0 – PR[2:0]** Main Routine Priority Selection

| Value | Description |
|---|---|
| 111 | System Arbiter Priority Level: 7 (Lowest Priority) |
| 110 | System Arbiter Priority Level: 6 |
| 101 | System Arbiter Priority Level: 5 |
| 100 | System Arbiter Priority Level: 4 |
| 011 | System Arbiter Priority Level: 3 |
| 010 | System Arbiter Priority Level: 2 |
| 001 | System Arbiter Priority Level: 1 |
| 000 | System Arbiter Priority Level: 0 (Highest Priority) |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

### 7.7.3 DMAxPR

**Name:** DMAxPR
**Address:** 0x0B6,0x0B7,0x0B8,0x0B9,0x0BA,0x0BB

DMAx Priority Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | PR[2:0] | | |
| Access | | | | | | R/W | R/W | R/W |
| Reset | | | | | | 1 | 1 | 1 |

**Bits 2:0 – PR[2:0]** DMAx Priority Selection

| Value | Description |
|---|---|
| 111 | System Arbiter Priority Level: 7 (Lowest Priority) |
| 110 | System Arbiter Priority Level: 6 |
| 101 | System Arbiter Priority Level: 5 |
| 100 | System Arbiter Priority Level: 4 |
| 011 | System Arbiter Priority Level: 3 |
| 010 | System Arbiter Priority Level: 2 |
| 001 | System Arbiter Priority Level: 1 |
| 000 | System Arbiter Priority Level: 0 (Highest Priority) |

### 7.7.4 SCANPR

**Name:** SCANPR
**Address:** 0x0B5

Scanner Priority Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | PR[2:0] | | |
| Access | | | | | | R/W | R/W | R/W |
| Reset | | | | | | 1 | 1 | 1 |

**Bits 2:0 – PR[2:0]** Scanner Priority Selection

| Value | Description |
|---|---|
| 111 | System Arbiter Priority Level: 7 (Lowest Priority) |
| 110 | System Arbiter Priority Level: 6 |
| 101 | System Arbiter Priority Level: 5 |
| 100 | System Arbiter Priority Level: 4 |
| 011 | System Arbiter Priority Level: 3 |
| 010 | System Arbiter Priority Level: 2 |
| 001 | System Arbiter Priority Level: 1 |
| 000 | System Arbiter Priority Level: 0 (Highest Priority) |

### 7.7.5 PRLOCK

**Name:** PRLOCK
**Address:** 0x0B4

Priority Lock Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | PRLOCKED |
| Access | | | | | | | | R/W |
| Reset | | | | | | | | 0 |

**Bit 0 – PRLOCKED** PR Register Lock

| Value | Description |
|---|---|
| 1 | Priority registers are locked and cannot be written; Peripherals have access to the memory |
| 0 | Priority registers can be modified by write operations; Peripherals do not have access to the memory |

**Important:**

1. The PRLOCKED bit can only be set or cleared after the unlock sequence.
2. If the Configuration Bit PR1WAY = 1, the PRLOCKED bit cannot be cleared after it has been set. A device Reset will clear the bit and allow one more set.

### 7.7.6 PROD

**Name:** PROD
**Address:** 0x4F3

Timer Register
Product Register Pair

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | PROD[15:8] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | PROD[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | x | x | x | x | x | x | x | x |

**Bits 15:0 – PROD[15:0]** PROD Most Significant

**Notes:** The individual bytes in this multibyte register can be accessed with the following register names:

- PRODH: Accesses the high byte PROD[15:8]
- PRODL: Accesses the low byte PROD[7:0]

### 7.7.7 STATUS

**Name:** STATUS
**Address:** 0x4D8

STATUS Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | $\overline{TO}$ | $\overline{PD}$ | N | OV | Z | DC | C |
| Access | | R | R | R/W | R/W | R/W | R/W | R/W |
| Reset | | 1 | 1 | 0 | 0 | 0 | 0 | 0 |

**Bit 6 – $\overline{TO}$** Time-Out

Reset States: POR/BOR = 1
All Other Resets = q

| Value | Description |
|---|---|
| 1 | Set at power-up or by execution of the CLRWDT or SLEEP instruction |
| 0 | A WDT time-out occurred |

**Bit 5 – $\overline{PD}$** Power-Down

Reset States: POR/BOR = 1
All Other Resets = q

| Value | Description |
|---|---|
| 1 | Set at power-up or by execution of the CLRWDT instruction |
| 0 | Cleared by execution of the SLEEP instruction |

**Bit 4 – N** Negative

Used for signed arithmetic (two's complement); indicates if the result is negative (ALU MSb = 1).

Reset States: POR/BOR = 0
All Other Resets = u

| Value | Description |
|---|---|
| 1 | The result is negative |
| 0 | The result is positive |

**Bit 3 – OV** Overflow

Used for signed arithmetic (two's complement); indicates an overflow of the 7-bit magnitude, which causes the sign bit (bit 7) to change state.

Reset States: POR/BOR = 0
All Other Resets = u

| Value | Description |
|---|---|
| 1 | Overflow occurred for current signed arithmetic operation |
| 0 | No overflow occurred |

**Bit 2 – Z** Zero

Reset States: POR/BOR = 0
All Other Resets = u

| Value | Description |
|---|---|
| 1 | The result of an arithmetic or logic operation is zero |
| 0 | The result of an arithmetic or logic operation is not zero |

**Bit 1 – DC** Digit Carry / $\overline{Borrow}$

ADDWF, ADDLW, SUBLW, SUBWF instructions(1)

Reset States: POR/BOR = 0
All Other Resets = u

| Value | Description |
|---|---|
| 1 | A carry-out from the 4th low-order bit of the result occurred |
| 0 | No carry-out from the 4th low-order bit of the result |

**Bit 0 – C** Carry / $\overline{\text{Borrow}}$

`ADDWF`, `ADDLW`, `SUBLW`, `SUBWF` instructions(1,2)

Reset States: POR/BOR = 0
All Other Resets = u

| Value | Description |
|---|---|
| 1 | A carry-out from the Most Significant bit of the result occurred |
| 0 | No carry-out from the Most Significant bit of the result occurred |

**Notes:**

1. For $\overline{\text{Borrow}}$, the polarity is reversed. A subtraction is executed by adding the two's complement of the second operand.
2. For Rotate (`RRCF`, `RLCF`) instructions, this bit is loaded with either the high or low-order bit of the Source register.

## 7.8 Register Summary - System Arbiter Control

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0xB4 | PRLOCK | 7:0 | | | | | | | | PRLOCKED |
| 0xB5 | SCANPR | 7:0 | | | | | | PR[2:0] | | |
| 0xB6 | DMA1PR | 7:0 | | | | | | PR[2:0] | | |
| 0xB7 | DMA2PR | 7:0 | | | | | | PR[2:0] | | |
| 0xB8 | DMA3PR | 7:0 | | | | | | PR[2:0] | | |
| 0xB9 | DMA4PR | 7:0 | | | | | | PR[2:0] | | |
| 0xBA | DMA5PR | 7:0 | | | | | | PR[2:0] | | |
| 0xBB | DMA6PR | 7:0 | | | | | | PR[2:0] | | |
| 0xBC ... 0xBD | Reserved | | | | | | | | | |
| 0xBE | MAINPR | 7:0 | | | | | | PR[2:0] | | |
| 0xBF | ISRPR | 7:0 | | | | | | PR[2:0] | | |
| 0xC0 ... 0x0372 | Reserved | | | | | | | | | |
| 0x0373 | STATUS_CSHAD | 7:0 | | $\overline{TO}$ | $\overline{PD}$ | N | OV | Z | DC | C |
| 0x0374 | WREG_CSHAD | 7:0 | WREG[7:0] | | | | | | | |
| 0x0375 | BSR_CSHAD | 7:0 | | | BSR[5:0] | | | | | |
| 0x0376 | Reserved | | | | | | | | | |
| 0x0377 | STATUS_SHAD | 7:0 | | $\overline{TO}$ | $\overline{PD}$ | N | OV | Z | DC | C |
| 0x0378 | WREG_SHAD | 7:0 | WREG[7:0] | | | | | | | |
| 0x0379 | BSR_SHAD | 7:0 | | | BSR[5:0] | | | | | |
| 0x037A | PCLAT_SHAD | 7:0 | PCLATH[7:0] | | | | | | | |
| | | 15:8 | | | | PCLATU[4:0] | | | | |
| 0x037C | FSR0_SHAD | 7:0 | FSRL[7:0] | | | | | | | |
| | | 15:8 | | | FSRH[5:0] | | | | | |
| 0x037E | FSR1_SHAD | 7:0 | FSRL[7:0] | | | | | | | |
| | | 15:8 | | | FSRH[5:0] | | | | | |
| 0x0380 | FSR2_SHAD | 7:0 | FSRL[7:0] | | | | | | | |
| | | 15:8 | | | FSRH[5:0] | | | | | |
| 0x0382 | PROD_SHAD | 7:0 | PROD[7:0] | | | | | | | |
| | | 15:8 | PROD[15:8] | | | | | | | |
| 0x0384 ... 0x04D7 | Reserved | | | | | | | | | |
| 0x04D8 | STATUS | 7:0 | | $\overline{TO}$ | $\overline{PD}$ | N | OV | Z | DC | C |
| 0x04D9 ... 0x04F2 | Reserved | | | | | | | | | |
| 0x04F3 | PROD | 7:0 | PROD[7:0] | | | | | | | |
| | | 15:8 | PROD[15:8] | | | | | | | |

# 8. Device Configuration

## 8.1 Configuration Settings

The Configuration settings allow the user to set up the device with several choices of oscillators, Resets and memory protection options. These are implemented at 0x300000 - 0x300009.

**Important:** The $\overline{\text{DEBUG}}$ Configuration bit is managed automatically by device development tools including debuggers and programmers. For normal device operation, this bit needs to be maintained as a '1'.

## 8.2 Code Protection

Code protection allows the device to be protected from unauthorized access. Internal access to the program memory is unaffected by any code protection setting. A single code-protect bit controls the access for both program memory and data EEPROM memory.

The entire program memory and Data EEPROM space is protected from external reads and writes by the $\overline{\text{CP}}$ bit. When $\overline{\text{CP}}$ = 0, external reads and writes are inhibited and a read will return all '0's. The CPU can continue to read the memory, regardless of the protection bit settings. Self-writing the program memory is dependent upon the write protection setting.

## 8.3 User ID

32 words in the memory space (0x200000 - 0x20003F) are designated as ID locations where the user can store checksum or other code identification numbers. These locations are readable and writable during normal execution. See the **"User ID, Device ID and Configuration Settings Access, DIA and DCI"** section in the **"NVM - Nonvolatile Memory Module"** chapter for more information on accessing these memory locations. For more information on checksum calculation, see the **"PIC18FXXQ43 Family Programming Specification"** (DS40002079D).

## 8.4 Device ID and Revision ID

The 16-bit device ID word is located at 0x3FFFFE and the 16-bit revision ID is located at 0x3FFFFC. These locations are read-only and cannot be erased or modified.

Development tools, such as device programmers and debuggers, may be used to read the Device ID, Revision ID and Configuration bits. Refer to the **"NVM - Nonvolatile Memory Module"** section for more information on accessing these locations.

## 8.5 Register Definitions: Configuration Settings

### 8.5.1 CONFIG1

**Name:** CONFIG1
**Address:** 0x300000

Configuration Byte 1

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | RSTOSC[2:0] | | | | FEXTOSC[2:0] | | |
| Access | | R/W | R/W | R/W | | R/W | R/W | R/W |
| Reset | | 1 | 1 | 1 | | 1 | 1 | 1 |

**Bits 6:4 – RSTOSC[2:0]** Power-Up Default Value for COSC

This value is the Reset default value for COSC and selects the oscillator first used by user software. Refer to COSC operation.

| Value | Description |
|---|---|
| 111 | EXTOSC operating per FEXTOSC bits |
| 110 | HFINTOSC with HFFRQ = 4 MHz and CDIV = 4:1. Resets COSC/NOSC to b'110'. |
| 101 | LFINTOSC |
| 100 | SOSC |
| 011 | Reserved |
| 010 | EXTOSC with 4x PLL, with EXTOSC operating per FEXTOSC bits |
| 001 | Reserved |
| 000 | HFINTOSC with HFFRQ = 64 MHz and CDIV = 1:1. Resets COSC/NOSC to b'110'. |

**Bits 2:0 – FEXTOSC[2:0]** External Oscillator Mode Selection

| Value | Description |
|---|---|
| 111 | ECH (external clock) above 8 MHz |
| 110 | ECM (external clock) for 500 kHz to 8 MHz |
| 101 | ECL (external clock) below 500 kHz |
| 100 | Oscillator not enabled |
| 011 | Reserved (do not use) |
| 010 | HS (crystal oscillator) above 4 MHz |
| 001 | XT (crystal oscillator) above 500 kHz, below 4 MHz |
| 000 | LP (crystal oscillator) optimized for 32.768 kHz |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

### 8.5.2 CONFIG2

**Name:** CONFIG2
**Address:** 0x300001

Configuration Byte 2

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | FCMEN | | CSWEN | | PR1WAY | $\overline{\text{CLKOUTEN}}$ |
| Access | | | R/W | | R/W | | R/W | R/W |
| Reset | | | 1 | | 1 | | 1 | 1 |

**Bit 5 – FCMEN** Fail-Safe Clock Monitor Enable

| Value | Description |
|---|---|
| 1 | Fail-Safe Clock Monitor enabled |
| 0 | Fail-Safe Clock Monitor disabled |

**Bit 3 – CSWEN** Clock Switch Enable

| Value | Description |
|---|---|
| 1 | Writing to NOSC and NDIV is allowed |
| 0 | The NOSC and NDIV bits cannot be changed by user software |

**Bit 1 – PR1WAY** PRLOCKED One-Way Set Enable

| Value | Description |
|---|---|
| 1 | The PRLOCKED bit can be cleared and set only once; Priority registers remain locked after one clear/set cycle |
| 0 | The PRLOCKED bit can be set and cleared repeatedly (subject to the unlock sequence) |

**Bit 0 – $\overline{\text{CLKOUTEN}}$** Clock Out Enable

If FEXTOSC = HS, XT, LP, then this bit is ignored.
Otherwise:

| Value | Description |
|---|---|
| 1 | CLKOUT function is disabled; I/O function on OSC2 |
| 0 | CLKOUT function is enabled; $F_{OSC}/4$ clock appears at OSC2 |

### 8.5.3 CONFIG3

**Name:** CONFIG3
**Address:** 0x300002

Configuration Byte 3

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | BOREN[1:0] | | $\overline{\text{LPBOREN}}$ | IVT1WAY | MVECEN | PWRTS[1:0] | | MCLRE |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Bits 7:6 – BOREN[1:0]** Brown-out Reset Enable
When enabled, Brown-out Reset Voltage ($V_{BOR}$) is set by the BORV bit.

| Value | Description |
|---|---|
| 11 | Brown-out Reset enabled, the SBOREN bit is ignored |
| 10 | Brown-out Reset enabled while running, disabled in Sleep; SBOREN is ignored |
| 01 | Brown-out Reset enabled according to SBOREN |
| 00 | Brown-out Reset disabled |

**Bit 5 – $\overline{\text{LPBOREN}}$** Low-Power BOR Enable

| Value | Description |
|---|---|
| 1 | Low-Power Brown-out Reset is disabled |
| 0 | Low-Power Brown-out Reset is enabled |

**Bit 4 – IVT1WAY** IVTLOCK One-Way Set Enable

| Value | Description |
|---|---|
| 1 | The IVTLOCK bit can be cleared and set only once; IVT registers remain locked after one clear/set cycle |
| 0 | The IVTLOCK bit can be set and cleared repeatedly (subject to the unlock sequence) |

**Bit 3 – MVECEN** Multivector Enable

| Value | Description |
|---|---|
| 1 | Multivector is enabled; vector table used for interrupts |
| 0 | Legacy interrupt behavior |

**Bits 2:1 – PWRTS[1:0]** Power-up Timer Selection

| Value | Description |
|---|---|
| 11 | PWRT is disabled |
| 10 | PWRT is set at 64 ms |
| 01 | PWRT is set at 16 ms |
| 00 | PWRT is set at 1 ms |

**Bit 0 – MCLRE** Master Clear ($\overline{\text{MCLR}}$) Enable

| Value | Condition | Description |
|---|---|---|
| x | If LVP = 1 | RE3 pin function is $\overline{\text{MCLR}}$ |
| 1 | If LVP = 0 | $\overline{\text{MCLR}}$ pin is $\overline{\text{MCLR}}$ |
| 0 | If LVP = 0 | $\overline{\text{MCLR}}$ pin function is a port-defined function |

### 8.5.4 CONFIG4

**Name:** CONFIG4
**Address:** 0x300003

Configuration Byte 4

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | $\overline{XINST}$ | | LVP | STVREN | PPS1WAY | $\overline{ZCD}$ | BORV[1:0] | |
| Access | R/W | | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 1 | | 1 | 1 | 1 | 1 | 1 | 1 |

**Bit 7 – $\overline{XINST}$** Extended Instruction Set Enable

| Value | Description |
|---|---|
| 1 | Extended Instruction Set and Indexed Addressing mode disabled (Legacy mode) |
| 0 | Extended Instruction Set and Indexed Addressing mode enabled |

**Bit 5 – LVP** Low-Voltage Programming Enable

The LVP bit cannot be written (to zero) while operating from the LVP programming interface. The purpose of this rule is to prevent the user from dropping out of LVP mode while programming from LVP mode, or accidentally eliminating LVP mode from the Configuration state.

| Value | Description |
|---|---|
| 1 | Low-Voltage Programming enabled. $\overline{MCLR}/V_{PP}$ pin function is $\overline{MCLR}$. The MCLRE Configuration bit is ignored. |
| 0 | HV on $\overline{MCLR}/V_{PP}$ must be used for programming |

**Bit 4 – STVREN** Stack Overflow/Underflow Reset Enable

| Value | Description |
|---|---|
| 1 | Stack Overflow or Underflow will cause a Reset |
| 0 | Stack Overflow or Underflow will not cause a Reset |

**Bit 3 – PPS1WAY** PPSLOCKED One-Way Set Enable

| Value | Description |
|---|---|
| 1 | The PPSLOCKED bit can only be set once after an unlocking sequence is executed; once PPSLOCK is set, all future changes to PPS registers are prevented |
| 0 | The PPSLOCKED bit can be set and cleared as needed (unlocking sequence is required) |

**Bit 2 – $\overline{ZCD}$** ZCD Disable

| Value | Description |
|---|---|
| 1 | ZCD disabled, ZCD can be enabled by setting the ZCDSEN bit of ZCDCON |
| 0 | ZCD always enabled, PMDx[ZCDMD] bit is ignored |

**Bits 1:0 – BORV[1:0]** Brown-out Reset Voltage Selection(1)

| Value | Description |
|---|---|
| 11 | Brown-out Reset Voltage ($V_{BOR}$) set to 1.90V |
| 10 | Brown-out Reset Voltage ($V_{BOR}$) set to 2.45V |
| 01 | Brown-out Reset Voltage ($V_{BOR}$) set to 2.7V |
| 00 | Brown-out Reset Voltage ($V_{BOR}$) set to 2.85V |

**Note:**

1. The higher voltage setting is recommended for an operation at or above 16 MHz.

### 8.5.5 CONFIG5

**Name:** CONFIG5
**Address:** 0x300004

Configuration Byte 5

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | WDTE[1:0] | | WDTCPS[4:0] | | | | |
| Access | | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Bits 6:5 – WDTE[1:0]** WDT Operating Mode

| Value | Description |
|---|---|
| 11 | WDT enabled regardless of Sleep; the SEN bit in WDTCON0 is ignored |
| 10 | WDT enabled while Sleep = 0, suspended when Sleep = 1; the SEN bit in WDTCON0 is ignored |
| 01 | WDT enabled/disabled by the SEN bit in WDTCON0 |
| 00 | WDT disabled, the SEN bit in WDTCON0 is ignored |

**Bits 4:0 – WDTCPS[4:0]** WDT Period Select

| WDTCPS | WDTCON0[WDTPS] at POR | | | | Software Control of WDTPS? |
|---|---|---|---|---|---|
| | Value | Divider Ratio | | Typical Time-Out ($F_{IN}$ = 31 kHz) | |
| 11111 | 01011 | 1:65536 | $2^{16}$ | 2s | Yes |
| 11110 to 10011 | 11110 to 10011 | 1:32 | $2^{5}$ | 1 ms | No |
| 10010 | 10010 | 1:8388608 | $2^{23}$ | 256s | No |
| 10001 | 10001 | 1:4194304 | $2^{22}$ | 128s | No |
| 10000 | 10000 | 1:2097152 | $2^{21}$ | 64s | No |
| 01111 | 01111 | 1:1048576 | $2^{20}$ | 32s | No |
| 01110 | 01110 | 1:524288 | $2^{19}$ | 16s | No |
| 01101 | 01101 | 1:262144 | $2^{18}$ | 8s | No |
| 01100 | 01100 | 1:131072 | $2^{17}$ | 4s | No |
| 01011 | 01011 | 1:65536 | $2^{16}$ | 2s | No |
| 01010 | 01010 | 1:32768 | $2^{15}$ | 1s | No |
| 01001 | 01001 | 1:16384 | $2^{14}$ | 512 ms | No |
| 01000 | 01000 | 1:8192 | $2^{13}$ | 256 ms | No |
| 00111 | 00111 | 1:4096 | $2^{12}$ | 128 ms | No |
| 00110 | 00110 | 1:2048 | $2^{11}$ | 64 ms | No |
| 00101 | 00101 | 1:1024 | $2^{10}$ | 32 ms | No |
| 00100 | 00100 | 1:512 | $2^{9}$ | 16 ms | No |
| 00011 | 00011 | 1:256 | $2^{8}$ | 8 ms | No |
| 00010 | 00010 | 1:128 | $2^{7}$ | 4 ms | No |
| 00001 | 00001 | 1:64 | $2^{6}$ | 2 ms | No |
| 00000 | 00000 | 1:32 | $2^{5}$ | 1 ms | No |

### 8.5.6 CONFIG6

**Name:** CONFIG6
**Address:** 0x300005

Configuration Byte 6

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | WDTCCS[2:0] | | | WDTCWS[2:0] | | |
| Access | | | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | | | 1 | 1 | 1 | 1 | 1 | 1 |

**Bits 5:3 – WDTCCS[2:0]** WDT Input Clock Selector

| Value | Condition | Description |
|---|---|---|
| x | WDTE = 00 | These bits have no effect |
| 111 | WDTE ≠ 00 | Software control |
| 110 to 011 | WDTE ≠ 00 | Reserved |
| 010 | WDTE ≠ 00 | WDT reference clock is the SOSC |
| 001 | WDTE ≠ 00 | WDT reference clock is the 31.25 kHz MFINTOSC |
| 000 | WDTE ≠ 00 | WDT reference clock is the 31.0 kHz LFINTOSC |

**Bits 2:0 – WDTCWS[2:0]** WDT Window Select

| WDTCWS | WDTCON1[WINDOW] at POR | | | Software Control of WINDOW | Keyed Access Required? |
|---|---|---|---|---|---|
| | Value | Window Delay Percent of Time | Window Opening Percent of Time | | |
| 111 | 111 | n/a | 100 | Yes | No |
| 110 | 110 | n/a | 100 | No | Yes |
| 101 | 101 | 25 | 75 | | |
| 100 | 100 | 37.5 | 62.5 | | |
| 011 | 011 | 50 | 50 | | |
| 010 | 010 | 62.5 | 37.5 | | |
| 001 | 001 | 75 | 25 | | |
| 000 | 000 | 87.5 | 12.5 | | |

### 8.5.7 CONFIG7

**Name:** CONFIG7
**Address:** 0x300006

Configuration Byte 7

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | $\overline{\text{DEBUG}}$ | $\overline{\text{SAFEN}}$ | $\overline{\text{BBEN}}$ | BBSIZE[2:0] | | |
| Access | | | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | | | 1 | 1 | 1 | 1 | 1 | 1 |

**Bit 5 – $\overline{\text{DEBUG}}$** Debugger Enable

| Value | Description |
|---|---|
| 1 | Background debugger disabled |
| 0 | Background debugger enabled |

**Bit 4 – $\overline{\text{SAFEN}}$** Storage Area Flash (SAF) Enable(1)

| Value | Description |
|---|---|
| 1 | SAF is disabled |
| 0 | SAF is enabled |

**Bit 3 – $\overline{\text{BBEN}}$** Boot Block Enable(1)

| Value | Description |
|---|---|
| 1 | Boot Block is disabled |
| 0 | Boot Block is enabled |

**Bits 2:0 – BBSIZE[2:0]** Boot Block Size Selection(2)

**Table 8-1. Boot Block Size**

| $\overline{\text{BBEN}}$ | BBSIZE | End Address of Boot Block | Boot Block Size (words) | | |
|---|---|---|---|---|---|
| | | | PIC18Fx5Q43 | PIC18Fx6Q43 | PIC18Fx7Q43 |
| 1 | xxx | – | – | | |
| 0 | 111 | 00 03FFh | 512 | | |
| 0 | 110 | 00 07FFh | 1024 | | |
| 0 | 101 | 00 0FFFh | 2048 | | |
| 0 | 100 | 00 1FFFh | 4096 | | |
| 0 | 011 | 00 3FFFh | 8192 | | |
| 0 | 010 | 00 7FFFh | – | 16384 | |
| 0 | 001 | 00 FFFFh | – | | 32768 |
| 0 | 000 | 01 FFFFh | – | | |

**Notes:**

1. Once protection is enabled through ICSP™ or a self-write, it can only be reset through a Bulk Erase.
2. BBSIZE[2:0] bits can only be changed when BBEN = 1. Once BBEN = 0, BBSIZE[2:0] can only be changed through a Bulk Erase.

### 8.5.8 CONFIG8

**Name:** CONFIG8
**Address:** 0x300007

Configuration Byte 8

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | $\overline{\text{WRTAPP}}$ | | | | $\overline{\text{WRTSAF}}$ | $\overline{\text{WRTD}}$ | $\overline{\text{WRTC}}$ | $\overline{\text{WRTB}}$ |
| Access | R/W | | | | R/W | R/W | R/W | R/W |
| Reset | 1 | | | | 1 | 1 | 1 | 1 |

**Bit 7 – $\overline{\text{WRTAPP}}$** Application Block Write Protection(1)

| Value | Description |
|---|---|
| 1 | Application Block is not write-protected |
| 0 | Application Block is write-protected |

**Bit 3 – $\overline{\text{WRTSAF}}$** Storage Area Flash (SAF) Write Protection(1,2)

| Value | Description |
|---|---|
| 1 | SAF is not write-protected |
| 0 | SAF is write-protected |

**Bit 2 – $\overline{\text{WRTD}}$** Data EEPROM Write Protection(1)

| Value | Description |
|---|---|
| 1 | Data EEPROM is not write-protected |
| 0 | Data EEPROM is write-protected |

**Bit 1 – $\overline{\text{WRTC}}$** Configuration Register Write Protection(1)

| Value | Description |
|---|---|
| 1 | Configuration registers are not write-protected |
| 0 | Configuration registers are write-protected |

**Bit 0 – $\overline{\text{WRTB}}$** Boot Block Write Protection(1,3)

| Value | Description |
|---|---|
| 1 | Boot Block is not write-protected |
| 0 | Boot Block is write-protected |

**Notes:**

1. Once protection is enabled through ICSP™ or a self-write, it can only be reset through a Bulk Erase.
2. Applicable only if $\overline{\text{SAFEN}}$ = 0.
3. Applicable only if $\overline{\text{BBEN}}$ = 0.

### 8.5.9 CONFIG9

**Name:** CONFIG9
**Address:** 0x300008

Configuration Byte 9

This register is reserved

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | |
| Access | | | | | | | | |
| Reset | | | | | | | | |

### 8.5.10 CONFIG10

**Name:** CONFIG10
**Address:** 0x300009

Configuration Byte 10

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | $\overline{CP}$ |
| Access | | | | | | | | R/W |
| Reset | | | | | | | | 1 |

**Bit 0 – $\overline{CP}$** User Program Flash Memory and Data EEPROM Code Protection(1)

| Value | Description |
|---|---|
| 1 | User Program Flash Memory and Data EEPROM code protection are disabled |
| 0 | User Program Flash Memory and Data EEPROM code protection are enabled |

**Note:**

1. Once this bit is enabled, it can only be reset through a Bulk Erase.

## 8.6 Register Summary - Configuration Settings

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x300000 | CONFIG1 | 7:0 | | RSTOSC[2:0] | | | | FEXTOSC[2:0] | | |
| 0x300001 | CONFIG2 | 7:0 | | | FCMEN | | CSWEN | | PR1WAY | CLKOUTEN |
| 0x300002 | CONFIG3 | 7:0 | BOREN[1:0] | | LPBOREN | IVT1WAY | MVECEN | PWRTS[1:0] | | MCLRE |
| 0x300003 | CONFIG4 | 7:0 | XINST | | LVP | STVREN | PPS1WAY | ZCD | BORV[1:0] | |
| 0x300004 | CONFIG5 | 7:0 | | WDTE[1:0] | | WDTCPS[4:0] | | | | |
| 0x300005 | CONFIG6 | 7:0 | | | WDTCCS[2:0] | | | WDTCWS[2:0] | | |
| 0x300006 | CONFIG7 | 7:0 | | | DEBUG | SAFEN | BBEN | BBSIZE[2:0] | | |
| 0x300007 | CONFIG8 | 7:0 | WRTAPP | | | | WRTSAF | WRTD | WRTC | WRTB |
| 0x300008 | CONFIG9 | 7:0 | | | | | | | | |
| 0x300009 | CONFIG10 | 7:0 | | | | | | | | CP |

## 8.7 Register Definitions: Device ID and Revision ID

### 8.7.1 Device ID

**Name:** DEVICEID
**Address:** 0x3FFFFE

Device ID Register

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | DEV[15:8] | | | | | | | |
| Access | R | R | R | R | R | R | R | R |
| Reset | q | q | q | q | q | q | q | q |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | DEV[7:0] | | | | | | | |
| Access | R | R | R | R | R | R | R | R |
| Reset | q | q | q | q | q | q | q | q |

**Bits 15:0 – DEV[15:0]** Device ID

| Device | Device ID |
|---|---|
| PIC18F27Q43 | 7480h |
| PIC18F47Q43 | 74A0h |
| PIC18F57Q43 | 74C0h |

### 8.7.2 Revision ID

**Name:** REVISIONID
**Address:** 0x3FFFFC

Revision ID Register

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | 1010[3:0] | | | | MJRREV[5:2] | | | |
| Access | R | R | R | R | R | R | R | R |
| Reset | 1 | 0 | 1 | 0 | q | q | q | q |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | MJRREV[1:0] | | MNRREV[5:0] | | | | | |
| Access | R | R | R | R | R | R | R | R |
| Reset | q | q | q | q | q | q | q | q |

**Bits 15:12 – 1010[3:0]** Read as `'b1010`
These bits are fixed with value `'b1010` for all devices in this family.

**Bits 11:6 – MJRREV[5:0]** Major Revision ID
These bits are used to identify a major revision (A0, B0, C0, etc.).
Revision A = `'b00 0000`
Revision B = `'b00 0001`

**Bits 5:0 – MNRREV[5:0]** Minor Revision ID
These bits are used to identify a minor revision.
Revision A0 = `'b00 0000`
Revision B0 = `'b00 0000`
Revision B1 = `'b00 0001`

**Tip:** For example, the REVISIONID register value for revision B1 will be `0xA041`.

## 8.8 Register Summary - DEVID/REVID

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x3FFFFC | REVISIONID | 7:0 | MJRREV[1:0] | | MNRREV[5:0] | | | | | |
| | | 15:8 | 1010[3:0] | | | | MJRREV[5:2] | | | |
| 0x3FFFFE | DEVICEID | 7:0 | DEV[7:0] | | | | | | | |
| | | 15:8 | DEV[15:8] | | | | | | | |

# 9. Memory Organization

There are three types of memory in PIC18 microcontroller devices:

- Program Memory
- Data RAM
- Data EEPROM

In Harvard architecture devices, the data and program memories use separate buses that allow for concurrent access of the two memory spaces. The data EEPROM, for practical purposes, can be regarded as a peripheral device, since it is addressed and accessed through a set of control registers.

Additional detailed information on the operation of the Program Flash Memory and data EEPROM memory is provided in the **"NVM - Nonvolatile Memory Module"** section.

## 9.1 Program Memory Organization

PIC18 microcontrollers implement a 21-bit Program Counter, which is capable of addressing a 2 Mbyte program memory space. Accessing a location between the upper boundary of the physically implemented memory and the 2 Mbyte address will return all '0's (a NOP instruction).

Refer to the following tables for device memory maps and code protection Configuration bits associated with the various sections of PFM.

The Reset vector address is at 000000h. The PIC18-Q43 devices feature a vectored interrupt controller with a dedicated interrupt vector table stored in the program memory. Refer to the **"VIC - Vectored Interrupt Controller Module"** chapter for more details.

**Figure 9-1. Program and Data Memory Map**

Rev. 40-000101E
4/20/2017

| Address | Device | | |
|---|---|---|---|
| | PIC18Fx5Q43 | PIC18Fx6Q43 | PIC18Fx7Q43 |
| 00 0000h to 00 3FFFh | Program Flash Memory (16 KW)(1) | Program Flash Memory (32 KW)(1) | Program Flash Memory (64 KW)(1) |
| 00 4000h to 00 7FFFh | | | |
| 00 8000h to 00 FFFFh | Not Present(2) | | |
| 01 0000h to 01 FFFFh | | Not Present(2) | |
| 02 0000h to 1F FFFFh | | | Not Present(2) |
| 20 0000h to 20 003Fh | User IDs (32 Words)(3) | | |
| 20 0040h to 2B FFFFh | Reserved | | |
| 2C 0000h to 2C 00FFh | Device Information Area (DIA)(3,5) | | |
| 2C 0100h to 2F FFFFh | Reserved | | |
| 30 0000h to 30 0009h | Configuration Bytes(3) | | |
| 30 000Ah to 37 FFFFh | Reserved | | |
| 38 0000h to 38 03FFh | Data EEPROM (1024 Bytes) | | |
| 38 0400h to 3B FFFFh | Reserved | | |
| 3C 0000h to 3C 0009h | Device Configuration Information(3,4,5) | | |
| 3C 000Ah to 3F FFFBh | Reserved | | |
| 3F FFFCh to 3F FFFDh | Revision ID (1 Word)(3,4,5) | | |
| 3F FFFEh to 3F FFFFh | Device ID (1 Word)(3,4,5) | | |

**Notes: 1.** Storage Area Flash is implemented as the last 128 Words of User Flash, if enabled.

**2.** The addresses do not roll over. The region is read as '0'.

**3.** Not code-protected.

**4.** Hard-coded in silicon.

**5.** This region cannot be written by the user and it is not affected by a Bulk Erase.

### 9.1.1 Memory Access Partition

In the PIC18-Q43 devices, the program memory can be further partitioned into the following sub-blocks:

- Application block
- Boot block
- Storage Area Flash (SAF) block

Refer to the **"Program Flash Memory Partition"** table for more details.

#### 9.1.1.1 Application Block

Application block is where the user's firmware resides by default. Default settings of the Configuration bits ($\overline{\text{BBEN}}$ = 1 and $\overline{\text{SAFEN}}$ = 1) assign all memory in the program Flash memory area to the application block. The $\overline{\text{WRTAPP}}$ Configuration bit is used to write-protect the application block.

#### 9.1.1.2 Boot Block

Boot block is an area in program memory that is ideal for storing bootloader code. Code placed in this area can be executed by the CPU. The boot block can be write-protected, independent of the main application block. The Boot Block is enabled by the $\overline{\text{BBEN}}$ Configuration bit and size is based on the value of the BBSIZE Configuration bits. The $\overline{\text{WRTB}}$ Configuration bit is used to write-protect the Boot Block.

#### 9.1.1.3 Storage Area Flash

Storage Area Flash (SAF) is the area in program memory that can be used as data storage. SAF is enabled by the $\overline{\text{SAFEN}}$ Configuration bit. If enabled, the code placed in this area cannot be executed by the CPU. The SAF block is placed at the end of memory and spans 128 Words. The $\overline{\text{WRTSAF}}$ Configuration bit is used to write-protect the Storage Area Flash.

**Important:** If write-protected locations are written to, memory is not changed and the WRERR bit is set.

**Table 9-1. Program Flash Memory Partition**

| Region | Address | Partition[3] | | | |
|---|---|---|---|---|---|
| | | $\overline{BBEN}$ = 1<br>$\overline{SAFEN}$ = 1 | $\overline{BBEN}$ = 1<br>$\overline{SAFEN}$ = 0 | $\overline{BBEN}$ = 0<br>$\overline{SAFEN}$ = 1 | $\overline{BBEN}$ = 0<br>$\overline{SAFEN}$ = 0 |
| Program Flash Memory | 00 0000h<br>....<br>Last Boot Block Memory Address | Application Block | Application Block | Boot Block | Boot Block |
| | Last Boot Block Memory Address[1] + 1<br>....<br>Last Program Memory Address[2] - 100h | | | Application Block | Application Block |
| | Last Program Memory Address[2] - FEh[4]<br>....<br>Last Program Memory Address[2] | | Storage Area Flash Block | | Storage Area Flash Block |

**Notes:**

1. Last Boot Block address is based on BBSIZE bits. Refer to the **"Device Configuration"** chapter for more details.
2. For Last Program Memory address refer the table above.
3. Refer to the **"Device Configuration"** chapter for $\overline{BBEN}$ and $\overline{SAFEN}$ bit definitions.
4. Storage Area Flash is implemented as the last 128 Words of user Flash memory.

### 9.1.2 Program Counter

The Program Counter (PC) specifies the address of the instruction to fetch for execution. The PC is 21 bits wide and is contained in three separate 8-bit registers. The low byte, known as the PCL register, is both readable and writable. The high byte, or PCH register, contains the PC[15:8] bits; it is not directly readable or writable. Updates to the PCH register are performed through the PCLATH register. The upper byte is called PCU. This register contains the PC[20:16] bits; it is also not directly readable or writable. Updates to the PCU register are performed through the PCLATU register.

The contents of PCLATH and PCLATU are transferred to the Program Counter by any operation that writes PCL. Similarly, the upper two bytes of the Program Counter are transferred to PCLATH and PCLATU by an operation that reads PCL. This is useful for computed offsets to the PC (see the Computed GOTO section).

The PC addresses bytes in the program memory. To prevent the PC from becoming misaligned with word instructions, the Least Significant bit of PCL is fixed to a value of '0'. The PC increments by two to address sequential instructions in the program memory.

The CALL, RCALL, GOTO and program branch instructions write to the Program Counter directly. For these instructions, the contents of PCLATH and PCLATU are not transferred to the Program Counter.

© 2019-2023 Microchip Technology Inc. and its subsidiaries

### 9.1.3 Return Address Stack

The return address stack allows any combination of up to 127 program calls and interrupts to occur. The PC is pushed onto the stack when a CALL or RCALL instruction is executed or an interrupt is Acknowledged. The PC value is pulled off the stack on a RETURN, RETLW or a RETFIE instruction. PCLATU and PCLATH are not affected by any of the RETURN or CALL instructions.

The Stack Pointer is readable and writable and the address on the top of the stack is readable and writable through the Top-of-Stack (TOS) Special File registers. Data can also be pushed to, or popped from the stack, using these registers.

A CALL type instruction causes a push onto the stack; the Stack Pointer is first incremented and the location pointed to by the Stack Pointer is written with the contents of the PC (already pointing to the instruction following the CALL). A RETURN type instruction causes a pop from the stack; the contents of the location pointed to by the STKPTR are transferred to the PC and then the Stack Pointer is decremented.

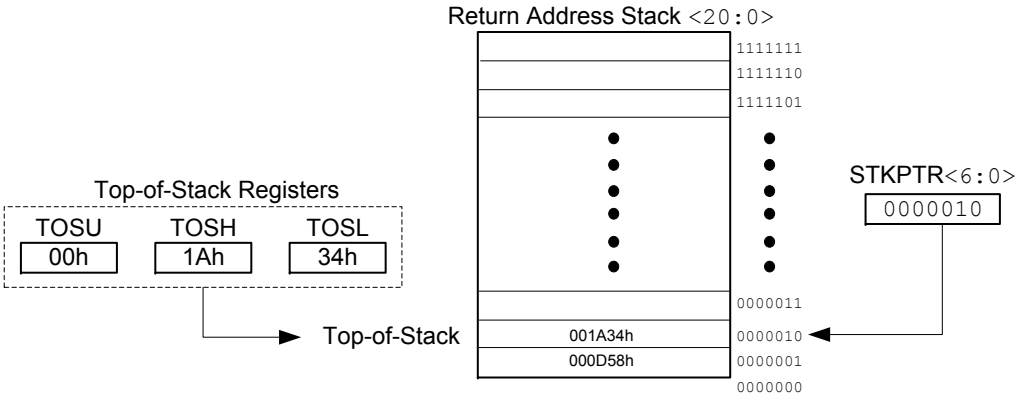
The Stack Pointer is initialized to 0x00 after all Resets.

#### 9.1.3.1 Top-of-Stack Access

Only the top of the return address stack (TOS) is readable and writable. A set of three registers, TOSU:TOSH:TOSL, hold the contents of the stack location pointed to by the STKPTR register (see Figure 9-2). This allows users to implement a software stack if necessary. After a CALL, RCALL or interrupt, the software can read the pushed value by reading the TOSU:TOSH:TOSL registers. These values can be placed on a user defined software stack. At return time, the software can return these values to TOSU:TOSH:TOSL and do a return.

The user must disable the Global Interrupt Enable (GIE) bits while accessing the stack to prevent inadvertent stack corruption.

**Figure 9-2. Return Address Stack and Associated Registers**




#### 9.1.3.2 Return Stack Pointer

The STKPTR register contains the Stack Pointer value. The Stack Overflow (STKOVF) Status bit and the Stack Underflow (STKUNF) Status bit can be accessed using the PCON0 register. The value of the Stack Pointer can be zero through 127. On Reset, the Stack Pointer value will be zero. The user may read and write the Stack Pointer value. After the PC is pushed onto the stack 128 times (without popping any values off the stack), the STKOVF bit is set. The STKOVF bit is cleared by software or by a POR. The action that takes place when the stack becomes full depends on the state of the Stack Overflow Reset Enable (STVREN) Configuration bit.

If STVREN is set (default), a Reset will be generated and a Stack Overflow will be indicated by the STKOVF bit. This includes CALL and CALLW instructions, as well as stacking the return address during an interrupt response. The STKOVF bit will remain set and the Stack Pointer will be set to zero.

If STVREN is cleared, the STKOVF bit will be set on the 128$^{th}$ push and the Stack Pointer will remain at 127 but no Reset will occur. Any additional pushes will overwrite the 127$^{st}$ push but the STKPTR will remain unchanged.

Setting STKOVF = 1 in software will change the bit, but will not generate a Reset.

© 2019-2023 Microchip Technology Inc. and its subsidiaries

The STKUNF bit is set when a stack pop returns a value of '0'. The STKUNF bit is cleared by software or by POR. The action that takes place when the stack becomes full depends on the state of the Stack Overflow Reset Enable (STVREN) Configuration bit.

If STVREN is set (default) and the stack has been popped enough times to unload the stack, the next pop will return a value of '0' to the PC, it will set the STKUNF bit and a Reset will be generated. This condition can be generated by the RETURN, RETLW and RETFIE instructions.

If STVREN is cleared, the STKUNF bit will be set, but no Reset will occur.

**Important:** Returning a value of '0' to the PC on an underflow has the effect of vectoring the program to the Reset vector, where the stack conditions can be verified and appropriate actions can be taken. This is not the same as a Reset, as the contents of the SFRs are not affected.

#### 9.1.3.3 PUSH and POP Instructions

Since the Top-of-Stack is readable and writable, the ability to push values onto the stack and pull values off the stack without disturbing normal program execution is a desirable feature. The PIC18 instruction set includes two instructions, PUSH and POP, that permit the TOS to be manipulated under software control. TOSU, TOSH and TOSL can be modified to place data or a return address on the stack.

The PUSH instruction places the current PC value onto the stack. This increments the Stack Pointer and loads the current PC value onto the stack.

The POP instruction discards the current TOS by decrementing the Stack Pointer. The previous value pushed onto the stack then becomes the TOS value.

#### 9.1.3.4 Fast Register Stack

There are three levels of fast stack registers available - one for CALL type instructions and two for interrupts. A fast register stack is provided for the STATUS, WREG and BSR registers, to provide a "fast return" option for interrupts. It is loaded with the current value of the corresponding register when the processor vectors for an interrupt. All interrupt sources will push values into the stack registers. The values in the registers are then loaded back into their associated registers if the RETFIE, FAST instruction is used to return from the interrupt. Refer to the **"Call Shadow Register"** section for interrupt call shadow registers.

The following example shows a source code example that uses the Fast Register Stack during a subroutine call and return.

**Example 9-1. Fast Register Stack Code Example**

```
CALL SUB1, FAST  ;STATUS, WREG, BSR SAVED IN FAST REGISTER STACK
          •
          •
SUB1:
          •
          •
     RETURN, FAST     ;RESTORE VALUES SAVED IN FAST REGISTER STACK
```

### 9.1.4 Look-up Tables in Program Memory

There may be programming situations that require the creation of data structures, or Look-up Tables, in program memory. For PIC18 devices, Look-up Tables can be implemented in two ways:

- Computed GOTO
- Table reads

#### 9.1.4.1 Computed GOTO

A computed GOTO is accomplished by adding an offset to the Program Counter. An example is shown in the following code example.

A Look-up Table can be formed with an ADDWF PCL instruction and a group of RETLW *nn* instructions. The W register is loaded with an offset into the table before executing a call to that table. The first instruction of the called

routine is the ADDWF PCL instruction. The next instruction executed will be one of the RETLW *nn* instructions that returns the value 'nn' to the calling function.

The offset value (in WREG) specifies the number of bytes that the Program Counter will advance and must be multiples of two (LSb = 0).

In this method, only one data byte may be stored in each instruction location and room on the return address stack is required.

**Example 9-2. Computed GOTO Using an Offset Value**

```
     RLNCF   OFFSET, W   ; W must be an even number, Max OFFSET = 127
     CALL    TABLE

     ORG     nn00h    ; 00 in LSByte ensures no addition overflow
TABLE:
     ADDWF   PCL      ; Add OFFSET to program counter
     RETLW   A        ; Value @ OFFSET=0
     RETLW   B        ; Value @ OFFSET=1
     RETLW   C        ; Value @ OFFSET=2
       .
       .
       .
```

#### 9.1.4.2 Program Flash Memory Access

A more compact method of storing data in program memory allows two bytes of data to be stored in each instruction location.

Look-up Table data may be stored two bytes per program word by using table reads and writes. The Table Pointer (TBLPTR) register specifies the byte address and the Table Latch (TABLAT) register contains the data that is read from or written to program memory. Data is transferred to or from program memory one byte at a time.

Table read and table write operations are discussed further in the **"Table Read Operations"** and **"Table Write Operations"** sections in the **"NVM - Nonvolatile Memory Module"** chapter.

## 9.2 Device Information Area

The Device Information Area (DIA) is a dedicated region in the program memory space. The DIA contains the calibration data for the internal temperature indicator module, the Microchip Unique Identifier words, and the Fixed Voltage Reference voltage readings measured in mV.

The complete DIA table is shown below, followed by a description of each region and its functionality. The data is mapped from 2C0000h to 2C003Fh. These locations are read-only and cannot be erased or modified. The data is programmed into the device during manufacturing.

**Table 9-2. Device Information Area**

| Address Range | Name of Region | Standard Device Information |
|---|---|---|
| 2C0000h-2C0011h | MUI0 | Microchip Unique Identifier (9 Words) |
| | MUI1 | |
| | MUI2 | |
| | MUI3 | |
| | MUI4 | |
| | MUI5 | |
| | MUI6 | |
| | MUI7 | |
| | MUI8 | |
| 2C0012h-2C0013h | MUI9 | Reserved (1 Word) |
| 2C0014h-2C0023h | EUI0 | Optional External Unique Identifier (8 Words) |
| | EUI1 | |
| | EUI2 | |
| | EUI3 | |
| | EUI4 | |
| | EUI5 | |
| | EUI6 | |
| | EUI7 | |
| 2C0024h-2C0025h | TSLR1(1) | Gain = $\frac{0.1C \times 256}{count}$ (low range setting) |
| 2C0026h-2C0027h | TSLR2(1) | Temperature indicator ADC reading at 90°C (low range setting) |
| 2C0028h-2C0029h | TSLR3(1) | Offset (low range setting) |
| 2C002Ah-2C002Bh | TSHR1(2) | Gain = $\frac{0.1C \times 256}{count}$ (high range setting) |
| 2C002Ch-2C002Dh | TSHR2(2) | Temperature indicator ADC reading at 90°C (high range setting) |
| 2C002Eh-2C002Fh | TSHR3(2) | Offset (high range setting) |
| 2C0030h-2C0031h | FVRA1X | ADC FVR1 Output voltage for 1x setting (in mV) |
| 2C0032h-2C0033h | FVRA2X | ADC FVR1 Output Voltage for 2x setting (in mV) |
| 2C0034h-2C0035h | FVRA4X | ADC FVR1 Output Voltage for 4x setting (in mV) |
| 2C0036h-2C0037h | FVRC1X | Comparator FVR2 output voltage for 1x setting (in mV) |
| 2C0038h-2C0039h | FVRC2X | Comparator FVR2 output voltage for 2x setting (in mV) |
| 2C003Ah-2C003Bh | FVRC4X | Comparator FVR2 output voltage for 4x setting (in mV) |

© 2019-2023 Microchip Technology Inc. and its subsidiaries

.....continued

| Address Range | Name of Region | Standard Device Information |
|---|---|---|
| 2C003Ch-2C003Fh | | Unassigned (2 Words) |

**Notes:**

1. TSLR: Address 2C0024h-2C0029h store the measurements for the low range setting of the temperature sensor at $V_{DD}$ = 3V, $V_{REF}$+ = 2.048V from FVR1.
2. TSHR: Address 2C002Ah-2C002Fh store the measurements for the high range setting of the temperature sensor at $V_{DD}$ = 3V, $V_{REF}$+ = 2.048V from FVR1.

### 9.2.1 Microchip Unique Identifier (MUI)

This family of devices is individually encoded during final manufacturing with a Microchip Unique Identifier (MUI). The MUI cannot be user-erased. This feature allows for manufacturing traceability of Microchip Technology devices in applications where this is required. It may also be used by the application manufacturer for a number of functions that require unverified unique identification, such as:

- Tracking the device
- Unique serial number

The MUI is stored in read-only locations, located between 2C0000h to 2C0013h in the DIA space. The DIA table lists the addresses of the identifier words.

**Important:** For applications that require verified unique identification, contact the Microchip Technology sales office to create a Serialized Quick Turn Programming option.

### 9.2.2 External Unique Identifier (EUI)

The EUI data is stored at locations 2C0014h-2C0023h in the program memory region. This region is an optional space for placing application specific information. The data is coded per customer requirements during manufacturing. The EUI cannot be erased by a Bulk Erase command.

**Important:** Data is stored in this address range on receiving a request from the customer. The customer may contact the local sales representative or Field Applications Engineer, and provide them the unique identifier information that is required to be stored in this region.

### 9.2.3 Standard Parameters for the Temperature Sensor

The purpose of the temperature indicator module is to provide a temperature-dependent voltage that can be measured by an analog module. The DIA table contains standard parameters for the temperature sensor for low and high range. The values are measured during test and are unique to each device. The calibration data can be used to plot the approximate sensor output voltage, $V_{TSENSE}$ vs. Temperature curve. The **"Temperature Indicator Module"** chapter explains the operation of the Temperature Indicator module and defines terms such as the low range and high range settings of the sensor.

### 9.2.4 Fixed Voltage Reference Data

The DIA stores measured FVR voltages for this device in mV for different buffer settings of 1x, 2x or 4x at program memory locations. For more information on the FVR, refer to the **"FVR - Fixed Voltage Reference"** chapter.

## 9.3 Device Configuration Information

The Device Configuration Information (DCI) is a dedicated region in the program memory mapped from 3C0000h to 3C0009h. The data stored in these location is read-only and cannot be erased. Refer to the table below for

the complete DCI table address and description. The DCI holds information about the device, which is useful for programming and Bootloader applications.

The erase size is the minimum erasable unit in the PFM, expressed as rows. The total device Flash memory capacity is (Erase size * Number of user-erasable pages).

**Table 9-3. Device Configuration Information for PIC18FxxQ43 Devices**

| ADDRESS | NAME | DESCRIPTION | VALUE | | | UNITS |
|---|---|---|---|---|---|---|
| | | | PIC18F25/45/55Q43 | PIC18F26/46/56Q43 | PIC18F27/47/57Q43 | |
| 3C0000h-3C0001h | ERSIZ | Erase page size | 128 | 128 | 128 | Words |
| 3C0002h-3C0003h | WLSIZ | Number of write latches per row | 0 | 0 | 0 | Words |
| 3C0004h-3C0005h | URSIZ | Number of user-erasable pages | 128 | 256 | 512 | Pages |
| 3C0006h-3C0007h | EESIZ | Data EEPROM memory size | 1024 | 1024 | 1024 | Bytes |
| 3C0008h-3C0009h | PCNT | Pin count | 28/40(1)/48 | 28/40(1)/48 | 28/40(1)/48 | Pins |

**Note:**
1. Pin count of 40 is also used for 44-pin part.

## 9.4 Data Memory Organization

**Important:** The operation of some aspects of data memory are changed when the PIC18 extended instruction set is enabled. See the PIC18 Instruction Execution and the Extended Instruction Set section for more information.

The data memory in PIC18 devices is implemented as static RAM. The memory space is divided into as many as 64 banks with 256 bytes each. Data Memory Map table below shows the data memory organization for all devices in the device family.

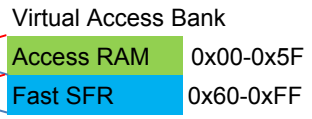
The data memory contains Special Function Registers (SFRs) and General Purpose Registers (GPRs). The SFRs are used for control and status of the controller and peripheral functions, while GPRs are used for data storage and scratchpad operations in the user's application. Any read of an unimplemented location will read as '0'.

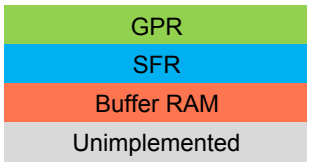
The value in the Bank Select Register (BSR) determines which bank is being accessed. The instruction set and architecture allow operations across all banks. The entire data memory may be accessed by Direct, Indirect or Indexed Addressing modes. Addressing modes are discussed later in this subsection.

To ensure that commonly used registers (SFRs and select GPRs) can be accessed in a single cycle, PIC18 devices implement an Access Bank. This is a virtual 256-byte memory space that provides fast access to SFRs and the top half of GPR Bank 5 without using the Bank Select Register. The Access Bank section provides a detailed description of the Access RAM.

**Figure 9-3. Data Memory Map**

| BanK | BSR addr[13:8] | addr[7:0] | PIC18F | | |
|---|---|---|---|---|---|
| | | | x5Q43 | x6Q43 | x7Q43 |
| 0 | 'b00 0000 | 0x00-0xFF | | | |
| 1 | 'b00 0001 | 0x00-0xFF | | | |
| 2 | 'b00 0010 | 0x00-0xFF | | | |
| 3 | 'b00 0011 | 0x00-0xFF | | | |
| 4 | 'b00 0100 | 0x00-0x5F | | | |
| | 'b00 0100 | 0x60-0xFF | | | |
| 5 | 'b00 0101 | 0x00-0x5F | | | |
| | 'b00 0101 | 0x60-0xFF | | | |
| 6 | 'b00 0110 | 0x00-0xFF | | | |
| 7 | 'b00 0111 | 0x00-0xFF | | | |
| 8 | 'b00 1000 | 0x00-0xFF | | | |
| 9 | 'b00 1001 | 0x00-0xFF | | | |
| 10 | 'b00 1010 | 0x00-0xFF | | | |
| 11 | 'b00 1011 | 0x00-0xFF | | | |
| 12 | 'b00 1100 | 0x00-0xFF | | | |
| 13 | 'b00 1101 | 0x00-0xFF | | | |
| 14 | 'b00 1110 | 0x00-0xFF | | | |
| 15 | 'b00 1111 | 0x00-0xFF | | | |
| 16 | 'b01 0000 | 0x00-0xFF | | | |
| 17 | 'b01 0001 | 0x00-0xFF | | | |
| 18 | 'b01 0010 | 0x00-0xFF | | | |
| 19 | 'b01 0011 | 0x00-0xFF | | | |
| 20 | 'b01 0100 | 0x00-0xFF | | | |
| 21 | 'b01 0101 | 0x00-0xFF | | | |
| 22 | 'b01 0110 | 0x00-0xFF | | | |
| 23 | 'b01 0111 | 0x00-0xFF | | | |
| 24 | 'b01 1000 | 0x00-0xFF | | | |
| 25 | 'b01 1001 | 0x00-0xFF | | | |
| 26 | 'b01 1010 | 0x00-0xFF | | | |
| 27 | 'b01 1011 | 0x00-0xFF | | | |
| 28 | 'b01 1100 | 0x00-0xFF | | | |
| 29 | 'b01 1101 | 0x00-0xFF | | | |
| 30 | 'b01 1110 | 0x00-0xFF | | | |
| 31 | 'b01 1111 | 0x00-0xFF | | | |
| 32 | 'b10 0000 | 0x00-0xFF | | | |
| 33 | 'b10 0001 | 0x00-0xFF | | | |
| 34 | 'b10 0010 | 0x00-0xFF | | | |
| 35 | 'b10 0011 | 0x00-0xFF | | | |
| 36 | 'b10 0100 | 0x00-0xFF | | | |
| 37 | 'b10 0101 | 0x00-0xFF | | | |
| 38 to 63 | 'b10 0110 – 'b11 1111 | 0x00-0xFF - 0x00-0xFF | | | |



### 9.4.1 Bank Select Register

To rapidly access the RAM space in PIC18 devices, the memory is split using the banking scheme. This divides the memory space into contiguous banks of 256 bytes each. Depending on the instruction, each location can be addressed directly by its full address, or an 8-bit low-order address and a bank pointer.

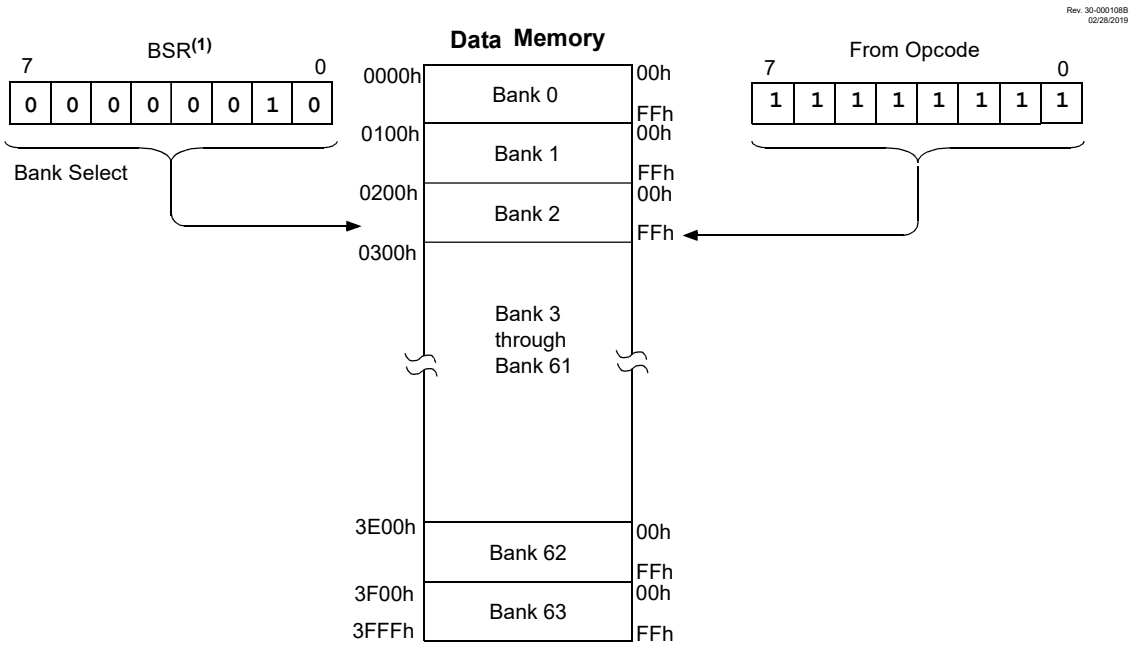
Most instructions in the PIC18 instruction set make use of the bank pointer known as the Bank Select Register (BSR). This SFR holds the Most Significant bits of a location's address; the instruction itself includes the eight Least Significant bits. The BSR can be loaded directly by using the `MOVLB` instruction.

The value of the BSR indicates the bank in data memory being accessed; the eight bits in the instruction show the location in the bank and can be thought of as an offset from the bank's lower boundary. The relationship between the BSR's value and the bank division in data memory is shown in Figure 9-4.

When writing the firmware in assembly, the user must always be careful to ensure that the proper bank is selected before performing a data read or write. When using the C compiler to write the firmware, the BSR is tracked and maintained by the compiler.

While any bank can be selected, only those banks that are actually implemented can be read or written to. Writes to unimplemented banks are ignored, while reads from unimplemented banks will return '0'. Refer Figure 9-3 for a list of implemented banks.

**Figure 9-4. Use of the Bank Select Register (Direct Addressing)**




**Note 1:** The Access RAM bit of the instruction can be used to force an override of the selected bank (BSR value) to the registers of the Access Bank.

### 9.4.2 Access Bank

While the use of the BSR with an embedded 8-bit address allows users to address the entire range of data memory, it also means that the user must always ensure that the correct bank is selected. Otherwise, data may be read from or written to the wrong location. Verifying and/or changing the BSR for each read or write to data memory can become very inefficient.

To streamline access for the most commonly used data memory locations, the data memory is configured with a virtual Access Bank, which allows users to access a mapped block of memory without specifying a BSR. The Access Bank consists of the first 96 bytes of memory in Bank 5 (`0500h-055Fh`) and the last 160 bytes of memory in Bank 4 (`0460h-04FFh`). The upper half is known as the "Access RAM" and is composed of GPRs. The lower half is where the device's SFRs are mapped. These two areas are mapped contiguously as the virtual Access Bank and can be addressed in a linear fashion by an 8-bit address (see the Data Memory Map section).

The Access Bank is used by core PIC18 instructions that include the Access RAM bit (the 'a' parameter in the instruction). When 'a' is equal to '1', the instruction uses the BSR and the 8-bit address included in the opcode for the data memory address. When 'a' is '0', the instruction ignores the BSR and uses the Access Bank address map.

Using this "forced" addressing allows the instruction to operate on a data address in a single cycle, without updating the BSR first. Access RAM also allows for faster and more code efficient context saving and switching of variables.

The mapping of the Access Bank is slightly different when the extended instruction set is enabled ($\overline{\text{XINST}}$ Configuration bit = 1). This is discussed in more detail in the Mapping the Access Bank in Indexed Liberal Offset Mode section.

## 9.5 Data Addressing Modes

**Important:** The execution of some instructions in the core PIC18 instruction set are changed when the PIC18 extended instruction set is enabled. See the Data Memory and the Extended Instruction Set section for more information.

Information in the data memory space can be addressed in several ways. For most instructions, the Addressing mode is fixed. Other instructions may use up to three modes, depending on which operands are used and whether or not the extended instruction set is enabled.

The Addressing modes are:

- Inherent
- Literal
- Direct
- Indirect

An additional Addressing mode, Indexed Literal Offset, is available when the extended instruction set is enabled (XINST Configuration bit = 1). Its operation is discussed in greater detail in the Indexed Addressing with Literal Offset section.

### 9.5.1 Inherent and Literal Addressing

Many PIC18 control instructions do not need any argument at all; they either perform an operation that globally affects the device or they operate implicitly on one register. This Addressing mode is known as Inherent Addressing. Examples include SLEEP, RESET and DAW.

Other instructions work in a similar way but require an additional explicit argument in the opcode. This is known as Literal Addressing mode because they require some literal value as an argument. Examples include ADDLW and MOVLW, which respectively, add or move a literal value to the W register. Other examples include CALL and GOTO, which include a program memory address.

### 9.5.2 Direct Addressing

Direct Addressing specifies all or part of the source and/or destination address of the operation within the opcode itself. The options are specified by the arguments accompanying the instruction.

In the core PIC18 instruction set, bit-oriented and byte-oriented instructions use some version of Direct Addressing by default. All of these instructions include some 8-bit literal address as their Least Significant Byte. This address specifies either a register address in one of the banks of data RAM (see the Data Memory Organization section) or a location in the Access Bank (see the Access Bank section) as the data source for the instruction.

The Access RAM bit 'a' determines how the address is interpreted. When 'a' is '1', the contents of the BSR (see the Bank Select Register section) are used with the address to determine the complete 12-bit address of the register. When 'a' is '0', the address is interpreted as being a register in the Access Bank.

The destination of the operation's results is determined by the destination bit 'd'. When 'd' is '1', the results are stored back in the source register, overwriting its original contents. When 'd' is '0', the results are stored in the W register.

© 2019-2023 Microchip Technology Inc. and its subsidiaries

Instructions without the ‘d’ argument have a destination that is implicit in the instruction; their destination is either the target register being operated on or the W register.

### 9.5.3 Indirect Addressing

Indirect Addressing allows the user to access a location in data memory without giving a fixed address in the instruction. This is done by using File Select Registers (FSRs) as pointers to the locations which are to be read or written. Since the FSRs are themselves located in RAM as Special File Registers, they can also be directly manipulated under program control. This makes FSRs very useful in implementing data structures, such as tables and arrays in data memory.

The registers for Indirect Addressing are also implemented with Indirect File Operands (INDFs) that permit automatic manipulation of the pointer value with auto-incrementing, auto-decrementing or offsetting with another value. This allows for efficient code, using loops, such as the following example of clearing an entire RAM bank.

**Example 9-3. How to Clear RAM (Bank 1) Using Indirect Addressing**

```
     LFSR    FSR0,100h   ; Set FSR0 to beginning of Bank1
NEXT:
     CLRF    POSTINC0    ; Clear location in Bank1 then increment FSR0

     BTFSS   FSR0H,1     ; Has high FSR0 byte incremented to next bank?
     BRA     NEXT        ; NO, clear next byte in Bank1

CONTINUE:                ; YES, continue
```

#### 9.5.3.1 FSR Registers and the INDF Operand

At the core of Indirect Addressing are three sets of registers: FSR0, FSR1 and FSR2. Each represent a pair of 8-bit registers, FSRnH and FSRnL. Each FSR pair holds the full address of the RAM location. The FSR value can address the entire range of the data memory in a linear fashion. The FSR register pairs, then, serve as pointers to data memory locations.

Indirect Addressing is accomplished with a set of Indirect File Operands, INDF0 through INDF2. These can be thought of as “virtual” registers; they are mapped in the SFR space but are not physically implemented. Reading or writing to a particular INDF register actually accesses its corresponding FSR register pair. A read from INDF1, for example, reads the data at the address indicated by FSR1H:FSR1L. Instructions that use the INDF registers as operands actually use the contents of their corresponding FSR as a pointer to the instruction’s target. The INDF operand is just a convenient way of using the pointer.

Because Indirect Addressing uses a full address, the FSR value can target any location in any bank regardless of the BSR value. However, the Access RAM bit must be cleared to zero to ensure that the INDF register in Access space is the object of the operation instead of a register in one of the other banks. The assembler default value for the Access RAM bit is zero when targeting any of the indirect operands.

#### 9.5.3.2 FSR Registers and POSTINC, POSTDEC, PREINC and PLUSW

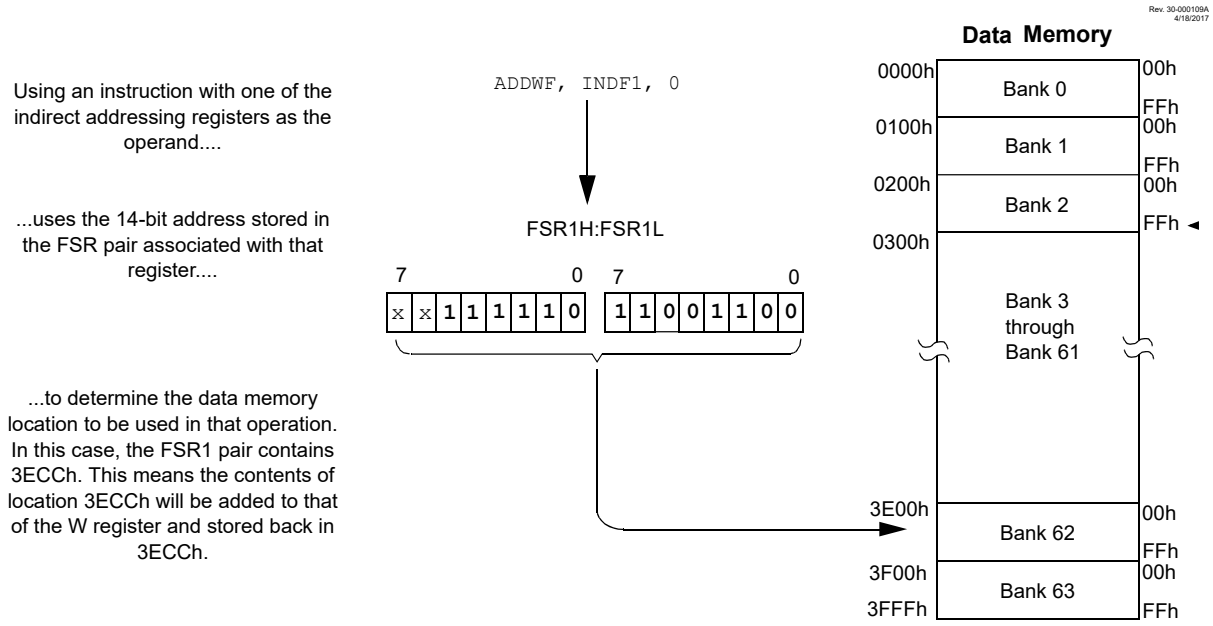
In addition to the INDF operand, each FSR register pair also has four additional indirect operands. Like INDF, these are “virtual” registers that cannot be directly read or written. Accessing these registers actually accesses the location to which the associated FSR register pair points, and also performs a specific action on the FSR value. They are:

- POSTDEC: Accesses the location to which the FSR points, then automatically decrements the FSR by 1 afterwards
- POSTINC: Accesses the location to which the FSR points, then automatically increments the FSR by 1 afterwards
- PREINC: Automatically increments the FSR by one, then uses the location to which the FSR points in the operation
- PLUSW: Adds the signed value of the W register (range of -127 to 128) to that of the FSR and uses the location to which the result points in the operation.

In this context, accessing an INDF register uses the value in the associated FSR register without changing it. Similarly, accessing a PLUSW register gives the FSR value an offset in the W register; however, neither W nor the FSR is actually changed in the operation. Accessing the other virtual registers changes the value of the FSR register.

© 2019-2023 Microchip Technology Inc. and its subsidiaries

**Figure 9-5. Indirect Addressing**




Operations on the FSRs with POSTDEC, POSTINC and PREINC affect the entire register pair; that is, rollovers of the FSRnL register from FFh to 00h carry over to the FSRnH register. On the other hand, results of these operations do not change the value of any flags in the STATUS register (e.g., Z, N, OV, etc.).

The PLUSW register can be used to implement a form of Indexed Addressing in the data memory space. By manipulating the value in the W register, users can reach addresses that are fixed offsets from pointer addresses. In some applications, this can be used to implement some powerful program control structure, such as software stacks, inside of data memory.

#### 9.5.3.3 Operations by FSRs on FSRs

Indirect Addressing operations that target other FSRs or virtual registers represent special cases. For example, using an FSR to point to one of the virtual registers will not result in successful operations. As a specific case, assume that FSR0H:FSR0L contains the address of INDF1. Attempts to read the value of the INDF1 using INDF0 as an operand will return 00h. Attempts to write to INDF1 using INDF0 as the operand will result in a `NOP`.

On the other hand, using the virtual registers to write to an FSR pair may not occur as planned. In these cases, the value will be written to the FSR pair but without any incrementing or decrementing. Thus, writing to either the INDF2 or POSTDEC2 register will write the same value to FSR2H:FSR2L.

Since the FSRs are physical registers mapped in the SFR space, they can be manipulated through all direct operations. Users need to proceed cautiously when working on these registers, particularly if their code uses Indirect Addressing.

Similarly, operations by Indirect Addressing are permitted on all other SFRs. Users need to exercise the appropriate caution that they do not inadvertently change settings that might affect the operation of the device.

## 9.6 Data Memory and the Extended Instruction Set

Enabling the PIC18 extended instruction set ($\overline{\text{XINST}}$ Configuration bit = `1`) significantly changes certain aspects of data memory and its addressing. Specifically, the use of the Access Bank for many of the core PIC18 instructions is different; this is due to the introduction of a new Addressing mode for the data memory space.

What does not change is just as important. The size of the data memory space is unchanged, as well as its linear addressing. The SFR map remains the same. Core PIC18 instructions can still operate in both Direct and Indirect

Addressing mode; inherent and literal instructions do not change at all. Indirect addressing with FSR0 and FSR1 also remain unchanged.

### 9.6.1 Indexed Addressing with Literal Offset

Enabling the PIC18 extended instruction set changes the behavior of Indirect Addressing using the FSR2 register pair within Access RAM. Under the proper conditions, instructions that use the Access Bank – that is, most bit-oriented and byte-oriented instructions – can invoke a form of Indexed Addressing using an offset specified in the instruction. This special Addressing mode is known as Indexed Addressing with Literal Offset, or Indexed Literal Offset mode.

When using the extended instruction set, this Addressing mode requires the following:

- The use of the Access Bank is forced (‘a’ = 0) and
- The file address argument is less than or equal to 5Fh.

Under these conditions, the file address of the instruction is not interpreted as the lower byte of an address (used with the BSR in Direct Addressing), or as an 8-bit address in the Access Bank. Instead, the value is interpreted as an offset value to an Address Pointer, specified by FSR2. The offset and the contents of FSR2 are added to obtain the target address of the operation.

### 9.6.2 Instructions Affected by Indexed Literal Offset Mode

Any of the core PIC18 instructions that can use Direct Addressing are potentially affected by the Indexed Literal Offset Addressing mode. This includes all byte-oriented and bit-oriented instructions, or almost one-half of the standard PIC18 instruction set. Instructions that only use Inherent or Literal Addressing modes are unaffected.

Additionally, byte-oriented and bit-oriented instructions are not affected if they do not use the Access Bank (Access RAM bit is ‘1’), or include a file address of 60h or above. Instructions meeting these criteria will continue to execute as before. A comparison of the different possible Addressing modes when the extended instruction set is enabled is shown in the following figure.

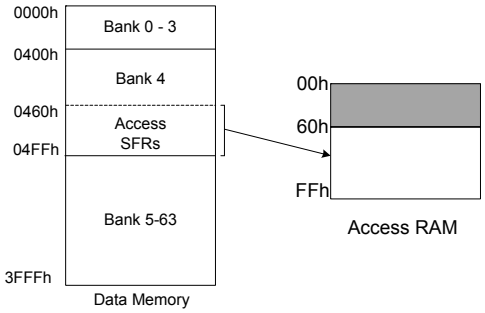
Those who desire to use byte-oriented or bit-oriented instructions in the Indexed Literal Offset mode need to note the changes to assembler syntax for this mode. This is described in more detail in the **“Extended Instruction Syntax”** section.

**Figure 9-6. Comparing Addressing Options for Bit-Oriented and Byte-Oriented Instructions (Extended Instruction Set Enabled)**

**EXAMPLE INSTRUCTION:** `ADDWF, f, d, a` (Opcode: `0010 01da ffff ffff`)

**When 'a' = 0 and f ≥ 60h**

The instruction executes in Direct Forced mode. 'f' is interpreted as a location in the Access RAM between 060h and 0FFh. This is the same as locations 460h to 4FFh (Bank4) of data memory. Locations below 60h are not available in this Addressing mode.

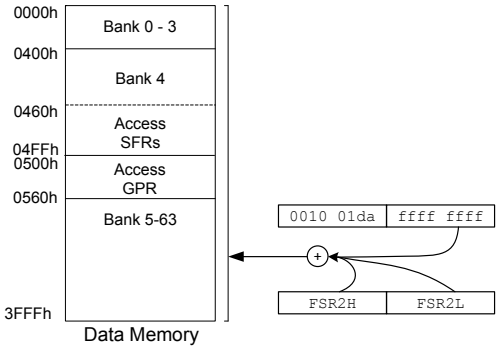



**When 'a' = 0 and f ≤ 5 Fh**

The instruction executes in Indexed Literal Offset mode. 'f' is interpreted as an offset to the address value in FSR2. The two are added together to obtain the address of the target register for the instruction. The address can be anywhere in the data memory space.

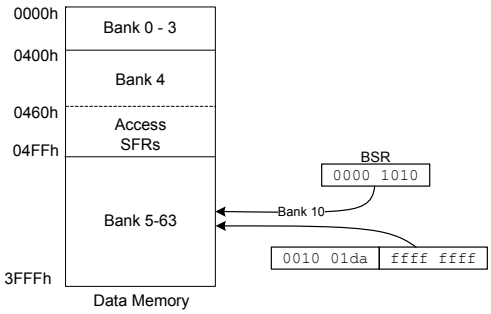
Note that in this mode, the correct syntax is now:

`ADDWF [k], d`
where 'k' is the same as 'f'.




**When 'a' = 1 (all values of f)**

The instruction executes in Direct mode (also known as Direct Long mode). 'f' is interpreted as a location in one of the 63 banks of the data memory space. The bank is designated by the Bank Select Register (BSR). The address can be in any implemented bank in the data memory space.



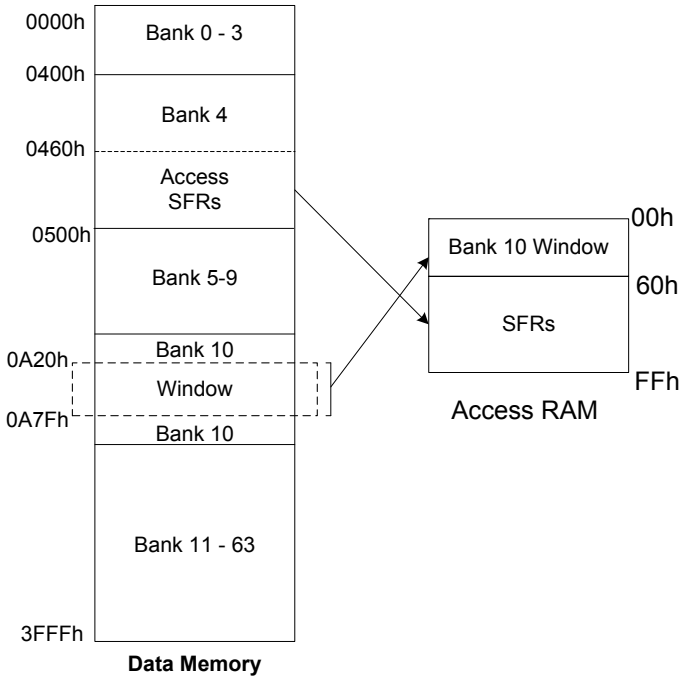

## 9.6.3 Mapping the Access Bank in Indexed Literal Offset Mode

The use of Indexed Literal Offset Addressing mode effectively changes how the first 96 locations of Access RAM (00h to 5Fh) are mapped. Rather than containing just the contents of the top section of Bank 5, this mode maps the contents from a user defined "window" that can be located anywhere in the data memory space. The value of FSR2 establishes the lower boundary of the addresses mapped into the window, while the upper boundary is defined by FSR2 plus 95 (5Fh). Addresses in the Access RAM above 5Fh are mapped as previously described (see the Access Bank section). An example of Access Bank remapping in this Addressing mode is shown in the following figure.

**Figure 9-7. Remapping the Access Bank with Indexed Literal Offset Addressing**

**EXAMPLE:**
`ADDWF, f, d, a`
FSR2H:FSR2L = `0x0A20`

Locations in the region from the FSR2 pointer (A20h) to the pointer plus 05Fh (A7Fh) are mapped to the Access RAM (000h-05Fh). Special File Registers at 460h through 4FFh are mapped to 60h through FFh, as usual. Bank 4 addresses below 5Fh can still be addressed by using the BSR.




Remapping of the Access Bank applies only to operations using the Indexed Literal Offset mode. Operations that use the BSR (Access RAM bit is '1') will continue to use Direct Addressing as before.

### 9.6.4 PIC18 Instruction Execution and the Extended Instruction Set

Enabling the extended instruction set adds additional commands to the existing PIC18 instruction set. These instructions are executed as described in the **"Extended Instruction Set"** section.

## 9.7 Register Definitions: Memory Organization

### 9.7.1 PCL

**Name:** PCL
**Address:** 0x4F9

Low byte of the Program Counter Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | PCL[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 7:0 – PCL[7:0]** Provides direct read and write access to the Program Counter

### 9.7.2 PCLAT

**Name:** PCLAT
**Address:** 0x4FA

Program Counter Latches

Holding register for bits [21:9] of the Program Counter (PC). Reads of the PCL register transfer the upper PC bits to the PCLAT register. Writes to PCL register transfer the PCLAT value to the PC.

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | | | | PCLATU[4:0] | | | | |
| Access | | | | R/W | R/W | R/W | R/W | R/W |
| Reset | | | | 0 | 0 | 0 | 0 | 0 |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | PCLATH[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 12:8 – PCLATU[4:0]** Upper PC Latch Register
Holding register for Program Counter [21:17]

**Bits 7:0 – PCLATH[7:0]** High PC Latch Register
Holding register for Program Counter [16:8]

### 9.7.3 TOS

**Name:** TOS
**Address:** 0x4FD

Top-of-Stack Register

Contents of the stack pointed to by the STKPTR register. This is the value that will be loaded into the Program Counter upon a `RETURN` or `RETFIE` instruction.

| Bit | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|
| | | | | TOS[20:16] | | | | |
| Access | | | | R/W | R/W | R/W | R/W | R/W |
| Reset | | | | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | TOS[15:8] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | TOS[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 20:0 – TOS[20:0]** Top-of-Stack

**Notes:** The individual bytes in this multibyte register can be accessed with the following register names:

- TOSU: Accesses the upper byte TOS[20:16]
- TOSH: Accesses the high byte TOS[15:8]
- TOSL: Accesses the low byte TOS[7:0]

### 9.7.4 STKPTR

**Name:** STKPTR
**Address:** 0x4FC

Stack Pointer Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | STKPTR[6:0] | | | | | | |
| Access | | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 6:0 – STKPTR[6:0]** Stack Pointer Location

### 9.7.5 WREG

**Name:** WREG
**Address:** 0x4E8

Working Data Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | WREG[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | x | x | x | x | x | x | x | x |

**Bits 7:0 – WREG[7:0]**

### 9.7.6 INDF

**Name:** INDFx
**Address:** 0x4EF,0x4E7,0x4DF

Indirect Data Register

This is a virtual register. The GPR/SFR register addressed by the FSRx register is the target for all operations involving the INDFx register.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | INDF[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 7:0 – INDF[7:0]** Indirect data pointed to by the FSRx register

### 9.7.7 POSTDEC

**Name:** POSTDECx
**Address:** 0x4ED,0x4E5,0x4DD

Indirect Data Register with post decrement

This is a virtual register. The GPR/SFR register addressed by the FSRx register is the target for all operations involving the POSTDECx register. FSRx is decrememted after the read or write operation.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | POSTDEC[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 7:0 – POSTDEC[7:0]**

### 9.7.8 POSTINC

**Name:** POSTINCx
**Address:** 0x4EE,0x4E6,0x4DE

Indirect Data Register with post increment

This is a virtual register. The GPR/SFR register addressed by the FSRx register is the target for all operations involving the POSTINCx register. FSRx is incremented after the read or write operation.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | POSTINC[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 7:0 – POSTINC[7:0]**

### 9.7.9 PREINC

**Name:** PREINCx
**Address:** 0x4EC,0x4E4,0x4DC

Indirect Data Register with pre-increment

This is a virtual register. The GPR/SFR register addressed by the FSRx register plus 1 is the target for all operations involving the PREINCx register. FSRx is incremented before the read or write operation.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | PREINC[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 7:0 – PREINC[7:0]**

### 9.7.10 PLUSW

**Name:** PLUSWx
**Address:** 0x4EB,0x4E3,0x4DB

Indirect Data Register with WREG offset

This is a virtual register. The GPR/SFR register addressed by the sum of the FSRx register plus the signed value of the W register is the target for all operations involving the PLUSWx register.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | PLUSW[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 7:0 – PLUSW[7:0]**

### 9.7.11 FSR

**Name:** FSRx
**Address:** 0x4E9,0x4E1,0x4D9

Indirect Address Register

The FSR value is the address of the data to which the INDF register points.

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | | | FSRH[5:0] | | | | | |
| Access | | | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | | | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | FSRL[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 13:8 – FSRH[5:0]** Most Significant address of INDF data

**Bits 7:0 – FSRL[7:0]** Least Significant address of INDF data

### 9.7.12 BSR

**Name:** BSR
**Address:** 0x4E0

Bank Select Register

The BSR indicates the data memory bank of the GPR address.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | BSR[5:0] | | | | | |
| Access | | | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | | | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 5:0 – BSR[5:0]** Most Significant bits of the data memory address

## 9.8 Register Summary - Memory Organization

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x04D9 | FSR2 | 7:0 | FSRL[7:0] | | | | | | | |
| | | 15:8 | | | FSRH[5:0] | | | | | |
| 0x04DB | PLUSW2 | 7:0 | PLUSW[7:0] | | | | | | | |
| 0x04DC | PREINC2 | 7:0 | PREINC[7:0] | | | | | | | |
| 0x04DD | POSTDEC2 | 7:0 | POSTDEC[7:0] | | | | | | | |
| 0x04DE | POSTINC2 | 7:0 | POSTINC[7:0] | | | | | | | |
| 0x04DF | INDF2 | 7:0 | INDF[7:0] | | | | | | | |
| 0x04E0 | BSR | 7:0 | | | BSR[5:0] | | | | | |
| 0x04E1 | FSR1 | 7:0 | FSRL[7:0] | | | | | | | |
| | | 15:8 | | | FSRH[5:0] | | | | | |
| 0x04E3 | PLUSW1 | 7:0 | PLUSW[7:0] | | | | | | | |
| 0x04E4 | PREINC1 | 7:0 | PREINC[7:0] | | | | | | | |
| 0x04E5 | POSTDEC1 | 7:0 | POSTDEC[7:0] | | | | | | | |
| 0x04E6 | POSTINC1 | 7:0 | POSTINC[7:0] | | | | | | | |
| 0x04E7 | INDF1 | 7:0 | INDF[7:0] | | | | | | | |
| 0x04E8 | WREG | 7:0 | WREG[7:0] | | | | | | | |
| 0x04E9 | FSR0 | 7:0 | FSRL[7:0] | | | | | | | |
| | | 15:8 | | | FSRH[5:0] | | | | | |
| 0x04EB | PLUSW0 | 7:0 | PLUSW[7:0] | | | | | | | |
| 0x04EC | PREINC0 | 7:0 | PREINC[7:0] | | | | | | | |
| 0x04ED | POSTDEC0 | 7:0 | POSTDEC[7:0] | | | | | | | |
| 0x04EE | POSTINC0 | 7:0 | POSTINC[7:0] | | | | | | | |
| 0x04EF | INDF0 | 7:0 | INDF[7:0] | | | | | | | |
| 0x04F0 ... 0x04F8 | Reserved | | | | | | | | | |
| 0x04F9 | PCL | 7:0 | PCL[7:0] | | | | | | | |
| 0x04FA | PCLAT | 7:0 | PCLATH[7:0] | | | | | | | |
| | | 15:8 | | | | PCLATU[4:0] | | | | |
| 0x04FC | STKPTR | 7:0 | | STKPTR[6:0] | | | | | | |
| 0x04FD | TOS | 7:0 | TOS[7:0] | | | | | | | |
| | | 15:8 | TOS[15:8] | | | | | | | |
| | | 23:16 | | | | TOS[20:16] | | | | |

# 10. NVM - Nonvolatile Memory Module

The Nonvolatile Memory (NVM) module provides run-time read and write access to the Program Flash Memory (PFM), Data Flash Memory (DFM) and Configuration bits. PFM includes the program memory and user ID space. DFM is also referred to as EEPROM which is accessed one byte at a time and the erase before write is automatic.

The Table Pointer provides read-only access to the PFM, DFM and Configuration bits. The NVM controls provide both read and write access to PFM, DFM and Configuration bits.

Reads and writes to and from the DFM are limited to single byte operations, whereas those for PFM are 16-bit word or 128-word page operations. The page buffer memory occupies one full bank of RAM space located in the RAM bank following the last occupied GPR bank. Refer to the **"Memory Organization"** chapter for more details about the buffer RAM.

The registers used for control, address and data are as follows:

- NVMCON0 - Operation start and active status
- NVMCON1 - Operation type and error status
- NVMLOCK - Write-only register to guard against accidental writes
- NVMADR - Read/write target address (multibyte register)
- NVMDAT - Read/write target data (multibyte register)
- TBLPTR - Table Pointer PFM target address for reads and buffer RAM address for writes (multibyte register)
- TABLAT - Table Pointer read/write target data (single byte register)

The write and erase times are controlled by an on-chip timer. The write and erase voltages are generated by an on-chip charge pump rated to function over the operating voltage range of the device.

PFM and DFM can be protected in two ways: code protection and write protection. Code protection (Configuration bit $\overline{CP}$) disables read and write access through an external device programmer. Write protection prevents user software writes to NVM areas tagged for protection by the $\overline{WRTn}$ Configuration bits. Code protection does not affect the self-write and erase functionality, whereas write protection does. Attempts to write a protected location will set the WRERR bit. Code protection and write protection can only be reset on a Bulk Erase performed by an external programmer.

The Bulk Erase command is used to completely erase different memory regions. The area to be erased is selected using a bit field combination. The Bulk Erase command can only be issued through an external programmer. There is no run time access for this command.

If the device is code-protected and a Bulk Erase command for the configuration memory is issued; all other memory regions are also erased. Refer to the **"Programming Specifications"** for more details.

## 10.1 Operations

NVM write operations are controlled by selecting the desired action with the NVMCMD bits and then starting the operation by executing the unlock sequence. NVM read operations are started by setting the GO bit after setting the read operation. Available NVM operations are shown in the following table.

**Table 10-1. NVM Operations**

| NVMCMD | Unlock | Operation | DFM | PFM | Source/Destination | WRERR | INT |
|---|---|---|---|---|---|---|---|
| 000 | No | Read | byte | word | NVM to NVMDAT | No | No |
| 001 | No | Read and Post Increment | byte | word | NVM to NVMDAT | No | No |
| 010 | No | Read Page | — | page | NVM to Buffer RAM | No | No |
| 011 | Yes | Write | byte | word | NVMDAT to NVM | Yes | Yes |
| 100 | Yes | Write and Post Increment | byte | word | NVMDAT to NVM | Yes | Yes |
| 101 | Yes | Write Page | — | page | Buffer RAM to NVM | Yes | Yes |

.....continued

| NVMCMD | Unlock | Operation | DFM | PFM | Source/Destination | WRERR | INT |
|---|---|---|---|---|---|---|---|
| 110 | Yes | Erase Page | — | page | n/a | Yes | Yes |
| 111 | No | Reserved (No Operation) | — | — | — | No | No |

**Important:** When the GO bit is set, writes operations are blocked on all NVM registers. The GO bit is cleared by hardware when the operation is complete. The GO bit cannot be cleared by software.

## 10.2 Unlock Sequence

As an additional layer of protection against memory corruption, a specific code execution unlock sequence is required to initiate a write or erase operation. All interrupts need to be disabled before starting the unlock sequence to ensure proper execution.

**Example 10-1. Unlock Sequence in C**

```
NVMLOCK = 0x55;
NVMLOCK = 0xAA;
NVMCON0bits.GO = 1;
```

## 10.3 Program Flash Memory (PFM)

The Program Flash Memory is readable, writable and erasable over the entire $V_{DD}$ range.

A 128-word PFM page is the only size that can be erased by user software. A Bulk Erase operation cannot be issued from user code. A read from program memory is executed either one byte, one word or a 128-word page at a time. A write to program memory can be executed as either 1 or 128 words at a time.

Writing or erasing program memory will cease instruction fetches until the operation is complete. The program memory cannot be accessed during the write or erase, so code cannot execute. An internal programming timer controls the write time of program memory writes and erases.

A value written to program memory does not need to be a valid instruction. Executing a program memory location that forms an invalid instruction results in a `NOP`.

It is important to understand the PFM memory structure for erase and programming operations. Program memory word size is 16 bits wide.

After a page has been erased, all or a portion of this page can be programmed. Data can be written directly into PFM one 16-bit word at a time using the NVMADR, NVMDAT and NVMCON1 controls or as a full page from the buffer RAM. The buffer RAM is directly accessible as any other SFR/GPR register and also may be loaded via sequential writes using the TABLAT and TBLPTR registers.

**Important:** To modify only a portion of a previously programmed page, the contents of the entire page must be read and saved in the buffer RAM prior to the page erase. The Read Page operation is the easiest way to do this. The page needs to be erased so that the new data can be written into the buffer RAM to reprogram the page of PFM. However, any unprogrammed locations can be written using the single word Write operation without first erasing the page.

© 2019-2023 Microchip Technology Inc. and its subsidiaries

### 10.3.1 Page Erase

The erase size is always 128 words. Only through the use of an external programmer can larger areas of program memory be Bulk Erased. Word erase in the program memory is not supported.

When initiating an erase sequence from user code, a page of 128 words of program memory is erased. The NVMADR[21:8] bits point to the page being erased. The NVMADR[7:0] bits are ignored. The NVMCON0 and NVMCON1 registers command the erase operation. The NVMCMD bits are set to select the erase operation. The GO bit is set to initiate the erase operation as the last step in the unlock sequence.

The NVM unlock sequence described in the Unlock Sequence section must be used; this guards against accidental writes. Instruction execution is halted during the erase cycle. The erase cycle is terminated by the internal programming timer.

The sequence of events for erasing a page of PFM is:

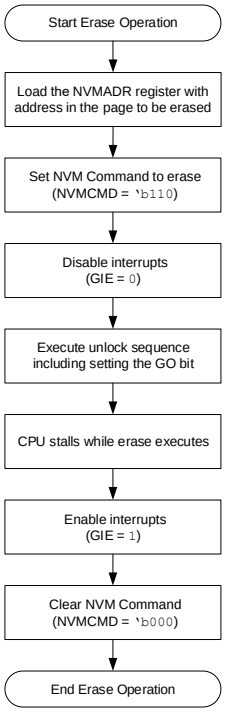
1. Set the NVMADR registers to an address within the intended page.
2. Set the NVMCMD control bits to `'b110` (Page Erase).
3. Disable all interrupts.
4. Perform the unlock sequence as described in the Unlock Sequence section.
5. Set the GO bit to start the PFM page erase.
6. Monitor the GO bit or NVMIF interrupt flag to determine when the erase has completed.
7. Interrupts can be enabled after the GO bit is clear.
8. Set the NVMCMD control bits to `'b000`.

If the PFM address is write-protected, the GO bit will be cleared, the erase operation will not take place, and the WRERR bit will be set.

While erasing the PFM page, the CPU operation is suspended and then resumes when the operation is complete. Upon erase completion, the GO bit is cleared in hardware, the NVMIF is set, and an interrupt will occur (if the NVMIE bit is set and interrupts are enabled).

The buffer RAM data are not affected by erase operations and the NVMCMD bits will remain unchanged throughout the erase opeation.

**Figure 10-1. PFM Page Erase Flowchart**



**Example 10-2. Erasing a Page of Program Flash Memory in C**

```
// Code sequence to erase one page of PFM
// PFM target address is specified by PAGE_ADDR

// Save interrupt enable bit value
uint8_t GIEBitValue = INTCON0bits.GIE;

// Load NVMADR with the base address of the memory page
NVMADR = PAGE_ADDR;
NVMCON1bits.CMD = 0x06;            // Set the page erase command
INTCON0bits.GIE = 0;               // Disable interrupts
//--------- Required Unlock Sequence ---------
NVMLOCK = 0x55;
NVMLOCK = 0xAA;
NVMCON0bits.GO = 1;                // Start page erase
//------------------------------------------------
while (NVMCON0bits.GO);            // Wait for the erase operation to complete
// Verify erase operation success and call the recovery function if needed
if (NVMCON1bits.WRERR){
   ERASE_FAULT_RECOVERY();
}

INTCON0bits.GIE = GIEBitValue;     // Restore interrupt enable bit value
NVMCON1bits.CMD = 0x00;            // Disable writes to memory
```

**Important:**

- If a write or erase operation is terminated by an unexpected Reset, the WRERR bit will be set and the user can check to decide whether a rewrite of the location(s) is needed.
- If a write or erase operation is attempted on a write-protected area, the WRERR bit will be set.
- If a write or erase operation is attempted on an invalid address location, the WRERR bit is set. (Refer to the Program and Data Memory Map in the **"Memory Organization"** chapter for more information on valid address locations.)

### 10.3.2 Page Read

PFM can be read one word or 128-word page at a time. A page is read by setting the NVMADR registers to an address within the target page and setting the NVMCMD bits to `'b010`. The page content is then transferred from PFM to the buffer RAM by starting the read operation by setting the GO bit.

The sequence of events for reading a 128-word page of PFM is:

1. Set the NVMADR registers to an address within the intended page.
2. Set the NVMCMD control bits to `'b010` (Page Read).
3. Set the GO bit to start the PFM page read.
4. Monitor the GO bit or NVMIF interrupt flag to determine when the read has completed.

**Example 10-3. Reading a Page of Program Flash Memory in C**

```
// Code sequence to read one page of PFM to Buffer Ram
// PFM target address is specified by PAGE_ADDR

// Load NVMADR with the base address of the memory page
NVMADR = PAGE_ADDR;

NVMCON1bits.CMD = 0x02;          // Set the page read command
NVMCON0bits.GO = 1;              // Start page read
while (NVMCON0bits.GO);          // Wait for the read operation to complete
```

### 10.3.3 Word Read

A single 16-bit word is read by setting the NVMADR registers to the target address and setting the NVMCMD bits to 'b000. The word is then transferred from PFM to the NVMDAT registers by starting the read operation by setting the GO bit.

The sequence of events for reading a word of PFM is:

1. Set the NVMADR registers to the target address.
2. Set the NVMCMD control bits to 'b000 (Word Read).
3. Set the GO bit to start the PFM word read.
4. Monitor the GO bit or NVMIF interrupt flag to determine when the read has completed.

**Example 10-4. Reading a Word from Program Flash Memory in C**

```
// Code sequence to read one word from PFM
// PFM target address is specified by WORD_ADDR

// Variable to store the word value from desired location in PFM
uint16_t WordValue;

// Load NVMADR with the desired word address
NVMADR = WORD_ADDR;
NVMCON1bits.CMD = 0x00;          // Set the word read command
NVMCON0bits.GO = 1;              // Start word read
while (NVMCON0bits.GO);          // Wait for the read operation to complete
WordValue = NVMDAT;              // Store the read value to a variable
```

### 10.3.4 Page Write

A page is written by first loading the buffer registers in the buffer RAM. All buffer registers are then written to PFM by setting the NVMADR to an address within the intended address range of the target PFM page, setting the NVMCMD bits to 'b101, and then executing the unlock sequence and setting the GO bit.

If the PFM address in the NVMADR is write-protected, or if NVMADR points to an invalid location, the GO bit is cleared without any effect and the WRERR bit is set.

CPU operation is suspended during a page write cycle and resumes when the operation is complete. The page write operation completes in one extended instruction cycle. When complete, the GO bit is cleared by hardware and NVMIF is set. An interrupt will occur if NVMIE is also set. The buffer registers and NVMCMD bits are not changed throughout the write operation.

The internal programming timer controls the write time. The write/erase voltages are generated by an on-chip charge pump and rated to operate over the voltage range of the device.

**Important:** Individual bytes of program memory may be modified, provided that the modification does not attempt to change any NVM bit from a '0' to a '1'. When modifying individual bytes with a page write operation, it is necessary to load all buffer registers with either 0xFF or the existing contents of memory before executing a page write operation. The fastest way to do this is by performing a page read operation.

In this device a PFM page is 128 words (256 bytes). This is the same size as one bank of general purpose RAM (GPR). This area of GPR space is dedicated as a buffer area for NVM page operations. The buffer areas for each device in the family are shown in the following table:

**Table 10-2. NVM Buffer Banks**

| Device | GPR Bank Number |
|---|---|
| PIC18Fx7Q43 | 37 |
| PIC18Fx6Q43 | 21 |
| PIC18Fx5Q43 | 13 |

There are several ways to address the data in the GPR buffer space:

- Using the TBLRD and TBLWT instructions
- Using the indirect FSR registers
- Direct read and writes to specific GPR locations

Neglecting the bank select bits, the 8 address bits of the GPR buffer space correspond to the 8 LSbs of each PFM page. In other words, there is a one-to-one correspondence between the NVMADRL register and the FSRxL register, where the x in FSRx is 0, 1 or 2.

The sequence of events for programming a page of PFM is:

1. Set the NVMADR registers to an address within the intended page.
2. Set the NVMCMD to 'b110 (Erase Page).
3. Disable all interrupts.
4. Perform the unlock sequence as described in the Unlock Sequence section.
5. Set the GO bit to start the PFM page erase.
6. Monitor the GO bit or NVMIF interrupt flag to determine when the erase has completed.
7. Set NVMCMD to 'b101 (Page Write).
8. Perform the unlock sequence.
9. Set the GO bit to start the PFM page write.
10. Monitor the GO bit or NVMIF interrupt flag to determine when the write has completed.
11. Interrupts can be enabled after the GO bit is clear.
12. Set the NVMCMD control bits to 'b000.

**Example 10-5. Writing a Page of Program Flash Memory in C**

```
// Code sequence to write a page of PFM
// Input[] is the user data that needs to be written to PFM
// PFM target address is specified by PAGE_ADDR

#define PAGESIZE 128                // PFM page size

// Save Interrupt Enable bit Value
uint8_t GIEBitValue = INTCON0bits.GIE;

// The BufferRAMStartAddr will be changed based on the device, refer
// to the "Memory Organization" chapter for more details
uint16_t bufferRAM __at(BufferRAMStartAddr);

// Defining a pointer to the first location of the Buffer RAM
uint16_t *bufferRamPtr = (uint16_t*) & bufferRAM;

//Copy application buffer contents to the Buffer RAM
for (uint8_t i = 0; i < PAGESIZE; i++) {
    *bufferRamPtr++ = Input[i];
}

// Load NVMADR with the base address of the memory page
NVMADR = PAGE_ADDR;
NVMCON1bits.CMD = 0x06;             // Set the page erase command
INTCON0bits.GIE = 0;                // Disable interrupts
//--------- Required Unlock Sequence ---------
NVMLOCK = 0x55;
NVMLOCK = 0xAA;
NVMCON0bits.GO = 1;                 // Start page erase
//--------------------------------------------
while (NVMCON0bits.GO);             // Wait for the erase operation to complete
// Verify erase operation success and call the recovery function if needed
if (NVMCON1bits.WRERR){
   ERASE_FAULT_RECOVERY();
}

// NVMADR is already pointing to target page
NVMCON1bits.CMD = 0x05;             // Set the page write command
//--------- Required Unlock Sequence ---------
NVMLOCK = 0x55;
NVMLOCK = 0xAA;
```

```
NVMCON0bits.GO = 1;                // Start page write
//-------------------------------------------------
while (NVMCON0bits.GO);            // Wait for the write operation to complete
// Verify write operation success and call the recovery function if needed
if (NVMCON1bits.WRERR){
   WRITE_FAULT_RECOVERY();
}

INTCON0bits.GIE = GIEBitValue;     // Restore interrupt enable bit value
NVMCON1bits.CMD = 0x00;            // Disable writes to memory
```

### 10.3.5 Word Write

PFM can be written one word at a time to a pre-erased memory location. Refer to the **"Word Modify"** section for more information on writing to a prewritten memory location.

A single word is written by setting the NVMADR to the target address and loading NVMDAT with the desired word. The word is then transferred to PFM by setting the NVMCMD bits to `'b011` then executing the unlock sequence and setting the GO bit.

The sequence of events for programming single word to a pre-erased location of PFM is:

1. Set the NVMADR registers to the target address.
2. Load the NVMDAT with desired word.
3. Set the NVMCMD control bits to `'b011` (Word Write).
4. Disable all interrupts.
5. Perform the unlock sequence as described in the Unlock Sequence section.
6. Set the GO bit to start the PFM word write.
7. Monitor the GO bit or NVMIF interrupt flag to determine when the write has completed.
8. Interrupts can be enabled after the GO bit is clear.
9. Set the NVMCMD control bits to `'b000`.

**Example 10-6. Writing a Word of Program Flash Memory in C**

```
// Code sequence to program one word to a pre-erased location in PFM
// PFM target address is specified by WORD_ADDR
// Target data is specified by WordValue

// Save interrupt enable bit value
uint8_t GIEBitValue = INTCON0bits.GIE;

// Load NVMADR with the target address of the word
NVMADR = WORD_ADDR;
NVMDAT = WordValue;                // Load NVMDAT with the desired value
NVMCON1bits.CMD = 0x03;            // Set the word write command
INTCON0bits.GIE = 0;               // Disable interrupts
//--------- Required Unlock Sequence ---------
NVMLOCK = 0x55;
NVMLOCK = 0xAA;
NVMCON0bits.GO = 1;                // Start word write
//-------------------------------------------------
while (NVMCON0bits.GO);            // Wait for the write operation to complete
// Verify word write operation success and call the recovery function if needed
if (NVMCON1bits.WRERR){
   WRITE_FAULT_RECOVERY();
}

INTCON0bits.GIE = GIEBitValue;     // Restore interrupt enable bit value
NVMCON1bits.CMD = 0x00;            // Disable writes to memory
```

### 10.3.6 Word Modify

Changing a word in PFM requires erasing the word before it is rewritten. However, the PFM cannot be erased by less than a page at a time. Changing a single word requires reading the page, erasing the page, and then rewriting the page with the modified word. The NVM command set includes page operations to simplify this task.

The steps necessary to change one or more words in PFM space are as follows:

1. Set the NVMADR registers to the target address.
2. Set the NVMCMD to 'b010 (Page Read).
3. Set the GO bit to start the PFM read into the GPR buffer.
4. Monitor the GO bit or NVMIF interrupt flag to determine when the read has completed.
5. Make the desired changes to the GPR buffer data.
6. Set NVMCMD to 'b110 (Page Erase).
7. Disable all interrupts.
8. Perform the unlock sequence as described in the Unlock Sequence section.
9. Set the GO bit to start the PFM page erase.
10. Monitor the GO bit or NVMIF interrupt flag to determine when the erase has completed.
11. Set NVMCMD to 'b101 (Page Write).
12. Perform the unlock sequence.
13. Set the GO bit to start the PFM page write.
14. Monitor the GO bit or NVMIF interrupt flag to determine when the write has completed.
15. Interrupts can be enabled after the GO bit is clear.
16. Set the NVMCMD control bits to 'b000.

**Example 10-7. Modifying a Word in Program Flash Memory in C**

```
// Code sequence to modify one word in a programmed page of PFM
// The variable with desired value is specified by ModifiedWord
// PFM target address is specified by WORD_ADDR
// PFM page size is specified by PAGESIZE
// The Buffer RAM start address is specified by BufferRAMStartAddr. This value
// will be changed based on the device, refer to the "Memory Organization"
//chapter for more details.

// Save Interrupt Enable bit Value
uint8_t GIEBitValue = INTCON0bits.GIE;

uint16_t bufferRAM __at(BufferRAMStartAddr);

// Defining a pointer to the first location of the Buffer RAM
uint16_t *bufferRamPtr = (uint16_t*) & bufferRAM;

// Load NVMADR with the base address of the memory page
NVMADR = WORD_ADDR;
NVMCON1bits.CMD = 0x02;            // Set the page read command
INTCON0bits.GIE = 0;               // Disable interrupts
NVMCON0bits.GO = 1;                // Start page read
while (NVMCON0bits.GO);            // Wait for the read operation to complete

// NVMADR is already pointing to target page
NVMCON1bits.CMD = 0x06;            // Set the page erase command
//--------- Required Unlock Sequence ---------
NVMLOCK = 0x55;
NVMLOCK = 0xAA;
NVMCON0bits.GO = 1;                // Start page erase
//--------------------------------------------
while (NVMCON0bits.GO);            // Wait for the erase operation to complete
// Verify erase operation success and call the recovery function if needed
if (NVMCON1bits.WRERR){
   ERASE_FAULT_RECOVERY();
}

//Modify Buffer RAM for the given word to be written to PFM
uint8_t offset = (uint8_t) ((WORD_ADDR & ((PAGESIZE * 2) - 1)) / 2);
bufferRamPtr += offset;
*bufferRamPtr = ModifiedWord;

// NVMADR is already pointing to target page
NVMCON1bits.CMD = 0x05;            // Set the page write command
//--------- Required Unlock Sequence ---------
NVMLOCK = 0x55;
NVMLOCK = 0xAA;
```

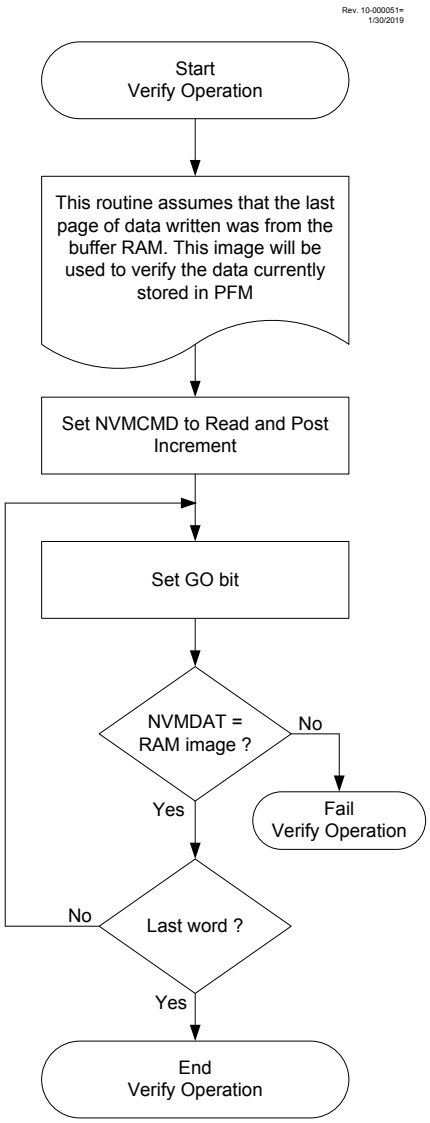
```
NVMCON0bits.GO = 1;               // Start page write
//-----------------------------------------------
while (NVMCON0bits.GO);           // Wait for the write operation to complete
// Verify write operation success and call the recovery function if needed
if (NVMCON1bits.WRERR){
   WRITE_FAULT_RECOVERY();
}

INTCON0bits.GIE = GIEBitValue;    // Restore interrupt enable bit value
NVMCON1bits.CMD = 0x00;           // Disable writes to memory
```

### 10.3.7 Write Verify

Depending on the application, good programming practice can dictate that the value written to the memory shall be verified against the original value. This can be used in applications where excessive writes can stress bits near the specification limit. Since program memory is stored as a full page, the stored program memory contents are compared with the intended data stored in the buffer RAM after the last write is complete.

**Figure 10-2. Program Flash Memory Write Verify Flowchart**



### 10.3.8 Unexpected Termination of Write Operation

If a write is terminated by an unplanned event, such as loss of power or an unexpected Reset, the memory location just programmed needs to be verified and reprogrammed, if needed. If the write operation is interrupted by a $\overline{MCLR}$ Reset or a WDT Time-out Reset during normal operation, the WRERR bit will be set, which the user can check to decide whether a rewrite of the location(s) is needed.

### 10.3.9 User ID, Device ID, Configuration Settings Access, DIA and DCI

The NVMADR value determines which NVM address space is accessed. The User IDs and Configuration areas allow read and write access, whereas Device and Revision IDs are limited to read-only.

Reading and writing User ID space is identical to reading and writing PFM space as described in the preceding paragraphs.

Writing to the Configuration bits is performed in the same manner as writing to the Data Flash Memory (DFM). Configuration settings are modified one byte at a time with the NVM Read and Write operations. When a Write operation is performed on a Configuration byte, an erase byte is performed automatically before the new byte is written. Any code protection settings that are not enabled will remain not enabled after the Write operation, unless the new values enable them. However, any code protection settings that are enabled cannot be disabled by a self-write of the configuration space. The user can modify the configuration space by the following steps:

1. Read the target Configuration byte by setting the NVMADR with the target address.
2. Retrieve the Configuration byte with the Read operation (NVMCMD = `'b000`).
3. Modify the Configuration byte in NVMDAT register.
4. Write the NVMDAT register to the Configuration byte using the Write operation (NVMCMD = `'b011`) and unlock sequence.

### 10.3.10 Table Pointer Operations

To read and write program memory, there are two operations that allow the processor to move bytes between the program memory space and the data RAM:

- Table Read (`TBLRD*`)
- Table Write (`TBLWT*`)

The SFR registers associated with these operations include:

- TABLAT register
- TBLPTR registers

The program memory space is 16 bits wide, while the data RAM space is eight bits wide. The TBLPTR registers determine the address of one byte of the NVM memory. Table reads move one byte of data from NVM space to the TABLAT register, and table writes move the TABLAT data to the buffer RAM ready for a subsequent write to NVM space with the NVM controls.

#### 10.3.10.1 Table Pointer Register

The Table Pointer (TBLPTR) register addresses a byte within the program memory. The TBLPTR comprises three SFR registers: Table Pointer Upper Byte, Table Pointer High Byte and Table Pointer Low Byte (TBLPTRU:TBLPTRH:TBLPTRL). These three registers join to form a 22-bit wide pointer (bits 0 through 21). The bits 0 through 20 allow the device to address up to 2 Mbytes of program memory space. Bit 21 allows access to the Device ID, the User ID, Configuration bits as well as the DIA and DCI.

The Table Pointer register, TBLPTR, is used by the `TBLRD` and `TBLWT` instructions. These instructions can increment and decrement TBLPTR, depending on specific appended characters shown in the following table. The increment and decrement operations on the TBLPTR affect only bits 0 through 20.

**Table 10-3. Table Pointer Operations with `TBLRD` and `TBLWT` Instructions**

| Example | Operation on Table Pointer |
|---|---|
| `TBLRD*`<br>`TBLWT*` | TBLPTR is not modified |

| | |
|---|---|
| TBLRD*+<br>TBLWT*+ | TBLPTR is incremented after the read/write |
| TBLRD*-<br>TBLWT*- | TBLPTR is decremented after the read/write |
| TBLRD+*<br>TBLWT+* | TBLPTR is incremented before the read/write |

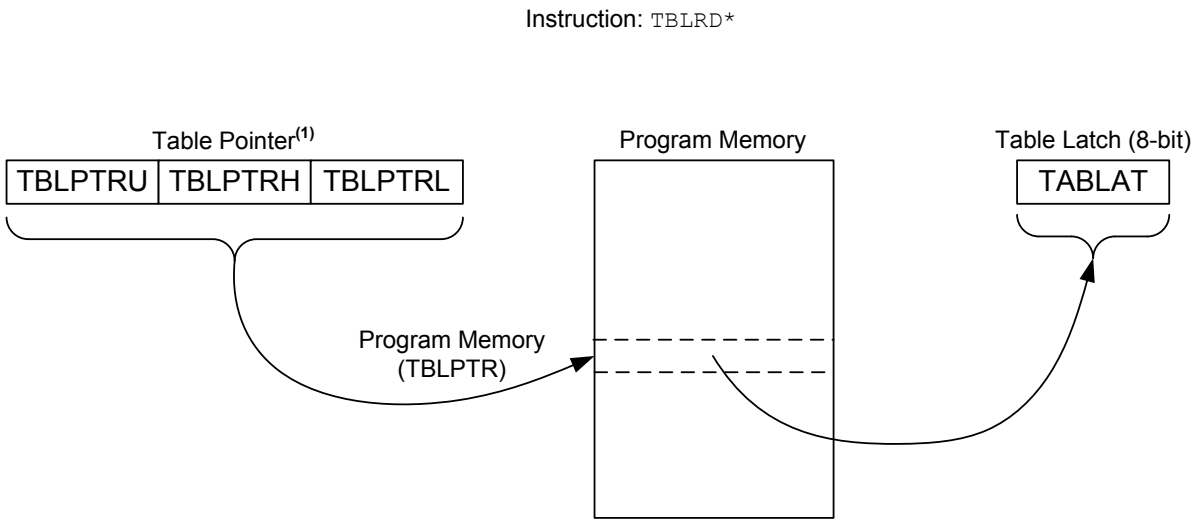
#### 10.3.10.2 Table Latch Register

The Table Latch (TABLAT) is an 8-bit register mapped into the SFR space. The Table Latch register receives one byte of NVM data resulting from a TBLRD* instruction and is the source of the 8-bit data sent to the holding register space as a result of a TBLWT* instruction.

#### 10.3.10.3 Table Read Operations

The table read operation retrieves one byte of data directly from program memory pointed to by the TBLPTR registers and places it into the TABLAT register. The following figure shows the operation of a table read.

**Figure 10-3. Table Read Operation**

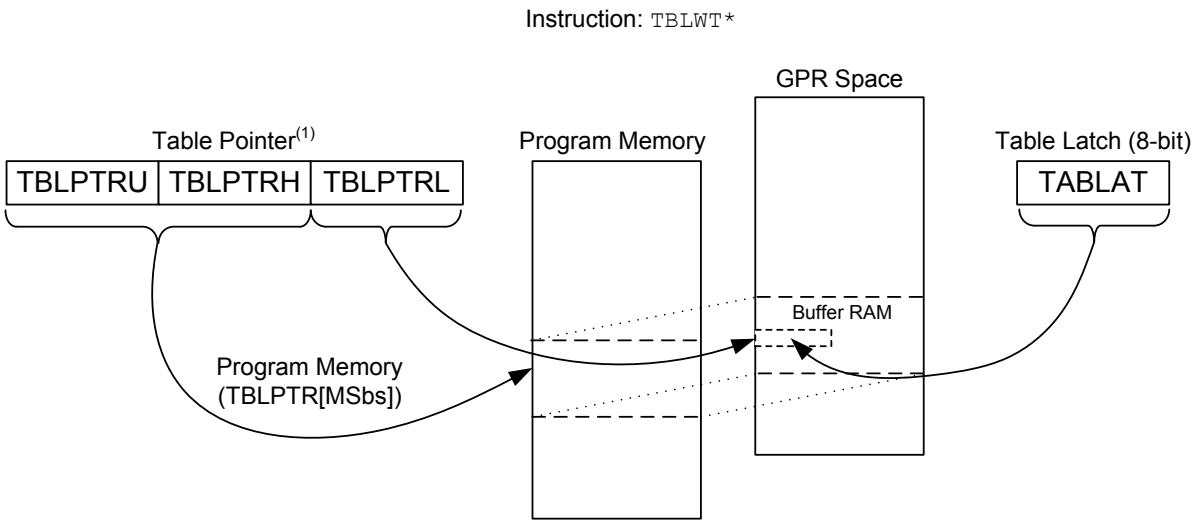



**Note:** 1. The Table Pointer register points to a byte in program memory.

#### 10.3.10.4 Table Write Operations

The table write operation stores one byte of data from the TABLAT register into a buffer RAM register. The following figure shows the operation of a table write from the TABLAT register to the buffer RAM space. The procedure to write the contents of the buffer RAM into program memory is detailed in the **"Page Write"** section.

**Figure 10-4. Table Write Operation**




**Note 1:** During table writes the Table Pointer does not point directly to program memory. TBLPTRL actually points to an address within the buffer registers. TBLPTRU:TBLPTRH points to program memory where the entire buffer space will eventually be written with the NVM commands.

Table operations work with byte entities. Tables containing data, rather than program instructions, are not required to be word-aligned. Therefore, a table can start and end at any byte address. If a table write is being used to write executable code into program memory, program instructions will need to be word-aligned.

### 10.3.10.5 Table Pointer Boundaries

The TBLPTR register is used in reads of the Program Flash Memory. Writes using the TBLPTR register go into a buffer RAM from which the data can eventually be transferred to Program Flash Memory using the NVMADR register and NVM commands.

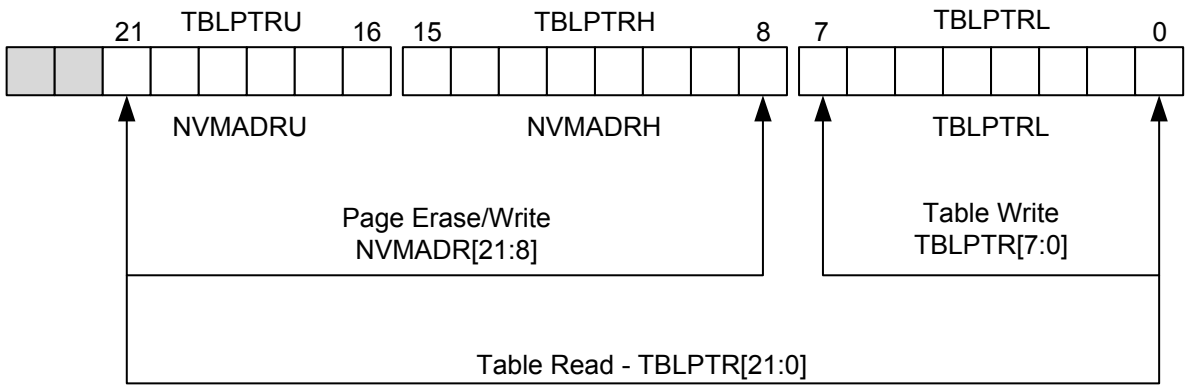
When a `TBLRD` instruction is executed, all 22 bits of the TBLPTR determine which byte is read from program memory directly into the TABLAT register.

When a `TBLWT` instruction is executed, the byte in the TABLAT register is written not to Flash memory but to a buffer register in preparation for a program memory write. All the buffer registers form a write block of size 128 words/256 bytes. The LSbs of the TBLPTR register determine to which specific address within the buffer register block the write affects. The size of the write block determines the number of LSbs that are affected. The MSbs of the TBLPTR register have no effect during `TBLWT` operations.

When a program memory page write is executed, the entire buffer register block is written to the Flash memory at the address determined by the MSbs of the NVMADR register. The LSbs are ignored during Flash memory writes.

The following figure illustrates the relevant boundaries of the TBLPTR register based on NVM operations.

**Figure 10-5. Table Pointer Boundaries Based on Operation**




**Note:**

1. Refer to the **"Memory Organization"** chapter for more details about the size of the buffer registers block.

### 10.3.10.6 Reading the Program Flash Memory

The `TBLRD` instruction retrieves data from program memory at the location to which the TBLPTR register points and places it into the TABLAT SFR register. Table reads from program memory are performed one byte at a time. The instruction set includes incrementing the TBLPTR register automatically for the next table read operation.

The CPU operation is suspended during the read, and resumes operation immediately after. From the user point of view, the value in the TABLAT register is valid in the next instruction cycle.

The internal program memory is typically organized by words. The Least Significant bit of the address selects between the high and low bytes of the word. The following figure illustrates the interface between the internal program memory and the TABLAT register.

**Figure 10-6. Reads from Program Flash Memory**

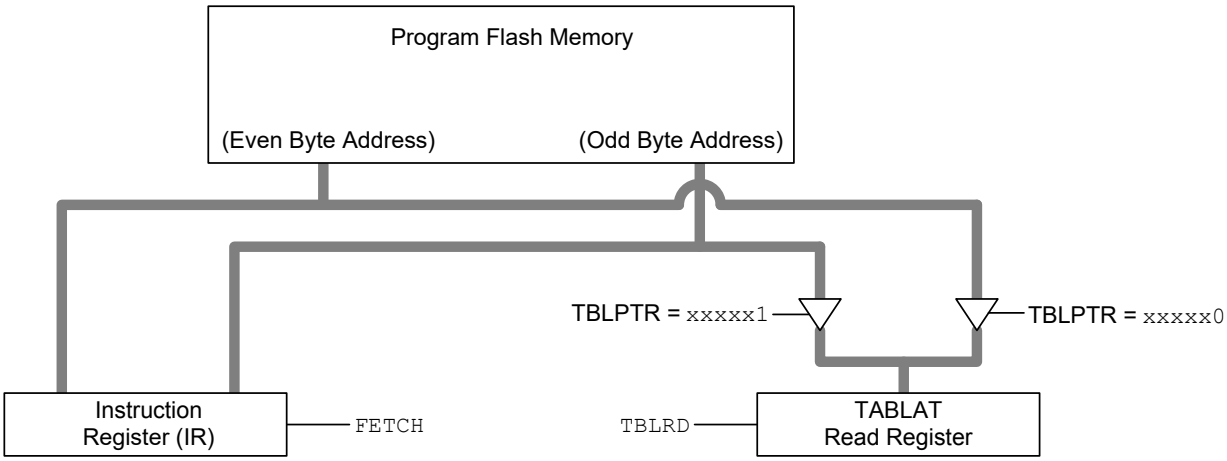
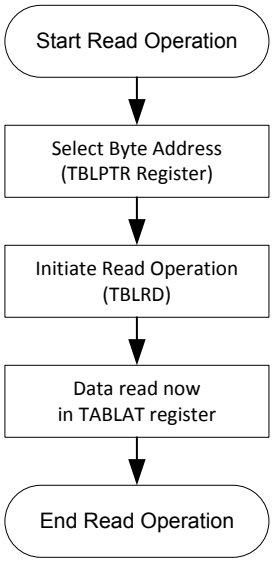


**Figure 10-7. Program Flash Memory Read Flowchart**




**Example 10-8. Reading a Program Flash Memory Word**

```
    MOVLW   CODE_ADDR_UPPER    ; Load TBLPTR with the base
    MOVWF   TBLPTRU            ; address of the word
    MOVLW   CODE_ADDR_HIGH
    MOVWF   TBLPTRH
    MOVLW   CODE_ADDR_LOW
    MOVWF   TBLPTRL
READ_WORD:
    TBLRD*+                    ; read into TABLAT and increment
    MOVF    TABLAT, W          ; get data
    MOVWF   WORD_EVEN
    TBLRD*+                    ; read into TABLAT and increment
    MOVFW   TABLAT, W          ; get data
    MOVF    WORD_ODD
```

## 10.4 Data Flash Memory (DFM)

The Data Flash Memory is a nonvolatile memory array, also referred to as EEPROM. The DFM is mapped above program memory space. The DFM can be accessed using the Table Pointer or NVM Special Function Registers (SFRs). The DFM is readable and writable during normal operation over the entire $V_{DD}$ range.

The DFM can only be read and written one byte at a time. When interfacing to the data memory block, the NVMDATL register holds the 8-bit data for read/write and the NVMADR register holds the address of the DFM location being accessed.

The DFM is rated for high erase/write cycle endurance. A byte write automatically erases the location and writes the new data (erase-before-write). The write time is controlled by an internal programming timer; it will vary with voltage and temperature as well as from device-to-device. Refer to the data EEPROM memory parameters in the **"Electrical Specifications"** chapter for the limits.

### 10.4.1 Reading the DFM

To read a DFM location, the user must write the address to the NVMADR register, set the NVMCMD bits for a single read operation (NVMCMD = 'b000), and then set the GO control bit. The data is available on the very next instruction cycle. Therefore, the NVMDATL register can be read by the next instruction. NVMDATL will hold this value until another read operation, or until it is written to by the user (during a write operation).

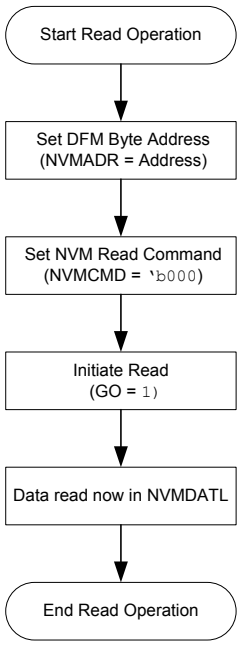
**Note:** Only byte reads are supported for DFM. Reading DFM with the Read Page operation is not supported.

The sequence of events for reading a byte of DFM is:

1. Set the NVMADR registers to an address within the intended page.
2. Set the NVMCMD control bits to 'b000 (Byte Read).
3. Set the GO bit to start the DFM byte read.
4. Monitor the GO bit or NVMIF interrupt flag to determine when the read has completed.

This process is also shown in the following flowchart.

**Figure 10-8. DFM Read Flowchart**




**Example 10-9. Reading a Byte from Data Flash Memory in C**

```
// Code sequence to read one byte from DFM
// DFM target address is specified by DFM_ADDR

// Variable to store the byte value from desired location in DFM
uint8_t ByteValue;

// Load NVMADR with the desired byte address
NVMADR = DFM_ADDR;
NVMCON1bits.CMD = 0x00;           // Set the byte read command
NVMCON0bits.GO = 1;               // Start byte read
while (NVMCON0bits.GO);           // Wait for the read operation to complete
ByteValue = NVMDATL;              // Store the read value to a variable
```

### 10.4.2 Writing to DFM

To write a DFM location, the address must first be written to the NVMADR register, the data written to the NVMDATL register, and the Write operation command set in the NVMCMD bits. The sequence shown in Unlock Sequence must be followed to initiate the write cycle. Multibyte Page writes are not supported for the DFM.

The write will not begin if the NVM unlock sequence is not exactly followed for each byte. It is strongly recommended to disable interrupts during this code segment.

When not actively writing to the DFM, the NVMCMD bits need to be kept clear at all times as an extra precaution against accidental writes. The NVMCMD bits are not cleared by hardware.

After a write sequence has been initiated, NVMCON0, NVMCON1, NVMADR and NVMDAT cannot be modified.

© 2019-2023 Microchip Technology Inc. and its subsidiaries

Each DFM write operation includes an implicit erase cycle for that byte. CPU execution continues in parallel and at the completion of the write cycle, the GO bit is cleared in hardware and the NVM Interrupt Flag (NVMIF) bit is set. The user can either enable the interrupt or poll the bit. NVMIF must be cleared by software.

The sequence of events for programming one byte of DFM is:

1. Set NVMADR registers with the target byte address.
2. Load NVMDATL register with desired byte.
3. Set the NVMCMD control bits to `'b011` (Byte Write).
4. Disable all interrupts.
5. Perform the unlock sequence as described in the Unlock Sequence section.
6. Set the GO bit to start the DFM byte write.
7. Monitor the GO bit or NVMIF interrupt flag to determine when the write has been completed.
8. Interrupts can be enabled after the GO bit is cleared.
9. Set the NVMCMD control bits to `'b000`.

**Example 10-10. Writing a Byte to Data Flash Memory in C**

```
// Code sequence to write one byte to a DFM
// DFM target address is specified by DFM_ADDR
// Target data is specified by ByteValue

// Save interrupt enable bit value
uint8_t GIEBitValue = INTCON0bits.GIE;

// Load NVMADR with the target address of the byte
NVMADR = DFM_ADDR;
NVMDATL = ByteValue;               // Load NVMDAT with the desired value
NVMCON1bits.CMD = 0x03;            // Set the byte write command
INTCON0bits.GIE = 0;               // Disable interrupts
//--------- Required Unlock Sequence ---------
NVMLOCK = 0x55;
NVMLOCK = 0xAA;
NVMCON0bits.GO = 1;                // Start byte write
//------------------------------------------------
while (NVMCON0bits.GO);            // Wait for the write operation to complete
// Verify byte write operation success and call the recovery function if needed
if (NVMCON1bits.WRERR){
   WRITE_FAULT_RECOVERY();
}

INTCON0bits.GIE = GIEBitValue;     // Restore interrupt enable bit value
NVMCON1bits.CMD = 0;               // Disable writes to memory
```

### 10.4.3 Erasing the DFM

The DFM does not support the Page Erase operation. However, the DFM can be erased by writing 0xFF to all locations in the memory that need to be erased. The simple code example bellow shows how to erase ‘n’ number of bytes in DFM. Refer to the **“Memory Organization”** chapter for more details about the DFM size and valid address locations.

**Example 10-11. Erasing n Bytes of Data Flash Memory in C**

```
// Code sequence to erase n bytes of DFM
// DFM target start address is specified by PAGE_ADDR
// Number of bytes to be eares is specified by n

// Save interrupt enable bit value
uint8_t GIEBitValue = INTCON0bits.GIE;

// Load NVMADR with the target address of the byte
NVMADR = DFM_ADDR;
NVMDATL = 0xFF;                    // Load NVMDATL with 0xFF
NVMCON1bits.CMD = 0x04;            // Set the write and post increment command
INTCON0bits.GIE = 0;               // Disable interrupts
```

```
for (uint8_t i = 0; i < n; i++}(
        NVMLOCK = 0x55;
        NVMLOCK = 0xAA;
        NVMCON0bits.GO = 1;
}

// Verify byte erase operation success and call the recovery function if needed
if (NVMCON1bits.WRERR){
   ERASE_FAULT_RECOVERY();
}

INTCON0bits.GIE = GIEBitValue;     // Restore interrupt enable bit value
NVMCON1bits.CMD = 0;               // Disable writes to memory
```

### 10.4.4 DFM Write Verify

Depending on the application, good programming practice can dictate that the value written to the memory shall be verified against the original value. This can be used in applications where excessive writes can stress bits near the specification limit to ensure that the intended values are written correctly to the specified memory locations.

### 10.4.5 Operation During Code-Protect and Write-Protect

The DFM can be code-protected using the $\overline{CP}$ Configuration bit. In-Circuit Serial Programming read and write operations are disabled when code protection is enabled. However, internal reads operate normally. Internal writes operate normally, provided that write protection is not enabled.

If the DFM is write-protected or if NVMADR points at an invalid address location, attempts to set the GO bit will fail and the WRERR bit will be set.

### 10.4.6 Protection Against Spurious Writes

A write sequence is valid only when both the following conditions are met. This prevents spurious writes that might lead to data corruption.

1. All NVM read, write and erase operations are enabled with the NVMCMD control bits. It is suggested to have the NVMCMD bits cleared at all times except during memory writes. This prevents memory operations if any of the control bits are set accidentally.
2. The NVM unlock sequence must be performed each time before all operations except the memory read operation.

## 10.5 Register Definitions: NVM

### 10.5.1 NVMCON0

**Name:** NVMCON0
**Address:** 0x040

Nonvolatile Memory Control Register 0

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | GO |
| Access | | | | | | | | R/S/HC |
| Reset | | | | | | | | 0 |

**Bit 0 – GO** Start Operation Control
Start the operation specified by the NVMCMD bits

| Value | Description |
|---|---|
| 1 | Start operation (must be set after UNLOCK sequence for all operations except READ) |
| 0 | Operation is complete |

### 10.5.2 NVMCON1

**Name:** NVMCON1
**Address:** 0x041

Nonvolatile Memory Control Register 1

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | WRERR | | | | | NVMCMD[2:0] | | |
| Access | R/C/HS | | | | | R/W | R/W | R/W |
| Reset | 0 | | | | | 0 | 0 | 0 |

**Bit 7 – WRERR** NVM Write Error
Reset States: POR = 0
All other Resets = u

| Value | Description |
|---|---|
| 1 | A write operation was interrupted by a Reset,<br>or a write or erase operation was attempted on a write-protected area,<br>or a write or erase operation was attempted on an unimplemented area,<br>or a write or erase operation was attempted while locked,<br>or a page operation was directed to a DFM area |
| 0 | All write/erase operations have completed successfully |

**Bits 2:0 – NVMCMD[2:0]** NVM Command

**Table 10-4. NVM Operations**

| NVMCMD | Unlock | Operation | DFM | PFM | Source/Destination | WRERR | INT |
|---|---|---|---|---|---|---|---|
| 000 | No | Read | byte | word | NVM to NVMDAT | No | No |
| 001 | No | Read and Post Increment | byte | word | NVM to NVMDAT | No | No |
| 010 | No | Read Page | — | page | NVM to Buffer RAM | No | No |
| 011 | Yes | Write | byte | word | NVMDAT to NVM | Yes | Yes |
| 100 | Yes | Write and Post Increment | byte | word | NVMDAT to NVM | Yes | Yes |
| 101 | Yes | Write Page | — | page | Buffer RAM to NVM | Yes | Yes |
| 110 | Yes | Erase Page | — | page | n/a | Yes | Yes |
| 111 | No | Reserved (No Operation) | — | — | — | No | No |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

### 10.5.3 NVMLOCK

**Name:** NVMLOCK
**Address:** 0x042

Nonvolatile Memory Write Restriction Control Register

NVM write and erase operations require writing 0x55 then 0xAA to this register immediately before the operation execution.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | NVMLOCK[7:0] | | | | | | | |
| Access | WO | WO | WO | WO | WO | WO | WO | WO |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 7:0 – NVMLOCK[7:0]**
Reading this register always returns ‘0’.

### 10.5.4 NVMADR

**Name:** NVMADR
**Address:** 0x043

Nonvolatile Memory Address Register

| Bit | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|
| | | | NVMADR[21:16] | | | | | |
| Access | | | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | | | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | NVMADR[15:8] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | NVMADR[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 21:0 – NVMADR[21:0]** NVM Address

**Notes:** The individual bytes in this multibyte register can be accessed with the following register names:

- NVMADRU: Accesses the upper byte NVMADR[21:16]
- NVMADRH: Accesses the high byte NVMADR[15:8]
- NVMADRL: Accesses the low byte NVMADR[7:0]

### 10.5.5 NVMDAT

**Name:** NVMDAT
**Address:** 0x046

Nonvolatile Memory Data Register

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | NVMDAT[15:8] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | NVMDAT[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 15:0 – NVMDAT[15:0]** NVM Data

**Notes:** The individual bytes in this multibyte register can be accessed with the following register names:

- NVMDATH: Accesses the high byte NVMDAT[15:8]
- NVMDATL: Accesses the low byte NVMDAT[7:0]

### 10.5.6 TBLPTR

**Name:** TBLPTR
**Address:** 0x4F6

Table Pointer Register

| Bit | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|
| | | | TBLPTR21 | TBLPTR[20:16] | | | | |
| Access | | | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | | | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | TBLPTR[15:8] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | TBLPTR[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bit 21 – TBLPTR21** NVM Most Significant Address bit

| Value | Description |
|---|---|
| 1 | Access Configuration, User ID, Device ID, and Revision ID spaces |
| 0 | Access Program Flash Memory space |

**Bits 20:0 – TBLPTR[20:0]** NVM Address bits

**Notes:** The individual bytes in this multibyte register can be accessed with the following register names:

- TBLPTRU: Accesses the upper byte TBLPTR[21:16]
- TBLPTRH: Accesses the high byte TBLPTR[15:8]
- TBLPTRL: Accesses the low byte TBLPTR[7:0]

### 10.5.7 TABLAT

**Name:** TABLAT
**Address:** 0x4F5

Table Latch Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | TABLAT[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 7:0 – TABLAT[7:0]** The value of the NVM memory byte returned from the address contained in TBLPTR after a `TBLRD` command, or the data written to the latch by a `TBLWT` command.

## 10.6 Register Summary - NVM

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x40 | NVMCON0 | 7:0 | | | | | | | | GO |
| 0x41 | NVMCON1 | 7:0 | WRERR | | | | | NVMCMD[2:0] | | |
| 0x42 | NVMLOCK | 7:0 | NVMLOCK[7:0] | | | | | | | |
| 0x43 | NVMADR | 7:0 | NVMADR[7:0] | | | | | | | |
| | | 15:8 | NVMADR[15:8] | | | | | | | |
| | | 23:16 | | | NVMADR[21:16] | | | | | |
| 0x46 | NVMDAT | 7:0 | NVMDAT[7:0] | | | | | | | |
| | | 15:8 | NVMDAT[15:8] | | | | | | | |
| 0x48 ... 0x04F4 | Reserved | | | | | | | | | |
| 0x04F5 | TABLAT | 7:0 | TABLAT[7:0] | | | | | | | |
| 0x04F6 | TBLPTR | 7:0 | TBLPTR[7:0] | | | | | | | |
| | | 15:8 | TBLPTR[15:8] | | | | | | | |
| | | 23:16 | | | TBLPTR21 | TBLPTR[20:16] | | | | |

# 11. VIC - Vectored Interrupt Controller Module

## 11.1 Overview

The Vectored Interrupt Controller (VIC) module reduces the numerous peripheral interrupt request signals to a single interrupt request signal to the CPU. This module includes the following major features:

- Interrupt Vector Table (IVT) with a unique vector for each interrupt source
- Fixed and ensured interrupt latency
- Programmable base address for IVT with lock
- Two user-selectable priority levels - High priority and low priority
- Two levels of context saving
- Interrupt state Status bits to indicate the current execution status of the CPU

The VIC module assembles all of the interrupt request signals and resolves the interrupts based on both a fixed natural order priority (i.e., determined by the IVT), and a user-assigned priority (i.e., determined by the IPRx registers), thereby eliminating scanning of interrupt sources.

## 11.2 Interrupt Control and Status Registers

The devices in this family implement the following registers for the interrupt controller:

- INTCON0, INTCON1 Control Registers
- PIRx - Peripheral Interrupt Status Registers
- PIEx - Peripheral Interrupt Enable Registers
- IPRx - Peripheral Interrupt Priority Registers
- IVTBASE Address Registers
- IVTLOCK Register

Global interrupt control functions and external interrupts are controlled from the INTCON0 register. The INTCON1 register contains the status flags for the interrupt controller.

The PIRx registers contain all of the interrupt request flags. Each source of interrupt has a Status bit, which is set by the respective peripherals or an external signal, and is either cleared via software or automatically cleared by hardware upon clearing of the interrupt condition, depending on the peripheral and bit.

The PIEx registers contain all of the interrupt enable bits. These control bits are used to individually enable interrupts from the peripherals or external signals.

The IPRx registers are used to set the interrupt priority level for each source of interrupt. Each user interrupt source can be assigned to either a high or low priority.

The IVTBASE register is user-programmable and is used to determine the start address of the IVT and the IVTLOCK register is used to prevent any unintended writes to the IVTBASE register.

There are two other Configuration bits that control the way the interrupt controller can be configured: The MVECEN and the IVT1WAY bits.

The MVECEN bit determines whether the IVT is used to determine the interrupt priorities. The IVT1WAY bit determines the number of times the IVTLOCKED bit can be cleared and set after a device Reset. See the Interrupt Vector Table Address Calculation section for details.

## 11.3 Interrupt Vector Table

The interrupt controller supports an IVT that contains the vector address location for each interrupt request source.

The IVT resides in program memory, starting at the address location determined by IVTBASE. The IVT contains one vector for each source of interrupt. Each interrupt vector location contains the starting address of the associated Interrupt Service Routine (ISR). The MVECEN Configuration bit controls the availability of the vector table.

© 2019-2023 Microchip Technology Inc. and its subsidiaries

### 11.3.1 Interrupt Vector Table Base Address (IVTBASE)

The start address of the vector table is user-programmable through the IVTBASE. The user must ensure the start address is such that it can encompass the entire vector table inside the program memory.

Each vector address is a 16-bit word (or two address locations on PIC18 devices). For 'n' interrupt sources, there are '2n' address locations necessary to hold the table, starting from IVTBASE as the first location. Thus, the starting address needs to be chosen such that the address range from IVTBASE to "IVTBASE+2n-1" can be encompassed within the program Flash memory.

For example, if the highest vector number was 81, IVTBASE needs to be chosen such that "IVTBASE+0xA1" is less than the last memory location in program Flash memory.

A programmable vector table base address is useful in situations to switch between different sets of vector tables, depending on the application. It can also be used when the application program needs to update the existing vector table (vector address values).

**Important:** It is required that the user assign an even address to IVTBASE for correct operation.

### 11.3.2 Interrupt Vector Table Contents

MVECEN = 0

When MVECEN = 0, the address location pointed to by IVTBASE has a GOTO instruction for a high-priority interrupt. Similarly, the corresponding low-priority vector also has a GOTO instruction, which is executed in case of a low-priority interrupt.

MVECEN = 1

When MVECEN = 1, the value in the vector table of each interrupt points to the address location of the first instruction of the Interrupt Service Routine, hence: ISR Location = Interrupt Vector Table entry << 2.

### 11.3.3 Interrupt Vector Table Address Calculation

MVECEN = 0

When the MVECEN Configuration bit is cleared, the address pointed to by IVTBASE is used as the high-priority interrupt vector address. The low-priority interrupt vector address is offset eight instruction words from the address in IVTBASE.

For PIC18 devices, IVTBASE defaults to 000008h, hence the high-priority interrupt vector address will be 000008h and the low-priority interrupt vector address will be 000018h.

MVECEN = 1

Each interrupt has a unique vector number associated with it, as defined in the IVT. This vector number is used for calculating the location of the interrupt vector for a particular interrupt source.

Interrupt Vector Address = IVTBASE + (2*Vector Number). This calculated interrupt vector address value is stored in the IVTAD register when an interrupt is received.

User-assigned software priority, when assigned using the IPRx registers, does not affect address calculation and is only used to resolve concurrent interrupts.

**Important:** If for any reason the address of the ISR cannot be fetched from the vector table, it will cause the system to reset and clear the Memory Execution Violation flag in the Power Control register. This can occur due to any one of the following:

- The entry for the interrupt in the vector table lies outside the executable program memory area
- ISR pointed by the vector table lies outside the executable program memory area

**Table 11-1. IVT Calculations Summary**

| IVT Address Calculation | | Interrupt Priority INTCON0 Register, IPEN Bit | |
|---|---|---|---|
| | | 0 | 1 |
| Multivector Enable, MVECEN Configuration bit | 0 | IVTBASE | High-priority IVTBASE |
| | | | Low-priority IVTBASE + 8 words |
| | 1 | IVTBASE + 2*(Vector Number) | |

### 11.3.4 Access Control for IVTBASE Registers

The interrupt controller has an IVTLOCKED bit, which can be set to avoid inadvertent changes to the contents of IVTBASE. Setting and clearing this bit requires a special sequence as an extra precaution against inadvertent changes.

To allow writes to IVTBASE, the interrupts must be disabled (GIEH = 0) and the IVTLOCKED bit must be cleared. The user must follow the sequence shown below to clear the IVTLOCKED bit.

**Example 11-1. IVT Unlock Sequence**

```
; Disable Interrupts:
    BCF INTCON0, GIE;

; Bank to IVTLOCK register
    BANKSEL IVTLOCK;
    MOVLW 55h;

; Required sequence, next 4 instructions
    MOVWF IVTLOCK;
    MOVLW AAh;
    MOVWF IVTLOCK;

; Clear IVTLOCKED bit to enable writes
    BCF IVTLOCK, IVTLOCKED;

; Enable Interrupts
    BSF INTCON0, GIE;
```

The user must follow the following sequence to set the IVTLOCKED bit.

**Example 11-2. IVT Lock Sequence**

```
; Disable Interrupts:
    BCF INTCON0, GIE;

; Bank to IVTLOCK register
    BANKSEL IVTLOCK;
    MOVLW 55h;

; Required sequence, next 4 instructions
    MOVWF IVTLOCK;
    MOVLW AAh;
    MOVWF IVTLOCK;

; Set IVTLOCKED bit to enable writes
    BSF IVTLOCK, IVTLOCKED;

; Enable Interrupts
    BSF INTCON0, GIE;
```

When the IVT1WAY Configuration bit is set, the IVTLOCKED bit can be cleared and set only once after a device Reset. The unlock operation will have no effect after the lock sequence is used to set the IVTLOCKED bit. Unlocking is inhibited until a system Reset occurs.

© 2019-2023 Microchip Technology Inc. and its subsidiaries

## 11.4 Interrupt Priority

The final priority level for any pending source of interrupt is determined first by the user-assigned priority of that source in the IPRx register, then by the natural order priority within the IVT. The sections below detail the operation of interrupt priorities.

### 11.4.1 User (Software) Priority

User-assigned interrupt priority is enabled by setting IPEN. Each peripheral interrupt source can be assigned a high- or low-priority level by the user. The user-assignable interrupt priority control bits for each interrupt are located in the IPRx registers, which are device-specific and can be found in the respective data sheet for each device.

The interrupts are serviced based on a predefined interrupt priority scheme detailed below.

1. Interrupts set by the user as a high-priority interrupt have higher precedence of execution. High-priority interrupts will override a low-priority request when:
   a. A low-priority interrupt has been requested or its request is already pending.
   b. A low and high-priority interrupt are triggered concurrently (i.e., on the same instruction cycle).(1)
   c. A low-priority interrupt was requested and the corresponding Interrupt Service Routine is currently executing. In this case, the lower priority interrupt routine will be interrupted then complete executing after the high-priority interrupt has been serviced.(2)
2. Interrupts set by the user as low priority have a lower priority of execution and are preempted by any high-priority interrupt.
3. Interrupts defined with the same software priority cannot preempt or interrupt each other. Concurrent pending interrupts with the same user priority are resolved using the natural order priority (when vectored interrupts are enabled) or in the order the interrupt flag bits are polled in the ISR (when vectored interrupts are disabled).

**Important:**

1. When a high-priority interrupt preempts a concurrent low-priority interrupt, GIEL may be cleared in the high-priority Interrupt Service Routine. If GIEL is cleared, the low-priority interrupt will NOT be serviced, even if it was originally requested. The corresponding interrupt flag needs to be cleared in user code.
2. When a high-priority interrupt is requested while a low-priority Interrupt Service Routine is executing, GIEL may be cleared in the high-priority Interrupt Service Routine. The pending low-priority interrupt will resume, even if GIEL is cleared.

### 11.4.2 Natural Order (Hardware) Priority

When vectored interrupts are enabled and more than one interrupt with the same user specified priority level is requested, the priority conflict is resolved by using a method called “Natural Order Priority”. Natural order priority is a fixed priority scheme that is based on the IVT.

**Table 11-2. Interrupt Vector Priority Table**

| Vector Number | Interrupt source | Vector Number (cont.) | Interrupt source (cont.) |
|---|---|---|---|
| 0x0 | Software Interrupt | 0x3E | PWM3RINT |
| 0x1 | HLVD (High/Low-Voltage Detect) | 0x3F | PWM3GINT |
| 0x2 | OSF (Oscillator Fail) | 0x40 | U2RX |
| 0x3 | CSW (Clock Switching) | 0x41 | U2TX |
| 0x4 | - | 0x42 | U2E |
| 0x5 | CLC1 (Configurable Logic Cell) | 0x43 | U2 |
| 0x6 | - | 0x44 | TMR5 |
| 0x7 | IOC (Interrupt-On-Change) | 0x45 | TMR5G |

.....continued

| Vector Number | Interrupt source | Vector Number (cont.) | Interrupt source (cont.) |
|---|---|---|---|
| 0x8 | INT0 | 0x46 | CCP2 |
| 0x9 | ZCD (Zero-Cross Detection) | 0x47 | SCAN |
| 0xA | AD (ADC Conversion Complete) | 0x48 | U3RX |
| 0xB | ACT (Active Clock Tuning) | 0x49 | U3TX |
| 0xC | CM1 (Comparator) | 0x4A | U3E |
| 0xD | SMT1 (Signal Measurement Timer) | 0x4B | U3 |
| 0xE | SMT1PRA | 0x4C | - |
| 0xF | SMT1PWA | 0x4D | CLC4 |
| 0x10 | ADT | 0x4E - 0x4F | - |
| 0x11 - 0x13 | - | 0x50 | INT2 |
| 0x14 | DMA1SCNT (Direct Memory Access) | 0x51 | CLC5 |
| 0x15 | DMA1DCNT | 0x52 | CWG2 (Complementary Waveform Generator) |
| 0x16 | DMA1OR | 0x53 | NCO2 |
| 0x17 | DMA1A | 0x54 | DMA3SCNT |
| 0x18 | SPI1RX (Serial Peripheral Interface) | 0x55 | DMA3DCNT |
| 0x19 | SPI1TX | 0x56 | DMA3OR |
| 0x1A | SPI1 | 0x57 | DMA3A |
| 0x1B | TMR2 | 0x58 | CCP3 |
| 0x1C | TMR1 | 0x59 | CLC6 |
| 0x1D | TMR1G | 0x5A | CWG3 |
| 0x1E | CCP1 (Capture/Compare/PWM) | 0x5B | TMR4 |
| 0x1F | TMR0 | 0x5C | DMA4SCNT |
| 0x20 | U1RX | 0x5D | DMA4DCNT |
| 0x21 | U1TX | 0x5E | DMA4OR |
| 0x22 | U1E | 0x5F | DMA4A |
| 0x23 | U1 | 0x60 | U4RX |
| 0x24 - 0x25 | - | 0x61 | U4TX |
| 0x26 | PWM1RINT | 0x62 | U4E |
| 0x27 | PWM1GINT | 0x63 | U4 |
| 0x28 | SPI2RX | 0x64 | DMA5SCNT |
| 0x29 | SPI2TX | 0x65 | DMA5DCNT |
| 0x2A | SPI2 | 0x66 | DMA5OR |
| 0x2B | - | 0x67 | DMA5A |
| 0x2C | TMR3 | 0x68 | U5RX |
| 0x2D | TMR3G | 0x69 | U5TX |
| 0x2E | PWM2RINT | 0x6A | U5E |
| 0x2F | PWM2GINT | 0x6B | U5 |
| 0x30 | INT1 | 0x6C | DMA6SCNT |
| 0x31 | CLC2 | 0x6D | DMA6DCNT |
| 0x32 | CWG1 (Complementary Waveform Generator) | 0x6E | DMA6OR |
| 0x33 | NCO1 (Numerically Controlled Oscillator) | 0x6F | DMA6A |
| 0x34 | DMA2SCNT | 0x70 | - |
| 0x35 | DMA2DCNT | 0x71 | CLC7 |

© 2019-2023 Microchip Technology Inc. and its subsidiaries

.....continued

| Vector Number | Interrupt source | Vector Number (cont.) | Interrupt source (cont.) |
|---|---|---|---|
| 0x36 | DMA2OR | 0x72 | CM2 |
| 0x37 | DMA2A | 0x73 | NCO3 |
| 0x38 | I2C1RX | 0x74 - 0x77 | - |
| 0x39 | I2C1TX | 0x78 | NVM |
| 0x3A | I2C1 | 0x79 | CLC8 |
| 0x3B | I2C1E | 0x7A | CRC (Cyclic Redundancy Check) |
| 0x3C | - | 0x7B | TMR6 |
| 0x3D | CLC3 | 0x7C - 0x8F | - |

The natural order priority scheme goes from high-to-low with increasing vector numbers, with 0 being the highest priority and decreasing from there.

For example, when two concurrently occurring interrupt sources that are both designated high priority, using the IPRx register will be resolved using the natural order priority (i.e., the interrupt with a lower corresponding vector number will preempt the interrupt with the higher vector number).

The ability for the user to assign every interrupt source to high- or low-priority levels means that the user program can give an interrupt with a low natural priority, a higher overall priority level.

## 11.5 Interrupt Operation

All pending interrupts are indicated by their respective flag bit being equal to a '1' in the PIRx register. All pending interrupts are resolved using the priority scheme explained in the Interrupt Priority section.

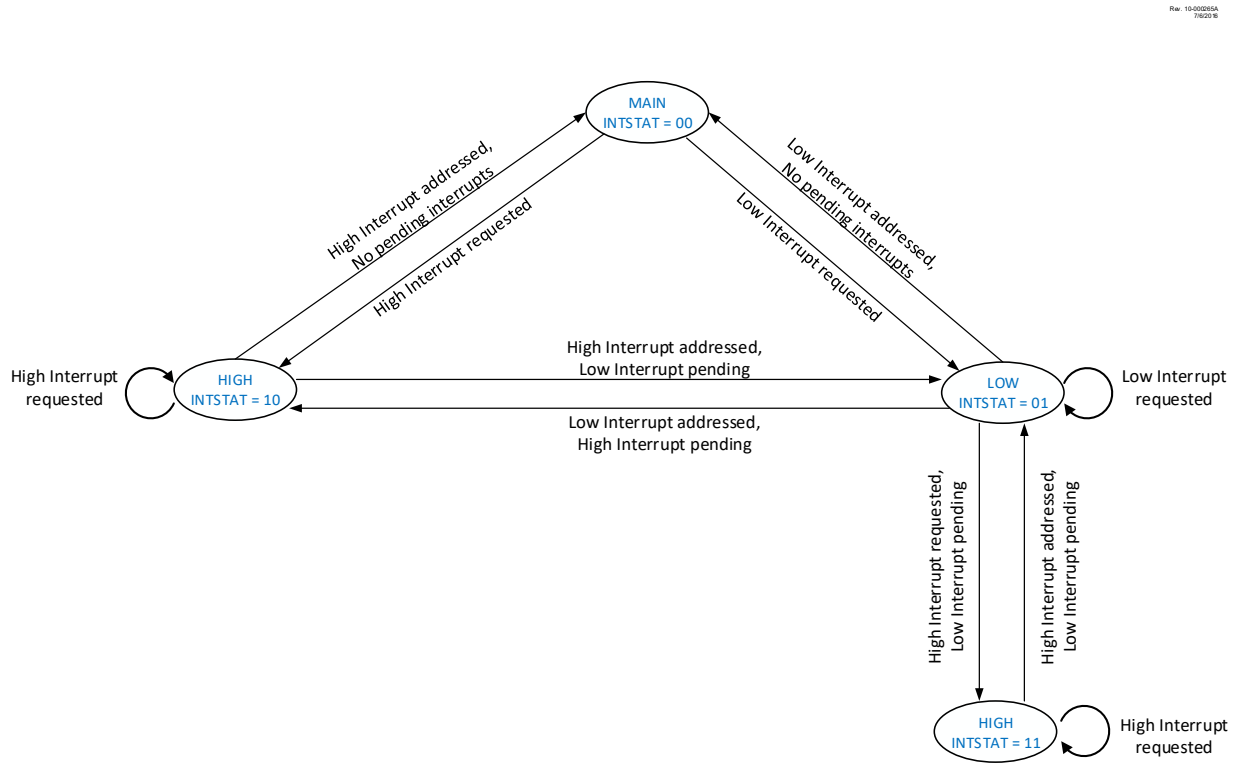
Once the interrupt source to be serviced is resolved, the program execution vectors to the resolved interrupt vector addresses, as explained in Interrupt Vector Table section. The vector number is also stored in the WREG register. Most of the flag bits are required to be cleared by the application software, but in some cases, device hardware clears the interrupt automatically. Some flag bits are read-only in the PIRx registers. These flags are a summary of the source interrupts, and the corresponding interrupt flags of the source must be cleared.

A valid interrupt can be either a high- or low-priority interrupt when in the main routine, or a high-priority interrupt when in a low-priority Interrupt Service Routine. Depending on the order of interrupt requests received and their relative timing, the CPU will be in a state of execution indicated by the STAT bit.

The state machine shown in Figure 11-1 and the subsequent sections detail the execution of interrupts when received in different orders.

**Important:** The state of GIEH/L is not changed by the hardware when servicing an interrupt. The internal state machine is used to keep track of execution states. These bits can be manipulated in the user code, resulting in transferring execution to the main routine and ignoring existing interrupts.

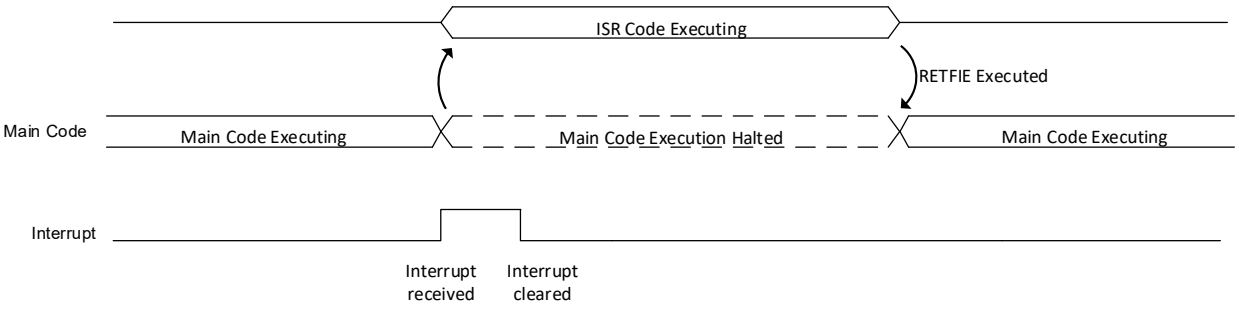
**Figure 11-1. Vectored Interrupts State Transition Diagram**




## 11.5.1 Serving a High- or Low-Priority Interrupt While the Main Routine Code Is Executing

When a high- or low-priority interrupt is requested while the main routine code is executing, the main routine execution is halted and the ISR is addressed. Upon a return from the ISR (by executing the `RETFIE` instruction), the main routine resumes execution.

**Figure 11-2. Interrupt Execution: High/Low-Priority Interrupt While Executing Main Routine**

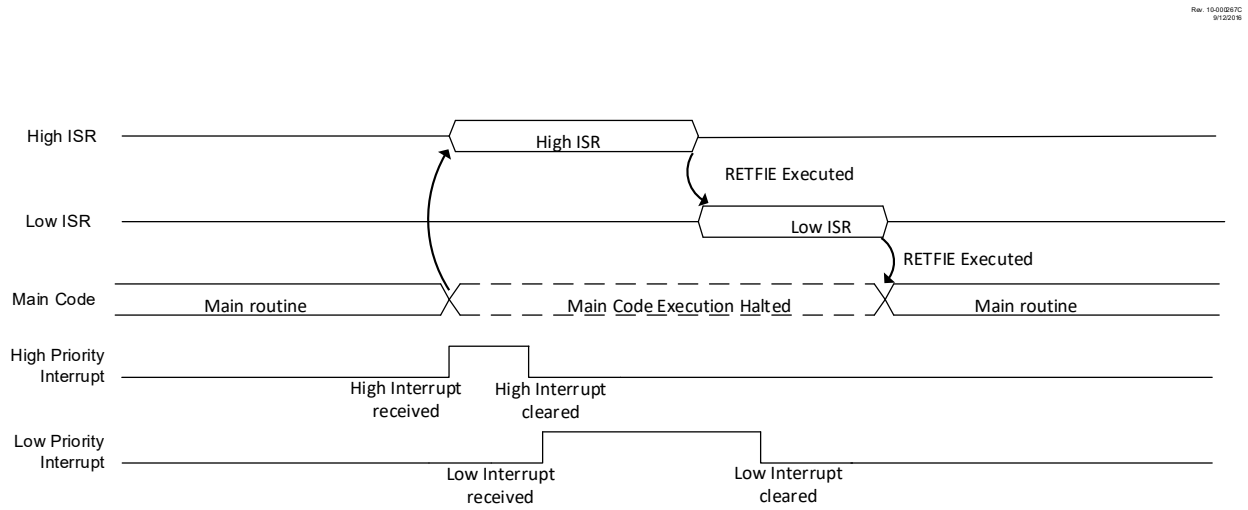



## 11.5.2 Serving a High-Priority Interrupt While a Low-Priority Interrupt Is Pending

A high priority interrupt request will always take precedence over any interrupt of a lower priority. The high-priority interrupt is acknowledged first, then the low-priority interrupt is acknowledged. Upon a return from the high-priority ISR (by executing the `RETFIE` instruction), the low-priority interrupt is serviced.

If any other high-priority interrupts are pending and enabled, they are serviced before servicing the pending low-priority interrupt. If no other high-priority interrupt requests are active, the low-priority interrupt is serviced.

**Figure 11-3. Interrupt Execution: High-Priority Interrupt with a Low-Priority Interrupt Pending**

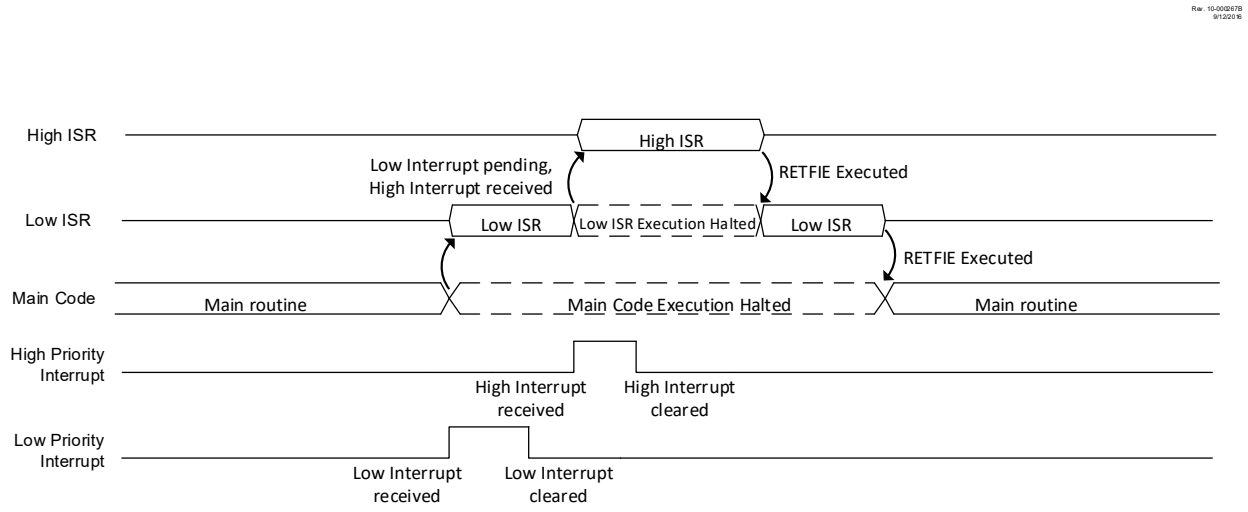



### 11.5.3 Preempting Low-Priority Interrupts

Low-priority interrupts can be preempted by high-priority interrupts. While in the low-priority ISR, if a high-priority interrupt arrives, the high-priority interrupt request is generated and the low-priority ISR is suspended, while the high-priority ISR is executed.

After the high-priority ISR is complete and if any other high-priority interrupt requests are not active, the execution returns to the preempted low-priority ISR.

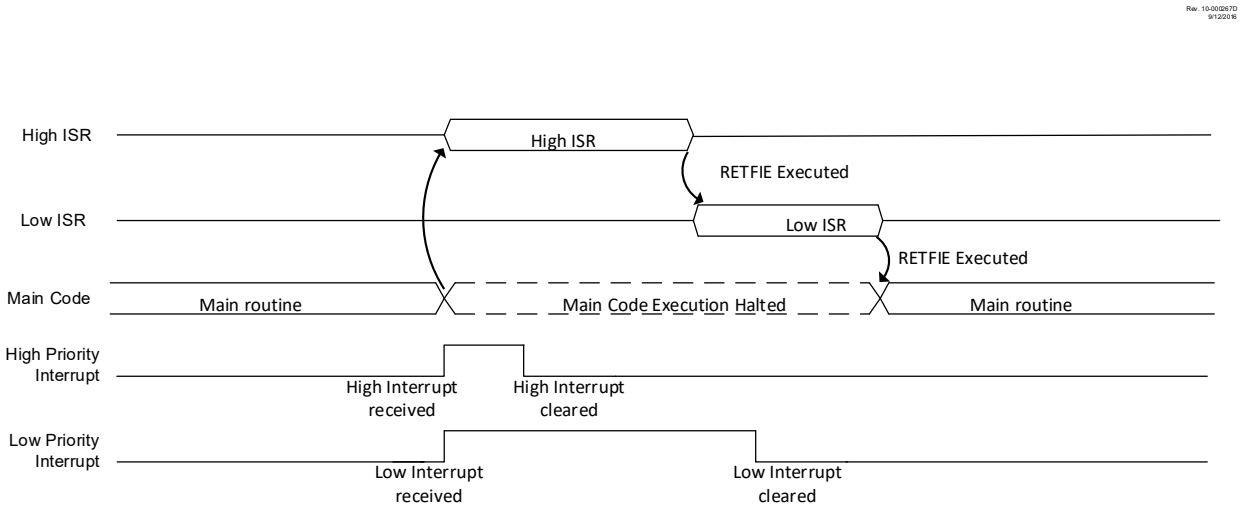
**Figure 11-4. Interrupt Execution: High-Priority Interrupt Preempting Low-Priority Interrupts**




### 11.5.4 Simultaneous High- and Low-Priority Interrupts

When both high- and low-priority interrupts are active in the same instruction cycle (i.e., simultaneous interrupt events), both the high- and low-priority requests are generated. The high-priority ISR is serviced first before servicing the low-priority interrupt.

**Figure 11-5. Interrupt Execution: Simultaneous High- and Low-Priority Interrupts**




## 11.6 Context Saving

The interrupt controller supports a two-level deep context saving system (main routine context and low ISR context). Refer to the state machine shown in Figure 11-6 for details.

The Program Counter (PC) is saved on the dedicated device PC stack. The CPU registers saved include STATUS, WREG, BSR, FSR0/1/2, PRODL/H and PCLATH/U.

After WREG has been saved to the context registers, the resolved vector number of the interrupt source to be serviced is copied into WREG. Context save and restore operation is completed by the interrupt controller based on the current state of the interrupts and the order in which they were sent to the CPU.

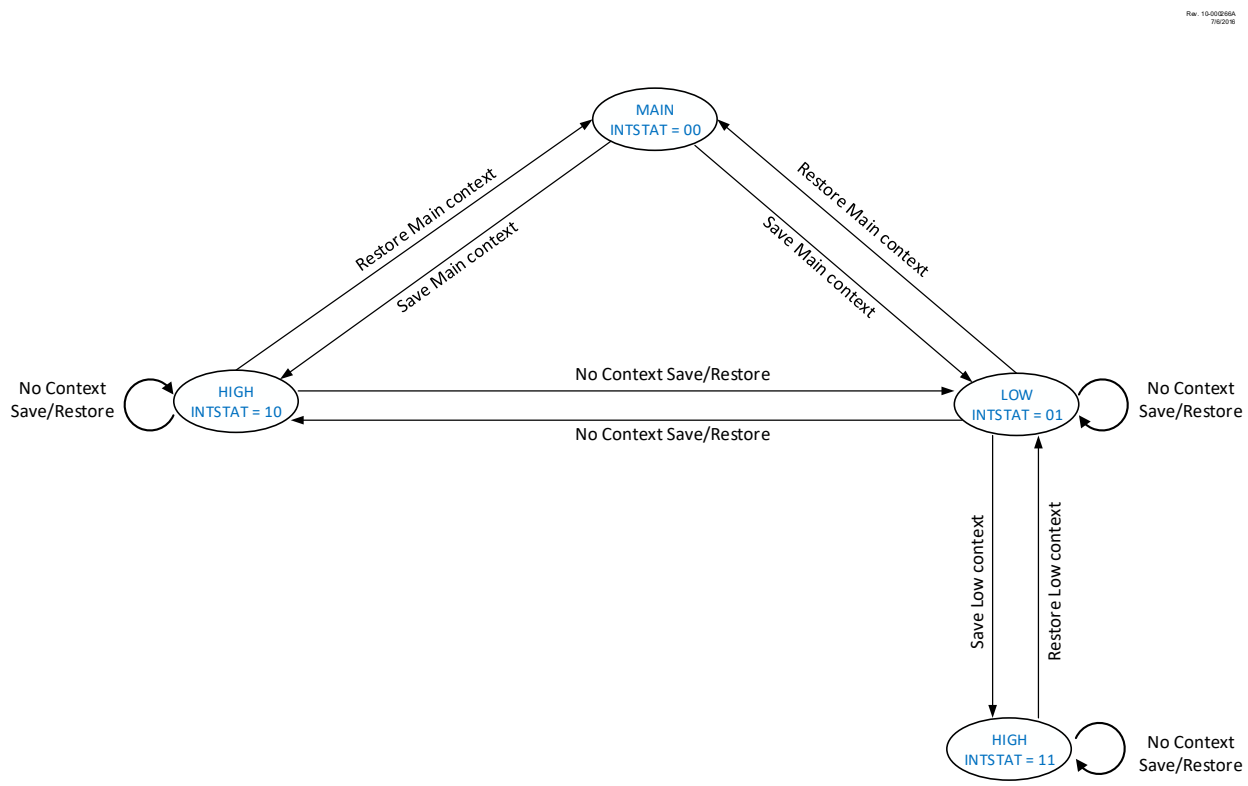
Context save/restore works the same way in both states of MVECEN. When IPEN = 0, there is only one level of interrupt active. Hence, only the main context is saved when an interrupt is received.

### 11.6.1 Accessing Shadow Registers

The interrupt controller automatically saves the context information in the shadow registers. Both the saved context values (i.e., main routine and low ISR) can be accessed using the same set of shadow registers. By clearing SHADLO, the CPU register values saved for main routine context can be accessed. Low ISR context is automatically restored to the CPU registers upon exiting the high ISR. Similarly, the main context is automatically restored to the CPU registers upon exiting the low ISR.

The shadow registers are readable and writable, so if the user desires to modify the context, then the corresponding shadow register needs to be modified and the value will be restored when exiting the ISR. Depending on the user's application, other registers may also need to be saved.

**Figure 11-6. Context Save State Machine Diagram**




## 11.7 Returning from Interrupt Service Routine (ISR)

The Return from Interrupt (`RETFIE`) instruction is used to mark the end of an ISR.

When the `RETFIE 1` instruction is executed, the PC is loaded with the saved PC value from the top of the PC stack. Saved context is also restored with the execution of this instruction. Thus, execution returns to the state of operation that existed before the interrupt occurred.

When the `RETFIE 0` instruction is executed, the saved context is not restored back to the registers.

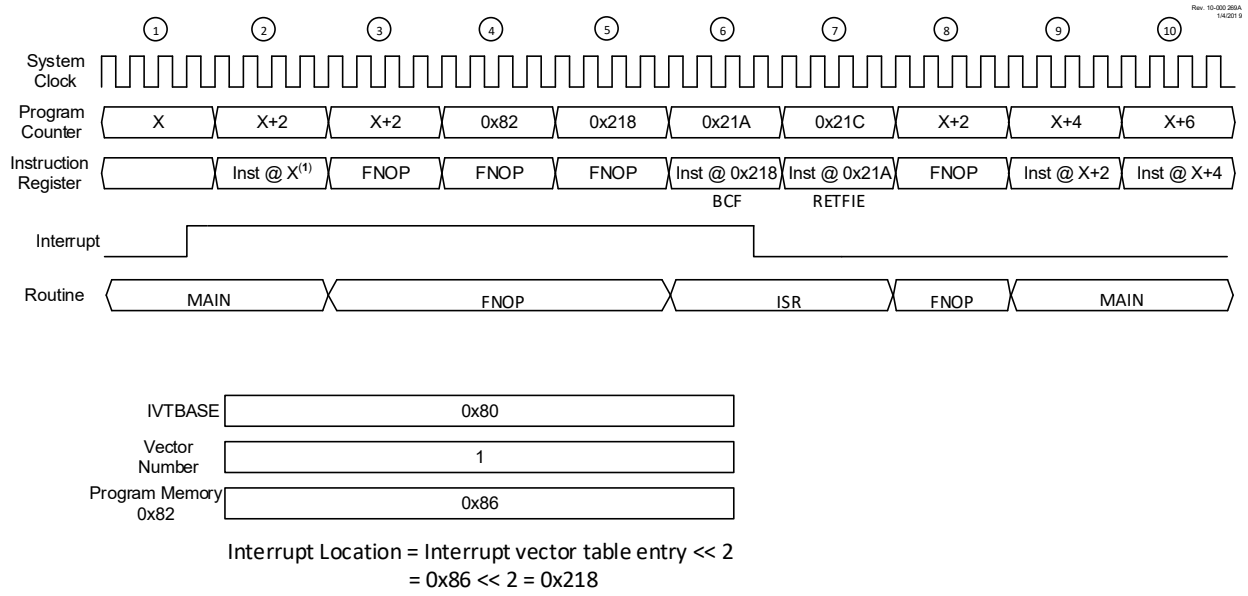
## 11.8 Interrupt Latency

When MVECEN = `1`, there is a fixed latency of three instruction cycles between the completion of the instruction active when the interrupt occurred, and the first instruction of the Interrupt Service Routine. Figure 11-7, Figure 11-8 and Figure 11-9 illustrate the sequence of events when a peripheral interrupt is asserted, when the last executed instruction is one-cycle, two-cycle and three-cycle, respectively.

After the Interrupt Flag Status bit is set, the current instruction completes executing. In the first latency cycle, the contents of the PC, STATUS, WREG, BSR, FSR0/1/2, PRODL/H and PCLATH/U registers are context saved, and the IVTBASE + Vector number is calculated. In the second latency cycle, the PC is loaded with the calculated vector table address for the interrupt source, and the starting address of the ISR is fetched. In the third latency cycle, the PC is loaded with the ISR address. All the latency cycles are executed as `NOP` instructions.

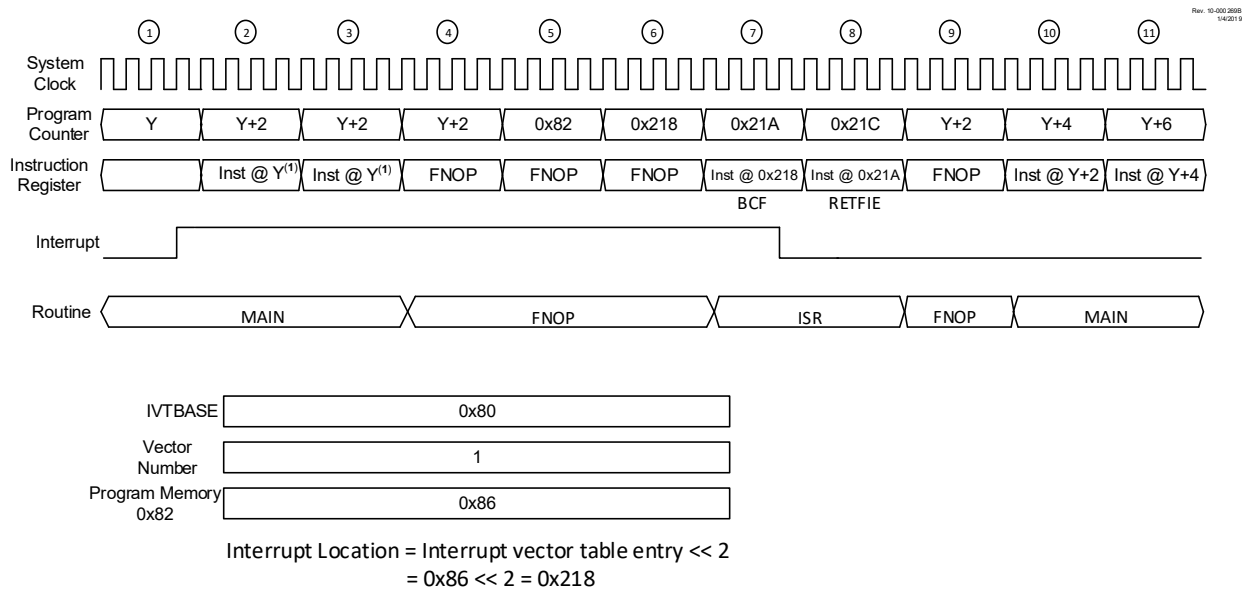
When MVECEN = `0`, the interrupt controller requires two clock cycles to vector to the ISR from the main routine. Note that, as this mode requires additional software to determine which interrupt source caused the interrupt, the actual latency between the trigger and the beginning of the specific ISR for each individual interrupt will be longer than two clock cycles and will vary, when not using vectored interrupts.

**Figure 11-7. Interrupt Timing Diagram: One-Cycle Instruction**



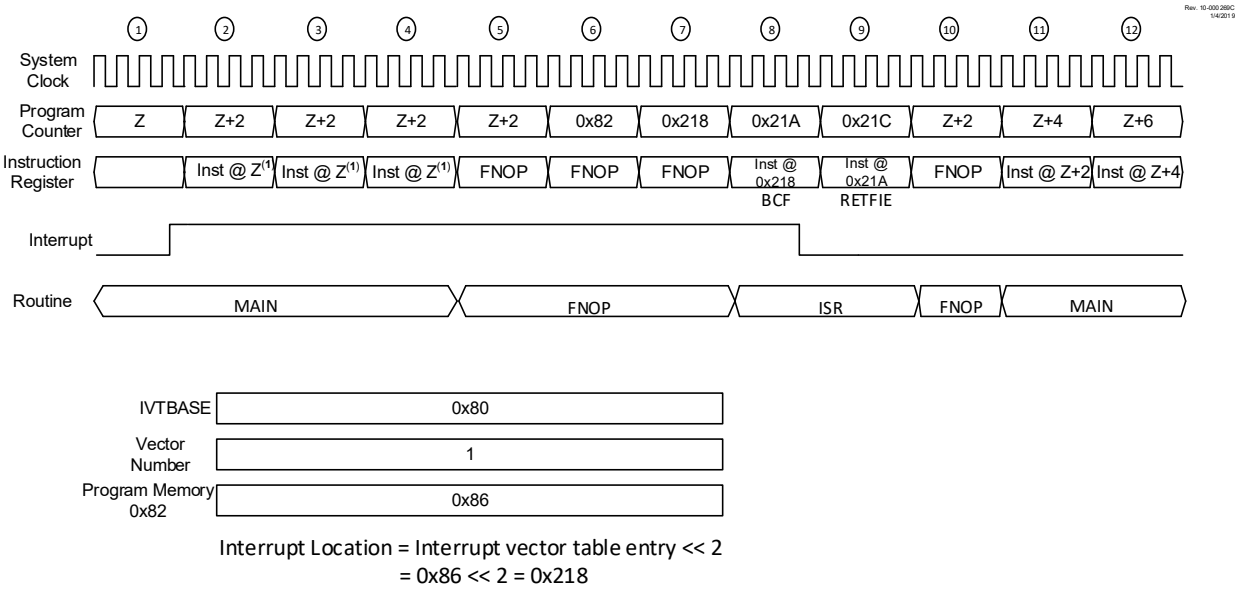

**Note**: 1. Instruction @ X is a One-cycle Instruction.

**Figure 11-8. Interrupt Timing Diagram: Two-Cycle Instruction**




**Note**: 1. Instruction @ Y is a Two-cycle Instruction.

**Figure 11-9. Interrupt Timing Diagram: Three-Cycle Instruction**




**Note**: 1. Instruction @ Z is a Three-cycle Instruction.

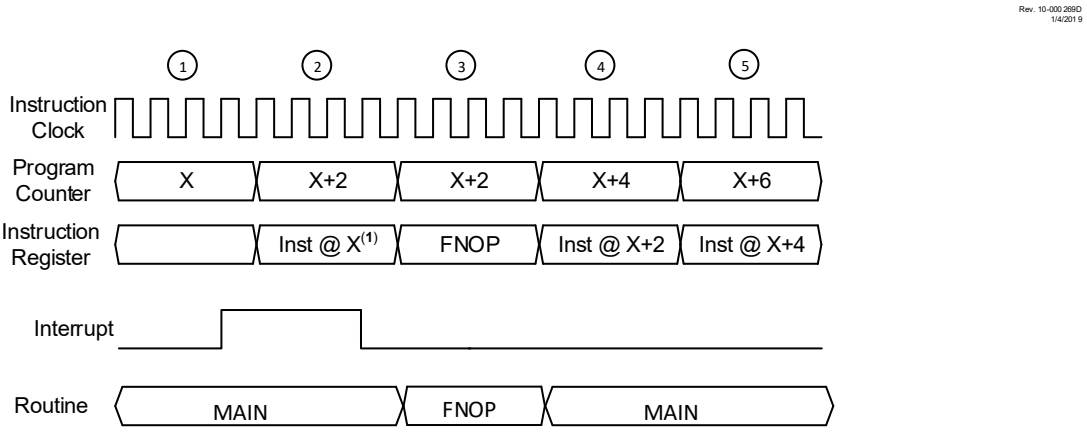
### 11.8.1 Aborting Interrupts

If the last instruction before the interrupt controller vectors to the ISR from the main routine clears the GIE, PIE, or PIR bit associated with the interrupt, the controller executes one forced `NOP` instruction cycle before it returns to the main routine.

Figure 11-10 illustrates the sequence of events when a peripheral interrupt is asserted and then cleared on the last executed instruction cycle.

If the GIE, PIE or PIR bit associated with the interrupt is cleared prior to vectoring to the ISR, then the controller continues executing the main routine.

**Figure 11-10. Interrupt Timing Diagram: Aborting Interrupts**




**Note**: 1. Inst @ X clears the interrupt flag, Example BCF INTCON0, GIE.

## 11.9 Interrupt Setup Procedure

1. When using interrupt priority levels, set IPEN and then select the user-assigned priority level for the interrupt source by writing the control bits in the appropriate IPRx control register.

**Important:** At a device Reset, the IPRx registers are initialized such that all user interrupt sources are assigned to high priority.

2. Clear the Interrupt Flag Status bit associated with the peripheral in the associated PIRx STATUS register.
3. Enable the interrupt source by setting the interrupt enable control bit associated with the source in the appropriate PIEx register.
4. If the vector table is used (MVECEN = 1), then set up the start address for the Interrupt Vector Table using IVTBASE. See the Interrupt Vector Table Contents section for more details.
5. Once IVTBASE is written to, set the interrupt enable bits in INTCON0.
6. An example of setting up interrupts and ISRs can be found below.

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

**Example 11-3. Setting Up Vectored Interrupts Using XC8**

```
// NOTE 1: If IVTBASE is changed from its default value of 0x000008, then the
// "base(...)" argument must be provided in the ISR. Otherwise the vector
// table will be placed at 0x0008 by default regardless of the IVTBASE value.

// NOTE 2: When MVECEN=0 and IPEN=1, a separate argument as "high_priority"
// or "low_priority" can be used to distinguish between the two ISRs.
// If the argument is not provided, the ISR is considered high priority
// by default.

// NOTE 3: Multiple interrupts can be handled by the same ISR if they are
// specified in the "irq(...)" argument. Ex: irq(IRQ_SW, IRQ_HLVD)

void __interrupt(irq(IRQ_SW), base(0x3008)) SW_ISR(void)
{
    PIR0bits.SWIF = 0;    // Clear the interrupt flag
    LATCbits.LATC0 ^= 1;    // ISR code goes here
}
void __interrupt(irq(default), base(0x3008)) DEFAULT_ISR(void)
{
    // Unhandled interrupts go here
}
void INTERRUPT_Initialize (void)
{
    INTCON0bits.GIEH = 1;    // Enable high priority interrupts
    INTCON0bits.GIEL = 1;    // Enable low priority interrupts
    INTCON0bits.IPEN = 1;    // Enable interrupt priority
    PIE0bits.SWIE = 1;     // Enable SW interrupt
    PIE0bits.HLVDIE = 1;     // Enable HLVD interrupt
    IPR0bits.SWIP = 0;     // Make SW interrupt low priority

    // Change IVTBASE if required
    IVTBASEU = 0x00;          // Optional
    IVTBASEH = 0x30;          // Default is 0x000008
    IVTBASEL = 0x08;
}
```

## 11.10 External Interrupt Pins

Devices may have several external interrupt sources that can be assigned to pins on different ports based on PPS settings. Refer to the **"PPS - Peripheral Pin Select Module"** chapter for possible routing options for these external interrupts. The external interrupt sources are edge-triggered. If the corresponding INTxEDG bit in INTCON0 is set, the interrupt is triggered by a rising edge. If the bit is clear, the trigger is on the falling edge.

When a valid edge appears on the INTx pin, the corresponding flag bit (INTxF in the PIRx registers) is set. This interrupt can be disabled by clearing the corresponding enable bit, INTxE. The flag bit INTxF must be cleared by software in the Interrupt Service Routine before re-enabling the interrupt.

All external interrupts can wake up the processor from Idle or Sleep modes if the INTxE bit was set prior to going into those modes. If GIE/GIEH bit is set, the processor will branch to the interrupt vector following wake-up. Interrupt priority is determined by the value contained in the respective INTxIP interrupt priority bits of the IPRx registers.

## 11.11 Wake-Up from Sleep

The interrupt controller provides a wake-up request to the CPU whenever an interrupt event occurs, if the interrupt event is enabled. This occurs regardless of whether the part is in Run, Idle/Doze or Sleep modes. The status of GIE/GIEH and GIEL bits have no effect on the wake-up request. This wake-up request is asynchronous to all clocks.

## 11.12 Interrupt Compatibility

When the MVECEN bit is cleared, the IVT feature is disabled and interrupts are compatible with previous high performance 8-bit PIC18 microcontroller devices. In this mode, the IVT priority has no effect.

© 2019-2023 Microchip Technology Inc. and its subsidiaries

When IPEN is also cleared, the interrupt priority feature is disabled and interrupts are compatible with PIC16 microcontroller midrange devices. All interrupts branch to address 0008h, since the interrupt priority is disabled.

## 11.13 Register Definitions: Interrupt Control

### 11.13.1 INTCON0

**Name:** INTCON0
**Address:** 0x4D6

Interrupt Control Register 0

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | GIE/GIEH | GIEL | IPEN | | | INT2EDG | INT1EDG | INT0EDG |
| Access | R/W | R/W | R/W | | | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | | | 1 | 1 | 1 |

**Bit 7 – GIE/GIEH** Global Interrupt Enable

| Value | Condition | Description |
|---|---|---|
| 1 | IPEN = 0 | Enables all masked interrupts |
| 0 | IPEN = 0 | Disables all interrupts |
| 1 | IPEN = 1 | Enables all unmasked high-priority interrupts: The bit also needs to be set for enabling low-priority interrupts |
| 0 | IPEN = 1 | Disables all interrupts |

**Bit 6 – GIEL** Global Low-Priority Interrupt Enable

| Value | Condition | Description |
|---|---|---|
| n | IPEN = 0 | Reserved, read as '0' |
| 1 | IPEN = 1 | Enables all unmasked low-priority interrupts, GIEH also needs to be set for low-priority interrupts |
| 0 | IPEN = 1 | Disables all low-priority interrupts |

**Bit 5 – IPEN** Interrupt Priority Enable

| Value | Description |
|---|---|
| 1 | Enable priority levels on interrupts |
| 0 | Disable priority levels on interrupts, all interrupts are treated as high-priority interrupts |

**Bit 2 – INT2EDG** External Interrupt 2 Edge Select

| Value | Description |
|---|---|
| 1 | Interrupt on rising edge of the INT2 pin |
| 0 | Interrupt on falling edge of the INT2 pin |

**Bit 1 – INT1EDG** External Interrupt 1 Edge Select

| Value | Description |
|---|---|
| 1 | Interrupt on rising edge of the INT1 pin |
| 0 | Interrupt on falling edge of the INT1 pin |

**Bit 0 – INT0EDG** External Interrupt 0 Edge Select

| Value | Description |
|---|---|
| 1 | Interrupt on rising edge of the INT0 pin |
| 0 | Interrupt on falling edge of the INT0 pin |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

### 11.13.2 INTCON1

**Name:** INTCON1
**Address:** 0x4D7

Interrupt Control Register 1

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | STAT[1:0] | | | | | | | |
| Access | R | R | | | | | | |
| Reset | 0 | 0 | | | | | | |

**Bits 7:6 – STAT[1:0]** Interrupt State Status

| Value | Description |
|---|---|
| 11 | High-priority ISR executing, high-priority interrupt was received while a low-priority ISR was executing |
| 10 | High-priority ISR executing, high-priority interrupt was received in main routine |
| 01 | Low-priority ISR executing, low-priority interrupt was received in main routine |
| 00 | Main routine executing |

© 2019-2023 Microchip Technology Inc. and its subsidiaries

### 11.13.3 IVTBASE

**Name:** IVTBASE
**Address:** 0x45D

Interrupt Vector Table Base Address Register

| Bit | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|
| | | | | IVTBASEU[4:0] | | | | |
| Access | | | | R/W | R/W | R/W | R/W | R/W |
| Reset | | | | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | IVTBASEH[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | IVTBASEL[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 20:16 – IVTBASEU[4:0]** Interrupt Vector Table Base Address Most Significant 5 bits

**Bits 15:8 – IVTBASEH[7:0]** Interrupt Vector Table Base Address Middle 8 bits

**Bits 7:0 – IVTBASEL[7:0]** Interrupt Vector Table Base Address Least Significant 8 bits

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

### 11.13.4 IVTAD

**Name:** IVTAD
**Address:** 0x45A

Interrupt Vector Table Address

| Bit | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|
| | | | | IVTADU[4:0] | | | | |
| Access | | | | R | R | R | R | R |
| Reset | | | | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | IVTADH[7:0] | | | | | | | |
| Access | R | R | R | R | R | R | R | R |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | IVTADL[7:0] | | | | | | | |
| Access | R | R | R | R | R | R | R | R |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 20:16 – IVTADU[4:0]** Interrupt Vector Table Address Most Significant 5 bits

**Bits 15:8 – IVTADH[7:0]** Interrupt Vector Table Address Middle 8 bits

**Bits 7:0 – IVTADL[7:0]** Interrupt Vector Table Address Least Significant 8 bits

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

### 11.13.5 IVTLOCK

**Name:** IVTLOCK
**Address:** 0x459

Interrupt Vector Table Lock Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | IVTLOCKED |
| Access | | | | | | | | R/W |
| Reset | | | | | | | | 0 |

**Bit 0 – IVTLOCKED** IVT Registers Lock(1,2)

| Value | Description |
|---|---|
| 1 | IVTBASE Registers are locked and cannot be written |
| 0 | IVTBASE Registers can be modified by write operations |

**Notes:**

1. The IVTLOCKED bit can only be set or cleared after the unlock sequence in Example 11-1.
2. If IVT1WAY = 1, the IVTLOCKED bit cannot be cleared after it has been set.

### 11.13.6 SHADCON

**Name:** SHADCON
**Address:** 0x376

Shadow Control Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | SHADLO |
| Access | | | | | | | | R/W |
| Reset | | | | | | | | 0 |

**Bit 0 – SHADLO** Interrupt Shadow Register Access Switch

| Value | Description |
|---|---|
| 1 | Access Main Context for Interrupt Shadow registers |
| 0 | Access Low-Priority Interrupt Context for Interrupt Shadow registers |

### 11.13.7 PIE0

**Name:** PIE0
**Address:** 0x49E

Peripheral Interrupt Enable Register 0

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | IOCIE | | CLC1IE | | CSWIE | OSFIE | HLVDIE | SWIE |
| Access | R/W | | R/W | | R/W | R/W | R/W | R/W |
| Reset | 0 | | 0 | | 0 | 0 | 0 | 0 |

**Bit 7 – IOCIE** Interrupt-on-Change Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 5 – CLC1IE** CLC1 Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 3 – CSWIE** Clock Switch Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 2 – OSFIE** Oscillator Failure Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 1 – HLVDIE** High/Low-Voltage Detect Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 0 – SWIE** Software Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

### 11.13.8 PIE1

**Name:** PIE1
**Address:** 0x49F

Peripheral Interrupt Enable Register 1

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | SMT1PWAIE | SMT1PRAIE | SMT1IE | CM1IE | ACTIE | ADIE | ZCDIE | INT0IE |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bit 7 – SMT1PWAIE** SMT1 Pulse-Width Acquisition Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 6 – SMT1PRAIE** SMT1 Period Acquisition Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 5 – SMT1IE** SMT1 Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 4 – CM1IE** CMP1 Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 3 – ACTIE** Active Clock Tuning Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 2 – ADIE** ADC Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 1 – ZCDIE** ZCD Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 0 – INT0IE** External Interrupt 0 Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

### 11.13.9 PIE2

**Name:** PIE2
**Address:** 0x4A0

Peripheral Interrupt Enable Register 2

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | DMA1AIE | DMA1ORIE | DMA1DCNTIE | DMA1SCNTIE | | | | ADTIE |
| Access | R/W | R/W | R/W | R/W | | | | R/W |
| Reset | 0 | 0 | 0 | 0 | | | | 0 |

**Bit 7 – DMA1AIE** DMA1 Abort Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 6 – DMA1ORIE** DMA1 Overrun Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 5 – DMA1DCNTIE** DMA1 Destination Count Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 4 – DMA1SCNTIE** DMA1 Source Count Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 0 – ADTIE** ADC Threshold Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

### 11.13.10 PIE3

**Name:** PIE3
**Address:** 0x4A1

Peripheral Interrupt Enable Register 3

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | TMR0IE | CCP1IE | TMR1GIE | TMR1IE | TMR2IE | SPI1IE | SPI1TXIE | SPI1RXIE |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bit 7 – TMR0IE** TMR0 Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 6 – CCP1IE** CCP1 Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 5 – TMR1GIE** TMR1 Gate Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 4 – TMR1IE** TMR1 Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 3 – TMR2IE** TMR2 Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 2 – SPI1IE** SPI1 Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 1 – SPI1TXIE** SPI1 Transmit Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 0 – SPI1RXIE** SPI1 Receive Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

### 11.13.11 PIE4

**Name:** PIE4
**Address:** 0x4A2

Peripheral Interrupt Enable Register 4

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | PWM1IE | PWM1PIE | | | U1IE | U1EIE | U1TXIE | U1RXIE |
| Access | R/W | R/W | | | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | | | 0 | 0 | 0 | 0 |

**Bit 7 – PWM1IE** PWM1 Parameter Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 6 – PWM1PIE** PWM1 Period Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 3 – U1IE** UART1 Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 2 – U1EIE** UART1 Framing Error Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 1 – U1TXIE** UART1 Transmit Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 0 – U1RXIE** UART 1 Receive Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

### 11.13.12 PIE5

**Name:** PIE5
**Address:** 0x4A3

Peripheral Interrupt Enable Register 5

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | PWM2IE | PWM2PIE | TMR3GIE | TMR3IE | | SPI2IE | SPI2TXIE | SPI2RXIE |
| Access | R/W | R/W | R/W | R/W | | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | | 0 | 0 | 0 |

**Bit 7 – PWM2IE** PWM2 Parameter Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 6 – PWM2PIE** PWM2 Period Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 5 – TMR3GIE** TMR3 Gate Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 4 – TMR3IE** TMR3 Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 2 – SPI2IE** SPI2 Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 1 – SPI2TXIE** SPI2 Transmit Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 0 – SPI2RXIE** SPI2 Receive Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

### 11.13.13 PIE6

**Name:** PIE6
**Address:** 0x4A4

Peripheral Interrupt Enable Register 6

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | DMA2AIE | DMA2ORIE | DMA2DCNTIE | DMA2SCNTIE | NCO1IE | CWG1IE | CLC2IE | INT1IE |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bit 7 – DMA2AIE** DMA2 Abort Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 6 – DMA2ORIE** DMA2 Overrun Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 5 – DMA2DCNTIE** DMA2 Destination Count Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 4 – DMA2SCNTIE** DMA2 Source Count Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 3 – NCO1IE** NCO1 Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 2 – CWG1IE** CWG1 Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 1 – CLC2IE** CLC2 Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 0 – INT1IE** External Interrupt 1 Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

### 11.13.14 PIE7

**Name:** PIE7
**Address:** 0x4A5

Peripheral Interrupt Enable Register 7

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | PWM3IE | PWM3PIE | CLC3IE | | I2C1EIE | I2C1IE | I2C1TXIE | I2C1RXIE |
| Access | R/W | R/W | R/W | | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | | 0 | 0 | 0 | 0 |

**Bit 7 – PWM3IE** PWM3 Parameter Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 6 – PWM3PIE** PWM3 Period Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 5 – CLC3IE** CLC3 Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 3 – I2C1EIE** I2C1 Error Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 2 – I2C1IE** I2C1 Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 1 – I2C1TXIE** I2C1 Transmit Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 0 – I2C1RXIE** I2C1 Receive Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

### 11.13.15 PIE8

**Name:** PIE8
**Address:** 0x4A6

Peripheral Interrupt Enable Register 8

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | SCANIE | CCP2IE | TMR5GIE | TMR5IE | U2IE | U2EIE | U2TXIE | U2RXIE |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bit 7 – SCANIE** Memory Scanner Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 6 – CCP2IE** CCP2 Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 5 – TMR5GIE** TMR5 Gate Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 4 – TMR5IE** TMR5 Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 3 – U2IE** UART2 Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 2 – U2EIE** UART2 Framing Error Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 1 – U2TXIE** UART2 Transmit Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 0 – U2RXIE** UART2 Receive Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

### 11.13.16 PIE9

**Name:** PIE9
**Address:** 0x4A7

Peripheral Interrupt Enable Register 9

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | CLC4IE | | U3IE | U3EIE | U3TXIE | U3RXIE |
| Access | | | R/W | | R/W | R/W | R/W | R/W |
| Reset | | | 0 | | 0 | 0 | 0 | 0 |

**Bit 5 – CLC4IE** CLC4 Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 3 – U3IE** UART3 Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 2 – U3EIE** UART3 Framing Error Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 1 – U3TXIE** UART3 Transmit Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 0 – U3RXIE** UART3 Receive Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

### 11.13.17 PIE10

**Name:** PIE10
**Address:** 0x4A8

Peripheral Interrupt Enable Register 10

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | DMA3AIE | DMA3ORIE | DMA3DCNTIE | DMA3SCNTIE | NCO2IE | CWG2IE | CLC5IE | INT2IE |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bit 7 – DMA3AIE** DMA3 Abort Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 6 – DMA3ORIE** DMA3 Overrun Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 5 – DMA3DCNTIE** DMA3 Destination Count Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 4 – DMA3SCNTIE** DMA3 Source Count Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 3 – NCO2IE** NCO2 Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 2 – CWG2IE** CWG2 Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 1 – CLC5IE** CLC5 Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 0 – INT2IE** External Interrupt 2 Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

### 11.13.18 PIE11

**Name:** PIE11
**Address:** 0x4A9

Peripheral Interrupt Enable Register 11

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | DMA4AIE | DMA4ORIE | DMA4DCNTIE | DMA4SCNTIE | TMR4IE | CWG3IE | CLC6IE | CCP3IE |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bit 7 – DMA4AIE** DMA4 Abort Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 6 – DMA4ORIE** DMA4 Overrun Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 5 – DMA4DCNTIE** DMA4 Destination Count Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 4 – DMA4SCNTIE** DMA4 Source Count Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 3 – TMR4IE** TMR4 Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 2 – CWG3IE** CWG3 Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 1 – CLC6IE** CLC6 Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 0 – CCP3IE** CCP3 Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

### 11.13.19 PIE12

**Name:** PIE12
**Address:** 0x4AA

Peripheral Interrupt Request Register 12

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | DMA5AIE | DMA5ORIE | DMA5DCNTIE | DMA5SCNTIE | U4IE | U4EIE | U4TXIE | U4RXIE |
| Access | R/W | R/W/HS | R/W/HS | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bit 7 – DMA5AIE** DMA5 Abort Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 6 – DMA5ORIE** DMA5 Overrun Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 5 – DMA5DCNTIE** DMA5 Destination Count Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 4 – DMA5SCNTIE** DMA5 Source Count Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 3 – U4IE** UART 4 Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 2 – U4EIE** UART4 Framing Error Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 1 – U4TXIE** UART4 Transmit Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 0 – U4RXIE** UART4 Receive Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

### 11.13.20 PIE13

**Name:** PIE13
**Address:** 0x4AB

Peripheral Interrupt Enable Register 13

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | DMA6AIE | DMA6ORIE | DMA6DCNTIE | DMA6SCNTIE | U5IE | U5EIE | U5TXIE | U5RXIE |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bit 7 – DMA6AIE** DMA6 Abort Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 6 – DMA6ORIE** DMA6 Overrun Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 5 – DMA6DCNTIE** DMA6 Destination Count Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 4 – DMA6SCNTIE** DMA6 Source Count Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 3 – U5IE** UART5 Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 2 – U5EIE** UART5 Framing Error Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 1 – U5TXIE** UART5 Transmit Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 0 – U5RXIE** UART5 Receive Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

### 11.13.21 PIE14

**Name:** PIE14
**Address:** 0x4AC

Peripheral Interrupt Enable Register 14

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | NCO3IE | CM2IE | CLC7IE | |
| Access | | | | | R/W | R/W | R/W | |
| Reset | | | | | 0 | 0 | 0 | |

**Bit 3 – NCO3IE** NCO3 Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 2 – CM2IE** CMP2 Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 1 – CLC7IE** CLC7 Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

### 11.13.22 PIE15

**Name:** PIE15
**Address:** 0x4AD

Peripheral Interrupt Enable Register 15

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | TMR6IE | CRCIE | CLC8IE | NVMIE |
| Access | | | | | R/W | R/W | R/W | R/W |
| Reset | | | | | 0 | 0 | 0 | 0 |

**Bit 3 – TMR6IE** TMR6 Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 2 – CRCIE** CRC Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 1 – CLC8IE** CLC8 Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

**Bit 0 – NVMIE** NVM Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enabled |
| 0 | Disabled |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

### 11.13.23 PIR0

**Name:** PIR0
**Address:** 0x4AE

Peripheral Interrupt Request Register 0

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | IOCIF | | CLC1IF | | CSWIF | OSFIF | HLVDIF | SWIF |
| Access | R | | R/W/HS | | R/W/HS | R/W/HS | R/W/HS | R/W |
| Reset | 0 | | 0 | | 0 | 0 | 0 | 0 |

**Bit 7 – IOCIF** Interrupt-on-Change Interrupt Flag (2)

| Value | Description |
|---|---|
| 1 | Interrupt has occurred |
| 0 | Interrupt event has not occurred |

**Bit 5 – CLC1IF** CLC1 Interrupt Flag

| Value | Description |
|---|---|
| 1 | Interrupt has occurred (must be cleared by software) |
| 0 | Interrupt event has not occurred |

**Bit 3 – CSWIF** Clock Switch Interrupt Flag(3)

| Value | Description |
|---|---|
| 1 | Interrupt has occurred (must be cleared by software) |
| 0 | Interrupt event has not occurred |

**Bit 2 – OSFIF** Oscillator Failure Interrupt Flag

| Value | Description |
|---|---|
| 1 | Interrupt has occurred (must be cleared by software) |
| 0 | Interrupt event has not occurred |

**Bit 1 – HLVDIF** High/Low-Voltage Detect Interrupt Flag

| Value | Description |
|---|---|
| 1 | Interrupt has occurred (must be cleared by software) |
| 0 | Interrupt event has not occurred |

**Bit 0 – SWIF** Software Interrupt Flag

| Value | Description |
|---|---|
| 1 | Interrupt will trigger (bit is set and cleared by user software) |
| 0 | Interrupt event has not occurred |

**Notes:**

1. Interrupt flag bits get set when an interrupt condition occurs, regardless of the state of its corresponding enable bit, or the global enable bit. User software will ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt.
2. IOCIF is a read-only bit. To clear the interrupt condition, all bits in the IOCxF registers must be cleared.
3. The CSWIF interrupt will not wake the system from Sleep mode. The system will sleep until another interrupt causes the wake-up.

### 11.13.24 PIR1

**Name:** PIR1
**Address:** 0x4AF

Peripheral Interrupt Request Register 1

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | SMT1PWAIF | SMT1PRAIF | SMT1IF | CM1IF | ACTIF | ADIF | ZCDIF | INT0IF |
| Access | R/W/HS | R/W/HS | R/W/HS | R/W/HS | R/W/HS | R/W/HS | R/W/HS | R/W/HS |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bit 7 – SMT1PWAIF** SMT1 Pulse-Width Acquisition Interrupt Flag

| Value | Description |
|---|---|
| 1 | Interrupt has occurred (must be cleared by software) |
| 0 | Interrupt event has not occurred |

**Bit 6 – SMT1PRAIF** SMT1 Period Acquisition Interrupt Flag

| Value | Description |
|---|---|
| 1 | Interrupt has occurred (must be cleared by software) |
| 0 | Interrupt event has not occurred |

**Bit 5 – SMT1IF** SMT1 Interrupt Flag

| Value | Description |
|---|---|
| 1 | Interrupt has occurred (must be cleared by software) |
| 0 | Interrupt event has not occurred |

**Bit 4 – CM1IF** CMP1 Interrupt Flag

| Value | Description |
|---|---|
| 1 | Interrupt has occurred (must be cleared by software) |
| 0 | Interrupt event has not occurred |

**Bit 3 – ACTIF** Active Clock Tuning Interrupt Flag

| Value | Description |
|---|---|
| 1 | Interrupt has occurred (must be cleared by software) |
| 0 | Interrupt event has not occurred |

**Bit 2 – ADIF** ADC Interrupt Flag

| Value | Description |
|---|---|
| 1 | Interrupt has occurred (must be cleared by software) |
| 0 | Interrupt event has not occurred |

**Bit 1 – ZCDIF** ZCD Interrupt Flag

| Value | Description |
|---|---|
| 1 | Interrupt has occurred (must be cleared by software) |
| 0 | Interrupt event has not occurred |

**Bit 0 – INT0IF** External Interrupt 0 Interrupt Flag(2)

| Value | Description |
|---|---|
| 1 | Interrupt has occurred (must be cleared by software) |
| 0 | Interrupt event has not occurred |

**Notes:**

1. Interrupt flag bits get set when an interrupt condition occurs, regardless of the state of its corresponding enable bit, or the global enable bit. User software will ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt.
2. The external interrupt GPIO pin is selected by the INTxPPS register.

### 11.13.25 PIR2

**Name:** PIR2
**Address:** 0x4B0

Peripheral Interrupt Request Register 2

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | DMA1AIF | DMA1ORIF | DMA1DCNTIF | DMA1SCNTIF | | | | ADTIF |
| Access | R/W/HS | R/W/HS | R/W/HS | R/W/HS | | | | R/W/HS |
| Reset | 0 | 0 | 0 | 0 | | | | 0 |

**Bit 7 – DMA1AIF** DMA1 Abort Interrupt Flag

| Value | Description |
|---|---|
| 1 | Interrupt has occurred (must be cleared by software) |
| 0 | Interrupt event has not occurred |

**Bit 6 – DMA1ORIF** DMA1 Overrun Interrupt Flag

| Value | Description |
|---|---|
| 1 | Interrupt has occurred |
| 0 | Interrupt event has not occurred |

**Bit 5 – DMA1DCNTIF** DMA1 Destination Count Interrupt Flag

| Value | Description |
|---|---|
| 1 | Interrupt has occurred |
| 0 | Interrupt event has not occurred |

**Bit 4 – DMA1SCNTIF** DMA1 Source Count Interrupt Flag

| Value | Description |
|---|---|
| 1 | Interrupt has occurred |
| 0 | Interrupt event has not occurred |

**Bit 0 – ADTIF** ADC Threshold Interrupt Flag

| Value | Description |
|---|---|
| 1 | Interrupt has occurred |
| 0 | Interrupt event has not occurred |

**Note:**

1. Interrupt flag bits get set when an interrupt condition occurs, regardless of the state of its corresponding enable bit, or the global enable bit. User software will ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt.

### 11.13.26 PIR3

**Name:** PIR3
**Address:** 0x4B1

Peripheral Interrupt Request Register 3

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | TMR0IF | CCP1IF | TMR1GIF | TMR1IF | TMR2IF | SPI1IF | SPI1TXIF | SPI1RXIF |
| Access | R/W/HS | R/W/HS | R/W/HS | R/W/HS | R/W/HS | R | R | R |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bit 7 – TMR0IF** TMR0 Interrupt Flag

| Value | Description |
|---|---|
| 1 | Interrupt has occurred (must be cleared by software) |
| 0 | Interrupt event has not occurred |

**Bit 6 – CCP1IF** CCP1 Interrupt Flag

| Value | Description |
|---|---|
| 1 | Interrupt has occurred (must be cleared by software) |
| 0 | Interrupt event has not occurred |

**Bit 5 – TMR1GIF** TMR1 Gate Interrupt Flag

| Value | Description |
|---|---|
| 1 | Interrupt has occurred (must be cleared by software) |
| 0 | Interrupt event has not occurred |

**Bit 4 – TMR1IF** TMR1 Interrupt Flag

| Value | Description |
|---|---|
| 1 | Interrupt has occurred (must be cleared by software) |
| 0 | Interrupt event has not occurred |

**Bit 3 – TMR2IF** TMR2 Interrupt Flag

| Value | Description |
|---|---|
| 1 | Interrupt has occurred (must be cleared by software) |
| 0 | Interrupt event has not occurred |

**Bit 2 – SPI1IF** SPI1 Interrupt Flag(2)

| Value | Description |
|---|---|
| 1 | Interrupt has occurred |
| 0 | Interrupt event has not occurred |

**Bit 1 – SPI1TXIF** SPI1 Transmit Interrupt Flag(3)

| Value | Description |
|---|---|
| 1 | Interrupt has occurred |
| 0 | Interrupt event has not occurred |

**Bit 0 – SPI1RXIF** SPI1 Receive Interrupt Flag(3)

| Value | Description |
|---|---|
| 1 | Interrupt has occurred |
| 0 | Interrupt event has not occurred |

© 2019-2023 Microchip Technology Inc. and its subsidiaries

**Notes:**

1. Interrupt flag bits get set when an interrupt condition occurs, regardless of the state of its corresponding enable bit, or the global enable bit. User software will ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt.
2. SPI1IF is a read-only bit. TO clear the interrupt condition, all bits in the SPI1INTF register must be cleared.
3. SPI1TXIF and SPI1RXIF are read-only bits and cannot be set/cleared by software.

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

### 11.13.27 PIR4

**Name:** PIR4
**Address:** 0x4B2

Peripheral Interrupt Request Register 4

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | PWM1IF | PWM1PIF | | | U1IF | U1EIF | U1TXIF | U1RXIF |
| Access | R | R/W/HS | | | R | R | R | R |
| Reset | 0 | 0 | | | 0 | 0 | 0 | 0 |

**Bit 7 – PWM1IF** PWM1 Parameter Interrupt Flag[2]

| Value | Description |
|---|---|
| 1 | Interrupt has occurred |
| 0 | Interrupt event has not occurred |

**Bit 6 – PWM1PIF** PWM1 Period Interrupt Flag

| Value | Description |
|---|---|
| 1 | Interrupt has occurred (must be cleared by software) |
| 0 | Interrupt event has not occurred |

**Bit 3 – U1IF** UART1 Interrupt Flag[3]

| Value | Description |
|---|---|
| 1 | Interrupt has occurred |
| 0 | Interrupt event has not occurred |

**Bit 2 – U1EIF** UART1 Framing Error Interrupt Flag[4]

| Value | Description |
|---|---|
| 1 | Interrupt has occurred |
| 0 | Interrupt event has not occurred |

**Bit 1 – U1TXIF** UART1 Transmit Interrupt Flag[5]

| Value | Description |
|---|---|
| 1 | Interrupt has occurred |
| 0 | Interrupt event has not occurred |

**Bit 0 – U1RXIF** UART 1 Receive Interrupt Flag[5]

| Value | Description |
|---|---|
| 1 | Interrupt has occurred |
| 0 | Interrupt event has not occurred |

**Notes:**

1. Interrupt flag bits get set when an interrupt condition occurs, regardless of the state of its corresponding enable bit, or the global enable bit. User software will ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt.
2. PWM1IF is a read-only bit. To clear the interrupt condition, all bits in the PWM1GIR register must be cleared.
3. U1IF is a read-only bit. To clear the interrupt condition, all bits in the U1UIR register must be cleared.
4. U1EIF is a read-only bit. To clear the interrupt condition, all bits in the U1ERR register must be cleared.
5. U1TXIF and U1RXIF are read-only bits and cannot be set/cleared by software.

### 11.13.28 PIR5

**Name:** PIR5
**Address:** 0x4B3

Peripheral Interrupt Request Register 5

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | PWM2IF | PWM2PIF | TMR3GIF | TMR3IF | | SPI2IF | SPI2TXIF | SPI2RXIF |
| Access | R | R/W/HS | R/W/HS | R/W/HS | | R | R | R |
| Reset | 0 | 0 | 0 | 0 | | 0 | 0 | 0 |

**Bit 7 – PWM2IF** PWM2 Parameter Interrupt Flag(2)

| Value | Description |
|---|---|
| 1 | Interrupt has occurred |
| 0 | Interrupt event has not occurred |

**Bit 6 – PWM2PIF** PWM2 Period Interrupt Flag

| Value | Description |
|---|---|
| 1 | Interrupt has occurred (must be cleared by software) |
| 0 | Interrupt event has not occurred |

**Bit 5 – TMR3GIF** TMR3 Gate Interrupt Flag

| Value | Description |
|---|---|
| 1 | Interrupt has occurred (must be cleared by software) |
| 0 | Interrupt event has not occurred |

**Bit 4 – TMR3IF** TMR3 Interrupt Flag

| Value | Description |
|---|---|
| 1 | Interrupt has occurred (must be cleared by software) |
| 0 | Interrupt event has not occurred |

**Bit 2 – SPI2IF** SPI2 Interrupt Flag(3)

| Value | Description |
|---|---|
| 1 | Interrupt has occurred |
| 0 | Interrupt event has not occurred |

**Bit 1 – SPI2TXIF** SPI2 Transmit Interrupt Flag(4)

| Value | Description |
|---|---|
| 1 | Interrupt has occurred |
| 0 | Interrupt event has not occurred |

**Bit 0 – SPI2RXIF** SPI2 Receive Interrupt Flag(4)

| Value | Description |
|---|---|
| 1 | Interrupt has occurred |
| 0 | Interrupt event has not occurred |

**Notes:**

1. Interrupt flag bits get set when an interrupt condition occurs, regardless of the state of its corresponding enable bit, or the global enable bit. User software will ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt.
2. PWM2IF is a read-only bit. To clear the interrupt condition, all bits in the PWM2GIR register must be cleared.
3. SPI2IF is a read-only bit. TO clear the interrupt condition, all bits in the SPI2INTF register must be cleared.
4. SPI2TXIF and SPI2RXIF are read-only bits and cannot be set/cleared by software.

© 2019-2023 Microchip Technology Inc. and its subsidiaries

### 11.13.29 PIR6

**Name:** PIR6
**Address:** 0x4B4

Peripheral Interrupt Request Register 6

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | DMA2AIF | DMA2ORIF | DMA2DCNTIF | DMA2SCNTIF | NCO1IF | CWG1IF | CLC2IF | INT1IF |
| Access | R/W/HS | R/W/HS | R/W/HS | R/W/HS | R/W/HS | R/W/HS | R/W/HS | R/W/HS |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bit 7 – DMA2AIF** DMA2 Abort Interrupt Flag

| Value | Description |
|---|---|
| 1 | Interrupt has occurred (must be cleared by software) |
| 0 | Interrupt event has not occurred |

**Bit 6 – DMA2ORIF** DMA2 Overrun Interrupt Flag

| Value | Description |
|---|---|
| 1 | Interrupt has occurred (must be cleared by software) |
| 0 | Interrupt event has not occurred |

**Bit 5 – DMA2DCNTIF** DMA2 Destination Count Interrupt Flag

| Value | Description |
|---|---|
| 1 | Interrupt has occurred (must be cleared by software) |
| 0 | Interrupt event has not occurred |

**Bit 4 – DMA2SCNTIF** DMA2 Source Count Interrupt Flag

| Value | Description |
|---|---|
| 1 | Interrupt has occurred (must be cleared by software) |
| 0 | Interrupt event has not occurred |

**Bit 3 – NCO1IF** NCO1 Interrupt Flag

| Value | Description |
|---|---|
| 1 | Interrupt has occurred (must be cleared by software) |
| 0 | Interrupt event has not occurred |

**Bit 2 – CWG1IF** CWG1 Interrupt Flag

| Value | Description |
|---|---|
| 1 | Interrupt has occurred (must be cleared by software) |
| 0 | Interrupt event has not occurred |

**Bit 1 – CLC2IF** CLC2 Interrupt Flag

| Value | Description |
|---|---|
| 1 | Interrupt has occurred (must be cleared by software) |
| 0 | Interrupt event has not occurred |

**Bit 0 – INT1IF** External Interrupt 1 Interrupt Flag

| Value | Description |
|---|---|
| 1 | Interrupt has occurred (must be cleared by software) |
| 0 | Interrupt event has not occurred |

**Note:**

1. Interrupt flag bits get set when an interrupt condition occurs, regardless of the state of its corresponding enable bit, or the global enable bit. User software will ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt.

### 11.13.30 PIR7

**Name:** PIR7
**Address:** 0x4B5

Peripheral Interrupt Request Register 7

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | PWM3IF | PWM3PIF | CLC3IF | | I2C1EIF | I2C1IF | I2C1TXIF | I2C1RXIF |
| Access | R | R/W/HS | R/W/HS | | R | R | R | R |
| Reset | 0 | 0 | 0 | | 0 | 0 | 0 | 0 |

**Bit 7 – PWM3IF** PWM3 Parameter Interrupt Flag(2)

| Value | Description |
|---|---|
| 1 | Interrupt has occurred |
| 0 | Interrupt event has not occurred |

**Bit 6 – PWM3PIF** PWM3 Period Interrupt Flag

| Value | Description |
|---|---|
| 1 | Interrupt has occurred (must be cleared by software) |
| 0 | Interrupt event has not occurred |

**Bit 5 – CLC3IF** CLC3 Interrupt Flag

| Value | Description |
|---|---|
| 1 | Interrupt has occurred (must be cleared by software) |
| 0 | Interrupt event has not occurred |

**Bit 3 – I2C1EIF** I2C1 Error Interrupt Flag(3)

| Value | Description |
|---|---|
| 1 | Interrupt has occurred |
| 0 | Interrupt event has not occurred |

**Bit 2 – I2C1IF** I2C1 Interrupt Flag(4)

| Value | Description |
|---|---|
| 1 | Interrupt has occurred |
| 0 | Interrupt event has not occurred |

**Bit 1 – I2C1TXIF** I2C1 Transmit Interrupt Flag(5)

| Value | Description |
|---|---|
| 1 | Interrupt has occurred |
| 0 | Interrupt event has not occurred |

**Bit 0 – I2C1RXIF** I2C1 Receive Interrupt Flag(5)

| Value | Description |
|---|---|
| 1 | Interrupt has occurred |
| 0 | Interrupt event has not occurred |

**Notes:**

1. Interrupt flag bits get set when an interrupt condition occurs, regardless of the state of its corresponding enable bit, or the global enable bit. User software will ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt.
2. PWM3IF is a read-only bit. To clear the interrupt condition, all bits in the PWM3GIR register must be cleared.
3. I2C1EIF is a read-only bit. To clear the interrupt condition, all bits in the I2C1ERR register must be cleared.
4. I2C1IF is a read-only bit. To clear the interrupt condition, all bits in the I2C1PIR register must be cleared.
5. I2C1TXIF and I2C1RXIF are read-only bits. To clear the interrupt condition, the CLRBF bit in I2C1STAT1 must be set.

© 2019-2023 Microchip Technology Inc. and its subsidiaries

### 11.13.31 PIR8

**Name:** PIR8
**Address:** 0x4B6

Peripheral Interrupt Request Register 8

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | SCANIF | CCP2IF | TMR5GIF | TMR5IF | U2IF | U2EIF | U2TXIF | U2RXIF |
| Access | R/W/HS | R/W/HS | R/W/HS | R/W/HS | R | R | R | R |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bit 7 – SCANIF** Memory Scanner Interrupt Flag

| Value | Description |
|---|---|
| 1 | Interrupt has occurred (must be cleared by software) |
| 0 | Interrupt event has not occurred |

**Bit 6 – CCP2IF** CCP2 Interrupt Flag

| Value | Description |
|---|---|
| 1 | Interrupt has occurred (must be cleared by software) |
| 0 | Interrupt event has not occurred |

**Bit 5 – TMR5GIF** TMR5 Gate Interrupt Flag

| Value | Description |
|---|---|
| 1 | Interrupt has occurred (must be cleared by software) |
| 0 | Interrupt event has not occurred |

**Bit 4 – TMR5IF** TMR5 Interrupt Flag

| Value | Description |
|---|---|
| 1 | Interrupt has occurred (must be cleared by software) |
| 0 | Interrupt event has not occurred |

**Bit 3 – U2IF** UART2 Interrupt Flag(2)

| Value | Description |
|---|---|
| 1 | Interrupt has occurred |
| 0 | Interrupt event has not occurred |

**Bit 2 – U2EIF** UART2 Framing Error Interrupt Flag(3)

| Value | Description |
|---|---|
| 1 | Interrupt has occurred |
| 0 | Interrupt event has not occurred |

**Bit 1 – U2TXIF** UART2 Transmit Interrupt Flag(4)

| Value | Description |
|---|---|
| 1 | Interrupt has occurred |
| 0 | Interrupt event has not occurred |

**Bit 0 – U2RXIF** UART2 Receive Interrupt Flag(4)

| Value | Description |
|---|---|
| 1 | Interrupt has occurred |
| 0 | Interrupt event has not occurred |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

**Notes:**

1. Interrupt flag bits get set when an interrupt condition occurs, regardless of the state of its corresponding enable bit, or the global enable bit. User software will ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt.
2. U2IF is a read-only bit. To clear the interrupt condition, all bits in the U2UIR register must be cleared.
3. U2EIF is a read-only bit. To clear the interrupt condition, all bits in the U2ERR register must be cleared.
4. U2TXIF and U2RXIF are read-only bits and cannot be set/cleared by software.

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

### 11.13.32 PIR9

**Name:** PIR9
**Address:** 0x4B7

Peripheral Interrupt Request Register 9

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | CLC4IF | | U3IF | U3EIF | U3TXIF | U3RXIF |
| Access | | | R/W/HS | | R | R | R | R |
| Reset | | | 0 | | 0 | 0 | 0 | 0 |

**Bit 5 – CLC4IF** CLC4 Interrupt Flag

| Value | Description |
|---|---|
| 1 | Interrupt has occurred (must be cleared by software) |
| 0 | Interrupt event has not occurred |

**Bit 3 – U3IF** UART3 Interrupt Flag(2)

| Value | Description |
|---|---|
| 1 | Interrupt has occurred |
| 0 | Interrupt event has not occurred |

**Bit 2 – U3EIF** UART3 Framing Error Interrupt Flag(3)

| Value | Description |
|---|---|
| 1 | Interrupt has occurred |
| 0 | Interrupt event has not occurred |

**Bit 1 – U3TXIF** UART3 Transmit Interrupt Flag(4)

| Value | Description |
|---|---|
| 1 | Interrupt has occurred |
| 0 | Interrupt event has not occurred |

**Bit 0 – U3RXIF** UART3 Receive Interrupt Flag(4)

| Value | Description |
|---|---|
| 1 | Interrupt has occurred |
| 0 | Interrupt event has not occurred |

**Notes:**

1. Interrupt flag bits get set when an interrupt condition occurs, regardless of the state of its corresponding enable bit, or the global enable bit. User software will ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt.
2. U3IF is a read-only bit. To clear the interrupt condition, all bits in the U3UIR register must be cleared.
3. U3EIF is a read-only bit. To clear the interrupt condition, all bits in the U3ERR register must be cleared.
4. U3TXIF and U3RXIF are read-only bits and cannot be set/cleared by software.

### 11.13.33 PIR10

**Name:** PIR10
**Address:** 0x4B8

Peripheral Interrupt Request Register 10

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | DMA3AIF | DMA3ORIF | DMA3DCNTIF | DMA3SCNTIF | NCO2IF | CWG2IF | CLC5IF | INT2IF |
| Access | R/W/HS | R/W/HS | R/W/HS | R/W/HS | R/W/HS | R/W/HS | R/W/HS | R/W/HS |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bit 7 – DMA3AIF** DMA3 Abort Interrupt Flag

| Value | Description |
|---|---|
| 1 | Interrupt has occurred (must be cleared by software) |
| 0 | Interrupt event has not occurred |

**Bit 6 – DMA3ORIF** DMA3 Overrun Interrupt Flag

| Value | Description |
|---|---|
| 1 | Interrupt has occurred (must be cleared by software) |
| 0 | Interrupt event has not occurred |

**Bit 5 – DMA3DCNTIF** DMA3 Destination Count Interrupt Flag

| Value | Description |
|---|---|
| 1 | Interrupt has occurred (must be cleared by software) |
| 0 | Interrupt event has not occurred |

**Bit 4 – DMA3SCNTIF** DMA3 Source Count Interrupt Flag

| Value | Description |
|---|---|
| 1 | Interrupt has occurred (must be cleared by software) |
| 0 | Interrupt event has not occurred |

**Bit 3 – NCO2IF** NCO2 Interrupt Flag

| Value | Description |
|---|---|
| 1 | Interrupt has occurred (must be cleared by software) |
| 0 | Interrupt event has not occurred |

**Bit 2 – CWG2IF** CWG2 Interrupt Flag

| Value | Description |
|---|---|
| 1 | Interrupt has occurred (must be cleared by software) |
| 0 | Interrupt event has not occurred |

**Bit 1 – CLC5IF** CLC5 Interrupt Flag

| Value | Description |
|---|---|
| 1 | Interrupt has occurred (must be cleared by software) |
| 0 | Interrupt event has not occurred |

**Bit 0 – INT2IF** External Interrupt 2 Interrupt Flag

| Value | Description |
|---|---|
| 1 | Interrupt has occurred (must be cleared by software) |
| 0 | Interrupt event has not occurred |

**Note:**

1. Interrupt flag bits get set when an interrupt condition occurs, regardless of the state of its corresponding enable bit, or the global enable bit. User software will ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt.

### 11.13.34 PIR11

**Name:** PIR11
**Address:** 0x4B9

Peripheral Interrupt Request Register 11

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | DMA4AIF | DMA4ORIF | DMA4DCNTIF | DMA4SCNTIF | TMR4IF | CWG3IF | CLC6IF | CCP3IF |
| Access | R/W/HS | R/W/HS | R/W/HS | R/W/HS | R/W/HS | R/W/HS | R/W/HS | R/W/HS |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bit 7 – DMA4AIF** DMA4 Abort Interrupt Flag

| Value | Description |
|---|---|
| 1 | Interrupt has occurred (must be cleared by software) |
| 0 | Interrupt event has not occurred |

**Bit 6 – DMA4ORIF** DMA4 Overrun Interrupt Flag

| Value | Description |
|---|---|
| 1 | Interrupt has occurred (must be cleared by software) |
| 0 | Interrupt event has not occurred |

**Bit 5 – DMA4DCNTIF** DMA4 Destination Count Interrupt Flag

| Value | Description |
|---|---|
| 1 | Interrupt has occurred (must be cleared by software) |
| 0 | Interrupt event has not occurred |

**Bit 4 – DMA4SCNTIF** DMA4 Source Count Interrupt Flag

| Value | Description |
|---|---|
| 1 | Interrupt has occurred (must be cleared by software) |
| 0 | Interrupt event has not occurred |

**Bit 3 – TMR4IF** TMR4 Interrupt Flag

| Value | Description |
|---|---|
| 1 | Interrupt has occurred (must be cleared by software) |
| 0 | Interrupt event has not occurred |

**Bit 2 – CWG3IF** CWG3 Interrupt Flag

| Value | Description |
|---|---|
| 1 | Interrupt has occurred (must be cleared by software) |
| 0 | Interrupt event has not occurred |

**Bit 1 – CLC6IF** CLC6 Interrupt Flag

| Value | Description |
|---|---|
| 1 | Interrupt has occurred (must be cleared by software) |
| 0 | Interrupt event has not occurred |

**Bit 0 – CCP3IF** CCP3 Interrupt Flag

| Value | Description |
|---|---|
| 1 | Interrupt has occurred (must be cleared by software) |
| 0 | Interrupt event has not occurred |

**Note:**

1. Interrupt flag bits get set when an interrupt condition occurs, regardless of the state of its corresponding enable bit, or the global enable bit. User software will ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt.

© 2019-2023 Microchip Technology Inc. and its subsidiaries

### 11.13.35 PIR12

**Name:** PIR12
**Address:** 0x4BA

Peripheral Interrupt Request Register 12

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | DMA5AIF | DMA5ORIF | DMA5DCNTIF | DMA5SCNTIF | U4IF | U4EIF | U4TXIF | U4RXIF |
| Access | R/W/HS | R/W/HS | R/W/HS | R/W/HS | R | R | R | R |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bit 7 – DMA5AIF** DMA5 Abort Interrupt Flag

| Value | Description |
|---|---|
| 1 | Interrupt has occurred (must be cleared by software) |
| 0 | Interrupt event has not occurred |

**Bit 6 – DMA5ORIF** DMA5 Overrun Interrupt Flag

| Value | Description |
|---|---|
| 1 | Interrupt has occurred (must be cleared by software) |
| 0 | Interrupt event has not occurred |

**Bit 5 – DMA5DCNTIF** DMA5 Destination Count Interrupt Flag

| Value | Description |
|---|---|
| 1 | Interrupt has occurred (must be cleared by software) |
| 0 | Interrupt event has not occurred |

**Bit 4 – DMA5SCNTIF** DMA5 Source Count Interrupt Flag

| Value | Description |
|---|---|
| 1 | Interrupt has occurred (must be cleared by software) |
| 0 | Interrupt event has not occurred |

**Bit 3 – U4IF** UART 4 Interrupt Flag(2)

| Value | Description |
|---|---|
| 1 | Interrupt has occurred |
| 0 | Interrupt event has not occurred |

**Bit 2 – U4EIF** UART4 Framing Error Interrupt Flag(3)

| Value | Description |
|---|---|
| 1 | Interrupt has occurred |
| 0 | Interrupt event has not occurred |

**Bit 1 – U4TXIF** UART4 Transmit Interrupt Flag(4)

| Value | Description |
|---|---|
| 1 | Interrupt has occurred |
| 0 | Interrupt event has not occurred |

**Bit 0 – U4RXIF** UART4 Receive Interrupt Flag(4)

| Value | Description |
|---|---|
| 1 | Interrupt has occurred |
| 0 | Interrupt event has not occurred |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

**Notes:**

1. Interrupt flag bits get set when an interrupt condition occurs, regardless of the state of its corresponding enable bit, or the global enable bit. User software will ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt.
2. U4IF is a read-only bit. To clear the interrupt condition, all bits in the U4UIR register must be cleared.
3. U4EIF is a read-only bit. To clear the interrupt condition, all bits in the U4ERR register must be cleared.
4. U4TXIF and U4RXIF are read-only bits and cannot be set/cleared by software.

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

### 11.13.36 PIR13

**Name:** PIR13
**Address:** 0x4BB

Peripheral Interrupt Request Register 13

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | DMA6AIF | DMA6ORIF | DMA6DCNTIF | DMA6SCNTIF | U5IF | U5EIF | U5TXIF | U5RXIF |
| Access | R/W/HS | R/W/HS | R/W/HS | R/W/HS | R | R | R | R |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bit 7 – DMA6AIF** DMA6 Abort Interrupt Flag

| Value | Description |
|---|---|
| 1 | Interrupt has occurred (must be cleared by software) |
| 0 | Interrupt event has not occurred |

**Bit 6 – DMA6ORIF** DMA6 Overrun Interrupt Flag

| Value | Description |
|---|---|
| 1 | Interrupt has occurred (must be cleared by software) |
| 0 | Interrupt event has not occurred |

**Bit 5 – DMA6DCNTIF** DMA6 Destination Count Interrupt Flag

| Value | Description |
|---|---|
| 1 | Interrupt has occurred (must be cleared by software) |
| 0 | Interrupt event has not occurred |

**Bit 4 – DMA6SCNTIF** DMA6 Source Count Interrupt Flag

| Value | Description |
|---|---|
| 1 | Interrupt has occurred (must be cleared by software) |
| 0 | Interrupt event has not occurred |

**Bit 3 – U5IF** UART5 Interrupt Flag(2)

| Value | Description |
|---|---|
| 1 | Interrupt has occurred |
| 0 | Interrupt event has not occurred |

**Bit 2 – U5EIF** UART5 Framing Error Interrupt Flag(3)

| Value | Description |
|---|---|
| 1 | Interrupt has occurred |
| 0 | Interrupt event has not occurred |

**Bit 1 – U5TXIF** UART5 Transmit Interrupt Flag(4)

| Value | Description |
|---|---|
| 1 | Interrupt has occurred |
| 0 | Interrupt event has not occurred |

**Bit 0 – U5RXIF** UART5 Receive Interrupt Flag(4)

| Value | Description |
|---|---|
| 1 | Interrupt has occurred |
| 0 | Interrupt event has not occurred |

© 2019-2023 Microchip Technology Inc. and its subsidiaries

**Notes:**

1. Interrupt flag bits get set when an interrupt condition occurs, regardless of the state of its corresponding enable bit, or the global enable bit. User software will ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt.
2. U5IF is a read-only bit. To clear the interrupt condition, all bits in the U5UIR register must be cleared.
3. U5EIF is a read-only bit. To clear the interrupt condition, all bits in the U5ERR register must be cleared.
4. U5TXIF and U5RXIF are read-only bits and cannot be set/cleared by software.

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

### 11.13.37 PIR14

**Name:** PIR14
**Address:** 0x4BC

Peripheral Interrupt Request Register 14

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | NCO3IF | CM2IF | CLC7IF | |
| Access | | | | | R/W/HS | R/W/HS | R/W/HS | |
| Reset | | | | | 0 | 0 | 0 | |

**Bit 3 – NCO3IF** NCO3 Interrupt Flag

| Value | Description |
|---|---|
| 1 | Interrupt has occurred (must be cleared by software) |
| 0 | Interrupt event has not occurred |

**Bit 2 – CM2IF** CMP2 Interrupt Flag

| Value | Description |
|---|---|
| 1 | Interrupt has occurred (must be cleared by software) |
| 0 | Interrupt event has not occurred |

**Bit 1 – CLC7IF** CLC7 Interrupt Flag

| Value | Description |
|---|---|
| 1 | Interrupt has occurred (must be cleared by software) |
| 0 | Interrupt event has not occurred |

**Note:**

1. Interrupt flag bits get set when an interrupt condition occurs, regardless of the state of its corresponding enable bit, or the global enable bit. User software will ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt.

© 2019-2023 Microchip Technology Inc. and its subsidiaries

### 11.13.38 PIR15

**Name:** PIR15
**Address:** 0x4BD

Peripheral Interrupt Request Register 15

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | TMR6IF | CRCIF | CLC8IF | NVMIF |
| Access | | | | | R/W/HS | R/W/HS | R/W/HS | R/W/HS |
| Reset | | | | | 0 | 0 | 0 | 0 |

**Bit 3 – TMR6IF** TMR6 Interrupt Flag

| Value | Description |
|---|---|
| 1 | Interrupt has occurred (must be cleared by software) |
| 0 | Interrupt event has not occurred |

**Bit 2 – CRCIF** CRC Interrupt Flag

| Value | Description |
|---|---|
| 1 | Interrupt has occurred (must be cleared by software) |
| 0 | Interrupt event has not occurred |

**Bit 1 – CLC8IF** CLC8 Interrupt Flag

| Value | Description |
|---|---|
| 1 | Interrupt has occurred (must be cleared by software) |
| 0 | Interrupt event has not occurred |

**Bit 0 – NVMIF** NVM Interrupt Flag

| Value | Description |
|---|---|
| 1 | Interrupt has occurred (must be cleared by software) |
| 0 | Interrupt event has not occurred |

**Note:**

1. Interrupt flag bits get set when an interrupt condition occurs, regardless of the state of its corresponding enable bit, or the global enable bit. User software will ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt.

© 2019-2023 Microchip Technology Inc. and its subsidiaries

### 11.13.39 IPR0

**Name:** IPR0
**Address:** 0x362

Peripheral Interrupt Request Register 0

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | IOCIP | | CLC1IP | | CSWIP | OSFIP | HLVDIP | SWIP |
| Access | R/W | | R/W | | R/W | R/W | R/W | R/W |
| Reset | 1 | | 1 | | 1 | 1 | 1 | 1 |

**Bit 7 – IOCIP** Interrupt-on-Change Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 5 – CLC1IP** CLC1 Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 3 – CSWIP** Clock Switch Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 2 – OSFIP** Oscillator Failure Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 1 – HLVDIP** High/Low-Voltage Detect Priority Flag

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 0 – SWIP** Software Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

### 11.13.40 IPR1

**Name:** IPR1
**Address:** 0x363

Peripheral Interrupt Priority Register 1

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | SMT1PWAIP | SMT1PRAIP | SMT1IP | CM1IP | ACTIP | ADIP | ZCDIP | INT0IP |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Bit 7 – SMT1PWAIP** SMT1 Pulse-Width Acquisition Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 6 – SMT1PRAIP** SMT1 Period Acquisition Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 5 – SMT1IP** SMT1 Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 4 – CM1IP** CMP1 Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 3 – ACTIP** Active Clock Tuning Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 2 – ADIP** ADC Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 1 – ZCDIP** ZCD Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 0 – INT0IP** External Interrupt 0 Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

### 11.13.41 IPR2

**Name:** IPR2
**Address:** 0x364

Peripheral Interrupt Priority Register 2

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | DMA1AIP | DMA1ORIP | DMA1DCNTIP | DMA1SCNTIP | | | | ADTIP |
| Access | R/W | R/W | R/W | R/W | | | | R/W |
| Reset | 1 | 1 | 1 | 1 | | | | 1 |

**Bit 7 – DMA1AIP** DMA1 Abort Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 6 – DMA1ORIP** DMA1 Overrun Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 5 – DMA1DCNTIP** DMA1 Destination Count Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 4 – DMA1SCNTIP** DMA1 Source Count Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 0 – ADTIP** ADC Threshold Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

### 11.13.42 IPR3

**Name:** IPR3
**Address:** 0x365

Peripheral Interrupt Priority Register 3

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | TMR0IP | CCP1IP | TMR1GIP | TMR1IP | TMR2IP | SPI1IP | SPI1TXIP | SPI1RXIP |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Bit 7 – TMR0IP** TMR0 Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 6 – CCP1IP** CCP1 Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 5 – TMR1GIP** TMR1 Gate Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 4 – TMR1IP** TMR1 Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 3 – TMR2IP** TMR2 Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 2 – SPI1IP** SPI1 Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 1 – SPI1TXIP** SPI1 Transmit Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 0 – SPI1RXIP** SPI1 Receive Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

### 11.13.43 IPR4

**Name:** IPR4
**Address:** 0x366

Peripheral Interrupt Priority Register 4

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | PWM1IP | PWM1PIP | | | U1IP | U1EIP | U1TXIP | U1RXIP |
| Access | R/W | R/W | | | R/W | R/W | R/W | R/W |
| Reset | 1 | 1 | | | 1 | 1 | 1 | 1 |

**Bit 7 – PWM1IP** PWM1 Parameter Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 6 – PWM1PIP** PWM1 Period Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 3 – U1IP** UART1 Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 2 – U1EIP** UART1 Framing Error Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 1 – U1TXIP** UART1 Transmit Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 0 – U1RXIP** UART 1 Receive Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

### 11.13.44 IPR5

**Name:** IPR5
**Address:** 0x367

Peripheral Interrupt Priority Register 5

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | PWM2IP | PWM2PIP | TMR3GIP | TMR3IP | | SPI2IP | SPI2TXIP | SPI2RXIP |
| Access | R/W | R/W | R/W | R/W | | R/W | R/W | R/W |
| Reset | 1 | 1 | 1 | 1 | | 1 | 1 | 1 |

**Bit 7 – PWM2IP** PWM2 Parameter Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 6 – PWM2PIP** PWM2 Period Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 5 – TMR3GIP** TMR3 Gate Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 4 – TMR3IP** TMR3 Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 2 – SPI2IP** SPI2 Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 1 – SPI2TXIP** SPI2 Transmit Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 0 – SPI2RXIP** SPI2 Receive Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

### 11.13.45 IPR6

**Name:** IPR6
**Address:** 0x368

Peripheral Interrupt Priority Register 6

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | DMA2AIP | DMA2ORIP | DMA2DCNTIP | DMA2SCNTIP | NCO1IP | CWG1IP | CLC2IP | INT1IP |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Bit 7 – DMA2AIP** DMA2 Abort Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 6 – DMA2ORIP** DMA2 Overrun Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 5 – DMA2DCNTIP** DMA2 Destination Count Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 4 – DMA2SCNTIP** DMA2 Source Count Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 3 – NCO1IP** NCO1 Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 2 – CWG1IP** CWG1 Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 1 – CLC2IP** CLC2 Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 0 – INT1IP** External Interrupt 1 Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

### 11.13.46 IPR7

**Name:** IPR7
**Address:** 0x369

Peripheral Interrupt Priority Register 7

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | PWM3IP | PWM3PIP | CLC3IP | | I2C1EIP | I2C1IP | I2C1TXIP | I2C1RXIP |
| Access | R/W | R/W | R/W | | R/W | R/W | R/W | R/W |
| Reset | 1 | 1 | 1 | | 1 | 1 | 1 | 1 |

**Bit 7 – PWM3IP** PWM3 Parameter Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 6 – PWM3PIP** PWM3 Period Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 5 – CLC3IP** CLC3 Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 3 – I2C1EIP** I2C1 Error Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 2 – I2C1IP** I2C1 Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 1 – I2C1TXIP** I2C1 Transmit Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 0 – I2C1RXIP** I2C1 Receive Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

### 11.13.47 IPR8

**Name:** IPR8
**Address:** 0x36A

Peripheral Interrupt Priority Register 8

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | SCANIP | CCP2IP | TMR5GIP | TMR5IP | U2IP | U2EIP | U2TXIP | U2RXIP |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Bit 7 – SCANIP** Memory Scanner Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 6 – CCP2IP** CCP2 Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 5 – TMR5GIP** TMR5 Gate Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 4 – TMR5IP** TMR5 Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 3 – U2IP** UART2 Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 2 – U2EIP** UART2 Framing Error Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 1 – U2TXIP** UART2 Transmit Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 0 – U2RXIP** UART2 Receive Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

### 11.13.48 IPR9

**Name:** IPR9
**Address:** 0x36B

Peripheral Interrupt Priority Register 9

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | CLC4IP | | U3IP | U3EIP | U3TXIP | U3RXIP |
| Access | | | R/W | | R/W | R/W | R/W | R/W |
| Reset | | | 1 | | 1 | 1 | 1 | 1 |

**Bit 5 – CLC4IP** CLC4 Priority Flag

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 3 – U3IP** UART3 Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 2 – U3EIP** UART3 Framing Error Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 1 – U3TXIP** UART3 Transmit Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 0 – U3RXIP** UART3 Receive Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

### 11.13.49 IPR10

**Name:** IPR10
**Address:** 0x36C

Peripheral Interrupt Priority Register 10

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | DMA3AIP | DMA3ORIP | DMA3DCNTIP | DMA3SCNTIP | NCO2IP | CWG2IP | CLC5IP | INT2IP |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Bit 7 – DMA3AIP** DMA3 Abort Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 6 – DMA3ORIP** DMA3 Overrun Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 5 – DMA3DCNTIP** DMA3 Destination Count Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 4 – DMA3SCNTIP** DMA3 Source Count Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 3 – NCO2IP** NCO2 Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 2 – CWG2IP** CWG2 Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 1 – CLC5IP** CLC5 Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 0 – INT2IP** External Interrupt 2 Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

### 11.13.50 IPR11

**Name:** IPR11
**Address:** 0x36D

Peripheral Interrupt Priority Register 11

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | DMA4AIP | DMA4ORIP | DMA4DCNTIP | DMA4SCNTIP | TMR4IP | CWG3IP | CLC6IP | CCP3IP |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Bit 7 – DMA4AIP** DMA4 Abort Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 6 – DMA4ORIP** DMA4 Overrun Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 5 – DMA4DCNTIP** DMA4 Destination Count Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 4 – DMA4SCNTIP** DMA4 Source Count Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 3 – TMR4IP** TMR4 Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 2 – CWG3IP** CWG3 Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 1 – CLC6IP** CLC6 Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 0 – CCP3IP** CCP3 Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

### 11.13.51 IPR12

**Name:** IPR12
**Address:** 0x36E

Peripheral Interrupt Priority Register 12

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | DMA5AIP | DMA5ORIP | DMA5DCNTIP | DMA5SCNTIP | U4IP | U4EIP | U4TXIP | U4RXIP |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Bit 7 – DMA5AIP** DMA5 Abort Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 6 – DMA5ORIP** DMA5 Overrun Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 5 – DMA5DCNTIP** DMA5 Destination Count Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 4 – DMA5SCNTIP** DMA5 Source Count Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 3 – U4IP** UART 4 Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 2 – U4EIP** UART4 Framing Error Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 1 – U4TXIP** UART4 Transmit Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 0 – U4RXIP** UART4 Receive Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

### 11.13.52 IPR13

**Name:** IPR13
**Address:** 0x36F

Peripheral Interrupt Priority Register 13

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | DMA6AIP | DMA6ORIP | DMA6DCNTIP | DMA6SCNTIP | U5IP | U5EIP | U5TXIP | U5RXIP |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Bit 7 – DMA6AIP** DMA6 Abort Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 6 – DMA6ORIP** DMA6 Overrun Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 5 – DMA6DCNTIP** DMA6 Destination Count Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 4 – DMA6SCNTIP** DMA6 Source Count Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 3 – U5IP** UART5 Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 2 – U5EIP** UART5 Framing Error Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 1 – U5TXIP** UART5 Transmit Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 0 – U5RXIP** UART5 Receive Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

### 11.13.53 IPR14

**Name:** IPR14
**Address:** 0x370

Peripheral Interrupt Priority Register 14

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | NCO3IP | CM2IP | CLC7IP | |
| Access | | | | | R/W | R/W | R/W | |
| Reset | | | | | 1 | 1 | 1 | |

**Bit 3 – NCO3IP** NCO3 Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 2 – CM2IP** CMP2 Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 1 – CLC7IP** CLC7 Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

### 11.13.54 IPR15

**Name:** IPR15
**Address:** 0x371

Peripheral Interrupt Priority Register 15

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | TMR6IP | CRCIP | CLC8IP | NVMIP |
| Access | | | | | R/W | R/W | R/W | R/W |
| Reset | | | | | 1 | 1 | 1 | 1 |

**Bit 3 – TMR6IP** TMR6 Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 2 – CRCIP** CRC Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 1 – CLC8IP** CLC8 Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

**Bit 0 – NVMIP** NVM Interrupt Priority

| Value | Description |
|---|---|
| 1 | High Priority |
| 0 | Low Priority |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

## 11.14 Register Summary - Interrupts

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0362 | IPR0 | 7:0 | IOCIP | | CLC1IP | | CSWIP | OSFIP | HLVDIP | SWIP |
| 0x0363 | IPR1 | 7:0 | SMT1PWAIP | SMT1PRAIP | SMT1IP | CM1IP | ACTIP | ADIP | ZCDIP | INT0IP |
| 0x0364 | IPR2 | 7:0 | DMA1AIP | DMA1ORIP | DMA1DCNTIP | DMA1SCNTIP | | | | ADTIP |
| 0x0365 | IPR3 | 7:0 | TMR0IP | CCP1IP | TMR1GIP | TMR1IP | TMR2IP | SPI1IP | SPI1TXIP | SPI1RXIP |
| 0x0366 | IPR4 | 7:0 | PWM1IP | PWM1PIP | | | U1IP | U1EIP | U1TXIP | U1RXIP |
| 0x0367 | IPR5 | 7:0 | PWM2IP | PWM2PIP | TMR3GIP | TMR3IP | | SPI2IP | SPI2TXIP | SPI2RXIP |
| 0x0368 | IPR6 | 7:0 | DMA2AIP | DMA2ORIP | DMA2DCNTIP | DMA2SCNTIP | NCO1IP | CWG1IP | CLC2IP | INT1IP |
| 0x0369 | IPR7 | 7:0 | PWM3IP | PWM3PIP | CLC3IP | | I2C1EIP | I2C1IP | I2C1TXIP | I2C1RXIP |
| 0x036A | IPR8 | 7:0 | SCANIP | CCP2IP | TMR5GIP | TMR5IP | U2IP | U2EIP | U2TXIP | U2RXIP |
| 0x036B | IPR9 | 7:0 | | | CLC4IP | | U3IP | U3EIP | U3TXIP | U3RXIP |
| 0x036C | IPR10 | 7:0 | DMA3AIP | DMA3ORIP | DMA3DCNTIP | DMA3SCNTIP | NCO2IP | CWG2IP | CLC5IP | INT2IP |
| 0x036D | IPR11 | 7:0 | DMA4AIP | DMA4ORIP | DMA4DCNTIP | DMA4SCNTIP | TMR4IP | CWG3IP | CLC6IP | CCP3IP |
| 0x036E | IPR12 | 7:0 | DMA5AIP | DMA5ORIP | DMA5DCNTIP | DMA5SCNTIP | U4IP | U4EIP | U4TXIP | U4RXIP |
| 0x036F | IPR13 | 7:0 | DMA6AIP | DMA6ORIP | DMA6DCNTIP | DMA6SCNTIP | U5IP | U5EIP | U5TXIP | U5RXIP |
| 0x0370 | IPR14 | 7:0 | | | | | NCO3IP | CM2IP | CLC7IP | |
| 0x0371 | IPR15 | 7:0 | | | | | TMR6IP | CRCIP | CLC8IP | NVMIP |
| 0x0372 ... 0x0375 | Reserved | | | | | | | | | |
| 0x0376 | SHADCON | 7:0 | | | | | | | | SHADLO |
| 0x0377 ... 0x0458 | Reserved | | | | | | | | | |
| 0x0459 | IVTLOCK | 7:0 | | | | | | | | IVTLOCKED |
| 0x045A | IVTAD | 7:0 | IVTADL[7:0] | | | | | | | |
| | | 15:8 | IVTADH[7:0] | | | | | | | |
| | | 23:16 | | | | IVTADU[4:0] | | | | |
| 0x045D | IVTBASE | 7:0 | IVTBASEL[7:0] | | | | | | | |
| | | 15:8 | IVTBASEH[7:0] | | | | | | | |
| | | 23:16 | | | | IVTBASEU[4:0] | | | | |
| 0x0460 ... 0x049D | Reserved | | | | | | | | | |
| 0x049E | PIE0 | 7:0 | IOCIE | | CLC1IE | | CSWIE | OSFIE | HLVDIE | SWIE |
| 0x049F | PIE1 | 7:0 | SMT1PWAIE | SMT1PRAIE | SMT1IE | CM1IE | ACTIE | ADIE | ZCDIE | INT0IE |
| 0x04A0 | PIE2 | 7:0 | DMA1AIE | DMA1ORIE | DMA1DCNTIE | DMA1SCNTIE | | | | ADTIE |
| 0x04A1 | PIE3 | 7:0 | TMR0IE | CCP1IE | TMR1GIE | TMR1IE | TMR2IE | SPI1IE | SPI1TXIE | SPI1RXIE |
| 0x04A2 | PIE4 | 7:0 | PWM1IE | PWM1PIE | | | U1IE | U1EIE | U1TXIE | U1RXIE |
| 0x04A3 | PIE5 | 7:0 | PWM2IE | PWM2PIE | TMR3GIE | TMR3IE | | SPI2IE | SPI2TXIE | SPI2RXIE |
| 0x04A4 | PIE6 | 7:0 | DMA2AIE | DMA2ORIE | DMA2DCNTIE | DMA2SCNTIE | NCO1IE | CWG1IE | CLC2IE | INT1IE |
| 0x04A5 | PIE7 | 7:0 | PWM3IE | PWM3PIE | CLC3IE | | I2C1EIE | I2C1IE | I2C1TXIE | I2C1RXIE |
| 0x04A6 | PIE8 | 7:0 | SCANIE | CCP2IE | TMR5GIE | TMR5IE | U2IE | U2EIE | U2TXIE | U2RXIE |
| 0x04A7 | PIE9 | 7:0 | | | CLC4IE | | U3IE | U3EIE | U3TXIE | U3RXIE |
| 0x04A8 | PIE10 | 7:0 | DMA3AIE | DMA3ORIE | DMA3DCNTIE | DMA3SCNTIE | NCO2IE | CWG2IE | CLC5IE | INT2IE |
| 0x04A9 | PIE11 | 7:0 | DMA4AIE | DMA4ORIE | DMA4DCNTIE | DMA4SCNTIE | TMR4IE | CWG3IE | CLC6IE | CCP3IE |
| 0x04AA | PIE12 | 7:0 | DMA5AIE | DMA5ORIE | DMA5DCNTIE | DMA5SCNTIE | U4IE | U4EIE | U4TXIE | U4RXIE |
| 0x04AB | PIE13 | 7:0 | DMA6AIE | DMA6ORIE | DMA6DCNTIE | DMA6SCNTIE | U5IE | U5EIE | U5TXIE | U5RXIE |
| 0x04AC | PIE14 | 7:0 | | | | | NCO3IE | CM2IE | CLC7IE | |
| 0x04AD | PIE15 | 7:0 | | | | | TMR6IE | CRCIE | CLC8IE | NVMIE |
| 0x04AE | PIR0 | 7:0 | IOCIF | | CLC1IF | | CSWIF | OSFIF | HLVDIF | SWIF |
| 0x04AF | PIR1 | 7:0 | SMT1PWAIF | SMT1PRAIF | SMT1IF | CM1IF | ACTIF | ADIF | ZCDIF | INT0IF |
| 0x04B0 | PIR2 | 7:0 | DMA1AIF | DMA1ORIF | DMA1DCNTIF | DMA1SCNTIF | | | | ADTIF |
| 0x04B1 | PIR3 | 7:0 | TMR0IF | CCP1IF | TMR1GIF | TMR1IF | TMR2IF | SPI1IF | SPI1TXIF | SPI1RXIF |
| 0x04B2 | PIR4 | 7:0 | PWM1IF | PWM1PIF | | | U1IF | U1EIF | U1TXIF | U1RXIF |
| 0x04B3 | PIR5 | 7:0 | PWM2IF | PWM2PIF | TMR3GIF | TMR3IF | | SPI2IF | SPI2TXIF | SPI2RXIF |
| 0x04B4 | PIR6 | 7:0 | DMA2AIF | DMA2ORIF | DMA2DCNTIF | DMA2SCNTIF | NCO1IF | CWG1IF | CLC2IF | INT1IF |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

.....continued

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x04B5 | PIR7 | 7:0 | PWM3IF | PWM3PIF | CLC3IF | | I2C1EIF | I2C1IF | I2C1TXIF | I2C1RXIF |
| 0x04B6 | PIR8 | 7:0 | SCANIF | CCP2IF | TMR5GIF | TMR5IF | U2IF | U2EIF | U2TXIF | U2RXIF |
| 0x04B7 | PIR9 | 7:0 | | | CLC4IF | | U3IF | U3EIF | U3TXIF | U3RXIF |
| 0x04B8 | PIR10 | 7:0 | DMA3AIF | DMA3ORIF | DMA3DCNTIF | DMA3SCNTIF | NCO2IF | CWG2IF | CLC5IF | INT2IF |
| 0x04B9 | PIR11 | 7:0 | DMA4AIF | DMA4ORIF | DMA4DCNTIF | DMA4SCNTIF | TMR4IF | CWG3IF | CLC6IF | CCP3IF |
| 0x04BA | PIR12 | 7:0 | DMA5AIF | DMA5ORIF | DMA5DCNTIF | DMA5SCNTIF | U4IF | U4EIF | U4TXIF | U4RXIF |
| 0x04BB | PIR13 | 7:0 | DMA6AIF | DMA6ORIF | DMA6DCNTIF | DMA6SCNTIF | U5IF | U5EIF | U5TXIF | U5RXIF |
| 0x04BC | PIR14 | 7:0 | | | | | NCO3IF | CM2IF | CLC7IF | |
| 0x04BD | PIR15 | 7:0 | | | | | TMR6IF | CRCIF | CLC8IF | NVMIF |
| 0x04BE ... 0x04D5 | Reserved | | | | | | | | | |
| 0x04D6 | INTCON0 | 7:0 | GIE/GIEH | GIEL | IPEN | | | INT2EDG | INT1EDG | INT0EDG |
| 0x04D7 | INTCON1 | 7:0 | STAT[1:0] | | | | | | | |

# 12. OSC - Oscillator Module (With Fail-Safe Clock Monitor)

The oscillator module contains multiple clock sources and selection features that allow it to be used in a wide range of applications while maximizing performance and minimizing power consumption.

Clock sources can be supplied either internally or externally. External sources include:

- External clock oscillators
- Quartz crystal resonators
- Ceramic resonators
- Secondary Oscillator (SOSC)

Internal sources include:

- High-Frequency Internal Oscillator (HFINTOSC)
- Low-Frequency Internal Oscillator (LFINTOSC)
- Analog-to-Digital Converter RC Oscillator (ADCRC)

Special features of the oscillator module include:

- Oscillator Start-up Timer (OST): Ensures stability of quartz crystal or ceramic resonators
- 4x Phase-Locked Loop (PLL): Frequency multiplier for external clock sources
- HFINTOSC Frequency Adjustment: Provides the ability to adjust the HFINTOSC frequency
- Clock switching: Allows the system clock to switch between internal or external sources via software during run time
- Fail-Safe Clock Monitor (FSCM): Designed to detect a failure of the external clock source and automatically switch to an internal clock source

The Reset Oscillator (RSTOSC) Configuration bits determine the type of oscillator that will be used when the device runs after a Reset, including when the device is first powered up (see the table below).

**Table 12-1. RSTOSC Selection Table**

| RSTOSC | SFR Reset Values | | | Clock Source |
|---|---|---|---|---|
| | NOSC / COSC | NDIV / CDIV | OSCFRQ | |
| 111 | 111 | 0000 (1:1) | 0010 (4 MHz) | EXTOSC per FEXTOSC |
| 110 | 110 | 0010 (4:1) | | HFINTOSC @ 1 MHz |
| 101 | 101 | 0000 (1:1) | | LFINTOSC |
| 100 | 100 | 0000 (1:1) | | SOSC |
| 011 | Reserved | | | |
| 010 | 010 | 0000 (1:1) | 0010 (4 MHz) | EXTOSC + 4x PLL(1) |
| 001 | Reserved | | | |
| 000 | 000 | 0000 (1:1) | 1000 (64 MHz) | HFINTOSC @ 64 MHz |

**Note:**

1. EXTOSC must meet the PLL specifications (see the data sheet Electrical Specifications).

If an external clock source is selected by the RSTOSC bits, the External Oscillator Mode Select (FEXTOSC) Configuration bits must be used to select the External Clock mode. These modes include:

- ECL: External Clock Low Power mode
- ECM: External Clock Medium Power mode
- ECH: External Clock High Power mode
- LP: 32 kHz Low-Gain Crystal mode
- XT: Medium-Gain Crystal or Ceramic Resonator mode

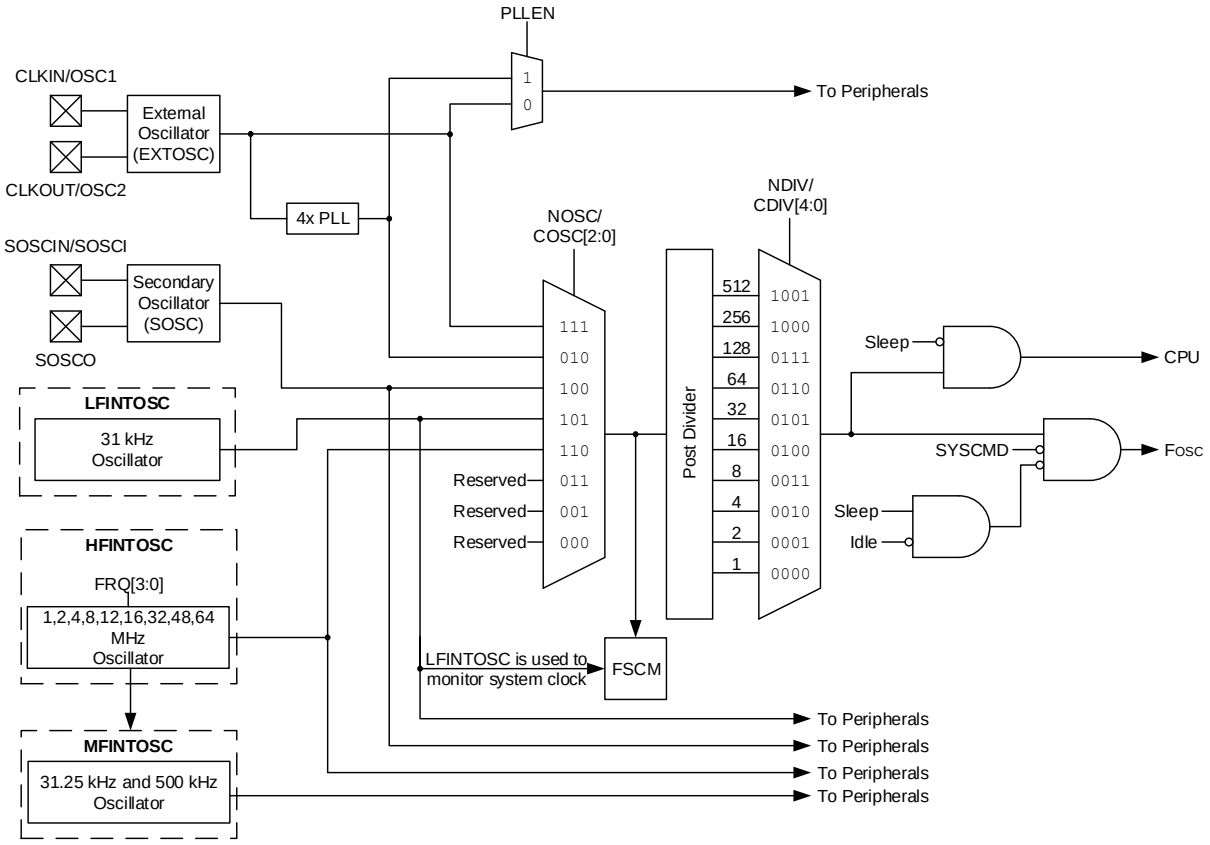
© 2019-2023 Microchip Technology Inc.
and its subsidiaries

- HS: High-Gain Crystal or Ceramic Resonator mode

The ECH, ECM and ECL modes rely on an external logic-level signal as the device clock source. The LP, XT and HS modes rely on an external quartz crystal or ceramic resonator as the device clock source. Each mode is optimized for a specific frequency range. The internal oscillator block produces both low-frequency and high-frequency clock signals, designated LFINTOSC and HFINTOSC, respectively. Multiple system operating frequencies may be derived from these clock sources.

The figure below illustrates a block diagram of the oscillator module.

**Figure 12-1. Clock Source Block Diagram**




## 12.1 Clock Source Types

Clock sources can be classified as external or internal.

External clock sources rely on external circuitry for the clock source to function. Examples of external clock sources include:

- Digital oscillator modules
- Quartz crystal resonators
- Ceramic resonators

A 4x PLL is provided for use with external clock sources.

Internal clock sources are contained within the oscillator module. The internal oscillator block features two internal oscillators that are used to generate internal system clock sources. The High-Frequency Internal Oscillator (HFINTOSC) can produce a wide range of frequencies which are determined via the HFINTOSC Frequency Selection (OSCFRQ) register. The Low-Frequency Internal Oscillator (LFINTOSC) generates a fixed nominal 31 kHz clock

signal. The internal oscillator block also features an RC oscillator which is dedicated to the Analog-to-Digital Converter (ADC).

The oscillator module allows the system clock source or system clock frequency to be changed through clock switching. Clock source selections are made via the New Oscillator Source Request (NOSC) bits. Once the clock source has been selected, the clock source base frequency can be divided (post-scaled) via the New Divider Selection Request (NDIV) bits.

The instruction clock ($F_{OSC}/4$) can be routed to the OSC2/CLKOUT pin when the pin is not in use. The Clock Out Enable ($\overline{CLKOUTEN}$) Configuration bit controls the functionality of the CLKOUT signal. When $\overline{CLKOUTEN}$ is clear ($\overline{CLKOUTEN}$ = 0), the CLKOUT signal is routed to the OSC2/CLKOUT pin. When $\overline{CLKOUTEN}$ is set ($\overline{CLKOUTEN}$ = 1), the OSC2/CLKOUT pin functions as an I/O pin.

### 12.1.1 External Clock Sources

An external clock source can be used as the device system clock by performing one of the following actions:

- Program the RSTOSC and FEXTOSC Configuration bits to select an external clock source that will be used as the default system clock upon a device Reset.
- Write the NOSC and NDIV bits to switch the system clock source during run time.

#### 12.1.1.1 EC Mode

The External Clock (EC) mode allows an externally generated logic level signal to be the system clock source. When operating in EC mode, an external clock source is connected to the OSC1/CLKIN input pin. The OSC2/CLKOUT pin is available as a general purpose I/O pin or as the CLKOUT signal pin.

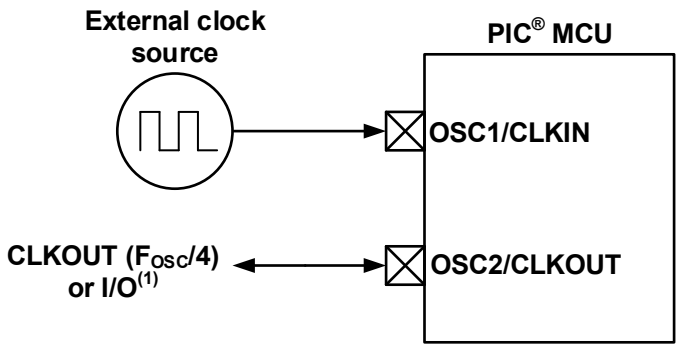
EC mode provides three Power mode selections:

- ECH: High Power mode
- ECM: Medium Power mode
- ECL: Low Power mode

The Oscillator Start-up Timer (OST) is disabled when EC mode is selected; therefore, there is no delay in operation after a Power-on Reset (POR) or wake-up from Sleep. Because the PIC® MCU design is fully static, stopping the external clock input will have the effect of halting the device while leaving all data intact. Upon restarting the external clock, the device will resume operation as if no time had elapsed.

The figure below shows the pin connections for EC mode.

**Figure 12-2. External Clock (EC) Mode Operation**




**Note:**

1. Output depends on the setting of the $\overline{CLKOUTEN}$ Configuration bit.

#### 12.1.1.2 LP, XT, HS Modes

The LP, XT and HS modes support the use of quartz crystals or ceramic resonators connected to the OSC1 and OSC2 pins, as shown in the figures below. These three modes select a low, medium, or high-gain setting of the internal inverter-amplifier to support various resonator types and speeds.

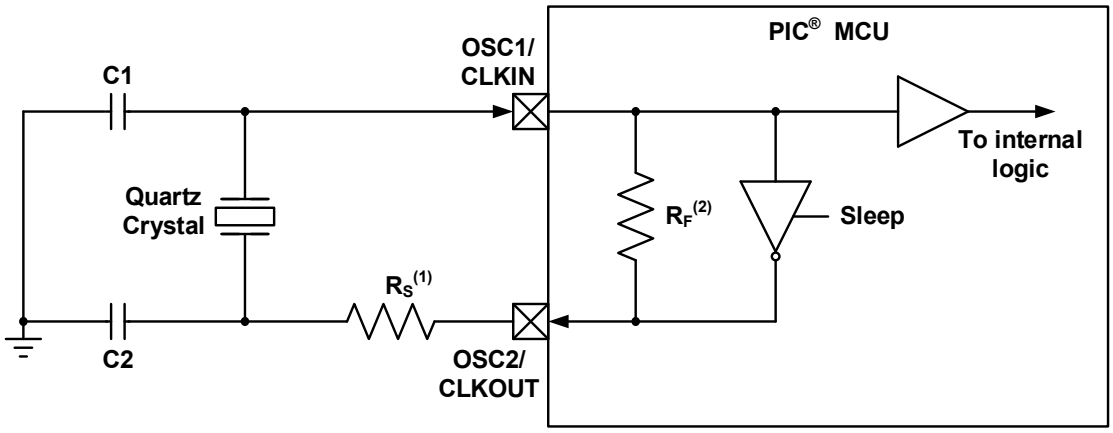
The LP Oscillator mode selects the lowest gain setting of the internal inverter-amplifier, and consumes the least amount of current. LP mode is designed to drive 32.768 kHz tuning-fork type crystals (watch crystals), but can operate up to 100 kHz.

The XT Oscillator mode selects the intermediate gain setting of the internal inverter-amplifier. Current consumption is at a medium level when compared to the other two modes. XT mode is best suited to drive crystal and ceramic resonators with a frequency range up to 4 MHz.

The HS Oscillator mode selects the highest gain setting of the internal inverter-amplifier, and consumes the most current. This mode is best suited for crystal and ceramic resonators that require operating frequencies up to 20 MHz.

The figures below show typical circuits for quartz crystal and ceramic resonators.

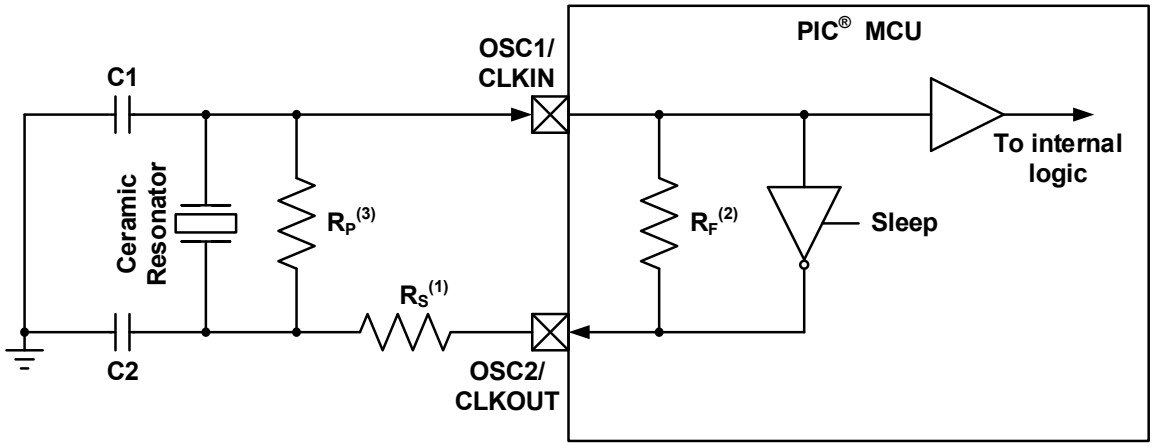
**Figure 12-3. Quartz Crystal Operation**




**Notes:**

1. A series resistor ($R_S$) may be required for quartz crystals with low drive level.
2. The value of $R_F$ varies with the Oscillator mode selected (typically between 2 MΩ and 10 MΩ).

**Figure 12-4. Ceramic Resonator Operation**




**Notes:**

1. A series resistor ($R_S$) may be required for ceramic resonators with low drive level.
2. The value of $R_F$ varies with the Oscillator mode selected (typically between 2 MΩ and 10 MΩ).
3. An additional parallel feedback resistor ($R_P$) may be required for proper ceramic resonator operation.

#### 12.1.1.3 Oscillator Start-Up Timer (OST)

The Oscillator Start-up Timer (OST) ensures that the oscillator circuit has started and is providing a stable system clock to the oscillator module. Quartz crystals or ceramic resonators do not start immediately, and may take a few hundred cycles before the oscillator becomes stable. The oscillations must build up until sufficient amplitude is generated to properly toggle between logic states. The OST counts 1024 oscillation periods from the OSC1 input following a Power-on Reset (POR), Brown-out Reset (BOR), or wake-up from Sleep event to ensure that the oscillator has enough time to reach stable and accurate operation. Once the OST has completed its count, module hardware sets the External Oscillator Ready (EXTOR) bit, indicating that the oscillator is stable and ready to use.

#### 12.1.1.4 4x PLL

The oscillator module contains a 4x Phase-Locked Loop (PLL) circuit that can be used with the external clock sources to provide a system clock source. The input frequency for the PLL must fall within a specified range. See the **"PLL Specifications"** table found in the **"Electrical Specifications"** chapter for more information.

The PLL can be enabled for use through one of two methods:

1. Program the RSTOSC Configuration bits to select the "EXTOSC with 4x PLL" option.
2. Write the NOSC bits to select the "EXTOSC with 4x PLL" option.

#### 12.1.1.5 Secondary Oscillator

The Secondary Oscillator (SOSC) is a separate external oscillator block that can be used as an alternate system clock source or as a Timer clock source. The SOSC is optimized for 32.768 kHz, and can be used with either an external quartz crystal connected to the SOSCI and SOSCO pins, or with an external clock source connected to the SOSCI pin as shown in the figures below.

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

**Figure 12-5. SOSC 32.768 kHz Quartz Crystal Oscillator Operation**

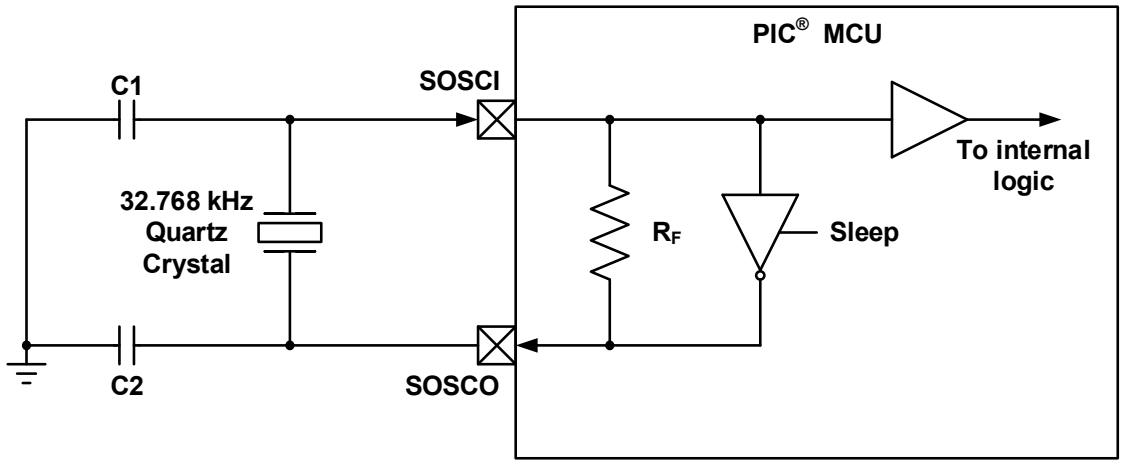
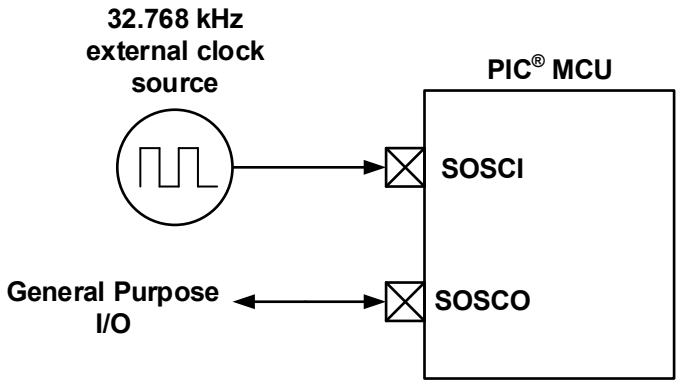



**Figure 12-6. SOSC 32.768 kHz External Clock Operation**




The SOSC can be enabled through one of two methods:

- Programming the RSTOSC Configuration bits to select the SOSC as the system clock.
- Programming the NOSC bits to select the SOSC during run time.

Two Power modes are available for the secondary oscillator, and are selected using the Secondary Oscillator Power Mode Select (SOSCPWR) bit. When SOSCPWR is clear (SOSCPWR = 0), the oscillator operates in Low Power mode, which is ideal for crystal oscillators with low drive strength. When SOSCPWR is set (SOSCPWR = 1), the oscillator operates in High Power mode, which is ideal for crystal oscillators with high drive strength or high Equivalent Series Resistance (ESR).

**Important:** The SOSC module must be disabled before changing Power modes. Changes to the Power mode during operation may result in undefined oscillator behavior.

#### 12.1.1.5.1 SOSC Start-Up Timing

The SOSC utilizes the Oscillator Start-up Timer (OST) to ensure that the 32.768 kHz crystal oscillator has started and is available for use. Since crystal oscillators do not start immediately and may take a few hundred cycles before achieving stable operation, the OST counts 1024 oscillation periods from the SOSCI input. Once the OST completes its count, module hardware sets the Secondary Oscillator Ready (SOR) bit, indicating that the SOSC is stable and ready to use.

### 12.1.2 Internal Clock Sources

The internal oscillator block contains two independent oscillators that can produce two internal system clock sources:

- High-Frequency Internal Oscillator (HFINTOSC)
- Low-Frequency Internal Oscillator (LFINTOSC)

Internal oscillator selection is performed one of two ways:

1. Program the RSTOSC Configuration bits to select one of the INTOSC sources which will be used upon a device Reset.
2. Write the New Oscillator Source Request (NOSC) bits to select an internal oscillator during run time.

In INTOSC mode, the OSC1/CLKIN and OSC2/CLKOUT pins are available for use as a general purpose I/Os, provided that no external oscillator is connected. The function of the OSC2/CLKOUT pin is determined by the $\overline{\text{CLKOUTEN}}$ Configuration bit. When $\overline{\text{CLKOUTEN}}$ is set ($\overline{\text{CLKOUTEN}}$ = 1), the pin functions as a general-purpose I/O. When $\overline{\text{CLKOUTEN}}$ is clear ($\overline{\text{CLKOUTEN}}$ = 0), the system instruction clock ($F_{OSC}/4$) is available as an output signal on the pin.

#### 12.1.2.1 HFINTOSC

The High-Frequency Internal Oscillator (HFINTOSC) is a factory-calibrated, precision digitally-controlled internal clock source that produces a wide range of stable clock frequencies. The HFINTOSC can be enabled through one of the following methods:

- Program the RSTOSC Configuration bits to select the HFINTOSC upon device Reset or power-up
- Write to the New Oscillator Source Request (NOSC) bits to select the HFINTOSC during run time.

The HFINTOSC frequency is selected via the HFINTOSC Frequency Selection (FRQ) bits. Fine-tuning of the HFINTOSC is done via the HFINTOSC Frequency Tuning (TUN) bits. The HFINTOSC output frequency can be divided (post-scaled) via the New Divider Selection Request (NDIV) bits.

##### 12.1.2.1.1 HFINTOSC Frequency Tuning

The HFINTOSC frequency can be fine-tuned via the HFINTOSC Tuning (OSCTUNE) register. The OSCTUNE register is used by Active Clock Tuning hardware or user software to provide small adjustments to the HFINTOSC nominal frequency.

The OSCTUNE register contains the HFINTOSC Frequency Tuning (TUN) bits. The TUN bits default to a 6-bit, two's compliment value of 0x00, which indicates that the oscillator is operating at the selected frequency. When a value between 0x01 and 0x1F is written to the TUN bits, the HFINTOSC frequency is increased. When a value between 0x3F and 0x20 is written to the TUN bits, the HFINTOSC frequency is decreased.

When the OSCTUNE register is modified, the oscillator will begin to shift to the new frequency. Code execution continues during this shift. There is no indication that the frequency shift occurred.

**Important:** OSCTUNE tuning does not affect the LFINTOSC frequency.

#### 12.1.2.2 MFINTOSC

The Medium-Frequency Internal Oscillator (MFINTOSC) generates two constant clock outputs (500 kHz and 31.25 kHz). The MFINTOSC clock signals are created from the HFINTOSC using dynamic divider logic, which provides constant MFINTOSC clock rates regardless of selected HFINTOSC frequency.

The MFINTOSC cannot be used as the system clock, but can be used as a clock source for certain peripherals, such as a Timer.

#### 12.1.2.3 LFINTOSC

The Low-Frequency Internal Oscillator (LFINTOSC) is a factory-calibrated 31 kHz internal clock source.

The LFINTOSC can be used as a system clock source, and may be used by certain peripheral modules as a clock source. Additionally, the LFINTOSC provides a time base for the following:

- Power-up Timer (PWRT)
- Watchdog Timer (WDT)/Windowed Watchdog Timer (WWDT)
- Fail-Safe Clock Monitor (FSCM)

The LFINTOSC is enabled through one of the following methods:

- Program the RSTOSC Configuration bits to select LFINTOSC
- Write the NOSC bits to select LFINTOSC during run time

#### 12.1.2.4 ADCRC

The Analog-to-Digital RC (ADCRC) oscillator is dedicated to the ADC module. ADCRC operates at a fixed frequency of approximately 600 kHz, and is used as a conversion clock source. The ADCRC allows the ADC module to operate in Sleep mode, which can reduce system noise during the ADC conversion. The ADCRC is automatically enabled when it is selected as the clock source for the ADC module, or when selected as the clock source of any peripheral that may use it. The ADCRC may also be manually enabled via the ADC Oscillator Enable (ADOEN) bit, thereby avoiding start-up delays when this source is used intermittently.

### 12.1.3 Oscillator Status and Manual Enable

The Oscillator Status (OSCSTAT) register displays the Ready status for each of the following oscillators:

- External oscillator
- HFINTOSC
- MFINTOSC
- LFINTOSC
- SOSC
- ADCRC

The OSCSTAT register also displays the Ready status for the 4xPLL.

The HFINTOSC Oscillator Ready (HFOR) and MFINTOSC Oscillator Ready (MFOR) Status bits indicate whether the respective oscillators are ready for use. Both clock sources are available for use at any time, but may require a finite amount of time before they have reached the specified accuracy levels. When the HFINTOSC or MFINTOSC are ready and achieved the specified accuracy, module hardware sets the HFOR/MFOR bits, respectively.

When a new value is loaded into the OSCFRQ register, the HFOR and MFOR bits are cleared by hardware, and will be set again once the respective oscillator is ready. During pending OSCFRQ changes, the MFINTOSC will stall at either a high or a low state until the HFINTOSC locks in the new frequency and resumes operation.

The Oscillator Enable (OSCEN) register can be used to manually enable the following oscillators:

- External oscillator
- HFINTOSC
- MFINTOSC
- LFINTOSC
- SOSC
- ADCRC

**Important:** OSCEN cannot be used to manually enable the 4xPLL.

## 12.2 Clock Switching

The system clock source can be switched between external and internal clock sources via software using the New Oscillator Source Request (NOSC) and New Divider Selection Request (NDIV) bits. The following sources can be selected:

- External Oscillator (EXTOSC)
- EXTOSC with 4x PLL
- High-Frequency Internal Oscillator (HFINTOSC)
- Low-Frequency Internal Oscillator (LFINTOSC)
- Secondary Oscillator (SOSC)

The Clock Switch Enable (CSWEN) Configuration bit can be used to enable or disable the clock switching capability. When CSWEN is set (CSWEN = 1), writes to NOSC and NDIV by user software will allow the system clock to switch between sources or frequencies. When CSWEN is clear (CSWEN = 0), writes to NOSC and NDIV are ignored, preventing the system clock from switching from one source to another.

### 12.2.1 NOSC and NDIV Bits

The New Oscillator Source Request (NOSC) and New Divider Selection Request (NDIV) bits are used to select the system clock source and clock frequency divider that will be used by the CPU and peripherals (see the tables below).

When new values are written into NOSC and/or NDIV, the current oscillator selection will continue to operate as the system clock while waiting for the new source to indicate that it is ready. Writes to NDIV without changing the clock source (e.g., changing the HFINTOSC frequency from 1 MHz to 2 MHz) are handled in the same manner as a clock switch.

When the new oscillator selection is ready, the New Oscillator is Ready (NOSCR) bit and the Clock Switch Interrupt Flag (CSWIF) are set by module hardware. If the Clock Switch Interrupt Enable (CSWIE) bit is set (CSWIE = 1), an interrupt will be generated when CSWIF is set. Additionally, the Oscillator Ready (ORDY) bit can be polled to determine that the clock switch has completed and the new oscillator source has replaced the old source as the system clock.

**Important:** The CSWIF interrupt does not wake the device from Sleep.

**Table 12-2. NOSC/COSC Clock Source Selection Table**

| NOSC / COSC | Clock Source |
|---|---|
| 111 | EXTOSC(1) |
| 110 | HFINTOSC(2) |
| 101 | LFINTOSC |
| 100 | SOSC |
| 011 | Reserved |
| 010 | EXTOSC + 4xPLL(3) |
| 001 | Reserved |
| 000 | Reserved |

**Notes:**

1. EXTOSC is configured via the FEXTOSC Configuration bits.
2. HFINTOSC frequency is determined by the FRQ bits.
3. EXTOSC must meet the PLL specifications (see the data sheet Electrical Specifications).

© 2019-2023 Microchip Technology Inc. and its subsidiaries

**Table 12-3. NDIV/CDIV Clock Divider Selection Table**

| NDIV / CDIV | Clock Divider |
|---|---|
| 1111-1010 | Reserved |
| 1001 | 512 |
| 1000 | 256 |
| 0111 | 128 |
| 0110 | 64 |
| 0101 | 32 |
| 0100 | 16 |
| 0011 | 8 |
| 0010 | 4 |
| 0001 | 2 |
| 0000 | 1 |

### 12.2.2 COSC and CDIV Bits

The Current Oscillator Source Select (COSC) bits and the Current Divider Select (CDIV) bits indicate the current oscillator source and clock divider, respectively. When a new oscillator or divider is requested via the NOSC/NDIV bits, the COSC and CDIV bits remain unchanged until the clock switch actually occurs. When the switch actually occurs, hardware copies the NOSC and NDIV values into COSC and CDIV, the Oscillator Ready (ORDY) bit is set, and the NOSCR bit is cleared by hardware, indicating that the clock switch is complete.

### 12.2.3 CSWHOLD

When the system oscillator changes frequencies, peripherals using the system clock may be affected. For example, if the $I^2C$ module is actively using the system clock as its Serial Clock (SCL) time base, changing the system clock frequency will change the SCL frequency. The Clock Switch Hold (CSWHOLD) bit can be used to suspend a requested clock switch. In this example, software can request a new clock source, use the CSWHOLD bit to suspend the switch, wait for the $I^2C$ bus to become Idle, then reconfigure the SCL frequency based on the new clock source. Once the $I^2C$ has been reconfigured, software can use CSWHOLD to complete the clock switch without causing any issues with the $I^2C$ bus.

When CSWHOLD is set (CSWHOLD = 1), a write to NOSC and/or NDIV is accepted, but the clock switch is suspended and does not automatically complete. While the switch is suspended, code execution continues using the old (current) clock source. Module hardware will still enable the new oscillator selection and set the NOSCR bit. Once the NOSCR bit is set, software will either:

- clear CSWHOLD so that the clock switch can complete, or
- copy the Current Oscillator Source Select (COSC) value into NOSC to abandon the clock switch.

When CSWHOLD is clear (CSWHOLD = 0), the clock switch will occur when the NOSCR bit is set. When NOSCR is set, the CSWIF is also set, and if CSWIE is set, the generated interrupt will be serviced using the new oscillator.

### 12.2.4 PLL Input Switch

Switching between the PLL and any non-PLL source is handled in the same manner as any other clock source change.

When the NOSC selects a source with a PLL, the system continues to operate using the current oscillator until the new oscillator is ready. When the new source is ready, the associated Status bit in the Oscillator Status (OSCSTAT) register is set, and once the PLL is locked and ready for use, the PLL is Ready (PLLR) bit is set. Once both the source and PLL are ready, the switch will complete.

© 2019-2023 Microchip Technology Inc. and its subsidiaries

### 12.2.5 Clock Switch and Sleep

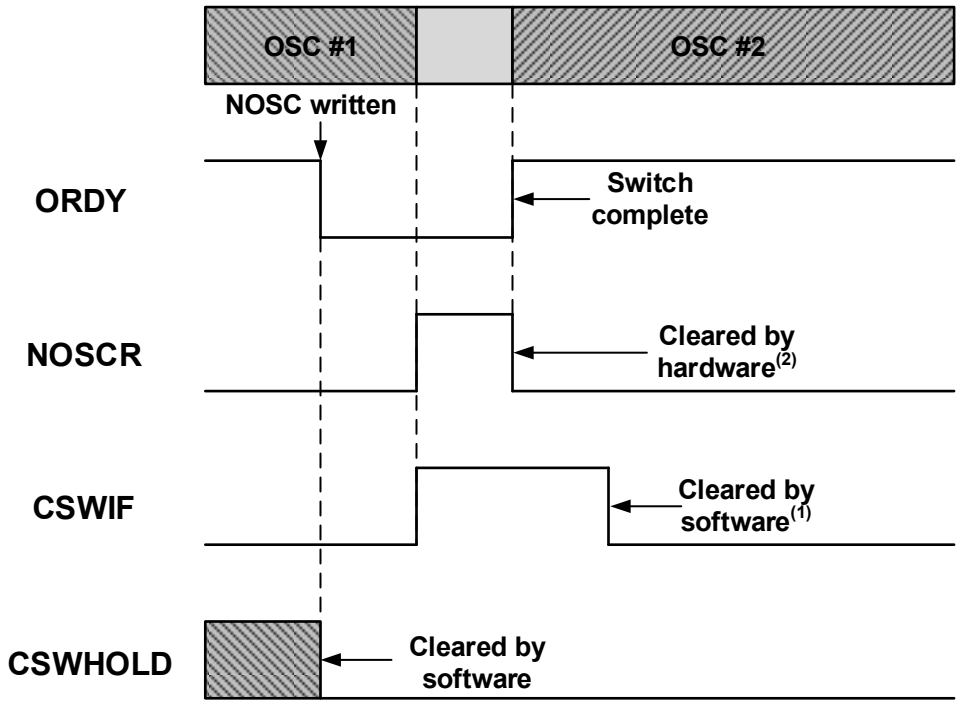
If the NOSC/NDIV bits are written with new values and the device is put to Sleep before the clock switch completes, the switch will not take place and the device will enter Sleep mode.

When the device wakes up from Sleep and CSWHOLD is clear (CSWHOLD = 0), the clock switch will complete and the device will wake with the new clock active, setting CSWIF.

When the device wakes from Sleep and CSWHOLD is set (CSWHOLD = 1), the device will wake up with the old clock active, and the new clock source will be requested again.

If Doze mode is in effect, the clock switch occurs on the next clock cycle regardless of whether or not the CPU is active during that clock cycle.

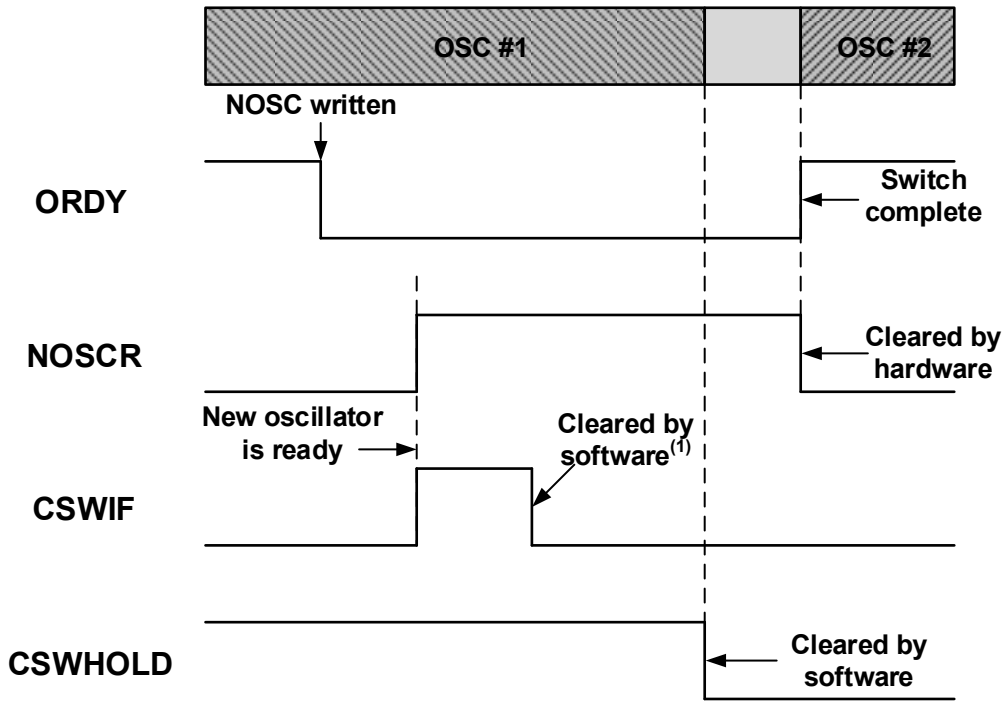
**Figure 12-7. Clock Switch (CSWHOLD = 0)**




**Notes:**

1. CSWIF is asserted coincident with NOSCR; interrupt is serviced at OSC#2 speed.
2. The assertion of NOSCR may not be seen by the user as it is only set for the duration of the switch.

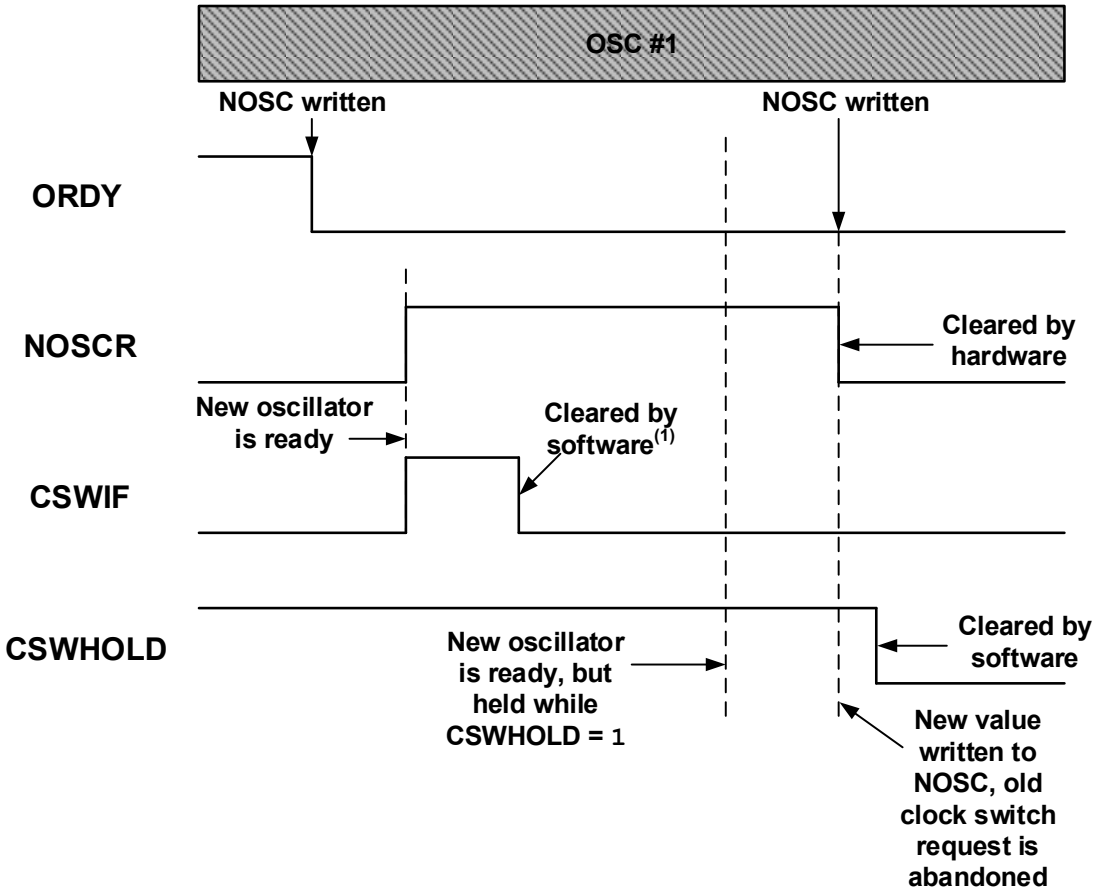
**Figure 12-8. Clock Switch (CSWHOLD = 1)**




**Note:**

1. CSWIF may be cleared before or after clearing CSWHOLD.

**Figure 12-9. Clock Switch Abandoned**




**Note:**

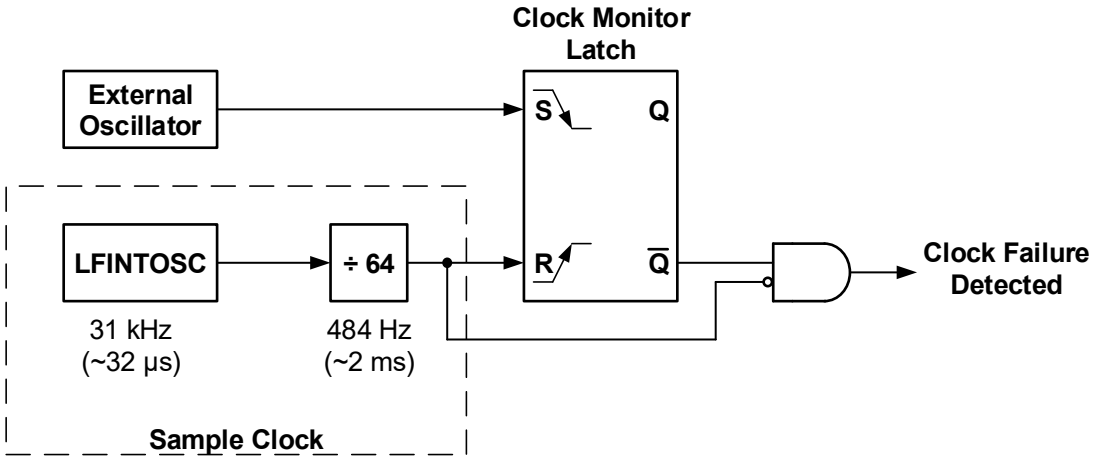
1. CSWIF may be cleared before or after rewriting NOSC; CSWIF is not automatically cleared.

## 12.3 Fail-Safe Clock Monitor (FSCM)

The Fail-Safe Clock Monitor (FSCM) allows the device to continue operating if the external oscillator fails. The FSCM is applicable to all external Oscillator modes. The FSCM is enabled by setting the Fail-Safe Clock Monitor Enable (FCMEN) Configuration bit.

The figure below shows the FSCM block diagram.

**Figure 12-10. FSCM Block Diagram**




### 12.3.1 Fail-Safe Detection

The FSCM detects a failed oscillator by comparing the external oscillator to the FSCM sample clock. The sample clock is generated by dividing the LFINTOSC by 64. The fail detector block contains a latch that is set upon each falling edge of the external clock. The latch is cleared on the rising edge of the sample clock. A failure is detected when a half-period of the sample clock elapses before the external clock goes low.

### 12.3.2 Fail-Safe Operation

When the external clock fails, the FSCM overwrites the NOSC/COSC bits to select the HFINTOSC, and sets the Oscillator Fail Interrupt Flag (OSFIF) bit of the PIR registers. If the Oscillator Fail Interrupt Enable (OSFIE) bit is set, an interrupt will be generated when OSFIF is set. The frequency of HFINTOSC depends on the previous state of the FRQ bits and the NDIV/CDIV bits.

Once a failure is detected, software can take steps to mitigate the effects of a failed oscillator. The system clock will continue to be sourced from the HFINTOSC until the external oscillator is restarted. Once the external source is operational, software can switch back to the external oscillator via the NOSC/NDIV bits.

### 12.3.3 Fail-Safe Condition Clearing

The Fail-Safe condition is cleared after a device Reset, execution of a `SLEEP` instruction, or a change to the NOSC/NDIV bits. When switching to the external oscillator or PLL, the Oscillator Start-up Timer (OST) is restarted. While the OST is running, the device continues to from HFINTOSC. When the OST expires, the Fail-Safe condition is cleared after successfully switching to the external clock source.

**Important:** Software must clear the OSFIF bit before switching to the external oscillator. If the Fail-Safe condition still exists, the OSFIF bit will be set again by module hardware.

### 12.3.4 Reset or Wake-Up from Sleep

The FSCM is designed to detect an oscillator failure after the OST has expired. The OST is used after waking up from Sleep or after any type of Reset, when in either LP, XT or HS mode. If the device is using the EC mode, the FSCM will be active as soon as the Reset or wake-up event has completed.

© 2019-2023 Microchip Technology Inc. and its subsidiaries

## 12.4 Active Clock Tuning (ACT)

Many applications, such as those using UART communication, require an oscillator with an accuracy of ± 1% over the full temperature and voltage range. To meet this level of accuracy, the Active Clock Tuning (ACT) feature utilizes the SOSC frequency of 32.768 kHz to adjust the frequency of the HFINTOSC over voltage and temperature.

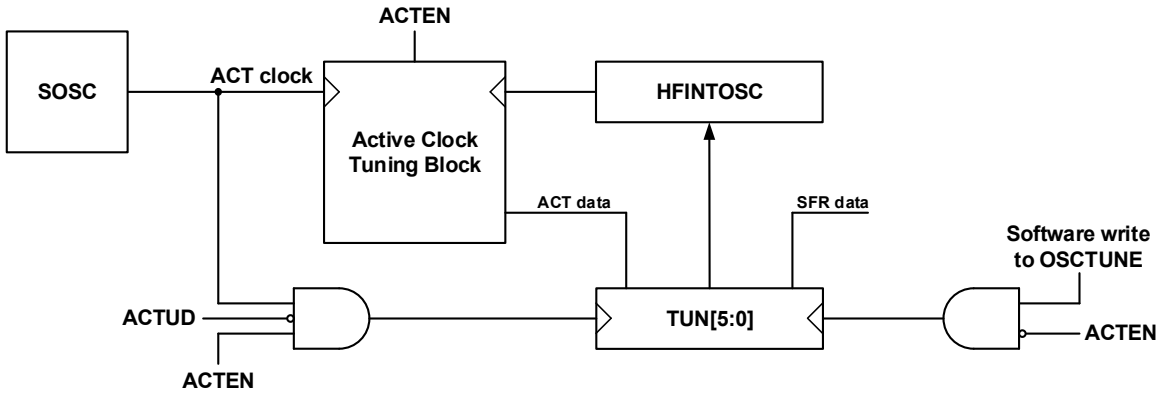
**Important:** Active Clock Tuning requires the use of a 32.768 kHz external oscillator connected to the SOSCI/SOSCO pins.

Active Clock Tuning is enabled via the Active Clock Tuning Enable (ACTEN) bit. When ACTEN is set (ACTEN = 1), the ACT module uses the SOSC time base to measure the HFINTOSC frequency, and uses the HFINTOSC Frequency Tuning (TUN) bits to adjust the HFINTOSC frequency. When ACTEN is clear (ACTEN = 0), the ACT feature is disabled, and user software can utilize the TUN bits to adjust the HFINTOSC frequency.

**Important:** When the ACT feature is enabled, the TUN bits are controlled directly through module hardware and become read-only bits to user software. Writes to the TUN bits when the ACT feature is enabled are ignored.

The figure below shows the Active Clock Tuning block diagram.

**Figure 12-11. Active Clock Tuning (ACT) Block Diagram**




### 12.4.1 ACT Lock Status

The Active Clock Tuning Lock Status (ACTLOCK) bit can be used to determine when the HFINTOSC has been tuned. When ACTLOCK is set (ACTLOCK = 1), the HFINTOSC frequency has been locked to within ± 1% of the nominal frequency. When ACTLOCK is clear (ACTLOCK = 0), the following conditions may be true:

- The HFINTOSC frequency has not been locked to within ± 1%
- A device Reset occurred
- The ACT feature is disabled

**Important:** The ACTLOCK bit is read-only. Writes to ACTLOCK are ignored.

### 12.4.2 ACT Out-of-Range Status

When Active Clock Tuning is enabled, module hardware uses the TUN bits to achieve high accuracy levels. If the module requires a TUN value outside of its range, the ACT Out-of-Range Status (ACTORS) bit is set by hardware (ACTORS = 1).

The ACTORS bit will be set when:

- The HFINTOSC is tuned to its lowest frequency as determined by the TUN bits, and will require a value lower than the TUN bits can provide to achieve accuracy within ± 1%.
- The HFINTOSC is tuned to its highest frequency as determined by the TUN bits, and will require a value higher than the TUN bits can provide to achieve accuracy within ± 1%.

When an ACT out-of-range event occurs, the HFINTOSC will continue to use the last TUN value until the HFINTOSC frequency returns to the tunable range. Once the HFINTOSC returns to the tunable range, module hardware clears the ACTORS bit.

**Important:** The ACTORS bit is read-only. Writes to ACTORS are ignored.

### 12.4.3 ACT Update Disable

When Active Clock Tuning is enabled, the OSCTUNE register is continuously updated every ACT clock cycle. The ACT Update Disable (ACTUD) bit can be used to suspend updates to the OSCTUNE register. When ACTUD is set (ACTUD = 1), updates to OSCTUNE are suspended, although the module continues to operate. The last value written to OSCTUNE is used for tuning, and the ACTLOCK bit is continually updated for each ACT cycle. When ACTUD is clear (ACTUD = 0), the module updates OSCTUNE register every ACT cycle.

### 12.4.4 ACT Interrupts

When Active Clock Tuning is enabled (ACTEN = 1) and either the ACTLOCK or ACTORS bits change state (e.g., from a Locked to an Unlocked state), the ACT Interrupt Flag (ACTIF) of the PIR registers is set (ACTIF = 1). If the ACT Interrupt Enable (ACTIE) bit is set (ACTIE = 1), an interrupt will be generated when ACTIF becomes set. No interrupts are generated for each OSCTUNE update unless the update results in a change of Lock status or Out-of-Range status.

## 12.5 Register Definitions: Oscillator Module

### 12.5.1 ACTCON

**Name:** ACTCON
**Address:** 0x0AC

Active Clock Tuning Control Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | ACTEN | ACTUD | | | ACTLOCK | | ACTORS | |
| Access | R/W | R/W | | | R | | R | |
| Reset | 0 | 0 | | | 0 | | 0 | |

**Bit 7 – ACTEN** Active Clock Tuning Enable

| Value | Description |
|---|---|
| 1 | ACT enabled: HFINTOSC tuning is controlled by the ACT |
| 0 | ACT disabled: HFINTOSC tuning is controlled by the OSCTUNE register via user software |

**Bit 6 – ACTUD** Active Clock Tuning Update Disable

| Value | Condition | Description |
|---|---|---|
| 1 | ACTEN = 1 | Updates to the OSCTUNE register from ACT hardware are disabled |
| 0 | ACTEN = 1 | Updates to the OSCTUNE register from ACT hardware are allowed |
| 1 | ACTEN = 0 | Updates to the OSCTUNE register through user software are disabled |
| 0 | ACTEN = 0 | Updates to the OSCTUNE register through user software are allowed |

**Bit 3 – ACTLOCK** Active Clock Tuning Lock Status

| Value | Description |
|---|---|
| 1 | Locked: HFINTOSC is within ± 1% of its nominal value |
| 0 | Not locked: HFINTOSC may or may not be within ± 1% of its nominal value |

**Bit 1 – ACTORS** Active Clock Tuning Out-of-Range Status

| Value | Description |
|---|---|
| 1 | Value required for tuning is outside of the OSCTUNE range |
| 0 | Value required for tuning is within the OSCTUNE range |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

### 12.5.2 OSCCON1

**Name:** OSCCON1
**Address:** 0x0AD

Oscillator Control Register 1

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | NOSC[2:0] | | | NDIV[3:0] | | | |
| Access | | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | | f | f | f | q | q | q | q |

**Bits 6:4 – NOSC[2:0]** New Oscillator Source Request(1,2,3)
Requests a new oscillator source per the NOSC/COSC Clock Source Selection Table.

**Bits 3:0 – NDIV[3:0]** New Divider Selection Request
Requests the new postscaler division ratio per the NDIV/CDIV Clock Divider Selection Table.

**Notes:**

1. The default value is determined by the RSTOSC Configuration bits. See the Reset Oscillator (RSTOSC) selection table for the RSTOSC selections.
2. If NOSC is written with a reserved value, the operation is ignored and neither NOSC nor NDIV is written.
3. When CSWEN = 0, these bits are read-only and cannot be changed from the RSTOSC value.

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

### 12.5.3 OSCCON2

**Name:** OSCCON2
**Address:** 0x0AE

Oscillator Control Register 2

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | COSC[2:0] | | | CDIV[3:0] | | | |
| Access | | R | R | R | R | R | R | R |
| Reset | | f | f | f | f | f | f | f |

**Bits 6:4 – COSC[2:0]** Current Oscillator Source Select (read-only)(1)
Indicates the current oscillator source per the NOSC/COSC Clock Source Selection Table.

**Bits 3:0 – CDIV[3:0]** Current Divider Select (read-only)
Indicates the current postscaler divider ratio per the NDIV/CDIV Clock Divider Table.

**Note:**

1. The RSTOSC value is the value present when user code execution begins. Refer to the RSTOSC Configuration bits or the RSTOSC selection table for the Reset Oscillator selections.

### 12.5.4 OSCCON3

**Name:** OSCCON3
**Address:** 0x0AF

Oscillator Control Register 3

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | CSWHOLD | SOSCPWR | | ORDY | NOSCR | | | |
| Access | R/W/HC | R/W | | R | R | | | |
| Reset | 0 | 1 | | 0 | 0 | | | |

**Bit 7 – CSWHOLD** Clock Switch Hold Control

| Value | Description |
|---|---|
| 1 | Clock switch (and interrupt) will hold when the oscillator selected by NOSC is ready |
| 0 | Clock switch will proceed when the oscillator selected by NOSC is ready |

**Bit 6 – SOSCPWR** Secondary Oscillator Power Mode Select

| Value | Description |
|---|---|
| 1 | Secondary Oscillator operates in High Power mode |
| 0 | Secondary Oscillator operates in Low Power mode |

**Bit 4 – ORDY** Oscillator Ready (read-only)

| Value | Description |
|---|---|
| 1 | OSCCON1 = OSCCON2; the current system clock is the clock specified by NOSC |
| 0 | A clock switch is in progress |

**Bit 3 – NOSCR** New Oscillator is Ready (read-only)(1)

| Value | Description |
|---|---|
| 1 | A clock switch is in progress and the oscillator selected by NOSC indicates a Ready condition |
| 0 | A clock switch is not in progress, or the NOSC-selected oscillator is not ready |

**Note:**

1. If CSWHOLD = 0, the user may not see this bit set (NOSCR = 1). When the oscillator becomes ready, there may be a delay of one instruction cycle before NOSCR is set. The clock switch occurs in the next instruction cycle and NOSCR is cleared.

### 12.5.5 OSCTUNE

**Name:** OSCTUNE
**Address:** 0x0B0

HFINTOSC Frequency Tuning Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | TUN[5:0] | | | | | |
| Access | | | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | | | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 5:0 – TUN[5:0]** HFINTOSC Frequency Tuning

| TUN | Condition |
|---|---|
| 01 1111 | Maximum frequency |
| • | • |
| • | • |
| • | • |
| 00 0000 | Center frequency. Oscillator is operating at the selected nominal frequency. (Default value) |
| • | • |
| • | • |
| • | • |
| 10 0000 | Minimum frequency |

### 12.5.6 OSCFRQ

**Name:** OSCFRQ
**Address:** 0x0B1

HFINTOSC Frequency Selection Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | FRQ[3:0] | | | |
| Access | | | | | R/W | R/W | R/W | R/W |
| Reset | | | | | 0 | 0 | 0 | 0 |

**Bits 3:0 – FRQ[3:0]** HFINTOSC Frequency Selection

| FRQ | Nominal Freq (MHz) |
|---|---|
| 1111-1001 | Reserved |
| 1000 | 64 |
| 0111 | 48 |
| 0110 | 32 |
| 0101 | 16 |
| 0100 | 12 |
| 0011 | 8 |
| 0010 | 4 |
| 0001 | 2 |
| 0000 | 1 |

### 12.5.7 OSCSTAT

**Name:** OSCSTAT
**Address:** 0x0B2

Oscillator Status Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | EXTOR | HFOR | MFOR | LFOR | SOR | ADOR | | PLLR |
| Access | R | R | R | R | R | R | | R |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | | 0 |

**Bit 7 – EXTOR** External Oscillator Ready

| Value | Description |
|---|---|
| 1 | The External oscillator is ready for use |
| 0 | The External oscillator is not enabled, or is not ready for use |

**Bit 6 – HFOR** HFINTOSC Ready

| Value | Description |
|---|---|
| 1 | The HFINTOSC is ready for use |
| 0 | The HFINTOSC is not enabled, or it is not ready for use |

**Bit 5 – MFOR** MFINTOSC Ready

| Value | Description |
|---|---|
| 1 | The MFINTOSC is ready for use |
| 0 | The MFINTOSC is not enabled, or it is not ready for use |

**Bit 4 – LFOR** LFINTOSC Ready

| Value | Description |
|---|---|
| 1 | The LFINTOSC is ready for use |
| 0 | The LFINTOSC is not enabled, or is not ready for use |

**Bit 3 – SOR** Secondary Oscillator (SOSC) Ready

| Value | Description |
|---|---|
| 1 | The Secondary oscillator is ready for use |
| 0 | The Secondary oscillator is not enabled, or is not ready for use |

**Bit 2 – ADOR** ADCRC Oscillator Ready

| Value | Description |
|---|---|
| 1 | The ADCRC oscillator is ready for use |
| 0 | The ADCRC oscillator is not enabled, or is not ready for use |

**Bit 0 – PLLR** PLL is Ready

| Value | Description |
|---|---|
| 1 | The PLL is ready for use |
| 0 | The PLL is not enabled, the required input source is not ready, or the PLL is not locked |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

### 12.5.8 OSCEN

**Name:** OSCEN
**Address:** 0x0B3

Oscillator Enable Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | EXTOEN | HFOEN | MFOEN | LFOEN | SOSCEN | ADOEN | | PLLEN |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | | 0 |

**Bit 7 – EXTOEN** External Oscillator Enable

| Value | Description |
|---|---|
| 1 | EXTOSC is explicitly enabled, operating as specified by FEXTOSC |
| 0 | EXTOSC can be enabled by a peripheral request |

**Bit 6 – HFOEN** HFINTOSC Enable

| Value | Description |
|---|---|
| 1 | HFINTOSC is explicitly enabled, operating as specified by OSCFRQ |
| 0 | HFINTOSC can be enabled by a peripheral request |

**Bit 5 – MFOEN** MFINTOSC Enable

| Value | Description |
|---|---|
| 1 | MFINTOSC is explicitly enabled |
| 0 | MFINTOSC can be enabled by a peripheral request |

**Bit 4 – LFOEN** LFINTOSC Enable

| Value | Description |
|---|---|
| 1 | LFINTOSC is explicitly enabled |
| 0 | LFINTOSC can be enabled by a peripheral request |

**Bit 3 – SOSCEN** Secondary Oscillator Enable

| Value | Description |
|---|---|
| 1 | SOSC is explicitly enabled, operating as specified by SOSCPWR |
| 0 | SOSC can be enabled by a peripheral request |

**Bit 2 – ADOEN** ADCRC Oscillator Enable

| Value | Description |
|---|---|
| 1 | ADCRC is explicitly enabled |
| 0 | ADCRC may be enabled by a peripheral request |

**Bit 0 – PLLEN** PLL Enable(1)

| Value | Description |
|---|---|
| 1 | EXTOSC multiplied by the 4x system PLL is used by a peripheral request |
| 0 | EXTOSC is used by a peripheral request |

**Note:**

1. This bit only controls external clock source supplied to the peripherals and has no effect on the system clock.

## 12.6 Register Summary - Oscillator Module

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0xAC | ACTCON | 7:0 | ACTEN | ACTUD | | | ACTLOCK | | ACTORS | |
| 0xAD | OSCCON1 | 7:0 | | NOSC[2:0] | | | NDIV[3:0] | | | |
| 0xAE | OSCCON2 | 7:0 | | COSC[2:0] | | | CDIV[3:0] | | | |
| 0xAF | OSCCON3 | 7:0 | CSWHOLD | SOSCPWR | | ORDY | NOSCR | | | |
| 0xB0 | OSCTUNE | 7:0 | | | TUN[5:0] | | | | | |
| 0xB1 | OSCFRQ | 7:0 | | | | | FRQ[3:0] | | | |
| 0xB2 | OSCSTAT | 7:0 | EXTOR | HFOR | MFOR | LFOR | SOR | ADOR | | PLLR |
| 0xB3 | OSCEN | 7:0 | EXTOEN | HFOEN | MFOEN | LFOEN | SOSCEN | ADOEN | | PLLEN |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

# 13. CRC - Cyclic Redundancy Check Module with Memory Scanner

The Cyclic Redundancy Check (CRC) module provides a software-configurable hardware-implemented CRC checksum generator. This module includes the following features:

- Any standard CRC up to 16 bits can be used
- Configurable polynomial
- Any seed value up to 16 bits can be used
- Standard and reversed bit order available
- Augmented zeros can be added automatically or by the user
- Memory scanner for fast CRC calculations on program memory user data
- Software loadable data registers for communication CRCs

## 13.1 CRC Module Overview

The CRC module provides a means for calculating a check value of program memory. The CRC module is coupled with a memory scanner for faster CRC calculations. The memory scanner can automatically provide data to the CRC module. The CRC module can also be operated by directly writing data to SFRs, without using a scanner.

## 13.2 CRC Functional Overview

The CRC module can be used to detect bit errors in the Flash memory using the built-in memory scanner or through user input RAM memory. The CRC module can accept up to a 16-bit polynomial with up to a 16-bit seed value. A CRC calculated check value (or checksum) will then be generated into the CRCACC register for user storage. The CRC module uses an XOR shift register implementation to perform the polynomial division required for the CRC calculation.

## 13.3 CRC Polynomial Implementation

Any polynomial can be used. The polynomial and accumulator sizes are determined by the PLEN bits. For an n-bit accumulator, PLEN = n-1 and the corresponding polynomial is n+1 bits. Therefore, the accumulator can be any size up to 16 bits with a corresponding polynomial up to 17 bits. The MSb and LSb of the polynomial are always '1' which is forced by hardware. Therefore, the LSb of the CRCXORL register is hardwired high and always reads as '1'.

All polynomial bits between the MSb and LSb are specified by the CRCXOR registers. For example, when using CRC16-ANSI, the polynomial is defined as $X^{16}+X^{15}+X^{2}+1$. The $X^{16}$ and $X^{0}$ = 1 terms are the MSb and LSb controlled by hardware. The $X^{15}$ and $X^{2}$ terms are specified by setting the corresponding CRCXOR[15:0] bits with the value of 0x8004. Reading the CRCXOR registers will return 0x8005 because the LSb is hardwired high. Refer to the following example.

**Figure 13-1. CRC Example**

Rev. 10-000206B
12/3/2018

CRC-16-ANSI

$x^{16} + x^{15} + x^2 + 1$ (17 bits)

Standard 16-bit representation = 0x8005

```
CRCXORH = 0b10000000
CRCXORL = 0b0000010- (1)
```

Data Sequence:
0x55, 0x66, 0x77, 0x88

```
DLEN = 0b0111
PLEN = 0b1111
```

Data entered into the CRC:
SHIFTM = 0:

```
01010101 01100110 01110111 10001000
```

SHIFTM = 1:

```
10101010 01100110 11101110 00010001
```

Check Value (ACCM = 1):

SHIFTM = 0: 0x32D6

```
CRCACCH = 0b00110010
CRCACCL = 0b11010110
```

SHIFTM = 1: 0x6BA2

```
CRCACCH = 0b01101011
CRCACCL = 0b10100010
```

Note 1: Bit 0 is unimplemented. The LSb of any CRC polynomial is always '1' and will always be treated as a '1' by the CRC for calculating the CRC check value. This bit will be read in software as a '0'.

## 13.4 CRC Data Sources

Data can be input to the CRC module in two ways:

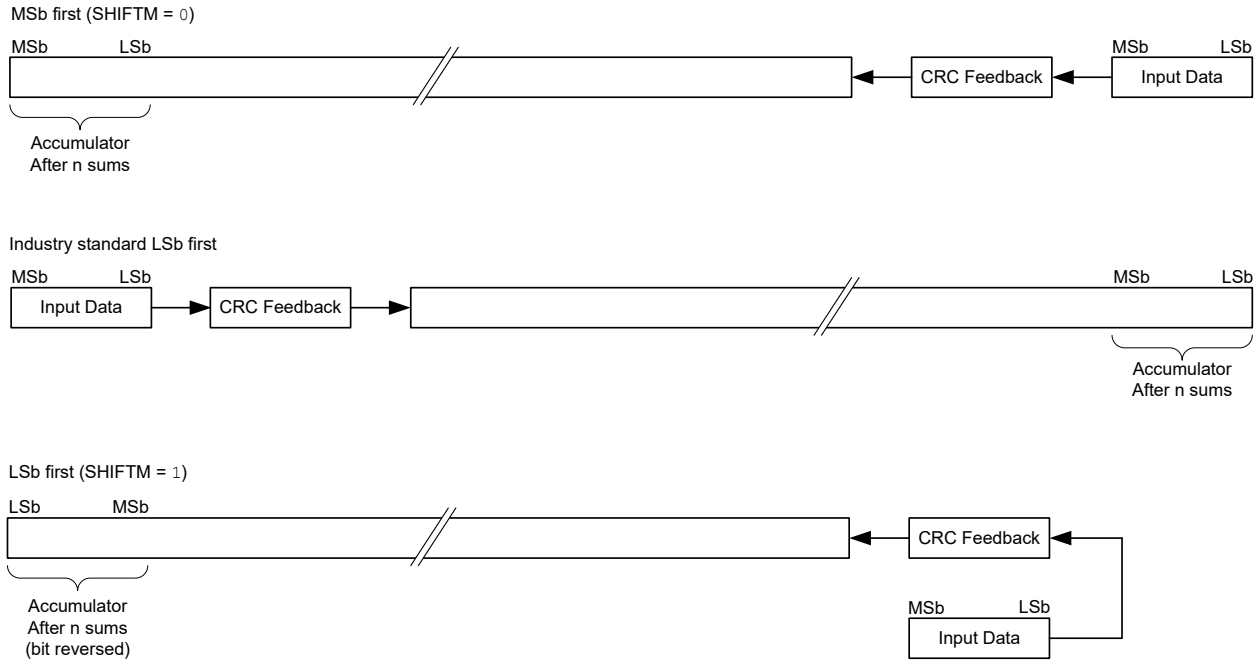
- User data using the CRCDATA registers
- From Flash memory using the program memory scanner

Up to 16 bits of data per word are specified with the DLEN bits. Only the number of data bits in the CRCDATA registers specified by DLEN will be used, other data bits in CRCDATA registers will be ignored.

Data is moved into the CRCSHIFT as an intermediate to calculate the check value located in the CRCACC registers.

The SHIFTM bit is used to determine the bit order of the data being shifted into the accumulator and the bit order of the result. The value of DLEN will determine which bit in CRCDATA is the MSb.

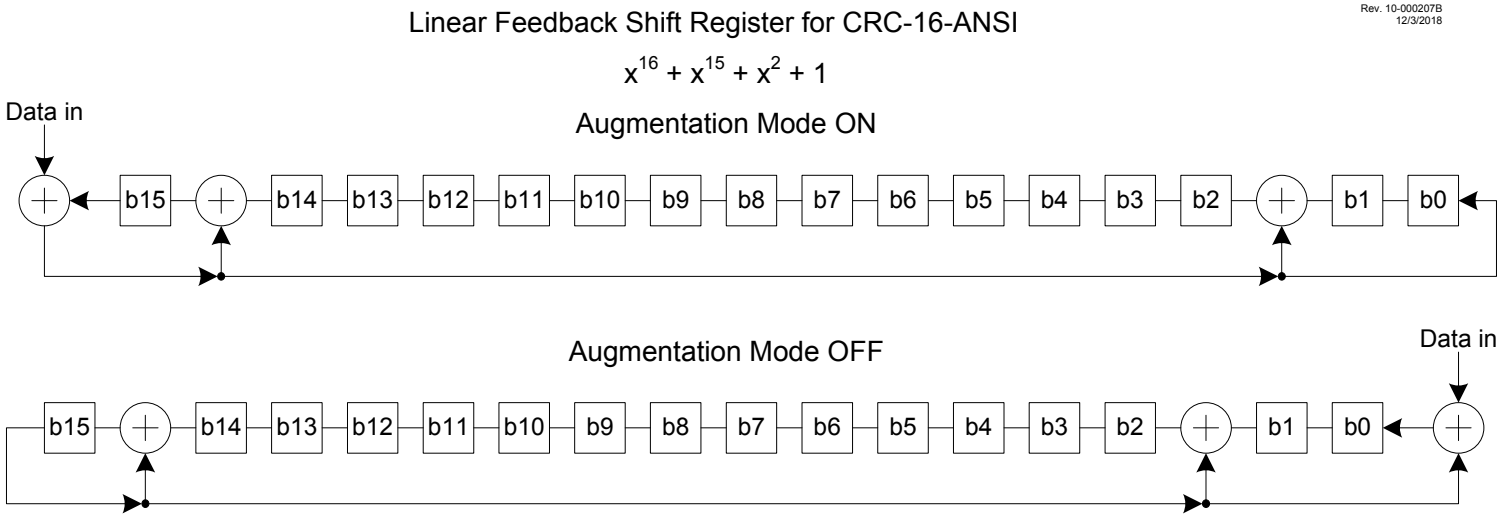
**Figure 13-2. CRC Process**




When SHIFTM is not set, the data will be shifted in MSb first and the result will be big-endian. When SHIFTM bit is set, the data will be shifted into the accumulator in reversed order, LSb first and the result will be little-endian.

The CRC module can be seeded with an initial value by setting the CRCACC registers to the appropriate value before beginning the CRC.

**Figure 13-3. CRC LFSR Example**




### 13.4.1 CRC from User Data

To use the CRC module on data input from the user, the user must write the data to the CRCDATA register. The data from both the CRCDATH and CRCDATL registers will be latched into the shift registers when the CRCDATL register is written.

### 13.4.2 CRC from Flash

To use the CRC module on data located in Flash memory, the user can initialize the program memory scanner as defined in the Scanner Module Overview section.

## 13.5 CRC Check Value

The CRC check value will be located in the CRCACC registers after the CRC calculation has finished. The check value will depend on the ACCM and SHIFTM mode settings.

When the ACCM bit is set, the CRC module augments the data with a number of zeros equal to the length of the polynomial to align the final check value. When the ACCM bit is not set, the CRC will stop at the end of the data. A number of zeros equal to the length of the polynomial can then be entered into CRCDATA to find the same check value as Augmented mode. Alternatively, the expected check value can be entered at this point to make the final result equal zero.

When the CRC check value is computed with the SHIFTM bit set, selecting LSb first, and the ACCM bit is set then the final value in the CRCACC registers will be reversed such that the LSb will be in the MSb position and vice versa (see Figure 13-2). This is the expected check value in bit reversed form. When creating a check value to be appended to a data stream, then a bit reversal must be performed on the final value to achieve the correct checksum. The CRC can be used to do this reversal by following the steps below:

1. Save CRCACC value in user RAM space.
2. Clear the CRCACC registers.
3. Clear the CRCXOR registers.
4. Write the saved CRCACC value to the CRCDATA input.

The properly oriented check value will be in the CRCACC registers as the result.

## 13.6 CRC Interrupt

The CRC will generate an interrupt when the BUSY bit transitions from '1' to '0'. The CRCIF Interrupt Flag bit of one of the PIR registers is set every time the BUSY bit transitions, regardless of whether or not the CRC interrupt is enabled. The CRCIF bit can only be cleared by software. The CRC interrupt enable is the CRCIE bit of the corresponding PIE register.

## 13.7 Configuring the CRC

The following steps illustrate how to properly configure the CRC:

1. Determine if the automatic program memory scan will be used with the scanner or manual calculation through the SFR interface and perform the actions specified in CRC Data Sources, depending on which decision was made.
2. If desired, seed a starting CRC value into the CRCACC registers.
3. Program the CRCXOR registers with the desired generator polynomial.
4. Program the DLEN bits with the length of the data word - 1 (refer to Figure 13-1). This determines how many times the shifter will shift into the accumulator for each data word.
5. Program the PLEN bits with the length of the polynomial - 2 (refer to Figure 13-1).
6. Determine whether shifting in trailing zeros is desired and set the ACCM bit accordingly.
7. Likewise, determine whether the MSb or LSb first shifting is desired, and write the SHIFTM bit accordingly.
8. Set the GO bit to begin the shifting process.
9. If manual SFR entry is used, monitor the FULL bit.
    a. When FULL = 0, another word of data can be written to the CRCDATA registers, keeping in mind that the Most Significant Byte (CRCDATH) must be written first if the data has more than eight bits, as the shifter will begin upon the CRCDATL register being written.
    b. If the scanner is used, the scanner will automatically load words into the CRCDATA registers as needed, as long as the GO bit is set.

10. If using the Flash memory scanner, monitor the PIRx SCANIF bit for the scanner to finish pushing information into the CRCDATA registers.
    a. After the scan is completed, monitor the SGO bit to determine that the CRC has been completed and the check value can be read from the CRCACC registers.
    b. When both the interrupt flags are set (or both BUSY and SCANGO bits are cleared), the completed CRC calculation can be read from the CRCACC registers.
11. If manual entry is used, monitor the BUSY bit to determine when the CRCACC registers hold the valid check value.

## 13.8 Scanner Module Overview

The Scanner allows segments of the Program Flash Memory or Data EEPROM, to be read out (scanned) to the CRC Peripheral. The scanner module interacts with the CRC module and supplies it data one word at a time. Data is fetched from the address range defined by SCANLADR registers up to the SCANHADR registers.

The Scanner begins operation when the SGO bit is set and ends when either SGO is cleared by the user or when SCANLADR increments past SCANHADR. The SGO bit is also cleared by clearing the EN bit in the CRCCON0 register.

## 13.9 Configuring the Scanner

The scanner module may be used in conjunction with the CRC module to perform a CRC calculation over a range of program memory or Data EEPROM addresses. In order to set up the scanner to work with the CRC, perform the following steps:

1. Set up the CRC module (See Configuring the CRC) and enable the scanner module by setting the EN bit in the SCANCON0 register.
2. Choose which memory region the scanner module will operate on and set the MREG bit appropriately.
3. If trigger is used for scanner operation, set the TRIGEN bit and select the trigger source using SCANTRIG register. Select the trigger source using SCANTRIG register and then set the TRIGEN bit.
4. If Burst mode of operation is desired, set the BURSTMD bit.
5. Set the SCANLADR and SCANHADR registers with the beginning and ending locations in memory that are to be scanned.
6. Select the priority level for the scanner module (Refer to the **“System Arbitration”** and the **“Priority Lock”** sections in the **“PIC18 CPU”** chapter for more details.)
7. Both EN and GO bits in the CRCCON0 register must be enabled to use the scanner. Setting the SGO bit will start the scanner operation.

## 13.10 Scanner Interrupt

The scanner will trigger an interrupt when the SCANGO bit transitions from ‘1’ to ‘0’. The SCANIF interrupt flag of one of the PIR registers is set when the last memory location is reached and the data is entered into the CRCDATA registers. The SCANIF bit must be cleared by software. The SCAN interrupt enable is the SCANIE bit of the corresponding PIE register.

## 13.11 Scanning Modes

The interaction of the scanner with the system operation is controlled by the priority selection in the System Arbiter (Refer to the **“Memory Access Scheme”** section for more details.) Additionally, BURSTMD and TRIGEN bits also determine the operation of the Scanner.

### 13.11.1 TRIGEN = 0, BURSTMD = 0

In this case, the memory access request is granted to the scanner if no other higher priority source is requesting access. All sources with lower priority than the scanner will get the memory access cycles that are not utilized by the scanner.

### 13.11.2 TRIGEN = 1, BURSTMD = 0

In this case, the memory access request is generated when the CRC module is ready to accept. The memory access request is granted to the scanner if no other higher priority source is requesting access. All sources with lower priority than the scanner will get the memory access cycles that are not utilized by the scanner.

### 13.11.3 TRIGEN = x, BURSTMD = 1

In this case, the memory access is always requested by the scanner. The memory access request is granted to the scanner if no other higher priority source is requesting access. The memory access cycles will not be granted to lower priority sources than the scanner until it completes operation i.e., SGO = 0.

**Important:** If TRIGEN = 1 and BURSTMD = 1, the user need to ensure that the trigger source is active for the Scanner operation to complete.

### 13.11.4 WWDT interaction

The Windowed Watchdog Timer (WWDT) operates in the background during scanner activity. It is possible that long scans, particularly in Burst mode, may exceed the WWDT time-out period and result in an undesired device Reset. This must be considered when performing memory scans with an application that also utilizes WWDT.

### 13.11.5 Peripheral Module Disable

Both the CRC and scanner module can be disabled individually by setting the CRCMD and SCANMD bits of one of the PMD registers (see the **“PMD - Peripheral Module Disable”** chapter for more details).

## 13.12 Register Definitions: CRC and Scanner Control

Long bit name prefixes for the CRC are shown in the table below. Refer to the **"Long Bit Names"** section in the **"Register and Bit Naming Conventions"** chapter for more information.

**Table 13-1. CRC Long Bit Name Prefixes**

| Peripheral | Bit Name Prefix |
|---|---|
| CRC | CRC |

### 13.12.1 CRCCON0

**Name:** CRCCON0
**Address:** 0x357

CRC Control Register 0

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | EN | GO | BUSY | ACCM | | | SHIFTM | FULL |
| Access | R/W | R/W | R | R/W | | | R/W | R |
| Reset | 0 | 0 | 0 | 0 | | | 0 | 0 |

**Bit 7 – EN** CRC Enable

| Value | Description |
|---|---|
| 1 | CRC module is released from Reset |
| 0 | CRC is disabled and consumes no operating current |

**Bit 6 – GO** CRC Start

| Value | Description |
|---|---|
| 1 | Start CRC serial shifter |
| 0 | CRC serial shifter turned off |

**Bit 5 – BUSY** CRC Busy

| Value | Description |
|---|---|
| 1 | Shifting in progress or pending |
| 0 | All valid bits in shifter have been shifted into accumulator and EMPTY = 1 |

**Bit 4 – ACCM** Accumulator Mode

| Value | Description |
|---|---|
| 1 | Data is augmented with zeros |
| 0 | Data is not augmented with zeros |

**Bit 1 – SHIFTM** Shift Mode

| Value | Description |
|---|---|
| 1 | Shift right (LSb first) |
| 0 | Shift left (MSb first) |

**Bit 0 – FULL** Data Path Full Indicator

| Value | Description |
|---|---|
| 1 | CRCDATH/L registers are full |
| 0 | CRCDATH/L registers have shifted their data into the shifter |

### 13.12.2 CRCCON1

**Name:** CRCCON1
**Address:** 0x358

CRC Control Register 1

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | DLEN[3:0] | | | | PLEN[3:0] | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 7:4 – DLEN[3:0]** Data Length
Denotes the length of the data word -1 (See Figure 13-1)

**Bits 3:0 – PLEN[3:0]** Polynomial Length
Denotes the length of the polynomial -1 (See Figure 13-1)

### 13.12.3 CRCDATA

**Name:** CRCDATA
**Address:** 0x34F

CRC Data Register

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | DATA[15:8] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | x | x | x | x | x | x | x | x |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | DATA[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | x | x | x | x | x | x | x | x |

**Bits 15:0 – DATA[15:0]** CRC Input/Output Data

**Notes:** The individual bytes in this multi-byte register can be accessed with the following register names:

- CRCDATH: Accesses the high byte DATA[15:8]
- CRCDATL: Accesses the low byte DATA[7:0]

### 13.12.4 CRCACC

**Name:** CRCACC
**Address:** 0x351

CRC Accumulator Register

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | ACC[15:8] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | ACC[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 15:0 – ACC[15:0]** CRC Accumulator Data

**Notes:** The individual bytes in this multi-byte register can be accessed with the following register names:

- CRCACCH: Accesses the high byte ACC[15:8]
- CRCACCL: Accesses the low byte ACC[7:0]

### 13.12.5 CRCSHIFT

**Name:** CRCSHFT
**Address:** 0x353

CRC Shift Register

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | SHFT[15:8] | | | | | | | |
| Access | R | R | R | R | R | R | R | R |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | SHFT[7:0] | | | | | | | |
| Access | R | R | R | R | R | R | R | R |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 15:0 – SHFT[15:0]** CRC Shifter Register Data

**Notes:** The individual bytes in this multi-byte register can be accessed with the following register names:

- CRCSHIFTH: Accesses the high byte SHFT[15:8]
- CRCSHIFTL: Accesses the low byte SHFT[7:0]

### 13.12.6 CRCXOR

**Name:** CRCXOR
**Address:** 0x355

CRC XOR Register

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | X[15:8] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | x | x | x | x | x | x | x | x |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | X[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | x | x | x | x | x | x | x | 1 |

**Bits 15:0 – X[15:0]** XOR of Polynomial Term XN Enable

**Important:** Bit 0 is not user accessible and is always set to the value of '1'.

**Notes:** The individual bytes in this multi-byte register can be accessed with the following register names:

- CRCXORH: Accesses the high byte X[15:8]
- CRCXORL: Accesses the low byte X[7:0]

### 13.12.7 SCANCON0

**Name:** SCANCON0
**Address:** 0x360

Scanner Access Control Register 0

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | EN | TRIGEN | SGO | | | MREG | BURSTMD | BUSY |
| Access | R/W | R/W | R/W/HC | | | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | | | 0 | 0 | 0 |

**Bit 7 – EN** Scanner Enable(1)

| Value | Description |
|---|---|
| 1 | Scanner is enabled |
| 0 | Scanner is disabled |

**Bit 6 – TRIGEN** Scanner Trigger Enable(2,5)

| Value | Description |
|---|---|
| 1 | Scanner trigger is enabled |
| 0 | Scanner trigger is disabled |

**Bit 5 – SGO** Scanner GO(3,4)

| Value | Description |
|---|---|
| 1 | When the CRC is ready, the Memory region set by the MREG bit will be accessed and data is passed to the CRC peripheral. |
| 0 | Scanner operations will not occur |

**Bit 2 – MREG** Scanner Memory Region Select(2)

| Value | Description |
|---|---|
| 1 | Scanner address points to Data EEPROM |
| 0 | Scanner address points to Program Flash Memory |

**Bit 1 – BURSTMD** Scanner Burst Mode(5)

| Value | Description |
|---|---|
| 1 | Memory access request to the CPU Arbiter is always true |
| 0 | Memory access request to the CPU Arbiter is dependent on the CRC request and Trigger |

**Bit 0 – BUSY** Scanner Busy Indicator

| Value | Description |
|---|---|
| 1 | Scanner cycle is in process |
| 0 | Scanner cycle is compete (or never started) |

**Notes:**

1. Setting EN = 0 does not affect any other register content.
2. Scanner trigger selection can be set using SCANTRIG register.
3. This bit can be cleared in software. It is cleared in hardware when LADR > HADR (and a data cycle is not occurring) or when CRCGO = 0.
4. CRCEN and CRCGO bits must be set before setting the SGO bit.
5. Refer to Scanning Modes.

### 13.12.8 SCANLADR

**Name:** SCANLADR
**Address:** 0x35A

Scan Low Address Register

| Bit | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|
| | | | SCANLADRU[5:0] | | | | | |
| Access | | | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | | | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | SCANLADRH[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | SCANLADRL[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 21:16 – SCANLADRU[5:0]** Scan Start/Current Address Upper Byte
Upper bits of the current address to be fetched from, value increments on each fetch of memory.

**Bits 15:8 – SCANLADRH[7:0]** Scan Start/Current Address High Byte
High byte of the current address to be fetched from, value increments on each fetch of memory.

**Bits 7:0 – SCANLADRL[7:0]** Scan Start/Current Address Low Byte
Low byte of the current address to be fetched from, value increments on each fetch of memory.

**Notes:**

1. Registers SCANLADRU/H/L form a 22-bit value, but are not guarded for atomic or asynchronous access; registers may only be read or written while SGO = 0.
2. While SGO = 1, writing to this register is ignored.

### 13.12.9 SCANHADR

**Name:** SCANHADR
**Address:** 0x35D

Scan High Address Register

| Bit | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|
| | | | SCANHADRU[5:0] | | | | | |
| Access | | | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | | | 1 | 1 | 1 | 1 | 1 | 1 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | SCANHADRH[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | SCANHADRL[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Bits 21:16 – SCANHADRU[5:0]** Scan End Address
Upper bits of the address at the end of the designated scan

**Bits 15:8 – SCANHADRH[7:0]** Scan End Address
High byte of the address at the end of the designated scan

**Bits 7:0 – SCANHADRL[7:0]** Scan End Address
Low byte of the address at the end of the designated scan

**Notes:**

1. Registers SCANHADRU/H/L form a 22-bit value but are not guarded for atomic or asynchronous access; registers may only be read or written while SGO = 0.
2. While SGO = 1, writing to this register is ignored.

### 13.12.10 SCANTRIG

**Name:** SCANTRIG
**Address:** 0x361

SCAN Trigger Selection Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | TSEL[4:0] | | | | |
| Access | | | | R/W | R/W | R/W | R/W | R/W |
| Reset | | | | 0 | 0 | 0 | 0 | 0 |

**Bits 4:0 – TSEL[4:0]** Scanner Data Trigger Input Selection

**Table 13-2. Scanner Data Trigger Input Sources**

| TSEL Value | Trigger Input Sources |
|---|---|
| 11111-10110 | — |
| 10110 | CLC8_OUT |
| 10101 | CLC7_OUT |
| 10100 | CLC6_OUT |
| 10011 | CLC5_OUT |
| 10010 | CLC4_OUT |
| 10001 | CLC3_OUT |
| 10000 | CLC2_OUT |
| 01111 | CLC1_OUT |
| 01110 | SMT1_OUT |
| 01101-01001 | — |
| 01000 | TMR6_Postscaler_OUT |
| 00111 | TMR5_OUT |
| 00110 | TMR4_Postscaler_OUT |
| 00101 | TMR3_OUT |
| 00100 | TMR2_Postscaler_OUT |
| 00011 | TMR1_OUT |
| 00010 | TMR0_OUT |
| 00001 | CLKREF_OUT |
| 00000 | LFINTOSC |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

## 13.13 Register Summary - CRC

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x034F | CRCDATA | 7:0 | DATA[7:0] | | | | | | | |
| | | 15:8 | DATA[15:8] | | | | | | | |
| 0x0351 | CRCACC | 7:0 | ACC[7:0] | | | | | | | |
| | | 15:8 | ACC[15:8] | | | | | | | |
| 0x0353 | CRCSHFT | 7:0 | SHFT[7:0] | | | | | | | |
| | | 15:8 | SHFT[15:8] | | | | | | | |
| 0x0355 | CRCXOR | 7:0 | X[7:0] | | | | | | | |
| | | 15:8 | X[15:8] | | | | | | | |
| 0x0357 | CRCCON0 | 7:0 | EN | GO | BUSY | ACCM | | | SHIFTM | FULL |
| 0x0358 | CRCCON1 | 7:0 | DLEN[3:0] | | | | PLEN[3:0] | | | |
| 0x0359 | Reserved | | | | | | | | | |
| 0x035A | SCANLADR | 7:0 | SCANLADRL[7:0] | | | | | | | |
| | | 15:8 | SCANLADRH[7:0] | | | | | | | |
| | | 23:16 | | | SCANLADRU[5:0] | | | | | |
| 0x035D | SCANHADR | 7:0 | SCANHADRL[7:0] | | | | | | | |
| | | 15:8 | SCANHADRH[7:0] | | | | | | | |
| | | 23:16 | | | SCANHADRU[5:0] | | | | | |
| 0x0360 | SCANCON0 | 7:0 | EN | TRIGEN | SGO | | | MREG | BURSTMD | BUSY |
| 0x0361 | SCANTRIG | 7:0 | | | | TSEL[4:0] | | | | |

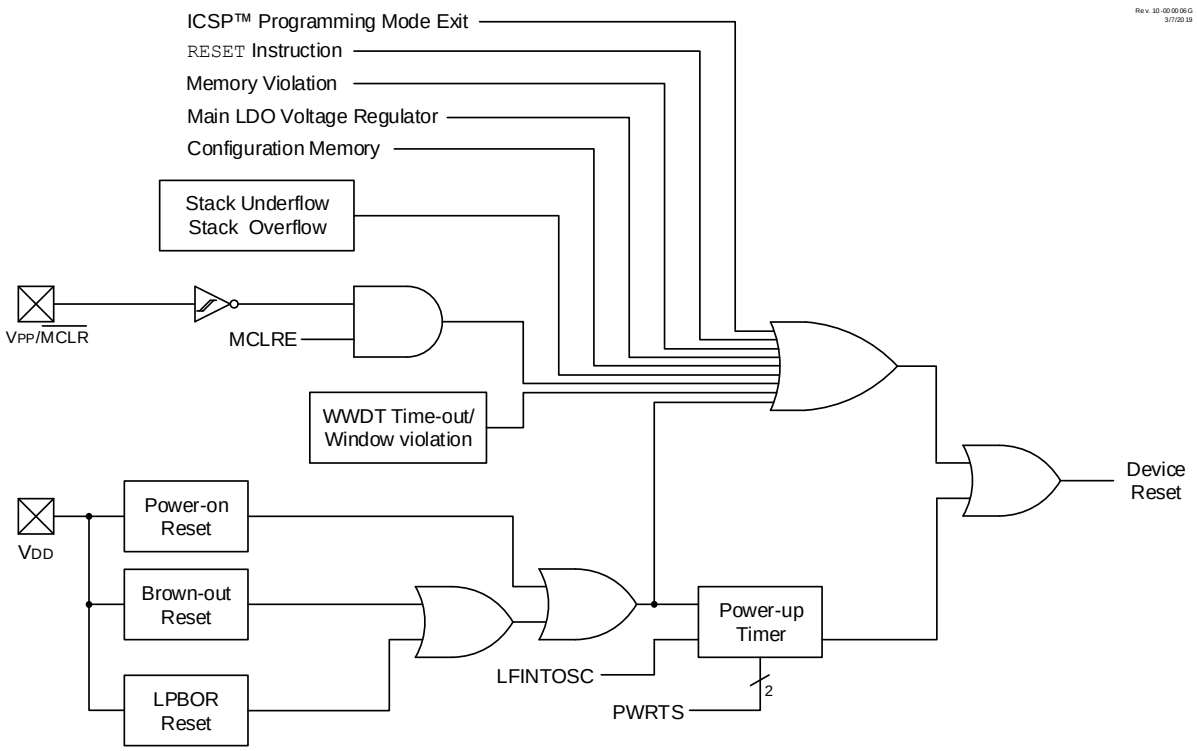
# 14. Resets

There are multiple ways to reset the device:

- Power-on Reset (POR)
- Brown-out Reset (BOR)
- Low-Power Brown-out Reset (LPBOR)
- $\overline{\text{MCLR}}$ Reset
- WDT Reset
- RESET instruction
- Stack Overflow
- Stack Underflow
- Programming mode exit
- Memory Execution Violation Reset
- Main LDO Voltage Regulator Reset
- Configuration Memory Reset

A simplified block diagram of the On-Chip Reset Circuit is shown in the block diagram below.

**Figure 14-1. Simplified Block Diagram of On-Chip Reset Circuit**




**Note:**

1. See the BOR Operating Modes table for BOR active conditions.

## 14.1 Power-on Reset (POR)

The POR circuit holds the device in Reset until $V_{DD}$ has reached an acceptable level for minimum operation. Slow rising $V_{DD}$, fast operating speeds or analog performance may require greater than minimum $V_{DD}$. The PWRT, BOR or $\overline{\text{MCLR}}$ features can be used to extend the start-up period until all device operation conditions have been met. The $\overline{\text{POR}}$ bit will be set to '0' if a Power-on Reset has occurred.

## 14.2 Brown-out Reset (BOR)

The BOR circuit holds the device in Reset when $V_{DD}$ reaches a selectable minimum level. Between the POR and BOR, complete voltage range coverage for execution protection can be implemented. The BOR bit will be set to '0' if a Brown-out Reset has occurred.

The Brown-out Reset module has four operating modes controlled by the BOREN Configuration bits. The four operating modes are:

- BOR is always on
- BOR is off when in Sleep
- BOR is controlled by software
- BOR is always off

Refer to the BOR Operating Modes table for more information.

A $V_{DD}$ noise rejection filter prevents the BOR from triggering on small events. If $V_{DD}$ falls below $V_{BOR}$ for a duration greater than parameter $T_{BORDC}$, the device will reset. Refer to the **"Electrical Specifications"** chapter for more details.

### 14.2.1 BOR Is Always On

When the BOREN Configuration bits are programmed to 'b11, the BOR is always on. The device start-up will be delayed until the BOR is ready and $V_{DD}$ is higher than the BOR threshold.

BOR protection is active during Sleep. The BOR does not delay wake-up from Sleep.

### 14.2.2 BOR Is Off in Sleep

When the BOREN Configuration bits are programmed to 'b10, the BOR is on, except in Sleep. The device start-up will be delayed until the BOR is ready and $V_{DD}$ is higher than the BOR threshold.

BOR protection is not active during Sleep. The device wake-up will be delayed until the BOR is ready.

### 14.2.3 BOR Controlled by Software

When the BOREN Configuration bits are programmed to 'b01, the BOR is controlled by the SBOREN bit. The device start-up is not delayed by the BOR Ready condition or the $V_{DD}$ level.

BOR protection begins as soon as the BOR circuit is ready. The status of the BOR circuit is reflected in the BORRDY bit.

BOR protection selected by SBOREN bit is unchanged by Sleep.

### 14.2.4 BOR Is Always Off

When the BOREN Configuration bits are programmed to 'b00, the BOR is off at all times. The device start-up is not delayed by the BOR Ready condition or the $V_{DD}$ level.

**Table 14-1. Reset Condition for Special Registers**

| Condition | Program Counter | STATUS Register(1,2) | PCON0 Register | PCON1 Register |
|---|---|---|---|---|
| Power-on Reset | 0 | -110 0000 | 0011 110x | ---- -111 |
| Brown-out Reset | 0 | -110 0000 | 0011 11u0 | ---- -u1u |
| $\overline{MCLR}$ Reset during normal operation | 0 | -uuu uuuu | uuuu 0uuu | ---- -uuu |
| $\overline{MCLR}$ Reset during Sleep | 0 | -10u uuuu | uuuu 0uuu | ---- -uuu |
| WDT Time-out Reset | 0 | -0uu uuuu | uuu0 uuuu | ---- -uuu |

.....continued

| Condition | Program Counter | STATUS Register(1,2) | PCON0 Register | PCON1 Register |
|---|---|---|---|---|
| WDT Wake-up from Sleep | PC + 2 | -00u uuuu | uuuu uuuu | ---- -uuu |
| WWDT Window Violation Reset | 0 | -uuu uuuu | uu0u uuuu | ---- -uuu |
| Interrupt Wake-up from Sleep | PC + 2(3) | -10u uuuu | uuuu uuuu | ---- -uuu |
| RESET Instruction Executed | 0 | -uuu uuuu | uuuu u0uu | ---- -uuu |
| Stack Overflow Reset (STVREN = 1) | 0 | -uuu uuuu | 1uuu uuuu | ---- -uuu |
| Stack Underflow Reset (STVREN = 1) | 0 | -uuu uuuu | u1uu uuuu | ---- -uuu |
| Data Protection (Fuse Fault) | 0 | -uuu uuuu | uuuu uuuu | ---- -uu0 |
| VREG or ULP Ready Fault | 0 | -110 0000 | 0011 110u | ---- -0u1 |
| Memory Violation Reset | 0 | -uuu uuuu | uuuu uuuu | ---- -u0u |

**Legend:** u = unchanged, x = unknown, - = unimplemented bit, reads as '0'.

**Notes:**

1. If a Status bit is not implemented, that bit will be read as '0'.
2. Status bits Z, C, DC are reset by POR/BOR.
3. When the wake-up is due to an interrupt and Global Interrupt Enable (GIE) bit is set, the return address is pushed on the stack and PC is loaded with the corresponding interrupt vector (depending on source, high or low priority) after execution of PC + 2.

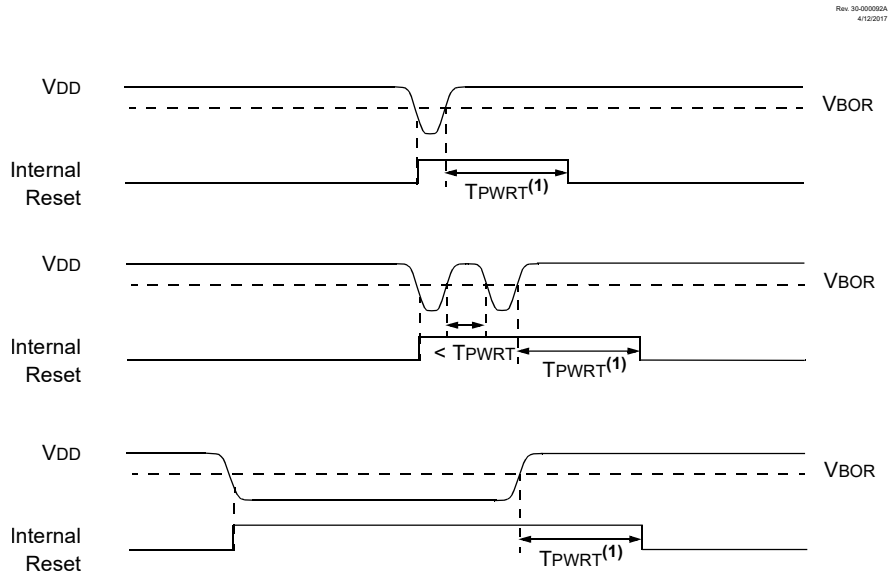
**Table 14-2. BOR Operating Modes**

| BOREN | SBOREN | Device Mode | BOR Mode | Instruction Execution upon: | |
|---|---|---|---|---|---|
| | | | | Release of POR | Wake-up from Sleep |
| 11(1) | X | X | Active | Wait for release of BOR (BORRDY = 1) | Begins immediately |
| 10 | X | Awake | Active | Wait for release of BOR (BORRDY = 1) | N/A |
| | | Sleep | Hibernate | N/A | Wait for release of BOR (BORRDY = 1) |
| 01 | 1 | X | Active | Wait for release of BOR (BORRDY = 1) | Begins immediately |
| | 0 | X | Hibernate | | |
| 00 | X | X | Disabled | Begins immediately | |

**Note:**

1. In this specific case, “Release of POR” and “Wake-up from Sleep”, there is no BOR ready delay in start-up. The BOR ready flag, (BORRDY = 1), will be set before the CPU is ready to execute instructions because the BOR circuit is forced on by the BOREN bits

**Figure 14-2. Brown-Out Situations**




**Note:**

1. $T_{PWRT}$ delay only if the Configuration bits enable the Power-up Timer.

### 14.2.5 BOR and Bulk Erase

BOR is forced ON during PFM Bulk Erase operations to make sure that the system code protection cannot be compromised by reducing $V_{DD}$.

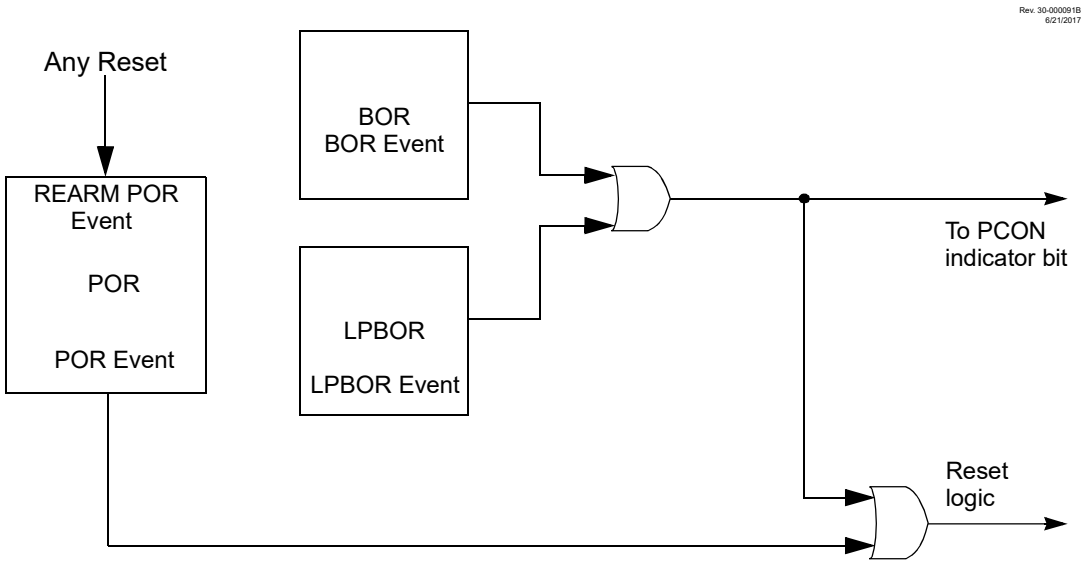
During Bulk Erase, the BOR is enabled at the lowest BOR threshold level, even if it is configured to some other value. If $V_{DD}$ falls, the erase cycle will be aborted, but the device will not be reset.

## 14.3 Low-Power Brown-out Reset (LPBOR)

The Low-Power Brown-out Reset (LPBOR) provides an additional BOR circuit for low-power operation. Refer to the figure below to see how the BOR interacts with other modules.

The LPBOR is used to monitor the external $V_{DD}$ pin. When too low of a voltage is detected, the device is held in Reset.

**Figure 14-3. LPBOR, BOR, POR Relationship**




### 14.3.1 Enabling LPBOR

The LPBOR is controlled by the $\overline{\text{LPBOREN}}$ Configuration bit. When the device is erased, the LPBOR module defaults to disabled.

### 14.3.2 LPBOR Module Output

The output of the LPBOR module indicates whether or not a Reset is to be asserted. This signal is OR'd with the Reset signal of the BOR module to provide the generic $\overline{\text{BOR}}$ signal, which goes to the PCON0 register and to the power control block.

## 14.4 $\overline{\text{MCLR}}$ Reset

$\overline{\text{MCLR}}$ is an optional external input that can reset the device. The $\overline{\text{MCLR}}$ function is controlled by the MCLRE and LVP Configuration bits (see the table below). The $\overline{\text{RMCLR}}$ bit will be set to '0' if a $\overline{\text{MCLR}}$ has occurred.

**Table 14-3. $\overline{\text{MCLR}}$ Configuration**

| MCLRE | LVP | $\overline{\text{MCLR}}$ |
|---|---|---|
| x | 1 | Enabled |
| 1 | 0 | Enabled |
| 0 | 0 | Disabled |

### 14.4.1 $\overline{\text{MCLR}}$ Enabled

When $\overline{\text{MCLR}}$ is enabled and the pin is held low, the device is held in Reset. The $\overline{\text{MCLR}}$ pin is connected to $V_{DD}$ through an internal weak pull-up.

The device has a noise filter in the $\overline{\text{MCLR}}$ Reset path. The filter will detect and ignore small pulses.

**Important:** An internal Reset event (RESET instruction, BOR, WWDT, POR, STKOVF, STKUNF) does not drive the $\overline{\text{MCLR}}$ pin low.

### 14.4.2 MCLR Disabled

When MCLR is disabled, the MCLR pin becomes input-only and pin functions such as internal weak pull-ups are under software control.

## 14.5 Windowed Watchdog Timer (WWDT) Reset

The Windowed Watchdog Timer generates a Reset if the firmware does not issue a CLRWDT instruction within the time-out period or window set. The TO and PD bits in the STATUS register and the RWDT bit are changed to indicate a WDT Reset. The WDTWV bit indicates if the WDT Reset has occurred due to a time-out or a window violation.

## 14.6 RESET Instruction

A RESET instruction will cause a device Reset. The RI bit will be set to '0'. See 14.11. Determining the Cause of a Reset for default conditions after a RESET instruction has occurred.

## 14.7 Stack Overflow/Underflow Reset

The device can be reset when the Stack Overflows or Underflows. The STKOVF or STKUNF bits indicate the Reset condition. These Resets are enabled by setting the STVREN Configuration bit.

## 14.8 Programming Mode Exit

Upon exit of Programming mode, the device will operate as if a POR had just occurred.

## 14.9 Power-up Timer (PWRT)

The Power-up Timer provides a selected time-out duration on POR or Brown-out Reset.

The device is held in Reset as long as PWRT is active. The PWRT delay allows additional time for $V_{DD}$ to rise to an acceptable level. The Power-up Timer is selected by setting the PWRTS Configuration bits accordingly.

The Power-up Timer starts after the release of the POR and BOR/LPBOR if enabled, as shown in Figure 14-4.

## 14.10 Start-Up Sequence

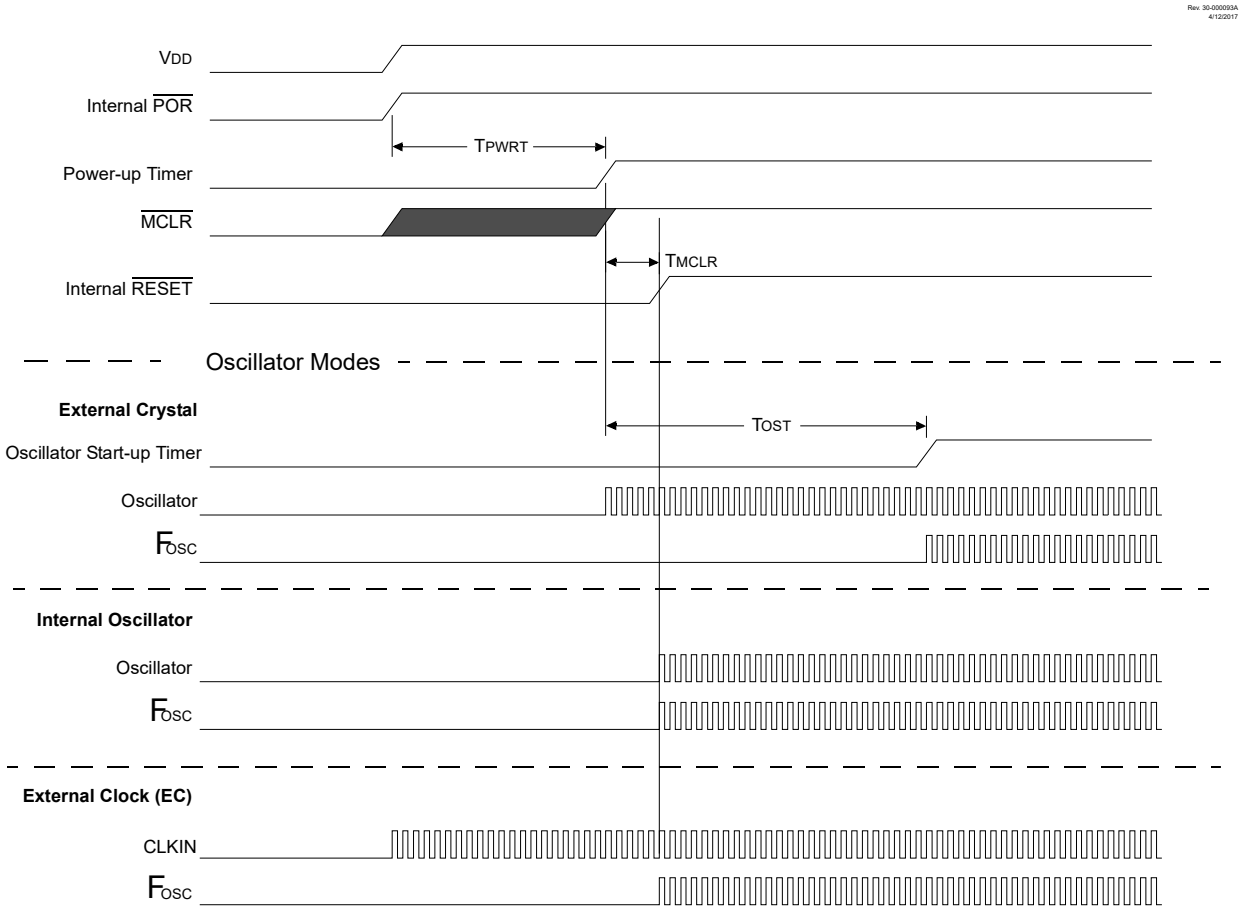
Upon the release of a POR or BOR, the following must occur before the device will begin executing:

1. Power-up Timer runs to completion (if enabled).
2. Oscillator Start-up Timer runs to completion (if required for selected oscillator source).
3. MCLR must be released (if enabled).

The total time-out will vary based on the oscillator configuration and Power-up Timer configuration.

The Power-up Timer and Oscillator Start-up Timer run independently of MCLR Reset. If MCLR is kept low long enough, the Power-up Timer and Oscillator Start-up Timer will expire. Upon bringing MCLR high, the device will begin execution after 10 $F_{OSC}$ cycles (see the figure below). This is useful for testing purposes or to synchronize more than one device operating in parallel.

**Figure 14-4. Reset Start-Up Sequence**




### 14.10.1 Memory Execution Violation

A memory execution violation Reset occurs if executing an instruction being fetched from outside the valid execution area. The invalid execution areas are:

1. Addresses outside implemented program memory.
2. Storage Area Flash (SAF) inside program memory, if it is enabled.

When a memory execution violation is generated, the device is reset and the $\overline{\text{MEMV}}$ bit is cleared to signal the cause of the Reset. The $\overline{\text{MEMV}}$ bit must be set in the user code after a memory execution violation Reset has occurred to detect further violation Resets.

## 14.11 Determining the Cause of a Reset

Upon any Reset, multiple bits in the STATUS, PCON0 and PCON1 registers are updated to indicate the cause of the Reset. The following table shows the Reset conditions of these registers.

**Table 14-4. Reset Condition for Special Registers**

| Condition | Program Counter | STATUS Register(1,2) | PCON0 Register | PCON1 Register |
|---|---|---|---|---|
| Power-on Reset | 0 | -110 0000 | 0011 110x | ---- -111 |
| Brown-out Reset | 0 | -110 0000 | 0011 11u0 | ---- -u1u |

.....continued

| Condition | Program Counter | STATUS Register(1,2) | PCON0 Register | PCON1 Register |
|---|---|---|---|---|
| MCLR Reset during normal operation | 0 | -uuu uuuu | uuuu 0uuu | ---- -uuu |
| MCLR Reset during Sleep | 0 | -10u uuuu | uuuu 0uuu | ---- -uuu |
| WDT Time-out Reset | 0 | -0uu uuuu | uuu0 uuuu | ---- -uuu |
| WDT Wake-up from Sleep | PC + 2 | -00u uuuu | uuuu uuuu | ---- -uuu |
| WWDT Window Violation Reset | 0 | -uuu uuuu | uu0u uuuu | ---- -uuu |
| Interrupt Wake-up from Sleep | PC + 2(3) | -10u uuuu | uuuu uuuu | ---- -uuu |
| RESET Instruction Executed | 0 | -uuu uuuu | uuuu u0uu | ---- -uuu |
| Stack Overflow Reset (STVREN = 1) | 0 | -uuu uuuu | 1uuu uuuu | ---- -uuu |
| Stack Underflow Reset (STVREN = 1) | 0 | -uuu uuuu | u1uu uuuu | ---- -uuu |
| Data Protection (Fuse Fault) | 0 | -uuu uuuu | uuuu uuuu | ---- -uu0 |
| VREG or ULP Ready Fault | 0 | -110 0000 | 0011 110u | ---- -0u1 |
| Memory Violation Reset | 0 | -uuu uuuu | uuuu uuuu | ---- -u0u |

**Legend:** u = unchanged, x = unknown, - = unimplemented bit, reads as '0'.

**Notes:**

1. If a Status bit is not implemented, that bit will be read as '0'.
2. Status bits Z, C, DC are reset by POR/BOR.
3. When the wake-up is due to an interrupt and Global Interrupt Enable (GIE) bit is set, the return address is pushed on the stack and PC is loaded with the corresponding interrupt vector (depending on source, high or low priority) after execution of PC + 2.

## 14.12 Power Control (PCON0/PCON1) Registers

The Power Control (PCON0/PCON1) registers contain flag bits to differentiate between the following Reset events:

- Brown-out Reset (BOR)
- Power-on Reset (POR)
- Reset Instruction Reset (RI)
- MCLR Reset (RMCLR)
- Watchdog Timer Reset (RWDT)
- Watchdog Window Violation (WDTWV)
- Stack Underflow Reset (STKUNF)
- Stack Overflow Reset (STKOVF)
- Configuration Memory Reset (RCM)

- Memory Violation Reset ($\overline{\text{MEMV}}$)
- Main LDO Voltage Regulator Reset ($\overline{\text{RVREG}}$)

Hardware will change the corresponding register bit or bits as a result of the Reset event. Bits for other Reset events remain unchanged. See 14.11. Determining the Cause of a Reset for more details.

Software will reset the bit to the Inactive state after restart (hardware will not reset the bit).

Software may also set any PCON0 bit to the Active state, so that user code may be tested, but no Reset action will be generated.

## 14.13 Register Definitions: Power Control

### 14.13.1 BORCON

**Name:** BORCON
**Address:** 0x049

Brown-out Reset Control Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | SBOREN | | | | | | | BORRDY |
| Access | R/W | | | | | | | R |
| Reset | 1 | | | | | | | q |

**Bit 7 – SBOREN** Software Brown-out Reset Enable
Reset States: POR/BOR = 1
All Other Resets = u

| Value | Condition | Description |
|---|---|---|
| – | If BOREN ≠ 01 | SBOREN is read/write, but has no effect on the BOR |
| 1 | If BOREN = 01 | BOR Enabled |
| 0 | If BOREN = 01 | BOR Disabled |

**Bit 0 – BORRDY** Brown-out Reset Circuit Ready Status
Reset States: POR/BOR = q
All Other Resets = u

| Value | Description |
|---|---|
| 1 | The Brown-out Reset Circuit is active and armed |
| 0 | The Brown-out Reset Circuit is disabled or is warming up |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

### 14.13.2 PCON0

**Name:** PCON0
**Address:** 0x4F0

Power Control Register 0

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | STKOVF | STKUNF | $\overline{\text{WDTWV}}$ | $\overline{\text{RWDT}}$ | $\overline{\text{RMCLR}}$ | $\overline{\text{RI}}$ | $\overline{\text{POR}}$ | $\overline{\text{BOR}}$ |
| Access | R/W/HS | R/W/HS | R/W/HC | R/W/HC | R/W/HC | R/W/HC | R/W/HC | R/W/HC |
| Reset | 0 | 0 | 1 | 1 | 1 | 1 | 0 | q |

**Bit 7 – STKOVF** Stack Overflow Flag
Reset States: POR/BOR = 0
All Other Resets = q

| Value | Description |
|---|---|
| 1 | A Stack Overflow occurred (more CALLs than fit on the stack) |
| 0 | A Stack Overflow has not occurred or set to '0' by firmware |

**Bit 6 – STKUNF** Stack Underflow Flag
Reset States: POR/BOR = 0
All Other Resets = q

| Value | Description |
|---|---|
| 1 | A Stack Underflow occurred (more RETURNs than CALLs) |
| 0 | A Stack Underflow has not occurred or set to '0' by firmware |

**Bit 5 – $\overline{\text{WDTWV}}$** Watchdog Window Violation Flag
Reset States: POR/BOR = 1
All Other Resets = q

| Value | Description |
|---|---|
| 1 | A WDT window violation has not occurred or set to '1' by firmware |
| 0 | A CLRWDT instruction was issued when the WDT Reset window was closed (set to '0' in hardware when a WDT window violation Reset occurs) |

**Bit 4 – $\overline{\text{RWDT}}$** WDT Reset Flag
Reset States: POR/BOR = 1
All Other Resets = q

| Value | Description |
|---|---|
| 1 | A WDT overflow/Time-out Reset has not occurred or set to '1' by firmware |
| 0 | A WDT overflow/Time-out Reset has occurred (set to '0' in hardware when a WDT Reset occurs) |

**Bit 3 – $\overline{\text{RMCLR}}$** $\overline{\text{MCLR}}$ Reset Flag
Reset States: POR/BOR = 1
All Other Resets = q

| Value | Description |
|---|---|
| 1 | A $\overline{\text{MCLR}}$ Reset has not occurred or set to '1' by firmware |
| 0 | A $\overline{\text{MCLR}}$ Reset has occurred (set to '0' in hardware when a $\overline{\text{MCLR}}$ Reset occurs) |

**Bit 2 – $\overline{\text{RI}}$** RESET Instruction Flag
Reset States: POR/BOR = 1
All Other Resets = q

| Value | Description |
|---|---|
| 1 | A RESET instruction has not been executed or set to '1' by firmware |
| 0 | A RESET instruction has been executed (set to '0' in hardware upon executing a RESET instruction) |

**Bit 1 – $\overline{\text{POR}}$** Power-on Reset Status

Reset States: POR/BOR = 0
All Other Resets = u

| Value | Description |
|---|---|
| 1 | No $V_{DD}$ Power-on Reset occurred or set to '1' by firmware |
| 0 | A $V_{DD}$ Power-on Reset occurred (set to '0' in hardware when a Power-on Reset occurs) |

**Bit 0 – $\overline{\text{BOR}}$** Brown-out Reset Status

Reset States: POR/BOR = q
All Other Resets = u

| Value | Description |
|---|---|
| 1 | No $V_{DD}$ Brown-out Reset occurred or set to '1' by firmware |
| 0 | A $V_{DD}$ Brown-out Reset occurred (set to '0' in hardware when a Brown-out Reset occurs) |

### 14.13.3 PCON1

**Name:** PCON1
**Address:** 0x4F1

Power Control Register 1

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | $\overline{\text{RVREG}}$ | $\overline{\text{MEMV}}$ | $\overline{\text{RCM}}$ |
| Access | | | | | | R/W/HC | R/W/HC | R/W/HC |
| Reset | | | | | | 1 | 0 | q |

**Bit 2 – $\overline{\text{RVREG}}$** Main LDO Voltage Regulator Reset Flag
Reset States: POR/BOR = 1
All Other Resets = q

| Value | Description |
|---|---|
| 1 | No LDO or ULP “ready” Reset has occurred or set to ‘1’ by firmware |
| 0 | LDO or ULP “ready” Reset has occurred (VDDCORE reached its minimum spec) |

**Bit 1 – $\overline{\text{MEMV}}$** Memory Violation Reset Flag
Reset States: POR/BOR = 0
All Other Resets = u

| Value | Description |
|---|---|
| 1 | No memory violation Reset occurred or set to ‘1’ by firmware |
| 0 | A memory violation Reset occurred (set to ‘0’ in hardware when a Memory Violation occurs) |

**Bit 0 – $\overline{\text{RCM}}$** Configuration Memory Reset Flag
Reset States: POR/BOR = q
All Other Resets = u

| Value | Description |
|---|---|
| 1 | A Reset occurred due to corruption of the configuration and/or calibration data latches |
| 0 | The configuration and calibration latches have not been corrupted |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

## 14.14 Register Summary - BOR Control and Power Control

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x49 | BORCON | 7:0 | SBOREN | | | | | | | BORRDY |
| 0x4A ... 0x04EF | Reserved | | | | | | | | | |
| 0x04F0 | PCON0 | 7:0 | STKOVF | STKUNF | WDTWV | RWDT | RMCLR | RI | POR | BOR |
| 0x04F1 | PCON1 | 7:0 | | | | | | RVREG | MEMV | RCM |

# 15. WWDT - Windowed Watchdog Timer

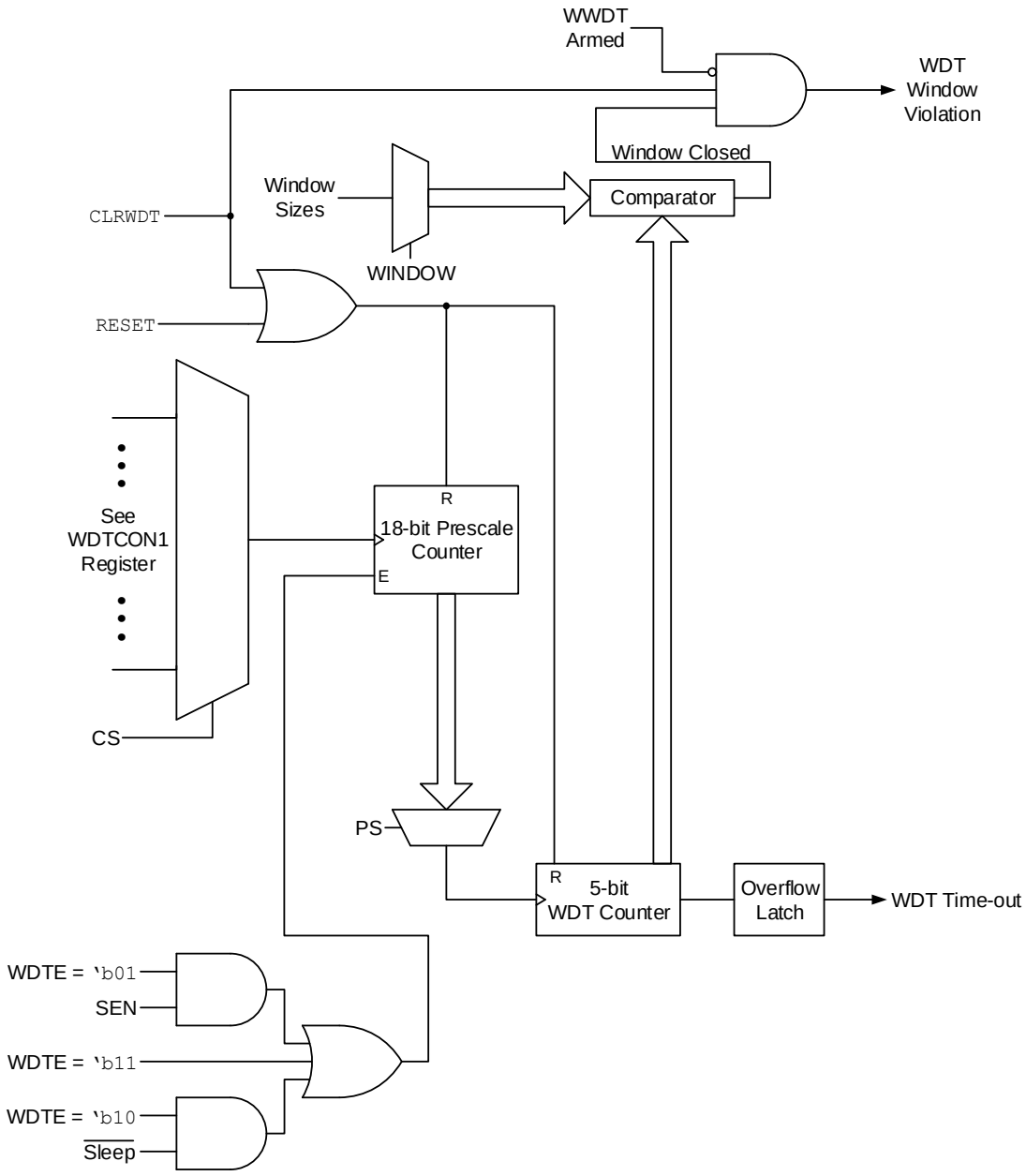
A Watchdog Timer (WDT) is a system timer that generates a Reset if the firmware does not issue a CLRWDT instruction within the time-out period. A Watchdog Timer is typically used to recover the system from unexpected events. The Windowed Watchdog Timer (WWDT) differs from nonwindowed operation in that CLRWDT instructions are only accepted when they are performed within a specific window during the time-out period.

The WWDT has the following features:

- Selectable clock source
- Multiple operating modes
  - WWDT is always on
  - WWDT is off when in Sleep
  - WWDT is controlled by software
  - WWDT is always off
- Configurable time-out period from 1 ms to 256s (nominal)
- Configurable window size from 12.5% to 100% of the time-out period
- Multiple Reset conditions

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

**Figure 15-1. Windowed Watchdog Timer Block Diagram**




## 15.1 Independent Clock Source

The WWDT can derive its time base from either the 31 KHz LFINTOSC or 31.25 kHz MFINTOSC internal oscillators, depending on the value of WDT Operating Mode (WDTE) Configuration bits. If WDTE = 'b1x, then the clock source will be enabled depending on the WDTCCS Configuration bits. If WDTE = 'b01, the SEN bit will be set by software to enable WWDT, and the clock source is enabled by the CS bits. Time intervals in this chapter are based on a minimum nominal interval of 1 ms. See the device Electrical Specifications for LFINTOSC and MFINTOSC tolerances.

## 15.2 WWDT Operating Modes

The Windowed Watchdog Timer module has four operating modes that are controlled by the WDTE Configuration bit. The table below summarizes the different WWDT operating modes.

**Table 15-1. WWDT Operating Modes**

| WDTE | SEN | Device Mode | WWDT Mode |
|---|---|---|---|
| 11 | X | X | Active |
| 10 | X | Awake | Active |
| | | Sleep | Disabled |
| 01 | 1 | X | Active |
| | 0 | X | Disabled |
| 00 | X | X | Disabled |

### 15.2.1 WWDT Is Always On

When the WDTE Configuration bits are set to 'b11, the WWDT is always on. WWDT protection is active during Sleep.

### 15.2.2 WWDT Is Off in Sleep

When the WDTE Configuration bits are set to 'b10, the WWDT is on, except in Sleep mode. WWDT protection is not active during Sleep.

### 15.2.3 WWDT Controlled by Software

When the WDTE Configuration bits are set to 'b01, the WWDT is controlled by the SEN bit. WWDT protection is unchanged by Sleep. See Table 15-1 for more details.

## 15.3 Time-Out Period

When the WDTCPS Configuration bits are set to the default value of 'b11111, the PS bits set the time-out period from 1 ms to 256 seconds (nominal). If any value other than the default value is assigned to the WDTCPS Configuration bits, then the timer period will be based on the WDTCPS Configuration bits. After a Reset, the default time-out period is 2s.

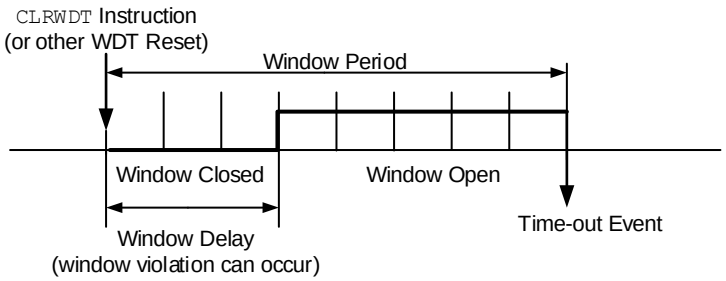
## 15.4 Watchdog Window

The Windowed Watchdog Timer has an optional Windowed mode that is controlled by either the WDTCWS Configuration bits or the WINDOW bits. In the Windowed mode (WINDOW < 'b1111), the CLRWDT instruction must occur within the allowed window of the WDT period. Any CLRWDT instruction that occurs outside of this window will trigger a window violation and will cause a WWDT Reset, similar to a WWDT time-out. See Figure 15-2 for an example.

When the WDTCWS Configuration bits are 'b111, then the window size is controlled by the WINDOW bits, otherwise the window size is controlled by the WDTCWS bits. The five Most Significant bits of the WDTTMR register are used to determine whether the window is open, as defined by the window size. In the event of a window violation, a Reset will be generated and the WDTWV bit of the PCON0 register will be cleared. This bit is set by a POR and can be set by software.

© 2019-2023 Microchip Technology Inc. and its subsidiaries

**Figure 15-2. Window Period and Delay**




## 15.5 Clearing the Watchdog Timer

The Watchdog Timer is cleared when any of the following conditions occur:

- Any Reset
- A valid CLRWDT instruction is executed
- The device enters Sleep
- The devices exits Sleep by Interrupt
- The WWDT is disabled
- The Oscillator Start-up Timer (OST) is running
- Any write to the WDTCON0 or WDTCON1 registers

### 15.5.1 CLRWDT Considerations (Windowed Mode)

When in Windowed mode, the WWDT must be armed before a CLRWDT instruction will clear the timer. This is performed by reading the WDTCON0 register. Executing a CLRWDT instruction without performing such an arming action will trigger a window violation regardless of whether the window is open or not. See Table 15-2 for more information.

## 15.6 Operation During Sleep

When the device enters Sleep, the Watchdog Timer is cleared. If the WWDT is enabled during Sleep, the Watchdog Timer resumes counting. When the device exits Sleep, the Watchdog Timer is cleared again. The Watchdog Timer remains clear until the Oscillator Start-up Timer (OST) completes, if enabled. When a WWDT time-out occurs while the device is in Sleep, no Reset is generated. Instead, the device wakes up and resumes operation. The $\overline{TO}$ and $\overline{PD}$ bits in the STATUS register are changed to indicate the event. The $\overline{RWDT}$ bit in the PCON0 register indicates that a Watchdog Reset has occurred.

**Table 15-2. WWDT Clearing Conditions**

| Conditions | WWDT |
|---|---|
| WDTE = 'b00 | Cleared |
| WDTE = 'b01 and SEN = 0 | |
| WDTE = 'b10 and enter Sleep | |
| CLRWDT Command | |
| Oscillator Fail Detected | |
| Exit Sleep + System Clock = SOSC, EXTRC, INTOSC, EXTCLK | |
| Exit Sleep + System Clock = XT, HS, LP | Cleared until the end of OST |
| Change INTOSC divider (IRCF bits) | Unaffected |

## 15.7 Register Definitions: Windowed Watchdog Timer Control

Long bit name prefixes for the Windowed Watchdog Timer peripherals are shown in the following table. Refer to the "**Long Bit Names**" section in the **"Register and Bit Naming Conventions"** chapter for more information.

**Table 15-3. WDT Long Bit Name Prefixes**

| Peripheral | Bit Name Prefix |
|---|---|
| WDT | WDT |

### 15.7.1 WDTCON0

**Name:** WDTCON0
**Address:** 0x078

Watchdog Timer Control Register 0

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | PS[4:0] | | | | | SEN |
| Access | | | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | | | q | q | q | q | q | 0 |

**Bits 5:1 – PS[4:0]** Watchdog Timer Prescaler Select[2]

| Value | Description |
|---|---|
| 11111 to 10011 | Reserved. Results in minimum interval (1 ms) |
| 10010 | 1:8388608 ($2^{23}$) (Interval 256s nominal) |
| 10001 | 1:4194304 ($2^{22}$) (Interval 128s nominal) |
| 10000 | 1:2097152 ($2^{21}$) (Interval 64s nominal) |
| 01111 | 1:1048576 ($2^{20}$) (Interval 32s nominal) |
| 01110 | 1:524288 ($2^{19}$) (Interval 16s nominal) |
| 01101 | 1:262144 ($2^{18}$) (Interval 8s nominal) |
| 01100 | 1:131072 ($2^{17}$) (Interval 4s nominal) |
| 01011 | 1:65536 (Interval 2s nominal) (Reset value) |
| 01010 | 1:32768 (Interval 1s nominal) |
| 01001 | 1:16384 (Interval 512 ms nominal) |
| 01000 | 1:8192 (Interval 256 ms nominal) |
| 00111 | 1:4096 (Interval 128 ms nominal) |
| 00110 | 1:2048 (Interval 64 ms nominal) |
| 00101 | 1:1024 (Interval 32 ms nominal) |
| 00100 | 1:512 (Interval 16 ms nominal) |
| 00011 | 1:256 (Interval 8 ms nominal) |
| 00010 | 1:128 (Interval 4 ms nominal) |
| 00001 | 1:64 (Interval 2 ms nominal) |
| 00000 | 1:32 (Interval 1 ms nominal) |

**Bit 0 – SEN** Software Enable/Disable for Watchdog Timer

| Value | Condition | Description |
|---|---|---|
| x | If WDTE = 1x | This bit is ignored |
| 1 | If WDTE = 01 | WDT is turned on |
| 0 | If WDTE = 01 | WDT is turned off |
| x | If WDTE = 00 | This bit is ignored |

**Notes:**

1. When the WDTCPS Configuration bits = 'b11111, the Reset value (q) of WDTPS is 'b01011. Otherwise, the Reset value of WDTPS is equal to the WDTCPS in Configuration bits.
2. When the WDTCPS in Configuration bits ≠ 'b11111, these bits are read-only.

© 2019-2023 Microchip Technology Inc. and its subsidiaries

### 15.7.2 WDTCON1

**Name:** WDTCON1
**Address:** 0x079

Watchdog Timer Control Register 1

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | CS[2:0] | | | | WINDOW[2:0] | | |
| Access | | R/W | R/W | R/W | | R/W | R/W | R/W |
| Reset | | q | q | q | | q | q | q |

**Bits 6:4 – CS[2:0]** Watchdog Timer Clock Select[1,3]

| CS | Clock Source |
|---|---|
| 111-100 | Reserved |
| 011 | EXTOSC |
| 010 | SOSC |
| 001 | MFINTOSC (31.25 kHz) |
| 000 | LFINTOSC (31 kHz) |

**Bits 2:0 – WINDOW[2:0]** Watchdog Timer Window Select[2,4]

| WINDOW | Window Delay Percent of Time | Window Opening Percent of Time |
|---|---|---|
| 111 | N/A | 100 |
| 110 | 12.5 | 87.5 |
| 101 | 25 | 75 |
| 100 | 37.5 | 62.5 |
| 011 | 50 | 50 |
| 010 | 62.5 | 37.5 |
| 001 | 75 | 25 |
| 000 | 87.5 | 12.5 |

**Notes:**

1. When the WDTCCS in Configuration bits = 'b111, the Reset value of WDTCS is 'b000.
2. The Reset value (q) of WINDOW is determined by the value of WDTCWS in the Configuration bits.
3. When the WDTCCS in Configuration bits ≠ 'b111, these bits are read-only.
4. When the WDTCWS in Configuration bits ≠ 'b111, these bits are read-only.

© 2019-2023 Microchip Technology Inc. and its subsidiaries

### 15.7.3 WDTPSH

**Name:** WDTPSH
**Address:** 0x07B

WWDT Prescaler Select Register (Read-Only)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | PSCNTH[7:0] | | | | | | | |
| Access | R | R | R | R | R | R | R | R |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 7:0 – PSCNTH[7:0]** Prescaler Select High Byte(1)

**Note:**

1. The 18-bit WDT prescaler value, PSCNT[17:0] includes the WDTPSL, WDTPSH and the lower bits of the WDTTMR registers. PSCNT[17:0] is intended for debug operations and will be read during normal operation.

### 15.7.4 WDTPSL

**Name:** WDTPSL
**Address:** 0x07A

WWDT Prescaler Select Register (Read-Only)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | PSCNTL[7:0] | | | | | | | |
| Access | R | R | R | R | R | R | R | R |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 7:0 – PSCNTL[7:0]** Prescaler Select Low Byte[1]

**Note:**

1. The 18-bit WDT prescaler value, PSCNT[17:0] includes the WDTPSL, WDTPSH and the lower bits of the WDTTMR registers. PSCNT[17:0] is intended for debug operations and will be read during normal operation.

### 15.7.5 WDTTMR

**Name:** WDTTMR
**Address:** 0x07C

WDT Timer Register (Read-Only)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | TMR[4:0] | | | | | STATE | PSCNT[17:16] | |
| Access | R | R | R | R | R | R | R | R |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 7:3 – TMR[4:0]** Watchdog Window Value

| WINDOW | WDT Window State | | Open Percent |
|---|---|---|---|
| | Closed | Open | |
| 111 | N/A | 00000-11111 | 100 |
| 110 | 00000-00011 | 00100-11111 | 87.5 |
| 101 | 00000-00111 | 01000-11111 | 75 |
| 100 | 00000-01011 | 01100-11111 | 62.5 |
| 011 | 00000-01111 | 10000-11111 | 50 |
| 010 | 00000-10011 | 10100-11111 | 37.5 |
| 001 | 00000-10111 | 11000-11111 | 25 |
| 000 | 00000-11011 | 11100-11111 | 12.5 |

**Bit 2 – STATE** WDT Armed Status

| Value | Description |
|---|---|
| 1 | WDT is armed |
| 0 | WDT is not armed |

**Bits 1:0 – PSCNT[17:16]** Prescaler Select Upper Byte[(1)]

**Note:**

1. The 18-bit WDT prescaler value, PSCNT[17:0] includes the WDTPSL, WDTPSH and the lower bits of the WDTTMR registers. PSCNT[17:0] is intended for debug operations and will not be read during normal operation.

© 2019-2023 Microchip Technology Inc. and its subsidiaries

## 15.8 Register Summary - WDT Control

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x78 | WDTCON0 | 7:0 | | | PS[4:0] | | | | | SEN |
| 0x79 | WDTCON1 | 7:0 | | CS[2:0] | | | | WINDOW[2:0] | | |
| 0x7A | WDTPSL | 7:0 | PSCNTL[7:0] | | | | | | | |
| 0x7B | WDTPSH | 7:0 | PSCNTH[7:0] | | | | | | | |
| 0x7C | WDTTMR | 7:0 | TMR[4:0] | | | | | STATE | PSCNT[17:16] | |

# 16. DMA - Direct Memory Access

The Direct Memory Access (DMA) module is designed to service data transfers between different memory regions directly, without intervention from the CPU. By eliminating the need for CPU-intensive management of handling interrupts intended for data transfers, the CPU now can spend more time on other tasks.

The DMA modules can be independently programmed to transfer data between different memory locations, move different data sizes, and use a wide range of hardware triggers to initiate transfers. The DMA modules can even be programmed to work together, to carry out more complex data transfers without CPU overhead.

Key features of the DMA module include:

- Support access to the following memory regions:
  - GPR and SFR space (R/W)
  - Program Flash memory (R only)
  - Data EEPROM memory (R only)
- Programmable priority between the DMA and CPU operations. Refer to the **"System Arbitration"** section in the **"PIC18 CPU"** chapter for details.
- Programmable Source and Destination Address modes:
  - Fixed address
  - Post-increment address
  - Post-decrement address
- Programmable source and destination sizes
- Source and Destination Pointer register, dynamically updated and reloadable
- Source and Destination Count register, dynamically updated and reloadable
- Programmable auto-stop based on source or destination counter
- Software triggered transfers
- Multiple user-selectable sources for hardware triggered transfers
- Multiple user-selectable sources for aborting DMA transfers

## 16.1 DMA Registers

The operation of the DMA module is controlled by the following registers:

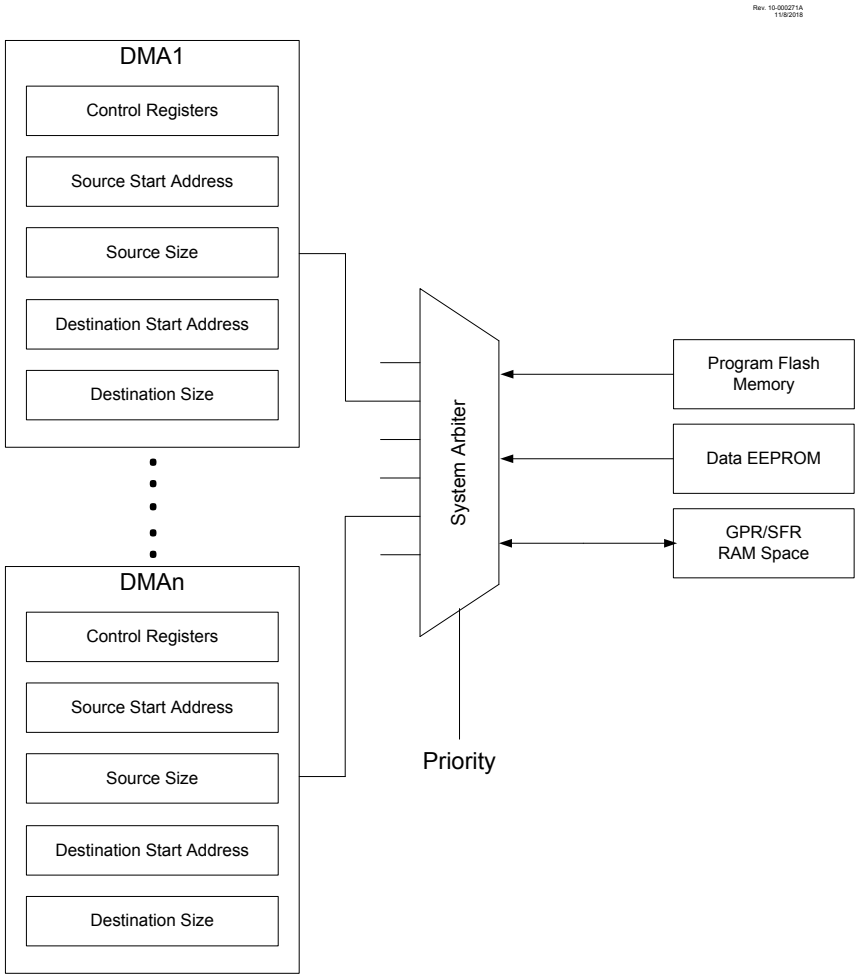
- DMA Instance Selection (DMASELECT) register
- Control (DMAnCON0, DMAnCON1) registers
- Data Buffer (DMAnBUF) register
- Source Start Address (DMAnSSA) register
- Source Pointer (DMAnSPTR) register
- Source Message Size (DMAnSSZ) register
- Source Count (DMAnSCNT) register
- Destination Start Address (DMAnDSA) register
- Destination Pointer (DMAnDPTR) register
- Destination Message Size (DMAnDSZ) register
- Destination Count (DMAnDCNT) register
- Start Interrupt Request Source (DMAnSIRQ) register
- Abort Interrupt Request Source (DMAnAIRQ) register

The registers are detailed in Register Definitions: DMA.

## 16.2 DMA Organization

The DMA module is designed to move data by using the existing instruction bus and data bus without the need for any dual-porting of memory or peripheral systems (Figure 16-1). The DMA accesses the required bus when granted by the system arbiter.

**Figure 16-1. DMA Functional Block Diagram**




Depending on the priority of the DMA with respect to CPU execution (refer to the **"Memory Access Scheme"** section in the **"PIC18 CPU"** chapter for more information), the DMA Controller can move data through two methods:

- Stalling the CPU execution until it has completed its transfers (DMA has higher priority over the CPU in this mode of operation)
- Utilizing unused CPU cycles for DMA transfers (CPU has higher priority over the DMA in this mode of operation). Unused CPU cycles are referred to as bubbles, which are instruction cycles available for use by the DMA to perform read and write operations. In this way, the effective bandwidth for handling data is increased; at the same time, DMA operations can proceed without causing a processor stall.

## 16.3 DMA Interface

The DMA module transfers data from the source to the destination one byte at a time, this smallest data movement is called a DMA data transaction. A DMA message refers to one or more DMA data transactions.

Each DMA data transaction consists of two separate actions:

- Reading the source address memory and storing the value in the DMA Buffer register
- Writing the contents of the DMA Buffer register to the destination address memory

**Important:** DMA data movement is a two-cycle operation.

The XIP bit is a Status bit to indicate whether or not the data in the DMAnBUF register has been written to the destination address. If the bit is set, then data is waiting to be written to the destination. If clear, it means that either data has been written to the destination or that no source read has occurred.

The DMA has read access to PFM, Data EEPROM, and SFR/GPR space, and write access to SFR/GPR space. Based on these memory access capabilities, the DMA can support the following memory transactions:

**Table 16-1. DMA Memory Access**

| Read Source | Write Destination |
|---|---|
| Program Flash Memory | GPR |
| Program Flash Memory | SFR |
| Data EE | GPR |
| Data EE | SFR |
| GPR | GPR |
| GPR | SFR |
| SFR | GPR |
| SFR | SFR |

**Important:** Even though the DMA module has access to all memory and peripherals that are also available to the CPU, it is recommended that the DMA does not access any register that is part of the system arbitration. The DMA, as a system arbitration client must not be read or written by itself or by another DMA instantiation.

The following sections discuss the various control interfaces required for DMA data transfers.

### 16.3.1 Special Function Registers with DMA Access only

The DMA can transfer data to any GPR or SFR location. For better user accessibility, some of the more commonly used SFR spaces have their mirror registers placed in a separate data memory location. These mirror registers can be only accessed by the DMA module through the DMA Source and Destination Address registers. The figure below shows the register map for these registers.

These registers are useful to multiple peripherals together like the Timers, PWMs and also other DMA modules using one of the DMA modules.

**Figure 16-2. Special Function Register Map (DMA Access Only)**

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 40FFh | - | 40DFh | - | 40BFh | - | 409Fh | - | 407Fh | - | 405Fh | - | 403Fh | - | 401Fh | - | 41FFh | - | 41DFh | - |
| 40FEh | - | 40DEh | - | 40BEh | - | 409Eh | - | 407Eh | - | 405Eh | - | 403Eh | - | 401Eh | - | 41FEh | - | 41DEh | - |
| 40FDh | - | 40DDh | - | 40BDh | - | 409Dh | - | 407Dh | ADRESH_M1 | 405Dh | - | 403Dh | - | 401Dh | - | 41FDh | - | 41DDh | - |
| 40FCh | - | 40DCh | - | 40BCh | - | 409Ch | - | 407Ch | ADRESL_M1 | 405Ch | - | 403Ch | - | 401Ch | - | 41FCh | - | 41DCh | - |
| 40FBh | - | 40DBh | - | 40BBh | - | 409Bh | - | 407Bh | ADPCH_M1 | 405Bh | - | 403Bh | - | 401Bh | PWM3S1P2H_M1 | 41FBh | TMR5H_M1 | 41DBh | - |
| 40FAh | - | 40DAh | - | 40BAh | - | 409Ah | - | 407Ah | ADCLK_M1 | 405Ah | - | 403Ah | - | 401Ah | PWM3S1P2L_M1 | 41FAh | TMR5L_M1 | 41DAh | - |
| 40F9h | - | 40D9h | - | 40B9h | - | 4099h | - | 4079h | ADACT_M1 | 4059h | - | 4039h | - | 4019h | PWM3S1P1H_M2 | 41F9h | TMR3H_M1 | 41D9h | - |
| 40F8h | - | 40D8h | - | 40B8h | - | 4098h | - | 4078h | ADREF_M1 | 4058h | - | 4038h | - | 4018h | PWM3S1P1L_M2 | 41F8h | TMR3L_M1 | 41D8h | - |
| 40F7h | - | 40D7h | - | 40B7h | - | 4097h | - | 4077h | ADCON3_M1 | 4057h | - | 4037h | - | 4017h | PWM2S1P2H_M1 | 41F7h | TMR1H_M1 | 41D7h | - |
| 40F6h | ADRESH_M2 | 40D6h | - | 40B6h | - | 4096h | - | 4076h | ADCON2_M1 | 4056h | - | 4036h | - | 4016h | PWM2S1P2L_M1 | 41F6h | TMR1L_M1 | 41D6h | - |
| 40F5h | ADRESL_M2 | 40D5h | - | 40B5h | - | 4095h | - | 4075h | ADCON1_M1 | 4055h | - | 4035h | - | 4015h | PWM2S1P1H_M2 | 41F5h | - | 41D5h | - |
| 40F4h | ADPCH_M2 | 40D4h | - | 40B4h | - | 4094h | - | 4074h | ADCON0_M1 | 4054h | - | 4034h | - | 4014h | PWM2S1P1L_M2 | 41F4h | - | 41D4h | - |
| 40F3h | ADCAP_M2 | 40D3h | - | 40B3h | - | 4093h | - | 4073h | ADCAP_M1 | 4053h | - | 4033h | - | 4013h | PWM1S1P2H_M1 | 41F3h | - | 41D3h | - |
| 40F2h | ADACQH_M2 | 40D2h | - | 40B2h | - | 4092h | - | 4072h | ADACQH_M1 | 4052h | - | 4032h | - | 4012h | PWM1S1P2L_M1 | 41F2h | - | 41D2h | - |
| 40F1h | ADACQL_M2 | 40D1h | - | 40B1h | - | 4091h | - | 4071h | ADACQL_M1 | 4051h | - | 4031h | PWM3PRH_M1 | 4011h | PWM1S1P1H_M2 | 41F1h | - | 41D1h | - |
| 40F0h | ADPREVH_M2 | 40D0h | - | 40B0h | - | 4090h | - | 4070h | ADPREVH_M1 | 4050h | - | 4030h | PWM3PRL_M1 | 4010h | PWM1S1P1L_M2 | 41F0h | - | 41D0h | - |
| 40EFh | ADPREVL_M2 | 40CFh | - | 40AFh | - | 408Fh | - | 406Fh | ADPREVL_M1 | 404Fh | - | 402Fh | PWM3S1P2H_M2 | 400Fh | - | 41EFh | - | 41CFh | - |
| 40EEh | ADRPT_M2 | 40CEh | - | 40AEh | - | 408Eh | - | 406Eh | ADRPT_M1 | 404Eh | - | 402Eh | PWM3S1P2L_M2 | 400Eh | - | 41EEh | - | 41CEh | - |
| 40EDh | ADCNT_M2 | 40CDh | - | 40ADh | - | 408Dh | - | 406Dh | ADCNT_M1 | 404Dh | - | 402Dh | PWM3S1P1H_M3 | 400Dh | - | 41EDh | - | 41CDh | - |
| 40ECh | ADACCU_M2 | 40CCh | - | 40ACh | - | 408Ch | - | 406Ch | ADACCU_M1 | 404Ch | - | 402Ch | PWM3S1P1L_M3 | 400Ch | - | 41ECh | - | 41CCh | - |
| 40CBh | ADACCH_M2 | 40CBh | - | 40ABh | - | 408Bh | - | 406Bh | ADACCH_M1 | 404Bh | - | 402Bh | PWM2PRH_M1 | 400Bh | PWM3S1P1H_M1 | 41CBh | - | 41CBh | - |
| 40EAh | ADACCL_M2 | 40CAh | - | 40AAh | - | 408Ah | - | 406Ah | ADACCL_M1 | 404Ah | - | 402Ah | PWM2PRL_M1 | 400Ah | PWM3S1P1L_M1 | 41EAh | - | 41CAh | - |
| 40E9h | ADFLTRH_M2 | 40C9h | - | 40A9h | - | 4089h | - | 4069h | ADFLTRH_M1 | 4049h | - | 4029h | PWM2S1P2H_M2 | 4009h | PWM2S1P1H_M1 | 41E9h | - | 41C9h | DMAnSIRQ_DMA6 |
| 40E8h | ADFLTRL_M2 | 40C8h | - | 40A8h | - | 4088h | - | 4068h | ADFLTRL_M1 | 4048h | T6PR_M1 | 4028h | PWM2S1P2L_M2 | 4008h | PWM2S1P1L_M1 | 41E8h | - | 41C8h | DMAnAIRQ_DMA6 |
| 40E7h | ADSTPTH_M2 | 40C7h | - | 40A7h | - | 4087h | - | 4067h | ADSTPTH_M1 | 4047h | CCPR3H_M2 | 4027h | PWM2S1P1H_M3 | 4007h | PWM1S1P1H_M1 | 41E7h | - | 41C7h | DMAnCON1_DMA6 |
| 40E6h | ADSTPTL_M2 | 40C6h | - | 40A6h | - | 4086h | - | 4066h | ADSTPTL_M1 | 4046h | CCPR3L_M2 | 4026h | PWM2S1P1L_M3 | 4006h | PWM1S1P1L_M1 | 41E6h | - | 41C6h | DMAnCON0_DMA6 |
| 40E5h | ADERRH_M2 | 40C5h | - | 40A5h | - | 4085h | - | 4065h | ADERRH_M1 | 4045h | T4PR_M1 | 4025h | PWM1PRH_M1 | 4005h | CCPR3H_M1 | 41E5h | - | 41C5h | DMAnSSAU_DMA6 |
| 40E4h | ADERRL_M2 | 40C4h | - | 40A4h | - | 4084h | - | 4064h | ADERRL_M1 | 4044h | CCPR2H_M2 | 4024h | PWM1PRL_M1 | 4004h | CCPR3L_M1 | 41E4h | - | 41C4h | DMAnSSAH_DMA6 |
| 40E3h | ADUTHH_M2 | 40C3h | - | 40A3h | - | 4083h | - | 4063h | ADUTHH_M1 | 4043h | CCPR2L_M2 | 4023h | PWM1S1P2H_M2 | 4003h | CCPR2H_M1 | 41E3h | IOCEF_M1 | 41C3h | DMAnSSAL_DMA6 |
| 40E2h | ADUTHL_M2 | 40C2h | - | 40A2h | - | 4082h | - | 4062h | ADUTHL_M1 | 4042h | T2PR_M1 | 4022h | PWM1S1P2L_M2 | 4002h | CCPR2L_M1 | 41E2h | IOCCF_M1 | 41C2h | DMAnSSZH_DMA6 |
| 40E1h | ADLTHH_M2 | 40C1h | - | 40A1h | - | 4081h | - | 4061h | ADLTHH_M1 | 4041h | CCPR1H_M2 | 4021h | PWM1S1P1H_M3 | 4001h | CCPR1H_M1 | 41E1h | IOCBF_M1 | 41C1h | DMAnSSZL_DMA6 |
| 40E0h | ADLTHL_M2 | 40C0h | - | 40A0h | - | 4080h | - | 4060h | ADLTHL_M1 | 4040h | CCPR1L_M2 | 4020h | PWM1S1P1L_M3 | 4000h | CCPR1L_M1 | 41E0h | IOCAF_M1 | 41C0h | DMAnSPTRU_DMA6 |

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 41FFh | - | 41DFh | - | 41BFh | - | 419Fh | - | 417Fh | DMAnSPTRH_DMA6 | 415Fh | DMAnDPTRL_DMA5 | 413Fh | DMAnSSAH_DMA3 | 411Fh | DMAnDSAH_DMA2 | 41FFh | - | 41DFh | - |
| 41FEh | - | 41DEh | - | 41BEh | - | 419Eh | - | 417Eh | DMAnSPTRL_DMA6 | 415Eh | DMAnDCNTH_DMA5 | 413Eh | DMAnSSAL_DMA3 | 411Eh | DMAnDSAL_DMA2 | 41FEh | - | 41DEh | - |
| 41FDh | - | 41DDh | - | 41BDh | - | 419Dh | - | 417Dh | DMAnSCNTH_DMA6 | 415Dh | DMAnDCNTL_DMA5 | 413Dh | DMAnSSZH_DMA3 | 411Dh | DMAnDSZH_DMA2 | 41FDh | - | 41DDh | - |
| 41FCh | - | 41DCh | - | 41BCh | - | 419Ch | - | 417Ch | DMAnSCNTL_DMA6 | 415Ch | DMAnBUF_DMA5 | 413Ch | DMAnSSZL_DMA3 | 411Ch | DMAnDSZL_DMA2 | 41FCh | - | 41DCh | - |
| 41FBh | TMR5H_M1 | 41DBh | - | 41BBh | - | 419Bh | - | 417Bh | DMAnDSAH_DMA6 | 415Bh | DMAnSIRQ_DMA4 | 413Bh | DMAnSPTRU_DMA3 | 411Bh | DMAnDPTRH_DMA2 | 41FBh | TMR5H_M1 | 41DBh | - |
| 41FAh | TMR5L_M1 | 41DAh | - | 41BAh | - | 419Ah | - | 417Ah | DMAnDSAL_DMA6 | 415Ah | DMAnAIRQ_DMA4 | 413Ah | DMAnSPTRH_DMA3 | 411Ah | DMAnDPTRL_DMA2 | 41FAh | TMR5L_M1 | 41DAh | - |
| 41F9h | TMR3H_M1 | 41D9h | - | 41B9h | - | 4199h | - | 4179h | DMAnDSZH_DMA6 | 4159h | DMAnCON1_DMA4 | 4139h | DMAnSPTRL_DMA3 | 4119h | DMAnDCNTH_DMA2 | 41F9h | TMR3H_M1 | 41D9h | - |
| 41F8h | TMR3L_M1 | 41D8h | - | 41B8h | - | 4198h | - | 4178h | DMAnDSZL_DMA6 | 4158h | DMAnCON0_DMA4 | 4138h | DMAnSCNTH_DMA3 | 4118h | DMAnDCNTL_DMA2 | 41F8h | TMR3L_M1 | 41D8h | - |
| 41F7h | TMR1H_M1 | 41D7h | - | 41B7h | - | 4197h | - | 4177h | DMAnDPTRH_DMA6 | 4157h | DMAnSSAU_DMA4 | 4137h | DMAnSCNTL_DMA3 | 4117h | DMAnBUF_DMA2 | 41F7h | TMR1H_M1 | 41D7h | - |
| 41F6h | TMR1L_M1 | 41D6h | - | 41B6h | - | 4196h | - | 4176h | DMAnDPTRL_DMA6 | 4156h | DMAnSSAH_DMA4 | 4136h | DMAnDSAH_DMA3 | 4116h | DMAnSIRQ_DMA1 | 41F6h | TMR1L_M1 | 41D6h | - |
| 41F5h | - | 41D5h | - | 41B5h | - | 4195h | - | 4175h | DMAnDCNTH_DMA6 | 4155h | DMAnSSAL_DMA4 | 4135h | DMAnDSAL_DMA3 | 4115h | DMAnAIRQ_DMA1 | 41F5h | - | 41D5h | - |
| 41F4h | - | 41D4h | - | 41B4h | - | 4194h | - | 4174h | DMAnDCNTL_DMA6 | 4154h | DMAnSSZH_DMA4 | 4134h | DMAnDSZH_DMA3 | 4114h | DMAnCON1_DMA1 | 41F4h | - | 41D4h | - |
| 41F3h | - | 41D3h | - | 41B3h | - | 4193h | - | 4173h | DMAnBUF_DMA6 | 4153h | DMAnSSZL_DMA4 | 4133h | DMAnDSZL_DMA3 | 4113h | DMAnCON0_DMA1 | 41F3h | - | 41D3h | - |
| 41F2h | - | 41D2h | - | 41B2h | - | 4192h | - | 4172h | DMAnSIRQ_DMA5 | 4152h | DMAnSPTRU_DMA4 | 4132h | DMAnDPTRH_DMA3 | 4112h | DMAnSSAU_DMA1 | 41F2h | - | 41D2h | - |
| 41F1h | - | 41D1h | - | 41B1h | - | 4191h | - | 4171h | DMAnAIRQ_DMA5 | 4151h | DMAnSPTRH_DMA4 | 4131h | DMAnDPTRL_DMA3 | 4111h | DMAnSSAH_DMA1 | 41F1h | - | 41D1h | - |
| 41F0h | - | 41D0h | - | 41B0h | - | 4190h | - | 4170h | DMAnCON1_DMA5 | 4150h | DMAnSPTRL_DMA4 | 4130h | DMAnDCNTH_DMA3 | 4110h | DMAnSSAL_DMA1 | 41F0h | - | 41D0h | - |
| 41EFh | - | 41CFh | - | 41AFh | - | 418Fh | - | 416Fh | DMAnCON0_DMA5 | 414Fh | DMAnSCNTH_DMA4 | 412Fh | DMAnDCNTL_DMA3 | 410Fh | DMAnSSZH_DMA1 | 41EFh | - | 41CFh | - |
| 41EEh | - | 41CEh | - | 41AEh | - | 418Eh | - | 416Eh | DMAnSSAU_DMA5 | 414Eh | DMAnSCNTL_DMA4 | 412Eh | DMAnBUF_DMA3 | 410Eh | DMAnSSZL_DMA1 | 41EEh | - | 41CEh | - |
| 41EDh | - | 41CDh | - | 41ADh | - | 418Dh | - | 416Dh | DMAnSSAH_DMA5 | 414Dh | DMAnDSAH_DMA4 | 412Dh | DMAnSIRQ_DMA2 | 410Dh | DMAnSPTRU_DMA1 | 41EDh | - | 41CDh | - |
| 41ECh | - | 41CCh | - | 41ACh | - | 418Ch | - | 416Ch | DMAnSSAL_DMA5 | 414Ch | DMAnDSAL_DMA4 | 412Ch | DMAnAIRQ_DMA2 | 410Ch | DMAnSPTRH_DMA1 | 41ECh | - | 41CCh | - |
| 41CBh | - | 41CBh | - | 41ABh | - | 418Bh | - | 416Bh | DMAnSSZH_DMA5 | 414Bh | DMAnDSZH_DMA4 | 412Bh | DMAnCON1_DMA2 | 410Bh | DMAnSPTRL_DMA1 | 41CBh | - | 41CBh | - |
| 41EAh | - | 41CAh | - | 41AAh | - | 418Ah | - | 416Ah | DMAnSSZL_DMA5 | 414Ah | DMAnDSZL_DMA4 | 412Ah | DMAnCON0_DMA2 | 410Ah | DMAnSCNTH_DMA1 | 41EAh | - | 41CAh | - |
| 41E9h | - | 41C9h | - | 41A9h | - | 4189h | DMAnSIRQ_DMA6 | 4169h | DMAnSPTRU_DMA5 | 4149h | DMAnDPTRH_DMA4 | 4129h | DMAnSSAU_DMA2 | 4109h | DMAnSCNTL_DMA1 | 41E9h | - | 41C9h | DMAnSIRQ_DMA6 |
| 41E8h | - | 41C8h | - | 41A8h | - | 4188h | DMAnAIRQ_DMA6 | 4168h | DMAnSPTRH_DMA5 | 4148h | DMAnDPTRL_DMA4 | 4128h | DMAnSSAH_DMA2 | 4108h | DMAnSSAH_DMA1 | 41E8h | - | 41C8h | DMAnAIRQ_DMA6 |
| 41E7h | - | 41C7h | - | 41A7h | - | 4187h | DMAnCON1_DMA6 | 4167h | DMAnSPTRL_DMA5 | 4147h | DMAnDCNTH_DMA4 | 4127h | DMAnSSAL_DMA2 | 4107h | DMAnSSAL_DMA1 | 41E7h | - | 41C7h | DMAnCON1_DMA6 |
| 41E6h | - | 41C6h | - | 41A6h | - | 4186h | DMAnCON0_DMA6 | 4166h | DMAnSCNTH_DMA5 | 4146h | DMAnDCNTL_DMA4 | 4126h | DMAnSSZH_DMA2 | 4106h | DMAnSSZH_DMA1 | 41E6h | - | 41C6h | DMAnCON0_DMA6 |
| 41E5h | - | 41C5h | - | 41A5h | - | 4185h | DMAnSSAU_DMA6 | 4165h | DMAnSCNTL_DMA5 | 4145h | DMAnBUF_DMA4 | 4125h | DMAnSSZL_DMA2 | 4105h | DMAnSSZL_DMA1 | 41E5h | - | 41C5h | DMAnSSAU_DMA6 |
| 41E4h | - | 41C4h | - | 41A4h | - | 4184h | DMAnSSAH_DMA6 | 4164h | DMAnDSAH_DMA5 | 4144h | DMAnSIRQ_DMA3 | 4124h | DMAnSPTRU_DMA2 | 4104h | DMAnDPTRH_DMA1 | 41E4h | - | 41C4h | DMAnSSAH_DMA6 |
| 41E3h | IOCEF_M1 | 41C3h | - | 41A3h | - | 4183h | DMAnSSAL_DMA6 | 4163h | DMAnDSAL_DMA5 | 4143h | DMAnAIRQ_DMA3 | 4123h | DMAnSPTRH_DMA2 | 4103h | DMAnDPTRL_DMA1 | 41E3h | IOCEF_M1 | 41C3h | DMAnSSAL_DMA6 |
| 41E2h | IOCCF_M1 | 41C2h | - | 41A2h | - | 4182h | DMAnSSZH_DMA6 | 4162h | DMAnDSZH_DMA5 | 4142h | DMAnCON1_DMA3 | 4122h | DMAnSPTRL_DMA2 | 4102h | DMAnDCNTH_DMA1 | 41E2h | IOCCF_M1 | 41C2h | DMAnSSZH_DMA6 |
| 41E1h | IOCBF_M1 | 41C1h | - | 41A1h | - | 4181h | DMAnSSZL_DMA6 | 4161h | DMAnDSZL_DMA5 | 4141h | DMAnCON0_DMA3 | 4121h | DMAnSCNTH_DMA2 | 4101h | DMAnDCNTL_DMA1 | 41E1h | IOCBF_M1 | 41C1h | DMAnSSZL_DMA6 |
| 41E0h | IOCAF_M1 | 41C0h | - | 41A0h | - | 4180h | DMAnSPTRU_DMA6 | 4160h | DMAnDPTRH_DMA5 | 4140h | DMAnSSAU_DMA3 | 4120h | DMAnSCNTL_DMA2 | 4100h | DMAnBUF_DMA1 | 41E0h | IOCAF_M1 | 41C0h | DMAnSPTRU_DMA6 |

## 16.3.2 DMA Addressing

The start addresses for the source read and destination write operations are set using the DMAnSSA and DMAnDSA registers, respectively.

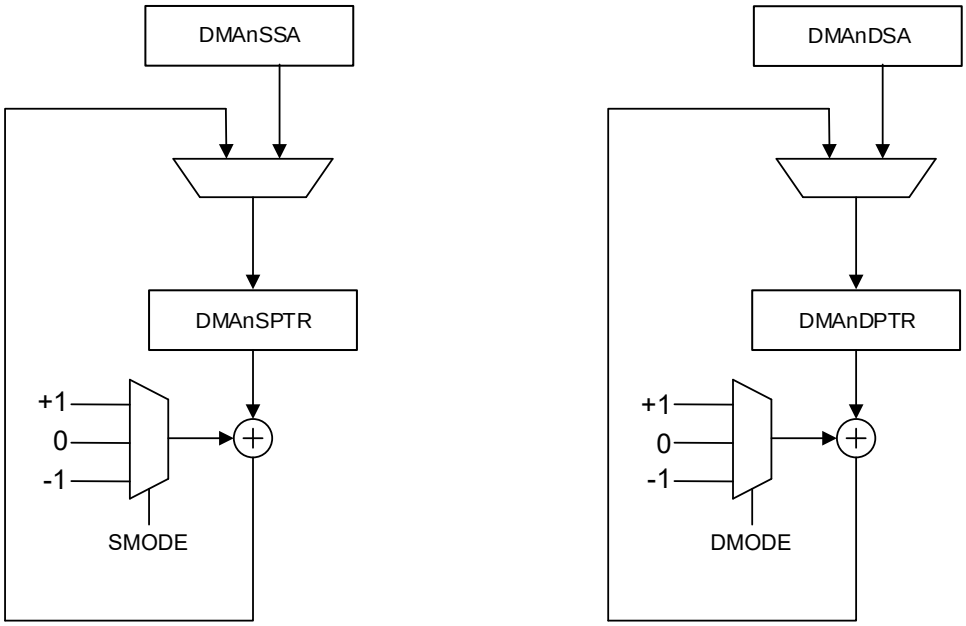
When the DMA message transfers are in progress, the DMAnSPTR and DMAnDPTR registers contain the current Address Pointers for each source read and destination write operation. These registers are modified after each transaction based on the Address mode selection bits.

The SMODE and DMODE bits determine the Address modes of operation by controlling how the DMAnSPTR and DMAnDPTR registers are updated after every DMA data transaction (Figure 16-3).

Each address can be separately configured to:

- Remain unchanged
- Increment by 1
- Decrement by 1

**Figure 16-3. DMA Pointers Block Diagram**




The DMA can initiate data transfers from the PFM, Data EEPROM or SFR/GPR space. The SMR bits are used to select the type of memory being pointed to by the Source Address Pointer. The SMR bits are required because the PFM and SFR/GPR spaces have overlapping addresses that do not allow the specified address to uniquely define the memory location to be accessed.

**Important:**

1. For proper memory read access to occur, the combination of address and space selection must be valid.
2. The destination does not have space selection bits because it can only write to the SFR/GPR space.

### 16.3.3 DMA Message Size/Counters

A transaction is the transfer of one byte. A message consists of one or more transactions. A complete DMA process consists of one or more messages. The size registers determine how many transactions are in a message. The DMAnSSZ registers determine the source size and DMAnDSZ registers determine the destination size.

When a DMA transfer is initiated, the size registers are copied to corresponding counter registers that control the duration of the message. The DMAnSCNT registers count the source transactions and the DMAnDCNT registers count the destination transactions. Both are simultaneously decremented by one after each transaction.

A message is started by setting the DGO bit and terminates when the smaller of the two counters reaches zero.

When either counter reaches zero, the DGO bit is cleared and the counter and pointer registers are immediately reloaded with the corresponding size and address data. If the other counter did not reach zero, then the next message will continue with the count and address corresponding to that register. Refer to Figure 16-4.

When the Source and Destination Size registers are not equal, then the ratio of the largest to the smallest size determines how many messages are in the DMA process. For example, when the destination size is six and the source size is two, then each message will consist of two transactions and the complete DMA process will consist of three messages. When the larger size is not an even integer of the smaller size, then the last message in the process will terminate early when the larger count reaches zero. In that case, the larger counter will reset and the smaller counter will have a remainder skewing any subsequent messages by that amount.

© 2019-2023 Microchip Technology Inc. and its subsidiaries

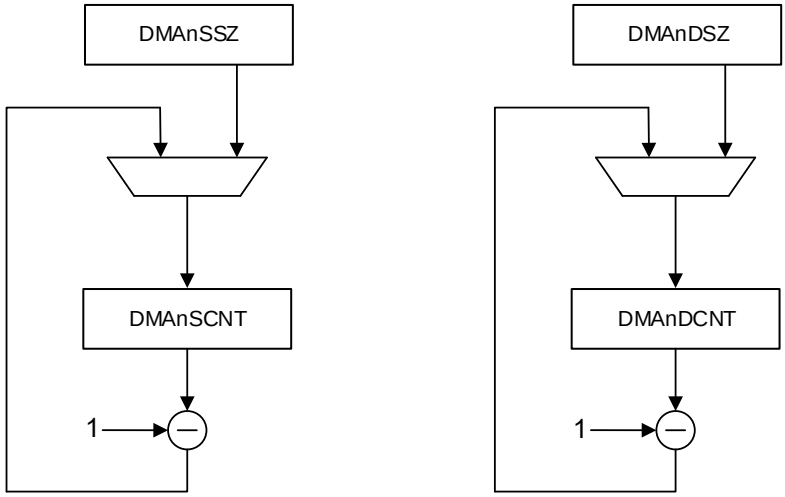
Table 16-2 has a few examples of configuring DMA Message sizes.

**Important:** Reading the DMAnSCNT or DMAnDCNT registers will never return zero. When either register is decremented from '1', it is immediately reloaded from the corresponding size register.

**Table 16-2. Example Message Size**

| Operation | Example | SCNT | DCNT | Comments |
|---|---|---|---|---|
| Read from single SFR location to RAM | UART Receive Buffer | 1 | N | N equals the number of bytes desired in the destination buffer. N ≥ 1. |
| Write to single SFR location from RAM | UART Transmit Buffer | N | 1 | N equals the number of bytes desired in the source buffer. N ≥ 1. |
| Read from multiple SFR location | ADC Result registers | 2 | 2*N | N equals the number of ADC results to be stored in memory. N ≥ 1 |
| Write to Multiple SFR registers | PWM Duty Cycle registers | 2*N | 2 | N equals the number of PWM duty cycle values to be loaded from a memory table. N ≥ 1 |

**Figure 16-4. DMA Counters Block Diagram**




### 16.3.4 DMA Message Transfers

Once the Enable bit is set to start DMA message transfers, the Source/Destination Pointer and Counter registers are initialized to the conditions shown in the table below.

**Table 16-3. DMA Initial Conditions**

| Register | Value Loaded |
|---|---|
| DMAnSPTR | DMAnSSA |
| DMAnSCNT | DMAnSSZ |
| DMAnDPTR | DMAnDSA |
| DMAnDCNT | DMAnDSZ |

During the DMA operation after each transaction, Table 16-4 and Table 16-5 indicate how the Source/Destination Pointer and Counter registers are modified.

The following sections discuss how to initiate and terminate DMA transfers.

**Table 16-4. DMA Source Pointer/Counter During Operation**

| Register | Modified Source Counter/Pointer Value |
|---|---|
| DMAnSCNT != 1 | DMAnSCNT = DMAnSCNT -1 |
| | SMODE = 00: DMAnSPTR = DMAnSPTR |
| | SMODE = 01: DMAnSPTR = DMAnSPTR + 1 |
| | SMODE = 10: DMAnSPTR = DMAnSPTR - 1 |
| DMAnSCNT == 1 | DMAnSCNT = DMAnSSZ |
| | DMAnSPTR = DMAnSSA |

**Table 16-5. DMA Destination Pointer/Counter During Operation**

| Register | Modified Destination Counter/Pointer Value |
|---|---|
| DMAnDCNT != 1 | DMAnDCNT = DMAnDCNT -1 |
| | DMODE = 00: DMAnDPTR = DMAnDPTR |
| | DMODE = 01: DMAnDPTR = DMAnDPTR + 1 |
| | DMODE = 10: DMAnDPTR = DMAnDPTR - 1 |
| DMAnDCNT == 1 | DMAnDCNT = DMAnDSZ |
| | DMAnDPTR = DMAnDSA |

### 16.3.4.1 Starting DMA Message Transfers

The DMA can initiate data transactions by either of the following two conditions:

- User software control
- Hardware trigger, SIRQ

#### 16.3.4.1.1 User Software Control

Software starts or stops DMA transaction by setting/clearing the DGO bit. The DGO bit is also used to indicate whether a DMA hardware trigger has been received and a message is in progress.

**Important:**

1. Software start can only occur when the EN bit is set.
2. If the CPU writes to the DGO bit while it is already set, there is no effect on the system, the DMA will continue to operate normally.

#### 16.3.4.1.2 Hardware Trigger, SIRQ

A hardware trigger is an interrupt request from another module sent to the DMA with the purpose of starting a DMA message. The DMA start trigger source is user-selectable using the DMAnSIRQ register.

The SIRQEN bit is used to enable sampling of external interrupt triggers by which a DMA transfer can be started. When set, the DMA will sample the selected interrupt source and when cleared, the DMA will ignore the interrupt source. Clearing the SIRQEN bit does not stop a DMA transaction currently in progress, it only stops more hardware request signals from being received.

### 16.3.4.2 Stopping DMA Message Transfers

The DMA controller can stop data transactions by any of the following conditions:

- Clearing the DGO bit
- Hardware abort trigger, AIRQ
- Source count reload
- Destination count reload
- Clearing the EN bit

#### 16.3.4.2.1 User Software Control

If the user clears the DGO bit, the message will be stopped and the DMA will remain in the current configuration.

For example, if the user clears the DGO bit after source data has been read, but before it is written to the destination, then the data in the DMAnBUF register will not reach its destination.

This is also referred to as a soft-stop as the operation can resume, if desired, by setting the DGO bit again.

#### 16.3.4.2.2 Hardware Trigger, AIRQ

The AIRQEN bit is used to enable sampling of external interrupt triggers by which a DMA transaction can be aborted.

Once an abort interrupt request has been received, the DMA will perform a soft-stop by clearing the DGO bit, as well as clearing the SIRQEN bit so overruns do not occur. The AIRQEN bit is also cleared to prevent additional abort signals from triggering false aborts.

If desired, the DGO bit can be set again and the DMA will resume operation from where it left off after the soft stop had occurred, as none of the DMA state information is changed in the event of an abort.

#### 16.3.4.2.3 Source Count Reload

A DMA message is considered to be complete when the Source Count register is decremented from '1' and then reloaded (i.e., once the last byte from either the source read or destination write has occurred). When the SSTP bit is set and the Source Count register is reloaded, then further message transfer is stopped.

#### 16.3.4.2.4 Destination Count Reload

A DMA message is considered to be complete when the Destination Count register is decremented from 1 and then reloaded (i.e., once the last byte from either the source read or destination write has occurred). When the DSTP bit is set and the Destination Count register is reloaded then further message transfer is stopped.

**Important:** Reading the DMAnSCNT or DMAnDCNT registers will never return zero. When either register is decremented from '1', it is immediately reloaded from the corresponding size register.

#### 16.3.4.2.5 Clearing the EN Bit

If the user clears the EN bit, the message will be stopped and the DMA will return to its default configuration. This is also referred to as a hard stop, as the DMA cannot resume operation from where it was stopped.

**Important:** After the DMA message transfer is stopped, it requires an extra instruction cycle before the Stop condition takes effect. Thus, after the Stop condition has occurred, a source read or a destination write can occur depending on the source or destination bus availability.

## 16.4 Disable DMA Message Transfer Upon Completion

Once the DMA message is complete, it may be desirable to disable the trigger source to prevent overrun or under run of data. This can be done by any of the following methods:

- Clearing the SIRQEN bit
- Setting the SSTP bit
- Setting the DSTP bit

### 16.4.1 Clearing the SIRQEN Bit

Clearing the SIRQEN bit stops the sampling of external start interrupt triggers, hence preventing further DMA message transfers.

An example is a communications peripheral with a level-triggered interrupt. The peripheral will continue to request data (because its buffer is empty) even though there is no more data to be moved. Disabling the SIRQEN bit prevents the DMA from processing these requests.

### 16.4.2 Source/Destination Stop

The SSTP and DSTP bits determine whether or not to disable the hardware triggers (SIRQEN = 0), once a DMA message has completed.

When the SSTP bit is set and the DMAnSCNT = 0, then the SIRQEN bit will be cleared. Similarly, when the DSTP bit is set and the DMAnDCNT = 0, the SIRQEN bit will be cleared.

**Important:** The SSTP and DSTP bits are independent functions and do not depend on each other. It is possible for a message to be stopped by either counter at message end or both counters at message end.

## 16.5 Types of Hardware Triggers

The DMA has two different trigger inputs, the source trigger and the abort trigger. Each of these trigger sources is user configurable using the DMAnSIRQ and DMAnAIRQ registers.

Based on the source selected for each trigger, there are two types of requests that can be sent to the DMA:

- Edge triggers
- Level triggers

### 16.5.1 Edge Trigger Requests

An edge request occurs only once when a given module interrupt requirements are true. Examples of edge triggers are the ADC conversion complete and the interrupt-on-change interrupts.

### 16.5.2 Level Trigger Requests

A level request is asserted as long as the condition that causes the interrupt is true. Examples of level triggers are the UART receive and transmit interrupts.

## 16.6 Types of Data Transfers

Based on the memory access capabilities of the DMA (see Table 16-1), the following sections discuss the different types of data movement between the source and destination memory regions.

- N:1
  This type of transfer is common when sending predefined data packets (such as strings) through a single interface point (such as communications modules transmit registers).
- N:N
  This type of transfer is useful for moving information out of the program Flash or Data EEPROM to SRAM for manipulation by the CPU or other peripherals.
- 1:1
  This type of transfer is common when bridging two different modules data streams together (communications bridge).
- 1:N
  This type of transfer is useful for moving information from a single data source into a memory buffer (communications receive registers).

## 16.7 DMA Interrupts

Each DMA has its own set of four interrupt flags, used to indicate a range of conditions during data transfers. The interrupt flag bits can be accessed using the corresponding PIR registers (refer to the **"VIC - Vectored Interrupt Controller"** chapter).

### 16.7.1 DMA Source Count Interrupt

The Source Count Interrupt Flag (DMAxSCNTIF) is set every time the DMAnSCNT register reaches zero and is reloaded to its starting value.

### 16.7.2 DMA Destination Count Interrupt

The Destination Count Interrupt Flag (DMAxDCNTIF) is set every time the DMAnDCNT register reaches zero and is reloaded to its starting value.

The DMA source and destination count interrupts signal the CPU when the DMA messages are completed.

### 16.7.3 Abort Interrupt

The Abort Interrupt Flag (DMAxAIF) is used to signal that the DMA has halted activity due to an abort signal from one of the abort sources. This is used to indicate that the transaction has been halted by a hardware event.

### 16.7.4 Overrun Interrupt

When the DMA receives a trigger to start a new message before the current message is completed, then the Overrun Interrupt Flag (DMAxORIF) bit is set.

This condition indicates that the DMA is being requested before its current transaction is finished. This implies that the active DMA may not be able to keep up with the demands from the peripheral module being serviced, which may result in data loss.

The DMAxORIF flag being set does not cause the current DMA transfer to terminate.

The overrun interrupt is only available for trigger sources that are edge-based, and not available for sources that are level-based. Therefore, a level-based interrupt source does not trigger a DMA overrun error due to the potential latency issues in the system.

An example of an interrupt that can use the overrun interrupt is a timer overflow (or period match) interrupt. This event only happens every time the timer rolls over and is not dependent on any other system conditions.

An example of an interrupt that does not allow the overrun interrupt is the UART TX buffer. The UART will continue to assert the interrupt until the DMA is able to process the message. Due to latency issues, the DMA may not be able to service an empty buffer immediately, but the UART continues to assert its transmit interrupt until it is serviced. If overrun was allowed in this case, the overrun would occur almost immediately, as the module samples the interrupt sources every instruction cycle.

## 16.8 DMA Setup and Operation

The following steps illustrate how to configure the DMA for data transfer:

1. Select the desired DMA using the DMASELECT register.
2. Program the appropriate source and destination addresses for the transaction into the DMAnSSA and DMAnDSA registers.
3. Select the source memory region that is being addressed by the DMAnSSA register, using the SMR bits.
4. Program the SMODE and DMODE bits to select the Addressing mode.
5. Program the source size (DMAnSSZ) and destination size (DMAnDSZ) registers with the number of bytes to be transferred. It is recommended for proper operation that the size registers be a multiple of each other.
6. If the user desires to disable data transfers once the message has completed, then the SSTP and DSTP bits need to be set. (See the Source/Destination Stop section).
7. If using hardware triggers for data transfer, set up the hardware trigger interrupt sources for the starting and aborting DMA transfers (DMAnSIRQ and DMAnAIRQ), and set the corresponding Interrupt Request Enable (SIRQEN and AIRQEN) bits.

© 2019-2023 Microchip Technology Inc. and its subsidiaries

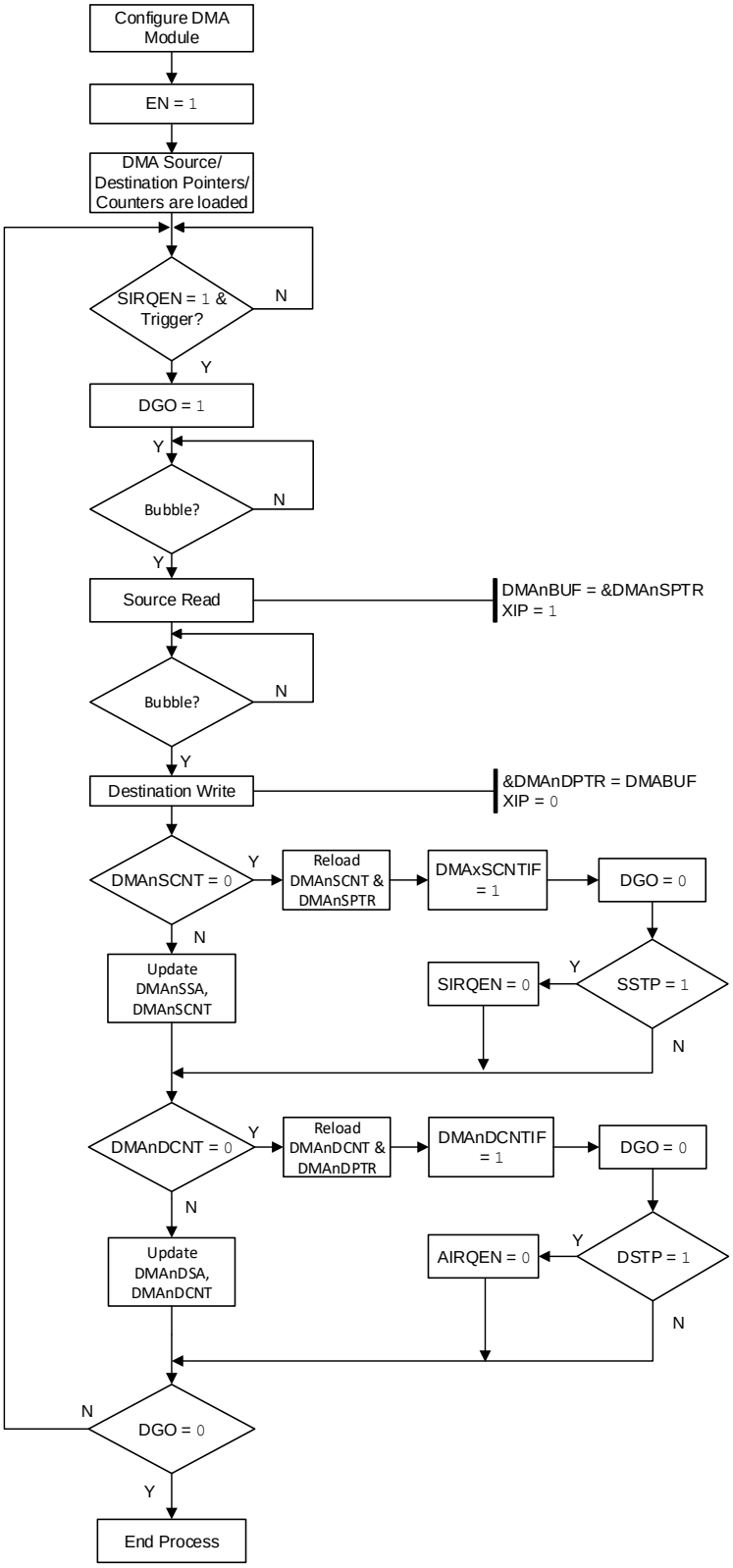
8. Select the priority level for the DMA (see the **“System Arbitration”** section in the **“PIC18 CPU”** chapter) and lock the priorities (see the **“Priority Lock”** section in the **“PIC18 CPU”** chapter).
9. Enable the DMA by setting the EN bit.
10. If using software control for data transfer, set the DGO bit, else this bit will be set by the hardware trigger.

Once the DMA is set up, Figure 16-5 describes the sequence of operation when the DMA uses hardware triggers and utilizes the unused CPU cycles (bubble) for DMA transfers.

The following sections describe with visual reference the sequence of events for different configurations of the DMA module.

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

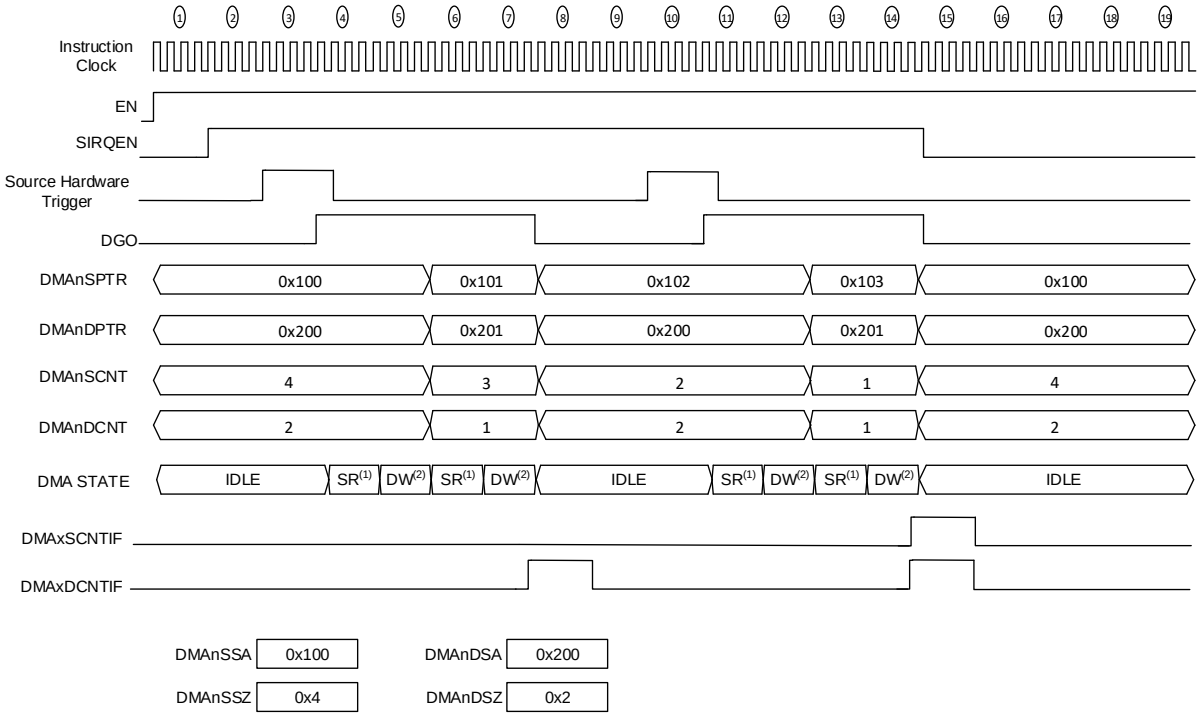
**Figure 16-5. DMA Operation with Hardware Trigger**



### 16.8.1 Source Stop

When the Source Stop bit is set (SSTP = 1) and the DMAnSCNT register reloads, the DMA clears the SIRQEN bit to stop receiving new start interrupt request signals and sets the DMAnSCNTIF flag. Refer to the figure below for more details.

**Figure 16-6. GPR-GPR Transactions with Hardware Triggers, SSTP = 1**



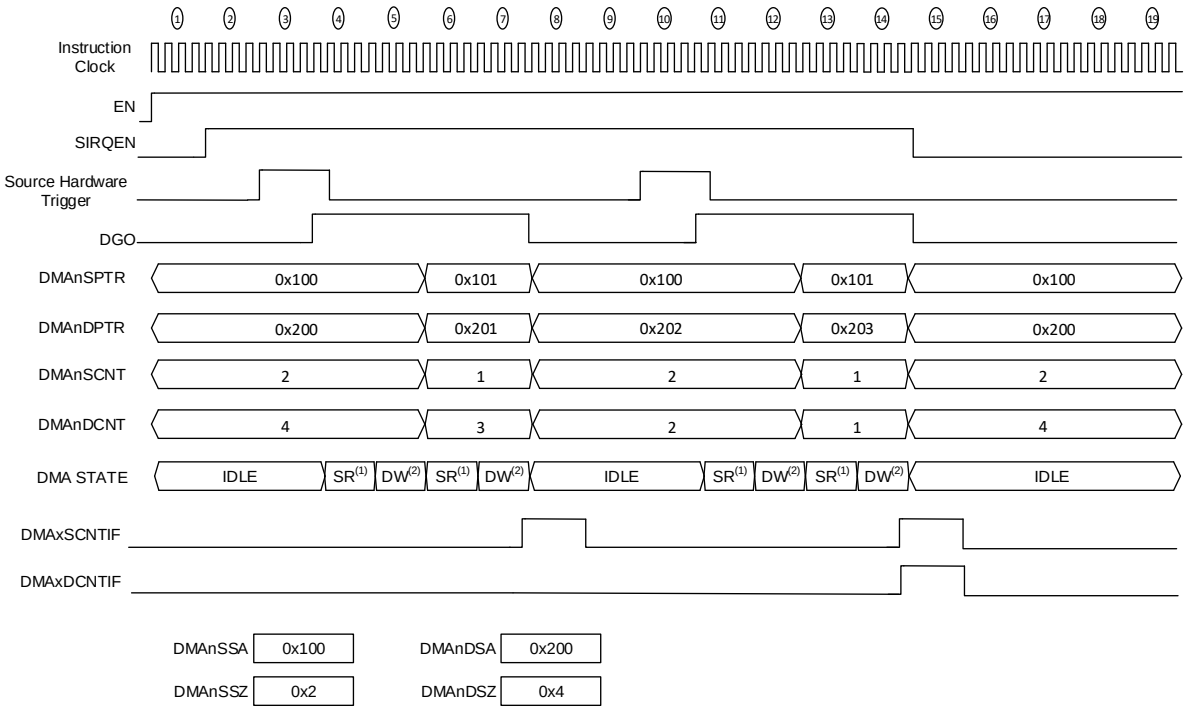

**Notes:**

1. SR - Source Read
2. DW - Destination Write

### 16.8.2 Destination Stop

When the Destination Stop bit is set (DSTP = 1) and the DMAxDCNT register reloads, the DMA clears the SIRQEN bit to stop receiving new start interrupt request signals and sets the DMAnDCNTIF flag.

**Figure 16-7. GPR-GPR Transactions with Hardware Triggers, DSTP = 1**



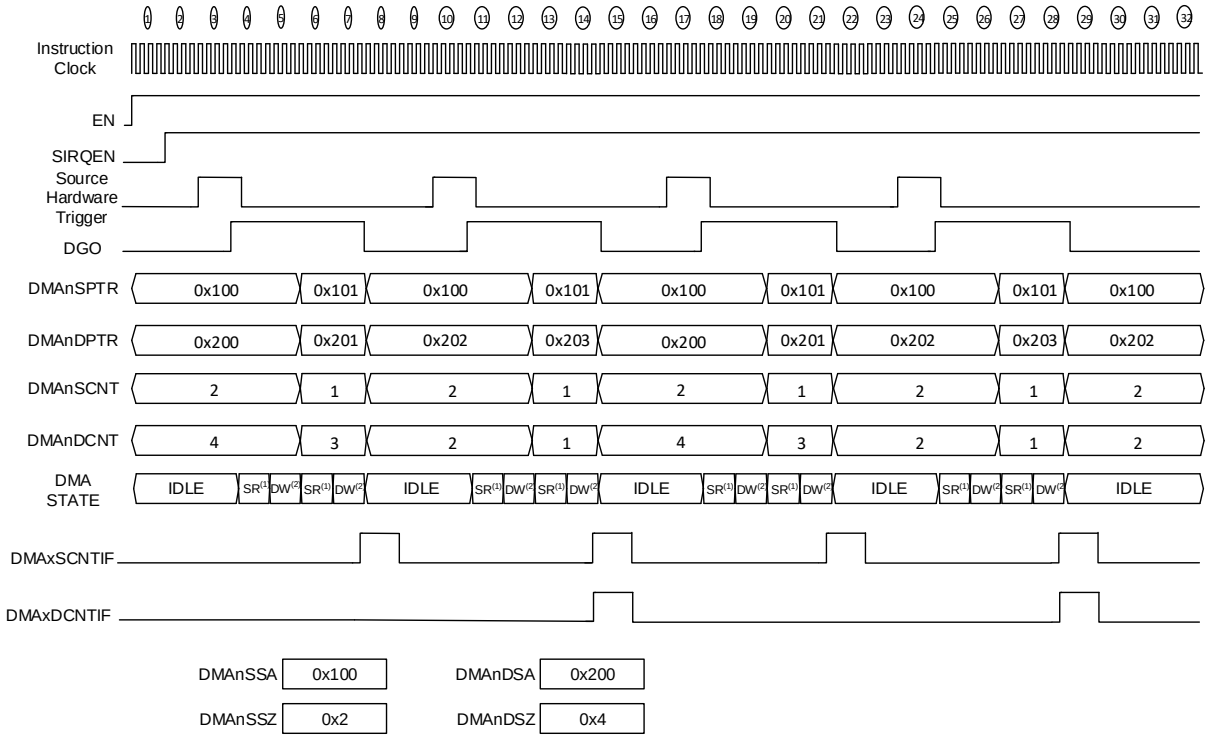

**Notes:**

1. SR - Source Read
2. DW - Destination Write

### 16.8.3 Continuous Transfer

When the Source or the Destination Stop bit is cleared (SSTP, DSTP = 0), the transactions continue unless stopped by the user. The DMAxSCNTIF and DMAxDCNTIF flags are set whenever the respective counter registers are reloaded.

**Figure 16-8. GPR-GPR Transactions with Hardware Triggers, SSTP, DSTP = 0**



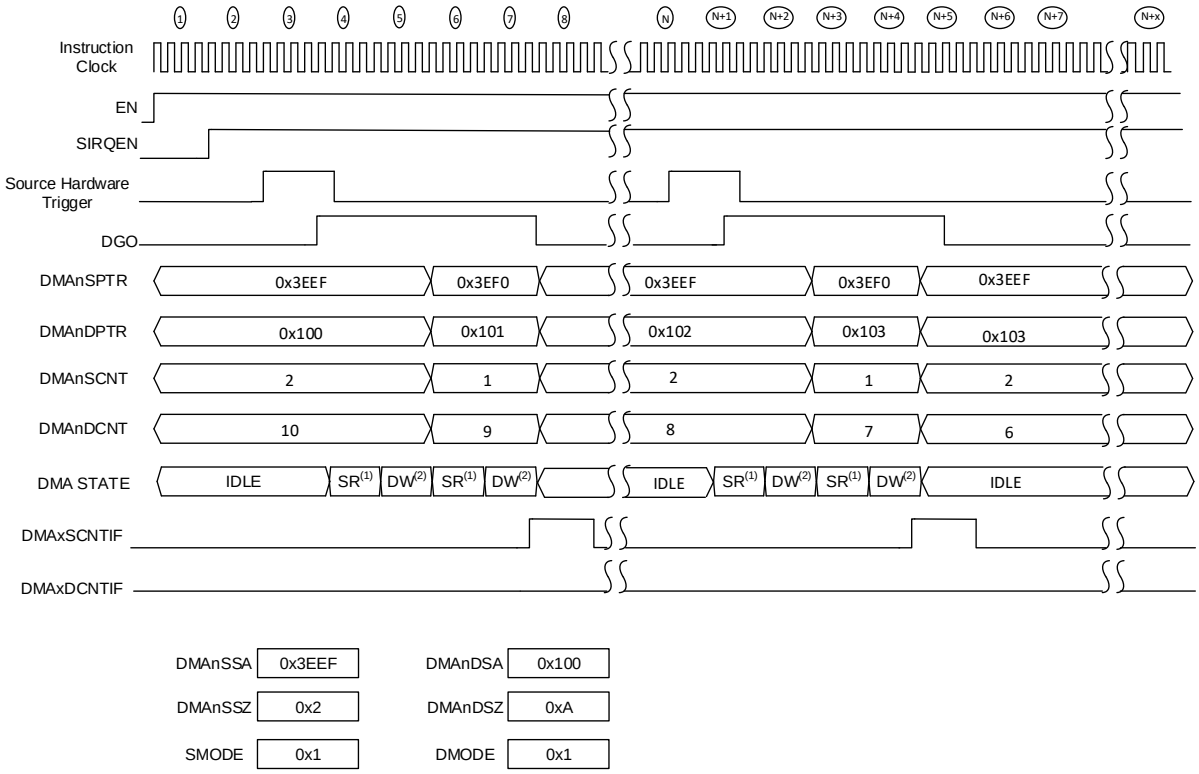

**Notes:**

1. SR - Source Read
2. DW - Destination Write

### 16.8.4 Transfer from SFR to GPR

The following visual reference describes the sequence of events when copying ADC results to a GPR location. The ADC interrupt flag can be chosen as the source hardware trigger, the source address can be set to point to the ADC Result registers (e.g., at 0x3EEF), and the destination address can be set to point to any chosen GPR location (e.g., at 0x100).

**Figure 16-9. SFR Space to GPR Space Transfer**



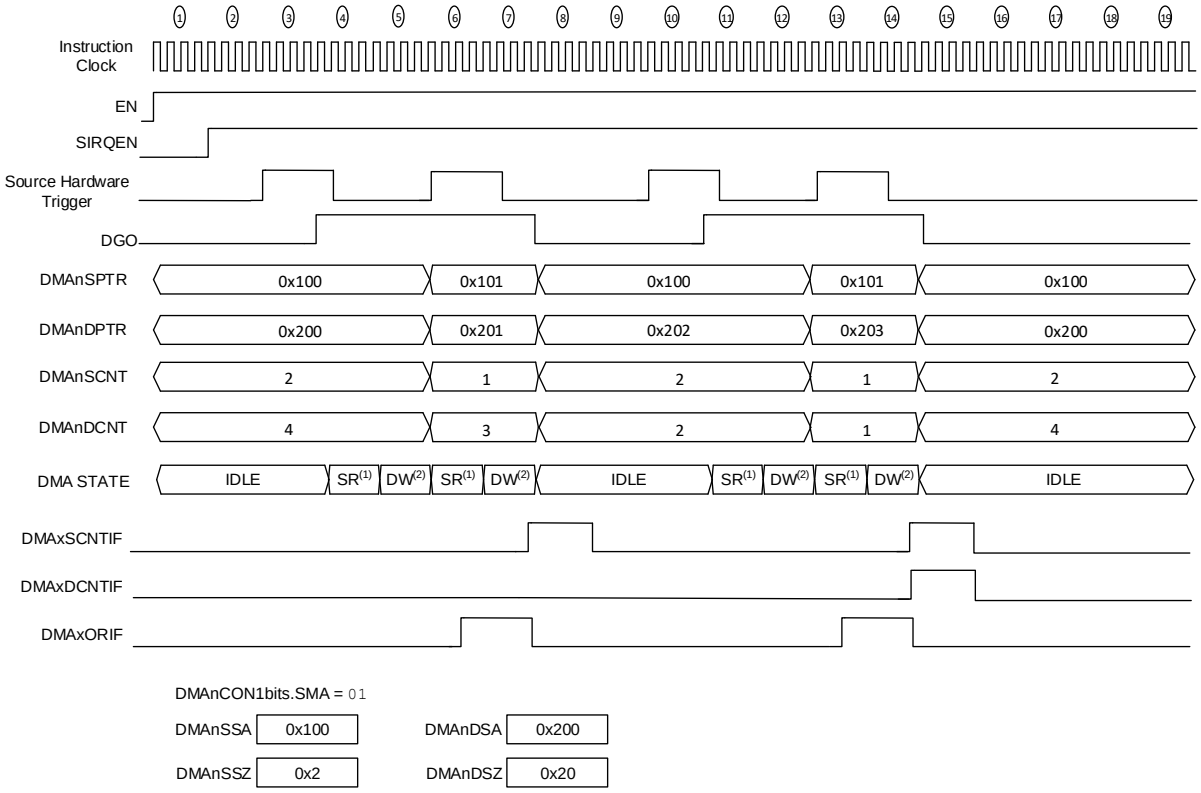

**Notes:**

1. SR - Source Read
2. DW - Destination Write

### 16.8.5 Overrun Condition

The Overrun Interrupt flag is set if the DMA receives a trigger to start a new message before the current message is completed.

**Figure 16-10. Overrun Interrupt**



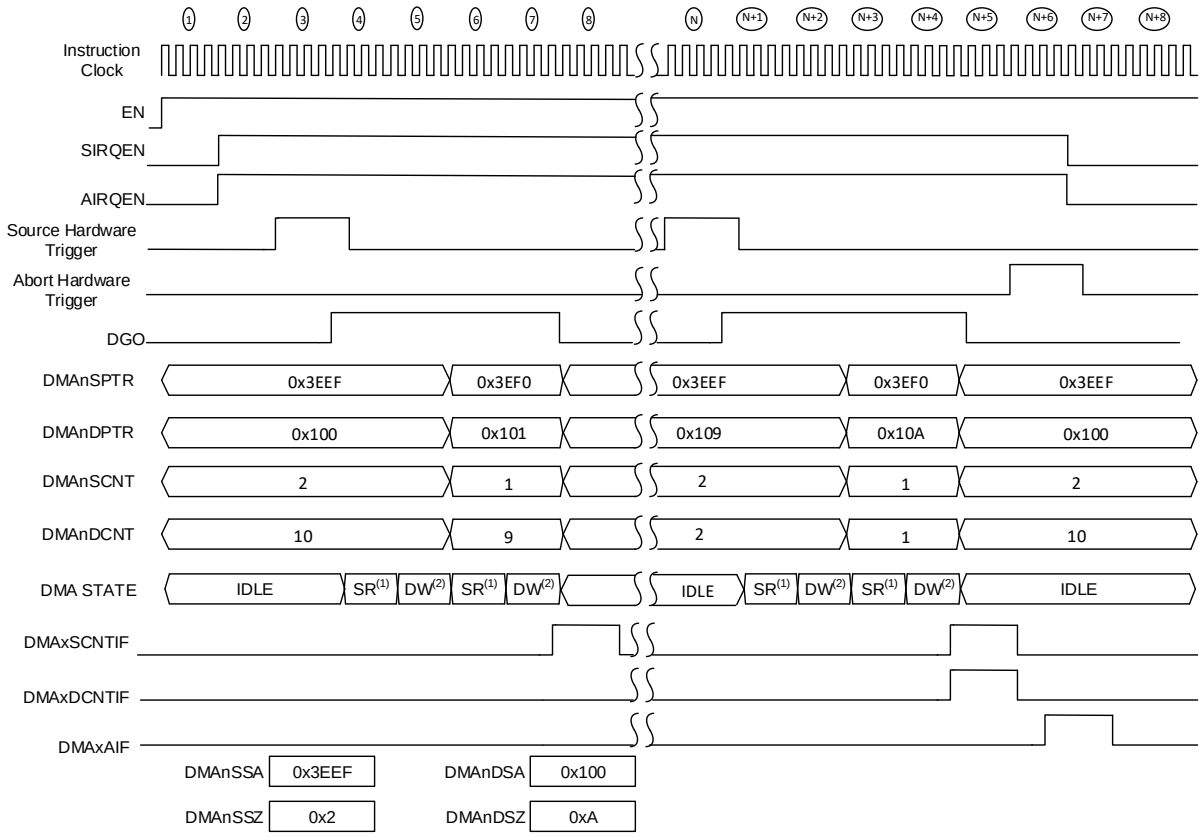

**Notes:**

1. SR - Source Read
2. DW - Destination Write

### 16.8.6 Abort Trigger, Message Complete

The AIRQEN needs to be set in order for the DMA to sample abort interrupt sources. When an abort interrupt is received, the SIRQEN bit is cleared and the AIRQEN bit is cleared to avoid receiving further abort triggers.

**Figure 16-11. Abort at the End of Message**




**Notes:**

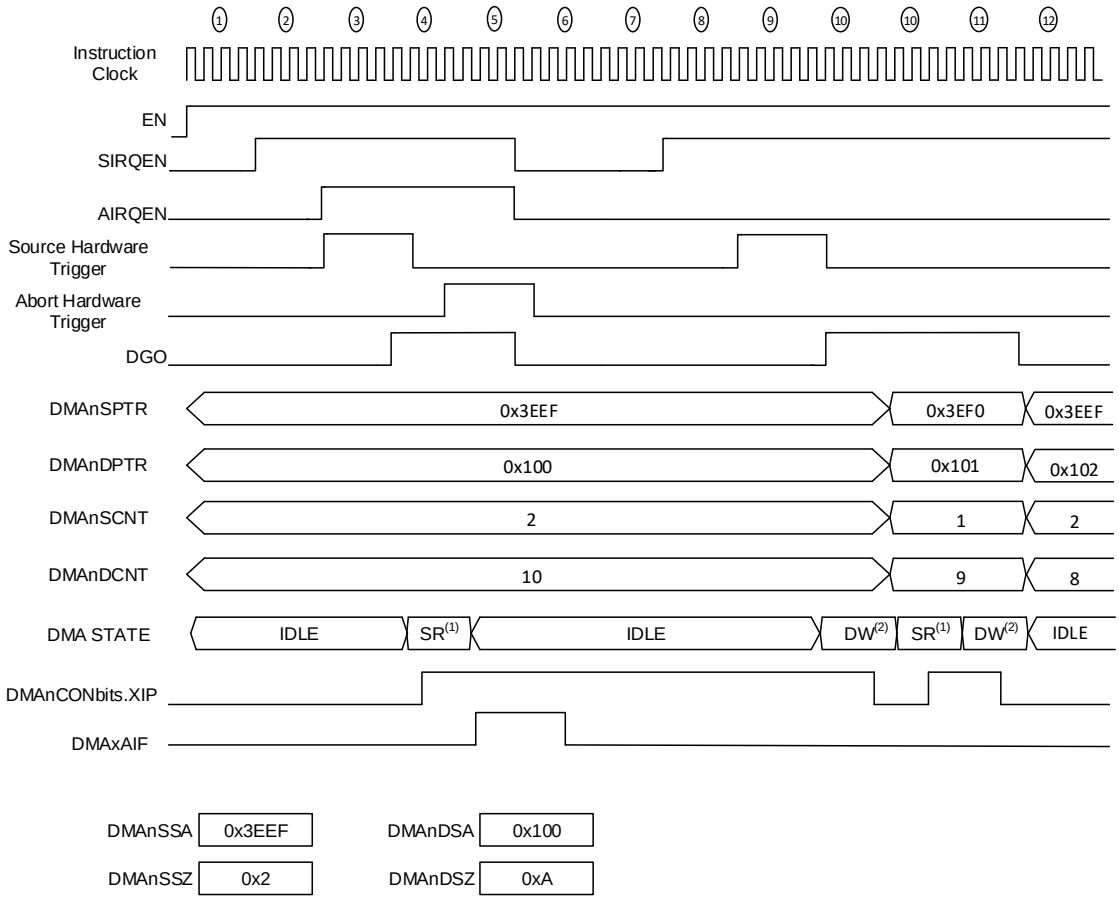
1. SR - Source Read
2. DW - Destination Write

### 16.8.7 Abort Trigger, Message in Progress

When an abort interrupt request is received in a DMA transaction, the DMA will perform a soft-stop by clearing the DGO bit (i.e., if the DMA was reading the source register, it will complete the read operation and then clear the DGO bit).

The SIRQEN bit is cleared to prevent any overrun and the AIRQEN bit is cleared to prevent any false aborts. When the DGO bit is set again, the DMA will resume operation from where it left off after the soft-stop.

**Figure 16-12. Abort During Message Transfer**




**Notes:**

1. SR - Source Read
2. DW - Destination Write

## 16.9 Reset

The DMA registers are set to the default state on any Reset. The registers are also reset to the default state when the enable bit is cleared (EN = 0). User firmware needs to setup all the registers to resume DMA operation.

## 16.10 Power-Saving Mode Operation

The DMA utilizes system clocks and it is treated as a peripheral when it comes to power-saving operations. Like other peripherals, the DMA also uses Peripheral Module Disable bits to further tailor its operation in low-power states.

### 16.10.1 Sleep Mode

When the device enters Sleep mode, the system clock to the module is shut down, therefore no DMA operation is supported in Sleep. Once the system clock is disabled, the requisite read and write clocks are also disabled, without which the DMA cannot perform any of its tasks.

Any transfers that may be in progress are resumed on exiting from Sleep mode. Register contents are not affected by the device entering or leaving Sleep mode. It is recommended that DMA transactions be allowed to finish before entering Sleep mode.

### 16.10.2 Idle Mode

In Idle mode, all of the system clocks (including the read and write clocks) are still operating, but the CPU is not using them to save power.

Therefore, every instruction cycle is available to the system arbiter and if the bubble is granted to the DMA, it may be utilized to move data.

### 16.10.3 Doze Mode

Similar to the Idle mode, the CPU does not utilize all of the available instruction cycles slots that are available to it to save power. It only executes instructions based on its Doze mode settings.

Therefore, every instruction not used by the CPU is available for system arbitration and may be utilized by the DMA, if granted by the arbiter.

### 16.10.4 Peripheral Module Disable

The Peripheral Module Disable (PMD) registers provide a method to disable DMA by gating all clock sources supplied to it. The respective DMAxMD bit needs to be set to disable the DMA.

## 16.11 Example Setup Code

This code example illustrates using DMA1 to transfer 10 bytes of data from 0x1000 in Flash memory to the UART transmit buffer.

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

```
void initializeDMA(){
//Select DMA1 by setting DMASELECT register to 0x00
    DMASELECT = 0x00;
//DMAnCON1 - DPTR remains, Source Memory Region PFM, SPTR increments, SSTP
    DMAnCON1 = 0x0B;
//Source registers
//Source size
    DMAnSSZH = 0x00;
    DMAnSSZL = 0x0A;
//Source start address, 0x1000
    DMAnSSAU = 0x00;
    DMAnSSAH = 0x10;
    DMAnSSAL = 0x00;
//Destination registers
//Destination size
    DMAnDSZH = 0x00;
    DMAnDSZL = 0x01;
//Destination start address,
    DMAnDSA = &U1TXB;
//Start trigger source U1TX. Refer the datasheet for the correct code
    DMAnSIRQ = 0xnn;
//Change arbiter priority if needed and perform lock operation
    DMA1PR = 0x01;        // Change the priority only if needed
    PRLOCK = 0x55;           // This sequence
    PRLOCK = 0xAA;           // is mandatory
    PRLOCKbits.PRLOCKED = 1;  // for DMA operation
//Enable the DMA & the trigger to start DMA transfer
    DMAnCON0 = 0xC0;
}
```

## 16.12 Register Overlay

All DMA instances in this device share the same set of registers. Only one DMA instance is accessible at a time. The value in the DMASELECT register is one less than the selected DMA instance. For example, a DMASELECT value of '0' selects DMA1.

## 16.13 Register Definitions: DMA

### 16.13.1 DMASELECT

**Name:** DMASELECT
**Address:** 0x0E8

DMA Instance Selection Register

Selects which DMA instance is accessed by the DMA registers

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | SLCT[2:0] | | |
| Access | | | | | | R/W | R/W | R/W |
| Reset | | | | | | 0 | 0 | 0 |

**Bits 2:0 – SLCT[2:0]** DMA Instance Selection

| Value | Description |
|---|---|
| n | Shared DMA registers of instance n+1 are selected for read and write operations |

### 16.13.2 DMAnCON0

**Name:** DMAnCON0
**Address:** 0x0FC

DMA Control Register 0

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | EN | SIRQEN | DGO | | | AIRQEN | | XIP |
| Access | R/W | R/W/HC | R/W/HS/HC | | | R/W/HC | | R/HS/HC |
| Reset | 0 | 0 | 0 | | | 0 | | 0 |

**Bit 7 – EN** DMA Module Enable

| Value | Description |
|---|---|
| 1 | Enables module |
| 0 | Disables module |

**Bit 6 – SIRQEN** Start of Transfer Interrupt Request Enable

| Value | Description |
|---|---|
| 1 | Hardware triggers are allowed to start DMA transfers |
| 0 | Hardware triggers are not allowed to start the DMA transfers |

**Bit 5 – DGO** DMA Transaction

| Value | Description |
|---|---|
| 1 | DMA transaction is in progress |
| 0 | DMA transaction is not in progress |

**Bit 2 – AIRQEN** Abort of Transfer Interrupt Request Enable

| Value | Description |
|---|---|
| 1 | Hardware triggers are allowed to abort DMA transfers |
| 0 | Hardware triggers are not allowed to abort the DMA transfers |

**Bit 0 – XIP** Transfer in Progress Status

| Value | Description |
|---|---|
| 1 | The DMA buffer register currently holds contents from a read operation and has not transferred data to the destination |
| 0 | The DMA buffer register is empty or has successfully transferred data to the destination address |

© 2019-2023 Microchip Technology Inc. and its subsidiaries

### 16.13.3 DMAnCON1

**Name:** DMAnCON1
**Address:** 0x0FD

DMA Control Register 1

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | DMODE[1:0] | | DSTP | SMR[1:0] | | SMODE[1:0] | | SSTP |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 7:6 – DMODE[1:0]** Destination Address Mode Selection

| Value | Description |
|---|---|
| 11 | Reserved, do not use |
| 10 | Destination Pointer (DMADPTR) is decremented after each transfer |
| 01 | Destination Pointer (DMADPTR) is incremented after each transfer |
| 00 | Destination Pointer (DMADPTR) remains unchanged after each transfer |

**Bit 5 – DSTP** Destination Counter Reload Stop

| Value | Description |
|---|---|
| 1 | SIRQEN bit is cleared when destination counter reloads |
| 0 | SIRQEN bit is not cleared when destination counter reloads |

**Bits 4:3 – SMR[1:0]** Source Memory Region Selection

| Value | Description |
|---|---|
| 1x | Data EEPROM is selected as the DMA source memory |
| 01 | Program Flash Memory is selected as the DMA source memory |
| 00 | SFR/GPR data space is selected as the DMA source memory |

**Bits 2:1 – SMODE[1:0]** Source Address Mode Selection

| Value | Description |
|---|---|
| 11 | Reserved, do not use |
| 10 | Source Pointer (DMASPTR) is decremented after each transfer |
| 01 | Source Pointer (DMASPTR) is incremented after each transfer |
| 00 | Source Pointer (DMASPTR) remains unchanged after each transfer |

**Bit 0 – SSTP** Source Counter Reload Stop

| Value | Description |
|---|---|
| 1 | SIRQEN bit is cleared when source counter reloads |
| 0 | SIRQEN bit is not cleared when source counter reloads |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

### 16.13.4 DMAnBUF

**Name:** DMAnBUF
**Address:** 0x0E9

DMA Data Buffer Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | BUF[7:0] | | | | | | | |
| Access | R | R | R | R | R | R | R | R |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 7:0 – BUF[7:0]** DMA Data Buffer

| Description |
|---|
| These bits reflect the content of the internal data buffer the DMA peripheral uses to hold the data being moved from the source to destination. |

### 16.13.5 DMAnSSA

**Name:** DMAnSSA
**Address:** 0x0F9

DMA Source Start Address Register

| Bit | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|
| | | | SSA[21:16] | | | | | |
| Access | | | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | | | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | SSA[15:8] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | SSA[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 21:0 – SSA[21:0]** Source Start Address

**Notes:** The individual bytes in this multibyte register can be accessed with the following register names.

1. DMAnSSAU: Accesses the upper most byte [23:16].
2. DMAnSSAH: Accesses the high byte [15:8].
3. DMAnSSAL: Access the low byte [7:0].

### 16.13.6 DMAnSSZ

**Name:** DMAnSSZ
**Address:** 0x0F7

DMA Source Size Register

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | | | | | SSZ[11:8] | | | |
| Access | | | | | R/W | R/W | R/W | R/W |
| Reset | | | | | 0 | 0 | 0 | 0 |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | SSZ[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 11:0 – SSZ[11:0]** Source Message Size

**Notes:** The individual bytes in this multibyte register can be accessed with the following register names.

1. DMAnSSZH: Accesses the high byte [15:8].
2. DMAnSSZL: Access the low byte [7:0].

### 16.13.7 DMAnSCNT

**Name:** DMAnSCNT
**Address:** 0x0F2

DMA Source Count Register

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | | | | | SCNT[11:8] | | | |
| Access | | | | | R/W | R/W | R/W | R/W |
| Reset | | | | | 0 | 0 | 0 | 0 |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | SCNT[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 11:0 – SCNT[11:0]** Current Source Byte Count

**Notes:** The individual bytes in this multibyte register can be accessed with the following register names.

1. DMAnSCNTH: Accesses the high byte [15:8].
2. DMAnSCNTL: Access the low byte [7:0].

### 16.13.8 DMAnSPTR

**Name:** DMAnSPTR
**Address:** 0x0F4

DMA Source Pointer Register

| Bit | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|
| | | | SPTR[21:16] | | | | | |
| Access | | | R | R | R | R | R | R |
| Reset | | | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | SPTR[15:8] | | | | | | | |
| Access | R | R | R | R | R | R | R | R |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | SPTR[7:0] | | | | | | | |
| Access | R | R | R | R | R | R | R | R |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 21:0 – SPTR[21:0]** Current Source Address Pointer

**Notes:** The individual bytes in this multibyte register can be accessed with the following register names.

1. DMAnSPTRU: Accesses the upper most byte [23:16].
2. DMAnSPTRH: Accesses the high byte [15:8].
3. DMAnSPTRL: Access the low byte [7:0].

### 16.13.9 DMAnDSA

**Name:** DMAnDSA
**Address:** 0x0F0

DMA Destination Start Address Register

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | DSA[15:8] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | DSA[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 15:0 – DSA[15:0]** Destination Start Address

**Notes:** The individual bytes in this multibyte register can be accessed with the following register names.

1. DMAnDSAH: Accesses the high byte [15:8].
2. DMAnDSAL: Access the low byte [7:0].

### 16.13.10 DMAnDSZ

**Name:** DMAnDSZ
**Address:** 0x0EE

DMA Destination Size Register

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | | | | | DSZ[11:8] | | | |
| Access | | | | | R/W | R/W | R/W | R/W |
| Reset | | | | | 0 | 0 | 0 | 0 |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | DSZ[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 11:0 – DSZ[11:0]** Destination Message Size

**Notes:** The individual bytes in this multibyte register can be accessed with the following register names.

1. DMAnDSZH: Accesses the high byte [15:8].
2. DMAnDSZL: Access the low byte [7:0].

### 16.13.11 DMAnDCNT

**Name:** DMAnDCNT
**Address:** 0x0EA

DMA Destination Count Register

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | | | | | DCNT[11:8] | | | |
| Access | | | | | R/W | R/W | R/W | R/W |
| Reset | | | | | 0 | 0 | 0 | 0 |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | DCNT[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 11:0 – DCNT[11:0]** Current Destination Byte Count

**Notes:** The individual bytes in this multibyte register can be accessed with the following register names.

1. DMAnDCNTH: Accesses the high byte [15:8].
2. DMAnDCNTL: Access the low byte Destination Message Size bits [7:0].

### 16.13.12 DMAnDPTR

**Name:** DMAnDPTR
**Address:** 0x0EC

DMA Destination Pointer Register

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | DPTR[15:8] | | | | | | | |
| Access | R | R | R | R | R | R | R | R |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | DPTR[7:0] | | | | | | | |
| Access | R | R | R | R | R | R | R | R |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 15:0 – DPTR[15:0]** Current Destination Address Pointer

**Notes:** The individual bytes in this multibyte register can be accessed with the following register names.

1. DMAnDPTRH: Accesses the high byte [15:8].
2. DMAnDPTRL: Access the low byte [7:0].

### 16.13.13 DMAnSIRQ

**Name:** DMAnSIRQ
**Address:** 0x0FF

DMA Start Interrupt Request Source Selection Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | SIRQ[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 7:0 – SIRQ[7:0]** DMA Start Interrupt Request Source Selection

**Table 16-6. DMA Start and Abort Interrupt Sources**

| Vector Number | Interrupt source | Vector Number (cont.) | Interrupt source (cont.) |
|---|---|---|---|
| 0x0 | - | 0x42 | U2E |
| 0x1 | HLVD (High/Low-Voltage Detect) | 0x43 | U2 |
| 0x2 | OSF (Oscillator Fail) | 0x44 | TMR5 |
| 0x3 | CSW (Clock Switching) | 0x45 | TMR5G |
| 0x4 | - | 0x46 | CCP2 |
| 0x5 | CLC1 (Configurable Logic Cell) | 0x47 | SCAN |
| 0x6 | - | 0x48 | U3RX |
| 0x7 | IOC (Interrupt-On-Change) | 0x49 | U3TX |
| 0x8 | INT0 | 0x4A | U3E |
| 0x9 | ZCD (Zero-Cross Detection) | 0x4B | U3 |
| 0xA | AD (ADC Conversion Complete) | 0x4C | - |
| 0xB | ACT (Active Clock Tuning) | 0x4D | CLC4 |
| 0xC | CM1 (Comparator) | 0x4E - 0x4F | - |
| 0xD | SMT1 (Signal Measurement Timer) | 0x50 | INT2 |
| 0xE | SMT1PRA | 0x51 | CLC5 |
| 0xF | SMT1PWA | 0x52 | CWG2 (Complementary Waveform Generator) |
| 0x10 | ADT (ADC Threshold Interrupt) | 0x53 | NCO2 |
| 0x11 - 0x13 | - | 0x54 | DMA3SCNT |
| 0x14 | DMA1SCNT (Direct Memory Access) | 0x55 | DMA3DCNT |
| 0x15 | DMA1DCNT | 0x56 | DMA3OR |
| 0x16 | DMA1OR | 0x57 | DMA3A |
| 0x17 | DMA1A | 0x58 | CCP3 |
| 0x18 | SPI1RX (Serial Peripheral Interface) | 0x59 | CLC6 |
| 0x19 | SPI1TX | 0x5A | CWG3 |
| 0x1A | SPI1 | 0x5B | TMR4 |
| 0x1B | TMR2 | 0x5C | DMA4SCNT |
| 0x1C | TMR1 | 0x5D | DMA4DCNT |
| 0x1D | TMR1G | 0x5E | DMA4OR |
| 0x1E | CCP1 (Capture/Compare/PWM) | 0x5F | DMA4A |
| 0x1F | TMR0 | 0x60 | U4RX |
| 0x20 | U1RX | 0x61 | U4TX |
| 0x21 | U1TX | 0x62 | U4E |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

.....continued

| Vector Number | Interrupt source | Vector Number (cont.) | Interrupt source (cont.) |
|---|---|---|---|
| 0x22 | U1E | 0x63 | U4 |
| 0x23 | U1 | 0x64 | DMA5SCNT |
| 0x24 - 0x25 | - | 0x65 | DMA5DCNT |
| 0x26 | PWM1RINT | 0x66 | DMA5OR |
| 0x27 | PWM1GINT | 0x67 | DMA5A |
| 0x28 | SPI2RX | 0x68 | U5RX |
| 0x29 | SPI2TX | 0x69 | U5TX |
| 0x2A | SPI2 | 0x6A | U5E |
| 0x2B | - | 0x6B | U5 |
| 0x2C | TMR3 | 0x6C | DMA6SCNT |
| 0x2D | TMR3G | 0x6D | DMA6DCNT |
| 0x2E | PWM2RINT | 0x6E | DMA6OR |
| 0x2F | PWM2GINT | 0x6F | DMA6A |
| 0x30 | INT1 | 0x70 | - |
| 0x31 | CLC2 | 0x71 | CLC7 |
| 0x32 | CWG1 (Complementary Waveform Generator) | 0x72 | CM2 |
| 0x33 | NCO1 (Numerically Controlled Oscillator) | 0x73 | NCO3 |
| 0x34 | DMA2SCNT | 0x74 - 0x77 | - |
| 0x35 | DMA2DCNT | 0x78 | NVM |
| 0x36 | DMA2OR | 0x79 | CLC8 |
| 0x37 | DMA2A | 0x7A | CRC (Cyclic Redundancy Check) |
| 0x38 | I2C1RX | 0x7B | TMR6 |
| 0x39 | I2C1TX | 0x7C - 0x8F | - |
| 0x3A | I2C1 | 0x90 | PWM1.S1P1 (PWM1 Parameter 1 of Slice 1) |
| 0x3B | I2C1E | 0x91 | PWM1.S1P2 (PWM1 Parameter 2 of Slice 1) |
| 0x3C | - | 0x92 | PWM1S2P1 |
| 0x3D | CLC3 | 0x93 | PWM1S2P2 |
| 0x3E | PWM3RINT | 0x94 | PWM1S3P1 |
| 0x3F | PWM3GINT | 0x95 | PWM1S3P2 |
| 0x40 | U2RX | 0x96 | - |
| 0x41 | U2TX | 0x97 | - |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

### 16.13.14 DMAnAIRQ

**Name:** DMAnAIRQ
**Address:** 0x0FE

DMA Abort Interrupt Request Source Selection Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | AIRQ[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 7:0 – AIRQ[7:0]** DMA Abort Interrupt Request Source Selection
Refer to the DMA Interrupt Sources table.

## 16.14 Register Summary - DMA

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0xE8 | DMASELECT | 7:0 | | | | | | SLCT[2:0] | | |
| 0xE9 | DMAnBUF | 7:0 | BUF[7:0] | | | | | | | |
| 0xEA | DMAnDCNT | 7:0 | DCNT[7:0] | | | | | | | |
| | | 15:8 | | | | | DCNT[11:8] | | | |
| 0xEC | DMAnDPTR | 7:0 | DPTR[7:0] | | | | | | | |
| | | 15:8 | DPTR[15:8] | | | | | | | |
| 0xEE | DMAnDSZ | 7:0 | DSZ[7:0] | | | | | | | |
| | | 15:8 | | | | | DSZ[11:8] | | | |
| 0xF0 | DMAnDSA | 7:0 | DSA[7:0] | | | | | | | |
| | | 15:8 | DSA[15:8] | | | | | | | |
| 0xF2 | DMAnSCNT | 7:0 | SCNT[7:0] | | | | | | | |
| | | 15:8 | | | | | SCNT[11:8] | | | |
| 0xF4 | DMAnSPTR | 7:0 | SPTR[7:0] | | | | | | | |
| | | 15:8 | SPTR[15:8] | | | | | | | |
| | | 23:16 | | | SPTR[21:16] | | | | | |
| 0xF7 | DMAnSSZ | 7:0 | SSZ[7:0] | | | | | | | |
| | | 15:8 | | | | | SSZ[11:8] | | | |
| 0xF9 | DMAnSSA | 7:0 | SSA[7:0] | | | | | | | |
| | | 15:8 | SSA[15:8] | | | | | | | |
| | | 23:16 | | | SSA[21:16] | | | | | |
| 0xFC | DMAnCON0 | 7:0 | EN | SIRQEN | DGO | | | AIRQEN | | XIP |
| 0xFD | DMAnCON1 | 7:0 | DMODE[1:0] | | DSTP | SMR[1:0] | | SMODE[1:0] | | SSTP |
| 0xFE | DMAnAIRQ | 7:0 | AIRQ[7:0] | | | | | | | |
| 0xFF | DMAnSIRQ | 7:0 | SIRQ[7:0] | | | | | | | |

# 17. Power-Saving Modes

The purpose of the Power-Saving modes is to reduce power consumption. There are three Power-Saving modes:

- Doze mode
- Sleep mode
- Idle mode

## 17.1 Doze Mode

Doze mode allows for power saving by reducing CPU operation and Program Flash Memory (PFM) access, without affecting peripheral operation. Doze mode differs from Sleep mode because the band gap and system oscillators continue to operate, while only the CPU and PFM are affected. The reduced execution saves power by eliminating unnecessary operations within the CPU and memory.

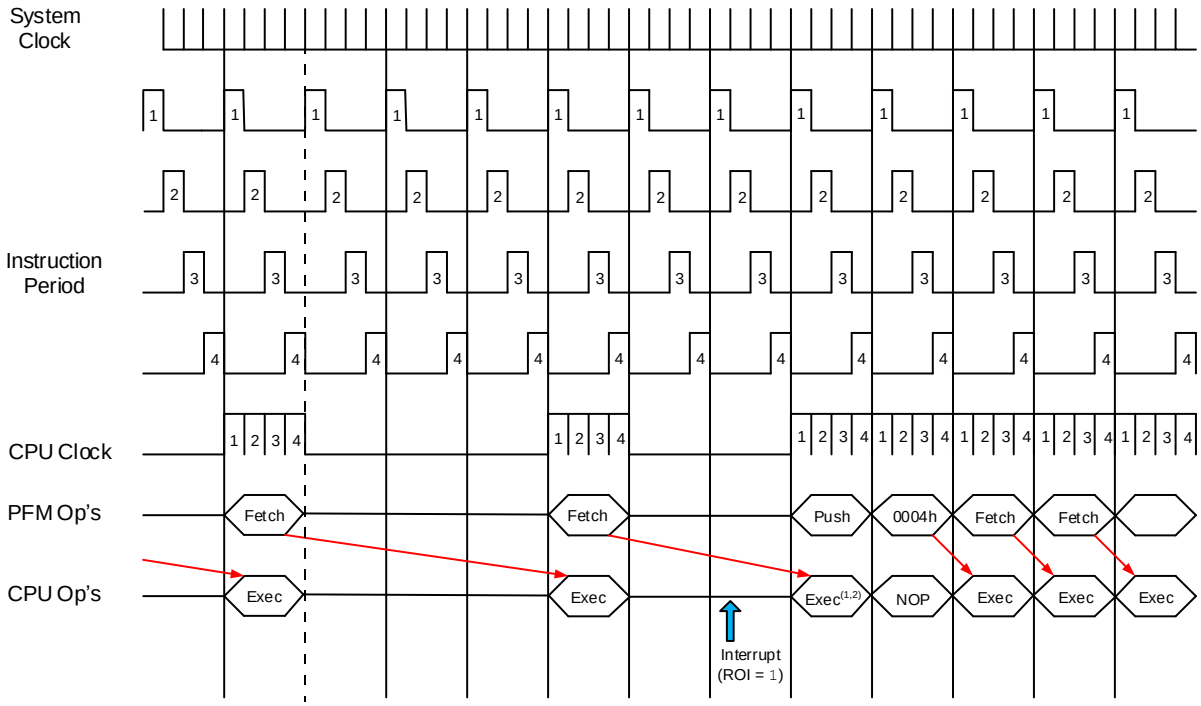
When the Doze Enable bit is set (DOZEN = 'b1) the CPU executes only one instruction cycle out of every N cycles as defined by the DOZE bits. For example, if DOZE = 001, the instruction cycle ratio is 1:4. The CPU and memory execute for one instruction cycle and then lay Idle for three instruction cycles. During the unused cycles, the peripherals continue to operate at the system clock speed.

### 17.1.1 Doze Operation

The Doze operation is illustrated in Figure 17-1. As with normal operation, the instruction is fetched for the next instruction cycle while the previous instruction is executed. The Q-clocks to the peripherals continue throughout the periods in which no instructions are fetched or executed. The following configuration settings apply for this example:

- Doze enabled (DOZEN = 1)
- CPU instruction cycle to peripheral instruction cycle ratio of 1:4
- Recover-on-Interrupt enabled (ROI = 1)

**Figure 17-1. Doze Mode Operation Example**



**Notes:**

1. Multicycle instructions are executed to completion before fetching `0x0004`.
2. If the prefetched instruction clears GIE, the ISR will not occur, but DOZEN is still cleared and the CPU will resume execution at full speed.

### 17.1.2 Interrupts During Doze

System behavior for interrupts that may occur during Doze mode are configured using the ROI and DOE bits. Refer to the example below for details about system behavior in all cases for a transition from Main to ISR back to Main.

**Example 17-1. Doze Software Example**

```
// Mainline operation
bool somethingToDo = FALSE;

void main() {
    initializeSystem();
    // DOZE = 64:1 (for example)
    // ROI = 1;
    GIE = 1; // enable interrupts
    while (1) {
        // If ADC completed, process data
        if (somethingToDo) {
            doSomething();
            DOZEN = 1; // resume low-power
        }
    }
}
// Data interrupt handler

void interrupt() {
    // DOZEN = 0 because ROI = 1
    if (ADIF) {
        somethingToDo = TRUE;
        DOE = 0; // make main() go fast
        ADIF = 0;
    }
    // else check other interrupts...
    if (TMR0IF) {
        timerTick++;
        DOE = 1; // make main() go slow
        TMR0IF = 0;
    }
}
```

**Note:** User software can change the DOE bit in the ISR.

## 17.2 Sleep Mode

Sleep mode provides the greatest power savings because both the CPU and selected peripherals cease to operate. However, some peripheral clocks continue to operate during Sleep. The peripherals that use those clocks also continue to operate. Sleep mode is entered by executing the `SLEEP` instruction, while the IDLEN bit is clear. Upon entering Sleep mode, the following conditions exist:

1. The WDT will be cleared, but keeps running if enabled for operation during Sleep.
2. The $\overline{PD}$ bit of the STATUS register is cleared.
3. The $\overline{TO}$ bit of the STATUS register is set.
4. The CPU clock is disabled.
5. LFINTOSC, SOSC, HFINTOSC and ADCRC are unaffected. Peripherals using them may continue operation during Sleep.
6. I/O ports maintain the status they had before Sleep was executed (driving high, low, or high-impedance).
7. Resets other than WDT are not affected by Sleep mode.

**Important:** Refer to individual chapters for more details on peripheral operation during Sleep.

To minimize current consumption, consider the following conditions:

- I/O pins must not be floating
- External circuitry sinking current from I/O pins
- Internal circuitry sourcing current to I/O pins
- Current draw from pins with internal weak pull-ups
- Peripherals using clock source unaffected by Sleep

I/O pins that are high-impedance inputs need to be pulled to $V_{DD}$ or $V_{SS}$ externally to avoid switching currents caused by floating inputs. Examples of internal circuitry that might be consuming current include modules such as the DAC and FVR peripherals.

### 17.2.1 Wake-Up from Sleep

The device can wake up from Sleep through one of the following events:

1. External Reset input on $\overline{MCLR}$ pin, if enabled.
2. BOR Reset, if enabled.
3. Low-Power Brown-out Reset (LPBOR), if enabled.
4. POR Reset.
5. Windowed Watchdog Timer, if enabled.
6. All interrupt sources except clock switch interrupt can wake up the part.

**Important:** The first five events will cause a device Reset. The last event in the list is considered a continuation of program execution. Fore more information about determining whether a device Reset or wake-up event occurred, refer to the **"Resets"** chapter.

When the `SLEEP` instruction is being executed, the next instruction (PC + 2) is prefetched. For the device to wake up through an interrupt event, the corresponding Interrupt Enable bit must be enabled in the PIEx register. Wake-up will occur regardless of the state of the Global Interrupt Enable (GIE) bit. If the GIE bit is disabled, the device will continue execution at the instruction after the `SLEEP` instruction. If the GIE bit is enabled, the device executes the instruction after the `SLEEP` instruction and then call the Interrupt Service Routine (ISR).

**Important:** It is recommended to add a `NOP` as the immediate instruction after the `SLEEP` instruction.

The WDT is cleared when the device wakes up from Sleep, regardless of the source of wake-up. Upon a wake-from-Sleep event, the core will wait for a combination of three conditions before beginning execution. The conditions are:

- PFM Ready
- System Clock Ready
- BOR Ready (unless BOR is disabled)

### 17.2.2 Wake-Up Using Interrupts

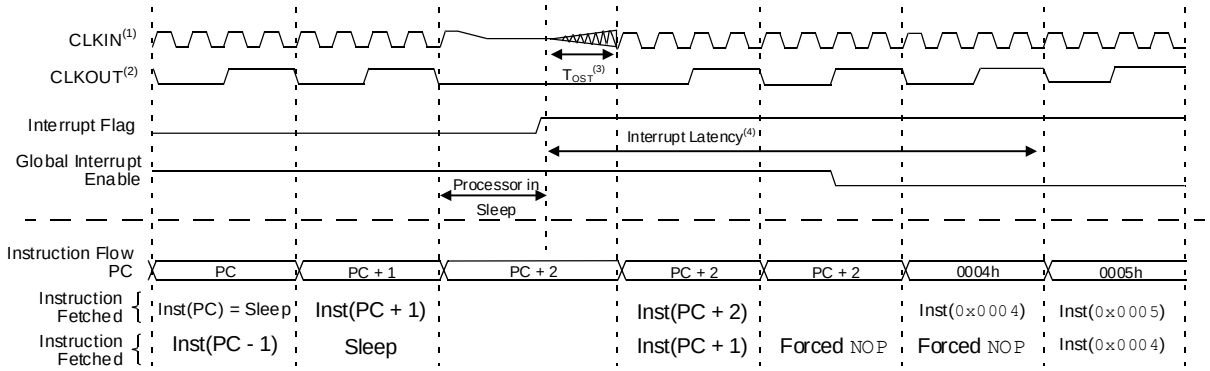
When global interrupts are disabled (GIE cleared) and any interrupt source, with the exception of the clock switch interrupt, has both its interrupt enable bit and interrupt flag bit set, one of the following will occur:

- If the interrupt occurs before the execution of a `SLEEP` instruction:

  - The SLEEP instruction will execute as a NOP
  - WDT and WDT prescaler will not be cleared
  - The $\overline{TO}$ bit of the STATUS register will not be set
  - The $\overline{PD}$ bit of the STATUS register will not be cleared
- If the interrupt occurs during or after the execution of a SLEEP instruction:
  - The SLEEP instruction will be completely executed
  - Device will immediately wake up from Sleep
  - WDT and WDT prescaler will be cleared
  - The $\overline{TO}$ bit of the STATUS register will be set
  - The $\overline{PD}$ bit of the STATUS register will be cleared

In the event where flag bits were checked before executing a SLEEP instruction, it may be possible for flag bits to become set before the SLEEP instruction completes. To determine whether a SLEEP instruction executed, test the $\overline{PD}$ bit. If the $\overline{PD}$ bit is set, the SLEEP instruction was executed as a NOP.

**Figure 17-2. Wake-Up from Sleep through Interrupt**




**Notes:**

1. External clock - High, Medium, Low mode assumed.
2. CLKOUT is shown here for timing reference.
3. $T_{OST}$ = 1024 $T_{OSC}$. This delay does not apply to EC and INTOSC Oscillator modes.
4. GIE = 1 assumed. In this case after wake-up, the processor calls the ISR at 0x0004. If GIE = 0, execution will continue in-line.

### 17.2.3 Low-Power Sleep Mode

This device family contains an internal Low Dropout (LDO) voltage regulator, which allows the device I/O pins to operate at voltages up to $V_{DD}$ while the internal device logic operates at a lower voltage. The LDO and its associated reference circuitry must remain active in Sleep but can operate in different Power modes. This allows the user to optimize the operating current in Sleep mode, depending on the application requirements.

#### 17.2.3.1 Sleep Current vs. Wake-Up Time

The Low-Power Sleep mode can be selected by setting the VREGPM bits as following:

- VREGPM = 'b00; the voltage regulator is in High Power mode. In this mode, the voltage regulator and reference circuitry remain in the normal configuration while in Sleep. Hence, there is no delay needed for these circuits to stabilize after wake-up (fastest wake-up from Sleep).
- VREGPM = 'b01; the voltage regulator is in Low Power mode. In this mode, when waking up from Sleep, an extra delay time is required for the voltage regulator and reference circuitry to return to the normal configuration and stabilize (faster wake-up from Sleep).
- VREGPM = 'b10; the voltage regulator is in Ultra-Low Power mode. In this mode, the voltage regulator and reference circuitry are in the lowest current consumption mode and all the auxiliary circuits remain shut down. Wake-up from Sleep in this mode needs the longest delay time for the voltage regulator and reference circuitry to stabilize (lowest current consumption).

- VREGPM = 'b11; this mode is the similar to the Ultra-Low Power mode (VREGPM = 'b10), and is recommended ONLY for extended temperature ranges at or above 70℃.

#### 17.2.3.2 Peripheral Usage in Sleep

Some peripherals that can operate in High-Power Sleep mode (VREGPM = 'b00) will not operate as intended in the Low-Power Sleep modes (VREGPM = 'b01 and 'b11). The Low-Power Sleep modes are intended for use with the following peripherals:

- Brown-out Reset (BOR)
- Windowed Watchdog Timer (WWDT)
- External interrupt pin/interrupt-on-change pins

It is the responsibility of the end user to determine what is acceptable for their application when setting the VREGPM settings to ensure correct operation in Sleep.

## 17.3 Idle Mode

When the IDLEN bit is clear, the SLEEP instruction will put the device into full Sleep mode. When IDLEN is set, the SLEEP instruction will put the device into Idle mode. In Idle mode, the CPU and memory operations are halted, but the peripheral clocks continue to run. This mode is similar to Doze mode, except that in Idle both the CPU and program memory are shut off.

**Important:**

1. Peripherals using $F_{OSC}$ will continue to operate while in Idle (but not in Sleep). Peripherals using HFINTOSC:LFINTOSC will continue running in both Idle and Sleep.
2. When the Clock Out Enable ($\overline{CLKOUTEN}$) Configuration bit is cleared, the CLKOUT pin will continue operating while in Idle.

### 17.3.1 Idle and Interrupts

Idle mode ends when an interrupt occurs (even if global interrupts are disabled), but IDLEN is not changed. The device can re-enter Idle by executing the SLEEP instruction. If Recover-on-Interrupt is enabled (ROI = 1), the interrupt that brings the device out of Idle also restores full-speed CPU execution when Doze is also enabled.

### 17.3.2 Idle and WWDT

When in Idle, the WWDT Reset is blocked and will instead wake the device. The WWDT wake-up is not an interrupt, therefore ROI does not apply.

**Important:** The WWDT can bring the device out of Idle, in the same way it brings the device out of Sleep. The DOZEN bit is not affected.

## 17.4 Peripheral Operation in Power-Saving Modes

All selected clock sources and the peripherals running from them are active in both Idle and Doze modes. Only in Sleep mode, both the $F_{OSC}$ and $F_{OSC}/4$ clocks are unavailable. However, all other clock sources enabled specifically or through peripheral clock selection before the part enters Sleep, remain operating in Sleep.

## 17.5 Register Definitions: Power-Savings Control

### 17.5.1 CPUDOZE

**Name:** CPUDOZE
**Address:** 0x4F2

Doze and Idle Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | IDLEN | DOZEN | ROI | DOE | | DOZE[2:0] | | |
| Access | R/W | R/W/HC/HS | R/W | R/W/HC/HS | | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | | 0 | 0 | 0 |

**Bit 7 – IDLEN** Idle Enable

| Value | Description |
|---|---|
| 1 | A SLEEP instruction places device into Idle mode |
| 0 | A SLEEP instruction places the device into Sleep mode |

**Bit 6 – DOZEN** Doze Enable(1)

| Value | Description |
|---|---|
| 1 | Places devices into Doze setting |
| 0 | Places devices into Normal mode |

**Bit 5 – ROI** Recover-on-Interrupt(1)

| Value | Description |
|---|---|
| 1 | Entering the Interrupt Service Routine (ISR) makes DOZEN = 0 |
| 0 | Entering the Interrupt Service Routine (ISR) does not change DOZEN |

**Bit 4 – DOE** Doze-on-Exit(1)

| Value | Description |
|---|---|
| 1 | Exiting the ISR makes DOZEN = 1 |
| 0 | Exiting the ISR does not change DOZEN |

**Bits 2:0 – DOZE[2:0]** Ratio of CPU Instruction Cycles to Peripheral Instruction Cycles

| Value | Description |
|---|---|
| 111 | 1:256 |
| 110 | 1:128 |
| 101 | 1:64 |
| 100 | 1:32 |
| 011 | 1:16 |
| 010 | 1:8 |
| 001 | 1:4 |
| 000 | 1:2 |

**Note:**

1. When ROI = 1 or DOE = 1.

### 17.5.2 VREGCON

**Name:** VREGCON
**Address:** 0x048

Voltage Regulator Control Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | PMSYS[1:0] | | | | VREGPM[1:0] | |
| Access | | | R | R | | | R/W | R/W |
| Reset | | | q | q | | | 1 | 0 |

**Bits 5:4 – PMSYS[1:0]** System Power Mode Status

| Value | Description |
|---|---|
| 11 | Regulator in Ultra-Low Power (ULP) mode for extended temperature range is active |
| 10 | Regulator in Ultra-Low Power (ULP) mode is active |
| 01 | Regulator in Low Power (LP) mode is active |
| 00 | Regulator in High Power (HP) mode is active |

**Bits 1:0 – VREGPM[1:0]** Voltage Regulator Power Mode Selection

| Value | Description |
|---|---|
| 11 | Regulator in Ultra-Low Power (ULP) mode. Use ONLY for extended temperature range |
| 10 | Regulator in Ultra-Low Power (ULP) mode (lowest current consumption) |
| 01 | Regulator in Low Power (LP) mode (faster wake-up from Sleep) |
| 00 | Regulator in High Power (HP) mode (fastest wake-up from Sleep) |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

## 17.6 Register Summary - Power-Savings Control

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x48 | VREGCON | 7:0 | | | PMSYS[1:0] | | | | VREGPM[1:0] | |
| 0x49<br>...<br>0x04F1 | Reserved | | | | | | | | | |
| 0x04F2 | CPUDOZE | 7:0 | IDLEN | DOZEN | ROI | DOE | | DOZE[2:0] | | |

# 18. PMD - Peripheral Module Disable

## 18.1 Overview

This module provides the ability to selectively enable or disable a peripheral. Disabling a peripheral places it in its lowest possible Power state. The user can selectively disable unused modules to reduce the overall power consumption.

**Important:** All modules are ON by default following any system Reset.

## 18.2 Disabling a Module

A peripheral can be disabled by setting the corresponding peripheral disable bit in the PMDx register. Disabling a module has the following effects:

- The module is held in Reset and does not function.
- All the SFRs pertaining to that peripheral become “unimplemented”
  - Writing is disabled
  - Reading returns `0x00`
- Module outputs are disabled

## 18.3 Enabling a Module

Clearing the corresponding module disable bit in the PMDx register, re-enables the module and the SFRs will reflect the Power-on Reset values.

**Important:** There will be no reads/writes to the module SFRs for at least two instruction cycles after it has been re-enabled.

## 18.4 Register Definitions: Peripheral Module Disable

© 2019-2023 Microchip Technology Inc. and its subsidiaries

### 18.4.1 PMD0

**Name:** PMD0
**Address:** 0x060

PMD Control Register 0

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | SYSCMD | FVRMD | HLVDMD | CRCMD | SCANMD | | CLKRMD | IOCMD |
| Access | R/W | R/W | R/W | R/W | R/W | | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | | 0 | 0 |

**Bit 7 – SYSCMD** Disable Peripheral System Clock Network(1)

| Value | Description |
|---|---|
| 1 | System clock network disabled ($F_{OSC}$) |
| 0 | System clock network enabled |

**Bit 6 – FVRMD** Disable Fixed Voltage Reference
Disable Fixed Voltage Reference

| Value | Description |
|---|---|
| 1 | FVR module disabled |
| 0 | FVR module enabled |

**Bit 5 – HLVDMD** Disable High/Low-Voltage Detect

| Value | Description |
|---|---|
| 1 | HLVD module disabled |
| 0 | HLVD module enabled |

**Bit 4 – CRCMD** Disable CRC Module

| Value | Description |
|---|---|
| 1 | CRC module disabled |
| 0 | CRC module enabled |

**Bit 3 – SCANMD** Disable NVM Memory Scanner

| Value | Description |
|---|---|
| 1 | NVM memory scanner module disabled |
| 0 | NVM memory scanner module enabled |

**Bit 1 – CLKRMD** Disable Clock Reference

| Value | Description |
|---|---|
| 1 | Clock reference module disabled |
| 0 | Clock reference module enabled |

**Bit 0 – IOCMD** Disable Interrupt-on-Change

| Value | Description |
|---|---|
| 1 | Interrupt-on-change module is disabled |
| 0 | Interrupt-on-change module is enabled |

**Note:**

1. Clearing the SYSCMD bit disables the system clock ($F_{OSC}$) to peripherals, however peripherals clocked by $F_{OSC}/4$ are not affected.

### 18.4.2 PMD1

**Name:** PMD1
**Address:** 0x061

PMD Control Register 1

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | SMT1MD | TMR6MD | TMR5MD | TMR4MD | TMR3MD | TMR2MD | TMR1MD | TMR0MD |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bit 7 – SMT1MD** Disable SMT1 Module

| Value | Description |
|---|---|
| 1 | SMT1 module disabled |
| 0 | SMT1 module enabled |

**Bits 0, 1, 2, 3, 4, 5, 6 – TMRnMD** Disable Timer TMRn

| Value | Description |
|---|---|
| 1 | TMRn module disabled |
| 0 | TMRn module enabled |

### 18.4.3 PMD3

**Name:** PMD3
**Address:** 0x063

PMD Control Register 3

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | ACTMD | DAC1MD | ADCMD | | | C2MD | C1MD | ZCDMD |
| Access | R/W | R/W | R/W | | | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | | | 0 | 0 | 0 |

**Bit 7 – ACTMD** Disable Active Clock Tuning

| Value | Description |
|---|---|
| 1 | Active Clock Tuning disabled |
| 0 | Active Clock Tuning enabled |

**Bit 6 – DAC1MD** Disable Digital-to-Analog Converter

| Value | Description |
|---|---|
| 1 | DAC module disabled |
| 0 | DAC module enabled |

**Bit 5 – ADCMD** Disable Analog-to-Digital Converter

| Value | Description |
|---|---|
| 1 | ADC module disabled |
| 0 | ADC module enabled |

**Bit 2 – C2MD** Disable Comparator 2

| Value | Description |
|---|---|
| 1 | CM2 module disabled |
| 0 | CM2 module enabled |

**Bit 1 – C1MD** Disable Comparator 1

| Value | Description |
|---|---|
| 1 | CM1 module disabled |
| 0 | CM1 module enabled |

**Bit 0 – ZCDMD** Disable Zero Cross Detect(1)

| Value | Description |
|---|---|
| 1 | ZCD module disabled |
| 0 | ZCD module enabled |

**Note:**

1. Subject to the value of $\overline{ZCD}$ Configuration bit.

### 18.4.4 PMD4

**Name:** PMD4
**Address:** 0x064

PMD Control Register 4

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | CWG3MD | CWG2MD | CWG1MD | DSM1MD | NCO3MD | NCO2MD | NCO1MD |
| Access | | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bit 6 – CWG3MD** Disable Complementary Waveform Generator 3

| Value | Description |
|---|---|
| 1 | CWG3 module disabled |
| 0 | CWG3 module enabled |

**Bit 5 – CWG2MD** Disable Complementary Waveform Generator 2

| Value | Description |
|---|---|
| 1 | CWG2 module disabled |
| 0 | CWG2 module enabled |

**Bit 4 – CWG1MD** Disable Complementary Waveform Generator 1

| Value | Description |
|---|---|
| 1 | CWG1 module disabled |
| 0 | CWG1 module enabled |

**Bit 3 – DSM1MD** Disable Digital Signal Modulator

| Value | Description |
|---|---|
| 1 | DSM module disabled |
| 0 | DSM module enabled |

**Bit 2 – NCO3MD** Disable Numerically Controlled Oscillator 3

| Value | Description |
|---|---|
| 1 | NCO3 module disabled |
| 0 | NCO3 module enabled |

**Bit 1 – NCO2MD** Disable Numerically Controlled Oscillator 2

| Value | Description |
|---|---|
| 1 | NCO2 module disabled |
| 0 | NCO2 module enabled |

**Bit 0 – NCO1MD** Disable Numerically Controlled Oscillator 1

| Value | Description |
|---|---|
| 1 | NCO1 module disabled |
| 0 | NCO1 module enabled |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

### 18.4.5 PMD5

**Name:** PMD5
**Address:** 0x065

PMD Control Register 5

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | PWM3MD | PWM2MD | PWM1MD | | CCP3MD | CCP2MD | CCP1MD |
| Access | | R/W | R/W | R/W | | R/W | R/W | R/W |
| Reset | | 0 | 0 | 0 | | 0 | 0 | 0 |

**Bit 6 – PWM3MD** Disable Pulse-Width Modulator 3

| Value | Description |
|---|---|
| 1 | PWM3 module disabled |
| 0 | PWM3 module enabled |

**Bit 5 – PWM2MD** Disable Pulse-Width Modulator 2

| Value | Description |
|---|---|
| 1 | PWM2 module disabled |
| 0 | PWM2 module enabled |

**Bit 4 – PWM1MD** Disable Pulse-Width Modulator 1

| Value | Description |
|---|---|
| 1 | PWM1 module disabled |
| 0 | PWM1 module enabled |

**Bit 2 – CCP3MD** Disable Capture Compare 3

| Value | Description |
|---|---|
| 1 | CCP3 module disabled |
| 0 | CCP3 module enabled |

**Bit 1 – CCP2MD** Disable Capture Compare 2

| Value | Description |
|---|---|
| 1 | CCP2 module disabled |
| 0 | CCP2 module enabled |

**Bit 0 – CCP1MD** Disable Capture Compare 1

| Value | Description |
|---|---|
| 1 | CCP1 module disabled |
| 0 | CCP1 module enabled |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

### 18.4.6 PMD6

**Name:** PMD6
**Address:** 0x066

PMD Control Register 6

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | U5MD | U4MD | U3MD | U2MD | U1MD | SPI2MD | SPI1MD | I2C1MD |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 3, 4, 5, 6, 7 – UnMD** Disable UART Un

| Value | Description |
|---|---|
| 1 | UARTn module disabled |
| 0 | UARTn module enabled |

**Bit 2 – SPI2MD** Disable Serial Peripheral Interface 2

| Value | Description |
|---|---|
| 1 | SPI2 module disabled |
| 0 | SPI2 module enabled |

**Bit 1 – SPI1MD** Disable Serial Peripheral Interface 1

| Value | Description |
|---|---|
| 1 | SPI1 module disabled |
| 0 | SPI1 module enabled |

**Bit 0 – I2C1MD** Disable $I^2C$

| Value | Description |
|---|---|
| 1 | $I^2C1$ module disabled |
| 0 | $I^2C1$ module enabled |

### 18.4.7 PMD7

**Name:** PMD7
**Address:** 0x067

PMD Control Register 7

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | CLC8MD | CLC7MD | CLC6MD | CLC5MD | CLC4MD | CLC3MD | CLC2MD | CLC1MD |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 0, 1, 2, 3, 4, 5, 6, 7 – CLCnMD** Disable CLCn

| Value | Description |
|---|---|
| 1 | CLCn module disabled |
| 0 | CLCn module enabled |

### 18.4.8 PMD8

**Name:** PMD8
**Address:** 0x068

PMD Control Register 8

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | DMA6MD | DMA5MD | DMA4MD | DMA3MD | DMA2MD | DMA1MD |
| Access | | | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | | | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 0, 1, 2, 3, 4, 5 – DMAnMD** Disable DMAn

| Value | Description |
|---|---|
| 1 | DMAn module disabled |
| 0 | DMAn module enabled |

## 18.5 Register Summary - PMD

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x60 | PMD0 | 7:0 | SYSCMD | FVRMD | HLVDMD | CRCMD | SCANMD | | CLKRMD | IOCMD |
| 0x61 | PMD1 | 7:0 | SMT1MD | TMR6MD | TMR5MD | TMR4MD | TMR3MD | TMR2MD | TMR1MD | TMR0MD |
| 0x62 | Reserved | | | | | | | | | |
| 0x63 | PMD3 | 7:0 | ACTMD | DAC1MD | ADCMD | | | C2MD | C1MD | ZCDMD |
| 0x64 | PMD4 | 7:0 | | CWG3MD | CWG2MD | CWG1MD | DSM1MD | NCO3MD | NCO2MD | NCO1MD |
| 0x65 | PMD5 | 7:0 | | PWM3MD | PWM2MD | PWM1MD | | CCP3MD | CCP2MD | CCP1MD |
| 0x66 | PMD6 | 7:0 | U5MD | U4MD | U3MD | U2MD | U1MD | SPI2MD | SPI1MD | I2C1MD |
| 0x67 | PMD7 | 7:0 | CLC8MD | CLC7MD | CLC6MD | CLC5MD | CLC4MD | CLC3MD | CLC2MD | CLC1MD |
| 0x68 | PMD8 | 7:0 | | | DMA6MD | DMA5MD | DMA4MD | DMA3MD | DMA2MD | DMA1MD |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

# 19. I/O Ports

## 19.1 Overview

**Table 19-1. Port Availability per Device**

| Device | PORTA | PORTB | PORTC | PORTD | PORTE | PORTF |
|---|---|---|---|---|---|---|
| 28-pin devices | • | • | • | | •(1) | |
| 40/44-pin devices | • | • | • | • | •(2) | |
| 48-pin devices | • | • | • | • | •(2) | • |

**Notes:**
1. Pin RE3 only
2. Pins RE0, RE1, RE2 and RE3 only

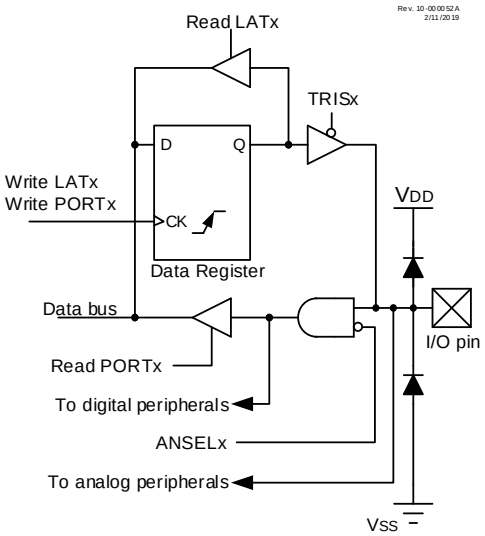
Each port has eight registers to control the operation. These registers are:

- PORTx registers (reads the levels on the pins of the device)
- LATx registers (output latch)
- TRISx registers (data direction)
- ANSELx registers (analog select)
- WPUx registers (weak pull-up)
- INLVLx (input level control)
- SLRCONx registers (slew rate control)
- ODCONx registers (open-drain control)

In this section, the generic names such as PORTx, LATx, TRISx, etc. can be associated with PORTA, PORTB, PORTC, etc., depending on availability per device.

A simplified model of a generic I/O port, without the interfaces to other peripherals, is shown in the following figure:

**Figure 19-1. Generic I/O Port Operation**



## 19.2 PORTx - Data Register

PORTx is a bidirectional port, and its corresponding data direction register is TRISx.

Reading the PORTx register reads the status of the pins, whereas writing to it will write to the PORT latch. All write operations are Read-Modify-Write operations. Therefore, a write to a port implies that the PORT pins are read, and this value is modified, then written to the PORT data latch (LATx). The PORT data latch LATx holds the output port data and contains the latest value of a LATx or PORTx write. The example below shows how to initialize PORTA.

**Example 19-1. Initializing PORTA in Assembly**

```
; This code example illustrates initializing the PORTA register.
; The other ports are initialized in the same manner.

    BANKSEL    PORTA        ;
    CLRF       PORTA        ;Clear PORTA
    BANKSEL    LATA         ;
    CLRF       LATA         ;Clear Data Latch
    BANKSEL    ANSELA       ;
    CLRF       ANSELA       ;Enable digital drivers
    BANKSEL    TRISA        ;
    MOVLW      B'00111000'  ;Set RA[5:3] as inputs
    MOVWF      TRISA        ;and set others as outputs
```

**Example 19-2. Initializing PORTA in C**

```
// This code example illustrates initializing the PORTA register.
// The other ports are initialized in the same manner.

    PORTA  = 0x00;       // Clear PORTA
    LATA   = 0x00;       // Clear Data Latch
    ANSELA = 0x00;       // Enable digital drivers
    TRISA  = 0x38;       // Set RA[5:3] as inputs and set others as outputs
```

**Important:** Most PORT pins share functions with device peripherals, both analog and digital. In general, when a peripheral is enabled on a PORT pin, that pin cannot be used as a general purpose output; however, the pin can still be read.

## 19.3 LATx - Output Latch

The Data Latch (LATx registers) is useful for Read-Modify-Write operations on the value that the I/O pins are driving.

A write operation to the LATx register has the same effect as a write to the corresponding PORTx register. A read of the LATx register reads the values held in the I/O PORT latches, while a read of the PORTx register reads the actual I/O pin value.

**Important:** As a general rule, output operations to a port must use the LAT register to avoid Read-Modify-Write issues. For example, a bit set or clear operation reads the port, modifies the bit, and writes the result back to the port. When two bit operations are executed in succession, output loading on the changed bit may delay the change at the output in which case the bit will be misread in the second bit operation and written to an unexpected level. The LAT registers are isolated from the port loading and therefore changes are not delayed.

© 2019-2023 Microchip Technology Inc. and its subsidiaries

## 19.4 TRISx - Direction Control

The TRISx register controls the PORTx pin output drivers, even when the pins are being used as analog inputs. The user must ensure the bits in the TRISx register are set when using the pins as analog inputs. I/O pins configured as analog inputs always read '0'.

Setting a TRISx bit (TRISx = 1) will make the corresponding PORTx pin an input (i.e., disable the output driver). Clearing a TRISx bit (TRISx = 0) will make the corresponding PORTx pin an output (i.e., it enables output driver and puts the contents of the output latch on the selected pin).

## 19.5 ANSELx - Analog Control

Ports that support analog inputs have an associated ANSELx register. The ANSELx register is used to configure the Input mode of an I/O pin to analog. Setting an ANSELx bit high will disable the digital input buffer associated with that bit and cause the corresponding input value to always read '0', whether the value is read in PORTx register or selected by PPS as a peripheral input.

Disabling the input buffer prevents analog signal levels on the pin between a logic high and low from causing excessive current in the logic input circuitry.

The state of the ANSELx bits has no effect on digital or analog output functions. A pin with TRIS clear and ANSEL set will still operate as a digital output, but the Input mode will be analog. This can cause unexpected behavior when executing Read-Modify-Write instructions on the PORTx register.

**Important:** The ANSELx bits default to the Analog mode after Reset. To use any pins as digital general purpose or peripheral inputs, the corresponding ANSEL bits must be changed to '0' by the user.

## 19.6 WPUx - Weak Pull-Up Control

The WPUx register controls the individual weak pull-ups for each PORT pin. When a WPUx bit is set (WPUx = 1), the weak pull-up will be enabled for the corresponding pin. When a WPUx bit is cleared (WPUx = 0), the weak pull-up will be disabled for the corresponding pin.

## 19.7 INLVLx - Input Threshold Control

The INLVLx register controls the input voltage threshold for each of the available PORTx input pins. A selection between the Schmitt Trigger CMOS or the TTL compatible thresholds is available. The input threshold is important in determining the value of a read of the PORTx register and also the level at which an interrupt-on-change occurs, if that feature is enabled. Refer to the I/O Ports table in the **"Electrical Specifications"** chapter for more details on threshold levels..

**Important:** Changing the input threshold selection must be performed while all peripheral modules are disabled. Changing the threshold level during the time a module is active may inadvertently generate a transition associated with an input pin, regardless of the actual voltage level on that pin.

## 19.8 SLRCONx - Slew Rate Control

The SLRCONx register controls the slew rate option for each PORT pin. Slew rate for each PORT pin can be controlled independently. When a SLRCONx bit is set (SLRCONx = 1), the corresponding PORT pin drive is slew

© 2019-2023 Microchip Technology Inc. and its subsidiaries

rate limited. When a SLRCONx bit is cleared (SLRCONx = 0), the corresponding PORT pin drive slews at the maximum rate possible.

## 19.9 ODCONx - Open-Drain Control

The ODCONx register controls the open-drain feature of the port. Open-drain operation is independently selected for each pin. When a ODCONx bit is set (ODCONx = 1), the corresponding port output becomes an open-drain driver capable of sinking current only. When a ODCONx bit is cleared (ODCONx = 0), the corresponding port output pin is the standard push-pull drive capable of sourcing and sinking current.

**Important:** It is necessary to set open-drain control when using the pin for $I^2C$.

## 19.10 Edge Selectable Interrupt-on-Change

An interrupt can be generated by detecting a signal at the PORT pin that has either a rising edge or a falling edge. Individual pins can be independently configured to generate an interrupt. Refer to the "**IOC - Interrupt-on-Change**" chapter for more details.

## 19.11 $I^2C$ Pad Control

For this family of devices, the $I^2C$ specific pads are available on RB1, RB2, RC3 and RC4 pins. The $I^2C$ characteristics of each of these pins is controlled by the RxyI2C registers. These characteristics include enabling $I^2C$ specific slew rate (over standard GPIO slew rate), selecting internal pull-ups for $I^2C$ pins, and selecting appropriate input threshold as per SMBus specifications.

**Important:** Any peripheral using the $I^2C$ pins reads the $I^2C$ input levels when enabled via RxyI2C.

## 19.12 I/O Priorities

Each pin defaults to the data latch after Reset. Other functions are selected with the Peripheral Pin Select logic. Refer to the **"PPS - Peripheral Pin Select Module"** chapter for more details.

Analog input functions, such as ADC and comparator inputs, are not shown in the Peripheral Pin Select lists. These inputs are active when the I/O pin is set for Analog mode using the ANSELx register. Digital output functions may continue to control the pin when it is in Analog mode.

Analog outputs, when enabled, take priority over digital outputs and force the digital output driver into a High-Impedance state.

The pin function priorities are as follows:

1. Port functions determined by the Configuration bits.
2. Analog outputs (input buffers must be disabled).
3. Analog inputs.
4. Port inputs and outputs from PPS.

© 2019-2023 Microchip Technology Inc. and its subsidiaries

## 19.13 $\overline{MCLR}/V_{PP}$/RE3 Pin

The $\overline{MCLR}/V_{PP}$ pin is an input-only pin. Its operation is controlled by the MCLRE Configuration bit. When selected as a PORT pin (MCLRE = 0), it functions as a digital input-only pin; as such, it does not have TRISx and LATx bits associated with its operation. Otherwise, it functions as the device's Master Clear input. In either configuration, the $\overline{MCLR}/V_{PP}$ pin also functions as the programming voltage input pin during high-voltage programming.

The $\overline{MCLR}/V_{PP}$ pin is a read-only bit and will read '1' when MCLRE = 1 (i.e., Master Clear enabled).

**Important:** On a Power-on Reset (POR), the $\overline{MCLR}/V_{PP}$ pin is enabled as a digital input-only if Master Clear functionality is disabled.

The $\overline{MCLR}/V_{PP}$ pin has an individually controlled internal weak pull-up. When set, the corresponding WPU bit enables the pull-up. When the $\overline{MCLR}/V_{PP}$ pin is configured as $\overline{MCLR}$ (MCLRE = 1 and LVP = 0), or configured for Low-Voltage Programming (MCLRE = x and LVP = 1), the pull-up is always enabled and the WPU bit has no effect.

## 19.14 Register Definitions: Port Control

### 19.14.1 PORTx

**Name:** PORTx

PORTx Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | Rx7 | Rx6 | Rx5 | Rx4 | Rx3 | Rx2 | Rx1 | Rx0 |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | x | x | x | x | x | x | x | x |

**Bits 0, 1, 2, 3, 4, 5, 6, 7 – Rxn** Port I/O Value
Reset States: POR/BOR = xxxxxxxx
All Other Resets = uuuuuuuu

| Value | Description |
|---|---|
| 1 | PORT pin is ≥ $V_{IH}$ |
| 0 | PORT pin is ≤ $V_{IL}$ |

**Important:**

- Writes to PORTx are actually written to the corresponding LATx register. Reads from PORTx register return actual I/O pin values.
- The PORT bit associated with the $\overline{MCLR}$ pin is read-only and will read '1' when the $\overline{MCLR}$ function is enabled (LVP = 1 or (LVP = 0 and MCLRE = 1))
- Refer to the **"Pin Allocation Table"** for details about $\overline{MCLR}$ pin and pin availability per port
- Unimplemented bits will read back as '0'
- Bits RB6 and RB7 read '1' while in Debug mode.

### 19.14.2 LATx

**Name:** LATx

Output Latch Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | LATx7 | LATx6 | LATx5 | LATx4 | LATx3 | LATx2 | LATx1 | LATx0 |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | x | x | x | x | x | x | x | x |

**Bits 0, 1, 2, 3, 4, 5, 6, 7 – LATxn** Output Latch Value
Reset States: POR/BOR = xxxxxxxx
All Other Resets = uuuuuuuu

**Important:**

- Writes to LATx are equivalent to writes to the corresponding PORTx register. Reads from LATx register return register values, not I/O pin values.
- Refer to the **"Pin Allocation Table"** for details about pin availability per port
- Unimplemented bits will read back as '0'

### 19.14.3 TRISx

**Name:** TRISx

Tri-State Control Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | TRISx7 | TRISx6 | TRISx5 | TRISx4 | TRISx3 | TRISx2 | TRISx1 | TRISx0 |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Bits 0, 1, 2, 3, 4, 5, 6, 7 – TRISxn** Port I/O Tri-state Control

| Value | Description |
|---|---|
| 1 | PORTx output driver is disabled. PORTx pin configured as an input (tri-stated) |
| 0 | PORTx output driver is enabled. PORTx pin configured as an output |

**Important:**

- The TRIS bit associated with the $\overline{\text{MCLR}}$ pin is read-only and the value is '1'
- Refer to the **"Pin Allocation Table"** for details about $\overline{\text{MCLR}}$ pin and pin availability per port
- Unimplemented bits will read back as '0'

### 19.14.4 ANSELx

**Name:** ANSELx

Analog Select Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | ANSELx7 | ANSELx6 | ANSELx5 | ANSELx4 | ANSELx3 | ANSELx2 | ANSELx1 | ANSELx0 |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Bits 0, 1, 2, 3, 4, 5, 6, 7 – ANSELxn** Analog Select on RX Pin

| Value | Description |
|---|---|
| 1 | Analog input. Pin is assigned as analog input. Digital input buffer disabled. |
| 0 | Digital I/O. Pin is assigned to port or digital special function. |

**Important:**

- When setting a pin as an analog input, the corresponding TRIS bit must be set to Input mode to allow external control of the voltage on the pin
- Refer to the "**Pin Allocation Table**" for details about pin availability per port
- Unimplemented bits will read back as '0'

### 19.14.5 WPUx

**Name:** WPUx

Weak Pull-Up Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | WPUx7 | WPUx6 | WPUx5 | WPUx4 | WPUx3 | WPUx2 | WPUx1 | WPUx0 |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 0, 1, 2, 3, 4, 5, 6, 7 – WPUxn** Weak Pull-up PORTx Control

| Value | Description |
|---|---|
| 1 | Weak pull-up enabled |
| 0 | Weak pull-up disabled |

**Important:**

- The weak pull-up device is automatically disabled if the pin is configured as an output, but this register remains unchanged
- If MCLRE = 1, the weak pull-up on $\overline{\text{MCLR}}$ pin is always enabled and the corresponding WPU bit is not affected
- Refer to the **"Pin Allocation Table"** for details about pin availability per port
- Unimplemented bits will read back as '0'

### 19.14.6 INLVLx

**Name:** INLVLx

Input Level Control Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | INLVLx7 | INLVLx6 | INLVLx5 | INLVLx4 | INLVLx3 | INLVLx2 | INLVLx1 | INLVLx0 |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Bits 0, 1, 2, 3, 4, 5, 6, 7 – INLVLxn** Input Level Select on RX Pin

| Value | Description |
|---|---|
| 1 | ST input used for port reads and interrupt-on-change |
| 0 | TTL input used for port reads and interrupt-on-change |

**Important:**

- Refer to the "**Pin Allocation Table**" for details about pin availability per port
- Unimplemented bits will read back as '0'
- Any peripheral using the $I^2C$ pins read the $I^2C$ ST inputs when enabled via RxyI2C.

### 19.14.7 SLRCONx

**Name:** SLRCONx

Slew Rate Control Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | SLRx7 | SLRx6 | SLRx5 | SLRx4 | SLRx3 | SLRx2 | SLRx1 | SLRx0 |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Bits 0, 1, 2, 3, 4, 5, 6, 7 – SLRxn** Slew Rate Control on RX Pin

| Value | Description |
|---|---|
| 1 | PORT pin slew rate is limited |
| 0 | PORT pin slews at maximum rate |

**Important:**
- Refer to the "**Pin Allocation Table**" for details about pin availability per port
- Unimplemented bits will read back as '0'

### 19.14.8 ODCONx

**Name:** ODCONx

Open-Drain Control Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | ODCx7 | ODCx6 | ODCx5 | ODCx4 | ODCx3 | ODCx2 | ODCx1 | ODCx0 |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 0, 1, 2, 3, 4, 5, 6, 7 – ODCxn** Open-Drain Configuration on Rx Pin

| Value | Description |
|---|---|
| 1 | PORT pin operates as open-drain drive (sink current only) |
| 0 | PORT pin operates as standard push-pull drive (source and sink current) |

**Important:**

- Refer to the "**Pin Allocation Table**" for details about pin availability per port
- Unimplemented bits will read back as '0'

### 19.14.9 RxyI2C

**Name:** RxyI2C

I²C Pad Rxy Control Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | SLEW[1:0] | | PU[1:0] | | | | TH[1:0] | |
| Access | R/W | R/W | R/W | R/W | | | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | | | 0 | 0 |

**Bits 7:6 – SLEW[1:0]** I²C Specific Slew Rate Limiting Control

| Value | Description |
|---|---|
| 11 | I²C Fast mode Plus (1 MHz) slew rate enabled. The SLRxy bit is ignored. |
| 10 | Reserved |
| 01 | I²C Fast mode (400 kHz) slew rate enabled. The SLRxy bit is ignored. |
| 00 | Standard GPIO Slew Rate; enabled/disabled via the SLRxy bit |

**Bits 5:4 – PU[1:0]** I²C Pull-Up Selection

| Value | Description | |
|---|---|---|
| | FME = 0x | FME = 10 |
| 11 | Reserved | 20x current of standard weak pull-up |
| 10 | 10x current of standard weak pull-up | 10x current of standard weak pull-up |
| 01 | 2x current of standard weak pull-up | 5x current of standard weak pull-up |
| 00 | Standard GPIO weak pull-up, enabled via the WPUxy bit | |

**Bits 1:0 – TH[1:0]** I²C Input Threshold Selection

| Value | Description |
|---|---|
| 11 | SMBus 3.0 (1.35V) input threshold |
| 10 | SMBus 2.0 (2.1V) input threshold |
| 01 | I²C-specific input thresholds |
| 00 | Standard GPIO Input pull-up, enabled via the INLVLxy registers |

**Important:**
- Refer to the "**Pin Allocation Table**" for details about pin availability per port
- Unimplemented bits will read back as '0'

## 19.15 Register Summary - IO Ports

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0286 | RC4I2C | 7:0 | SLEW[1:0] | | PU[1:0] | | | | TH[1:0] | |
| 0x0287 | RC3I2C | 7:0 | SLEW[1:0] | | PU[1:0] | | | | TH[1:0] | |
| 0x0288 | RB2I2C | 7:0 | SLEW[1:0] | | PU[1:0] | | | | TH[1:0] | |
| 0x0289 | RB1I2C | 7:0 | SLEW[1:0] | | PU[1:0] | | | | TH[1:0] | |
| 0x028A ... 0x03FF | Reserved | | | | | | | | | |
| 0x0400 | ANSELA | 7:0 | ANSELA7 | ANSELA6 | ANSELA5 | ANSELA4 | ANSELA3 | ANSELA2 | ANSELA1 | ANSELA0 |
| 0x0401 | WPUA | 7:0 | WPUA7 | WPUA6 | WPUA5 | WPUA4 | WPUA3 | WPUA2 | WPUA1 | WPUA0 |
| 0x0402 | ODCONA | 7:0 | ODCA7 | ODCA6 | ODCA5 | ODCA4 | ODCA3 | ODCA2 | ODCA1 | ODCA0 |
| 0x0403 | SLRCONA | 7:0 | SLRA7 | SLRA6 | SLRA5 | SLRA4 | SLRA3 | SLRA2 | SLRA1 | SLRA0 |
| 0x0404 | INLVLA | 7:0 | INLVLA7 | INLVLA6 | INLVLA5 | INLVLA4 | INLVLA3 | INLVLA2 | INLVLA1 | INLVLA0 |
| 0x0405 ... 0x0407 | Reserved | | | | | | | | | |
| 0x0408 | ANSELB | 7:0 | ANSELB7 | ANSELB6 | ANSELB5 | ANSELB4 | ANSELB3 | ANSELB2 | ANSELB1 | ANSELB0 |
| 0x0409 | WPUB | 7:0 | WPUB7 | WPUB6 | WPUB5 | WPUB4 | WPUB3 | WPUB2 | WPUB1 | WPUB0 |
| 0x040A | ODCONB | 7:0 | ODCB7 | ODCB6 | ODCB5 | ODCB4 | ODCB3 | ODCB2 | ODCB1 | ODCB0 |
| 0x040B | SLRCONB | 7:0 | SLRB7 | SLRB6 | SLRB5 | SLRB4 | SLRB3 | SLRB2 | SLRB1 | SLRB0 |
| 0x040C | INLVLB | 7:0 | INLVLB7 | INLVLB6 | INLVLB5 | INLVLB4 | INLVLB3 | INLVLB2 | INLVLB1 | INLVLB0 |
| 0x040D ... 0x040F | Reserved | | | | | | | | | |
| 0x0410 | ANSELC | 7:0 | ANSELC7 | ANSELC6 | ANSELC5 | ANSELC4 | ANSELC3 | ANSELC2 | ANSELC1 | ANSELC0 |
| 0x0411 | WPUC | 7:0 | WPUC7 | WPUC6 | WPUC5 | WPUC4 | WPUC3 | WPUC2 | WPUC1 | WPUC0 |
| 0x0412 | ODCONC | 7:0 | ODCC7 | ODCC6 | ODCC5 | ODCC4 | ODCC3 | ODCC2 | ODCC1 | ODCC0 |
| 0x0413 | SLRCONC | 7:0 | SLRC7 | SLRC6 | SLRC5 | SLRC4 | SLRC3 | SLRC2 | SLRC1 | SLRC0 |
| 0x0414 | INLVLC | 7:0 | INLVLC7 | INLVLC6 | INLVLC5 | INLVLC4 | INLVLC3 | INLVLC2 | INLVLC1 | INLVLC0 |
| 0x0415 ... 0x0417 | Reserved | | | | | | | | | |
| 0x0418 | ANSELD | 7:0 | ANSELD7 | ANSELD6 | ANSELD5 | ANSELD4 | ANSELD3 | ANSELD2 | ANSELD1 | ANSELD0 |
| 0x0419 | WPUD | 7:0 | WPUD7 | WPUD6 | WPUD5 | WPUD4 | WPUD3 | WPUD2 | WPUD1 | WPUD0 |
| 0x041A | ODCOND | 7:0 | ODCD7 | ODCD6 | ODCD5 | ODCD4 | ODCD3 | ODCD2 | ODCD1 | ODCD0 |
| 0x041B | SLRCOND | 7:0 | SLRD7 | SLRD6 | SLRD5 | SLRD4 | SLRD3 | SLRD2 | SLRD1 | SLRD0 |
| 0x041C | INLVLD | 7:0 | INLVLD7 | INLVLD6 | INLVLD5 | INLVLD4 | INLVLD3 | INLVLD2 | INLVLD1 | INLVLD0 |
| 0x041D ... 0x041F | Reserved | | | | | | | | | |
| 0x0420 | ANSELE | 7:0 | | | | | | ANSELE2 | ANSELE1 | ANSELE0 |
| 0x0421 | WPUE | 7:0 | | | | | WPUE3 | WPUE2 | WPUE1 | WPUE0 |
| 0x0422 | ODCONE | 7:0 | | | | | | ODCE2 | ODCE1 | ODCE0 |
| 0x0423 | SLRCONE | 7:0 | | | | | | SLRE2 | SLRE1 | SLRE0 |
| 0x0424 | INLVLE | 7:0 | | | | | INLVLE3 | INLVLE2 | INLVLE1 | INLVLE0 |
| 0x0425 ... 0x0427 | Reserved | | | | | | | | | |
| 0x0428 | ANSELF | 7:0 | ANSELF7 | ANSELF6 | ANSELF5 | ANSELF4 | ANSELF3 | ANSELF2 | ANSELF1 | ANSELF0 |
| 0x0429 | WPUF | 7:0 | WPUF7 | WPUF6 | WPUF5 | WPUF4 | WPUF3 | WPUF2 | WPUF1 | WPUF0 |
| 0x042A | ODCONF | 7:0 | ODCF7 | ODCF6 | ODCF5 | ODCF4 | ODCF3 | ODCF2 | ODCF1 | ODCF0 |
| 0x042B | SLRCONF | 7:0 | SLRF7 | SLRF6 | SLRF5 | SLRF4 | SLRF3 | SLRF2 | SLRF1 | SLRF0 |
| 0x042C | INLVLF | 7:0 | INLVLF7 | INLVLF6 | INLVLF5 | INLVLF4 | INLVLF3 | INLVLF2 | INLVLF1 | INLVLF0 |
| 0x042D ... 0x04BD | Reserved | | | | | | | | | |
| 0x04BE | LATA | 7:0 | LATA7 | LATA6 | LATA5 | LATA4 | LATA3 | LATA2 | LATA1 | LATA0 |
| 0x04BF | LATB | 7:0 | LATB7 | LATB6 | LATB5 | LATB4 | LATB3 | LATB2 | LATB1 | LATB0 |

.....continued

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x04C0 | LATC | 7:0 | LATC7 | LATC6 | LATC5 | LATC4 | LATC3 | LATC2 | LATC1 | LATC0 |
| 0x04C1 | LATD | 7:0 | LATD7 | LATD6 | LATD5 | LATD4 | LATD3 | LATD2 | LATD1 | LATD0 |
| 0x04C2 | LATE | 7:0 | | | | | | LATE2 | LATE1 | LATE0 |
| 0x04C3 | LATF | 7:0 | LATF7 | LATF6 | LATF5 | LATF4 | LATF3 | LATF2 | LATF1 | LATF0 |
| 0x04C4 ... 0x04C5 | Reserved | | | | | | | | | |
| 0x04C6 | TRISA | 7:0 | TRISA7 | TRISA6 | TRISA5 | TRISA4 | TRISA3 | TRISA2 | TRISA1 | TRISA0 |
| 0x04C7 | TRISB | 7:0 | TRISB7 | TRISB6 | TRISB5 | TRISB4 | TRISB3 | TRISB2 | TRISB1 | TRISB0 |
| 0x04C8 | TRISC | 7:0 | TRISC7 | TRISC6 | TRISC5 | TRISC4 | TRISC3 | TRISC2 | TRISC1 | TRISC0 |
| 0x04C9 | TRISD | 7:0 | TRISD7 | TRISD6 | TRISD5 | TRISD4 | TRISD3 | TRISD2 | TRISD1 | TRISD0 |
| 0x04CA | TRISE | 7:0 | | | | | Reserved | TRISE2 | TRISE1 | TRISE0 |
| 0x04CB | TRISF | 7:0 | TRISF7 | TRISF6 | TRISF5 | TRISF4 | TRISF3 | TRISF2 | TRISF1 | TRISF0 |
| 0x04CC ... 0x04CD | Reserved | | | | | | | | | |
| 0x04CE | PORTA | 7:0 | RA7 | RA6 | RA5 | RA4 | RA3 | RA2 | RA1 | RA0 |
| 0x04CF | PORTB | 7:0 | RB7 | RB6 | RB5 | RB4 | RB3 | RB2 | RB1 | RB0 |
| 0x04D0 | PORTC | 7:0 | RC7 | RC6 | RC5 | RC4 | RC3 | RC2 | RC1 | RC0 |
| 0x04D1 | PORTD | 7:0 | RD7 | RD6 | RD5 | RD4 | RD3 | RD2 | RD1 | RD0 |
| 0x04D2 | PORTE | 7:0 | | | | | RE3 | RE2 | RE1 | RE0 |
| 0x04D3 | PORTF | 7:0 | RF7 | RF6 | RF5 | RF4 | RF3 | RF2 | RF1 | RF0 |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

# 20. IOC - Interrupt-on-Change

## 20.1 Overview

The pins denoted in the table below can be configured to operate as interrupt-on-change (IOC) pins for this device. An interrupt can be generated by detecting a signal that has either a rising edge or a falling edge. Any individual PORT pin, or combination of PORT pins, can be configured to generate an interrupt.

**Table 20-1. IOC Pin Availability per Device**

| Device | PORTA | PORTB | PORTC | PORTD | PORTE | PORTF |
|---|---|---|---|---|---|---|
| 28-pin devices | • | • | • | | •(1) | |
| 40/44-pin devices | • | • | • | | •(1) | |
| 48-pin devices | • | • | • | | •(1) | |

**Note:**
1. Pin RE3 only

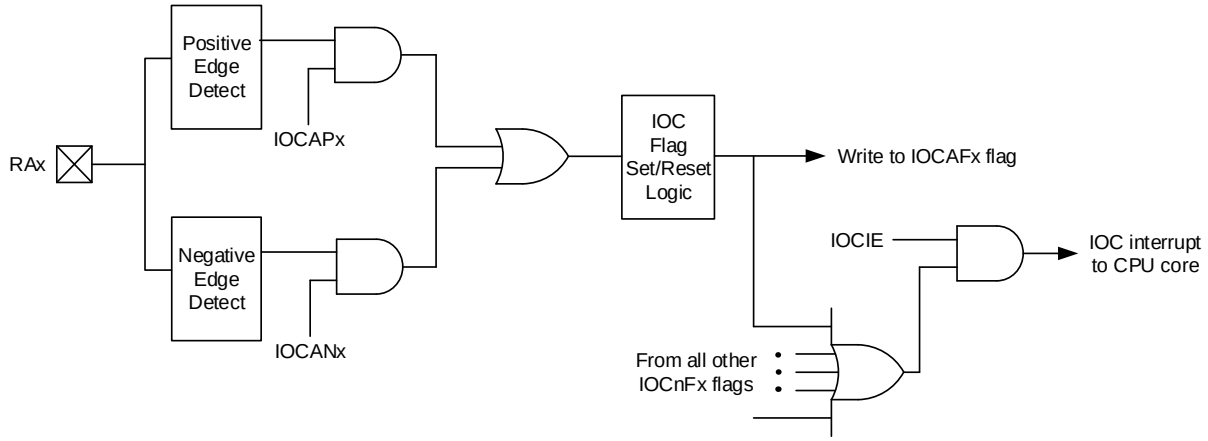
**Important:** If MCLRE = 1 or LVP = 1, the $\overline{MCLR}$ pin port functionality is disabled and IOC on that pin is not available.

The interrupt-on-change module has the following features:

- Interrupt-on-change enable (Host Switch)
- Individual pin configuration
- Rising and falling edge detection
- Individual pin interrupt flags

The following figure is a block diagram of the IOC module.

**Figure 20-1. Interrupt-on-Change Block Diagram (PORTA Example)**




## 20.2 Enabling the Module

For individual PORT pins to generate an interrupt, the IOC Interrupt Enable (IOCIE) bit of the Peripheral Interrupt Enable (PIEx) register must be set. If the IOC Interrupt Enable bit is disabled, the edge detection on the pin will still occur, but an interrupt will not be generated.

## 20.3 Individual Pin Configuration

A rising edge detector and a falling edge detector are present for each PORT pin. To enable a pin to detect a rising edge, the associated bit of the IOCxP register must be set. To enable a pin to detect a falling edge, the associated bit of the IOCxN register must be set. A PORT pin can be configured to detect rising and falling edges simultaneously by setting both associated bits of the IOCxP and IOCxN registers, respectively.

## 20.4 Interrupt Flags

The bits located in the IOCxF registers are status flags that correspond to the interrupt-on-change pins of each port. If an expected edge is detected on an appropriately enabled pin, then the status flag for that pin will be set, and an interrupt will be generated if the IOCIE bit is set. The IOCIF bit located in the corresponding Peripheral Interrupt Request (PIRx) register, is all the IOCxF bits ORd together. The IOCIF bit is read-only. All of the IOCxF Status bits must be cleared to clear the IOCIF bit.

## 20.5 Clearing Interrupt Flags

The individual status flags (IOCxF register bits) will be cleared by resetting them to zero. If another edge is detected during this clearing operation, the associated status flag will be set at the end of the sequence, regardless of the value actually being written.

To ensure that no detected edge is lost while clearing flags, only `AND` operations masking out known changed bits must be performed. The following sequence is an example of clearing an IOC interrupt flag using this method.

**Example 20-1. Clearing Interrupt Flags (PORTA Example)**

```
MOVLW    0xff
XORWF    IOCAF, W
ANDWF    IOCAF, F
```

## 20.6 Operation in Sleep

An interrupt-on-change event will wake the device from Sleep mode, if the IOCIE bit is set. If an edge is detected while in Sleep mode, the IOCxF register will be updated prior to the first instruction executed out of Sleep.

## 20.7 Register Definitions: Interrupt-on-Change Control

© 2019-2023 Microchip Technology Inc. and its subsidiaries

### 20.7.1 IOCxF

**Name:** IOCxF

Interrupt-on-Change Flag Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | IOCxF7 | IOCxF6 | IOCxF5 | IOCxF4 | IOCxF3 | IOCxF2 | IOCxF1 | IOCxF0 |
| Access | R/W/HS | R/W/HS | R/W/HS | R/W/HS | R/W/HS | R/W/HS | R/W/HS | R/W/HS |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 0, 1, 2, 3, 4, 5, 6, 7 – IOCxFn** Interrupt-on-Change Flag

| Value | Condition | Description |
|---|---|---|
| 1 | IOCxP[n] = 1 | A positive edge was detected on the Rx[n] pin |
| 1 | IOCxN[n] = 1 | A negative edge was detected on the Rx[n] pin |
| 0 | IOCxP[n] = x and IOCxN[n] = x | No change was detected, or the user cleared the detected change |

**Important:**

- If MCLRE = 1 or LVP = 1, the $\overline{\text{MCLR}}$ pin port functionality is disabled and IOC on that pin is not available
- Refer to the **"Pin Allocation Table"** for details about pins with configurable IOC per port

### 20.7.2 IOCxN

**Name:** IOCxN

Interrupt-on-Change Negative Edge Register Example

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | IOCxN7 | IOCxN6 | IOCxN5 | IOCxN4 | IOCxN3 | IOCxN2 | IOCxN1 | IOCxN0 |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 0, 1, 2, 3, 4, 5, 6, 7 – IOCxNn** Interrupt-on-Change Negative Edge Enable

| Value | Description |
|---|---|
| 1 | Interrupt-on-change enabled on the IOCx pin for a negative-going edge. Associated Status bit and interrupt flag will be set upon detecting an edge. |
| 0 | Falling edge interrupt-on-change disabled for the associated pin |

**Important:**

- If MCLRE = 1 or LVP = 1, the $\overline{\text{MCLR}}$ pin port functionality is disabled and IOC on that pin is not available
- Refer to the **"Pin Allocation Table"** for details about pins with configurable IOC per port

### 20.7.3 IOCxP

**Name:** IOCxP

Interrupt-on-Change Positive Edge Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | IOCxP7 | IOCxP6 | IOCxP5 | IOCxP4 | IOCxP3 | IOCxP2 | IOCxP1 | IOCxP0 |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 0, 1, 2, 3, 4, 5, 6, 7 – IOCxPn** Interrupt-on-Change Positive Edge Enable

| Value | Description |
|---|---|
| 1 | Interrupt-on-change enabled on the IOCx pin for a positive-going edge. Associated Status bit and interrupt flag will be set upon detecting an edge. |
| 0 | Rising edge interrupt-on-change disabled for the associated pin. |

**Important:**

- If MCLRE = 1 or LVP = 1, the $\overline{\text{MCLR}}$ pin port functionality is disabled and IOC on that pin is not available
- Refer to the **"Pin Allocation Table"** for details about pins with configurable IOC per port

## 20.8 Register Summary - Interrupt-on-Change Control

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0405 | IOCAP | 7:0 | IOCAP7 | IOCAP6 | IOCAP5 | IOCAP4 | IOCAP3 | IOCAP2 | IOCAP1 | IOCAP0 |
| 0x0406 | IOCAN | 7:0 | IOCAN7 | IOCAN6 | IOCAN5 | IOCAN4 | IOCAN3 | IOCAN2 | IOCAN1 | IOCAN0 |
| 0x0407 | IOCAF | 7:0 | IOCAF7 | IOCAF6 | IOCAF5 | IOCAF4 | IOCAF3 | IOCAF2 | IOCAF1 | IOCAF0 |
| 0x0408 ... 0x040C | Reserved | | | | | | | | | |
| 0x040D | IOCBP | 7:0 | IOCBP7 | IOCBP6 | IOCBP5 | IOCBP4 | IOCBP3 | IOCBP2 | IOCBP1 | IOCBP0 |
| 0x040E | IOCBN | 7:0 | IOCBN7 | IOCBN6 | IOCBN5 | IOCBN4 | IOCBN3 | IOCBN2 | IOCBN1 | IOCBN0 |
| 0x040F | IOCBF | 7:0 | IOCBF7 | IOCBF6 | IOCBF5 | IOCBF4 | IOCBF3 | IOCBF2 | IOCBF1 | IOCBF0 |
| 0x0410 ... 0x0414 | Reserved | | | | | | | | | |
| 0x0415 | IOCCP | 7:0 | IOCCP7 | IOCCP6 | IOCCP5 | IOCCP4 | IOCCP3 | IOCCP2 | IOCCP1 | IOCCP0 |
| 0x0416 | IOCCN | 7:0 | IOCCN7 | IOCCN6 | IOCCN5 | IOCCN4 | IOCCN3 | IOCCN2 | IOCCN1 | IOCCN0 |
| 0x0417 | IOCCF | 7:0 | IOCCF7 | IOCCF6 | IOCCF5 | IOCCF4 | IOCCF3 | IOCCF2 | IOCCF1 | IOCCF0 |
| 0x0418 ... 0x0424 | Reserved | | | | | | | | | |
| 0x0425 | IOCEP | 7:0 | | | | | IOCEP3 | | | |
| 0x0426 | IOCEN | 7:0 | | | | | IOCEN3 | | | |
| 0x0427 | IOCEF | 7:0 | | | | | IOCEF3 | | | |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

# 21. PPS - Peripheral Pin Select Module

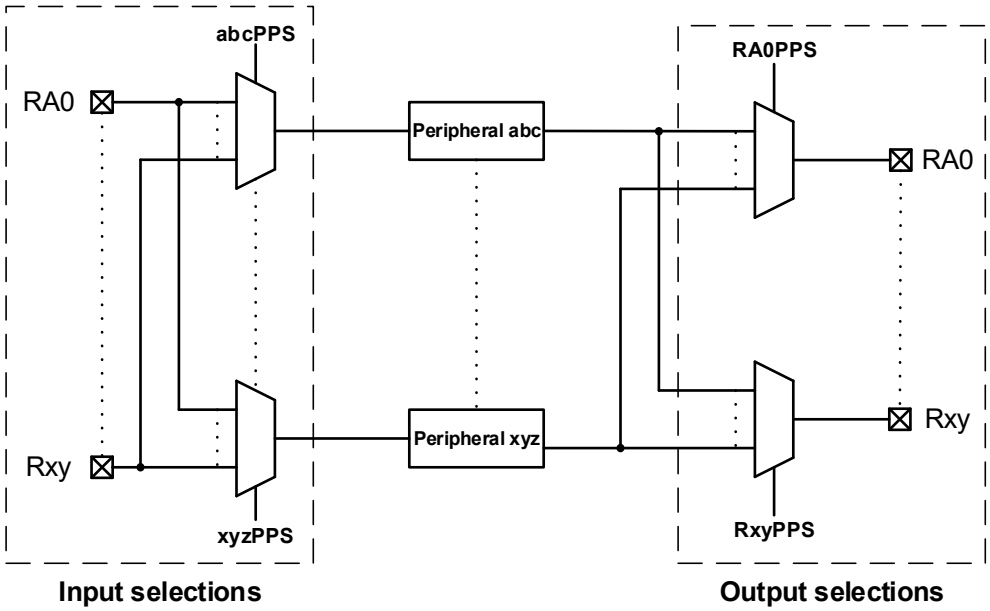
## 21.1 Overview

The Peripheral Pin Select (PPS) module connects peripheral inputs and outputs to the device I/O pins. Only digital signals are included in the selections.

**Important:** All analog inputs and outputs remain fixed to their assigned pins and cannot be changed through PPS.

Input and output selections are independent as shown in the figure below.

**Figure 21-1. PPS Block Diagram**




## 21.2 PPS Inputs

Each digital peripheral has a dedicated PPS Peripheral Input Selection (xxxPPS) register with which the input pin to the peripheral is selected. Devices that have 20 leads or less (8/14/16/20) allow PPS routing to any I/O pin, while devices with 28 leads or more allow PPS routing to I/Os contained within two ports (see the table below).

**Important:** The notation “xxx” in the generic register name is a placeholder for the peripheral identifier. For example, xxx = T0CKI for the T0CKIPPS register.

Multiple peripherals can operate from the same source simultaneously. Port reads always return the pin level regardless of peripheral PPS selection. If a pin also has analog functions associated, the ANSEL bit for that pin must be cleared to enable the digital input buffer.

**Table 21-1. PPS Input Selection Table**

| Peripheral | PPS Input Register | Default Pin Selection at POR | Register Reset Value at POR | Available Input Port | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | 28-Pin Devices | | | 40-Pin Devices | | | | | | 48-Pin Devices | | | | |
| Interrupt 0 | INT0PPS | RB0 | 'b001 000 | A | B | — | A | B | — | — | — | A | B | — | — | — | — |
| Interrupt 1 | INT1PPS | RB1 | 'b001 001 | A | B | — | A | B | — | — | — | — | B | — | D | — | — |
| Interrupt 2 | INT2PPS | RB2 | 'b001 010 | A | B | — | A | B | — | — | — | — | B | — | — | — | F |
| Timer0 Clock | T0CKIPPS | RA4 | 'b000 100 | A | B | — | A | B | — | — | — | A | — | — | — | — | F |
| Timer1 Clock | T1CKIPPS | RC0 | 'b010 000 | A | — | C | A | — | C | — | — | — | — | C | — | E | — |
| Timer1 Gate | T1GPPS | RB5 | 'b001 101 | — | B | C | — | B | C | — | — | — | B | C | — | — | — |
| Timer3 Clock | T3CKIPPS | RC0 | 'b010 000 | — | B | C | — | B | C | — | — | — | — | C | — | E | — |
| Timer3 Gate | T3GPPS | RC0 | 'b010 000 | A | — | C | A | — | C | — | — | A | — | C | — | — | — |
| Timer5 Clock | T5CKIPPS | RC2 | 'b010 010 | A | — | C | A | — | C | — | — | — | — | C | — | E | — |
| Timer5 Gate | T5GPPS | RB4 | 'b001 100 | — | B | C | — | B | — | D | — | — | B | — | D | — | — |
| Timer2 Input | T2INPPS | RC3 | 'b010 011 | A | — | C | A | — | C | — | — | A | — | C | — | — | — |
| Timer4 Input | T4INPPS | RC5 | 'b010 101 | — | B | C | — | B | C | — | — | — | B | C | — | — | — |
| Timer6 Input | T6INPPS | RB7 | 'b001 111 | — | B | C | — | B | — | D | — | — | B | — | D | — | — |
| CCP1 | CCP1PPS | RC2 | 'b010 010 | — | B | C | — | B | C | — | — | — | — | C | — | — | F |
| CCP2 | CCP2PPS | RC1 | 'b010 001 | — | B | C | — | B | C | — | — | — | — | C | — | — | F |
| CCP3 | CCP3PPS | RB5 | 'b001 101 | — | B | C | — | B | — | D | — | — | B | — | D | — | — |
| SMT1 Window | SMT1WINPPS | RC0 | 'b010 000 | — | B | C | — | B | C | — | — | — | — | C | — | — | F |
| SMT1 Signal | SMT1SIGPPS | RC1 | 'b010 001 | — | B | C | — | B | C | — | — | — | — | C | — | — | F |
| PWM Input 0 | PWMIN0PPS | RC2 | 'b010 010 | — | B | C | — | B | C | — | — | — | — | C | — | — | F |
| PWM Input 1 | PWMIN1PPS | RC6 | 'b010 110 | A | — | C | A | — | — | — | E | A | — | — | — | E | — |
| PWM1 External Reset Source | PWM1ERSPPS | RC3 | 'b010 011 | A | — | C | A | — | C | — | — | A | — | C | — | | — |
| PWM2 External Reset Source | PWM2ERSPPS | RC5 | 'b010 101 | A | — | C | A | — | C | — | — | — | — | C | — | E | — |
| PWM3 External Reset Source | PWM3ERSPPS | RB7 | 'b001 111 | — | B | C | — | B | — | D | — | — | B | — | D | — | — |
| CWG1 | CWG1PPS | RB0 | 'b001 000 | — | B | C | — | B | — | D | — | — | B | — | D | — | — |
| CWG2 | CWG2PPS | RB1 | 'b001 001 | — | B | C | — | B | — | D | — | — | B | — | D | — | — |
| CWG3 | CWG3PPS | RB2 | 'b001 010 | — | B | C | — | B | — | D | — | — | B | — | D | — | — |
| DSM1 Carrier Low | MD1CARLPPS | RA3 | 'b000 011 | A | — | C | A | — | — | D | — | A | — | — | D | — | — |
| DSM1 Carrier High | MD1CARHPPS | RA4 | 'b000 100 | A | — | C | A | — | — | D | — | A | — | — | D | — | — |
| DSM1 Source | MD1SRCPPS | RA5 | 'b000 101 | A | — | C | A | — | — | D | — | A | — | — | D | — | — |
| CLCx Input 1 | CLCIN0PPS | RA0 | 'b000 000 | A | — | C | A | — | C | — | — | A | — | C | — | — | — |
| CLCx Input 2 | CLCIN1PPS | RA1 | 'b000 001 | A | — | C | A | — | C | — | — | A | — | C | — | — | — |
| CLCx Input 3 | CLCIN2PPS | RB6 | 'b001 110 | — | B | C | — | B | — | D | — | — | B | — | D | — | — |
| CLCx Input 4 | CLCIN3PPS | RB7 | 'b001 111 | — | B | C | — | B | — | D | — | — | B | — | D | — | — |
| CLCx Input 5 | CLCIN4PPS | RA0 | 'b000 000 | A | — | C | A | — | C | — | — | A | — | C | — | — | — |
| CLCx Input 6 | CLCIN5PPS | RA1 | 'b000 001 | A | — | C | A | — | C | — | — | A | — | C | — | — | — |
| CLCx Input 7 | CLCIN6PPS | RB6 | 'b001 110 | — | B | C | — | B | — | D | — | — | B | — | D | — | — |
| CLCx Input 8 | CLCIN7PPS | RB7 | 'b001 111 | — | B | C | — | B | — | D | — | — | B | — | D | — | — |
| ADC Conversion Trigger | ADACTPPS | RB4 | 'b001 100 | — | B | C | — | B | — | D | — | — | B | — | D | — | — |
| SPI1 Clock | SPI1SCKPPS | RC3 | 'b010 011 | — | B | C | — | B | C | — | — | — | B | C | — | — | — |
| SPI1 Data | SPI1SDIPPS | RC4 | 'b010 100 | — | B | C | — | B | C | — | — | — | B | C | — | — | — |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

.....continued

| Peripheral | PPS Input Register | Default Pin Selection at POR | Register Reset Value at POR | Available Input Port | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | 28-Pin Devices | | | 40-Pin Devices | | | | | 48-Pin Devices | | | | | |
| SPI1 Client Select | SPI1SSPPS | RA5 | 'b000 101 | A | — | C | A | — | — | D | — | A | — | — | D | — | — |
| SPI2 Clock | SPI2SCKPPS | RB3 | 'b001 011 | — | B | C | — | B | — | D | — | — | B | — | D | — | — |
| SPI2 Data | SPI2SDIPPS | RB2 | 'b001 010 | — | B | C | — | B | — | D | — | — | B | — | D | — | — |
| SPI2 Client Select | SPI2SSPPS | RA4 | 'b000 100 | A | — | C | A | — | — | D | — | A | — | — | D | — | — |
| I2C1 Clock | I2C1SCLPPS(1) | RC3 | 'b010 011 | — | B | C | — | B | C | — | — | — | B | C | — | — | — |
| I2C1 Data | I2C1SDAPPS(1) | RC4 | 'b010 100 | — | B | C | — | B | C | — | — | — | B | C | — | — | — |
| UART1 Receive | U1RXPPS | RC7 | 'b010 111 | — | B | C | — | B | C | — | — | — | — | C | — | — | F |
| UART1 Clear to Send | U1CTSPPS | RC6 | 'b010 110 | — | B | C | — | B | C | — | — | — | — | C | — | — | F |
| UART2 Receive | U2RXPPS | RB7 | 'b001 111 | — | B | C | — | B | — | D | — | — | B | — | D | — | — |
| UART2 Clear to Send | U2CTSPPS | RB6 | 'b001 110 | — | B | C | — | B | — | D | — | — | B | — | D | — | — |
| UART3 Receive | U3RXPPS | RA7 | 'b000 111 | A | B | — | A | B | — | — | — | A | — | — | — | — | F |
| UART3 Clear to Send | U3CTSPPS | RA6 | 'b000 110 | A | B | — | A | B | — | — | — | A | — | — | — | — | F |
| UART4 Receive | U4RXPPS | RB5 | 'b001 101 | — | B | C | — | B | — | D | — | — | B | — | D | — | — |
| UART4 Clear to Send | U4CTSPPS | RB4 | 'b001 100 | — | B | C | — | B | — | D | — | — | B | — | D | — | — |
| UART5 Receive | U5RXPPS | RA5 | 'b000 101 | A | — | C | A | — | C | — | — | A | — | — | — | — | F |
| UART5 Clear to Send | U5CTSPPS | RA4 | 'b000 100 | A | — | C | A | — | C | — | — | A | — | — | — | — | F |

**Note:**

1. Bidirectional pin. The corresponding output must select the same pin.

## 21.3 PPS Outputs

Each digital peripheral has a dedicated Pin Rxy Output Source Selection (RxyPPS) register with which the pin output source is selected. With few exceptions, the port TRIS control associated with that pin retains control over the pin output driver. Peripherals that control the pin output driver as part of the peripheral operation will override the TRIS control as needed. The $I^2C$ module is an example of such a peripheral.

**Important:** The notation 'Rxy' is a placeholder for the pin identifier. The 'x' holds the place of the PORT letter and the 'y' holds the place of the bit number. For example, Rxy = RA0 for the RA0PPS register.

The table below shows the output codes for each peripheral, as well as the available Port selections.

**Table 21-2. PPS Output Selection Table**

| RxyPPS | Output Source | Available Output Ports | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 28-Pin Devices | | | 40-Pin Devices | | | | | 48-Pin Devices | | | | | |
| 0x45 | ADGRDB | A | — | C | A | — | C | — | — | A | — | — | — | — | F |
| 0x44 | ADGRDA | A | — | C | A | — | C | — | — | A | — | — | — | — | F |
| 0x43 | DSM1 | A | — | C | A | — | — | D | — | A | — | — | D | — | — |
| 0x42 | CLKR | — | B | C | — | B | C | — | — | — | B | — | — | E | — |
| 0x41 | NCO3 | — | B | C | — | B | — | — | E | — | B | — | — | E | — |
| 0x40 | NCO2 | — | B | C | — | B | — | D | — | — | B | — | D | — | — |
| 0x3F | NCO1 | A | — | C | A | — | — | D | — | A | — | — | D | — | — |
| 0x3E - 0x3A | **Reserved** | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| 0x39 | TMR0 | — | B | C | — | B | C | — | — | — | — | C | — | — | F |
| 0x38 | I2C1 SDA(1) | — | B | C | — | B | C | — | — | — | B | C | — | — | — |

.....continued

| RxyPPS | Output Source | Available Output Ports | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 28-Pin Devices | | | 40-Pin Devices | | | | | 48-Pin Devices | | | | | |
| 0x37 | I2C1 SCL(1) | — | B | C | — | B | C | — | — | — | B | C | — | — | — |
| 0x36 | SPI2 SS | A | — | C | A | — | — | D | — | A | — | — | D | — | — |
| 0x35 | SPI2 SDO | — | B | C | — | B | — | D | — | — | B | — | D | — | — |
| 0x34 | SPI2 SCK | — | B | C | — | B | — | D | — | — | B | — | D | — | — |
| 0x33 | SPI1 SS | A | — | C | A | — | — | D | — | A | — | — | D | — | — |
| 0x32 | SPI1 SDO | — | B | C | — | B | C | — | — | — | B | C | — | — | — |
| 0x31 | SPI1 SCK | — | B | C | — | B | C | — | — | — | B | C | — | — | — |
| 0x30 | C2OUT | A | — | C | A | — | — | — | E | A | — | — | — | E | — |
| 0x2F | C1OUT | A | — | C | A | — | — | D | — | A | — | — | D | — | — |
| 0x2E | UART5 RTS | — | B | C | — | B | C | — | — | — | — | C | — | — | F |
| 0x2D | UART5 TXDE | — | B | C | — | B | C | — | — | — | — | C | — | — | F |
| 0x2C | UART5 TX | — | B | C | — | B | C | — | — | — | — | C | — | — | F |
| 0x2B | UART4 RTS | A | B | — | A | — | — | D | — | A | — | — | D | — | — |
| 0x2A | UART4 TXDE | A | B | — | A | — | — | D | — | A | — | — | D | — | — |
| 0x29 | UART4 TX | A | B | — | A | — | — | D | — | A | — | — | D | — | — |
| 0x28 | UART3 RTS | A | B | — | A | B | — | — | — | A | — | — | — | — | F |
| 0x27 | UART3 TXDE | A | B | — | A | B | — | — | — | A | — | — | — | — | F |
| 0x26 | UART3 TX | A | B | — | A | B | — | — | — | A | — | — | — | — | F |
| 0x25 | UART2 RTS | — | B | C | — | B | — | D | — | — | B | — | D | — | — |
| 0x24 | UART2 TXDE | — | B | C | — | B | — | D | — | — | B | — | D | — | — |
| 0x23 | UART2 TX | — | B | C | — | B | — | D | — | — | B | — | D | — | — |
| 0x22 | UART1 RTS | — | B | C | — | B | C | — | — | — | — | C | — | — | F |
| 0x21 | UART1 TXDE | — | B | C | — | B | C | — | — | — | — | C | — | — | F |
| 0x20 | UART1 TX | — | B | C | — | B | C | — | — | — | — | C | — | — | F |
| 0x1F | **Reserved** | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| 0x1E | **Reserved** | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| 0x1D | PWM3S1P2_OUT | — | B | C | — | B | — | D | — | — | B | — | D | — | — |
| 0x1C | PWM3S1P1_OUT | — | B | C | — | B | — | D | — | — | B | — | D | — | — |
| 0x1B | PWM2S1P2_OUT | — | B | C | — | B | — | D | — | — | B | — | D | — | — |
| 0x1A | PWM2S1P1_OUT | — | B | C | — | B | — | D | — | — | B | — | D | — | — |
| 0x19 | PWM1S1P2_OUT | — | B | C | — | B | C | — | — | — | — | C | — | — | F |
| 0x18 | PWM1S1P1_OUT | — | B | C | — | B | C | — | — | — | — | C | — | — | F |
| 0x17 | CCP3 | — | B | C | — | B | — | D | — | — | B | — | D | — | — |
| 0x16 | CCP2 | — | B | C | — | B | C | — | — | — | — | C | — | — | F |
| 0x15 | CCP1 | — | B | C | — | B | C | — | — | — | — | C | — | — | F |
| 0x14 | CWG3D | A | — | C | A | — | — | D | — | A | — | — | D | — | — |
| 0x13 | CWG3C | A | — | C | A | — | — | D | — | A | — | — | D | — | — |
| 0x12 | CWG3B | A | — | C | A | — | — | — | E | A | — | — | — | E | — |
| 0x11 | CWG3A | — | B | C | — | B | C | — | — | — | B | C | — | — | — |
| 0x10 | CWG2D | — | B | C | — | B | — | D | — | — | B | — | D | — | — |
| 0x0F | CWG2C | — | B | C | — | B | — | D | — | — | B | — | D | — | — |
| 0x0E | CWG2B | — | B | C | — | B | — | D | — | — | B | — | D | — | — |
| 0x0D | CWG2A | — | B | C | — | B | C | — | — | — | B | C | — | — | — |
| 0x0C | CWG1D | — | B | C | — | B | — | D | — | — | B | — | D | — | — |
| 0x0B | CWG1C | — | B | C | — | B | — | D | — | — | B | — | D | — | — |

| .....continued | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RxyPPS | Output Source | Available Output Ports | | | | | | | | | | | | | |
| | | 28-Pin Devices | | | 40-Pin Devices | | | | | 48-Pin Devices | | | | | |
| 0x0A | CWG1B | — | B | C | — | B | — | D | — | — | B | — | D | — | — |
| 0x09 | CWG1A | — | B | C | — | B | C | — | — | — | B | C | — | — | — |
| 0x08 | CLC8OUT | — | B | C | — | B | — | D | — | — | B | — | D | — | — |
| 0x07 | CLC7OUT | — | B | C | — | B | — | D | — | — | B | — | D | — | — |
| 0x06 | CLC6OUT | A | — | C | A | — | C | — | — | A | — | — | — | — | F |
| 0x05 | CLC5OUT | A | — | C | A | — | C | — | — | A | — | — | — | — | F |
| 0x04 | CLC4OUT | — | B | C | — | B | — | D | — | — | B | — | D | — | — |
| 0x03 | CLC3OUT | — | B | C | — | B | — | D | — | — | B | — | D | — | — |
| 0x02 | CLC2OUT | A | — | C | A | — | C | — | — | A | — | — | — | — | F |
| 0x01 | CLC1OUT | A | — | C | A | — | C | — | — | A | — | — | — | — | F |
| 0x00 | LATxy | A | B | C | A | B | C | D | E | A | B | C | D | E | F |

**Note:**

1. Bidirectional pin. The corresponding input must select the same pin.

## 21.4 Bidirectional Pins

PPS selections for peripherals with bidirectional signals on a single pin must be made so that the PPS input and PPS output select the same pin. The $I^2C$ Serial Clock (SCL) and Serial Data (SDA) are examples of such pins.

**Important:** The $I^2C$ default pins and a limited number of other alternate pins are $I^2C$ and SMBus compatible. SDA and SCL signals can be routed to any pin; however, pins without $I^2C$ compatibility will operate at standard TTL/ST logic levels as selected by the port's INLVL register.

## 21.5 PPS Lock

The PPS module provides an extra layer of protection to prevent inadvertent changes to the PPS selection registers. The PPSLOCKED bit is used in combination with specific code execution blocks to lock/unlock the PPS selection registers.

**Important:** The PPSLOCKED bit is clear by default (PPSLOCKED = 0), which allows the PPS selection registers to be modified without an unlock sequence.

PPS selection registers are locked when the PPSLOCKED bit is set (PPSLOCKED = 1). Setting the PPSLOCKED bit requires a specific lock sequence as shown in the examples below in both C and assembly languages.

PPS selection registers are unlocked when the PPSLOCKED bit is clear (PPSLOCKED = 0). Clearing the PPSLOCKED bit requires a specific unlock sequence as shown in the examples below in both C and assembly languages.

**Important:** All interrupts must be disabled before starting the lock/unlock sequence to ensure proper execution.

**Example 21-1. PPS Lock Sequence (assembly language)**

```
; suspend interrupts
   BCF      INTCON0,GIE
   BANKSEL  PPSLOCK
; required sequence, next 5 instructions
   MOVLW    0x55
   MOVWF    PPSLOCK
   MOVLW    0xAA
   MOVWF    PPSLOCK
; Set PPSLOCKED bit
   BSF      PPSLOCK,PPSLOCKED
; restore interrupts
   BSF      INTCON0,GIE
```

**Example 21-2. PPS Lock Sequence (C language)**

```
INTCON0bits.GIE = 0;          //Suspend interrupts
PPSLOCK = 0x55;               //Required sequence
PPSLOCK = 0xAA;               //Required sequence
PPSLOCKbits.PPSLOCKED = 1;    //Set PPSLOCKED bit
INTCON0bits.GIE = 1;          //Restore interrupts
```

**Example 21-3. PPS Unlock Sequence (assembly language)**

```
; suspend interrupts
   BCF      INTCON0,GIE
   BANKSEL  PPSLOCK
; required sequence, next 5 instructions
   MOVLW    0x55
   MOVWF    PPSLOCK
   MOVLW    0xAA
   MOVWF    PPSLOCK
; Clear PPSLOCKED bit
   BCF      PPSLOCK,PPSLOCKED
; restore interrupts
   BSF      INTCON0,GIE
```

**Example 21-4. PPS Unlock Sequence (C language)**

```
INTCON0bits.GIE = 0;          //Suspend interrupts
PPSLOCK = 0x55;               //Required sequence
PPSLOCK = 0xAA;               //Required sequence
PPSLOCKbits.PPSLOCKED = 0;    //Clear PPSLOCKED bit
INTCON0bits.GIE = 1;          //Restore interrupts
```

### 21.5.1 PPS One-Way Lock

The PPS1WAY Configuration bit can also be used to prevent inadvertent modification to the PPS selection registers.

When the PPS1WAY bit is set (PPS1WAY = 1), the PPSLOCKED bit can only be set one time after a device Reset. Once the PPSLOCKED bit has been set, it cannot be cleared again unless a device Reset is executed.

When the PPS1WAY bit is clear (PPS1WAY = 0), the PPSLOCKED bit can be set or cleared as needed; however, the PPS lock/unlock sequences must be executed.

## 21.6 Operation During Sleep

PPS input and output selections are unaffected by Sleep.

## 21.7 Effects of a Reset

A device Power-on Reset (POR) or Brown-out Reset (BOR) returns all PPS input selection registers to their default values and clears all PPS output selection registers. All other Resets leave the selections unchanged. Default input selections are shown in the PPS input register details table. The PPSLOCKED bit is cleared in all Reset conditions.

## 21.8 Register Definitions: Peripheral Pin Select (PPS)

### 21.8.1 xxxPPS

**Name:** xxxPPS

Peripheral Input Selection Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | PORT[2:0] | | | PIN[2:0] | | |
| Access | | | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | | | m | m | m | m | m | m |

**Bits 5:3 – PORT[2:0]** Peripheral Input PORT Selection(1)
See the PPS Input Selection Table for the list of available Ports and default pin locations.

| PORT | Selection |
|---|---|
| 101 | PORTF |
| 100 | PORTE |
| 011 | PORTD |
| 010 | PORTC |
| 001 | PORTB |
| 000 | PORTA |

Reset States: POR = mmm
All other Resets = uuu

**Bits 2:0 – PIN[2:0]** Peripheral Input PORT Pin Selection(2)
Reset States: POR = mmm
All other Resets = uuu

| Value | Description |
|---|---|
| 111 | Peripheral input is from PORTx Pin 7 (Rx7) |
| 110 | Peripheral input is from PORTx Pin 6 (Rx6) |
| 101 | Peripheral input is from PORTx Pin 5 (Rx5) |
| 100 | Peripheral input is from PORTx Pin 4 (Rx4) |
| 011 | Peripheral input is from PORTx Pin 3 (Rx3) |
| 010 | Peripheral input is from PORTx Pin 2 (Rx2) |
| 001 | Peripheral input is from PORTx Pin 1 (Rx1) |
| 000 | Peripheral input is from PORTx Pin 0 (Rx0) |

**Notes:**
1. The Reset value ‘m’ is determined by device default locations for that input.
2. Refer to the **“Pin Allocation Table”** for details about available pins per port.

### 21.8.2 RxyPPS

**Name:** RxyPPS

Pin Rxy Output Source Selection Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | RxyPPS[6:0] | | | | | | |
| Access | | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 6:0 – RxyPPS[6:0]** Pin Rxy Output Source Selection
See the PPS Output Selection Table for the list of RxyPPS Output Source codes
Reset States: POR = 0000000
All other Resets = uuuuuuu

### 21.8.3 PPSLOCK

**Name:** PPSLOCK

PPS Lock Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | PPSLOCKED |
| Access | | | | | | | | R/W |
| Reset | | | | | | | | 0 |

**Bit 0 – PPSLOCKED** PPS Locked
Reset States: POR = 0
All other Resets = 0

| Value | Description |
|---|---|
| 1 | PPS is locked. PPS selections cannot be changed. Writes to any PPS register are ignored. |
| 0 | PPS is not locked. PPS selections can be changed, but may require the PPS lock/unlock sequence. |

## 21.9 Register Summary - Peripheral Pin Select Module

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0200 | PPSLOCK | 7:0 | | | | | | | | PPSLOCKED |
| 0x0201 | RA0PPS | 7:0 | | | | | RA0PPS[6:0] | | | |
| 0x0202 | RA1PPS | 7:0 | | | | | RA1PPS[6:0] | | | |
| 0x0203 | RA2PPS | 7:0 | | | | | RA2PPS[6:0] | | | |
| 0x0204 | RA3PPS | 7:0 | | | | | RA3PPS[6:0] | | | |
| 0x0205 | RA4PPS | 7:0 | | | | | RA4PPS[6:0] | | | |
| 0x0206 | RA5PPS | 7:0 | | | | | RA5PPS[6:0] | | | |
| 0x0207 | RA6PPS | 7:0 | | | | | RA6PPS[6:0] | | | |
| 0x0208 | RA7PPS | 7:0 | | | | | RA7PPS[6:0] | | | |
| 0x0209 | RB0PPS | 7:0 | | | | | RB0PPS[6:0] | | | |
| 0x020A | RB1PPS | 7:0 | | | | | RB1PPS[6:0] | | | |
| 0x020B | RB2PPS | 7:0 | | | | | RB2PPS[6:0] | | | |
| 0x020C | RB3PPS | 7:0 | | | | | RB3PPS[6:0] | | | |
| 0x020D | RB4PPS | 7:0 | | | | | RB4PPS[6:0] | | | |
| 0x020E | RB5PPS | 7:0 | | | | | RB5PPS[6:0] | | | |
| 0x020F | RB6PPS | 7:0 | | | | | RB6PPS[6:0] | | | |
| 0x0210 | RB7PPS | 7:0 | | | | | RB7PPS[6:0] | | | |
| 0x0211 | RC0PPS | 7:0 | | | | | RC0PPS[6:0] | | | |
| 0x0212 | RC1PPS | 7:0 | | | | | RC1PPS[6:0] | | | |
| 0x0213 | RC2PPS | 7:0 | | | | | RC2PPS[6:0] | | | |
| 0x0214 | RC3PPS | 7:0 | | | | | RC3PPS[6:0] | | | |
| 0x0215 | RC4PPS | 7:0 | | | | | RC4PPS[6:0] | | | |
| 0x0216 | RC5PPS | 7:0 | | | | | RC5PPS[6:0] | | | |
| 0x0217 | RC6PPS | 7:0 | | | | | RC6PPS[6:0] | | | |
| 0x0218 | RC7PPS | 7:0 | | | | | RC7PPS[6:0] | | | |
| 0x0219 | RD0PPS | 7:0 | | | | | RD0PPS[6:0] | | | |
| 0x021A | RD1PPS | 7:0 | | | | | RD1PPS[6:0] | | | |
| 0x021B | RD2PPS | 7:0 | | | | | RD2PPS[6:0] | | | |
| 0x021C | RD3PPS | 7:0 | | | | | RD3PPS[6:0] | | | |
| 0x021D | RD4PPS | 7:0 | | | | | RD4PPS[6:0] | | | |
| 0x021E | RD5PPS | 7:0 | | | | | RD5PPS[6:0] | | | |
| 0x021F | RD6PPS | 7:0 | | | | | RD6PPS[6:0] | | | |
| 0x0220 | RD7PPS | 7:0 | | | | | RD7PPS[6:0] | | | |
| 0x0221 | RE0PPS | 7:0 | | | | | RE0PPS[6:0] | | | |
| 0x0222 | RE1PPS | 7:0 | | | | | RE1PPS[6:0] | | | |
| 0x0223 | RE2PPS | 7:0 | | | | | RE2PPS[6:0] | | | |
| 0x0224 ... 0x0228 | Reserved | | | | | | | | | |
| 0x0229 | RF0PPS | 7:0 | | | | | RF0PPS[6:0] | | | |
| 0x022A | RF1PPS | 7:0 | | | | | RF1PPS[6:0] | | | |
| 0x022B | RF2PPS | 7:0 | | | | | RF2PPS[6:0] | | | |
| 0x022C | RF3PPS | 7:0 | | | | | RF3PPS[6:0] | | | |
| 0x022D | RF4PPS | 7:0 | | | | | RF4PPS[6:0] | | | |
| 0x022E | RF5PPS | 7:0 | | | | | RF5PPS[6:0] | | | |
| 0x022F | RF6PPS | 7:0 | | | | | RF6PPS[6:0] | | | |
| 0x0230 | RF7PPS | 7:0 | | | | | RF7PPS[6:0] | | | |
| 0x0231 ... 0x023D | Reserved | | | | | | | | | |
| 0x023E | INT0PPS | 7:0 | | | | | PORT | PIN[2:0] | | |
| 0x023F | INT1PPS | 7:0 | | | | PORT[1:0] | | PIN[2:0] | | |
| 0x0240 | INT2PPS | 7:0 | | | PORT[2:0] | | | PIN[2:0] | | |
| 0x0241 | T0CKIPPS | 7:0 | | | PORT[2:0] | | | PIN[2:0] | | |
| 0x0242 | T1CKIPPS | 7:0 | | | PORT[2:0] | | | PIN[2:0] | | |
| 0x0243 | T1GPPS | 7:0 | | | | PORT[1:0] | | PIN[2:0] | | |

.....continued

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0244 | T3CKIPPS | 7:0 | | | PORT[2:0] | | | PIN[2:0] | | |
| 0x0245 | T3GPPS | 7:0 | | | | PORT[1:0] | | PIN[2:0] | | |
| 0x0246 | T5CKIPPS | 7:0 | | | PORT[2:0] | | | PIN[2:0] | | |
| 0x0247 | T5GPPS | 7:0 | | | | PORT[1:0] | | PIN[2:0] | | |
| 0x0248 | T2INPPS | 7:0 | | | | PORT[1:0] | | PIN[2:0] | | |
| 0x0249 | T4INPPS | 7:0 | | | | PORT[1:0] | | PIN[2:0] | | |
| 0x024A | T6INPPS | 7:0 | | | | PORT[1:0] | | PIN[2:0] | | |
| 0x024B ... 0x024E | Reserved | | | | | | | | | |
| 0x024F | CCP1PPS | 7:0 | | | PORT[2:0] | | | PIN[2:0] | | |
| 0x0250 | CCP2PPS | 7:0 | | | PORT[2:0] | | | PIN[2:0] | | |
| 0x0251 | CCP3PPS | 7:0 | | | | PORT[1:0] | | PIN[2:0] | | |
| 0x0252 | Reserved | | | | | | | | | |
| 0x0253 | PWM1ERSPPS | 7:0 | | | | PORT[1:0] | | PIN[2:0] | | |
| 0x0254 | PWM2ERSPPS | 7:0 | | | PORT[2:0] | | | PIN[2:0] | | |
| 0x0255 | PWM3ERSPPS | 7:0 | | | | PORT[1:0] | | PIN[2:0] | | |
| 0x0256 | Reserved | | | | | | | | | |
| 0x0257 | PWMIN0PPS | 7:0 | | | PORT[2:0] | | | PIN[2:0] | | |
| 0x0258 | PWMIN1PPS | 7:0 | | | PORT[2:0] | | | PIN[2:0] | | |
| 0x0259 | SMT1WINPPS | 7:0 | | | PORT[2:0] | | | PIN[2:0] | | |
| 0x025A | SMT1SIGPPS | 7:0 | | | PORT[2:0] | | | PIN[2:0] | | |
| 0x025B | CWG1PPS | 7:0 | | | | PORT[1:0] | | PIN[2:0] | | |
| 0x025C | CWG2PPS | 7:0 | | | | PORT[1:0] | | PIN[2:0] | | |
| 0x025D | CWG3PPS | 7:0 | | | | PORT[1:0] | | PIN[2:0] | | |
| 0x025E | MD1CARLPPS | 7:0 | | | | PORT[1:0] | | PIN[2:0] | | |
| 0x025F | MD1CARHPPS | 7:0 | | | | PORT[1:0] | | PIN[2:0] | | |
| 0x0260 | MD1SRCPPS | 7:0 | | | | PORT[1:0] | | PIN[2:0] | | |
| 0x0261 | CLCIN0PPS | 7:0 | | | | PORT[1:0] | | PIN[2:0] | | |
| 0x0262 | CLCIN1PPS | 7:0 | | | | PORT[1:0] | | PIN[2:0] | | |
| 0x0263 | CLCIN2PPS | 7:0 | | | | PORT[1:0] | | PIN[2:0] | | |
| 0x0264 | CLCIN3PPS | 7:0 | | | | PORT[1:0] | | PIN[2:0] | | |
| 0x0265 | CLCIN4PPS | 7:0 | | | | PORT[1:0] | | PIN[2:0] | | |
| 0x0266 | CLCIN5PPS | 7:0 | | | | PORT[1:0] | | PIN[2:0] | | |
| 0x0267 | CLCIN6PPS | 7:0 | | | | PORT[1:0] | | PIN[2:0] | | |
| 0x0268 | CLCIN7PPS | 7:0 | | | | PORT[1:0] | | PIN[2:0] | | |
| 0x0269 | ADACTPPS | 7:0 | | | | PORT[1:0] | | PIN[2:0] | | |
| 0x026A | SPI1SCKPPS | 7:0 | | | | PORT[1:0] | | PIN[2:0] | | |
| 0x026B | SPI1SDIPPS | 7:0 | | | | PORT[1:0] | | PIN[2:0] | | |
| 0x026C | SPI1SSPPS | 7:0 | | | | PORT[1:0] | | PIN[2:0] | | |
| 0x026D | SPI2SCKPPS | 7:0 | | | | PORT[1:0] | | PIN[2:0] | | |
| 0x026E | SPI2SDIPPS | 7:0 | | | | PORT[1:0] | | PIN[2:0] | | |
| 0x026F | SPI2SSPPS | 7:0 | | | | PORT[1:0] | | PIN[2:0] | | |
| 0x0270 | I2C1SDAPPS | 7:0 | | | | PORT[1:0] | | PIN[2:0] | | |
| 0x0271 | I2C1SCLPPS | 7:0 | | | | PORT[1:0] | | PIN[2:0] | | |
| 0x0272 | U1RXPPS | 7:0 | | | PORT[2:0] | | | PIN[2:0] | | |
| 0x0273 | U1CTSPPS | 7:0 | | | PORT[2:0] | | | PIN[2:0] | | |
| 0x0274 | U2RXPPS | 7:0 | | | | PORT[1:0] | | PIN[2:0] | | |
| 0x0275 | U2CTSPPS | 7:0 | | | | PORT[1:0] | | PIN[2:0] | | |
| 0x0276 | U3RXPPS | 7:0 | | | PORT[2:0] | | | PIN[2:0] | | |
| 0x0277 | U3CTSPPS | 7:0 | | | PORT[2:0] | | | PIN[2:0] | | |
| 0x0278 | U4RXPPS | 7:0 | | | | PORT[1:0] | | PIN[2:0] | | |
| 0x0279 | U4CTSPPS | 7:0 | | | | PORT[1:0] | | PIN[2:0] | | |
| 0x027A | U5RXPPS | 7:0 | | | PORT[2:0] | | | PIN[2:0] | | |
| 0x027B | U5CTSPPS | 7:0 | | | PORT[2:0] | | | PIN[2:0] | | |

© 2019-2023 Microchip Technology Inc. and its subsidiaries

# 22. CLC - Configurable Logic Cell

The Configurable Logic Cell (CLC) module provides programmable logic that operates outside the speed limitations of software execution. The logic cell takes up to 256 input signals and, through the use of configurable gates, reduces those inputs to four logic lines that drive one of eight selectable single-output logic functions.

Input sources are a combination of the following:

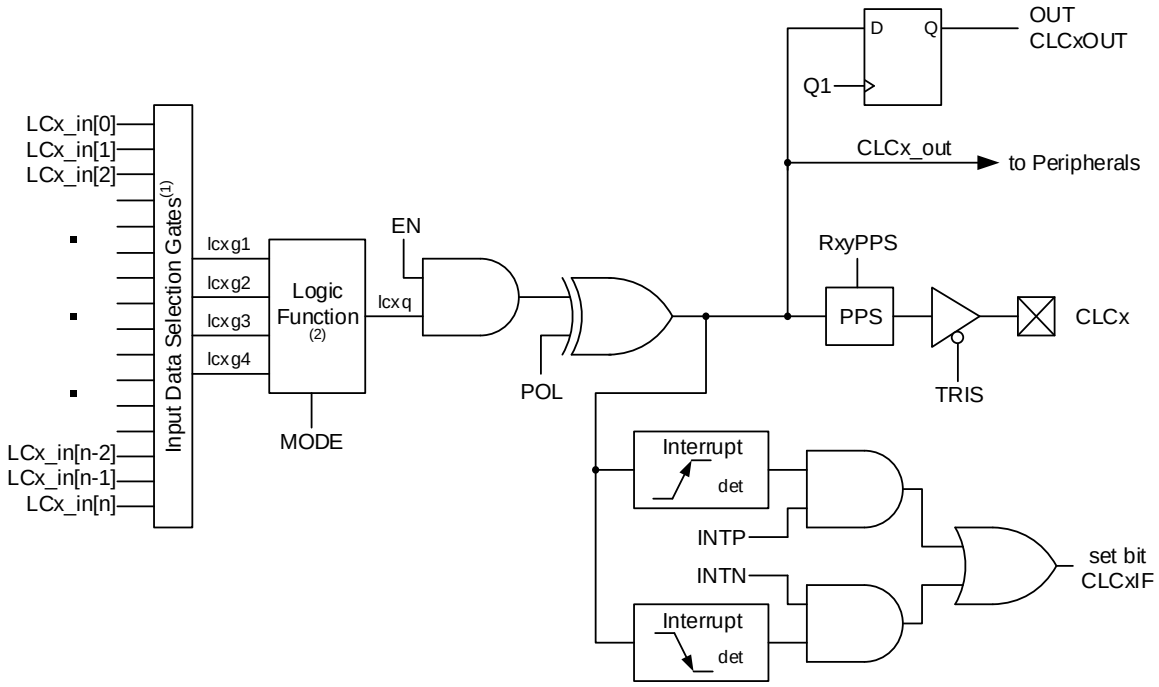
- I/O pins
- Internal clocks
- Peripherals
- Register bits

The output can be directed internally to peripherals and to an output pin.

The following figure is a simplified diagram showing signal flow through the CLC. Possible configurations include:

- Combinatorial Logic
  - AND
  - NAND
  - AND-OR
  - AND-OR-INVERT
  - OR-XOR
  - OR-XNOR
- Latches
  - SR
  - Clocked D with Set and Reset
  - Transparent D with Set and Reset

**Figure 22-1. CLC Simplified Block Diagram**



**Notes:**

1. See Figure 22-2 for input data selection and gating.
2. See Figure 22-3 for programmable logic functions.

## 22.1 CLC Setup

Programming the CLC module is performed by configuring the four stages in the logic signal flow. The four stages are:

- Data selection
- Data gating
- Logic function selection
- Output polarity

Each stage is set up at run time by writing to the corresponding CLC Special Function Registers. This has the added advantage of permitting logic reconfiguration on-the-fly during program execution.

### 22.1.1 Data Selection

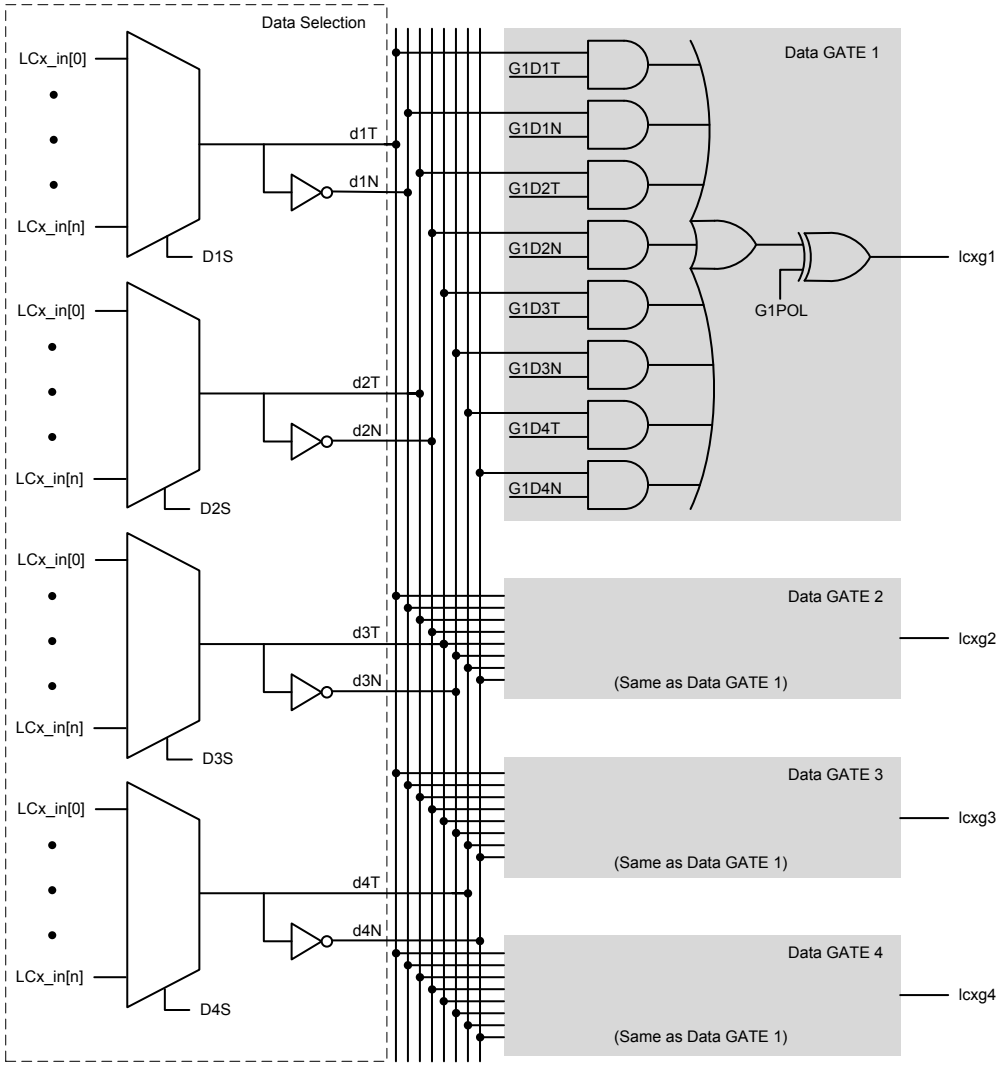
Data inputs are selected with CLCnSEL0 through CLCnSEL3 registers.

**Important:** Data selections are undefined at power-up.

Depending on the number of bits implemented in the CLCnSELy registers, there can be as many as 256 sources available as inputs to the configurable logic. Four multiplexers are used to independently select these inputs to pass on to the next stage as indicated on the left side of the following diagram.

Data inputs in the figure are identified by a generic numbered input name.

**Figure 22-2. Input Data Selection and Gating**




**Note:** All controls are undefined at power-up.

The CLC Input Selection table correlates the generic input name to the actual signal for each CLC module. The table column labeled ‘DyS Value’ indicates the MUX selection code for the selected data input. DyS is an abbreviation for the MUX select input codes, D1S through D4S, where ‘y’ is the gate number.

### 22.1.2 Data Gating

Outputs from the input multiplexers are directed to the desired logic function input through the data gating stage. Each data gate can direct any combination of the four selected inputs.

The gate stage is more than just signal direction. The gate can be configured to direct each input signal as inverted or noninverted data. Directed signals are ANDed together in each gate. The output of each gate can be inverted before going on to the logic function stage.

The gating is in essence a 1-to-4 input AND/NAND/OR/NOR gate. When every input is inverted and the output is inverted, the gate is an AND of all enabled data inputs. When the inputs and output are not inverted, the gate is an OR or all enabled inputs.

Table 22-1 summarizes the basic logic that can be obtained in gate 1 by using the gate logic select bits. The table shows the logic of four input variables, but each gate can be configured to use less than four. If no inputs are selected, the output will be ‘0’ or ‘1’, depending on the gate output polarity bit.

**Table 22-1. Data Gating Logic**

| CLCnGLSy | GyPOL | Gate Logic |
|---|---|---|
| 0x55 | 1 | AND |
| 0x55 | 0 | NAND |
| 0xAA | 1 | NOR |
| 0xAA | 0 | OR |
| 0x00 | 0 | Logic '0' |
| 0x00 | 1 | Logic '1' |

It is possible (but not recommended) to select both the true and negated values of an input. When this is done, the gate output is '0', regardless of the other inputs, but may emit logic glitches (transient-induced pulses). If the output of the channel must be '0' or '1', the recommended method is to set all gate bits to '0' and use the gate polarity bit to set the desired level.

Data gating is configured with the logic gate select registers as follows:

- Gate 1: CLCnGLS0
- Gate 2: CLCnGLS1
- Gate 3: CLCnGLS2
- Gate 4: CLCnGLS3

**Note:** Register number suffixes are different than the gate numbers because other variations of this module have multiple gate selections in the same register.

Data gating is indicated in the right side of Figure 22-2. Only one gate is shown in detail. The remaining three gates are configured identically, except when the data enables correspond to the enables for that gate.

### 22.1.3 Logic Function

There are eight available logic functions including:

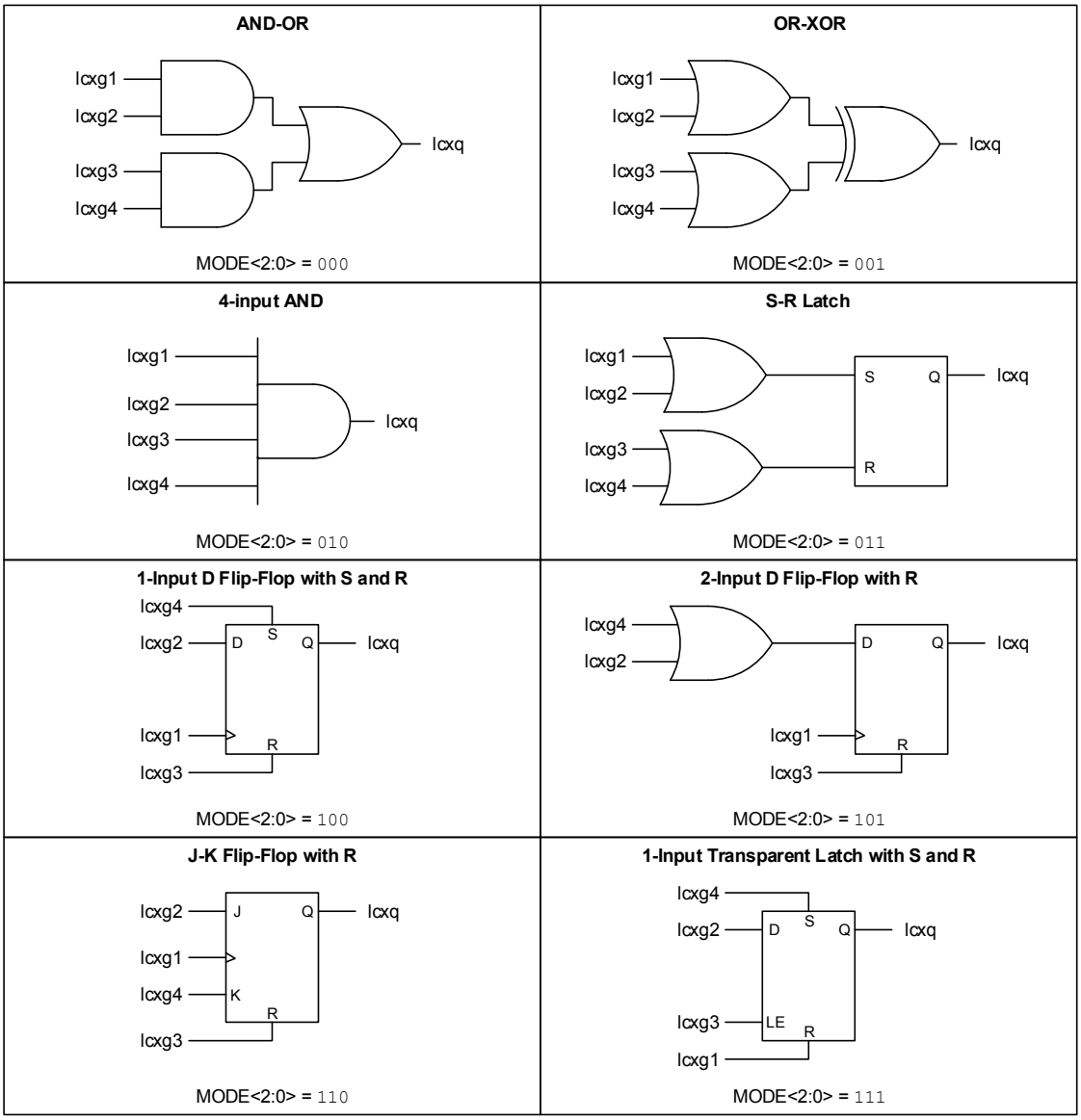
- AND-OR
- OR-XOR
- AND
- SR Latch
- D Flip-Flop with Set and Reset
- D Flip-Flop with Reset
- J-K Flip-Flop with Reset
- Transparent Latch with Set and Reset

Logic functions are shown in the following diagram. Each logic function has four inputs and one output. The four inputs are the four data gate outputs of the previous stage. The output is fed to the inversion stage and from there to other peripherals, an output pin, and back to the CLC itself.

© 2019-2023 Microchip Technology Inc. and its subsidiaries

**Figure 22-3. Programmable Logic Functions**

Rev. 10-000122B
9/13/2016




### 22.1.4 Output Polarity

The last stage in the Configurable Logic Cell is the output polarity. Setting the POL bit inverts the output signal from the logic stage. Changing the polarity while the interrupts are enabled will cause an interrupt for the resulting output transition.

## 22.2 CLC Interrupts

An interrupt will be generated upon a change in the output value of the CLCx when the appropriate interrupt enables are set. A rising edge detector and a falling edge detector are present in each CLC for this purpose.

The CLCxIF bit of the associated PIR register will be set when either edge detector is triggered and its associated enable bit is set. The INTP bit enables rising edge interrupts and the INTN bit enables falling edge interrupts.

To fully enable the interrupt, set the following bits:

- The CLCxIE bit of the respective PIE register
- The INTP bit (for a rising edge detection)
- The INTN bit (for a falling edge detection)

If priority interrupts are not used:

1. Clear the IPEN bit of the INTCON register.
2. Set the GIE bit of the INTCON register.
3. Set the GIEL bit of the INTCON register.

If the CLC is a high-priority interrupt:

1. Set the IPEN bit of the INTCON register.
2. Set the CLCxIP bit of the respective IPR register.
3. Set the GIEH bit of the INTCON register.

If the CLC is a low-priority interrupt:

1. Set the IPEN bit of the INTCON register.
2. Clear the CLCxIP bit of the respective IPR register.
3. Set the GIEL bit of the INTCON register.

The CLCxIF bit of the respective PIR register must be cleared in software as part of the interrupt service. If another edge is detected while this flag is being cleared, the flag will still be set at the end of the sequence.

## 22.3 Effects of a Reset

The CLCnCON register is cleared to '0' as the result of a Reset. All other selection and gating values remain unchanged.

## 22.4 Output Mirror Copies

Mirror copies of all CLCxOUT bits are contained in the CLCDATA register. Reading this register reads the outputs of all CLCs simultaneously. This prevents any reading skew introduced by testing or reading the OUT bits in the individual CLCnCON registers.

## 22.5 Operation During Sleep

The CLC module operates independently from the system clock and will continue to run during Sleep, provided that the input sources selected remain Active.

The HFINTOSC remains Active during Sleep when the CLC module is enabled and the HFINTOSC is selected as an input source, regardless of the system clock source selected.

In other words, if the HFINTOSC is simultaneously selected as both the system clock and as a CLC input source then, when the CLC is enabled, the CPU will go Idle during Sleep, but the CLC will continue to operate and the HFINTOSC will remain Active. This will have a direct effect on the Sleep mode current.

## 22.6 CLC Setup Steps

These steps need to be followed when setting up the CLC:

1. Disable the CLC by clearing the EN bit.
2. Select the desired inputs using the CLCnSEL0 through CLCnSEL3 registers.
3. Clear any ANSEL bits associated with CLC input pins.
4. Set all TRIS bits associated with inputs. However, a CLC input will also operate if the pin is configured as an output, in which case the TRIS bits must be cleared.

© 2019-2023 Microchip Technology Inc. and its subsidiaries

5. Enable the chosen inputs through the four gates using the CLCnGLS0 through CLCnGLS3 registers.
6. Select the gate output polarities with the GyPOL bits.
7. Select the desired logic function with the MODE bits.
8. Select the desired polarity of the logic output with the POL bit (this step may be combined with the previous gate output polarity step).
9. If driving a device pin, configure the associated pin PPS control register and also clear the TRIS bit corresponding to that output.
10. Configure the interrupts (optional). See the CLC Interrupts section.
11. Enable the CLC by setting the EN bit.

## 22.7 Register Overlay

All CLCs in this device share the same set of registers. Only one CLC instance is accessible at a time. The value in the CLCSELECT register is one less than the selected CLC instance. For example, a CLCSELECT value of '0' selects CLC1.

## 22.8 Register Definitions: Configurable Logic Cell

### 22.8.1 CLCSELECT

**Name:** CLCSELECT
**Address:** 0x0D5

CLC Instance Selection Register

Selects which CLC instance is accessed by the CLC registers

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | SLCT[2:0] | | |
| Access | | | | | | R/W | R/W | R/W |
| Reset | | | | | | 0 | 0 | 0 |

**Bits 2:0 – SLCT[2:0]** CLC instance selection

| Value | Description |
|---|---|
| n | Shared CLC registers of instance n+1 are selected for read and write operations |

### 22.8.2 CLCnCON

**Name:** CLCnCON
**Address:** 0x0D6

Configurable Logic Cell Control Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | EN | | OUT | INTP | INTN | MODE[2:0] | | |
| Access | R/W | | R | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | | 0 | 0 | 0 | 0 | 0 | 0 |

**Bit 7 – EN** CLC Enable

| Value | Description |
|---|---|
| 1 | Configurable logic cell is enabled and mixing signals |
| 0 | Configurable logic cell is disabled and has logic zero output |

**Bit 5 – OUT** Logic cell output data, after LCPOL. Sampled from CLCxOUT.

**Bit 4 – INTP** Configurable Logic Cell Positive Edge Going Interrupt Enable

| Value | Description |
|---|---|
| 1 | CLCxIF will be set when a rising edge occurs on CLCxOUT |
| 0 | Rising edges on CLCxOUT have no effect on CLCxIF |

**Bit 3 – INTN** Configurable Logic Cell Negative Edge Going Interrupt Enable

| Value | Description |
|---|---|
| 1 | CLCxIF will be set when a falling edge occurs on CLCxOUT |
| 0 | Falling edges on CLCxOUT have no effect on CLCxIF |

**Bits 2:0 – MODE[2:0]** Configurable Logic Cell Functional Mode Selection

| Value | Description |
|---|---|
| 111 | Cell is 1-input transparent latch with Set and Reset |
| 110 | Cell is J-K flip-flop with Reset |
| 101 | Cell is 2-input D flip-flop with Reset |
| 100 | Cell is 1-input D flip-flop with Set and Reset |
| 011 | Cell is SR latch |
| 010 | Cell is 4-input AND |
| 001 | Cell is OR-XOR |
| 000 | Cell is AND-OR |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

### 22.8.3 CLCnPOL

**Name:** CLCnPOL
**Address:** 0x0D7

Signal Polarity Control Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | POL | | | | G4POL | G3POL | G2POL | G1POL |
| Access | R/W | | | | R/W | R/W | R/W | R/W |
| Reset | 0 | | | | x | x | x | x |

**Bit 7 – POL** CLCxOUT Output Polarity Control

| Value | Description |
|---|---|
| 1 | The output of the logic cell is inverted |
| 0 | The output of the logic cell is not inverted |

**Bits 0, 1, 2, 3 – GyPOL** Gate Output Polarity Control
Reset States: POR/BOR = xxxx
All Other Resets = uuuu

| Value | Description |
|---|---|
| 1 | The gate output is inverted when applied to the logic cell |
| 0 | The output of the gate is not inverted |

### 22.8.4 CLCnSEL0

**Name:** CLCnSEL0
**Address:** 0x0D8

Generic CLCn Data 1 Select Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | D1S[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | x | x | x | x | x | x | x | x |

**Bits 7:0 – D1S[7:0]** CLCn Data1 Input Selection

**Table 22-2. CLC Input Selection**

| DyS | Input Source | DyS (cont.) | Input Source (cont.) | DyS (cont.) | Input Source (cont.) |
|---|---|---|---|---|---|
| [0] 0000 0000 | CLCIN0PPS | [32] 0010 0000 | CCP2 | [64] 0100 0000 | SPI1_SDO |
| [1] 0000 0001 | CLCIN1PPS | [33] 0010 0001 | CCP3 | [65] 0100 0001 | SPI1_SCK |
| [2] 0000 0010 | CLCIN2PPS | [34] 0010 0010 | PWM1S1P1_OUT | [66] 0100 0010 | SPI1_SS |
| [3] 0000 0011 | CLCIN3PPS | [35] 0010 0011 | PWM1S1P2_OUT | [67] 0100 0011 | SPI2_SDO |
| [4] 0000 0100 | CLCIN4PPS | [36] 0010 0100 | PWM2S1P1_OUT | [68] 0100 0100 | SPI2_SCK |
| [5] 0000 0101 | CLCIN5PPS | [37] 0010 0101 | PWM2S1P2_OUT | [69] 0100 0101 | SPI2_SS |
| [6] 0000 0110 | CLCIN6PPS | [38] 0010 0110 | PWM3S1P1_OUT | [70] 0100 0110 | $I^2C$_SCL |
| [7] 0000 0111 | CLCIN7PPS | [39] 0010 0111 | PWM3S1P2_OUT | [71] 0100 0111 | $I^2C$_SDA |
| [8] 0000 1000 | $F_{OSC}$ | [40] 0010 1000 | — | [72] 0100 1000 | CWG1A |
| [9] 0000 1001 | HFINTOSC(1) | [41] 0010 1001 | — | [73] 0100 1001 | CWG1B |
| [10] 0000 1010 | LFINTOSC(1) | [42] 0010 1010 | NCO1 | [74] 0100 1010 | CWG2A |
| [11] 0000 1011 | MFINTOSC(1) | [43] 0010 1011 | NCO2 | [75] 0100 1011 | CWG2B |
| [12] 0000 1100 | MFINTOSC (31.25 kHz)(1) | [44] 0010 1100 | NCO3 | [76] 0100 1100 | CWG3A |
| [13] 0000 1101 | SFINTOSC (1 MHz)(1) | [45] 0010 1101 | CMP1_OUT | [77] 0100 1101 | CWG3B |
| [14] 0000 1110 | SOSC(1) | [46] 0010 1110 | CMP2_OUT | ... | — |
| [15] 0000 1111 | EXTOSC(1) | [47] 0010 1111 | ZCD | ... | — |
| [16] 0001 0000 | ADCRC(1) | [48] 0011 0000 | IOC | ... | — |
| [17] 0001 0001 | CLKR | [49] 0011 0001 | DSM1 | ... | — |
| [18] 0001 0010 | TMR0 | [50] 0011 0010 | HLVD_OUT | ... | — |
| [19] 0001 0011 | TMR1 | [51] 0011 0011 | CLC1 | ... | — |
| [20] 0001 0100 | TMR2 | [52] 0011 0100 | CLC2 | ... | — |
| [21] 0001 0101 | TMR3 | [53] 0011 0101 | CLC3 | ... | — |
| [22] 0001 0110 | TMR4 | [54] 0011 0110 | CLC4 | ... | — |
| [23] 0001 0111 | TMR5 | [55] 0011 0111 | CLC5 | ... | — |
| [24] 0001 1000 | TMR6 | [56] 0011 1000 | CLC6 | ... | — |
| [25] 0001 1001 | — | [57] 0011 1001 | CLC7 | ... | — |
| [26] 0001 1010 | — | [58] 0011 1010 | CLC8 | ... | — |
| [27] 0001 1011 | — | [59] 0011 1011 | U1TX | ... | — |
| [28] 0001 1100 | — | [60] 0011 1100 | U2TX | ... | — |
| [29] 0001 1101 | — | [61] 0011 1101 | U3TX | ... | — |
| [30] 0001 1110 | SMT1 | [62] 0011 1110 | U4TX | ... | — |
| [31] 0001 1111 | CCP1 | [63] 0011 1111 | U5TX | [127] 0111 1111 | — |

**Note:**
1. Requests clock.

Reset States: POR/BOR = xxxxxxxx
All Other Resets = uuuuuuuu

### 22.8.5 CLCnSEL1

**Name:** CLCnSEL1
**Address:** 0x0D9

Generic CLCn Data 1 Select Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | D2S[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | x | x | x | x | x | x | x | x |

**Bits 7:0 – D2S[7:0]** CLCn Data2 Input Selection
Reset States: POR/BOR = xxxxxxxx
All Other Resets = uuuuuuuu

| Value | Description |
|---|---|
| n | Refer to the CLC Input Selection table for input selections |

### 22.8.6 CLCnSEL2

**Name:** CLCnSEL2
**Address:** 0x0DA

Generic CLCn Data 1 Select Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | D3S[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | x | x | x | x | x | x | x | x |

**Bits 7:0 – D3S[7:0]** CLCn Data3 Input Selection
Reset States: POR/BOR = xxxxxxxx
All Other Resets = uuuuuuuu

| Value | Description |
|---|---|
| n | Refer to the CLC Input Selection table for input selections |

### 22.8.7 CLCnSEL3

**Name:** CLCnSEL3
**Address:** 0x0DB

Generic CLCn Data 4 Select Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | D4S[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | x | x | x | x | x | x | x | x |

**Bits 7:0 – D4S[7:0]** CLCn Data4 Input Selection
Reset States: POR/BOR = xxxxxxxx
All Other Resets = uuuuuuuu

| Value | Description |
|---|---|
| n | Refer to the CLC Input Selection table for input selections |

### 22.8.8 CLCnGLS0

**Name:** CLCnGLS0
**Address:** 0x0DC

CLCn Gate1 Logic Select Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | G1D4T | G1D4N | G1D3T | G1D3N | G1D2T | G1D2N | G1D1T | G1D1N |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | x | x | x | x | x | x | x | x |

**Bits 1, 3, 5, 7 – G1DyT** dyT: Gate1 Data ‘y’ True (noninverted)
Reset States: POR/BOR = xxxx
All Other Resets = uuuu

| Value | Description |
|---|---|
| 1 | dyT is gated into g1 |
| 0 | dyT is not gated into g1 |

**Bits 0, 2, 4, 6 – G1DyN** dyN: Gate1 Data ‘y’ Negated (inverted)
Reset States: POR/BOR = xxxx
All Other Resets = uuuu

| Value | Description |
|---|---|
| 1 | dyN is gated into g1 |
| 0 | dyN is not gated into g1 |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

### 22.8.9 CLCnGLS1

**Name:** CLCnGLS1
**Address:** 0x0DD

CLCn Gate2 Logic Select Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | G2D4T | G2D4N | G2D3T | G2D3N | G2D2T | G2D2N | G2D1T | G2D1N |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | x | x | x | x | x | x | x | x |

**Bits 1, 3, 5, 7 – G2DyT** dyT: Gate2 Data 'y' True (noninverted)
Reset States: POR/BOR = xxxx
All Other Resets = uuuu

| Value | Description |
|---|---|
| 1 | dyT is gated into g2 |
| 0 | dyT is not gated into g2 |

**Bits 0, 2, 4, 6 – G2DyN** dyN: Gate2 Data 'y' Negated (inverted)
Reset States: POR/BOR = xxxx
All Other Resets = uuuu

| Value | Description |
|---|---|
| 1 | dyN is gated into g2 |
| 0 | dyN is not gated into g2 |

© 2019-2023 Microchip Technology Inc. and its subsidiaries

### 22.8.10 CLCnGLS2

**Name:** CLCnGLS2
**Address:** 0x0DE

CLCn Gate3 Logic Select Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | G3D4T | G3D4N | G3D3T | G3D3N | G3D2T | G3D2N | G3D1T | G3D1N |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | x | x | x | x | x | x | x | x |

**Bits 1, 3, 5, 7 – G3DyT** dyT: Gate3 Data 'y' True (noninverted)
Reset States: POR/BOR = xxxx
All Other Resets = uuuu

| Value | Description |
|---|---|
| 1 | dyT is gated into g3 |
| 0 | dyT is not gated into g3 |

**Bits 0, 2, 4, 6 – G3DyN** dyN: Gate3 Data 'y' Negated (inverted)
Reset States: POR/BOR = xxxx
All Other Resets = uuuu

| Value | Description |
|---|---|
| 1 | dyN is gated into g3 |
| 0 | dyN is not gated into g3 |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

### 22.8.11 CLCnGLS3

**Name:** CLCnGLS3
**Address:** 0x0DF

CLCn Gate4 Logic Select Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | G4D4T | G4D4N | G4D3T | G4D3N | G4D2T | G4D2N | G4D1T | G4D1N |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | x | x | x | x | x | x | x | x |

**Bits 1, 3, 5, 7 – G4DyT** dyT: Gate4 Data 'y' True (noninverted)
Reset States: POR/BOR = xxxx
All Other Resets = uuuu

| Value | Description |
|---|---|
| 1 | dyT is gated into g4 |
| 0 | dyT is not gated into g4 |

**Bits 0, 2, 4, 6 – G4DyN** dyN: Gate4 Data 'y' Negated (inverted)
Reset States: POR/BOR = xxxx
All Other Resets = uuuu

| Value | Description |
|---|---|
| 1 | dyN is gated into g4 |
| 0 | dyN is not gated into g4 |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

### 22.8.12 CLCDATA

**Name:** CLCDATA
**Address:** 0x0D4

CLC Data Output Register

Mirror copy of CLC outputs

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | CLC8OUT | CLC7OUT | CLC6OUT | CLC5OUT | CLC4OUT | CLC3OUT | CLC2OUT | CLC1OUT |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 0, 1, 2, 3, 4, 5, 6, 7 – CLCxOUT** Mirror copy of CLCx_out

| Value | Description |
|---|---|
| 1 | CLCx_out is 1 |
| 0 | CLCx_out is 0 |

## 22.9 Register Summary - CLC Control

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0xD4 | CLCDATA | 7:0 | CLC8OUT | CLC7OUT | CLC6OUT | CLC5OUT | CLC4OUT | CLC3OUT | CLC2OUT | CLC1OUT |
| 0xD5 | CLCSELECT | 7:0 | | | | | | SLCT[2:0] | | |
| 0xD6 | CLCnCON | 7:0 | EN | | OUT | INTP | INTN | MODE[2:0] | | |
| 0xD7 | CLCnPOL | 7:0 | POL | | | | G4POL | G3POL | G2POL | G1POL |
| 0xD8 | CLCnSEL0 | 7:0 | D1S[7:0] | | | | | | | |
| 0xD9 | CLCnSEL1 | 7:0 | D2S[7:0] | | | | | | | |
| 0xDA | CLCnSEL2 | 7:0 | D3S[7:0] | | | | | | | |
| 0xDB | CLCnSEL3 | 7:0 | D4S[7:0] | | | | | | | |
| 0xDC | CLCnGLS0 | 7:0 | G1D4T | G1D4N | G1D3T | G1D3N | G1D2T | G1D2N | G1D1T | G1D1N |
| 0xDD | CLCnGLS1 | 7:0 | G2D4T | G2D4N | G2D3T | G2D3N | G2D2T | G2D2N | G2D1T | G2D1N |
| 0xDE | CLCnGLS2 | 7:0 | G3D4T | G3D4N | G3D3T | G3D3N | G3D2T | G3D2N | G3D1T | G3D1N |
| 0xDF | CLCnGLS3 | 7:0 | G4D4T | G4D4N | G4D3T | G4D3N | G4D2T | G4D2N | G4D1T | G4D1N |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

# 23. CLKREF - Reference Clock Output Module

The reference clock output module provides the ability to send a clock signal to the clock reference output pin (CLKR). The reference clock output can be routed internally as an input signal for other peripherals, such as the timers and CLCs.

The reference clock output module has the following features:

- Selectable clock source using the CLKRCLK register
- Programmable clock divider
- Selectable duty cycle

The figure below shows the simplified block diagram of the clock reference module.

**Figure 23-1. Clock Reference Block Diagram**

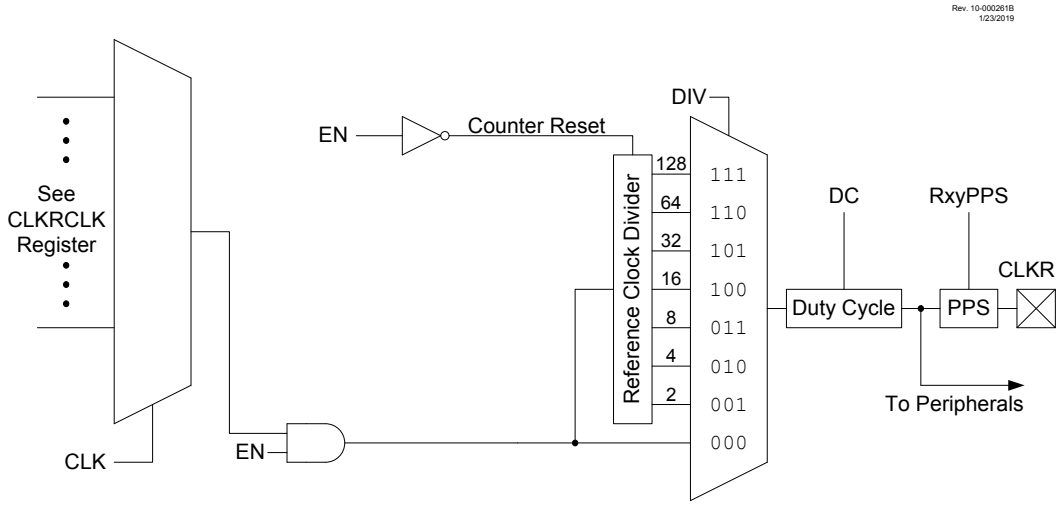
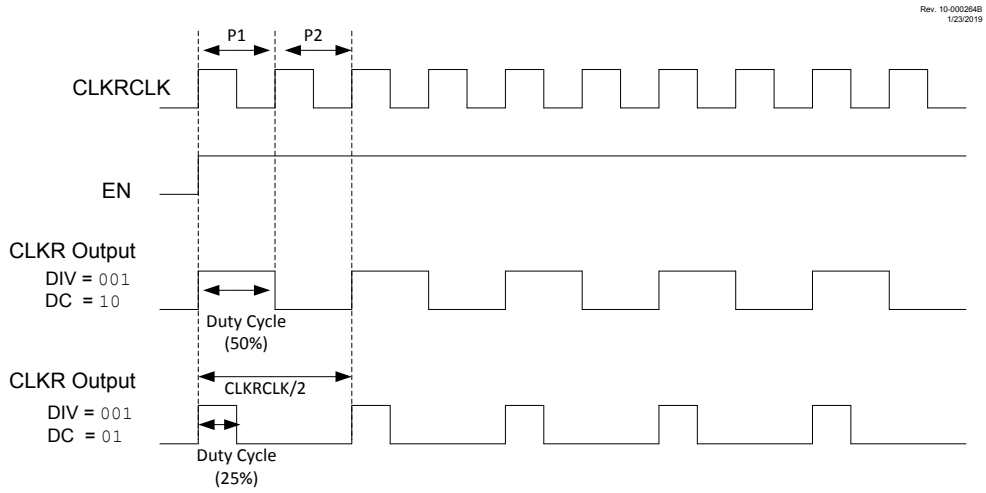



**Figure 23-2. Clock Reference Timing**




## 23.1 Clock Source

The clock source of the reference clock peripheral is selected with the CLK bits.

### 23.1.1 Clock Synchronization

The CLKR output signal is ensured to be glitch-free when the EN bit is set to start the module and enable the CLKR output. When the reference clock output is disabled, the output signal will be disabled immediately.

## 23.2 Programmable Clock Divider

The module takes the clock input and divides it based on the value of the DIV bits.

The following configurations are available:

- Base clock frequency value
- Base clock frequency divided by 2
- Base clock frequency divided by 4
- Base clock frequency divided by 8
- Base clock frequency divided by 16
- Base clock frequency divided by 32
- Base clock frequency divided by 64
- Base clock frequency divided by 128

## 23.3 Selectable Duty Cycle

The DC bits are used to modify the duty cycle of the output clock. A duty cycle of 0%, 25%, 50%, or 75% can be selected for all clock rates when the DIV value is not `0b000`. When DIV = `0b000`, the duty cycle defaults to 50% for all values of DC except `0b00`, in which case the duty cycle is 0% (constant low output).

**Important:** The DC value at Reset is `10`. This makes the default duty cycle 50% and not 0%.

**Important:** Clock dividers and clock duty cycles can be changed while the module is enabled but doing so may cause glitches to occur on the output. To avoid possible glitches, clock dividers and clock duty cycles will be changed only when the EN bit is clear.

## 23.4 Operation in Sleep Mode

The reference clock module continues to operate and provide a signal output in Sleep for all clock source selections except $F_{OSC}$ (CLK = `0`).

## 23.5 Register Definitions: Reference Clock

Long bit name prefixes for the Reference Clock peripherals are shown in the following table. Refer to the **"Long Bit Names"** section in the **"Register and Bit Naming Conventions"** chapter for more information.

**Table 23-1. CLKR Long Bit Name Prefixes**

| Peripheral | Bit Name Prefix |
|---|---|
| CLKR | CLKR |

### 23.5.1 CLKRCON

**Name:** CLKRCON
**Address:** 0x000039

Reference Clock Control Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | EN | | | DC[1:0] | | DIV[2:0] | | |
| Access | R/W | | | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | | | 1 | 0 | 0 | 0 | 0 |

**Bit 7 – EN** Reference Clock Module Enable

| Value | Description |
|---|---|
| 1 | Reference clock module enabled |
| 0 | Reference clock module is disabled |

**Bits 4:3 – DC[1:0]** Reference Clock Duty Cycle(1)

| Value | Description |
|---|---|
| 11 | Clock outputs duty cycle of 75% |
| 10 | Clock outputs duty cycle of 50% |
| 01 | Clock outputs duty cycle of 25% |
| 00 | Clock outputs duty cycle of 0% |

**Bits 2:0 – DIV[2:0]** Reference Clock Divider

| Value | Description |
|---|---|
| 111 | Base clock value divided by 128 |
| 110 | Base clock value divided by 64 |
| 101 | Base clock value divided by 32 |
| 100 | Base clock value divided by 16 |
| 011 | Base clock value divided by 8 |
| 010 | Base clock value divided by 4 |
| 001 | Base clock value divided by 2 |
| 000 | Base clock value |

**Note:**

1. Bits are valid for DIV ≥ 001. For DIV = 000, duty cycle is fixed at 50%.

### 23.5.2 CLKRCLK

**Name:** CLKRCLK
**Address:** 0x03A

Clock Reference Clock Selection Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | CLK[4:0] | | | | |
| Access | | | | R/W | R/W | R/W | R/W | R/W |
| Reset | | | | 0 | 0 | 0 | 0 | 0 |

**Bits 4:0 – CLK[4:0]** CLKR Clock Selection

**Table 23-2. Clock Reference Module Clock Sources**

| CLK | Clock Source |
|---|---|
| 11111-10010 | Reserved |
| 10001 | CLC8_OUT |
| 10000 | CLC7_OUT |
| 01111 | CLC6_OUT |
| 01110 | CLC5_OUT |
| 01101 | CLC4_OUT |
| 01100 | CLC3_OUT |
| 01011 | CLC2_OUT |
| 01010 | CLC1_OUT |
| 01001 | NCO3_OUT |
| 01000 | NCO2_OUT |
| 00111 | NCO1_OUT |
| 00110 | EXTOSC |
| 00101 | SOSC |
| 00100 | MFINTOSC (31.25 kHz) |
| 00011 | MFINTOSC (500 kHz) |
| 00010 | LFINTOSC |
| 00001 | HFINTOSC |
| 00000 | $F_{OSC}$ |

## 23.6 Register Summary - Reference CLK

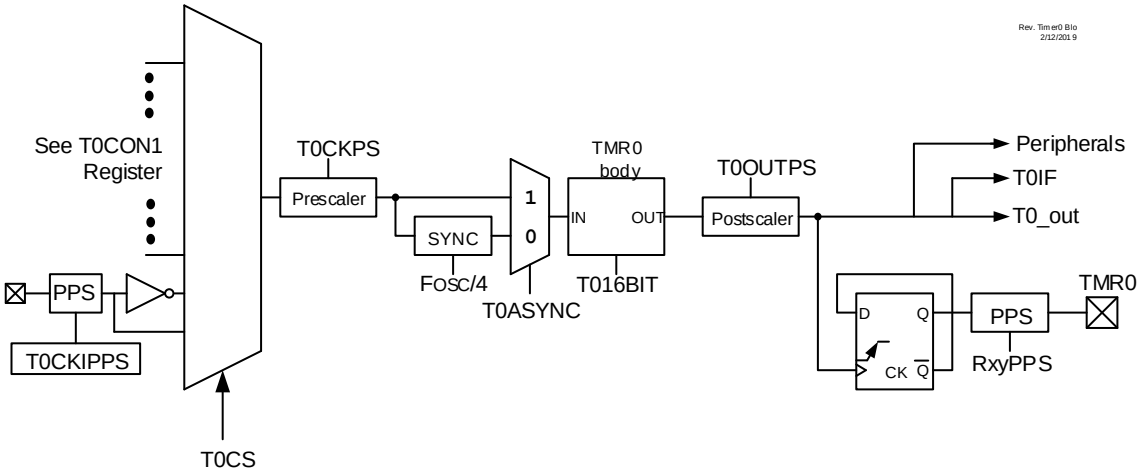
| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x39 | CLKRCON | 7:0 | EN | | | DC[1:0] | | DIV[2:0] | | |
| 0x3A | CLKRCLK | 7:0 | | | | CLK[4:0] | | | | |

# 24. TMR0 - Timer0 Module

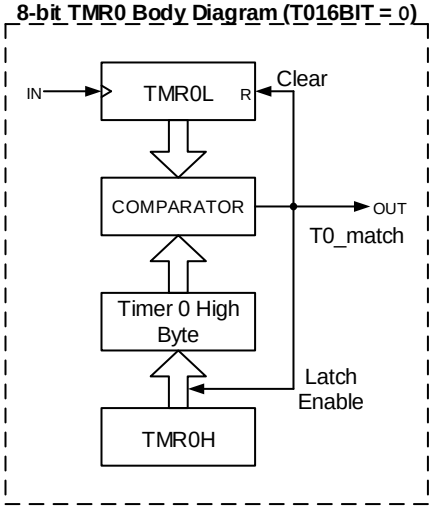
The Timer0 module has the following features:

- 8-bit timer with programmable period
- 16-bit timer
- Selectable clock sources
- Synchronous and asynchronous operation
- Programmable prescaler (Independent of Watchdog Timer)
- Programmable postscaler
- Interrupt on match or overflow
- Output on I/O pin (via PPS) or to other peripherals
- Operation during Sleep

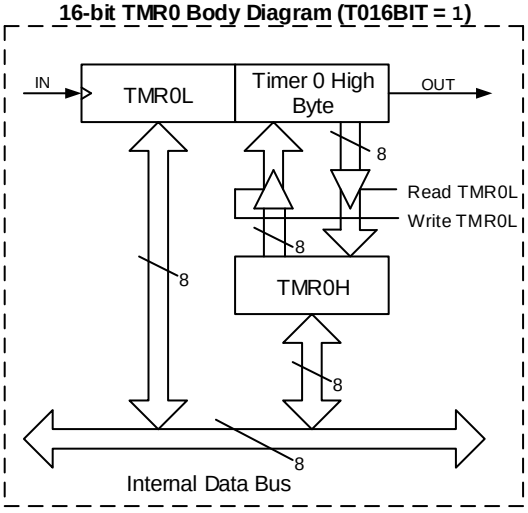
**Figure 24-1. Timer0 Block Diagram**




**8-bit TMR0 Body Diagram (T016BIT = 0)**




**16-bit TMR0 Body Diagram (T016BIT = 1)**




## 24.1 Timer0 Operation

Timer0 can operate as either an 8-bit or 16-bit timer. The mode is selected with the MD16 bit.

### 24.1.1 8-Bit Mode

In this mode, Timer0 increments on the rising edge of the selected clock source. A prescaler on the clock input gives several prescale options (see the prescaler control bits, CKPS). In this mode, as shown in Figure 24-1, a buffered version of TMR0H is maintained.

This is compared with the value of TMR0L on each cycle of the selected clock source. When the two values match, the following events occur:

- TMR0L is reset
- The contents of TMR0H are copied to the TMR0H buffer for next comparison

### 24.1.2 16-Bit Mode

In this mode, Timer0 increments on the rising edge of the selected clock source. A prescaler on the clock input gives several prescale options (see the prescaler control bits, CKPS). In this mode, TMR0H:TMR0L form the 16-bit timer value. As shown in Figure 24-1, reads and writes of the TMR0H register are buffered. The TMR0H register is updated with the contents of the high byte of Timer0 when the TMR0L register is read. Similarly, writing the TMR0L register causes a transfer of the TMR0H register value to the Timer0 high byte.

This buffering allows all 16 bits of Timer0 to be read and written at the same time. Timer0 rolls over to `0x0000` on incrementing past `0xFFFF`. This makes the timer free-running. While actively operating in 16-bit mode, the Timer0 value can be read but not written.

## 24.2 Clock Selection

Timer0 has several options for clock source selections, the option to operate synchronously/asynchronously and an available programmable prescaler. The CS bits are used to select the clock source for Timer0.

### 24.2.1 Synchronous Mode

When the ASYNC bit is clear, Timer0 clock is synchronized to the system clock ($F_{OSC}/4$). When operating in Synchronous mode, Timer0 clock frequency cannot exceed $F_{OSC}/4$. During Sleep mode, the system clock is not available and Timer0 cannot operate.

### 24.2.2 Asynchronous Mode

When the ASYNC bit is set, Timer0 increments with each rising edge of the input source (or output of the prescaler, if used). Asynchronous mode allows Timer0 to continue operation during Sleep mode provided the selected clock source operates during Sleep.

### 24.2.3 Programmable Prescaler

Timer0 has 16 programmable input prescaler options ranging from 1:1 to 1:32768. The prescaler values are selected using the CKPS bits. The prescaler counter is not directly readable or writable. The prescaler counter is cleared on the following events:

- A write to the TMR0L register
- A write to either the T0CON0 or T0CON1 registers
- Any device Reset

### 24.2.4 Programmable Postscaler

Timer0 has 16 programmable output postscaler options ranging from 1:1 to 1:16. The postscaler values are selected using the OUTPS bits. The postscaler divides the output of Timer0 by the selected ratio. The postscaler counter is not directly readable or writable. The postscaler counter is cleared on the following events:

- A write to the TMR0L register
- A write to either the T0CON0 or T0CON1 registers
- Any device Reset

© 2019-2023 Microchip Technology Inc. and its subsidiaries

## 24.3 Timer0 Output and Interrupt

### 24.3.1 Timer0 Output

TMR0_out toggles on every match between TMR0L and TMR0H in 8-bit mode, or when TMR0H:TMR0L rolls over in 16-bit mode. If the output postscaler is used, the output is scaled by the ratio selected. The Timer0 output can be routed to an I/O pin via the RxyPPS output selection register, or internally to a number of Core Independent Peripherals. The Timer0 output can be monitored through software via the OUT output bit.

### 24.3.2 Timer0 Interrupt

The Timer0 Interrupt Flag (TMR0IF) bit is set when the TMR0_out toggles. If the Timer0 interrupt is enabled (TMR0IE), the CPU will be interrupted when the TMR0IF bit is set. When the postscaler bits (T0OUTPS) are set to 1:1 operation (no division), the T0IF flag bit will be set with every TMR0 match or rollover. In general, the TMR0IF flag bit will be set every T0OUTPS +1 matches or rollovers.

### 24.3.3 Timer0 Example

Timer0 Configuration:

- Timer0 mode = 16-bit
- Clock Source = $F_{OSC}/4$ (250 kHz)
- Synchronous operation
- Prescaler = 1:1
- Postscaler = 1:2 (T0OUTPS = `1`)

In this case, the TMR0_out toggles every two rollovers of TMR0H:TMR0L.
i.e., (0xFFFF)*2*(1/250 kHz) = 524.28 ms

## 24.4 Operation During Sleep

When operating synchronously, Timer0 will halt when the device enters Sleep mode. When operating asynchronously and the selected clock source is active, Timer0 will continue to increment and wake the device from Sleep mode if the Timer0 interrupt is enabled.

## 24.5 Register Definitions: Timer0 Control

### 24.5.1 T0CON0

**Name:** T0CON0
**Address:** 0x31A

Timer0 Control Register 0

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | EN | | OUT | MD16 | OUTPS[3:0] | | | |
| Access | R/W | | R | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | | 0 | 0 | 0 | 0 | 0 | 0 |

**Bit 7 – EN** TMR0 Enable

| Value | Description |
|---|---|
| 1 | The module is enabled and operating |
| 0 | The module is disabled |

**Bit 5 – OUT** TMR0 Output

**Bit 4 – MD16** 16-Bit Timer Operation Select

| Value | Description |
|---|---|
| 1 | TMR0 is a 16-bit timer |
| 0 | TMR0 is an 8-bit timer |

**Bits 3:0 – OUTPS[3:0]** TMR0 Output Postscaler (Divider) Select

| Value | Description |
|---|---|
| 1111 | 1:16 Postscaler |
| 1110 | 1:15 Postscaler |
| 1101 | 1:14 Postscaler |
| 1100 | 1:13 Postscaler |
| 1011 | 1:12 Postscaler |
| 1010 | 1:11 Postscaler |
| 1001 | 1:10 Postscaler |
| 1000 | 1:9 Postscaler |
| 0111 | 1:8 Postscaler |
| 0110 | 1:7 Postscaler |
| 0101 | 1:6 Postscaler |
| 0100 | 1:5 Postscaler |
| 0011 | 1:4 Postscaler |
| 0010 | 1:3 Postscaler |
| 0001 | 1:2 Postscaler |
| 0000 | 1:1 Postscaler |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

### 24.5.2 T0CON1

**Name:** T0CON1
**Address:** 0x31B

Timer0 Control Register 1

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | CS[2:0] | | | ASYNC | CKPS[3:0] | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 7:5 – CS[2:0]** Timer0 Clock Source Select

| Value | Description |
|---|---|
| 111 | CLC1_OUT |
| 110 | SOSC |
| 101 | MFINTOSC (500 kHz) |
| 100 | LFINTOSC |
| 011 | HFINTOSC |
| 010 | $F_{OSC}/4$ |
| 001 | Pin selected by T0CKIPPS (Inverted) |
| 000 | Pin selected by T0CKIPPS (Non-inverted) |

**Bit 4 – ASYNC** TMR0 Input Asynchronization Enable

| Value | Description |
|---|---|
| 1 | The input to the TMR0 counter is not synchronized to system clocks |
| 0 | The input to the TMR0 counter is synchronized to Fosc/4 |

**Bits 3:0 – CKPS[3:0]** Prescaler Rate Select

| Value | Description |
|---|---|
| 1111 | 1:32768 |
| 1110 | 1:16384 |
| 1101 | 1:8192 |
| 1100 | 1:4096 |
| 1011 | 1:2048 |
| 1010 | 1:1024 |
| 1001 | 1:512 |
| 1000 | 1:256 |
| 0111 | 1:128 |
| 0110 | 1:64 |
| 0101 | 1:32 |
| 0100 | 1:16 |
| 0011 | 1:8 |
| 0010 | 1:4 |
| 0001 | 1:2 |
| 0000 | 1:1 |

© 2019-2023 Microchip Technology Inc. and its subsidiaries

### 24.5.3 TMR0H

**Name:** TMR0H
**Address:** 0x319

Timer0 Period/Count High Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | TMR0H[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Bits 7:0 – TMR0H[7:0]** TMR0 Most Significant Counter

| Value | Condition | Description |
|---|---|---|
| 0 to 255 | MD16 = 0 | 8-bit Timer0 Period Value. TMR0L continues counting from 0 when this value is reached. |
| 0 to 255 | MD16 = 1 | 16-bit Timer0 Most Significant Byte |

### 24.5.4 TMR0L

**Name:** TMR0L
**Address:** 0x318

Timer0 Period/Count Low Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | TMR0L[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 7:0 – TMR0L[7:0]** TMR0 Least Significant Counter

| Value | Condition | Description |
|---|---|---|
| 0 to 255 | MD16 = 0 | 8-bit Timer0 Counter bits |
| 0 to 255 | MD16 = 1 | 16-bit Timer0 Least Significant Byte |

## 24.6 Register Summary - Timer0

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0318 | TMR0L | 7:0 | TMR0L[7:0] | | | | | | | |
| 0x0319 | TMR0H | 7:0 | TMR0H[7:0] | | | | | | | |
| 0x031A | T0CON0 | 7:0 | EN | | OUT | MD16 | OUTPS[3:0] | | | |
| 0x031B | T0CON1 | 7:0 | CS[2:0] | | | ASYNC | CKPS[3:0] | | | |

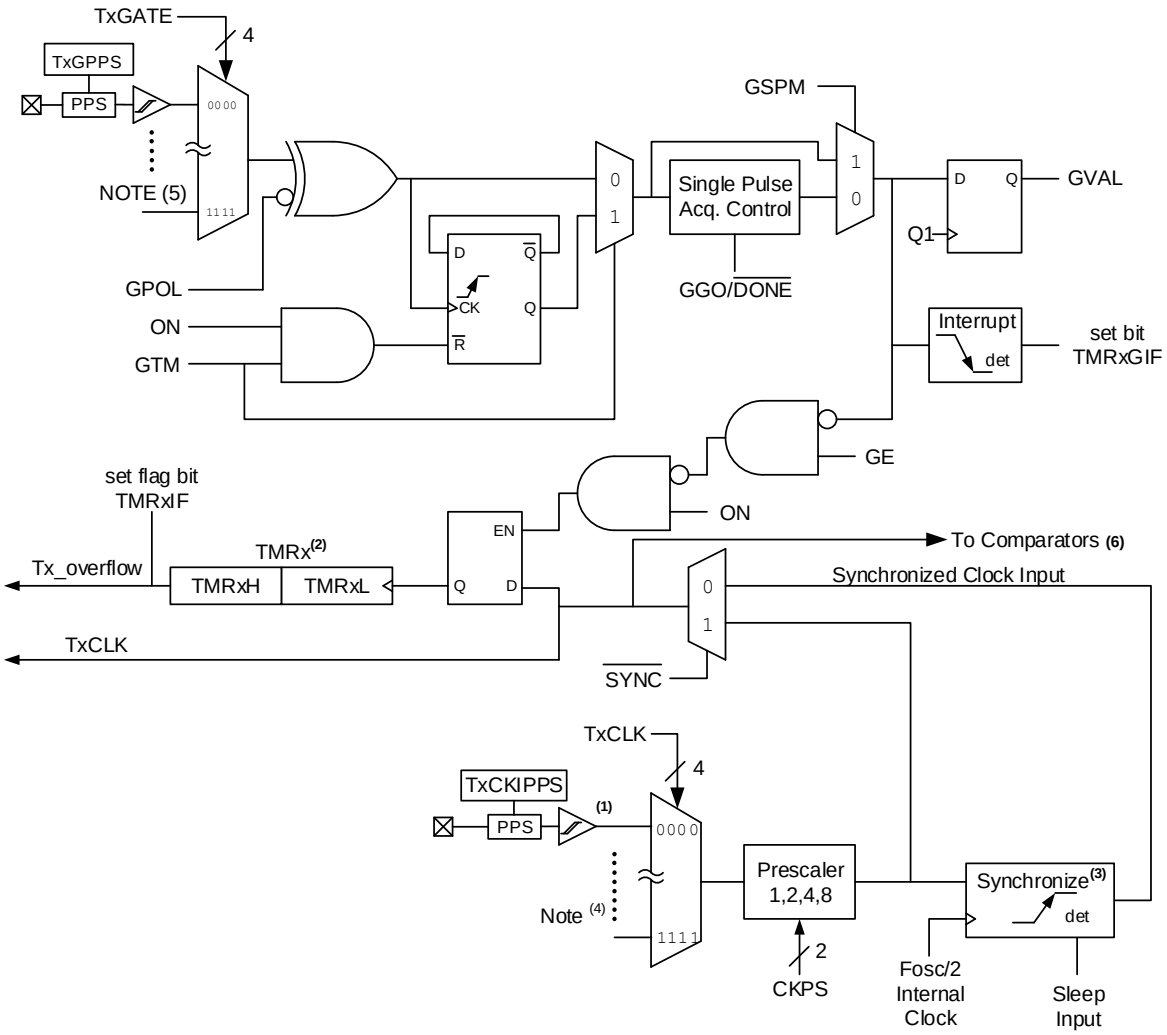
# 25. TMR1 - Timer1 Module with Gate Control

The Timer1 module is a 16-bit timer/counter with the following features:

- 16-bit timer/counter register pair (TMRxH:TMRxL)
- Programmable internal or external clock source
- 2-bit prescaler
- Clock source for optional comparator synchronization
- Multiple Timer1 gate (count enable) sources
- Interrupt-on-overflow
- Wake-up on overflow (external clock, Asynchronous mode only)
- 16-bit read/write operation
- Time base for the capture/compare function with the CCP modules
- Special event trigger (with CCP)
- Selectable gate source polarity
- Gate Toggle mode
- Gate Single Pulse mode
- Gate value status
- Gate event interrupt

**Important:** References to the module Timer1 apply to all the odd numbered timers on this device.

**Figure 25-1. Timer1 Block Diagram**




**Notes:**

1. This signal comes from the pin selected by Timer1 PPS register.
2. TMRx register increments on rising edge.
3. Synchronize does not operate while in Sleep.
4. See TxCLK for clock source selections.
5. See TxGATE for gate source selections.
6. Synchronized comparator output must not be used in conjunction with synchronized input clock.

## 25.1 Timer1 Operation

The Timer1 module is a 16-bit incrementing counter accessed through the TMRx register. Writes to TMRx directly update the counter. When used with an internal clock source, the module is a timer that increments on every instruction cycle. When used with an external clock source, the module can be used as either a timer or counter and increments on every selected edge of the external source.

Timer1 is enabled by configuring the ON and GE bits. Table 25-1 shows the possible Timer1 enable selections.

**Table 25-1. Timer1 Enable Selections**

| ON | GE | Timer1 Operation |
|---|---|---|
| 1 | 1 | Count enabled |
| 1 | 0 | Always on |
| 0 | 1 | Off |
| 0 | 0 | Off |

## 25.2 Clock Source Selection

The CS bits select the clock source for Timer1. These bits allow the selection of several possible synchronous and asynchronous clock sources.

### 25.2.1 Internal Clock Source

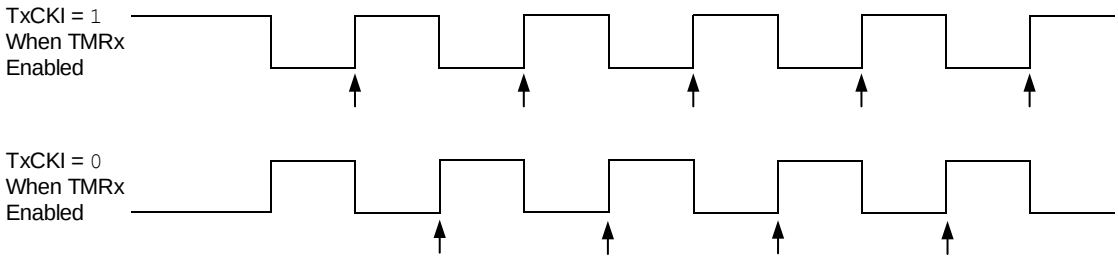
When the internal clock source is selected, the TMRx register will increment on multiples of $F_{OSC}$ as determined by the Timer1 prescaler.

When the $F_{OSC}$ internal clock source is selected, the TMRx register value will increment by four counts every instruction clock cycle. Due to this condition, a two LSB error in resolution will occur when reading the TMRx value. To utilize the full resolution of Timer1, an asynchronous input signal must be used to gate the Timer1 clock input.

**Important:** In Counter mode, a falling edge must be registered by the counter prior to the first incrementing rising edge after any one or more of the following conditions:

- Timer1 enabled after POR
- Write to TMRxH or TMRxL
- Timer1 is disabled
- Timer1 is disabled (ON = 0) when TxCKI is high, then Timer1 is enabled (ON = 1) when TxCKI is low. Refer to the figure below.

**Figure 25-2. Timer1 Incrementing Edge**




**Notes:**

1. Arrows indicate counter increments.
2. In Counter mode, a falling edge must be registered by the counter prior to the first incrementing rising edge of the clock.

### 25.2.2 External Clock Source

When the external clock source is selected, the TMRx module may work as a timer or a counter. When enabled to count, Timer1 is incremented on the rising edge of the external clock input of the TxCKIPPS pin. This external clock source can be synchronized to the system clock or it can run asynchronously.

## 25.3 Timer1 Prescaler

Timer1 has four prescaler options allowing 1, 2, 4 or 8 divisions of the clock input. The CKPS bits control the prescale counter. The prescale counter is not directly readable or writable; however, the prescaler counter is cleared upon a write to TMRx.

## 25.4 Secondary Oscillator

A secondary low-power 32.768 kHz oscillator circuit is built-in between pins SOSCI (input) and SOSCO (amplifier output). This internal circuit is to be used in conjunction with an external 32.768 kHz crystal. The secondary oscillator is not dedicated only to Timer1; it can also be used by other modules.

The oscillator circuit is enabled by setting the SOSCEN bit of the OSCEN register. This can be used as one of the Timer1 clock sources selected with the CS bits. The oscillator will continue to run during Sleep.

**Important:** The oscillator requires a start-up and stabilization time before use. Thus, the SOSCEN bit of the OSCEN register must be set and a suitable delay observed prior to enabling Timer1. A software check can be performed to confirm if the secondary oscillator is enabled and ready to use. This is done by polling the secondary oscillator ready Status bit. Refer to the **"OSC - Oscillator Module (with Fail-Safe Clock Monitor)"** chapter for more details.

## 25.5 Timer1 Operation in Asynchronous Counter Mode

When the SYNC Control bit is set, the external clock input is not synchronized. The timer increments asynchronously to the internal phase clocks. If the external clock source is selected, then the timer will continue to run during Sleep and can generate an interrupt on overflow, which will wake up the processor. However, special precautions in software are needed to read/write the timer.

**Important:** When switching from synchronous to asynchronous operation, it is possible to skip an increment. When switching from asynchronous to synchronous operation, it is possible to produce an additional increment.

### 25.5.1 Reading and Writing TMRx in Asynchronous Counter Mode

Reading TMRxH or TMRxL while the timer is running from an external asynchronous clock will ensure a valid read (taken care of in hardware). However, the user must keep in mind that reading the 16-bit timer in two 8-bit values itself poses certain problems, since there may be a carry-out of TMRxL to TMRxH between the reads.

For writes, it is recommended that the user simply stop the timer and write the desired values. A write contention may occur by writing to the timer registers, while the register is incrementing. This may produce an unpredictable value in the TMRxH:TMRxL register pair.

## 25.6 Timer1 16-Bit Read/Write Mode

Timer1 can be configured to read and write all 16 bits of data to and from the 8-bit TMRxL and TMRxH registers, simultaneously. The 16-bit read and write operations are enabled by setting the RD16 bit. To accomplish this function, the TMRxH register value is mapped to a buffer register called the TMRxH buffer register. While in 16-bit mode, the TMRxH register is not directly readable or writable and all read and write operations take place through the use of this TMRxH buffer register.

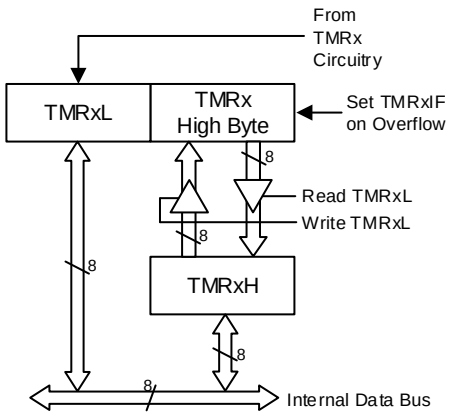
When a read from the TMRxL register is requested, the value of the TMRxH register is simultaneously loaded into the TMRxH buffer register. When a read from the TMRxH register is requested, the value is provided from the TMRxH buffer register instead. This provides the user with the ability to accurately read all 16 bits of the Timer1 value from a

© 2019-2023 Microchip Technology Inc. and its subsidiaries

single instance in time (refer to Figure 25-3 for more details). In contrast, when not in 16-bit mode, the user must read each register separately and determine if the values have become invalid due to a rollover that may have occurred between the read operations.

When a write request of the TMRxL register is requested, the TMRxH buffer register is simultaneously updated with the contents of the TMRxH register. The value of TMRxH must be preloaded into the TMRxH buffer register prior to the write request for the TMRxL register. This provides the user with the ability to write all 16 bits to the TMRx register at the same time. Any requests to write to TMRxH directly does not clear the Timer1 prescaler value. The prescaler value is only cleared through write requests to the TMRxL register.

**Figure 25-3. Timer1 16-Bit Read/Write Mode Block Diagram**




## 25.7 Timer1 Gate

Timer1 can be configured to count freely or the count can be enabled and disabled using Timer1 gate circuitry. This is also referred to as Timer1 gate enable. Timer1 gate can also be driven by multiple selectable sources.

### 25.7.1 Timer1 Gate Enable

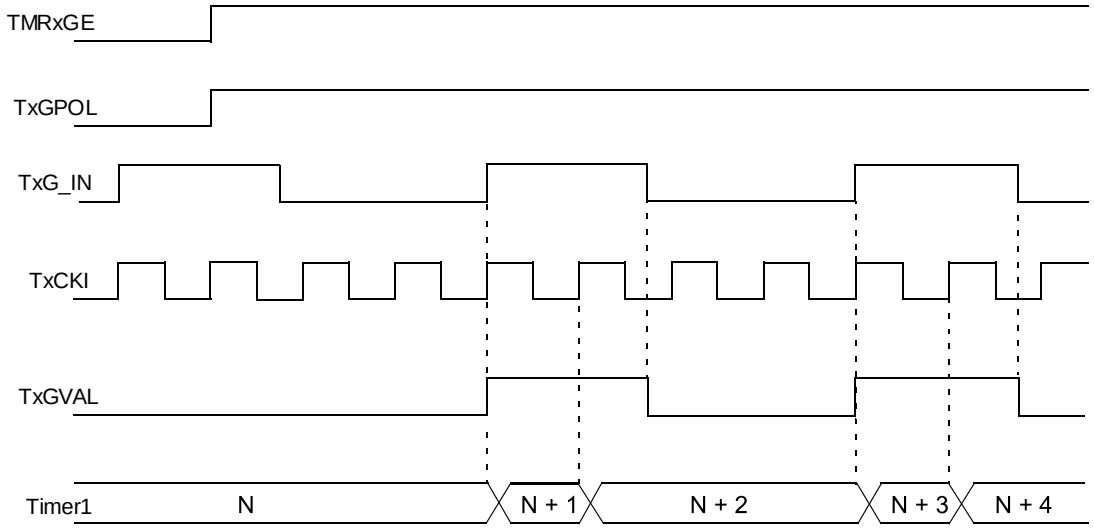
The Timer1 Gate Enable mode is enabled by setting the GE bit. The polarity of the Timer1 Gate Enable mode is configured using the GPOL bit.

When Timer1 Gate Enable mode is enabled, Timer1 will increment on the rising edge of the Timer1 clock source. When Timer1 Gate signal is inactive, the timer will not increment and hold the current count. Enable mode is disabled, no incrementing will occur and Timer1 will hold the current count. See Figure 25-4 for timing details.

**Table 25-2. Timer1 Gate Enable Selections**

| TMRxCLK | GPOL | TxG | Timer1 Operation |
|---|---|---|---|
| ↑ | 1 | 1 | Counts |
| ↑ | 1 | 0 | Holds Count |
| ↑ | 0 | 1 | Holds Count |
| ↑ | 0 | 0 | Counts |

**Figure 25-4. Timer1 Gate Enable Mode**




### 25.7.2 Timer1 Gate Source Selection

The gate source for Timer1 is selected using the GSS bits. The polarity selection for the gate source is controlled by the GPOL bit.

Any of the above mentioned signals can be used to trigger the gate. The output of the CMPx can be synchronized to the Timer1 clock or left asynchronous. For more information refer to the **"Comparator Output Synchronization"** section.

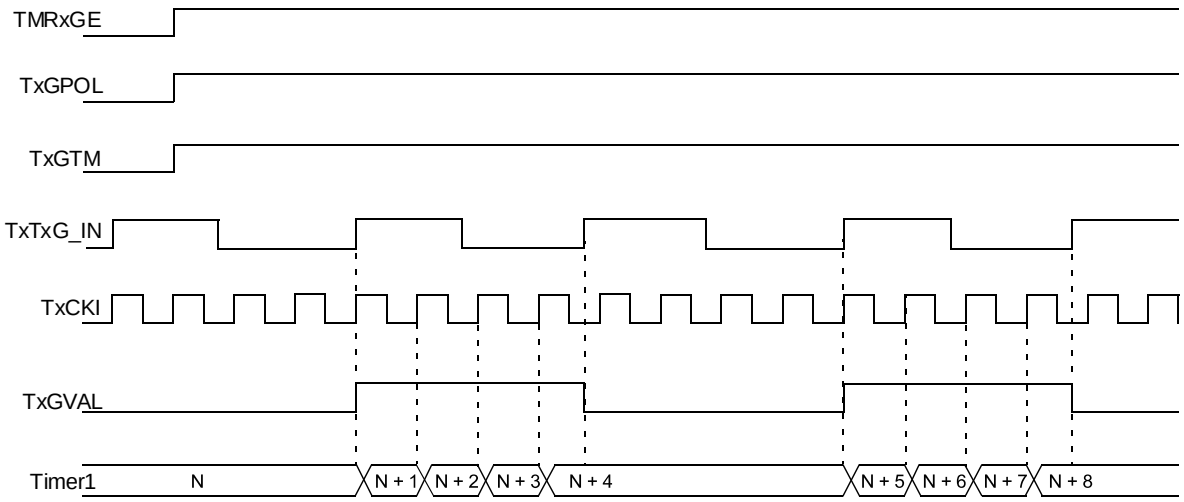
### 25.7.3 Timer1 Gate Toggle Mode

When Timer1 Gate Toggle mode is enabled, it is possible to measure the full-cycle length of a Timer1 Gate signal, as opposed to the duration of a single-level pulse. The Timer1 gate source is routed through a flip-flop that changes state on every incrementing edge of the signal. See the figure below for timing details.

Timer1 Gate Toggle mode is enabled by setting the GTM bit. When the GTM bit is cleared, the flip-flop is cleared and held clear. This is necessary to control which edge is measured.

**Important:** Enabling Toggle mode at the same time as changing the gate polarity may result in indeterminate operation.

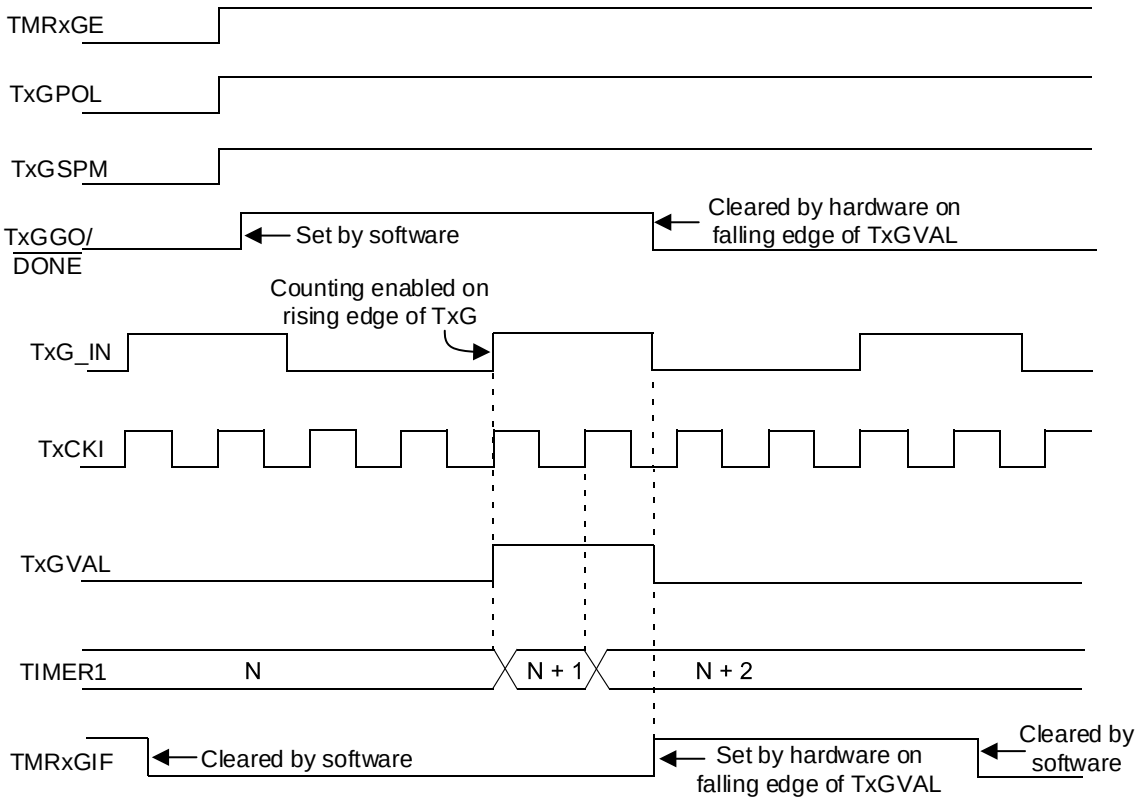
**Figure 25-5. Timer1 Gate Toggle Mode**




### 25.7.4 Timer1 Gate Single Pulse Mode

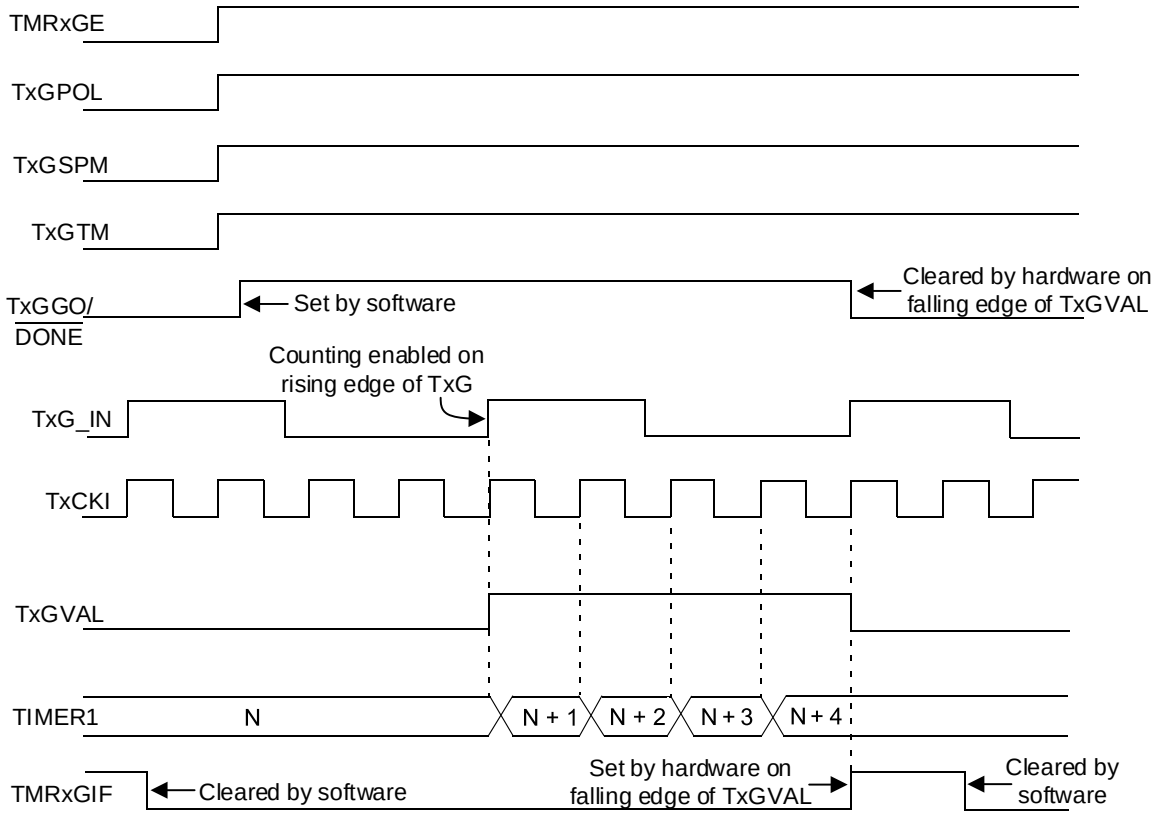
When Timer1 Gate Single Pulse mode is enabled, it is possible to capture a single pulse gate event. Timer1 Gate Single Pulse mode is first enabled by setting the GSPM bit. Next, the GGO/DONE must be set. The Timer1 will be fully enabled on the next incrementing edge. On the next trailing edge of the pulse, the GGO/$\overline{\text{DONE}}$ bit will automatically be cleared. No other gate events will be allowed to increment Timer1 until the GGO/$\overline{\text{DONE}}$ bit is once again set in software.

**Figure 25-6. Timer1 Gate Single Pulse Mode**



Clearing the GSPM bit will also clear the GGO/$\overline{\text{DONE}}$ bit. See the figure below for timing details. Enabling the Toggle mode and the Single Pulse mode simultaneously will permit both sections to work together. This allows the cycle times on the Timer1 gate source to be measured. See the figure below for timing details.

**Figure 25-7. Timer1 Gate Single Pulse and Toggle Combined Mode**




### 25.7.5 Timer1 Gate Value Status

When Timer1 gate value status is utilized, it is possible to read the most current level of the gate control value. The value is stored in the GVAL bit in the TxGCON register. The GVAL bit is valid even when the Timer1 gate is not enabled (GE bit is cleared).

### 25.7.6 Timer1 Gate Event Interrupt

When Timer1 gate event interrupt is enabled, it is possible to generate an interrupt upon the completion of a gate event. When the falling edge of GVAL occurs, the TMRxGIF flag bit in one of the PIR registers will be set. If the TMRxGIE bit in the corresponding PIE register is set, then an interrupt will be recognized.

The TMRxGIF flag bit operates even when the Timer1 gate is not enabled (the GE bit is cleared).

## 25.8 Timer1 Interrupt

The TMRx register increments to FFFFh and rolls over to 0000h. When TMRx rolls over, the Timer1 interrupt flag bit of the PIRx register is set. To enable the interrupt-on-rollover, the following bits must be set:

- The ON bit of the TxCON register
- The TMRxIE bits of the PIEx register
- Global interrupts must be enabled

The interrupt is cleared by clearing the TMRxIF bit as a task in the Interrupt Service Routine.

**Important:** The TMRx register and the TMRxIF bit must be cleared before enabling interrupts.

## 25.9 Timer1 Operation During Sleep

Timer1 can only operate during Sleep when configured as an asynchronous counter. In this mode, many clock sources can be used to increment the counter. To set up the timer to wake the device:

- The ON bit must be set
- The TMRxIE bit of the PIEx register must be set
- Global interrupts must be enabled
- The SYNC bit must be set
- Configure the TxCLK register for using any clock source other than $F_{OSC}$ and $F_{OSC}/4$

The device will wake up on an overflow and execute the next instruction. If global interrupts are enabled, the device will call the IRS. The secondary oscillator will continue to operate in Sleep regardless of the SYNC bit setting.

## 25.10 CCP Capture/Compare Time Base

The CCP modules use TMRx as the time base when operating in Capture or Compare mode. In Capture mode, the value in TMRx is copied into the CCPRx register on a capture event. In Compare mode, an event is triggered when the value in the CCPRx register matches the value in TMRx. This event can be a Special Event Trigger.

## 25.11 CCP Special Event Trigger

When any of the CCPs are configured to trigger a special event, the trigger will clear the TMRx register. This special event does not cause a Timer1 interrupt. The CCP module may still be configured to generate a CCP interrupt. In this mode of operation, the CCPRx register becomes the period register for Timer1. Timer1 must be synchronized and $F_{OSC}/4$ must be selected as the clock source to utilize the Special Event Trigger. Asynchronous operation of Timer1 can cause a Special Event Trigger to be missed. In the event that a write to TMRxH or TMRxL coincides with a Special Event Trigger from the CCP, the write will take precedence.

## 25.12 Peripheral Module Disable

When a peripheral is not used or inactive, the module can be disabled by setting the Module Disable bit in the PMD registers. This will reduce power consumption to an absolute minimum. Setting the PMD bits holds the module in Reset and disconnects the module's clock source. The Module Disable bits for Timer1 (TMR1MD) are in the PMDx register. See the **"PMD - Peripheral Module Disable"** chapter for more information.

## 25.13 Register Definitions: Timer1 Control

Long bit name prefixes for the Timer registers are shown in the table below, where 'x' refers to the Timer instance number. Refer to the "**Long Bit Names**" section in the "**Register and Bit Naming Conventions**" chapter for more information.

**Table 25-3. Timer1 Register Bit Name Prefixes**

| Peripheral | Bit Name Prefix |
|---|---|
| Timer1 | T1 |
| Timer3 | T3 |

.....continued

| Peripheral | Bit Name Prefix |
|---|---|
| Timer5 | T5 |

### 25.13.1 TxCON

**Name:** TxCON
**Address:** 0x31E,0x32A,0x336

Timer Control Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | CKPS[1:0] | | | SYNC | RD16 | ON |
| Access | | | R/W | R/W | | R/W | R/W | R/W |
| Reset | | | 0 | 0 | | 0 | 0 | 0 |

**Bits 5:4 – CKPS[1:0]** Timer Input Clock Prescaler Select
Reset States: POR/BOR = 00
All Other Resets = uu

| Value | Description |
|---|---|
| 11 | 1:8 Prescaler value |
| 10 | 1:4 Prescaler value |
| 01 | 1:2 Prescaler value |
| 00 | 1:1 Prescaler value |

**Bit 2 – $\overline{\text{SYNC}}$** Timer External Clock Input Synchronization Control
Reset States: POR/BOR = 0
All Other Resets = u

| Value | Condition | Description |
|---|---|---|
| x | CS = $F_{OSC}/4$ or $F_{OSC}$ | This bit is ignored. Timer uses the incoming clock as is. |
| 1 | All other clock sources | Do not synchronize external clock input |
| 0 | All other clock sources | Synchronize external clock input with system clock |

**Bit 1 – RD16** 16-Bit Read/Write Mode Enable
Reset States: POR/BOR = 0
All Other Resets = u

| Value | Description |
|---|---|
| 1 | Enables register read/write of Timer in one 16-bit operation |
| 0 | Enables register read/write of Timer in two 8-bit operations |

**Bit 0 – ON** Timer On
Reset States: POR/BOR = 0
All Other Resets = u

| Value | Description |
|---|---|
| 1 | Enables Timer |
| 0 | Disables Timer |

### 25.13.2 TxGCON

**Name:** TxGCON
**Address:** 0x31F,0x32B,0x337

Timer Gate Control Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | GE | GPOL | GTM | GSPM | GGO/$\overline{\text{DONE}}$ | GVAL | | |
| Access | R/W | R/W | R/W | R/W | R/W | R | | |
| Reset | 0 | 0 | 0 | 0 | 0 | x | | |

**Bit 7 – GE** Timer Gate Enable
Reset States: POR/BOR = 0
All Other Resets = u

| Value | Condition | Description |
|---|---|---|
| 1 | ON = 1 | Timer counting is controlled by the Timer gate function |
| 0 | ON = 1 | Timer is always counting |
| X | ON = 0 | This bit is ignored |

**Bit 6 – GPOL** Timer Gate Polarity
Reset States: POR/BOR = 0
All Other Resets = u

| Value | Description |
|---|---|
| 1 | Timer gate is active-high (Timer counts when gate is high) |
| 0 | Timer gate is active-low (Timer counts when gate is low) |

**Bit 5 – GTM** Timer Gate Toggle Mode
Timer Gate flip-flop toggles on every rising edge when Toggle mode is enabled.
Reset States: POR/BOR = 0
All Other Resets = u

| Value | Description |
|---|---|
| 1 | Timer Gate Toggle mode is enabled |
| 0 | Timer Gate Toggle mode is disabled and Toggle flip-flop is cleared |

**Bit 4 – GSPM** Timer Gate Single Pulse Mode
Reset States: POR/BOR = 0
All Other Resets = u

| Value | Description |
|---|---|
| 1 | Timer Gate Single Pulse mode is enabled and is controlling Timer gate |
| 0 | Timer Gate Single Pulse mode is disabled |

**Bit 3 – GGO/$\overline{\text{DONE}}$** Timer Gate Single Pulse Acquisition Status
This bit is automatically cleared when TxGSPM is cleared.
Reset States: POR/BOR = 0
All Other Resets = u

| Value | Description |
|---|---|
| 1 | Timer Gate Single Pulse Acquisition is ready, waiting for an edge |
| 0 | Timer Gate Single Pulse Acquisition has completed or has not been started |

**Bit 2 – GVAL** Timer Gate Current State
Indicates the current state of the timer gate that can be provided to TMRxH:TMRxL
Unaffected by the Timer Gate Enable (GE) bit

### 25.13.3 TxCLK

**Name:** TxCLK
**Address:** 0x321,0x32D,0x339

Timer Clock Source Selection Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | CS[4:0] | | | | |
| Access | | | | R/W | R/W | R/W | R/W | R/W |
| Reset | | | | 0 | 0 | 0 | 0 | 0 |

**Bits 4:0 – CS[4:0]** Timer Clock Source Selection

**Table 25-4. Timer Clock Sources**

| CS | Clock Source | | |
|---|---|---|---|
| | Timer1 | Timer3 | Timer5 |
| 11111-10110 | Reserved | | |
| 10101 | CLC8_OUT | | |
| 10100 | CLC7_OUT | | |
| 10011 | CLC6_OUT | | |
| 10010 | CLC5_OUT | | |
| 10001 | CLC4_OUT | | |
| 10000 | CLC3_OUT | | |
| 01111 | CLC2_OUT | | |
| 01110 | CLC1_OUT | | |
| 01101 | TMR5_OUT | TMR5_OUT | Reserved |
| 01100 | TMR3_OUT | Reserved | TMR3_OUT |
| 01011 | Reserved | TMR1_OUT | TMR1_OUT |
| 01010 | TMR0_OUT | | |
| 01001 | CLKREF_OUT | | |
| 01000 | EXTOSC | | |
| 00111 | SOSC | | |
| 00110 | MFINTOSC (31.25 kHz) | | |
| 00101 | MFINTOSC (500 kHz) | | |
| 00100 | LFINTOSC | | |
| 00011 | HFINTOSC | | |
| 00010 | $F_{OSC}$ | | |
| 00001 | $F_{OSC}/4$ | | |
| 00000 | Pin selected by T1CKIPPS | Pin selected by T3CKIPPS | Pin selected by T5CKIPPS |

Reset States: POR/BOR = 00000
All Other Resets = uuuuu

### 25.13.4 TxGATE

**Name:** TxGATE
**Address:** 0x320,0x32C,0x338

Timer Gate Source Selection Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | GSS[5:0] | | | | | |
| Access | | | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | | | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 5:0 – GSS[5:0]** Timer Gate Source Selection

**Table 25-5. Timer Gate Sources**

| GSS | Gate Source | | |
|---|---|---|---|
| | Timer1 | Timer3 | Timer5 |
| 111111-100010 | Reserved | | |
| 100001 | CLC8_OUT | | |
| 100000 | CLC7_OUT | | |
| 011111 | CLC6_OUT | | |
| 011110 | CLC5_OUT | | |
| 011101 | CLC4_OUT | | |
| 011100 | CLC3_OUT | | |
| 011011 | CLC2_OUT | | |
| 011010 | CLC1_OUT | | |
| 011001 | ZCD_OUT | | |
| 011000 | CMP2_OUT | | |
| 010111 | CMP1_OUT | | |
| 010110 | NCO3_OUT | | |
| 010101 | NCO2_OUT | | |
| 010100 | NCO1_OUT | | |
| 010010-010011 | Reserved | | |
| 010001 | PWM3S1P2_OUT | | |
| 010000 | PWM3S1P1_OUT | | |
| 001111 | PWM2S1P2_OUT | | |
| 001110 | PWM2S1P1_OUT | | |
| 001101 | PWM1S1P2_OUT | | |
| 001100 | PWM1S1P1_OUT | | |
| 001011 | CCP3_OUT | | |
| 001010 | CCP2_OUT | | |
| 001001 | CCP1_OUT | | |
| 001000 | SMT1_OUT | | |
| 000111 | TMR6_Postscaler_OUT | | |
| 000110 | TMR5_OUT | TMR5_OUT | Reserved |
| 000101 | TMR4_Postscaler_OUT | | |
| 000100 | TMR3_OUT | Reserved | TMR3_OUT |
| 000011 | TMR2_Postscaler_OUT | | |
| 000010 | Reserved | TMR1_OUT | TMR1_OUT |
| 000001 | TMR0_OUT | | |

© 2019-2023 Microchip Technology Inc. and its subsidiaries

**.....continued**

| GSS | Gate Source | | |
|---|---|---|---|
| | Timer1 | Timer3 | Timer5 |
| 000000 | Pin selected by T1GPPS | Pin selected by T3GPPS | Pin selected by T5GPPS |

### 25.13.5 TMRx

**Name:** TMRx
**Address:** 0x31C,0x328,0x334

Timer Register

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | TMRx[15:8] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | TMRx[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 15:0 – TMRx[15:0]** Timer Register Value
Reset States: POR/BOR = 0000000000000000
All Other Resets = uuuuuuuuuuuuuuuu

**Notes:** The individual bytes in this multibyte register can be accessed with the following register names:

- TMRxH: Accesses the high byte TMRx[15:8]
- TMRxL: Accesses the low byte TMRx[7:0]

## 25.14 Register Summary - Timer1

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x031C | TMR1 | 7:0 | TMR1[7:0] | | | | | | | |
| | | 15:8 | TMR1[15:8] | | | | | | | |
| 0x031E | T1CON | 7:0 | | | CKPS[1:0] | | | SYNC | RD16 | ON |
| 0x031F | T1GCON | 7:0 | GE | GPOL | GTM | GSPM | GGO/DONE | GVAL | | |
| 0x0320 | T1GATE | 7:0 | | | GSS[5:0] | | | | | |
| 0x0321 | T1CLK | 7:0 | | | | CS[4:0] | | | | |
| 0x0322 ... 0x0327 | Reserved | | | | | | | | | |
| 0x0328 | TMR3 | 7:0 | TMR3[7:0] | | | | | | | |
| | | 15:8 | TMR3[15:8] | | | | | | | |
| 0x032A | T3CON | 7:0 | | | CKPS[1:0] | | | SYNC | RD16 | ON |
| 0x032B | T3GCON | 7:0 | GE | GPOL | GTM | GSPM | GGO/DONE | GVAL | | |
| 0x032C | T3GATE | 7:0 | | | GSS[5:0] | | | | | |
| 0x032D | T3CLK | 7:0 | | | | CS[4:0] | | | | |
| 0x032E ... 0x0333 | Reserved | | | | | | | | | |
| 0x0334 | TMR5 | 7:0 | TMR5[7:0] | | | | | | | |
| | | 15:8 | TMR5[15:8] | | | | | | | |
| 0x0336 | T5CON | 7:0 | | | CKPS[1:0] | | | SYNC | RD16 | ON |
| 0x0337 | T5GCON | 7:0 | GE | GPOL | GTM | GSPM | GGO/DONE | GVAL | | |
| 0x0338 | T5GATE | 7:0 | | | GSS[5:0] | | | | | |
| 0x0339 | T5CLK | 7:0 | | | | CS[4:0] | | | | |

# 26. TMR2 - Timer2 Module

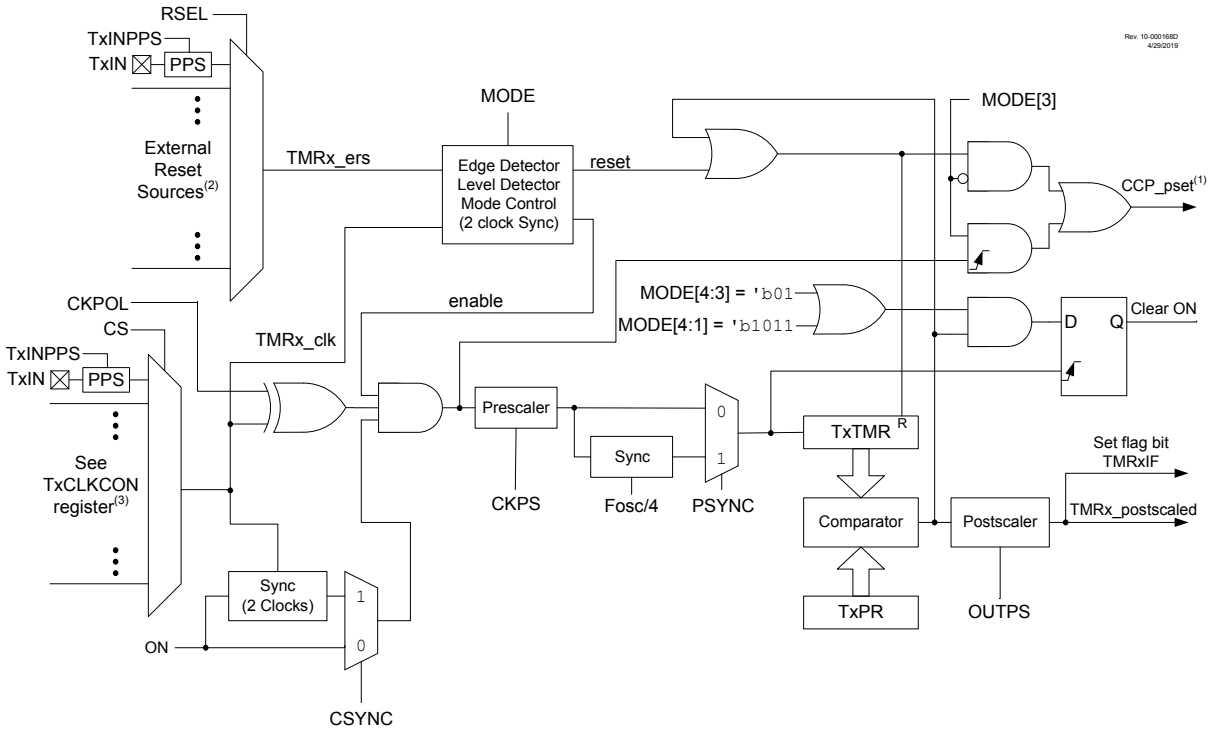
The Timer2 module is an 8-bit timer that incorporates the following features:

- 8-bit timer and period registers
- Readable and writable
- Software programmable prescaler (1:1 to 1:128)
- Software programmable postscaler (1:1 to 1:16)
- Interrupt on T2TMR match with T2PR
- One-shot operation
- Full asynchronous operation
- Includes Hardware Limit Timer (HLT)
- Alternate clock sources
- External timer Reset signal sources
- Configurable timer Reset operation

See figure below for a block diagram of Timer2.

**Important:** References to module Timer2 apply to all the even numbered timers on this device (Timer2, Timer4, etc.).

**Figure 26-1. Timer2 with Hardware Limit Timer (HLT) Block Diagram**




**Notes:**

1. Signal to the CCP peripheral for PWM pulse trigger in PWM mode.
2. See RSEL for external Reset sources.
3. See CS for clock source selections.

## 26.1 Timer2 Operation

Timer2 operates in three major modes:

- Free-Running Period
- One Shot
- Monostable

Within each operating mode, there are several options for starting, stopping and Reset. Table 26-1 lists the options.

In all modes, the T2TMR count register increments on the rising edge of the clock signal from the programmable prescaler. When T2TMR equals T2PR, a high level output to the postscaler counter is generated. T2TMR is cleared on the next clock input.

An external signal from hardware can also be configured to gate the timer operation or force a T2TMR count Reset. In Gate modes, the counter stops when the gate is disabled and resumes when the gate is enabled. In Reset modes, the T2TMR count is reset on either the level or edge from the external source.

The T2TMR and T2PR registers are both directly readable and writable. The T2TMR register is cleared and the T2PR register initializes to 0xFF on any device Reset. Both the prescaler and postscaler counters are cleared on the following events:

- A write to the T2TMR register
- A write to the T2CON register
- Any device Reset
- External Reset source event that resets the timer

**Important:** T2TMR is not cleared when T2CON is written.

### 26.1.1 Free-Running Period Mode

The value of T2TMR is compared to that of the period register, T2PR, on each clock cycle. When the two values match, the comparator resets the value of T2TMR to 0x00 on the next cycle and increments the output postscaler counter. When the postscaler count equals the value in the OUTPS bits of the T2CON register then a one clock period wide pulse occurs on the TMR2_postscaled output, and the postscaler count is cleared.

### 26.1.2 One Shot Mode

The One Shot mode is identical to the Free-Running Period mode except that the ON bit is cleared and the timer is stopped when T2TMR matches T2PR and will not restart until the ON bit is cycled off and on. Postscaler (OUTPS) values other than zero are ignored in this mode because the timer is stopped at the first period event and the postscaler is reset when the timer is restarted.

### 26.1.3 Monostable Mode

Monostable modes are similar to One Shot modes except that the ON bit is not cleared and the timer can be restarted by an external Reset event.

## 26.2 Timer2 Output

The Timer2 module's primary output is TMR2_postscaled, which pulses for a single TMR2_clk period upon each match of the postscaler counter and the OUTPS bits of the T2CON register. The postscaler is incremented each time the T2TMR value matches the T2PR value. This signal can also be selected as an input to other Core Independent Peripherals.

In addition, the Timer2 is also used by the CCP module for pulse generation in PWM mode. See the "**PWM Overview**" and "**PWM Period**" sections in the **"CCP - Capture/Compare/PWM Module"** chapter for more details on setting up Timer2 for use with the CCP and PWM modules.

© 2019-2023 Microchip Technology Inc. and its subsidiaries

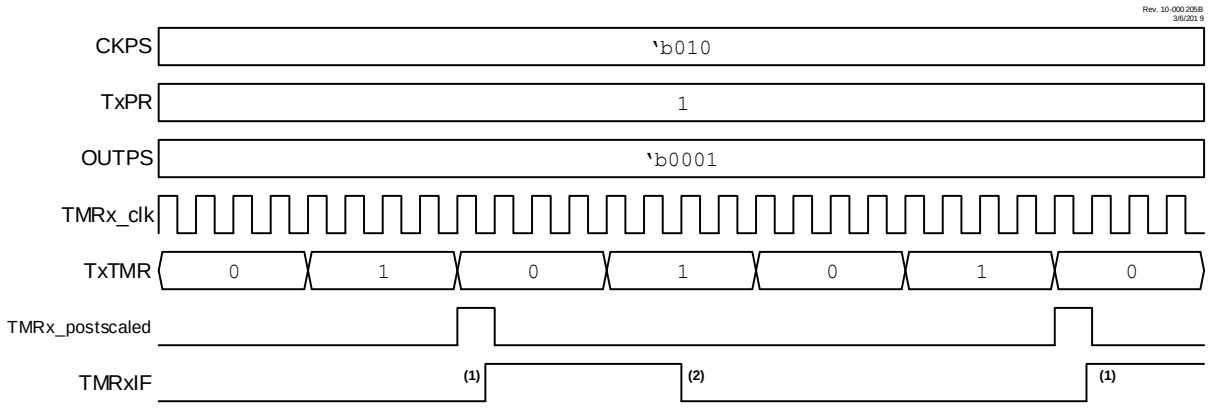
## 26.3 External Reset Sources

In addition to the clock source, the Timer2 can also be driven by an external Reset source input. This external Reset input is selected for each timer with the corresponding TxRST register. The external Reset input can control starting and stopping of the timer, as well as resetting the timer, depending on the mode used.

## 26.4 Timer2 Interrupt

Timer2 can also generate a device interrupt. The interrupt is generated when the postscaler counter matches the selected postscaler value (OUTPS bits of T2CON register). The interrupt is enabled by setting the TMR2IE interrupt enable bit. Interrupt timing is illustrated in the figure below.

**Figure 26-2. Timer2 Prescaler, Postscaler, and Interrupt Timing Diagram**




**Notes:** 1. Setting the interrupt flag is synchronized with the instruction clock. Synchronization may take as many as two instruction cycles.
2. Cleared by software.

## 26.5 PSYNC Bit

Setting the PSYNC bit synchronizes the prescaler output to $F_{OSC}/4$. Setting this bit is required for reading the Timer2 counter register while the selected Timer clock is asynchronous to $F_{OSC}/4$.

**Note:** Setting PSYNC requires that the output of the prescaler is slower than $F_{OSC}/4$. Setting PSYNC when the output of the prescaler is greater than or equal to $F_{OSC}/4$ may cause unexpected results.

## 26.6 CSYNC Bit

All bits in the Timer2 SFRs are synchronized to $F_{OSC}/4$ by default, not the Timer2 input clock. As such, if the Timer2 input clock is not synchronized to $F_{OSC}/4$, it is possible for the Timer2 input clock to transition at the same time as the ON bit is set in software, which may cause undesirable behavior and glitches in the counter. Setting the CSYNC bit remedies this problem by synchronizing the ON bit to the Timer2 input clock instead of $F_{OSC}/4$. However, as this synchronization uses an edge of the TMR2 input clock, up to one input clock cycle will be consumed and not counted by the Timer2 when CSYNC is set. Conversely, clearing the CSYNC bit synchronizes the ON bit to $F_{OSC}/4$, which does not consume any clock edges, but has the previously stated risk of glitches.

© 2019-2023 Microchip Technology Inc. and its subsidiaries

## 26.7 Operating Modes

The mode of the timer is controlled by the MODE bits. Edge Triggered modes require six Timer clock periods between external triggers. Level Triggered modes require the triggering level to be at least three Timer clock periods long. External triggers are ignored while in Debug mode.

**Table 26-1. Operating Modes Table**

| Mode | MODE | | Output Operation | Operation | Timer Control | | |
|---|---|---|---|---|---|---|---|
| | [4:3] | [2:0] | | | Start | Reset | Stop |
| Free-Running Period | 00 | 000 | Period Pulse | Software gate (Figure 26-3) | ON = 1 | — | ON = 0 |
| | | 001 | | Hardware gate, active-high (Figure 26-4) | ON = 1 and TMRx_ers = 1 | — | ON = 0 or TMRx_ers = 0 |
| | | 010 | | Hardware gate, active-low | ON = 1 and TMRx_ers = 0 | — | ON = 0 or TMRx_ers = 1 |
| | | 011 | Period Pulse with Hardware Reset | Rising or falling edge Reset | ON = 1 | TMRx_ers ↕ | ON = 0 |
| | | 100 | | Rising edge Reset (Figure 26-5) | | TMRx_ers ↑ | |
| | | 101 | | Falling edge Reset | | TMRx_ers ↓ | |
| | | 110 | | Low-level Reset | | TMRx_ers = 0 | ON = 0 or TMRx_ers = 0 |
| | | 111 | | High-level Reset (Figure 26-6) | | TMRx_ers = 1 | ON = 0 or TMRx_ers = 1 |
| One Shot | 01 | 000 | One-shot | Software start (Figure 26-7) | ON = 1 | — | ON = 0 or Next clock after TxTMR = TxPR (Note 2) |
| | | 001 | Edge-Triggered Start (Note 1) | Rising edge start (Figure 26-8) | ON = 1 and TMRx_ers ↑ | — | |
| | | 010 | | Falling edge start | ON = 1 and TMRx_ers ↓ | — | |
| | | 011 | | Any edge start | ON = 1 and TMRx_ers ↕ | — | |
| | | 100 | Edge-Triggered Start and Hardware Reset (Note 1) | Rising edge start and Rising edge Reset (Figure 26-9) | ON = 1 and TMRx_ers ↑ | TMRx_ers ↑ | |
| | | 101 | | Falling edge start and Falling edge Reset | ON = 1 and TMRx_ers ↓ | TMRx_ers ↓ | |
| | | 110 | | Rising edge start and Low-level Reset (Figure 26-10) | ON = 1 and TMRx_ers ↑ | TMRx_ers = 0 | |
| | | 111 | | Falling edge start and High-level Reset | ON = 1 and TMRx_ers ↓ | TMRx_ers = 1 | |

.....continued

| Mode | MODE | | Output Operation | Operation | Timer Control | | |
|---|---|---|---|---|---|---|---|
| | [4:3] | [2:0] | | | Start | Reset | Stop |
| Monostable | 10 | 000 | Reserved | | | | |
| | | 001 | Edge-Triggered Start (Note 1) | Rising edge start (Figure 26-11) | ON = 1 and TMRx_ers ↑ | — | ON = 0 or Next clock after TxTMR = TxPR (Note 3) |
| | | 010 | | Falling edge start | ON = 1 and TMRx_ers ↓ | — | |
| | | 011 | | Any edge start | ON = 1 and TMRx_ers ↕ | — | |
| Reserved | | 100 | Reserved | | | | |
| Reserved | | 101 | Reserved | | | | |
| One Shot | | 110 | Level-Triggered Start and Hardware Reset | High-level start and Low-level Reset (Figure 26-12) | ON = 1 and TMRx_ers = 1 | TMRx_ers = 0 | ON = 0 or Held in Reset (Note 2) |
| | | 111 | | Low-level start and High-level Reset | ON = 1 and TMRx_ers = 0 | TMRx_ers = 1 | |
| Reserved | 11 | xxx | Reserved | | | | |

**Notes:**

1. If ON = 0, then an edge is required to restart the timer after ON = 1.
2. When T2TMR = T2PR, the next clock clears ON and stops T2TMR at 00h.
3. When T2TMR = T2PR, the next clock stops T2TMR at 00h but does not clear ON.

## 26.8 Operation Examples

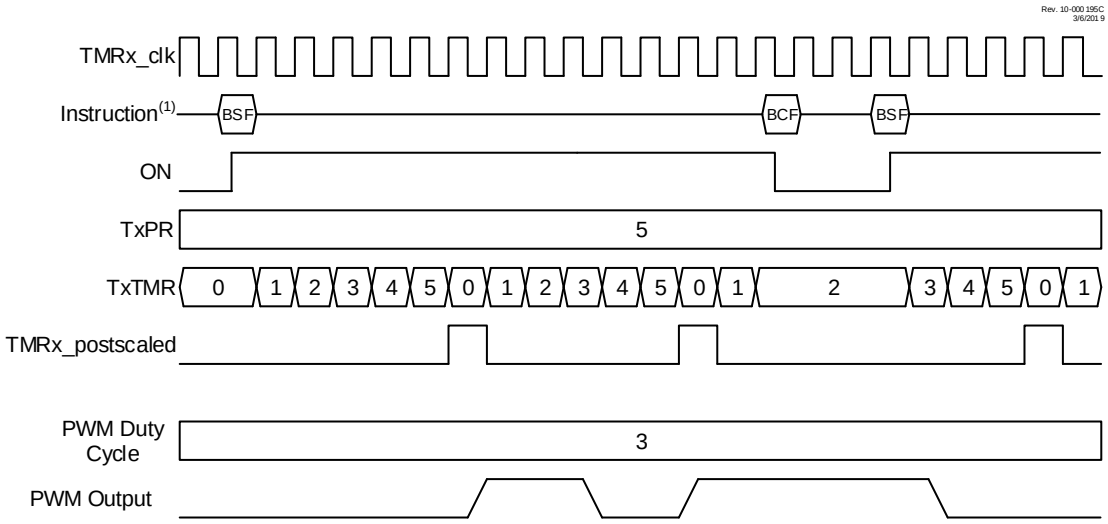
Unless otherwise specified, the following notes apply to the following timing diagrams:

- Both the prescaler and postscaler are set to 1:1 (both the CKPS and OUTPS bits).
- The diagrams illustrate any clock except $F_{OSC}/4$ and show clock-sync delays of at least two full cycles for both ON and TMRx_ers. When using $F_{OSC}/4$, the clock-sync delay is at least one instruction period for TMRx_ers; ON applies in the next instruction period.
- ON and TMRx_ers are somewhat generalized, and clock-sync delays may produce results that are slightly different than illustrated.
- The PWM Duty Cycle and PWM output are illustrated assuming that the timer is used for the PWM function of the CCP module as described in the "**PWM Overview**" section. The signals are not a part of the Timer2 module.

### 26.8.1 Software Gate Mode

This mode corresponds to legacy Timer2 operation. The timer increments with each clock input when ON = 1, and does not increment when ON = 0. When the TxTMR count equals the TxPR period count, the timer resets on the next clock and continues counting from zero. Operation with the ON bit software controlled is illustrated in Figure 26-3. With TxPR = 5, the counter advances until TxTMR = 5, and goes to zero with the next clock.

**Figure 26-3. Software Gate Mode Timing Diagram (MODE = 'b00000)**




**Note:** 1. BSF and BCF represent Bit-Set File and Bit-Clear File instructions executed by the CPU to set or clear the ON bit of TxCON. CPU execution is asynchronous to the timer clock input.

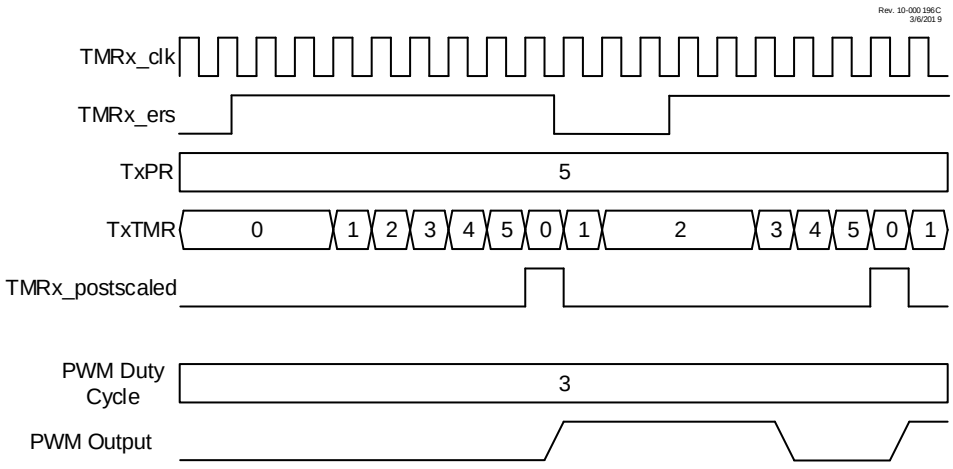
### 26.8.2 Hardware Gate Mode

The Hardware Gate modes operate the same as the Software Gate mode, except the TMRx_ers external signal can also gate the timer. When used with the CCP, the gating extends the PWM period. If the timer is stopped when the PWM output is high, then the duty cycle is also extended.

When MODE = 'b00001, then the timer is stopped when the external signal is high. When MODE = 'b00010, then the timer is stopped when the external signal is low.

Figure 26-4 illustrates the Hardware Gating mode for MODE = 'b00001 in which a high input level starts the counter.

**Figure 26-4. Hardware Gate Mode Timing Diagram (MODE = 'b00001)**



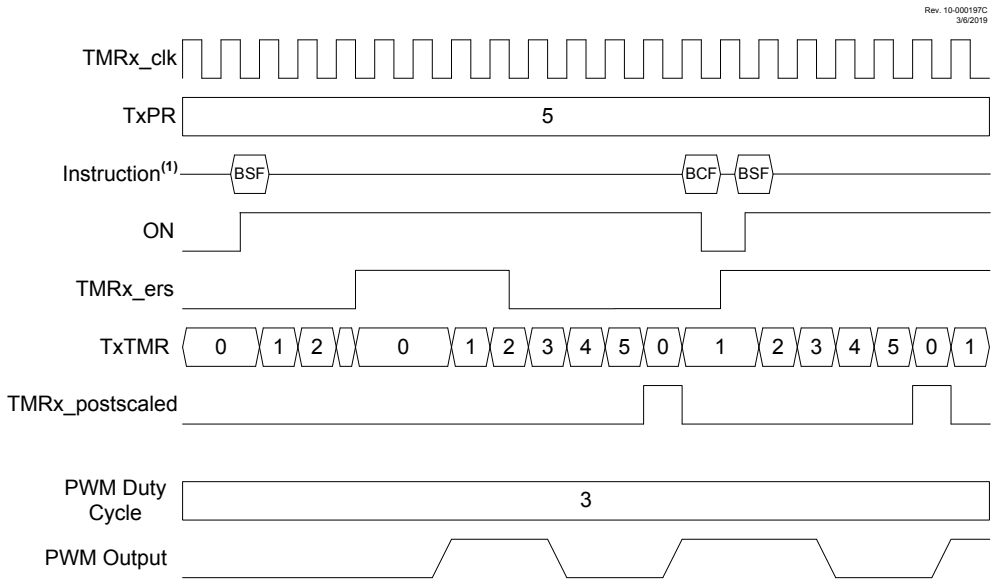

### 26.8.3 Edge Triggered Hardware Limit Mode

In Hardware Limit mode, the timer can be reset by the TMRx_ers external signal before the timer reaches the period count. Three types of Resets are possible:

- Reset on rising or falling edge (MODE = `'b00011`)
- Reset on rising edge (MODE = `'b00100`)
- Reset on falling edge (MODE = `'b00101`)

When the timer is used in conjunction with the CCP in PWM mode then an early Reset shortens the period and restarts the PWM pulse after a two clock delay. Refer to Figure 26-5.

**Figure 26-5. Edge Triggered Hardware Limit Mode Timing Diagram (MODE = `'b00100`)**




**Note:** 1. `BSF` and `BCF` represent Bit-Set File and Bit-Clear File instructions executed by the CPU to set or clear the ON bit of TxCON. CPU execution is asynchronous to the timer clock input.

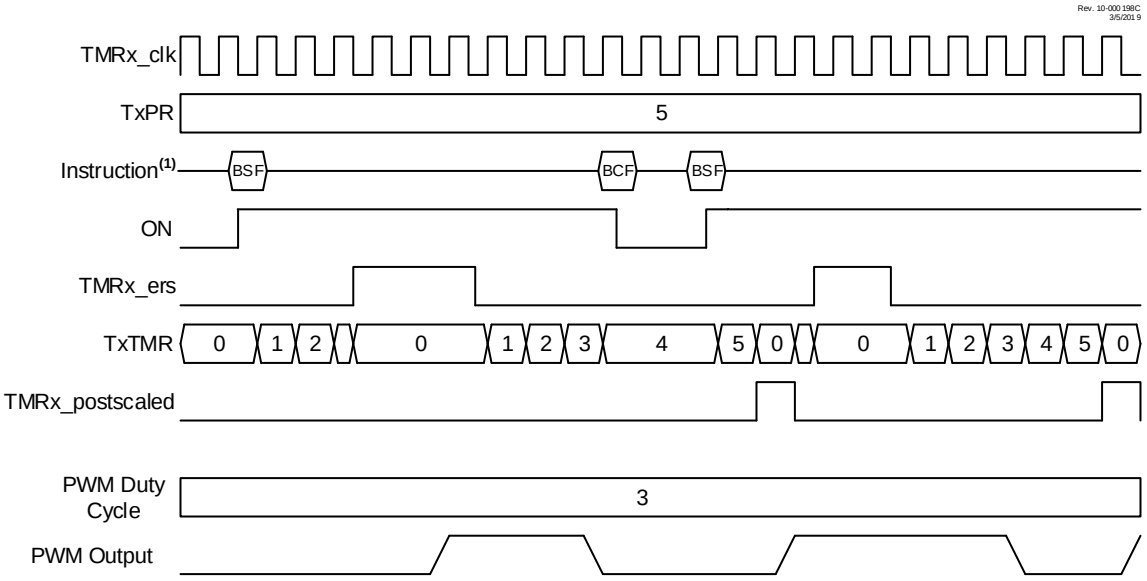
### 26.8.4 Level Triggered Hardware Limit Mode

In the Level Triggered Hardware Limit Timer modes the counter is reset by high or low levels of the external signal TMRx_ers, as shown in Figure 26-6. Selecting MODE = `'b00110` will cause the timer to reset on a low-level external signal. Selecting MODE = `'b00111` will cause the timer to reset on a high-level external signal. In the example, the counter is reset while TMRx_ers = `1`. ON is controlled by `BSF` and `BCF` instructions. When ON = `0`, the external signal is ignored.

When the CCP uses the timer as the PWM time base, then the PWM output will be set high when the timer starts counting and then set low only when the timer count matches the CCPRx value. The timer is reset when either the timer count matches the TxPR value or two clock periods after the external Reset signal goes true and stays true.

The timer starts counting, and the PWM output is set high, on either the clock following the TxPR match or two clocks after the external Reset signal relinquishes the Reset. The PWM output will remain high until the timer counts up to match the CCPRx pulse-width value. If the external Reset signal goes true while the PWM output is high, then the PWM output will remain high until the Reset signal is released allowing the timer to count up to match the CCPRx value.

**Figure 26-6. Level Triggered Hardware Limit Mode Timing Diagram (MODE = `'b00111`)**



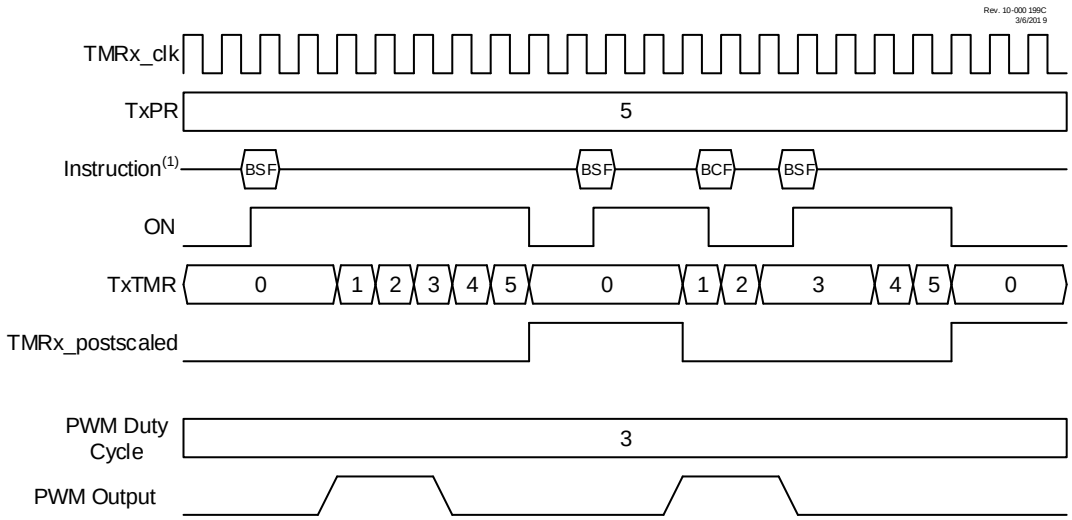

**Note:** 1. `BSF` and `BCF` represent Bit-Set File and Bit-Clear File instructions executed by the CPU to set or clear the ON bit of TxCON. CPU execution is asynchronous to the timer clock input.

### 26.8.5 Software Start One Shot Mode

In One Shot mode, the timer resets and the ON bit is cleared when the timer value matches the TxPR period value. The ON bit must be set by software to start another timer cycle. Setting MODE = `'b01000` selects One Shot mode which is illustrated in Figure 26-7. In the example, ON is controlled by `BSF` and `BCF` instructions. In the first case, a `BSF` instruction sets ON and the counter runs to completion and clears ON. In the second case, a `BSF` instruction starts the cycle, the `BCF`/`BSF` instructions turn the counter off and on during the cycle, and then it runs to completion.

When One Shot mode is used in conjunction with the CCP PWM operation, the PWM pulse drive starts concurrent with setting the ON bit. Clearing the ON bit while the PWM drive is active will extend the PWM drive. The PWM drive will terminate when the timer value matches the CCPRx pulse-width value. The PWM drive will remain off until the software sets the ON bit to start another cycle. If the software clears the ON bit after the CCPRx match but before the TxPR match, then the PWM drive will be extended by the length of time the ON bit remains cleared. Another timing cycle can only be initiated by setting the ON bit after it has been cleared by a TxPR period count match.

**Figure 26-7. Software Start One Shot Mode Timing Diagram (MODE = `'b01000`)**




**Note:** 1. `BSF` and `BCF` represent Bit-Set File and Bit-Clear File instructions executed by the CPU to set or clear the ON bit of TxCON. CPU execution is asynchronous to the timer clock input.

### 26.8.6 Edge Triggered One Shot Mode

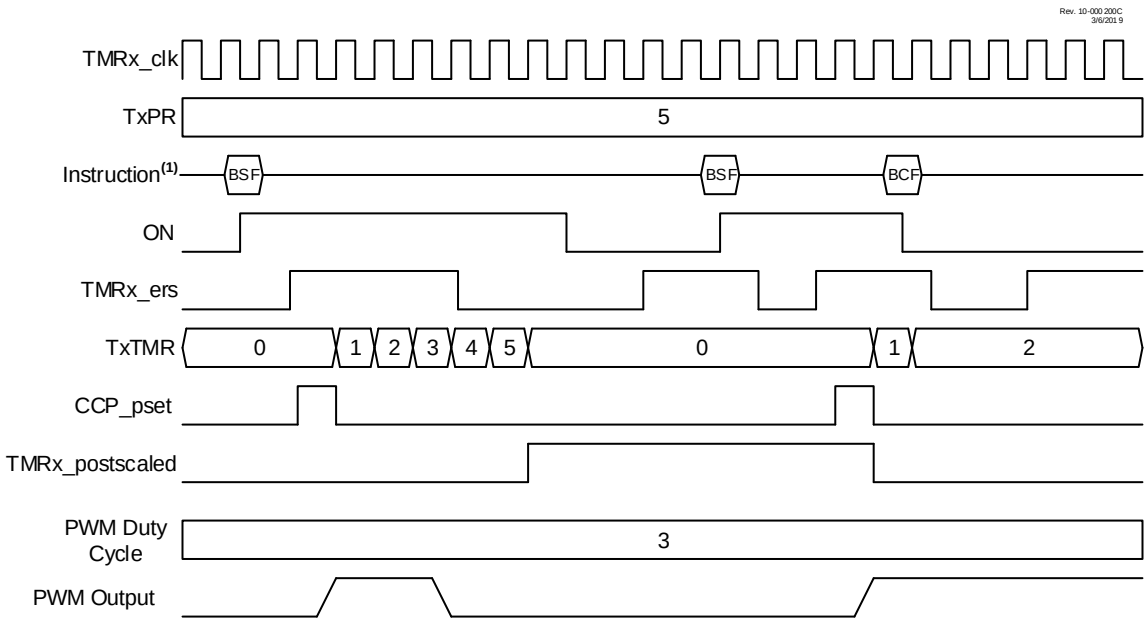
The Edge Triggered One Shot modes start the timer on an edge from the external signal input, after the ON bit is set, and clear the ON bit when the timer matches the TxPR period value. The following edges will start the timer:

- Rising edge (MODE = `'b01001`)
- Falling edge (MODE = `'b01010`)
- Rising or Falling edge (MODE = `'b01011`)

If the timer is halted by clearing the ON bit, then another TMRx_ers edge is required after the ON bit is set to resume counting. Figure 26-8 illustrates operation in the rising edge One Shot mode.

When Edge Triggered One Shot mode is used in conjunction with the CCP, then the edge-trigger will activate the PWM drive and the PWM drive will deactivate when the timer matches the CCPRx pulse-width value and stay deactivated when the timer halts at the TxPR period count match.

**Figure 26-8. Edge Triggered One Shot Mode Timing Diagram (MODE = `'b01001`)**




**Note:** 1. `BSF` and `BCF` represent Bit-Set File and Bit-Clear File instructions executed by the CPU to set or clear the ON bit of TxCON. CPU execution is asynchronous to the timer clock input.

### 26.8.7 Edge Triggered Hardware Limit One Shot Mode

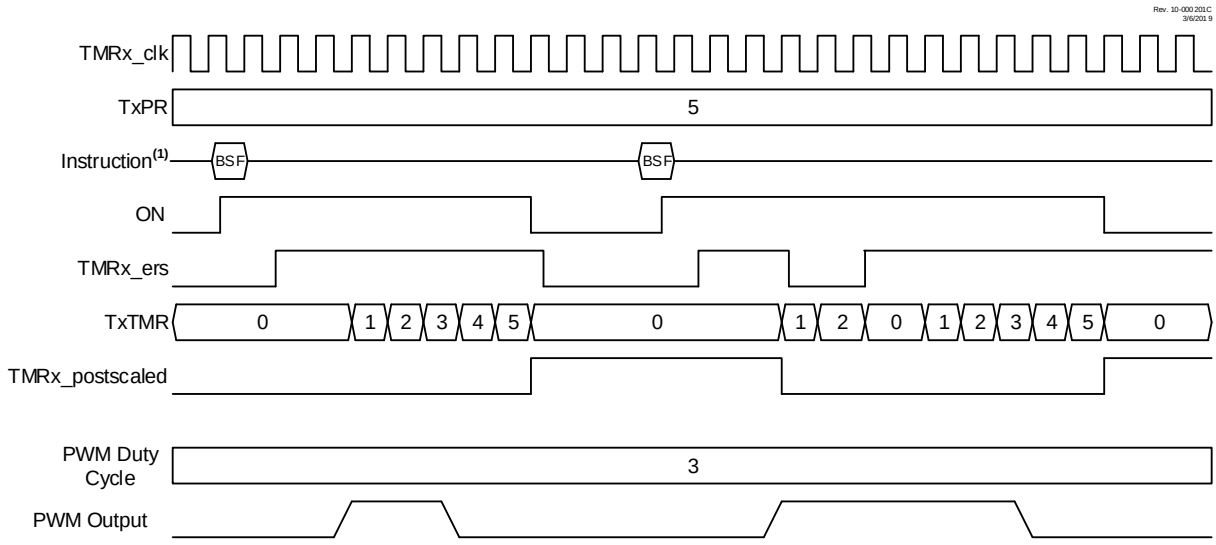
In Edge Triggered Hardware Limit One Shot modes, the timer starts on the first external signal edge after the ON bit is set and resets on all subsequent edges. Only the first edge after the ON bit is set is needed to start the timer. The counter will resume counting automatically two clocks after all subsequent external Reset edges. Edge triggers are as follows:

- Rising edge start and Reset (MODE = `'b01100`)
- Falling edge start and Reset (MODE = `'b01101`)

The timer resets and clears the ON bit when the timer value matches the TxPR period value. External signal edges will have no effect until after software sets the ON bit. Figure 26-9 illustrates the rising edge hardware limit one-shot operation.

When this mode is used in conjunction with the CCP, then the first starting edge trigger, and all subsequent Reset edges, will activate the PWM drive. The PWM drive will deactivate when the timer matches the CCPRx pulse-width value and stay deactivated until the timer halts at the TxPR period match unless an external signal edge resets the timer before the match occurs.

**Figure 26-9. Edge Triggered Hardware Limit One Shot Mode Timing Diagram (MODE = `'b01100`)**




**Note:** 1. `BSF` and `BCF` represent Bit-Set File and Bit-Clear File instructions executed by the CPU to set or clear the ON bit of TxCON. CPU execution is asynchronous to the timer clock input.

### 26.8.8 Level Reset, Edge Triggered Hardware Limit One Shot Modes

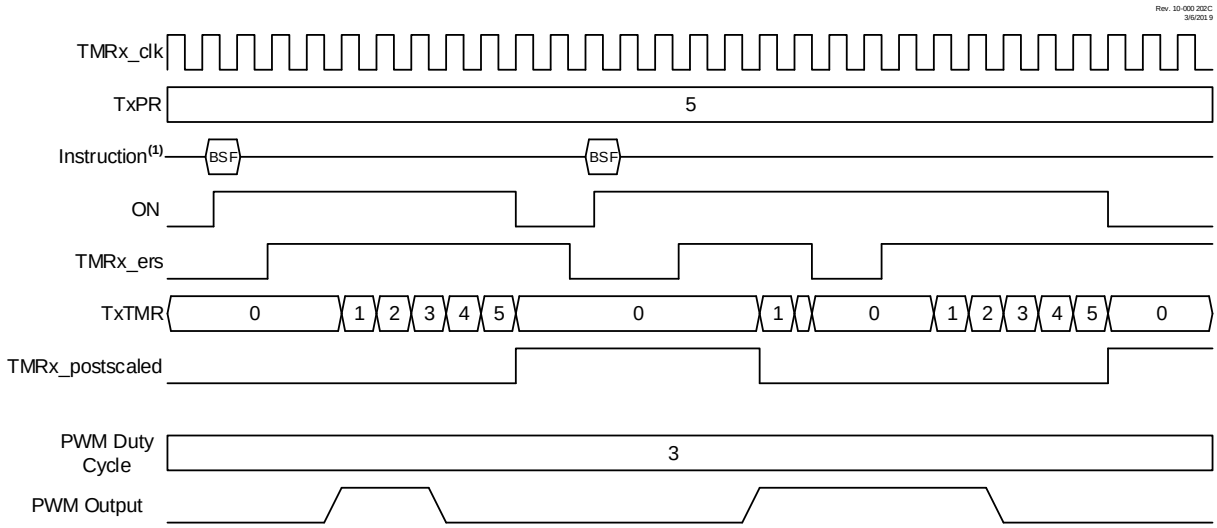
In Level Triggered One Shot mode, the timer count is reset on the external signal level and starts counting on the rising/falling edge of the transition from Reset level to the active level while the ON bit is set. Reset levels are selected as follows:

- Low Reset level (MODE = 'b01110)
- High Reset level (MODE = 'b01111)

When the timer count matches the TxPR period count, the timer is reset and the ON bit is cleared. When the ON bit is cleared by either a TxPR match or by software control, a new external signal edge is required after the ON bit is set to start the counter.

When Level-Triggered Reset One Shot mode is used in conjunction with the CCP PWM operation, the PWM drive goes active with the external signal edge that starts the timer. The PWM drive goes inactive when the timer count equals the CCPRx pulse-width count. The PWM drive does not go active when the timer count clears at the TxPR period count match.

**Figure 26-10. Low Level Reset, Edge Triggered Hardware Limit One Shot Mode Timing Diagram (MODE = 'b01110)**




**Note:** 1. BSF and BCF represent Bit-Set File and Bit-Clear File instructions executed by the CPU to set or clear the ON bit of TxCON. CPU execution is asynchronous to the timer clock input.

### 26.8.9 Edge Triggered Monostable Modes

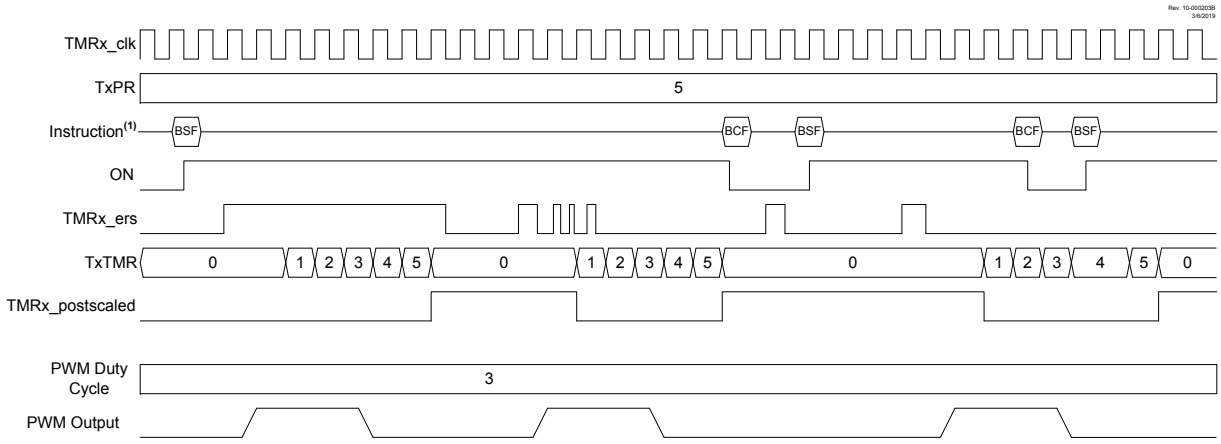
The Edge Triggered Monostable modes start the timer on an edge from the external Reset signal input, after the ON bit is set, and stop incrementing the timer when the timer matches the TxPR period value. The following edges will start the timer:

- Rising edge (MODE = 'b10001)
- Falling edge (MODE = 'b10010)
- Rising or Falling edge (MODE = 'b10011)

When an Edge Triggered Monostable mode is used in conjunction with the CCP PWM operation, the PWM drive goes active with the external Reset signal edge that starts the timer, but will not go active when the timer matches the TxPR value. While the timer is incrementing, additional edges on the external Reset signal will not affect the CCP PWM.

© 2019-2023 Microchip Technology Inc. and its subsidiaries

**Figure 26-11. Rising Edge Triggered Monostable Mode Timing Diagram (MODE = 'b10001)**




**Note:** 1. BSF and BCF represent Bit-Set File and Bit-Clear File instructions executed by the CPU to set or clear the ON bit of TxCON. CPU execution is asynchronous to the timer clock input.

### 26.8.10 Level Triggered Hardware Limit One Shot Modes

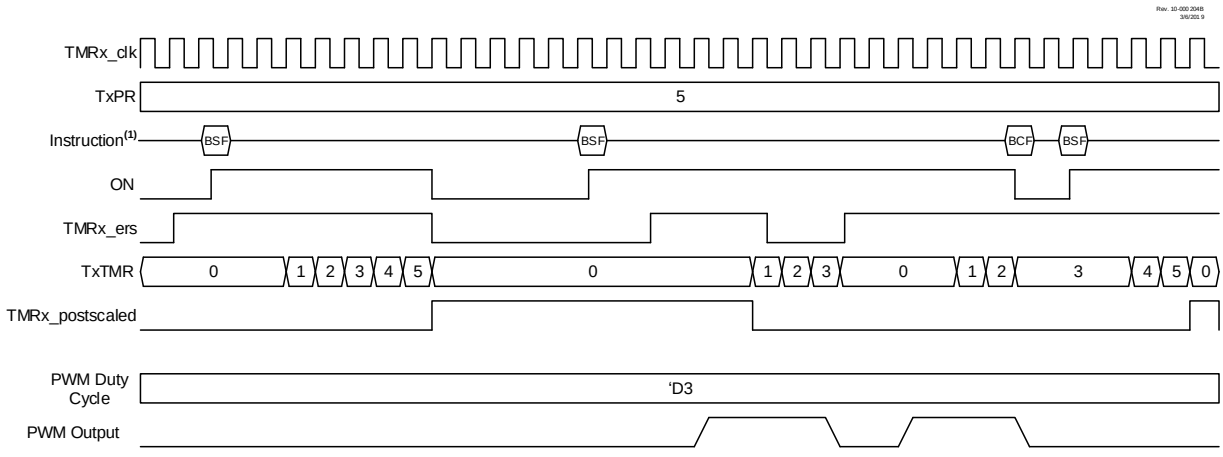
The Level Triggered Hardware Limit One Shot modes hold the timer in Reset on an external Reset level and start counting when both the ON bit is set and the external signal is not at the Reset level. If one of either the external signal is not in Reset or the ON bit is set, then the other signal being set/made active will start the timer. Reset levels are selected as follows:

- Low Reset level (MODE = 'b10110)
- High Reset level (MODE = 'b10111)

When the timer count matches the TxPR period count, the timer is reset and the ON bit is cleared. When the ON bit is cleared by either a TxPR match or by software control, the timer will stay in Reset until both the ON bit is set and the external signal is not at the Reset level.

When Level Triggered Hardware Limit One Shot modes are used in conjunction with the CCP PWM operation, the PWM drive goes active with either the external signal edge or the setting of the ON bit, whichever of the two starts the timer.

**Figure 26-12. Level Triggered Hardware Limit One Shot Mode Timing Diagram (MODE = 'b10110)**




**Note:** 1. BSF and BCF represent Bit-Set File and Bit-Clear File instructions executed by the CPU to set or clear the ON bit of TxCON. CPU execution is asynchronous to the timer clock input.

## 26.9 Timer2 Operation During Sleep

When PSYNC = 1, Timer2 cannot be operated while the processor is in Sleep mode. The contents of the T2TMR and T2PR registers will remain unchanged while the processor is in Sleep mode.

When PSYNC = 0, Timer2 will operate in Sleep as long as the clock source selected is also still running. If any internal oscillator is selected as the clock source, it will stay active during Sleep mode.

## 26.10 Register Definitions: Timer2 Control

Long bit name prefixes for the Timer2 peripherals are shown in the table below. Refer to the **"Long Bit Names"** section of the **"Register and Bit Naming Conventions"** chapter for more information.

**Table 26-2. Timer2 Long Bit Name Prefixes**

| Peripheral | Bit Name Prefix |
|---|---|
| Timer2 | T2 |
| Timer4 | T4 |
| Timer6 | T6 |

**Important:** References to module Timer2 apply to all the even numbered timers on this device (Timer2, Timer4, etc.).

### 26.10.1 TxTMR

**Name:** TxTMR
**Address:** 0x322,0x32E,0x33A

Timer Counter Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | TxTMR[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 7:0 – TxTMR[7:0]** Timerx Counter

### 26.10.2 TxPR

**Name:** TxPR
**Address:** 0x323,0x32F,0x33B

Timer Period Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | TxPR[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Bits 7:0 – TxPR[7:0]** Timer Period Register

| Value | Description |
|---|---|
| 0 to 255 | The timer restarts at '0' when TxTMR reaches the TxPR value |

### 26.10.3 TxCON

**Name:** TxCON
**Address:** 0x324,0x330,0x33C

Timerx Control Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | ON | CKPS[2:0] | | | OUTPS[3:0] | | | |
| Access | R/W/HC | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bit 7 – ON** Timer On[(1)]

| Value | Description |
|---|---|
| 1 | Timer is on |
| 0 | Timer is off: All counters and state machines are reset |

**Bits 6:4 – CKPS[2:0]** Timer Clock Prescale Select

| Value | Description |
|---|---|
| 111 | 1:128 Prescaler |
| 110 | 1:64 Prescaler |
| 101 | 1:32 Prescaler |
| 100 | 1:16 Prescaler |
| 011 | 1:8 Prescaler |
| 010 | 1:4 Prescaler |
| 001 | 1:2 Prescaler |
| 000 | 1:1 Prescaler |

**Bits 3:0 – OUTPS[3:0]** Timer Output Postscaler Select

| Value | Description |
|---|---|
| 1111 | 1:16 Postscaler |
| 1110 | 1:15 Postscaler |
| 1101 | 1:14 Postscaler |
| 1100 | 1:13 Postscaler |
| 1011 | 1:12 Postscaler |
| 1010 | 1:11 Postscaler |
| 1001 | 1:10 Postscaler |
| 1000 | 1:9 Postscaler |
| 0111 | 1:8 Postscaler |
| 0110 | 1:7 Postscaler |
| 0101 | 1:6 Postscaler |
| 0100 | 1:5 Postscaler |
| 0011 | 1:4 Postscaler |
| 0010 | 1:3 Postscaler |
| 0001 | 1:2 Postscaler |
| 0000 | 1:1 Postscaler |

**Note:**

1. In certain modes, the ON bit will be auto-cleared by hardware. See Table 26-1.

### 26.10.4 TxHLT

**Name:** TxHLT
**Address:** 0x325,0x331,0x33D

Timer Hardware Limit Control Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | PSYNC | CPOL | CSYNC | MODE[4:0] | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bit 7 – PSYNC** Timer Prescaler Synchronization Enable[1, 2]

| Value | Description |
|---|---|
| 1 | Timer Prescaler Output is synchronized to $F_{OSC}/4$ |
| 0 | Timer Prescaler Output is not synchronized to $F_{OSC}/4$ |

**Bit 6 – CPOL** Timer Clock Polarity Selection[3]

| Value | Description |
|---|---|
| 1 | Falling edge of input clock clocks timer/prescaler |
| 0 | Rising edge of input clock clocks timer/prescaler |

**Bit 5 – CSYNC** Timer Clock Synchronization Enable[4, 5]

| Value | Description |
|---|---|
| 1 | ON bit is synchronized to timer clock input |
| 0 | ON bit is not synchronized to timer clock input |

**Bits 4:0 – MODE[4:0]** Timer Control Mode Selection[6, 7]

| Value | Description |
|---|---|
| 00000 to 11111 | See Table 26-1 |

**Notes:**

1. Setting this bit ensures that reading TxTMR will return a valid data value.
2. When this bit is '1', the Timer cannot operate in Sleep mode.
3. CKPOL must not be changed while ON = 1.
4. Setting this bit ensures glitch-free operation when the ON is enabled or disabled.
5. When this bit is set, then the timer operation will be delayed by two input clocks after the ON bit is set.
6. Unless otherwise indicated, all modes start upon ON = 1 and stop upon ON = 0 (stops occur without affecting the value of TxTMR).
7. When TxTMR = TxPR, the next clock clears TxTMR, regardless of the operating mode.

### 26.10.5 TxCLKCON

**Name:** TxCLKCON
**Address:** 0x326,0x332,0x33E

Timer Clock Source Selection Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | CS[4:0] | | | | |
| Access | | | | R/W | R/W | R/W | R/W | R/W |
| Reset | | | | 0 | 0 | 0 | 0 | 0 |

**Bits 4:0 – CS[4:0]** Timer Clock Source Selection

**Table 26-3. Clock Source Selection**

| CS | Clock Source | | |
|---|---|---|---|
| | Timer2 | Timer4 | Timer6 |
| 11111-10110 | Reserved | | |
| 10101 | CLC8_OUT | | |
| 10100 | CLC7_OUT | | |
| 10011 | CLC6_OUT | | |
| 10010 | CLC5_OUT | | |
| 10001 | CLC4_OUT | | |
| 10000 | CLC3_OUT | | |
| 01111 | CLC2_OUT | | |
| 01110 | CLC1_OUT | | |
| 01101 | ZCD_OUT | | |
| 01100 | NCO3_OUT | | |
| 01011 | NCO2_OUT | | |
| 01010 | NCO1_OUT | | |
| 01001 | CLKREF_OUT | | |
| 01000 | EXTOSC | | |
| 00111 | SOSC | | |
| 00110 | MFINTOSC (31.25 kHz) | | |
| 00101 | MFINTOSC (500 kHz) | | |
| 00100 | LFINTOSC | | |
| 00011 | HFINTOSC | | |
| 00010 | $F_{OSC}$ | | |
| 00001 | $F_{OSC}/4$ | | |
| 00000 | Pin selected by T2INPPS | Pin selected by T4INPPS | Pin selected by T6INPPS |

### 26.10.6 TxRST

**Name:** TxRST
**Address:** 0x327,0x333,0x33F

Timer External Reset Signal Selection Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | RSEL[5:0] | | | | | |
| Access | | | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | | | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 5:0 – RSEL[5:0]** External Reset Source Selection

**Table 26-4. External Reset Sources**

| RSEL | Reset Source | | |
|---|---|---|---|
| | TMR2 | TMR4 | TMR6 |
| 111111-100100 | Reserved | | |
| 100011 | U5TX_Edge (Positive/Negative) | | |
| 100010 | U5RX_Edge (Positive/Negative) | | |
| 100001 | U4TX_Edge (Positive/Negative) | | |
| 100000 | U4RX_Edge (Positive/Negative) | | |
| 011111 | U3TX_Edge (Positive/Negative) | | |
| 011110 | U3RX_Edge (Positive/Negative) | | |
| 011101 | U2TX_Edge (Positive/Negative) | | |
| 011100 | U2RX_Edge (Positive/Negative) | | |
| 011011 | U1TX_Edge (Positive/Negative) | | |
| 011010 | U1RX_Edge (Positive/Negative) | | |
| 011001 | CLC8_OUT | | |
| 011000 | CLC7_OUT | | |
| 010111 | CLC6_OUT | | |
| 010110 | CLC5_OUT | | |
| 010101 | CLC4_OUT | | |
| 010100 | CLC3_OUT | | |
| 010011 | CLC2_OUT | | |
| 010010 | CLC1_OUT | | |
| 010001 | ZCD_OUT | | |
| 010000 | CMP2_OUT | | |
| 001111 | CMP1_OUT | | |
| 001110 | Reserved | | |
| 001101 | Reserved | | |
| 001100 | PWM3S1P2_OUT | | |
| 001011 | PWM3S1P1_OUT | | |
| 001010 | PWM2S1P2_OUT | | |
| 001001 | PWM2S1P1_OUT | | |
| 001000 | PWM1S1P2_OUT | | |
| 000111 | PWM1S1P1_OUT | | |
| 000110 | CCP3_OUT | | |
| 000101 | CCP2_OUT | | |
| 000100 | CCP1_OUT | | |

© 2019-2023 Microchip Technology Inc. and its subsidiaries

.....continued

| RSEL | Reset Source | | |
|---|---|---|---|
| | TMR2 | TMR4 | TMR6 |
| 000011 | TMR6_Postscaler_OUT | TMR6_Postscaler_OUT | Reserved |
| 000010 | TMR4_Postscaler_OUT | Reserved | TMR4_Postscaler_OUT |
| 000001 | Reserved | TMR2_Postscaler_OUT | TMR2_Postscaler_OUT |
| 000000 | Pin selected by T2INPPS | Pin selected by T4INPPS | Pin selected by T6INPPS |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

## 26.11 Register Summary - Timer2

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0322 | T2TMR | 7:0 | T2TMR[7:0] | | | | | | | |
| 0x0323 | T2PR | 7:0 | T2PR[7:0] | | | | | | | |
| 0x0324 | T2CON | 7:0 | ON | | CKPS[2:0] | | OUTPS[3:0] | | | |
| 0x0325 | T2HLT | 7:0 | PSYNC | CPOL | CSYNC | MODE[4:0] | | | | |
| 0x0326 | T2CLKCON | 7:0 | | | | CS[4:0] | | | | |
| 0x0327 | T2RST | 7:0 | | | RSEL[5:0] | | | | | |
| 0x0328 ... 0x032D | Reserved | | | | | | | | | |
| 0x032E | T4TMR | 7:0 | T4TMR[7:0] | | | | | | | |
| 0x032F | T4PR | 7:0 | T4PR[7:0] | | | | | | | |
| 0x0330 | T4CON | 7:0 | ON | | CKPS[2:0] | | OUTPS[3:0] | | | |
| 0x0331 | T4HLT | 7:0 | PSYNC | CPOL | CSYNC | MODE[4:0] | | | | |
| 0x0332 | T4CLKCON | 7:0 | | | | CS[4:0] | | | | |
| 0x0333 | T4RST | 7:0 | | | RSEL[5:0] | | | | | |
| 0x0334 ... 0x0339 | Reserved | | | | | | | | | |
| 0x033A | T6TMR | 7:0 | T6TMR[7:0] | | | | | | | |
| 0x033B | T6PR | 7:0 | T6PR[7:0] | | | | | | | |
| 0x033C | T6CON | 7:0 | ON | | CKPS[2:0] | | OUTPS[3:0] | | | |
| 0x033D | T6HLT | 7:0 | PSYNC | CPOL | CSYNC | MODE[4:0] | | | | |
| 0x033E | T6CLKCON | 7:0 | | | | CS[4:0] | | | | |
| 0x033F | T6RST | 7:0 | | | RSEL[5:0] | | | | | |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

# 27. SMT - Signal Measurement Timer

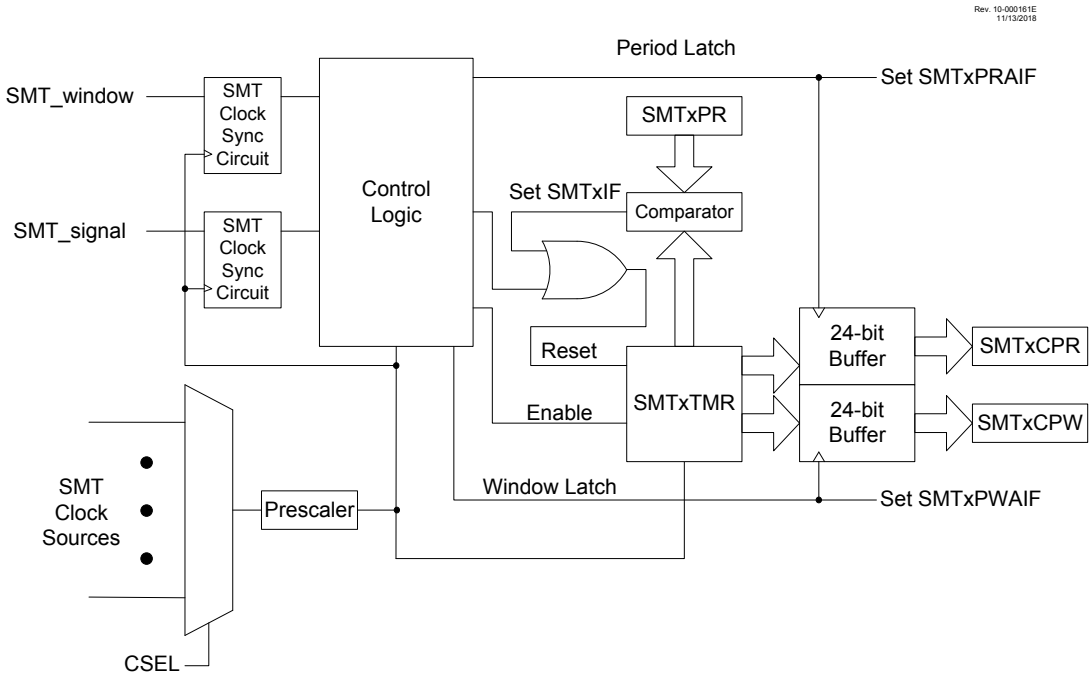
The Signal Measurement Timer (SMT) is a 24-bit counter with advanced clock and gating logic, which can be configured for measuring a variety of digital signal parameters such as pulse width, frequency and duty cycle, and the time difference between edges on two signals.

Features of the SMT include:

- 24-Bit Timer/Counter
- Two 24-Bit Measurement Capture Registers
- One 24-Bit Period Match Register
- Multi-Mode Operation, Including Relative Timing Measurement
- Interrupt-on-Period Match and Acquisition Complete
- Multiple Clock, Signal and Window Sources

Below is the block diagram for the SMT module.

**Figure 27-1. Signal Measurement Timer Block Diagram**




## 27.1 SMT Operation

### 27.1.1 Clock Source Selection

The SMT clock source is selected by configuring the CSEL bits. The clock source is prescaled by using the PS bits. The prescaled clock source is used to clock both the counter and any synchronization logic used by the module.

The polarity of the clock source is selected by using the CPOL bit.

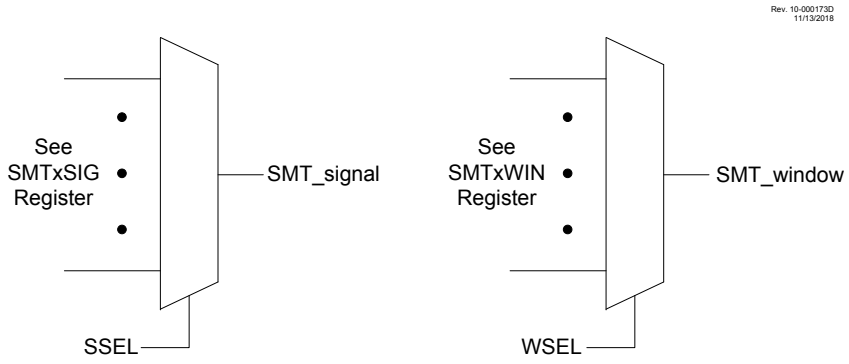
### 27.1.2 Signal and Window Source Selection

The SMT signal and window sources are selected by configuring the SSEL bits and the WSEL bits (refer to the figure below).

The polarity of the signal and window sources is selected by using the SPOL and WPOL bits, respectively.

© 2019-2023 Microchip Technology Inc. and its subsidiaries

**Figure 27-2. SMT Signal and SMT Window Source Selections**




### 27.1.3 Time Base

The SMTxTMR register is the 24-bit counter/timer used for measurement in each of the modes of the SMT. Setting the RST bit clears the SMTxTMR register to 0x000000. It can be written to and read by software. It is not guarded for atomic access, therefore reads and writes to the SMTxTMR register must be made only when GO = 0.

The counter can be prevented from resetting at the end of the timer period by using the STP bit. When STP = 1, the SMTxTMR will stop and remain equal to the SMTxPR register. When STP = 0, the SMTxTMR register resets to 0x000000 at the end of the period.

### 27.1.4 Pulse-Width and Period Captures

The SMTxCPW and SMTxCPR registers are used to latch in the value of the SMTxTMR register, based on the SMT mode of operation. These registers can also be updated with the current value of the SMTxTMR value by setting the CPWUP and CPRUP bits, respectively.

### 27.1.5 Status Information

The SMT provides input status information for the user without requiring the need to monitor the raw incoming signals.

**Go Status:** Timer run status is indicated by the TS bit. The TS bit is delayed in time by synchronizer delays in non-counter modes.

**Signal Status:** Signal status is indicated by the AS bit. This bit is used in all modes, except Window Measure, Time-of-Flight, and Capture modes, and is only valid when TS = 1. The signal status is delayed in time by synchronizer delays in non-counter modes.

**Window Status:** Window status is indicated by the WS bit. This bit is only used in Windowed Measure, Gated Counter, and Gated Window Measure modes, and is only valid when TS = 1. Window status is delayed in time by synchronizer delays in non-counter modes.

### 27.1.6 Modes of Operation

The modes of operation are summarized in the table below. The sections following the table provide descriptions and examples of how each mode can be used. Note that all waveforms assume WPOL/SPOL/CPOL = 0.

For all modes, the REPEAT bit controls whether the acquisition happens only once or is repeated. When REPEAT = 0 (Single Acquisition mode), the timer will stop incrementing and the GO bit will be cleared upon the completion of an acquisition. Otherwise, the timer will continue and allow for continued acquisitions to overwrite the previous ones, until the timer is stopped by software.

**Table 27-1. Modes of Operation**

| MODE | Mode of Operation | Synchronous Operation |
|---|---|---|
| 1111-1011 | Reserved | - |

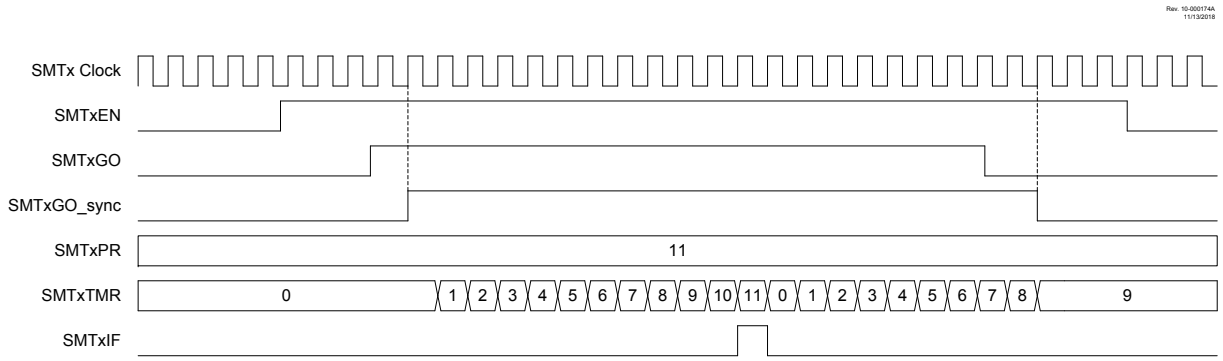
.....continued

| MODE | Mode of Operation | Synchronous Operation |
|---|---|---|
| 1010 | Windowed Counter | No |
| 1001 | Gated Counter | No |
| 1000 | Counter | No |
| 0111 | Capture | Yes |
| 0110 | Time of Flight Measurement | Yes |
| 0101 | Gated Windowed Measurement | Yes |
| 0100 | Windowed Measurement | Yes |
| 0011 | High and Low Time Measurement | Yes |
| 0010 | Period and Duty Cycle Measurement | Yes |
| 0001 | Gated Timer | Yes |
| 0000 | Timer | Yes |

### 27.1.6.1 Timer Mode

Timer mode is the basic mode of operation where the SMTxTMR register is used as a 24-bit timer. No data acquisition takes place in this mode. The timer increments as long as the GO bit has been set by software. No SMT window or SMT signal events affect the GO bit. Everything is synchronized to the SMT clock source. When the timer experiences a period match (SMTxTMR = SMTxPR), the SMTxTMR register is reset and the period match interrupt is set. Refer to the figure below.

**Figure 27-3. Timer Mode Timing Diagram**




### 27.1.6.2 Gated Timer Mode

Gated Timer mode uses the SMT_signal input, selected with the SSEL bits, to control whether or not the SMTxTMR register will increment. Upon a falling edge of the signal, the SMTxCPW register will update to the current value of the SMTxTMR register. Example waveforms for both repeated and single acquisitions are provided in the figures below.

**Figure 27-4. Gated Timer Mode, Repeat Acquisition Timing Diagram**

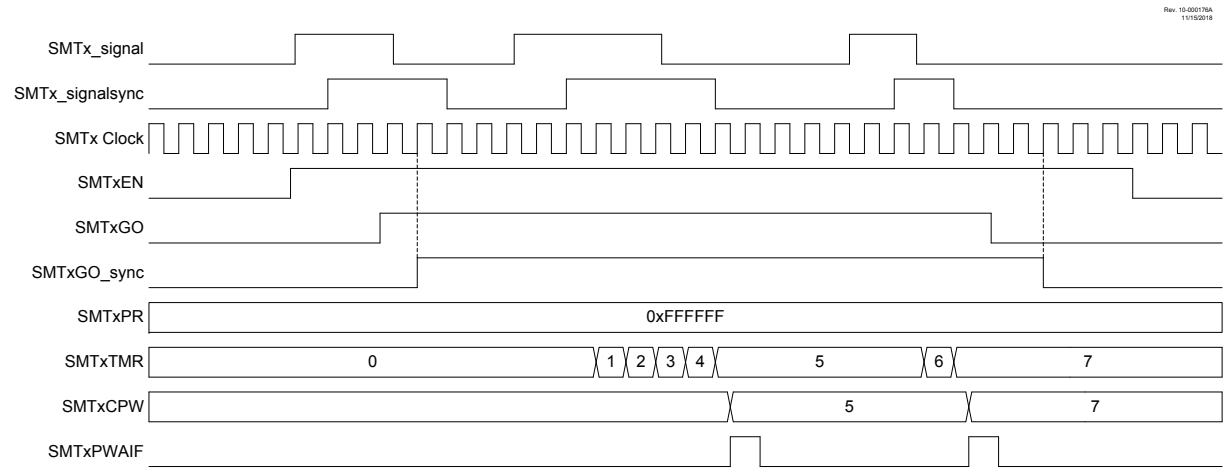
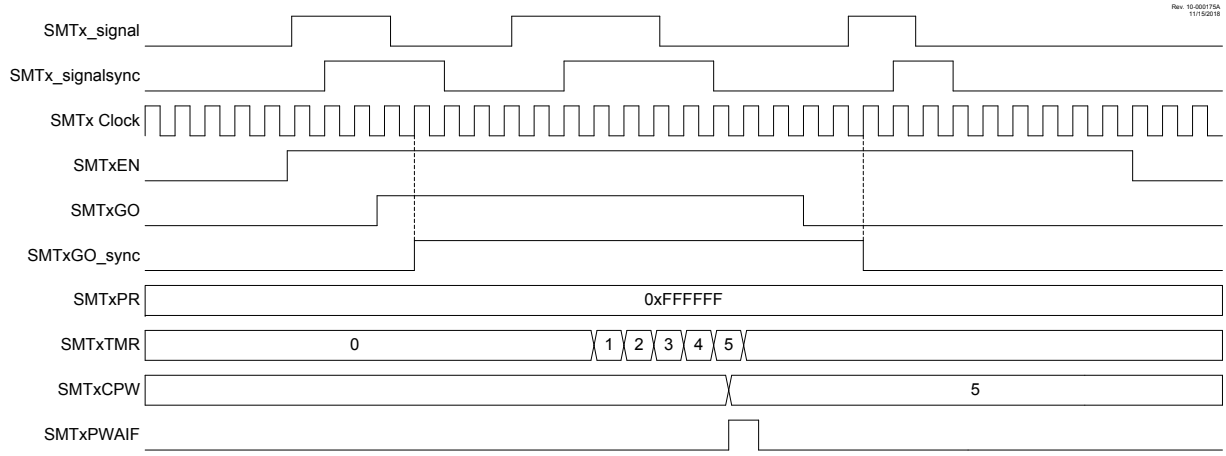



**Figure 27-5. Gated Timer Mode, Single Acquisition Timing Diagram**



### 27.1.6.3 Period and Duty Cycle Measurement Mode

In this mode, either the duty cycle or period of the input signal can be acquired relative to the SMT clock. The SMTxCPW register is updated on a falling edge of the signal, and the SMTxCPR register is updated on a rising edge of the signal. The rising edge also resets the SMTxTMR register to 0x000001. The GO bit is reset on a rising edge when the SMT is in Single Acquisition mode. Refer to the figures below.

**Figure 27-6. Period and Duty Cycle, Repeat Acquisition Mode Timing Diagram**

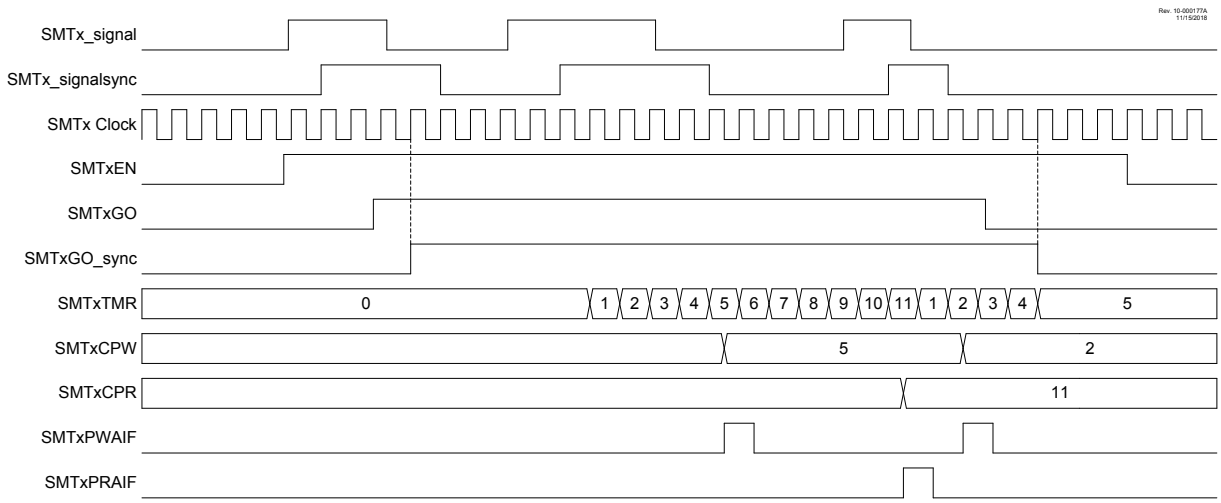
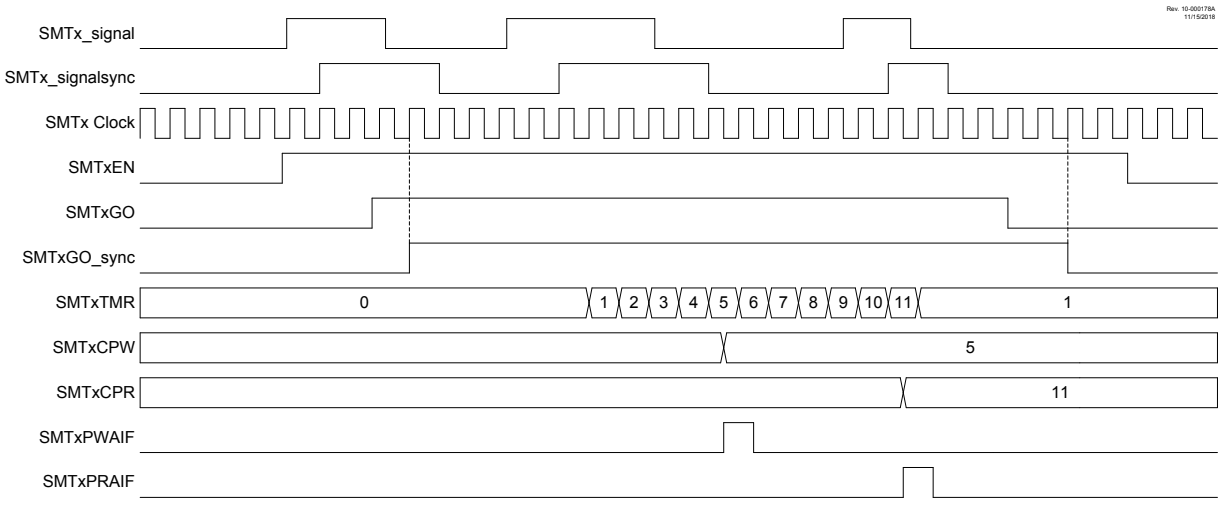



**Figure 27-7. Period and Duty Cycle, Single Acquisition Mode Timing Diagram**



### 27.1.6.4 High and Low Measurement Mode

This mode measures the high and low pulse time of the SMT_signal, relative to the SMT clock. The SMTxTMR register starts incrementing on a rising edge of the input signal. On the falling edge, the SMTxTMR register value is written to the SMTxCPW register. The SMTxTMR register is then reset and continues to increment. On the next rising edge, the SMTxTMR register value is written to the SMTxCPR register. The SMTxTMR register is then reset and continues to increment. Refer to the figures below.

**Figure 27-8. High and Low Measurement Mode, Repeat Acquisition Timing Diagram**

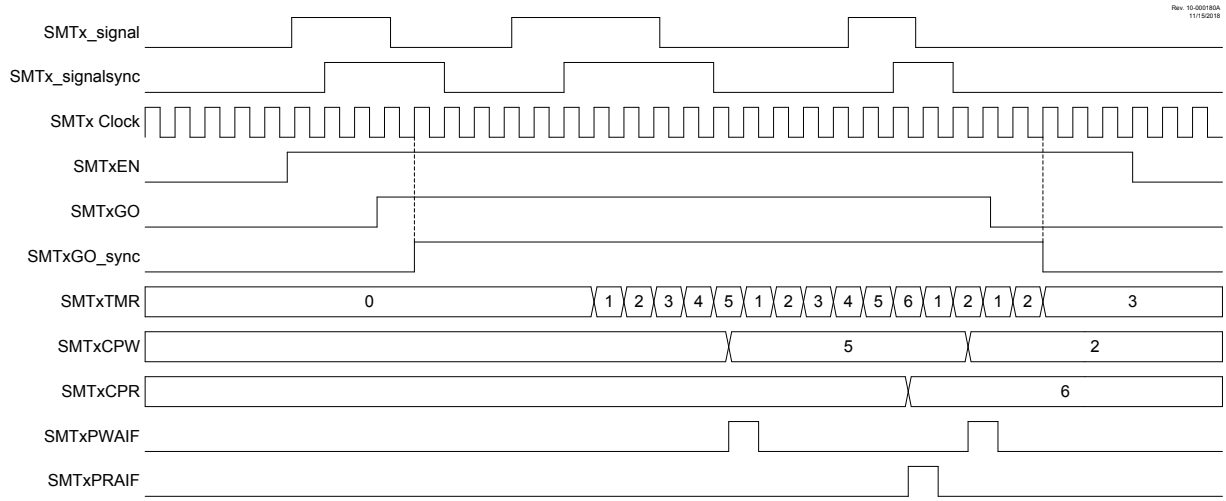
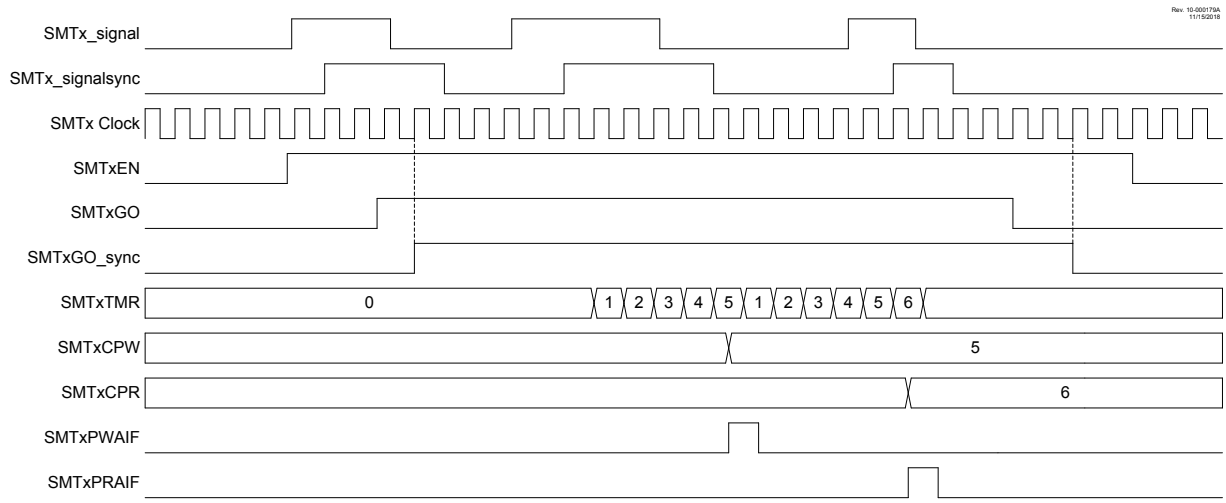



**Figure 27-9. High and Low Measurement Mode, Single Acquisition Timing Diagram**



### 27.1.6.5 Windowed Measurement Mode

This mode measures the period of the SMT_window input, selected with the WSEL bits, relative to the SMT clock. On the rising edge of the window input, the SMTxTMR register value is written to the SMTxCPR register. In Repeat mode, the SMTxTMR register is reset and continues to increment. The capture and Reset process repeats on the next rising edge. Refer to the figures below.

**Figure 27-10. Windowed Measurement Mode, Repeat Acquisition Timing Diagram**

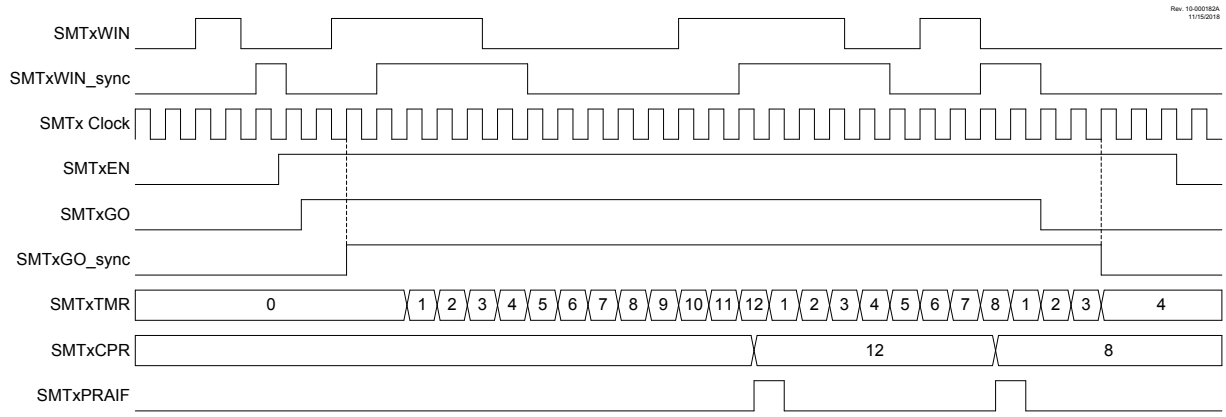
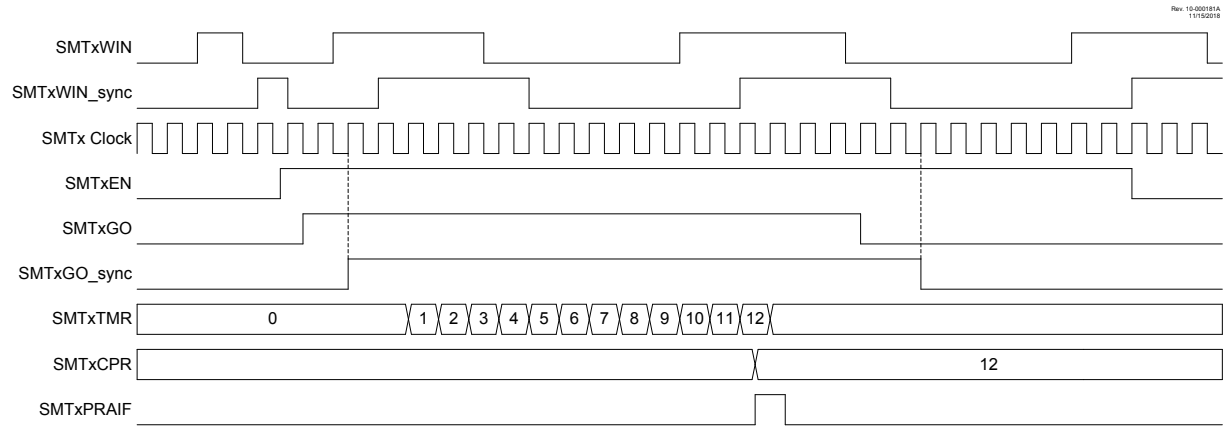



**Figure 27-11. Windowed Measurement Mode, Single Acquisition Timing Diagram**



### 27.1.6.6 Gated Window Measurement Mode

This mode measures the duty cycle of the SMT_signal input over a known input window. It does so by incrementing the SMTxTMR register on each rising edge of the SMTx clock signal when the SMT_signal input is high. The accumulated SMTxTMR register value is written to the SMTxCPR register, and the SMTxTMR register is reset on every rising edge of the window input after the first. Refer to the figures below.

**Figure 27-12. Gated Windowed Measurement Mode, Repeat Acquisition Timing Diagram**

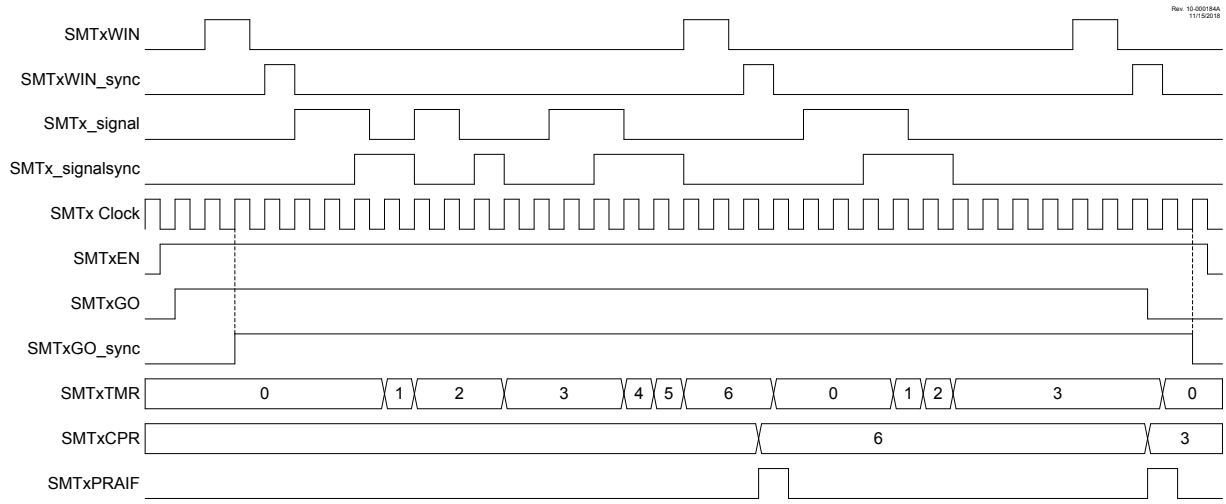
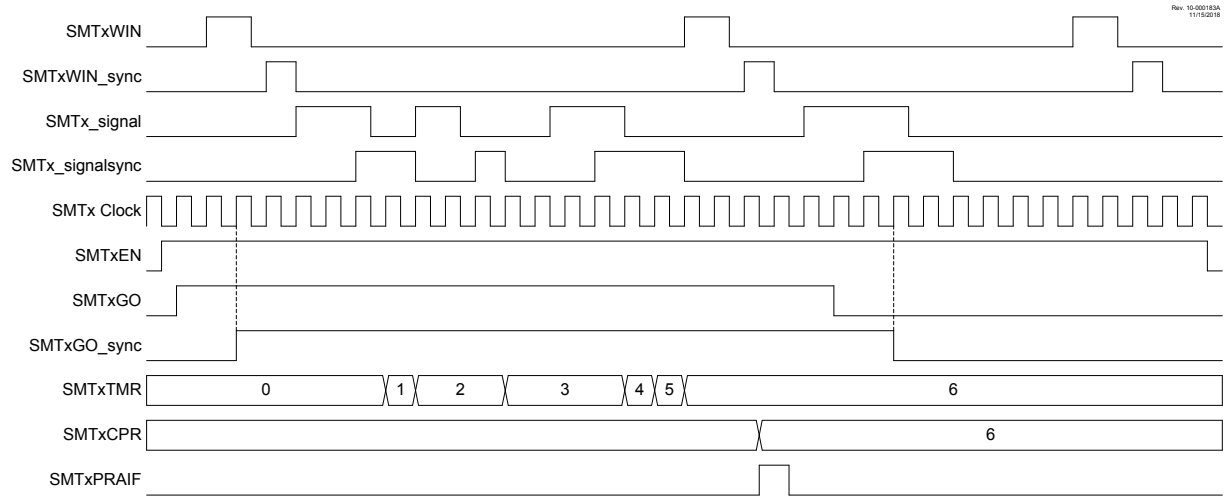



**Figure 27-13. Gated Windowed Measurement Mode, Single Acquisition Timing Diagram**



### 27.1.6.7 Time-of-Flight Measurement Mode

This mode measures the time interval between a rising edge on the SMT_window input and a rising edge on the SMT_signal input. The SMTxTMR register starts incrementing on the rising edge of the window input. The SMTxTMR register value is written to the SMTxCPR register and the SMTxTMR register is reset on a rising edge of the signal input. In the event of two rising edges of the window signal without a signal rising edge, the SMTxCPW register will be written with the current value of the SMTxTMR register, which will then be reset. Refer to the figures below.

**Figure 27-14. Time-of-Flight Mode, Repeat Acquisition Timing Diagram**

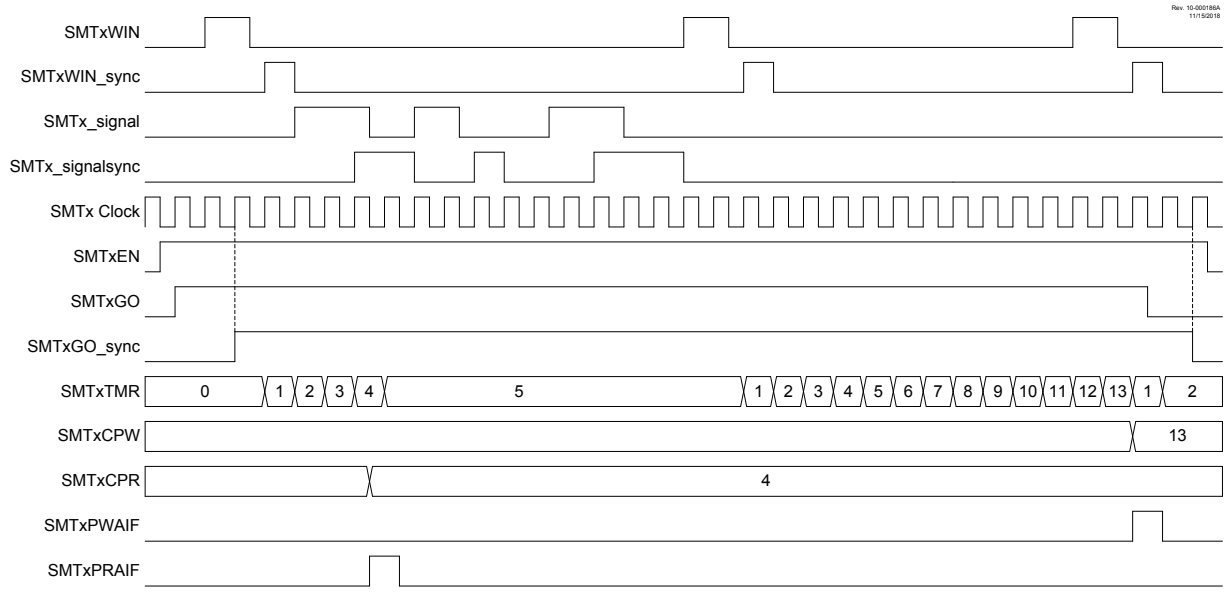
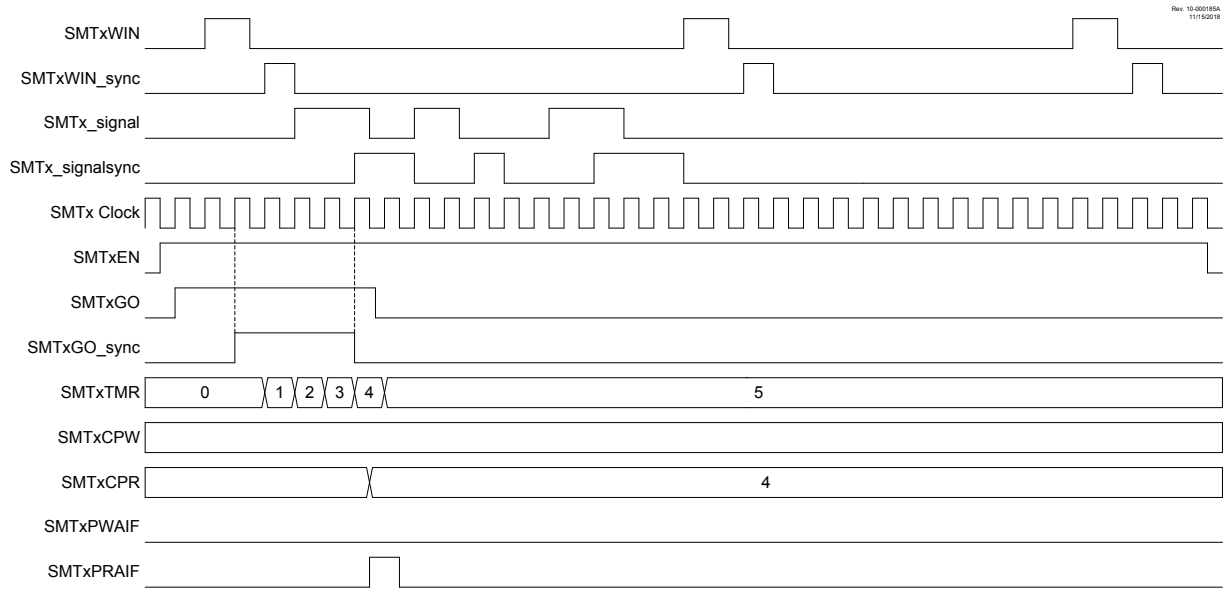



**Figure 27-15. Time-of-Flight Mode, Single Acquisition Timing Diagram**



### 27.1.6.8 Capture Mode

This mode captures the SMTxTMR register value based on a rising or falling edge of the SMT_window input and triggers an interrupt. This mimics the capture feature of a CCP module. The timer begins incrementing upon the GO bit being set. The SMTxTMR register value is written to the SMTxCPR register on each rising edge of the SMT_window input. The SMTxTMR register value is written to the SMTxCPW register on each falling edge of the SMT_window input. The timer is not reset by any hardware conditions in this mode and must be reset by software, if desired. Refer to the figures below.

**Figure 27-16. Capture Mode, Repeat Acquisition Timing Diagram**

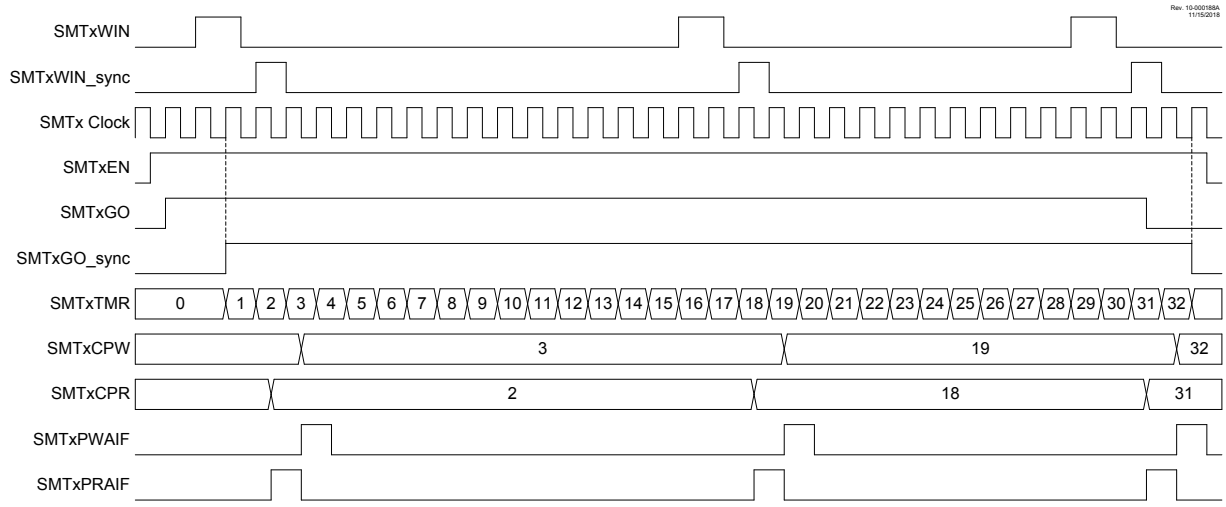
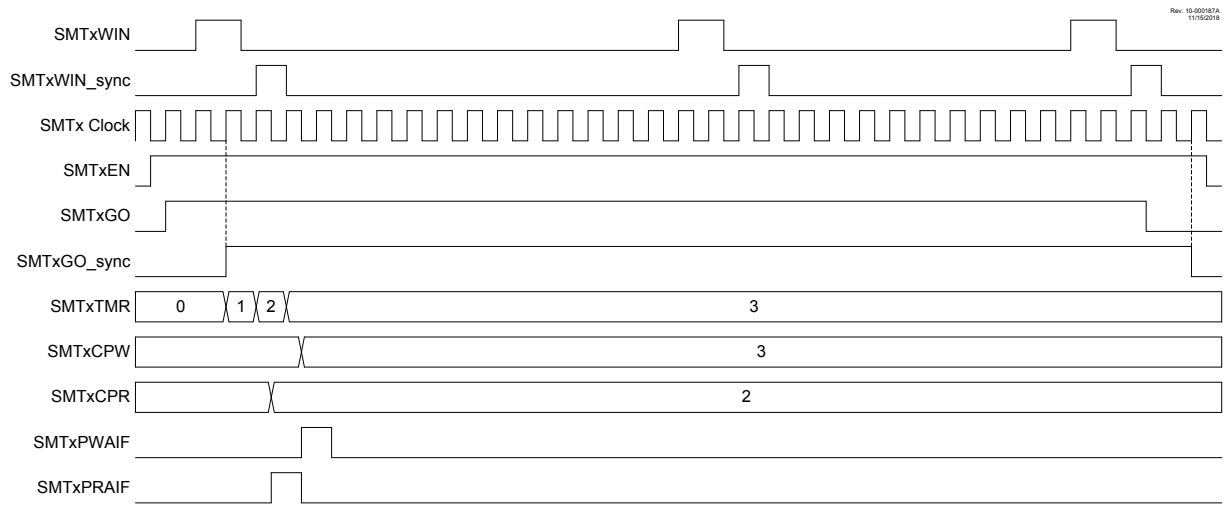


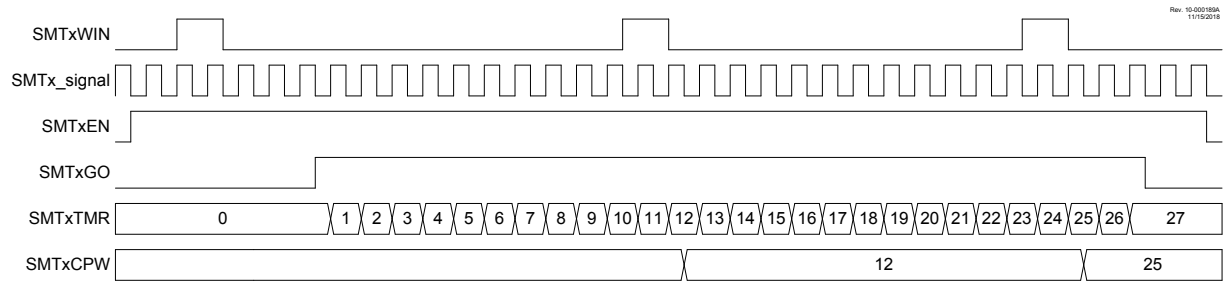

**Figure 27-17. Capture Mode, Single Acquisition Timing Diagram**



### 27.1.6.9 Counter Mode

This mode increments the SMTxTMR register on each rising edge of the SMT_signal input. This mode is asynchronous to the SMT clock and uses the SMT_signal input as a time source. The SMTxCPW register will be updated with the current SMTxTMR register value on the falling edge of the SMT_window input. Refer to the figure below.

**Figure 27-18. Counter Mode Timing Diagram**




### 27.1.6.10 Gated Counter Mode

This mode counts rising edges on the SMT_signal input, gated by the SMT_window input. It increments the SMTxTMR register for each rising edge of the SMT_signal input while the SMT_window input is high. The SMTxTMR register value is written to the SMTxCPW register upon a falling edge of the SMT_window input. Refer to the figures below.

**Figure 27-19. Gated Counter Mode, Repeat Acquisition Timing Diagram**

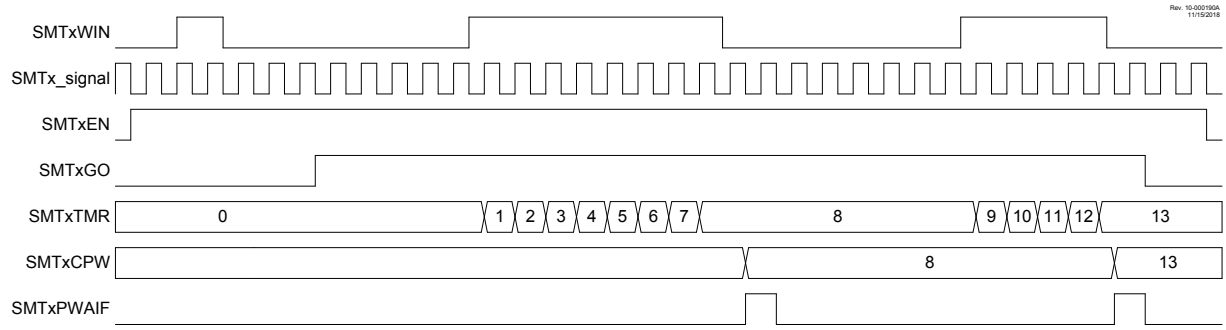
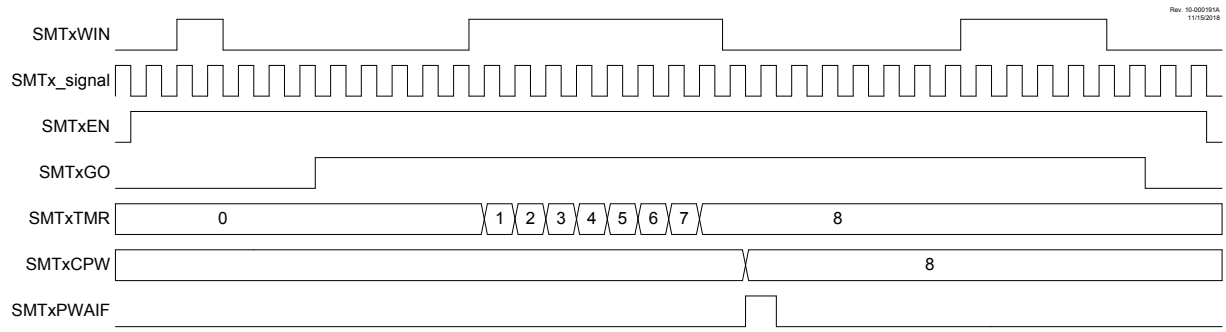



**Figure 27-20. Gated Counter Mode, Single Acquisition Timing Diagram**



#### 27.1.6.11 Windowed Counter Mode

This mode counts rising edges of the SMT_signal between rising edges of the SMT_window input. Beginning with the rising edge of the SMT_window input, the SMTxTMR register is incremented for every rising edge of the SMT_signal input. The SMTxTMR register value is written to the SMTxCPW register on the falling edge of the SMT_window input and the SMTxTMR register continues to increment. The SMTxTMR register value is written to the SMTxCPR register, then reset on each rising edge of the SMT_window input after the first. Refer to the figures below.

**Figure 27-21. Windowed Counter Mode, Repeat Acquisition Timing Diagram**

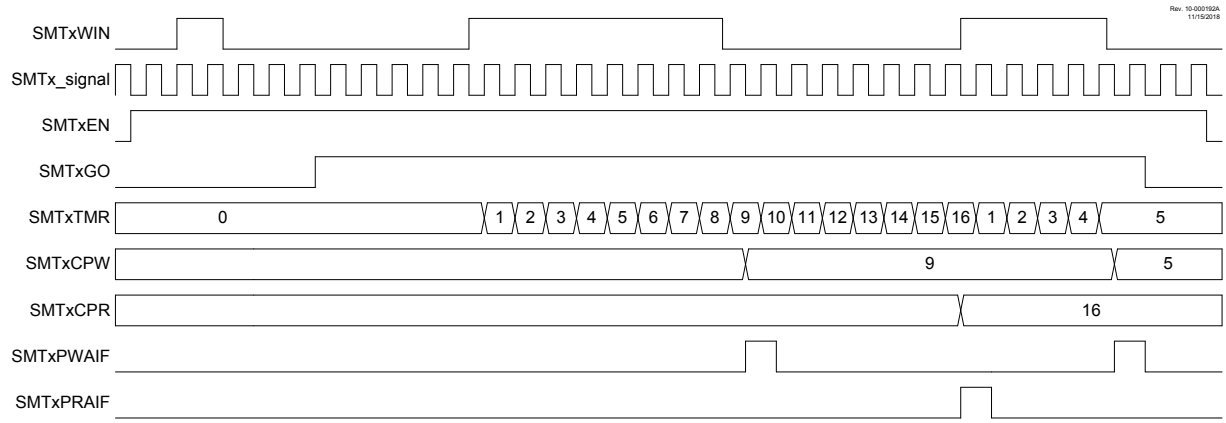
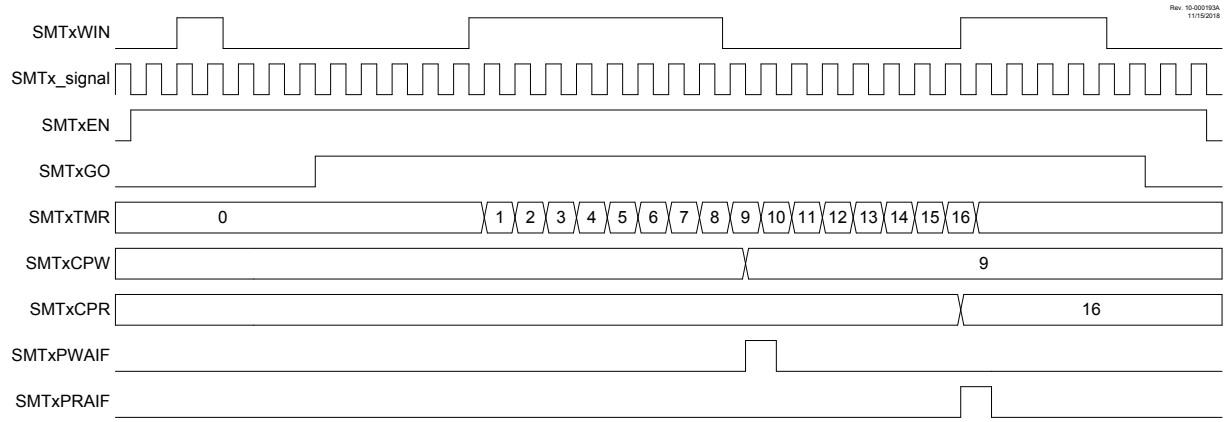



**Figure 27-22. Windowed Counter Mode, Single Acquisition Timing Diagram**




### 27.1.7 Interrupts

The SMT has three interrupts located in one of the PIR registers:

- **Pulse-Width Acquisition Interrupt (SMTxPWAIF):** Interrupt triggers when the SMTxCPW register is updated with the SMTxTMR register value.
- **Period Acquisition Interrupt (SMTxPRAIF):** Interrupt triggers when the SMTxCPR register is updated with the SMTxTMR register value.
- **Counter Period Match Interrupt (SMTxIF):** Interrupt triggers when the SMTxTMR register equals the SMTxPR register.

Each of the above interrupts can be enabled/disabled using the corresponding bits in the PIE register.

### 27.1.8 Operation During Sleep

The SMT can operate during Sleep mode, provided that the clock and signal sources continue to function. In general, internal clock sources, such as HFINTOSC, continue to operate in Sleep mode when selected as the clock source, whereas external oscillators, such as $F_{OSC}$ and $F_{OSC}/4$ cease to operate in Sleep.

## 27.2 Register Definitions: SMT Control

Long bit name prefixes for the SMT peripherals are shown in the table below. Replace the x in SMTx with the SMT peripheral instance number. Refer to the **"Long Bit Names"** section in the **"Register and Nit Naming Conventions"** chapter for more information.

**Table 27-2. SMT Long Bit Name Prefixes**

| Peripheral | Bit Name Prefix |
|---|---|
| SMT1 | SMT1 |

### 27.2.1 SMTxCON0

**Name:** SMTxCON0
**Address:** 0x030C

SMT Control Register 0

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | EN | | STP | WPOL | SPOL | CPOL | PS[1:0] | |
| Access | R/W | | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | | 0 | 0 | 0 | 0 | 0 | 0 |

**Bit 7 – EN** SMT Enable

| Value | Description |
|---|---|
| 1 | SMT is enabled |
| 0 | SMT is disabled; internal states are reset, clock requests are disabled |

**Bit 5 – STP** SMT Counter Halt Enable

| Value | Condition | Description |
|---|---|---|
| 1 | When SMTxTMR = SMTxPR | Counter remains at SMTxPR; period match interrupt occurs when clocked |
| 0 | When SMTxTMR = SMTxPR | Counter resets to 0x000000; period match interrupt occurs when clocked |

**Bit 4 – WPOL** SMT_window Input Polarity Control

| Value | Description |
|---|---|
| 1 | SMT_window input is active-low/falling edge enabled |
| 0 | SMT_window input is active-high/rising edge enabled |

**Bit 3 – SPOL** SMT_signal Input Polarity Control

| Value | Description |
|---|---|
| 1 | SMT_signal input is active-low/falling edge enabled |
| 0 | SMT_signal input is active-high/rising edge enabled |

**Bit 2 – CPOL** SMT Clock Input Polarity Control

| Value | Description |
|---|---|
| 1 | SMTxTMR increments on the falling edge of the selected clock signal |
| 0 | SMTxTMR increments on the rising edge of the selected clock signal |

**Bits 1:0 – PS[1:0]** SMT Prescale Select

| Value | Description |
|---|---|
| 11 | Prescaler = 1:8 |
| 10 | Prescaler = 1:4 |
| 01 | Prescaler = 1:2 |
| 00 | Prescaler = 1:1 |

### 27.2.2 SMTxCON1

**Name:** SMTxCON1
**Address:** 0x030D

SMT Control Register 1

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | GO | REPEAT | | | MODE[3:0] | | | |
| Access | R/W | R/W | | | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | | | 0 | 0 | 0 | 0 |

**Bit 7 – GO** SMT GO Data Acquisition

| Value | Description |
|---|---|
| 1 | Incrementing, acquiring data is enabled |
| 0 | Incrementing, acquiring data is disabled |

**Bit 6 – REPEAT** SMT Repeat Acquisition Enable

| Value | Description |
|---|---|
| 1 | Repeat Data Acquisition mode is enabled |
| 0 | Single Acquisition mode is enabled |

**Bits 3:0 – MODE[3:0]** SMT Operation Mode Select

| Value | Description |
|---|---|
| 1111 | Reserved |
| 1110 | Reserved |
| 1101 | Reserved |
| 1100 | Reserved |
| 1011 | Reserved |
| 1010 | Windowed Counter |
| 1001 | Gated Counter |
| 1000 | Counter |
| 0111 | Capture |
| 0110 | Time-of-Flight |
| 0101 | Gated Windowed Measurement |
| 0100 | Windowed Measurement |
| 0011 | High and Low Time Measurement |
| 0010 | Period and Duty Cycle Acquisition |
| 0001 | Gated Timer |
| 0000 | Timer |

© 2019-2023 Microchip Technology Inc. and its subsidiaries

### 27.2.3 SMTxSTAT

**Name:** SMTxSTAT
**Address:** 0x030E

SMT Status Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | CPRUP | CPWUP | RST | | | TS | WS | AS |
| Access | R/W/HC | R/W/HC | R/W | | | R | R | R |
| Reset | 0 | 0 | 0 | | | 0 | 0 | 0 |

**Bit 7 – CPRUP** SMT Manual Period Buffer Update

| Value | Description |
|---|---|
| 1 | Request write of SMTxTMR value to SMTxCPR registers |
| 0 | SMTxCPR registers update is complete |

**Bit 6 – CPWUP** SMT Manual Pulse-Width Buffer Update

| Value | Description |
|---|---|
| 1 | Request write of SMTxTMR value to SMTxCPW registers |
| 0 | SMTxCPW registers update is complete |

**Bit 5 – RST** SMT Manual Timer Reset

| Value | Description |
|---|---|
| 1 | Request Reset to SMTxTMR registers |
| 0 | SMTxTMR registers update is complete |

**Bit 2 – TS** SMT GO Value Status

| Value | Description |
|---|---|
| 1 | SMTxTMR is incrementing |
| 0 | SMTxTMR is not incrementing |

**Bit 1 – WS** SMT Window Status

| Value | Description |
|---|---|
| 1 | SMT window is open |
| 0 | SMT window is closed |

**Bit 0 – AS** SMT Signal Value Status

| Value | Description |
|---|---|
| 1 | SMT acquisition is in progress |
| 0 | SMT acquisition is not in progress |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

### 27.2.4 SMTxCLK

**Name:** SMTxCLK
**Address:** 0x030F

SMT Clock Selection Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | CSEL[3:0] | | | |
| Access | | | | | R/W | R/W | R/W | R/W |
| Reset | | | | | 0 | 0 | 0 | 0 |

**Bits 3:0 – CSEL[3:0]** SMT Clock Selection

| CSEL Value | SOURCE | Active in Sleep |
|---|---|---|
| 1111-1001 | Reserved | No |
| 1000 | CLKR | No |
| 0111 | EXTOSC | Yes |
| 0110 | SOSC | Yes |
| 0101 | MFINTOSC (31.25 kHz) | Yes |
| 0100 | MFINTOSC (500 kHz) | Yes |
| 0011 | LFINTOSC | Yes |
| 0010 | HFINTOSC | Yes |
| 0001 | FOSC | No |
| 0000 | FOSC/4 | No |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

### 27.2.5 SMTxWIN

**Name:** SMTxWIN
**Address:** 0x0311

SMT Window Input Select Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | WSEL[5:0] | | | | | |
| Access | | | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | | | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 5:0 – WSEL[5:0]** SMT Window Signal Selection

| WSEL Value | Window Source | Active in Sleep |
|---|---|---|
| 111111-101000 | Reserved | No |
| 100111 | CLC8_OUT | No |
| 100110 | CLC7_OUT | No |
| 100101 | CLC6_OUT | No |
| 100100 | CLC5_OUT | No |
| 100011 | CLC4_OUT | No |
| 100010 | CLC3_OUT | No |
| 100001 | CLC2_OUT | No |
| 100000 | CLC1_OUT | No |
| 011111 | ZCD_OUT | No |
| 011110 | CMP2_OUT | No |
| 011101 | CMP1_OUT | No |
| 011100 | NCO3_OUT | No |
| 011011 | NCO2_OUT | No |
| 011010 | NCO1_OUT | No |
| 011001 | Reserved | No |
| 011000 | Reserved | No |
| 010111 | PWM3S1P2_OUT | No |
| 010110 | PWM3S1P1_OUT | No |
| 010101 | PWM2S1P2_OUT | No |
| 010100 | PWM2S1P1_OUT | No |
| 010011 | PWM1S1P2_OUT | No |
| 010010 | PWM1S1P1_OUT | No |
| 010001 | CCP3_OUT | No |
| 010000 | CCP2_OUT | No |
| 001111 | CCP1_OUT | No |
| 001110-001010 | Reserved | No |
| 001001 | TMR6_Postscaler_OUT | No |
| 001000 | TMR4_Postscaler_OUT | No |
| 000111 | TMR2_Postscaler_OUT | No |
| 000110 | TMR0_OUT | No |
| 000101 | CLKREF | No |
| 000100 | EXTOSC | Yes |
| 000011 | SOSC | Yes |
| 000010 | MFINTOSC (31.25 kHz) | Yes |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

.....continued

| WSEL Value | Window Source | Active in Sleep |
|---|---|---|
| 000001 | LFINTOSC | Yes |
| 000000 | SMT1WINPPS | No |

### 27.2.6 SMTxSIG

**Name:** SMTxSIG
**Address:** 0x0310

SMT Signal Selection Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | SSEL[5:0] | | | | | |
| Access | | | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | | | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 5:0 – SSEL[5:0]** SMT Signal Selection

| SSEL Value | Source |
|---|---|
| 111111-100110 | Reserved |
| 100101 | CLC8_OUT |
| 100100 | CLC7_OUT |
| 001101 | CCP1_OUT |
| 001111 | CCP3_OUT |
| 010000 | PWM1S1P1_OUT |
| 010010 | PWM2S1P1_OUT |
| 010011 | PWM2S1P2_OUT |
| 010100 | PWM3S1P1_OUT |
| 010101 | PWM3S1P2_OUT |
| 010110 | Reserved |
| 010111 | Reserved |
| 011000 | NCO1_OUT |
| 011001 | NCO2_OUT |
| 011010 | NCO3_OUT |
| 011011 | CMP1_OUT |
| 011100 | CMP2_OUT |
| 011101 | ZCD_OUT |
| 011110 | CLC1_OUT |
| 011111 | CLC2_OUT |
| 100000 | CLC3_OUT |
| 100010 | CLC5_OUT |
| 100001 | CLC4_OUT |
| 100011 | CLC6_OUT |
| 001110 | CCP2_OUT |
| 010001 | PWM1S1P2_OUT |
| 001100-001000 | Reserved |
| 000111 | TMR6_Postscaler_OUT |
| 000110 | TMR5_OUT |
| 000101 | TMR4_Postscaler_OUT |
| 000100 | TMR3_OUT |
| 000011 | TMR2_Postscaler_OUT |
| 000010 | TMR1_OUT |
| 000001 | TMR0_OUT |
| 000000 | SMT1SIGPPS |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

### 27.2.7 SMTxTMR

**Name:** SMTxTMR
**Address:** 0x0300

SMT Timer Register

| Bit | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|
| | TMR[23:16] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | TMR[15:8] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | TMR[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 23:0 – TMR[23:0]** SMT Timer Value

**Notes:** The individual bytes in this multibyte register can be accessed with the following register names:

- SMTxTMRU: Accesses the upper byte TMR[23:16]
- SMTxTMRH: Accesses the high byte TMR[15:8]
- SMTxTMRL: Accesses the low byte TMR[7:0]

### 27.2.8 SMTxCPR

**Name:** SMTxCPR
**Address:** 0x0303

SMT Captured Period Register

| Bit | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|
| | CPR[23:16] | | | | | | | |
| Access | R | R | R | R | R | R | R | R |
| Reset | x | x | x | x | x | x | x | x |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | CPR[15:8] | | | | | | | |
| Access | R | R | R | R | R | R | R | R |
| Reset | x | x | x | x | x | x | x | x |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | CPR[7:0] | | | | | | | |
| Access | R | R | R | R | R | R | R | R |
| Reset | x | x | x | x | x | x | x | x |

**Bits 23:0 – CPR[23:0]** SMTxTMR Value at Time of Period Capture Event
Reset States: POR/BOR = xxxxxxxxxxxxxxxxxxxxxxxx
All Other Resets = uuuuuuuuuuuuuuuuuuuuuuuu

**Notes:** The individual bytes in this multibyte register can be accessed with the following register names:

- SMTxCPRU: Accesses the upper byte CPR[23:16]
- SMTxCPRH: Accesses the high byte CPR[15:8]
- SMTxCPRL: Accesses the low byte CPR[7:0]

### 27.2.9 SMTxCPW

**Name:** SMTxCPW
**Address:** 0x0306

SMT Captured Pulse-Width Register

| Bit | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|
| | CPW[23:16] | | | | | | | |
| Access | R | R | R | R | R | R | R | R |
| Reset | x | x | x | x | x | x | x | x |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | CPW[15:8] | | | | | | | |
| Access | R | R | R | R | R | R | R | R |
| Reset | x | x | x | x | x | x | x | x |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | CPW[7:0] | | | | | | | |
| Access | R | R | R | R | R | R | R | R |
| Reset | x | x | x | x | x | x | x | x |

**Bits 23:0 – CPW[23:0]** SMTxTMR Value at Time of Capture Event
Reset States: POR/BOR = xxxxxxxxxxxxxxxxxxxxxxxx
All Other Resets = uuuuuuuuuuuuuuuuuuuuuuuu

**Notes:** The individual bytes in this multibyte register can be accessed with the following register names:

- SMTxCPWU: Accesses the upper byte CPW[23:16]
- SMTxCPWH: Accesses the high byte CPW[15:8]
- SMTxCPWL: Accesses the low byte CPW[7:0]

### 27.2.10 SMTxPR

**Name:** SMTxPR
**Address:** 0x0309

SMT Period Register

| Bit | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|
| | PR[23:16] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | PR[15:8] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | PR[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Bits 23:0 – PR[23:0]** The SMTxTMR Value at Which the SMTxTMR Resets to Zero

**Notes:** The individual bytes in this multibyte register can be accessed with the following register names:

- SMTxPRU: Accesses the upper byte PR[23:16]
- SMTxPRH: Accesses the high byte PR[15:8]
- SMTxPRL: Accesses the low byte PR[7:0]

## 27.3 Register Summary - SMT Control

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0300 | SMT1TMR | 7:0 | TMR[7:0] | | | | | | | |
| | | 15:8 | TMR[15:8] | | | | | | | |
| | | 23:16 | TMR[23:16] | | | | | | | |
| 0x0303 | SMT1CPR | 7:0 | CPR[7:0] | | | | | | | |
| | | 15:8 | CPR[15:8] | | | | | | | |
| | | 23:16 | CPR[23:16] | | | | | | | |
| 0x0306 | SMT1CPW | 7:0 | CPW[7:0] | | | | | | | |
| | | 15:8 | CPW[15:8] | | | | | | | |
| | | 23:16 | CPW[23:16] | | | | | | | |
| 0x0309 | SMT1PR | 7:0 | PR[7:0] | | | | | | | |
| | | 15:8 | PR[15:8] | | | | | | | |
| | | 23:16 | PR[23:16] | | | | | | | |
| 0x030C | SMT1CON0 | 7:0 | EN | | STP | WPOL | SPOL | CPOL | PS[1:0] | |
| 0x030D | SMT1CON1 | 7:0 | GO | REPEAT | | | MODE[3:0] | | | |
| 0x030E | SMT1STAT | 7:0 | CPRUP | CPWUP | RST | | | TS | WS | AS |
| 0x030F | SMT1CLK | 7:0 | | | | | CSEL[3:0] | | | |
| 0x0310 | SMT1SIG | 7:0 | | | SSEL[5:0] | | | | | |
| 0x0311 | SMT1WIN | 7:0 | | | WSEL[5:0] | | | | | |

# 28. CCP - Capture/Compare/PWM Module

The Capture/Compare/PWM module is a peripheral that allows the user to time and control different events, and to generate Pulse-Width Modulation (PWM) signals. In Capture mode, the peripheral allows the timing of the duration of an event. The Compare mode allows the user to trigger an external event when a predetermined amount of time has expired. The PWM mode can generate Pulse-Width Modulated signals of varying frequency and duty cycle.

Each individual CCP module can select the timer source that controls the module. The default timer selection is Timer1 when using Capture/Compare mode and Timer2 when using PWM mode in the CCPx module.

Note that the Capture/Compare mode operation is described with respect to Timer1 and the PWM mode operation is described with respect to Timer2 in the following sections.

The Capture and Compare functions are identical for all CCP modules.

**Important:** In devices with more than one CCP module, it is very important to pay close attention to the register names used. Throughout this section, the prefix “CCPx” is used as a generic replacement for specific numbering. A number placed where the “x” is in the prefix is used to distinguish between separate modules. For example, CCP1CON and CCP2CON control the same operational aspects of two completely different CCP modules.

## 28.1 CCP Module Configuration

Each Capture/Compare/PWM module is associated with a control register (CCPxCON), a capture input selection register (CCPxCAP) and a data register (CCPRx). The data register, in turn, is comprised of two 8-bit registers: CCPRxL (low byte) and CCPRxH (high byte).

### 28.1.1 CCP Modules and Timer Resources

The CCP modules utilize Timers 1 through 6 that vary with the selected mode. Various timers are available to the CCP modules in Capture, Compare or PWM modes, as shown in the table below.

**Table 28-1. CCP Mode - Timer Resources**

| CCP Mode | Timer Resource |
|---|---|
| Capture | Timer1, Timer3 or Timer5 |
| Compare | |
| PWM | Timer2, Timer4 or Timer6 |

The assignment of a particular timer to a module is selected as shown in the **“Capture, Compare, and PWM Timers Selection”** chapter. All of the modules may be active at once and may share the same timer resource if they are configured to operate in the same mode (Capture/Compare or PWM) at the same time.

### 28.1.2 Open-Drain Output Option

When operating in Output mode (the Compare or PWM modes), the drivers for the CCPx pins can be optionally configured as open-drain outputs. This feature allows the voltage level on the pin to be pulled to a higher level through an external pull-up resistor and allows the output to communicate with external circuits without the need for additional level shifters.

## 28.2 Capture Mode

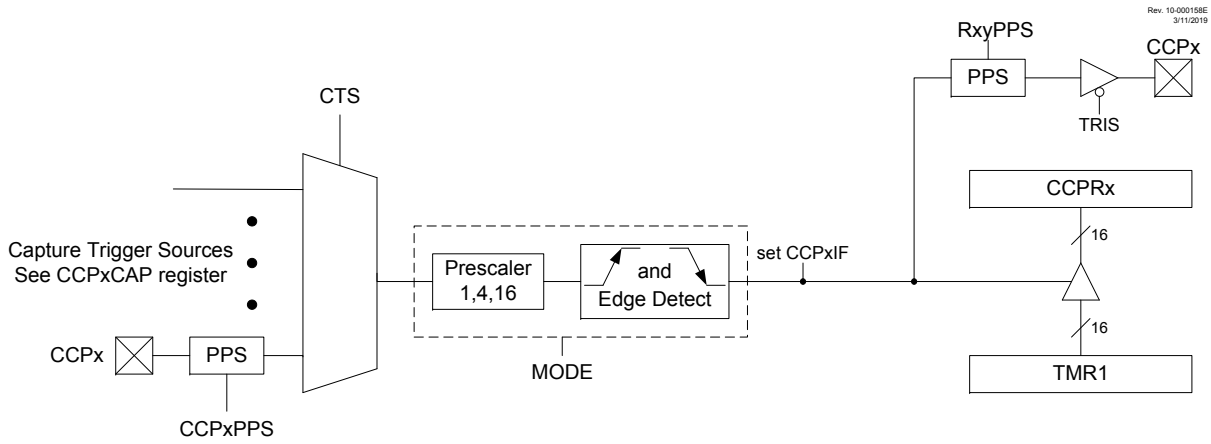
Capture mode makes use of the 16-bit odd numbered timer resources (Timer1, Timer3, etc.). When an event occurs on the capture source, the 16-bit CCPRx register captures and stores the 16-bit value of the TMRx register. An event is defined as one of the following and is configured by the MODE bits:

- Every falling edge of CCPx input
- Every rising edge of CCPx input
- Every 4th rising edge of CCPx input
- Every 16th rising edge of CCPx input
- Every edge of CCPx input (rising or falling)

When a capture is made, the Interrupt Request Flag bit CCPxIF of the PIRx register is set. The interrupt flag must be cleared in software. If another capture occurs before the value in the CCPRx register is read, the old captured value is overwritten by the new captured value. The following figure shows a simplified diagram of the capture operation.

**Important:** If an event occurs during a 2-byte read, the high and low-byte data will be from different events. It is recommended while reading the CCPRx register pair to either disable the module or read the register pair twice for data integrity.

**Figure 28-1. Capture Mode Operation Block Diagram**




### 28.2.1 Capture Sources

The capture source is selected with the CTS bits.

In Capture mode, the CCPx pin must be configured as an input by setting the associated TRIS control bit.

**Important:** If the CCPx pin is configured as an output, a write to the port can cause a capture event.

### 28.2.2 Timer1 Mode for Capture

Timer1 must be running in Timer mode or Synchronized Counter mode for the CCP module to use the capture feature. In Asynchronous Counter mode, the capture operation may not work.

See the **"TMR1 - Timer1 Module with Gate Control"** chapter for more information on configuring Timer1.

### 28.2.3 Software Interrupt Mode

When the Capture mode is changed, a false capture interrupt may be generated. The user will keep the CCPxIE Interrupt Enable bit of the PIEx register clear to avoid false interrupts. Additionally, the user will clear the CCPxIF Interrupt Flag bit of the PIRx register following any change in Operating mode.

© 2019-2023 Microchip Technology Inc. and its subsidiaries

**Important:** Clocking Timer1 from the system clock ($F_{OSC}$) must not be used in Capture mode. For Capture mode to recognize the trigger event on the CCPx pin, Timer1 must be clocked from the instruction clock ($F_{OSC}/4$) or from an external clock source.

### 28.2.4 CCP Prescaler

There are four prescaler settings specified by the MODE bits. Whenever the CCP module is turned off, or the CCP module is not in Capture mode, the prescaler counter is cleared. Any Reset will clear the prescaler counter.

Switching from one capture prescaler to another does not clear the prescaler and may generate a false interrupt. To avoid this unexpected operation, turn the module off by clearing the CCPxCON register before changing the prescaler. The example below demonstrates the code to perform this function.

**Example 28-1. Changing between Capture Prescalers**

```
BANKSEL CCP1CON         ;only needed when CCP1CON is not in ACCESS space
CLRF    CCP1CON         ;Turn CCP module off
MOVLW   NEW_CAPT_PS     ;CCP ON and Prescaler select → W
MOVWF   CCP1CON         ;Load CCP1CON with this value
```

### 28.2.5 Capture During Sleep

Capture mode depends upon the Timer1 module for proper operation. There are two options for driving the Timer1 module in Capture mode. It can be driven by the instruction clock ($F_{OSC}/4$), or by an external clock source.

When Timer1 is clocked by $F_{OSC}/4$, Timer1 will not increment during Sleep. When the device wakes from Sleep, Timer1 will continue from its previous state.

Capture mode will operate during Sleep when Timer1 is clocked by an external clock source.

## 28.3 Compare Mode

The Compare mode function described in this section is available and identical for all CCP modules.

Compare mode makes use of the 16-bit odd numbered Timer resources (Timer1, Timer3, etc.). The 16-bit value of the CCPRx register is constantly compared against the 16-bit value of the TMRx register. When a match occurs, one of the following events can occur:

- Toggle the CCPx output and clear TMRx
- Toggle the CCPx output without clearing TMRx
- Set the CCPx output
- Clear the CCPx output
- Generate a Pulse output
- Generate a Pulse output and clear TMRx

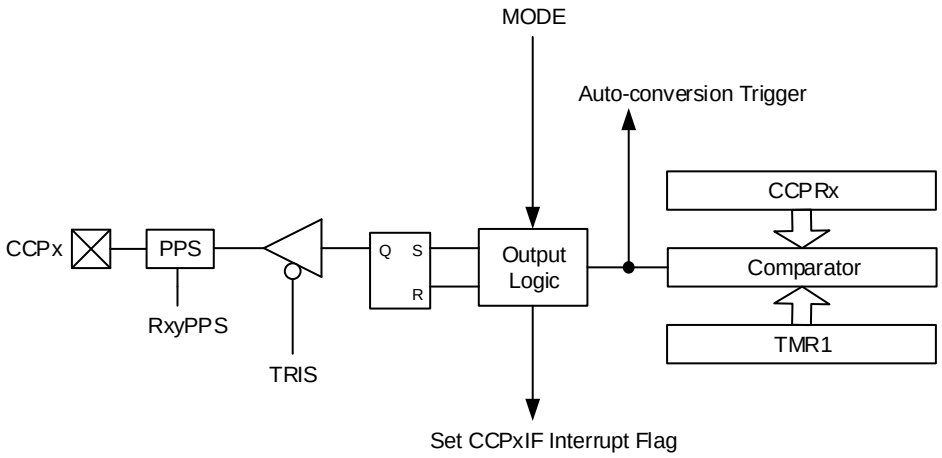
The action on the pin is based on the value of the MODE control bits.

All Compare modes can generate an interrupt. When MODE = `'b0001` or `'b1011`, the CCP resets the TMRx register.

The following figure shows a simplified diagram of the compare operation.

© 2019-2023 Microchip Technology Inc. and its subsidiaries

**Figure 28-2. Compare Mode Operation Block Diagram**




### 28.3.1 CCPx Pin Configuration

The CCPx pin must be configured as an output in software by clearing the associated TRIS bit and defining the appropriate output pin through the RxyPPS registers. See the **"PPS - Peripheral Pin Select Module"** section for more details.

The CCP output can also be used as an input for other peripherals.

**Important:** Clearing the CCPxCON register will force the CCPx compare output latch to the default low level. This is not the PORT I/O data latch.

### 28.3.2 Timer1 Mode for Compare

In Compare mode, Timer1 must be running in either Timer mode or Synchronized Counter mode. The compare operation may not work in Asynchronous Counter mode.

See the "**TMR1 - Timer1 Module with Gate Control**" section for more information on configuring Timer1.

**Important:** Clocking Timer1 from the system clock ($F_{OSC}$) must not be used in Compare mode. For Compare mode to recognize the trigger event on the CCPx pin, Timer1 must be clocked from the instruction clock ($F_{OSC}/4$) or from an external clock source.

### 28.3.3 Compare During Sleep

Since $F_{OSC}$ is shut down during Sleep mode, the Compare mode will not function properly during Sleep, unless the timer is running. The device will wake on interrupt (if enabled).

## 28.4 PWM Overview

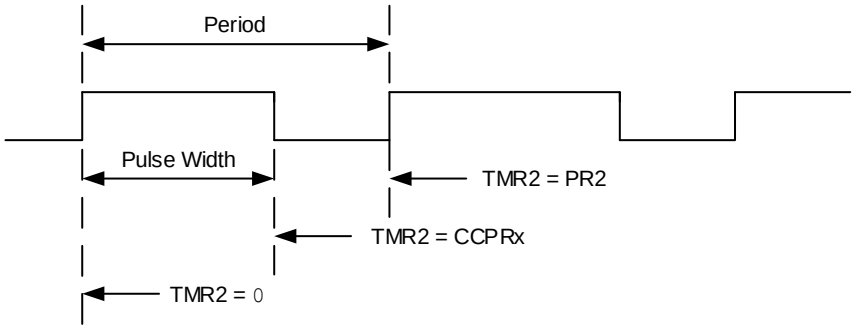
Pulse-Width Modulation (PWM) is a scheme that controls power to a load by switching quickly between fully ON and fully OFF states. The PWM signal resembles a square wave where the high portion of the signal is considered the ON state and the low portion of the signal is considered the OFF state. The high portion, also known as the pulse width, can vary in time and is defined in steps. A larger number of steps applied, which lengthens the pulse width, also supplies more power to the load. Lowering the number of steps applied, which shortens the pulse width, supplies less power. The PWM period is defined as the duration of one complete cycle or the total amount of ON and OFF time combined.

© 2019-2023 Microchip Technology Inc. and its subsidiaries

PWM resolution defines the maximum number of steps that can be present in a single PWM period. A higher resolution allows for more precise control of the power applied to the load.

The term duty cycle describes the proportion of the ON time to the OFF time and is expressed in percentages, where 0% is fully OFF and 100% is fully ON. A lower duty cycle corresponds to less power applied and a higher duty cycle corresponds to more power applied. The figure below shows a typical waveform of the PWM signal.

**Figure 28-3. CCP PWM Output Signal**



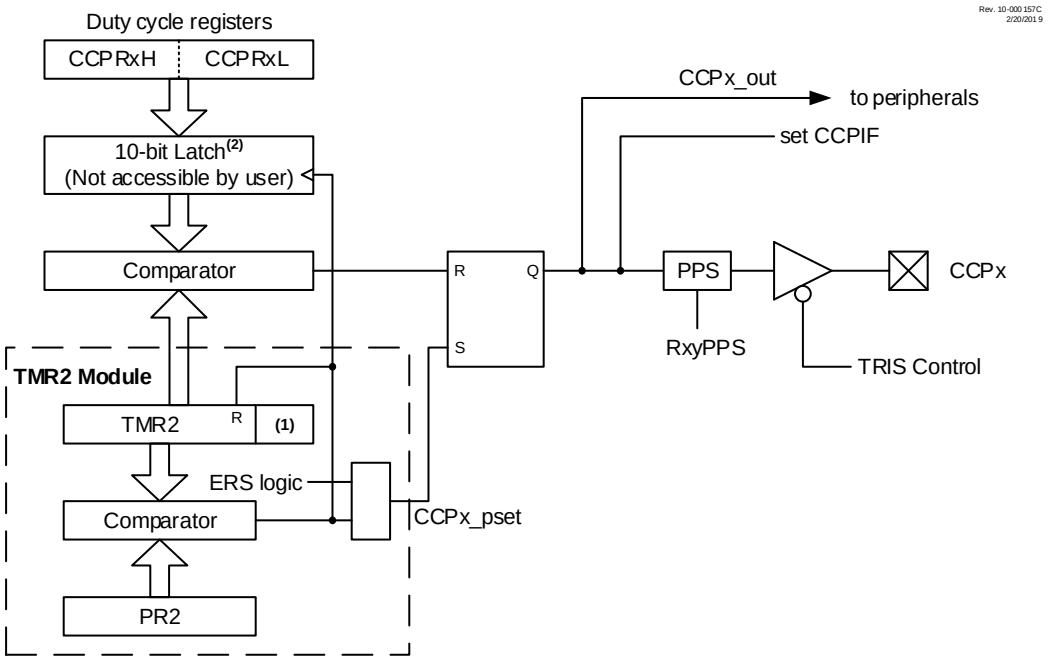

### 28.4.1 Standard PWM Operation

The standard PWM function described in this section is available and identical for all CCP modules. It generates a Pulse-Width Modulation (PWM) signal on the CCPx pin with up to ten bits of resolution. The period, duty cycle, and resolution are controlled by the following registers:

- Even numbered TxPR registers (T2PR, T4PR, etc.)
- Even numbered TxCON registers (T2CON, T4CON, etc.)
- 16-bit CCPRx registers
- CCPxCON registers

It is required to have $F_{OSC}/4$ as the clock input to TxTMR for correct PWM operation. The following figure shows a simplified block diagram of the PWM operation.

**Figure 28-4. Simplified PWM Block Diagram**




**Notes**: 1. An 8-bit timer is concatenated with two bits generated by Fosc or two bits of the internal prescaler to create 10-bit time base.
2. The alignment of the 10 bits from the CCPR register is determined by the CCPxFMT bit.

**Important:** The corresponding TRIS bit must be cleared to enable the PWM output on the CCPx pin.

### 28.4.2 Setup for PWM Operation

The following steps illustrate how to configure the CCP module for standard PWM operation:

1. Select the desired output pin with the RxyPPS control to select CCPx as the source. Disable the selected pin output driver by setting the associated TRIS bit. The output will be enabled later at the end of the PWM setup.
2. Load the selected timer TxPR period register with the PWM period value.
3. Configure the CCP module for the PWM mode by loading the CCPxCON register with the appropriate values.
4. Load the CCPRx register with the PWM duty cycle value and configure the FMT bit to set the proper register alignment.
5. Configure and start the selected timer:
   - Clear the TMRxIF Interrupt Flag bit of the PIRx register. See the Note below.
   - Select the timer clock source to be as $F_{OSC}/4$. This is required for correct operation of the PWM module.
   - Configure the TxCKPS bits of the TxCON register with the desired timer prescale value.
   - Enable the timer by setting the TxON bit.
6. Enable the PWM output:
   - Wait until the timer overflows and the TMRxIF bit of the PIRx register is set. See the Note below.
   - Enable the CCPx pin output driver by clearing the associated TRIS bit.

**Important:** To send a complete duty cycle and period on the first PWM output, the above steps must be included in the setup sequence. If it is not critical to start with a complete PWM signal on the first output, then step 6 may be ignored.

### 28.4.3 Timer2 Timer Resource

The PWM Standard mode makes use of the 8-bit Timer2 timer resources to specify the PWM period.

### 28.4.4 PWM Period

The PWM period is specified by the T2PR register of Timer2. The PWM period can be calculated using the formula in the equation below.

**Equation 28-1. PWM Period**

$$PWM\ Period = [(T2PR + 1)] \bullet 4 \bullet T_{OSC} \bullet (TMR2\ Prescale\ Value)$$

where $T_{OSC} = 1/F_{OSC}$

When T2TMR is equal to T2PR, the following three events occur on the next increment event:

- T2TMR is cleared
- The CCPx pin is set (Exception: If the PWM duty cycle = 0%, the pin will not be set)
- The PWM duty cycle is transferred from the CCPRx register into a 10-bit buffer

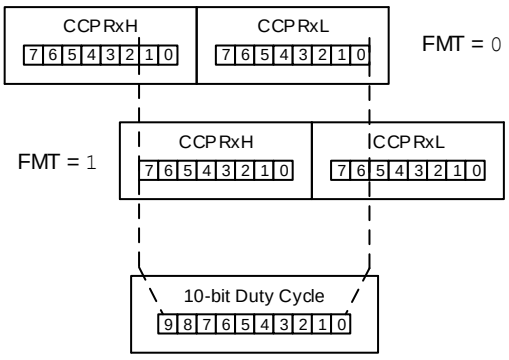
**Important:** The Timer postscaler (see the "**Timer2 Interrupt**" section) is not used in the determination of the PWM frequency.

### 28.4.5 PWM Duty Cycle

The PWM duty cycle is specified by writing a 10-bit value to the CCPRx register. The alignment of the 10-bit value is determined by the FMT bit (see Figure 28-5). The CCPRx register can be written to at any time. However, the duty cycle value is not latched onto the 10-bit buffer until after a match between T2PR and T2TMR.

The equations below are used to calculate the PWM pulse width and the PWM duty cycle ratio.

**Figure 28-5. PWM 10-Bit Alignment**




**Equation 28-2. Pulse Width**

$$Pulse\ Width = (CCPRxH{:}CCPRxL\ register\ value) \bullet T_{OSC} \bullet (TMR2\ Prescale\ Value)$$

**Equation 28-3. Duty Cycle**

$$DutyCycleRatio = \frac{(CCPRxH{:}CCPRxL\ register\ value)}{4(T2PR + 1)}$$

The CCPRx register is used to double buffer the PWM duty cycle. This double buffering is essential for glitchless PWM operation.

The 8-bit timer T2TMR register is concatenated with either the 2-bit internal system clock ($F_{OSC}$), or two bits of the prescaler, to create the 10-bit time base. The system clock is used if the Timer2 prescaler is set to 1:1.

When the 10-bit time base matches the CCPRx register, then the CCPx pin is cleared (see Figure 28-4).

### 28.4.6 PWM Resolution

The resolution determines the number of available duty cycles for a given period. For example, a 10-bit resolution will result in 1024 discrete duty cycles, whereas an 8-bit resolution will result in 256 discrete duty cycles.

The maximum PWM resolution is 10 bits when T2PR is `0xFF`. The resolution is a function of the T2PR register value, as shown below.

**Equation 28-4. PWM Resolution**

$$\text{Resolution} = \frac{\log[4(T2PR+1)]}{\log(2)} bits$$

**Important:** If the pulse-width value is greater than the period, the assigned PWM pin(s) will remain unchanged.

**Table 28-2. Example PWM Frequencies and Resolutions ($F_{OSC}$ = 20 MHz)**

| PWM Frequency | 1.22 kHz | 4.88 kHz | 19.53 kHz | 78.12 kHz | 156.3 kHz | 208.3 kHz |
|---|---|---|---|---|---|---|
| Timer Prescale | 16 | 4 | 1 | 1 | 1 | 1 |
| T2PR Value | 0xFF | 0xFF | 0xFF | 0x3F | 0x1F | 0x17 |
| Maximum Resolution (bits) | 10 | 10 | 10 | 8 | 7 | 6.6 |

**Table 28-3. Example PWM Frequencies and Resolutions ($F_{OSC}$ = 8 MHz)**

| PWM Frequency | 1.22 kHz | 4.90 kHz | 19.61 kHz | 76.92 kHz | 153.85 kHz | 200.0 kHz |
|---|---|---|---|---|---|---|
| Timer Prescale | 16 | 4 | 1 | 1 | 1 | 1 |
| T2PR Value | 0x65 | 0x65 | 0x65 | 0x19 | 0x0C | 0x09 |
| Maximum Resolution (bits) | 8 | 8 | 8 | 6 | 5 | 5 |

### 28.4.7 Operation in Sleep Mode

In Sleep mode, the T2TMR register will not increment and the state of the module will not change. If the CCPx pin is driving a value, it will continue to drive that value. When the device wakes up, T2TMR will continue from the previous state.

### 28.4.8 Changes in System Clock Frequency

The PWM frequency is derived from the system clock frequency. Any changes in the system clock frequency will result in changes to the PWM frequency. See the **"OSC - Oscillator Module (with Fail-Safe Clock Monitor)"** chapter for additional details.

### 28.4.9 Effects of Reset

Any Reset will force all ports to Input mode and the CCP registers to their Reset states.

## 28.5 Register Definitions: CCP Control

Long bit name prefixes for the CCP peripherals are shown in the following table. Refer to the **"Long Bit Names"** section in the **"Register and Bit Naming Conventions"** chapter for more information.

**Table 28-4. CCP Long Bit Name Prefixes**

| Peripheral | Bit Name Prefix |
|---|---|
| CCP1 | CCP1 |
| CCP2 | CCP2 |
| CCP3 | CCP3 |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

### 28.5.1 CCPxCON

**Name:** CCPxCON
**Address:** 0x342,0x346,0x34A

CCP Control Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | EN | | OUT | FMT | MODE[3:0] | | | |
| Access | R/W | | R | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | | x | 0 | 0 | 0 | 0 | 0 |

**Bit 7 – EN** CCP Module Enable

| Value | Description |
|---|---|
| 1 | CCP is enabled |
| 0 | CCP is disabled |

**Bit 5 – OUT** CCP Output Data (read-only)

**Bit 4 – FMT** CCPxRH:L Value Alignment (PWM mode)

| Value | Condition | Description |
|---|---|---|
| x | Capture mode | Not used |
| x | Compare mode | Not used |
| 1 | PWM mode | Left aligned format |
| 0 | PWM mode | Right aligned format |

**Bits 3:0 – MODE[3:0]** CCP Mode Select

**Table 28-5. CCPx Mode Select**

| MODE Value | Operating Mode | Operation | Set CCPxIF |
|---|---|---|---|
| 11xx | PWM | PWM operation | Yes |
| 1011 | Compare | Pulse output; clear TMR1(2) | Yes |
| 1010 | | Pulse output | Yes |
| 1001 | | Clear output(1) | Yes |
| 1000 | | Set output(1) | Yes |
| 0111 | Capture | Every 16th rising edge of CCPx input | Yes |
| 0110 | | Every 4th rising edge of CCPx input | Yes |
| 0101 | | Every rising edge of CCPx input | Yes |
| 0100 | | Every falling edge of CCPx input | Yes |
| 0011 | | Every edge of CCPx input | Yes |
| 0010 | Compare | Toggle output | Yes |
| 0001 | | Toggle output; clear TMR1(2) | Yes |
| 0000 | Disabled | | — |

**Notes:**

1. The set and clear operations of the Compare mode are reset by setting MODE = 'b0000 or EN = 0.
2. When MODE = 'b0001 or 'b1011, then the timer associated with the CCP module is cleared. TMR1 is the default selection for the CCP module, so it is used for indication purposes only.

### 28.5.2 CCPxCAP

**Name:** CCPxCAP
**Address:** 0x343,0x347,0x34B

Capture Trigger Input Selection Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | CTS[3:0] | | | |
| Access | | | | | R/W | R/W | R/W | R/W |
| Reset | | | | | 0 | 0 | 0 | 0 |

**Bits 3:0 – CTS[3:0]** Capture Trigger Input Selection

**Table 28-6. Capture Trigger Sources**

| CTS Value | Source |
|---|---|
| 1111-1100 | Reserved |
| 1011 | CLC8_OUT |
| 1010 | CLC7_OUT |
| 1001 | CLC6_OUT |
| 1000 | CLC5_OUT |
| 0111 | CLC4_OUT |
| 0110 | CLC3_OUT |
| 0101 | CLC2_OUT |
| 0100 | CLC1_OUT |
| 0011 | IOC Interrupt |
| 0010 | CMP2_OUT |
| 0001 | CMP1_OUT |
| 0000 | Pin selected by CCPxPPS |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

### 28.5.3 CCPRx

**Name:** CCPRx
**Address:** 0x340,0x344,0x348

Capture/Compare/Pulse-Width Register

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | CCPR[15:8] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | x | x | x | x | x | x | x | x |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | CCPR[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | x | x | x | x | x | x | x | x |

**Bits 15:0 – CCPR[15:0]** Capture/Compare/Pulse-Width
Reset States: POR/BOR = xxxxxxxxxxxxxxxx
All other Resets = uuuuuuuuuuuuuuuu

**Notes:** The individual bytes in this multibyte register can be accessed with the following register names:

- When MODE = Capture or Compare
  - CCPRxH: Accesses the high byte CCPR[15:8]
  - CCPRxL: Accesses the low byte CCPR[7:0]
- When MODE = PWM and FMT = 0
  - CCPRx[15:10]: Not used
  - CCPRxH[1:0]: Accesses the two Most Significant bits CCPR[9:8]
  - CCPRxL: Accesses the eight Least Significant bits CCPR[7:0]
- When MODE = PWM and FMT = 1
  - CCPRxH: Accesses the eight Most Significant bits CCPR[9:2]
  - CCPRxL[7:6]: Accesses the two Least Significant bits CCPR[1:0]
  - CCPRx[5:0]: Not used

## 28.6 Register Summary - CCP Control

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0340 | CCPR1 | 7:0 | CCPR[7:0] | | | | | | | |
| | | 15:8 | CCPR[15:8] | | | | | | | |
| 0x0342 | CCP1CON | 7:0 | EN | | OUT | FMT | MODE[3:0] | | | |
| 0x0343 | CCP1CAP | 7:0 | | | | | CTS[3:0] | | | |
| 0x0344 | CCPR2 | 7:0 | CCPR[7:0] | | | | | | | |
| | | 15:8 | CCPR[15:8] | | | | | | | |
| 0x0346 | CCP2CON | 7:0 | EN | | OUT | FMT | MODE[3:0] | | | |
| 0x0347 | CCP2CAP | 7:0 | | | | | CTS[3:0] | | | |
| 0x0348 | CCPR3 | 7:0 | CCPR[7:0] | | | | | | | |
| | | 15:8 | CCPR[15:8] | | | | | | | |
| 0x034A | CCP3CON | 7:0 | EN | | OUT | FMT | MODE[3:0] | | | |
| 0x034B | CCP3CAP | 7:0 | | | | | CTS[3:0] | | | |

# 29. Capture, Compare, and PWM Timers Selection

Each of these modules has an independent timer selection which can be accessed using the timer selection register. The default timer selection is Timer1 for capture or compare functions and Timer2 for PWM functions.

## 29.1 Register Definitions: Capture, Compare, and PWM Timers Selection

### 29.1.1 CCPTMRS0

**Name:** CCPTMRS0
**Address:** 0x34C

CCP Timers Selection Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | C3TSEL[1:0] | | C2TSEL[1:0] | | C1TSEL[1:0] | |
| Access | | | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | | | 0 | 1 | 0 | 1 | 0 | 1 |

**Bits 0:1, 2:3, 4:5 – CnTSEL** CCPn Timer Selection

| CnTSEL Value | Capture/Compare | PWM |
|---|---|---|
| 11 | Timer5 | Timer6 |
| 10 | Timer3 | Timer4 |
| 01 | Timer1 | Timer2 |
| 00 | Reserved | |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

## 29.2 Register Summary - Capture, Compare, and PWM Timers Selection

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x034C | CCPTMRS0 | 7:0 | | | C3TSEL[1:0] | | C2TSEL[1:0] | | C1TSEL[1:0] | |

# 30. PWM - Pulse-Width Modulator with Compare

This module is a 16-bit Pulse-Width Modulator (PWM) with a compare feature and multiple outputs. The outputs are grouped in slices where each slice has two outputs. There can be up to four slices in each PWM module. The EN bit enables the PWM operation for all slices simultaneously. The prescale counter, postscale counter, and all internal logic is held in Reset while the EN bit is low.

Features of this module include the following:

- Five main operating modes:
  - Left Aligned
  - Right Aligned
  - Center-Aligned
  - Variable Aligned
  - Compare
    - Pulsed
    - Toggled
- Push-pull operation (available in Left and Right Aligned modes only)
- Independent 16-bit period timer
- Programmable clock sources
- Programmable trigger sources for synchronous duty cycle and period changes
- Programmable synchronous/asynchronous Reset sources
- Programmable Reset source polarity control
- Programmable PWM output polarity control
- Up to four two-output slices per module

Block diagrams of each PWM mode are shown in their respective sections.

## 30.1 Output Slices

A PWM module can have up to four output slices. An output slice consists of two PWM outputs, PWMx_SaP1_out and PWMx_SaP2_out. Both share the same operating mode. However, other slices may operate in a different mode. PWMx_SaP1_out and PWMx_SaP2_out have independent duty cycles which are set with the respective P1 and P2 parameter registers.

### 30.1.1 Output Polarity

The polarity for the PWMx_SaP1_out and PWMx_SaP2_out is controlled with the respective POL1 and POL2 bits. Setting the polarity bit inverts the output Active state to Low True. Toggling the polarity bit toggles the output whether or not the PWM module is enabled.

### 30.1.2 Operating Modes

Each output slice can operate in one of six modes selected with the MODE bits. The Left and Right Aligned modes can also be operated in Push-Pull mode by setting the PPEN bit. The following sections provide more details on each mode, including block diagrams.

### 30.1.2.1 Left Aligned Mode

In Left Aligned mode, the active part of the duty cycle is at the beginning of the period. The outputs start active and stay active for the number of prescaled PWM clock periods specified by the P1 and P2 parameter registers, then go inactive for the remainder of the period. Block and timing diagrams follow.

**Figure 30-1. Left-Aligned Block Diagram**

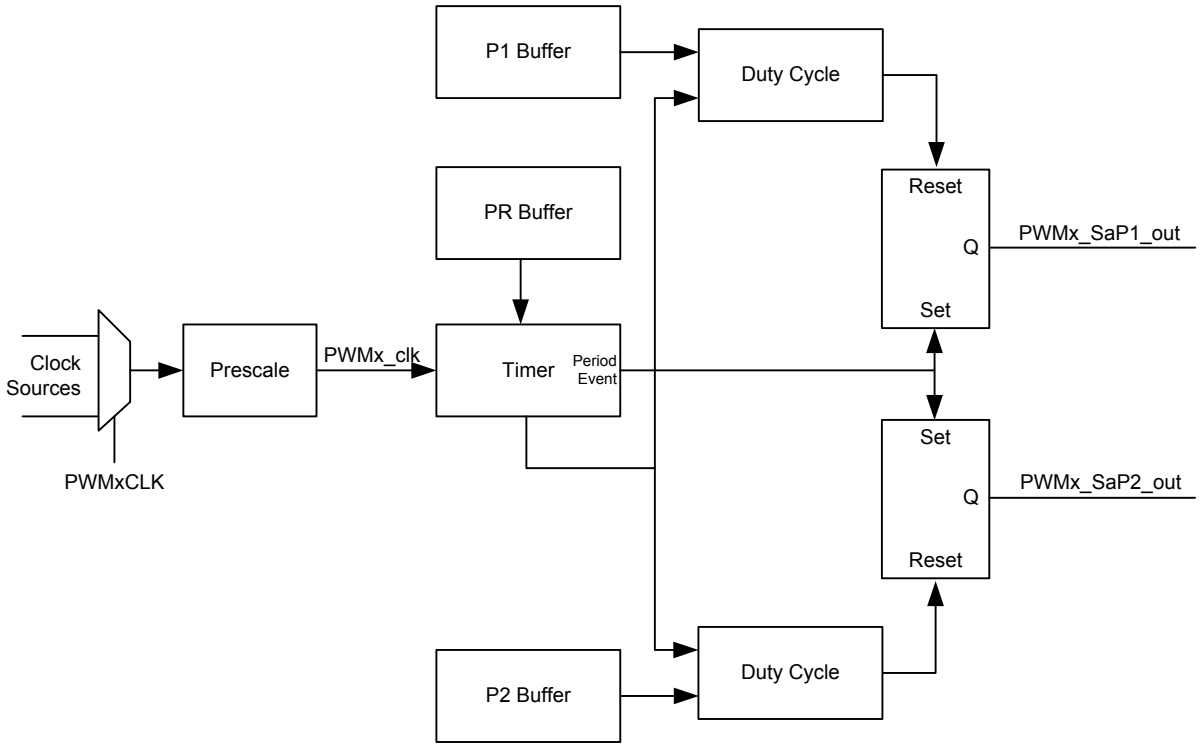
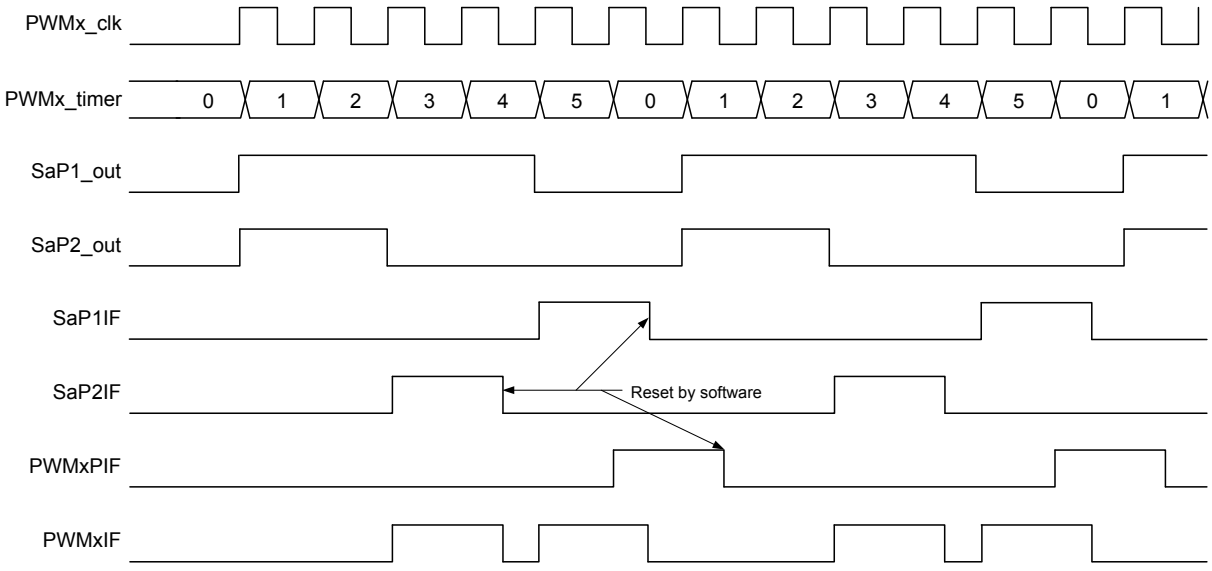



**Figure 30-2. Left-Aligned Timing Diagram**




**Note:** MODE = `'b000`, PR = 5, P1 = 4, P2 = 2.

### 30.1.2.2 Right Aligned Mode

In Right Aligned mode, the active part of the duty cycle is at the end of the period. The outputs start in the Inactive state and then go Active the number of prescaled PWM clock periods specified by the P1 and P2 parameter registers before the end of the period. Block and timing diagrams follow.

**Figure 30-3. Right-Aligned Block Diagram**

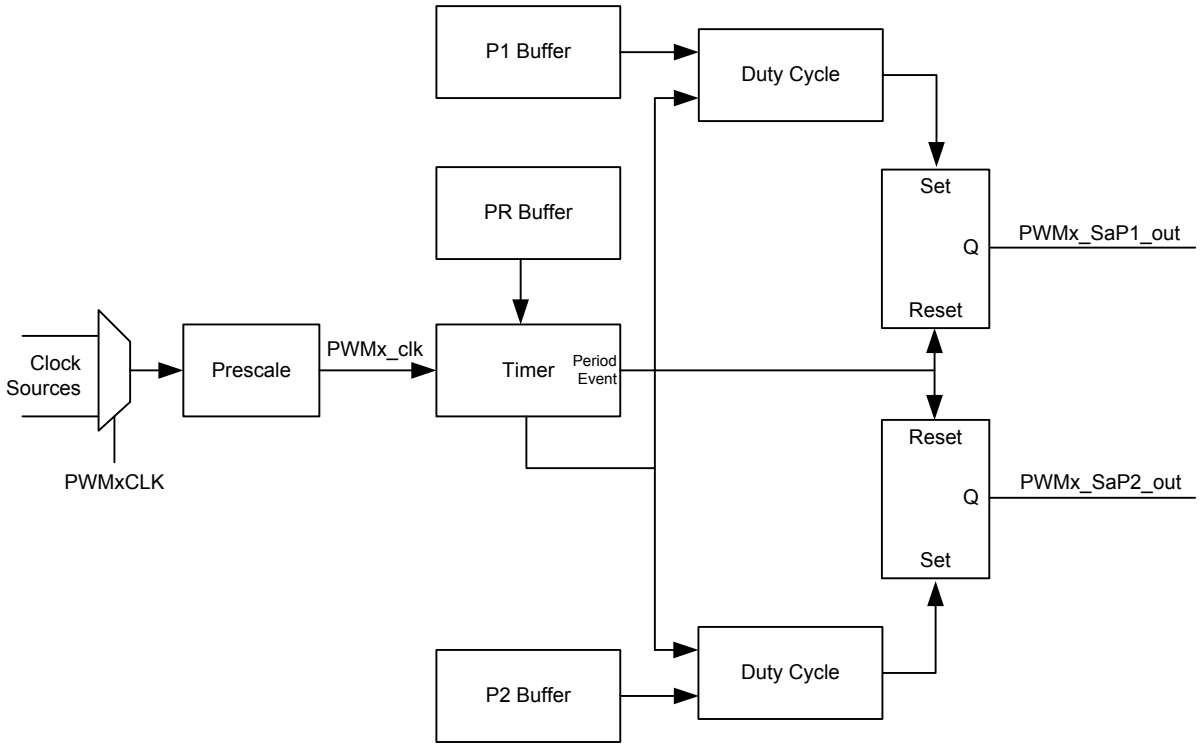
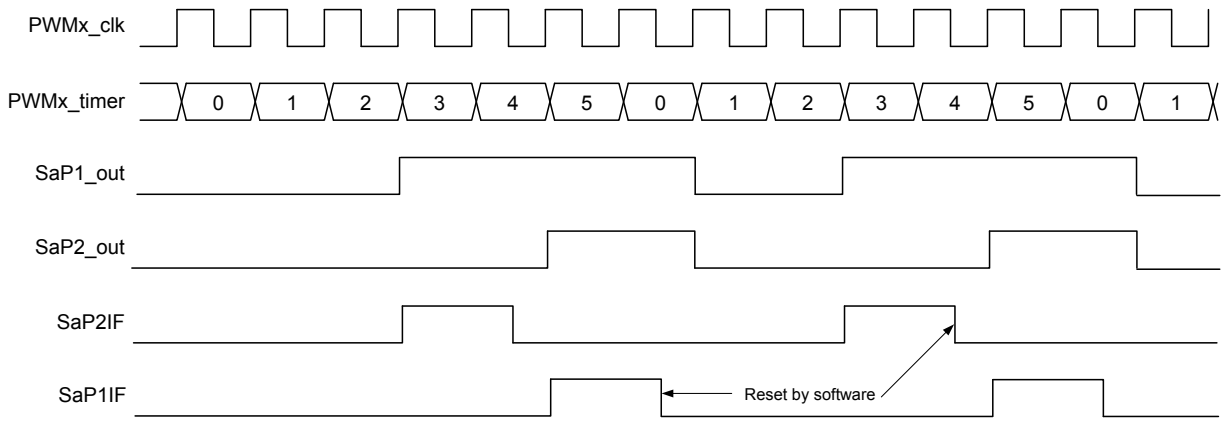



**Figure 30-4. Right-Aligned Timing Diagram**




**Note:** MODE = `'b001`, PR = 5, P1 = 4, P2 = 2.

### 30.1.2.3 Center-Aligned Mode

In Center-Aligned mode, the active duty cycle is centered in the period. The period for this mode is twice that of other modes, as shown in the following equation.

**Equation 30-1. Center-Aligned Period**

$$Period = \frac{(PR + 1) \times 2}{F_{PWMx_clk}}$$

The parameter register specifies the number of PWM clock periods that the output goes Active before the period center. The output goes inactive the same number of prescaled PWM clock periods after the period center. Block and timing diagrams follow.

**Figure 30-5. Center-Aligned Block Diagram**

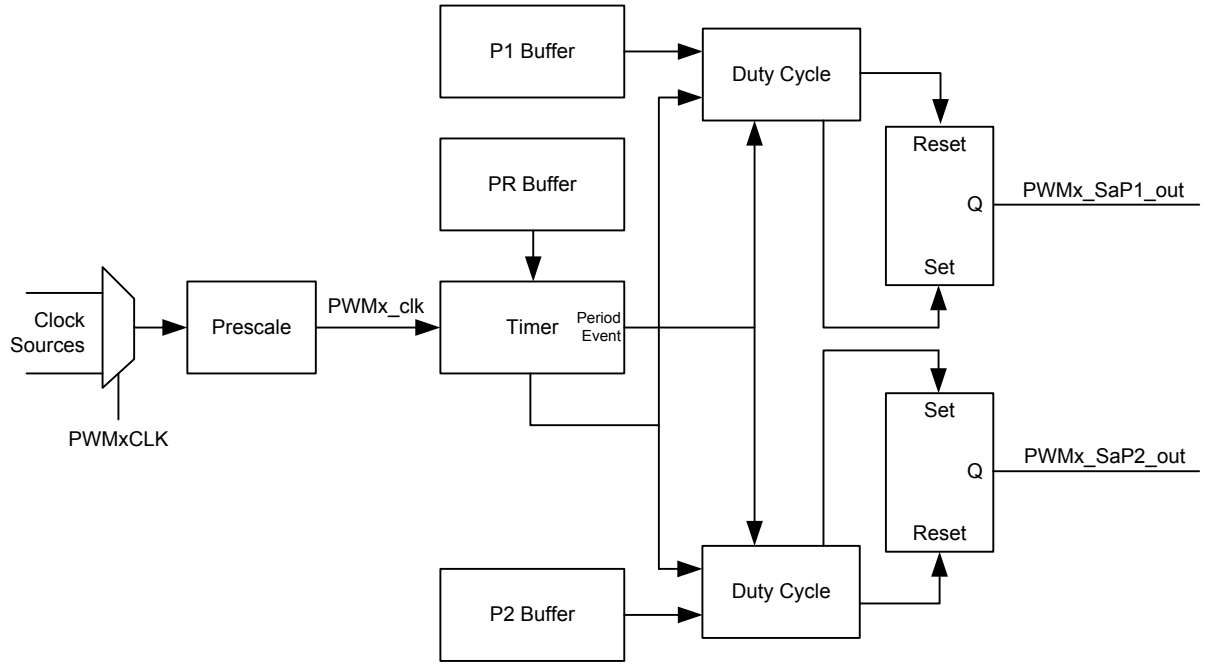
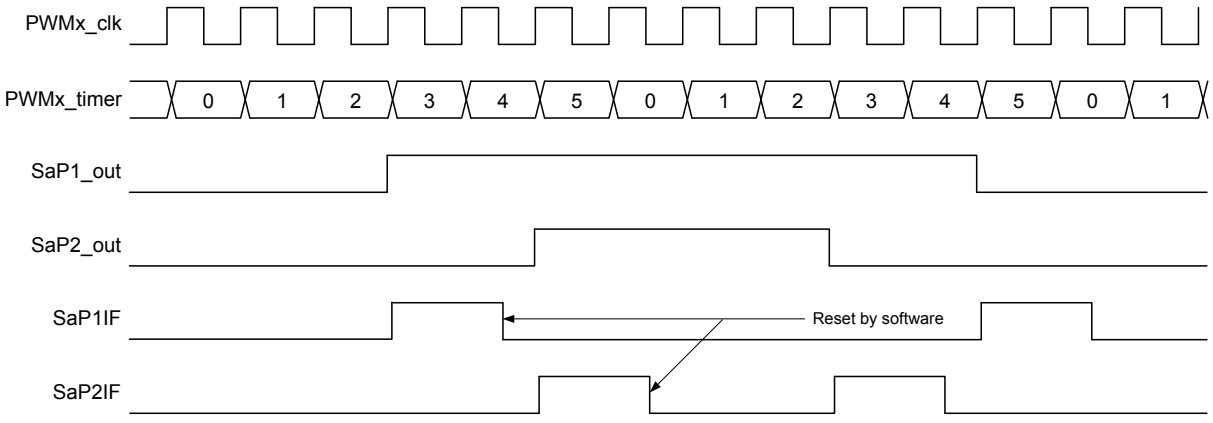



**Figure 30-6. Center-Aligned Timing Diagram**




**Note:** MODE = `'b010`, PR = 5, P1 = 4, P2 = 2.

### 30.1.2.4 Variable Alignment Mode

In Variable Alignment mode, the active part of the duty cycle starts when the parameter 1 value (P1) matches the timer and ends when the parameter 2 value (P2) matches the timer. Both outputs are identical because both parameter values are used for the same duty cycle. Block and timing diagrams follow.

**Figure 30-7. Variable Alignment Block Diagram**

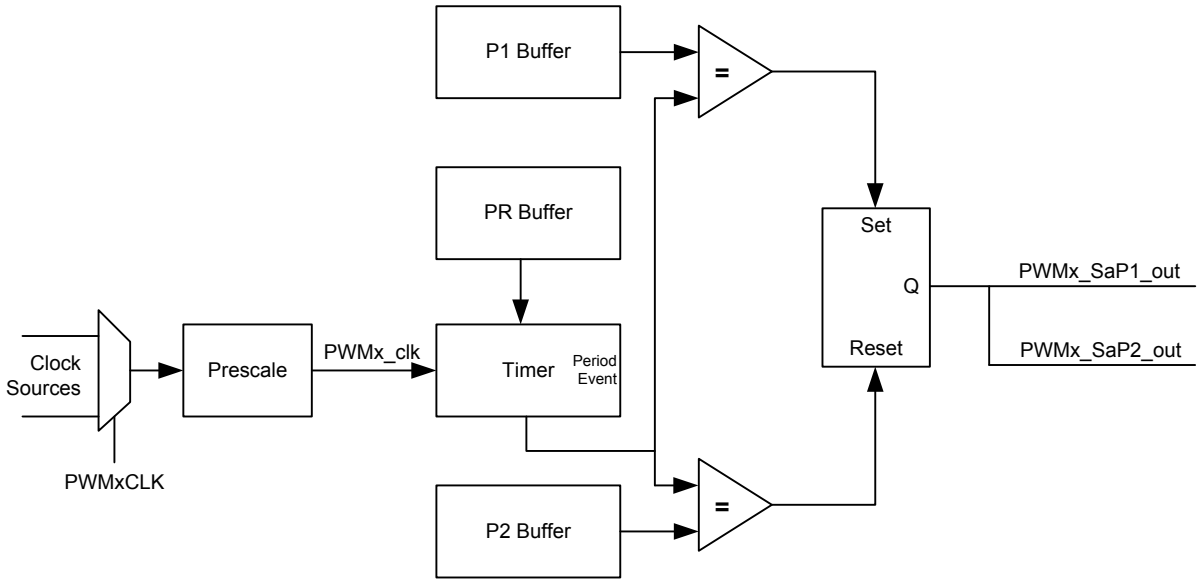
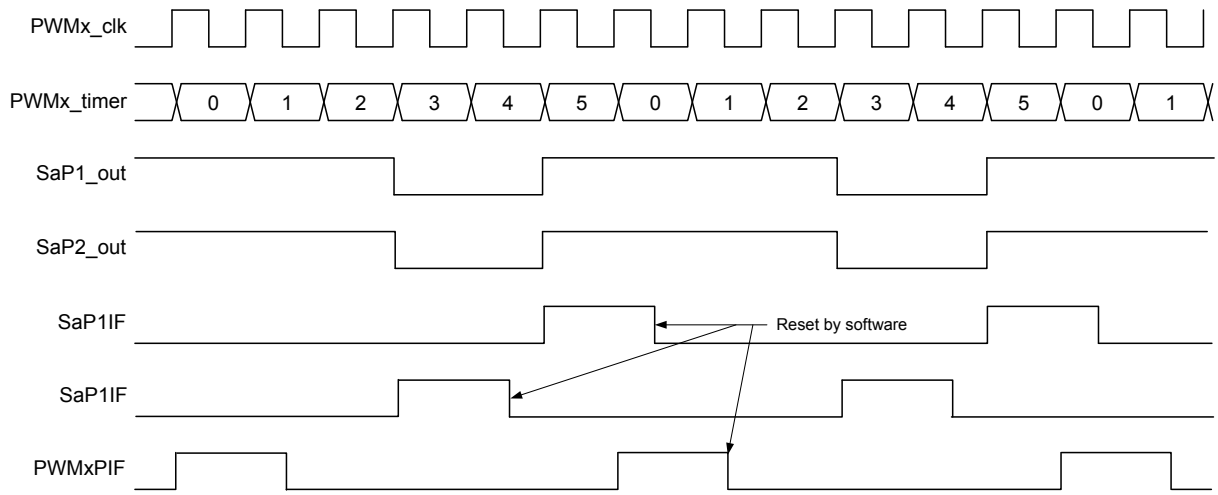



**Figure 30-8. Variable Alignment Timing Diagram**




**Note:** MODE = `'b011`, PR = 5, P1 = 4, P2 = 2.

### 30.1.2.5 Compare Modes

In the Compare modes, the PWM timer is compared to the P1 and P2 parameter values. When a match occurs, the output is either pulsed or toggled. In Pulsed Compare mode, the duty cycle is always one prescaled PWM clock period. In Toggle Compare mode, the duty cycle is always one full PWM period. Refer to the following sections for more details.

#### 30.1.2.5.1 Pulsed Compare Mode

In Pulsed Compare mode, the duty cycle is one prescaled PWM clock period that starts when the timer matches the parameter value and ends one prescaled PWM clock period later. The outputs start in the Inactive state and then go Active during the duty cycle. Block and timing diagrams follow.

**Figure 30-9. Pulsed Compare Block Diagram**

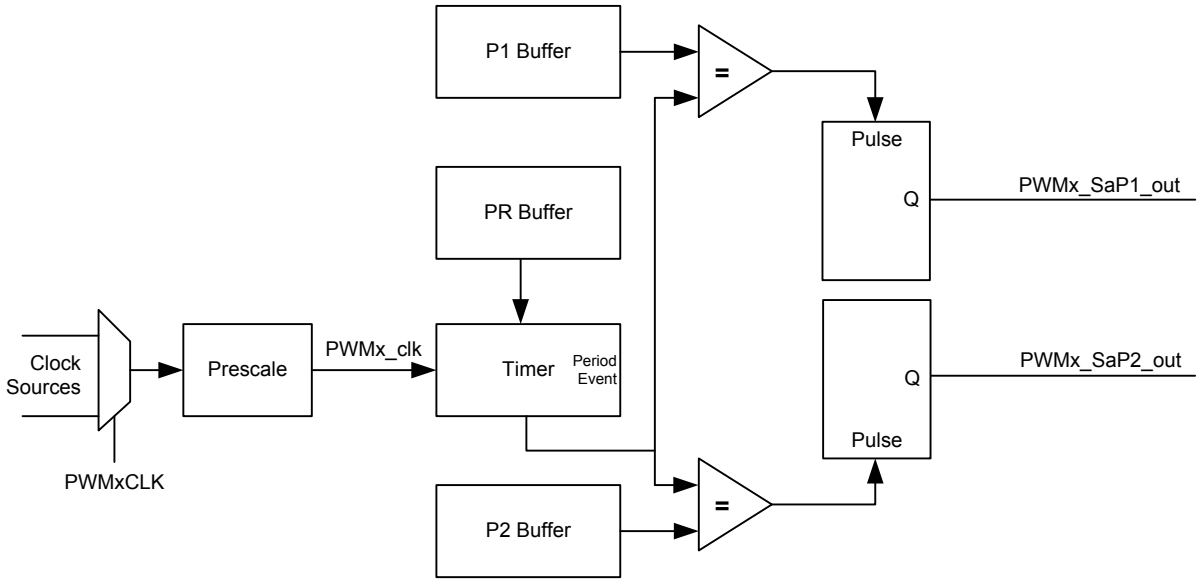
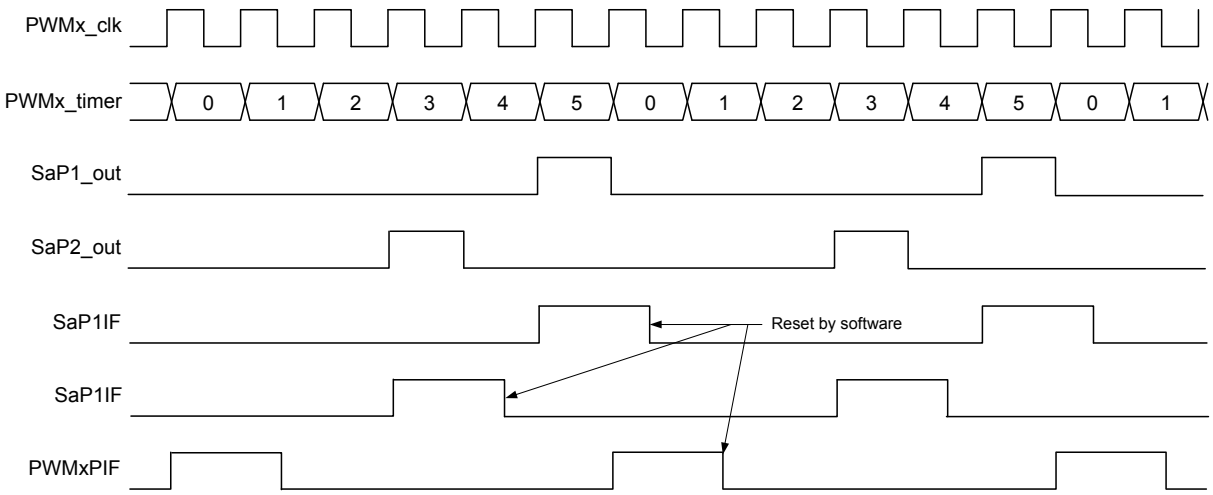



**Figure 30-10. Pulsed Compare Timing Diagram**




**Note:** MODE = `'b100`, PR = 5, P1 = 4, P2 = 2.

#### 30.1.2.5.2 Toggled Compare

In Toggled Compare mode, the duty cycle is alternating full PWM periods. The output goes Active when the PWM timer matches the P1 or P2 parameter value and goes Inactive in the next period at the same match point. Block and timing diagrams follow.

**Figure 30-11. Toggled Compare Block Diagram**

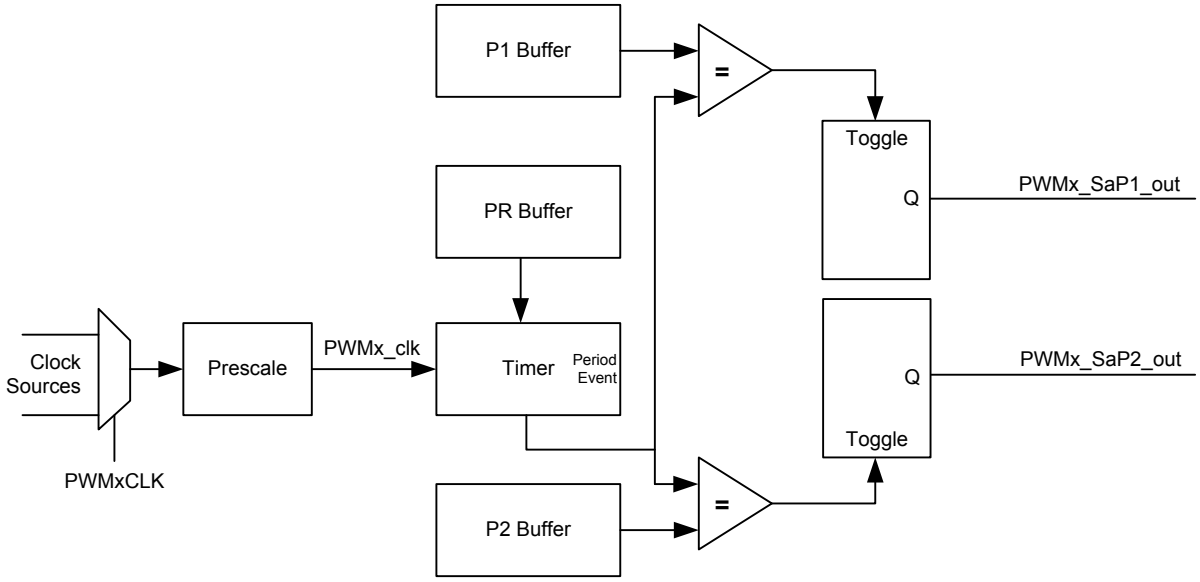
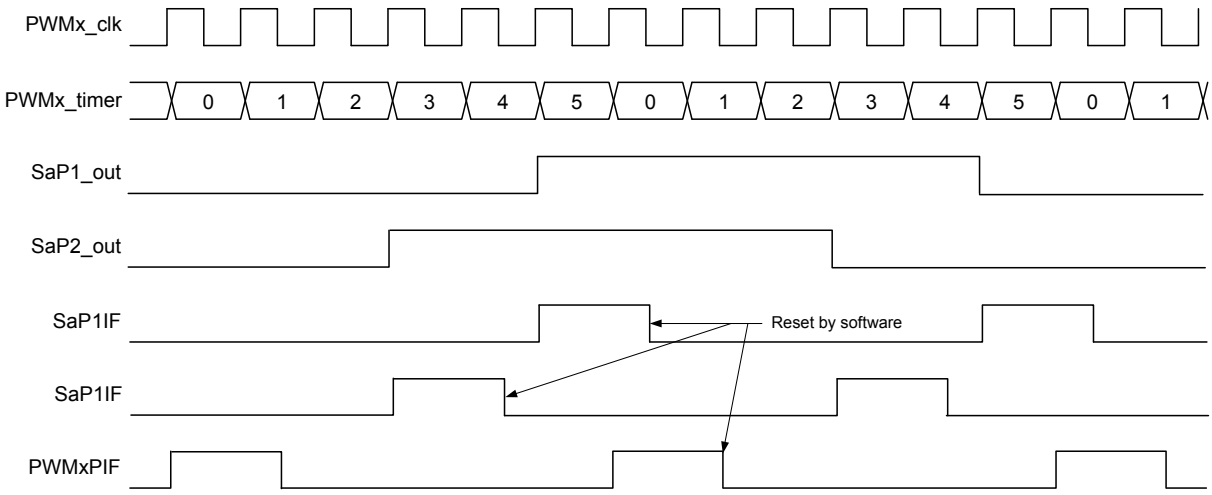



**Figure 30-12. Toggled Compare Timing Diagram**

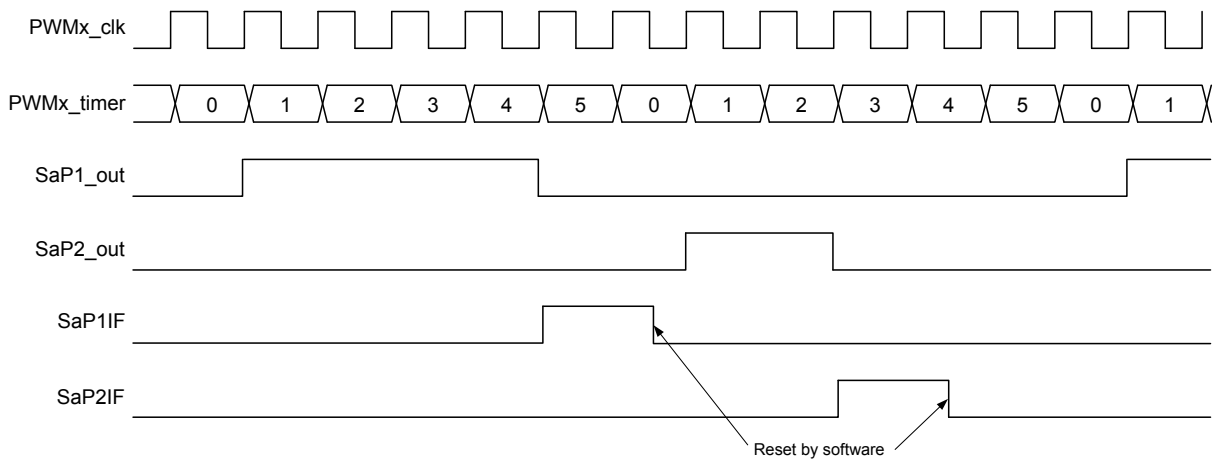



**Note:** MODE = `'b101`, PR = 5, P1 = 4, P2 = 2.

### 30.1.3 Push-Pull Mode

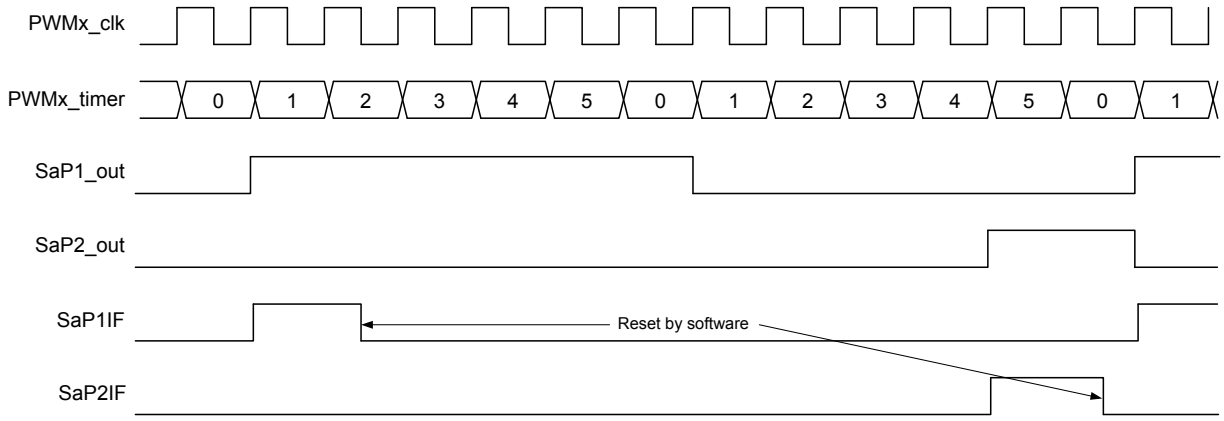
The Push-Pull mode is enabled by setting the PPEN bit. Push-Pull operates only in the Left Aligned and Right Aligned modes. In the Push-Pull mode, the outputs are Active every other PWM period. PWMx_SaP1_out is Active when the PWMx_SaP2_out is not and the PWMx_SaP2_out is Active when the PWMx_SaP1_out is not. When the parameter value (P1 or P2) is greater than the period value (PR), then the corresponding output is Active for one full PWM period. The following figures illustrate timing examples of Left and Right Aligned Push-Pull modes.

**Figure 30-13. Left Aligned Push-Pull Mode Timing Diagram**




**Note:** MODE = `'b000`, PR = 5, P1 = 4, P2 = 2, PPEN = `1`.

**Figure 30-14. Right Aligned Push-Pull Mode Timing Diagram**




**Note:** MODE = `'b001`, PR = 5, P1 = 6, P2 = 2, PPEN = `1`.

## 30.2 Period Timer

All slices in a PWM instance operate with the same period. The value written to the PWMxPR register is one less than the number of prescaled PWM clock periods (PWM_clk) in the PWM period.

The PWMxPR register is double-buffered. When the PWM is operating, writes to the PWMxPR register are transferred to the period buffer only after the LD bit is set or an external load event occurs. The transfer occurs at the next period Reset event. If the LD bit is set less than three PWM clock periods before the end of the period, then the transfer may be one full period later.

Loading the buffers of multiple PWM instances can be coordinated using the PWMLOAD register. See the Buffered Period and Parameter Registers section for more details.

## 30.3 Clock Sources

The time base for the PWM period prescaler is selected with the CLK bits. Changes take effect immediately when written. Clearing the EN bit before making clock source changes is recommended to avoid unexpected behavior.

### 30.3.1 Clock Prescaler

The PWM clock frequency can be reduced with the clock prescaler. There are 256 prescale selections from 1:1 to 1:256.

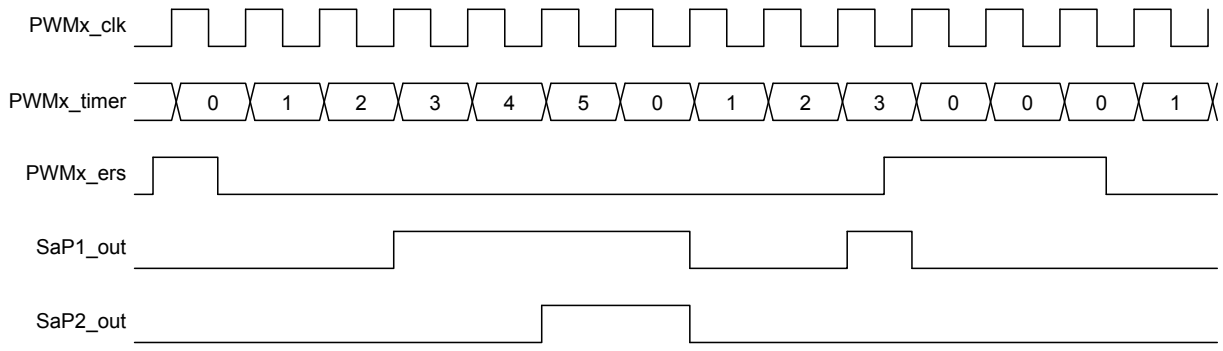
The CPRE bits select the prescale value. Changes to the prescale value take effect immediately. Clearing the EN bit before making prescaler changes is recommended to avoid unexpected behavior. The prescale counter is reset when the EN bit is cleared.

## 30.4 External Period Resets

The period timer can be reset and held at zero by a logic level from one of various sources. The Reset event also resets the postscaler counter. The resetting source is selected with the ERS bits.

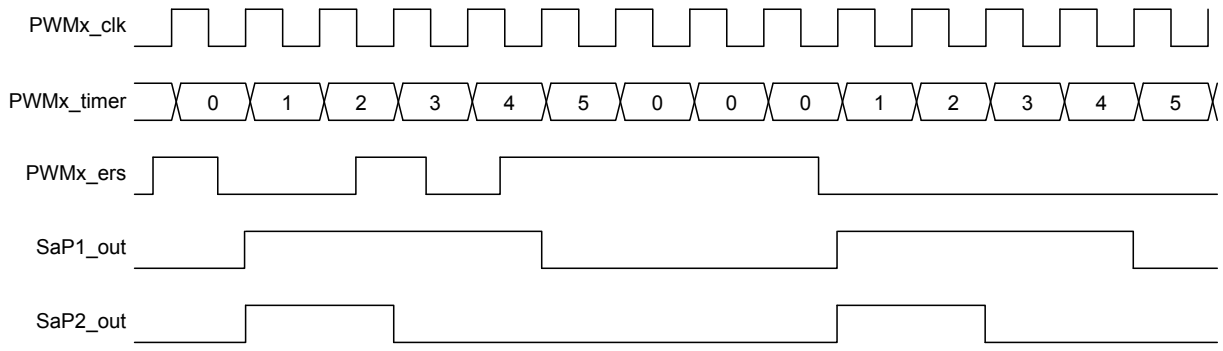
The Reset can be configured with the ERSNOW bit to occur on either the next PWM clock or the next PWM period Reset event. When the ERSNOW bit is set, then the Reset will occur on the next PWM clock. When the ERSNOW bit is cleared, then the Reset will be held off until the timer resets at the end of the period. The difference between a normal period Reset and an ERS Reset is that once the timer is reset, it is held at zero until the ERS signal goes false. The following timing diagrams illustrate the two types of external Reset.

**Figure 30-15. Right Aligned Mode with ERSNOW = 1**




**Note:** PR = 5, P1 = 4, P2 = 2.

**Figure 30-16. Left Aligned Mode with ERSNOW = 0**




**Note:** PR = 5, P1 = 4, P2 = 2.

## 30.5 Buffered Period and Parameter Registers

The PWMxPR, PWMxSaP1 and PWMxSaP2 registers are double-buffered. The PWM module operates on the buffered copies. The values in all these registers are copied to the buffer registers when the PWM module is enabled.

Changes to the PWMxPR, PWMxSaP1 and PWMxSaP2 registers do not affect the buffer registers while the PWM is operating until either software sets the LD bit or an external load event occurs. For all operating modes except Center-Aligned, the values are copied to the buffer registers when the PWM timer is reloaded at the end of the period in which the load request occurred. In the Center-Aligned mode, the buffer update occurs on every other period Reset event because one full center-aligned period uses two period cycles. Load requests occurring three or less clocks before the end of the period may not be serviced until the following period.

A list of external load trigger sources is shown in the PWMxLDS register. Software can set the LD bits of multiple PWM instances simultaneously with the PWMLOAD register.

**Important:** No changes are allowed after the LD bit is set until after the LD bit is cleared by hardware. Unexpected behavior may result if the LD bit is cleared by software.

## 30.6 Synchronizing Multiple PWMs

To synchronize multiple PWMs, the PWMEN register is used to enable selected PWMs simultaneously. The bits in the PWMEN register are mirror copies of the EN bit of every PWM in the device. Setting or clearing the EN bits in the PWMEN register enables or disables all the corresponding PWMs simultaneously.

## 30.7 Interrupts

Each PWM instance has a period interrupt and interrupts associated with the mode and parameter settings.

### 30.7.1 Period Interrupt

The period interrupt occurs when the PWMx timer value matches the PR value, thereby also resetting the PWMx timer. Refer to Figure 30-2 for a timing example. The period interrupt is indicated with the PWMxPIF flag bit in one of the PIR registers and is set whether or not the interrupt is enabled. This flag must be reset by software. The PWMxPIF interrupt is enabled with the PWMxPIE bit in the corresponding PIE register.

#### 30.7.1.1 Period Interrupt Postscaler

The frequency of the period interrupt events can be reduced with the period interrupt postscaler. A postscaler counter suppresses period interrupts until the postscale count is reached. Only one PWM period interrupt is generated for every postscale counts. There are 256 postscale selections from 1:1 to 1:256.

The PIPOS bits select the postscale value. Changes to the postscale value take effect immediately. Clearing the EN bit before making postscaler changes is recommended to avoid unexpected behavior. The postscale counter is reset when the EN bit is cleared.

### 30.7.2 Parameter Interrupts

The P1 and P2 parameters in each slice have interrupts that occur depending on the selected mode. The individual parameter interrupts are indicated in the PWMxGIR register and enabled by the corresponding bits in the PWMxGIE register.

A timing example is shown in Figure 30-2. Refer to the timing diagrams of each of the other modes for more details.

All the enabled PWMxGIR interrupts of one PMW instance are OR'd together into the PWMxIF bit in one of the PIR registers. The PWMxIF bit is read-only. When any of the PWMxGIR bits are set then the PWMxIF bit is true. All PWMxGIF flags must be reset to clear the PWMxIF bit. The PWMxIF interrupt is enabled with the PWMxIE bit in the corresponding PIE register.

© 2019-2023 Microchip Technology Inc. and its subsidiaries

## 30.8 Operation During Sleep

The PWM module operates in Sleep only if the PWM clock is Active. Some internal clock sources are automatically enabled to operate in Sleep when a peripheral using them is enabled. Those clock sources are identified in the clock source table shown in the PWMxCLK clock source selection register.

## 30.9 Register Definitions: PWM Control

Long bit name prefixes for the PWM peripherals are shown in the table below. Refer to the **"Long Bit Names"** section in the **"Register and Bit Naming Conventions"** chapter for more information.

**Table 30-1. PWM Bit Name Prefixes**

| Peripheral | Bit Name Prefix |
|---|---|
| PWM1 | PWM1 |
| PWM2 | PWM2 |
| PWM3 | PWM3 |

### 30.9.1 PWMxERS

**Name:** PWMxERS
**Address:** 0x460,0x46F,0x47E

PWMx External Reset Source

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | ERS[4:0] | | | | |
| Access | | | | R/W | R/W | R/W | R/W | R/W |
| Reset | | | | 0 | 0 | 0 | 0 | 0 |

**Bits 4:0 – ERS[4:0]** External Reset Source Select

| ERS | Reset Source | | |
|---|---|---|---|
| | PWM1 | PWM2 | PWM3 |
| 11111-10010 | Reserved (ERS Disabled) | | |
| 10001 | CLC8_OUT | | |
| 10000 | CLC7_OUT | | |
| 01111 | CLC6_OUT | | |
| 01110 | CLC5_OUT | | |
| 01101 | CLC4_OUT | | |
| 01100 | CLC3_OUT | | |
| 01011 | CLC2_OUT | | |
| 01010 | CLC1_OUT | | |
| 01001 | Reserved (ERS Disabled) | | |
| 01000 | Reserved (ERS Disabled) | | |
| 00111 | PWM3S1P2_OUT | PWM3S1P2_OUT | Reserved |
| 00110 | PWM3S1P1_OUT | PWM3S1P1_OUT | Reserved |
| 00101 | PWM2S1P2_OUT | Reserved | PWM2S1P2_OUT |
| 00100 | PWM2S1P1_OUT | Reserved | PWM2S1P1_OUT |
| 00011 | Reserved | PWM1S1P2_OUT | PWM1S1P2_OUT |
| 00010 | Reserved | PWM1S1P1_OUT | PWM1S1P1_OUT |
| 00001 | PWM1ERSPPS | PWM2ERSPPS | PWM3ERSPPS |
| 00000 | ERS Disabled | | |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

### 30.9.2 PWMxCLK

**Name:** PWMxCLK
**Address:** 0x461,0x470,0x47F

PWMx Clock Source

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | CLK[4:0] | | | | |
| Access | | | | R/W | R/W | R/W | R/W | R/W |
| Reset | | | | 0 | 0 | 0 | 0 | 0 |

**Bits 4:0 – CLK[4:0]** PWM Clock Source Select

| CLK | Source | Operates in sleep |
|---|---|---|
| 11111-10110 | Reserved | N/A |
| 10100 | CLC8_OUT | Yes(1) |
| 10011 | CLC7_OUT | Yes(1) |
| 10010 | CLC6_OUT | Yes(1) |
| 10001 | CLC5_OUT | Yes(1) |
| 10000 | CLC4_OUT | Yes(1) |
| 01111 | CLC3_OUT | Yes(1) |
| 01110 | CLC2_OUT | Yes(1) |
| 01101 | CLC1_OUT | Yes(1) |
| 01100 | NCO3_OUT | Yes(1) |
| 01011 | NCO2_OUT | Yes(1) |
| 01010 | NCO1_OUT | Yes(1) |
| 01001 | CLKREF | Yes(1) |
| 01000 | EXTOSC | Yes |
| 00111 | SOSC | Yes |
| 00110 | MFINTOSC (31.25 kHz) | Yes |
| 00101 | MFINTOSC (500 kHz) | Yes |
| 00100 | LFINTOSC | Yes |
| 00011 | HFINTOSC | Yes |
| 00010 | $F_{OSC}$ | No |
| 00001 | PWMIN1PPS | Yes(1) |
| 00000 | PWMIN0PPS | Yes(1) |

**Note:** Operation during Sleep is possible if the clock supplying the source peripheral operates in Sleep.

### 30.9.3 PWMxLDS

**Name:** PWMxLDS
**Address:** 0x462,0x471,0x480

PWMx Auto-load Trigger Source Select Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | LDS[4:0] | | | | |
| Access | | | | R/W | R/W | R/W | R/W | R/W |
| Reset | | | | 0 | 0 | 0 | 0 | 0 |

**Bits 4:0 – LDS[4:0]** Auto-load Trigger Source Select

| LDS | Source |
|---|---|
| 11111-10011 | Auto-load Disabled |
| 10010 | Reserved |
| 10001 | Reserved |
| 10000 | DMA6_Destination_Count_Done |
| 01111 | DMA5_Destination_Count_Done |
| 01110 | DMA4_Destination_Count_Done |
| 01101 | DMA3_Destination_Count_Done |
| 01100 | DMA2_Destination_Count_Done |
| 01011 | DMA1_Destination_Count_Done |
| 01010 | CLC8_OUT |
| 01001 | CLC7_OUT |
| 01000 | CLC6_OUT |
| 00111 | CLC5_OUT |
| 00110 | CLC4_OUT |
| 00101 | CLC3_OUT |
| 00100 | CLC2_OUT |
| 00011 | CLC1_OUT |
| 00010 | PWMIN1PPS |
| 00001 | PWMIN0PPS |
| 00000 | Auto-load Disabled |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

### 30.9.4 PWMxPR

**Name:** PWMxPR
**Address:** 0x463,0x472,0x481

PWMx Period Register

Determines the PWMx period

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | PR[15:8] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | PR[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 15:0 – PR[15:0]** PWM Period
Number of PWM clocks periods in the PWM period

**Notes:** The individual bytes in this multibyte register can be accessed with the following register names:

- PWMxPRH: Accesses the high byte PR[15:8]
- PWMxPRL: Accesses the low byte PR[7:0]

### 30.9.5 PWMxCPRE

**Name:** PWMxCPRE
**Address:** 0x465,0x474,0x483

PWMx Clock Prescaler Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | CPRE[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 7:0 – CPRE[7:0]** PWM Clock Prescale Value

| Value | Description |
|---|---|
| n | PWM clock is prescaled by n+1 |

### 30.9.6 PWMxPIPOS

**Name:** PWMxPIPOS
**Address:** 0x466,0x475,0x484

PWMx Period Interrupt Postscaler Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | PIPOS[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 7:0 – PIPOS[7:0]** Period Interrupt Postscale Value

| Value | Description |
|---|---|
| n | Period interrupt occurs after n+1 period events |

### 30.9.7 PWMxGIR

**Name:** PWMxGIR
**Address:** 0x467,0x476,0x485

PWMx Interrupt Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | S1P2 | S1P1 |
| Access | | | | | | | R/W/HS | R/W/HS |
| Reset | | | | | | | 0 | 0 |

**Bit 1 – SaP2** Slice “a” Parameter 2 Interrupt Flag

| Value | Mode | Description |
|---|---|---|
| 1 | Variable Aligned or Compare | Compare match between P2 and PWM counter has occurred |
| 1 | Center-Aligned | PWMx_SaP2_out has changed |
| 1 | Right Aligned | Left edge of PWMx_SaP2_out pulse has occurred |
| 1 | Left Aligned | Right edge of PWMx_SaP2_out pulse has occurred |
| 0 | All | Interrupt event has not occurred |

**Bit 0 – SaP1** Slice “a” Parameter 1 Interrupt Flag

| Value | Mode | Description |
|---|---|---|
| 1 | Variable Aligned or Compare | Compare match between P1 and PWM counter has occurred |
| 1 | Center-Aligned | PWMx_SaP1_out has changed |
| 1 | Right Aligned | Left edge of PWMx_SaP1_out pulse has occurred |
| 1 | Left Aligned | Right edge of PWMx_SaP1_out pulse has occurred |
| 0 | All | Interrupt event has not occurred |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

### 30.9.8 PWMxGIE

**Name:** PWMxGIE
**Address:** 0x468,0x477,0x486

PWMx Interrupt Enable Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | S1P2 | S1P1 |
| Access | | | | | | | R/W | R/W |
| Reset | | | | | | | 0 | 0 |

**Bit 1 – SaP2** Slice “a” Parameter 2 Interrupt Enable

| Value | Description |
|---|---|
| 1 | Slice “a” Parameter 2 match interrupt is enabled |
| 0 | Slice “a” Parameter 2 match interrupt is not enabled |

**Bit 0 – SaP1** Slice “a” Parameter 1 Interrupt Enable

| Value | Description |
|---|---|
| 1 | Slice “a” Parameter 1 match interrupt is enabled |
| 0 | Slice “a” Parameter 1 match interrupt is not enabled |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

### 30.9.9 PWMxCON

**Name:** PWMxCON
**Address:** 0x469,0x478,0x487

PWM Control Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | EN | | | | | LD | ERSPOL | ERSNOW |
| Access | R/W | | | | | R/W/HC | R/W | R/W |
| Reset | 0 | | | | | 0 | 0 | 0 |

**Bit 7 – EN** PWM Module Enable

| Value | Description |
|---|---|
| 1 | PWM module is enabled |
| 0 | PWM module is disabled. The prescaler, postscaler, and all internal logic is reset. Outputs go to their default states. Register values remain unchanged. |

**Bit 2 – LD** Reload Registers

Reload the period and duty cycle registers on the next period event

| Value | Description |
|---|---|
| 1 | Reload PR/P1/P2 registers |
| 0 | Reload not enabled or reload complete |

**Bit 1 – ERSPOL** External Reset Polarity Select

| Value | Description |
|---|---|
| 1 | External Reset input is active-low |
| 0 | External Reset input is active-high |

**Bit 0 – ERSNOW** External Reset Mode Select

Determines when an external Reset event takes effect.

| Value | Description |
|---|---|
| 1 | Stop counter on the next PWM clock. Output goes to the Inactive state. |
| 0 | Stop counter at the end of the period. Output goes to the Inactive state. |

### 30.9.10 PWMxSaCFG

**Name:** PWMxSaCFG

PWM Slice “a” Configuration Register(1)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | POL2 | POL1 | | | PPEN | MODE[2:0] | | |
| Access | R/W | R/W | | | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | | | 0 | 0 | 0 | 0 |

**Bit 7 – POL2** PWM Slice “a” Parameter 2 Output Polarity

| Value | Description |
|---|---|
| 1 | PWMx_SaP2_out is low true |
| 0 | PWMx_SaP2_out is high true |

**Bit 6 – POL1** PWM Slice “a” Parameter 1 Output Polarity

| Value | Description |
|---|---|
| 1 | PWMx_SaP1_out is low true |
| 0 | PWMx_SaP1_out is high true |

**Bit 3 – PPEN** Push-Pull Mode Enable

Each period the output alternates between PWMx_SaP1_out and PWMx_SaP2_out. Only Left and Right Aligned modes are supported. Other modes may exhibit unexpected results.

| Value | Description |
|---|---|
| 1 | PWMx Slice “a” Push-Pull mode is enabled |
| 0 | PWMx Slice “a” Push-Pull mode is not enabled |

**Bits 2:0 – MODE[2:0]** PWM Module Slice “a” Operating Mode Select

Selects operating mode for both PWMx_SaP1_out and PWMx_SaP2_out

| Value | Description |
|---|---|
| 11x | Reserved. Outputs go to Reset state. |
| 101 | Compare mode: Toggle PWMx_SaP1_out and PWMx_SaP2_out on PWM timer match with corresponding parameter register |
| 100 | Compare mode: Set PWMx_SaP1_out and PWMx_SaP2_out high on PWM timer match with corresponding parameter register |
| 011 | Variable Aligned mode |
| 010 | Center-Aligned mode |
| 001 | Right Aligned mode |
| 000 | Left Aligned mode |

**Note:**

1. Changes to this register must be done only when the EN bit is cleared.

© 2019-2023 Microchip Technology Inc. and its subsidiaries

### 30.9.11 PWMxSaP1

**Name:** PWMxSaP1

PWM Slice “a” Parameter 1 Register

Determines the active period of slice “a”, parameter 1 output

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | P1[15:8] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | P1[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 15:0 – P1[15:0]** Parameter 1 Value

| Value | Mode | Description |
|---|---|---|
| n | Compare | Compare match event occurs when PWMx timer = n (Refer to MODE selections) |
| n | Variable Aligned | PWMx_SaP1_out and PWMx_SaP2 both go high when PWMx timer = n |
| n | Center-Aligned | PWMx_SaP1_out is high 2*n PWMx clock periods centered around PWMx period event |
| n | Right Aligned | PWMx_SaP1_out is high n PWMx clock periods at end of PWMx period |
| n | Left Aligned | PWMx_SaP1_out is high n PWMx clock periods at beginning of PWMx period |

**Notes:** The individual bytes in this multibyte register can be accessed with the following register names:

- PWMxSaP1H: Accesses the high byte P1[15:8]
- PWMxSaP1L: Accesses the low byte P1[7:0]

### 30.9.12 PWMxSaP2

**Name:** PWMxSaP2

PWM Slice “a” Parameter 2 Register

Determines the active period of slice “a”, parameter 2 output

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | P2[15:8] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | P2[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 15:0 – P2[15:0]** Parameter 2 Value

| Value | Mode | Description |
|---|---|---|
| n | Compare | Compare match event occurs when PWMx timer = n (Refer to MODE selections) |
| n | Variable Aligned | PWMx_SaP1_out and PWMx_SaP2 both go low when PWMx timer = n |
| n | Center-Aligned | PWMx_SaP2_out is high 2*n PWMx clock periods centered around PWMx period event |
| n | Right Aligned | PWMx_SaP2_out is high n PWMx clock periods at end of PWMx period |
| n | Left Aligned | PWMx_SaP2_out is high n PWMx clock periods at beginning of PWMx period |

**Notes:** The individual bytes in this multibyte register can be accessed with the following register names:

- PWMxSaP2H: Accesses the high byte P2[15:8]
- PWMxSaP2L: Accesses the low byte P2[7:0]

### 30.9.13 PWMLOAD

**Name:** PWMLOAD
**Address:** 0x49C

Mirror copies of all PWMxLD bits

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | MPWM3LD | MPWM2LD | MPWM1LD |
| Access | | | | | | R/W | R/W | R/W |
| Reset | | | | | | 0 | 0 | 0 |

**Bits 0, 1, 2 – MPWMxLD** Mirror copy of PWMxLD bit

Mirror copies of all PWMxLD bits can be set simultaneously to synchronize the load event across all PWMs

| Value | Description |
|---|---|
| 1 | PWMx parameter and period values will be transferred to their buffer registers at the next period Reset event |
| 0 | There are no PWMx period and parameter value transfers pending |

### 30.9.14 PWMEN

**Name:** PWMEN
**Address:** 0x49D

Mirror copies of all PWMxEN bits

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | MPWM3EN | MPWM2EN | MPWM1EN |
| Access | | | | | | R/W | R/W | R/W |
| Reset | | | | | | 0 | 0 | 0 |

**Bits 0, 1, 2 – MPWMxEN** Mirror copy of PWMxEN bit

Mirror copies of all PWMxEN bits can be set simultaneously to synchronize the enable event across all PWMs

| Value | Description |
|---|---|
| 1 | PWMx is enabled |
| 0 | PWMx is not enabled |

## 30.10 Register Summary - PWM

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0460 | PWM1ERS | 7:0 | | | | | | ERS[4:0] | | |
| 0x0461 | PWM1CLK | 7:0 | | | | | | CLK[4:0] | | |
| 0x0462 | PWM1LDS | 7:0 | | | | | | LDS[4:0] | | |
| 0x0463 | PWM1PR | 7:0 | PR[7:0] | | | | | | | |
| | | 15:8 | PR[15:8] | | | | | | | |
| 0x0465 | PWM1CPRE | 7:0 | CPRE[7:0] | | | | | | | |
| 0x0466 | PWM1PIPOS | 7:0 | PIPOS[7:0] | | | | | | | |
| 0x0467 | PWM1GIR | 7:0 | | | | | | | S1P2 | S1P1 |
| 0x0468 | PWM1GIE | 7:0 | | | | | | | S1P2 | S1P1 |
| 0x0469 | PWM1CON | 7:0 | EN | | | | | LD | ERSPOL | ERSNOW |
| 0x046A | PWM1S1CFG | 7:0 | POL2 | POL1 | | | PPEN | | MODE[2:0] | |
| 0x046B | PWM1S1P1 | 7:0 | P1[7:0] | | | | | | | |
| | | 15:8 | P1[15:8] | | | | | | | |
| 0x046D | PWM1S1P2 | 7:0 | P2[7:0] | | | | | | | |
| | | 15:8 | P2[15:8] | | | | | | | |
| 0x046F | PWM2ERS | 7:0 | | | | | | ERS[4:0] | | |
| 0x0470 | PWM2CLK | 7:0 | | | | | | CLK[4:0] | | |
| 0x0471 | PWM2LDS | 7:0 | | | | | | LDS[4:0] | | |
| 0x0472 | PWM2PR | 7:0 | PR[7:0] | | | | | | | |
| | | 15:8 | PR[15:8] | | | | | | | |
| 0x0474 | PWM2CPRE | 7:0 | CPRE[7:0] | | | | | | | |
| 0x0475 | PWM2PIPOS | 7:0 | PIPOS[7:0] | | | | | | | |
| 0x0476 | PWM2GIR | 7:0 | | | | | | | S1P2 | S1P1 |
| 0x0477 | PWM2GIE | 7:0 | | | | | | | S1P2 | S1P1 |
| 0x0478 | PWM2CON | 7:0 | EN | | | | | LD | ERSPOL | ERSNOW |
| 0x0479 | PWM2S1CFG | 7:0 | POL2 | POL1 | | | PPEN | | MODE[2:0] | |
| 0x047A | PWM2S1P1 | 7:0 | P1[7:0] | | | | | | | |
| | | 15:8 | P1[15:8] | | | | | | | |
| 0x047C | PWM2S1P2 | 7:0 | P2[7:0] | | | | | | | |
| | | 15:8 | P2[15:8] | | | | | | | |
| 0x047E | PWM3ERS | 7:0 | | | | | | ERS[4:0] | | |
| 0x047F | PWM3CLK | 7:0 | | | | | | CLK[4:0] | | |
| 0x0480 | PWM3LDS | 7:0 | | | | | | LDS[4:0] | | |
| 0x0481 | PWM3PR | 7:0 | PR[7:0] | | | | | | | |
| | | 15:8 | PR[15:8] | | | | | | | |
| 0x0483 | PWM3CPRE | 7:0 | CPRE[7:0] | | | | | | | |
| 0x0484 | PWM3PIPOS | 7:0 | PIPOS[7:0] | | | | | | | |
| 0x0485 | PWM3GIR | 7:0 | | | | | | | S1P2 | S1P1 |
| 0x0486 | PWM3GIE | 7:0 | | | | | | | S1P2 | S1P1 |
| 0x0487 | PWM3CON | 7:0 | EN | | | | | LD | ERSPOL | ERSNOW |
| 0x0488 | PWM3S1CFG | 7:0 | POL2 | POL1 | | | PPEN | | MODE[2:0] | |
| 0x0489 | PWM3S1P1 | 7:0 | P1[7:0] | | | | | | | |
| | | 15:8 | P1[15:8] | | | | | | | |
| 0x048B | PWM3S1P2 | 7:0 | P2[7:0] | | | | | | | |
| | | 15:8 | P2[15:8] | | | | | | | |
| 0x048D ... 0x049B | Reserved | | | | | | | | | |
| 0x049C | PWMLOAD | 7:0 | | | | | | MPWM3LD | MPWM2LD | MPWM1LD |
| 0x049D | PWMEN | 7:0 | | | | | | MPWM3EN | MPWM2EN | MPWM1EN |

# 31. CWG - Complementary Waveform Generator Module

The Complementary Waveform Generator (CWG) produces half-bridge, full-bridge, and steering of PWM waveforms. It is backwards compatible with previous CCP functions.

The CWG has the following features:

- Six Operating modes:
  - Synchronous Steering mode
  - Asynchronous Steering mode
  - Full Bridge mode, Forward
  - Full Bridge mode, Reverse
  - Half Bridge mode
  - Push-Pull mode
- Output Polarity Control
- Output Steering
- Independent 6-bit Rising and Falling Event Dead-Band Timers:
  - Clocked dead band
  - Independent rising and falling dead-band enables
- Auto-Shutdown Control with:
  - Selectable shutdown sources
  - Auto-restart option
  - Auto-shutdown pin override control

## 31.1 Fundamental Operation

The CWG generates two output waveforms from the selected input source.

The off-to-on transition of each output can be delayed from the on-to-off transition of the other output, thereby creating a time delay immediately where neither output is driven. This is referred to as dead time and is covered in the Dead-Band Control section.

It may be necessary to guard against the possibility of circuit faults or a feedback event arriving too late or not at all. In this case, the active drive must be terminated before the Fault condition causes damage. This is referred to as auto-shutdown and is covered in the Auto-Shutdown section.

## 31.2 Operating Modes

The CWG module can operate in six different modes, as specified by the MODE bits:

- Half Bridge mode
- Push-Pull mode
- Asynchronous Steering mode
- Synchronous Steering mode
- Full Bridge mode, Forward
- Full Bridge mode, Reverse

All modes accept a single pulse input, and provide up to four outputs as described in the following sections.

All modes include auto-shutdown control as described in the Auto-Shutdown section.

**Important:** Except as noted for Full Bridge mode, mode changes must only be performed while EN = 0.

### 31.2.1 Half Bridge Mode

In Half Bridge mode, two output signals are generated as true and inverted versions of the input as illustrated in Figure 31-1. A nonoverlap (dead band) time is inserted between the two outputs to prevent shoot-through current in various power supply applications. Dead-band control is described in the Dead-Band Control section. The output steering feature cannot be used in this mode. A basic block diagram of this mode is shown in Figure 31-2.

The unused outputs CWGxC and CWGxD drive similar signals as CWGxA and CWGxB, with polarity independently controlled by the POLC and POLD bits, respectively.

**Figure 31-1. CWG Half Bridge Mode Operation**

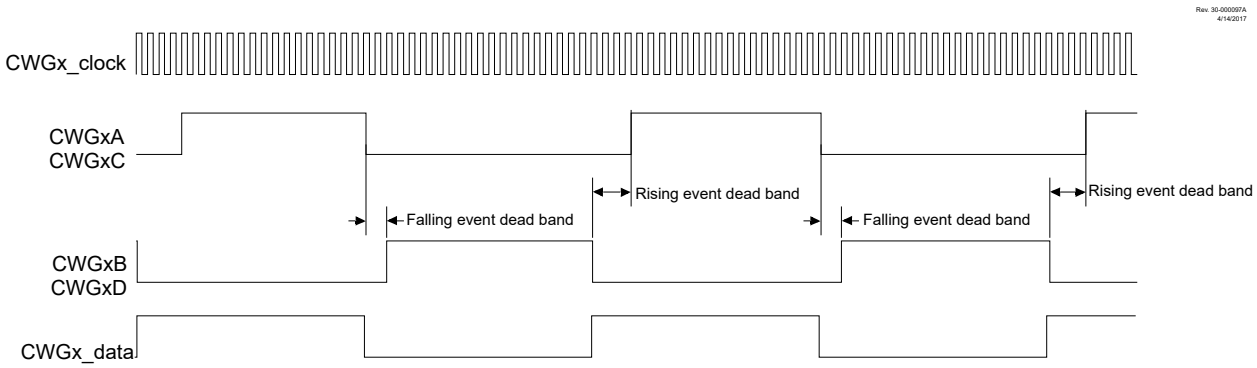
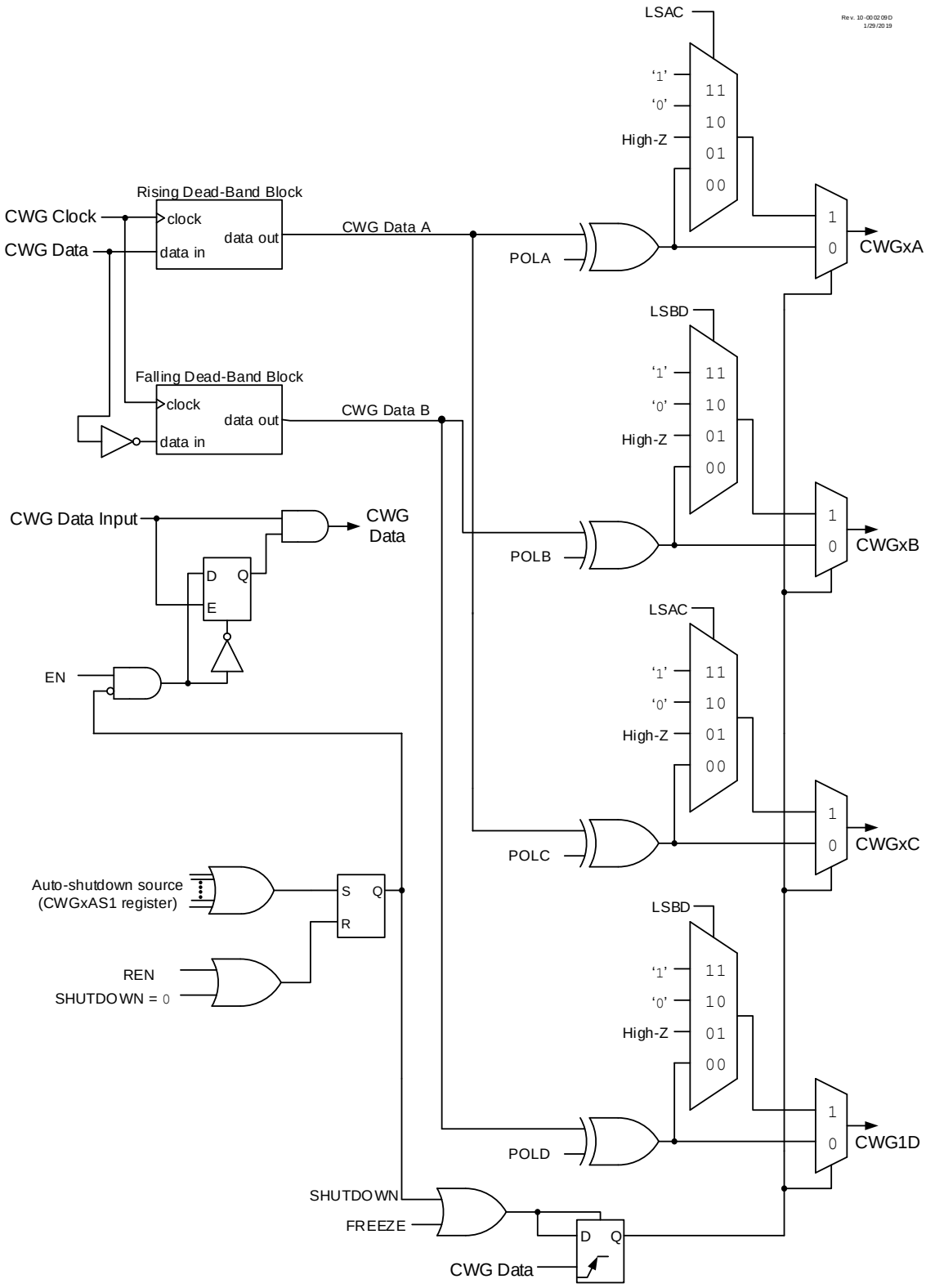


**Figure 31-2. Simplified CWG Block Diagram (Half Bridge Mode, MODE = 'b100)**



### 31.2.2 Push-Pull Mode

In Push-Pull mode, two output signals are generated, alternating copies of the input as illustrated in Figure 31-3. This alternation creates the Push-Pull effect required for driving some transformer-based power supply designs. Steering modes are not used in Push-Pull mode. A basic block diagram for the Push-Pull mode is shown in Figure 31-4.

The Push-Pull sequencer is reset whenever EN = 0 or if an auto-shutdown event occurs. The sequencer is clocked by the first input pulse, and the first output appears on CWGxA.

The unused outputs CWGxC and CWGxD drive copies of CWGxA and CWGxB, respectively, but with polarity controlled by the POLC and POLD bits, respectively.

**Figure 31-3. CWG Push-Pull Mode Operation**

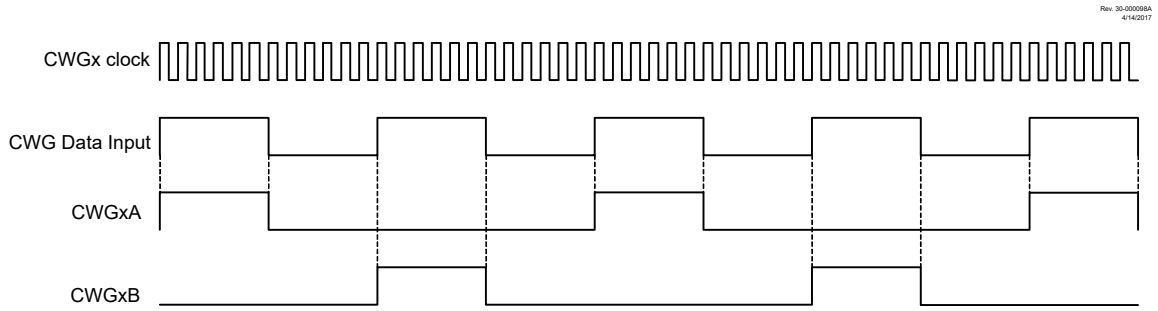
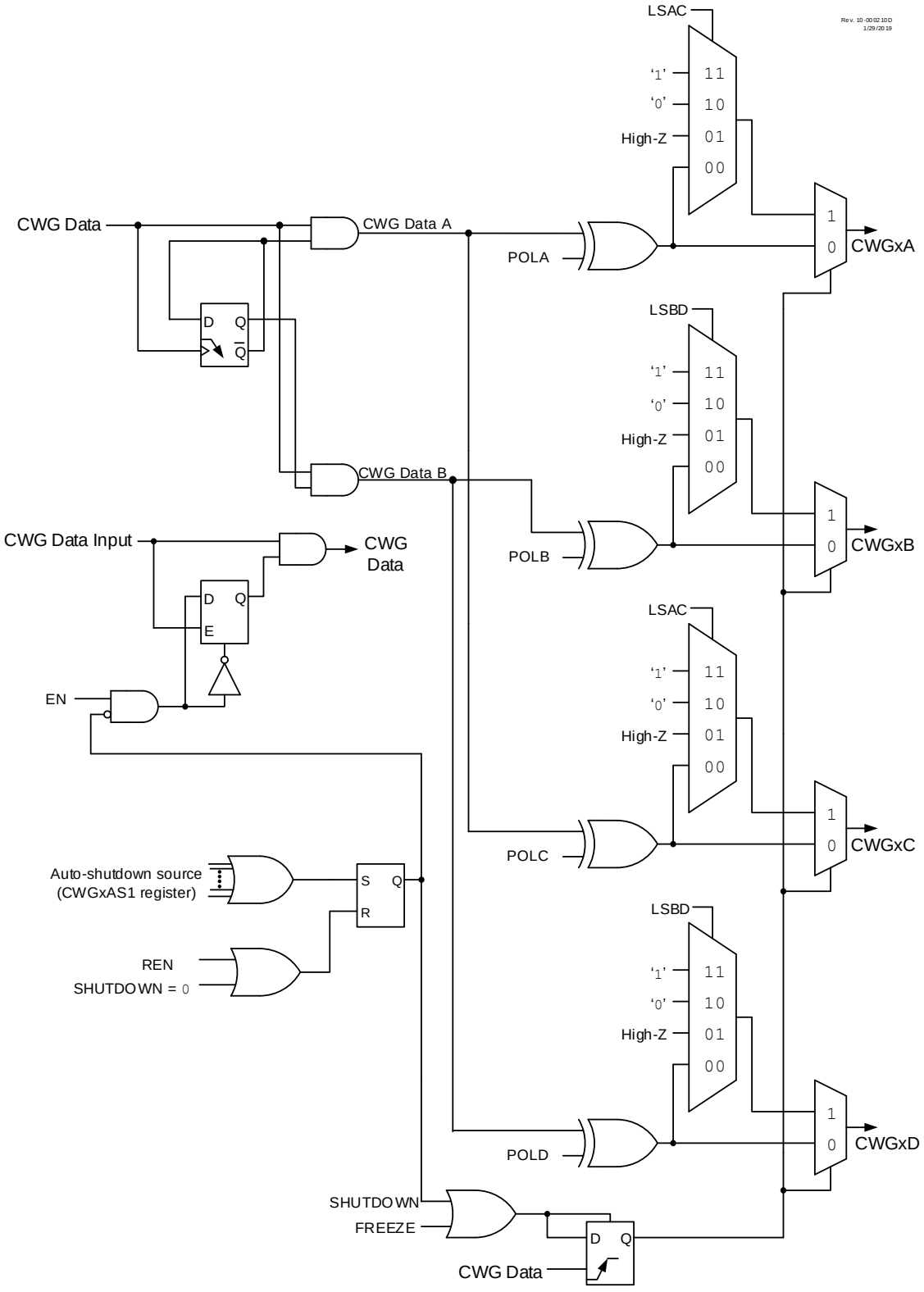


**Figure 31-4. Simplified CWG Block Diagram (Push-Pull Mode, MODE = `b101)**



### 31.2.3 Full Bridge Mode

In Forward and Reverse Full Bridge modes, three outputs drive static values while the fourth is modulated by the input data signal. The mode selection may be toggled between forward and reverse by toggling the MODE[0] bit of the CWGxCON0 register while keeping the MODE[2:1] bits static, without disabling the CWG module. When connected, as shown in Figure 31-5, the outputs are appropriate for a full-bridge motor driver. Each CWG output signal has independent polarity control, so the circuit can be adapted to high-active and low-active drivers. A simplified block diagram for the Full Bridge modes is shown in Figure 31-6.

**Figure 31-5. Example of Full-Bridge Application**

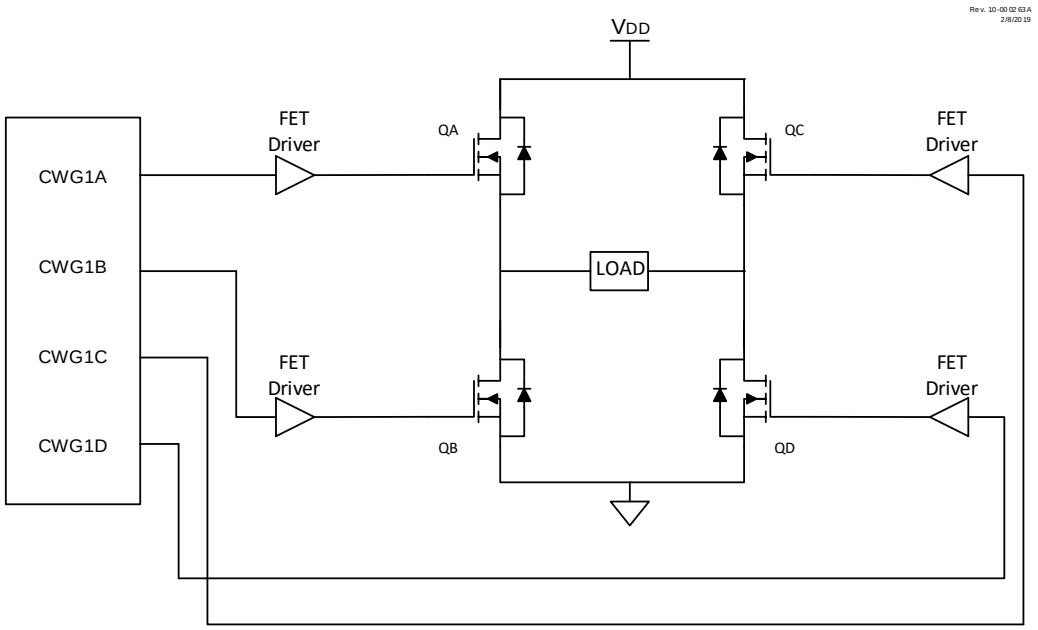
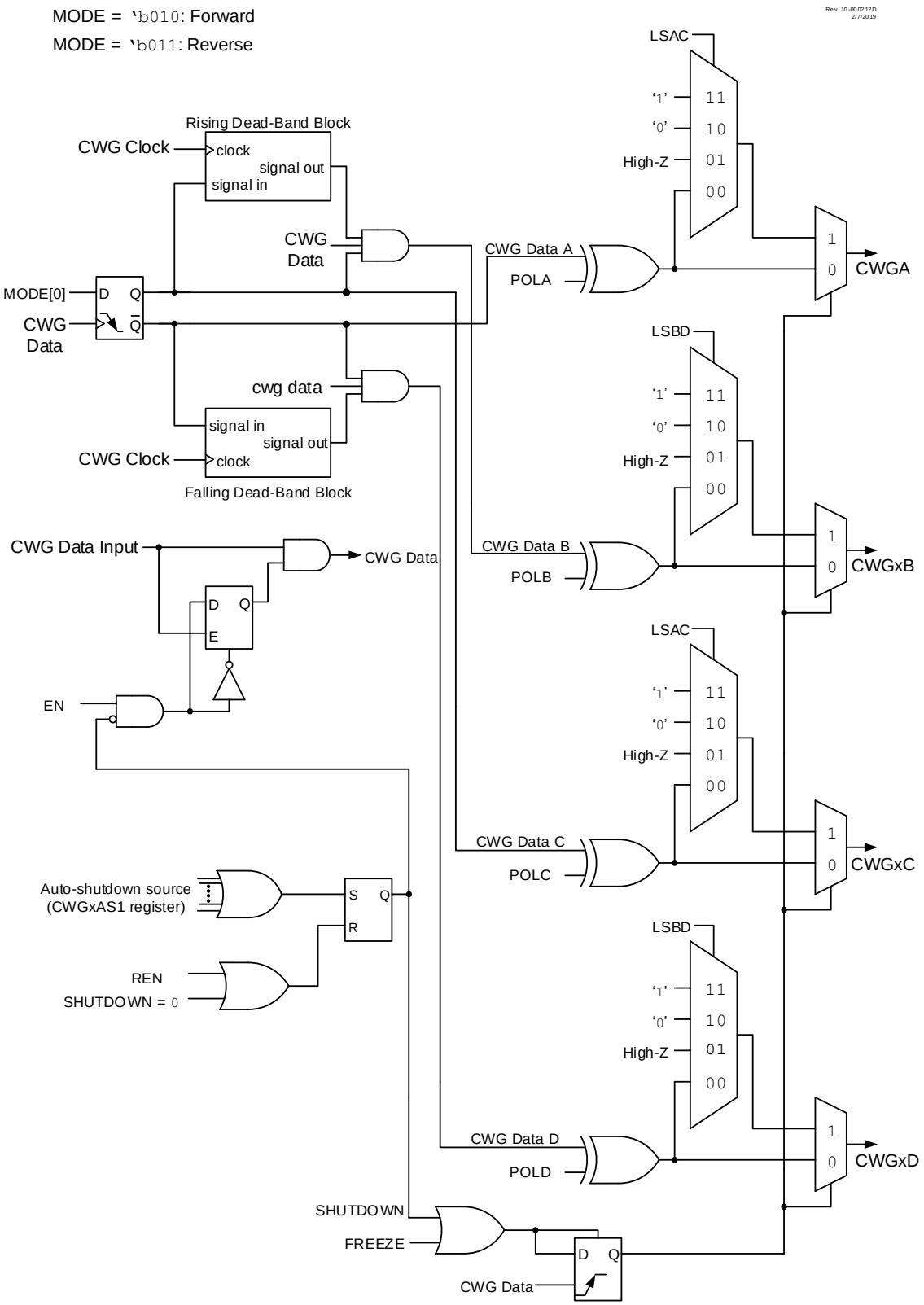


**Figure 31-6. Simplified CWG Block Diagram (Forward and Reverse Full Bridge Modes)**

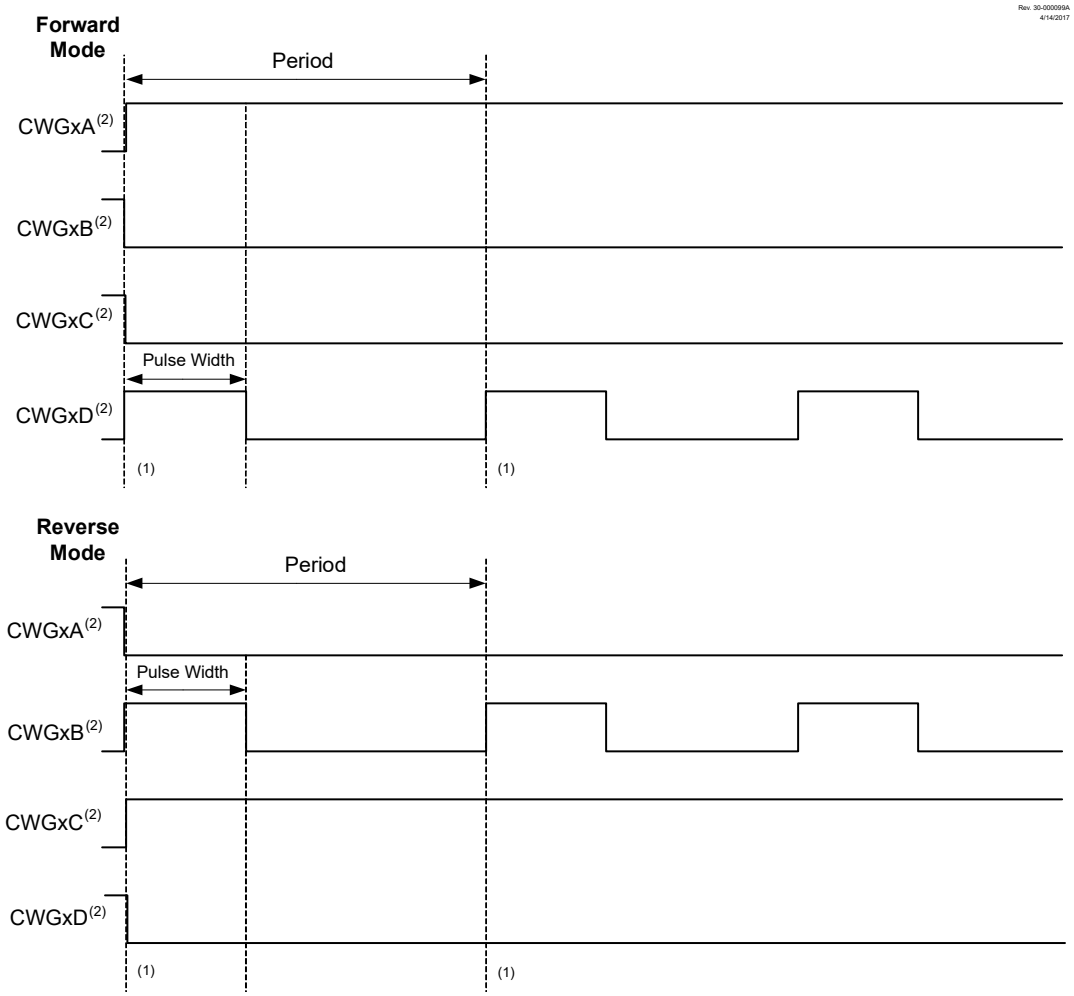


In Forward Full Bridge mode (MODE = 'b010), CWGxA is driven to its Active state, CWGxB and CWGxC are driven to their Inactive state, and CWGxD is modulated by the input signal, as shown in Figure 31-7.

In Reverse Full Bridge mode (MODE = 'b011), CWGxC is driven to its Active state, CWGxA and CWGxD are driven to their Inactive states, and CWGxB is modulated by the input signal, as shown in Figure 31-7.

In Full Bridge mode, the dead-band period is used when there is a switch from forward to reverse or vice versa. This dead-band control is described in the Dead-Band Control section, with additional details in the Rising Edge and Reverse Dead Band and Falling Edge and Forward Dead Band sections. Steering modes are not used with either of the Full Bridge modes.

**Figure 31-7. Example of Full-Bridge Output**




**Notes:**

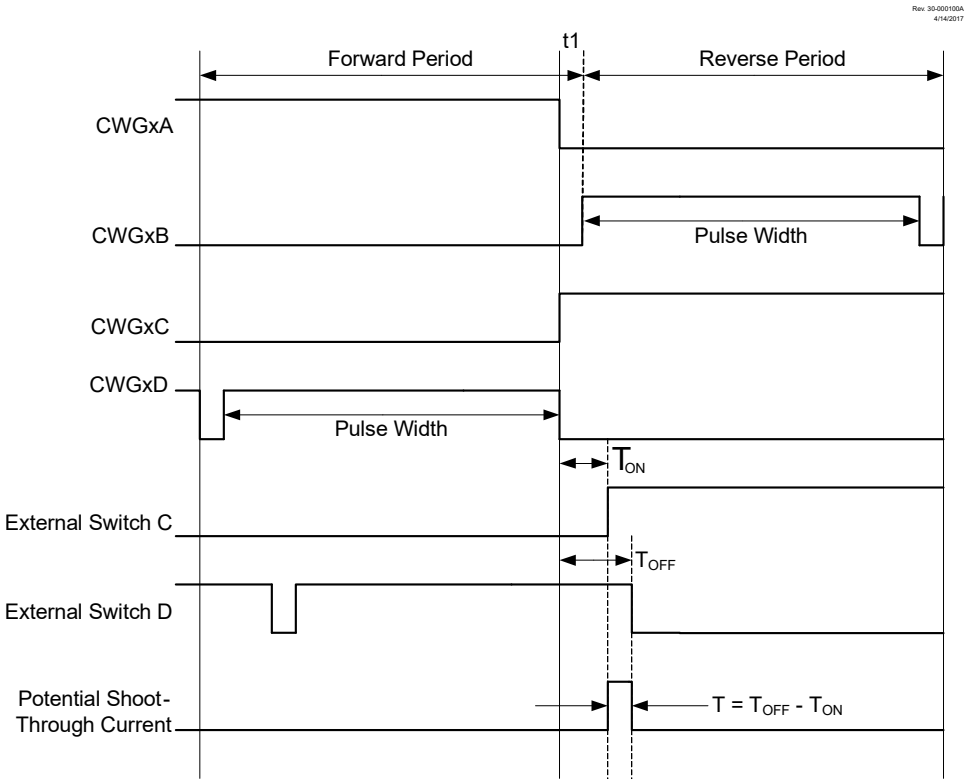
1. A rising CWG data input creates a rising event on the modulated output.
2. Output signals shown as active-high; all POLy bits are clear.

### 31.2.3.1 Direction Change in Full Bridge Mode

In Full Bridge mode, changing the MODE[0] bit controls the forward/reverse direction. Direction changes occur on the next rising edge of the modulated input. The sequence, described as follows, is illustrated in Figure 31-8.

1. The associated active output CWGxA and the inactive output CWGxC are switched to drive in the opposite direction.
2. The previously modulated output CWGxD is switched to the Inactive state, and the previously inactive output CWGxB begins to modulate.
3. CWG modulation resumes after the direction-switch dead band has elapsed.

**Figure 31-8. Example of PWM Direction Change at Near 100% Duty Cycle**




### 31.2.3.2 Dead-Band Delay in Full Bridge Mode

Dead-band delay is important when either of the following conditions is true:

- The direction of the CWG output changes when the duty cycle of the data input is at or near 100%.
- The turn-off time of the power switch, including the power device and driver circuit, is greater than the turn-on time.

The dead-band delay is inserted only when changing directions, and only the modulated output is affected. The statically-configured outputs (CWGxA and CWGxC) are not afforded dead band, and switch essentially simultaneously.

Figure 31-8 shows an example of the CWG outputs changing directions from forward to reverse, at near 100% duty cycle. In this example, at time t1, the output of CWGxA and CWGxD becomes inactive, while the output of CWGxC becomes active. Since the turn-off time of the power devices is longer than the turn-on time, a shoot-through current will flow through the power devices QC and QD for the duration of 'T'. The same phenomenon will occur to power devices QA and QB for the CWG direction change from reverse to forward.

When changing the CWG direction at high duty cycle is required for an application, two possible solutions for eliminating the shoot-through current are:

1. Reduce the CWG duty cycle for one period before changing directions.
2. Use switch drivers that can drive the switches off faster than they can drive them on.

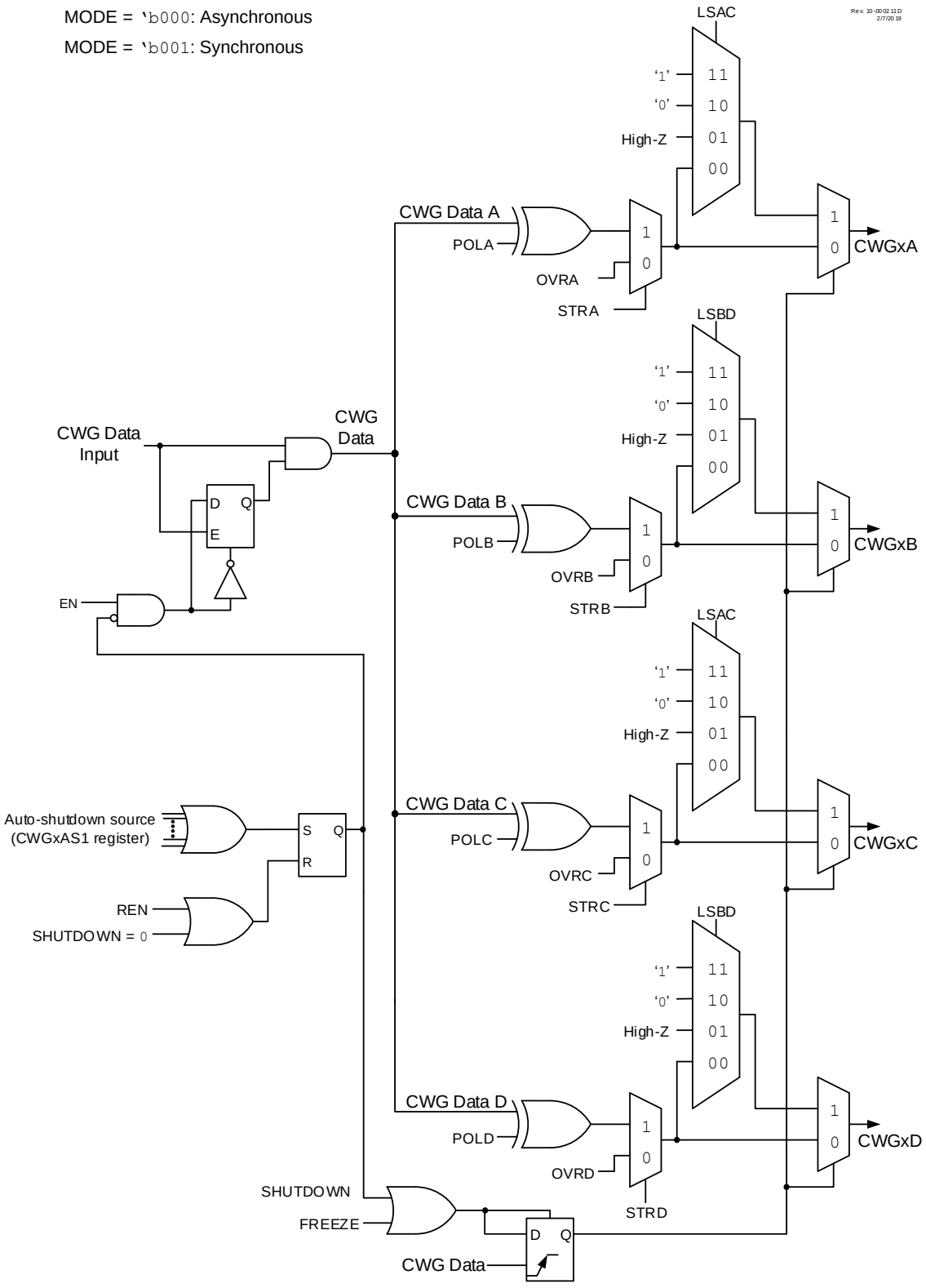
## 31.2.4 Steering Modes

In both Synchronous and Asynchronous Steering modes, the CWG Data can be steered to any combination of four CWG outputs. A fixed value will be presented on all the outputs not used for the PWM output. Each output has independent polarity, steering, and shutdown options. Dead-band control is not used in either Steering mode.

For example, when STRA = 0, the corresponding pin is held at the level defined by OVRA. When STRA = 1, the pin is driven by the CWG Data signal. The POLy bits control the signal polarity only when STRy = 1.

The CWG auto-shutdown operation also applies in Steering modes as described in the Auto-Shutdown section. An auto-shutdown event will only affect pins that have STRy = 1.

**Figure 31-9. Simplified CWG Block Diagram (Output Steering Modes)**

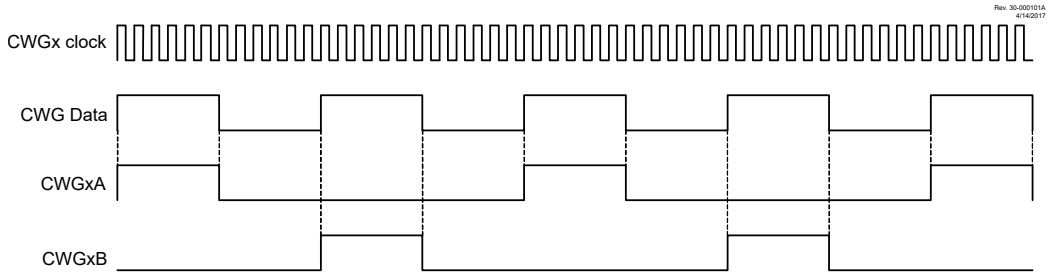


#### 31.2.4.1 Synchronous Steering Mode

In Synchronous Steering mode (MODE = 'b001), the changes to steering selection registers take effect on the next rising edge of CWG Data (see the figure below). In Synchronous Steering mode, the output will always produce a complete waveform.

**Important:** Only the STRx bits are synchronized; the OVRx bits are not synchronized.

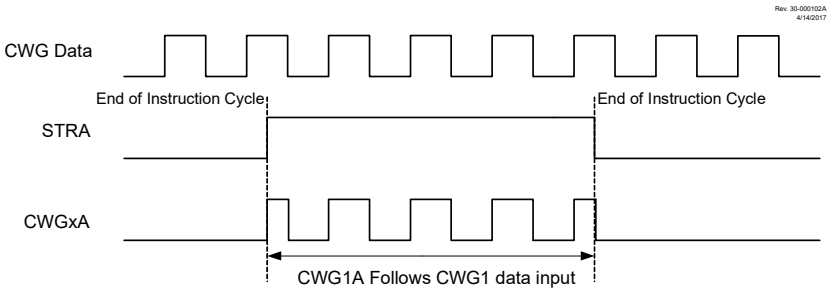
**Figure 31-10. Example of Synchronous Steering (MODE = 'b001)**




#### 31.2.4.2 Asynchronous Steering Mode

In Asynchronous mode (MODE = 'b000), steering takes effect at the end of the instruction cycle that writes to STRx. In Asynchronous Steering mode, the output signal may be an incomplete waveform (see the figure below). This operation may be useful when the user firmware needs to immediately remove a signal from the output pin.

**Figure 31-11. Example of Asynchronous Steering (MODE = 'b000)**




#### 31.2.4.3 Start-Up Considerations

The application hardware must use the proper external pull-up and/or pull-down resistors on the CWG output pins. This is required because all I/O pins are forced to high-impedance at Reset.

The Polarity Control (POLy) bits allow the user to choose whether the output signals are active-high or active-low.

## 31.3 Clock Source

The clock source is used to drive the dead-band timing circuits. The CWG module allows the following clock sources to be selected:

- $F_{OSC}$ (system clock)
- HFINTOSC

When the HFINTOSC is selected, the HFINTOSC will be kept running during Sleep. Therefore, the CWG modes requiring dead band can operate in Sleep, provided that the CWG data input is also active during Sleep. The clock sources are selected using the CS bit. The system clock $F_{OSC}$ is disabled in Sleep and thus dead-band control cannot be used.

## 31.4 Selectable Input Sources

The CWG generates the output waveforms from the input sources which are selected with the ISM bits. Refer to the CWGxISM register for more details.

## 31.5 Output Control

### 31.5.1 CWG Output

Each CWG output can be routed to a Peripheral Pin Select (PPS) output via the RxyPPS register. Refer to the **"PPS - Peripheral Pin Select Module"** chapter for more details.

### 31.5.2 Polarity Control

The polarity of each CWG output can be selected independently. When the output polarity bit is set, the corresponding output is active-high. Clearing the output polarity bit configures the corresponding output as active-low. However, polarity does not affect the override levels. Output polarity is selected with the POLy bits. Auto-shutdown and steering options are unaffected by polarity.

## 31.6 Dead-Band Control

The dead-band control provides nonoverlapping complementary outputs to prevent shoot-through current when the outputs switch. Dead-band operation is employed for Half Bridge and Full Bridge modes. The CWG contains two 6-bit dead-band counters. One is used for the rising edge of the input source control in Half Bridge mode or for reverse direction change dead band in Full Bridge mode. The other is used for the falling edge of the input source control in Half Bridge mode or for forward direction change dead band in Full Bridge mode.

Dead band is timed by counting CWG clock periods from zero up to the value in the rising or falling dead-band counter registers.

### 31.6.1 Dead-Band Functionality in Half Bridge Mode

In Half Bridge mode, the dead-band counters dictate the delay between the falling edge of the normal output and the rising edge of the inverted output. This can be seen in Figure 31-1.

### 31.6.2 Dead-Band Functionality in Full Bridge Mode

In Full Bridge mode, the dead-band counters are used when undergoing a direction change. The MODE[0] bit can be set or cleared while the CWG is running, allowing for changes from Forward to Reverse mode. The CWGxA and CWGxC signals will change immediately upon the first rising input edge following a direction change, but the modulated signals (CWGxB or CWGxD, depending on the direction of the change) will experience a delay dictated by the dead-band counters.

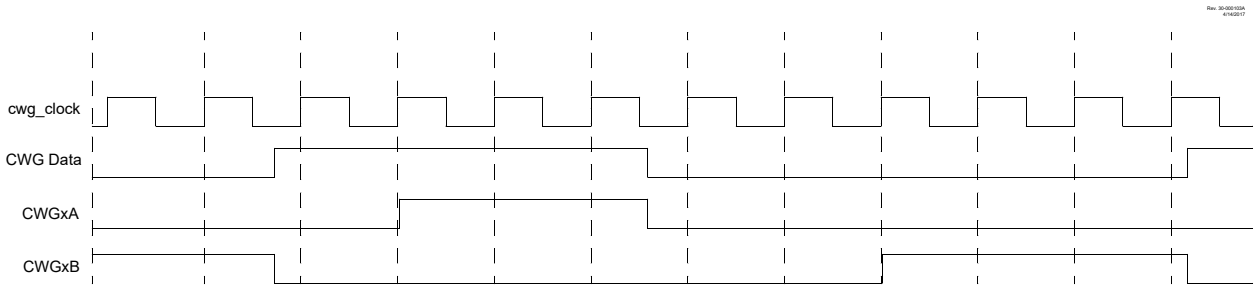
## 31.7 Rising Edge and Reverse Dead Band

In Half Bridge mode, the rising edge dead band delays the turn-on of the CWGxA output after the rising edge of the CWG data input. In Full Bridge mode, the reverse dead-band delay is only inserted when changing directions from Forward mode to Reverse mode, and only the modulated output, CWGxB, is affected.

The CWGxDBR register determines the duration of the dead-band interval on the rising edge of the input source signal. This duration is from 0 to 64 periods of the CWG clock. The following figure illustrates different dead-band delays for rising and falling CWG Data events.

© 2019-2023 Microchip Technology Inc. and its subsidiaries

**Figure 31-12. Dead-Band Operation, CWGxDBR = 0x01, CWGxDBF = 0x02**




Dead band is always initiated on the edge of the input source signal. A count of zero indicates that no dead band is present.

If the input source signal reverses polarity before the dead-band count is completed, then no signal will be seen on the respective output.

The CWGxDBR register value is double-buffered. When EN = 0, the buffer is loaded when CWGxDBR is written. When EN = 1, the buffer will be loaded at the rising edge following the first falling edge of the CWG Data, after the LD bit is set.

## 31.8 Falling Edge and Forward Dead Band

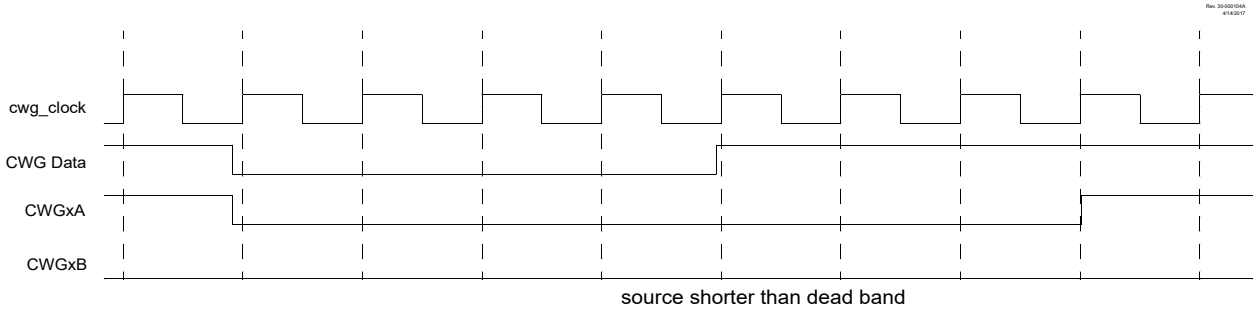
In Half Bridge mode, the falling edge dead band delays the turn-on of the CWGxB output at the falling edge of the CWG data input. In Full Bridge mode, the forward dead-band delay is only inserted when changing directions from Reverse mode to Forward mode, and only the modulated output, CWGxD, is affected.

The CWGxDBF register determines the duration of the dead-band interval on the falling edge of the input source signal. This duration is from 0 to 64 periods of the CWG clock.

Dead-band delay is always initiated on the edge of the input source signal. A count of zero indicates that no dead band is present.

If the input source signal reverses polarity before the dead-band count is completed, then no signal will be seen on the respective output.

**Figure 31-13. Dead-Band Operation, CWGxDBR = 0x03, CWGxDBF = 0x06, Source Shorter Than Dead Band**




The CWGxDBF register value is double-buffered. When EN = 0, the buffer is loaded when CWGxDBF is written. When EN = 1, the buffer will be loaded at the rising edge following the first falling edge of the data input after the LD bit is set.

## 31.9 Dead-Band Jitter

When the rising and falling edges of the input source are asynchronous to the CWG clock, it creates jitter in the dead-band time delay. The maximum jitter is equal to one CWG clock period. Refer to the equations below for more details.

**Equation 31-1. Dead-Band Delay Time Calculation**

$$T_{DEAD-BAND_MIN} = \frac{1}{F_{CWG_CLOCK}} \bullet DBx$$

$$T_{DEAD-BAND_MAX} = \frac{1}{F_{CWG_CLOCK}} \bullet (DBx + 1)$$

$$T_{JITTER} = T_{DEAD-BAND_MAX} - T_{DEAD-BAND_MIN}$$

$$T_{JITTER} = \frac{1}{F_{CWG_CLOCK}}$$

$$T_{DEAD-BAND_MAX} = T_{DEAD-BAND_MIN} + T_{JITTER}$$

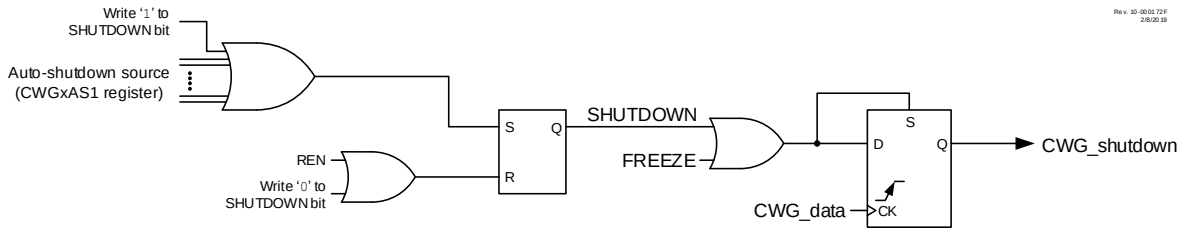
**Dead-Band Delay Example Calculation**

$DBx = 0x0A = 10$
$F_{CWG_CLOCK} = 8\ MHz$
$T_{JITTER} = \frac{1}{8\ MHz} = 125\ ns$
$T_{DEAD-BAND_MIN} = 125\ ns \bullet 10 = 1.25\ \mu s$
$T_{DEAD-BAND_MAX} = 1.25\ \mu s + 0.125\ \mu s = 1.37\ \mu s$

## 31.10 Auto-Shutdown

Auto-shutdown is a method to immediately override the CWG output levels with specific overrides that allow for safe shutdown of the circuit. The Shutdown state can be either cleared automatically or held until cleared by software. The auto-shutdown circuit is illustrated in the following figure.

**Figure 31-14. CWG Shutdown Block Diagram**




### 31.10.1 Shutdown

The Shutdown state can be entered by either of the following two methods:

- Software Generated
- External Input

### 31.10.2 Software Generated Shutdown

Setting the SHUTDOWN bit will force the CWG into the Shutdown state.

When the auto-restart is disabled, the Shutdown state will persist as long as the SHUTDOWN bit is set.

When auto-restart is enabled, the SHUTDOWN bit will clear automatically and resume operation on the next rising edge event. The SHUTDOWN bit indicates when a Shutdown condition exists. The bit may be set or cleared in software or by hardware.

### 31.10.3 External Input Source

External shutdown inputs provide the fastest way to safely suspend CWG operation in the event of a Fault condition. When any of the selected shutdown inputs goes active, the CWG outputs will immediately go to the selected override levels without software delay. The override levels are selected by the LSBD and LSAC bits. Several input sources can be selected to cause a Shutdown condition. All input sources are active-low. The shutdown input sources are individually enabled by the ASyE bits.

**Important:** Shutdown inputs are level sensitive, not edge sensitive. The Shutdown state cannot be cleared, except by disabling auto-shutdown, as long as the shutdown input level persists.

### 31.10.4 Pin Override Levels

The levels driven to the CWG outputs during an auto-shutdown event are controlled by the LSBD and LSAC bits. The LSBD bits control CWGxB/D output levels, while the LSAC bits control the CWGxA/C output levels.

### 31.10.5 Auto-Shutdown Interrupts

When an auto-shutdown event occurs, either by software or hardware setting SHUTDOWN, the CWGxIF flag bit of the PIRx register is set.

## 31.11 Auto-Shutdown Restart

After an auto-shutdown event has occurred, there are two ways to resume operation:

- Software controlled
- Auto-restart

In either case, the shutdown source must be cleared before the restart can take place. That is, either the Shutdown condition must be removed, or the corresponding ASyE bit must be cleared.

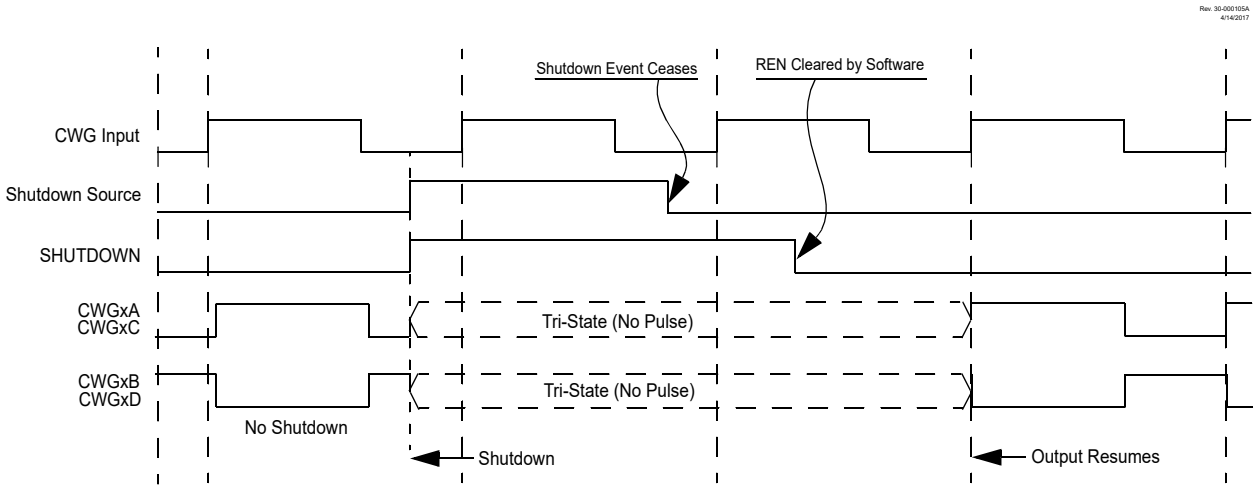
### 31.11.1 Software-Controlled Restart

When the REN bit is clear (REN = 0), the CWG module must be restarted after an auto-shutdown event through software.

Once all auto-shutdown sources are removed, the software must clear the SHUTDOWN bit. Once SHUTDOWN is cleared, the CWG module will resume operation upon the first rising edge of the CWG data input.

**Important:** The SHUTDOWN bit cannot be cleared in software if the Auto-Shutdown condition is still present.

**Figure 31-15. Shutdown Functionality, Auto-Restart Disabled (REN = 0, LSAC = 'b01, LSBD = 'b01)**



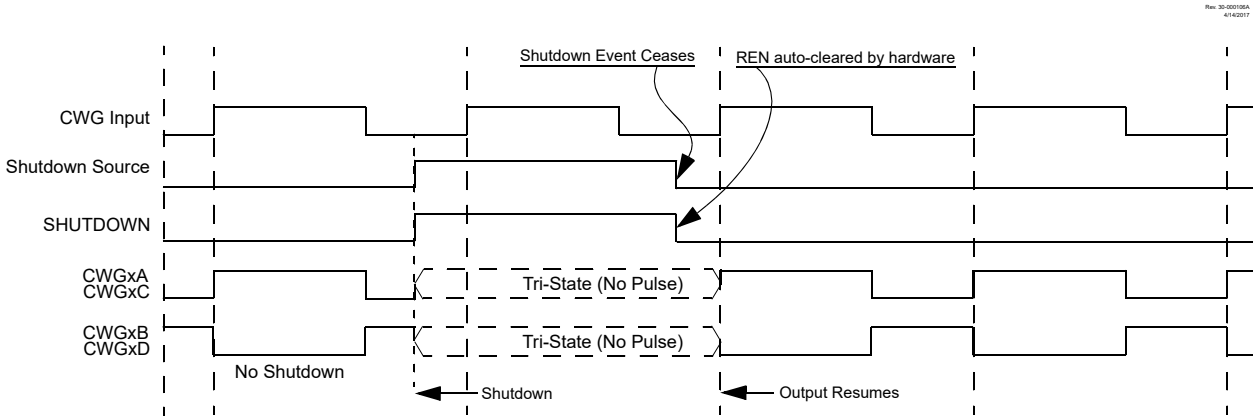

### 31.11.2 Auto-Restart

When the REN bit is set (REN = 1), the CWG module will restart from the Shutdown state automatically.

Once all Auto-Shutdown conditions are removed, the hardware will automatically clear the SHUTDOWN bit. Once SHUTDOWN is cleared, the CWG module will resume operation upon the first rising edge of the CWG data input.

**Important:** The SHUTDOWN bit cannot be cleared in software if the Auto-Shutdown condition is still present.

**Figure 31-16. Shutdown Functionality, Auto-Restart Enabled (REN = 1, LSAC = 'b01, LSBD = 'b01)**




## 31.12 Operation During Sleep

The CWG module operates independently from the system clock and will continue to run during Sleep, provided that the clock and input sources selected remain active.

The HFINTOSC remains active during Sleep when all the following conditions are met:

- CWG module is enabled
- Input source is active
- HFINTOSC is selected as the clock source, regardless of the system clock source selected.

In other words, if the HFINTOSC is simultaneously selected as the system clock and the CWG clock source, when the CWG is enabled and the input source is active, then the CPU will go Idle during Sleep, but the HFINTOSC will remain active and the CWG will continue to operate. This will have a direct effect on the Sleep mode current.

## 31.13 Configuring the CWG

1. Ensure that the TRIS control bits corresponding to CWG outputs are set so that all are configured as inputs, ensuring that the outputs are inactive during setup. External hardware must ensure that pin levels are held to safe levels.
2. Clear the EN bit, if not already cleared.
3. Configure the MODE bits to set the output operating mode.
4. Configure the POLy bits to set the output polarities.
5. Configure the ISM bits to select the data input source.
6. If a Steering mode is selected, configure the STRy bits to select the desired output on the CWG outputs.
7. Configure the LSBD and LSAC bits to select the Auto-Shutdown Output Override states (this is necessary even if not using auto-shutdown, because start-up will be from a Shutdown state).
8. If auto-restart is desired, set the REN bit.
9. If auto-shutdown is desired, configure the ASyE bits to select the shutdown source.
10. Set the desired rising and falling dead-band times with the CWGxDBR and CWGxDBF registers.
11. Select the clock source with the CS bit.

© 2019-2023 Microchip Technology Inc. and its subsidiaries

12. Set the EN bit to enable the module.
13. Clear the TRIS bits that correspond to the CWG outputs to set them as outputs.

If auto-restart is to be used, set the REN bit and the SHUTDOWN bit will be cleared automatically. Otherwise, clear the SHUTDOWN bit in software to start the CWG.

## 31.14 Register Definitions: CWG Control

Long bit name prefixes for the CWG peripherals are shown in the table below. Refer to the **"Long Bit Names"** section in the **"Register and Bit Naming Conventions"** chapter for more information.

**Table 31-1. CWG Long Bit Name Prefixes**

| Peripheral | Bit Name Prefix |
|---|---|
| CWG1 | CWG1 |
| CWG2 | CWG2 |
| CWG3 | CWG3 |

### 31.14.1 CWGxCON0

**Name:** CWGxCON0
**Address:** 0x3C0,0x3C9,0x3D2

CWG Control Register 0

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | EN | LD | | | | MODE[2:0] | | |
| Access | R/W | R/W/HC | | | | R/W | R/W | R/W |
| Reset | 0 | 0 | | | | 0 | 0 | 0 |

**Bit 7 – EN** CWG Enable

| Value | Description |
|---|---|
| 1 | Module is enabled |
| 0 | Module is disabled |

**Bit 6 – LD** CWG1 Load Buffers(1)

| Value | Description |
|---|---|
| 1 | Dead-band count buffers to be loaded on CWG data rising edge, following first falling edge after this bit is set |
| 0 | Buffers remain unchanged |

**Bits 2:0 – MODE[2:0]** CWG Mode

| Value | Description |
|---|---|
| 111 | Reserved |
| 110 | Reserved |
| 101 | CWG outputs operate in Push-Pull mode |
| 100 | CWG outputs operate in Half Bridge mode |
| 011 | CWG outputs operate in Reverse Full Bridge mode |
| 010 | CWG outputs operate in Forward Full Bridge mode |
| 001 | CWG outputs operate in Synchronous Steering mode |
| 000 | CWG outputs operate in Asynchronous Steering mode |

**Note:**

1. This bit can only be set after EN = 1; it cannot be set in the same cycle when EN is set.

### 31.14.2 CWGxCON1

**Name:** CWGxCON1
**Address:** 0x3C1,0x3CA,0x3D3

CWG Control Register 1

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | IN | | POLD | POLC | POLB | POLA |
| Access | | | R | | R/W | R/W | R/W | R/W |
| Reset | | | x | | 0 | 0 | 0 | 0 |

**Bit 5 – IN** CWG Input Value (read-only)

| Value | Description |
|---|---|
| 1 | CWG data input is a logic '1' |
| 0 | CWG data input is a logic '0' |

**Bits 0, 1, 2, 3 – POLy** CWG Output 'y' Polarity

| Value | Description |
|---|---|
| 1 | Signal output is inverted polarity |
| 0 | Signal output is normal polarity |

### 31.14.3 CWGxCLK

**Name:** CWGxCLK
**Address:** 0x3BC,0x3C5,0x3CE

CWG Clock Input Selection Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | CS |
| Access | | | | | | | | R/W |
| Reset | | | | | | | | 0 |

**Bit 0 – CS** CWG Clock Source Selection Select

| Value | Description |
|---|---|
| 1 | HFINTOSC (remains operating during Sleep) |
| 0 | $F_{OSC}$ |

### 31.14.4 CWGxISM

**Name:** CWGxISM
**Address:** 0x3BD,0x3C6,0x3CF

CWGx Input Selection Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | ISM[4:0] | | | | |
| Access | | | | R/W | R/W | R/W | R/W | R/W |
| Reset | | | | 0 | 0 | 0 | 0 | 0 |

**Bits 4:0 – ISM[4:0]** CWG Data Input Source Select

| ISM | Input Selection | | |
|---|---|---|---|
| | CWG1 | CWG2 | CWG3 |
| 11111-11010 | Reserved | | |
| 11001 | CLC8_OUT | | |
| 11000 | CLC7_OUT | | |
| 10111 | CLC6_OUT | | |
| 10110 | CLC5_OUT | | |
| 10101 | CLC4_OUT | | |
| 10100 | CLC3_OUT | | |
| 10011 | CLC2_OUT | | |
| 10010 | CLC1_OUT | | |
| 10001 | DSM1_OUT | | |
| 10000 | CMP2_OUT | | |
| 01111 | CMP1_OUT | | |
| 01110 | NCO3_OUT | | |
| 01101 | NCO2_OUT | | |
| 01100 | NCO1_OUT | | |
| 01011-01010 | Reserved | | |
| 01001 | PWM3S1P1_OUT | | |
| 01000 | PWM3S1P2_OUT | | |
| 00111 | PWM2S1P1_OUT | | |
| 00110 | PWM2S1P2_OUT | | |
| 00101 | PWM1S1P1_OUT | | |
| 00100 | PWM1S1P2_OUT | | |
| 00011 | CCP3_OUT | | |
| 00010 | CCP2_OUT | | |
| 00001 | CCP1_OUT | | |
| 00000 | Pin selected by CWG1PPS | Pin selected by CWG2PPS | Pin selected by CWG3PPS |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

### 31.14.5 CWGxSTR

**Name:** CWGxSTR
**Address:** 0x3C4,0x3CD,0x3D6

CWG Steering Control Register[1]

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | OVRD | OVRC | OVRB | OVRA | STRD | STRC | STRB | STRA |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 4, 5, 6, 7 – OVRy** Steering Data OVR'y'

| Value | Condition | Description |
|---|---|---|
| x | STRy = 1 | CWGx'y' output has the CWG data input waveform with polarity control from POLy bit |
| 1 | STRy = 0 and POLy = x | CWGx'y' output is high |
| 0 | STRy = 0 and POLy = x | CWGx'y' output is low |

**Bits 0, 1, 2, 3 – STRy** STR'y' Steering Enable[2]

| Value | Description |
|---|---|
| 1 | CWGx'y' output has the CWG data input waveform with polarity control from the POLy bit |
| 0 | CWGx'y' output is assigned to value of the OVRy bit |

**Notes:**

1. The bits in this register apply only when MODE = 'b00x (CWGxCON0, Steering modes).
2. This bit is double-buffered when MODE = 'b001.

### 31.14.6 CWGxAS0

**Name:** CWGxAS0
**Address:** 0x3C2,0x3CB,0x3D4

CWG Auto-Shutdown Control Register 0

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | SHUTDOWN | REN | LSBD[1:0] | | LSAC[1:0] | | | |
| Access | R/W/HS/HC | R/W | R/W | R/W | R/W | R/W | | |
| Reset | 0 | 0 | 0 | 1 | 0 | 1 | | |

**Bit 7 – SHUTDOWN** Auto-Shutdown Event Status[1,2]

| Value | Description |
|---|---|
| 1 | An Auto-Shutdown state is in effect |
| 0 | No auto-shutdown event has occurred |

**Bit 6 – REN** Auto-Restart Enable

| Value | Description |
|---|---|
| 1 | Auto-restart is enabled |
| 0 | Auto-restart is disabled |

**Bits 5:4 – LSBD[1:0]** CWGxB and CWGxD Auto-Shutdown State Control

| Value | Description |
|---|---|
| 11 | A logic '1' is placed on CWGxB/D when an auto-shutdown event occurs |
| 10 | A logic '0' is placed on CWGxB/D when an auto-shutdown event occurs |
| 01 | Pin is tri-stated on CWGxB/D when an auto-shutdown event occurs |
| 00 | The Inactive state of the pin, including polarity, is placed on CWGxB/D after the required dead-band interval when an auto-shutdown event occurs |

**Bits 3:2 – LSAC[1:0]** CWGxA and CWGxC Auto-Shutdown State Control

| Value | Description |
|---|---|
| 11 | A logic '1' is placed on CWGxA/C when an auto-shutdown event occurs |
| 10 | A logic '0' is placed on CWGxA/C when an auto-shutdown event occurs |
| 01 | Pin is tri-stated on CWGxA/C when an auto-shutdown event occurs |
| 00 | The Inactive state of the pin, including polarity, is placed on CWGxA/C after the required dead-band interval when an auto-shutdown event occurs |

**Notes:**

1. This bit may be written while EN = 0, to place the outputs into the shutdown configuration.
2. The outputs will remain in Auto-Shutdown state until the next rising edge of the CWG data input after this bit is cleared.

### 31.14.7 CWGxAS1

**Name:** CWGxAS1
**Address:** 0x3C3,0x3CC,0x3D5

CWG Auto-Shutdown Control Register 1

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | AS7E | AS6E | AS5E | AS4E | AS3E | AS2E | AS1E | AS0E |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 0, 1, 2, 3, 4, 5, 6, 7 – ASyE** CWG Auto-Shutdown Source Enable(1,2)

| ASyE | Auto-Shutdown Source | | |
|---|---|---|---|
| | CWG1 | CWG2 | CWG3 |
| AS7E | CLC6_OUT | | |
| AS6E | CLC2_OUT | CLC3_OUT | CLC4_OUT |
| AS5E | CMP2_OUT | | |
| AS4E | CMP1_OUT | | |
| AS3E | TMR6_Postscaler_OUT | | |
| AS2E | TMR4_Postscaler_OUT | | |
| AS1E | TMR2_Postscaler_OUT | | |
| AS0E | Pin selected by CWG1PPS | Pin selected by CWG2PPS | Pin selected by CWG3PPS |

**Notes:**

1. This bit may be written while EN = 0, to place the outputs into the shutdown configuration.
2. The outputs will remain in Auto-Shutdown state until the next rising edge of the CWG data input after this bit is cleared.

### 31.14.8 CWGxDBR

**Name:** CWGxDBR
**Address:** 0x3BE,0x3C7,0x3D0

CWG Rising Dead-Band Count Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | DBR[5:0] | | | | | |
| Access | | | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | | | x | x | x | x | x | x |

**Bits 5:0 – DBR[5:0]** CWG Rising Edge-Triggered Dead-Band Count
Reset States: POR/BOR = xxxxxx
All Other Resets = uuuuuu

| Value | Description |
|---|---|
| n | Dead band is active no less than n and no more than n+1 CWG clock periods after the rising edge |
| 0 | 0 CWG clock periods. Dead-band generation is bypassed. |

### 31.14.9 CWGxDBF

**Name:** CWGxDBF
**Address:** 0x3BF,0x3C8,0x3D1

CWG Falling Dead-Band Count Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | DBF[5:0] | | | | | |
| Access | | | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | | | x | x | x | x | x | x |

**Bits 5:0 – DBF[5:0]** CWG Falling Edge-Triggered Dead-Band Count
Reset States: POR/BOR = xxxxxx
All Other Resets = uuuuuu

| Value | Description |
|---|---|
| n | Dead band is active no less than n and no more than n+1 CWG clock periods after the falling edge |
| 0 | 0 CWG clock periods. Dead-band generation is bypassed. |

## 31.15 Register Summary - CWG

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x03BC | CWG1CLK | 7:0 | | | | | | | | CS |
| 0x03BD | CWG1ISM | 7:0 | | | | ISM[4:0] | | | | |
| 0x03BE | CWG1DBR | 7:0 | | | DBR[5:0] | | | | | |
| 0x03BF | CWG1DBF | 7:0 | | | DBF[5:0] | | | | | |
| 0x03C0 | CWG1CON0 | 7:0 | EN | LD | | | | MODE[2:0] | | |
| 0x03C1 | CWG1CON1 | 7:0 | | | IN | | POLD | POLC | POLB | POLA |
| 0x03C2 | CWG1AS0 | 7:0 | SHUTDOWN | REN | LSBD[1:0] | | LSAC[1:0] | | | |
| 0x03C3 | CWG1AS1 | 7:0 | AS7E | AS6E | AS5E | AS4E | AS3E | AS2E | AS1E | AS0E |
| 0x03C4 | CWG1STR | 7:0 | OVRD | OVRC | OVRB | OVRA | STRD | STRC | STRB | STRA |
| 0x03C5 | CWG2CLK | 7:0 | | | | | | | | CS |
| 0x03C6 | CWG2ISM | 7:0 | | | | ISM[4:0] | | | | |
| 0x03C7 | CWG2DBR | 7:0 | | | DBR[5:0] | | | | | |
| 0x03C8 | CWG2DBF | 7:0 | | | DBF[5:0] | | | | | |
| 0x03C9 | CWG2CON0 | 7:0 | EN | LD | | | | MODE[2:0] | | |
| 0x03CA | CWG2CON1 | 7:0 | | | IN | | POLD | POLC | POLB | POLA |
| 0x03CB | CWG2AS0 | 7:0 | SHUTDOWN | REN | LSBD[1:0] | | LSAC[1:0] | | | |
| 0x03CC | CWG2AS1 | 7:0 | AS7E | AS6E | AS5E | AS4E | AS3E | AS2E | AS1E | AS0E |
| 0x03CD | CWG2STR | 7:0 | OVRD | OVRC | OVRB | OVRA | STRD | STRC | STRB | STRA |
| 0x03CE | CWG3CLK | 7:0 | | | | | | | | CS |
| 0x03CF | CWG3ISM | 7:0 | | | | ISM[4:0] | | | | |
| 0x03D0 | CWG3DBR | 7:0 | | | DBR[5:0] | | | | | |
| 0x03D1 | CWG3DBF | 7:0 | | | DBF[5:0] | | | | | |
| 0x03D2 | CWG3CON0 | 7:0 | EN | LD | | | | MODE[2:0] | | |
| 0x03D3 | CWG3CON1 | 7:0 | | | IN | | POLD | POLC | POLB | POLA |
| 0x03D4 | CWG3AS0 | 7:0 | SHUTDOWN | REN | LSBD[1:0] | | LSAC[1:0] | | | |
| 0x03D5 | CWG3AS1 | 7:0 | AS7E | AS6E | AS5E | AS4E | AS3E | AS2E | AS1E | AS0E |
| 0x03D6 | CWG3STR | 7:0 | OVRD | OVRC | OVRB | OVRA | STRD | STRC | STRB | STRA |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

# 32. NCO - Numerically Controlled Oscillator Module

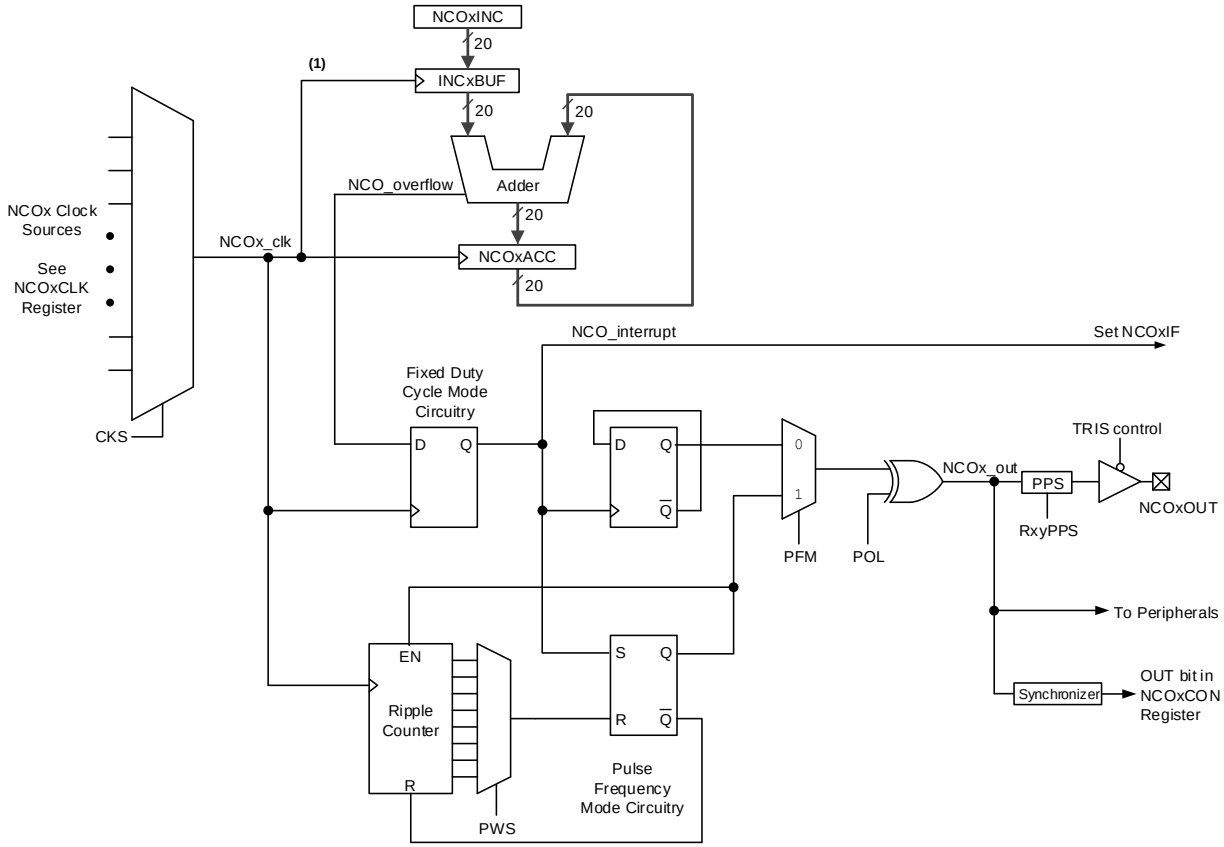
The Numerically Controlled Oscillator (NCO) module is a timer that uses overflow from the addition of an increment value to divide the input frequency. The advantage of the addition method over a simple counter driven timer is that the output frequency resolution does not vary with the divider value. The NCO is most useful for applications that require frequency accuracy and fine resolution at a fixed duty cycle.

Features of the NCO include:

- 20-Bit Increment Function
- Fixed Duty Cycle (FDC) mode
- Pulse Frequency (PF) mode
- Output Pulse-Width Control
- Multiple Clock Input Sources
- Output Polarity Control
- Interrupt Capability

The following figure is a simplified block diagram of the NCO module.

**Figure 32-1. Numerically Controlled Oscillator Module Simplified Block Diagram**




**Note 1:** The increment registers are double-buffered to allow for value changes to be made without first disabling the NCO module. The full increment value is loaded into the buffer registers on the second rising edge of the NCOx_clk signal that occurs immediately after a write to the NCOxINCL register. The buffers are not user-accessible and are shown here for reference.

## 32.1 NCO Operation

The NCO operates by repeatedly adding a fixed value to an accumulator. Additions occur at the input clock rate. The accumulator will overflow with a carry periodically, which is the raw NCO output (NCO_overflow). This effectively reduces the input clock by the ratio of the addition value to the maximum accumulator value. See the following equation.

**Equation 32-1. NCO Overflow Frequency**

$$F_{OVERFLOW} = \frac{NCO\ Clock\ Frequency \times Increment\ Value}{2^{20}}$$

It is apparent from the equation that there is a linear relationship between the increment value and the overflow frequency. This linear advantage over divide-by-n timers comes at the cost of output jitter. However, the jitter is always plus or minus one NCO clock period that occurs periodically, depending on the division remainder. For example, there is no jitter when there is no division remainder, whereas a division remainder of 0.5 will result in a jitter frequency one half of the overflow frequency.

### 32.1.1 NCO Clock Sources

The NCO can be clocked from a variety of sources including the system clock, internal timers, and other peripherals. The NCO clock source is selected by configuring the CKS bits.

### 32.1.2 Accumulator

The accumulator is a 20-bit register. Read and write access to the accumulator is available through three registers:

- NCOxACCL
- NCOxACCH
- NCOxACCU

### 32.1.3 Adder

The NCO adder is a full adder, which operates synchronously from the source clock. The addition of the previous result and the increment value replaces the accumulator value on the rising edge of each input clock.

### 32.1.4 Increment Registers

The increment value is stored in three registers making up a 20-bit word. In order of LSB to MSB, they are:

- NCOxINCL
- NCOxINCH
- NCOxINCU

The increment registers are readable and writable and are double-buffered to allow value changes to be made without first disabling the NCO module.

When the NCO module is enabled, the NCOxINCU and NCOxINCH registers will be written first, then the NCOxINCL register. Writing to the NCOxINCL register initiates the increment buffer registers to be loaded simultaneously on the second rising edge of the NCO_clk signal.

When the NCO module is disabled, the increment buffers are loaded immediately after a write to the increment registers.

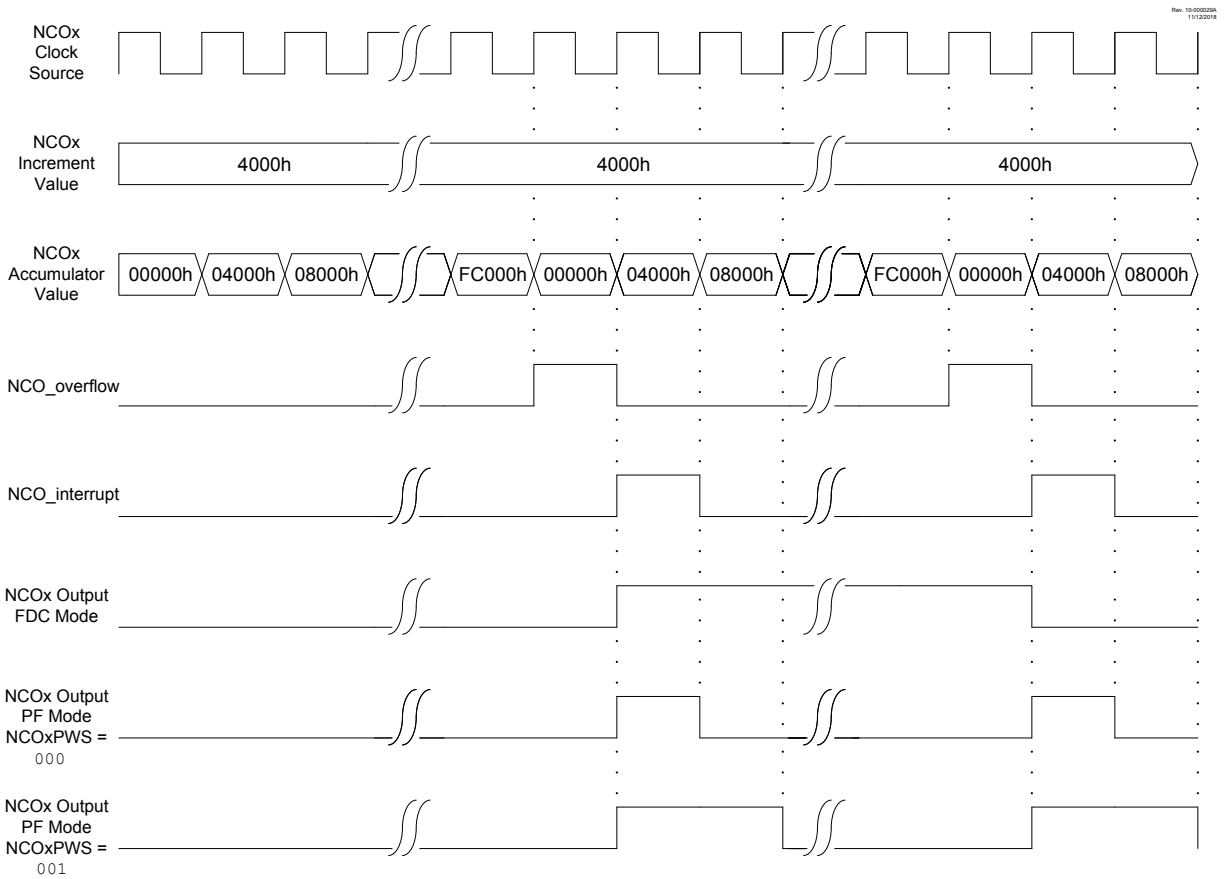
**Important:** The increment buffer registers are not user-accessible.

## 32.2 Fixed Duty Cycle Mode

In Fixed Duty Cycle (FDC) mode, every time the accumulator overflows, the output is toggled. This provides a 50% duty cycle at half the $F_{OVERFLOW}$ frequency, provided that the increment value remains constant. For more information, see the figure below.

The FDC mode is selected by clearing the PFM bit.

**Figure 32-2. FDC Output Mode Timing Diagram**




## 32.3 Pulse Frequency Mode

In Pulse Frequency (PF) mode, the output becomes active on the rising clock edge immediately following the overflow event, and goes inactive 1 to 128 clock periods later, determined by the PWS bits. This provides a pulsed output at the $F_{OVERFLOW}$ frequency. For more information, refer to the figure above.

**Important:** When the selected pulse width is greater than the accumulator overflow time frame, then the NCO output does not toggle.

The level of the Active and Inactive states is determined by the POL bit.

PF mode is selected by setting the PFM bit.

## 32.4 Output Polarity Control

The last stage in the NCO module is the output polarity. The POL bit selects the output polarity. The active level of the Pulse Frequency mode is high true when the POL bit is cleared.

Changing the polarity while the interrupts are enabled will cause an interrupt for the resulting output transition.

The NCO output signal (NCOx_out) is available by internal routing to several other peripherals.

## 32.5 Interrupts

When the accumulator overflows, the NCO Interrupt Flag bit, NCOxIF, in the associated PIR register is set. To enable interrupt service on this event, the following bits must be set:

- EN bit
- NCOxIE bit in the associated PIE register
- Peripheral and Global Interrupt Enable bits

The interrupt must be cleared by software by clearing the NCOxIF bit in the Interrupt Service Routine.

## 32.6 Effects of a Reset

All of the NCO registers are cleared to zero as the result of any Reset.

## 32.7 Operation in Sleep

The NCO module operates independently from the system clock and will continue to run during Sleep, provided that the clock source selected remains active.

The HFINTOSC remains active during Sleep when the NCO module is enabled and the HFINTOSC is selected as the clock source, regardless of the system clock source selected.

In other words, if the HFINTOSC is simultaneously selected as the system clock and the NCO clock source, when the NCO is enabled, the CPU will go Idle during Sleep, but the NCO will continue to operate and the HFINTOSC will remain active.

With a clock running, it will have a direct effect on the Sleep mode current.

## 32.8 Register Definitions: NCO

Long bit name prefixes for the NCO peripherals are shown in the table below. Refer to the **“Long Bit Names”** section in the **“Register and Bit Naming Conventions”** chapter for more information.

**Table 32-1. NCO Long Bit Name Prefixes**

| Peripheral | Bit Name Prefix |
|---|---|
| NCO1 | NCO1 |
| NCO2 | NCO2 |
| NCO3 | NCO3 |

### 32.8.1 NCOxCON

**Name:** NCOxCON
**Address:** 0x446,0x44E,0x456

NCO Control Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | EN | | OUT | POL | | | | PFM |
| Access | R/W | | R | R/W | | | | R/W |
| Reset | 0 | | 0 | 0 | | | | 0 |

**Bit 7 – EN** NCO Enable

| Value | Description |
|---|---|
| 1 | NCO module is enabled |
| 0 | NCO module is disabled |

**Bit 5 – OUT** NCO Output

Displays the current logic level of the NCO module output.

**Bit 4 – POL** NCO Polarity

| Value | Description |
|---|---|
| 1 | NCO output signal is inverted |
| 0 | NCO output signal is not inverted |

**Bit 0 – PFM** NCO Pulse Frequency Mode

| Value | Description |
|---|---|
| 1 | NCO operates in Pulse Frequency mode. Output frequency is $F_{OVERFLOW}$. |
| 0 | NCO operates in Fixed Duty Cycle mode. Output frequency is $F_{OVERFLOW}$ divided by 2. |

### 32.8.2 NCOxCLK

**Name:** NCOxCLK
**Address:** 0x447,0x44F,0x457

NCO Input Clock Control Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | PWS[2:0] | | | CKS[4:0] | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 7:5 – PWS[2:0]** NCO Output Pulse-Width Select(1)

| Value | Description |
|---|---|
| 111 | NCO output is active for 128 input clock periods |
| 110 | NCO output is active for 64 input clock periods |
| 101 | NCO output is active for 32 input clock periods |
| 100 | NCO output is active for 16 input clock periods |
| 011 | NCO output is active for 8 input clock periods |
| 010 | NCO output is active for 4 input clock periods |
| 001 | NCO output is active for 2 input clock periods |
| 000 | NCO output is active for 1 input clock periods |

**Bits 4:0 – CKS[4:0]** NCO Clock Source Select

| CKS | Clock Source | | | Active in Sleep |
|---|---|---|---|---|
| Value | NCO1 | NCO2 | NC03 | |
| 11111-11011 | Reserved | | | - |
| 11010 | CLC8_OUT | | | No |
| 11001 | CLC7_out | | | No |
| 11000 | CLC6_out | | | No |
| 10111 | CLC5_OUT | | | No |
| 10110 | CLC4_OUT | | | No |
| 10101 | CLC3_out | | | No |
| 10100 | CLC2_OUT | | | No |
| 10011 | CLC1_OUT | | | No |
| 10010 | NCO3_OUT | NCO3_OUT | Reserved | No |
| 10001 | NCO2_OUT | Reserved | NCO2_OUT | No |
| 10000 | Reserved | NCO1_OUT | NCO1_OUT | No |
| 01111-01011 | Reserved | | | - |
| 01010 | TMR6_OUT | | | No |
| 01001 | TMR4_OUT | | | No |
| 01000 | TMR2_OUT | | | No |
| 00111 | CLKREF | | | No |
| 00110 | EXTOSC | | | Yes |
| 00101 | SOSC | | | Yes |
| 00100 | MFINTOSC (31.25 kHz) | | | Yes |
| 00011 | MFINTOSC (500 kHz) | | | Yes |
| 00010 | LFINTOSC | | | Yes |
| 00001 | HFINTOSC | | | Yes |
| 00000 | $F_{OSC}$ | | | No |

**Note:**

1. PWS applies only when operating in Pulse Frequency mode.

### 32.8.3 NCOxACC

**Name:** NCOxACC
**Address:** 0x440,0x448,0x450

NCO Accumulator Register

| Bit | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|
| | | | | | ACC[19:16] | | | |
| Access | | | | | R/W | R/W | R/W | R/W |
| Reset | | | | | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | ACC[15:8] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | ACC[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 19:0 – ACC[19:0]** Accumulated sum of NCO additions

**Notes:**

1. The individual bytes in this multibyte register can be accessed with the following register names:
   - NCOxACCU: Accesses the upper byte ACC[23:16]
   - NCOxACCH: Accesses the high byte ACC[15:8]
   - NCOxACCL: Accesses the low byte ACC[7:0].
2. The accumulator spans registers NCOxACCU:NCOxACCH:NCOxACCL. The 24 bits are reserved, but not all are used. This register updates in real-time, asynchronously to the CPU; there is no provision to ensure atomic access to this 24-bit space using an 8-bit bus. Writing to this register while the module is operating will produce undefined results.

© 2019-2023 Microchip Technology Inc. and its subsidiaries

### 32.8.4 NCOxINC

**Name:** NCOxINC
**Address:** 0x443,0x44B,0x453

NCO Increment Register

| Bit | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|
| | | | | | INC[19:16] | | | |
| Access | | | | | R/W | R/W | R/W | R/W |
| Reset | | | | | 0 | 0 | 0 | 0 |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | INC[15:8] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | INC[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

**Bits 19:0 – INC[19:0]** Value by which the NCOxACC is increased by each NCO clock

**Notes:**

1. The individual bytes in this multibyte register can be accessed with the following register names:
   – NCOxINCU: Accesses the upper byte INC[19:16]
   – NCOxINCH: Accesses the high byte INC[15:8]
   – NCOxINCL: Accesses the low byte INC[7:0].
2. The logical increment spans NCOxINCU:NCOxINCH:NCOxINCL.
3. NCOxINC is double-buffered as INCBUF:
   – INCBUF is updated on the next falling edge of NCOxCLK after writing to NCOxINCL
   – NCOxINCU and NCOxINCH will be written prior to writing NCOxINCL.

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

## 32.9 Register Summary - NCO

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0440 | NCO1ACC | 7:0 | ACC[7:0] | | | | | | | |
| | | 15:8 | ACC[15:8] | | | | | | | |
| | | 23:16 | | | | | ACC[19:16] | | | |
| 0x0443 | NCO1INC | 7:0 | INC[7:0] | | | | | | | |
| | | 15:8 | INC[15:8] | | | | | | | |
| | | 23:16 | | | | | INC[19:16] | | | |
| 0x0446 | NCO1CON | 7:0 | EN | | OUT | POL | | | | PFM |
| 0x0447 | NCO1CLK | 7:0 | | PWS[2:0] | | | CKS[4:0] | | | |
| 0x0448 | NCO2ACC | 7:0 | ACC[7:0] | | | | | | | |
| | | 15:8 | ACC[15:8] | | | | | | | |
| | | 23:16 | | | | | ACC[19:16] | | | |
| 0x044B | NCO2INC | 7:0 | INC[7:0] | | | | | | | |
| | | 15:8 | INC[15:8] | | | | | | | |
| | | 23:16 | | | | | INC[19:16] | | | |
| 0x044E | NCO2CON | 7:0 | EN | | OUT | POL | | | | PFM |
| 0x044F | NCO2CLK | 7:0 | | PWS[2:0] | | | CKS[4:0] | | | |
| 0x0450 | NCO3ACC | 7:0 | ACC[7:0] | | | | | | | |
| | | 15:8 | ACC[15:8] | | | | | | | |
| | | 23:16 | | | | | ACC[19:16] | | | |
| 0x0453 | NCO3INC | 7:0 | INC[7:0] | | | | | | | |
| | | 15:8 | INC[15:8] | | | | | | | |
| | | 23:16 | | | | | INC[19:16] | | | |
| 0x0456 | NCO3CON | 7:0 | EN | | OUT | POL | | | | PFM |
| 0x0457 | NCO3CLK | 7:0 | | PWS[2:0] | | | CKS[4:0] | | | |

# 33. DSM - Data Signal Modulator Module

The Data Signal Modulator (DSM) is a peripheral that allows the user to mix a data stream, also known as a modulator signal, with a carrier signal to produce a modulated output. Both the carrier and the modulator signals are supplied to the DSM module either internally, from the output of a peripheral, or externally through an input pin. The modulated output signal is generated by performing a logical “AND” operation of both the carrier and modulator signals, and then provided to the DSM_out pin.

The carrier signal is comprised of two distinct and separate signals. A Carrier High (CARH) signal and a Carrier Low (CARL) signal. During the time in which the modulator (MOD) signal is in a Logic High state, the DSM mixes the CARH signal with the modulator signal. When the modulator signal is in a Logic Low state, the DSM mixes the CARL signal with the modulator signal.

Using this method, the DSM can generate the following types of key modulation schemes:

- Frequency Shift Keying (FSK)
- Phase-Shift Keying (PSK)
- ON-OFF Keying (OOK)

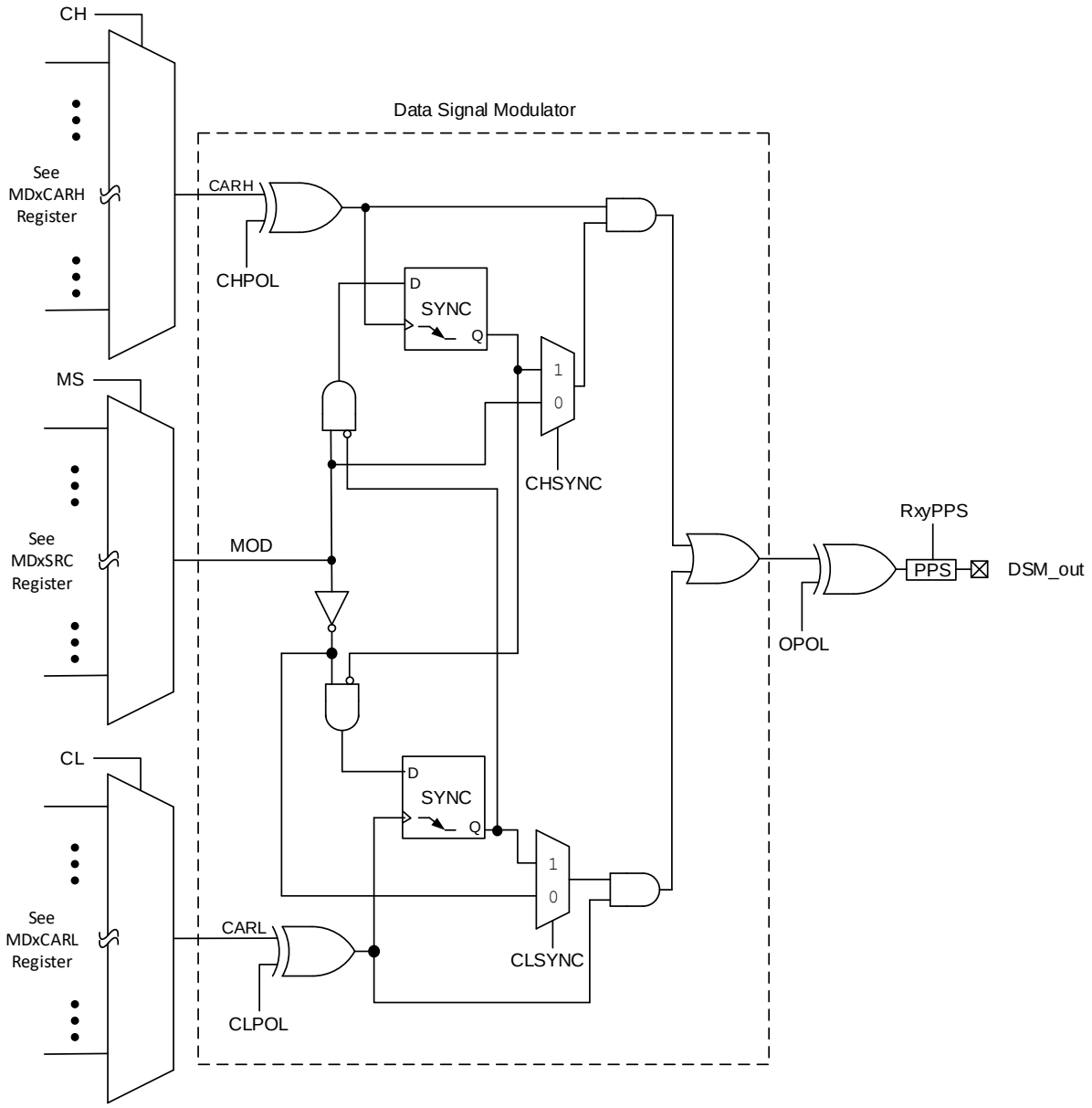
Additionally, the following features are provided within the DSM module:

- Carrier Synchronization
- Carrier Source Polarity Select
- Programmable Modulator Data
- Modulated Output Polarity Select
- Peripheral Module Disable, which provides the ability to place the DSM module in the lowest power consumption mode

The figure below shows a simplified block diagram of the data signal modulator peripheral.

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

**Figure 33-1. Simplified Block Diagram of the Data Signal Modulator**




## 33.1 DSM Operation

The DSM module is enabled by setting the EN bit. Clearing the EN bit disables the output of the module, but retains the carrier and source signal selections. The module will resume operation when the EN bit is set again. The output of the DSM module can be rerouted to several pins using the PPS output source selection register. When the EN bit is cleared the output pin is held low.

### 33.1.1 Modulator Signal Sources

The modulator signal can be supplied from several different sources, and is selected by configuring the MS bits.

### 33.1.2 Carrier Signal Sources

The carrier high signal and carrier low signal can be supplied from several different sources, and is selected by the CH bits and CL bits, respectively.

## 33.2 Carrier Synchronization

During the time when the DSM switches between carrier high and carrier low signal sources, the carrier data in the modulated output signal can become truncated. To prevent this, the carrier signal can be synchronized to the modulator signal. When synchronization is enabled, the carrier pulse that is being mixed at the time of the transition is allowed to transition low before the DSM switches over to the next carrier source.

Synchronization is enabled separately for the carrier high and carrier low signal sources. Synchronization for the carrier high signal is enabled by setting the CHSYNC bit. Synchronization for the carrier low signal is enabled by setting the CLSYNC bit. The figures below show the timing diagrams of using various synchronization methods.

**Figure 33-2. On-Off Keying (OOK) Synchronization**

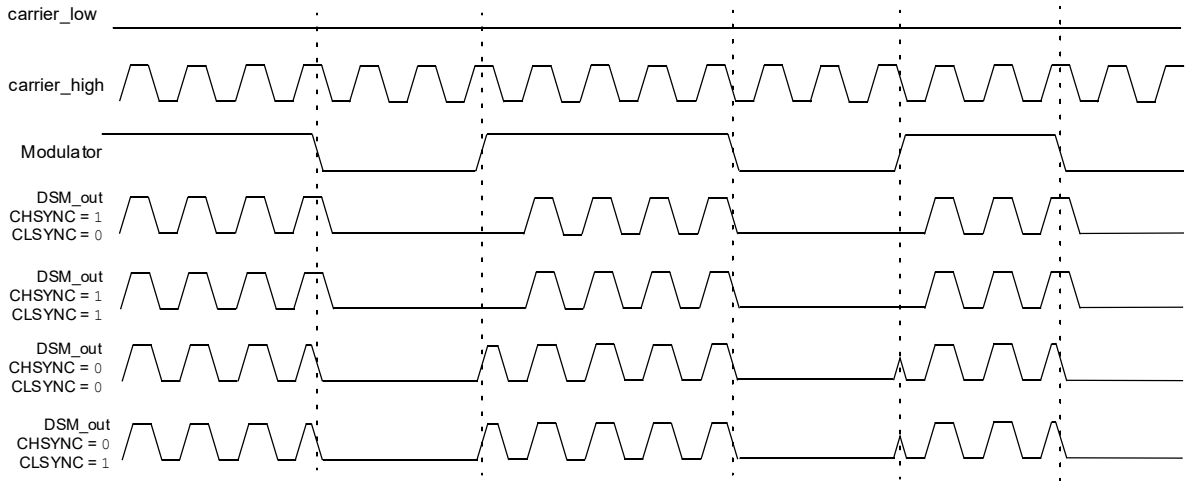



**Figure 33-3. No Synchronization (CHSYNC = 0, CLSYNC = 0)**

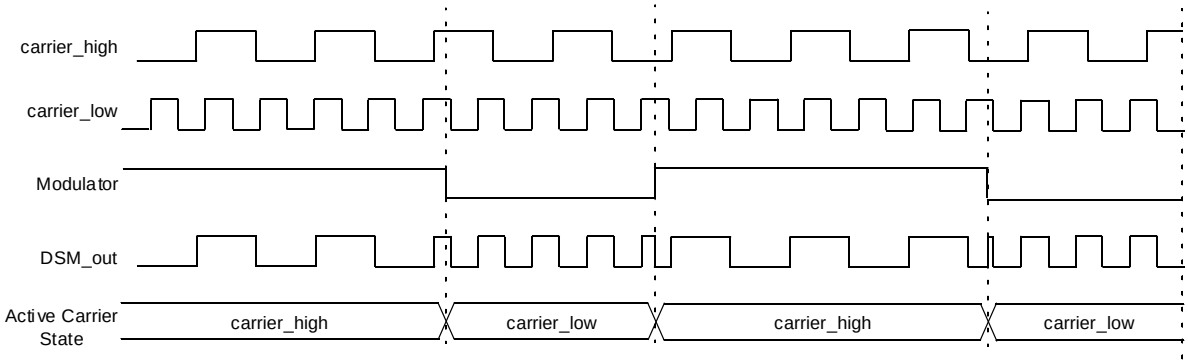


**Figure 33-4. Carrier High Synchronization (CHSYNC = 1, CLSYNC = 0)**

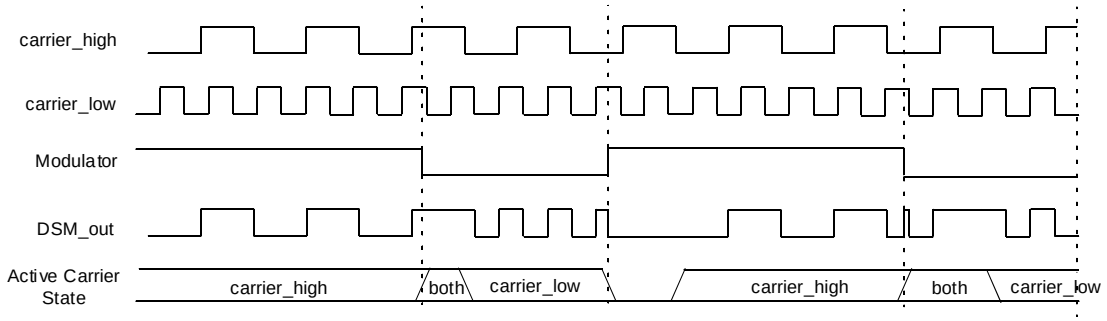



**Figure 33-5. Carrier Low Synchronization (CHSYNC = 0, CLSYNC = 1)**

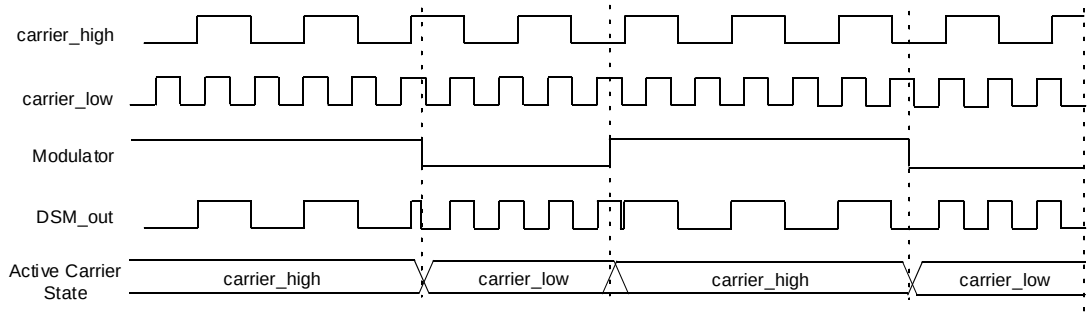
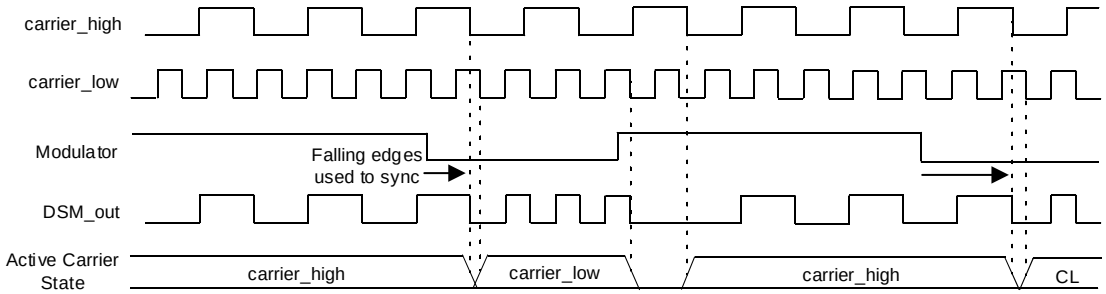



**Figure 33-6. Full Synchronization (CHSYNC = 1, CLSYNC = 1)**




## 33.3 Carrier Source Polarity Select

The signal provided from any selected input source for the carrier high and carrier low signals can be inverted. Inverting the signal for the carrier high and low source is enabled by setting the CHPOL bit and the CLPOL bit, respectively.

## 33.4 Programmable Modulator Data

The BIT control bit can used to generate the modulation signal. This gives the user the ability to provide software driven modulation.

## 33.5 Modulated Output Polarity

The modulated output signal provided on the DSM_out pin can also be inverted. Inverting the modulated output signal is enabled by setting the OPOL bit.

## 33.6 Operation in Sleep Mode

The DSM can operate during Sleep, if the carrier and modulator input sources are also operable during Sleep. Refer to the **"Power-Saving Modes"** chapter for more details.

## 33.7 Effects of a Reset

Upon any device Reset, the DSM module is disabled. The user's firmware is responsible for initializing the module before enabling the output. All the registers are reset to their default values.

## 33.8 Peripheral Module Disable

The DSM module can be completely disabled using the PMD module to achieve maximum power saving. When the DSMMD bit of the PMD registers is set, the DSM module is completely disabled. This puts the module in its lowest power consumption state. When enabled again all the registers of the DSM module default to POR status.

## 33.9 Register Definitions: Modulation Control

Long bit name prefixes for the modulation control peripherals are shown in the table below. Refer to the **"Long Bit Names"** section in the **"Register and Bit Naming Conventions"** chapter for more information.

**Table 33-1. Modulation Control Long Bit Name Prefixes**

| Peripheral | Bit Name Prefix |
|---|---|
| DSM1 | MD1 |

### 33.9.1 MDxCON0

**Name:** MDxCON0
**Address:** 0x6A

Modulation Control Register 0

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | EN | | OUT | OPOL | | | | BIT |
| Access | R/W | | R/W | R/W | | | | R/W |
| Reset | 0 | | 0 | 0 | | | | 0 |

**Bit 7 – EN** Modulator Module Enable

| Value | Description |
|---|---|
| 1 | DSM is enabled and mixing input signals |
| 0 | DSM is disabled and has no output |

**Bit 5 – OUT** Modulator Output(1)
Displays the current DSM_out value

**Bit 4 – OPOL** Modulator Output Polarity Select

| Value | Description |
|---|---|
| 1 | DSM output signal is inverted; idle high output |
| 0 | DSM output signal is not inverted; idle low output |

**Bit 0 – BIT** Modulation Source Signal(2)
Allows direct software control of the modulation signal

**Notes:**

1. The modulated output frequency can be greater and asynchronous from the clock that updates this register bit. The bit value may not be valid for higher speed modulator or carrier signals.
2. MDBIT must be selected as the modulation source in the MDxSRC register for this operation.

### 33.9.2 MDxCON1

**Name:** MDxCON1
**Address:** 0x6B

Modulation Control Register 1

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | CHPOL | CHSYNC | | | CLPOL | CLSYNC |
| Access | | | R/W | R/W | | | R/W | R/W |
| Reset | | | 0 | 0 | | | 0 | 0 |

**Bit 5 – CHPOL** Modulator High Carrier Polarity Select

| Value | Description |
|---|---|
| 1 | Selected high carrier signal is inverted |
| 0 | Selected high carrier signal is not inverted |

**Bit 4 – CHSYNC** Modulator High Carrier Synchronization Enable

| Value | Description |
|---|---|
| 1 | Modulator waits for a falling edge on the high time carrier signal before allowing a switch to the low time carrier |
| 0 | Modulator output is not synchronized to the high time carrier signal(1) |

**Bit 1 – CLPOL** Modulator Low Carrier Polarity Select

| Value | Description |
|---|---|
| 1 | Selected low carrier signal is inverted |
| 0 | Selected low carrier signal is not inverted |

**Bit 0 – CLSYNC** Modulator Low Carrier Synchronization Enable

| Value | Description |
|---|---|
| 1 | Modulator waits for a falling edge on the low time carrier signal before allowing a switch to the high time carrier |
| 0 | Modulator output is not synchronized to the low time carrier signal(1) |

**Note:**

1. Narrowed carrier pulse widths or spurs may occur in the signal stream if the carrier is not synchronized.

### 33.9.3 MDxCARH

**Name:** MDxCARH
**Address:** 0x6E

Modulation High Carrier Control Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | CH[4:0] | | | | |
| Access | | | | R/W | R/W | R/W | R/W | R/W |
| Reset | | | | 0 | 0 | 0 | 0 | 0 |

**Bits 4:0 – CH[4:0]** Modulator Carrier High Selection

| CH | Connection |
|---|---|
| 11111-10111 | Reserved |
| 10110 | CLC8_OUT |
| 10101 | CLC7_OUT |
| 10100 | CLC6_OUT |
| 10011 | CLC5_OUT |
| 10010 | CLC4_OUT |
| 10001 | CLC3_OUT |
| 10000 | CLC2_OUT |
| 01111 | CLC1_OUT |
| 01110 | NCO3_OUT |
| 01101 | NCO2_OUT |
| 01100 | NCO1_OUT |
| 01011 | Reserved |
| 01010 | PWM3S1P1_OUT |
| 01001 | PWM2S1P1_OUT |
| 01000 | PWM1S1P1_OUT |
| 00111 | CCP3_OUT |
| 00110 | CCP2_OUT |
| 00101 | CCP1_OUT |
| 00100 | CLKREF_OUT |
| 00011 | EXTOSC |
| 00010 | HFINTOSC |
| 00001 | $F_{OSC}$ (System Clock) |
| 00000 | Pin selected by MDCARHPPS |

### 33.9.4 MDxCARL

**Name:** MDxCARL
**Address:** 0x6D

Modulation Low Carrier Control Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | CL[4:0] | | | | |
| Access | | | | R/W | R/W | R/W | R/W | R/W |
| Reset | | | | 0 | 0 | 0 | 0 | 0 |

**Bits 4:0 – CL[4:0]** Modulator Carrier Low Input Selection

| CL | Connection |
|---|---|
| 11111-10111 | Reserved |
| 10110 | CLC8_OUT |
| 10101 | CLC7_OUT |
| 10100 | CLC6_OUT |
| 10011 | CLC5_OUT |
| 10010 | CLC4_OUT |
| 10001 | CLC3_OUT |
| 10000 | CLC2_OUT |
| 01111 | CLC1_OUT |
| 01110 | NCO3_OUT |
| 01101 | NCO2_OUT |
| 01100 | NCO1_OUT |
| 01011 | Reserved |
| 01010 | PWM3S1P2_OUT |
| 01001 | PWM2S1P2_OUT |
| 01000 | PWM1S1P2_OUT |
| 00111 | CCP3_OUT |
| 00110 | CCP2_OUT |
| 00101 | CCP1_OUT |
| 00100 | CLKREF_OUT |
| 00011 | EXTOSC |
| 00010 | HFINTOSC |
| 00001 | $F_{OSC}$ (System Clock) |
| 00000 | Pin selected by MDCARLPPS |

### 33.9.5 MDxSRC

**Name:** MDxSRC
**Address:** 0x6C

Modulation Source Control Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | MS[5:0] | | | | | |
| Access | | | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | | | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 5:0 – MS[5:0]** Modulator Source Selection

| MS | Connection |
|---|---|
| 111111-100000 | Reserved |
| 100000 | SPI2_SDO |
| 011111 | SPI1_SDO |
| 011110 | UART5_TX |
| 011101 | UART4_TX |
| 011100 | UART3_TX |
| 011011 | UART2_TX |
| 011010 | UART1_TX |
| 011001 | CLC8_OUT |
| 011000 | CLC7_OUT |
| 010111 | CLC6_OUT |
| 010110 | CLC5_OUT |
| 010101 | CLC4_OUT |
| 010100 | CLC3_OUT |
| 010011 | CLC2_OUT |
| 010010 | CLC1_OUT |
| 010001 | CMP2_OUT |
| 010000 | CMP1_OUT |
| 001111 | NCO3_OUT |
| 001110 | NCO2_OUT |
| 001101 | NCO1_OUT |
| 001100 | Reserved |
| 001011 | Reserved |
| 001010 | PWM3S1P2_OUT |
| 001001 | PWM3S1P1_OUT |
| 001000 | PWM2S1P2_OUT |
| 000111 | PWM2S1P1_OUT |
| 000110 | PWM1S1P2_OUT |
| 000101 | PWM1S1P1_OUT |
| 000100 | CCP3_OUT |
| 000011 | CCP2_OUT |
| 000010 | CCP1_OUT |
| 000001 | MDBIT |
| 000000 | Pin selected by MDSRCPPS |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

## 33.10 Register Summary - DSM

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x6A | MD1CON0 | 7:0 | EN | | OUT | OPOL | | | | BIT |
| 0x6B | MD1CON1 | 7:0 | | | CHPOL | CHSYNC | | | CLPOL | CLSYNC |
| 0x6C | MD1SRC | 7:0 | | | MS[5:0] | | | | | |
| 0x6D | MD1CARL | 7:0 | | | | CL[4:0] | | | | |
| 0x6E | MD1CARH | 7:0 | | | | CH[4:0] | | | | |

# 34. UART - Universal Asynchronous Receiver Transmitter with Protocol Support

The Universal Asynchronous Receiver Transmitter (UART) module is a serial I/O communications peripheral. It contains all the clock generators, shift registers and data buffers necessary to perform an input or output serial data transfer, independent of device program execution. The UART, also known as a Serial Communications Interface (SCI), can be configured as a full-duplex asynchronous system or one of several automated protocols. The Full Duplex mode is useful for communications with peripheral systems, such as wireless modems and USB to serial interface modules.

Supported protocols include:

- LIN Host and Client
- DMX Controller and Receiver
- DALI Control Gear and Control Device

The UART module includes the following capabilities:

- Half and full-duplex asynchronous transmit and receive
- Two-byte input buffer
- One-byte output buffer
- Programmable 7-bit or 8-bit byte width
- 9th bit address detection
- 9th bit even or odd parity
- Input buffer overrun error detection
- Receive framing error detection
- Hardware and software flow control
- Automatic checksum calculation and verification
- Programmable 1, 1.5, and 2 Stop bits
- Programmable data polarity
- Manchester encoder/decoder
- Operation in Sleep
- Automatic detection and calibration of the baud rate
- Wake-up on Break reception
- Automatic and user timed Break period generation
- RX and TX inactivity time-outs (with Timer2)

The operation of the UART module is controlled through 19 8-bit registers:

- Three control registers (UxCON0-UxCON2)
- Error enable and status (UxERRIE, UxERRIR, UxUIR)
- UART buffer status and control (UxFIFO)
- Three 9-bit protocol parameters (UxP1-UxP3)
- 16-bit Baud Rate Generator (UxBRG)
- Transmit buffer write (UxTXB)
- Receive buffer read (UxRXB)
- Receive checksum (UxRXCHK)
- Transmit checksum (UxTXCHK)

The UART transmit output (TX_out) is available to the TX pin and internally to various peripherals.

Block diagrams of the UART transmitter and receiver are shown in the following figures.

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

**Figure 34-1. UART Transmitter Block Diagram**

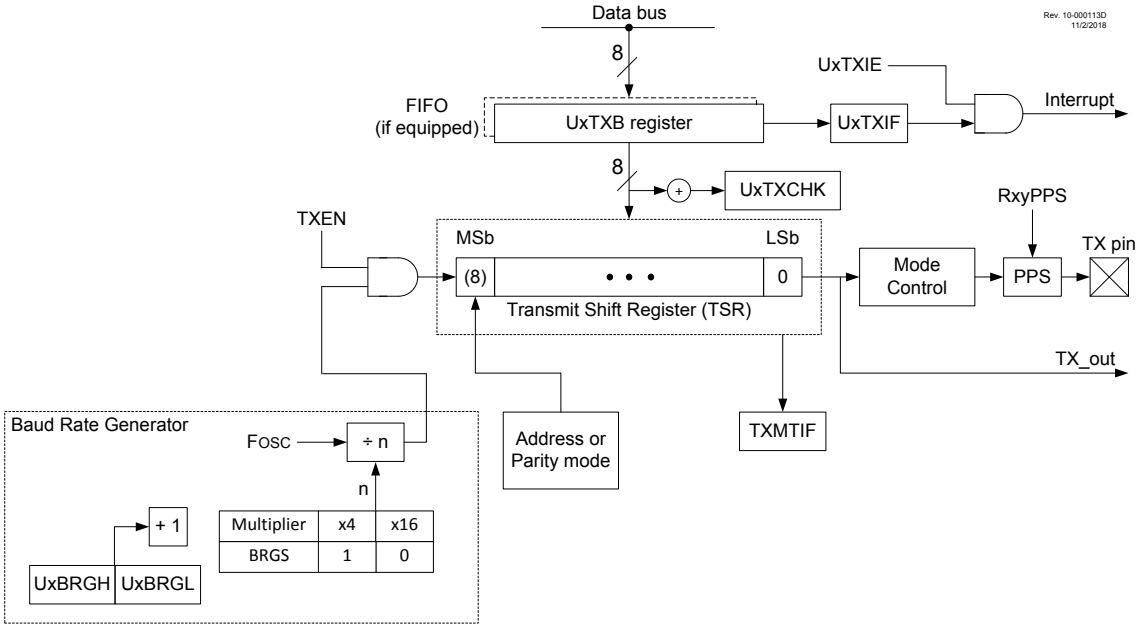
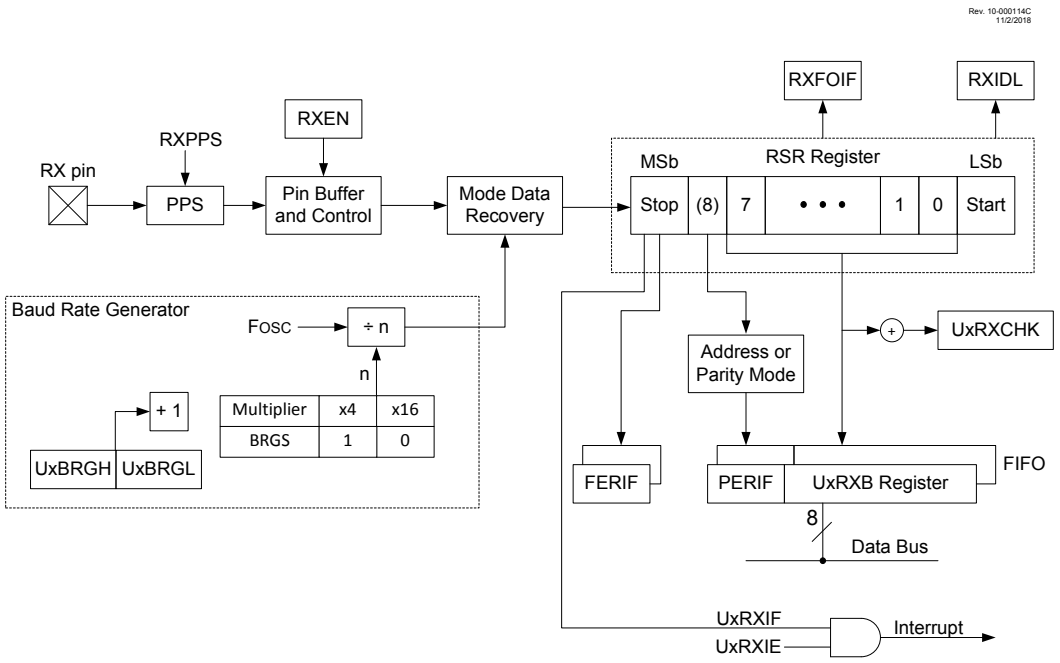



**Figure 34-2. UART Receiver Block Diagram**




## 34.1 UART I/O Pin Configuration

The RX input pin is selected with the UxRPPS register. The TX output pin is selected with each pin's RxyPPS register. When the TRIS control for the pin corresponding to the TX output is cleared, the UART will control the logic level on the TX pin. Changing the TXPOL bit in UxCON2 will immediately change the TX pin logic level, regardless of the value of EN or TXEN.

## 34.2 UART Asynchronous Modes

The UART has five Asynchronous modes:

- 7-bit
- 8-bit
- 8-bit with even parity in the 9th bit
- 8-bit with odd parity in the 9th bit
- 8-bit with address indicator in the 9th bit

The UART transmits and receives data using the standard Non-Return-to-Zero (NRZ) format. NRZ is implemented with two levels: A VOH Mark state, which represents a '1' data bit, and a VOL Space state, which represents a '0' data bit. NRZ implies that consecutively transmitted data bits of the same value stay at the output level of that bit without returning to a neutral level between each bit transmission. An NRZ transmission port idles in the Mark state. Each character transmission consists of one Start bit followed by seven or eight data bits, one optional parity or address bit, and is always terminated by one or more Stop bits. The Start bit is always a space and the Stop bits are always marks. The most common data format is eight bits with no parity. Each transmitted bit persists for a period of 1/ (Baud Rate). An on-chip dedicated 16-bit Baud Rate Generator is used to derive standard baud rate frequencies from the system oscillator. See the UART Baud Rate Generator section for more information.

In all Asynchronous modes, the UART transmits and receives the LSb first. The UART's transmitter and receiver are functionally independent, but share the same data format and baud rate. Parity is supported by the hardware with even and odd parity modes.

### 34.2.1 UART Asynchronous Transmitter

The UART transmitter block diagram is shown in Figure 34-1. The heart of the transmitter is the serial Transmit Shift Register (TSR), which is not directly accessible by software. The TSR obtains its data from the transmit buffer, which is the UxTXB register.

#### 34.2.1.1 Enabling the Transmitter

The UART transmitter is enabled for asynchronous operations by configuring the following control bits:

- TXEN = 1
- MODE = 0000 through 0011
- UxBRG = desired baud rate
- BRGS = desired baud rate multiplier
- RxyPPS = code for desired output pin
- ON = 1

All other UART control bits are assumed to be in their default state.

Setting the TXEN bit enables the transmitter circuitry of the UART. The MODE bits select the desired mode. Setting the ON bit enables the UART. When TXEN is set and the transmitter is not Idle, the TX pin is automatically configured as an output. When the transmitter is Idle, the TX pin drive is relinquished to the port TRIS control. If the TX pin is shared with an analog peripheral, the analog I/O function will be disabled by clearing the corresponding ANSEL bit.

**Important:** The UxTXIF Transmitter Interrupt flag is set when the TXEN Enable bit is set and the UxTXB register can accept data.

#### 34.2.1.2 Transmitting Data

A transmission is initiated by writing a character to the UxTXB register. If this is the first character, or the previous character has been completely transmitted from the TSR, the data in the UxTXB is immediately transferred to the TSR register. If the TSR still contains all or part of a previous character, the new character data is held in the UxTXB until the previous character transmission is complete. The pending character in the UxTXB is then transferred to the TSR at the beginning of the previous character Stop bit transmission. The transmission of the Start bit, data bits and Stop bit sequence commences immediately following the completion of all of the previous character's Stop bits.

© 2019-2023 Microchip Technology Inc. and its subsidiaries

#### 34.2.1.3 Transmit Data Polarity

The polarity of the transmit data is controlled with the TXPOL bit. The default state of this bit is '0', which selects high true transmit Idle and data bits. Setting the TXPOL bit to '1' will invert the transmit data, resulting in low true Idle and data bits. The TXPOL bit controls transmit data polarity in all modes.

#### 34.2.1.4 Transmit Interrupt Flag

The UxTXIF Interrupt Flag bit in the PIR register is set whenever the UART transmitter is enabled and no character is being held for transmission in the UxTXB register. In other words, the UxTXIF bit is clear only when the TSR is busy with a character and a new character has been queued for transmission in the UxTXB register.

The UxTXIF interrupt is enabled by setting the UxTXIE Interrupt Enable bit in the PIE register. However, the UxTXIF Flag bit will be set whenever the UxTXB register is empty, regardless of the state of the UxTXIE Enable bit. The UxTXIF bit is read-only and cannot be set or cleared by software.

To use interrupts when transmitting data, set the UxTXIE bit only when there is more data to send. Clear the UxTXIE Interrupt Enable bit upon writing the UxTXB register with the last character of the transmission.

#### 34.2.1.5 TSR Status

The TXMTIF bit indicates the status of the TSR. This is a read-only bit. The TXMTIF bit is set when the TSR is empty and Idle. The TXMTIF bit is cleared when a character is transferred to the TSR from the UxTXB. The TXMTIF bit remains clear until all bits, including the Stop bits, have been shifted out of the TSR and a byte is not waiting in the UxTXB register.

The TXMTIF will generate a summary UxEIF interrupt when the TXMTIE bit is set.

**Important:** The TSR is not mapped in data memory, so it is not available to the user.

#### 34.2.1.6 Transmitter 7-Bit Mode

The 7-bit mode is selected when the MODE bits are set to '0001'. In 7-bit mode, only the seven Least Significant bits of the data written to UxTXB are transmitted. The Most Significant bit is ignored.

#### 34.2.1.7 Transmitter Parity Modes

When Odd or Even Parity mode is selected, all data is sent as nine bits. The first eight bits are data and the 9th bit is parity. Even and odd parity is selected when the MODE bits are set to '0011' and '0010', respectively. Parity is automatically determined by the module and inserted in the serial data stream.

#### 34.2.1.8 Asynchronous Transmission Setup

Use the following steps as a guide for configuring the UART for asynchronous transmissions.

1. Initialize the UxBRG register pair and the BRGS bit to achieve the desired baud rate.
2. Set the MODE bits to the desired Asynchronous mode.
3. Set the TXPOL bit if inverted TX output is desired.
4. Enable the asynchronous serial port by setting the ON bit.
5. Enable the transmitter by setting the TXEN Control bit. This will cause the UxTXIF Interrupt flag to be set.
6. If the device has PPS, configure the desired I/O pin RxyPPS register with the code for the TX output.
7. If interrupts are desired, set the UxTXIE Interrupt Enable bit in the respective PIE register. An interrupt will occur immediately provided that global interrupts are also enabled.
8. Write one byte of data into the UxTXB register. This will start the transmission.
9. Subsequent bytes may be written when the UxTXIF bit is '1'.

© 2019-2023 Microchip Technology Inc. and its subsidiaries

**Figure 34-3. UART Asynchronous Transmission**

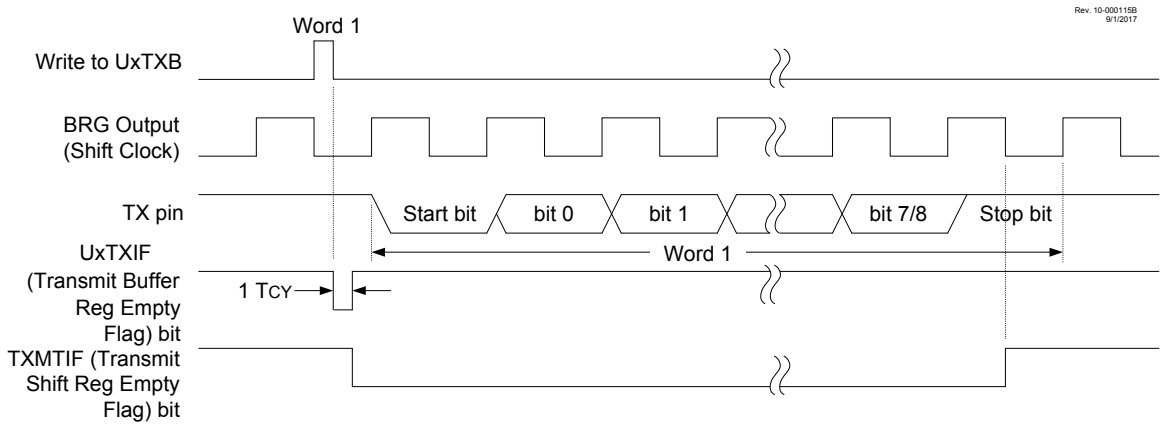
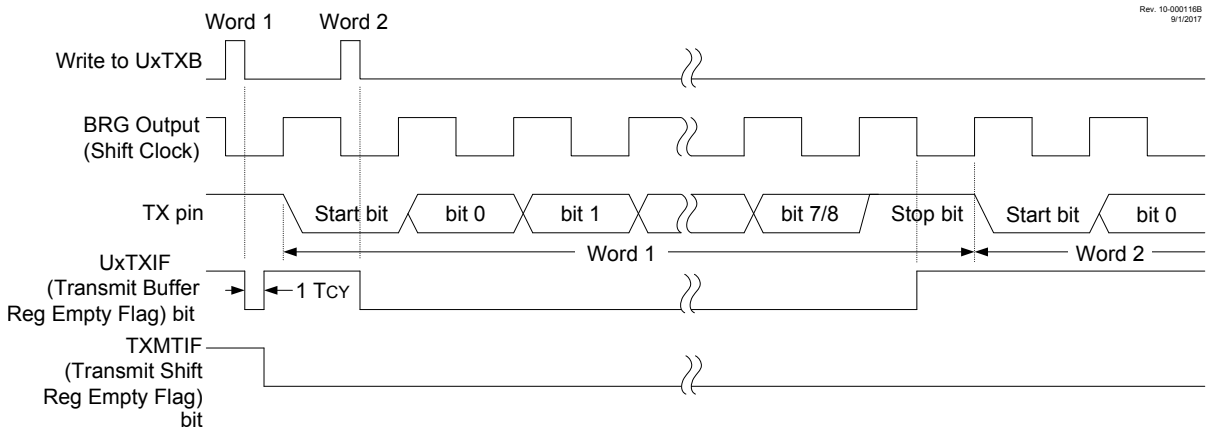



**Figure 34-4. UART Asynchronous Transmission (Back-to-Back)**




### 34.2.2 UART Asynchronous Receiver

The Asynchronous mode is typically used in RS-232 systems. The receiver block diagram is shown in Figure 34-2. The data is received on the RX pin and drives the data recovery block. The data recovery block is actually a high-speed shifter operating at 4 or 16 times the baud rate, whereas the serial Receive Shift Register (RSR) operates at the bit rate. When all bits of the character have been shifted in, they are immediately transferred to a two-character First-In First-Out (FIFO) memory. The FIFO buffering allows reception of two complete characters and the start of a third character before software must begin servicing the UART receiver. The FIFO registers and RSR are not directly accessible by software. Access to the received data is made via the UxRXB register.

#### 34.2.2.1 Enabling the Receiver

The UART receiver is enabled for asynchronous operation by configuring the following control bits:

- RXEN = 1
- MODE = 0000 through 0011
- UxBRG = desired baud rate
- BRGS = desired baud rate multiplier
- RXPPS = code for desired input pin
- Input pin ANSEL bit = 0
- ON = 1

All other UART control bits are assumed to be in their default state.

Setting the RXEN bit enables the receiver circuitry of the UART. Setting the MODE bits configures the UART for the desired Asynchronous mode. Setting the ON bit enables the UART. The TRIS bit corresponding to the selected RX I/O pin must be set to configure the pin as an input.

**Important:** If the RX function is on an analog pin, the corresponding ANSEL bit must be cleared for the receiver to function.

#### 34.2.2.2 Receiving Data

Data is recovered from the bit stream by timing to the center of the bits and sampling the input level. In High Speed mode, there are four BRG clocks per bit and only one sample is taken per bit. In Normal Speed mode, there are 16 BRG clocks per bit and three samples are taken per bit.

The receiver data recovery circuit initiates character reception on the falling edge of the Start bit. The Start bit is always a '0'. The Start bit is qualified in the middle of the bit. In Normal Speed mode only, the Start bit is also qualified at the leading edge of the bit. The following paragraphs describe the majority-detect sampling of the Normal Speed mode without inverted polarity.

The falling edge starts the Baud Rate Generator (BRG) clock. The input is sampled at the first and second BRG clocks.

If both samples are high, then the falling edge is deemed a glitch and the UART returns to the Start bit detection state without generating an error.

If either sample is low, the data recovery circuit continues counting BRG clocks and takes samples at clock counts: 7, 8 and 9. When less than two samples are low, the Start bit is deemed invalid and the data recovery circuit aborts character reception, without generating an error, and resumes looking for the falling edge of the Start bit.

When two or more samples are low, the Start bit is deemed valid and the data recovery continues. After a valid Start bit is detected, the BRG clock counter continues and resets at count 16. This is the beginning of the first data bit.

The data recovery circuit counts the BRG clocks from the beginning of the bit and takes samples at clocks 7, 8 and 9. The bit value is determined from the majority of the samples. The resulting '0' or '1' is shifted into the RSR. The BRG clock counter continues and resets at count 16. This sequence repeats until all data bits have been sampled and shifted into the RSR.

After all data bits have been shifted in, the first Stop bit is sampled. Stop bits are always a '1'. If the bit sampling determines that a '0' is in the Stop bit position, the framing error is set for this character. Otherwise, the framing error is cleared for this character. See the Receive Framing Error section for more information on framing errors.

#### 34.2.2.3 Receive Data Polarity

The polarity of the receive data is controlled with the RXPOL bit. The default state of this bit is '0', which selects high true receive Idle and data bits. Setting the RXPOL bit to '1' will invert the receive data, resulting in low true Idle and data bits. The RXPOL bit controls receive data polarity in all modes.

#### 34.2.2.4 Receive Interrupts

Immediately after all data bits and the Stop bit have been received, the character in the RSR is transferred to the UART receive FIFO. The UxRXIF Interrupt flag in the respective PIR register is set at this time, provided it is not being suppressed.

The UxRXIF is suppressed by any of the following:

- FERIF when FERIE is set
- PERIF when PERIE is set

When the UART uses DMA for reception, suppressing the UxRXIF suspends the DMA transfer of data until software processes the error and reads UxRXB to advance the FIFO beyond the error.

The UxRXIF interrupts are enabled by setting all of the following bits:

- UxRXIE, Interrupt Enable bit in the PIE register
- Global Interrupt Enable bits

© 2019-2023 Microchip Technology Inc. and its subsidiaries

The UxRXIF Interrupt Flag bit will be set when it is not suppressed and there is an unread character in the FIFO, regardless of the state of interrupt enable bits. Reading the UxRXB register will transfer the top character out of the FIFO and reduce the FIFO contents by one. The UxRXIF Interrupt Flag bit is read-only and therefore cannot be set or cleared by software.

#### 34.2.2.5 Receive Framing Error

Each character in the receive FIFO buffer has a corresponding Framing Error Flag bit. A framing error indicates that the Stop bit was not seen at the expected time. For example, a Break condition will be received as a 0x00 byte with the framing error bit set.

The Framing Error flag is accessed via the FERIF bit. The FERIF bit represents the frame status of the top unread character of the receive FIFO. Therefore, the FERIF bit must be read before reading UxRXB.

The FERIF bit is read-only and only applies to the top unread character of the receive FIFO. A framing error (FERIF = 1) does not preclude reception of additional characters. It is neither necessary nor possible to clear the FERIF bit directly. Reading the next character from the FIFO buffer will advance the FIFO to the next character and the next corresponding framing error, if any.

The FERIF bit is cleared when the character at the top of the FIFO does not have a framing error or when all bytes in the receive FIFO have been read. Clearing the ON bit resets the receive FIFO, thereby also clearing the FERIF bit.

A framing error will generate a summary UxEIF interrupt when the FERIE bit is set. The summary error is reset when the FERIF bit of the top of the FIFO is '0' or when all FIFO characters have been retrieved.

**Important:** When FERIE is set, UxRXIF interrupts are suppressed by FERIF = 1.

#### 34.2.2.6 Receiver Parity Modes

Even or odd parity is automatically detected when the MODE bits are set to '0011' or '0010', respectively. The parity modes receive eight data bits and one parity bit for a total of nine bits for each character. The PERIF bit represents the parity error of the top unread character of the receive FIFO rather than the parity bit itself. The parity error must be read before the UxRXB register is read because reading the UxRXB register will advance the FIFO pointer to the next byte with its associated PERIF flag.

A parity error will generate a summary UxEIF interrupt when the PERIE bit is set. The summary error is reset when the PERIF bit of the top of the FIFO is '0' or when all FIFO characters have been retrieved.

**Important:** When PERIE is set, the UxRXIF interrupts are suppressed by PERIF = 1.

#### 34.2.2.7 Receive FIFO Overflow

When more characters are received than the receive FIFO can hold, the RXFOIF bit is set. The character causing the Overflow condition is discarded. The RUNOVF bit determines how the receive circuit responds to characters while the Overflow condition persists. When RUNOVF is set, the receive shifter stays synchronized to the incoming data stream by responding to Start, data, and Stop bits. However, all received bytes not already in the FIFO are discarded. When RUNOVF is cleared, the receive shifter ceases operation and Start, data, and Stop bits are ignored. The Receive Overflow condition is cleared by reading the UxRXB register and clearing the RXFOIF bit. If the UxRXB register is not read, thereby opening a space in the FIFO, the next character received will be discarded and cause another Overflow condition.

A receive overflow error will generate a summary UxEIF interrupt when the RXFOIE bit is set.

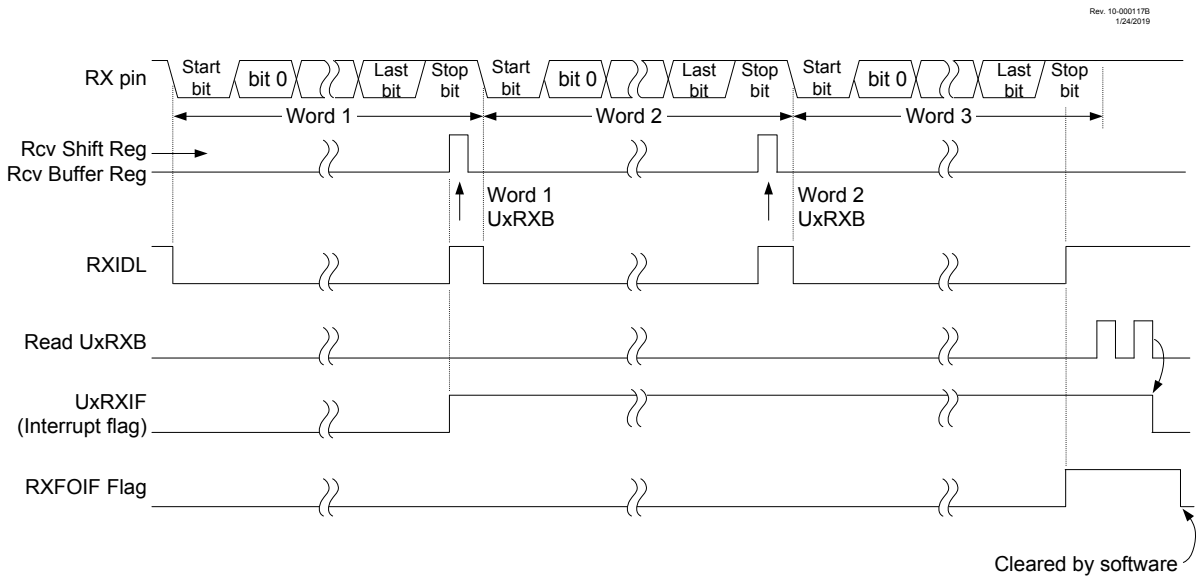
#### 34.2.2.8 Asynchronous Reception Setup

Use the following steps as a guide for configuring the UART for asynchronous reception:

1. Initialize the UxBRG register pair and the BRGS bit to achieve the desired baud rate.

2. Configure the RXPPS register for the desired RX pin.
3. Clear the ANSEL bit for the RX pin (if applicable).
4. Set the MODE bits to the desired Asynchronous mode.
5. Set the RXPOL bit if the data stream is inverted.
6. Enable the serial port by setting the ON bit.
7. If interrupts are desired, set the UxRXIE bit in the PIEx register and enable global interrupts.
8. Enable reception by setting the RXEN bit.
9. Read the UxERRIR register to get the error flags.
10. The UxRXIF Interrupt Flag bit will be set when a character is transferred from the RSR to the receive buffer. An interrupt will be generated if the UxRXIE interrupt enable bit is also set.
11. Read the UxRXB register to get the received byte.
12. If an overrun occurred, clear the RXFOIF bit.

**Figure 34-5. UART Asynchronous Reception**




**Note:** This timing diagram shows three bytes appearing on the RX input. The UxRXB is not read before the third word is received, causing the RXFOIF (FIFO overrun) bit to be set. STPMD = 0, STP = 00.

### 34.2.3 Asynchronous Address Mode

A special Address Detection mode is available for use when multiple receivers share the same transmission line, as seen in RS-485 systems.

When Asynchronous Address mode is enabled, all data is transmitted and received as 9-bit characters. The 9th bit determines whether the character is address or data. When the 9th bit is set, the eight Least Significant bits are the address. When the 9th bit is clear, the Least Significant bits are data. In either case, the 9th bit is stored in PERIF when the byte is written to the receive FIFO. When PERIE is also set, the RXIF will be suppressed, thereby suspending DMA transfers allowing software to process the received address.

An address character will enable all receivers that match the address and disable all other receivers. Once a receiver is enabled, all non-address characters will be received until an address character that does not match is received.

#### 34.2.3.1 Address Mode Transmit

The UART transmitter is enabled for asynchronous address operation by configuring the following control bits:

- TXEN = 1
- MODE = 0100
- UxBRG = desired baud rate

- BRGS = desired baud rate multiplier
- RxyPPS = code for desired output pin
- ON = 1

Addresses are sent by writing to the UxP1L register. This transmits the written byte with the 9th bit set, which indicates that the byte is an address.

Data is sent by writing to the UxTXB register. This transmits the written byte with the 9th bit cleared, which indicates that the byte is data.

To send data to a particular device on the transmission bus, first transmit the address of the intended device. All subsequent data will be accepted only by that device until an address of another device is transmitted.

Writes to UxP1L take precedence over writes to UxTXB. When both the UxP1L and UxTXB registers are written while the TSR is busy, the next byte to be transmitted will be from UxP1L.

To ensure all data intended for one device are sent before the address is changed, wait until the TXMTIF bit is high before writing UxP1L with the new address.

#### 34.2.3.2 Address Mode Receive

The UART receiver is enabled for asynchronous address operation by configuring the following control bits:

- RXEN = 1
- MODE = 0100
- UxBRG = desired baud rate
- BRGS = desired baud rate multiplier
- RXPPS = code for desired input pin
- Input pin ANSEL bit = 0
- UxP2L = receiver address
- UxP3L = address mask
- ON = 1

In Address mode, no data will be transferred to the input FIFO until a valid address is received. This is the default state. Any of the following conditions will cause the UART to revert to the default state:

- ON = 0
- RXEN = 0
- Received address does not match

When a character with the 9th bit set is received, the Least Significant eight bits of that character will be qualified by the values in the UxP2L and UxP3L registers.

The byte is XORed with UxP2L then ANDed with UxP3L. A match occurs when the result is 0h, in which case, the unaltered received character is stored in the receive FIFO, thereby setting the UxRXIF Interrupt bit. The 9th bit is stored in the corresponding PERIF bit, identifying this byte as an address.

An address match also enables the receiver for all data such that all subsequent characters without the 9th bit set will be stored in the receive FIFO.

When the 9th bit is set and a match does not occur, the character is not stored in the receive FIFO and all subsequent data is ignored.

The UxP3L register mask allows a range of addresses to be accepted. Software can then determine the sub-address of the range by processing the received address character.

## 34.3 DMX Mode (Full-Featured UARTs Only)

DMX is a protocol used in stage and show equipment. This includes lighting, fog machines, motors, etc. The protocol consists of a controller that sends out commands, and a receiver such as theater lights that receive these commands. The DMX protocol is usually unidirectional, but can be a bidirectional protocol in either Half or Full Duplex mode. An example of a Half Duplex mode is the RDM (Remote Device Management) protocol that sits on DMX512A.

© 2019-2023 Microchip Technology Inc. and its subsidiaries

The controller transmits commands and the receiver receives them. There are no Error conditions or retransmit mechanisms.

DMX, or DMX512A, consists of a “universe” of 512 channels. This means that one controller can output up to 512 bytes on a single DMX link. Each piece of equipment on the line is programmed to listen to a consecutive sequence of one or more of these bytes.

For example, a fog machine connected to one of the universes may be programmed to receive one byte, starting at byte number 10, and a lighting unit may be programmed to receive four bytes starting at byte number 22.

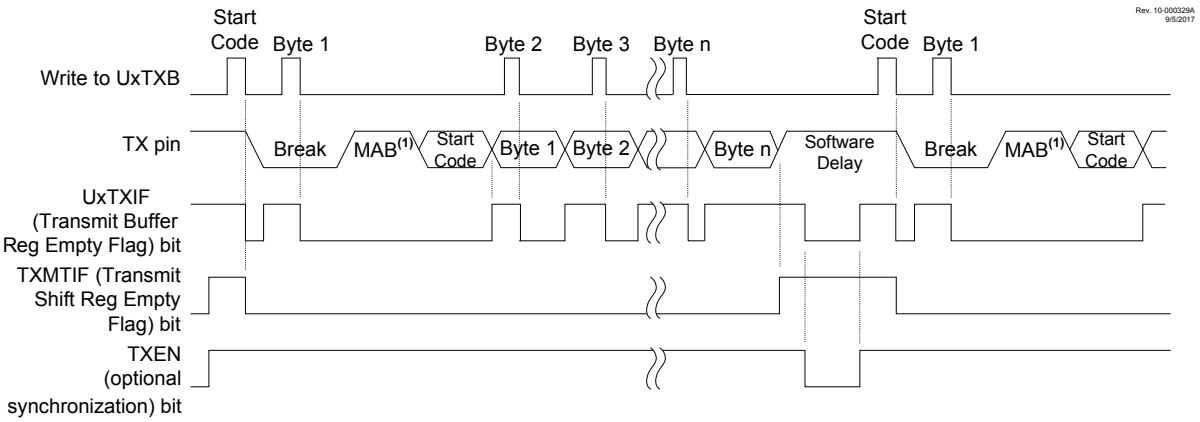
### 34.3.1 DMX Controller

The DMX Controller mode is configured with the following settings:

- MODE = 1010
- TXEN = 1
- RXEN = 0
- TXPOL = 0
- UxP1 = one less than the number of bytes to transmit (excluding the Start code)
- UxBRG = value to achieve 250K baud rate
- STP = 10 for two Stop bits
- RxyPPS = TX pin output code
- ON = 1

Each DMX transmission begins with a Break followed by a byte called the “Start Code”. The width of the Break is fixed at 25 bit times. The Break is followed by a “Mark After Break” (MAB) Idle period. After this Idle period, the first through the ‘n’th byte is transmitted, where ‘n-1’ is the value in UxP1. See the following figure.

**Figure 34-6. DMX Transmit Sequence**




**Note:** 1. The MAB period is fixed at 3 bit times.

Software sends the Start Code and the ‘n’ data bytes by writing the UxTXB register with each byte to be sent in the desired order. A UxTXIF value of ‘1’ indicates when the UxTXB is ready to accept the next byte.

The internal byte counter is not accessible to software. Software needs to keep track of the number of bytes written to UxTXB to ensure that no more and no less than ‘n’ bytes are sent because the DMX state machine will automatically insert a Break and reset its internal counter after ‘n’ bytes are written. One way to ensure synchronization between hardware and software is to toggle TXEN after the last byte of the universe is completely free of the transmit shift register, as indicated by the TXMTIF bit.

### 34.3.2 DMX Receiver

The DMX Receiver mode is configured with the following settings:

- MODE = 1010

- TXEN = 0
- RXEN = 1
- RXPOL = 0
- UxP2 = number of first byte to receive
- UxP3 = number of last byte to receive
- UxBRG = value to achieve 250K baud rate
- STP = 10 for two Stop bits
- ON = 1
- UxRXPPS = code for desired input pin
- Input pin ANSEL bit = 0

When configured as a DMX Receiver, the UART listens for a Break character that is at least 23 bit periods wide. If the Break is shorter than 23 bit times, the Break is ignored and the DMX state machine remains in Idle mode. Upon receiving the Break, the DMX counters will be reset to align with the incoming data stream. Immediately after the Break, the UART will see the “Mark after Break” (MAB). This space is ignored by the UART. The Start Code follows the MAB and will always be stored in the receive FIFO.

After the Start Code, the first through the 512th byte will be received, but not all of them are stored in the receive FIFO. The UART ignores all received bytes until the ones of interest are received. This is done using the UxP2 and UxP3 registers. The UxP2 register holds the value of the byte number to start the receive process. The byte counter starts at ‘0’ for the first byte after the Start Code. For example, to receive four bytes starting at the 10th byte after the Start Code, write 009h (9 decimal) to UxP2H:L and 00Ch (12 decimal) to UxP3H:L. The receive FIFO depth is limited, therefore the bytes must be retrieved by reading UxRXB as they come in to avoid a receive FIFO Overrun condition.

Typically, two Stop bits are inserted between bytes. If either Stop bit is detected as a ‘0’, the framing error for that byte will be set.

Since the DMX sequence always starts with a Break, the software can verify that it is in sync with the sequence by monitoring the RXBKIF flag to ensure that the next byte received after the RXBKIF flag is processed as the Start Code and subsequent bytes are processed as the expected data.

## 34.4 LIN Modes (Full-Featured UARTs Only)

LIN is a protocol used primarily in automotive applications. The LIN network consists of two kinds of software processes: A Host process and a Client process. Each network has only one Host process and one or more Client processes.

From a physical layer point of view, the UART on one processor may be driven by both a Host and a Client process, as long as only one Host process exists on the network.

A LIN transaction consists of a Host process followed by a Client process. The Client process may involve more than one client where one is transmitting and the other(s) receiving. The transaction begins by the following Host process transmission sequence:

1. Break.
2. Delimiter bit.
3. Sync Field.
4. PID byte.

The PID determines which Client processes are expected to respond to the host. When the PID byte is complete, the TX output remains in the Idle state. One or more of the Client processes may respond to the Host process. If no one responds within the inter-byte period, the host is free to start another transmission. The inter-byte period is timed by software using a means other than the UART.

The Client process follows the Host process. When the client software recognizes the PID, that Client process responds by either transmitting the required response or by receiving the transmitted data. Only Client processes send data. Therefore, Client processes receiving data are receiving that of another Client process.

When a client sends data, the client UART automatically calculates the checksum for the transmitted bytes as they are sent and appends the inverted checksum byte to the client response.

© 2019-2023 Microchip Technology Inc. and its subsidiaries

When a client receives data, the checksum is accumulated on each byte as it is received using the same algorithm as the sending process. The last byte, which is the inverted checksum value calculated by the sending process, is added to the locally calculated checksum by the UART. The check passes when the result is all '1's, otherwise the check fails and the CERIF bit is set.

Two methods for computing the checksum are available: legacy and enhanced. The legacy checksum includes only the data bytes. The enhanced checksum includes the PID and the data. The C0EN control bit determines the checksum method. Setting C0EN to '1' selects the enhanced method. Software must select the appropriate method before the Start bit of the checksum byte is received.

### 34.4.1 LIN Host/Client Mode

The LIN Host mode includes capabilities to generate client processes. The host process stops at the PID transmission. Any data that is transmitted in Host/Client mode is done as a client process. LIN Host/Client mode is configured by the following settings:

- MODE = 1100
- TXEN = 1
- RXEN = 1
- UxBRG = value to achieve desired baud rate
- TXPOL = 0 (for high Idle state)
- STP = desired Stop bits selection
- C0EN = desired Checksum mode
- RxyPPS = TX pin selection code
- TX pin TRIS control = 0
- ON = 1

**Important:** The TXEN bit must be set before the Host process is received and remain set while in LIN mode whether or not the Client process is a transmitter.

The Host process is started by writing the PID to the UxP1L register when UxP2 is '0' and the UART is Idle. The UxTXIF will not be set in this case. Only the six Least Significant bits of UxP1L are used in the PID transmission.

The two Most Significant bits of the transmitted PID are PID parity bits. PID[6] is the exclusive-or of PID bits 0, 1, 2 and 4. PID[7] is the inverse of the exclusive-or of PID bits 1, 3, 4 and 5.

The UART hardware calculates and inserts these bits in the serial stream.

Writing UxP1L automatically clears the UxTXCHK and UxRXCHK registers and generates the Break, the delimiter bit, the Sync character (55h), and the PID transmission portion of the transaction. The data portion of the transaction that follows, if there is one, is a Client process. See the LIN Client Mode section for more details of that process. The host receives its own PID if RXEN is set. Software performs the Client process corresponding to the PID that was sent and received. Attempting to write UxP1L before an active Host process is complete will not succeed. Instead, the TXWRE bit will be set.

### 34.4.2 LIN Client Mode

The LIN Client mode is configured by the following settings:

- MODE = 1011
- TXEN = 1
- RXEN = 1
- UxP2 = number of data bytes to transmit
- UxP3 = number of data bytes to receive
- UxBRG = value to achieve default baud rate
- TXPOL = 0 (for high Idle state)
- STP = desired Stop bits selection

- C0EN = desired Checksum mode
- RxyPPS = TX pin selection code
- TX pin TRIS control = 0
- ON = 1

The Client process starts upon detecting a Break on the RX pin. The Break clears the UxTXCHK, UxRXCHK, UxP2 and UxP3 registers. At the end of the Break, the auto-baud circuity is activated and the baud rate is automatically set using the Sync character following the Break. The character following the Sync character is received as the PID code and is saved in the receive FIFO. The UART computes the two PID parity bits from the six Least Significant bits of the PID. If either parity bit does not match the corresponding bit of the received PID code, the PERIF flag is set and saved at the same FIFO location as the PID code. The UxRXIF bit is set indicating that the PID is available.

Software retrieves the PID by reading the UxRXB register and determines the Client process to execute from that. The checksum method, number of data bytes, and whether to send or receive data are defined by software according to the PID code.

#### 34.4.2.1 LIN Client Receiver

When the Client process is a Receiver, the software performs the following tasks:

- The UxP3 register is written with a value equal to the number of data bytes to receive
- The C0EN bit is set or cleared to select the appropriate checksum. This must be completed before the Start bit of the checksum byte is received.
- Each byte of the process response is read from UxRXB when UxRXIF is set

The UART updates the checksum on each received byte. When the last data byte is received, the computed checksum total is stored in the UxRXCHK register. The next received byte is saved in the receive FIFO and added with the value in UxRXCHK. The result of this addition is not accessible. However, if the result is not all '1's, the CERIF bit is set. The CERIF flag persists until cleared by software. Software needs to read UxRXB to remove the checksum byte from the FIFO, but the byte can be discarded if not needed for any other purpose.

After the checksum is received, the UART ignores all activity on the RX pin until a Break starts the next transaction.

#### 34.4.2.2 LIN Client Transmitter

When the Client process is a transmitter, software performs the following tasks in the order shown:

- The UxP2 register is written with a value equal to the number of bytes to transmit. This will enable the UxTXIF flag which is disabled when UxP2 is '0'.
- The C0EN bit is set or cleared to select the appropriate checksum
- Each byte of the process response is written to UxTXB when UxTXIF is set

The UART accumulates the checksum as each byte is written to UxTXB. After the last byte is written, the UART stores the calculated checksum in the UxTXCHK register and transmits the inverted result as the last byte in the response.

The UxTXIF flag is disabled when the number of bytes specified by the value in the UxP2 register have been written. Any writes to UxTXB that exceed the UxP2 count will be ignored and set the TXWRE flag.

## 34.5 DALI Mode (Full-Featured UARTs Only)

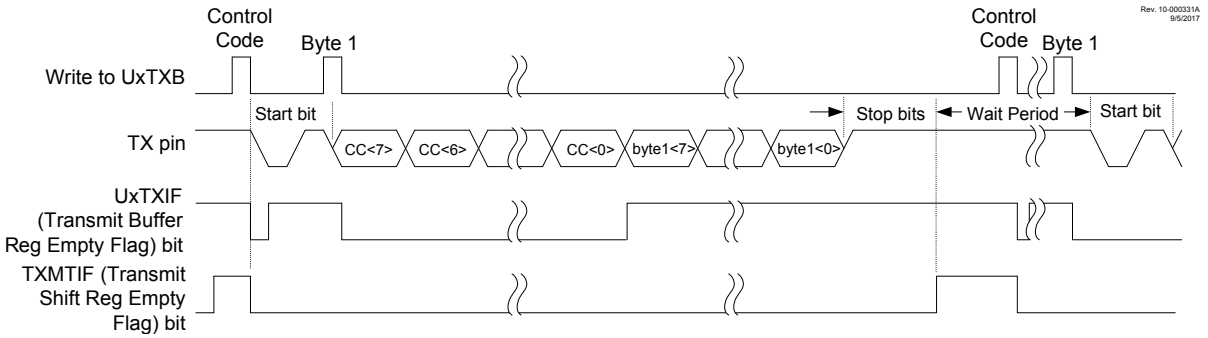
DALI is a protocol used for intelligent lighting control for building automation. The protocol consists of Control Devices and Control Gear. A Control Device is an application controller that sends out commands to the light fixtures. The light fixture itself is termed as a Control Gear. The communication is done using Manchester encoding, which is performed by the UART hardware.

Manchester encoding consists of the clock and data in a single bit stream (refer to Figure 34-9). A high-to-low or a low-to-high transition always occurs in the middle of the bit period and may or may not occur at the bit period boundaries. When the consecutive bits in the bit stream are of the same value (i.e., consecutive '1's or consecutive '0's) a transition occurs at the bit boundary. However, when the bit value changes, there is no transition at the bit boundary. According to the standard, a half-bit time is typically 416.7 µs long. A double half-bit time or a single bit is typically 833.3 µs.

The protocol is inherently half-duplex. Communication over the bus occurs in the form of forward and backward frames. Wait times between the frames are defined in the standard to prevent collision between the frames.

A Control Device transmission is termed as the forward frame. In the DALI 2.0 standard, a forward frame can be two or three bytes in length. The two-byte forward frame is used for communication between Control Device and Control Gear whereas the three-byte forward frame is used for communication between Control Devices on the bus. The first byte in the forward frame is the control byte and is followed by either one or two data bytes. The transaction begins when the Control Device starts a transmission. Unlike other protocols, each byte in the frame is transmitted MSb first. Typical frame timing is shown below.

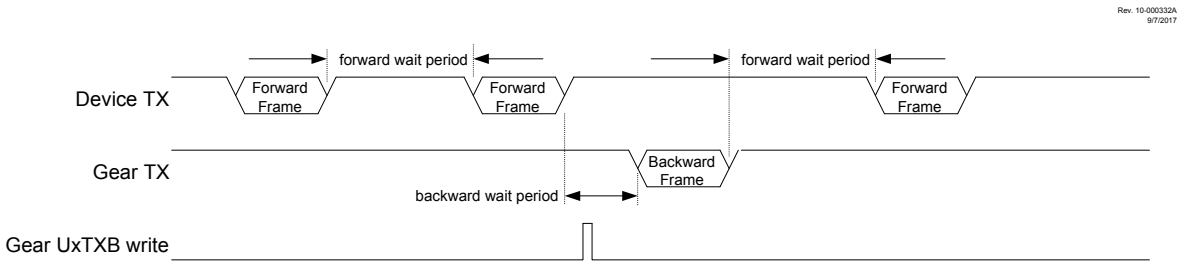
**Figure 34-7. DALI Frame Timing**




During the communication between two Control Devices, three bytes are required to be transmitted. In this case, the software must write the third byte to UxTXB as soon as UxTXIF goes true and before the output shifter becomes empty. This ensures that the three bytes of the forward frame are transmitted back-to-back without any interruption.

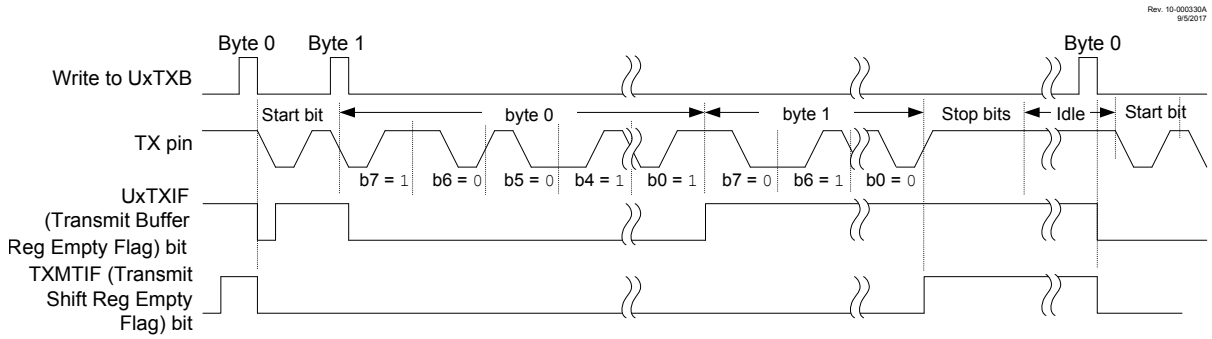
All Control Gear on the bus receive the forward frame. If the forward frame requires a reply to be sent, one of the Control Gear may respond with a single byte, called the backward frame. The 2.0 standard requires the Control Gear to begin transmission of the backward frame between 5.5 ms to 10.5 ms (~14 to 22 half-bit times) after reception of the forward frame. Once the backward frame is received by the Control Device, it is required to wait a minimum of 2.4 ms (~6 half-bit times). After this wait time, the Control Device is free to transmit another forward frame. Refer to the figure below.

**Figure 34-8. DALI Forward/Backward Frame Timing**




A Start bit is used to indicate the start of the forward and backward frames. When ABDEN = 0, the receiver bit rate is determined by the BRG register. When ABDEN = 1, the first bit synchronizes the receiver with the transmitter and sets the receiver bit rate. The low period of the Start bit is measured and is used as the timing reference for all data bits in the forward and backward frames. The ABDOVF bit is set if the Start bit low period causes the measurement counter to overflow. All the bits following the Start bit are data bits. The bit stream terminates when no transition is detected in the middle of a bit period. Refer to the figure below.

**Figure 34-9. Manchester Timing**




The forward and backward frames are terminated by two Idle bit periods or Stop bits. Normally, these start in the first bit period of a byte. If both Stop bits are valid, the byte reception is terminated.

If either of the Stop bits is invalid, the frame is tagged as invalid by saving it as a null byte and setting the framing error in the receive FIFO.

A framing error also occurs when no transition is detected on the bus in the middle of a bit period when the byte reception is not complete. In such a scenario, the byte will be saved with the FERIF bit set.

### 34.5.1 Control Device

The Control Device mode is configured with the following settings:

- MODE = `'b1000`
- TXEN = `1`
- RXEN = `1`
- UxP1 = forward frames are held for transmission with this number of half-bit periods after the completion of a forward or backward frame
- UxP2 = forward/backward frame threshold delimiter. Any reception that starts this number of half-bit periods after the completion of a forward or backward frame is detected as forward frame and sets the PERIF flag of the corresponding received byte.
- UxBRG = value to achieve 1200 baud rate
- TXPOL = appropriate polarity for interface circuit
- STP = `'b10` for two Stop bits
- RxyPPS = TX pin selection code
- TX pin TRIS control = `0`
- ON = `1`

A forward frame is initiated by writing the control byte to the UxTXB register. After sending the control byte, each data byte must be written to the UxTXB register as soon as UxTXIF goes true. It is necessary to perform every write after UxTXIF goes true, to ensure that the transmit buffer is ready to accept the byte. Each write must also occur before the TXMTIF bit goes true, to ensure that the bit stream of the forward frame is generated without interruption.

When TXMTIF goes true, indicating the transmit shift register has completed sending the last byte in the frame, the TX output is held in Idle state for the number of half-bit periods selected by the STP bits.

After the last Stop bit, the TX output is held in the Idle state for an additional wait time determined by the half-bit period count in the UxP1 register. For example, a 2450 µs delay (~6 half-bit times) requires a value of 6 in UxP1L.

Any writes to the UxTXB register that occur after TXMTIF goes true, but before the UxP1 wait time expires, are held and then transmitted immediately following the wait time. If a backward frame is received during the wait time, any bytes that may have been written to UxTXB will be transmitted after completion of the backward frame reception plus the UxP1 wait time.

The wait timer is reset by the backward frame and starts over immediately following the reception of the Stop bits of the backward frame. Data pending in the transmit shift register will be sent when the wait time elapses.

© 2019-2023 Microchip Technology Inc. and its subsidiaries

To replace or delete any pending forward frame data, the TXBE bit needs to be set to flush the shift register and transmit buffer. A new control byte can then be written to the UxTXB register. The control byte will be held in the buffer and sent at the beginning of the next forward frame following the UxP1 wait time.

In Control Device mode, PERIF is set when a forward frame is received. This helps the software to determine whether the received byte is part of a forward frame from a Control Device (either from the Control Device under consideration or from another Control Device on the bus) or a backward frame from a Control Gear.

### 34.5.2 Control Gear

The Control Gear mode is configured with the following settings:

- MODE = `'b1001`
- TXEN = `1`
- RXEN = `1`
- UxP1 = back frames are held for transmission with this number of half-bit periods after the completion of a forward frame
- UxP2 = forward/back frame threshold delimiter. Idle periods longer than this number of half-bit periods are detected as forward frames.
- UxBRG = value to achieve 1200 baud rate
- TXPOL = appropriate polarity for interface circuit
- RXPOL = same as TXPOL
- STP = `'b10` for two Stop bits
- RxyPPS = TX pin output code
- TX pin TRIS control = `0`
- RXPPS = RX pin selection code
- RX pin TRIS control = `1`
- Input pin ANSEL bit = `0`
- ON = `1`

The UART starts listening for a forward frame when the Control Gear mode is entered. Only the frames that follow an Idle period longer than UxP2 half-bit periods are detected as forward frames. Backward frames from other Control Gear are ignored. Only forward frames will be stored in UxRXB. This is necessary because a backward frame can be sent only as a response to a forward frame.

The forward frame is received one byte at a time in the receive FIFO and retrieved by reading the UxRXB register. The end of the forward frame starts a timer to delay the backward frame response by a wait time equal to the number of half-bit periods stored in UxP1.

The data received in the forward frame is processed by the application software. If the application decides to send a backward frame in response to the forward frame, the value of the backward frame is written to UxTXB. This value is held for transmission in the transmit shift register until the wait time expires, being transmitted afterwards.

If the backward frame data is written to UxTXB after the wait time has expired, it is held in the UxTXB register until the end of the wait time following the next forward frame. The TXMTIF bit is false when the backward frame data is held in the transmit shift register. Receiving a UxRXIF interrupt before the TXMTIF goes true indicates that the backward frame write was too late and another forward frame was received before sending the backward frame. The pending backward frame is flushed by setting the TXBE bit to prevent it from being sent after the next forward frame.

## 34.6 General Purpose Manchester (Full-Featured UARTs Only)

General purpose Manchester is a subset of the DALI mode. When the UxP1L register is cleared, there is no minimum wait time between frames. This allows full and half-duplex operation because writes to the UxTXB register are not held waiting for a receive operation to complete.

General purpose Manchester operation maintains all other aspects of DALI mode as shown in Figure 34-9 such as:

- Single-pulse Start bit
- Most Significant bit first

- No stop periods between back-to-back bytes

The general purpose Manchester mode is configured with the following settings:

- MODE = 'b1000
- TXEN = 1
- RXEN = 1
- UxP1 = 0h
- UxBRG = desired baud rate
- TXPOL and RXPOL = desired Idle state
- STP = desired number of stop periods
- RxyPPS = TX pin selection code
- TX pin TRIS control = 0
- RXPPS = RX pin selection code
- RX pin TRIS control = 1
- Input pin ANSEL bit = 0
- ON = 1

The Manchester bit stream timing is shown in Figure 34-9.

## 34.7 Polarity

Receive and transmit polarity is user selectable and affects all modes of operation.

The idle level is programmable with the TXPOL and RXPOL polarity control bits. Both control bits default to '0', which selects a high idle level for transmit and receive. The low level Idle state is selected by setting the control bit to '1'. TXPOL controls the TX idle level. RXPOL controls the RX idle level.

## 34.8 Stop Bits

The number of Stop bits is user selectable with the STP bits. The STP bits affect all modes of operation.

Stop bits selections are shown in the table below:

**Table 34-1. Stop Bits Selections**

| Transmitter Stop Bits | Receiver Verification |
|---|---|
| 1 | Verify Stop bit |
| 1.5 | Verify first Stop bit |
| 2 | Verify both Stop bits |
| 2 | Verify only first Stop bit |

In all modes, except DALI, the transmitter is Idle for the number of Stop bit periods between each consecutively transmitted word. In DALI, the Stop bits are generated after the last bit in the transmitted data stream.

The input is checked for the idle level in the middle of the first Stop bit, when receive verify on first is selected, as well as in the middle of the second Stop bit, when verify on both is selected. If any Stop bit verification indicates a nonidle level, the framing error FERIF bit is set for the received word.

### 34.8.1 Delayed Receive Interrupt

When operating in Half Duplex mode, where the microcontroller needs to reverse the transceiver direction after a reception, it may be more convenient to hold off the UxRXIF interrupt until the end of the Stop bits to avoid line contention. The user selects when the UxRXIF interrupt occurs with the STPMD bit. When STPMD is '1', the UxRXIF interrupt occurs at the end of the last Stop bit. When STPMD is '0', the UxRXIF interrupt occurs when the received

byte is stored in the receive FIFO. When STP = 10, the store operation is performed in the middle of the second Stop bit, otherwise, it is performed in the middle of the first Stop bit.

The FERIF and PERIF interrupts are not delayed with STPMD. When STPMD is set, the preferred indicator for reversing transceiver direction is the UxRXIF interrupt because it is delayed whereas the others are not.

## 34.9 Operation After FIFO Overflow

The Receive Shift Register (RSR) can be configured to stop or continue running during a receive FIFO Overflow condition. Stopped operation is the Legacy mode.

When the RSR continues to run during an Overflow condition, the first word received after clearing the overflow will always be valid.

When the RSR is stopped during an Overflow condition, the synchronization with the Start bits is lost. Therefore, the first word received after the overflow is cleared may start in the middle of a word.

Operation during overflow is selected with the RUNOVF bit. When the RUNOVF bit is set, the receiver maintains synchronization with the Start bits throughout the Overflow condition.

## 34.10 Receive and Transmit Buffers

The UART uses small buffer areas to transmit and receive data. These are sometimes referred to as FIFOs.

The receiver has a Receive Shift Register (RSR) and two or more buffer registers. The buffer at the top of the FIFO (earliest byte to enter the FIFO) is by retrieved by reading the UxRXB register.

The transmitter has one or more Transmit Shift Register (TSR) and one buffer register. Writes to UxTXB go to the transmit buffer then immediately to the TSR, if it is empty. When the TSR is not empty, writes to UxTXB are held then transferred to the TSR when it becomes available.

### 34.10.1 FIFO Status

The UxFIFO register contains several Status bits for determining the state of the receive and transmit buffers.

The RXBE bit indicates that the receive FIFO is empty. This bit is essentially the inverse of UxRXIF. The RXBF bit indicates that the receive FIFO is full.

The TXBE bit indicates that the transmit buffer is empty (same as UxTXIF) and the TXBF bit indicates that the buffer is full. A third transmitter Status bit, TXWRE (transmit write error), is set whenever a UxTXB write is performed when the TXBF bit is set. This indicates that the write was unsuccessful.

### 34.10.2 FIFO Reset

All modes support resetting the receive and transmit buffers.

The receive buffer is flushed and all unread data discarded when the RXBE bit is written to '1'. Instead of using a BSF instruction to set RXBE, the MOVWF instruction with the TXBE bit cleared will be used to avoid inadvertently clearing a byte pending in the TSR when UxTXB is empty.

Data written to UxTXB when TXEN is low will be held in the Transmit Shift Register (TSR), then sent when TXEN is set. The transmit buffer and inactive TSR are flushed by setting the TXBE bit. Setting TXBE while a character is actively transmitting from the TSR will complete the transmission without being flushed.

Clearing the ON bit will discard all received data and transmit data pending in the TSR and UxTXB.

## 34.11 Flow Control

This section does not apply to the LIN, DALI, or DMX modes.

Flow control is the means by which a sending UART data stream can be suspended by a receiving UART. Flow control prevents input buffers from overflowing without software intervention. The UART supports both hardware and XON/XOFF methods of flow control.

© 2019-2023 Microchip Technology Inc. and its subsidiaries

The flow control method is selected with the FLO bits. Flow control is disabled when both bits are cleared.

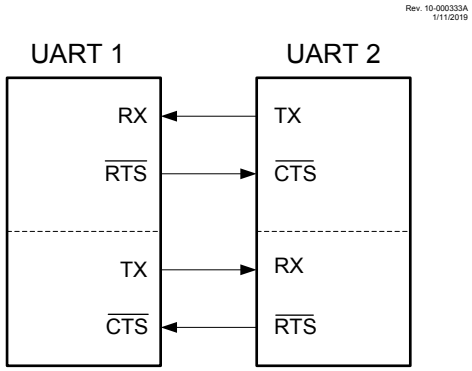
### 34.11.1 Hardware Flow Control

The hardware flow control is selected by setting the FLO bits to '10'.

The hardware flow control consists of three lines. The RS-232 signal names for two of these are $\overline{RTS}$ and $\overline{CTS}$. Both are low true. The third line is called TXDE for transmit drive enable which may be used to control an RS-485 transceiver. This output is high when the TX output is actively sending a character and low at all other times. The UART is configured as DTE (computer) equipment, which means $\overline{RTS}$ is an output and $\overline{CTS}$ is an input.

The $\overline{RTS}$ and $\overline{CTS}$ signals work as a pair to control the transmission flow. A DTE-to-DTE configuration connects the $\overline{RTS}$ output of the receiving UART to the $\overline{CTS}$ input of the sending UART. Refer to the following figure.

**Figure 34-10. Hardware Flow Control Connections**



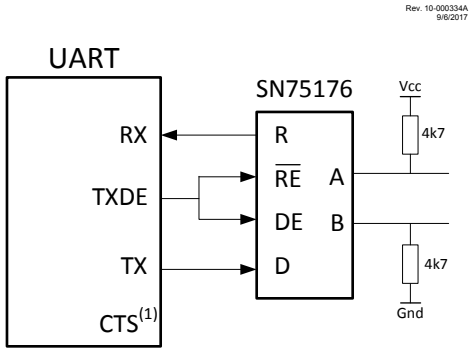

The UART receiving data asserts the $\overline{RTS}$ output low when the input FIFO is empty. When a character is received, the $\overline{RTS}$ output goes high until the UxRXB is read to free up both FIFO locations.

When the $\overline{CTS}$ input goes high after a byte has started to transmit, the transmission will complete normally. The receiver accommodates this by accepting the character in the second FIFO location even when the $\overline{CTS}$ input is high.

### 34.11.2 RS-485 Transceiver Control

The hardware flow control can be used to control the direction of an RS-485 transceiver as shown in the following figure. The $\overline{CTS}$ input will be configured to be always enabled by setting the UxCTSPPS selection to an unimplemented PORT pin, such as RD0. When the signal and control lines are configured as shown in the figure below, the UART will not receive its own transmissions. To verify that there are no collisions on the RS-485 lines, the transceiver RE control can be disconnected from TXDE and tied low, thereby enabling loopback reception of all transmissions. See the Collision Detection section for more information.

**Figure 34-11. RS-485 Configuration**




**Note 1:** Configure UxCTSPPS to an unimplemented input such as RD0. (e.g. UxCTSPPS = 0x18)

### 34.11.3 XON/XOFF Flow Control

XON/XOFF flow control is selected by setting the FLO bits to '01'.

XON/XOFF is a data-based flow control method. The signals to suspend and resume transmission are special characters sent by the receiver to the transmitter. The advantage is that additional hardware lines are not needed.

XON/XOFF flow control requires full-duplex operation because the transmitter must be able to receive the signal to suspend transmitting while the transmission is in progress. Although XON and XOFF are not defined in the ASCII code, the generally accepted values are 13h for XOFF and 11h for XON. The UART uses those codes.

The transmitter defaults to XON, or transmitter enabled. This state is also indicated by the read-only XON bit.

When an XOFF character is received, the transmitter stops transmitting after completing the character actively being transmitted. The transmitter remains disabled until an XON character is received.

XON will be forced on when software toggles the TXEN bit.

When the RUNOVF bit is set, the XON and XOFF characters continue to be received and processed without the need to clear the input FIFO by reading UxRXB. However, if the RUNOVF bit is clear then UxRXB must be read to avoid a receive overflow which will suspend flow control when the receive buffer overflows.

## 34.12 Checksum (Full-Featured UARTs Only)

This section does not apply to the LIN mode, which handles checksums automatically.

The transmit and receive checksum adders are enabled when the C0EN bit is set. When enabled, the adders accumulate every byte that is transmitted or received. The accumulated sum includes the carry of the addition. Software is responsible for clearing the checksum registers before a transaction and performing the check at the end of the transaction.

The following examples illustrate how the checksum registers can be used in the Asynchronous modes.

### 34.12.1 Transmit Checksum Method

1. Clear the UxTXCHK register.
2. Set the C0EN bit.
3. Send all bytes of the transaction output.
4. Invert UxTXCHK and send the result as the last byte of the transaction.

### 34.12.2 Receive Checksum Method

1. Clear the UxRXCHK register.
2. Set the C0EN bit.
3. Receive all bytes in the transaction including the checksum byte.
4. Set MSb of UxRXCHK if 7-bit mode is selected.
5. Add '1' to UxRXCHK.
6. If the result is '0', the checksum passes, otherwise it fails.

The CERIF Checksum Interrupt flag is not active in any mode other than LIN.

## 34.13 Collision Detection (Full-Featured UARTs Only)

External forces that interfere with the transmit line are detected in all modes of operation with collision detection. Collision detection is always active when RXEN and TXEN are both set. When the receive input is connected to the transmit output through either the same I/O pin or external circuitry, a character will be received for every character transmitted. The collision detection circuit provides a warning when the word received does not match the word transmitted.

The TXCIF flag is used to signal collisions. This signal is only useful when the TX output is looped back to the RX input and everything that is transmitted is expected to be received. If more than one transmitter is active at the same time, it can be assumed that the TX word will not match the RX word. The TXCIF detects this mismatch and flags an interrupt. The TXCIF bit will also be set in DALI mode transmissions when the received bit is missing the expected mid-bit transition.

Collision detection is always active, regardless of whether or not the RX input is connected to the TX output. It is up to the user to disable the TXCIE bit when collision interrupts are not required. The software overhead of unloading the receive buffer of transmitted data is avoided by setting the RUNOVF bit and ignoring the receive interrupt and letting the receive buffer overflow. When the transmission is complete, prepare for receiving data by flushing the receive buffer (see the FIFO Reset section) and clearing the RXFOIF overflow flag.

## 34.14 RX/TX Activity Time-Out

The UART works in conjunction with the HLT timers to monitor activity on the RX and TX lines. Use this feature to determine when there has been no activity on the receive or transmit lines for a user-specified period of time.

To use this feature, set the HLT to the desired time-out period by a combination of the HLT clock source, timer prescale value, and timer period registers. Configure the HLT to reset on the UART TX or RX line and start the HLT at the same time the UART is started. UART activity will keep resetting the HLT to prevent a full HLT period from elapsing. When there has been no activity on the selected TX or RX line for longer than the HLT period, then an HLT interrupt will occur signaling the time-out event.

For example, the following register settings will configure HLT2 for a 5 ms time-out of no activity on U1RX:

- T2PR = 0x9C (156 prescale periods)
- T2CLKCON = 0x05 (500 kHz internal oscillator)
- T2HLT = 0x04 (free running, reset on rising edge)
- T2RST = 0x15 (reset on U1RX)
- T2CON = 0xC0 (Timer2 on with 1:16 prescale)

## 34.15 Clock Accuracy with Asynchronous Operation

The factory calibrates the internal oscillator block output (INTOSC). However, the INTOSC frequency may drift as $V_{DD}$ or temperature changes, and this directly affects the asynchronous baud rate. Two methods may be used to adjust the baud rate clock, but both require a reference clock source of some kind.

The first (preferred) method uses the OSCTUNE register to adjust the INTOSC output. Adjusting the value of the OSCTUNE register allows for fine resolution changes to the system clock source. See the **"HFINTOSC Frequency Tuning"** section for more information.

The other method adjusts the value of the Baud Rate Generator. This can be done automatically with the Auto-Baud Detect feature (see the Auto-Baud Detect section). There may not be fine enough resolution when adjusting the Baud Rate Generator to compensate for a gradual change of the peripheral clock frequency.

## 34.16 UART Baud Rate Generator

The Baud Rate Generator (BRG) is a 16-bit timer that is dedicated to the support of the UART operation. The UxBRG register pair determines the period of the free-running baud rate timer. The multiplier of the baud rate period is determined by the BRGS bit.

The high baud rate range (BRGS = 1) is intended to extend the baud rate range up to a faster rate when the desired baud rate is not possible otherwise, and to improve the baud rate resolution at high baud rates. Using the normal baud rate range (BRGS = 0) is recommended when the desired baud rate is achievable with either range.

**Important:** BRGS = 1 is not supported in the DALI mode.

Writing a new value to UxBRG causes the BRG timer to be reset (or cleared). This ensures that the BRG does not wait for a timer overflow before outputting the new baud rate.

If the system clock is changed during an active receive operation, a receive error or data loss may result. To avoid this problem, check the status of the RXIDL bit to make sure that the receive operation is Idle before changing the system clock. The following table contains formulas for determining the baud rate.

**Table 34-2. Baud Rate Formulas**

| BRGS | BRG/UART Mode | Baud Rate Formula |
|---|---|---|
| 1 | High Rate | Fosc/[4(UxBRG+1)] |
| 0 | Normal Rate | Fosc/[16(UxBRG+1)] |

The following example provides a sample calculation for determining the baud rate and baud rate error.

**Example 34-1. Baud Rate Error Calculation**

For a device with Fosc of 16 MHz, desired baud rate of 9600, Asynchronous mode, and BRGS = 0.

$$DesiredBaudrate = \frac{F_{OSC}}{16 \times (UxBRG + 1)}$$

Solving for UxBRG:

$$UxBRG = \frac{F_{OSC}}{16 \times DesiredBaudrate} - 1$$

$$UxBRG = \frac{16000000}{16 \times 9600} - 1$$

$$UxBRG = 103.17 \simeq 103$$

$$CalculatedBaudrate = \frac{16000000}{16 \times (103 + 1)}$$

$$CalculatedBaudrate = 9615$$

$$Error = \frac{CalculatedBaudrate - DesiredBaudrate}{DesiredBaudrate}$$

$$Error = \frac{9615 - 9600}{9600}$$

$$Error \simeq 0.16\,\%$$

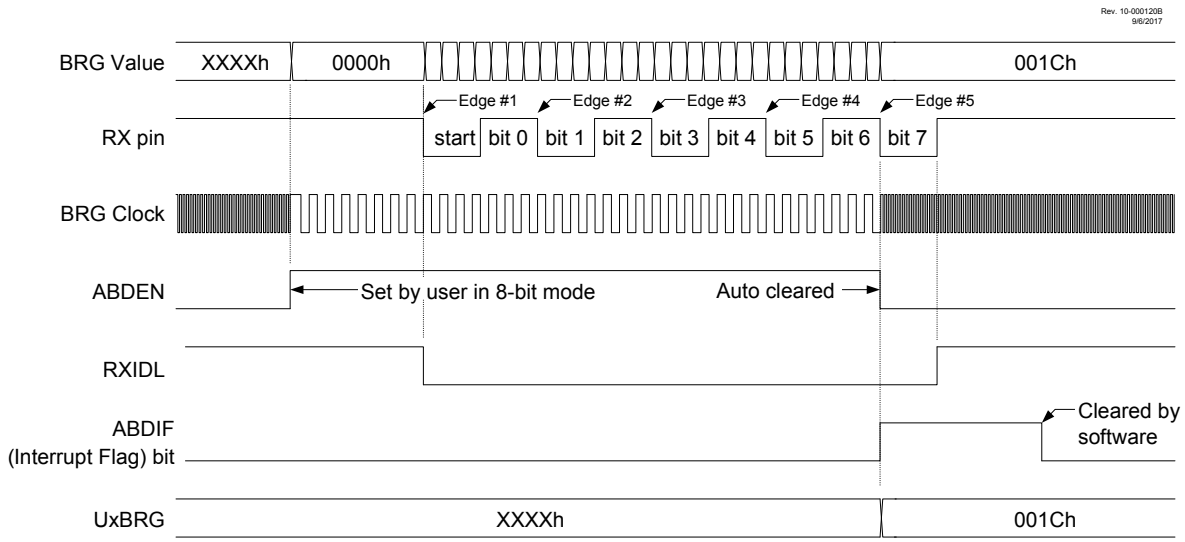
### 34.16.1 Auto-Baud Detect

The UART module supports automatic detection and calibration of the baud rate in the 8-bit Asynchronous and LIN modes. However, setting ABDEN to start auto-baud detection is neither necessary, nor possible in LIN mode because that mode supports auto-baud detection automatically at the beginning of every data packet. Enabling auto-baud detect with the ABDEN bit applies to the Asynchronous modes only.

When Auto-Baud Detect (ABD) is active, the clock to the BRG is reversed. Rather than the BRG clocking the incoming RX signal, the RX signal is timing the BRG. The Baud Rate Generator is used to time the period of a received 55h (ASCII “U”), which is the Sync character for the LIN bus. The unique feature of this character is that it has five falling edges, including the Start bit edge, and five rising edges, including the Stop bit edge.

In 8-bit Asynchronous mode, setting the ABDEN bit enables the auto-baud calibration sequence. The first falling edge of the RX input after ABDEN is set will start the auto-baud calibration sequence. While the ABD sequence takes place, the UART state machine is held in Idle. On the first falling edge of the receive line, the UxBRG begins counting up using the BRG counter clock, as shown in the following figure. The fifth falling edge will occur on the RX pin at the beginning of the bit 7 period. At that time, an accumulated value totaling the proper BRG period is left in the UxBRG register pair, the ABDEN bit is automatically cleared and the ABDIF interrupt flag is set. ABDIF must be cleared by software.

**Figure 34-12. Automatic Baud Rate Calibration**




RXIDL indicates that the sync input is active. RXIDL will go low on the first falling edge and go high on the fifth rising edge.

The BRG auto-baud clock is determined by the BRGS bit, as shown in the following table.

**Table 34-3. BRG Counter Clock Rates**

| BRGS | BRG Base Clock | BRG ABD Clock |
|---|---|---|
| 1 | Fosc/4 | Fosc/32 |
| 0 | Fosc/16 | Fosc/128 |

During ABD, the internal BRG register is used as a 16-bit counter. However, the UxBRG registers retain the previous BRG value until the auto-baud process is successfully completed. While calibrating the baud rate period, the internal BRG register is clocked at 1/8th the BRG base clock rate. The resulting byte measurement is the average bit time when clocked at full speed and is transferred to the UxBRG registers when complete.

© 2019-2023 Microchip Technology Inc. and its subsidiaries

**Important:**

1. When both the WUE and ABDEN bits are set, the auto-baud detection will occur on the byte following the Break character (see the Auto Wake-on-Break section).
2. It is up to the user to verify the incoming character baud rate is within the range of the selected BRG clock source. Some combinations of oscillator frequency and UART baud rates are not possible.

### 34.16.2 Auto-Baud Overflow

During the course of automatic baud detection, the ABDOVF bit will be set if the baud rate counter overflows before the fifth falling edge is detected on the RX pin. The ABDOVF bit indicates that the counter has exceeded the maximum count that can fit in the 16 bits of the UxBRG register pair. After the ABDOVF bit has been set, the state machine continues to search until the fifth falling edge is detected on the RX pin. Upon detecting the fifth falling RX edge, the hardware will set the ABDIF Interrupt flag and clear the ABDEN bit. The UxBRG register values retain their previous value. The ABDIF flag and ABDOVF flag can be cleared by software directly. To generate an interrupt on an Auto-Baud Overflow condition, all the following bits must be set:

- ABDOVE bit
- UxEIE bit in the PIEx register
- Global Interrupt Enable bits

To terminate the auto-baud process before the ABDIF flag is set, clear the ABDEN bit, then clear the ABDOVF bit.

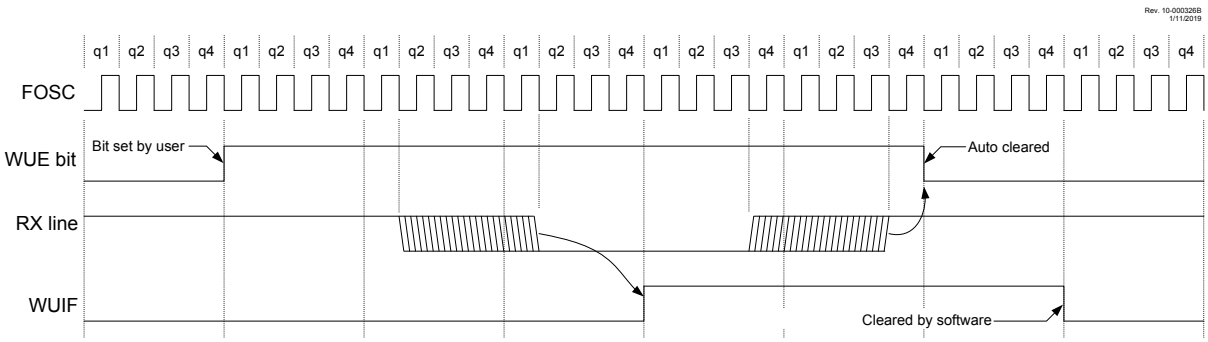
### 34.16.3 Auto Wake-on-Break

During Sleep mode, all clocks to the UART are suspended. Because of this, the Baud Rate Generator is inactive and a proper character reception cannot be performed. The Auto Wake-on-Break feature allows the controller to wake up due to activity on the RX line.

The Auto-Wake-up feature is enabled by setting both the WUE bit and the UxIE bit in the PIEx register. Once set, the normal receive sequence on RX is disabled, and the UART remains in an Idle state, monitoring for a wake-up event independent of the CPU mode. A wake-up event consists of a transition out of the Idle state on the RX line (this coincides with the start of a Break or a wake-up signal character for the LIN protocol).

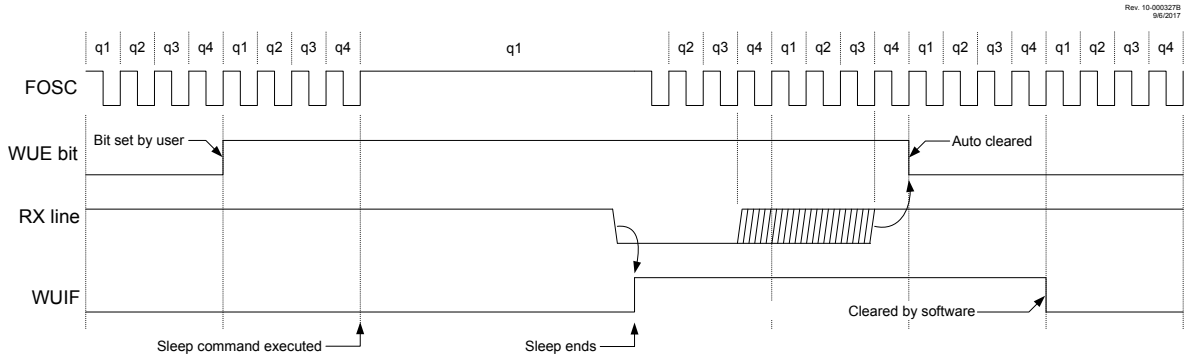
The UART module generates a WUIF interrupt coincident with the wake-up event. The interrupt is generated synchronously to the Q clocks in normal CPU operating modes (Figure 34-13), and asynchronously, if the device is in Sleep mode (Figure 34-14). The interrupt condition is cleared by clearing the WUIF bit.

**Figure 34-13. Auto-Wake-Up Timing During Normal Operation**




**Note 1:** The UART remains in Idle while the WUE bit is set.

**Figure 34-14. Auto-Wake-Up Timing During Sleep**




**Note 1:** The UART remains in Idle while the WUE bit is set.

To generate an interrupt on a wake-up event, all the following bits must be set:

- The UxIE bit in the PIEx register
- Global interrupt enables

The WUE bit is automatically cleared by the transition to the Idle state on the RX line at the end of the Break. This signals to the user that the Break event is over. At this point, the UART module is in Idle mode, waiting to receive the next character.

### 34.16.3.1 Auto-Wake-Up Special Considerations

**Break Character**

To avoid character errors or character fragments during a wake-up event, all bits in the character causing the Wake event must be zero.

When the wake-up is enabled, the function works independent of the low time on the data stream. If the WUE bit is set and a valid nonzero character is received, the low time from the Start bit to the first rising edge will be interpreted as the wake-up event. The remaining bits of the character will be received as a fragmented character and subsequent characters can result in framing or overrun errors.

Therefore, the initial character of the transmission must be all zeros. This must be eleven or more bit times, 13 bit times recommended for LIN bus, or any number of bit times for standard RS-232 devices.

**Oscillator Start-Up Time**

The oscillator start-up time must be considered, especially in applications using oscillators with longer start-up intervals (i.e., LP, XT or HS/PLL modes). The Sync Break (or wake-up signal) character must be of sufficient length, and be followed by a sufficient interval, to allow enough time for the selected oscillator to start and provide proper initialization of the UART.

**The WUE Bit**

To ensure that no actual data is lost, check the RXIDL bit to verify that a receive operation is not in process before setting the WUE bit. If a receive operation is not occurring, the WUE bit may then be set just prior to entering the Sleep mode.

## 34.17 Transmitting a Break

The UART module has the capability of sending either a fixed length Break period or a software-timed Break period. The fixed length Break consists of a Start bit, followed by 12 '0' bits and a Stop bit. The software-timed Break is generated by setting and clearing the BRKOVR bit.

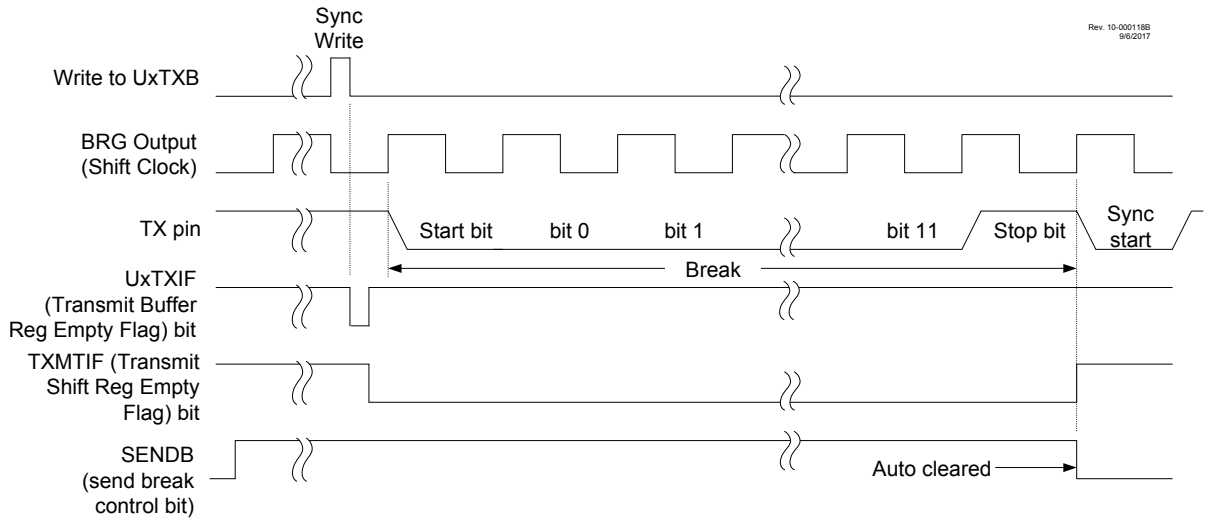
To send the fixed length Break, set the SENDB and TXEN bits. The Break sequence is then initiated by a write to UxTXB. The timed Break will occur first, followed by the character written to UxTXB that initiated the Break. The initiating character is typically the Sync character of the LIN specification.

SENDB is disabled in the LIN and DMX modes because those modes generate the Break sequence automatically.

The SENDB bit is automatically reset by hardware after the Break Stop bit is complete.

The TXMTIF bit indicates when the transmit operation is Active or Idle, just as it does during normal transmission. The following figure illustrates the Break sequence.

**Figure 34-15. Send-Break Sequence**




## 34.18 Receiving a Break

The UART has counters to detect when the RX input remains in the Space state for an extended period of time. When this happens, the RXBKIF bit is set.

A Break is detected when the RX input remains in the Space state for 11 bit periods for asynchronous and LIN modes, and 23 bit periods for DMX mode.

The user can select to receive the Break interrupt as soon as the Break is detected or at the end of the Break, when the RX input returns to the Idle state. When the RXBIMD bit is '1', then RXBKIF is set immediately upon Break detection. When RXBIMD is '0', then RXBKIF is set when the RX input returns to the Idle state.

## 34.19 UART Operation During Sleep

The UART ceases to operate during Sleep. The safe way to wake the device from Sleep by a serial operation is to use the Wake-on-Break feature of the UART. See the Auto Wake-on-Break section.

## 34.20 Register Definitions: UART

Long bit name prefixes for the UART peripherals are shown in the following table. Refer to the **"Long Bit Names"** section in the **"Register and Bit Naming Conventions"** chapter for more information.

**Table 34-4. UART Long Bit Name Prefixes**

| Peripheral | Bit Name Prefix |
|---|---|
| UART1 (full featured) | U1 |
| UART2 (limited features) | U2 |

.....continued

| Peripheral | Bit Name Prefix |
|---|---|
| UART3 (limited features) | U3 |
| UART4 (limited features) | U4 |
| UART5 (limited features) | U5 |

### 34.20.1 UxCON0

**Name:** UxCON0
**Address:** 0x2AB,0x2BE,0x2D1,0x2E4,0x2F7

UART Control Register 0

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | BRGS | ABDEN | TXEN | RXEN | MODE[3:0] | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bit 7 – BRGS** Baud Rate Generator Speed Select

| Value | Description |
|---|---|
| 1 | Baud Rate Generator is high speed with 4 baud clocks per bit |
| 0 | Baud Rate Generator is normal speed with 16 baud clocks per bit |

**Bit 6 – ABDEN** Auto-Baud Detect Enable(3)

| Value | Description |
|---|---|
| 1 | Auto-baud is enabled. Receiver is waiting for Sync character (0x55). |
| 0 | Auto-baud is not enabled or auto-baud is complete |

**Bit 5 – TXEN** Transmit Enable Control(2)

| Value | Description |
|---|---|
| 1 | Transmit is enabled. TX output pin drive is forced on when transmission is active, and controlled by PORT TRIS control when transmission is Idle. |
| 0 | Transmit is disabled. TX output pin drive is controlled by PORT TRIS control. |

**Bit 4 – RXEN** Receive Enable Control(2)

| Value | Description |
|---|---|
| 1 | Receiver is enabled |
| 0 | Receiver is disabled |

**Bits 3:0 – MODE[3:0]** UART Mode Select(1)

| Value | Description |
|---|---|
| 1111 – 1101 | Reserved |
| 1100 | LIN Host/Client mode(4) |
| 1011 | LIN Client Only mode(4) |
| 1010 | DMX mode(4) |
| 1001 | DALI Control Gear mode(4) |
| 1000 | DALI Control Device mode(4) |
| 0111 – 0101 | Reserved |
| 0100 | Asynchronous 9-bit UART Address mode. 9th bit: 1 = address, 0 = data |
| 0011 | Asynchronous 8-bit UART mode with 9th bit even parity |
| 0010 | Asynchronous 8-bit UART mode with 9th bit odd parity |
| 0001 | Asynchronous 7-bit UART mode |
| 0000 | Asynchronous 8-bit UART mode |

**Notes:**

1. Changing the UART MODE while ON = 1 may cause unexpected results.
2. Clearing TXEN or RXEN will not clear the corresponding buffers. Use TXBE or RXBE to clear the buffers.
3. ABDEN is read-only when MODE > 'b0111.
4. Full-featured UARTs only.

### 34.20.2 UxCON1

**Name:** UxCON1
**Address:** 0x2AC,0x2BF,0x2D2,0x2E5,0x2F8

UART Control Register 1

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | ON | | | WUE | RXBIMD | | BRKOVR | SENDB |
| Access | R/W | | | R/W/HC | R/W | | R/W | R/W/HC |
| Reset | 0 | | | 0 | 0 | | 0 | 0 |

**Bit 7 – ON** Serial Port Enable

| Value | Description |
|---|---|
| 1 | Serial port enabled |
| 0 | Serial port disabled (held in Reset) |

**Bit 4 – WUE** Wake-Up Enable

| Value | Description |
|---|---|
| 1 | Receiver is waiting for falling RX input edge which will set the UxIF bit. Cleared by hardware on wake-up event. Also requires the UxIE bit of PIEx to enable wake. |
| 0 | Receiver operates normally |

**Bit 3 – RXBIMD** Receive Break Interrupt Mode Select

| Value | Description |
|---|---|
| 1 | Set RXBKIF immediately when RX in has been low for the minimum Break time |
| 0 | Set RXBKIF on rising RX input after RX in has been low for the minimum Break time |

**Bit 1 – BRKOVR** Send Break Software Override

| Value | Description |
|---|---|
| 1 | TX output is forced to non-Idle state |
| 0 | TX output is driven by transmit shift register |

**Bit 0 – SENDB** Send Break Control(1)

| Value | Description |
|---|---|
| 1 | Output Break upon UxTXB write. Written byte follows Break. Bit is cleared by hardware. |
| 0 | Break transmission completed or disabled |

**Note:**

1. This bit is read-only in LIN, DMX and DALI modes.

### 34.20.3 UxCON2

**Name:** UxCON2

UART Control Register 2

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | RUNOVF | RXPOL | STP[1:0] | | C0EN | TXPOL | FLO[1:0] | |
| Access | R/W | R/W/HC | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bit 7 – RUNOVF** Run During Overflow Control

| Value | Description |
|---|---|
| 1 | RX input shifter continues to synchronize with Start bits after Overflow condition |
| 0 | RX input shifter stops all activity on receiver Overflow condition |

**Bit 6 – RXPOL** Receive Polarity Control

| Value | Description |
|---|---|
| 1 | Invert RX polarity, Idle state is low |
| 0 | RX polarity is not inverted, Idle state is high |

**Bits 5:4 – STP[1:0]** Stop Bit Mode Control(1)

| Value | Description |
|---|---|
| 11 | Transmit 2 Stop bits, receiver verifies first Stop bit |
| 10 | Transmit 2 Stop bits, receiver verifies first and second Stop bits |
| 01 | Transmit 1.5 Stop bits, receiver verifies first Stop bit |
| 00 | Transmit 1 Stop bit, receiver verifies first Stop bit |

**Bit 3 – C0EN** Checksum Mode Select(2)

| Value | Condition | Description |
|---|---|---|
| 1 | MODE = LIN | Enhanced LIN checksum includes PID in sum |
| 0 | MODE = LIN | Legacy LIN checksum does not include PID in sum |
| 1 | MODE = not LIN | Checksum is the sum of all TX and RX characters |
| 0 | MODE = not LIN | Checksum is disabled |

**Bit 2 – TXPOL** Transmit Control Polarity(1)

| Value | Description |
|---|---|
| 1 | Output data is inverted, TX output is low in Idle state |
| 0 | Output data is not inverted, TX output is high in Idle state |

**Bits 1:0 – FLO[1:0]** Handshake Flow Control

| Value | Description |
|---|---|
| 11 | Reserved |
| 10 | $\overline{RTS}$/$\overline{CTS}$ and TXDE Hardware flow control |
| 01 | XON/XOFF Software flow control |
| 00 | Flow control is off |

**Notes:**

1. All modes transmit selected number of Stop bits.
2. Full-featured UARTs only.

### 34.20.4 UxERRIR

**Name:** UxERRIR

UART Error Interrupt Flag Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | TXMTIF | PERIF | ABDOVF | CERIF | FERIF | RXBKIF | RXFOIF | TXCIF |
| Access | R/S/C | R/W/HC | R/W/S | R/W/S | R/S/C | R/W/S | R/W/S | R/W/S |
| Reset | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bit 7 – TXMTIF** Transmit Shift Register Empty Interrupt Flag

| Value | Description |
|---|---|
| 1 | Transmit shift register is empty (Set at end of Stop bits) |
| 0 | Transmit shift register is actively shifting data |

**Bit 6 – PERIF** Parity Error Interrupt Flag

| Value | Condition | Description |
|---|---|---|
| 1 | MODE = LIN or Parity | Unread byte at top of input FIFO has parity error |
| 0 | MODE = LIN or Parity | Unread byte at top of input FIFO does not have parity error |
| 1 | MODE = DALI Device | Unread byte at top of input FIFO received as Forward Frame |
| 0 | MODE = DALI Device | Unread byte at top of input FIFO received as Back Frame |
| 1 | MODE = Address | Unread byte at top of input FIFO received as address |
| 0 | MODE = Address | Unread byte at top of input FIFO received as data |
| x | MODE = All others | Not used |

**Bit 5 – ABDOVF** Auto-baud Detect Overflow Interrupt Flag

| Value | Condition | Description |
|---|---|---|
| 1 | MODE = DALI | Start bit measurement overflowed counter |
| 0 | MODE = DALI | No overflow during Start bit measurement |
| 1 | MODE = All others | Baud Rate Generator overflowed during the auto-detection sequence |
| 0 | MODE = All others | Baud Rate Generator has not overflowed |

**Bit 4 – CERIF** Checksum Error Interrupt Flag

| Value | Condition | Description |
|---|---|---|
| 1 | MODE = LIN | Checksum error |
| 0 | MODE = LIN | No checksum error |
| x | MODE = not LIN | Not used |

**Bit 3 – FERIF** Framing Error Interrupt Flag

| Value | Description |
|---|---|
| 1 | Unread byte at top of input FIFO has framing error |
| 0 | Unread byte at top of input FIFO does not have framing error |

**Bit 2 – RXBKIF** Break Reception Interrupt Flag

| Value | Description |
|---|---|
| 1 | Break detected |
| 0 | No break detected |

**Bit 1 – RXFOIF** Receive FIFO Overflow Interrupt Flag

| Value | Description |
|---|---|
| 1 | Receive FIFO has overflowed |
| 0 | Receive FIFO has not overflowed |

**Bit 0 – TXCIF** Transmit Collision Interrupt Flag(1)

| Value | Description |
|---|---|
| 1 | Transmitted word is not equal to the word received during transmission |
| 0 | Transmitted word equals the word received during transmission |

**Note:**

1. Full-featured UARTs only.

### 34.20.5 UxERRIE

**Name:** UxERRIE

UART Error Interrupt Enable Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | TXMTIE | PERIE | ABDOVE | CERIE | FERIE | RXBKIE | RXFOIE | TXCIE |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bit 7 – TXMTIE** Transmit Shift Register Empty Interrupt Enable

| Value | Description |
|---|---|
| 1 | Interrupt enabled |
| 0 | Interrupt not enabled |

**Bit 6 – PERIE** Parity Error Interrupt Enable

| Value | Description |
|---|---|
| 1 | Interrupt enabled |
| 0 | Interrupt not enabled |

**Bit 5 – ABDOVE** Auto-baud Detect Overflow Interrupt Enable

| Value | Description |
|---|---|
| 1 | Interrupt enabled |
| 0 | Interrupt not enabled |

**Bit 4 – CERIE** Checksum Error Interrupt Enable

| Value | Description |
|---|---|
| 1 | Interrupt enabled |
| 0 | Interrupt not enabled |

**Bit 3 – FERIE** Framing Error Interrupt Enable

| Value | Description |
|---|---|
| 1 | Interrupt enabled |
| 0 | Interrupt not enabled |

**Bit 2 – RXBKIE** Break Reception Interrupt Enable

| Value | Description |
|---|---|
| 1 | Interrupt enabled |
| 0 | Interrupt not enabled |

**Bit 1 – RXFOIE** Receive FIFO Overflow Interrupt Enable

| Value | Description |
|---|---|
| 1 | Interrupt enabled |
| 0 | Interrupt not enabled |

**Bit 0 – TXCIE** Transmit Collision Interrupt Enable(1)

| Value | Description |
|---|---|
| 1 | Interrupt enabled |
| 0 | Interrupt not enabled |

**Note:**

1. Full-featured UARTs only.

### 34.20.6 UxUIR

**Name:** UxUIR
**Address:** 0x2B1,0x2C4,0x2D7,0x2EA,0x2FD

UART General Interrupt Flag Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | WUIF | ABDIF | | | | ABDIE | | |
| Access | R/W/S | R/W/S | | | | R/W | | |
| Reset | 0 | 0 | | | | 0 | | |

**Bit 7 – WUIF** Wake-Up Interrupt

| Value | Description |
|---|---|
| 1 | Idle to non-Idle transition on RX line detected when WUE is set. Also sets UxIF. (WUIF must be cleared by software to clear UxIF) |
| 0 | WUE not enabled by software or no transition detected |

**Bit 6 – ABDIF** Auto-Baud Detect Interrupt

| Value | Description |
|---|---|
| 1 | Auto-baud detection complete. Status shown in UxIF when ABDIE is set. (Must be cleared by software) |
| 0 | Auto-baud not enabled or auto-baud enabled and auto-baud detection not complete |

**Bit 2 – ABDIE** Auto-Baud Detect Interrupt Enable

| Value | Description |
|---|---|
| 1 | ABDIF will set the UxIF bit in the PIRx register |
| 0 | ABDIF will not set UxIF |

### 34.20.7 UxFIFO

**Name:** UxFIFO
**Address:** 0x2B0,0x2C3,0x2D6,0x2E9,0x2FC

UART FIFO Status Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | TXWRE | STPMD | TXBE | TXBF | RXIDL | XON | RXBE | RXBF |
| Access | R/W/S | R/W | R/W/S/C | R/S/C | R/S/C | S/C | R/W/S/C | R/S/C |
| Reset | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 |

**Bit 7 – TXWRE** Transmit Write Error Status (must be cleared by software)

| Value | Condition | Description |
|---|---|---|
| 1 | MODE = LIN Host | UxP1L was written when a host process was active |
| 1 | MODE = LIN Client | UxTXB was written when UxP2 = 0 or more than UxP2 bytes have been written to UxTXB since last Break |
| 1 | MODE = Address detect | UxP1L was written before the previous data in UxP1L was transferred to TX shifter |
| 1 | MODE = All | A new byte was written to UxTXB when the output FIFO was full |
| 0 | MODE = All | No error |

**Bit 6 – STPMD** Stop Bit Detection Mode

| Value | Condition | Description |
|---|---|---|
| 1 | STP = 11 | Assert UxRXIF at end of first Stop bit |
| 1 | STP ≠ 11 | Assert UxRXIF at end of last Stop bit |
| 0 | STP = xx | Assert UxRXIF in middle of first Stop bit |

**Bit 5 – TXBE** Transmit Buffer Empty Status

| Value | Description |
|---|---|
| 1 | Transmit buffer is empty. Setting this bit will clear the transmit buffer and output shift register. |
| 0 | Transmit buffer is not empty. Software cannot clear this bit. |

**Bit 4 – TXBF** Transmit Buffer Full Status

| Value | Description |
|---|---|
| 1 | Transmit buffer is full |
| 0 | Transmit buffer is not full |

**Bit 3 – RXIDL** Receive Pin Idle Status

| Value | Description |
|---|---|
| 1 | Receive pin is in Idle state |
| 0 | UART is receiving Start, Stop, Data, Auto-baud, or Break |

**Bit 2 – XON** Software Flow Control Transmit Enable Status

| Value | Description |
|---|---|
| 1 | Transmitter is enabled |
| 0 | Transmitter is disabled |

**Bit 1 – RXBE** Receive Buffer Empty Status

| Value | Description |
|---|---|
| 1 | Receive buffer is empty. Setting this bit will clear the RX buffer(1). |
| 0 | Receive buffer is not empty. Software cannot clear this bit. |

**Bit 0 – RXBF** Receive Buffer Full Status

| Value | Description |
|---|---|
| 1 | Receive buffer is full |

| Value | Description |
|---|---|
| 0 | Receive buffer is not full |

**Note:**

1. The BSF instruction will not be used to set RXBE because doing so will clear a byte pending in the transmit shift register when the UxTXB register is empty. Instead, use the MOVWF instruction with a '0' in the TXBE bit location.

### 34.20.8 UxBRG

**Name:** UxBRG
**Address:** 0x2AE,0x2C1,0x2D4,0x2E7,0x2FA

UART Baud Rate Generator

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | BRG[15:8] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | BRG[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 15:0 – BRG[15:0]** Baud Rate Generator Value
The UART Baud Rate equals [Fosc*(1+(BRGS*3)]/[(16*(BRG+1))]

**Notes:**

1. The individual bytes in this multibyte register can be accessed with the following register names:
   – UxBRGH: Accesses the high byte BRG[15:8]
   – UxBRGL: Accesses the low byte BRG[7:0]
2. The UxBRG registers will only be written when ON = `0`.
3. Maximum BRG value when MODE = `'100x` and BRGS = `1` is `0x7FFE`.
4. Maximum BRG value when MODE = `'100x` and BRGS = `0` is `0x1FFE`.

### 34.20.9 UxRXB

**Name:** UxRXB
**Address:** 0x2A1,0x2B4,0x2C7,0x2DA,0x2ED

UART Receive Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | RXB[7:0] | | | | | | | |
| Access | R | R | R | R | R | R | R | R |
| Reset | x | x | x | x | x | x | x | x |

**Bits 7:0 – RXB[7:0]** Top of Receive FIFO

### 34.20.10 UxTXB

**Name:** UxTXB
**Address:** 0x2A3,0x2B6,0x2C9,0x2DC,0x2EF

UART Transmit Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | TXB[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 7:0 – TXB[7:0]** Bottom of Transmit FIFO

### 34.20.11 UxP1

**Name:** UxP1

UART Parameter 1

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | P1[8] |
| Access | | | | | | | | R/W |
| Reset | | | | | | | | 0 |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | P1[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bit 8 – P1[8]** Parameter 1 Most Significant bit
UART mode operating parameter values

| Value | Condition | Description |
|---|---|---|
| n | MODE = DMX | Most Significant bit of number of bytes to transmit between Start Code and automatic Break generation |
| n | MODE = DALI Control Device | Most Significant bit of Idle time delay after which a Forward Frame is sent. Measured in half-bit periods. |
| n | MODE = DALI Control Gear | Most Significant bit of delay between the end of a Forward Frame and the start of the Back Frame. Measured in half-bit periods. |
| x | All other modes/Limited featured UART | Not used |

**Bits 7:0 – P1[7:0]** Parameter 1 Least Significant bits
UART mode operating parameter values

| Value | Condition | Description |
|---|---|---|
| n | MODE = DMX | Least Significant bits of number of bytes to transmit between Start Code and automatic Break generation |
| n | MODE = DALI Control Device | Least Significant bits of Idle time delay after which a Forward Frame is sent. Measured in half-bit periods. |
| n | MODE = DALI Control Gear | Least Significant bits of delay between the end of a Forward Frame and the start of the Back Frame. Measured in half-bit periods. |
| n | MODE = LIN | PID to transmit (Only Least Significant six bits used) |
| n | MODE = Asynchronous Address | Address to transmit (9th transmit bit automatically set to '1') |
| x | All other modes | Not used |

**Notes:** The individual bytes in this multibyte register can be accessed with the following register names:

- UxP1H: Accesses the high byte P1[8]
- UxP1L: Accesses the low byte P1[7:0]

### 34.20.12 UxP2

**Name:** UxP2

UART Parameter 2

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | P2[8] |
| Access | | | | | | | | R/W |
| Reset | | | | | | | | 0 |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | P2[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bit 8 – P2[8]** Parameter 2 Most Significant bit
UART mode operating parameter values

| Value | Condition | Description |
|---|---|---|
| n | MODE = DMX | Most Significant bit of first address of receive block |
| n | MODE = DALI | Most Significant bit of number of half-bit periods of Idle time in Forward Frame detection threshold |
| x | All other modes/Limited featured UART | Not used |

**Bits 7:0 – P2[7:0]** Parameter 2 Least Significant bits
UART mode operating parameter values

| Value | Condition | Description |
|---|---|---|
| n | MODE = DMX | Least Significant bits of first address of receive block |
| n | MODE = DALI | Least Significant bits of number of half-bit periods of Idle time in Forward Frame detection threshold |
| n | MODE = LIN | Number of data bytes to transmit |
| n | MODE = Asynchronous Address | Receiver address |
| x | All other modes | Not used |

**Notes:** The individual bytes in this multibyte register can be accessed with the following register names:

- UxP2H: Accesses the high byte P2[8]
- UxP2L: Accesses the low byte P2[7:0]

### 34.20.13 UxP3

**Name:** UxP3

UART Parameter 3

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | P3[8] |
| Access | | | | | | | | R/W |
| Reset | | | | | | | | 0 |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | P3[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bit 8 – P3[8]** Parameter 3 Most Significant bit

UART mode operating parameter values

| Value | Condition | Description |
|---|---|---|
| n | MODE = DMX | Most Significant bit of last address of receive block |
| x | All other modes/Limited featured UART | Not used |

**Bits 7:0 – P3[7:0]** Parameter 3 Least Significant bits

UART mode operating parameter values

| Value | Condition | Description |
|---|---|---|
| n | MODE = DMX | Least Significant bits of last address of receive block |
| n | MODE = LIN Client | Number of data bytes to receive |
| n | MODE = Asynchronous Address | Receiver address mask. Received address is XOR'd with UxP2L, then AND'd with UxP3L. Match occurs when result is zero. |
| x | All other modes | Not used |

**Notes:** The individual bytes in this multibyte register can be accessed with the following register names:

- UxP3H: Accesses the high byte P3[8]
- UxP3L: Accesses the low byte P3[7:0]

### 34.20.14 UxTXCHK

**Name:** UxTXCHK
**Address:** 0x02A4

UART Transmit Checksum Result Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | TXCHK[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 7:0 – TXCHK[7:0]** Transmit Checksum Value

| Value | Condition | Description |
|---|---|---|
| n | MODE = LIN and C0EN = 1 | Sum of all transmitted bytes including PID |
| n | MODE = LIN and C0EN = 0 | Sum of all transmitted bytes except PID |
| n | MODE = All others and C0EN = 1 | Sum of all transmitted bytes since last clear |
| x | MODE = All others and C0EN = 0 | Not used |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

### 34.20.15 UxRXCHK

**Name:** UxRXCHK
**Address:** 0x02A2

UART Receive Checksum Result Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | RXCHK[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 7:0 – RXCHK[7:0]** Receive Checksum Value

| Value | Condition | Description |
|---|---|---|
| n | MODE = LIN and C0EN = 1 | Sum of all received bytes including PID |
| n | MODE = LIN and C0EN = 0 | Sum of all received bytes except PID |
| n | MODE = All others and C0EN = 1 | Sum of all received bytes since last clear |
| x | MODE = All others and C0EN = 0 | Not used |

## 34.21 Register Summary - UART

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x02A1 | U1RXB | 7:0 | RXB[7:0] | | | | | | | |
| 0x02A2 | U1RXCHK | 7:0 | RXCHK[7:0] | | | | | | | |
| 0x02A3 | U1TXB | 7:0 | TXB[7:0] | | | | | | | |
| 0x02A4 | U1TXCHK | 7:0 | TXCHK[7:0] | | | | | | | |
| 0x02A5 | U1P1 | 7:0 | P1[7:0] | | | | | | | |
| | | 15:8 | | | | | | | | P1[8] |
| 0x02A7 | U1P2 | 7:0 | P2[7:0] | | | | | | | |
| | | 15:8 | | | | | | | | P2[8] |
| 0x02A9 | U1P3 | 7:0 | P3[7:0] | | | | | | | |
| | | 15:8 | | | | | | | | P3[8] |
| 0x02AB | U1CON0 | 7:0 | BRGS | ABDEN | TXEN | RXEN | MODE[3:0] | | | |
| 0x02AC | U1CON1 | 7:0 | ON | | | WUE | RXBIMD | | BRKOVR | SENDB |
| 0x02AD | U1CON2 | 7:0 | RUNOVF | RXPOL | STP[1:0] | | C0EN | TXPOL | FLO[1:0] | |
| 0x02AE | U1BRG | 7:0 | BRG[7:0] | | | | | | | |
| | | 15:8 | BRG[15:8] | | | | | | | |
| 0x02B0 | U1FIFO | 7:0 | TXWRE | STPMD | TXBE | TXBF | RXIDL | XON | RXBE | RXBF |
| 0x02B1 | U1UIR | 7:0 | WUIF | ABDIF | | | | ABDIE | | |
| 0x02B2 | U1ERRIR | 7:0 | TXMTIF | PERIF | ABDOVF | CERIF | FERIF | RXBKIF | RXFOIF | TXCIF |
| 0x02B3 | U1ERRIE | 7:0 | TXMTIE | PERIE | ABDOVE | CERIE | FERIE | RXBKIE | RXFOIE | TXCIE |
| 0x02B4 | U2RXB | 7:0 | RXB[7:0] | | | | | | | |
| 0x02B5 | Reserved | | | | | | | | | |
| 0x02B6 | U2TXB | 7:0 | TXB[7:0] | | | | | | | |
| 0x02B7 | Reserved | | | | | | | | | |
| 0x02B8 | U2P1 | 7:0 | P1[7:0] | | | | | | | |
| | | 15:8 | | | | | | | | |
| 0x02BA | U2P2 | 7:0 | P2[7:0] | | | | | | | |
| | | 15:8 | | | | | | | | |
| 0x02BC | U2P3 | 7:0 | P3[7:0] | | | | | | | |
| | | 15:8 | | | | | | | | |
| 0x02BE | U2CON0 | 7:0 | BRGS | ABDEN | TXEN | RXEN | MODE[3:0] | | | |
| 0x02BF | U2CON1 | 7:0 | ON | | | WUE | RXBIMD | | BRKOVR | SENDB |
| 0x02C0 | U2CON2 | 7:0 | RUNOVF | RXPOL | STP[1:0] | | | TXPOL | FLO[1:0] | |
| 0x02C1 | U2BRG | 7:0 | BRG[7:0] | | | | | | | |
| | | 15:8 | BRG[15:8] | | | | | | | |
| 0x02C3 | U2FIFO | 7:0 | TXWRE | STPMD | TXBE | TXBF | RXIDL | XON | RXBE | RXBF |
| 0x02C4 | U2UIR | 7:0 | WUIF | ABDIF | | | | ABDIE | | |
| 0x02C5 | U2ERRIR | 7:0 | TXMTIF | PERIF | ABDOVF | CERIF | FERIF | RXBKIF | RXFOIF | |
| 0x02C6 | U2ERRIE | 7:0 | TXMTIE | PERIE | ABDOVE | CERIE | FERIE | RXBKIE | RXFOIE | |
| 0x02C7 | U3RXB | 7:0 | RXB[7:0] | | | | | | | |
| 0x02C8 | Reserved | | | | | | | | | |
| 0x02C9 | U3TXB | 7:0 | TXB[7:0] | | | | | | | |
| 0x02CA | Reserved | | | | | | | | | |
| 0x02CB | U3P1 | 7:0 | P1[7:0] | | | | | | | |
| | | 15:8 | | | | | | | | |
| 0x02CD | U3P2 | 7:0 | P2[7:0] | | | | | | | |
| | | 15:8 | | | | | | | | |
| 0x02CF | U3P3 | 7:0 | P3[7:0] | | | | | | | |
| | | 15:8 | | | | | | | | |
| 0x02D1 | U3CON0 | 7:0 | BRGS | ABDEN | TXEN | RXEN | MODE[3:0] | | | |
| 0x02D2 | U3CON1 | 7:0 | ON | | | WUE | RXBIMD | | BRKOVR | SENDB |
| 0x02D3 | U3CON2 | 7:0 | RUNOVF | RXPOL | STP[1:0] | | | TXPOL | FLO[1:0] | |
| 0x02D4 | U3BRG | 7:0 | BRG[7:0] | | | | | | | |
| | | 15:8 | BRG[15:8] | | | | | | | |
| 0x02D6 | U3FIFO | 7:0 | TXWRE | STPMD | TXBE | TXBF | RXIDL | XON | RXBE | RXBF |
| 0x02D7 | U3UIR | 7:0 | WUIF | ABDIF | | | | ABDIE | | |
| 0x02D8 | U3ERRIR | 7:0 | TXMTIF | PERIF | ABDOVF | CERIF | FERIF | RXBKIF | RXFOIF | |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

.....continued

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x02D9 | U3ERRIE | 7:0 | TXMTIE | PERIE | ABDOVE | CERIE | FERIE | RXBKIE | RXFOIE | |
| 0x02DA | U4RXB | 7:0 | RXB[7:0] | | | | | | | |
| 0x02DB | Reserved | | | | | | | | | |
| 0x02DC | U4TXB | 7:0 | TXB[7:0] | | | | | | | |
| 0x02DD | Reserved | | | | | | | | | |
| 0x02DE | U4P1 | 7:0 | P1[7:0] | | | | | | | |
| | | 15:8 | | | | | | | | |
| 0x02E0 | U4P2 | 7:0 | P2[7:0] | | | | | | | |
| | | 15:8 | | | | | | | | |
| 0x02E2 | U4P3 | 7:0 | P3[7:0] | | | | | | | |
| | | 15:8 | | | | | | | | |
| 0x02E4 | U4CON0 | 7:0 | BRGS | ABDEN | TXEN | RXEN | MODE[3:0] | | | |
| 0x02E5 | U4CON1 | 7:0 | ON | | | WUE | RXBIMD | | BRKOVR | SENDB |
| 0x02E6 | U4CON2 | 7:0 | RUNOVF | RXPOL | STP[1:0] | | | TXPOL | FLO[1:0] | |
| 0x02E7 | U4BRG | 7:0 | BRG[7:0] | | | | | | | |
| | | 15:8 | BRG[15:8] | | | | | | | |
| 0x02E9 | U4FIFO | 7:0 | TXWRE | STPMD | TXBE | TXBF | RXIDL | XON | RXBE | RXBF |
| 0x02EA | U4UIR | 7:0 | WUIF | ABDIF | | | | ABDIE | | |
| 0x02EB | U4ERRIR | 7:0 | TXMTIF | PERIF | ABDOVF | CERIF | FERIF | RXBKIF | RXFOIF | |
| 0x02EC | U4ERRIE | 7:0 | TXMTIE | PERIE | ABDOVE | CERIE | FERIE | RXBKIE | RXFOIE | |
| 0x02ED | U5RXB | 7:0 | RXB[7:0] | | | | | | | |
| 0x02EE | Reserved | | | | | | | | | |
| 0x02EF | U5TXB | 7:0 | TXB[7:0] | | | | | | | |
| 0x02F0 | Reserved | | | | | | | | | |
| 0x02F1 | U5P1 | 7:0 | P1[7:0] | | | | | | | |
| | | 15:8 | | | | | | | | |
| 0x02F3 | U5P2 | 7:0 | P2[7:0] | | | | | | | |
| | | 15:8 | | | | | | | | |
| 0x02F5 | U5P3 | 7:0 | P3[7:0] | | | | | | | |
| | | 15:8 | | | | | | | | |
| 0x02F7 | U5CON0 | 7:0 | BRGS | ABDEN | TXEN | RXEN | MODE[3:0] | | | |
| 0x02F8 | U5CON1 | 7:0 | ON | | | WUE | RXBIMD | | BRKOVR | SENDB |
| 0x02F9 | U5CON2 | 7:0 | RUNOVF | RXPOL | STP[1:0] | | | TXPOL | FLO[1:0] | |
| 0x02FA | U5BRG | 7:0 | BRG[7:0] | | | | | | | |
| | | 15:8 | BRG[15:8] | | | | | | | |
| 0x02FC | U5FIFO | 7:0 | TXWRE | STPMD | TXBE | TXBF | RXIDL | XON | RXBE | RXBF |
| 0x02FD | U5UIR | 7:0 | WUIF | ABDIF | | | | ABDIE | | |
| 0x02FE | U5ERRIR | 7:0 | TXMTIF | PERIF | ABDOVF | CERIF | FERIF | RXBKIF | RXFOIF | |
| 0x02FF | U5ERRIE | 7:0 | TXMTIE | PERIE | ABDOVE | CERIE | FERIE | RXBKIE | RXFOIE | |

# 35. SPI - Serial Peripheral Interface Module

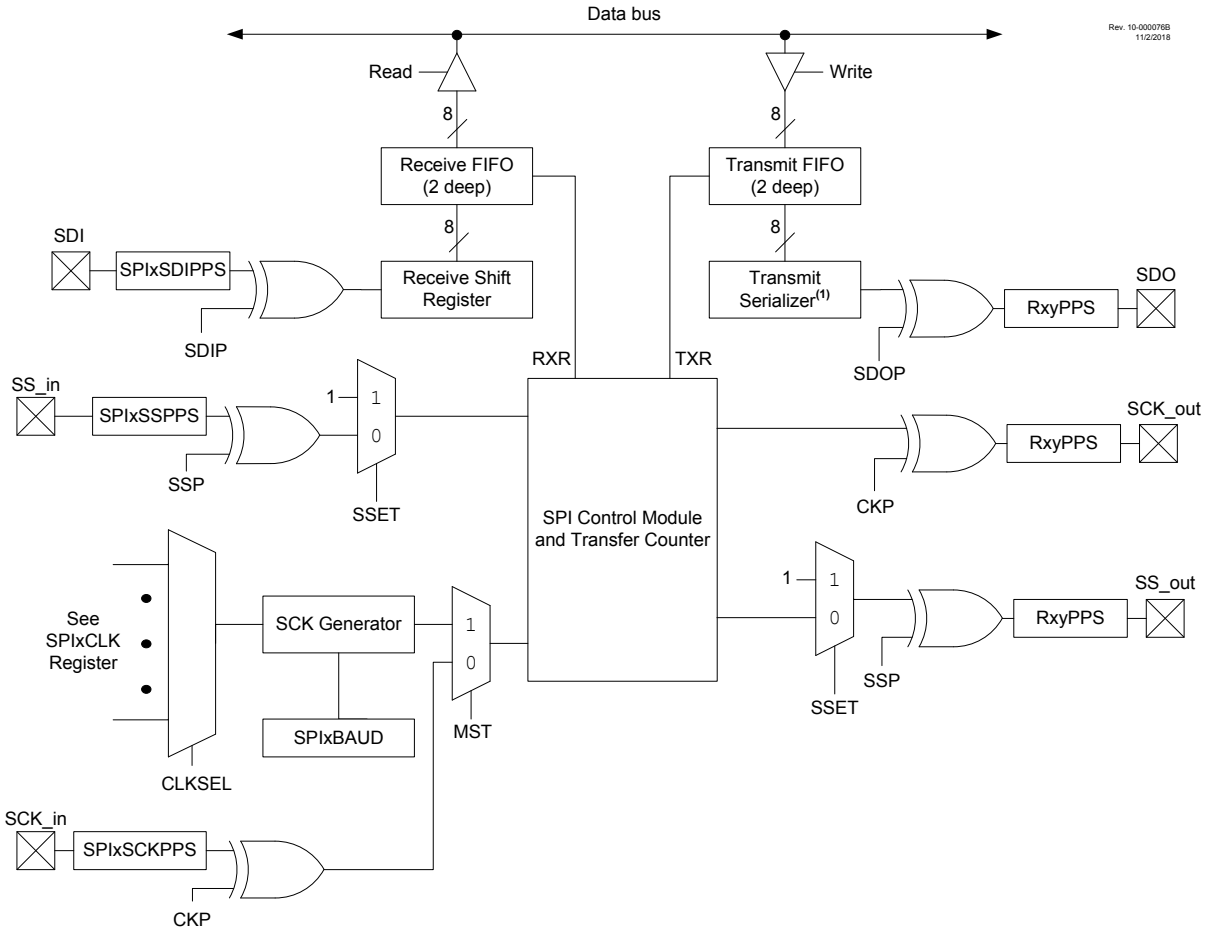
The Serial Peripheral Interface (SPI) module is a synchronous serial data communication bus that operates in Full Duplex mode. Devices communicate in a host/client environment where the host device initiates the communication. A client device is typically controlled through a chip select known as Client Select. Some examples of client devices include serial EEPROMs, shift registers, display drivers, A/D converters, and other PIC® devices with SPI capabilities.

The SPI bus specifies four signal connections:

- Serial Clock (SCK)
- Serial Data Out (SDO)
- Serial Data In (SDI)
- Client Select (SS)

The following figure shows the block diagram of the SPI module.

**Figure 35-1. SPI Module Simplified Block Diagram**




**Note:** 1. If the transmit FIFO is empty and TXR = 1, the previous value of the receive shift register will be sent to the transmit serializer.

The SPI transmit output (SDO_out) is available to the remappable PPS SDO pin and internally to the select peripherals.

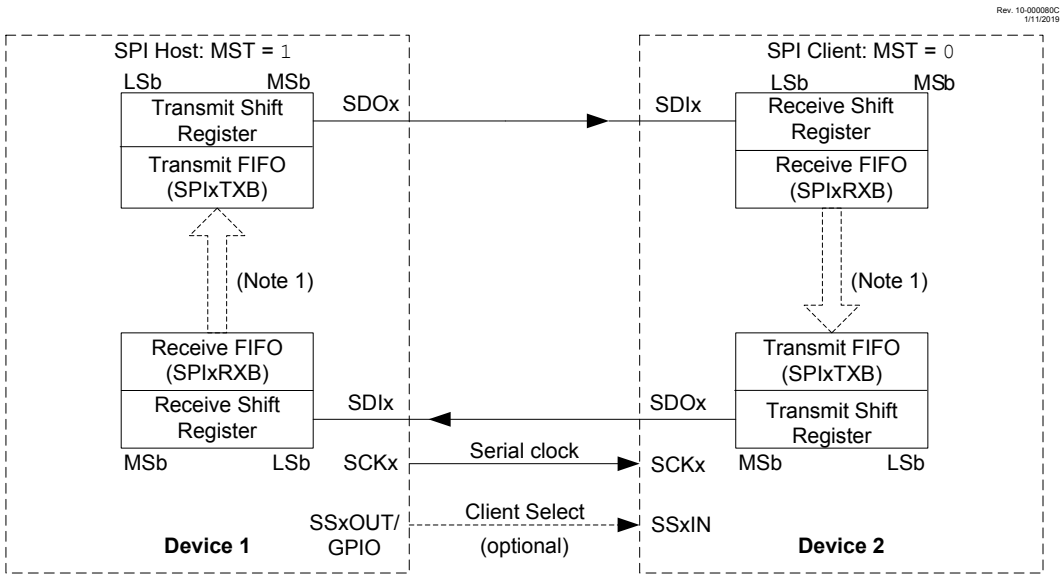
The SPI bus typically operates with a single host device and one or more client devices. When multiple client devices are used, an independent Client Select connection is required from the host device to each client device.

The host selects only one client at a time. Most client devices have tri-state outputs so their output signal appears disconnected from the bus when they are not selected.

Transmissions typically involve Shift registers, eight bits in size, one in the host and one in the client. With either the host or the client device, data is always shifted out one bit at a time, with the Most Significant bit (MSb) shifted out first. At the same time, a new bit is shifted into the device. Unlike older Microchip devices, the SPI module on this device contains one register for incoming data and another register for outgoing data. Both registers also have multibyte FIFO buffers and allow for DMA bus connections.

The figure below shows a typical connection between two devices configured as host and client devices.

**Figure 35-2. SPI Host/Client Connection with FIFOs**




**Notes:** 1. In some modes, if the Transmit FIFO is empty, the most recently received byte of data will be transmitted.
2. This diagram assumes that the LSBF bit is cleared (communications are MSb-first). When LSBF is set, the communications will be LSb-first.

Data is shifted out of the transmit FIFO on the programmed clock edge and into the receive Shift register on the opposite edge of the clock.

The host device transmits information on its SDO output pin which is connected to, and received by, the client's SDI input pin. The client device transmits information on its SDO output pin, which is connected to, and received by, the host's SDI input pin.

The host device sends out the clock signal. Both the host and the client devices need to be configured for the same clock phase and clock polarity.

During each SPI clock cycle, a full-duplex data transmission occurs. This means that while the host device is sending out the MSb from its output register (on its SDO pin) and the client device is reading this bit and saving it as the LSb of its input register. The client device is also sending out the MSb from its Shift register (on its SDO pin) and the host device is reading this bit and saving it as the LSb of its input register.

After eight bits have been shifted out, the host and client have exchanged register values and stored the incoming data into the receiver FIFOs.

If there is more data to exchange, the registers are loaded with new data and the process repeats.

Whether the data is meaningful or not (dummy data) depends on the application software. This leads to three scenarios for data transmission:

- Host sends useful data and client sends dummy data
- Host sends useful data and client sends useful data
- Host sends dummy data and client sends useful data

In this SPI module, dummy data may be sent without software involvement. Dummy transmit data is automatically handled by clearing the TXR bit and receive data is ignored by clearing the RXR bit. See Table 35-1 as well as Host Mode and Client Mode for further TXR/RXR setting details.

This SPI module can send transmissions of any number of bits, and can send information in segments of varying size (from 1-8 bits in width). As such, transmissions may involve any number of clock cycles, depending on the amount of data to be transmitted.

When there is no more data to be transmitted, the host stops sending the clock signal and deselects the client. Every client device connected to the bus that has not been selected through its Client Select line disregards the clock and transmission signals and does not transmit out any data of its own.

## 35.1 SPI Controls

The following registers control the SPI operation:

- SPI Interrupt Flag (SPIxINTF) Register
- SPI Interrupt Enable (SPIxINTE) Register
- SPI Byte Count High and Low (SPIxTCNTH/L) Registers
- SPI Bit Count (SPIxTWIDTH) Register
- SPI Baud Rate (SPIxBAUD) Register
- SPI Control (SPIxCON0) Register 0
- SPI Control (SPIxCON1) Register 1
- SPI Control (SPIxCON2) Register 2
- SPI FIFO Status (SPIxSTATUS) Register
- SPI Receiver Buffer (SPIxRXB) Register
- SPI Transmit Buffer (SPIxTXB) Register
- SPI Clock Select (SPIxCLK) Register

SPIxCON0, SPIxCON1 and SPIxCON2 are control registers for the SPI module.

SPIxSTATUS reflects the status of both the SPI module and the receive and transmit FIFOs.

SPIxBAUD and SPIxCLK control the Baud Rate Generator (BRG) of the SPI module when in Host mode. The SPIxCLK selects the clock source that is used by the BRG. The SPIxBAUD configures the clock divider used on that clock source. More information on the BRG is available in the Host Mode SPI Clock Configuration section.

SPIxTxB and SPIxRxB are the Transmit and Receive Buffer registers used to send and receive data on the SPI bus. The Transmit and Receive Buffer registers offer indirect access to Shift registers that are used for shifting the data in and out. Both registers access the multibyte FIFOs, allowing for multiple transmissions or receptions to be stored between software transfers of the data.

The SPIxTCNTH:L register pair either count or control the number of bits or bytes in a data transfer. When BMODE = 1, the SPIxTCNT value signifies bytes and the SPIxTWIDTH value signifies the number of bits in a byte. When BMODE = 0, the SPIxTCNT value is concatenated with the SPIxTWIDTH register to signify bits. In Host Receive Only mode (TXR = 0 and RXR = 1), the data transfer is initiated by writing SPIxTCNT with the desired bit or byte value to transfer. In Host Transmit mode (TXR = 1), the data transfer is initiated by writing the SPIxTxB register, in which case the SPIxTCNT is a down counter for the bits or bytes transferred.

The SPIxINTF and SPIxINTE are the flags and enables, respectively, for SPI specific interrupts. They are tied to the SPIxIF flag and SPIxIE enable bit in the PIR and PIE registers, which is triggered when any interrupt contained in the SPIxINTF/SPIxINTE registers is triggered. The PIR/PIE registers also contain SPIxTXIF/SPIxTXIE bits, which are the Interrupt flag and Enable bit for the SPI Transmit Interrupt, as well as the SPIxRXIF/SPIxRXIE bits, which are the Interrupt flag and Enable bit for the SPI receive interrupt.

## 35.2 SPI Operation

When initializing the SPI, several options need to be specified. This is done by programming the appropriate control bits of the SPIxCON0, SPIxCON1 and SPIxCON2 registers. These control bits allow the following to be configured:

© 2019-2023 Microchip Technology Inc. and its subsidiaries

- Host mode (SCK is the clock output)
- Client mode (SCK is the clock input)
- Clock Polarity (Idle state of SCK)
- Input, Output, and Client Select Polarity
- Data Input Sample Phase (middle or end of data output time)
- Clock Edge (output data on first/second edge of SCK)
- Clock Rate (Host mode only)
- Client Select mode (Host or Client mode)
- MSB-First or LSB-First
- Receive/Transmit modes:
  - Full Duplex
  - Receive Only (receive without transmit)
  - Transmit Only (transmit without receive)
- Transfer Counter mode (only available in Transmit Only mode)

### 35.2.1 Enabling and Disabling the SPI Module

Setting the EN bit enables the SPI peripheral. However, to reset or reconfigure the SPI mode, the EN bit must be cleared.

Setting the EN bit enables the SPI inputs and outputs: SDI, SDO, SCK_out, SCK_in, SS_out and SS_in. The pins for all of these inputs and outputs are selected by the PPS controls, and thus must have their functions mapped properly to the device pins to function. Refer to the **"PPS - Peripheral Pin Select Module"** chapter for more details.

SS_out and SCK_out must have the pins to which they are assigned set as outputs (TRIS bits must be '0') to properly output. Clearing the TRIS bit of the SDO pin will cause the SPI module to always control that pin, but is not necessary for SDO functionality (see the Input and Output Polarity Control section).

Configurations selected by the following registers will not be changed while the EN bit is set:

- SPIxBAUD
- SPIxCON1
- SPIxCON0 (with the exception of clearing the EN bit)

Clearing the EN bit aborts any transmissions in progress, disables the setting of interrupt flags by hardware, and resets the FIFO occupancy (see the Transmit and Receive FIFOs section).

### 35.2.2 BUSY Bit

While a data transfer is in progress, the SPI hardware sets the BUSY bit. This bit can be polled by the user to determine the current status of the SPI module, and to know when a communication is complete. The following registers and bits will not be changed by software while the BUSY bit is set:

- SPIxTCNT
- SPIxTWIDTH
- SPIxCON2
- The CLB bit

**Important:**

1. The BUSY bit is subject to synchronization delay of up to two instruction cycles. The user must wait for it to set after loading the transmit buffer (SPIxTXB register) before using it to determine the status of the SPI module.
2. It is also not recommended to read SPIxTCNT while the BUSY bit is set, as the value in the registers may not be a reliable indicator of the transfer counter. Use the TCZIF bit to accurately determine that the transfer counter has reached zero.

### 35.2.3 Transmit and Receive FIFOs

The transmission and reception of data from the SPI module is handled by two FIFOs, one for reception and one for transmission. These are addressed by the SFRs, SPIxRXB and SPIxTXB, respectively.

The transmit FIFO is written to by software and is read by the SPI module to shift the data onto the SDO pin. The receive FIFO is written to by the SPI module as it shifts in the data from the SDI pin and is read by software. Setting the CLB bit resets the occupancy for both FIFOs, emptying both buffers. The FIFOs are also reset by clearing the EN bit, thus disabling the SPI module.

**Important:** The transmit and receive FIFO occupancy refer to the number of bytes that are currently being stored in each FIFO. These values are used in this chapter to illustrate the function of these FIFOs and are not directly accessible through software.

The SPIxRXB register addresses the receive FIFO and is read-only. Reading from this register will read from the first FIFO location that was written to by hardware and decrease the receive FIFO occupancy. If the FIFO is empty, reading from this register will instead return a value of '0' and set the RXRE (Receive Buffer Read Error) bit. The RXRE bit must then be cleared in software to properly reflect the status of the read error. When the receive FIFO is full, the RXBF bit will be set.

The SPIxTXB register addresses the transmit FIFO and is write-only. Writing to the register will write to the first empty FIFO location and increase the occupancy. If the FIFO is full, writing to this register will not affect the data and will set the TXWE bit. When the transmit FIFO is empty, the TXBE bit will be set.

More details on enabling and disabling the receive and transmit functions is summarized in Table 35-1 and Client Mode Transmit Options.

### 35.2.4 LSb vs. MSb-First Operation

Typically, the SPI communication outputs the Most Significant bit first, but some devices or buses may not conform to this standard. In this case, the LSBF bit may be used to alter the order in which bits are shifted out during the data exchange. In both Host and Client mode, the LSBF bit controls whether data is shifted MSb or LSb first. Clearing the bit (default) configures the data to transfer MSb first, which conforms to traditional SPI operation, while setting the bit configures the data to transfer LSb first.

### 35.2.5 Input and Output Polarity Control

SPIxCON1 has three bits that control the polarity of the SPI inputs and outputs:

- The SDIP bit controls the polarity of the SDI input
- The SDOP bit controls the polarity of the SDO output
- The SSP bit controls the polarity of both the client SS input and the host SS output

For all three bits, when the bit is clear, the input or output is active-high, and when the bit is set, the input or output is active-low. When the EN bit is cleared, SS_out and SCK_out both revert to the Inactive state dictated by their polarity bits. The SDO Output state, when the EN bit is cleared, is determined by several factors as follows:

- When the associated TRIS bit for the SDO pin is cleared, and the SPI goes Idle after a transmission, the SDO output will remain at the last bit level.
- When the associated TRIS bit for the SDO pin is set, its behavior varies in Client and Host modes:
  - In Client mode, the SDO pin tri-states when any of the following is true:
    - Client Select is inactive
    - EN = 0
    - TXR = 0
  - In Host mode:
    - The SDO pin tri-states when TXR = 0
    - When TXR = 1 and the SPI goes Idle after a transmission, the SDO output will remain at the last bit level. The SDO pin will revert to the Idle state when EN is cleared.

### 35.2.6 Transfer Counter

In all Host modes, the transfer counter can be used to determine how many data transfers the SPI will send/receive. The transfer counter is comprised of the SPIxTCNT registers, and is also partially controlled by the SPIxTWIDTH register.

The transfer counter has two primary modes, determined by the BMODE bit. Each mode uses the SPIxTCNT and SPIxTWIDTH registers to determine the number and size of the transfers. In both modes, when the transfer counter reaches zero, the TCZIF interrupt flag is set.

**Important:**
In all Client modes and when BMODE = 1 in Host modes, the transfer counter will still decrement as transfers occur and can be used to count the number of messages sent/received, control SS_out, and trigger TCZIF. Also, when BMODE = 1, the SPIxTWIDTH register can be used in Host and Client modes to determine the size of messages sent and received by the SPI, even if the transfer counter is not being actively used to control the number of messages being sent/received by the SPI module.

#### 35.2.6.1 Total Bit Count Mode (BMODE = 0)

In this mode, SPIxTCNT and SPIxTWIDTH are concatenated to determine the total number of bits to be transferred. These bits will be loaded from/into the transmit/receive FIFOs in 8-bit increments and the transfer counter will be decremented by eight until the total number of remaining bits is less than eight. If there are any remaining bits (SPIxTWIDTH ≠ 0), the transmit FIFO will send out one final message with any extra bits greater than the remainder ignored.

The SPIxTWIDTH is the remaining bit count but the value does not change as it does for the SPIxTCNT value. The receiver will load a final byte into the receiver FIFO, and pad the extra bits with zeros. The LSBF bit determines whether the Most Significant or Least Significant bits of this final byte are ignored or padded. For example, when LSBF = 0 and the final transfer contains only two bits, then if the last byte sent was 0x5F, the RXB of the receiver will contain 0x40 which are the two MSbs of the final byte padded with zeros in the LSbs.

In this mode, the SPI host will only transmit messages when the SPIxTCNT value is greater than zero, regardless of the TXR and RXR settings.

In Host Transmit mode, the transfer starts with the data write to the SPIxTXB register or the count value written to the SPIxTCNTL register, whichever occurs last.

In Host Receive Only mode, the transfer clocks start when the SPIxTCNTL value is written. Transfer clocks are suspended when the receive FIFO is full and resume as the FIFO is read.

#### 35.2.6.2 Variable Transfer Size Mode (BMODE = 1)

In this mode, SPIxTWIDTH specifies the width of every individual piece of the data transfer in bits. SPIxTCNT specifies the number of transfers of this bit length. If SPIxTWIDTH = 0, each piece is a full byte of data. If SPIxTWIDTH ≠ 0, then only that specified number of bits from the transmit FIFO are shifted out, with the unused bits ignored.

Received data is padded with zeros in the unused bit areas when transferred into the receive FIFO. The LSBF bit determines whether the Most Significant or Least Significant bits of the transfers are ignored or padded.

In this mode, the transfer counter being zero only stops messages from being sent or received when in Receive Only mode.

**Important:**
With BMODE = 1, it is possible for the transfer counter (SPIxTCNT) to decrement below zero, although when in Host Receive Only mode, transfer clocks will cease when the transfer counter reaches zero.

#### 35.2.6.3 Transfer Counter in Client Mode

In Client mode, the transfer counter will still decrement as data is shifted in and out of the SPI module, but it will not control data transfers. The BMODE bit along with the transfer counter is used to determine when the device will look for Client Select faults.

When BMODE = 0, the SSFLT bit will be set if Client Select transitions from its Active to Inactive state during bytes of data, or if it transitions before the last bit sent during the final byte (if SPIxTWIDTH ≠ 0).

When BMODE = 1, the SSFLT bit will be set if Client Select transitions from its Active to Inactive state before the final bit of each individual transfer is completed.

**Note:** SSFLT does not have an associated interrupt, so it will be checked in software. An ideal time to do this is when the End of Client Select Interrupt (EOSIF) is triggered (see the Start of Client Select and End of Client Select Interrupts section).

## 35.3 Host Mode

In Host mode, the device controls the SCK line, and as such, initiates data transfers and determines when any clients broadcast data onto the SPI bus.

Host mode can be configured in four different modes, configured by the TXR and RXR bits:

- Full Duplex mode
- Receive Only mode
- Transmit Only mode
- Transfer Off mode

The modes are illustrated in the following table:

**Table 35-1. Host Mode TXR/RXR Settings**

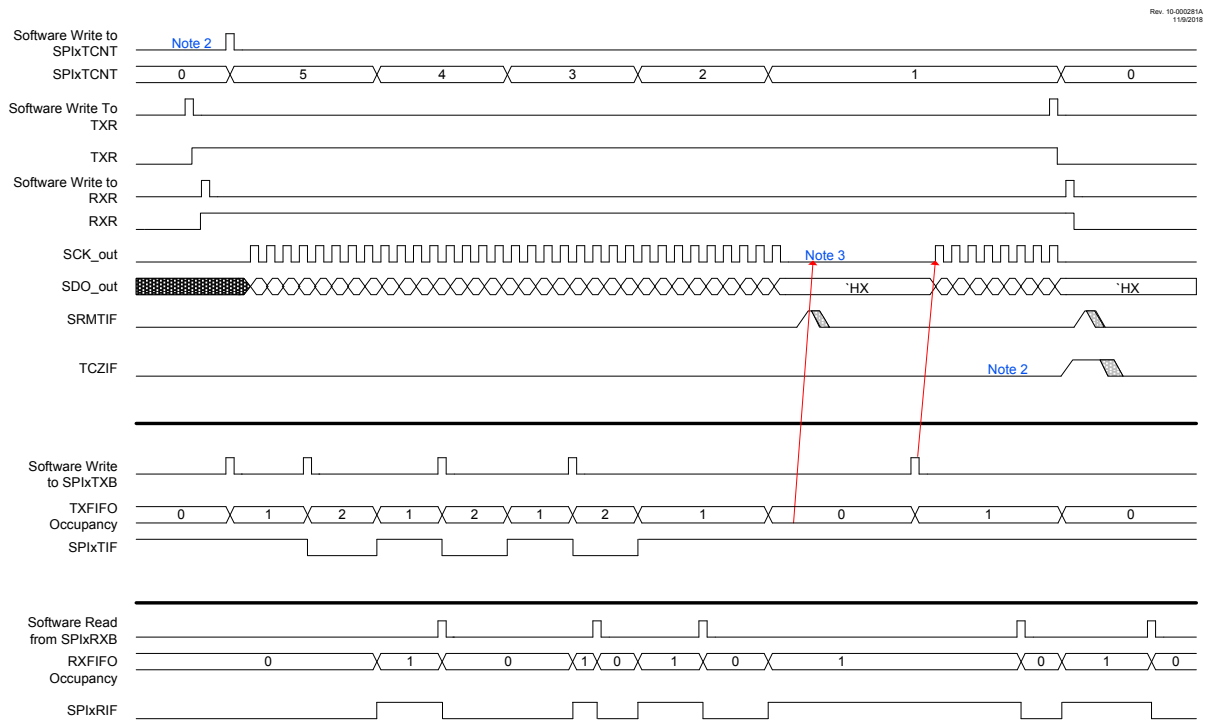
| | TXR = 1 | TXR = 0 |
|---|---|---|
| **RXR = 1** | **Full Duplex mode**<br>BMODE = 1: Transfer when RxFIFO is not full and TxFIFO is not empty<br>BMODE = 0: Transfer when RXFIFO is not full, TXFIFO is not empty, and the Transfer Counter is nonzero | **Receive Only mode**<br>Transfer when RxFIFO is not full and the Transfer Counter is nonzero<br>Transmitted data is either the top of the FIFO or the most recently received data |
| **RXR = 0** | **Transmit Only mode**<br>BMODE = 1: Transfer when TxFIFO is not empty<br>BMODE = 0: Transfer when TXFIFO is not empty and the Transfer Counter is nonzero<br>Received data is not stored | No Transfers |

### 35.3.1 Full Duplex Mode

When both TXR and RXR are set, the SPI host is in Full Duplex mode. In this mode, data transfer triggering is affected by the BMODE bit.

When BMODE = 1, data transfers will occur whenever the receive FIFO is not full and data is present in the transmit FIFO. In practice, as long as the receive FIFO is not full, data will be transmitted/received as soon as the SPIxTXB register is written to, matching the functionality of SPI (MSSP) modules on older 8-bit Microchip devices. The SPIxTCNT will decrement with each transfer. However, when SPIxTCNT is zero, the next transfer is not inhibited and the corresponding SPIxTCNT decrement will cause the count to roll over to the maximum value. The following figure shows an example of a communication using this mode.

**Figure 35-3. SPI Host Operation - Data Exchange, RXR = 1, TXR = 1**




**Notes:** 1. SS(out) is not shown on this diagram.
2. SPIxTCNT write is optional when TXR/RXR = 1/1 and BMODE = 1. If BMODE = 0, a write to SPIxTCNT is required to start transmission; TCZIF signals the transition of SPIxTCNT from 1 to 0.
3. Transmission gap occurs while waiting for transmitter data.

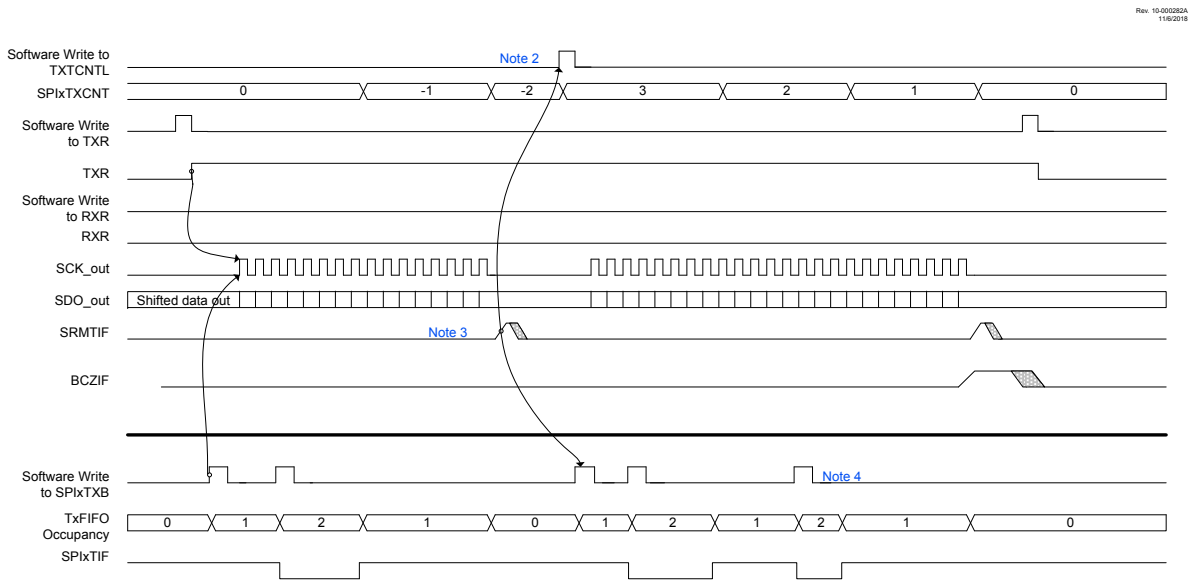
When BMODE = 0, the transfer counter (SPIxTCNT) must also be written to before transfers will occur. Transfers will cease when the transfer counter reaches '0'. For example, if SPIxTXB is written twice and then SPIxTCNTL is written with '3', the transfer will start with the SPIxTCNTL write. The two bytes in the TXFIFO will be sent after which the transfer will suspend until the third and last byte is written to SPIxTXB.

### 35.3.2 Transmit Only Mode

When TXR is set and RXR is clear, the SPI host is in Transmit Only mode. In this mode, data transfer triggering is affected by the BMODE bit.

When BMODE = 1, data transfers will occur whenever the transmit FIFO is not empty. Data will be transmitted as soon as the SPIxTXB register is written to, matching the functionality of the SPI (MSSP) modules on previous 8-bit devices. The SPIxTCNT will decrement with each transfer. However, when SPIxTCNT is zero the next transfer is not inhibited and the corresponding SPIxTCNT decrement will cause the count to roll over to the maximum value. Any data received in this mode is not stored in the receive FIFO. The following figure shows an example of sending a command and then sending a byte of data, using this mode.

**Figure 35-4. SPI Host Operation - Command+Write Data, TXR = 1, RXR = 0**




**Notes:** 1. SS_out is not shown.
2. The byte counter is optional when TXR/RXR = 1/0.
3. After the command bytes, wait for SRMTIF before loading SPIxTXB, otherwise the command data will decrement SPIxTXCNT. Alternatively, load SPIxTXCNT = 5 and count the command bytes also; TCZIF signals the end of the transmission.
4. Transmit data interrupt handler (or DMA) must write only the bytes necessary; the byte counter is not available as an indicator.
5. Reading the SPIxRXB is not required because RXR = 0.

When BMODE = 0, the transfer counter (SPIxTCNT) must also be written to before transfers will occur, and transfers will cease when the transfer counter reaches '0'.

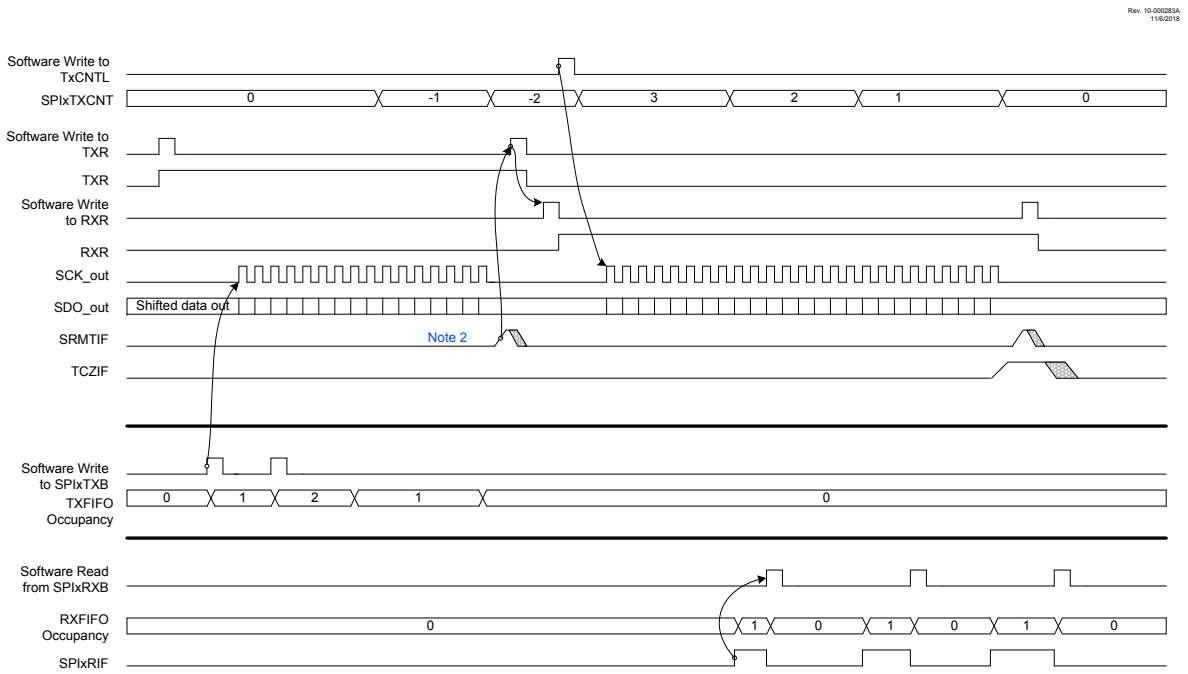
For example, if SPIxTXB is written twice and then SPIxTCNTL is written with '3', the transfer will start with the SPIxTCNTL write. The two bytes in the TXFIFO will be sent after which the transfer will suspend until the third and last byte is written to SPIxTXB.

### 35.3.3 Receive Only Mode

When RXR is set and TXR is clear, the SPI host is in Receive Only mode. In this mode, data transfers when the receive FIFO is not full and the transfer counter is nonzero. In this mode, writing a value to SPIxTCNTL will start the clocks for transfer. The clocks will suspend while the receive FIFO is full and cease when the SPIxTCNT reaches zero (see the Transfer Counter section). If there is any data in the transmit FIFO, the first data written to SPIxTXB will be transmitted on each data exchange, although the transmit FIFO occupancy will not change, meaning that the same message will be sent on each transmission. If there is no data in the transmit FIFO, the most recently received data will be transmitted. The following figure shows an example of sending a command using the Transmit Only mode and then receiving a byte of data using the Receive Only mode.

**Important:** When operating in Receive Only mode and the size of every SPI transaction is less than 8 bits, it is recommended to operate in BMODE = 1 mode. The size of the packet can be configured using the SPIxTWIDTH register.

**Figure 35-5. SPI Host Operation - Command+Read Data, TXR = 0, RXR = 1**




**Notes:** 1. SS_out is not shown.
2. Software must wait for shift-register empty (SRMTIF) before changing TXR, RXR, SPIxTCNT and SPIxTWIDTH controls. This is not considered an imposition in this case, because the client likely needs time to load output data.

### 35.3.4 Transfer Off Mode

When both TXR and RXR are cleared, the SPI host is in Transfer Off mode. In this mode, SCK will not toggle and no data is exchanged. However, writes to SPIxTXB will be transferred to the transmit FIFO which will then be transmitted when the TXR bit is set.

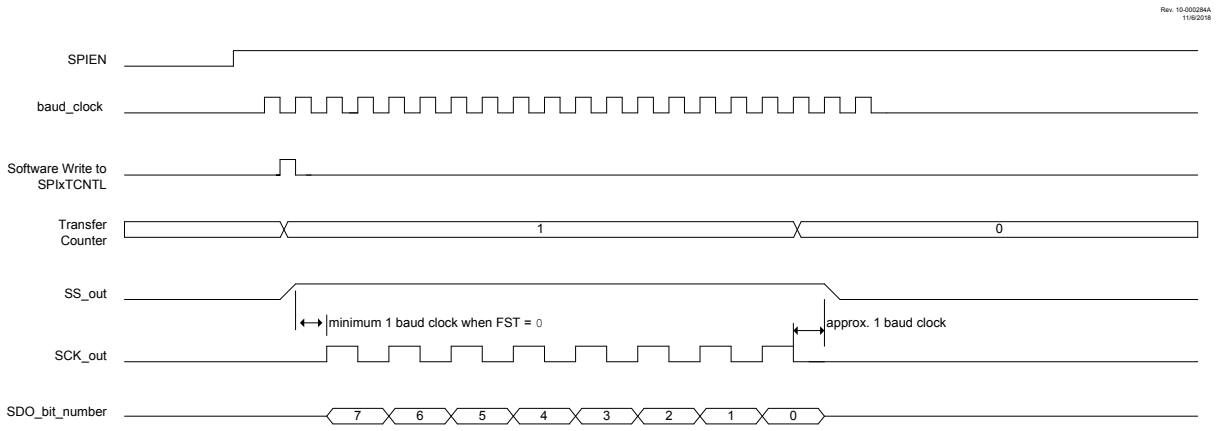
### 35.3.5 Host Mode Client Select Control

#### 35.3.5.1 Hardware Client Select Control

The SPI module allows for direct hardware control of a Client Select output. The Client Select output (SS_out) is controlled both directly, through the SSET bit, and indirectly by the hardware while the transfer counter is nonzero (see the Transfer Counter section). The SS_out pin is selected with the PPS controls. The SS_out polarity is controlled by the SSP bit.

Setting the SSET bit will assert SS_out. Clearing the SSET bit will leave SS_out to be controlled by the transfer counter. When the transfer counter is loaded, the SPI module will automatically assert SS_out. When the transfer counter decrements to zero, the SPI module will deassert SS_out either one baud period after the final SCK pulse of the final transfer (when CKE/SMP = 0/1) or one half baud period otherwise, as shown in the following figure.

**Figure 35-6. SPI Host SS Operation - CKE = 0, BMODE = 1, TWIDTH = 0, SSP = 0**




**Notes:** 1. *SDO bit number* illustrates the transmitted bit number, and is not intended to imply SDO_out tristate operation.
2. Assumes SPIxTXB holds data when SPIxTCNTL is written.

#### 35.3.5.2 Software Client Select Control

Client Select can be controlled through software via a general purpose I/O pin. In this case, ensure that the desired pin is configured as a general purpose output with the PPS and TRIS controls. In this case, SSET will not affect the Client Select, the Transfer Counter will not automatically control the Client Select output, and all setting and clearing of the Client Select output line must be directly controlled by software.

### 35.3.6 Host Mode SPI Clock Configuration

#### 35.3.6.1 SPI Clock Selection

The clock source for SPI Host modes is selected by the SPIxCLK register.

The SPIxBAUD register allows for dividing this clock. The frequency of the SCK output is defined by the following equation:

**Equation 35-1. SCK Output Frequency**

$$F_{BAUD} = \frac{F_{CSEL}}{2 \times (BAUD + 1)}$$

where $F_{BAUD}$ is the baud rate frequency output on the SCK pin, $F_{CSEL}$ is the frequency of the input clock selected by the SPIxCLK register, and BAUD is the value contained in the SPIxBAUD register.

#### 35.3.6.2 Clock and Data Change Alignment

The CKP, CKE and SMP bits control the relationship between the SCK clock output, SDO output data changes, and SDI input data sampling. The bit functions are as follows:

- CKP controls SCK output polarity
- CKE controls SDO output change relative to the SCK clock
- SMP controls SDI input sampling relative to the clock edges

The CKE bit, when set, inverts the low Idle state of the SCK output to a high Idle state.

The following figures illustrate the eight possible combinations of the CKP, CKE and SMP bit selections.

**Important:** All timing diagrams assume the LSBF bit is cleared.

**Figure 35-7. Clocking Detail - Host Mode, CKE = 0, SMP = 0**

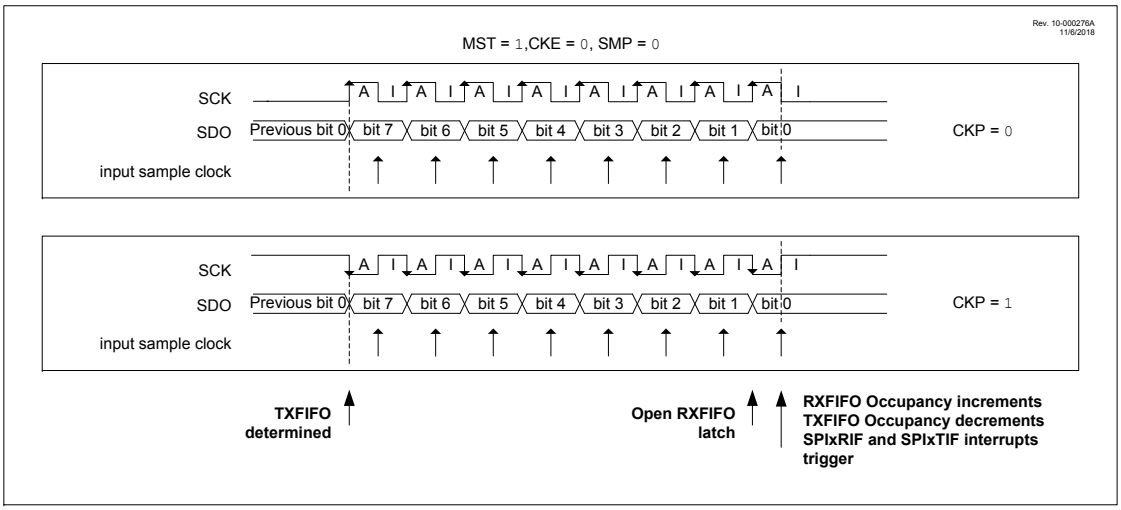



**Figure 35-8. Clocking Detail - Host Mode, CKE = 1, SMP = 1**

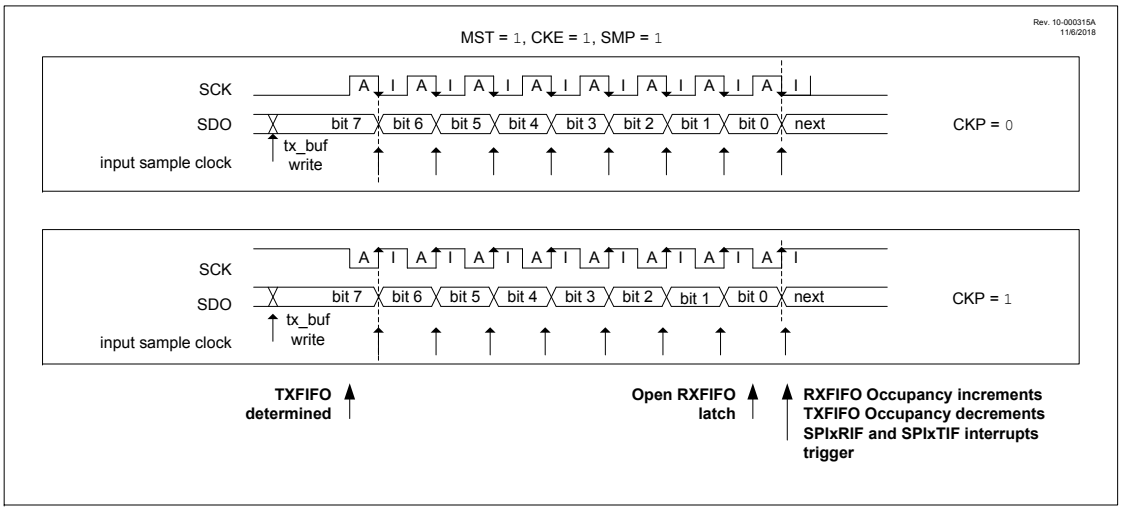


**Figure 35-9. Clocking Detail - Host Mode, CKE = 0, SMP = 1**

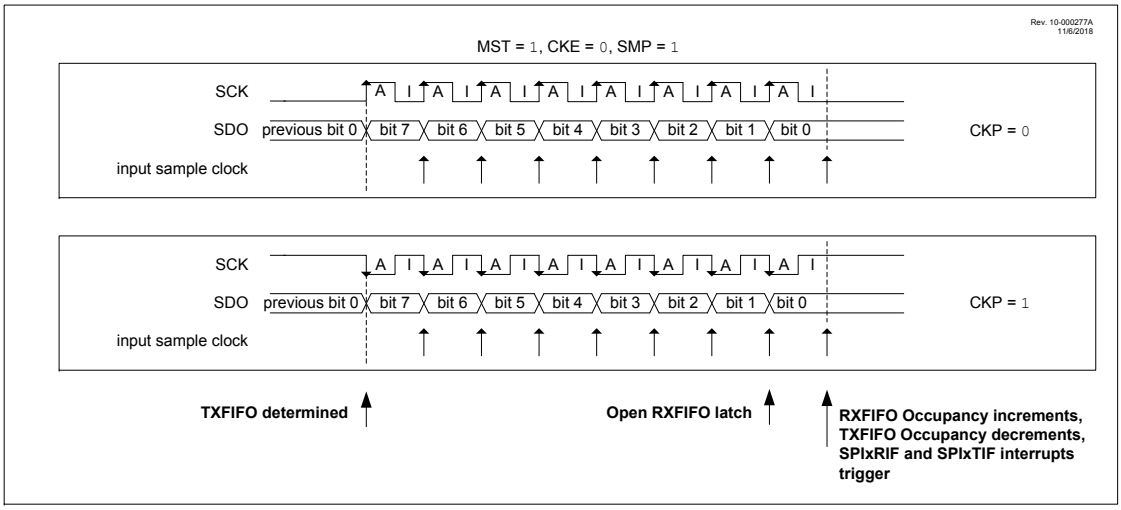
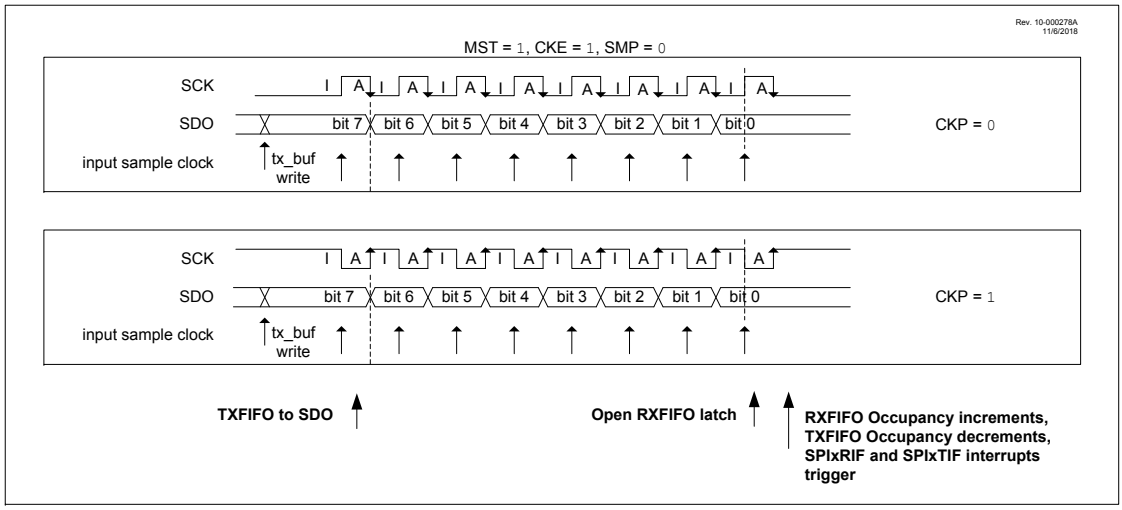



**Figure 35-10. Clocking Detail - Host Mode, CKE = 1, SMP = 0**




### 35.3.6.3 SCK Start-Up Delay

When starting an SPI data exchange, the host device asserts the SS output, by either setting the SSET bit or loading the TCNT value, and then triggers the module to send data by writing SPIxTXB. These data triggers are synchronized to the clock selected by the SPIxCLK register before the first SCK pulse appears, usually requiring one or two clock periods of the selected SPI source clock.

The SPI module includes additional synchronization delays on SCK generation specifically designed to ensure that the Client Select output timing is correct, without requiring precision software timing loops. By default, this synchronization delay is ½ baud period.

When the value of the SPIxBAUD register is a small number (indicating higher SCK frequencies), the code execution delay between asserting SS and writing SPIxTXB is relatively long compared to the added synchronization delay before the first SCK edge. With larger values of SPIxBAUD (indicating lower SCK frequencies), the code execution delay is much smaller relative to the synchronization delay. Therefore, the first SCK edge after SS is asserted will be closer to the synchronization delay.

Setting the FST bit removes the synchronization delay, allowing systems with low SPIxBAUD values (and thus, long synchronization delays) to forgo this extra delay, in which case the time between the SS assertion and the first SCK edge depends entirely on the code execution delay.

## 35.4 Client Mode

### 35.4.1 Client Mode Transmit Options

The SDO output of the SPI module in Client mode is controlled by the following:

- TXR bit
- TRIS bit associated with the SDO pin
- Client Select input
- Current state of the transmit FIFO

This control is summarized in the following table where TRISxn refers to the bit in the TRIS register corresponding to the pin that SDO has been assigned with PPS, TXR is the Transmit Data Required Control bit, SS is the state of the Client Select input, and TXBE is the transmit FIFO Buffer Empty bit.

**Table 35-2. Client Mode Transmit**

| TRISxn(1) | TXR | SS | TXBE | SDO State |
|---|---|---|---|---|
| 0 | 0 | FALSE | 0 | Drives state determined by LATxn(2) |
| 0 | 0 | FALSE | 1 | Drives state determined by LATxn(2) |
| 0 | 0 | TRUE | 0 | Outputs the oldest byte in the transmit FIFO Does not remove data from the transmit FIFO |
| 0 | 0 | TRUE | 1 | Outputs the most recently received byte |
| 0 | 1 | FALSE | 0 | Drives state determined by LATxn(2) |
| 0 | 1 | FALSE | 1 | Drives state determined by LATxn(2) |
| 0 | 1 | TRUE | 0 | Outputs the oldest byte in the transmit FIFO<br>Removes transmitted byte from the transmit FIFO<br>Decrements occupancy of transmit FIFO |
| 0 | 1 | TRUE | 1 | Outputs the most recently received byte<br>Sets the TXUIF bit |
| 1 | 0 | FALSE | 0 | Tri-stated |
| 1 | 0 | FALSE | 1 | Tri-stated |
| 1 | 0 | TRUE | 0 | Tri-stated |
| 1 | 0 | TRUE | 1 | Tri-stated |
| 1 | 1 | FALSE | 0 | Tri-stated |
| 1 | 1 | FALSE | 1 | Tri-stated |
| 1 | 1 | TRUE | 0 | Outputs the oldest byte in the transmit FIFO<br>Removes transmitted byte from the transmit FIFO<br>Decrements the FIFO occupancy |
| 1 | 1 | TRUE | 1 | Outputs the most recently received byte<br>Sets the TXUIF bit |

**Notes:**

1. TRISxn is the bit in the TRISx register corresponding to the pin to which SDO has been assigned with PPS.
2. LATxn is the bit in the LATx register corresponding to the pin to which SDO has been assigned with PPS.

#### 35.4.1.1 SDO Drive/Tri-State

The TRIS bit associated with the SDO pin controls whether the SDO pin will tri-state. When this TRIS bit is cleared, the pin will always be driving to a level, even when the SPI module is inactive. When the SPI module is inactive

(either due to the host not clocking the SCK line or the SS being false), the SDO pin will be driven to the value of the LAT bit associated with the SDO pin. When the SPI module is active, its output is determined by both TXR and whether there is data in the transmit FIFO.

When the TRIS bit associated with the SDO pin is set, the pin will only have an output level driven to it when TXR = 1 and the Client Select input is true. In all other cases, the pin will be tri-stated.

**Table 35-3. Client Mode Transmit**

| TRISxn(1) | TXR | SS | TXBE | SDO State |
|---|---|---|---|---|
| 0 | 0 | FALSE | 0 | Output level determined by LATxn(2) |
| 0 | 0 | FALSE | 1 | Output level determined by LATxn(2) |
| 0 | 0 | TRUE | 0 | Outputs the oldest byte in the TXFIFO.<br>Does not remove data from the TXFIFO. |
| 0 | 0 | TRUE | 1 | Outputs the most recently received byte |
| 0 | 1 | FALSE | 0 | Output level determined by LATxn(2) |
| 0 | 1 | FALSE | 1 | Output level determined by LATxn(2) |
| 0 | 1 | TRUE | 0 | Outputs the oldest byte in the TXFIFO.<br>Removes transmitted byte from the TXFIFO.<br>Decrements occupancy of TXFIFO. |
| 0 | 1 | TRUE | 1 | Outputs the most recently received byte.<br>Sets the TXUIF bit. |
| 1 | 0 | FALSE | 0 | Tri-stated |
| 1 | 0 | FALSE | 1 | Tri-stated |
| 1 | 0 | TRUE | 0 | Tri-stated |
| 1 | 0 | TRUE | 1 | Tri-stated |
| 1 | 1 | FALSE | 0 | Tri-stated |
| 1 | 1 | FALSE | 1 | Tri-stated |
| 1 | 1 | TRUE | 0 | Outputs the oldest byte in the TXFIFO.<br>Removes transmitted byte from the TXFIFO.<br>Decrements occupancy of TXFIFO. |
| 1 | 1 | TRUE | 1 | Outputs the most recently received byte.<br>Sets the TXUIF bit. |

**Notes:**

1. TRISxn is the bit in the TRISx register corresponding to the pin that SDO has been assigned with PPS.
2. LATxn is the bit in the LATx register corresponding to the pin that SDO has been assigned with PPS.

### 35.4.1.2 SDO Output Data

The TXR bit controls the nature of the data that is transmitted in Client mode. When TXR is set, transmitted data is taken from the transmit FIFO. If the FIFO is empty, the most recently received data will be transmitted and the TXUIF flag will be set to indicate that a transmit FIFO underflow has occurred.

When TXR is cleared, the data will be taken from the transmit FIFO, and the FIFO occupancy will not decrease. If the transmit FIFO is empty, the most recently received data will be transmitted, and the TXUIF bit will not be set. However, if the TRIS bit associated with the SDO pin is set, clearing the TXR bit will cause the SPI module to not output any data to the SDO pin.

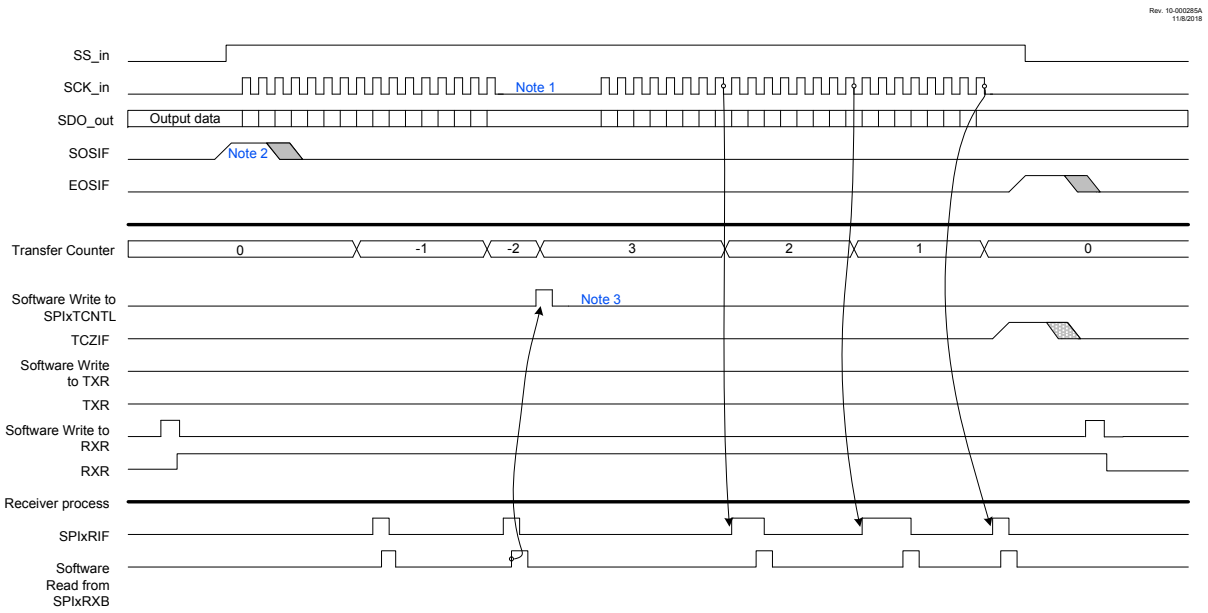
© 2019-2023 Microchip Technology Inc.
and its subsidiaries

### 35.4.2 Client Mode Receive Options

The RXR bit controls the nature of receptions in Client mode. When RXR is set, the SDI input data will be stored in the receive FIFO if it is not full. If the receive FIFO is full, the RXOIF bit will be set to indicate a receive FIFO overflow error and the data is discarded. When RXR is cleared, all received data will be ignored and not stored in the receive FIFO (although it may still be used for transmission if the transmit FIFO is empty).

The following figure presents a typical Client mode communication, showing a case where the host writes two then three bytes, showing interrupts as well as the behavior of the transfer counter in Client mode (see the Transfer Counter in Client Mode section for more details on the transfer counter in Client mode as well as the SPI Interrupts section for more information on interrupts).

**Figure 35-11. SPI Client Mode Operation – Interrupt-Driven, Host Writes 2+3 Bytes**




**Notes:** 1. This delay is exaggerated for illustration, and can be as short as1/2 bit period.
2. If the device is sleeping, SOSIF will wake it up for interrupt service.
3. Setting SPIxTCNTL is optional in this example, otherwise it will count -3, -4, -5, and TCZIF will not occur.

### 35.4.3 Client Mode Client Select

In Client mode, an external Client Select signal can be used to synchronize communication with the host device. The Client Select line is held in its Inactive state (high by default) until the host device is ready to communicate. When the Client Select transitions to its Active state, the client knows that a new transmission is starting.

When the Client Select goes false at the end of the transmission, the receive function of the selected SPI client device returns to the Inactive state. The client is then ready to receive a new transmission when the Client Select goes true again.

The Client Select signal is received on the SS input pin. This pin is selected with the SPIxSSPPS register (refer to the **"PPS Inputs"** section). When the input on this pin is true, transmission and reception are enabled, and the SDO pin is driven. When the input on this pin is false, the SDO pin is either tri-stated (if the TRIS bit associated with the SDO pin is set) or driven to the value of the LAT bit associated with the SDO pin (if the TRIS bit associated with the SDO pin is cleared). The SCK input is ignored when the SS input is false.

If the SS input goes false, while a data transfer is still in progress, it is considered a Client Select fault. The SSFLT bit indicates whether such an event has occurred. The transfer counter value determines the number of bits in a valid data transfer (see the Transfer Counter section for more details).

The Client Select polarity is controlled by the SSP bit. When SSP is set (its default state), the Client Select input is active-low, and when it is cleared, the Client Select input is active-high.

The Client Select for the SPI module is controlled by the SSET bit. When SSET is cleared (its default state), the Client Select will act as described above. When the bit is set, the SPI module will behave as if the SS input is always in its Active state.

**Important:**
When SSET is set, the effective SS_in signal is always active. Hence, the SSFLT bit may be disregarded.

### 35.4.4 Client Mode Clock Configuration

In Client mode, SCK is an input, and must be configured to the same polarity and clock edge as the host device. As in Host mode, the polarity of the clock input is controlled by the CKP bit and the clock edge used for transmitting data is controlled by the CKE bit.

### 35.4.5 Daisy-Chain Configuration

The SPI bus can be connected in a daisy-chain configuration. The first client output is connected to the second client input, the second client output is connected to the third client input, and so on. The final client output is connected to the host input. Each client sends out, during a second group of clock pulses, an exact copy of what was received during the first group of clock pulses. The whole chain acts as one large communication shift register. The daisy-chain feature only requires a single Client Select line from the host device connected to all client devices (alternately, the client devices can be configured to ignore the Client Select line by setting the SSET bit).

In a typical daisy-chain configuration, the SCK signal from the host is connected to each of the client device SCK inputs. However, the SCK input and output are separate signals selected by the PPS control. When the PPS selection is made to configure the SCK input and SCK output on separate pins, then the SCK output will follow the SCK input, allowing for SCK signals to be daisy-chained like the SDO/SDI signals.

The following two figures show block diagrams of a typical daisy-chain connection, and a daisy-chain connection with daisy-chained SPI clocks, respectively.

**Figure 35-12. Traditional SPI Daisy-Chain Connection**

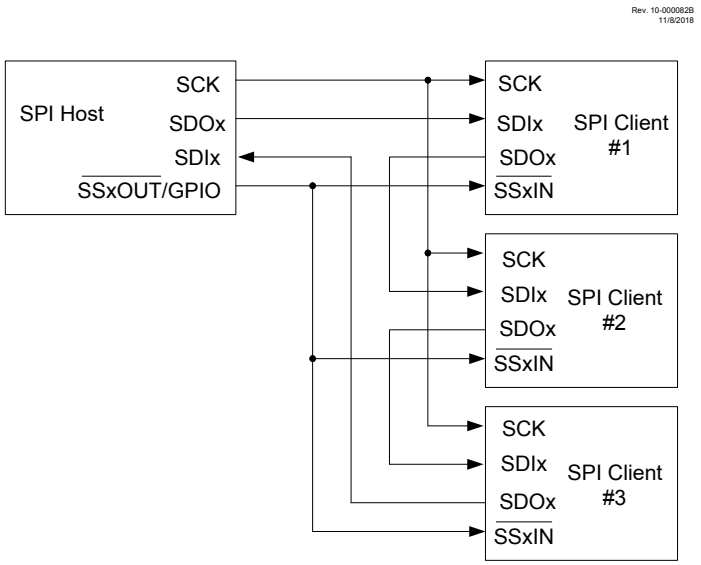
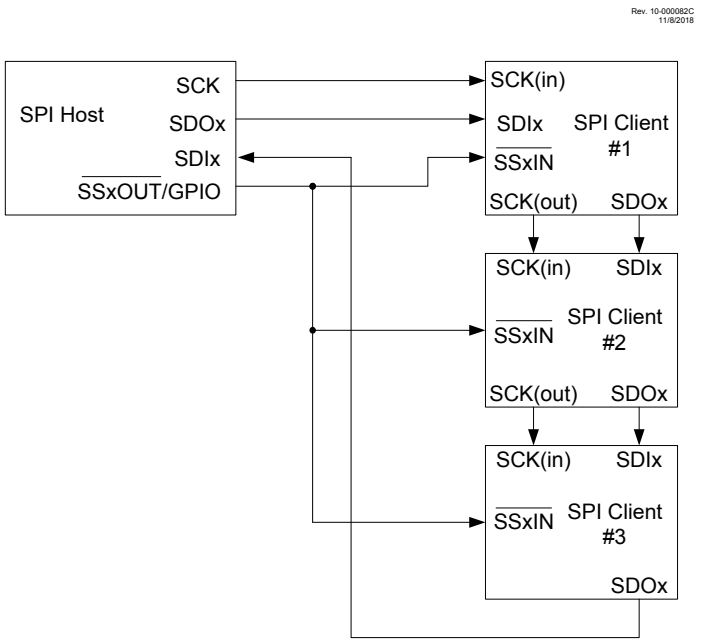


**Figure 35-13. SPI Daisy-Chain Connection with Chained SCK**




## 35.5 SPI Operation in Sleep Mode

The SPI Host mode will operate in Sleep, provided the clock source selected by SPIxCLK is active in Sleep mode. FIFOs will operate as they do when the part is awake. When TXR = 1, the transmit FIFO will need to contain data for transfers to take place in Sleep. All interrupts will still set the interrupt flags in Sleep, but only enabled interrupts will wake the device from Sleep.

The SPI Client mode will operate in Sleep because the clock is provided by an external host device. FIFOs will still operate and interrupts will set interrupt flags, and enabled interrupts will wake the device from Sleep.

## 35.6 SPI Interrupts

There are three top level SPI interrupts in the PIRx register:

- SPI Transmit (SPIxTXIF)
- SPI Receive (SPIxRXIF)
- SPI Module status (SPIxIF)

The SPI Module status interrupts are enabled at the module level in the SPIxINTE register. Only enabled status interrupts will cause the single top level SPIxIF flag to be set.

### 35.6.1 SPI Receive Interrupt

The SPI receive interrupt is set when the receive FIFO contains data, and is cleared when the receive FIFO is empty. The interrupt flag, SPIxRXIF, is located in one of the PIR registers. The interrupt enable, SPIxRXIE, is located in the corresponding PIE register. The SPIxRXIF interrupt flag is read-only.

### 35.6.2 SPI Transmit Interrupt

The SPI Transmit interrupt is set when the transmit FIFO is not full and can accept a character, and is cleared when the transmit FIFO is full and cannot accept a character. The interrupt flag, SPIxTXIF, is located in one of the PIR registers. The interrupt enable, SPIxTXIE, is located in the corresponding PIE register. The SPIxTXIF interrupt flag is read-only.

© 2019-2023 Microchip Technology Inc. and its subsidiaries

### 35.6.3 SPI Status Interrupts

The SPIxIF flag is located in one of the PIR registers. This flag is set when any of the individual status flags in SPIxINTF and their respective SPIxINTE bits are set. For any specific interrupt flag to interrupt normal program flow, both the SPIxIE bit, in the PIE register corresponding to the PIR register, and the specific bit in SPIxINTE associated with that interrupt must be set.

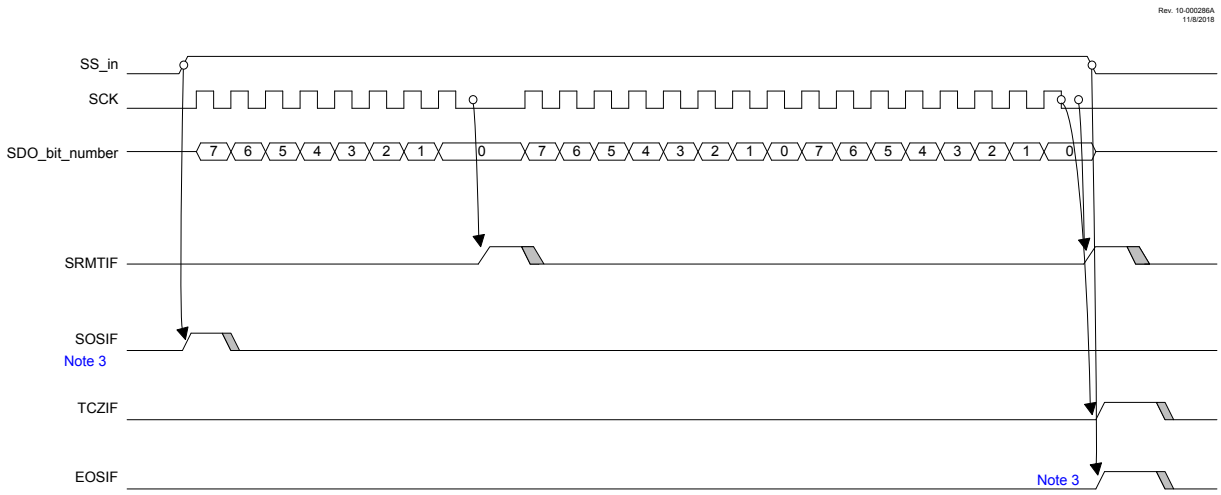
The Status Interrupts include the following:

- Shift Register Empty (SRMTIF)
- Transfer Counter is Zero (TCZIF)
- Start of Client Select (SOSIF)
- End of Client Select (EOSIF)
- Receiver Overflow (RXOIF)
- Transmitter Underflow (TXUIF)

#### 35.6.3.1 Shift Register Empty Interrupt

The Shift Register Empty Interrupt Flag and Shift Register Empty Interrupt Enable are the SRMTIF and SRMTIE bits respectively. This interrupt is only available in Host mode and triggers when a data transfer completes and conditions are not present to start a new transfer, as dictated by the TXR and RXR bits (see Table 35-1 for conditions for starting a new Host mode data transfer with different TXR/ RXR settings). This interrupt will be triggered at the end of the last full bit period, after SCK has been low for one ½-baud period. See the figure below for more details of the timing of this interrupt as well as other interrupts. This bit will not clear itself when the conditions for starting a new transfer occur, and must be cleared in software.

**Figure 35-14. Transfer And Client Select Interrupt Timing**




**Notes:**
1. SRMTIF available only in Host mode.
2. Clearing of interrupt flags is shown for illustration; actual interrupt flags must be cleared in software.
3. SOSIF and EOSIF are set according to SS_in, even in Host mode.

#### 35.6.3.2 Transfer Counter Is Zero Interrupt

The Transfer Counter Is Zero Interrupt Flag and Transfer Counter Is Zero Interrupt Enable are the TCZIF and TCZIE bits, respectively. This interrupt will trigger when the transfer counter (defined by BMODE, SPIxTCNT and SPIxTWIDTH) decrements from one to zero. See Figure 35-14 for more details on the timing of this interrupt as well as other interrupts. This bit must be cleared in software.

**Important:**
The TCZIF flag only indicates that the transfer counter has decremented from one to zero, and may not indicate that the entire data transfer process is complete. Either poll the BUSY bit and wait for it to be cleared or use the Shift Register Empty Interrupt (SRMTIF) to determine when a data transfer is fully complete.

#### 35.6.3.3 Start of Client Select and End of Client Select Interrupts

The Start of Client Select Interrupt Flag and Start of Client Select Interrupt Enable are the SOSIF and SOSIE bits, respectively. The End of Client Select Interrupt Flag and End of Client Select Interrupt Enable are the EOSIF and EOSIE bits, respectively. These interrupts trigger at the leading and trailing edges of the Client Select input.

The interrupts are active in both Host and Client mode, and will trigger on transitions of the Client Select input regardless of which mode the SPI is in. In Host mode, the PPS controls will be used to assign the Client Select input to the same pin as the Client Select output, allowing these interrupts to trigger on changes to the Client Select output.

In Client mode, changing the SSET bit can trigger these interrupts, as it changes the effective input value of Client Select.

Both SOSIF and EOSIF must be cleared in software.

#### 35.6.3.4 Receiver Overflow and Transmitter Underflow Interrupts

The receiver overflow interrupt triggers if data is received when the receive FIFO is already full and RXR = 1. In this case, the data will be discarded and the RXOIF bit will be set. The Receiver Overflow Interrupt Enable bit is RXOIE.

The Transmitter Underflow Interrupt flag triggers if a data transfer begins when the transmit FIFO is empty and TXR = 1. In this case, the most recently received data will be transmitted and the TXUIF bit will be set. The Transmitter Underflow Interrupt Enable bit is TXUIE.

Both these interrupts will only occur in Client mode, as Host mode will not allow the receive FIFO to overflow or the transmit FIFO to underflow.

## 35.7 Register Definitions: Serial Peripheral Interface

Long bit name prefixes for the SPI peripherals are shown in the table below where “x” refers to the SPI instance number. Refer to the **“Long Bit Names”** section in the **“Register and Bit Naming Conventions”** chapter for more information.

**Table 35-4. SPI Bit Name Prefixes**

| Peripheral | Bit Name Prefix |
|---|---|
| SPI1 | SPI1 |
| SPI2 | SPI2 |

### 35.7.1 SPIxCON0

**Name:** SPIxCON0
**Address:** 0x084,0x091

SPI Control Register 0

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | EN | | | | | LSBF | MST | BMODE |
| Access | R/W | | | | | R/W | R/W | R/W |
| Reset | 0 | | | | | 0 | 0 | 0 |

**Bit 7 – EN** SPI Enable

| Value | Description |
|---|---|
| 1 | SPI is enabled |
| 0 | SPI is disabled |

**Bit 2 – LSBF** LSb-First Data Exchange Select(1)

| Value | Description |
|---|---|
| 1 | Data is exchanged LSb first |
| 0 | Data is exchanged MSb first (traditional SPI operation) |

**Bit 1 – MST** SPI Host Operating Mode Select(1)

| Value | Description |
|---|---|
| 1 | SPI module operates as the bus host |
| 0 | SPI module operates as a bus client |

**Bit 0 – BMODE** Bit-Length Mode Select(1)

| Value | Description |
|---|---|
| 1 | SPIxTWIDTH setting applies to every byte: total bits sent is SPIxTWIDTH*SPIxTCNT, end-of-packet occurs when SPIxTCNT = 0 |
| 0 | SPIxTWIDTH setting applies only to the last byte exchanged; total bits sent is SPIxTWIDTH + (SPIxTCNT*8) |

**Note:**

1. Do not change this bit when EN = 1.

### 35.7.2 SPIxCON1

**Name:** SPIxCON1
**Address:** 0x085,0x092

SPI Control Register 1

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | SMP | CKE | CKP | FST | | SSP | SDIP | SDOP |
| Access | R/W | R/W | R/W | R/W | | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | | 1 | 0 | 0 |

**Bit 7 – SMP** SPI Input Sample Phase Control

| Value | Mode | Description |
|---|---|---|
| 1 | Client | Reserved |
| 1 | Host | SDI input is sampled at the end of data output time |
| 0 | Client or Host | SDI input is sampled in the middle of data output time |

**Bit 6 – CKE** Clock Edge Select

| Value | Description |
|---|---|
| 1 | Output data changes on transition from Active to Idle clock state |
| 0 | Output data changes on transition from Idle to Active clock state |

**Bit 5 – CKP** Clock Polarity Select

| Value | Description |
|---|---|
| 1 | Idle state for SCK is high level |
| 0 | Idle state for SCK is low level |

**Bit 4 – FST** Fast Start Enable

| Value | Mode | Description |
|---|---|---|
| x | Client | This bit is ignored |
| 1 | Host | Delay to first SCK may be less than ½ baud period |
| 0 | Host | Delay to first SCK will be at least ½ baud period |

**Bit 2 – SSP** Client Select Input/Output Polarity Control

| Value | Description |
|---|---|
| 1 | SS is active-low |
| 0 | SS is active-high |

**Bit 1 – SDIP** SPI Input Polarity Control

| Value | Description |
|---|---|
| 1 | SDI input is active-low |
| 0 | SDI input is active-high |

**Bit 0 – SDOP** SPI Output Polarity Control

| Value | Description |
|---|---|
| 1 | SDO output is active-low |
| 0 | SDO output is active-high |

© 2019-2023 Microchip Technology Inc. and its subsidiaries

### 35.7.3 SPIxCON2

**Name:** SPIxCON2
**Address:** 0x086,0x093

SPI Control Register 2[3]

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | BUSY | SSFLT | | | | SSET | TXR | RXR |
| Access | R | R | | | | R/W | R/W | R/W |
| Reset | 0 | 0 | | | | 0 | 0 | 0 |

**Bit 7 – BUSY** SPI Module Busy Status[1]

| Value | Description |
|---|---|
| 1 | Data exchange is busy |
| 0 | Data exchange is not taking place |

**Bit 6 – SSFLT** SS_in Fault Status

| Value | Condition | Description |
|---|---|---|
| x | SSET = 1 | This bit is unchanged |
| 1 | SSET = 0 | SS_in ended the transaction unexpectedly, and the data byte being received was lost |
| 0 | SSET = 0 | SS_in ended normally |

**Bit 2 – SSET** Client Select Enable

| Value | Mode | Description |
|---|---|---|
| 1 | Host | SS_out is driven to the Active state continuously |
| 0 | Host | SS_out is driven to the Active state while the transmit counter is not zero |
| 1 | Client | SS_in is ignored and data is clocked on all SCK_in (as though SS = TRUE at all times) |
| 0 | Client | SS_in enables/disables data input and tri-states SDO if the TRIS bit associated with the SDO pin is set (see the Client Mode Transmit table for details) |

**Bit 1 – TXR** Transmit Data-Required Control[2]

| Value | Description |
|---|---|
| 1 | TxFIFO data is required for a transfer |
| 0 | TxFIFO data is not required for a transfer |

**Bit 0 – RXR** Receive FIFO Space-Required Control[2]

| Value | Description |
|---|---|
| 1 | Data transfers are suspended when RxFIFO is full |
| 0 | Received data is not stored in the FIFO |

**Notes:**

1. The BUSY bit is subject to synchronization delay of up to two instruction cycles. The user must wait after loading the transmit buffer (the SPIxTXB register) before using it to determine the status of the SPI module.
2. See the Host Mode TXR/RXR Settings table as well as the Host Mode and Client Mode sections for more details pertaining to TXR and RXR function.
3. This register will not be written to while a transfer is in progress (the BUSY bit is set).

### 35.7.4 SPIxCLK

**Name:** SPIxCLK
**Address:** 0x08C,0x099

SPI Clock Selection Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | CLKSEL[4:0] | | | | |
| Access | | | | R/W | R/W | R/W | R/W | R/W |
| Reset | | | | 0 | 0 | 0 | 0 | 0 |

**Bits 4:0 – CLKSEL[4:0]** SPI Clock Source Selection

**Table 35-5. SPI CLK Source Selections**

| CLK | Selection |
|---|---|
| 10111-11111 | Reserved |
| 10110 | CLC8_OUT |
| 10101 | CLC7_OUT |
| 10100 | CLC6_OUT |
| 10011 | CLC5_OUT |
| 10010 | CLC4_OUT |
| 10001 | CLC3_OUT |
| 10000 | CLC2_OUT |
| 01111 | CLC1_OUT |
| 01110 | SMT1_OUT |
| 01001-01101 | Reserved |
| 01000 | TMR6_Postscaler_OUT |
| 00111 | TMR4_Postscaler_OUT |
| 00110 | TMR2_Postscaler_OUT |
| 00101 | TMR0_OUT |
| 00100 | Clock Reference Output |
| 00011 | EXTOSC |
| 00010 | MFINTOSC (500 kHz) |
| 00001 | HFINTOSC |
| 00000 | $F_{OSC}$ (System Clock) |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

### 35.7.5 SPIxBAUD

**Name:** SPIxBAUD
**Address:** 0x089,0x096

SPI Baud Rate Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | BAUD[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 7:0 – BAUD[7:0]** Baud Clock Prescaler Select

| Value | Description |
|---|---|
| n | SCK high or low time: TSC = SPI Clock Period*(n+1)<br>SCK toggle frequency: FSCK = FBAUD = SPI Clock Frequency/(2*(n+1)) |

### 35.7.6 SPIxTCNT

**Name:** SPIxTCNT
**Address:** 0x082,0x08F

SPI Transfer Counter Register

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | TCNTH[2:0] | | |
| Access | | | | | | R/W | R/W | R/W |
| Reset | | | | | | 0 | 0 | 0 |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | TCNTL[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 10:8 – TCNTH[2:0]** SPI Transfer Counter Most Significant Byte

| Value | Condition | Description |
|---|---|---|
| n | BMODE = 0 | Bits 13-11 of the transfer bit count |
| n | BMODE = 1 | Bits 10-8 of the transfer byte count |

**Bits 7:0 – TCNTL[7:0]** SPI Transfer Counter Least Significant Byte

| Value | Condition | Description |
|---|---|---|
| n | BMODE = 0 | Bits 10-3 of the transfer bit count |
| n | BMODE = 1 | Bits 7-0 of the transfer byte count |

### 35.7.7 SPIxTWIDTH

**Name:** SPIxTWIDTH
**Address:** 0x088,0x095

SPI Transfer Width Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | TWIDTH[2:0] | | |
| Access | | | | | | R/W | R/W | R/W |
| Reset | | | | | | 0 | 0 | 0 |

**Bits 2:0 – TWIDTH[2:0]** SPI Transfer Count Byte Width or three LSbs of the Transfer Bit Count

| Value | Condition | Description |
|---|---|---|
| n | BMODE = 0 | Bits 2-0 of the transfer bit count |
| n | BMODE = 1 | Number of bits in each transfer byte count. Bits = n (when n > 0) or 8 (when n = 0). |

### 35.7.8 SPIxSTATUS

**Name:** SPIxSTATUS
**Address:** 0x087,0x094

SPI Status Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | TXWE | | TXBE | | RXRE | CLB | | RXBF |
| Access | R/C/HS | | R | | R/C/HS | S | | R |
| Reset | 0 | | 1 | | 0 | 0 | | 0 |

**Bit 7 – TXWE** Transmit Buffer Write Error

| Value | Description |
|---|---|
| 1 | SPIxTXB was written while TxFIFO was full |
| 0 | No error has occurred |

**Bit 5 – TXBE** Transmit Buffer Empty

| Value | Description |
|---|---|
| 1 | Transmit buffer TxFIFO is empty |
| 0 | Transmit buffer is not empty |

**Bit 3 – RXRE** Receive Buffer Read Error

| Value | Description |
|---|---|
| 1 | SPIxRXB was read while RxFIFO was empty |
| 0 | No error has occurred |

**Bit 2 – CLB** Clear Buffer Control

| Value | Description |
|---|---|
| 1 | Reset the receive and transmit buffers, making both buffers empty |
| 0 | Take no action |

**Bit 0 – RXBF** Receive Buffer Full

| Value | Description |
|---|---|
| 1 | Receive buffer is full |
| 0 | Receive buffer is not full |

### 35.7.9 SPIxRXB

**Name:** SPIxRXB
**Address:** 0x080,0x08D

SPI Receive Buffer

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | RXB[7:0] | | | | | | | |
| Access | R | R | R | R | R | R | R | R |
| Reset | x | x | x | x | x | x | x | x |

**Bits 7:0 – RXB[7:0]** Receive Buffer

| Value | Condition | Description |
|---|---|---|
| n | Receive buffer is not empty | Contains the top-most byte of the RXFIFO. Reading this register will remove the RXFIFO top-most byte and decrease the occupancy of the RXFIFO by 1. |
| 0 | Receive buffer is empty | Reading this register will return '0', leave the occupancy unchanged, and set the RXRE Status bit |

### 35.7.10 SPIxTXB

**Name:** SPIxTXB
**Address:** 0x081,0x08E

SPI Transmit Buffer

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | TXB[7:0] | | | | | | | |
| Access | W | W | W | W | W | W | W | W |
| Reset | x | x | x | x | x | x | x | x |

**Bits 7:0 – TXB[7:0]** Transmit Buffer

| Value | Condition | Description |
|---|---|---|
| n | Transmit buffer is not full | Writing to this register adds the data to the top of the TXFIFO and increases the occupancy of the TXFIFO by 1. |
| x | Transmit buffer is full | Writing to this register does not affect the data in the TXFIFO or the occupancy count. The TXWE Status bit will be set. |

### 35.7.11 SPIxINTE

**Name:** SPIxINTE
**Address:** 0x08B,0x098

SPI Interrupt Enable Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | SRMTIE | TCZIE | SOSIE | EOSIE | | RXOIE | TXUIE | |
| Access | R/W | R/W | R/W | R/W | | R/W | R/W | |
| Reset | 0 | 0 | 0 | 0 | | 0 | 0 | |

**Bit 7 – SRMTIE** Shift Register Empty Interrupt Enable

| Value | Description |
|---|---|
| 1 | Interrupt is enabled |
| 0 | Interrupt is not enabled |

**Bit 6 – TCZIE** Transfer Counter is Zero Interrupt Enable

| Value | Description |
|---|---|
| 1 | Interrupt is enabled |
| 0 | Interrupt is not enabled |

**Bit 5 – SOSIE** Start of Client Select Interrupt Enable

| Value | Description |
|---|---|
| 1 | Interrupt is enabled |
| 0 | Interrupt is not enabled |

**Bit 4 – EOSIE** End of Client Select Interrupt Enable

| Value | Description |
|---|---|
| 1 | Interrupt is enabled |
| 0 | Interrupt is not enabled |

**Bit 2 – RXOIE** Receiver Overflow Interrupt Enable

| Value | Description |
|---|---|
| 1 | Interrupt is enabled |
| 0 | Interrupt is not enabled |

**Bit 1 – TXUIE** Transmitter Underflow Interrupt Enable

| Value | Description |
|---|---|
| 1 | Interrupt is enabled |
| 0 | Interrupt is not enabled |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

### 35.7.12 SPIxINTF

**Name:** SPIxINTF
**Address:** 0x08A,0x097

SPI Interrupt Flag Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | SRMTIF | TCZIF | SOSIF | EOSIF | | RXOIF | TXUIF | |
| Access | R/W/HS | R/W/HS | R/W/HS | R/W/HS | | R/W/HS | R/W/HS | |
| Reset | 0 | 0 | 0 | 0 | | 0 | 0 | |

**Bit 7 – SRMTIF** Shift Register Empty Interrupt Flag

| Value | Mode | Description |
|---|---|---|
| x | Client | This bit is ignored |
| 1 | Host | The data transfer is complete |
| 0 | Host | Either no data transfers have occurred or a data transfer is in progress |

**Bit 6 – TCZIF** Transfer Counter is Zero Interrupt Flag

| Value | Description |
|---|---|
| 1 | The transfer counter has decremented to zero |
| 0 | No interrupt pending |

**Bit 5 – SOSIF** Start of Client Select Interrupt Flag

| Value | Description |
|---|---|
| 1 | SS_in transitioned from false to true |
| 0 | No interrupt pending |

**Bit 4 – EOSIF** End of Client Select Interrupt Flag

| Value | Description |
|---|---|
| 1 | SS_in transitioned from true to false |
| 0 | No interrupt pending |

**Bit 2 – RXOIF** Receiver Overflow Interrupt Flag

| Value | Description |
|---|---|
| 1 | Data transfer completed when RXBF = 1 (edge-triggered) and RXR = 1 |
| 0 | No interrupt pending |

**Bit 1 – TXUIF** Transmitter Underflow Interrupt Flag

| Value | Description |
|---|---|
| 1 | Client Data transfer started when TXBE = 1 and TXR = 1 |
| 0 | No interrupt pending |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

## 35.8 Register Summary - SPI Control

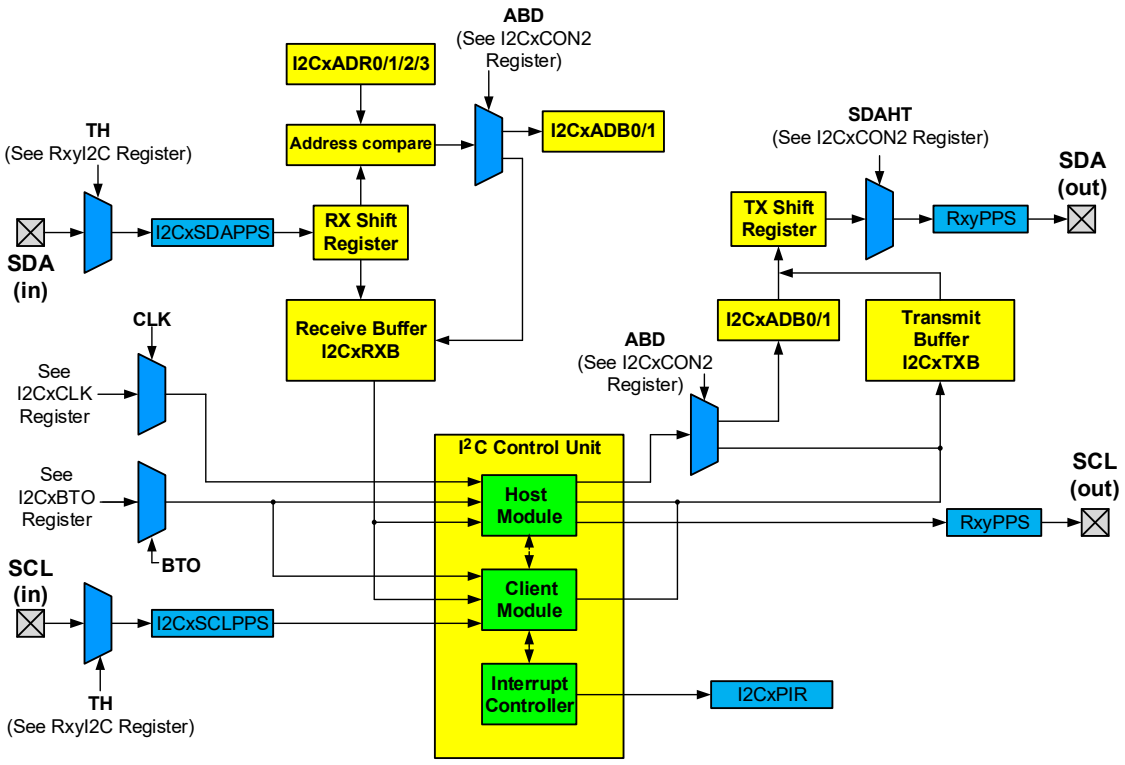
| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x80 | SPI1RXB | 7:0 | RXB[7:0] | | | | | | | |
| 0x81 | SPI1TXB | 7:0 | TXB[7:0] | | | | | | | |
| 0x82 | SPI1TCNT | 7:0 | TCNTL[7:0] | | | | | | | |
| | | 15:8 | | | | | | TCNTH[2:0] | | |
| 0x84 | SPI1CON0 | 7:0 | EN | | | | | LSBF | MST | BMODE |
| 0x85 | SPI1CON1 | 7:0 | SMP | CKE | CKP | FST | | SSP | SDIP | SDOP |
| 0x86 | SPI1CON2 | 7:0 | BUSY | SSFLT | | | | SSET | TXR | RXR |
| 0x87 | SPI1STATUS | 7:0 | TXWE | | TXBE | | RXRE | CLB | | RXBF |
| 0x88 | SPI1TWIDTH | 7:0 | | | | | | TWIDTH[2:0] | | |
| 0x89 | SPI1BAUD | 7:0 | BAUD[7:0] | | | | | | | |
| 0x8A | SPI1INTF | 7:0 | SRMTIF | TCZIF | SOSIF | EOSIF | | RXOIF | TXUIF | |
| 0x8B | SPI1INTE | 7:0 | SRMTIE | TCZIE | SOSIE | EOSIE | | RXOIE | TXUIE | |
| 0x8C | SPI1CLK | 7:0 | | | | CLKSEL[4:0] | | | | |
| 0x8D | SPI2RXB | 7:0 | RXB[7:0] | | | | | | | |
| 0x8E | SPI2TXB | 7:0 | TXB[7:0] | | | | | | | |
| 0x8F | SPI2TCNT | 7:0 | TCNTL[7:0] | | | | | | | |
| | | 15:8 | | | | | | TCNTH[2:0] | | |
| 0x91 | SPI2CON0 | 7:0 | EN | | | | | LSBF | MST | BMODE |
| 0x92 | SPI2CON1 | 7:0 | SMP | CKE | CKP | FST | | SSP | SDIP | SDOP |
| 0x93 | SPI2CON2 | 7:0 | BUSY | SSFLT | | | | SSET | TXR | RXR |
| 0x94 | SPI2STATUS | 7:0 | TXWE | | TXBE | | RXRE | CLB | | RXBF |
| 0x95 | SPI2TWIDTH | 7:0 | | | | | | TWIDTH[2:0] | | |
| 0x96 | SPI2BAUD | 7:0 | BAUD[7:0] | | | | | | | |
| 0x97 | SPI2INTF | 7:0 | SRMTIF | TCZIF | SOSIF | EOSIF | | RXOIF | TXUIF | |
| 0x98 | SPI2INTE | 7:0 | SRMTIE | TCZIE | SOSIE | EOSIE | | RXOIE | TXUIE | |
| 0x99 | SPI2CLK | 7:0 | | | | CLKSEL[4:0] | | | | |

# 36. I²C - Inter-Integrated Circuit Module

The Inter-Integrated Circuit (I²C) bus is a multi-host serial data communication bus. Devices communicate in a host/client environment where the host devices initiate the communication. A client device is controlled through addressing.

The following figure shows a block diagram of the I²C interface module, and shows both Host and Client modes together.

**Figure 36-1. I²C Block Diagram**




## 36.1 I²C Features

The I²C supports the following modes and features:

- Modes
  - Host mode
  - Client mode
  - Multi-Host mode
- Features
  - Supports Standard-mode (100 kHz), Fast-mode (400 kHz) and Fast-mode plus (1 MHz) modes of operation
  - Dedicated Address, Receive, and Transmit buffers
  - Up to four unique Client addresses
  - General Call addressing
  - 7-bit and 10-bit addressing with optional masking
  - Interrupts for:

- Start condition
- Restart condition
- Stop condition
- Address match
- Data Write
- Acknowledge Status
- NACK detection
- Data Byte Count
- Bus Collision
- Bus Time-out

– Clock Stretching for:
  - RX buffer full
  - TX buffer empty
  - Incoming address match
  - Data Write
  - Acknowledge Status
– Bus Collision Detection with Arbitration
– Bus Time-out Detection
– Selectable Serial Data (SDA) Hold Time
– Dedicated $I^2C$ Pad (I/O) Control
  - Standard GPIO or $I^2C$-specific slew rate control
  - Selectable $I^2C$ pull-up levels
  - $I^2C$-specific, SMBus 2.0/3.0, or standard GPIO input threshold level selections
– Integrated Direct Memory Access (DMA) support
– Remappable pin locations using Peripheral Pin Select (PPS)

## 36.2 $I^2C$ Terminology

The $I^2C$ communication protocol terminology used throughout this document have been adapted from the Phillips $I^2C$ Specification and can be found in the table below.

**$I^2C$ Bus Terminology and Definitions**

| Term | Definition |
|---|---|
| Host | The device that initiates a transfer, generates the clock signal and terminates a transfer |
| Client | The device addressed by the host |
| Multi-Host | A bus containing more than one host device that can initiate communication |
| Transmitter | The device that shifts data out onto the bus |
| Receiver | The device that shifts data in from the bus |
| Arbitration | Procedure that ensures only one host at a time controls the bus |
| Synchronization | Procedure that synchronizes the clock signal between two or more devices on the bus |
| Idle | The state in which no activity occurs on the bus and both bus lines are at a high logic level |
| Active | The state in which one or more devices are communicating on the bus |
| Matching Address | The address byte received by a client that matches the value that is stored in the I2CxADR0/1/2/3 registers |
| Addressed Client | Client device that has received a matching address and is actively being clocked by a host device |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

| | |
|---|---|
| Write Request | Host transmits an address with the R/$\overline{W}$ bit clear indicating that it wishes to transmit data to a client device |
| Read Request | Host transmits an address with the R/$\overline{W}$ bit set indicating that it wishes to receive data from a client device |
| Clock Stretching | The action in which a device holds the SCL line low to stall communication |
| Bus Collision | Occurs when the module samples the SDA line and returns a low state while expecting a high state |
| Bus Time-out | Occurs whenever communication stalls for a period longer than acceptable |

## 36.3 I²C Module Overview

The I²C module provides a synchronous serial interface between the microcontroller and other I²C-compatible devices using a bidirectional two-wire bus. Devices operate in a host/client environment that may contain one or more host devices and one or more client devices. The host device always initiates communication.

The I²C bus consists of two signal connections:

- Serial Clock (SCL)
- Serial Data (SDA)

Both the SCL and SDA connections are open-drain lines, each line requiring pull-up resistors to the application's supply voltage. Pulling the line to ground is considered a logic '0', while allowing the line to float is considered a logic '1'. It is important to note that the voltage levels of the logic low and logic high are not fixed and are dependent on the bus supply voltage. According to the I²C Specification, a logic low input level is up to 30% of $V_{DD}$ ($V_{IL} \leq 0.3V_{DD}$), while the logic high input level is 70% to 100% of $V_{DD}$ ($V_{IH} \geq 0.7V_{DD}$). Both signal connections are considered bidirectional, although the SCL signal can only be an output in Host mode and an input in Client mode.

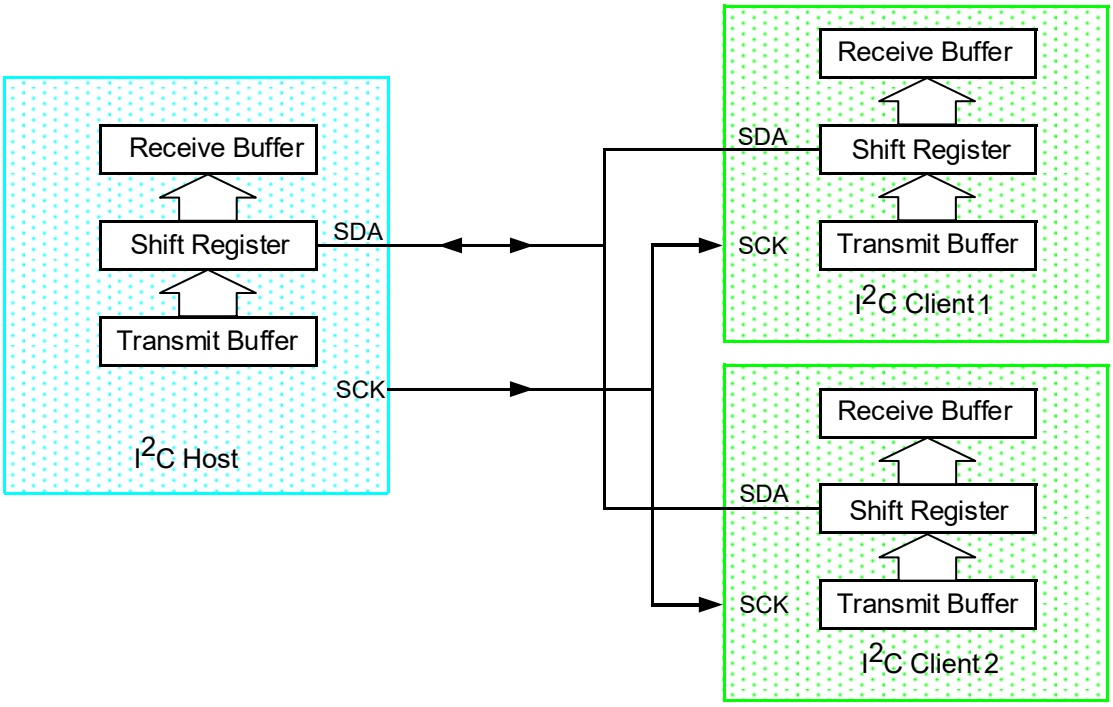
All transactions on the bus are initiated and terminated by the host device. Depending on the direction of the data being transferred, there are four main operations performed by the I²C module:

- Host Transmit: host is transmitting data to a client
- Host Receive: host is receiving data from a client
- Client Transmit: client is transmitting data to a host
- Client Receive: client is receiving data from a host

The I²C interface allows for a multi-host bus, meaning that there can be several host devices present on the bus. A host can select a client device by transmitting a unique address on the bus. When the address matches a client's address, the client responds with an Acknowledge condition ($\overline{ACK}$), and communication between the host and that client can commence. All other devices connected to the bus must ignore any transactions not intended for them.

The following figure shows a typical I²C bus configuration with one host and two clients.

**Figure 36-2. I²C Host-Client Connections**




### 36.3.1 Byte Format

As previously mentioned, all I²C communication is performed in 9-bit segments. The transmitting device sends a byte to a receiver, and once the byte is processed by the receiver, the receiver returns an Acknowledge bit. There are no limits to the amount of data bytes in a I²C transmission.

After the 8th falling edge of the SCL line, the transmitting device releases control of the SDA line to allow the receiver to respond with either an Acknowledge ($\overline{ACK}$) sequence or a Not Acknowledge (NACK) sequence. At this point, if the receiving device is a client, it can hold the SCL line low (clock stretch) to allow itself time to process the incoming byte. Once the byte has been processed, the receiving device releases the SCL line, allowing the host device to provide the 9th clock pulse, within which the client responds with either an $\overline{ACK}$ or a NACK sequence. If the receiving device is a host, it may also hold the SCL line low until it has processed the received byte. Once the byte has been processed, the host device will generate the 9th clock pulse and transmit the $\overline{ACK}$ or NACK sequence.

Data is valid to change only while the SCL signal is in a low state, and sampled on the rising edge of SCL. Changes on the SDA line while the SCL line is high indicate either a Start or Stop condition.

### 36.3.2 SDA and SCL Pins

The SDA and SCL pins must be configured as open-drain outputs. Open-drain configuration is accomplished by setting the appropriate bits in the Open-Drain Control (ODCONx) registers, while output direction configuration is handled by clearing the appropriate bits in the Tri-State Control (TRISx) registers. Input threshold, slew rate, and internal pull-up settings are configured using the RxyI2C registers. The RxyI2C registers are used exclusively on the default I²C pin locations, and provide the following selections:

- Input threshold levels:
  - SMBus 3.0 (1.35V) input threshold
  - SMBus 2.0 (2.1V) input threshold
  - I²C-specific input thresholds

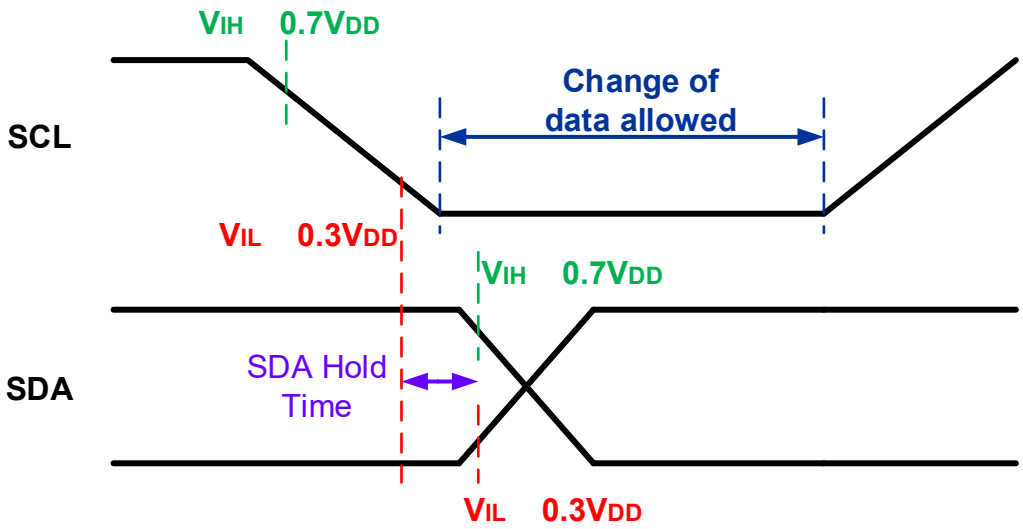
  - Standard GPIO input thresholds (controlled by the Input Level Control (INLVLx) registers)
- Slew rate limiting:
  - $I^2C$-specific slew rate limiting
  - Standard GPIO slew rate (controlled by the Slew Rate Control (SLRCONx) registers)
- $I^2C$ pull-ups:
  - Programmable ten or two times the current of the standard internal pull-up
  - Standard GPIO pull-up (controlled by the Weak Pull-Up Control (WPUx) registers)

**Important:** The pin locations for SDA and SCL are remappable through the Peripheral Pin Select (PPS) registers. If new pin locations for SDA and SCL are desired, user software must configure the INLVLx, SLRCONx, ODCONx, and TRISx registers for each new pin location. The RxyI2C registers cannot be used since they are dedicated to the default pin locations. Additionally, the internal pull-ups for non-$I^2C$ pins are not strong enough to drive the pins; therefore, external pull-up resistors must be used.

### 36.3.2.1 SDA Hold Time

SDA hold time refers to the amount of time between the low threshold region of the falling edge of SCL ($V_{IL} \le 0.3V_{DD}$) and either the low threshold region of the rising edge of SDA ($V_{IL} \le 0.3V_{DD}$) or the high threshold region of the falling edge of SDA ($V_{IH} \ge 0.7V_{DD}$) (see figure below). If the SCL fall time is long or close to the maximum allowable time set by the $I^2C$ Specification, data may be sampled in the undefined logic state between the 70% and 30% region of the falling SCL edge, leading to data corruption. The $I^2C$ module offers selectable SDA hold times, which can be useful to ensure valid data transfers at various bus data rates and capacitance loads.

**Figure 36-3. SDA Hold Time**




### 36.3.3 Start Condition

All $I^2C$ transmissions begin with a Start condition. The Start condition is used to synchronize the SCL signals between the host and client devices. The $I^2C$ Specification defines a Start condition as a transition of the SDA line from a logic high level (Idle state) to a logic low level (Active state) while the SCL line is at a logic high (see figure below). A Start condition is always generated by the host, and is initiated by either writing to the Start (S) bit or by writing to the $I^2C$ Transmit Buffer (I2CxTXB) register, depending on the Address Buffer Disable (ABD) bit setting.

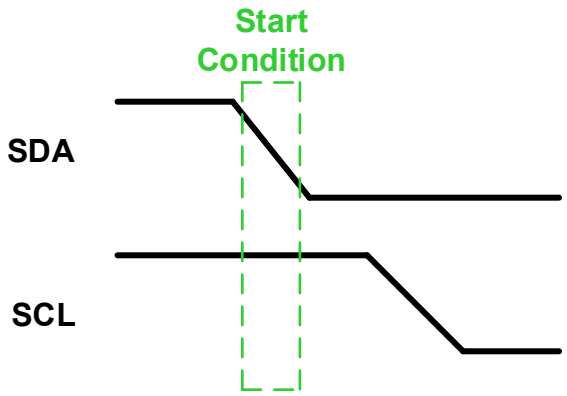
When the $I^2C$ module is configured in Host mode, module hardware waits until the bus is free (Idle state). Module hardware checks the Bus Free Status (BFRE) bit to ensure the bus is idle before initiating a Start condition. When the BFRE bit is set, the bus is considered idle, and indicates that the SCL and SDA lines have been in a logic high state

for the amount of $I^2C$ clock cycles as selected by the Bus Free Time Selection (BFRET) bits. When a Start condition is detected on the bus, module hardware clears the BFRE bit, indicating an active bus.

In Multi-Host mode, it is possible for two host devices to issue Start conditions at the same time. If two or more hosts initiate a Start at the same time, a bus collision will occur; however, the $I^2C$ Specification states that a bus collision cannot occur on a Start. In this case, the competing host devices must go through bus arbitration during the addressing phase.

The figure below shows a Start condition.

**Figure 36-4. Start Condition**




### 36.3.4 Acknowledge Sequence

The 9th SCL pulse for any transferred address/data byte is reserved for the Acknowledge ($\overline{ACK}$) sequence. During an Acknowledge sequence, the transmitting device relinquishes control of the SDA line to the receiving device. At this time, the receiving device must decide whether to pull the SDA line low ($\overline{ACK}$) or allow the line to float high (NACK). Since the Acknowledge sequence is an active-low signal, pulling the SDA line low informs the transmitter that the receiver has successfully received the transmitted data.

The Acknowledge Data (ACKDT) bit holds the value to be transmitted during an Acknowledge sequence while the I2CxCNT register is non-zero (I2CxCNT != 0). When a client device receives a matching address, or a receiver receives valid data, the ACKDT bit is cleared by user software to indicate an $\overline{ACK}$. If the client does not receive a matching address, user software sets the ACKDT bit, indicating a NACK. In Client or Multi-Host modes, if the Address Interrupt and Hold Enable (ADRIE) or Write Interrupt and Hold Enable (WRIE) bits are set, the clock is stretched after receiving a matching address or after the 8th falling edge of SCL when a data byte is received. This allows user software time to determine the $\overline{ACK}$/NACK response to send back to the transmitter.

The Acknowledge End of Count (ACKCNT) bit holds the value that will be transmitted once the I2CxCNT register reaches a zero value (I2CxCNT = 0). When the I2CxCNT register reaches a zero value, the ACKCNT bit can be cleared (ACKCNT = 0), indicating an $\overline{ACK}$, or ACKCNT can be set (ACKCNT = 1), indicating a NACK. It is important to note that the ACKCNT bit is only used when the I2CxCNT register is zero, otherwise the ACKDT bit is used for $\overline{ACK}$/NACK sequences.

In Host Write or Client Read modes, the Acknowledge Status (ACKSTAT) bit holds the result of the Acknowledge sequence transmitted by the receiving device. The ACKSTAT bit is cleared when the receiver sends an $\overline{ACK}$, and is set when the receiver does not Acknowledge (NACK).

The Acknowledge Time Status (ACKT) bit indicates whether or not the bus is in an Acknowledge sequence. The ACKT bit is set during an $\overline{ACK}$/NACK sequence on the 8th falling edge of SCL, and is cleared on the 9th rising edge of SCL, indicating that the bus is not in an $\overline{ACK}$/NACK sequence.

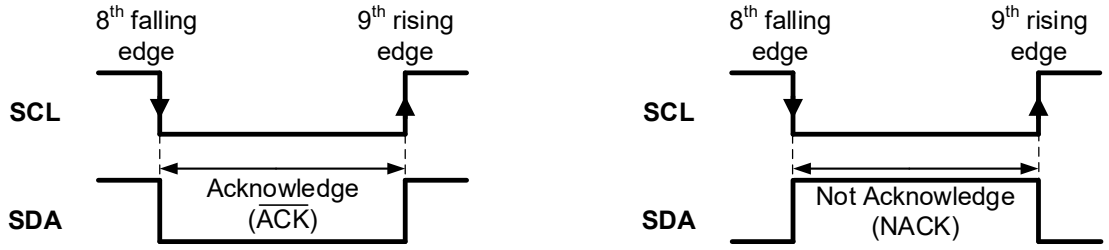
Certain conditions will cause a NACK sequence to be sent automatically. A NACK sequence is generated by module hardware when any of the following bits are set:

- Transmit Write Error Status (TXWE)
- Transmit Underflow Status (TXU)
- Receive Read Error Status (RXRE)
- Receive Overflow Status (RXO)

**Important:** Once a NACK is detected on the bus, all subsequent Acknowledge sequences will consist of a NACK until all error conditions are cleared.

The following figure shows $\overline{ACK}$ and NACK sequences.

**Figure 36-5. $\overline{ACK}$/NACK Sequences**




### 36.3.5 Restart Condition

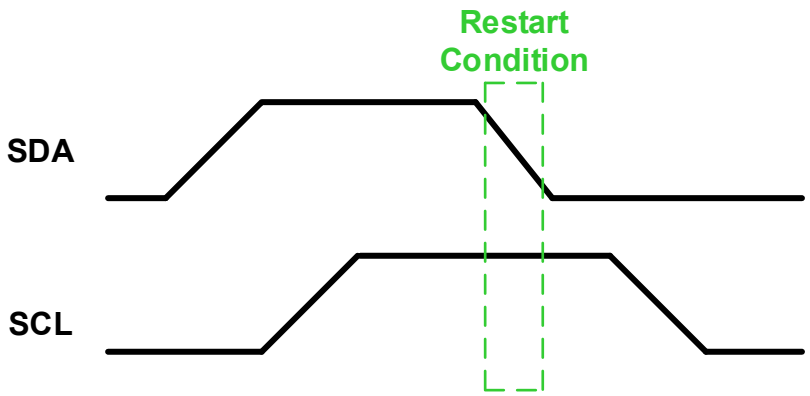
A Restart condition is essentially the same as a Start condition – the SDA line transitions from an idle level to an active level while the SCL line is Idle – but may be used in place of a Stop condition whenever the host device has completed its current transfer but wishes to keep control of the bus. A Restart condition has the same effect as a Start condition, resetting all client logic and preparing it to receive an address.

A Restart condition is also used when the host wishes to use a combined data transfer format. A combined data transfer format is used when a host wishes to communicate with a specific register address or memory location. In a combined format, the host issues a Start condition, followed by the client's address, followed by a data byte which represents the desired client register or memory address. Once the client address and data byte have been acknowledged by the client, the host issues a Restart condition, followed by the client address. If the host wishes to write data to the client, the LSb of the client address, the Read/not Write (R/$\overline{W}$) bit, will be clear. If the host wishes to read data from the client, the R/$\overline{W}$ bit will be set. Once the client has acknowledged the second address byte, the host issues a Restart condition, followed by the upper byte of the client address with the R/$\overline{W}$ bit set. Client logic will then acknowledge the upper byte, and begin to transmit data to the host.

**Important:** In 10-bit Client mode, a Restart is required for the host to read data out of the client, regardless of which data transfer format is used – host read-only or combined. For example, if the host wishes to perform a bulk read, it will transmit the client's 10-bit address with the R/$\overline{W}$ bit clear.

The figure below shows a Restart condition.

**Figure 36-6. Restart Condition**

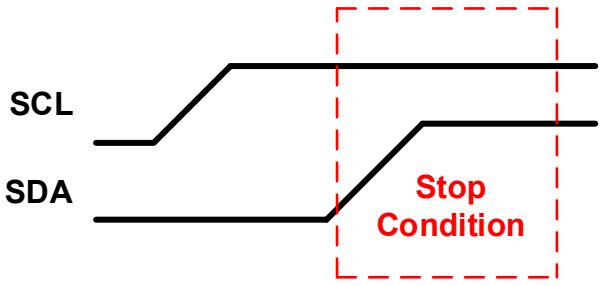



### 36.3.6 Stop Condition

All I²C transmissions end with a Stop condition. A Stop condition occurs when the SDA line transitions from a logic low (Active) level to a logic high (Idle) level while the SCL line is at a logic high level. A Stop condition is always generated by the host device, and is generated by module hardware when a Not Acknowledge (NACK) is detected on the bus, a bus time-out event occurs, or when the I²C Byte Count (I2CxCNT) register reaches a zero count.

The figure below shows a Stop condition.

**Figure 36-7. Stop Condition**




### 36.3.7 Bus Time-out

The SMBus protocol requires a bus watchdog to prevent a stalled device from holding the bus indefinitely. The I²C Bus Time-Out (I2CxBTO) register provides several timer sources that can be configured as the time-out time base. When the selected resource expires, the module is reset and the bus is forced into an Idle state.

For example, Timer2 can be selected as the bus time-out source and set to begin counting when the SCL line is low. If Timer2 rolls over before the SCL line transitions high, the module is reset.

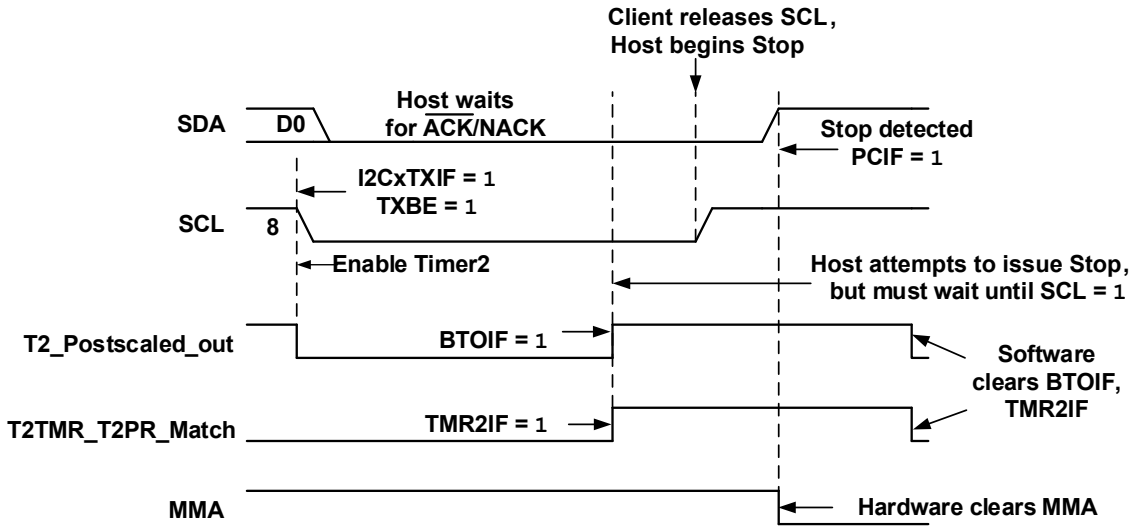
If the module is configured in Client mode, and a bus time-out event occurs while the Client is active (Client mode Active (SMA) bit = 1), the module is immediately reset, the SMA and Client Clock Stretching (CSTR) bits are cleared, and the Bus Time-out Interrupt Flag (BTOIF) bit is set.

If the module is configured in Host mode, and the bus time-out event occurs while the host is active (Host mode Active (MMA) = 1), the module will immediately attempt to transmit a Stop condition, and sets the BTOIF bit. Stop condition generation may be delayed if a client device is stretching the clock, but will resume once the clock is

released, or if the client holding the bus also times out. The MMA bit is only cleared after the Stop condition has been generated.

The figure below shows an example of a bus time-out event when the module is operating in Host mode.

**Figure 36-8. Host Mode Bus Time-out Example**




### 36.3.8 Address Buffers

The I$^2$C module has two address buffer registers, I2CxADB0 and I2CxADB1, which can be used as address receive buffers in Client mode, address transmit buffers in Host mode, or both address transmit and address receive buffers in 7-bit Multi-Host mode (see table below). The address buffers are enabled/disabled via the Address Buffer Disable (ABD) bit.

When the ABD bit is clear (ABD = 0), the buffers are enabled, which means:

- In 7-bit Host mode, the desired client address with the R/$\overline{W}$ value is transmitted from the I2CxADB1 register, bypassing the I$^2$C Transmit Buffer (I2CxTXB). I2CxADB0 is unused.
- In 10-bit Host mode, I2CxADB1 holds the upper bits and R/$\overline{W}$ value of the desired client address, while I2CxADB0 holds the lower eight bits of the desired client address. Host hardware copies the contents of I2CxADB1 to the transmit shift register, and waits for an $\overline{ACK}$ from the client. Once the $\overline{ACK}$ is received, host hardware copies the contents of I2CxADB0 to the transmit shift register.
- In 7-bit Client mode, a matching received address is loaded into I2CxADB0, bypassing the I$^2$C Receive Buffer (I2CxRXB). I2CxADB1 is unused.
- In 10-bit Client mode, I2CxADB0 is loaded with the lower eight bits of the matching received address, while I2CxADB1 is loaded with the upper bits and R/$\overline{W}$ value of the matching received address.
- In 7-bit Multi-Host mode, the device can be both a host and a client depending on the sequence of events on the bus. When being addressed as a client, the matching received address with R/$\overline{W}$ value is stored into I2CxADB0. When being used as a host, the desired client address and R/$\overline{W}$ value are loaded into the I2CxADB1 register.

When the ABD bit is set (ABD = 1), the buffers are disabled, which means:

- In Host mode, the desired client address is transmitted from the I2CxTXB register.
- In Client mode, a matching received address is loaded into the I2CxRXB register.

**Table 36-1. Address Buffer Direction**

| Mode | I2CxADB0 | I2CxADB1 |
|---|---|---|

| | | |
|---|---|---|
| Client (7-bit) | RX | Unused |
| Client (10-bit) | RX (address low byte) | RX (address high byte) |
| Host (7-bit) | Unused | TX |
| Host (10-bit) | TX (address low byte) | TX (address high byte) |
| Multi-Host (7-bit) | RX | TX |

### 36.3.9 Transmit Buffer

The I²C module has a dedicated transmit buffer, I2CxTXB, which is independent from the receive buffer.

The transmit buffer is loaded with an address byte (when ABD = 1), or a data byte, that is copied into the transmit shift register and transmitted onto the bus. When the I2CxTXB register does not contain any transmit data, the Transmit Buffer Empty Status (TXBE) bit is set (TXBE = 1), allowing user software or the DMA to load a new byte into the buffer. When the TXBE bit is set and the I2CxCNT register is non-zero I2CxCNT( != 0), the I²C Transmit Interrupt Flag (I2CxTXIF) bit of the PIR registers is set, and can be used as a DMA trigger. A write to I2CxTXB will clear both the I2CxTXB and I2CxTXIF bits. Setting the Clear Buffer (CLRBF) bit clears I2CxTXIF, the I2Cx Receive Buffer (I2CxRXB) and I2CxTXB.

If user software attempts to load I2CxTXB while it is full, the Transmit Write Error Status (TXWE) bit is set, a NACK is generated, and the new data is ignored. If TXWE is set, user software must clear the bit before attempting to load the buffer again.

When module hardware attempts to transfer the contents of I2CxTXB to the transmit shift register while I2CxTXB is empty (TXBE = 1), the Transmit Underflow Status (TXU) bit is set, I2CxTXB is loaded with 0xFF, and a NACK is generated.

**Important:** A transmit underflow can only occur when clock stretching is disabled (Clock Stretching Disable (CSD) bit = 1). Clock stretching prevents transmit underflows because the clock is stretched after the 8th falling SCL edge, and is only released upon the write of new data into I2CxTXB.

### 36.3.10 Receive Buffer

The I²C module has a dedicated receive buffer, I2CxRXB, which is independent from the transmit buffer.

Data received through the shift register is transferred to I2CxRXB when the byte is complete. User software or the DMA can access the byte by reading the I2CxRXB register. When new data is loaded into I2CxRXB, the Receive Buffer Full Status (RXBF) bit is set, allowing user software or the DMA to read the new data. When the RXBF bit is set, the I²C Receive Interrupt Flag (I2CxRXIF) bit of the PIR registers is set, and can be used to trigger the DMA. A read of the I2CxRXB register will clear both RXBF and I2CxRXIF bits. Setting the CLRBF bit clears the I2CxRXIF bit, I2CxRXB, and I2CxTXB.

If the buffer is read while empty (RXBF = 0), the Receive Read Error Status (RXRE) bit is set, and the module generates a NACK. User software must clear RXRE to resume normal operation.

When the module attempts to transfer the contents of the receive shift register to I2CxRXB while I2CxRXB is full (RXBF = 1), the Receive Overflow Status (RXO) bit is set, and a NACK is generated. The data currently stored in I2CxRXB remains unchanged, but the data in the receive shift register is lost.

**Important:** A receive overflow can only occur when clock stretching is disabled. Clock stretching prevents receive overflows because the receive shift register cannot receive any more data until user software or the DMA reads I2CxRXB and the SCL line is released.

### 36.3.11 Clock Stretching

Clock stretching occurs when a client device holds the SCL line low to pause bus communication. A client device may stretch the clock to allow more time to process incoming data, prepare a response for the host device, or to

prevent receive overflow or transmit underflow conditions. Clock stretching is enabled by clearing the Clock Stretch Disable (CSD) bit, and is only available in Client and Multi-Host modes.

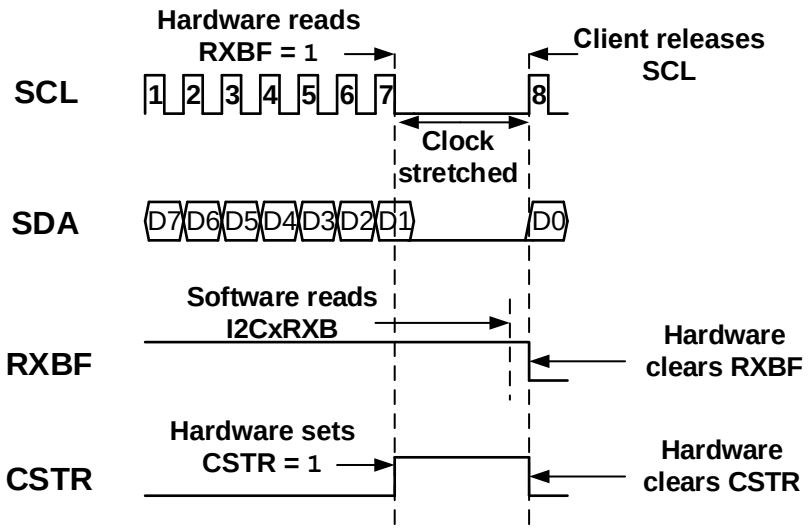
When clock stretching is enabled (CSD = 0), the Client Clock Stretching (CSTR) bit can be used to determine if the clock is currently being stretched. While the client is actively stretching the clock, CSTR is set by hardware (CSTR = 1). Once the client has completed its current transaction and clock stretching is no longer required, either module hardware or user software must clear CSTR to release the clock and resume communication.

#### 36.3.11.1 Clock Stretching for Buffer Operations

When enabled (CSD = 0), clock stretching is forced during buffer read/write operations. This allows the client device time to either load I2CxTXB with transmit data, or read data from I2CxRXB to clear the buffer.

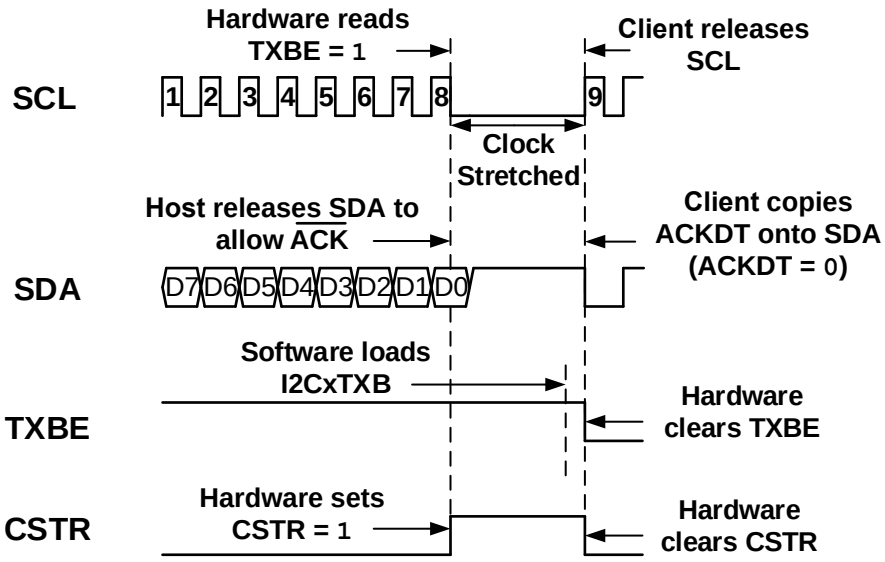
In Client Receive mode, clock stretching prevents receive data overflows. When the first seven bits of a new byte are received into the receive shift register while I2CxRXB is full (RXBF = 1), client hardware automatically stretches the clock and sets CSTR. When the client has read the data in I2CxRXB, client hardware automatically clears CSTR to release the SCL line and continue communication (see figure below).

**Figure 36-9. Receive Buffer Clock Stretching**




In Client Transmit mode, clock stretching prevents transmit underflows. When I2CxTXB is empty (TXBE = 1) and the I2CxCNT register is non-zero (I2CxCNT != 0), client hardware stretches the clock and sets CSTR upon the 8th falling SCL edge. Once the client has loaded new data into I2CxTXB, client hardware automatically clears CSTR to release the SCL line and allow further communication (see figure below).

**Figure 36-10. Transmit Buffer Clock Stretching**




### 36.3.11.2 Clock Stretching for Other Client Operations

The I²C module provides three Interrupt and Hold Enable features:

- Address Interrupt and Hold Enable
- Data Write Interrupt and Hold Enable
- Acknowledge Status Time Interrupt and Hold Enable

When clock stretching is enabled (CSD = 0), the Interrupt and Hold Enable features provide an interrupt response, and stretches the clock to allow time for address recognition, data processing, or an $\overline{ACK}$/NACK response.

The Address Interrupt and Hold Enable feature will generate an interrupt event and stretch the SCL line when a matching address is received. This feature is enabled by setting the Address Interrupt and Hold Enable (ADRIE) bit. When enabled ADRIE( = 1), the CSTR bit and the Address Interrupt Flag (ADRIF) bit are set by the module hardware, and the SCL line is stretched following the 8th falling SCL edge of a received matching address. Once the client has completed processing the address, software determines whether to send an $\overline{ACK}$ or a NACK back to the host device. Client software must clear both the ADRIF and CSTR bits to resume communication.

**Important:** In 10-bit Client Addressing mode, clock stretching occurs only after the client receives a matching low address byte, or a matching high address byte with the R/$\overline{W}$ bit = 1 (Host read) while the Client Mode Active (SMA) bit is set (SMA = 1). Clock stretching does not occur after the client receives a matching high address byte with the R/$\overline{W}$ bit = 0 (Host write).

The Data Write Interrupt and Hold Enable feature provides an interrupt event and stretches the SCL signal after the client receives a data byte. This feature is enabled by setting the Data Write Interrupt and Hold Enable (WRIE) bit. When enabled (WRIE = 1), module hardware sets both the CSTR bit and the Data Write Interrupt Flag (WRIF) bit and stretches the SCL line after the 8th falling edge of SCL. Once the client has read the new data, software determines whether to send an $\overline{ACK}$ or a NACK back to the host device. Client software must clear both the CSTR and WRIF bits to resume communication.

The Acknowledge Status Time Interrupt and Hold Enable feature generates an interrupt event and stretches the SCL line after the acknowledgement phase of a transaction. This feature is enabled by setting the Acknowledge Status Time Interrupt and Hold Enable (ACKTIE) bit. When enabled (ACKTIE = 1), the module hardware sets the CSTR bit and the Acknowledge Status Time Interrupt Flag (ACKTIF) bit and stretches the clock after the 9th falling edge of

SCL for all address, read, or write operations. Client software must clear both the ACKTIF and CSTR bits to resume communication.

### 36.3.12 Data Byte Count

The data byte count refers to the number of data bytes in a complete I²C packet. The data byte count does not include address bytes. The I²C Byte Count (I2CxCNT) register is used to specify the length, in bytes, of the complete transaction. The value loaded into I2CxCNT will be decremented by module hardware each time a data byte is transmitted or received by the module.

**Important:** The I2CxCNT register will not decrement past a zero value.

When a byte transfer causes the I2CxCNT register to decrement to '0', the Byte Count Interrupt Flag (CNTIF) bit is set, and if the Byte Count Interrupt Enable (CNTIE) is set, the general purpose I²C Interrupt Flag (I2CxIF) bit of the Peripheral Interrupt Registers (PIR) is also set. If the I²C Interrupt Enable (I2CxIE) bit of the Peripheral Interrupt Enable (PIE) registers is set, module hardware will generate an interrupt event.

**Important:** The I2CxIF bit is read-only and can only be cleared by clearing all the interrupt flag bits of the I2CxPIR register.

The I2CxCNT register can be read at any time, but it is recommended that a double read is performed to ensure a valid count value.

The I2CxCNT register can be written to; however, care is required to prevent register corruption. If the I2CxCNT register is written to during the 8th falling SCL edge of a reception, or during the 9th falling SCL edge of a transmission, the register value may be corrupted. In Client mode, I2CxCNT can be safely written to any time the clock is not being stretched (CSTR = 0), or after a Stop condition has been received (Stop Condition Interrupt Flag (PCIF) = 1). In Host mode, I2CxCNT can be safely written to any time the Host Data Ready (MDR) or Bus Free (BFRE) bits are set. If the I²C packet is longer than 255 bytes, the I2CxCNT register can be updated mid-message to prevent the count from reaching zero; however, the preventative measures listed above must be followed.

When in either Client Read or Host Write mode and the I2CxCNT value is non-zero (I2CxCNT != 0), the value of the ACKDT bit is used as the acknowledgement response. When I2CxCNT reaches zero (I2CxCNT = 0), the value of the Acknowledge End of Count (ACKCNT) bit is used for the acknowledgement response.

In Host read or write operations, when the I2CxCNT register is clear (I2CxCNT = 0) and the Restart Enable (RSEN) bit is clear, host hardware automatically generates a Stop condition upon the 9th falling edge of SCL. When I2CxCNT is clear (I2CxCNT = 0) and RSEN is set (RSEN = 1), host hardware will stretch the clock while it waits for the Start (S) bit to be set (S = 1). When the Start bit has been set, module hardware transmits a Restart condition followed by the address of the client it wishes to communicate with.

#### 36.3.12.1 Auto-Load I2CxCNT

The I2CxCNT register can be automatically loaded. Auto-loading of the I2CxCNT register is enabled when the Auto-Load I²C Count Register Enable (ACNT) bit is set (ACNT = 1).

In Host Transmit mode, the first byte following either the 7-bit or 10-bit client address is transferred from I2CxTXB into both I2CxCNT and the transmit shift register.

In Host Reception mode, the first byte received from the client is loaded into both I2CxCNT and I2CxRXB. The value of the Acknowledge Data (ACKDT) bit is used as the host's acknowledgement response to prevent a false NACK from being generated before the I2CxCNT register is updated with the new count value.

In Client Reception mode, the first byte received after a receiving a matching 7-bit or 10-bit address is loaded into both I2CxCNT and I2CxRXB, and the value of the ACKDT bit is used as the client's acknowledgement response.

In Client Transmit mode, the first byte loaded into I2CxTXB following the reception of a matching 7-bit or 10-bit address is transferred into both I2CxCNT and the transmit shift register.

© 2019-2023 Microchip Technology Inc. and its subsidiaries

**Important:** It is not necessary to preload the I2CxCNT register when using the auto-load feature. If no value is loaded by the 9th falling SCL edge following an address transmission or reception, the Byte Count Interrupt Flag (CNTIF) will be set by module hardware, and must be cleared by software to prevent an interrupt event before I2CxCNT is updated. Alternatively, I2CxCNT can be preloaded with a non-zero value to prevent the CNTIF from being set. In this case, the preloaded value will be overwritten once the new count value has been loaded into I2CxCNT.

### 36.3.13 DMA Integration

The I²C module can be used with the DMA for data transfers. The DMA can be triggered through software via the DMA Transaction (DGO) bit, or through the use of the following hardware triggers:

- I²C Transmit Interrupt Flag (I2CxTXIF)
- I²C Receive Interrupt Flag (I2CxRXIF)
- I²C Interrupt Flag (I2CxIF)
- I²C Error Interrupt Flag (I2CxEIF)

For I²C communication, the I2CxTXIF is commonly used as the hardware trigger source for host or client transmission, and I2CxRXIF is commonly used as the hardware trigger source for host or client reception.

#### 36.3.13.1 7-Bit Host Transmission

When address buffers are enabled (ABD = 0), I2CxADB1 is loaded with the client address, and I2CxCNT is loaded with a count value. At this point, I2CxTXB does not contain data, and the Transmit Buffer Empty (TXBE) bit is set (TXBE= 1). The I2CxTXIF bit is not set since it can only be set when the Host Mode Active (MMA) and TXBE bits are set. Once software sets the Start (S) bit, the MMA bit is set, and hardware transmits the client address. Upon the 8th falling SCL edge, since TXBE = 1, the Host Data Request (MDR) and I2CxTXIF bits are set, and hardware stretches the clock while the DMA loads I2CxTXB with data. Once the DMA loads I2CxTXB, the TXBE, MDR, and I2CxTXIF bits are cleared by hardware, and the DMA waits for the next occurrence of I2CxTXIF being set.

When address buffers are disabled (ABD = 1), software must load I2CxTXB with the client address to begin transmission. This is because I2CxTXIF can only be set when MMA = 1, and since a Start has not occurred, MMA = 0. Once the address has been transmitted, I2CxTXIF will be set, triggering the DMA to load I2CxTXB with data.

#### 36.3.13.2 10-Bit Host Transmission

When address buffers are enabled (ABD = 0), I2CxADB1 is loaded with the client high address, I2CxADB0 is loaded with the client low address, and I2CxCNT is loaded with a count value. Once software sets the Start (S) bit, the MMA bit is set, and hardware transmits the 10-bit client address. Upon the 8th falling SCL edge of the transmitted address low byte, since TXBE = 1, the MDR and I2CxTXIF bits are set, and hardware stretches the clock while the DMA loads I2CxTXB with data. Once the DMA loads I2CxTXB, the TXBE, MDR, and I2CxTXIF bits are cleared by hardware, and the DMA waits for the next occurrence of I2CxTXIF being set.

When address buffers are disabled (ABD = 1), software must load I2CxTXB with the client high address to begin transmission. Once the client high address has been transmitted, I2CxTXIF will be set, triggering the DMA to load I2CxTXB with client low address. Once the DMA loads I2CxTXB with the client low address, the TXBE, MDR, and I2CxTXIF bits are cleared by hardware, and the DMA waits for the next occurrence of I2CxTXIF being set.

#### 36.3.13.3 7/10-Bit Host Reception

In both 7-bit and 10-bit Host Receive modes, the state of the ABD bit is ignored. Once the complete 7-bit or 10-bit address has been received by the client, the client will transmit a data byte. Once the byte has been received by the host, hardware sets the I2CxRXIF bit, which triggers the DMA to read I2CxRXB. Once the DMA has read I2CxRXB, I2CxRXIF is cleared by hardware and the DMA waits for the next occurrence of I2CxRXIF being set.

#### 36.3.13.4 7-Bit Client Transmission

In 7-bit Client Transmission mode, the state of ABD is ignored. If the client receives the matching 7-bit address and TXBE is set, I2CxTXIF is set by hardware, triggering the DMA to load data into I2CxTXB. Once the data is transmitted from I2CxTXB, I2CxTXIF is set by hardware, triggering the DMA to once again load I2CxTXB with data. The DMA will continue to load data into I2CxTXB until I2CxCNT reaches a zero value. Once I2CxCNT reaches zero and the data is transmitted from I2CxTXB, I2CxTXIF will not be set, and the DMA will stop loading data.

#### 36.3.13.5 10-Bit Client Transmission

In 10-bit Client Transmission mode, the state of ABD is ignored. If there is no data in I2CxTXB after the client has received the address high byte with the R/$\overline{W}$ bit set, hardware sets I2CxTXIF, triggering the DMA to load I2CxTXB. The DMA will continue to load data into I2CxTXB until I2CxCNT reaches a zero value. Once I2CxCNT reaches zero and the data is transmitted from I2CxTXB, I2CxTXIF will not be set, and the DMA will stop loading data.

#### 36.3.13.6 7/10-Bit Client Reception

When address buffers are enabled (ABD = 0), client hardware loads I2CxADB0/I2CxADB1 with the matching address, while all data is received by I2CxRXB. Once the client loads I2CxRXB with a received data byte, hardware sets I2CxRXIF, which triggers the DMA to read I2CxRXB. The DMA will continue to read I2CxRXB whenever I2CxRXIF is set.

When address buffers are disabled (ABD = 1), the client loads I2CxRXB with the matching address byte(s) as they are received. Each received address byte sets I2CxRXIF, which triggers the DMA to read I2CxRXB. The DMA will continue to read I2CxRXB whenever I2CxRXIF is set.

### 36.3.14 Interrupts

The I$^2$C module offers several interrupt features designed to assist with communication functions. The interrupt hardware contains four high-level interrupts and several condition-specific interrupts.

#### 36.3.14.1 High-Level Interrupts

Module hardware provides four high-level interrupts:

- Transmit
- Receive
- General Purpose
- Error

These flag bits are read-only bits, and cannot be cleared by software.

The I$^2$C Transmit Interrupt Flag (I2CxTXIF) bit is set when the I2CxCNT register is non-zero (I2CxCNT != 0), and the transmit buffer, I2CxTXB, is empty as indicated by the Transmit Buffer Empty Status (TXBE) bit (TXBE = 1). If the I$^2$C Transmit Interrupt Enable (I2CxTXIE) bit is set, an interrupt event will occur when the I2CxTXIF bit becomes set. Writing new data to I2CxTXB, or setting the Clear Buffer (CLRBF) bit, will clear the interrupt condition. The I2CxTXIF bit is also used by the DMA as a trigger source.

**Important:** I2CxTXIF can only be set when either the Client Mode Active (SMA) or Host Mode Active (MMA) bits are set, and the I2CxCNT register is non-zero (I2CxCNT != 0). The SMA bit is only set after an address has been successfully acknowledged by a client device, which prevents false interrupts from being triggered on address reception. The MMA bit is set once the host completes the transmission of a Start condition.

The I$^2$C Receive Interrupt Flag (I2CxRXIF) bit is set when the receive shift register has loaded new data into the receive buffer, I2CxRXB. When new data is loaded into I2CxRXB, the Receive Buffer Full Status (RXBF) bit is set (RXBF = 1), which also sets I2CxRXIF. If the I$^2$C Receive Interrupt Enable (I2CxRXIE) bit is set, an interrupt event will occur when the I2CxRXIF bit becomes set. Reading data from I2CxRXB, or setting the CLRBF bit, will clear the interrupt condition. The I2CxRXIF bit is also used by the DMA as a trigger source.

**Important:** I2CxRXIF can only be set when either the Client Mode Active (SMA) or Host Mode Active (MMA) bits are set.

The I$^2$C Interrupt Flag (I2CxIF) is the general purpose interrupt. I2CxIF is set whenever any of the interrupt flag bits contained in the I$^2$C Peripheral Interrupt Register (I2CxPIR) and the associated interrupt enable bits contained in the I$^2$C Peripheral Interrupt Enable (I2CxPIE) register are set. If I2CxIF becomes set while the I$^2$C Interrupt Enable (I2CxIE) bit is set, an interrupt event will occur. I2CxIF is cleared by module hardware when all enabled interrupt flag bits in I2CxPIR are clear.

© 2019-2023 Microchip Technology Inc. and its subsidiaries

The I²C Error Interrupt Flag (I2CxEIF) is set whenever any of the interrupt flag bits contained in the I²C Error Register (I2CxERR) and their associated interrupt enable bits are set. If I2CxEIF becomes set while the I²C Error Interrupt Enable (I2CxEIE) bit is set, an interrupt event will occur. I2CxEIF is cleared by hardware when all enabled error interrupt flag bits in the I2CxERR register are clear.

#### 36.3.14.2 Condition-Specific Interrupts

In addition to the high-level interrupts, module hardware provides several condition-specific interrupts.

The I²C Peripheral Interrupt Register (I2CxPIR) contains the following interrupt flag bits:

- CNTIF: Byte Count Interrupt Flag
- ACKTIF: Acknowledge Status Time Interrupt Flag
- WRIF: Data Write Interrupt Flag
- ADRIF: Address Interrupt Flag
- PCIF: Stop Condition Interrupt Flag
- RSCIF: Restart Condition Interrupt Flag
- SCIF: Start Condition Interrupt Flag

When any of the flag bits in I2CxPIR become set and the associated interrupt enable bits in I2CxPIE are set, the generic I2CxIF is also set. If the generic I2CxIE bit is set, an interrupt event is generated whenever one of the I2CxPIR flag bits becomes set. If the I2CxIE bit is clear, the I2CxPIR flag bit will still be set by hardware; however, no interrupt event will be triggered.

CNTIF becomes set (CNTIF = 1) when the I2CxCNT register value reaches zero, indicating that all data bytes in the I²C packet have been transmitted or received. CNTIF is set after the 9th falling SCL edge when I2CxCNT reaches zero (I2CxCNT = 0).

ACKTIF is set (ACKTIF = 1) by the 9th falling edge of SCL for any byte when the device is addressed as a client in any Client or Multi-Host mode. If the Acknowledge Interrupt and Hold Enable (ACKTIE) bit is set and ACKTIF becomes set:

- If an $\overline{ACK}$ is detected, clock stretching is also enabled (CSTR = 1).
- If a NACK is detected, no clock stretching occurs (CSTR = 0).

WRIF is set (WRIF = 1) after the 8th falling edge of SCL when the module receives a data byte in Client or Multi-Host modes. Once the data byte is received, WRIF is set, as is the Receive Buffer Full Status (RXBF) bit, the I2CxRXIF bit, and if the Data Write Interrupt and Hold Enable (WRIE) bit is set, the generic I2CxIF bit is also set. WRIF is a read/write bit and must be cleared in software, while the RXBF, I2CxRXIF, and I2CxIF bits are read-only and are cleared by reading I2CxRXB or by setting the Clear Buffer bit (CLRBF = 1).

ADRIF is set on the 8th falling edge of SCL after the module has received a matching 7-bit address, after receiving a matching 10-bit upper address byte, and after receiving a matching 10-bit lower address byte in Client or Multi-Host modes. Upon receiving a matching 7-bit address or 10-bit upper address, the address is copied to I2CxADB0, the R/$\overline{W}$ bit setting is copied to the Read Information (R) bit, the Data (D) bit is cleared, and the ADRIF bit is set. If the Address Interrupt and Hold Enable (ADRIE) bit is set, I2CxIF is set, and the clock will be stretched while the module determines whether to $\overline{ACK}$ or NACK the transmitter. Upon receiving the matching 10-bit lower address, the address is copied to I2CxADB1, and the ADRIF bit is set. If ADRIE is also set, the clock is stretched while the module determines the $\overline{ACK}$/NACK response to return to the transmitter.

PCIF is set whenever a Stop condition is detected on the bus.

RSCIF is set upon the detection of a Restart condition.

SCIF is set upon the detection of a Start condition.

In addition to the I2CxPIR register, the I²C Error (I2CxERR) register contains three interrupt flag bits that are used to detect bus errors. These read/write bits are set by module hardware, but must be cleared by user software. The I2CxERR register also includes the interrupt enable bits for these three error conditions, and when set, will cause an interrupt event whenever the associated interrupt flag bit becomes set.

I2CxERR contains the following interrupt flag bits:

- BTOIF: Bus Time-out Interrupt Flag
- BCLIF: Bus Collision Interrupt Flag

- NACKIF: NACK Detect Interrupt Flag

BTOIF is set when a bus time-out occurs. The bus time-out period is configured using one of the time-out sources selected by the I²C Bus Time-out (I2CxBTO) register.

If a bus time-out event occurs while the module is configured as a client and is active (SMA = 1), BTOIF is set, the SMA and CSTR bits are cleared, and if the Bus Time-out Interrupt Enable (BTOIE) bit is set, the generic I²C Error Interrupt Flag (I2CxEIF) bit is set, and the module is reset.

If a bus time-out event occurs and the module is configured as a host and is active (MMA = 1), BTOIF is set, and if BTOIE is set, I2CxEIF is set, and the module attempts to issue a Stop condition. The Stop condition is generated only after the module detects an Idle condition, and the MMA bit is only cleared by hardware after the detection of a Stop condition.

BCLIF is set upon the detection of a bus collision. A bus collision occurs any time the SDA line is sampled at a logic low while the module expects both SCL and SDA lines to be at a high logic level. When a bus collision occurs, BCLIF is set, and if the Bus Collision Detect Interrupt Enable (BCLIE) bit is set, I2CxEIF is also set, and the module is reset.

NACKIF is set when either the host or client is active (SMA = 1 || MMA = 1) and a NACK response is detected on the bus. A NACK response occurs during the 9th SCL pulse in which the SDA line is released to a logic high. In Host mode, a NACK can be issued when the host has finished receiving data from a client, or when the host receives incorrect data. In Client mode, a NACK is issued when the client does not receive a matching address, or when it receives incorrect data. A NACK can also be automatically issued when any of the following bits become set, which will also set NACKIF and I2CxEIF:

- TXWE: Transmit Write Error Status
- RXRE: Receive Read Error Status
- TXU: Transmit Underflow Status
- RXO: Receive Overflow Status

**Important:** The I2CxEIF bit is read-only, and is only cleared by hardware after all enabled I2CxERR error flags have been cleared.

### 36.3.15 Operation in Sleep

The I²C module can operate while in Sleep mode.

In Client mode, the module can transmit and receive data as long as the system clock source operates in Sleep. If the generic I²C Interrupt Enable (I2CxIE) bit is set and the client receives or transmits a complete byte, I2CxIF is set and the device wakes up from Sleep.

In Host mode, both the system clock and the selected I2CxCLK source must be able to operate in Sleep. If the I2CxIE bit is set and the I2CxIF bit becomes set, the device wakes from Sleep.

## 36.4 I²C Operation

All I²C communication is performed in 9-bit segments consisting of an 8-bit address/data segment followed by a 1-bit acknowledgement segment. Address and data bytes are transmitted with the Most Significant bit (MSb) first. Interaction between the I²C module and other devices on the bus is controlled and monitored through several I²C Control, Status, and Interrupt registers.

To begin any I²C communication, mater hardware checks to ensure that the bus is in an Idle state as indicated by the Bus Free Status (BFRE) bit. When BFRE = 1, both SDA and SCL lines are floating to a logic high and the bus is considered ‘idle’. When the host detects an idle bus, it transmits a Start condition, followed by the address of the client it intends to communicate with. The client address can be either 7-bit or 10-bit, depending on the application design.

In 7-bit Addressing mode, the Least Significant bit (LSb) of the 7-bit client address is reserved for the Read/not Write (R/$\overline{W}$) bit, while in 10-bit Addressing mode, the LSb of the high address byte is reserved as the R/$\overline{W}$ bit. If the R/$\overline{W}$ bit is clear (R/$\overline{W}$ = 0), the host intends to read information from the client. If R/$\overline{W}$ is set (R/$\overline{W}$ = 1), the host intends

to write information to the client. If the addressed client exists on the bus, it must respond with an Acknowledgement ($\overline{ACK}$) condition.

Once a client has been successfully addressed, the host will continue to receive data from the client, write data to the client, or a combination of both. Data is always transmitted Most Significant bit (MSb) first. When the host has completed its transactions, it can either issue a Stop condition, signaling to the client that communication is to be terminated, or a Restart condition, informing the bus that the current host wishes to hold the bus to communicate with the same or other client devices.

### 36.4.1 I²C Client mode Operation

The I²C module provides four client operation modes as selected by the I²C Mode Select (MODE) bits:

- I²C Client mode with recognition of up to four 7-bit addresses
- I²C Client mode with recognition of up to two masked 7-bit addresses
- I²C Client mode with recognition of up to two 10-bit addresses
- I²C Client mode with recognition of one masked 10-bit address

During operation, the client device waits until module hardware detects a Start condition on the bus. Once the Start condition is detected, the client waits for the incoming address information to be received by the receive shift register. The address is then compared to the addresses stored in the I²C Address 0/1/2/3 registers (I2CxADR0, I2CxADR1, I2CxADR2, I2CxADR3), and if an address match is detected, client hardware transfers the matching address into either the I2CxADB0/I2CxADB1 registers or the I2CxRXB register, depending on the state of the Address Buffer Disable (ABD) bit. If there are no address matches, there is no response from the client.

#### 36.4.1.1 Client Addressing Modes

The I2CxADR0, I2CxADR1, I2CxADR2, and I2CxADR3 registers contain the client's addresses. The first byte (7-bit mode) or first and second bytes (10-bit mode) following a Start or Restart condition are compared to the values stored in the I2CxADR registers (see figure below). If an address match occurs, the valid address is transferred to the I2CxADB0/I2CxADB1 registers or I2CxRXB register, depending on the addressing mode and the state of the ABD bit.

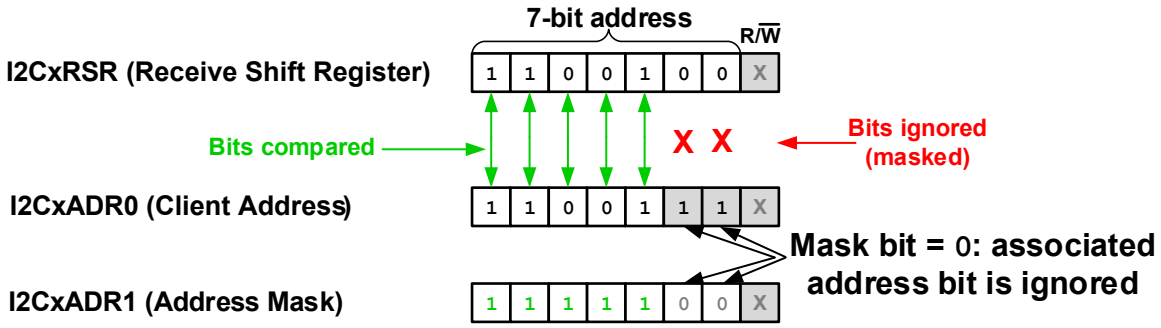
**Table 36-2. I²C Address Registers**

| Mode | I2CxADR0 | I2CxADR1 | I2CxADR2 | I2CxADR3 |
|---|---|---|---|---|
| 7-bit | 7-bit address | 7-bit address | 7-bit address | 7-bit address |
| 7-bit w/ masking | 7-bit address | 7-bit mask for I2CxADR0 | 7-bit address | 7-bit mask for I2CxADR2 |
| 10-bit | Address low byte | Address high byte | Address low byte | Address high byte |
| 10-bit w/ masking | Address low byte | Address high byte | Address low byte mask | Address high byte mask |

In 7-bit Address mode, the received address byte is compared to all four I2CxADR registers independently to determine a match. The R/$\overline{W}$ bit is ignored during address comparison. If a match occurs, the matching received address is transferred from the receive shift register to either the I2CxADR0 register (when ABD = 0) or to the I2CxRXB register (when ABD = 1), and the value of the R/$\overline{W}$ bit is loaded into the Read Information (R) bit.

In 7-bit Address with Masking mode, I2CxADR0 holds one client address and I2CxADR1 holds the mask value for I2CxADR0, while I2CxADR2 holds a second client address and I2CxADR3 holds the mask value for I2CxADR2. A zero bit in a mask register means that the associated bit in the address register is a 'don't care', which means that the particular address bit is not used in the address comparison between the received address in the shift register and the address stored in either I2CxADR0 or I2CxADR2 (see figure below).

**Figure 36-11. 7-Bit Address with Masking Example**




In 10-bit Address mode, I2CxADR0 and I2CxADR1, and I2CxADR2 and I2CxADR3, are combined to create two 10-bit addresses. I2CxADR0 and I2CxADR2 hold the lower eight bits of the address, while I2CxADR1 and I2CxADR3 hold the upper two bits of the address, the R/$\overline{W}$ bit, and the five-digit '11110' code assigned to the five Most Significant bits of the high address byte.

**Important:** The '11110' code is specified by the I²C Specification, but is not supported by Microchip. It is up to the user to ensure the correct bit values are loaded into the address high byte. If a host device has included the five-digit code in the address it intends to transmit, the client must also include those bits in client address.

The upper received address byte is compared to the values in I2CxADR1 and I2CxADR3, and if a match occurs, the address is stored in either I2CxADB1 (when ABD = 0) or in I2CxRXB (when ABD = 1), and the value of the R/$\overline{W}$ bit is transferred into the R bit. The lower received address byte is compared to the values in I2CxADR0 and I2CxADR2, and if a match occurs, the address is stored in either I2CxADB0 (when ABD = 0) or in I2CxRXB (when ABD = 1).

In 10-bit Address with Masking mode, I2CxADR0 and I2CxADR1 are combined to form the 10-bit address, while I2CxADR2 and I2CxADR3 are combined to form the 10-bit mask. The upper received address byte is compared to the masked value in I2CxADR1, and if a match occurs, the address is stored in either I2CxADB1 (when ABD = 0) or in I2CxRXB (when ABD = 1), and the value of the R/$\overline{W}$ bit is transferred into the R bit. The lower received address byte is compared to the value in I2CxADR0, and if a match occurs, the address is stored in either I2CxADB0 (when ABD = 0) or in I2CxRXB (when ABD = 1).

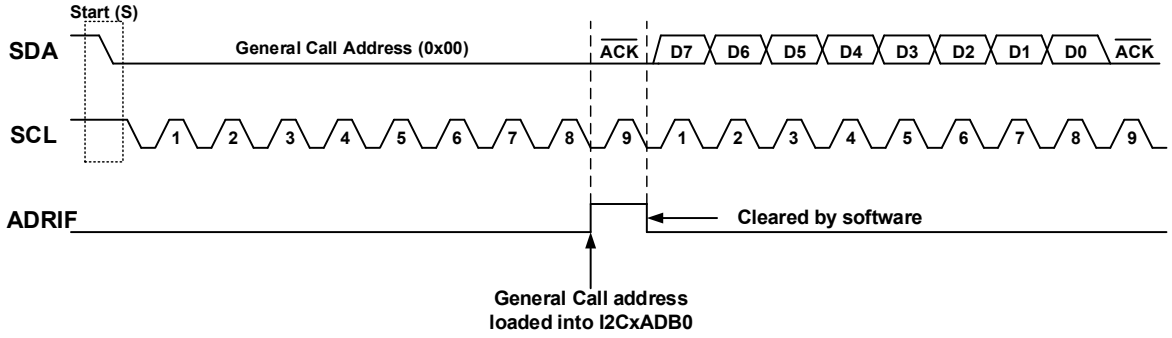
### 36.4.1.2 General Call Addressing Support

The I²C Specification reserves the address 0x00 as the General Call address. The General Call address is used to address all client modules connected to the bus at the same time. When a host issues a General Call, all client devices should, in theory, respond with an $\overline{ACK}$. The General Call Enable (GCEN) bit determines whether client hardware will respond to a General Call address. When GCEN is set (GCEN = 1), client hardware will respond to a General Call with an $\overline{ACK}$, and when GCEN is clear (GCEN = 0), the General Call is ignored, and the client responds with a NACK.

When the module receives a General Call, the ADRIF bit is set and the address is stored in I2CxADB0. If the ADRIE bit is set, the module will generate an interrupt and stretch the clock after the 8th falling edge of SCL. This allows the client to determine the acknowledgement response to return to the host (see figure below).

**Important:** When using the General Call addressing feature, loading the I2CxADR0/1/2/3 registers with the 0x00 address is not recommended. Additionally, client hardware only supports General Call addressing in 7-bit Addressing modes.

**Figure 36-12. General Call Addressing**




### 36.4.1.3 Client Operation in 7-Bit Addressing Modes

The upper seven bits of an address byte are used to determine a client's address, while the LSb of the address byte is reserved as the Read/not Write (R/$\overline{W}$) bit. When R/$\overline{W}$ is set (R/$\overline{W}$ = 1), the host device intends to read data from the client. When R/$\overline{W}$ is clear (R/$\overline{W}$ = 0), the host device intends to write data to the client. When an address match occurs, the R/$\overline{W}$ bit is copied to the Read Information (R) bit, and the 7-bit address is copied to I2CxADB0.

#### 36.4.1.3.1 Client Transmission (7-bit Addressing Mode)

The following section describes the sequence of events that occur when the module is transmitting data in 7-bit Addressing mode:

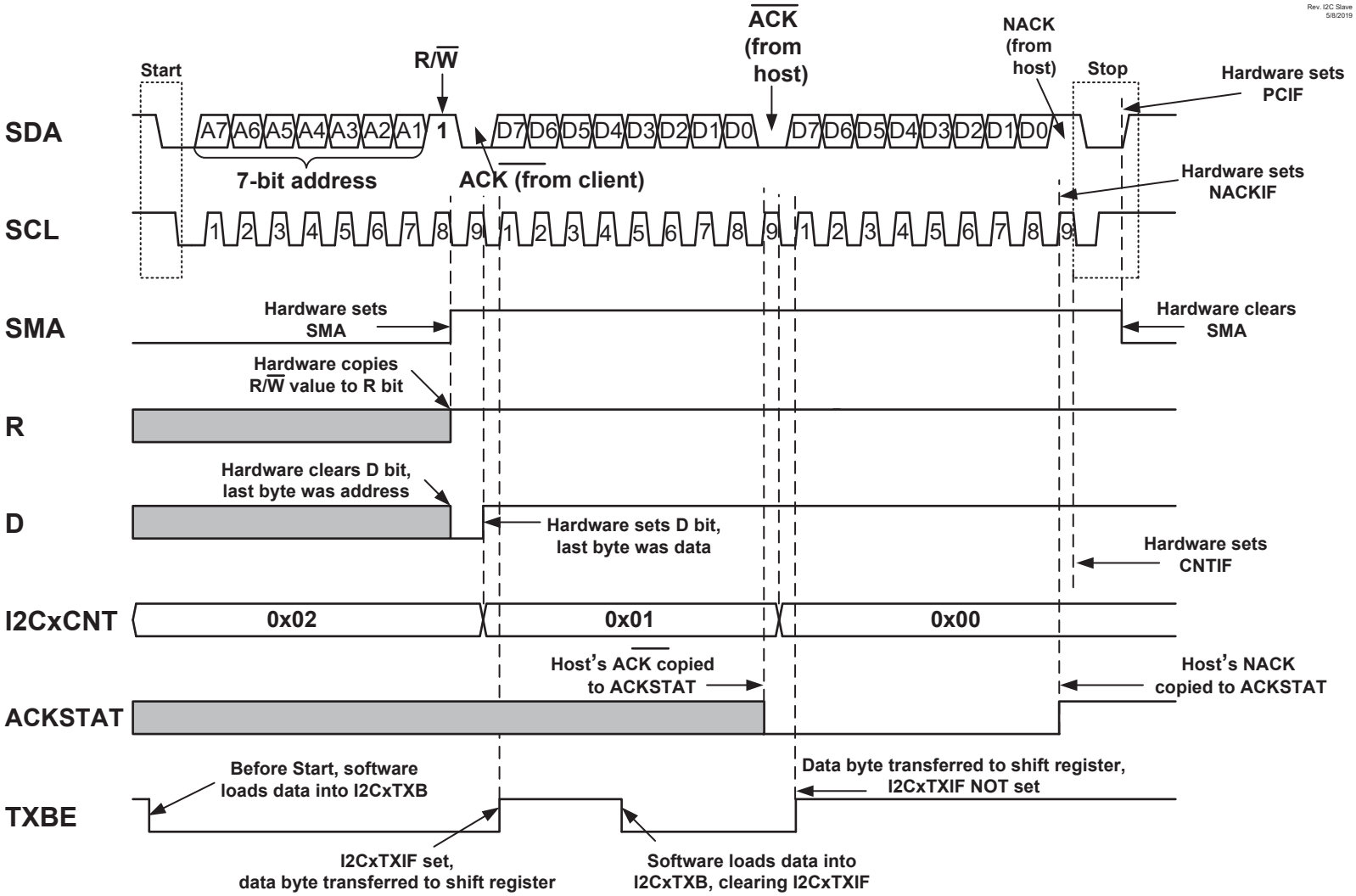
1. The host device issues a Start condition. Once the Start condition has been detected, client hardware sets the Start Condition Interrupt Flag (SCIF) bit. If the Start Condition Interrupt Enable (SCIE) bit is also set, the generic I2CxIF is also set.
2. Host hardware transmits the 7-bit client address with the R/$\overline{W}$ bit set, indicating that it intends to read data from the client.
3. The received address is compared to the values in the I2CxADR registers. If the client is configured in 7-bit Addressing mode (no masking), the received address is independently compared to each of the I2CxADR0/1/2/3 registers. In 7-bit Addressing with Masking mode, the received address is compared to the masked value of I2CxADR0 and I2CxADR2.
   If an address match occurs:
   - The Client mode Active (SMA) bit is set by module hardware.
   - The R/$\overline{W}$ bit value is copied to the Read Information (R) bit by module hardware.
   - The Data (D) bit is cleared by hardware, indicating the last received byte was an address.
   - The Address Interrupt Flag (ADRIF) bit is set. If the Address Interrupt and Hold Enable (ADRIE) bit is set, and the Clock Stretching Disable (CSD) bit is clear, hardware sets the Client Clock Stretching (CSTR) bit and the generic I2CxIF bit. This allows time for the client to read either I2CxADB0 or I2CxRXB and selectively $\overline{ACK}$/NACK based on the received address. When the client has finished processing the address, software must clear CSTR to resume operation.
   - The matching received address is loaded into either the I2CxADB0 register or into the I2CxRXB register as determined by the Address Buffer Disable (ABD) bit. When ABD is clear (ABD = 0), the matching address is copied to I2CxADB0. When ABD is set (ABD = 1), the matching address is copied to I2CxRXB, which also sets the Receive Buffer Full Status (RXBF) bit and the $I^2C$ Receive Interrupt Flag (I2CxRXIF) bit. I2CxRXIF is a read-only bit, and must be cleared by either reading I2CxRXB or by setting the Clear Buffer (CLRBF) bit (CLRBF = 1).

   If no address match occurs, the module remains idle.
4. If the Transmit Buffer Empty Status (TXBE) bit is set (TXBE = 1), I2CxCNT has a non-zero value (I2CxCNT != 0), and the $I^2C$ Transmit Interrupt Flag (I2CxTXIF) is set (I2CxTXIF = 1), client hardware sets CSTR, stretches the clock (when CSD = 0), and waits for software to load I2CxTXB with data. I2CxTXB must be loaded to clear I2CxTXIF. Once data is loaded into I2CxTXB, hardware automatically clears CSTR to resume communication.

5. The host device transmits the 9th clock pulse, and client hardware transfers the value of the ACKDT bit onto the SDA line. If there are pending errors, such as a receive overflow (RXO = 1), client hardware automatically generates a NACK condition. NACKIF is set, and the module goes idle.
6. Upon the 9th falling SCL edge, the data byte in I2CxTXB is transferred to the transmit shift register, and I2CxCNT is decremented by one. Additionally, the Acknowledge Status Time Interrupt Flag (ACKTIF) bit is set. If the Acknowledge Status Time Interrupt and Hold Enable (ACKTIE) bit is also set, the generic I2CxIF is set, and if client hardware generated an $\overline{\text{ACK}}$, the CSTR bit is also set and the clock is stretched (when CSD = 0). If a NACK was generated, the CSTR bit remains unchanged. Once complete, software must clear CSTR and ACKTIF to release the clock and continue operation.
7. If the client generated an $\overline{\text{ACK}}$ and I2CxCNT is non-zero, host hardware transmits eight clock pulses, and client hardware begins to shift the data byte out of the shift register starting with the Most Significant bit (MSb).
8. After the 8th falling edge of SCL, client hardware checks the status of TXBE and I2CxRXB. If TXBE is set and I2CxCNT has a non-zero count value, hardware sets CSTR and the clock is stretched (when CSD = 0) until software loads I2CxTXB with new data. Once I2CxTXB has been loaded, hardware clears TXBE, I2CxTXIF, and CSTR to resume communication.
9. Once the host hardware clocks in all eight data bits, it transmits the 9th clock pulse along with the $\overline{\text{ACK}}$/NACK response back to the client. Client hardware copies the $\overline{\text{ACK}}$/NACK value to the Acknowledge Status (ACKSTAT) bit and sets ACKTIF. If ACKTIE is also set, client hardware sets the generic I2CxIF bit and CSTR, and stretches the clock (when CSD = 0). Software must clear CSTR to resume operation.
10. After the 9th falling edge of SCL, data currently loaded in I2CxTXB is transferred to the transmit shift register, setting both TXBE and I2CxTXIF. I2CxCNT is decremented by one. If I2CxCNT is zero (I2CxCNT = 0), CNTIF is set.
11. If I2CxCNT is non-zero and the host issued an $\overline{\text{ACK}}$ on the last byte (ACKSTAT = 0), the host transmits eight clock pulses, and client hardware begins to shift data out of the shift register.
12. Repeat steps 8 – 11 until the host has received all the requested data (I2CxCNT = 0). Once all data has been received, the host issues a NACK, followed by either a Stop or Restart condition. Once the NACK has been received by the client, hardware sets NACKIF and clears SMA. If the NACK Detect Interrupt Enable (NACKIE) bit is also set, the generic $I^2C$ Error Interrupt Flag (I2CxEIF) is set. If the host issued a Stop condition, client hardware sets the Stop Condition Interrupt Flag (PCIF). If the host issued a Restart condition, client hardware sets the Restart Condition Interrupt Flag (RSCIF). If the associated interrupt enable bits are also set, the generic I2CxIF is also set.

**Important:** I2CxEIF is read-only, and is cleared by hardware when all enable interrupt flag bits in I2CxERR are cleared.

**Figure 36-13. 7-Bit Client mode Transmission (No Clock Stretching)**




© 2019-2023 Microchip Technology Inc. and its subsidiaries

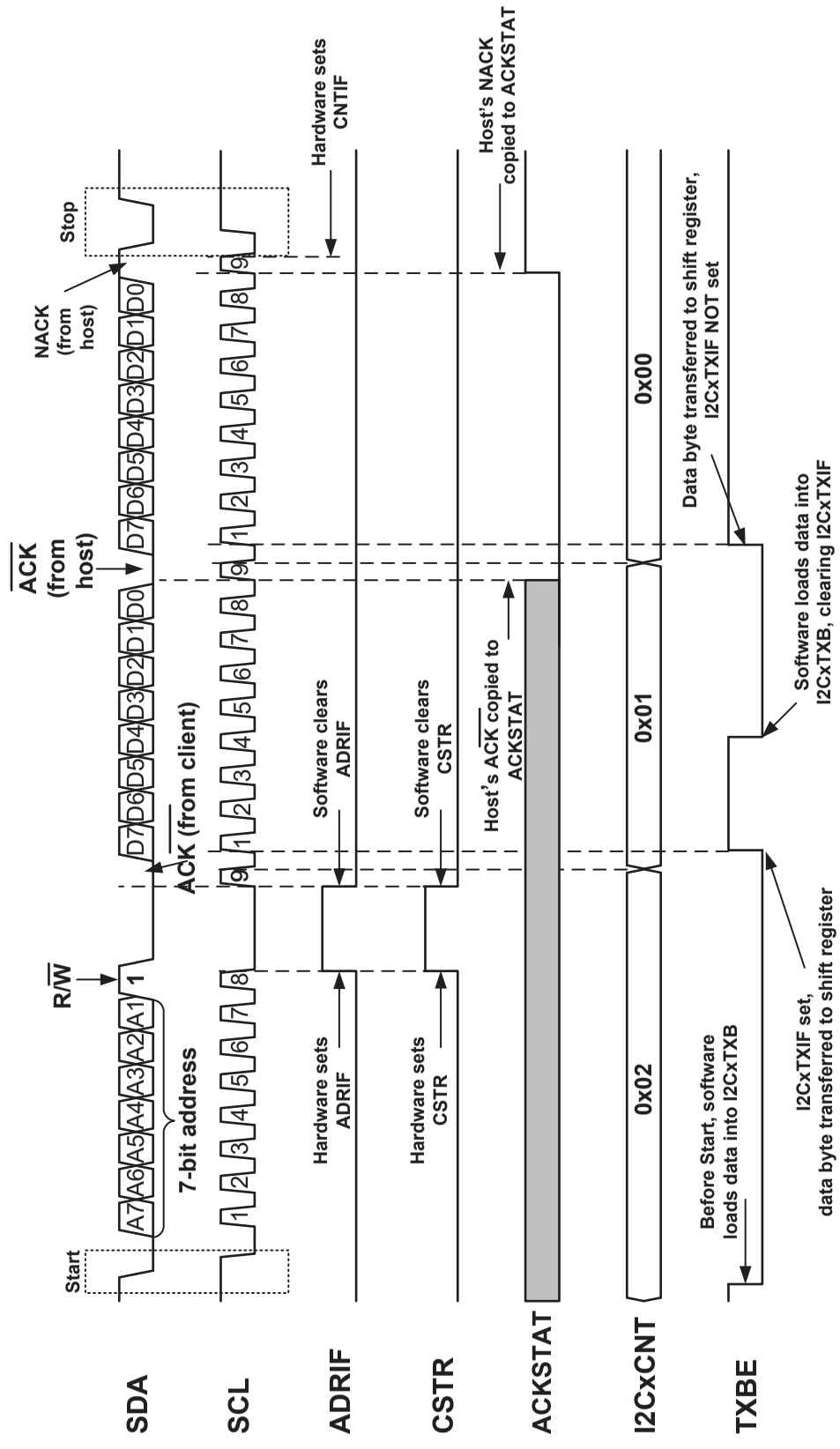
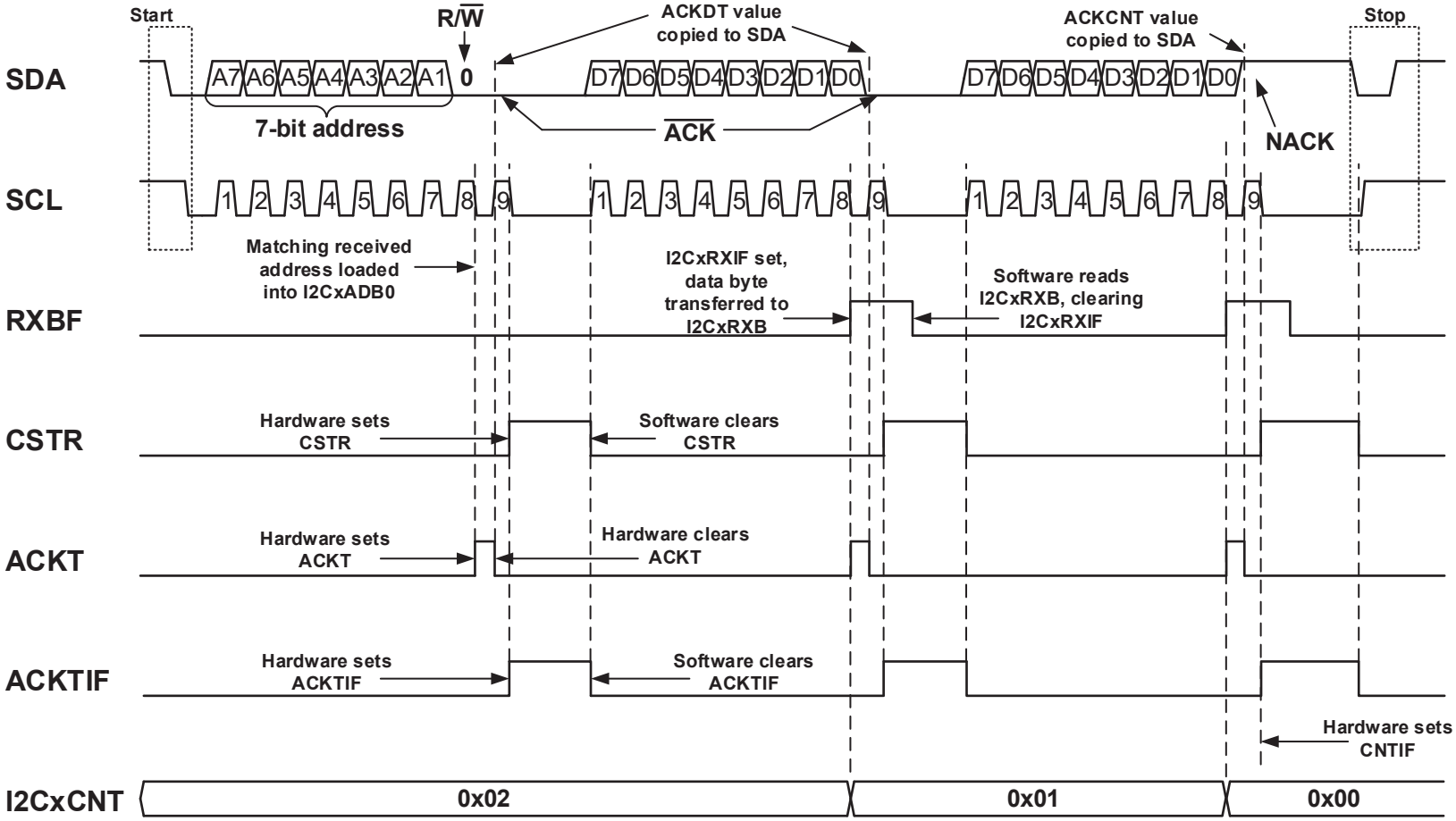



**Figure 36-14. 7-Bit Client mode Transmission (ADRIE = 1)**

**Figure 36-15. 7-Bit Client mode Transmission (ACKTIE = 1)**




© 2019-2023 Microchip Technology Inc. and its subsidiaries

#### 36.4.1.3.2 Client Reception (7-Bit Addressing Mode)

The following section describes the sequence of events that occur when the module is receiving data in 7-bit Addressing mode:

1. The host issues a Start condition. Once the Start is detected, client hardware sets the Start Condition Interrupt Flag (SCIF) bit. If the Start Condition Interrupt Enable (SCIE) bit is also set, the generic I2CxIF bit is also set.
2. The host transmits the 7-bit client address with the R/$\overline{W}$ bit clear, indicating that it intends to write data to the client.
3. The received address is compared to the values in the I2CxADR registers. If the client is configured in 7-bit Addressing mode (no masking), the received address is independently compared to each of the I2CxADR0/1/2/3 registers. In 7-bit Addressing with Masking mode, the received address is compared to the masked value of I2CxADR0 and I2CxADR2.
   If an address match occurs:
   - The Client Mode Active (SMA) bit is set by module hardware.
   - The R/$\overline{W}$ bit value is copied to the Read Information (R) bit by module hardware.
   - The Data (D) bit is cleared (D = 0) by hardware, indicating the last received byte was an address.
   - The Address Interrupt Flag (ADRIF) bit is set (ADRIF = 1). If the Address Interrupt and Hold Enable (ADRIE) bit is set (ADRIE = 1), and the Clock Stretching Disable (CSD) bit is clear (CSD = 0), hardware sets the Client Clock Stretching (CSTR) bit and the generic I2CxIF bit. This allows time for the client to read either I2CxADB0 or I2CxRXB and selectively $\overline{ACK}$/NACK based on the received address. When the client has finished processing the address, software must clear CSTR to resume operation.
   - The matching received address is loaded into either the I2CxADB0 register or into the I2CxRXB register as determined by the Address Buffer Disable (ABD) bit. When ABD is clear (ABD = 0), the matching address is copied to I2CxADB0. When ABD is set (ABD = 1), the matching address is copied to I2CxRXB, which also sets the Receive Buffer Full Status (RXBF) bit and the $I^2C$ Receive Interrupt Flag (I2CxRXIF) bit. I2CxRXIF is a read-only bit, and must be cleared by either reading I2CxRXB or by setting the Clear Buffer (CLRBF) bit (CLRBF = 1).

   If no address match occurs, the module remains Idle.
4. The host device transmits the 9th clock pulse, and the client hardware transfers the value of the ACKDT bit onto the SDA line. If there are pending errors, such as a receive overflow (RXO = 1), client hardware automatically generates a NACK condition. NACKIF is set, and the module goes Idle.
5. Upon the 9th falling SCL edge, the Acknowledge Status Time Interrupt Flag (ACKTIF) bit is set. If the Acknowledge Interrupt and Hold Enable (ACKTIE) bit is also set, the generic I2CxIF is set, and if the client hardware generated an $\overline{ACK}$, the CSTR bit is also set and the clock is stretched (when CSD = 0). If a NACK was generated, the CSTR bit remains unchanged. Once complete, software must clear CSTR and ACKTIF to release the clock and continue operation.
6. If the client hardware generated a NACK, the host hardware generates a Stop condition, the Stop Condition Interrupt Flag (PCIF) bit is set when the client hardware detects the Stop condition, and the client goes Idle. If an $\overline{ACK}$ was generated, the host hardware transmits the first seven bits of the 8-bit data byte.
7. If data remains in I2CxRXB (RXBF = 1 and I2CxRXIF = 1) when the first seven bits of the new byte are received by the shift register, CSTR is set, and if CSD is clear, the clock is stretched after the 7th falling edge of SCL. This allows time for the client to read I2CxRXB, which clears RXBF and I2CxRXIF, and prevents a receive buffer overflow. Once RXBF and I2CxRXIF are cleared, hardware releases SCL.
8. Host hardware transmits the 8th bit of the current data byte into the client receive shift register. Client hardware then transfers the complete byte into I2CxRXB on the 8th falling edge of SCL, and sets the following bits:
   - I2CxRXIF
   - I2CxIF
   - Data Write Interrupt Flag (WRIF)
   - Data (D)
   - RXBF

   I2CxCNT is decremented by one. If the Data Write Interrupt and Hold Enable (WRIE) bit is set (WRIE = 1), hardware sets CSTR (when CSD = 0) and stretches the clock, allowing time for client software to read

I2CxRXB and determine the state of the ACKDT bit that is transmitted back to the host. Once the client determines the acknowledgement response, software clears CSTR to allow further communication.

9. Host hardware transmits the 9th clock pulse. If there are pending errors, such as receive buffer overflow, client hardware automatically generates a NACK condition, sets NACKIF, and the module goes Idle. If I2CxCNT is non-zero (I2CxCNT != 0), client hardware transmits the value of ACKDT as the acknowledgement response to the host. It is up to software to configure ACKDT appropriately. In most cases, the ACKDT bit should be clear (ACKDT = 0) so that the host receives an $\overline{\text{ACK}}$ response (logic low level on SDA during the 9th clock pulse). If I2CxCNT is zero (I2CxCNT = 0), the client hardware transmits the value of the Acknowledge End of Count (ACKCNT) bit as the Acknowledgement response, rather than the value of ACKDT. It is up to software to configure ACKCNT appropriately. In most cases, ACKCNT must be set (ACKCNT = 1), which represents a NACK condition. When host hardware detects a NACK on the bus, it will generate a Stop condition. If ACKCNT is clear (ACKCNT = 0), an $\overline{\text{ACK}}$ will be issued, and host hardware will not issue a Stop condition.
10. Upon the 9th falling edge of SCL, the ACKTIF bit is set. If ACKTIE is also set, the generic I2CxIF is set, and if CSD is clear, client hardware sets CSTR and stretches the clock. This allows time for software to read I2CxRXB. Once complete, software must clear both CSTR and ACKTIF to release the clock and continue communication.
11. Repeat steps 6 -10 until the host has transmitted all the data (I2CxCNT = 0), or until the host issues a Stop or Restart condition.

**Figure 36-16. 7-Bit Client Mode Reception (No Clock Stretching)**

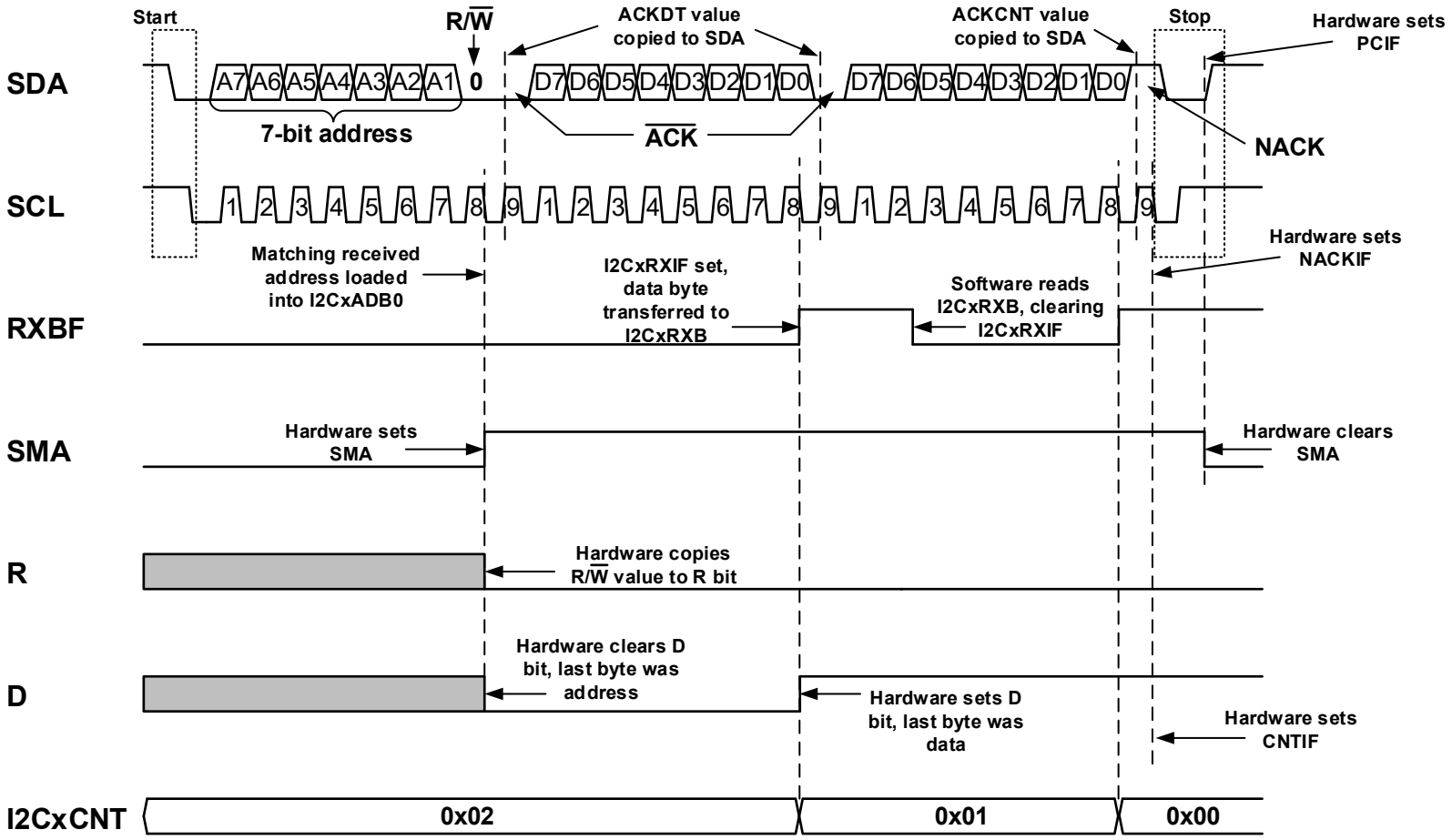
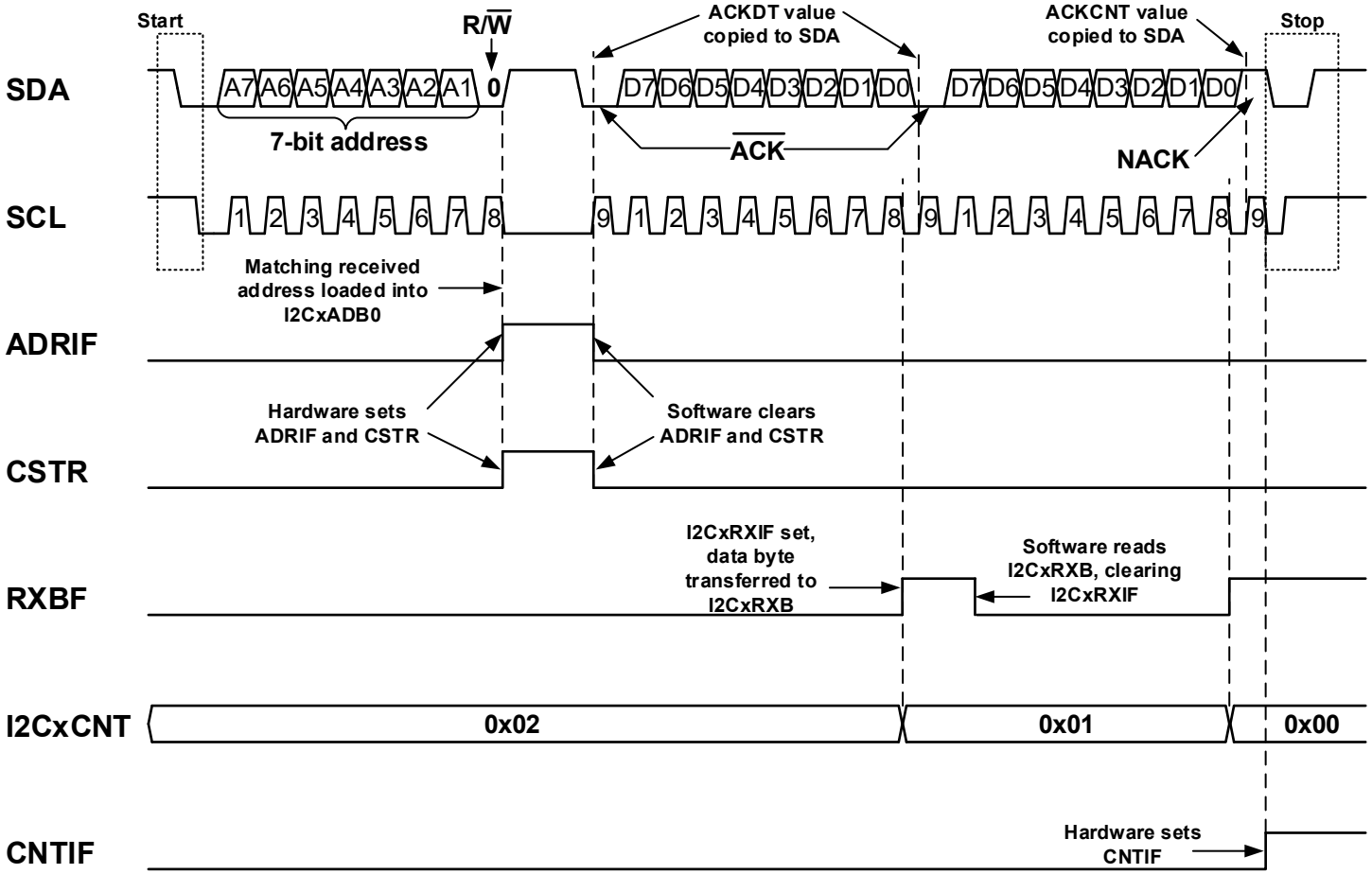
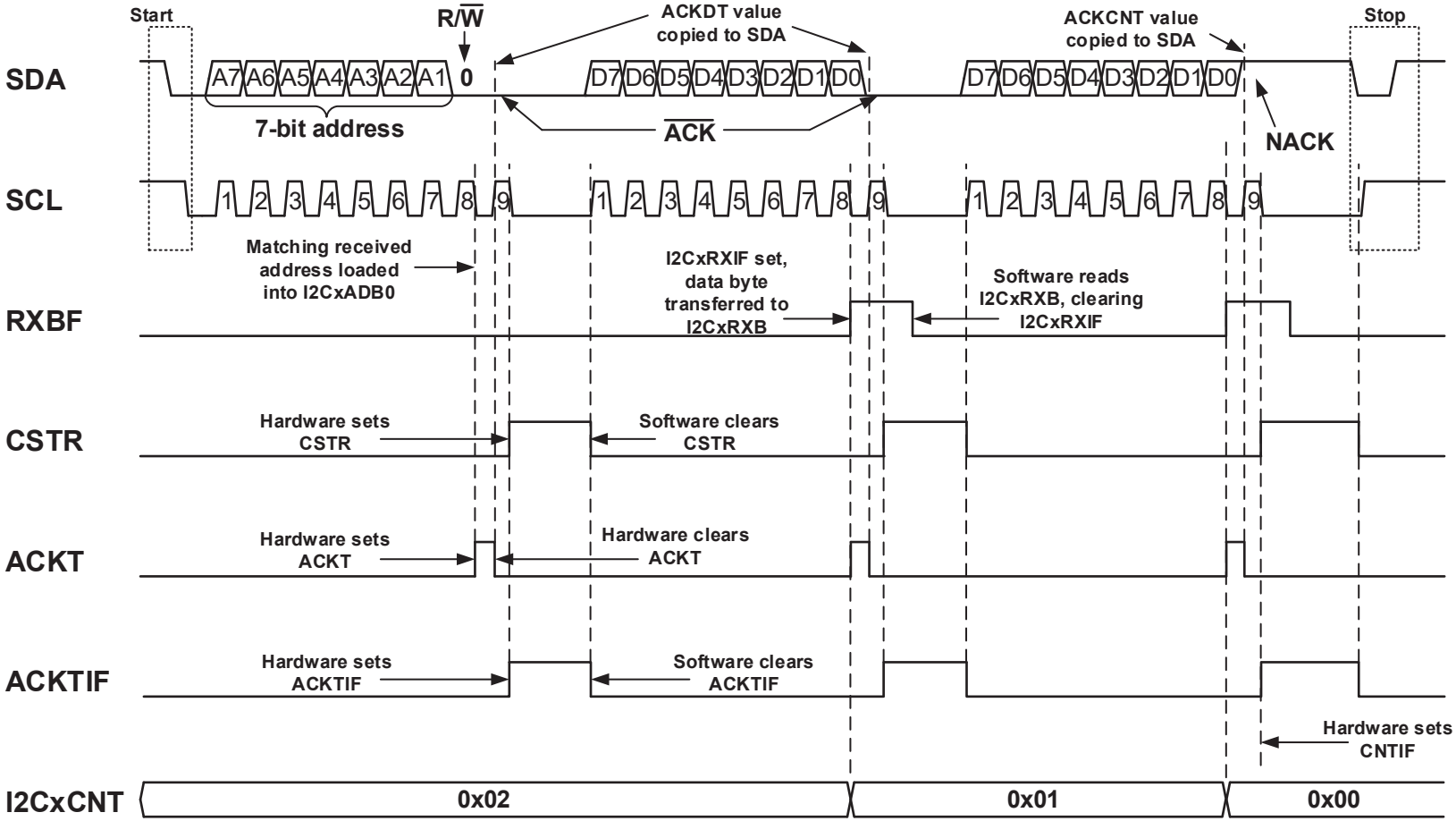


**Figure 36-17. 7-Bit Client Mode Reception (ADRIE = 1)**



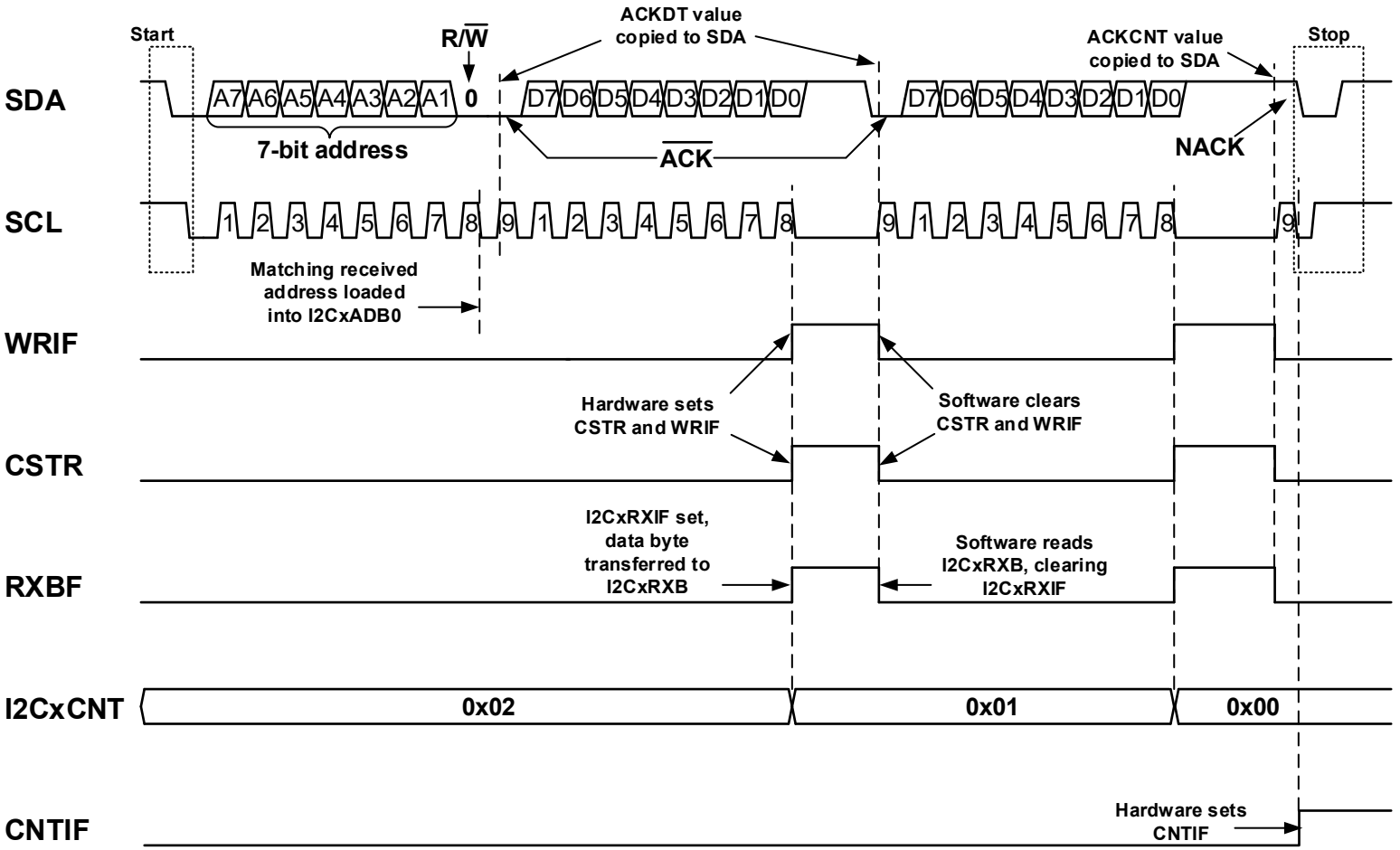

© 2019-2023 Microchip Technology Inc. and its subsidiaries

**Figure 36-18. 7-Bit Client Mode Reception (ACKTIE = 1)**




© 2019-2023 Microchip Technology Inc. and its subsidiaries

**Figure 36-19. 7-Bit Client Mode Reception (WRIE = 1)**



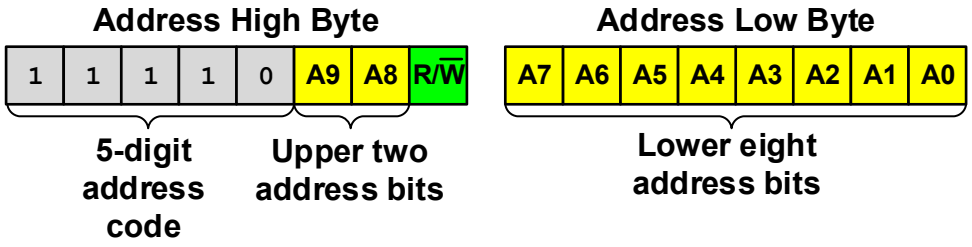

© 2019-2023 Microchip Technology Inc. and its subsidiaries

#### 36.4.1.4 Client Operation in 10-Bit Addressing Modes

In 10-bit Addressing modes, the first two bytes following a Start condition form the 10-bit address (see figure below). The first byte (address high byte) holds the upper two address bits, the R/$\overline{W}$ bit, and a five digit code (11110) as defined by the I$^2$C Specification. The second byte (address low byte) holds the lower eight address bits. In all 10-bit Addressing modes, the R/$\overline{W}$ value contained in the first byte must always be zero (R/$\overline{W}$ = 0). If the host intends to read data from the client, it must issue a Restart condition, followed by the address high byte with R/$\overline{W}$ set (R/$\overline{W}$ = 1).

The first byte is compared to the values in the I2CxADR1 and I2CxADR3 registers in 10-bit Addressing mode, or to the masked value of I2CxADR1 in 10-bit Addressing with Masking mode. The second byte is compared to the values in the I2CxADR0 and I2CxADR2 registers in 10-bit Addressing mode, or to the masked value of I2CxADR0 in 10-bit Addressing with Masking mode. If an address high byte match occurs, the high address byte is copied to I2CxADB1 and the R/$\overline{W}$ bit value is copied to the Read Information (R) bit, and if an address low byte match occurs, the low address byte is copied to I2CxADB0.

**Figure 36-20. Upper and Lower 10-Bit Address Bytes**




##### 36.4.1.4.1 Client Transmission (10-Bit Addressing Mode)

The following section describes the sequence of events that occur when the module is transmitting data in 10-bit Addressing mode:

1. The host device issues a Start condition. Once the Start condition has been detected, client hardware sets the Start Condition Interrupt Flag (SCIF) bit. If the Start Condition Interrupt Enable (SCIE) bit is also set, the generic I2CxIF is also set.
2. Host hardware transmits the 10-bit high address byte with the R/$\overline{W}$ bit clear (R/$\overline{W}$ = 0).
3. Client hardware compares the received address to the values in the I2CxADR registers. If the client is configured in 10-bit Addressing mode (no masking), the received high address byte is compared to the values in I2CxADR1 and I2CxADR3. In 10-bit Addressing with Masking mode, the received high address byte is compared to the masked value of I2CxADR1.
   If an address match occurs:
   - The R/$\overline{W}$ value is copied to the Read Information (R) bit by module hardware.
   - The Data (D) bit is cleared by hardware.
   - The Address Interrupt Flag (ADRIF) bit is set (ADRIF = 1).
   - The matching address is loaded into either the I2CxADB1 register or into the I2CxRXB register as determined by the Address Buffer Disable (ABD) bit. When ABD is clear (ABD = 0), the matching address is copied to I2CxADB1. When ABD is set (ABD = 1), the matching address is copied to I2CxRXB, which also sets the Receive Buffer Full Status (RXBF) bit and the I$^2$C Receive Interrupt Flag (I2CxRXIF) bit.

**Important:** Regardless of whether the Address Interrupt and Hold Enable (ADRIE) bit is set, clock stretching does not occur when the R/$\overline{W}$ bit is clear in 10-bit Addressing modes.

If no address match occurs, the module remains Idle.

4. The host device transmits the 9th clock pulse and client hardware transfers the value of the ACKDT bit onto the SDA line. If there are pending errors, such as a receive buffer overflow (RXO = 1), client hardware generates a NACK and the module goes Idle.
5. The host device transmits the low address byte. If the client is configured in 10-bit Addressing mode (no masking), the received low address byte is compared to the values in I2CxADR0 and I2CxADR2. In 10-bit Addressing with Masking mode, the received low address byte is compared to the masked value of I2CxADR0.
   If a match occurs:
   - The Client Mode Active (SMA) bit is set by module hardware.
   - ADRIF is set. If ADRIE is set, and the Clock Stretching Disable (CSD) bit is clear, hardware sets the Client Clock Stretching (CSTR) bit and the generic I2CxIF bit. This allows time for the client to read either I2CxADB0 or I2CxRXB and selectively $\overline{ACK}$/NACK based on the received address. When the client has finished processing the address, software must clear CSTR to resume operation.
   - The matching received address is loaded into either the I2CxADB0 register or into the I2CxRXB register as determined by the ABD bit. When ABD is clear (ABD = 0), the matching address is copied to I2CxADB0. When ABD is set (ABD = 1), the matching address is copied to I2CxRXB, which also sets RXBF and I2CxRXIF. I2CxRXIF is a read-only bit and must be cleared by either reading I2CxRXB or by setting the Clear Buffer (CLRBF) bit (CLRBF = 1).

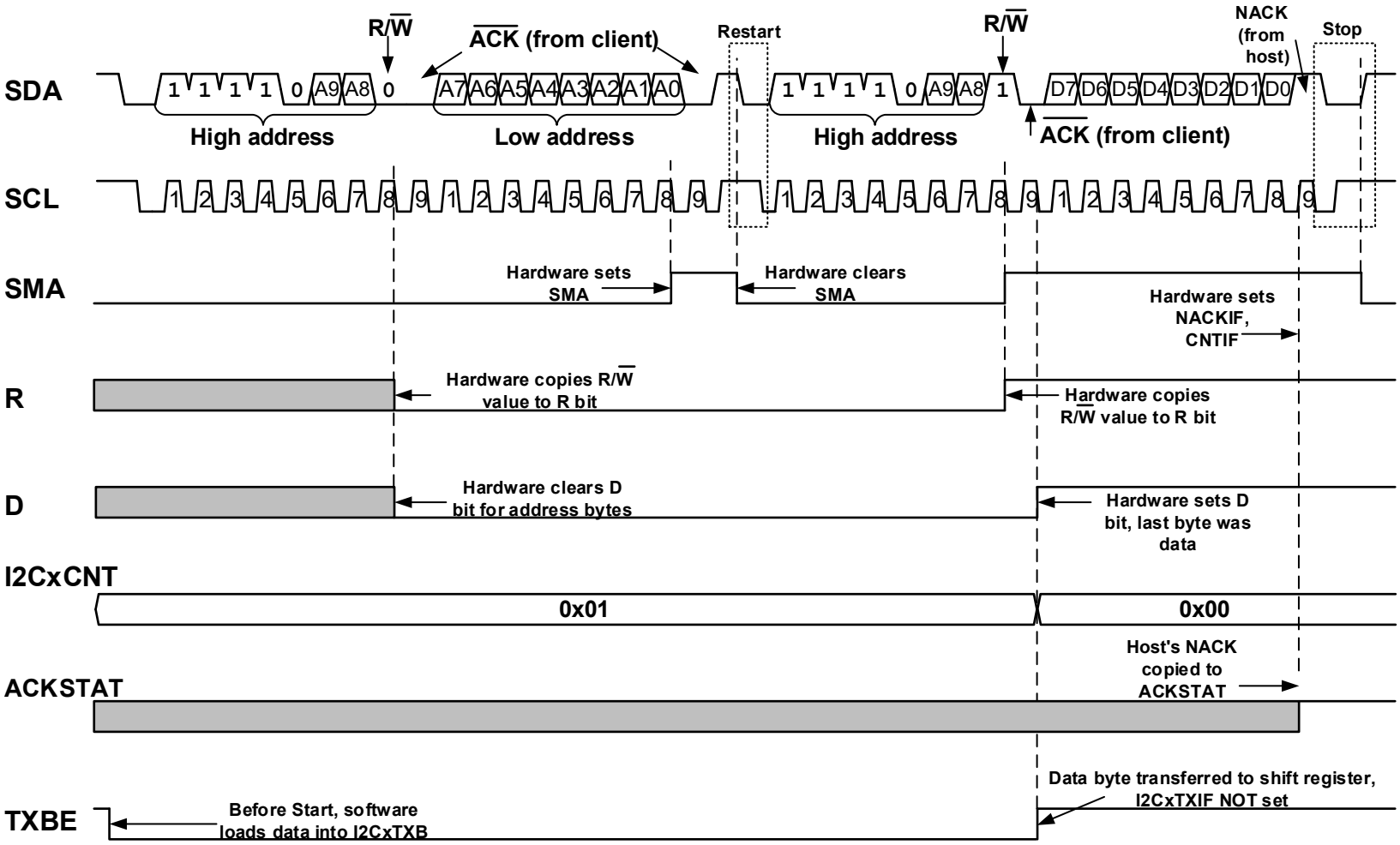
   If no match occurs, the module goes Idle.
6. The host device transmits the 9th clock pulse and client hardware transfers the value of the ACKDT bit onto the SDA line. If there are pending errors, such as a receive buffer overflow (RXO = 1), client hardware generates a NACK and the module goes Idle.
7. After the 9th falling edge of SCL, the Acknowledge Status Time Interrupt Flag (ACKTIF) bit is set. If the Acknowledge Time Interrupt and Hold Enable (ACKTIE) bit is also set, the generic I2CxIF is set, and if client hardware generated an $\overline{ACK}$, the CSTR bit is also set and the clock is stretched (when CSD = 0). If a NACK was generated, the CSTR bit remains unchanged. Once completed, software must clear CSTR and ACKTIF to release the clock and resume operation.
8. Host hardware issues a Restart condition (cannot be a Start condition), and once the client detects the Restart, hardware sets the Restart Condition Interrupt Flag (RSCIF). If the Restart Condition Interrupt Enable (RSCIE) bit is also set, the generic I2CxIF is also set.
9. Host hardware transmits the client's high address byte with R/$\overline{W}$ set.
   If the received high address byte matches:
   - The R/$\overline{W}$ bit value is copied to the R bit.
   - The SMA bit is set.
   - The D bit is cleared, indicating the last byte as an address.
   - ADRIF is set. If ADRIE is set, and the CSD bit is clear, hardware sets CSTR and the generic I2CxIF bit. This allows time for the client to read either I2CxADB1 or I2CxRXB and selectively $\overline{ACK}$/NACK based on the received address. When the client has finished processing the address, software must clear CSTR to resume operation.
   - The matching received address is loaded into either the I2CxADB1 register or into the I2CxRXB register as determined by the ABD bit. When ABD is clear (ABD = 0), the matching address is copied to I2CxADB1. When ABD is set (ABD = 1), the matching address is copied to I2CxRXB, which also sets RXBF and I2CxRXIF. I2CxRXIF is a read-only bit and must be cleared by either reading I2CxRXB or by setting CLRBF (CLRBF = 1).

   If the address does not match, the module goes Idle.
10. If the Transmit Buffer Empty Status (TXBE) bit is set (TXBE = 1), I2CxCNT has a non-zero value (I2CxCNT != 0), and the $I^2C$ Transmit Interrupt Flag (I2CxTXIF) is set (I2CxTXIF = 1), client hardware sets CSTR, stretches the clock (when CSD = 0) and waits for software to load I2CxTXB with data. I2CxTXB must be loaded to clear I2CxTXIF. Once data is loaded into I2CxTXB, hardware automatically clears CSTR to resume communication.
11. The host device transmits the 9th clock pulse and client hardware transfers the value of the ACKDT bit onto the SDA line. If there are pending errors, such as a receive overflow (RXO = 1), client hardware automatically generates a NACK condition. NACKIF is set and the module goes Idle.

12. Upon the 9th falling SCL edge, the data byte in I2CxTXB is transferred to the transmit shift register and I2CxCNT is decremented by one. Additionally, the ACKTIF bit is set. If the ACKTIE bit is also set, the generic I2CxIF is set, and if client hardware generated an $\overline{\text{ACK}}$, the CSTR bit is also set and the clock is stretched (when CSD = 0). If a NACK was generated, the CSTR bit remains unchanged. Once complete, software must clear CSTR and ACKTIF to release the clock and continue operation.
13. If the client generated an $\overline{\text{ACK}}$ and I2CxCNT is non-zero, host hardware transmits eight clock pulses and client hardware begins to shift the data byte out of the shift register starting with the Most Significant bit (MSb).
14. After the 8th falling edge of SCL, client hardware checks the status of TXBE and I2CxCNT. If TXBE is set and I2CxCNT has a non-zero count value, hardware sets CSTR and the clock is stretched (when CSD = 0) until software loads I2CxTXB with new data. Once I2CxTXB has been loaded, hardware clears CSTR to resume communication.
15. Once the host hardware clocks in all eight data bits, it transmits the 9th clock pulse along with the $\overline{\text{ACK}}$/NACK response back to the client. Client hardware copies the $\overline{\text{ACK}}$/NACK value to the Acknowledge Status (ACKSTAT) bit and sets ACKTIF. If ACKTIE is also set, client hardware sets the generic I2CxIF bit and CSTR, and stretches the clock (when CSD = 0). Software must clear CSTR to resume operation.
16. After the 9th falling edge of SCL, data currently loaded in I2CxTXB is transferred to the transmit shift register, setting both TXBE and I2CxTXIF. I2CxCNT is decremented by one. If I2CxCNT is zero (I2CxCNT = 0), CNTIF is set.
17. If I2CxCNT is non-zero and the host issued an $\overline{\text{ACK}}$ on the last byte (ACKSTAT = 0), the host transmits eight clock pulses, and client hardware begins to shift data out of the shift register.
18. Repeat Steps 13-17 until the host has received all the requested data (I2CxCNT = 0). Once all data is received, host hardware transmits a NACK condition, followed by either a Stop or Restart condition. Once the NACK has been received by the client, hardware sets NACKIF and clears SMA. If the NACK Detect Interrupt Enable (NACKIE) bit is also set, the generic $I^2C$ Error Interrupt Flag (I2CxEIF) is set. If the host issued a Stop condition, client hardware sets the Stop Condition Interrupt Flag (PCIF). If the host issued a Restart condition, client hardware sets the Restart Condition Interrupt Flag (RSCIF) bit. If the associated interrupt enable bits are also set, the generic I2CxIF is also set.

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

**Figure 36-21. 10-Bit Client Mode Transmission**




© 2019-2023 Microchip Technology Inc. and its subsidiaries

#### 36.4.1.4.2 Client Reception (10-Bit Addressing Mode)

The following section describes the sequence of events that occur when the module is receiving data in 7-bit Addressing mode:

1. The host issues a Start condition. Once the Start is detected, the client hardware sets the Start Condition Interrupt Flag (SCIF) bit. If the Start Condition Interrupt Enable (SCIE) bit is also set, the generic I2CxIF bit is also set.
2. Host hardware transmits the address high byte with the R/$\overline{W}$ bit clear (R/$\overline{W}$ = 0).
3. The received high address byte is compared to the values in the I2CxADR registers. If the client is configured in 10-bit Addressing mode (no masking), the received high address byte is compared to the values in the I2CxADR1 and I2CxADR3 registers. If the client is configured in 10-bit Addressing with Masking mode, the received high address byte is compared to the masked value in the I2CxADR1 register.
   If a high address match occurs:
   - The R/$\overline{W}$ bit value is copied to the Read Information (R) bit by module hardware.
   - The Data (D) bit is cleared (D = 0) by hardware, indicating the last received byte was an address.
   - The Address Interrupt Flag (ADRIF) bit is set (ADRIF = 1). It is important to note that regardless of whether the Address Interrupt and Hold Enable (ADRIE) bit is set, clock stretching does not occur when the R/$\overline{W}$ bit is clear in 10-bit Addressing modes.
   - The matching address is loaded into either the I2CxADB1 register or into the I2CxRXB register as determined by the Address Buffer Disable (ABD) bit. When ABD is clear (ABD = 0), the matching address is copied to I2CxADB1. When ABD is set (ABD = 1), the matching address is copied to I2CxRXB, which also sets the Receive Buffer Full Status (RXBF) bit and the $I^2C$ Receive Interrupt Flag (I2CxRXIF) bit.

   If no address match occurs, the module remains Idle.
4. The host device transmits the 9th clock pulse and the client hardware transfers the value of the ACKDT bit onto the SDA line. If there are pending errors, such as a receive buffer overflow (RXO = 1), the client hardware generates a NACK and the module goes Idle.
5. The host device transmits the low address byte. If the client is configured in 10-bit Addressing mode (no masking), the received low address byte is compared to the values in I2CxADR0 and I2CxADR2. In 10-bit Addressing with Masking mode, the received low address byte is compared to the masked value of I2CxADR0.
   If a match occurs:
   - The Client Mode Active (SMA) bit is set by module hardware.
   - ADRIF is set. If ADRIE is set, and the Clock Stretching Disable (CSD) bit is clear, hardware sets the Client Clock Stretching (CSTR) bit and the generic I2CxIF bit. This allows time for the client to read either I2CxADB0 or I2CxRXB and selectively $\overline{ACK}$/NACK based on the received address. When the client has finished processing the address, software must clear CSTR to resume operation.
   - The matching received address is loaded into either the I2CxADB0 register or into the I2CxRXB register as determined by the ABD bit. When ABD is clear (ABD = 0), the matching address is copied to I2CxADB0. When ABD is set (ABD = 1), the matching address is copied to I2CxRXB, which also sets the RXBF and the I2CxRXIF bits. I2CxRXIF is a read-only bit and must be cleared by either reading I2CxRXB or by setting the Clear Buffer (CLRBF) bit (CLRBF = 1).

   If no match occurs, the module goes Idle.
6. The host device transmits the 9th clock pulse and the client hardware transfers the value of the ACKDT bit onto the SDA line. If there are pending errors, such as a receive buffer overflow (RXO = 1), the client hardware generates a NACK and the module goes Idle.
7. After the 9th falling edge of SCL, the Acknowledge Status Time Interrupt Flag (ACKTIF) bit is set. If the Acknowledge Time Interrupt and Hold Enable (ACKTIE) bit is also set, the generic I2CxIF is set, and if the client hardware generated an $\overline{ACK}$, the CSTR bit is also set and the clock is stretched (when CSD = 0). If a NACK was generated, the CSTR bit remains unchanged. Once completed, software must clear CSTR and ACKTIF to release the clock and resume operation.
8. If the client hardware generated a NACK, host hardware generates a Stop condition, the Stop Condition Interrupt Flag (PCIF) is set when the client hardware detects the Stop condition, and the client goes Idle. If an $\overline{ACK}$ was generated, host hardware transmits the first seven bits of the 8-bit data byte.

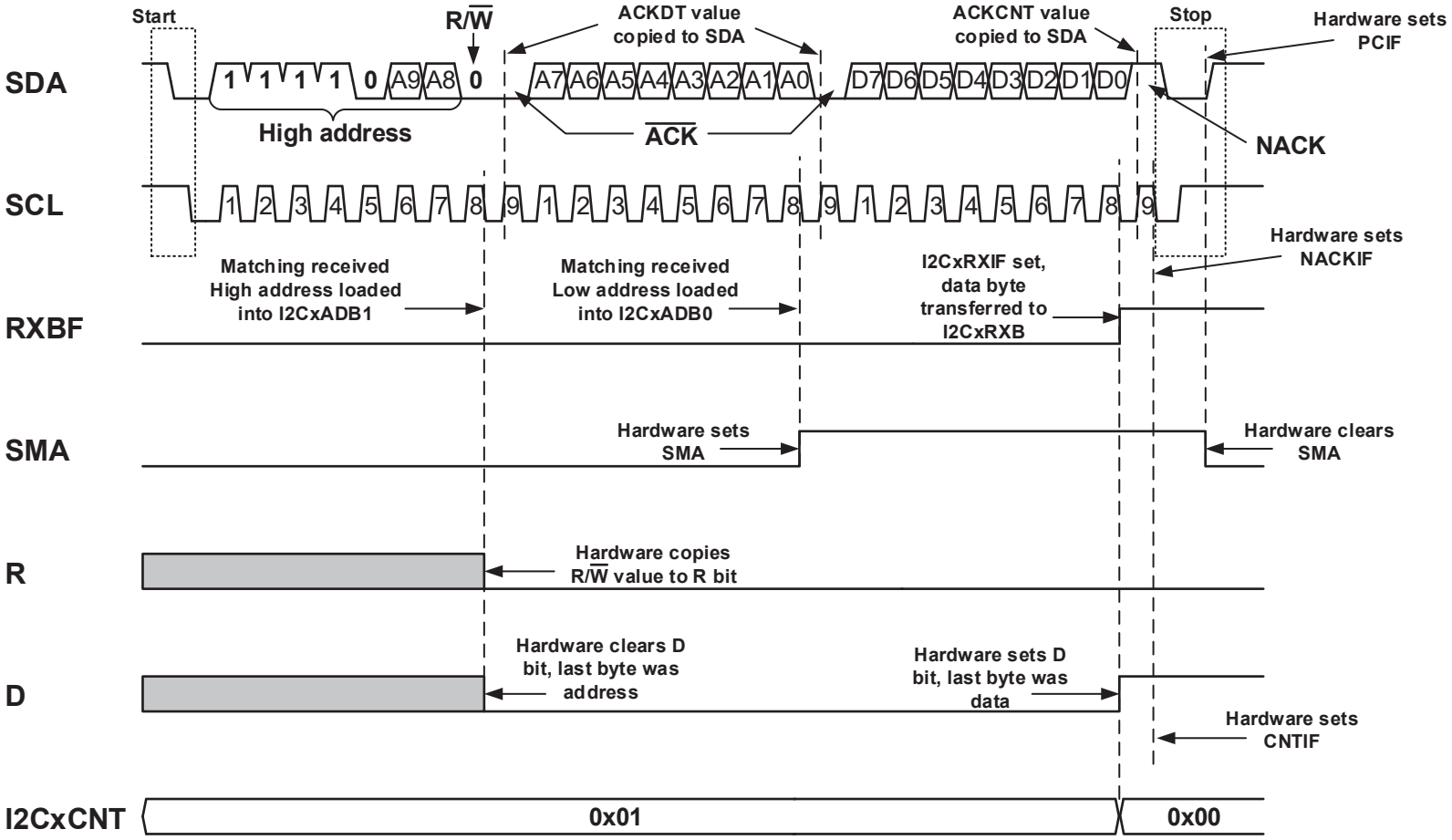
© 2019-2023 Microchip Technology Inc. and its subsidiaries

9. If data remains in I2CxRXB (RXBF = 1 and I2CxRXIF = 1) when the first seven bits of the new byte are received by the shift register, CSTR is set, and if CSD is clear, the clock is stretched after the 7th falling edge of SCL. This allows time for the client to read I2CxRXB, which clears RXBF and I2CxRXIF, and prevents a receive buffer overflow. Once I2CxRXB has been read, RXBF and I2CxRXIF are cleared and hardware releases SCL.
10. Host hardware transmits the 8th bit of the current data byte into the client receive shift register. The client hardware then transfers the complete byte into I2CxRXB on the 8th falling edge of SCL and sets the following bits:
    - I2CxRXIF
    - I2CxIF
    - Data Write Interrupt Flag (WRIF)
    - Data (D)
    - RXBF

    I2CxCNT is decremented by one. If the Data Write Interrupt and Hold Enable (WRIE) bit is set (WRIE = 1), hardware sets CSTR (when CSD = 0) and stretches the clock, allowing time for client software to read I2CxRXB and determine the state of the ACKDT bit that is transmitted back to the host. Once the client determines the acknowledgement response, software clears CSTR to allow further communication.
11. Upon the 9th falling edge of SCL, the ACKTIF bit is set. If ACKTIE is also set, the generic I2CxIF is set, and if CSD is clear, the client hardware sets CSTR and stretches the clock. This allows time for software to read I2CxRXB. Once complete, software must clear both CSTR and ACKTIF to release the clock and continue communication.
12. Repeat Steps 8 – 11 until the host has transmitted all the data (I2CxCNT = 0), or until the host issues a Stop or Restart condition.

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

**Figure 36-22. 10-Bit Client Mode Reception**




© 2019-2023 Microchip Technology Inc. and its subsidiaries

### 36.4.2 I²C Host Mode Operation

The I²C module provides two host operation modes as selected by the I²C Mode Select (MODE) bits:

- I²C Host mode with 7-bit addressing
- I²C Host mode with 10-bit addressing

To begin any I²C communication, host hardware checks to ensure that the bus is in an Idle state, which means both the SCL and SDA lines are floating in a high logic state as indicated by the Bus Free Status (BFRE) bit.

Once Host hardware has determined that the bus is free (BFRE = 1), it examines the state of the Address Buffer Disable (ABD) bit. The ABD bit determines whether the I2CxADB registers are used.

When ABD is clear (ABD = 0), address buffers I2CxADB0 and I2CxADB1 are active. In 7-bit Addressing mode, software loads I2CxADB1 with the 7-bit client address and R/$\overline{W}$ bit setting, and also loads I2CxTXB with the first byte of data . In 10-bit Addressing mode, software loads I2CxADB1 with the address high byte and I2CxADB0 with the address low byte, and also loads I2CxTXB with the first data byte. Software must issue a Start condition to initiate communication with the client.

When ABD is set (ABD = 1), the address buffers are inactive. In this case, communication begins as soon as software loads the client address into I2CxTXB. Writes to the Start bit (S) are ignored.

In 7-bit Addressing mode, the Least Significant bit (LSb) of the 7-bit address byte acts as the Read/not Write (R/$\overline{W}$) information bit, while in 10-bit Addressing mode, the LSb of the address high byte is reserved as the R/$\overline{W}$ bit. When R/$\overline{W}$ is set, the host intends to read data from the client (see figure below). When R/$\overline{W}$ is clear, the host intends to write data to the client (see figure below). The host may also wish to read or write data to a specific location, such as writing to a specific EEPROM location. In this case, the host issues a Start condition, followed by the client's address with the R/$\overline{W}$ bit clear. Once the client acknowledges the address, the first data byte following the 7-bit or 10-bit address is used as the client's specific register location. If the host intends to read data from the specific location, it must issue a Restart condition, followed by the client address with the R/$\overline{W}$ bit set (see figure below). If the addressed client device exists on the bus, it must respond with an Acknowledge ($\overline{ACK}$) sequence.

Once a client has acknowledged its address, the host begins to receive data from the client or transmits data to the client. Data is always transmitted Most Significant bit (MSb) first. When the host wishes to halt further communication, it transmits either a Stop condition, signaling to the client that communication is to be terminated, or a Restart condition, informing the bus that the current host wishes to hold the bus to communicate with the same or other client devices.

**Figure 36-23. 7-Bit Host Read Diagram**

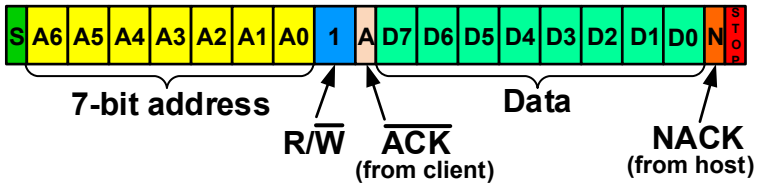


**Figure 36-24. 7-Bit Host Read Diagram (From a Specific Memory/Register Location)**

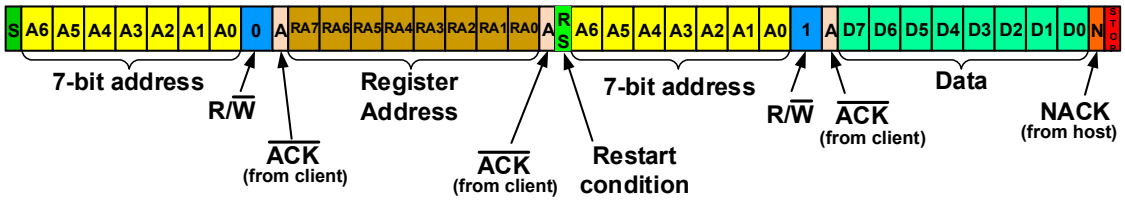



**Figure 36-25. 10-Bit Host Read Diagram**

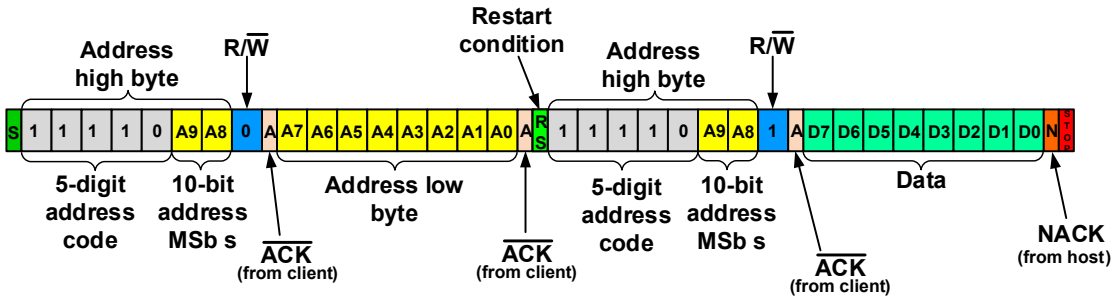



**Figure 36-26. 7-Bit Host Write Diagram**

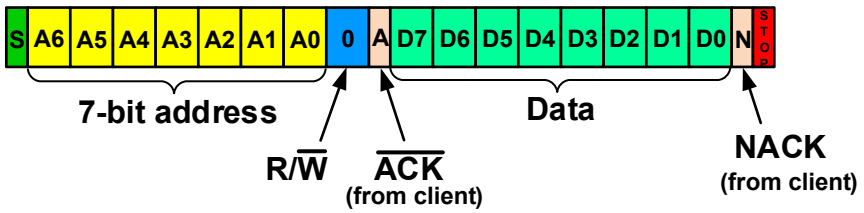


**Figure 36-27. 7-Bit Host Write Diagram (To a Specific Memory/Register Location)**

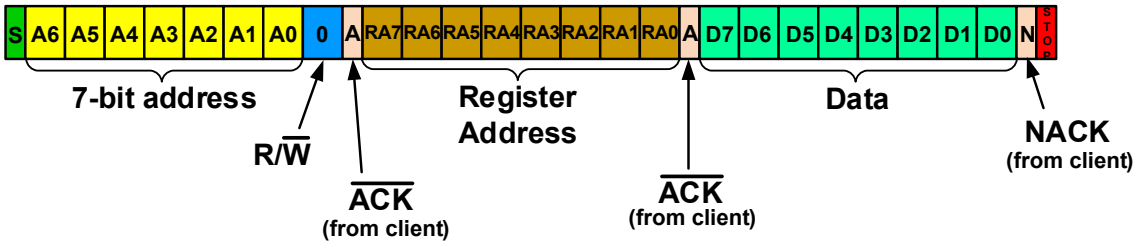
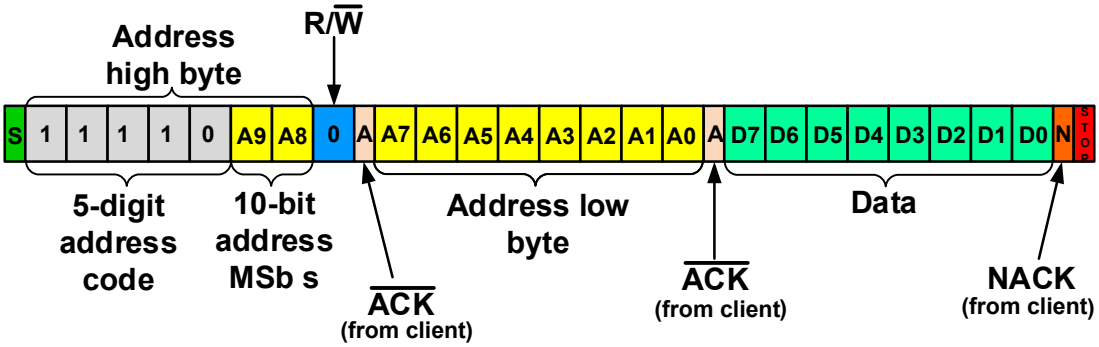



**Figure 36-28. 10-Bit Host Write Diagram**




### 36.4.2.1 Bus Free Time

The Bus Free Status (BFRE) bit indicates the activity status of the bus. When BFRE is set (BFRE = 1), the bus is in an Idle state (both SDA and SCL are floating high), and any host device residing on the bus can compete for control of the bus. When BFRE is clear (BFRE = 0), the bus is in an Active state, and any attempts by a host to control the bus will cause a collision.

The Bus Free Time (BFRET) bits determine the length of time, in terms of $I^2C$ clock pulses, before the bus is considered idle. Once module hardware detects logic high levels on both SDA and SCL, it monitors the $I^2C$ clock signal, and when the desired number of pulses have occurred, module hardware sets BFRE. The BFRET bits are also used to ensure that the module meets the minimum Stop hold time as defined by the $I^2C$ Specification.

### 36.4.2.2 Host Clock Timing

The Serial Clock (SCL) signal is generated by module hardware via the $I^2C$ Clock Selection (I2CxCLK) register, and the Fast Mode Enable (FME) bit.

I2CxCLK contains several clock source selections. The clock source selections typically include variants of the system clock and timer resources.

**Important:** When using a timer as the clock source, the timer must also be configured. Additionally, when using the HFINTOSC as a clock source it is important to understand that the HFINTOSC frequency selected by the OSCFRQ register is used as the clock source. The clock divider selected by the NDIV bits is not used. For example, if OSCFRQ selects 4 MHz as the HFINTOSC clock frequency, and the NDIV bits select a divide by four scaling factor, the $I^2C$ Clock Frequency will be 4 MHz and not 1 MHz since the divider is ignored.

The FME bit acts as a prescaler (clock divider) to the clock source selected by I2CxCLK.

When FME is clear (FME = 0), one SCL period ($T_{SCL}$) is equal to five clock periods of the selected I2CxCLK source. In other words, the I2CxCLK source is divided by five. For example, if the MFINTOSC (500 kHz) clock source is selected and the FME bit is clear, the actual SCL frequency is 100 kHz (see Equation below).

**Equation 36-1. SCL Frequency (FME = 0)**

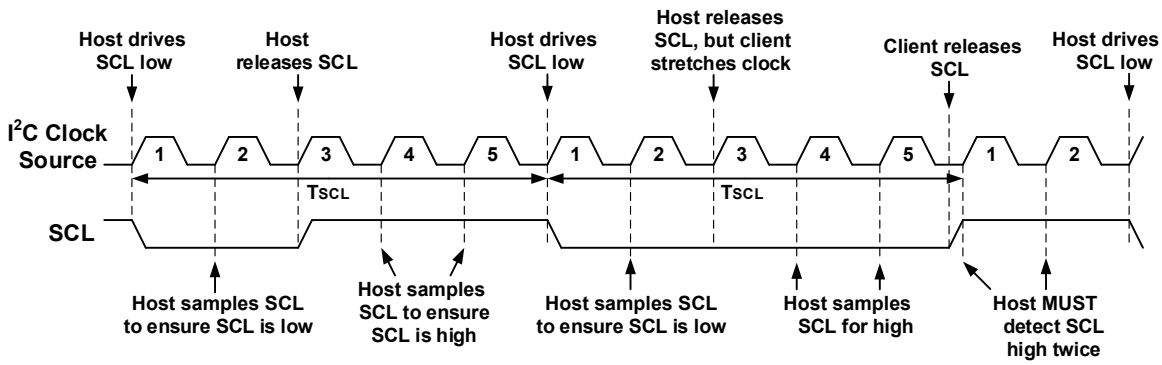
$$f_{SCL} = \frac{f_{I2CxCLK}}{5}$$

Example:

- I2CxCLK: MFINTOSC (500 kHz)
- FME: FME = 0

When FME is clear, host hardware uses the first I2CxCLK source period to drive SCL low (see figure below). During the second I2CxCLK period, hardware verifies that SCL is in fact low. During the third period, hardware releases SCL, allowing it to float high. Host hardware then uses the fourth and fifth I2CxCLK periods to sample SCL to verify that SCL is high. If a client is holding SCL low (clock stretch) during the fourth and/or fifth I2CxCLK period, host hardware samples each successive I2CxCLK period until a high level is detected on SCL. Once the high level is detected, host hardware samples SCL during the next two I2CxCLK periods to verify that SCL is high.

**Figure 36-29. SCL Timing (FME = 0)**




When FME is set (FME = 1), one SCL period ($T_{SCL}$) is equal to four clock periods of the selected I2CxCLK source. In other words, the I2CxCLK source is divided by four. Using the example from above, if the MFINTOSC (500 kHz) clock source is selected and the FME bit is set, the actual SCL frequency is 125 kHz (see Equation below).

**Equation 36-2. SCL Frequency (FME = 1)**

$$f_{SCL} = \frac{f_{I2CxCLK}}{4}$$

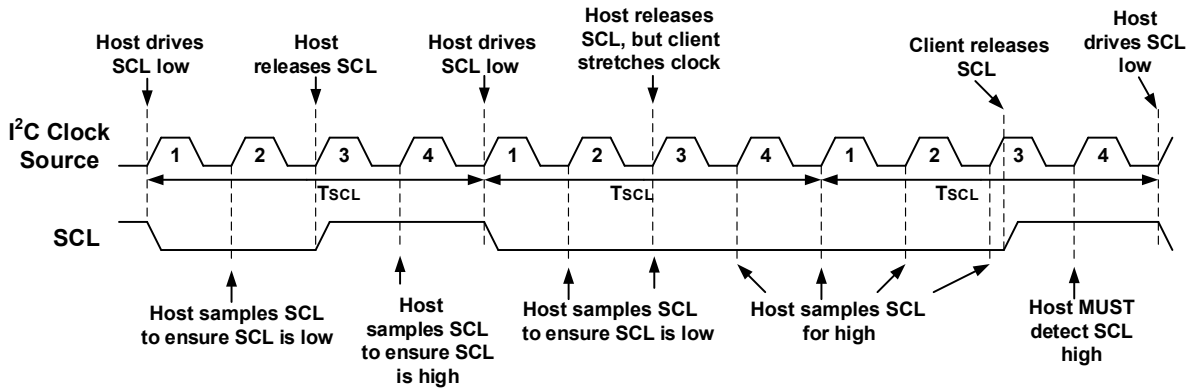
Example:

- I2CxCLK: MFINTOSC (500 kHz)
- FME: FME = 1

$$f_{SCL} = \frac{500\ kHz}{4} = 125\ kHz$$

When FME is set, host hardware uses the first I2CxCLK source period to drive SCL low (see figure below). During the second I2CxCLK period, hardware verifies that SCL is in fact low. During the third period, hardware releases SCL, allowing it to float high. Host hardware then uses the fourth I2CxCLK period to sample SCL to verify that SCL is high. If a client is holding SCL low (clock stretch) during the fourth I2CxCLK period, host hardware samples each successive I2CxCLK period until a high level is detected on SCL. Once the high level is detected, host hardware samples SCL during the next I2CxCLK period to verify that SCL is high.

**Figure 36-30. SCL Timing (FME = 1)**

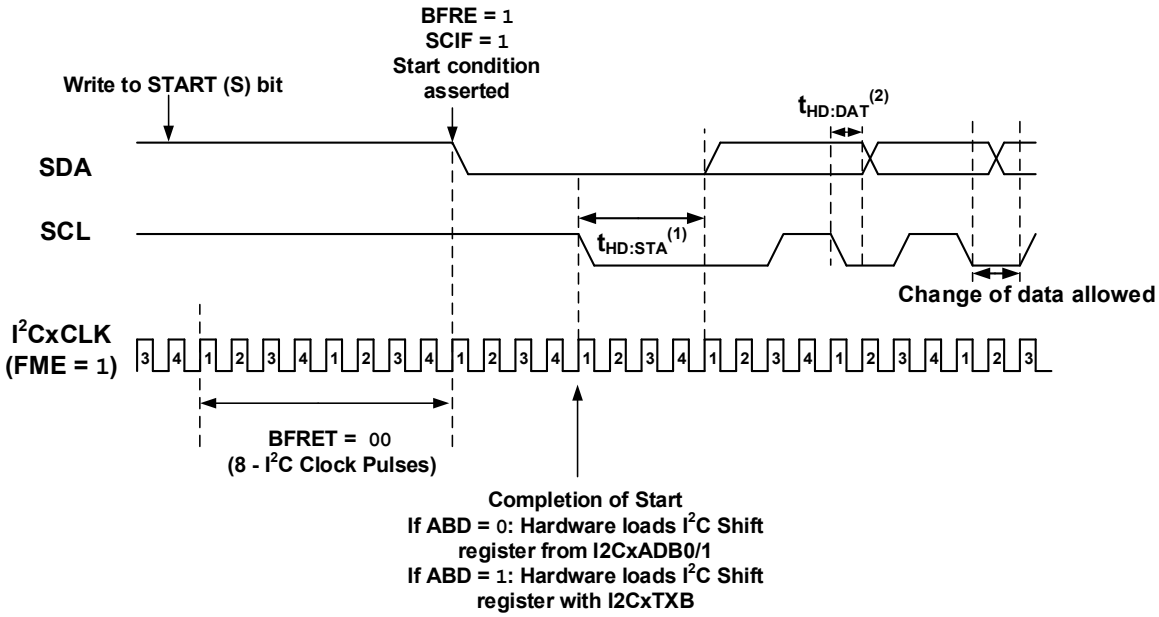



### 36.4.2.3 Start Condition Timing

A Start condition is initiated by either writing to the Start (S) bit (when ABD = 0), or by writing to I2CxTXB (when ABD = 1). When the Start condition is initiated, host hardware verifies that the bus is idle, then begins to count the number of I2CxCLK periods as determined by the Bus Free Time Status (BFRET) bits. Once the Bus Free Time period has been reached, hardware sets BFRE (BFRE = 1), the Start condition is asserted on the bus, which pulls the SDA line low, and the Start Condition Interrupt Flag (SCIF) bit is set (SCIF = 1). Host hardware then waits one full SCL period ($T_{SCL}$) before pulling the SCL line low, signaling the end of the Start condition. At this point, hardware loads the transmit shift register from either I2CxADB0/I2CxADB1 (ABD = 0) or I2CxTXB (ABD = 1).

The figure below shows an example of a Start condition.

**Figure 36-31. Start Condition Timing**



**Important:**
1. Refer to the **"Electrical Specifications"** chapter for Start condition hold time parameters.
2. SDA hold times are configured via the SDAHT bits.

### 36.4.2.4 Acknowledge Sequence Timing

As previously mentioned, the 9th SCL pulse for any transferred address/data byte is reserved for the Acknowledge ($\overline{ACK}$) sequence. During an Acknowledge sequence, the transmitting device relinquishes control of the SDA line to the receiving device. At this time, the receiving device must decide whether to pull the SDA line low ($\overline{ACK}$) or allow the line to float high (NACK).

An Acknowledge sequence is enabled automatically by module hardware following an address/data byte reception. On the 8th falling edge of SCL, the value of either the ACKDT or ACKCNT bits are copied to the SDA output, depending on the state of I2CxCNT. When I2CxCNT holds a non-zero value (I2CxCNT != 0), the value of ACKDT is copied to SDA (see figure below). When I2CxCNT reaches a zero count (I2CxCNT = 0), the value of ACKCNT is copied to SDA (see figure below). In most applications, the value of ACKDT will be zero (ACKDT = 0), which represents an $\overline{ACK}$, while the value of ACKCNT will be one (ACKCNT = 1), which represents a NACK.

**Figure 36-32. Acknowledge ($\overline{ACK}$) Sequence Timing**

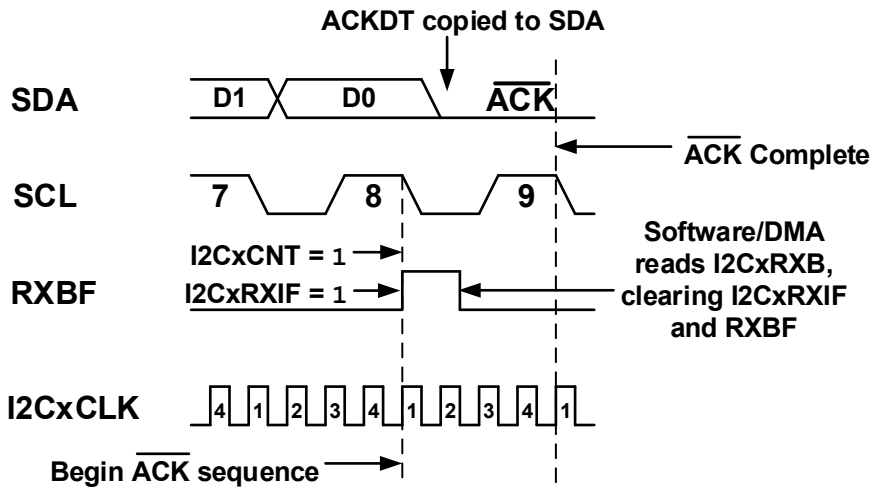
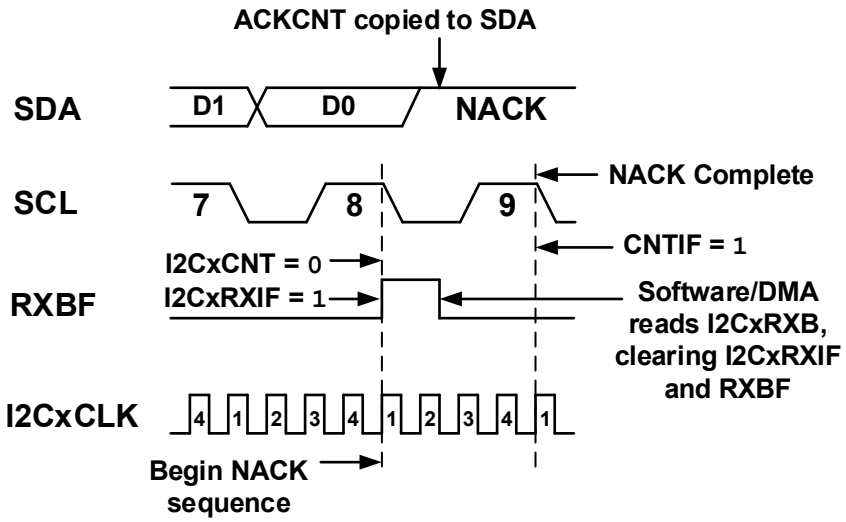


**Figure 36-33. Not Acknowledge (NACK) Sequence Timing**

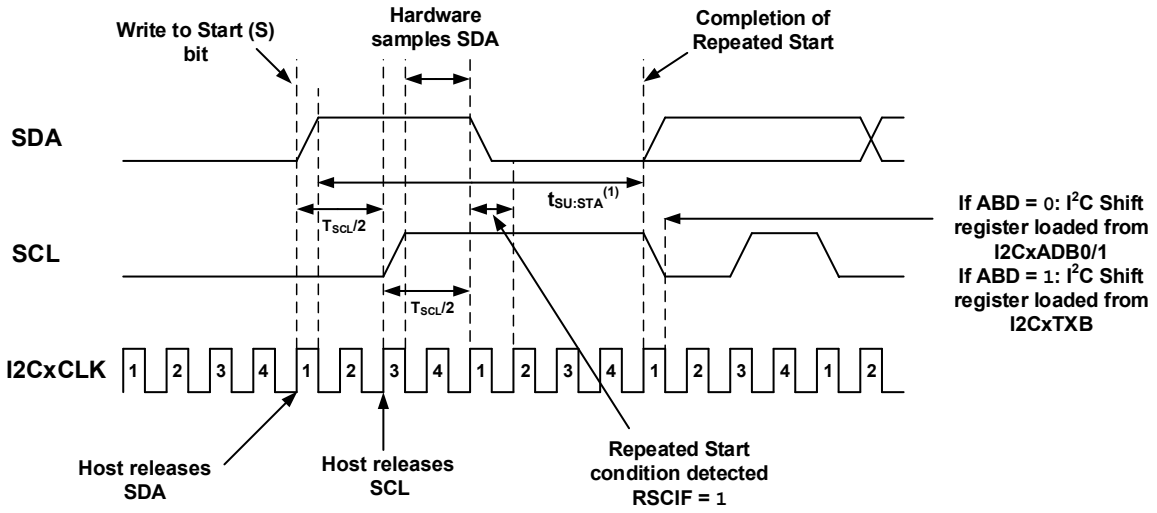



### 36.4.2.5 Restart Condition Timing

A Restart condition is identical to a Start condition. A host device may issue a Restart instead of a Stop condition if it intends to hold the bus after completing the current data transfer. A Restart condition occurs when the Restart Enable (RSEN) bit is set (RSEN = 1), either I2CxCNT is zero (I2CxCNT = 0) or ACKSTAT is set (ACKSTAT = 1), and either host hardware (ABD = 1) or user software (ABD = 0) sets the Start (S) bit.

When the Start bit is set, host hardware releases SDA (SDA floats high) for half of an SCL clock period ($T_{SCL}/2$), and then releases SCL for another half of an SCL period, then samples SDA (see figure below). If SDA is sampled low while SCL is sampled high, a bus collision has occurred. In this case, the Bus Collision Detect Interrupt Flag (BCLIF) is set, and if the Bus Collision Detect Interrupt Enable (BCLIE) bit is also set, the generic I2CxEIF is set, and the module goes Idle. If SDA is sampled high while SCL is also sampled high, host hardware issues a Start condition. Once the Restart condition is detected on the bus, the Restart Condition Interrupt Flag (RSCIF) is set by hardware. If the Restart Condition Interrupt Enable (RSCIE) bit is set, the generic I2CxIF is also set.

**Figure 36-34. Restart Condition Timing**




**Important:**
1. Refer to the **"Electrical Specifications"** chapter for Restart condition setup times.

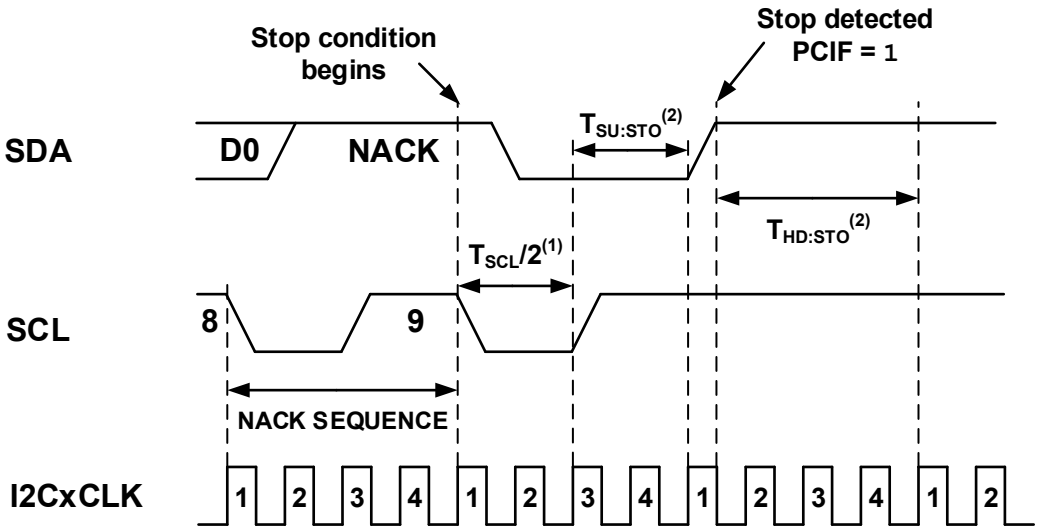
### 36.4.2.6 Stop Condition Timing

A Stop condition occurs when SDA transitions from an Active state to an Idle state while SCL is Idle. Host hardware will issue a Stop condition when it has completed its current transmission and is ready to release control of the bus. A Stop condition is also issued after an error condition occurs, such as a bus time-out, or when a NACK condition is detected on the bus.

After the $\overline{ACK}$/NACK sequence of the final byte of the transmitted/received packet, hardware pulls SCL low for half of an SCL period ($T_{SCL}/2$) (see figure below). After the half SCL period, hardware releases SCL, then samples SCL to ensure it is in an Idle state (SCL = 1). Host hardware then waits the duration of the Stop condition setup time ($T_{SU:STO}$) and releases SDA, setting the Stop Condition Interrupt Flag (PCIF). If the Stop Condition Interrupt Enable (PCIE) bit is also set, the generic I2CxIF is also set.

**Important:** At least one SCL low period must appear before a Stop condition is valid. If the SDA line transitions low, then high again, while SCL is high, the Stop condition is ignored and a Start condition will be detected by the receiver.

**Figure 36-35. Stop Condition Timing**




**Important:**
1. At least one SCL low period must appear before a Stop is valid.
2. Refer to the **"Electrical Specifications"** chapter for Stop condition setup and hold times.

### 36.4.2.7 Host Operation in 7-Bit Addressing Modes

In Host 7-bit Addressing modes, the client's 7-bit address and R/$\overline{W}$ bit value are loaded into either I2CxADB1 or I2CxTXB, depending on the Address Buffer Disable (ABD) bit setting. When the host wishes to read data from the client, software must set the R/$\overline{W}$ bit (R/$\overline{W}$ = 1). When the host wishes to write data to the client, software must clear the R/$\overline{W}$ bit (R/$\overline{W}$ = 0).

#### 36.4.2.7.1 Host Transmission (7-Bit Addressing Mode)

The following section describes the sequence of events that occur when the module is transmitting data in 7-bit Addressing mode:

1. Depending on the configuration of the Address Buffer Disable (ABD) bit, one of two methods may be used to begin communication:
   a. When ABD is clear (ABD = 0), the address buffer, I2CxADB1, is enabled. In this case, the 7-bit client address and R/$\overline{W}$ bit are loaded into I2CxADB1, with the R/$\overline{W}$ bit clear (R/$\overline{W}$ = 0). The number of data bytes are loaded into I2CxCNT and the first data byte is loaded into I2CxTXB. After these registers are loaded, software must set the Start (S) bit to begin communication. Once the S bit is set, host hardware waits for the Bus Free (BFRE) bit to be set before transmitting the Start condition to avoid bus collisions.
   b. When ABD is set (ABD = 1), the address buffer is disabled. In this case, the number of data bytes are loaded into I2CxCNT, and the client's 7-bit address and R/$\overline{W}$ bit are loaded into I2CxTXB. A write to I2CxTXB will cause host hardware to automatically issue a Start condition once the bus is idle (BFRE = 1). Software writes to the Start bit are ignored.
2. Host hardware waits for BFRE to be set, then shifts out the Start condition. Module hardware sets the Host Mode Active (MMA) bit and the Start Condition Interrupt Flag (SCIF). If the Start Condition Interrupt Enable (SCIE) bit is set, the generic I2CxIF is also set.
3. Host hardware transmits the 7-bit client address and R/$\overline{W}$ bit.

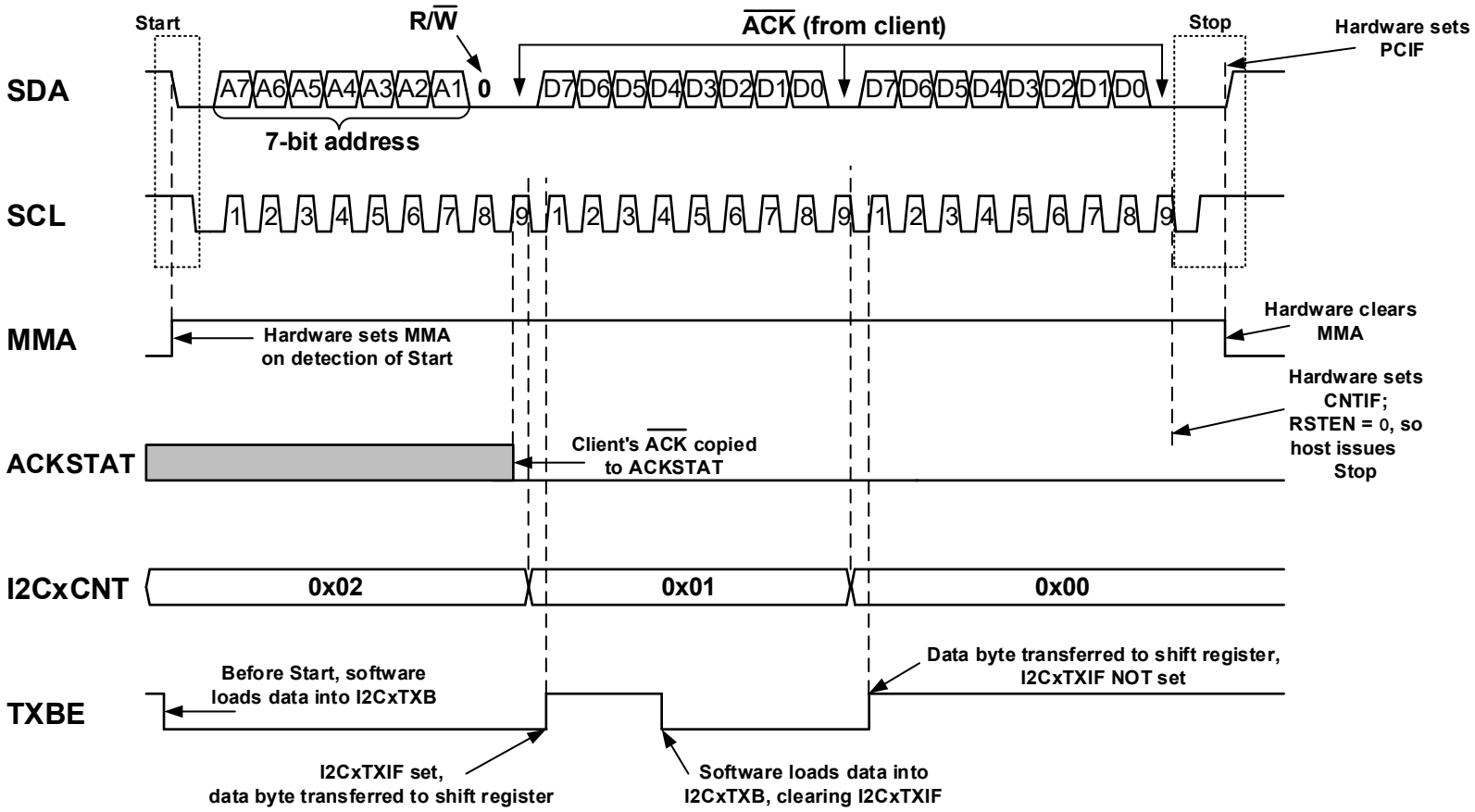
4. If upon the 8th falling edge of SCL, I2CxTXB is empty (Transmit Buffer Empty Status (TXBE) = 1), I2CxCNT is non-zero (I2CxCNT != 0), and the Clock Stretching Disable (CSD) bit is clear (CSD = 0):
   – The $I^2C$ Transmit Interrupt Flag (I2CxTXIF) is set. If the $I^2C$ Transmit Interrupt Enable (I2CxTXIE) bit is also set, the generic I2CxIF is also set.
   – The Host Data Request (MDR) bit is set, and the clock is stretched, allowing time for software to load I2CxTXB with new data. Once I2CxTXB has been written, hardware releases SCL and clears MDR.
5. Hardware transmits the 9th clock pulse and waits for an $\overline{ACK}$/NACK response from the client. If the host receives an $\overline{ACK}$, module hardware transfers the data from I2CxTXB into the transmit shift register, and I2CxCNT is decremented by one. If the host receives a NACK, hardware will attempt to issue a Stop condition. If the clock is currently being stretched by a client, the host must wait until the bus is free before issuing the Stop.
6. Host hardware checks I2CxCNT for a zero value. If I2CxCNT is zero:
   a. If ABD is clear (ABD = 0), host hardware issues a Stop condition, or sets MDR if the Restart Enable (RSEN) bit is set and waits for software to set the Start bit to issue a Restart condition. CNTIF is set.
   b. If ABD is set (ABD = 1), host hardware issues a Stop condition, or sets MDR if RSEN is set and waits for software to load I2CxTXB with a new client address. CNTIF is set.
7. Host hardware transmits the data byte.
8. If upon the 8th falling edge of SCL I2CxTXB is empty (TXBE = 1), I2CxCNT is non-zero (I2CxCNT != 0), and CSD is clear (CSD = 0):
   – I2CxTXIF is set. If the I2CxTXIE bit is also set, the generic I2CxIF is also set.
   – The MDR bit is set and the clock is stretched, allowing time for software to load I2CxTXB with new data. Once I2CxTXB has been written, hardware releases SCL and clears MDR.

   If TXBE is set (TXBE = 1) and I2CxCNT is zero (I2CxCNT = 0):

   – I2CxTXIF is NOT set.
   – CNTIF is set.
   – Host hardware issues a Stop condition, setting PCIF.
9. Repeat Steps 5 – 8 until all data has been transmitted.

© 2019-2023 Microchip Technology Inc. and its subsidiaries

**Figure 36-36. 7-Bit Host Mode Transmission**




© 2019-2023 Microchip Technology Inc. and its subsidiaries

#### 36.4.2.7.2 Host Reception (7-Bit Addressing Mode)

The following section describes the sequence of events that occur when the module is receiving data in 7-bit Addressing mode:

1. Depending on the configuration of the Address Buffer Disable (ABD) bit, one of two methods may be used to begin communication:
   a. When ABD is clear (ABD = 0), the address buffer, I2CxADB1, is enabled. In this case, the 7-bit client address and R/$\overline{W}$ bit are loaded into I2CxADB1, with the R/$\overline{W}$ bit set (R/$\overline{W}$ = 1). The number of expected received data bytes are loaded into I2CxCNT. After these registers are loaded, software must set the Start (S) bit to begin communication. Once the S bit is set, host hardware waits for the Bus Free (BFRE) bit to be set before transmitting the Start condition to avoid bus collisions.
   b. When ABD is set (ABD = 1), the address buffer is disabled. In this case, the number of expected received data bytes are loaded into I2CxCNT, and the client's 7-bit address and R/$\overline{W}$ bit are loaded into I2CxTXB. A write to I2CxTXB will cause host hardware to automatically issue a Start condition once the bus is idle (BFRE = 1). Software writes to the Start bit are ignored.
2. Host hardware waits for BFRE to be set, then shifts out the Start condition. Module hardware sets the Host Mode Active (MMA) bit and the Start Condition Interrupt Flag (SCIF). If the Start Condition Interrupt Enable (SCIE) bit is set, the generic I2CxIF is also set.
3. Host hardware transmits the 7-bit client address and R/$\overline{W}$ bit.
4. Host hardware samples SCL to determine if the client is stretching the clock, and continues to sample SCL until the line is sampled high.
5. Host hardware transmits the 9th clock pulse, and receives the $\overline{ACK}$/NACK response from the client.
   If an $\overline{ACK}$ is received, host hardware receives the first seven bits of the data byte into the receive shift register.

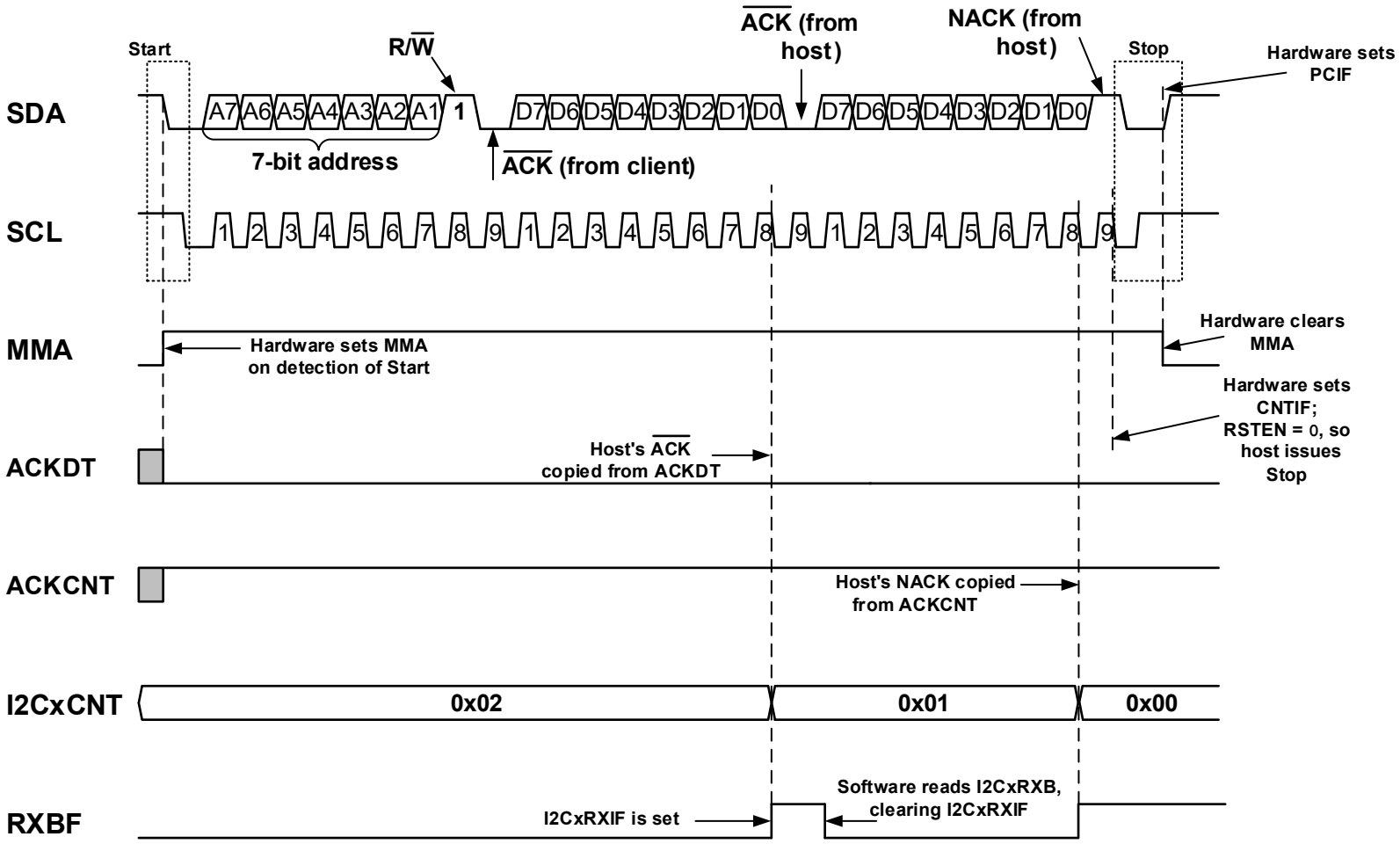
   If a NACK is received, hardware sets the NACK Detect Interrupt Flag (NACKIF), and:

   a. ABD = 0: Host generates a Stop condition, or sets the MDR bit (if RSEN is also set) and waits for software to set the Start bit to generate a Restart condition.
   b. ABD = 1: Host generates a Stop condition, or sets the MDR bit (if RSEN is also set) and waits for software to load a new address into I2CxTXB. Software writes to the Start bit are ignored.

   If the NACK Detect Interrupt Enable (NACKIE) is also set, hardware sets the generic I2CxEIF bit.
6. If previous data remains in the $I^2C$ Receive Buffer (I2CxRXB) when the first seven bits of the new byte are received into the receive shift register (RXBF = 1), the MDR bit is set (MDR = 1), and the clock is stretched after the 7th falling edge of SCL. This allows the host time to read I2CxRXB, which clears the RXBF bit, and prevents receive buffer overflows. Once RXBF is clear, hardware releases SCL.
7. The host clocks in the 8th bit of the data byte into the receive shift register, then transfers the full byte into I2CxRXB. Host hardware sets the $I^2C$ Receive Interrupt Flag (I2CxRXIF) and RXBF, and if the $I^2C$ Receive Interrupt Enable (I2CxRXIE) is set, the generic I2CxIF is also set. Finally, I2CxCNT is decremented by one.
8. Host hardware checks I2CxCNT for a zero value.
   If I2CxCNT is non-zero (I2CxCNT != 0), hardware transmits the value of the Acknowledge Data (ACKDT) bit as the acknowledgement response to the client. It is up to user software to properly configure ACKDT. In most cases, ACKDT should be clear (ACKDT = 0), which indicates an $\overline{ACK}$ response.

   If I2CxCNT is zero (I2CxCNT = 0), hardware transmits the value of the Acknowledge End of Count (ACKCNT) bit as the acknowledgement response to the client. CNTIF is set, and host hardware either issues a Stop condition or a Restart condition. It is up to user software to properly configure ACKCNT. In most cases, ACKCNT should be set (ACKCNT = 1), which indicates a NACK response. When hardware detects a NACK on the bus, it automatically issues a Stop condition. If a NACK is not detected, the Stop will not be generated, which may lead to a stalled bus condition.
9. Host hardware receives the first seven bits of the next data byte into the receive shift register.
10. Repeat Steps 6 – 9 until all expected bytes have been received.

**Figure 36-37. 7-Bit Host Mode Reception**




© 2019-2023 Microchip Technology Inc. and its subsidiaries

#### 36.4.2.8 Host Operation in 10-Bit Addressing Modes

In Host 10-bit Addressing modes, the client's 10-bit address and R/$\overline{W}$ bit value are loaded into either the I2CxADB0 and I2CxADB1 registers (when ABD = 0), or I2CxTXB (when ABD = 1). When the host intends to read data from the client, it must first transmit the full 10-bit address with the R/$\overline{W}$ bit clear (R/$\overline{W}$ = 0), issue a Restart condition, then transmit the address high byte with the R/$\overline{W}$ bit set (R/$\overline{W}$ = 1). When the host intends to write data to the client, it must transmit the full 10-bit address with the R/$\overline{W}$ bit clear (R/$\overline{W}$ = 0).

##### 36.4.2.8.1 Host Transmission (10-Bit)

The following section describes the sequence of events that occur when the module is transmitting data in 10-bit Addressing mode:

1. Depending on the configuration of the Address Buffer Disable (ABD) bit, one of two methods may be used to begin communication:
   a. When ABD is clear (ABD = 0), the address buffers, I2CxADB0 and I2CxADB1, are enabled. In this case, the address high byte is loaded into I2CxADB1 with the R/$\overline{W}$ bit clear, while the address low byte is loaded into I2CxADB0. I2CxCNT is loaded with the total number of data bytes to transmit, and the first data byte is loaded into I2CxTXB. After these registers are loaded, software must set the Start bit to begin communication.
   b. When ABD is set (ABD = 1), the address buffers are disabled. In this case, I2CxCNT must be loaded with the total number of bytes to transmit prior to loading I2CxTXB with the address high byte and R/$\overline{W}$ bit. A write to I2CxTXB forces module hardware to issue a Start condition automatically; software writes to the S bit are ignored.
2. Host hardware waits for BFRE to be set, then shifts out the Start condition. Module hardware sets the Host Mode Active (MMA) bit and the Start Condition Interrupt Flag (SCIF). If the Start Condition Interrupt Enable (SCIE) bit is also set, the generic I2CxIF is also set.
3. Host hardware transmits the address high byte and R/$\overline{W}$ bit from I2CxADB1.
4. Host hardware transmits the 9th clock pulse and shifts in the $\overline{ACK}$/NACK response from the client.
   If the host receives a NACK, it issues a Stop condition.

   If the host receives and $\overline{ACK}$ and:

   a. ABD = 0: Hardware transmits the address low byte from I2CxADB0.
   b. ABD = 1: Hardware sets I2CxTXIF and the Host Data Request (MDR) bit and waits for software to load I2CxTXB with the address low byte. Software must load I2CxTXB to resume communication.
5. If upon the 8th falling edge of SCL I2CxTXB is empty (TXBE = 1), I2CxCNT is non-zero (I2CxCNT != 0), and the Clock Stretching Disable (CSD) bit is clear (CSD = 0):
   – I2CxTXIF is set. If the $I^2C$ Transmit Interrupt Enable (I2CxTXIE) bit is also set, the generic I2CxIF is also set.
   – MDR bit is set and the clock is stretched, allowing time for software to load I2CxTXB with the address low byte. Once I2CxTXB has been written, hardware releases SCL and clears MDR.
6. Hardware transmits the 9th clock pulse and waits for an $\overline{ACK}$/NACK response from the client. If the host receives an $\overline{ACK}$, module hardware transfers the data from I2CxTXB into the transmit shift register and I2CxCNT is decremented by one. If the host receives a NACK, hardware will attempt to issue a Stop condition. If the clock is currently being stretched by a client, the host must wait until the bus is free before issuing the Stop.
7. Host hardware checks I2CxCNT for a zero value. If I2CxCNT is zero:
   a. If ABD is clear (ABD = 0), host hardware issues a Stop condition, or sets MDR if the Restart Enable (RSEN) bit is set and waits for software to set the Start bit to issue a Restart condition. CNTIF is set.
   b. If ABD is set (ABD = 1), host hardware issues a Stop condition, or sets MDR if RSEN is set and waits for software to load I2CxTXB with a new client address. CNTIF is set.
8. Host hardware transmits the data byte.
9. If upon the 8th falling edge of SCL I2CxTXB is empty (TXBE = 1), I2CxCNT is non-zero (I2CxCNT != 0), and CSD is clear (CSD = 0):
   – The I2CxTXIF bit is set. If the I2CxTXIE bit is also set, the generic I2CxIF is also set.
   – The MDR bit is set, and the clock is stretched, allowing time for software to load I2CxTXB with new data. Once I2CxTXB has been written, hardware releases SCL and clears MDR.

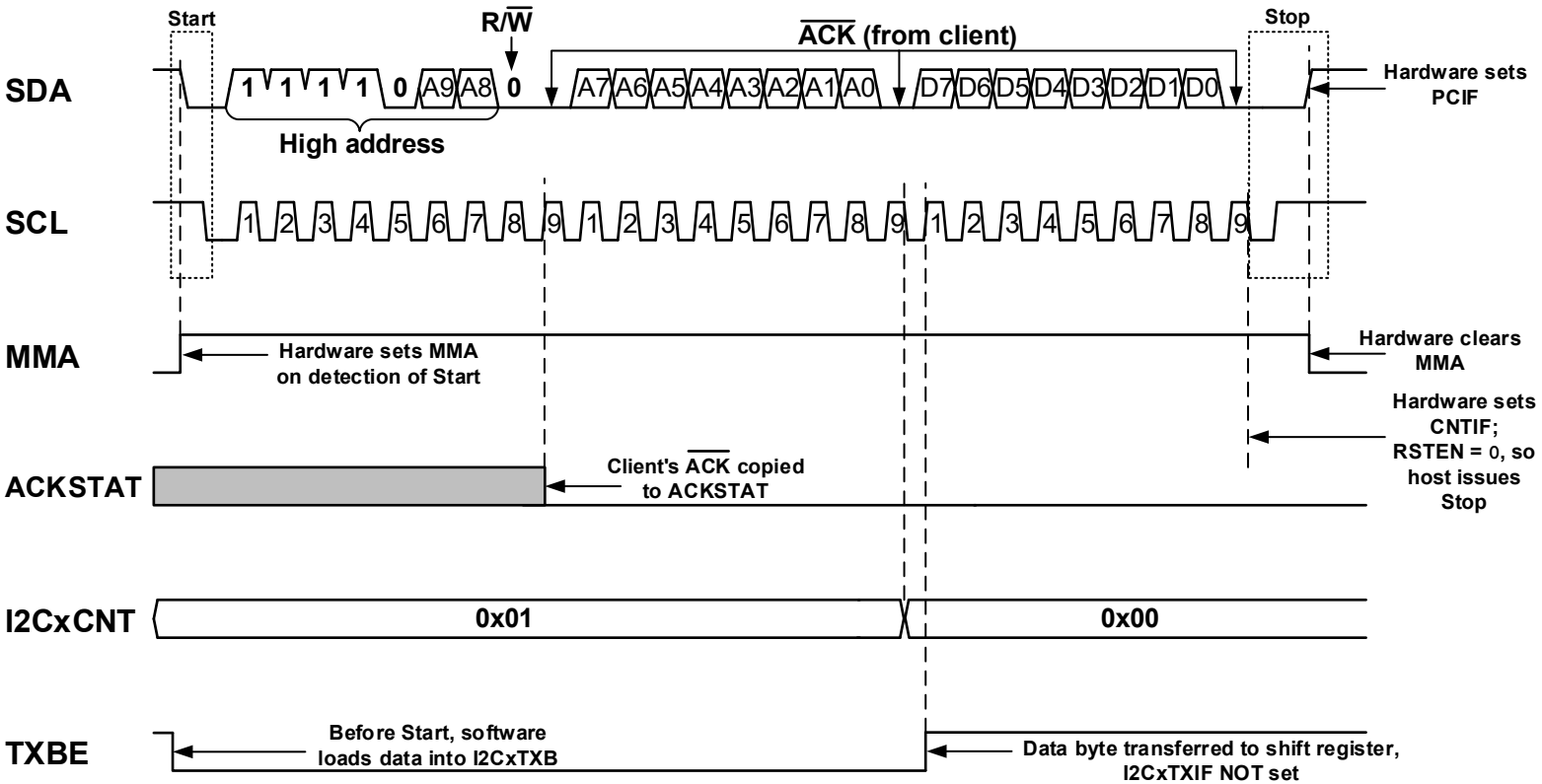
© 2019-2023 Microchip Technology Inc. and its subsidiaries

If TXBE is set (TXBE = 1) and I2CxCNT is zero (I2CxCNT = 0):

- I2CxTXIF is NOT set
- CNTIF is set
- Host hardware issues a Stop condition, setting PCIF

10. Repeat Steps 6 – 9 until all data has been transmitted.

**Figure 36-38. 10-Bit Host Mode Transmission**




© 2019-2023 Microchip Technology Inc. and its subsidiaries

#### 36.4.2.8.2 Host Reception (10-Bit Addressing Mode)

The following section describes the sequence of events that occur when the module is receiving data in 10-bit addressing mode:

1. Depending on the configuration of the Address Buffer Disable (ABD) bit, one of two methods may be used to begin communication:
   a. When ABD is clear (ABD = 0), the address buffers, I2CxADB0 and I2CxADB1, are enabled. In this case, the address high byte and R/$\overline{W}$ bit are loaded into I2CxADB1, with R/$\overline{W}$ clear (R/$\overline{W}$ = 0). The address low byte is loaded into I2CxADB0 and the Restart Enable (RSEN) bit is set by software. After these registers are loaded, software must set the Start (S) bit to begin communication. Once the S bit is set, host hardware waits for the Bus Free (BFRE) bit to be set before transmitting the Start condition to avoid bus collisions.
   b. When ABD is set (ABD = 1), the address buffers are disabled. In this case, the number of expected received bytes are loaded into I2CxCNT, the address high byte and R/$\overline{W}$ bit are loaded into I2CxTXB, with R/$\overline{W}$ clear (R/$\overline{W}$ = 0). A write to I2CxTXB will cause host hardware to automatically issue a Start condition once the bus is idle (BFRE = 1). Software writes to the Start bit are ignored.
2. Host hardware waits for BFRE to be set, then shifts out the Start condition. Module hardware sets the Host Mode Active (MMA) bit and the Start Condition Interrupt Flag (SCIF). If the Start Condition Interrupt Enable (SCIE) bit is set, the generic I2CxIF is also set.
3. Host hardware transmits the address high byte and R/$\overline{W}$ bit.
4. Host hardware samples SCL to determine if the client is stretching the clock and continues to sample SCL until the line is sampled high.
5. Host hardware transmits the 9th clock pulse and receives the $\overline{ACK}$/NACK response from the client.
   If a NACK was received, the NACK Detect Interrupt Flag (NACKIF) is set and the host immediately issues a Stop condition.

   If an $\overline{ACK}$ was received, module hardware transmits the address low byte.
6. Host hardware samples SCL to determine if the client is stretching the clock and continues to sample SCL until the line is sampled high.
7. Host hardware transmits the 9th clock pulse and receives the $\overline{ACK}$/NACK response from the client.
   If an $\overline{ACK}$ was received, hardware sets MDR and waits for hardware or software to set the Start bit.

   If a NACK is received, hardware sets NACKIF, and:

   a. ABD = 0: Host generates a Stop condition, or sets the MDR bit (if RSEN is also set) and waits for software to set the Start bit to generate a Restart condition.
   b. ABD = 1: Host generates a Stop condition, or sets the MDR bit (if RSEN is also set) and waits for software to load a new address into I2CxTXB. Software writes to the Start bit are ignored.

   If the NACK Detect Interrupt Enable (NACKIE) is also set, hardware sets the generic I2CxEIF bit.
8. Software loads I2CxCNT with the expected number of received bytes.
9. If ABD is clear (ABD = 0), software sets the Start bit. If ABD is set (ABD = 1), software writes the address high byte with R/$\overline{W}$ bit into I2CxTXB, with R/$\overline{W}$ set (R/$\overline{W}$ = 1).
10. Host hardware transmits the Restart condition, which sets the Restart Condition Interrupt Flag (RSCIF) bit. If the Restart Condition Interrupt Enable (RSCIE) bit is set, the generic I2CxIF is set by hardware.
11. Host hardware transmits the high address byte and R/$\overline{W}$ bit.
12. Host hardware samples SCL to determine if the client is stretching the clock and continues to sample SCL until the line is sampled high.
13. Host hardware transmits the 9th clock pulse and receives the $\overline{ACK}$/NACK response from the client.
    If an $\overline{ACK}$ is received, host hardware receives the first seven bits of the data byte into the receive shift register.

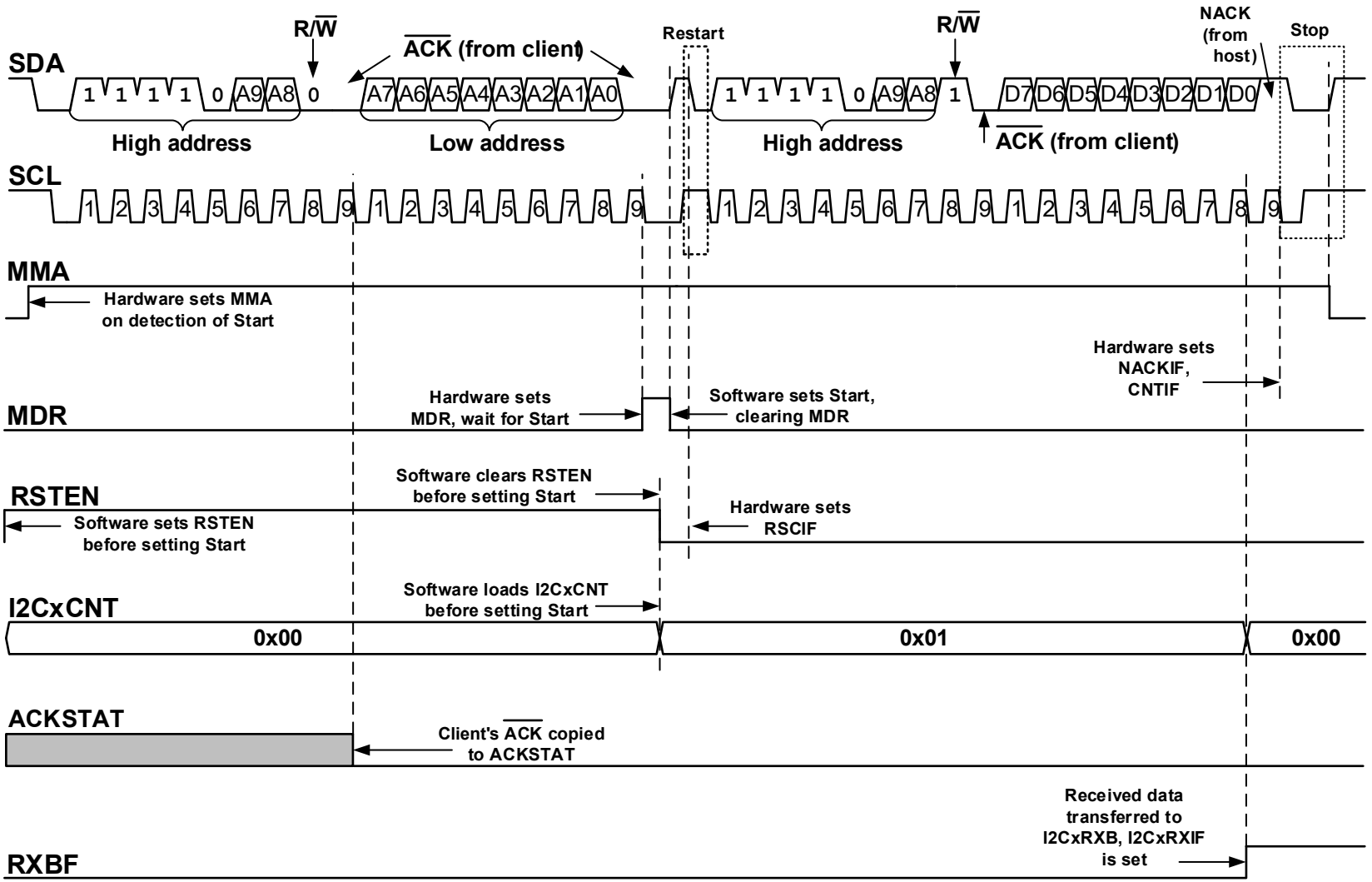
    If a NACK is received, and:

    a. ABD = 0: Host generates a Stop condition, or sets the MDR bit (if RSEN is also set) and waits for software to set the Start bit to generate a Restart condition.
    b. ABD = 1: Host generates a Stop condition, or sets the MDR bit (if RSEN is also set) and waits for software to load a new address into I2CxTXB. Software writes to the Start bit are ignored.

14. If previous data is currently in I2CxRXB (RXBF = 1) when the first seven bits are received by the receive shift register, hardware sets MDR, and the clock is stretched after the 7th falling edge of SCL. This allows software to read I2CxRXB, which clears the RXBF bit, and prevents a receive buffer overflow. Once the RXBF bit is cleared, hardware releases SCL.
15. Host hardware clocks in the 8th bit of the data byte into the receive shift register, then transfers the complete byte into I2CxRXB, which sets the I2CxRXIF and RXBF bits. If I2CxRXIE is also set, hardware sets the generic I2CxIF bit. I2CxCNT is decremented by one.
16. Hardware checks I2CxCNT for a zero value.
    If I2CxCNT is non-zero (I2CxCNT != 0), hardware transmits the value of the Acknowledge Data (ACKDT) bit as the acknowledgement response to the client. It is up to user software to properly configure ACKDT. In most cases, ACKDT will be clear (ACKDT = 0), which indicates an $\overline{\text{ACK}}$ response.

    If I2CxCNT is zero (I2CxCNT = 0), hardware transmits the value of the Acknowledge End of Count (ACKCNT) bit as the acknowledgement response to the client. CNTIF is set, and host hardware either issues a Stop condition or a Restart condition. It is up to user software to properly configure ACKCNT. In most cases, ACKCNT will be set (ACKCNT = 1), which indicates a NACK response. When hardware detects a NACK on the bus, it automatically issues a Stop condition. If a NACK is not detected, the Stop will not be generated, which may lead to a stalled bus condition.
17. Host hardware receives the first seven bits of the next data byte into the receive shift register.
18. Repeat Steps 14 – 17 until all expected bytes have been received.

**Figure 36-39. 10-Bit Host Mode Reception**




© 2019-2023 Microchip Technology Inc. and its subsidiaries

### 36.4.3 I²C Multi-Host Mode Operation

In Multi-Host mode, multiple host devices reside on the same bus. A single device, or all devices, may act as both a host and a client. Control of the bus is achieved through clock synchronization and bus arbitration.

The Bus Free (BFRE) bit is used to determine if the bus is free. When BFRE is set (BFRE = 1), the bus is in an Idle state, allowing a host device to take control of the bus.

In Multi-Host mode, the Address Interrupt and Hold Enable (ADRIE) bit must be set (ADRIE = 1) and the Clock Stretching Disable (CSD) bit must be clear (CSD = 0), in order for a host device to be addressed as a client.

When a matching address is received into the receive shift register, the SMA and the Address Interrupt Flag (ADRIF) bits are set. Since ADRIE is also set, hardware sets the Client Clock Stretching (CSTR) bit, and hardware stretches the clock to allow time for software to respond to the host device being addressed as a client. Once the address has been processed, software must clear CSTR to resume communication.

**Important:** Client hardware has priority over host hardware in Multi-Host mode. Host mode communication can only be initiated when SMA = 0.

#### 36.4.3.1 Multi-Host Mode Clock Synchronization

In a multi-host system, each host may begin to generate a clock signal as soon as the bus is idle. Clock synchronization allows all devices on the bus to use a single SCL signal.

When a high-to-low transition on SCL occurs, all active host devices begin SCL low period timing, with their clocks held low until their low hold time expires and the high state is reached. If one host's clock signal is still low, SCL will be held low until that host reaches its high state. During this time, all other host devices are held in a Wait state (see figure below).

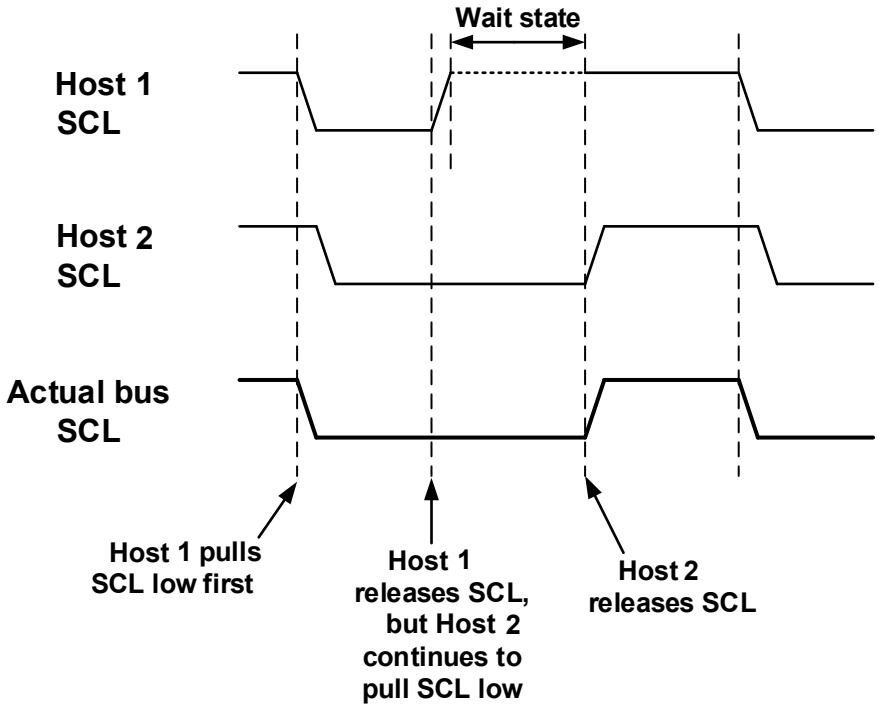
Once all hosts have counted off their low period times, SCL is released high and all host devices begin counting their high periods. The first host to complete its high period pulls the SCL line low again.

This means that when the clocks are synchronized, the SCL low period is determined by the host with the longest SCL low period, while the SCL high period is determined by the host device with the shortest SCL high period.

**Important:** The I²C Specification does not require the SCL signal to have a 50% duty cycle. In other words, one host's clock signal may have a low time that is 60% of the SCL period and a high time that is 40% of the SCL period, while another host may be 50/50. This creates a timing difference between the two clock signals, which may result in data loss.

© 2019-2023 Microchip Technology Inc. and its subsidiaries

**Figure 36-40. Clock Synchronization During Arbitration**




### 36.4.3.2 Multi-Host Mode Bus Arbitration

When the bus is idle, any host device may attempt to take control of the bus. Two or more host devices may issue a Start condition within the minimum hold time ($T_{HD:STA}$), which triggers a valid Start on the bus. The host devices must then compete using bus arbitration to determine who takes control of the bus and completes their transaction.

Bus arbitration takes place bit by bit, and it may be possible for two hosts who have identical messages to complete the entire transaction without either device losing arbitration.

During every bit period, while SCL is high each host device compares the actual signal level of SDA to the signal level the host actually transmitted. SDA sampling is performed during the SCL high period because the SDA data must be stable during this period; therefore, the first host to detect a low signal level on SDA while it expects a high signal level loses arbitration. In this case, the 'losing' host device detects a bus collision and sets the Bus Collision Detect Interrupt Flag (BCLIF), and if the Bus Collision Detect Interrupt Enable (BCLIE) bit is set, the generic I2CxEIF is also set.

Arbitration can be lost in any of the following states:

- Address transfer
- Data transfer
- Start condition
- Restart condition
- Acknowledge sequence
- Stop condition

If a collision occurs during the data transfer phase, the transmission is halted and both SCL and SDA are released by hardware. If a collision occurs during a Start, Restart, Acknowledge, or Stop, the operation is aborted and hardware releases SCL and SDA. If a collision occurs during the addressing phase, the host that 'wins' arbitration may be

attempting to address the 'losing' host as a client. In this case, the host that lost arbitration must switch to its Client mode and check to see if an address matches.

**Important:** The I²C Specification states that a bus collision cannot occur during a Start condition. If a collision occurs during a Start, BCLIF will be set during the addressing phase.

User software must clear BCLIF to resume operation.

**Figure 36-41. Bus Collision**

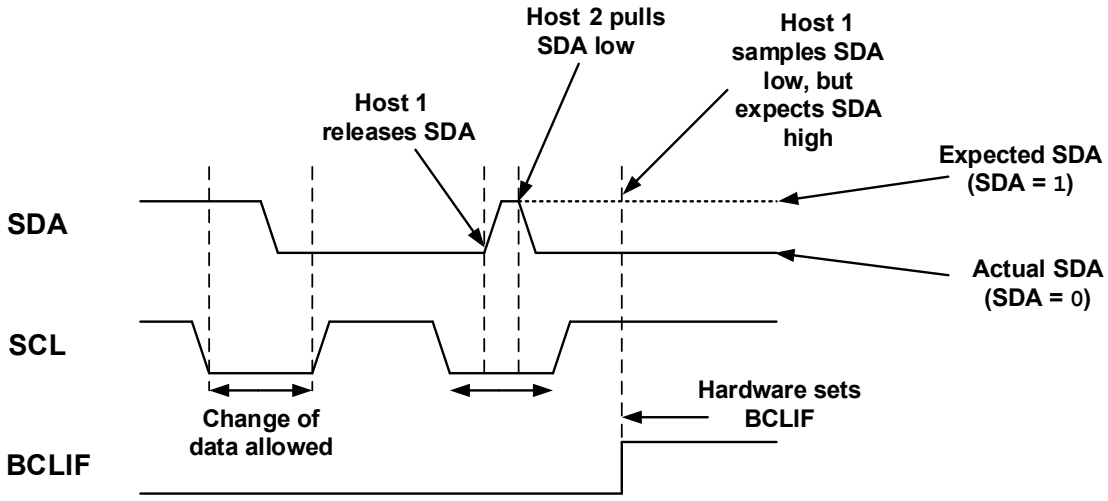
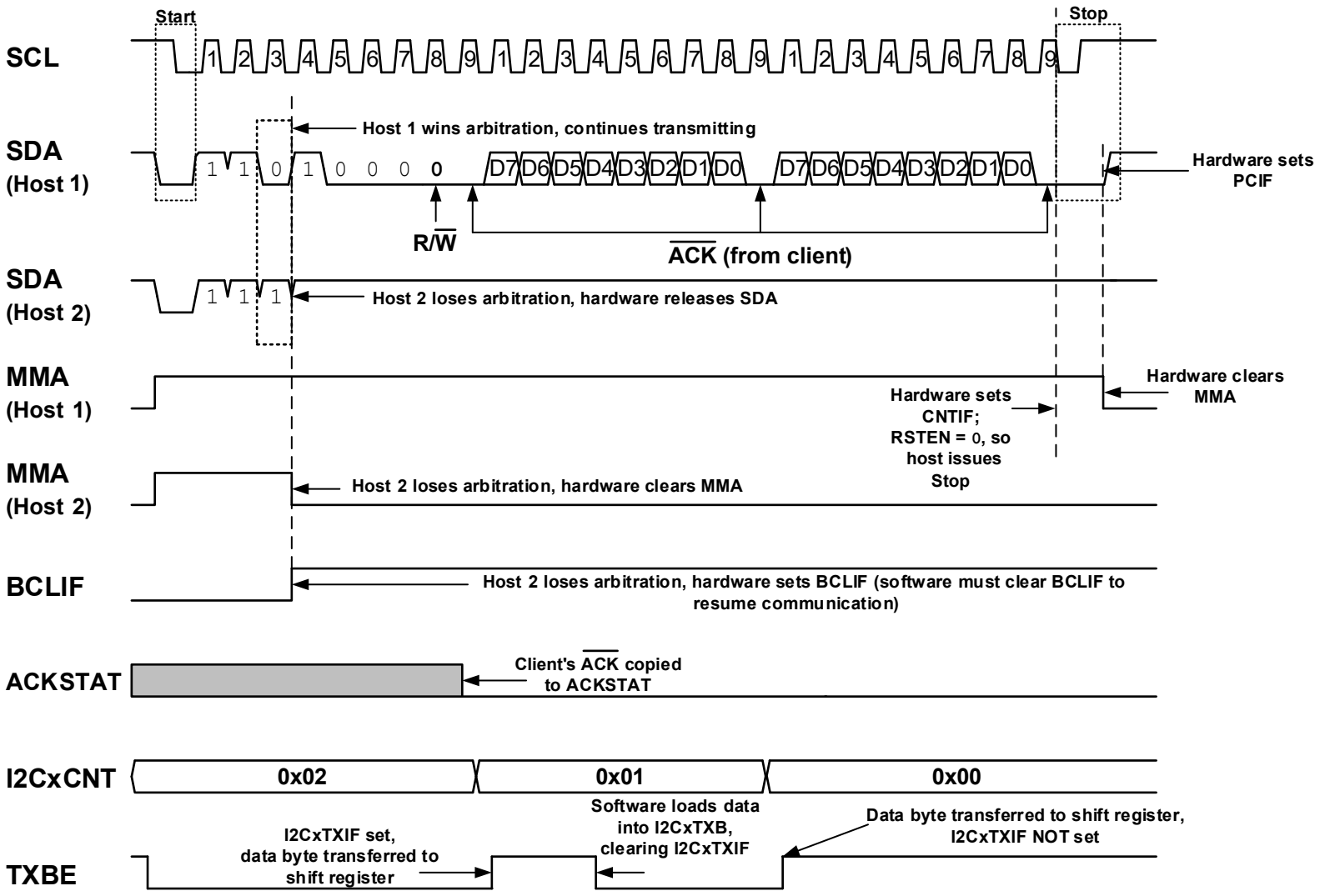


**Figure 36-42. Multi-Host Mode Transmission**




© 2019-2023 Microchip Technology Inc.
and its subsidiaries

## 36.5 Register Definitions: I²C Control

Long bit name prefixes for the I²C peripherals are shown in the following table. Refer to the **"Long Bit Names"** section in the **"Register and Bits Naming Conventions"** chapter for more information.

**Table 36-3. I²C Long Bit Name Prefixes**

| Peripheral | Bit Name Prefix |
|---|---|
| I2C1 | I2C1 |

### 36.5.1 I2CxCON0

**Name:** I2CxCON0
**Address:** 0x0294

I2C Control Register 0

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | EN | RSEN | S | CSTR | MDR | MODE[2:0] | | |
| Access | R/W | R/W | R/W/HS/HC | R/C/HS/HC | R | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bit 7 – EN** I²C Module Enable(1,2)

| Value | Description |
|---|---|
| 1 | The I²C module is enabled |
| 0 | The I²C module is disabled |

**Bit 6 – RSEN** Restart Enable (*used only when MODE* = `1xx`)

| Value | Description |
|---|---|
| 1 | Hardware sets MDR on 9th falling SCL edge (when I2CxCNT = 0 or ACKSTAT = 1) |
| 0 | Hardware issues Stop condition on 9th falling SCL edge (when I2CxCNT = 0 or ACKSTAT = 1) |

**Bit 5 – S** Host Start (*used only when MODE* = `1xx`)

| Value | Condition | Description |
|---|---|---|
| 1 | MMA = 0: | Set by write to I2CxTXB or S bit, hardware issues Start condition |
| 0 | MMA = 0: | Cleared by hardware after sending Start condition |
| 1 | MMA = 1 and MDR = 1: | Set by write to I2CxTXB or S bit, communication resumes with a Restart condition |
| 0 | MMA = 1 and MDR = 1: | Cleared by hardware after sending Restart condition |

**Bit 4 – CSTR** Client Clock Stretching(3)

| Value | Condition | Description |
|---|---|---|
| 1 | | Clock is held low (clock stretching) |
| 0 | | Enable clocking, SCL control is released |
| | SMA = 1 and RXBF = 1(6): | Set by hardware on 7th falling SCL edge<br>User must read I2CxRXB and clear CSTR to release SCL |
| | SMA = 1 and TXBE = 1 and I2CxCNT != 0: | Set by hardware on 8th falling SCL edge<br>User must write to I2CxTXB and clear CSTR to release SCL |
| | when ADRIE = 1(4): | Set by hardware on 8th falling edge of matching received address<br>User must clear CSTR to release SCL |
| | SMA = 1 and WRIE = 1: | Set by hardware on 8th falling SCL edge of received data byte<br>User must clear CSTR to release SCL |
| | SMA = 1 and ACKTIE = 1: | Set by hardware on 9th falling SCL edge<br>User must clear CSTR to release SCL |

**Bit 3 – MDR** Host Data Request (*Host pause*)

| Value | Condition | Description |
|---|---|---|
| 1 | | Host state machine pauses until data is read/written (SCL is held low) |
| 0 | | Host clocking of data is enabled |
| | MMA = 1 and RXBF = 1 (*pause for RX*): | Set by hardware on 7th falling SCL edge<br>User must read I2CxRXB to release SCL |
| | MMA = 1 and TXBE = 1 and I2CxCNT != 0 (*pause for TX*): | Set by hardware on the 8th falling SCL edge<br>User must write to I2CxTXB to release SCL |

| Value | Condition | Description |
|---|---|---|
| | RSEN = 1 and MMA = 1 and (I2CxCNT = 0 or ACKSTAT = 1) (*pause for Restart*): | Set by hardware on 9th falling SCL edge<br>User must set S bit or write to I2CxTXB to release SCL and issue a Restart condition |

**Bits 2:0 – MODE[2:0]** $I^2C$ Mode Select

| Value | Description |
|---|---|
| 111 | $I^2C$ Multi-Host mode (SMBus 2.0 Host)(5) |
| 110 | $I^2C$ Multi-Host mode (SMBus 2.0 Host)(5) |
| 101 | $I^2C$ Host mode, 10-bit address |
| 100 | $I^2C$ Host mode, 7-bit address |
| 011 | $I^2C$ Client mode, one 10-bit address with masking |
| 010 | $I^2C$ Client mode, two 10-bit addresses |
| 001 | $I^2C$ Client mode, two 7-bit addresses with masking |
| 000 | $I^2C$ Client mode, four 7-bit addresses |

**Notes:**

1. SDA and SCL pins must be configured as open-drain I/Os and use either internal or external pull-up resistors.
2. SDA and SCL signals must configure both the input and output PPS registers for each signal.
3. CSTR can be set by multiple hardware sources; all sources must be addressed by user software before the SCL line can be released.
4. SMA is set on the same SCL edge as CSTR for a matching received address.
5. In this mode, ADRIE will be set, allowing an interrupt to clear the BCLIF condition and the $\overline{ACK}$ of a matching address.
6. In 10-bit Client mode (when ABD = 1), CSTR will be set when the high address has not been read from I2CxRXB before the low address is shifted in.

© 2019-2023 Microchip Technology Inc. and its subsidiaries

### 36.5.2 I2CxCON1

**Name:** I2CxCON1
**Address:** 0x0295

I2C Control Register 1

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | ACKCNT | ACKDT | ACKSTAT | ACKT | | RXO | TXU | CSD |
| Access | R/W | R/W | R | R | | R/W/HS | R/W/HS | R/W |
| Reset | 0 | 0 | 0 | 0 | | 0 | 0 | 0 |

**Bit 7 – ACKCNT** Acknowledge End of Count(2)

| Value | Condition | Description |
|---|---|---|
| 1 | I2CxCNT = 0 | Not Acknowledge (NACK) copied to SDA output |
| 0 | I2CxCNT = 0 | Acknowledge ($\overline{ACK}$) copied to SDA output |

**Bit 6 – ACKDT** Acknowledge Data(1,2)

| Value | Condition | Description |
|---|---|---|
| 1 | Matching received address | Not Acknowledge (NACK) copied to SDA output |
| 0 | Matching received address | Acknowledge ($\overline{ACK}$) copied to SDA output |
| 1 | I2CxCNT != 0 | Not Acknowledge (NACK) copied to SDA output |
| 0 | I2CxCNT != 0 | Acknowledge ($\overline{ACK}$) copied to SDA output |

**Bit 5 – ACKSTAT** Acknowledge Status (*Transmission only*)

| Value | Description |
|---|---|
| 1 | Acknowledge was not received for the most recent transaction |
| 0 | Acknowledge was received for the most recent transaction |

**Bit 4 – ACKT** Acknowledge Time Status

| Value | Description |
|---|---|
| 1 | Indicates that the bus is in an Acknowledge sequence, set on the 8th falling SCL edge |
| 0 | Not in an Acknowledge sequence, cleared on the 9th rising SCL edge |

**Bit 2 – RXO** Receive Overflow Status (*used only when* MODE = *0xx or* MODE = *11x*)(3)

| Value | Description |
|---|---|
| 1 | Set when SMA = 1 and a host receives data when RXBF = 1 |
| 0 | No client receive Overflow condition |

**Bit 1 – TXU** Transmit Underflow Status (*used only when* MODE = *0xx or* MODE = *11x*)(3)

| Value | Description |
|---|---|
| 1 | Set when SMA = 1 and a host transmits data when TXBE = 1 |
| 0 | No client transmit Underflow condition |

**Bit 0 – CSD** Clock Stretching Disable (*used only when* MODE = *0xx or* MODE = *11x*)

| Value | Description |
|---|---|
| 1 | When SMA = 1, the CSTR bit will not be set |
| 0 | Client clock stretching proceeds normally |

**Notes:**

1. Software writes to ACKDT must be followed by a minimum SDA setup time before clearing CSTR.
2. A NACK may still be generated by hardware when bus errors are present as indicated by the I2CxSTAT1 or I2CxERR registers.
3. This bit can only be set when CSD = 1.

### 36.5.3 I2CxCON2

**Name:** I2CxCON2
**Address:** 0x0296

I2C Control Register 2

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | ACNT | GCEN | FME | ABD | SDAHT[1:0] | | BFRET[1:0] | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bit 7 – ACNT** Auto-Load I²C Count Register Enable

| Value | Description |
|---|---|
| 1 | The first transmitted/received byte after the address is automatically loaded into the I2CxCNT register |
| 0 | Auto-load of I2CxCNT is disabled |

**Bit 6 – GCEN** General Call Address Enable (*used when* MODE = *00x or* MODE = *11x*)

| Value | Description |
|---|---|
| 1 | General Call Address (0x00) causes an address match event |
| 0 | General Call Addressing is disabled |

**Bit 5 – FME** Fast Mode Enable

| Value | Description |
|---|---|
| 1 | SCL frequency ($F_{SCL}$) = $F_{I2CxCLK}$/**4** |
| 0 | SCL frequency ($F_{SCL}$) = $F_{I2CxCLK}$/**5** |

**Bit 4 – ABD** Address Buffer Disable

| Value | Description |
|---|---|
| 1 | Address buffers are disabled;<br>Received address is loaded into I2CxRXB, address to transmit is loaded into I2CxTXB |
| 0 | Address buffers are enabled;<br>Received address is loaded into I2CxADB0/I2CxADB1, address to transmit is loaded into I2CxADB0/I2CxADB1 |

**Bits 3:2 – SDAHT[1:0]** SDA Hold Time Selection

| Value | Description |
|---|---|
| 11 | **Reserved** |
| 10 | Minimum of 30 ns hold time on SDA after the falling SCL edge |
| 01 | Minimum of 100 ns hold time on SDA after the falling SCL edge |
| 00 | Minimum of 300 ns hold time on SDA after the falling SCL edge |

**Bits 1:0 – BFRET[1:0]** Bus Free Time Selection

| Value | Description |
|---|---|
| 11 | 64 I2CxCLK pulses |
| 10 | 32 I2CxCLK pulses |
| 01 | 16 I2CxCLK pulses |
| 00 | 8 I2CxCLK pulses |

### 36.5.4 I2CxSTAT0

**Name:** I2CxSTAT0
**Address:** 0x0298

I2C Status Register 0

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | BFRE | SMA | MMA | R | D | | | |
| Access | R | R | R | R | R | | | |
| Reset | 0 | 0 | 0 | 0 | 0 | | | |

**Bit 7 – BFRE** Bus Free Status(2)

| Value | Description |
|---|---|
| 1 | Indicates an idle bus; both SCL and SDA have been high for the time selected by the BFRET bits |
| 0 | Bus is not idle |

**Bit 6 – SMA** Client Mode Active Status

| Value | Description |
|---|---|
| 1 | Client mode is active<br>Set after the 8th falling SCL edge of a received matching 7-bit client address<br>Set after the 8th falling SCL edge of a matching received 10-bit client **low** address<br>Set after the 8th falling SCL edge of a received matching 10-bit client **high w/read** address, only after a previous received matching **high and low w/write** address |
| 0 | Client mode is not active<br>Cleared when any Restart/Stop condition is detected on the bus<br>Cleared by BTOIF and BCLIF conditions |

**Bit 5 – MMA** Host Mode Active Status

| Value | Description |
|---|---|
| 1 | Host mode is active<br>Set when host state machine asserts a Start condition |
| 0 | Host mode is not active<br>Cleared when BCLIF is set<br>Cleared when Stop condition is issued<br>Cleared for BTOIF condition after the host successfully shifts out a Stop condition |

**Bit 4 – R** Read Information(1)

| Value | Description |
|---|---|
| 1 | Indicates that the last matching received address was a Read request |
| 0 | Indicates that the last matching received address was a Write request |

**Bit 3 – D** Data

| Value | Description |
|---|---|
| 1 | Indicates that the last byte received or transmitted was data |
| 0 | Indicates that the last byte received or transmitted was an address |

**Notes:**

1. This bit holds the R/$\overline{W}$ bit information following the last received address match. Addresses transmitted by the host do not affect the host's R bit, and addresses appearing on the bus without a match do not affect the R bit.
2. I2CxCLK must have a valid clock source selected for this bit to function.

### 36.5.5 I2CxSTAT1

**Name:** I2CxSTAT1
**Address:** 0x0299

I2C Status Register 1

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | TXWE | | TXBE | | RXRE | CLRBF | | RXBF |
| Access | R/W/HS | | R | | R/W/HS | R/S | | R |
| Reset | 0 | | 1 | | 0 | 0 | | 0 |

**Bit 7 – TXWE** Transmit Write Error Status(1)

| Value | Description |
|---|---|
| 1 | A new byte of data was written into I2CxTXB when it was full (*must be cleared by software*) |
| 0 | No transmit write error occurred |

**Bit 5 – TXBE** Transmit Buffer Empty Status(2)

| Value | Description |
|---|---|
| 1 | I2CxTXB is empty (*cleared by writing to the I2CxTXB register*) |
| 0 | I2CxTXB is full |

**Bit 3 – RXRE** Receive Read Error Status(1)

| Value | Description |
|---|---|
| 1 | A byte of data was read from I2CxRXB when it was empty (*must be cleared by software*) |
| 0 | No receive overflow occurred |

**Bit 2 – CLRBF** Clear Buffer(3)

| Value | Description |
|---|---|
| 1 | Setting this bit clears/empties the receive and transmit buffers, causing a Reset of RXBF and TXBE<br>Setting this bit clears the I2CxRXIF and I2CxTXIF interrupt flags |

**Bit 0 – RXBF** Receive Buffer Full Status(2)

| Value | Description |
|---|---|
| 1 | I2CxRXB is full (*cleared by reading the I2CxRXB register*) |
| 0 | I2CxRXB is empty |

**Notes:**

1. This bit, when set, will cause a NACK to be issued.
2. Used as a trigger source for DMA operations.
3. This bit is special function; it can only be set by user software and always reads '0'.

### 36.5.6 I2CxPIR

**Name:** I2CxPIR
**Address:** 0x029A

I2C Interrupt Flag Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | CNTIF | ACKTIF | | WRIF | ADRIF | PCIF | RSCIF | SCIF |
| Access | R/W/HS | R/W/HS | | R/W/HS | R/W/HS | R/W/HS | R/W/HS | R/W/HS |
| Reset | 0 | 0 | | 0 | 0 | 0 | 0 | 0 |

**Bit 7 – CNTIF** Byte Count Interrupt Flag(1)

| Value | Description |
|---|---|
| 1 | Set on the 9th falling SCL edge when I2CxCNT = 0 |
| 0 | I2CxCNT value is not zero |

**Bit 6 – ACKTIF** Acknowledge Status Time Interrupt Flag (*used only when* MODE = *0xx or* MODE = *11x*)(1,2)

| Value | Description |
|---|---|
| 1 | Acknowledge sequence detected, set on the 9th falling SCL edge for any byte when addressed as a client |
| 0 | Acknowledge sequence not detected |

**Bit 4 – WRIF** Data Write Interrupt Flag (*used only when* MODE = *0xx or* MODE = *11x*)(1)

| Value | Description |
|---|---|
| 1 | Data byte detected, set on the 8th falling SCL edge for a received data byte |
| 0 | Data byte not detected |

**Bit 3 – ADRIF** Address Interrupt Flag (*used only when* MODE = *0xx or* MODE = *11x*)(1)

| Value | Description |
|---|---|
| 1 | Address detected, set on the 8th falling SCL edge for a matching received address byte |
| 0 | Address not detected |

**Bit 2 – PCIF** Stop Condition Interrupt Flag(1)

| Value | Description |
|---|---|
| 1 | Stop condition detected |
| 0 | Stop condition not detected |

**Bit 1 – RSCIF** Restart Condition Interrupt Flag(1)

| Value | Description |
|---|---|
| 1 | Restart condition detected |
| 0 | Restart condition not detected |

**Bit 0 – SCIF** Start Condition Interrupt Flag(1)

| Value | Description |
|---|---|
| 1 | Start condition detected |
| 0 | Start condition not detected |

**Notes:**

1. Enabled interrupt flags are OR'ed to produce the PIRx[I2CxIF] bit.
2. ACKTIF is not set by a matching 10-bit high address byte with the R/$\overline{W}$ bit clear. It is only set after the matching low address byte is shifted in.

### 36.5.7 I2CxPIE

**Name:** I2CxPIE
**Address:** 0x029B

I2C Interrupt and Hold Enable Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | CNTIE | ACKTIE | | WRIE | ADRIE | PCIE | RSCIE | SCIE |
| Access | R/W | R/W | | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | | 0 | 0 | 0 | 0 | 0 |

**Bit 7 – CNTIE** Byte Count Interrupt Enable(1)

| Value | Description |
|---|---|
| 1 | Enables Byte Count interrupts |
| 0 | Disables Byte Count interrupts |

**Bit 6 – ACKTIE** Acknowledge Status Time Interrupt and Hold Enable(1,2)

| Value | Description |
|---|---|
| 1 | Enables Acknowledge Status Time Interrupt and Hold condition |
| 0 | Disables Acknowledge Status Time Interrupt and Hold condition |

**Bit 4 – WRIE** Data Write Interrupt and Hold Enable(1,3)

| Value | Description |
|---|---|
| 1 | Enables Data Write Interrupt and Hold condition |
| 0 | Disables Data Write Interrupt and Hold condition |

**Bit 3 – ADRIE** Address Interrupt and Hold Enable(1,4)

| Value | Description |
|---|---|
| 1 | Enables Address Interrupt and Hold condition |
| 0 | Disables Address Interrupt and Hold condition |

**Bit 2 – PCIE** Stop Condition Interrupt Enable(1)

| Value | Description |
|---|---|
| 1 | Enables interrupt on the detection of a Stop condition |
| 0 | Disables interrupt on the detection of a Stop condition |

**Bit 1 – RSCIE** Restart Condition Interrupt Enable(1)

| Value | Description |
|---|---|
| 1 | Enables interrupt on the detection of a Restart condition |
| 0 | Disables interrupt on the detection of a Restart condition |

**Bit 0 – SCIE** Stop Condition Interrupt Enable(1)

| Value | Description |
|---|---|
| 1 | Enables interrupt on the detection of a Start condition |
| 0 | Disables interrupt on the detection of a Start condition |

**Notes:**

1. Enabled interrupt flags are OR'ed to produce the PIRx[I2CxIF] bit.
2. When ACKTIE is set (ACKTIE = 1) and ACKTIF becomes set (ACKTIF = 1), if an $\overline{\text{ACK}}$ is generated, CSTR is also set. If a NACK is generated, CSTR remains unchanged.
3. When WRIE is set (WRIE = 1) and WRIF becomes set (WRIF = 1), CSTR is also set.
4. When ADRIE is set (ADRIE = 1) and ADRIF becomes set (ADRIF = 1), CSTR is also set.

### 36.5.8 I2CxERR

**Name:** I2CxERR
**Address:** 0x0297

I2C Error Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | BTOIF | BCLIF | NACKIF | | BTOIE | BLCIE | NACKIE |
| Access | | R/W/HS | R/W/HS | R/W/HS | | R/W | R/W | R/W |
| Reset | | 0 | 0 | 0 | | 0 | 0 | 0 |

**Bit 6 – BTOIF** Bus Time-out Interrupt Flag(1,2)

| Value | Description |
|---|---|
| 1 | Bus time-out event occurred |
| 0 | No bus time-out event occurred |

**Bit 5 – BCLIF** Bus Collision Detect Interrupt Flag(1)

| Value | Description |
|---|---|
| 1 | Bus collision detected |
| 0 | No bus collision occurred |

**Bit 4 – NACKIF** NACK Detect Interrupt Flag(1,3,4)

| Value | Description |
|---|---|
| 1 | NACK detected on the bus (when SMA = 1 or MMA = 1) |
| 0 | No NACK detected on the bus |

**Bit 2 – BTOIE** Bus Time-out Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enable Bus Time-out interrupts |
| 0 | Disable Bus Time-out interrupts |

**Bit 1 – BLCIE** Bus Collision Detect Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enable Bus Collision interrupts |
| 0 | Disable Bus Collision interrupts |

**Bit 0 – NACKIE** NACK Detect Interrupt Enable

| Value | Description |
|---|---|
| 1 | Enable NACK detect interrupts |
| 0 | Disable NACK detect interrupts |

**Notes:**

1. Enabled error interrupt flags are OR'ed to produce the PIRx[I2CxEIF] bit.
2. User software must select the Bus Time-out source in the I2CxBTO register.
3. NACKIF is also set when any of the TXWE, RXRE, TXU, or RXO bits are set.
4. NACKIF is not set for the NACK response to a non-matching client address.

### 36.5.9 I2CxCLK

**Name:** I2CxCLK
**Address:** 0x029C

I2C Clock Selection Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | CLK[4:0] | | | | |
| Access | | | | R/W | R/W | R/W | R/W | R/W |
| Reset | | | | 0 | 0 | 0 | 0 | 0 |

**Bits 4:0 – CLK[4:0]** $I^2C$ Clock Selection

**Table 36-4.**

| CLK | Selection |
|---|---|
| 11111-11000 | Reserved |
| 10111 | CLC8_OUT |
| 10110 | CLC7_OUT |
| 10101 | CLC6_OUT |
| 10100 | CLC5_OUT |
| 10011 | CLC4_OUT |
| 10010 | CLC3_OUT |
| 10001 | CLC2_OUT |
| 10000 | CLC1_OUT |
| 01111 | SMT1_OUT |
| 01110-01010 | Reserved |
| 01001 | TMR6_Postscaler_OUT |
| 01000 | TMR4_Postscaler_OUT |
| 00111 | TMR2_Postscaler_OUT |
| 00110 | TMR0_OUT |
| 00101 | EXTOSC |
| 00100 | Clock Reference Output |
| 00011 | MFINTOSC (500 kHz) |
| 00010 | HFINTOSC |
| 00001 | $F_{OSC}$ (System Clock) |
| 00000 | $F_{OSC}/4$ |

© 2019-2023 Microchip Technology Inc. and its subsidiaries

### 36.5.10 I2CxCNT

**Name:** I2CxCNT
**Address:** 0x028D

I2C Byte Count Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | CNT[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | x | x | x | x | x | x | x | x |

**Bits 7:0 – CNT[7:0]** I$^2$C Byte Count Register(1,2)
Reset States: POR/BOR = xxxxxxxx
All Other Resets = uuuuuuuu

| Condition | Description |
|---|---|
| **If receiving data:** | Count value decremented on 8th falling SCL edge when a new byte is loaded into I2CxRXB |
| **If transmitting data:** | Count value is decremented on the 9th falling SCL edge when a new byte is moved from I2CxTXB |

**Notes:**

1. It is recommended to write this register only when the module is idle (MMA = 0 or SMA = 0), or when the module is clock stretching (CSTR =1 or MDR = 1).
2. CNTIF is set on the 9th falling SCL edge when I2CxCNT = 0.

### 36.5.11 I2CxBTO

**Name:** I2CxBTO
**Address:** 0x029D

I2C Bus Time-out Selection Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | BTO[3:0] | | | |
| Access | | | | | R/W | R/W | R/W | R/W |
| Reset | | | | | 0 | 0 | 0 | 0 |

**Bits 3:0 – BTO[3:0]** $I^2C$ Bus Time-out Selection

**Table 36-5.**

| BTO | Selection |
|---|---|
| 1111 | Reserved |
| 1110 | CLC8_OUT |
| 1101 | CLC7_OUT |
| 1100 | CLC6_OUT |
| 1011 | CLC5_OUT |
| 1010 | CLC4_OUT |
| 1001 | CLC3_OUT |
| 1000 | CLC2_OUT |
| 0111 | CLC1_OUT |
| 0110-0100 | Reserved |
| 0011 | TMR6_Postscaler_OUT |
| 0010 | TMR4_Postscaler_OUT |
| 0001 | TMR2_Postscaler_OUT |
| 0000 | Reserved |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

### 36.5.12 I2CxADB0

**Name:** I2CxADB0
**Address:** 0x028E

I2C Address Buffer 0 Register(1)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | ADB[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | x | x | x | x | x | x | x | x |

**Bits 7:0 – ADB[7:0]** $I^2C$ Address Buffer 0
Reset States: POR/BOR = xxxxxxxx
All Other Resets = uuuuuuuu

| Condition | Description |
|---|---|
| 7-bit Client/Multi-Host modes *(*MODE = *00x or 11x)*: | **ADB[7:1]:** Received matching 7-bit client address<br>**ADB[0]:** Received R/$\overline{W}$ value from 7-bit address |
| 10-bit Client modes *(*MODE = *01x)*: | **ADB[7:0]:** Received matching lower eight bits of 10-bit client address |
| 7-bit Host mode *(*MODE = *100)*: | Unused in this mode |
| 10-bit Host mode *(*MODE = *101)*: | **ADB[7:0]:** Eight Least Significant bits of the 10-bit client address |

**Note:**

1. This register is read-only except in Host 10-bit Address mode *(*MODE = *101)*.

### 36.5.13 I2CxADB1

**Name:** I2CxADB1
**Address:** 0x028F

I2C Address Buffer 1 Register(1)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | ADB[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | x | x | x | x | x | x | x | x |

**Bits 7:0 – ADB[7:0]** $I^2C$ Address Buffer 1
Reset States: POR/BOR = xxxxxxxx
All Other Resets = uuuuuuuu

| Condition | Description |
|---|---|
| 7-bit Client modes *(*MODE = *00x)*: | Unused in this mode |
| 10-bit Client modes *(*MODE = *01x)*: | **ADB[7:1]:** Received matching 10-bit client address high byte<br>**ADB[0]:** Received R/$\overline{W}$ value from 10-bit high address byte |
| 7-bit Host mode *(*MODE = *100)*: | **ADB[7:1]:** 7-bit client address<br>**ADB[0]:** R/$\overline{W}$ value |
| 10-bit Host mode *(*MODE = *101)*: | **ADB[7:1]:** 10-bit client high address byte<br>**ADB[0]:** R/$\overline{W}$ value |
| 7-bit Multi-Host modes *(*MODE = *11x)*: | **ADB[7:1]:** 7-bit client address<br>**ADB[0]:** R/$\overline{W}$ value |

**Note:**

1. This register is read-only in 7-bit Client Address modes *(*MODE = *0xx)*.

### 36.5.14 I2CxADR0

**Name:** I2CxADR0
**Address:** 0x0290

I2C Address 0 Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | ADR[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Bits 7:0 – ADR[7:0]** $I^2C$ Client Address 0

| Condition | Description |
|---|---|
| 7-bit Client/Multi-Host modes *(*MODE = *00x or 11x)*: | **ADR[7:1]:** 7-bit client address<br>**ADR[0]:** Unused; bit state is 'don't care' |
| 10-bit Client modes *(*MODE = *01x)*: | **ADR[7:0]:** Eight Least Significant bits of first 10-bit address |

### 36.5.15 I2CxADR1

**Name:** I2CxADR1
**Address:** 0x0291

I2C Address 1 Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | ADR[6:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | |

**Bits 7:1 – ADR[6:0]** I²C Client Address 1

| Condition | Description |
|---|---|
| 7-bit Client/Multi-Host modes *(*MODE = *000* or *110)*: | 7-bit client address 1 |
| 7-bit Client/Multi-Host modes with Masking *(*MODE = *011* or *111)*: | 7-bit Client address mask for I2CxADR0 |
| 10-bit Client mode *(*MODE = *010)*: | **ADR[7:3]:** Bit pattern (11110) as defined by the I²C Specification(1)<br>**ADR[2:1]:** Two Most Significant bits of first 10-bit address |
| 10-bit Client mode with Masking *(*MODE = *011)*: | **ADR[7:3]:** Bit pattern (11110) as defined by the I²C Specification(1)<br>**ADR[2:1]:** Two Most Significant bits of 10-bit address |

**Note:**

1. The '11110' bit pattern used in the 10-bit address high byte is defined by the I²C Specification. It is up to the user to define these bits. These bit values are compared to the received address by hardware to determine a match. The bit pattern transmitted by the host must be the same as the client address's bit pattern used for comparison or a match will not occur.

### 36.5.16 I2CxADR2

**Name:** I2CxADR2
**Address:** 0x0292

I2C Address 2 Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | ADR[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

**Bits 7:0 – ADR[7:0]** $I^2C$ Client Address 2

| Condition | Description |
|---|---|
| 7-bit Client/Multi-Host modes *(*MODE = *000* or *110)*: | **ADR[7:1]:** 7-bit client address 2<br>**ADR[0]:** Unused; bit state is 'don't care' |
| 7-bit Client/Multi-Host modes with Masking *(*MODE = *001* or *111)*: | **ADR[7:1]:** 7-bit client address<br>**ADR[0]:** Unused; bit state is 'don't care' |
| 10-bit Client mode *(*MODE = *010)*: | **ADR[7:0]:** Eight Least Significant bits of the second 10-bit address |
| 10-bit Client mode with Masking *(*MODE = *011)*: | **ADR[7:0]:** Eight Least Significant bits of 10-bit address mask |

### 36.5.17 I2CxADR3

**Name:** I2CxADR3
**Address:** 0x0293

I2C Address 3 Register(1)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | ADR[6:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | |
| Reset | 1 | 1 | 1 | 1 | 1 | 1 | 1 | |

**Bits 7:1 – ADR[6:0]** I²C Client Address 3

| Name | Description |
|---|---|
| 7-bit Client/Multi-Host modes *(*MODE = *000* or *110)*: | 7-bit client address 3 |
| 7-bit Client/Multi-Host modes with Masking *(*MODE = *001* or *111)*: | 7-bit client address mask for I2CxADR2 |
| 10-bit Client mode *(*MODE = *010)*: | **ADR[7:3]:** Bit pattern (11110) as defined by the I²C Specification(1)<br>**ADR[2:1]:** Two Most Significant bits of second 10-bit address |
| 10-bit Client mode with Masking *(*MODE = *011)*: | **ADR[7:3]:** Bit pattern (11110) as defined by the I²C Specification(1)<br>**ADR[2:1]:** Two Most Significant bits of 10-bit address mask |

**Note:**

1. The '11110' bit pattern used in the 10-bit address high byte is defined by the I²C Specification. It is up to the user to define these bits. These bit values are compared to the received address by hardware to determine a match. The bit pattern transmitted by the host must be the same as the client address's bit pattern used for comparison or a match will not occur.

### 36.5.18 I2CxTXB

**Name:** I2CxTXB
**Address:** 0x028C

I2C Transmit Buffer Register(1)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | TXB[7:0] | | | | | | | |
| Access | W | W | W | W | W | W | W | W |
| Reset | x | x | x | x | x | x | x | x |

**Bits 7:0 – TXB[7:0]** $I^2C$ Transmit Buffer

**Note:** This register is write-only. Reading this register will return a value of `0x00`.

### 36.5.19 I2CxRXB

**Name:** I2CxRXB
**Address:** 0x028B

I2C Receive Buffer(1)

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | RXB[7:0] | | | | | | | |
| Access | R | R | R | R | R | R | R | R |
| Reset | x | x | x | x | x | x | x | x |

**Bits 7:0 – RXB[7:0]** $I^2C$ Receive Buffer

**Note:** This register is read-only. Writes to this register are ignored.

## 36.6 Register Summary - I2C

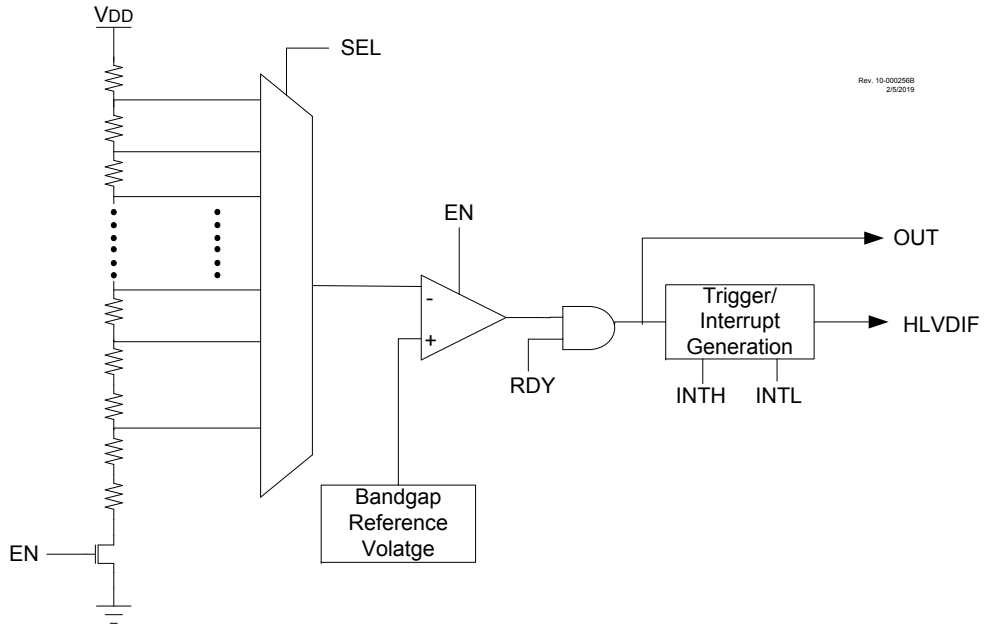
| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x028B | I2C1RXB | 7:0 | RXB[7:0] | | | | | | | |
| 0x028C | I2C1TXB | 7:0 | TXB[7:0] | | | | | | | |
| 0x028D | I2C1CNT | 7:0 | CNT[7:0] | | | | | | | |
| 0x028E | I2C1ADB0 | 7:0 | ADB[7:0] | | | | | | | |
| 0x028F | I2C1ADB1 | 7:0 | ADB[7:0] | | | | | | | |
| 0x0290 | I2C1ADR0 | 7:0 | ADR[7:0] | | | | | | | |
| 0x0291 | I2C1ADR1 | 7:0 | ADR[6:0] | | | | | | | |
| 0x0292 | I2C1ADR2 | 7:0 | ADR[7:0] | | | | | | | |
| 0x0293 | I2C1ADR3 | 7:0 | ADR[6:0] | | | | | | | |
| 0x0294 | I2C1CON0 | 7:0 | EN | RSEN | S | CSTR | MDR | MODE[2:0] | | |
| 0x0295 | I2C1CON1 | 7:0 | ACKCNT | ACKDT | ACKSTAT | ACKT | | RXO | TXU | CSD |
| 0x0296 | I2C1CON2 | 7:0 | ACNT | GCEN | FME | ABD | SDAHT[1:0] | | BFRET[1:0] | |
| 0x0297 | I2C1ERR | 7:0 | | BTOIF | BCLIF | NACKIF | | BTOIE | BLCIE | NACKIE |
| 0x0298 | I2C1STAT0 | 7:0 | BFRE | SMA | MMA | R | D | | | |
| 0x0299 | I2C1STAT1 | 7:0 | TXWE | | TXBE | | RXRE | CLRBF | | RXBF |
| 0x029A | I2C1PIR | 7:0 | CNTIF | ACKTIF | | WRIF | ADRIF | PCIF | RSCIF | SCIF |
| 0x029B | I2C1PIE | 7:0 | CNTIE | ACKTIE | | WRIE | ADRIE | PCIE | RSCIE | SCIE |
| 0x029C | I2C1CLK | 7:0 | | | | CLK[4:0] | | | | |
| 0x029D | I2C1BTO | 7:0 | | | | | BTO[3:0] | | | |

# 37. HLVD - High/Low-Voltage Detect

The HLVD module can be configured to monitor the device voltage. This is useful in battery monitoring applications. Complete control of the HLVD module is provided through the HLVDCON0 and HLVDCON1 registers.

Refer below for a simplified block diagram of the HLVD module.

**Figure 37-1. HLVD Module Block Diagram**




Since the HLVD can be software enabled through the EN bit, setting and clearing the enable bit does not produce a false HLVD event glitch. Each time the HLVD module is enabled, the RDY bit can be used to detect when the module is stable and ready to use.

The INTH and INTL bits determine the overall operation of the module. When INTH is set, the module monitors for rises in $V_{DD}$ above the trip point set by the bits. When INTL is set, the module monitors for drops in $V_{DD}$ below the trip point set by the SEL bits. When both the INTH and INTL bits are set, any changes above or below the trip point set by the SEL bits can be monitored.

The OUT bit can be read to determine if the voltage is greater than or less than the selected trip point.

## 37.1 Operation

When the HLVD module is enabled, a comparator uses an internally generated voltage reference as the set point. The set point is compared with the trip point, where each node in the resistor divider represents a trip point voltage. The “trip point” voltage is the voltage level at which the device detects a high or low-voltage event, depending on the configuration of the module.

When the supply voltage is equal to the trip point, the voltage tapped off of the resistor array is equal to the internal reference voltage generated by the voltage reference module. The comparator then generates an interrupt signal by setting the HLVDIF bit.

The trip point voltage is software programmable using the SEL bits.

## 37.2 Setup

To set up the HLVD module:

1. Select the desired HLVD trip point by writing the value to the SEL bits.
2. Depending on the application to detect high-voltage peaks or low-voltage drops or both, set the INTH or INTL bit appropriately.
3. Enable the HLVD module by setting the EN bit.
4. Clear the HLVD Interrupt Flag (HLVDIF), which may have been set from a previous interrupt.
5. If interrupts are desired, enable the HLVD interrupt by setting the HLVDIE and GIE bits.
   An interrupt will not be generated until the RDY bit is set.

**Important:** Before changing any module settings (interrupts and tripping point), first disable the module (EN = 0), make the changes and re-enable the module. This prevents the generation of false HLVD events.

## 37.3 Current Consumption

When the module is enabled, the HLVD comparator and voltage divider are enabled and consume static current. The total current consumption, when enabled, is specified in the **"Electrical Specifications"** chapter.

Depending on the application, the HLVD module does not need to operate constantly. To reduce the current consumption, the module can disabled when not in use. Refer to the **"PMD - Peripheral Module Disable"** chapter for more details.

## 37.4 HLVD Start-Up Time

If the HLVD or other circuits using the internal voltage reference are disabled to lower the device's current consumption, the reference voltage circuit will require time to become stable before a Low or High Voltage condition can be reliably detected. This start-up time, $T_{FVRST}$, is an interval that is independent of device clock speed. It is specified in the **"Electrical Specifications"** chapter of the device specific data sheet.

The HLVD interrupt flag is not enabled until $T_{FVRST}$ has expired and a stable reference voltage is reached. For this reason, brief excursions beyond the set point may not be detected during this interval (see the figures below).

**Figure 37-2. Low-Voltage Detect Operation (INTL = 1)**

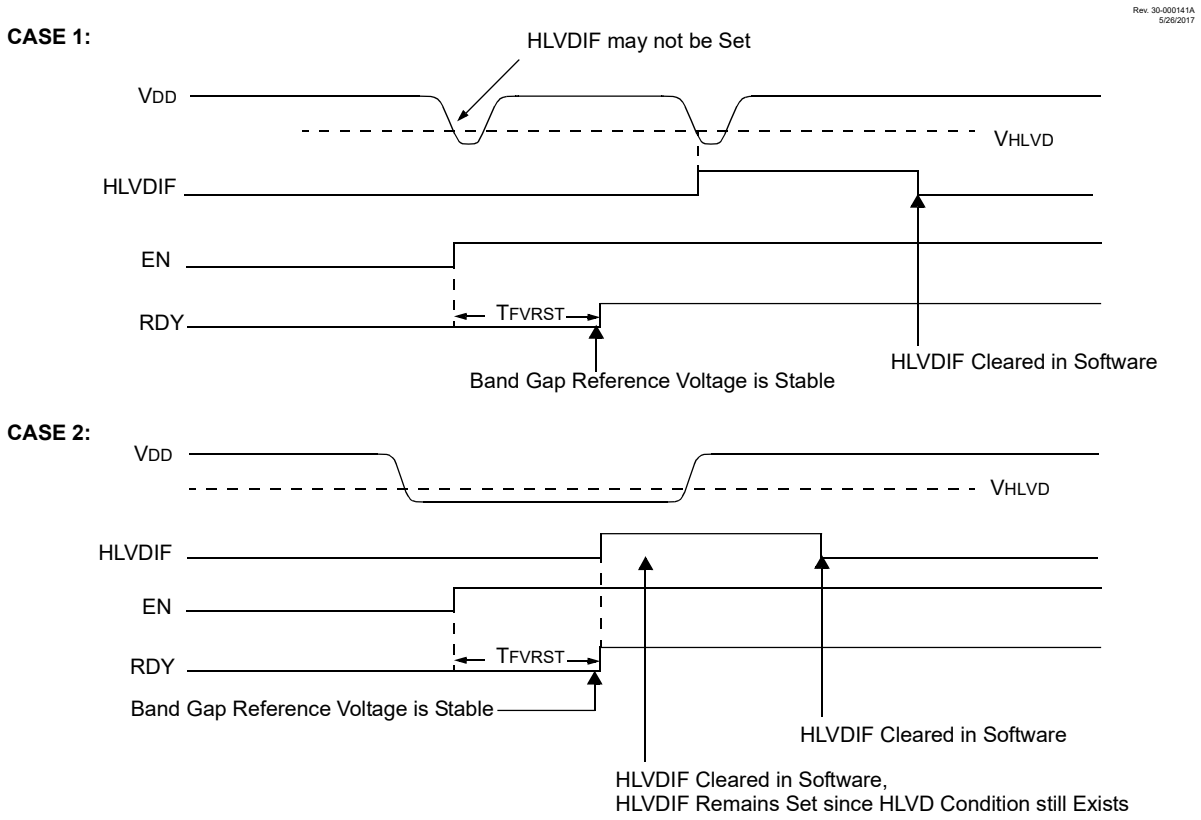
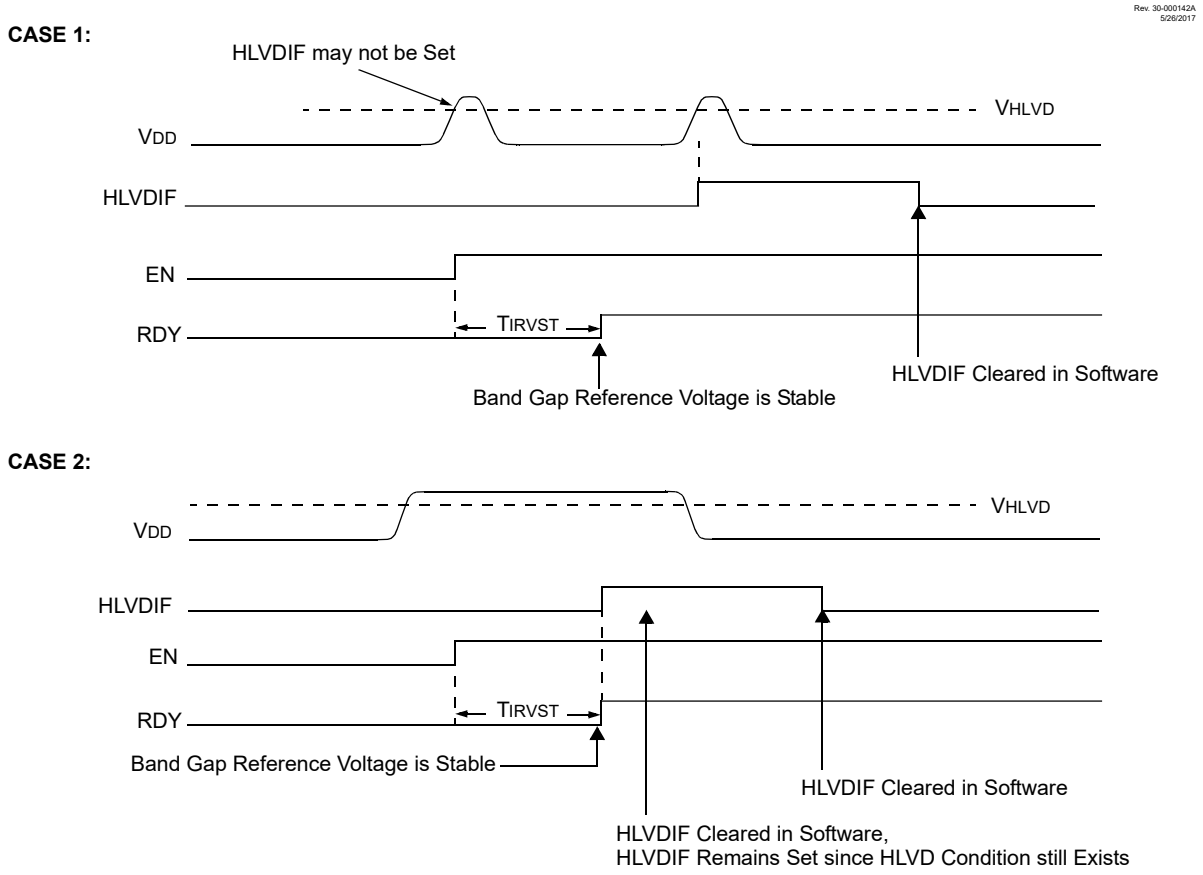


**Figure 37-3. High-Voltage Detect Operation (INTH = 1)**

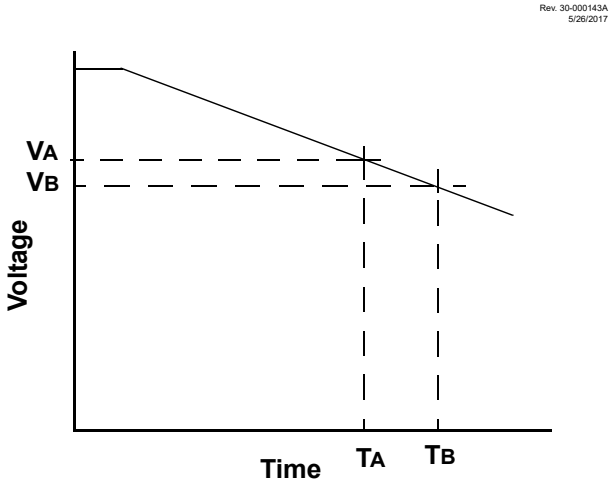



## 37.5 Applications

In many applications, it is desirable to detect a drop below, or rise above, a particular voltage threshold. For example, the HLVD module can be periodically enabled to detect Universal Serial Bus (USB) attach or detach. This assumes the device is powered by a lower voltage source than the USB when detached. An attach indicates a High-Voltage Detect from, for example, 3.3V to 5V (the voltage on USB) and vice versa for a detach. This feature can save a design a few extra components and an attach signal (input pin).

For general battery applications, the figure below shows a possible voltage curve. Over time, the device voltage decreases. When the device voltage reaches voltage, $V_A$, the HLVD logic generates an interrupt at time, $T_A$. The interrupt can cause the execution of an Interrupt Service Routine (ISR), which will allow the application to perform “housekeeping tasks” and a controlled shutdown before the device voltage exits the valid operating range at $T_B$. This will give the application a time window, represented by the difference between $T_A$ and $T_B$, to safely exit.

**Figure 37-4. Typical Low-Voltage Detect Application**




**Legend:** $V_A$ = HLVD trip point
$V_B$ = Minimum valid device operating voltage

## 37.6 Operation During Sleep

When enabled, the HLVD circuitry continues to operate during Sleep. When the device voltage crosses the trip point, the HLVDIF bit will be set and the device will wake up from Sleep. If interrupts are enabled, the device will execute code from the interrupt vector. If interrupts are disabled, the device will continue execution from the next instruction after `SLEEP`.

## 37.7 Operation During Idle and Doze Modes

The performance of the module is independent of the Idle and Doze modes. The module will generate the events based on the trip points. The response to these events will depend on the Doze and Idle mode settings.

## 37.8 Effects of a Reset

A device Reset forces all registers to their Reset state. This forces the HLVD module to be turned off. User firmware has to configure the module again.

## 37.9 Register Definitions: HLVD Control

Long bit name prefixes for the HLVD peripheral is shown in the following table. Refer to the **"Long Bit Names"** section in the **"Register and Bits Naming Conventions"** chapter for more information.

**Table 37-1. HLVD Long Bit Name Prefixes**

| Peripheral | Bit Name Prefix |
|---|---|
| HLVD | HLVD |

### 37.9.1 HLVDCON0

**Name:** HLVDCON0
**Address:** 0x04A

High/Low-Voltage Detect Control Register 0

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | EN | | OUT | RDY | | | INTH | INTL |
| Access | R/W | | R | R | | | R/W | R/W |
| Reset | 0 | | x | x | | | 0 | 0 |

**Bit 7 – EN** High/Low-voltage Detect Power Enable

| Value | Description |
|---|---|
| 1 | Enables the HLVD module |
| 0 | Disables the HLVD module |

**Bit 5 – OUT** HLVD Comparator Output

| Value | Description |
|---|---|
| 1 | Voltage < selected detection limit (SEL) |
| 0 | Voltage > selected detection limit (SEL) |

**Bit 4 – RDY** Band Gap Reference Voltages Stable Status Flag

| Value | Description |
|---|---|
| 1 | Indicates HLVD Module is ready and output is stable |
| 0 | Indicates HLVD Module is not ready |

**Bit 1 – INTH** HLVD Positive going (High Voltage) Interrupt Enable

| Value | Description |
|---|---|
| 1 | HLVDIF will be set when voltage ≥ selected detection limit (SEL) |
| 0 | HLVDIF will not be set |

**Bit 0 – INTL** HLVD Negative going (Low Voltage) Interrupt Enable

| Value | Description |
|---|---|
| 1 | HLVDIF will be set when voltage ≤ selected detection limit (SEL) |
| 0 | HLVDIF will not be set |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

### 37.9.2 HLVDCON1

**Name:** HLVDCON1
**Address:** 0x04B

Low-Voltage Detect Control Register 1

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | SEL[3:0] | | | |
| Access | | | | | R/W | R/W | R/W | R/W |
| Reset | | | | | 0 | 0 | 0 | 0 |

**Bits 3:0 – SEL[3:0]** High/Low-Voltage Detection Limit Selection
Refer to the **"High/Low Voltage Detect Characteristic"** table in the **"Electrical Specification"** chapter for more details about the voltage detection limit selection.
Reset States: POR/BOR = 0000
All other Resets = uuuu

## 37.10 Register Summary - HLVD

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x4A | HLVDCON0 | 7:0 | EN | | OUT | RDY | | | INTH | INTL |
| 0x4B | HLVDCON1 | 7:0 | | | | | SEL[3:0] | | | |

# 38. FVR - Fixed Voltage Reference

The Fixed Voltage Reference (FVR) is a stable voltage reference, independent of $V_{DD}$, with 1.024V, 2.048V or 4.096V selectable output levels. The output of the FVR can be configured to supply a reference voltage to analog peripherals such as those listed below.

- ADC input channel
- ADC positive reference
- Comparator input
- Digital-to-Analog Converter (DAC)

The FVR can be enabled by setting the EN bit to '1'.

**Note:** Fixed Voltage Reference output cannot exceed $V_{DD}$.

## 38.1 Independent Gain Amplifiers

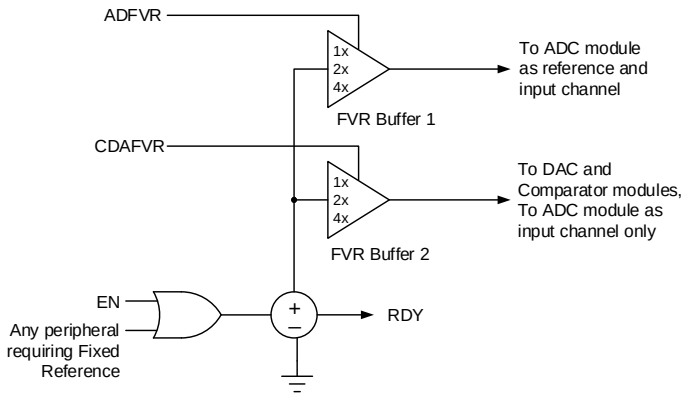
The output of the FVR is routed through two independent programmable gain amplifiers. Each amplifier can be programmed for a gain of 1x, 2x or 4x, to produce the three possible voltage levels.

The ADFVR bits are used to enable and configure the gain amplifier settings for the reference supplied to the ADC module. Refer to the **"ADCC - Analog-to-Digital Converter with Computation Module"** chapter for additional information.

The CDAFVR bits are used to enable and configure the gain amplifier settings for the reference supplied to the DAC and comparator modules. Refer to the **"DAC - Digital-to-Analog Converter Module"** and **"CMP - Comparator Module"** chapters for additional information.

Refer to the figure below for the block diagram of the FVR module.

**Figure 38-1. Fixed Voltage Reference Block Diagram**




## 38.2 FVR Stabilization Period

When the Fixed Voltage Reference module is enabled, it requires time for the reference and amplifier circuits to stabilize. Once the circuits stabilize and are ready for use, the RDY bit will be set.

## 38.3 Register Definitions: FVR

Long bit name prefixes for the FVR peripherals are shown in the following table. Refer to the **"Long Bit Names"** section in the **"Register and Bits Naming Conventions"** chapter for more information.

**Table 38-1. FVR Long Bit Name Prefixes**

| Peripheral | Bit Name Prefix |
|---|---|
| FVR | FVR |

### 38.3.1 FVRCON

**Name:** FVRCON
**Address:** 0x3D7

FVR Control Register

**Important:** This register is shared between the Fixed Voltage Reference (FVR) module and the temperature indicator module.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | EN | RDY | TSEN | TSRNG | CDAFVR[1:0] | | ADFVR[1:0] | |
| Access | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | q | 0 | 0 | 0 | 0 | 0 | 0 |

**Bit 7 – EN** Fixed Voltage Reference Enable

| Value | Description |
|---|---|
| 1 | Enables module |
| 0 | Disables module |

**Bit 6 – RDY** Fixed Voltage Reference Ready Flag

| Value | Description |
|---|---|
| 1 | Fixed Voltage Reference output is ready for use |
| 0 | Fixed Voltage Reference output is not ready for use or not enabled |

**Bit 5 – TSEN** Temperature Indicator Enable

| Value | Description |
|---|---|
| 1 | Temperature Indicator is enabled |
| 0 | Temperature Indicator is disabled |

**Bit 4 – TSRNG** Temperature Indicator Range Selection

| Value | Description |
|---|---|
| 1 | $V_{OUT} = 3V_T$ (High Range) |
| 0 | $V_{OUT} = 2V_T$ (Low Range) |

**Bits 3:2 – CDAFVR[1:0]** FVR Buffer 2 Gain Selection(1)

| Value | Description |
|---|---|
| 11 | FVR Buffer 2 Gain is 4x, (4.096V)(3) |
| 10 | FVR Buffer 2 Gain is 2x, (2.048V)(3) |
| 01 | FVR Buffer 2 Gain is 1x, (1.024V) |
| 00 | FVR Buffer 2 is OFF |

**Bits 1:0 – ADFVR[1:0]** FVR Buffer 1 Gain Selection(2)

| Value | Description |
|---|---|
| 11 | FVR Buffer 1 Gain is 4x, (4.096V)(3) |
| 10 | FVR Buffer 1 Gain is 2x, (2.048V)(3) |
| 01 | FVR Buffer 1 Gain is 1x, (1.024V) |
| 00 | FVR Buffer 1 is OFF |

**Notes:**

1. This output goes to the DAC and comparator modules, and to the ADC module as an input channel only.
2. This output goes to the ADC module as a reference and an input channel.
3. Fixed Voltage Reference output cannot exceed $V_{DD}$.

## 38.4 Register Summary - FVR

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x03D7 | FVRCON | 7:0 | EN | RDY | TSEN | TSRNG | CDAFVR[1:0] | | ADFVR[1:0] | |

# 39. Temperature Indicator Module

This family of devices is equipped with a temperature circuit designed to measure the operating temperature of the silicon die. The temperature indicator module provides a temperature-dependent voltage that can be measured by the internal Analog-to-Digital Converter.

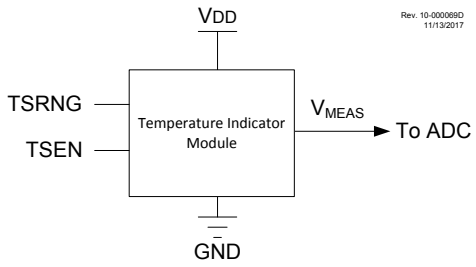
The circuit's range of operating temperature falls between -40℃ and +125℃. The circuit may be used as a temperature threshold detector or a more accurate temperature indicator, depending on the level of calibration performed. A one-point calibration allows the circuit to indicate a temperature closely surrounding that point. A two-point calibration allows the circuit to sense the entire range of temperature more accurately.

## 39.1 Module Operation

The temperature indicator module consists of a temperature-sensing circuit that provides a voltage to the device ADC. The analog voltage output varies inversely to the device temperature. The output of the temperature indicator is referred to as $V_{MEAS}$.

The following figure shows a simplified block diagram of the temperature indicator module.

**Figure 39-1. Temperature Indicator Module Block Diagram**




The output of the circuit is measured using the internal Analog-to-Digital Converter. A channel is reserved for the temperature circuit output. Refer to the **"ADCC - Analog-to-Digital Converter with Computation Module"** chapter for more details.

The ON/OFF bit for the module is located in the FVRCON register. The circuit is enabled by setting the TSEN bit. When the module is disabled, the circuit draws no current. Refer to the **"FVR - Fixed Reference Voltage"** chapter for more details.

### 39.1.1 Temperature Indicator Range

The temperature indicator circuit operates in either high or low range. The high range, selected by setting the TSRNG bit, provides a wider output voltage. This provides more resolution over the temperature range. High range requires a higher bias voltage to operate and thus, a higher $V_{DD}$ is needed. The low range is selected by clearing the TSRNG bit. The low range generates a lower sensor voltage and thus, a lower $V_{DD}$ voltage is needed to operate the circuit.

The output voltage of the sensor is the highest value at -40℃ and the lowest value at +125℃.

- **High Range:** The high range is used in applications with the reference for the ADC, $V_{REF}$ = 2.048V. This range may not be suitable for battery-powered applications.
- **Low Range:** This mode is useful in applications in which the $V_{DD}$ is too low for high-range operation. The $V_{DD}$ in this mode can be as low as 1.8V. However, $V_{DD}$ must be at least 0.5V higher than the maximum sensor voltage depending on the expected low operating temperature.

**Important:** The standard parameters for the Temperature Sensor for both high range and low range are stored in the DIA table. Refer to the DIA table in the **"Memory Organization"** chapter for more details.

### 39.1.2 Minimum Operating $V_{DD}$

When the temperature circuit is operated in low range, the device may be operated at any operating voltage that is within the device specifications. When the temperature circuit is operated in high range, the device operating voltage, $V_{DD}$, must be high enough to ensure that the temperature circuit is correctly biased.

The following table shows the recommended minimum $V_{DD}$ vs. Range setting.

**Table 39-1. RECOMMENDED $V_{DD}$ vs. RANGE**

| Min. $V_{DD}$, TSRNG = 1 (High Range) | Min. $V_{DD}$, TSRNG = 0 (Low Range) |
|---|---|
| ≥ 2.5 | ≥ 1.8 |

## 39.2 Temperature Calculation

This section describes the steps involved in calculating the die temperature, $T_{MEAS}$:

1. Obtain the ADC count value of the measured analog voltage: The analog output voltage, $V_{MEAS}$, is converted to a digital count value by the Analog-to-Digital Converter (ADC) and is referred to as $ADC_{MEAS}$.
2. Obtain the Gain value from the DIA table. This parameter is TSLR1 for the low range setting or TSHR1 for the high range setting of the temperature indicator module. Refer to the DIA table in the **"Memory Organization"** chapter for more details.
3. Obtain the Offset value from the DIA table. This parameter is TSLR3 for the low range setting or TSHR3 for the high range setting of the temperature indicator module. Refer to the DIA table in the **"Memory Organization"** chapter for more details.

The following equation provides an estimate for the die temperature based on the above parameters:

**Equation 39-1. Sensor Temperature (in ℃)**

$$T_{MEAS} = \frac{\frac{(ADC_{MEAS} \times Gain)}{256} + Offset}{10}$$

Where:
$ADC_{MEAS}$ = ADC reading at temperature being estimated

Gain = Gain value stored in the DIA table

Offset = Offset value stored in the DIA table

**Note:** It is recommended to take the average of ten measurements of $ADC_{MEAS}$ to reduce noise and improve accuracy.

**Example 39-1. Temperature Calculation (C)**

```
// offset is int16_t data type
// gain is int16_t data type
// ADC_MEAS is uint16_t data type
// Temp_in_C is int24_t data type

ADC_MEAS = ((ADRESH << 8) + ADRESL);      // Store the ADC Result
Temp_in_C = (int24_t)(ADC_MEAS) * gain;   // Multiply the ADC Result by
                                          // Gain and store the result in a
                                          // signed variable
Temp_in_C = Temp_in_C / 256;              // Divide (ADC Result * Gain) by 256
Temp_in_C = Temp_in_C + offset;           // Add (Offset) to the result
Temp_in_C = Temp_in_C / 10;               // Divide the result by 10 and store
                                          // the calculated temperature
```

### 39.2.1 Higher-Order Calibration

If the application requires more precise temperature measurement, additional calibrations steps will be necessary. For these applications, two-point or three-point calibration is recommended. For additional information on two-

point calibration method, refer to the following Microchip application note, available at the corporate website (www.microchip.com):

- AN2798, *"Using the PIC16F/PIC18F Ground Referenced Temperature Indicator Module"*

## 39.3 ADC Acquisition Time

To ensure accurate temperature measurements, the user must wait a certain minimum acquisition time (parameter TS01) after the temperature indicator output is selected as ADC input. This is required for the ADC sampling circuit to settle before the conversion is performed.

**Note:** Parameter TS01 can be found in the Temperature Indicator Requirements table of the **"Electrical Specifications"** chapter.

## 39.4 Register Definitions: Temperature Indicator

### 39.4.1 FVRCON

**Name:** FVRCON
**Address:** 0x3D7

FVR Control Register

**Important:** This register is shared between the Fixed Voltage Reference (FVR) module and the temperature indicator module.

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | EN | RDY | TSEN | TSRNG | CDAFVR[1:0] | | ADFVR[1:0] | |
| Access | R/W | R | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | q | 0 | 0 | 0 | 0 | 0 | 0 |

**Bit 7 – EN** Fixed Voltage Reference Enable

| Value | Description |
|---|---|
| 1 | Enables module |
| 0 | Disables module |

**Bit 6 – RDY** Fixed Voltage Reference Ready Flag

| Value | Description |
|---|---|
| 1 | Fixed Voltage Reference output is ready for use |
| 0 | Fixed Voltage Reference output is not ready for use or not enabled |

**Bit 5 – TSEN** Temperature Indicator Enable

| Value | Description |
|---|---|
| 1 | Temperature Indicator is enabled |
| 0 | Temperature Indicator is disabled |

**Bit 4 – TSRNG** Temperature Indicator Range Selection

| Value | Description |
|---|---|
| 1 | $V_{OUT} = 3V_T$ (High Range) |
| 0 | $V_{OUT} = 2V_T$ (Low Range) |

**Bits 3:2 – CDAFVR[1:0]** FVR Buffer 2 Gain Selection(1)

| Value | Description |
|---|---|
| 11 | FVR Buffer 2 Gain is 4x, (4.096V)(3) |
| 10 | FVR Buffer 2 Gain is 2x, (2.048V)(3) |
| 01 | FVR Buffer 2 Gain is 1x, (1.024V) |
| 00 | FVR Buffer 2 is OFF |

**Bits 1:0 – ADFVR[1:0]** FVR Buffer 1 Gain Selection(2)

| Value | Description |
|---|---|
| 11 | FVR Buffer 1 Gain is 4x, (4.096V)(3) |
| 10 | FVR Buffer 1 Gain is 2x, (2.048V)(3) |
| 01 | FVR Buffer 1 Gain is 1x, (1.024V) |
| 00 | FVR Buffer 1 is OFF |

**Notes:**

1. This output goes to the DAC and comparator modules, and to the ADC module as an input channel only.
2. This output goes to the ADC module as a reference and an input channel.
3. Fixed Voltage Reference output cannot exceed $V_{DD}$.

## 39.5 Register Summary - Temperature Indicator

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x03D7 | FVRCON | 7:0 | EN | RDY | TSEN | TSRNG | CDAFVR[1:0] | | ADFVR[1:0] | |

# 40. ADCC - Analog-to-Digital Converter with Computation Module

The Analog-to-Digital Converter with Computation (ADCC) allows conversion of an analog input signal to a 12-bit binary representation of that signal. This device uses analog inputs that are multiplexed into a single Sample-and-Hold circuit. The output of the Sample-and-Hold is connected to the input of the converter. The converter generates a 12-bit binary result via successive approximation and stores the conversion result into the ADC result registers.

Additionally, the following features are provided within the ADC module:

- Acquisition Timer
- Hardware Capacitive Voltage Divider (CVD) support:
  - Precharge timer
  - Adjustable Sample-and-Hold capacitor array
  - Guard ring digital output drive
- Automatic Repeat and Sequencing:
  - Automated double sample conversion for CVD
  - Two sets of Result registers (Current Result and Previous Result)
  - Auto-conversion trigger
  - Internal re-trigger
- Computation Features:
  - Averaging and low-pass filter functions
  - Reference comparison
  - 2-level threshold comparison
  - Selectable interrupts

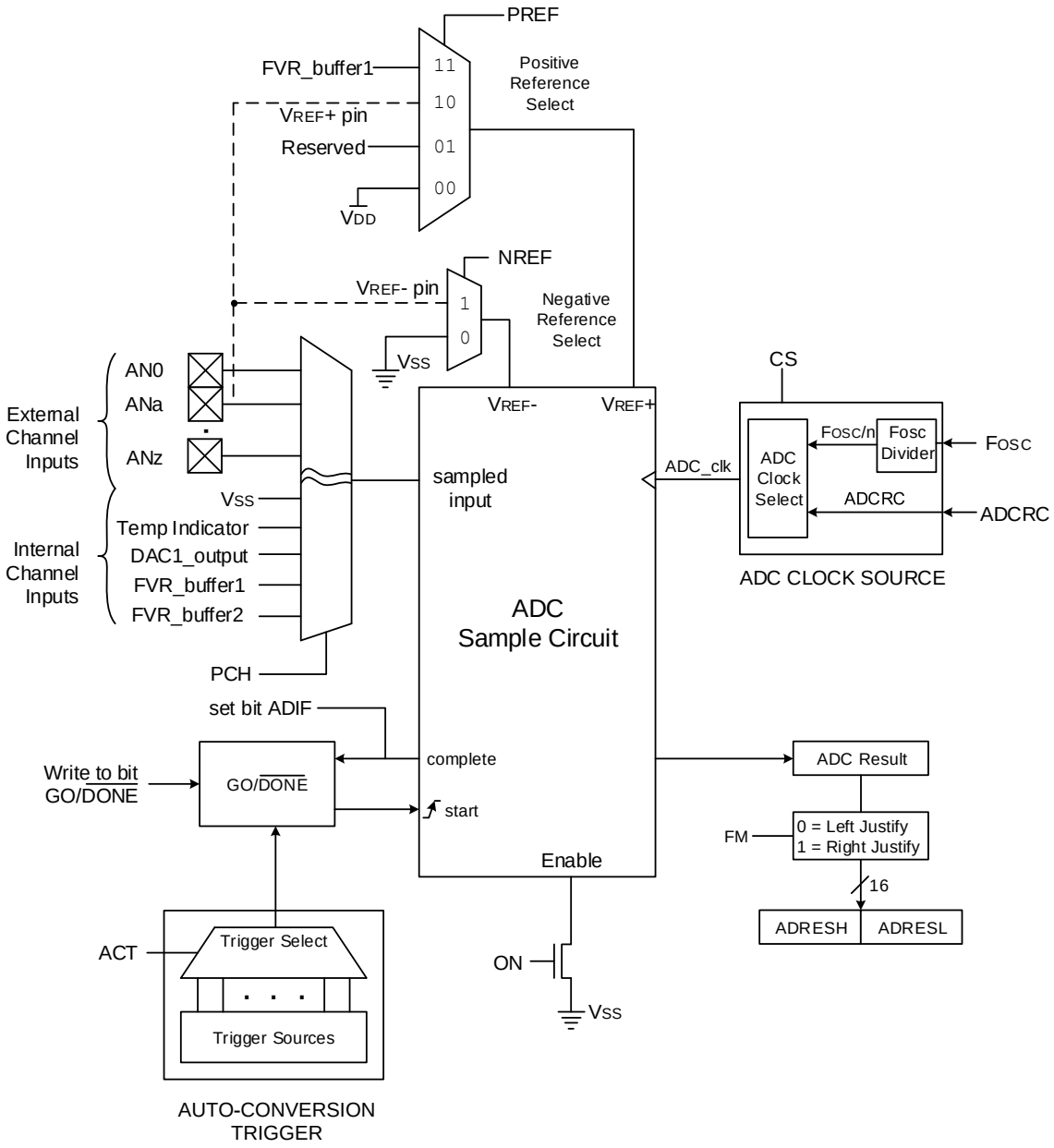
Figure 40-1 shows the block diagram of the ADC.

The ADC voltage reference is software selectable to be either internally generated or externally supplied.

The ADC can generate an interrupt upon completion of a conversion and upon threshold comparison. These interrupts can be used to wake up the device from Sleep.

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

**Figure 40-1. ADCC Block Diagram**




## 40.1 ADC Configuration

When configuring the ADC the following functions must be considered:

- Port Configuration
- Channel Selection
- ADC Voltage Reference Selection
- ADC Conversion Clock Source
- Interrupt Control
- Result Formatting
- Conversion Trigger Selection

- ADC Acquisition Time
- ADC Precharge Time
- Additional Sample-and-Hold Capacitor
- Single/Double Sample Conversion
- Guard Ring Outputs

### 40.1.1 Port Configuration

The ADC will convert the voltage level on a pin whether or not the ANSEL bit is set. When converting analog signals, the I/O pin will be configured for analog by setting the associated TRIS and ANSEL bits. Refer to the **"I/O Ports"** chapter for more information.

**Important:** Analog voltages on any pin that is defined as a digital input may cause the input buffer to conduct excess current.

### 40.1.2 Channel Selection

The ADPCH register determines which channel is connected to the Sample-and-Hold circuit for conversion. When switching channels, it is recommended to have some acquisition time (ADACQ register) before starting the next conversion. Refer to the ADC Operation section for more information.

**Important:** To reduce the chance of measurement error, it is recommended to discharge the Sample-and-Hold capacitor when switching between ADC channels by starting a conversion on a channel connected to $V_{SS}$ and terminating the conversion after the acquisition time has elapsed. If the ADC does not have a dedicated $V_{SS}$ input channel, the $V_{SS}$ selection through the DAC output channel can be used. If the DAC is in use, a free input channel can be connected to $V_{SS}$, and can be used in place of the DAC.

### 40.1.3 ADC Voltage Reference

The PREF bits provide control of the positive voltage reference. The NREF bit provides control of the negative voltage reference. Refer to the ADREF register for the list of available positive and negative sources.

### 40.1.4 Conversion Clock

The conversion clock source is selected with the CS bit. When CS = 1 the ADC clock source is an internal fixed-frequency clock referred to as ADCRC. When CS = 0 the ADC clock source is derived from $F_{OSC}$.

**Important:** When CS = 0, the clock can be divided using the ADCLK register to meet the ADC clock period requirements.

The time to complete one bit conversion is defined as the $T_{AD}$. Refer to Figure 40-2 for the complete timing details of the ADC conversion.

For correct conversion, the appropriate $T_{AD}$ specification must be met. Refer to the ADC Timing Specifications table in the **"Electrical Specifications"** chapter for more details. The table below gives examples of appropriate ADC clock selections.

**Table 40-1. ADC Clock Period ($T_{AD}$) Vs. Device Operating Frequencies(1,3)**

| ADC Clock Source | ADCLK | ADC Clock Period ($T_{AD}$) for Different Device Frequency ($F_{OSC}$) | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | 64 MHz | 32 MHz | 20 MHz | 16 MHz | 8 MHz | 4 MHz | 1 MHz |
| $F_{OSC}/2$ | 'b000000 | 31.25 ns(2) | 62.5 ns(2) | 100 ns(2) | 125 ns(2) | 250 ns(2) | 500 ns | 2.0 µs |

.....continued

| ADC Clock Source | ADCLK | ADC Clock Period ($T_{AD}$) for Different Device Frequency ($F_{OSC}$) | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | 64 MHz | 32 MHz | 20 MHz | 16 MHz | 8 MHz | 4 MHz | 1 MHz |
| $F_{OSC}$/4 | 'b000001 | 62.5 ns(2) | 125 ns(2) | 200 ns(2) | 250 ns(2) | 500 ns | 1.0 µs | 4.0 µs |
| $F_{OSC}$/6 | 'b000010 | 93.75 ns(2) | 187.5 ns(2) | 300 ns(2) | 375 ns(2) | 750 ns | 1.5 µs | 6.0 µs |
| $F_{OSC}$/8 | 'b000011 | 125 ns(2) | 250 ns(2) | 400 ns(2) | 500 ns | 1.0 µs | 2.0 µs | 8.0 µs |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| $F_{OSC}$/16 | 'b000111 | 250 ns(2) | 500 ns | 800 ns | 1.0 µs | 2.0 µs | 4.0 µs | 16.0 µs(2) |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| $F_{OSC}$/32 | 'b001111 | 500 ns | 1.0 µs | 1.6 µs | 2.0 µs | 4.0 µs | 8.0 µs | 32.0 µs(2) |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| $F_{OSC}$/64 | 'b0111111 | 1.0 µs | 2.0 µs | 3.2 µs | 4.0 µs | 8.0 µs | 16.0 µs(3) | 64.0 µs(2) |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| $F_{OSC}$/128 | 'b111111 | 2.0 µs | 4.0 µs | 6.4 µs | 8.0 µs | 16.0 µs(2) | 32.0 µs(2) | 128.0 µs(2) |
| ADCRC | CS=1 | 1.0-6.0 µs | 1.0-6.0 µs | 1.0-6.0 µs | 1.0-6.0 µs | 1.0-6.0 µs | 1.0-6.0 µs | 1.0-6.0 µs |

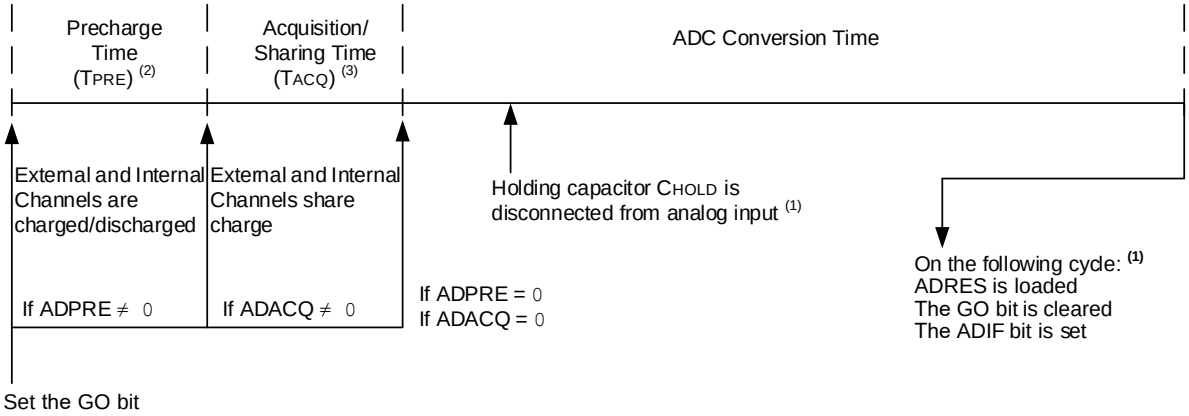
**Notes:**

1. Refer to the **"Electrical Specifications"** chapter to see the $T_{AD}$ parameter for the ADCRC source typical $T_{AD}$ value.
2. These values violate the required $T_{AD}$ time.
3. The ADC clock period ($T_{AD}$) and total ADC conversion time can be minimized when the ADC clock is derived from the system clock $F_{OSC}$. However, the ADCRC oscillator source must be used when conversions are to be performed with the device in Sleep mode.

**Important:**

- Except for the ADCRC clock source, any changes in the system clock frequency will change the ADC clock frequency, which may adversely affect the ADC result.
- The internal control logic of the ADC runs off of the clock selected by the CS bit. When the CS bit is set to '1' (ADC runs on ADCRC), there may be unexpected delays in operation when setting the ADC control bits.

**Figure 40-2. Analog-to-Digital Conversion Cycles**




**Note:**
1. Refer to the ADC Conversion Timing Specifications table in the **"Electrical Specifications"** chapter.
2. Refer to the ADPRE register for more details.
3. Refer to the ADACQ register for more details.

### 40.1.5 Interrupts

The ADC module allows for the ability to generate an interrupt upon completion of an Analog-to-Digital Conversion. The ADC Interrupt Flag is the ADIF bit in the PIRx register. The ADC Interrupt Enable is the ADIE bit in the PIEx register. The ADIF bit must be cleared by software.

**Important:**
1. The ADIF bit is set at the completion of every conversion, regardless of whether or not the ADC interrupt is enabled.
2. The ADC operates during Sleep only when the ADCRC oscillator is selected.

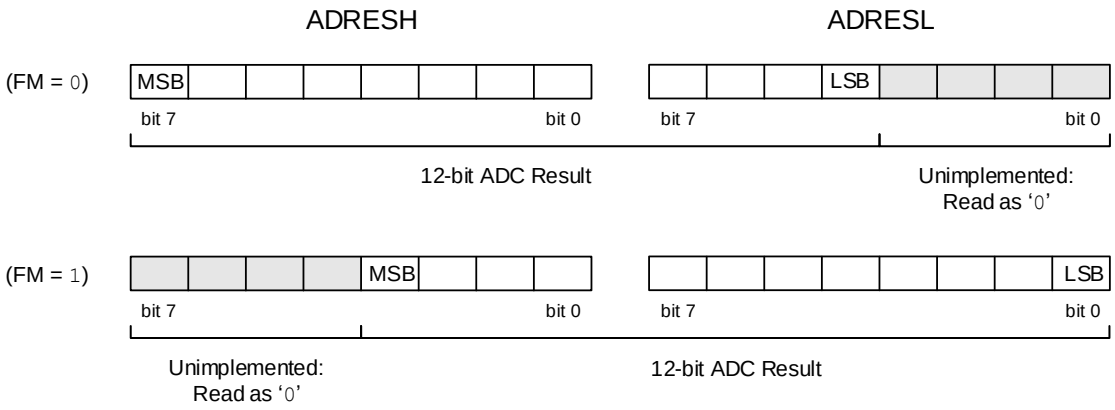
While the device is operating in Sleep:

- If ADIE = 1 and GIE = 0 : An interrupt will wake up the device from Sleep. Upon waking from Sleep, the instructions following the SLEEP instruction is executed. Interrupt Service Routine is not executed.
- If ADIE = 1 and GIE = 1 : An interrupt will wake up the device from Sleep. Upon waking from Sleep, the instruction following the SLEEP instruction is always executed. Then the execution will switch to the Interrupt Service Routine.

### 40.1.6 Result Formatting

The ADC conversion result can be supplied in two formats, left justified or right justified. The FM bit controls the output format as shown in Figure 40-3.

**Figure 40-3. 12-Bit ADC Conversion Result Format**




**Important:** Writes to the ADRES register pair are always right justified regardless of the selected format mode. Therefore, a data read after writing to ADRES when FM = 0 will be shifted left four places.

## 40.2 ADC Operation

### 40.2.1 Starting a Conversion

To enable the ADC module, the ON bit must be set to '1'. A conversion may be started by any of the following:

- Software setting the GO bit to '1'
- An external trigger (source selected by ADACT)
- A Continuous-mode retrigger (see the Continuous Sampling Mode section for more details)

**Important:** The GO bit will not be set in the same instruction that turns on the ADC. Refer to the ADC Conversion Procedure (Basic Mode) section for more details.

### 40.2.2 Completion of a Conversion

When any individual conversion is complete, the existing value in ADRES is written into ADPREV (if PSIS = 0) and the new conversion results appear in ADRES. When the conversion completes, the ADC module will:

- Clear the GO bit (unless the CONT bit is set)
- Set the ADIF Interrupt Flag bit
- Set the MATH bit
- Update ADACC

After every conversion when DSEN = 0, or after every other conversion when DSEN = 1, the following events occur:

- ADERR is calculated
- ADTIF interrupt is set if ADERR calculation meets threshold comparison

### 40.2.3 ADC Operation During Sleep

The ADC module can operate during Sleep. This requires the ADC clock source to be set to the ADCRC option. When the ADCRC oscillator source is selected, the ADC waits one additional instruction before starting the

conversion. This allows the SLEEP instruction to be executed, which can reduce system noise during the conversion. If the ADC interrupt is enabled, the device will wake up from Sleep when the conversion completes. If the ADC interrupt is disabled, the device remains in Sleep and the ADC module is turned off after the conversion completes, although the ON bit remains set.

### 40.2.4 External Trigger During Sleep

If the external trigger is received during Sleep while the ADC clock source is set to the ADCRC, the ADC module will perform the conversion and set the ADIF bit upon completion.

If an external trigger is received when the ADC clock source is something other than ADCRC, the trigger will be recorded, but the conversion will not begin until the device exits Sleep.

### 40.2.5 Auto-Conversion Trigger

The auto-conversion trigger allows periodic ADC measurements without software intervention. When a rising edge of the selected source occurs, the GO bit is set by hardware.

The auto-conversion trigger source is selected with the ACT bits.

Using the auto-conversion trigger does not ensure proper ADC timing. It is the user's responsibility to ensure that the ADC timing requirements are met.

### 40.2.6 ADC Conversion Procedure (Basic Mode)

This is an example procedure for using the ADC to perform an Analog-to-Digital Conversion:

1. Configure Port:
   - Disable pin output driver (refer to the TRISx register)
   - Configure pin as analog (refer to the ANSELx register)
2. Configure the ADC module:
   - Select ADC conversion clock
   - Configure voltage reference
   - Select ADC input channel
   - Configure precharge (ADPRE) and acquisition (ADACQ) time period
   - Turn on ADC module
3. Configure ADC interrupt (optional):
   - Clear ADC interrupt flag
   - Enable ADC interrupt
   - Enable global interrupt (GIE bit)(1)
4. If ADACQ = 0, software must wait the required acquisition time(2).
5. Start conversion by setting the GO bit.
6. Wait for ADC conversion to complete by one of the following:
   - Polling the GO bit
   - Waiting for the ADC interrupt (if interrupt is enabled)
7. Read ADC Result.
8. Clear the ADC interrupt flag (if interrupt is enabled).

**Notes:**

1. With global interrupts disabled (GIE = 0), the device will wake from Sleep but will not enter an Interrupt Service Routine.
2. Refer to the ADC Acquisition Requirements section for more details.

**Example 40-1. ADC Conversion (assembly)**

```
; This code block configures the ADC for polling, Vdd and Vss references,
; ADCRC oscillator, and AN0 input.
; Conversion start & polling for completion are included.

    BANKSEL ADCON1      ;
    clrf    ADCON1      ;
```

```
    clrf    ADCON2      ; Legacy mode, no filtering, ADRES->ADPREV
    clrf    ADCON3      ; no math functions
    clrf    ADREF       ; Vref = Vdd & Vss
    clrf    ADPCH       ; select RA0/AN0
    clrf    ADACQ       ; software controlled acquisition time
    clrf    ADCAP       ; default S&H capacitance
    clrf    ADRPT       ; no repeat measurements
    clrf    ADACT       ; auto-conversion disabled
    movlw   B'10010100' ; ADC On, right-justified, ADCRC clock
    movwf   ADCON0
    BANKSEL TRISA       ;
    bsf     TRISA,0     ; Set RA0 to input
    BANKSEL ANSEL       ;
    bsf     ANSEL,0     ; Set RA0 to analog
    call    SampleTime  ; Acquisiton delay
    BANKSEL ADCON0
    bsf     ADCON0,GO   ; Start conversion
    btfsc   ADCON0,GO   ; Is conversion done?
    goto    $-2         ; No, test again
    BANKSEL ADRESH      ;
    movf    ADRESH,W    ; Read upper byte
    movwf   RESULTHI    ; store in GPR space
    movf    ADRESL,W    ; Read lower byte
    movwf   RESULTLO    ; Store in GPR space
```

**Example 40-2. ADC Conversion (C)**

```
/*This code block configures the ADC
for polling, VDD and VSS references,
ADCRC oscillator and AN0 input.
Conversion start & polling for completion
are included.
  */
    void main() {
        //System Initialize
        initializeSystem();

        //Setup ADC
        ADCON0bits.FM = 1;      //right justify
        ADCON0bits.CS = 1;      //ADCRC Clock
        ADPCH = 0x00;           //RA0 is Analog channel
        TRISAbits.TRISA0 = 1;   //Set RA0 to input
        ANSELAbits.ANSELA0 = 1; //Set RA0 to analog
        ADCON0bits.ON = 1;      //Turn ADC On

    while (1) {
        ADCON0bits.GO = 1;     //Start conversion
        while (ADCON0bits.GO); //Wait for conversion done
        resultHigh = ADRESH;   //Read result
        resultLow = ADRESL;    //Read result
    }
 }
```

## 40.3 ADC Acquisition Requirements

For the ADC to meet its specified accuracy, the charge holding capacitor ($C_{HOLD}$) must be allowed to fully charge to the input channel voltage level. The analog input model is shown in Figure 40-4. The source impedance ($R_S$) and the internal sampling switch ($R_{SS}$) impedance directly affect the time required to charge the capacitor $C_{HOLD}$. The sampling switch ($R_{SS}$) impedance varies over the device voltage ($V_{DD}$). The maximum recommended impedance for analog sources is 10 kΩ. As the source impedance is decreased, the acquisition time may be decreased. After the analog input channel is selected (or changed), an ADC acquisition time must be completed before the conversion can be started. To calculate the minimum acquisition time, Equation 40-1 may be used. This equation assumes an error of 1/2 LSb. The 1/2 LSb error is the maximum error allowed for the ADC to meet its specified resolution.

**Equation 40-1. Acquisition Time Example**

Assumptions: Temperature = 50°C; External impedance = 10 kΩ; $V_{DD}$ = 5.0V

$T_{ACQ}$ = Amplifier Settling Time + Hold Capacitor Charging Time + Temperature Coefficient

$$T_{ACQ} = T_{AMP} + T_C + T_{COFF}$$

$$T_{ACQ} = 2\,\mu s + T_C + \left[\left(Temperature - 25°C\right)\left(0.05\,\mu s/°C\right)\right]$$

The value for $T_C$ can be approximated with the following equations:

$$V_{APPLIED}\left(1 - \frac{1}{\left(2^{n+1}\right) - 1}\right) = V_{CHOLD}$$ ; [1] $V_{CHOLD}$ charged to within ½ LSb

$$V_{APPLIED}\left(1 - e^{\frac{-T_C}{RC}}\right) = V_{CHOLD}$$ ; [2] $V_{CHOLD}$ charge response to $V_{APPLIED}$

$$V_{APPLIED}\left(1 - e^{\frac{-T_C}{RC}}\right) = V_{APPLIED}\left(1 - \frac{1}{\left(2^{n+1}\right) - 1}\right)$$ ; Combining [1] and [2]

Note: Where n = ADC resolution in bits

Solving for $T_C$:

$$T_C = -C_{HOLD}(R_{IC} + R_{SS} + R_S)\ ln\ (1/8191)$$

$$T_C = -28\,pF(1\,k\Omega + 7\,k\Omega + 10\,k\Omega)\ ln\ (0.0001221)$$

$$T_C = 4.54\,\mu s$$

Therefore:

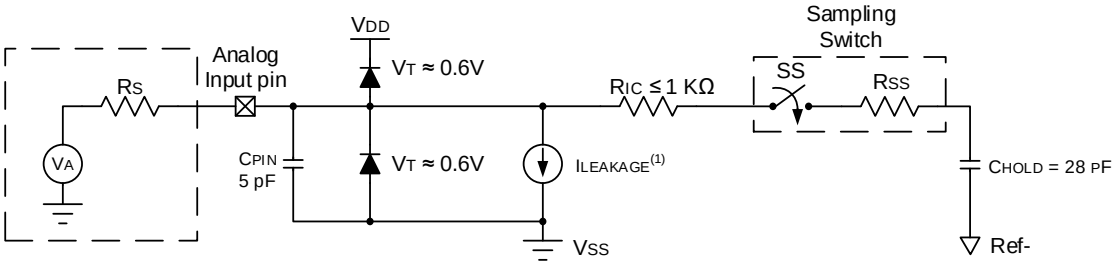
$$T_{ACQ} = 2\,\mu s + 4.54\,\mu s + \left[\left(50°C - 25°C\right)\left(0.05\,\mu s/°C\right)\right]$$

$$T_{ACQ} = 7.79\,\mu s$$

**Important:**

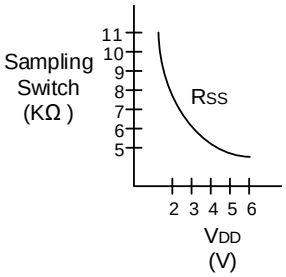
- The reference voltage ($V_{REF}$) has no effect on the equation, since it cancels itself out.
- The charge holding capacitor ($C_{HOLD}$) is not discharged after each conversion.
- The maximum recommended impedance for analog sources is 10 kΩ. This is required to meet the pin leakage specification.

**Figure 40-4. Analog Input Model**



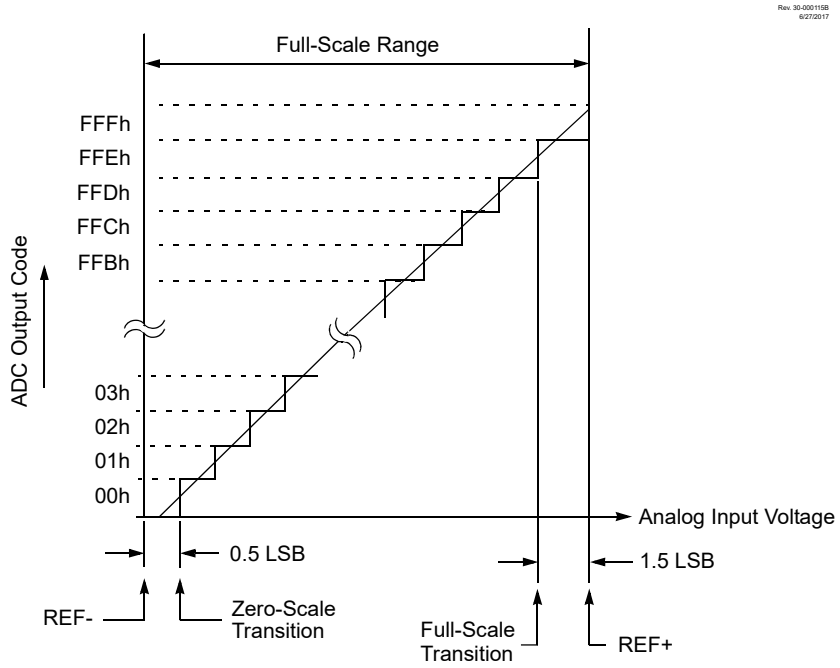

**Legend:**

| | |
|---|---|
| CPIN | = Input Capacitance |
| ILEAKAGE | = Leakage Current at the pin due to various junctions |
| RIC | = Interconnect Resistance |
| RS | = Source Impedance |
| VA | = Analog Voltage |
| VT | = Diode Forward Voltage |
| SS | = Sampling Switch |
| RSS | = Resistance of the Sampling Switch |
| CHOLD | = Sample/Hold Capacitance |




**Note:**
1. Refer to the **"Electrical Specifications"** chapter.

**Figure 40-5. ADC Transfer Function**




## 40.4 ADC Charge Pump

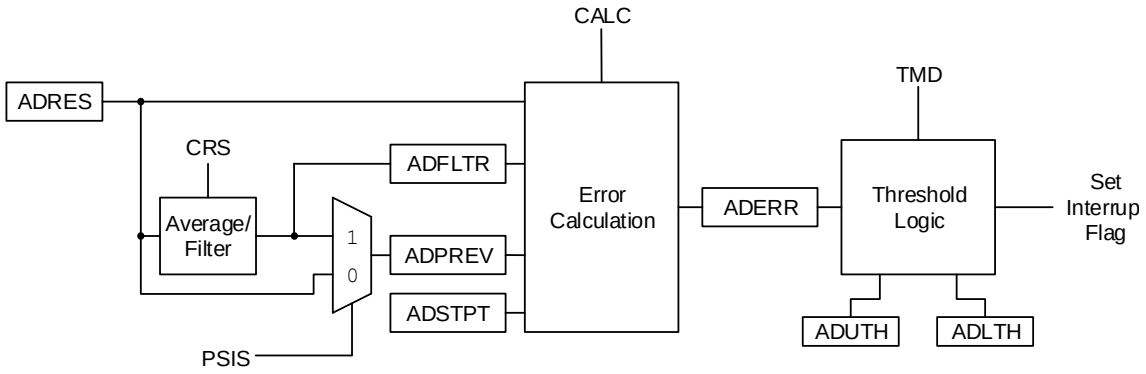
The ADC module has a dedicated charge pump which can be controlled through the ADCP register. The primary purpose of the charge pump is to supply a constant voltage to the gates of transistor devices in the Analog-to-Digital Converter, signal and reference input pass-gates, to prevent degradation of transistor performance at low operating voltage.

The charge pump can be enabled by setting the CPON bit. Once enabled, the pump will undergo a start-up time to stabilize the charge pump output. Once the output stabilizes and is ready for use, the CPRDY bit will be set.

## 40.5 Computation Operation

The ADC module hardware is equipped with post-conversion computation features. These features provide post-processing functions such as digital filtering/averaging and threshold comparison. Based on computation results, the module can be configured to take additional samples or stop conversions, and an interrupt may be asserted.

**Figure 40-6. Computational Features Simplified Block Diagram**




The operation of the ADC computational features is controlled by the MD bits.

The module can be operated in one of five modes:

- Basic: This is a Legacy mode. In this mode, ADC conversion occurs on single (DSEN = 0) or double (DSEN = 1) samples. ADIF is set after each conversion is complete. ADTIF is set according to the Calculation mode.
- Accumulate: With each trigger, the ADC conversion result is added to the accumulator and CNT increments. ADIF is set after each conversion. ADTIF is set according to the Calculation mode.
- Average: With each trigger, the ADC conversion result is added to the accumulator. When the RPT number of samples have been accumulated, a threshold test is performed. Upon the next trigger, the accumulator is cleared. For the subsequent tests, additional ADRPT samples are required to be accumulated.
- Burst Average: At the trigger, the accumulator is cleared. The ADC conversion results are then collected repetitively until ADRPT samples are accumulated and finally the threshold is tested.
- Low-Pass Filter (LPF): With each trigger, the ADC conversion result is sent through a filter. When ADRPT samples have occurred, a threshold test is performed. Every trigger after that the ADC conversion result is sent through the filter and another threshold test is performed.

The five modes are summarized in the following table.

**Table 40-2. Computation Modes**

| Mode | MD | Register Clear Event | Value after Cycle[1] Completion | | Threshold Operations | | | Value at ADTIF Interrupt | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | ADACC and CNT | ADACC | ADCNT | Retrigger | Threshold Test | Interrupt | AOV | ADFLTR | ADCNT |
| Basic | 0 | ACLR = 1 | Unchanged | Unchanged | No | Every Sample | If threshold = true | N/A | N/A | count |
| Accumulate | 1 | ACLR = 1 | S1 + ADACC or (S2 - S1) + ADACC | If (ADCNT = 0xFF): ADCNT, otherwise: ADCNT+1 | No | Every Sample | If threshold = true | ADACC Overflow | ADACC/$2^{CRS}$ | count |
| Average | 2 | ACLR = 1 or ADCNT ≥ ADRPT at GO set or retrigger | S1 + ADACC or (S2 -S1) + ADACC | If (ADCNT = 0xFF): ADCNT, otherwise: ADCNT + 1 | No | If ADCNT ≥ ADRPT | If threshold = true | ADACC Overflow | ADACC/$2^{CRS}$ | count |
| Burst Average | 3 | ACLR = 1 or at GO set or retrigger | Each repetition: same as Average End with sum of all samples | Each repetition: same as Average End with ADCNT = ADRPT | Repeat while ADCNT < ADRPT | If ADCNT ≥ ADRPT | If threshold = true | ADACC Overflow | ADACC/$2^{CRS}$ | ADRPT |
| Low-pass Filter | 4 | ACLR = 1 | S1 + ADACC-ADACC/ $2^{CRS}$ or (S2 - S1) + ADACC-ADACC/ $2^{CRS}$ | If (ADCNT = 0xFF): ADCNT, otherwise: ADCNT + 1 | No | If ADCNT ≥ ADRPT | If threshold = true | ADACC Overflow | ADACC/$2^{CRS}$ (Filtered Value) | count |

**Notes:**

1. When DSEN = 0, Cycle means one conversion. When DSEN = 1, Cycle means two conversions.
2. S1 and S2 are abbreviations for Sample 1 and Sample 2, respectively. When DSEN = 0, S1 = ADRES; When DSEN = 1, S1 = ADPREV and S2 = ADRES.

© 2019-2023 Microchip Technology Inc. and its subsidiaries

### 40.5.1 Digital Filter/Average

The digital filter/average module consists of an accumulator with data feedback options, and control logic to determine when threshold tests need to be applied. The accumulator can be accessed through the ADACC register.

Upon each trigger event (the GO bit set or external event trigger), the ADC conversion result is added to or subtracted from the accumulator. If the accumulated value exceeds $2^{(accumulator_width)}$-1 = $2^{18}$-1 = 262143, the AOV overflow bit is set.

The number of samples to be accumulated is determined by the ADRPT (ADC Repeat Setting) register. Each time a sample is added to the accumulator, the ADCNT register is incremented. Once ADRPT samples are accumulated (ADCNT = ADRPT), the accumulator may be cleared automatically depending on ADC Operation mode. An accumulator clear command can be issued in software by setting the ACLR bit. Setting the ACLR bit will also clear the AOV (Accumulator Overflow) bit, as well as the ADCNT register. The ACLR bit is cleared by the hardware when accumulator clearing action is complete.

**Important:** When ADC is operating from ADCRC, up to five ADCRC clock cycles are required to execute the ADACC clearing operation.

The CRS bits control the data shift on the accumulator result, which effectively divides the value in the accumulator registers. For the Accumulate mode of the digital filter, the shift provides a simple scaling operation. For the Average/Burst Average mode, the calculated average is only accurate when the number of samples agrees with the number of bits shifted. For the Low-Pass Filter mode, the shift is an integral part of the filter, and determines the cutoff frequency of the filter. Table 40-3 shows the -3 dB cutoff frequency in ωT (radians) and the highest signal attenuation obtained by this filter at Nyquist frequency (ωT = π).

**Table 40-3. Low-Pass Filter -3 dB Cutoff Frequency**

| CRS | ωT (radians) @ -3 dB Frequency | dB @ $F_{Nyquist}$=1/(2T) |
|---|---|---|
| 1 | 0.72 | -9.5 |
| 2 | 0.284 | -16.9 |
| 3 | 0.134 | -23.5 |
| 4 | 0.065 | -29.8 |
| 5 | 0.032 | -36.0 |
| 6 | 0.016 | -42.0 |

### 40.5.2 Basic Mode

Basic mode (MD = `'b000`) disables all additional computation features. In this mode, no accumulation occurs but threshold error comparison is performed. Double sampling, Continuous mode, and all CVD features are still available, but no digital filter/average calculations are performed.

### 40.5.3 Accumulate Mode

In Accumulate mode (MD = `'b001`), after every conversion, the ADC result is added to the ADACC register. The ADACC register is right-shifted by the value of the CRS bits. This right-shifted value is copied into the ADFLTR register. The Formatting mode does not affect the right-justification of the ADACC or ADFLTR values. Upon each sample, ADCNT is incremented, counting the number of samples accumulated. After each sample and accumulation, the ADFLTR value has a threshold comparison performed on it (see the Threshold Comparison section) and the ADTIF interrupt may trigger.

### 40.5.4 Average Mode

In Average mode (MD = `'b010`), the ADACC registers accumulate with each ADC sample, much as in Accumulate mode, and the ADCNT register increments with each sample. The ADFLTR register is also updated with the right-shifted value of the ADACC register. The value of the CRS bits governs the number of right shifts. However, in Average mode, the threshold comparison is performed upon ADCNT being greater than or equal to a user-defined

ADRPT value. In this mode, when ADRPT = 2^CRS, the final accumulated value will be divided by the number of samples, allowing for a threshold comparison operation on the average of all gathered samples.

### 40.5.5 Burst Average Mode

The Burst Average mode (MD = `'b011`) acts the same as the Average mode in most respects. The one way it differs is that it continuously retriggers ADC sampling until the CNT value is equal to ADRPT, even if Continuous Sampling mode (see the Continuous Sampling Mode section) is not enabled. This provides a threshold comparison on the average of a short burst of ADC samples.

### 40.5.6 Low-Pass Filter Mode

The Low-Pass Filter mode (MD = `'b100`) acts similarly to the Average mode in how it handles samples (accumulates samples until the ADCNT value is greater than or equal to RPT, then triggers a threshold comparison), but instead of a simple average, it performs a low-pass filter operation on all of the samples, reducing the effect of high-frequency noise on the total, then performs a threshold comparison on the results. In this mode, the CRS bits determine the cutoff frequency of the low-pass filter (as demonstrated by Digital Filter/Average). Refer to the Computation Operation section for a more detailed description of the mathematical operation.

For more information about Low-Pass Filter mode, refer to the following Microchip application note, available in the corporate website (www.microchip.com):

- AN2749, *"PIC18 12-bit ADCC in Low-Pass Filter Mode"*

### 40.5.7 Threshold Comparison

At the end of each computation:

- The conversion results are captured at the end-of-conversion.
- The error (ADERR) is calculated based on a difference calculation which is selected by the CALC bits. The value can be one of the following calculations:
  - The first derivative of single measurements
  - The CVD result when double sampling is enabled
  - The current result vs. setpoint value in the ADSTPT register
  - The current result vs. the filtered/average result
  - The first derivative of the filtered/average value
  - Filtered/average value vs. setpoint value in the ADSTPT register
- The result of the calculation (ADERR) is compared to the upper and lower thresholds, ADUTH and ADLTH registers, to set the UTHR and LTHR flag bits. The threshold logic is selected by the TMD bits. The threshold trigger option can be one of the following:
  - Never interrupt
  - Error is less than lower threshold
  - Error is greater than or equal to lower threshold
  - Error is between thresholds (inclusive)
  - Error is outside of thresholds
  - Error is less than or equal to upper threshold
  - Error is greater than upper threshold
  - Always interrupt regardless of threshold test results
  - If the Threshold condition is met, the threshold interrupt flag ADTIF is set.

**Important:**

- The threshold tests are signed operations.
- If the AOV bit is set, a threshold interrupt is signaled. It is good practice for threshold interrupt handlers to verify the validity of the threshold by checking AOV bit.

### 40.5.8 Repetition and Sampling Options

#### 40.5.8.1 Continuous Sampling Mode

Setting the CONT bit automatically retriggers a new conversion cycle after updating the ADACC register. That means the GO bit remains set to generate automatic retriggering. If SOI = 1, a Threshold Interrupt condition will clear GO bit and the conversion will stop.

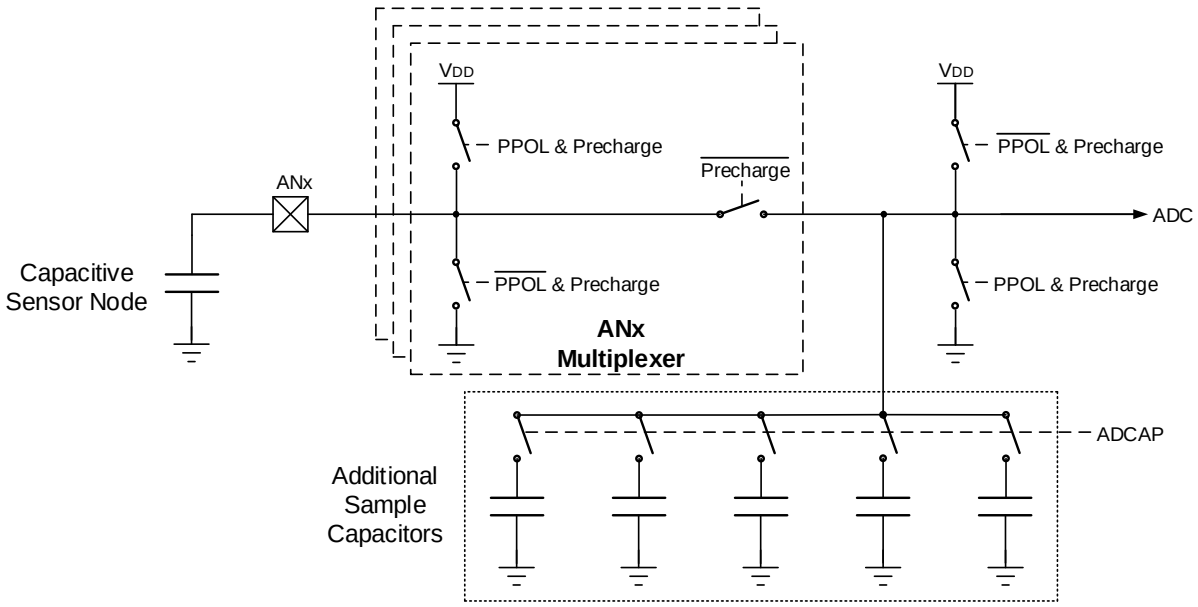
#### 40.5.8.2 Double Sample Conversion

Double sampling is enabled by setting the DSEN bit. When this bit is set, two conversions are required before the module calculates the threshold error. Each conversion must be triggered separately when CONT = 0 but will repeat automatically form a single trigger when CONT = 1. The first conversion will set the MATH bit and update ADACC, but will not calculate ADERR or trigger ADTIF. When the second conversion completes, the first value is transferred to ADPREV (depending on the setting of PSIS) and the value of the second conversion is placed into ADRES. Only upon the completion of the second conversion is ADERR calculated and ADTIF triggered (depending on the value of CALC).

## 40.6 Capacitive Voltage Divider (CVD) Features

The ADC module contains several features that allow the user to perform a relative capacitance measurement on any ADC channel using the internal ADC Sample-and-Hold capacitance as a reference. This relative capacitance measurement can be used to implement capacitive touch or proximity sensing applications. The following figure shows the basic block diagram of the CVD portion of the ADC module.

**Figure 40-7. Hardware Capacitive Voltage Divider Block Diagram**




This is an example on how to configure ADC for CVD operation:

1. Configure Port:
   a. Disable pin output driver (refer to the TRISx register)
   b. Configure pin as analog (refer to the ANSELx register)
2. Configure the ADC module:
   a. Select ADC conversion clock
   b. Configure voltage reference
   c. Select ADC input channel
   d. Configure precharge (ADPRE) and acquisition (ADACQ) time period
   e. Select precharge polarity (PPOL)

f. Enable Double Sampling (DSEN)
g. Turn on ADC module

3. Configure ADC interrupt (optional):
   a. Clear ADC interrupt flag
   b. Enable ADC interrupt
   c. Enable global interrupt (GIE bit)(1)
4. Start double sample conversion by setting the GO bit.
5. Wait for ADC conversion to complete by one of the following:
   a. Polling the GO bit
   b. Waiting for the ADC interrupt (if interrupt is enabled)
6. Second ADC conversion depends on the state of CONT:
   a. If CONT = 1, both conversion will repeat automatically from a single trigger
   b. If CONT = 0, each conversion must be triggered separately
7. The ADERR register contains the CVD result.
8. Clear the ADC interrupt flag (if interrupt is enabled).

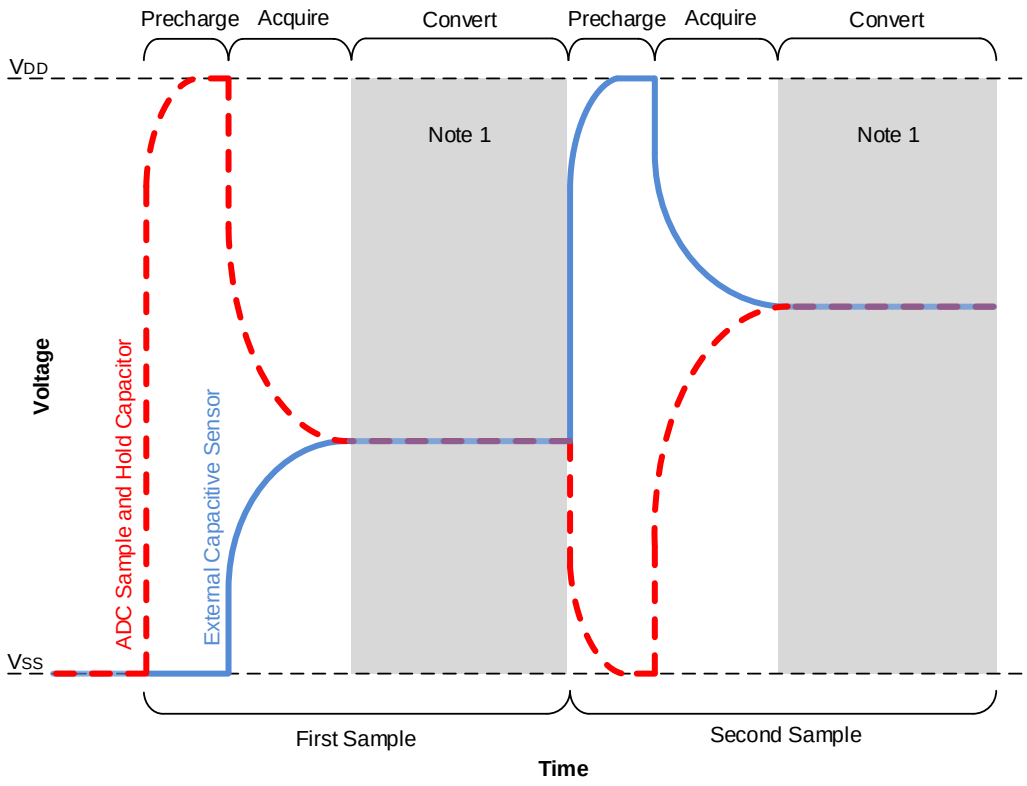
**Note:**

1. With global interrupts disabled (GIE = 0), the device will wake from Sleep but will not enter an Interrupt Service Routine.

### 40.6.1 CVD Operation

A CVD operation begins with the ADC's internal Sample-and-Hold capacitor ($C_{HOLD}$) being disconnected from the path which connects it to the external capacitive sensor node. While disconnected, $C_{HOLD}$ is precharged to $V_{DD}$ or discharged to $V_{SS}$. The sensor node is either discharged or charged to $V_{SS}$ or $V_{DD}$, respectively to the opposite level of $C_{HOLD}$. When the precharge phase is complete, the $V_{DD}/V_{SS}$ bias paths for the two nodes are disconnected and the paths between $C_{HOLD}$ and the external sensor node is reconnected, at which time the acquisition phase of the CVD operation begins. During acquisition, a capacitive voltage divider is formed between the precharged $C_{HOLD}$ and sensor nodes, which results in a final voltage level setting on $C_{HOLD}$ which is determined by the capacitances and precharge levels of the two nodes. After acquisition, the ADC converts the voltage level on $C_{HOLD}$. This process is then repeated with the selected precharge levels inverted for both the $C_{HOLD}$ and the sensor nodes. The waveform for two CVD measurements, which is known as differential CVD measurement, is shown in the following figure.

**Figure 40-8. Differential CVD Measurement Waveform**




**Note 1:** **External Capacitive Sensor voltage during the conversion phase may vary as per the configuration of the corresponding pin.**

### 40.6.2 Precharge Control

The Precharge stage is the period of time that brings the external channel and internal Sample-and-Hold capacitor to known voltage levels. Precharge is enabled by writing a nonzero value to the ADPRE register. This stage is initiated when an ADC conversion begins, either from setting the GO bit, a Special Event Trigger, or a conversion restart from the computation functionality. If the ADPRE register is cleared when an ADC conversion begins, this stage is skipped.

During the precharge time, $C_{HOLD}$ is disconnected from the outer portion of the sample path that leads to the external capacitive sensor and is connected to either $V_{DD}$ or $V_{SS}$, depending on the value of the PPOL bit. At the same time, the PORT pin logic of the selected analog channel is overridden to drive a digital high or low out, to precharge the outer portion of the ADC's sample path, which includes the external sensor. The output polarity of this override is determined by the PPOL bit such that the external sensor cap is charged opposite of the internal $C_{HOLD}$ cap. The amount of time for precharge is controlled by the ADPRE register.

**Important:** The external charging overrides the TRIS/LAT/Guard outputs setting of the respective I/O pin. If there is a device attached to this pin, precharge will not be used.

### 40.6.3 Acquisition Control for CVD (ADPRE > 0)

The Acquisition stage allows time for the voltage on the internal Sample-and-Hold capacitor to charge or discharge from the selected analog channel. This acquisition time is controlled by the ADACQ register. The acquisition stage begins when precharge stage ends.

At the start of the acquisition stage, the PORT pin logic of the selected analog channel is overridden to turn off the digital high/low output drivers so they do not affect the final result of the charge averaging. Also, the selected ADC

channel is connected to $C_{HOLD}$. This allows charge averaging to proceed between the precharged channel and the $C_{HOLD}$ capacitor.

**Important:** When ADPRE > 0 setting ADACQ to '0' will set a maximum acquisition time. When precharge is disabled, setting ADACQ to '0' will disable hardware acquisition time control.

### 40.6.4 Guard Ring Outputs

Figure 40-9 shows a typical guard ring circuit. $C_{GUARD}$ represents the capacitance of the guard ring trace placed on the PCB. The user selects values for $R_A$ and $R_B$ that will create a voltage profile on $C_{GUARD}$, which will match the selected acquisition channel.

The purpose of the guard ring is to generate a signal in phase with the CVD sensing signal to minimize the effects of the parasitic capacitance on sensing electrodes. It also can be used as a mutual drive for mutual capacitive sensing. For more information about active guard and mutual drive, refer to the following Microchip application note, available on the corporate website (www.microchip.com):

- AN1478, *"mTouch™ Sensing Solution Acquisition Methods Capacitive Voltage Divider"*

The ADC has two guard ring drive outputs, ADGRDA and ADGRDB. These outputs are routed through PPS controls to I/O pins. Refer to the **"PPS - Peripheral Pin Select Module"** chapter for more details. The polarity of these outputs is controlled by the GPOL and IPEN bits.

At the start of the first precharge stage, both outputs are set to match the GPOL bit. Once the acquisition stage begins, ADGRDA changes polarity, while ADGRDB remains unchanged. When performing a double sample conversion, setting the IPEN bit causes both guard ring outputs to transition to the opposite polarity of GPOL at the start of the second precharge stage, and ADGRDA toggles again for the second acquisition. For more information on the timing of the guard ring output, refer to Figure 40-10.

**Figure 40-9. Guard Ring Circuit**

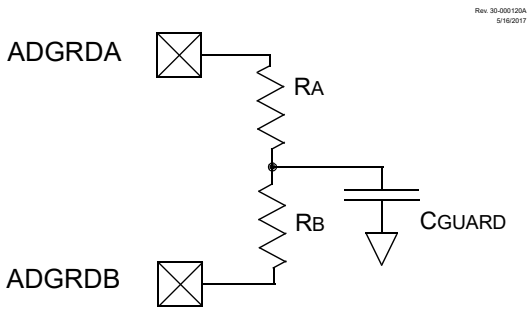
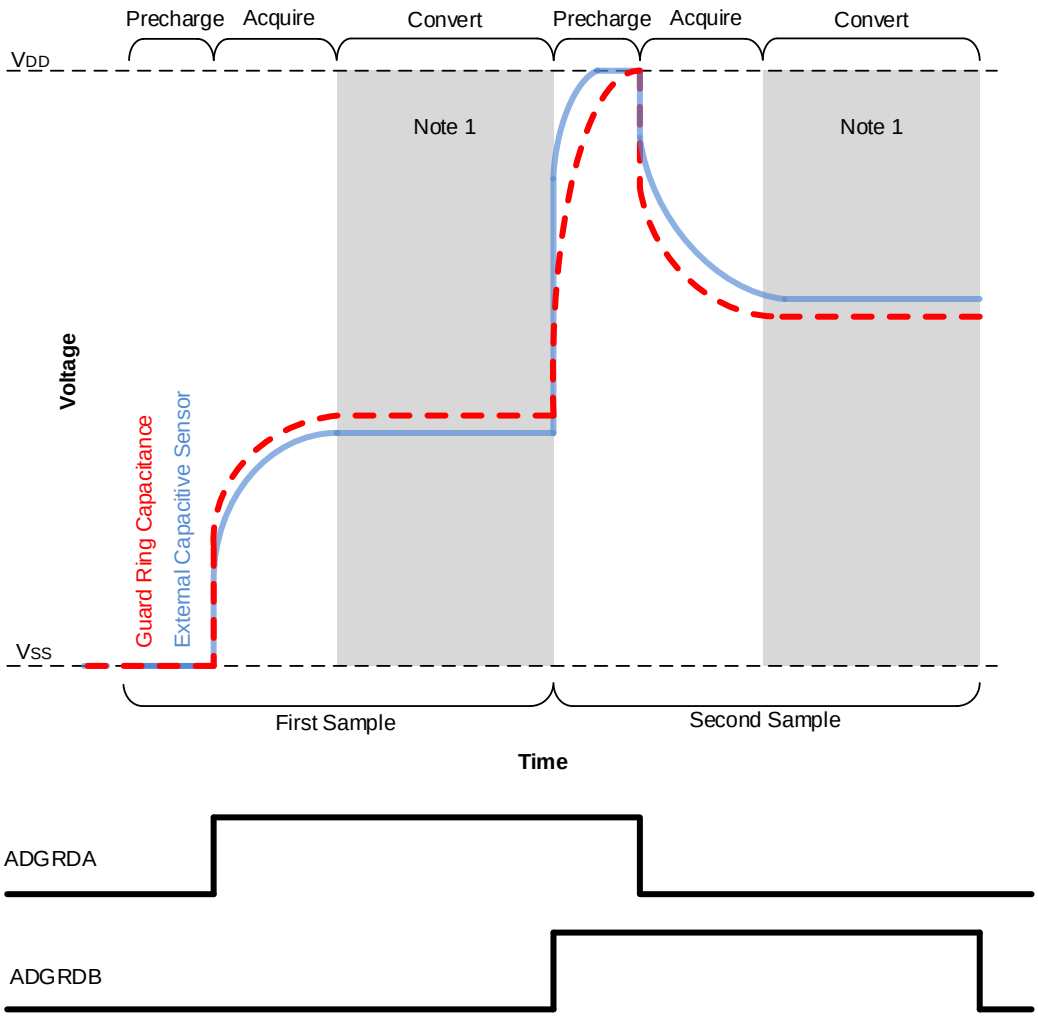


**Figure 40-10. Differential CVD with Guard Ring Output Waveform**




**Note 1:** **External Capacitive Sensor voltage during the conversion phase may vary as per the configuration of the corresponding pin.**

### 40.6.5 Additional Sample-and-Hold Capacitance

Additional capacitance can be added in parallel with the internal Sample-and-Hold capacitor ($C_{HOLD}$) by using the ADCAP register. This register selects a digitally programmable capacitance that is added to the ADC conversion bus, increasing the effective internal capacitance of the Sample-and-Hold capacitor in the ADC module. This is used to improve the match between internal and external capacitance for a better sensing performance. The additional capacitance does not affect analog performance of the ADC because it is not connected during conversion.

## 40.7 Register Definitions: ADC Control

Long bit name prefixes for the ADC peripherals are shown in the following table. Refer to the **"Long Bit Names"** section of the **"Register and Bit Naming Conventions"** chapter for more information.

**Table 40-4. ADC Long Bit Name Prefixes**

| Peripheral | Bit Name Prefix |
|---|---|
| ADC | AD |

### 40.7.1 ADCON0

**Name:** ADCON0
**Address:** 0x3F3

ADC Control Register 0

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | ON | CONT | | CS | | FM | | GO |
| Access | R/W | R/W | | R/W | | R/W | | R/W/HC/HS |
| Reset | 0 | 0 | | 0 | | 0 | | 0 |

**Bit 7 – ON** ADC Enable

| Value | Description |
|---|---|
| 1 | ADC is enabled |
| 0 | ADC is disabled |

**Bit 6 – CONT** ADC Continuous Operation Enable

| Value | Description |
|---|---|
| 1 | GO is retriggered upon completion of each conversion trigger until ADTIF is set (if SOI is set) or until GO is cleared (regardless of the value of SOI) |
| 0 | ADC is cleared upon completion of each conversion trigger |

**Bit 4 – CS** ADC Clock Selection

| Value | Description |
|---|---|
| 1 | Clock supplied from ADCRC dedicated oscillator |
| 0 | Clock supplied by $F_{OSC}$, divided according to ADCLK register |

**Bit 2 – FM** ADC Results Format/Alignment Selection

| Value | Description |
|---|---|
| 1 | ADRES and ADPREV data are right justified |
| 0 | ADRES and ADPREV data are left justified, zero-filled |

**Bit 0 – GO** ADC Conversion Status(1,2)

| Value | Description |
|---|---|
| 1 | ADC conversion cycle in progress. Setting this bit starts an ADC conversion cycle. The bit is cleared by hardware as determined by the CONT bit |
| 0 | ADC conversion completed/not in progress |

**Notes:**

1. This bit requires ON bit to be set.
2. If cleared by software while a conversion is in progress, the results of the conversion up to this point will be transferred to ADRES and the state machine will be reset, but the ADIF Interrupt Flag bit will not be set; filter and threshold operations will not be performed.

### 40.7.2 ADCON1

**Name:** ADCON1
**Address:** 0x3F4

ADC Control Register 1

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | PPOL | IPEN | GPOL | | | | | DSEN |
| Access | R/W | R/W | R/W | | | | | R/W |
| Reset | 0 | 0 | 0 | | | | | 0 |

**Bit 7 – PPOL** Precharge Polarity
Action During 1st Precharge Stage

| Value | Condition | Description |
|---|---|---|
| x | ADPRE = 0 | Bit has no effect |
| 1 | ADPRE > 0 | External analog I/O pin is connected to $V_{DD}$.<br>Internal AD sampling capacitor ($C_{HOLD}$) is connected to $V_{SS}$. |
| 0 | ADPRE > 0 | External analog I/O pin is connected to $V_{SS}$.<br>Internal AD sampling capacitor ($C_{HOLD}$) is connected to $V_{DD}$. |

**Bit 6 – IPEN** A/D Inverted Precharge Enable

| Value | Condition | Description |
|---|---|---|
| x | DSEN = 0 | Bit has no effect |
| 1 | DSEN = 1 | The precharge and guard signals in the second conversion cycle are the opposite polarity of the first cycle |
| 0 | DSEN = 1 | Both conversion cycles use the precharge and guards specified by PPOL and GPOL |

**Bit 5 – GPOL** Guard Ring Polarity Selection

| Value | Description |
|---|---|
| 1 | ADC guard Ring outputs start as digital high during Precharge stage |
| 0 | ADC guard Ring outputs start as digital low during Precharge stage |

**Bit 0 – DSEN** Double-Sample Enable

| Value | Description |
|---|---|
| 1 | Two conversions are processed as a pair. The selected computation is performed after every second conversion. |
| 0 | Selected computation is performed after every conversion |

© 2019-2023 Microchip Technology Inc. and its subsidiaries

### 40.7.3 ADCON2

**Name:** ADCON2
**Address:** 0x3F5

ADC Control Register 2

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | PSIS | CRS[2:0] | | | ACLR | MD[2:0] | | |
| Access | R/W | R/W | R/W | R/W | R/W/HC | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bit 7 – PSIS** ADC Previous Sample Input Select

| Value | Description |
|---|---|
| 1 | ADFLTR is transferred to ADPREV at start-of-conversion |
| 0 | ADRES is transferred to ADPREV at start-of-conversion |

**Bits 6:4 – CRS[2:0]** ADC Accumulated Calculation Right Shift Select

| Value | Condition | Description |
|---|---|---|
| 1 to 6 | MD = 'b100 | Low-pass filter time constant is $2^{CRS}$, filter gain is 1:1(2) |
| 1 to 6 | MD = 'b011 to 'b001 | The accumulated value is right-shifted by CRS (divided by $2^{CRS}$)(1,2) |
| x | MD = 'b000 | These bits are ignored |

**Bit 3 – ACLR** A/D Accumulator Clear Command(3)

| Value | Description |
|---|---|
| 1 | The ADACC and ADCNT registers and the AOV bit are cleared |
| 0 | Clearing action is complete (or not started) |

**Bits 2:0 – MD[2:0]** ADC Operating Mode Selection(4)

| Value | Description |
|---|---|
| 111-101 | Reserved |
| 100 | Low-Pass Filter mode |
| 011 | Burst Average mode |
| 010 | Average mode |
| 001 | Accumulate mode |
| 000 | Basic (Legacy) mode |

**Notes:**

1. To correctly calculate an average, the number of samples (set in ADRPT) must be $2^{CRS}$.
2. CRS = 'b111 and 'b000 are reserved.
3. This bit is cleared by hardware when the accumulator operation is complete; depending on oscillator selections, the delay may be many instructions.
4. See the Computation Operation section for full mode descriptions.

### 40.7.4 ADCON3

**Name:** ADCON3
**Address:** 0x3F6

ADC Control Register 3

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | CALC[2:0] | | | SOI | TMD[2:0] | | |
| Access | | R/W | R/W | R/W | R/W/HC | R/W | R/W | R/W |
| Reset | | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 6:4 – CALC[2:0]** ADC Error Calculation Mode Select

| CALC | ADERR | | Application |
|---|---|---|---|
| | DSEN = 0 Single-Sample Mode | DSEN = 1 CVD Double-Sample Mode(1) | |
| 111 | Reserved | Reserved | Reserved |
| 110 | Reserved | Reserved | Reserved |
| 101 | ADFLTR-ADSTPT | ADFLTR-ADSTPT | Average/filtered value vs. setpoint |
| 100 | ADPREV-ADFLTR | ADPREV-ADFLTR | First derivative of filtered value(3) (negative) |
| 011 | Reserved | Reserved | Reserved |
| 010 | ADRES-ADFLTR | (ADRES-ADPREV)-ADFLTR | Actual result vs. averaged/filtered value |
| 001 | ADRES-ADSTPT | (ADRES-ADPREV)-ADSTPT | Actual result vs. setpoint |
| 000 | ADRES-ADPREV | ADRES-ADPREV | First derivative of single measurement(2) |
| | | | Actual CVD result(2) |

**Notes:**
1. When DSEN = 1 and PSIS = 0, ADERR is computed only after every second sample.
2. When PSIS = 0.
3. When PSIS = 1.

**Bit 3 – SOI** ADC Stop-on-Interrupt

| Value | Condition | Description |
|---|---|---|
| x | CONT = 0 | This bit is not used |
| 1 | CONT = 1 | GO is cleared when the threshold conditions are met, otherwise the conversion is retriggered |
| 0 | CONT = 1 | GO is not cleared by hardware, must be cleared by software to stop retriggers |

**Bits 2:0 – TMD[2:0]** Threshold Interrupt Mode Select

| Value | Description |
|---|---|
| 111 | Interrupt regardless of threshold test results |
| 110 | Interrupt if ADERR > ADUTH |
| 101 | Interrupt if ADERR ≤ ADUTH |
| 100 | Interrupt if ADERR < ADLTH or ADERR > ADUTH |
| 011 | Interrupt if ADERR > ADLTH and ADERR < ADUTH |
| 010 | Interrupt if ADERR ≥ ADLTH |
| 001 | Interrupt if ADERR < ADLTH |
| 000 | Never interrupt |

© 2019-2023 Microchip Technology Inc. and its subsidiaries

### 40.7.5 ADSTAT

**Name:** ADSTAT
**Address:** 0x3F7

ADC Status Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | AOV | UTHR | LTHR | MATH | | STAT[2:0] | | |
| Access | R/C/HS/HC | R | R | R/C/HS | | R | R | R |
| Reset | 0 | 0 | 0 | 0 | | 0 | 0 | 0 |

**Bit 7 – AOV** ADC Accumulator Overflow

| Value | Description |
|---|---|
| 1 | The ADACC or ADFLTR or ADERR registers have overflowed |
| 0 | The ADACC, ADFLTR and ADERR registers have not overflowed |

**Bit 6 – UTHR** ADC Module Greater-than Upper Threshold Flag

| Value | Description |
|---|---|
| 1 | ADERR > ADUTH |
| 0 | ADERR ≤ ADUTH |

**Bit 5 – LTHR** ADC Module Less-than Lower Threshold Flag

| Value | Description |
|---|---|
| 1 | ADERR < ADLTH |
| 0 | ADERR ≥ ADLTH |

**Bit 4 – MATH** ADC Module Computation Status

ADC Module Computation Status(1)

| Value | Description |
|---|---|
| 1 | The ADACC, ADFLTR, ADUTH and ADLTH registers and the AOV bit are updating or have already updated |
| 0 | Associated registers/bits have not changed since this bit was last cleared |

**Bits 2:0 – STAT[2:0]** ADC Module Cycle Multi-Stage Status

| Value | Description |
|---|---|
| 111 | ADC module is in 2nd conversion stage |
| 110 | ADC module is in 2nd acquisition stage |
| 101 | ADC module is in 2nd precharge stage |
| 100 | ADC computation is suspended between 1st and 2nd sample; the computation results are incomplete and awaiting data from the 2nd sample(2,3) |
| 011 | ADC module is in 1st conversion stage |
| 010 | ADC module is in 1st acquisition stage |
| 001 | ADC module is in 1st precharge stage |
| 000 | ADC module is not converting |

**Notes:**

1. MATH bit cannot be cleared by software while STAT = 'b100.
2. If ADC clock source is ADCRC, and $F_{OSC}$ < ADCRC, the indicated status may not be valid.
3. STAT = 'b100 appears between the two triggers when DSEN = 1 and CONT = 0.

### 40.7.6 ADCLK

**Name:** ADCLK
**Address:** 0x3FA

ADC Clock divider Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | CS[5:0] | | | | | |
| Access | | | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | | | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 5:0 – CS[5:0]** ADC Clock divider Select

| Value | Description |
|---|---|
| n | ADC Clock frequency = $F_{OSC}/(2*(n+1))$ |

**Note:** ADC Clock divider is only available if $F_{OSC}$ is selected as the ADC clock source (CS = 0).

### 40.7.7 ADREF

**Name:** ADREF
**Address:** 0x3F8

ADC Reference Selection Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | NREF | | | PREF[1:0] | |
| Access | | | | R/W | | | R/W | R/W |
| Reset | | | | 0 | | | 0 | 0 |

**Bit 4 – NREF** ADC Negative Voltage Reference Selection

| Value | Description |
|---|---|
| 1 | $V_{REF}$- is connected to external $V_{REF}$- |
| 0 | $V_{REF}$- is connected to $AV_{SS}$ |

**Bits 1:0 – PREF[1:0]** ADC Positive Voltage Reference Selection

| Value | Description |
|---|---|
| 11 | $V_{REF}$+ is connected to internal Fixed Voltage Reference (FVR) module |
| 10 | $V_{REF}$+ is connected to external $V_{REF}$+ |
| 01 | Reserved |
| 00 | $V_{REF}$+ is connected to $V_{DD}$ |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

### 40.7.8 ADPCH

**Name:** ADPCH
**Address:** 0x3EC

ADC Positive Channel Selection Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | PCH[5:0] | | | | | |
| Access | | | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | | | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 5:0 – PCH[5:0]** ADC Positive Input Channel Selection

| PCH | ADC Positive Channel Input |
|---|---|
| 111111 | Fixed Voltage Reference (FVR) Buffer 2(1) |
| 111110 | Fixed Voltage Reference (FVR) Buffer 1(1) |
| 111101 | DAC1 output(2) |
| 111100 | Temperature Indicator(3) |
| 111011 | $V_{SS}$ (Analog Ground) |
| 111010-110000 | Reserved. No channel connected. |
| 101111 | RF7/ANF7(5) |
| 101110 | RF6/ANF6(5) |
| 101101 | RF5/ANF5(5) |
| 101100 | RF4/ANF4(5) |
| 101011 | RF3/ANF3(5) |
| 101010 | RF2/ANF2(5) |
| 101001 | RF1/ANF1(5) |
| 101000 | RF0/ANF0(5) |
| 100111-100011 | Reserved. No channel connected. |
| 100010 | RE2/ANE2(4) |
| 100001 | RE1/ANE1(4) |
| 100000 | RE0/ANE0(4) |
| 011111 | RD7/AND7(4) |
| 011110 | RD6/AND6(4) |
| 011101 | RD5/AND5(4) |
| 011100 | RD4/AND4(4) |
| 011011 | RD3/AND3(4) |
| 011010 | RD2/AND2(4) |
| 011001 | RD1/AND1(4) |
| 011000 | RD0/AND0(4) |
| 010111 | RC7/ANC7 |
| 010110 | RC6/ANC6 |
| 010101 | RC5/ANC5 |
| 010100 | RC4/ANC4 |
| 010011 | RC3/ANC3 |
| 010010 | RC2/ANC2 |
| 010001 | RC1/ANC1 |
| 010000 | RC0/ANC0 |
| 001111 | RB7/ANB7 |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

.....continued

| PCH | ADC Positive Channel Input |
|---|---|
| 001110 | RB6/ANB6 |
| 001101 | RB5/ANB5 |
| 001100 | RB4/ANB4 |
| 001011 | RB3/ANB3 |
| 001010 | RB2/ANB2 |
| 001001 | RB1/ANB1 |
| 001000 | RB0/ANB0 |
| 000111 | RA7/ANA7 |
| 000110 | RA6/ANA6 |
| 000101 | RA5/ANA5 |
| 000100 | RA4/ANA4 |
| 000011 | RA3/ANA3 |
| 000010 | RA2/ANA2 |
| 000001 | RA1/ANA1 |
| 000000 | RA0/ANA0 |

**Notes:**

1. Refer to the **"Fixed Voltage Reference Module"** chapter for more details.
2. Refer to the **"Digital-to-Analog Converter Module"** chapter for more details.
3. Refer to the **"Temperature Indicator Module"** chapter for more details.
4. 40/44/48-pin devices only.
5. 48-pin devices only.

### 40.7.9 ADPRE

**Name:** ADPRE
**Address:** 0x3F1

ADC Precharge Time Control Register

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | | | | PRE[12:8] | | | | |
| Access | | | | R/W | R/W | R/W | R/W | R/W |
| Reset | | | | 0 | 0 | 0 | 0 | 0 |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | PRE[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 12:0 – PRE[12:0]** Precharge Time Select

| PRE | Precharge Time | |
|---|---|---|
| | CS = 0 | CS = 1 |
| 1 1111 1111 1111 | 8191 clocks of $F_{OSC}$ | 8191 clocks of ADCRC |
| 1 1111 1111 1110 | 8190 clocks of $F_{OSC}$ | 8190 clocks of ADCRC |
| 1 1111 1111 1101 | 8189 clocks of $F_{OSC}$ | 8189 clocks of ADCRC |
| ... | ... | ... |
| 0 0000 0000 0010 | 2 clocks of $F_{OSC}$ | 2 clocks of ADCRC |
| 0 0000 0000 0001 | 1 clocks of $F_{OSC}$ | 1 clocks of ADCRC |
| 0 0000 0000 0000 | Not included in the data conversion cycle | |

**Notes:** The individual bytes in this multibyte register can be accessed with the following register names:
1. ADPREH: Accesses the high byte ADPRE[12:8].
2. ADPREL: Accesses the low byte ADPRE[7:0].

### 40.7.10 ADACQ

**Name:** ADACQ
**Address:** 0x3EE

ADC Acquisition Time Control Register

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | | | | ACQ[12:8] | | | | |
| Access | | | | R/W | R/W | R/W | R/W | R/W |
| Reset | | | | 0 | 0 | 0 | 0 | 0 |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | ACQ[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 12:0 – ACQ[12:0]** Acquisition (charge share time) Select

| ACQ | Acquisition Time | |
|---|---|---|
| | CS = 0 | CS = 1 |
| 1 1111 1111 1111 | 8191 clocks of $F_{OSC}$ | 8191 clocks of ADCRC |
| 1 1111 1111 1110 | 8190 clocks of $F_{OSC}$ | 8190 clocks of ADCRC |
| 1 1111 1111 1101 | 8189 clocks of $F_{OSC}$ | 8189 clocks of ADCRC |
| ... | ... | ... |
| 0 0000 0000 0010 | 2 clocks of $F_{OSC}$ | 2 clocks of ADCRC |
| 0 0000 0000 0001 | 1 clocks of $F_{OSC}$ | 1 clocks of ADCRC |
| 0 0000 0000 0000 | Not included in the data conversion cycle(1) | |

**Note:**
1. If ADPRE is not equal to '0', then ACQ = 0 means Acquisition Time is 8192 clocks of $F_{OSC}$ or ADCRC.

**Notes:** The individual bytes in this multibyte register can be accessed with the following register names:
- ADACQH: Accesses the high byte ADACQ[12:8]
- ADACQL: Accesses the low byte ADACQ[7:0]

### 40.7.11 ADCAP

**Name:** ADCAP
**Address:** 0x3F0

ADC Additional Sample Capacitor Selection Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | CAP[4:0] | | | | |
| Access | | | | R/W | R/W | R/W | R/W | R/W |
| Reset | | | | 0 | 0 | 0 | 0 | 0 |

**Bits 4:0 – CAP[4:0]** ADC Additional Sample Capacitor Selection

| Value | Description |
|---|---|
| 1 to 31 | Number of pF in the additional capacitance |
| 0 | No additional capacitance |

### 40.7.12 ADRPT

**Name:** ADRPT
**Address:** 0x3E7

ADC Repeat Setting Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | RPT[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 7:0 – RPT[7:0]** ADC Repeat Threshold
Determines the number of times the ADC is triggered for a threshold check. When CNT reaches this value, the error threshold is checked. Used when the computation mode is Low-Pass Filter, Burst Average, or Average. See the Computation Operation section for more details.

### 40.7.13 ADCNT

**Name:** ADCNT
**Address:** 0x3E6

ADC Repeat Counter Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | CNT[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 7:0 – CNT[7:0]** ADC Repeat Count
Counts the number of times the ADC is triggered before the threshold is checked. When this value reaches RPT, the threshold is checked. Used when the computation mode is Low-Pass Filter, Burst Average, or Average. See the Computation Operation section for more details.

### 40.7.14 ADFLTR

**Name:** ADFLTR
**Address:** 0x3E1

ADC Filter Register

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | FLTR[15:8] | | | | | | | |
| Access | R | R | R | R | R | R | R | R |
| Reset | x | x | x | x | x | x | x | x |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | FLTR[7:0] | | | | | | | |
| Access | R | R | R | R | R | R | R | R |
| Reset | x | x | x | x | x | x | x | x |

**Bits 15:0 – FLTR[15:0]** ADC Filter Output - Signed Two's Complement
In Accumulate, Average and Burst Average modes, this is equal to ACC right shifted by the CRS bits. In LPF mode, this is the output of the Low-Pass Filter.

**Notes:** The individual bytes in this multibyte register can be accessed with the following register names:

- ADFLTRH: Accesses the high byte ADFLTR[15:8]
- ADFLTRL: Accesses the low byte ADFLTR[7:0]

### 40.7.15 ADRES

**Name:** ADRES
**Address:** 0x3EA

ADC Result Register

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | RES[15:8] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | RES[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 15:0 – RES[15:0]** ADC Sample Result

**Notes:** The individual bytes in this multibyte register can be accessed with the following register names:

- ADRESH: Accesses the high byte ADRES[15:18]
- ADRESL: Accesses the low byte ADRES[7:0]

### 40.7.16 ADPREV

**Name:** ADPREV
**Address:** 0x3E8

ADC Previous Result Register

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | PREV[15:8] | | | | | | | |
| Access | R | R | R | R | R | R | R | R |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | PREV[7:0] | | | | | | | |
| Access | R | R | R | R | R | R | R | R |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 15:0 – PREV[15:0]** Previous ADC Result

| Value | Condition | Description |
|---|---|---|
| n | PSIS = 1 | n = ADFLTR value at the start of current ADC conversion |
| n | PSIS = 0 | n = ADRES at the start of current ADC conversion(1) |

**Notes:**

1. If PSIS = 0, ADPREV is formatted the same way as ADRES is, depending on the FM bit.
2. The individual bytes in this multibyte register can be accessed with the following register names:
   – ADPREVH: Accesses ADPREV[15:8]
   – ADPREVL: Accesses ADPREV[7:0]

### 40.7.17 ADACC

**Name:** ADACC
**Address:** 0x3E3

ADC Accumulator Register[1]

See the Computation Operation section for more details.

**Important:** This register contains signed two's complement accumulator value and the upper unused bits contain copies of the sign bit.

| Bit | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | ACC[17:16] | |
| Access | | | | | | | R/W | R/W |
| Reset | | | | | | | x | x |

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | ACC[15:8] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | x | x | x | x | x | x | x | x |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | ACC[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | x | x | x | x | x | x | x | x |

**Bits 17:0 – ACC[17:0]** ADC Accumulator - Signed Two's Complement

**Notes:**

1. This register can only be written when GO = 0.
2. The individual bytes in this multibyte register can be accessed with the following register names:
   – ADACCU: Accesses the upper byte ADACC[17:16]
   – ADACCH: Accesses the high byte ADACC[15:8]
   – ADACCL: Accesses the low byte ADACC[7:0]

### 40.7.18 ADSTPT

**Name:** ADSTPT
**Address:** 0x3DF

ADC Threshold Setpoint Register
Depending on CALC, may be used to determine ADERR.

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | STPT[15:8] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | STPT[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 15:0 – STPT[15:0]** ADC Threshold Setpoint - Signed Two's Complement

**Notes:** The individual bytes in this multibyte register can be accessed with the following register names:

- ADSTPTH: Accesses the high byte ADSTPT[15:8]
- ADSTPTH: Accesses the low byte ADSTPT[7:0]

### 40.7.19 ADERR

**Name:** ADERR
**Address:** 0x3DD

ADC Setpoint Error Register
ADC Setpoint Error calculation is determined by the CALC bits.

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | ERR[15:8] | | | | | | | |
| Access | R | R | R | R | R | R | R | R |
| Reset | x | x | x | x | x | x | x | x |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | ERR[7:0] | | | | | | | |
| Access | R | R | R | R | R | R | R | R |
| Reset | x | x | x | x | x | x | x | x |

**Bits 15:0 – ERR[15:0]** ADC Setpoint Error - Signed Two's Complement

**Notes:** The individual bytes in this multibyte register can be accessed with the following register names:

- ADERRH: Accesses the high byte ADERR[15:8]
- ADERRL: Accesses the low byte ADERR[7:0]

### 40.7.20 ADLTH

**Name:** ADLTH
**Address:** 0x3D9

ADC Lower Threshold Register
ADLTH and ADUTH are compared with ADERR to set the UTHR and LTHR bits. Depending on the setting of TMD, an interrupt may be triggered by the results of this comparison.

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | LTH[15:8] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | LTH[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 15:0 – LTH[15:0]** ADC Lower Threshold - Signed Two's Complement

**Notes:** The individual bytes in this multibyte register can be accessed with the following register names:

- ADLTHH: Accesses the high byte ADLTH[15:8]
- ADLTHL: Accesses the low byte ADLTH[7:0]

### 40.7.21 ADUTH

**Name:** ADUTH
**Address:** 0x3DB

ADC Upper Threshold Register
ADLTH and ADUTH are compared with ADERR to set the UTHR and LTHR bits. Depending on the setting of TMD, an interrupt may be triggered by the results of this comparison.

| Bit | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
|---|---|---|---|---|---|---|---|---|
| | UTH[15:8] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | UTH[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 15:0 – UTH[15:0]** ADC Upper Threshold - Signed Two's Complement

**Notes:** The individual bytes in this multibyte register can be accessed with the following register names:

- ADUTHH: Accesses the high byte ADUTH[15:8]
- ADUTHL: Accesses the low byte ADUTH[7:0]

### 40.7.22 ADACT

**Name:** ADACT
**Address:** 0x3F9

ADC Auto Conversion Trigger Source Selection Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | ACT[5:0] | | | | | |
| Access | | | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | | | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 5:0 – ACT[5:0]** Auto-Conversion Trigger Select

| ACT | Auto-conversioin Trigger Source |
|---|---|
| 111111 | Software write to ADPCH |
| 111110 | Software read of ADRESH |
| 111101 | Software read of ADERRH |
| 111100-101000 | Reserved |
| 100111 | CLC8_OUT |
| 100110 | CLC7_OUT |
| 100101 | CLC6_OUT |
| 100100 | CLC5_OUT |
| 100011 | CLC4_OUT |
| 100010 | CLC3_OUT |
| 100001 | CLC2_OUT |
| 100000 | CLC1_OUT |
| 011111 | Interrupt-on-change Interrupt Flag |
| 011110 | CMP2_OUT |
| 011101 | CMP1_OUT |
| 011100 | NCO3_OUT |
| 011011 | NCO2_OUT |
| 011010 | NCO1_OUT |
| 011001-011000 | Reserved |
| 010111 | PWM3S1P2_OUT |
| 010110 | PWM3S1P1_OUT |
| 010101 | PWM2S1P2_OUT |
| 010100 | PWM2S1P1_OUT |
| 010011 | PWM1S1P2_OUT |
| 010010 | PWM1S1P1_OUT |
| 010001 | CCP3_trigger |
| 010000 | CCP2_trigger |
| 001111 | CCP1_trigger |
| 001110 | SMT1_overflow |
| 001101-001001 | Reserved |
| 001000 | TMR6_postscaled |
| 000111 | TMR5_overflow |
| 000110 | TMR4_postscaled |
| 000101 | TMR3_overflow |
| 000100 | TMR2_postscaled |

.....continued

| ACT | Auto-conversioin Trigger Source |
|---|---|
| 000011 | TMR1_overflow |
| 000010 | TMR0_overflow |
| 000001 | Pin selected by ADACTPPS |
| 000000 | External Trigger Disabled |

### 40.7.23 ADCP

**Name:** ADCP
**Address:** 0x3D8

ADC Charge Pump Control Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | CPON | | | | | | | CPRDY |
| Access | R/W | | | | | | | R |
| Reset | 0 | | | | | | | 0 |

**Bit 7 – CPON** Charge Pump On Control

| Value | Description |
|---|---|
| 1 | Charge Pump On when requested by the ADC |
| 0 | Charge Pump Off |

**Bit 0 – CPRDY** Charge Pump Ready Status

| Value | Description |
|---|---|
| 1 | Charge Pump is ready |
| 0 | Charge Pump is not ready (or never started) |

## 40.8 Register Summary - ADC

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x03D8 | ADCP | 7:0 | CPON | | | | | | | CPRDY |
| 0x03D9 | ADLTH | 7:0 | LTH[7:0] | | | | | | | |
| | | 15:8 | LTH[15:8] | | | | | | | |
| 0x03DB | ADUTH | 7:0 | UTH[7:0] | | | | | | | |
| | | 15:8 | UTH[15:8] | | | | | | | |
| 0x03DD | ADERR | 7:0 | ERR[7:0] | | | | | | | |
| | | 15:8 | ERR[15:8] | | | | | | | |
| 0x03DF | ADSTPT | 7:0 | STPT[7:0] | | | | | | | |
| | | 15:8 | STPT[15:8] | | | | | | | |
| 0x03E1 | ADFLTR | 7:0 | FLTR[7:0] | | | | | | | |
| | | 15:8 | FLTR[15:8] | | | | | | | |
| 0x03E3 | ADACC | 7:0 | ACC[7:0] | | | | | | | |
| | | 15:8 | ACC[15:8] | | | | | | | |
| | | 23:16 | | | | | | | ACC[17:16] | |
| 0x03E6 | ADCNT | 7:0 | CNT[7:0] | | | | | | | |
| 0x03E7 | ADRPT | 7:0 | RPT[7:0] | | | | | | | |
| 0x03E8 | ADPREV | 7:0 | PREV[7:0] | | | | | | | |
| | | 15:8 | PREV[15:8] | | | | | | | |
| 0x03EA | ADRES | 7:0 | RES[7:0] | | | | | | | |
| | | 15:8 | RES[15:8] | | | | | | | |
| 0x03EC | ADPCH | 7:0 | | | PCH[5:0] | | | | | |
| 0x03ED | Reserved | | | | | | | | | |
| 0x03EE | ADACQ | 7:0 | ACQ[7:0] | | | | | | | |
| | | 15:8 | | | | ACQ[12:8] | | | | |
| 0x03F0 | ADCAP | 7:0 | | | | CAP[4:0] | | | | |
| 0x03F1 | ADPRE | 7:0 | PRE[7:0] | | | | | | | |
| | | 15:8 | | | | PRE[12:8] | | | | |
| 0x03F3 | ADCON0 | 7:0 | ON | CONT | | CS | | FM | | GO |
| 0x03F4 | ADCON1 | 7:0 | PPOL | IPEN | GPOL | | | | | DSEN |
| 0x03F5 | ADCON2 | 7:0 | PSIS | | CRS[2:0] | | ACLR | | MD[2:0] | |
| 0x03F6 | ADCON3 | 7:0 | | | CALC[2:0] | | SOI | | TMD[2:0] | |
| 0x03F7 | ADSTAT | 7:0 | AOV | UTHR | LTHR | MATH | | | STAT[2:0] | |
| 0x03F8 | ADREF | 7:0 | | | | NREF | | | PREF[1:0] | |
| 0x03F9 | ADACT | 7:0 | | | ACT[5:0] | | | | | |
| 0x03FA | ADCLK | 7:0 | | | CS[5:0] | | | | | |

# 41. DAC - Digital-to-Analog Converter Module - 8-Bit

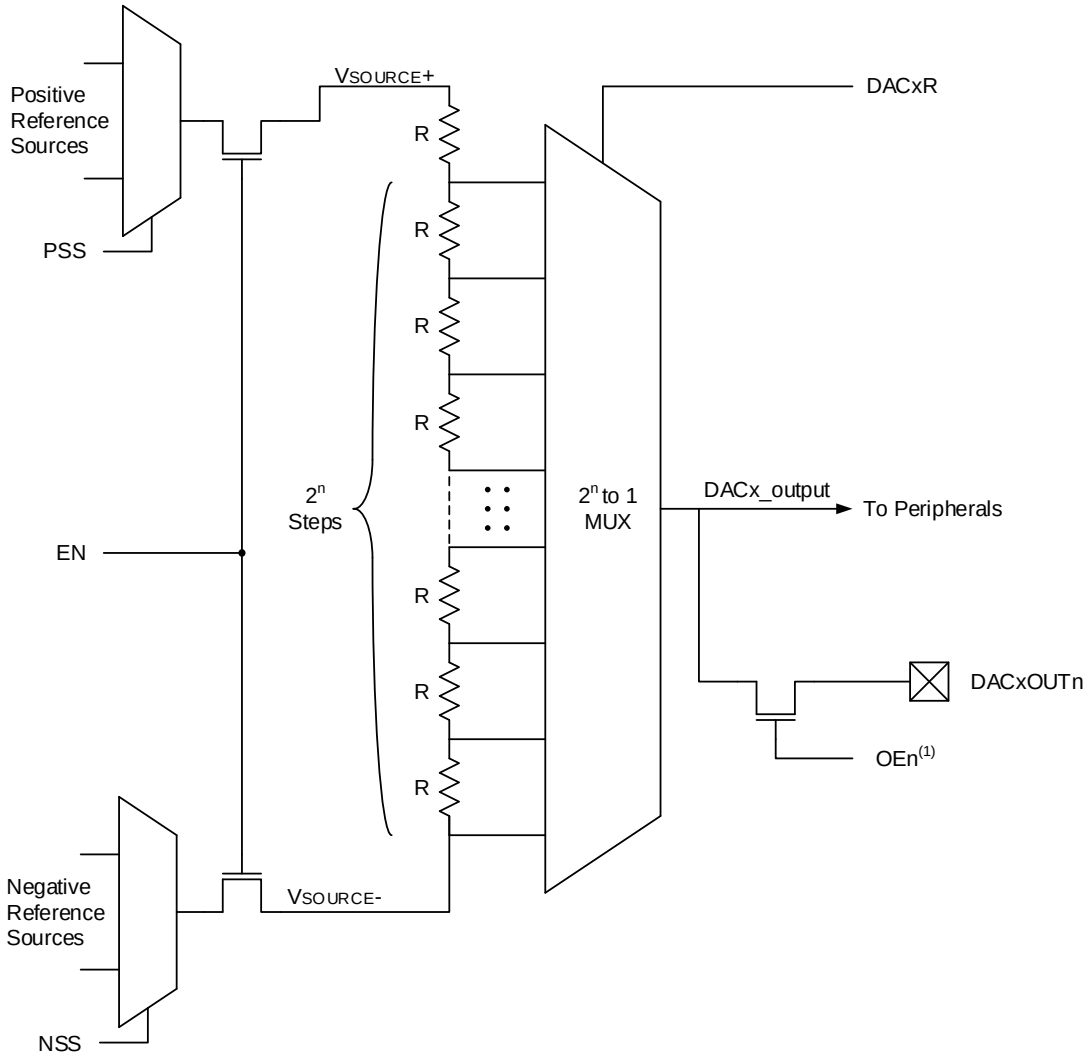
The Digital-to-Analog Converter (DAC) supplies a variable voltage reference, ratiometric with the input source, with programmable selectable output levels.

The positive and negative input references ($V_{SOURCE+}$ and $V_{SOURCE-}$) can each be selected from several sources.

The output of the DAC (DACx_output) can be selected as a reference voltage to several other peripherals or routed to output pins.

The Digital-to-Analog Converter (DAC) is enabled by setting the EN bit.

**Figure 41-1. Digital-to-Analog Converter Block Diagram**




**Note:** 1. The output enable bits are configured so that they act as a “one-hot” system, meaning only one DAC output can be enabled at a time.

## 41.1 Output Voltage Selection

The DAC has $2^n$ voltage level ranges, where n is the number of bits in DACR. Each level is determined by the DACxR bits. The DAC output voltage can be determined by using Equation 41-1.

**Equation 41-1. DAC Output Equation**

$$DACx_output = \left( (V_{REF+} - V_{REF-}) \times \frac{DACR}{2^n} \right) + V_{REF-}$$

## 41.2 Ratiometric Output Level

The DAC output value is derived using a resistor ladder with each end of the ladder tied to a positive and negative voltage reference input source. If the voltage of either input source fluctuates, a similar fluctuation will result in the DAC output value.

The value of the individual resistors within the ladder can be found in the **"Electrical Specifications"** chapter for each respective device.

## 41.3 Operation During Sleep

When the device wakes from Sleep through an interrupt or a WWDT Time-out Reset, the contents of the DACxCON and DACxDATL registers are not affected. To minimize current consumption in Sleep mode, the voltage reference will be disabled.

## 41.4 Effects of a Reset

A device Reset affects the following:

- The DAC module is disabled
- The DAC output voltage is removed from the DACxOUTn pin(s)
- The DACxR bits are cleared

## 41.5 Register Definitions: DAC Control

Long bit name prefixes for the DAC are shown in the table below. Refer to the **"Long Bit Names"** section in the **"Register and Bit Naming Conventions"** chapter for more information.

**Table 41-1. DAC Long Bit Name Prefixes**

| Peripheral | Bit Name Prefix |
|---|---|
| DAC1 | DAC1 |

### 41.5.1 DACxCON

**Name:** DACxCON
**Address:** 0x7F

Digital-to-Analog Converter Control Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | EN | | OE[1:0] | | PSS[1:0] | | | NSS |
| Access | R/W | | R/W | R/W | R/W | R/W | | R/W |
| Reset | 0 | | 0 | 0 | 0 | 0 | | 0 |

**Bit 7 – EN** DAC Enable

| Value | Description |
|---|---|
| 1 | DAC is enabled |
| 0 | DAC is disabled |

**Bits 5:4 – OE[1:0]** DAC Output Enable

| OE | DAC Outputs |
|---|---|
| 11 | DACxOUT is disabled |
| 10 | DACxOUT is enabled on pin RA2 only |
| 01 | DACxOUT is enabled on pin RB7 only |
| 00 | DACxOUT is disabled |

**Bits 3:2 – PSS[1:0]** DAC Positive Reference Selection

| PSS | DAC Positive Reference |
|---|---|
| 11 | Reserved, do not use |
| 10 | FVR Buffer 2 |
| 01 | $V_{REF}+$ |
| 00 | $V_{DD}$ |

**Bit 0 – NSS** DAC Negative Reference Selection

| NSS | DAC Negative Reference |
|---|---|
| 1 | $V_{REF}-$ |
| 0 | $V_{SS}$ |

### 41.5.2 DACxDATL

**Name:** DACxDATL
**Address:** 0x7D

Digital-to-Analog Converter Data Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | DACxR[7:0] | | | | | | | |
| Access | R/W | R/W | R/W | R/W | R/W | R/W | R/W | R/W |
| Reset | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

**Bits 7:0 – DACxR[7:0]** Data Input Bits for DAC Value

## 41.6 Register Summary - DAC

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x7D | DAC1DATL | 7:0 | DAC1R[7:0] | | | | | | | |
| 0x7E | Reserved | | | | | | | | | |
| 0x7F | DAC1CON | 7:0 | EN | | OE[1:0] | | PSS[1:0] | | | NSS |

# 42. CMP - Comparator Module

Comparators are used to interface analog circuits to a digital circuit by comparing two analog voltages and providing a digital indication of their relative magnitudes. Comparators are very useful mixed signal building blocks because they provide analog functionality independent of program execution.

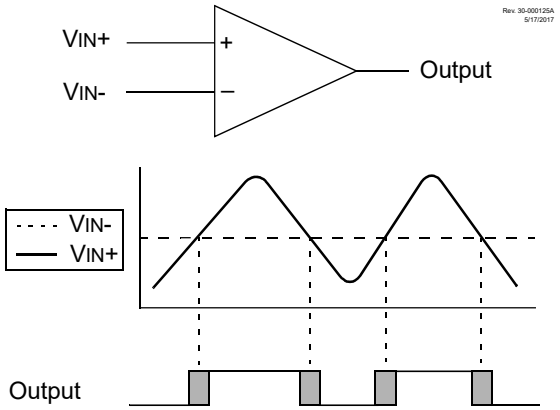
The analog comparator module includes the following features:

- Programmable input selection
- Programmable output polarity
- Rising/falling output edge interrupts
- Wake-up from Sleep
- Selectable voltage reference
- ADC auto-trigger
- Inter-connections with other available modules (e.g., timer clocks)

## 42.1 Comparator Overview

A single comparator is shown in Figure 42-1 along with the relationship between the analog input levels and the digital output. When the analog voltage at $V_{IN}$+ is less than the analog voltage at $V_{IN}$-, the output of the comparator is a digital low level. When the analog voltage at $V_{IN}$+ is greater than the analog voltage at $V_{IN}$-, the output of the comparator is a digital high level.

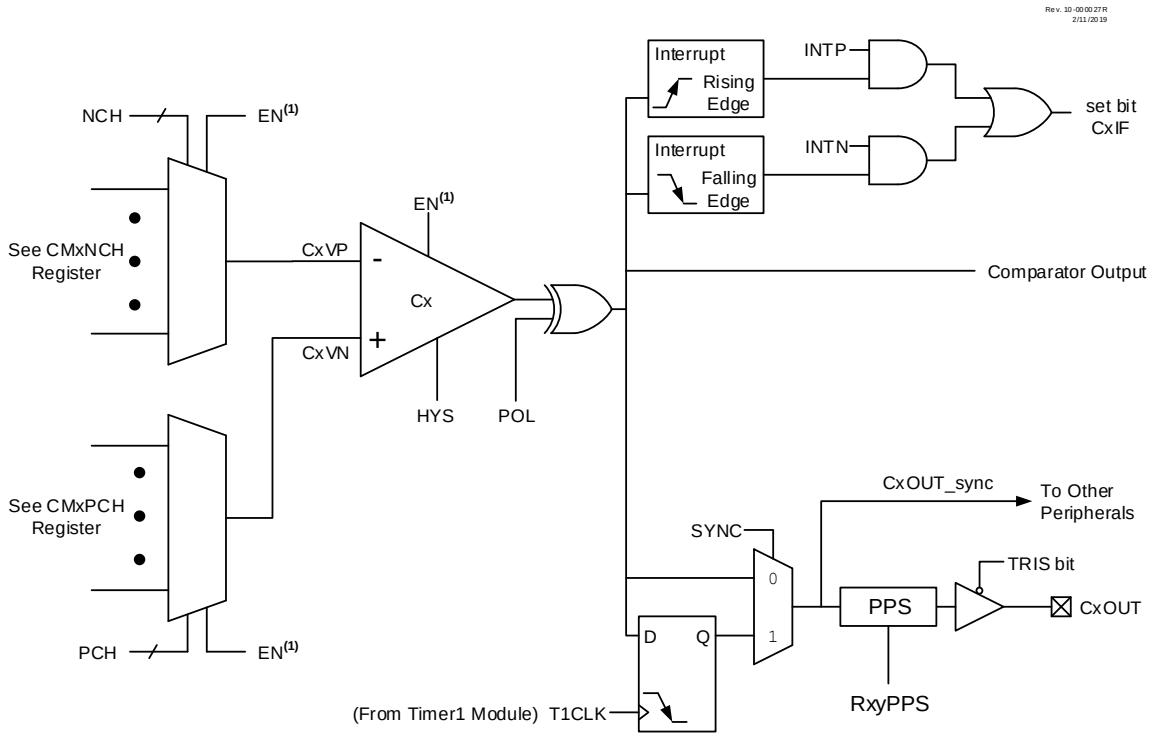
**Figure 42-1. Single Comparator**




**Note:**

1. The black areas of the output of the comparator represent the uncertainty due to input offsets and response time.

**Figure 42-2. Comparator Module Simplified Block Diagram**




**Note 1:** When EN = 0, all multiplexer inputs are disconnected and the Comparator will produce a '0' at the output.

## 42.2 Comparator Control

Each comparator has two control registers: CMxCON0 and CMxCON1.

The CMxCON0 register contains Control and Status bits for the following:

- Enable
- Output
- Output Polarity
- Hysteresis Enable
- Timer1 Output Synchronization

The CMxCON1 register contains Control bits for the following:

- Interrupt on Positive/Negative Edge Enables

The CMxPCH and CMxNCH registers are used to select the positive and negative input channels, respectively.

### 42.2.1 Comparator Enable

Setting the EN bit enables the comparator for operation. Clearing the EN bit disables the comparator, resulting in minimum current consumption.

### 42.2.2 Comparator Output

The output of the comparator can be monitored in two different registers. Each output can be read individually by reading the OUT bit. Outputs of all the comparators can be collectively accessed by reading the CMOUT register.

The comparator output can also be routed to an external pin through the RxyPPS register. Refer to the **"PPS - Peripheral Pin Select Module"** chapter for more details. The corresponding TRIS bit must be clear to enable the pin as an output.

**Important:** The internal output of the comparator is latched with each instruction cycle. Unless otherwise specified, external outputs are not latched.

### 42.2.3 Comparator Output Polarity

Inverting the output of the comparator is functionally equivalent to swapping the comparator inputs. The polarity of the comparator output can be inverted by setting the POL bit. Clearing the POL bit results in a noninverted output. Table 42-1 shows the Output state versus Input conditions, including polarity control.

**Table 42-1. Comparator Output State vs. Input Conditions**

| Input Condition | POL | OUT |
|---|---|---|
| CxVn > CxVp | 0 | 0 |
| CxVn < CxVp | 0 | 1 |
| CxVn > CxVp | 1 | 1 |
| CxVn < CxVp | 1 | 0 |

## 42.3 Comparator Output Synchronization

The output from a comparator can be synchronized with Timer1 by setting the SYNC bit.

Once enabled, the comparator output is latched on the falling edge of the Timer1 source clock. If a prescaler is used with Timer1, the comparator output is latched after the prescaling function. To prevent a Race condition, the comparator output is latched on the falling edge of the Timer1 clock source and Timer1 increments on the rising edge of its clock source. A simplified block diagram of the comparator module is shown in Figure 42-2. Refer to the **"TMR1 - Timer1 Module with Gate Control"** chapter for more details.

## 42.4 Comparator Hysteresis

A selectable amount of separation voltage can be added to the input pins of each comparator to provide a hysteresis function to the overall operation. Hysteresis is enabled by setting the HYS bit.

See the **"Comparator Specifications"** section for more information.

## 42.5 Comparator Interrupt

An interrupt can be generated for every rising or falling edge of the comparator output.

When either edge detector is triggered and its associated enable bit is set (INTP and/or INTN bits), the Corresponding Interrupt Flag bit (CxIF bit of the respective PIR register) will be set.

To enable the interrupt, the following bits must be set:

- EN bit
- INTP bit (for a rising edge detection)
- INTN bit (for a falling edge detection)
- CxIE bit of the respective PIE register
- GIE bit of the INTCON0 register

© 2019-2023 Microchip Technology Inc. and its subsidiaries

The associated interrupt flag bit, CxIF bit of the respective PIR register, must be cleared in software to successfully detect another edge.

**Important:** Although a comparator is disabled, an interrupt will be generated by changing the output polarity with the POL bit.

## 42.6 Comparator Positive Input Selection

Configuring the PCH bits direct an internal voltage reference or an analog pin to the noninverting input of the comparator.

Any time the comparator is disabled (EN = 0), all comparator inputs are disabled.

## 42.7 Comparator Negative Input Selection

The NCH bits direct an analog input pin, internal reference voltage or analog ground to the inverting input of the comparator.

**Important:** To use CxINy+ and CxINy- pins as analog input, the appropriate bits must be set in the ANSEL register and the corresponding TRIS bits must also be set to disable the output drivers.

## 42.8 Comparator Response Time

The comparator output is indeterminate for a period of time after the change of an input source or the selection of a new reference voltage. This period is referred to as the response time. The response time of the comparator differs from the settling time of the voltage reference. Therefore, both of these times must be considered when determining the total response time to a comparator input change. See the Comparator and Voltage Reference Specifications in the **"Comparator Specifications"** and **"Fixed Voltage Reference (FVR) Specifications"** sections for more details.

## 42.9 Analog Input Connection Considerations

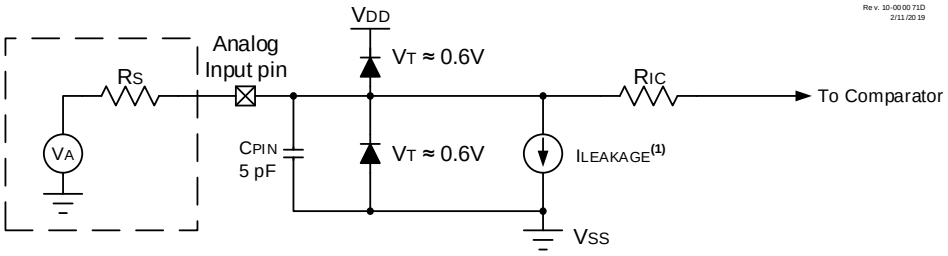
A simplified circuit for an analog input is shown in Figure 42-3. Since the analog input pins share their connection with a digital input, they have reverse biased ESD protection diodes to $V_{DD}$ and $V_{SS}$. The analog input, therefore, must be between $V_{SS}$ and $V_{DD}$. If the input voltage deviates from this range by more than 0.6V in either direction, one of the diodes is forward biased and abnormal behavior may occur.

A maximum source impedance of 10 kΩ is recommended for the analog sources. Also, any external component connected to an analog input pin, such as a capacitor or a Zener diode, will have very little leakage current to minimize corrupting the result.

**Notes:**

1. When reading a PORT register, all pins configured as analog inputs will read as a '0'. Pins configured as digital inputs will convert as an analog input, according to the input specification.
2. Analog levels on any pin defined as a digital input may cause the input buffer to consume more current than specified.

**Figure 42-3. Analog Input Model**




**Legend:** CPIN = Input Capacitance
ILEAKAGE = Leakage Current at the pin due to various junctions
RIC = Interconnect Resistance
RS = Source Impedance
VA = Analog Voltage
VT = Diode Forward Voltage

**Note:**
1. See the **"Electrical Specifications"** chapter.

## 42.10 Operation in Sleep Mode

The comparator module can operate during Sleep. A comparator interrupt will wake the device from Sleep. The CxIE bits of the respective PIE register must be set to enable comparator interrupts.

The comparator clock source is based on the Timer1 clock source. If the Timer1 clock source is either the system clock ($F_{OSC}$) or the instruction clock ($F_{OSC}/4$), Timer1 will not operate during Sleep, and synchronized comparator outputs will not operate.

## 42.11 ADC Auto-Trigger Source

The output of the comparator module can be used to trigger an ADC conversion. When the ADACT register is set to trigger on a comparator output, an ADC conversion will trigger when the comparator output goes high.

## 42.12 Register Definitions: Comparator Control

Long bit name prefixes for the Comparator peripherals are shown in the table below. Refer to the **“Long Bit Names”** section in the **“Register and Bit Naming Conventions”** chapter for more information.

**Table 42-2. Comparator Long Bit Name Prefixes**

| Peripheral | Bit Name Prefix |
|---|---|
| C1 | C1 |
| C2 | C2 |

### 42.12.1 CMxCON0

**Name:** CMxCON0
**Address:** 0x070,0x074

Comparator Control Register 0

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | EN | OUT | | POL | | | HYS | SYNC |
| Access | R/W | R | | R/W | | | R/W | R/W |
| Reset | 0 | 0 | | 0 | | | 0 | 0 |

**Bit 7 – EN** Comparator Enable

| Value | Description |
|---|---|
| 1 | Comparator is enabled |
| 0 | Comparator is disabled and consumes no active power |

**Bit 6 – OUT** Comparator Output

| Value | Condition | Description |
|---|---|---|
| 1 | If POL = 0 (noninverted polarity): | CxVP > CxVN |
| 0 | If POL = 0 (noninverted polarity): | CxVP < CxVN |
| 1 | If POL = 1 (inverted polarity): | CxVP < CxVN |
| 0 | If POL = 1 (inverted polarity): | CxVP > CxVN |

**Bit 4 – POL** Comparator Output Polarity Select

| Value | Description |
|---|---|
| 1 | Comparator output is inverted |
| 0 | Comparator output is not inverted |

**Bit 1 – HYS** Comparator Hysteresis Enable

| Value | Description |
|---|---|
| 1 | Comparator hysteresis enabled |
| 0 | Comparator hysteresis disabled |

**Bit 0 – SYNC** Comparator Output Synchronous Mode

| Value | Description |
|---|---|
| 1 | Comparator output to Timer1 and I/O pin is synchronous to changes on Timer1 clock source. Output updated on the falling edge of Timer1 clock source. |
| 0 | Comparator output to Timer1 and I/O pin is asynchronous |

© 2019-2023 Microchip Technology Inc. and its subsidiaries

### 42.12.2 CMxCON1

**Name:** CMxCON1
**Address:** 0x071,0x075

Comparator Control Register 1

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | INTP | INTN |
| Access | | | | | | | R/W | R/W |
| Reset | | | | | | | 0 | 0 |

**Bit 1 – INTP** Comparator Interrupt on Positive-Going Edge Enable

| Value | Description |
|---|---|
| 1 | The CxIF interrupt flag will be set upon a positive-going edge of the CxOUT bit |
| 0 | No interrupt flag will be set on a positive-going edge of the CxOUT bit |

**Bit 0 – INTN** Comparator Interrupt on Negative-Going Edge Enable

| Value | Description |
|---|---|
| 1 | The CxIF interrupt flag will be set upon a negative-going edge of the CxOUT bit |
| 0 | No interrupt flag will be set on a negative-going edge of the CxOUT bit |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

### 42.12.3 CMxNCH

**Name:** CMxNCH
**Address:** 0x072,0x076

Comparator Inverting Channel Select Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | NCH[2:0] | | |
| Access | | | | | | R/W | R/W | R/W |
| Reset | | | | | | 0 | 0 | 0 |

**Bits 2:0 – NCH[2:0]** Comparator Inverting Input Channel Select

| NCH | Negative Input Sources |
|---|---|
| 111 | $V_{SS}$ |
| 110 | FVR_Buffer2 |
| 101 | NCH not connected |
| 100 | NCH not connected |
| 011 | CxIN3- |
| 010 | CxIN2- |
| 001 | CxIN1- |
| 000 | CxIN0- |

### 42.12.4 CMxPCH

**Name:** CMxPCH
**Address:** 0x073,0x077

Comparator Noninverting Channel Select Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | PCH[2:0] | | |
| Access | | | | | | R/W | R/W | R/W |
| Reset | | | | | | 0 | 0 | 0 |

**Bits 2:0 – PCH[2:0]** Comparator Noninverting Input Channel Select

| PCH | Positive Input Sources |
|---|---|
| 111 | $V_{SS}$ |
| 110 | FVR_Buffer2 |
| 101 | DAC_Output |
| 100 | PCH not connected |
| 011 | PCH not connected |
| 010 | PCH not connected |
| 001 | CxIN1+ |
| 000 | CxIN0+ |

### 42.12.5 CMOUT

**Name:** CMOUT
**Address:** 0x06F

Comparator Output Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | C2OUT | C1OUT |
| Access | | | | | | | R | R |
| Reset | | | | | | | 0 | 0 |

**Bits 0, 1 – CxOUT** Mirror copy of the CMxCON0.OUT

## 42.13 Register Summary - Comparator

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x6F | CMOUT | 7:0 | | | | | | | C2OUT | C1OUT |
| 0x70 | CM1CON0 | 7:0 | EN | OUT | | POL | | | HYS | SYNC |
| 0x71 | CM1CON1 | 7:0 | | | | | | | INTP | INTN |
| 0x72 | CM1NCH | 7:0 | | | | | | NCH[2:0] | | |
| 0x73 | CM1PCH | 7:0 | | | | | | PCH[2:0] | | |
| 0x74 | CM2CON0 | 7:0 | EN | OUT | | POL | | | HYS | SYNC |
| 0x75 | CM2CON1 | 7:0 | | | | | | | INTP | INTN |
| 0x76 | CM2NCH | 7:0 | | | | | | NCH[2:0] | | |
| 0x77 | CM2PCH | 7:0 | | | | | | PCH[2:0] | | |

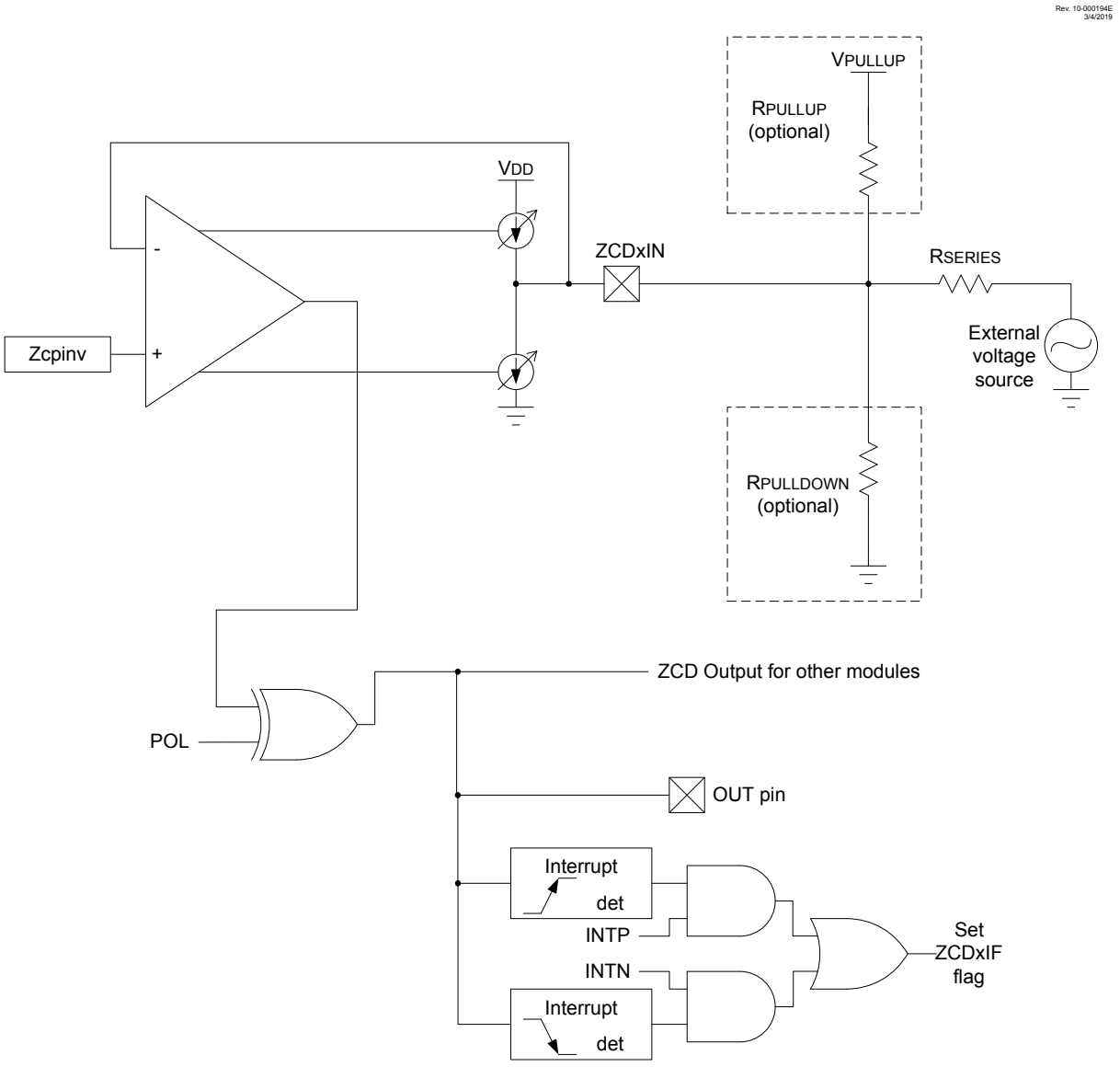
© 2019-2023 Microchip Technology Inc.
and its subsidiaries

# 43. ZCD - Zero-Cross Detection Module

The ZCD module detects when an A/C signal crosses through the ground potential. The actual zero-crossing threshold is the zero-crossing reference voltage, $Z_{CPINV}$, which is typically 0.75V above ground.

The connection to the signal to be detected is through a series current-limiting resistor. The module applies a current source or sink to the ZCD pin to maintain a constant voltage on the pin, thereby preventing the pin voltage from forward biasing the ESD protection diodes. When the applied voltage is greater than the reference voltage, the module sinks current. When the applied voltage is less than the reference voltage, the module sources current. The current source and sink action keeps the pin voltage constant over the full range of the applied voltage. The ZCD module is shown in the following simplified block diagram.

**Figure 43-1. Simplified ZCD Block Diagram**




The ZCD module is useful when monitoring an A/C waveform for, but not limited to, the following purposes:

- A/C period measurement
- Accurate long term time measurement

- Dimmer phase delayed drive
- Low EMI cycle switching

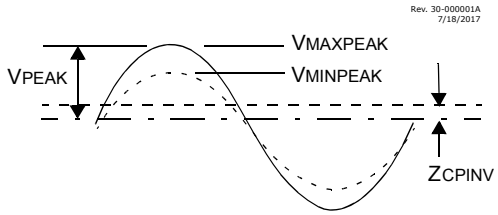
## 43.1 External Resistor Selection

The ZCD module requires a current-limiting resistor in series with the external voltage source. The impedance and rating of this resistor depends on the external source peak voltage. Select a resistor value that will drop all of the peak voltage when the current through the resistor is less than the maximum input current (ZC02). Refer to the **"Electrical Specifications"** chapter for more details. Make sure that the ZCD I/O pin internal weak pull-up is disabled so it does not interfere with the current source and sink.

**Equation 43-1. External Resistor**

$$R_{SERIES} = \frac{V_{PEAK}}{I_{ZCD}}$$

**Figure 43-2. External Voltage Source**




## 43.2 ZCD Logic Output

The ZCD module includes a Status bit, which can be read to determine whether the current source or sink is active. The OUT bit is set when the current sink is active, and cleared when the current source is active. The OUT bit is affected by the polarity bit.

The OUT signal can also be used as input to other modules. This is controlled by the registers of the corresponding module.

## 43.3 ZCD Logic Polarity

The POL bit inverts the OUT bit relative to the current source and sink output. When the POL bit is set, a OUT high indicates that the current source is active, and a low output indicates that the current sink is active. The POL bit affects the ZCD interrupts.

## 43.4 ZCD Interrupts

An interrupt will be generated upon a change in the ZCD logic output when the appropriate interrupt enables are set. The ZCD module has a rising edge detector and a falling edge detector.

The ZCDIF bit of the PIRx register will be set when either edge detector is triggered and its associated enable bit is set. The INTP enables rising edge interrupts and the INTN bit enables falling edge interrupts.

To fully enable the interrupt, the following bits must be set:

- ZCDIE bit of the PIEx register
- INTP bit for rising edge detection
- INTN bit for falling edge detection
- GIEL and GIE bits of the INTCON0 register

Changing the POL bit will cause an interrupt, regardless of the level of the SEN bit.

The ZCDIF bit of the PIRx register must be cleared in software as part of the interrupt service. If another edge is detected while this flag is being cleared, the flag will still be set at the end of the sequence.

© 2019-2023 Microchip Technology Inc. and its subsidiaries

## 43.5 Correction for $Z_{CPINV}$ Offset

The actual voltage at which the ZCD switches is the reference voltage at the noninverting input of the ZCD op amp. For external voltage source waveforms other than square waves, this voltage offset from zero causes the zero-cross event to occur either too early or too late.

### 43.5.1 Correction by AC Coupling

When the external voltage source is sinusoidal, the effects of the $Z_{CPINV}$ offset can be eliminated by isolating the external voltage source from the ZCD pin with a capacitor, in addition to the voltage reducing resistor. The capacitor will cause a phase shift resulting in the ZCD output switch in advance of the actual zero-crossing event. The phase shift will be the same for both rising and falling zero-crossings, which can be compensated for by either delaying the CPU response to the ZCD switch by a timer or other means, or selecting a capacitor value large enough that the phase shift is negligible.

To determine the series resistor and capacitor values for this configuration, start by computing the impedance, Z, to obtain a peak current less than the maximum input current (ZC02). Refer to the **"Electrical Specifications"** chapter for more details. Next, arbitrarily select a suitably large nonpolar capacitor and compute its reactance, Xc, at the external voltage source frequency. Finally, compute the series resistor, capacitor peak voltage, and phase shift using the formulas shown below.

When this technique is used and the input signal is not present, the ZCD will tend to oscillate. To avoid this oscillation, connect the ZCD pin to $V_{DD}$ or GND with a high-impedance resistor.

**Note:** In this example, the impedance value is calculated for a peak current of 300 µA.

**Equation 43-2. R-C Equations**

$V_{PEAK}$ = external voltage source peak voltage

f = external voltage source frequency

C = series capacitor

R = series resistor

$V_C$ = peak capacitor voltage

Φ = capacitor induced zero-crossing phase advance in radians

$T_\Phi$ = time ZC event occurs before actual zero-crossing

$$Z = \frac{V_{PEAK}}{3 \times 10^{-4}}$$

$$X_C = \frac{1}{2\pi f C}$$

$$R = \sqrt{Z^2 - {X_C}^2}$$

$$V_C = X_C\left(3 \times 10^{-4}\right)$$

$$\Phi = \tan^{-1}\left(\frac{X_C}{R}\right)$$

$$T_\Phi = \frac{\Phi}{2\pi f}$$

**Equation 43-3. R-C Calculation Example**

$$V_{rms} = 120$$

$$V_{PEAK} = V_{rms} \times \sqrt{2} = 169.7$$

$$f = 60\,Hz$$

$C = 0.1\ \mu F$

$Z = \frac{V_{PEAK}}{3 \times 10^{-4}} = \frac{169.7}{3 \times 10^{-4}} = 565.7\ k\Omega$

$X_C = \frac{1}{2\pi f C} = \frac{1}{2\pi \times 60 \times 10^{-7}} = 26.53\ k\Omega$

$R = \sqrt{Z^2 - {X_C}^2} = 565.1\ k\Omega\ (computed)$

$R_a = 560\ k\Omega\ (used)$

$Z_R = \sqrt{{R_a}^2 + {X_C}^2} = 560.6\ k\Omega$

$I_{PEAK} = \frac{V_{PEAK}}{Z_R} = 302.7 \times 10^{-6} A$

$V_C = X_C \times I_{PEAK} = 8.0\ V$

$\Phi = \tan^{-1}\left(\frac{X_C}{R}\right) = 0.047\ radians$

$T_\Phi = \frac{\Phi}{2\pi f} = 125.6\ \mu s$

### 43.5.2 Correction by Offset Current

When the waveform is varying relative to $V_{SS}$, the zero-cross is detected too early as the waveform falls and too late as the waveform rises. When the waveform is varying relative to $V_{DD}$, the zero-cross is detected too late as the waveform rises and too early as the waveform falls. The actual offset time can be determined for sinusoidal waveforms with the corresponding equations shown below.

**Equation 43-4. ZCD Event Offset**

When External Voltage source is relative to $V_{SS}$:

$$T_{offset} = \frac{\sin^{-1}\left(\frac{Z_{CPINV}}{V_{PEAK}}\right)}{2\pi f}$$

When External Voltage source is relative to $V_{DD}$:

$$T_{offset} = \frac{\sin^{-1}\left(\frac{V_{DD} - Z_{CPINV}}{V_{PEAK}}\right)}{2\pi f}$$

This offset time can be compensated for by adding a pull-up or pull-down biasing resistor to the ZCD pin. A pull-up resistor is used when the external voltage source is varying relative to $V_{SS}$. A pull-down resistor is used when the voltage is varying relative to $V_{DD}$. The resistor adds a bias to the ZCD pin so that the target external voltage source must go to zero to pull the pin voltage to the $Z_{CPINV}$ switching voltage. The pull-up or pull-down value can be determined with the equations shown below.

**Equation 43-5. ZCD Pull-up/Pull-down Resistor**

When External Voltage source is relative to $V_{SS}$:

$$R_{pullup} = \frac{R_{SERIES}\left(V_{pullup} - Z_{CPINV}\right)}{Z_{CPINV}}$$

When External Voltage source is relative to $V_{DD}$:

$$R_{pulldown} = \frac{R_{SERIES}(Z_{CPINV})}{(V_{DD} - Z_{CPINV})}$$

## 43.6 Handling $V_{PEAK}$ Variations

If the peak amplitude of the external voltage is expected to vary, the series resistor must be selected to keep the ZCD current source, and sink below the design maximum range specified by ZC02 and above a reasonable minimum range depending on the application. The compensating pull-up for this series resistance can be determined with the equations shown in Equation 43-5 because the pull-up value is independent from the peak voltage.

**Tip:** It is recommended that the maximum peak voltage be no more than six times the minimum peak voltage.

## 43.7 Operation During Sleep

The ZCD current sources and interrupts are unaffected by Sleep.

## 43.8 Effects of a Reset

The ZCD circuit can be configured to default to the Active or Inactive state on Power-on Reset (POR). When the $\overline{ZCD}$ Configuration bit is cleared, the ZCD circuit will be active at POR. When the $\overline{ZCD}$ Configuration bit is set, the SEN bit must be set to enable the ZCD module.

## 43.9 Disabling the ZCD Module

The ZCD module can be disabled in two ways:

1. The $\overline{ZCD}$ Configuration bit disables the ZCD module when set. When this is the case, then the ZCD module will be enabled by setting the SEN bit. When the $\overline{ZCD}$ bit is clear, the ZCD is always enabled and the SEN bit has no effect.
2. The ZCD can also be disabled using the ZCDMD bit of the PMDx register. This is subject to the status of the $\overline{ZCD}$ bit.

## 43.10 Register Definitions: ZCD Control

Long bit name prefixes for the ZCD peripherals are shown in the table below. Refer to the **"Long Bit Names"** section of the **"Register and Bit Naming Conventions"** chapter for more information.

**Table 43-1. ZCD Long Bit Name Prefixes**

| Peripheral | Bit Name Prefix |
|---|---|
| ZCD | ZCD |

### 43.10.1 ZCDCON

**Name:** ZCDCON
**Address:** 0x04C

Zero-Cross Detect Control Register

| Bit | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|
| | SEN | | OUT | POL | | | INTP | INTN |
| Access | R/W | | R | R/W | | | R/W | R/W |
| Reset | 0 | | x | 0 | | | 0 | 0 |

**Bit 7 – SEN** Zero-Cross Detect Software Enable
This bit is ignored when the ZCD fuse is cleared.

| Value | Condition | Description |
|---|---|---|
| X | ZCD Config fuse = 0 | Zero-cross detect is always enabled. This bit is ignored. |
| 1 | ZCD Config fuse = 1 | Zero-cross detect is enabled. ZCD pin is forced to output to source and sink current. |
| 0 | ZCD Config fuse = 1 | Zero-cross detect is disabled. ZCD pin operates according to PPS and TRIS controls. |

**Bit 5 – OUT** Zero-Cross Detect Data Output

| Value | Condition | Description |
|---|---|---|
| 1 | POL = 0 | ZCD pin is sinking current |
| 0 | POL = 0 | ZCD pin is sourcing current |
| 1 | POL = 1 | ZCD pin is sourcing current |
| 0 | POL = 1 | ZCD pin is sinking current |

**Bit 4 – POL** Zero-Cross Detect Polarity

| Value | Description |
|---|---|
| 1 | ZCD logic output is inverted |
| 0 | ZCD logic output is not inverted |

**Bit 1 – INTP** Zero-Cross Detect Positive-Going Edge Interrupt Enable

| Value | Description |
|---|---|
| 1 | The ZCDIF bit is set on low-to-high ZCD_output transition |
| 0 | The ZCDIF bit is unaffected by low-to-high ZCD_output transition |

**Bit 0 – INTN** Zero-Cross Detect Negative-Going Edge Interrupt Enable

| Value | Description |
|---|---|
| 1 | The ZCDIF bit is set on high-to-low ZCD_output transition |
| 0 | The ZCDIF bit is unaffected by high-to-low ZCD_output transition |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

## 43.11 Register Summary - ZCD

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x4C | ZCDCON | 7:0 | SEN | | OUT | POL | | | INTP | INTN |

# 44. Instruction Set Summary

The PIC18 devices incorporate the standard set of PIC18 core instructions, as well as an extended set of instructions to optimize code that is recursive or that utilizes a software stack. The extended set is discussed later in this section.

## 44.1 Standard Instruction Set

The standard PIC18 instruction set adds many enhancements to the previous PIC® MCU instruction sets, while maintaining an easy migration from these PIC MCU instruction sets. Most instructions are a single program memory word (16 bits), but there are a few instructions that require two- or three-program memory locations.

Each single-word instruction is a 16-bit word divided into an opcode that specifies the instruction type and one or more operands, which further specifies the operation of the instruction.

The instruction set is highly orthogonal and is grouped into four basic categories:

- Byte-oriented operations
- Bit-oriented operations
- Literal operations
- Control operations

The PIC18 instruction set summary in Table 44-2 lists byte-oriented, bit-oriented, literal and control operations. Table 44-1 shows the opcode field descriptions.

Most byte-oriented instructions have three operands:

- The file register (specified by ‘f’)
- The destination of the result (specified by ‘d’)
- The accessed memory (specified by ‘a’)

The file register designator ‘f’ specifies which file register is to be used by the instruction. The destination designator ‘d’ specifies where the result of the operation is to be placed. If ‘d’ is zero, the result is placed in the WREG register. If ‘d’ is one, the result is placed in the file register specified in the instruction.

All bit-oriented instructions have three operands:

- The file register (specified by ‘f’)
- The bit in the file register (specified by ‘b’)
- The accessed memory (specified by ‘a’)

The bit field designator ‘b’ selects the number of the bit affected by the operation, while the file register designator ‘f’ represents the number of the file in which the bit is located.

The literal instructions may use some of the following operands:

- A literal value to be loaded into a file register (specified by ‘k’)
- The desired FSR register to load the literal value into (specified by ‘f’)
- No operand required (specified by ‘—’)

The control instructions may use some of the following operands:

- A program memory address (specified by ‘n’)
- The mode of the `CALL` or `RETURN` instructions (specified by ‘s’)
- The mode of the table read and table write instructions (specified by ‘m’)
- No operand required (specified by ‘—’)

All instructions are a single word, except for a few two- or three-word instructions. These instructions were made two- or three-word to contain the required information in 32 or 48 bits. In the second and third words, the four MSbs are ‘`1`’s. If this second or third word is executed as an instruction (by itself), it will execute as a `NOP`.

All single-word instructions are executed in a single instruction cycle, unless a conditional test is true or the Program Counter is changed as a result of the instruction. In these cases, the execution takes two instruction cycles, with the additional instruction cycle(s) executed as a `NOP`.

The two-word instructions execute in two instruction cycles and three-word instructions execute in three instruction cycles.

One instruction cycle consists of four oscillator periods. Thus, for an oscillator frequency of 4 MHz, the normal instruction execution time is 1 µs. If a conditional test is true, or the Program Counter is changed as a result of an instruction, the instruction execution time is 2 µs. Two-word branch instructions (if true) take 3 µs.

Figure 44-1, Figure 44-2 and Figure 44-3 show the general formats that the instructions can have. All examples use the convention 'nnh' to represent a hexadecimal number.

The Instruction Set Summary, shown in Table 44-2, lists the standard instructions recognized by the Microchip MPASM™ Assembler.

Standard Instruction Set provides a description of each instruction.

**Table 44-1. Opcode Field Descriptions**

| Field | Description |
|---|---|
| a | RAM access bit<br>a = 0: RAM location in Access RAM (BSR register is ignored)<br>a = 1: RAM bank is specified by BSR register (default) |
| ACCESS | ACCESS = 0: RAM access bit symbol |
| BANKED | BANKED = 1: RAM access bit symbol |
| bbb | Bit address within an 8-bit file register (0 to 7) |
| BSR | Bank Select Register (BSR). Used to select the current RAM bank. |
| d | Destination select bit<br>d = 0: store result in WREG<br>d = 1: store result in file register f (default) |
| dest | Destination: either the WREG register or the specified register file location |
| f | 8-bit register file address (00h to FFh) |
| $f_n$ | FSR Number (0 to 2) |
| $f_s$ | 12-bit register file address (000h to FFFh) or 14-bit register file address (0000h to 3FFFh). This is the source address. |
| $f_d$ | 12-bit register file address (000h to FFFh) or 14-bit register file address (0000h to 3FFFh). This is the destination address. |
| $z_s$ | 7-bit literal offset for FSR2 to used as register file address (000h to FFFh). This is the source address. |
| $z_d$ | 7-bit literal offset for FSR2 to used as register file address (000h to FFFh). This is the destination address. |
| k | Literal field, constant data or label (may be either a 6-bit, 8-bit, 12-bit or a 20-bit value) |
| label | Label name |
| mm<br><br>*<br>*+<br>*-<br>+* | The mode of the TBLPTR register for the table read and table write instructions.<br>Only used with table read and table write instructions:<br><br>No change to register (such as TBLPTR with table reads and writes)<br>Post-Increment register (such as TBLPTR with table reads and writes)<br>Post-Decrement register (such as TBLPTR with table reads and writes)<br>Pre-Increment register (such as TBLPTR with table reads and writes) |

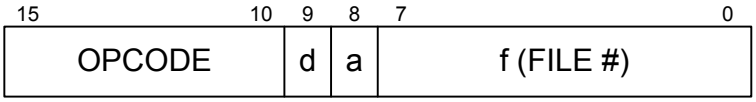
.....continued

| Field | Description |
|---|---|
| n | The relative address (two's complement number) for relative branch instructions, or the direct address for call/branch and return instructions. |
| PRODH | Product of multiply high byte |
| PRODL | Product of multiply low byte |
| s | Fast Call/Return mode select bit<br>s = 0: do not update into/from shadow registers (default)<br>s = 1: certain registers loaded into/from shadow registers (Fast mode) |
| u | Unused or unchanged |
| W | W = 0: Destination select bit symbol |
| WREG | Working register (accumulator) |
| x | Don't care ('0' or '1'). The assembler will generate code with x = 0. It is the recommended form of use for compatibility with all Microchip software tools. |
| TBLPTR | 21-bit Table Pointer (points to a program memory location) |
| TABLAT | 8-bit table latch |
| TOS | Top-of-stack (TOS) |
| PC | Program Counter |
| PCL | Program Counter low byte |
| PCH | Program Counter high byte |
| PCLATH | Program Counter high byte latch |
| PCLATU | Program Counter upper byte Latch |
| GIE | Global Interrupt Enable bit |
| WDT | Watchdog Timer |
| $\overline{TO}$ | Time-Out bit |
| $\overline{PD}$ | Power-Down bit |
| C, DC, Z, OV, N | ALU Status bits: Carry, Digit Carry, Zero, Overflow, Negative |
| { } | Optional argument |
| [ ] | Indexed address |
| ( ) | Contents |
| < > | Register bit field |
| [expr]<n> | Specifies bit n of the register indicated by pointer expr |
| → | Assigned to |
| ∈ | In the set of |
| italics | User defined term (font is Courier) |

**Figure 44-1. General Format for Byte-Oriented Instructions**

**Byte-oriented** file register operations

**Example Instruction**

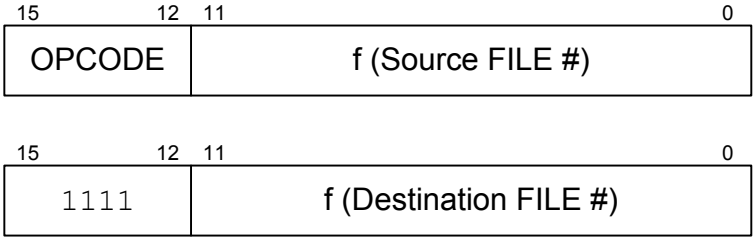



```
ADDWF MYREG, W, B
```

d = 0 for result destination to be WREG register
d = 1 for result destination to be file register (f)
a = 0 to force Access Bank
a = 1 for BSR to select bank
f = 8-bit file register address

**Byte to Byte** move operations (two-word)

**Example Instruction**

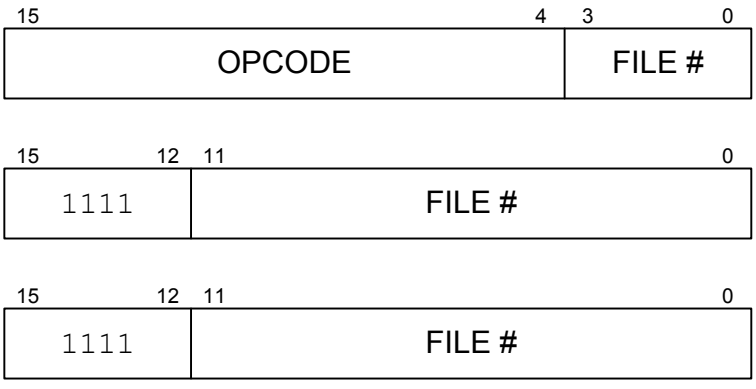



```
MOVFF MYREG1, MYREG2
```

f = 12-bit file register address

**Byte to Byte** move operations (three-word)

**Example Instruction**

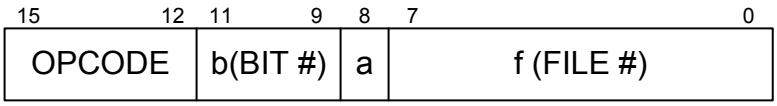



```
MOVFFL MYREG1, MYREG2
```

**Figure 44-2. General Format for Bit-Oriented and Literal Instructions**

**Bit-oriented** file register operations

**Example Instruction**

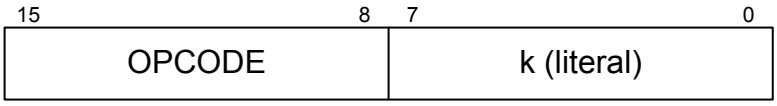



```
BSF MYREG, bit, B
```

b = 3-bit position of bit in file register (f)
a = 0 to force Access Bank
a = 1 for BSR to select bank
f = 8-bit file register address

**Literal** operations

**Example Instruction**



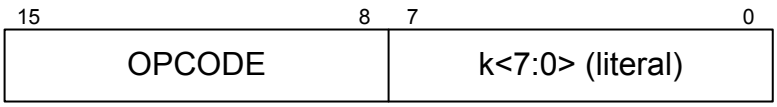

```
MOVLW 7Fh
```

k = 8-bit immediate value

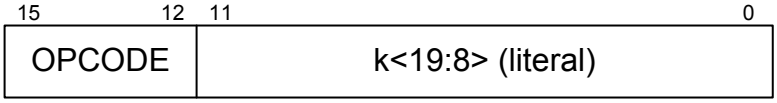
**Figure 44-3. General Format for Control Instructions**

**Control** operations
**CALL, GOTO and Branch** operations

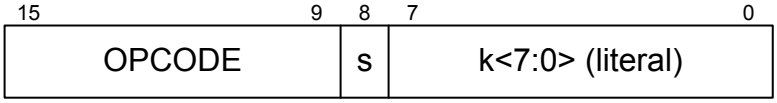
**Example Instruction**



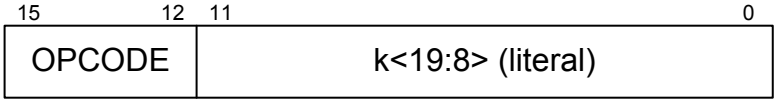

```
GOTO Label
```



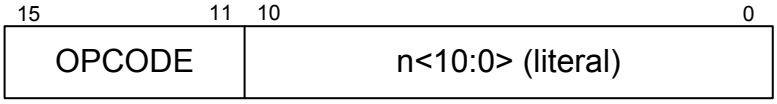

k = 20-bit immediate value




```
CALL MYFUNC
```

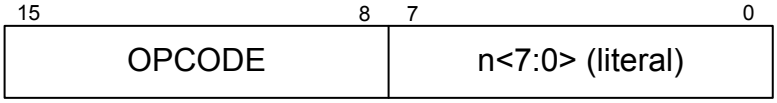



k = 20-bit immediate value
s = Fast bit




```
BRA MYFUNC
```

n = 11-bit immediate value




```
BC MYFUNC
```

n = 8-bit immediate value

**Table 44-2. Standard Instruction Set**

| Mnemonic, Operands | | Description | Cycles | 16-Bit Instruction Word | | | | Status Affected | Notes |
|---|---|---|---|---|---|---|---|---|---|
| | | | | MSb | | | LSb | | |
| **BYTE-ORIENTED FILE REGISTER INSTRUCTIONS** | | | | | | | | | |
| ADDWF | f, d, a | Add WREG and f | 1 | 0010 | 01da | ffff | ffff | C, DC, Z, OV, N | 1 |
| ADDWFC | f, d, a | Add WREG and Carry bit to f | 1 | 0010 | 00da | ffff | ffff | C, DC, Z, OV, N | 1 |
| ANDWF | f, d, a | AND WREG with f | 1 | 0001 | 01da | ffff | ffff | Z, N | 1 |
| CLRF | f, a | Clear f | 1 | 0110 | 101a | ffff | ffff | Z | |
| COMF | f, d, a | Complement f | 1 | 0001 | 11da | ffff | ffff | Z, N | 1 |
| DECF | f, d, a | Decrement f | 1 | 0000 | 01da | ffff | ffff | C, DC, Z, OV, N | 1 |
| INCF | f, d, a | Increment f | 1 | 0010 | 10da | ffff | ffff | C, DC, Z, OV, N | 1 |
| IORWF | f, d, a | Inclusive OR WREG with f | 1 | 0001 | 00da | ffff | ffff | Z, N | 1 |
| MOVF | f, d, a | Move f to WREG or f | 1 | 0101 | 00da | ffff | ffff | Z, N | 1 |
| MOVFF | $f_s$, $f_d$ | Move $f_s$ (12-bit source) to $f_d$ (12-bit destination) | 2 | 1100 | $f_sf_sf_sf_s$ | $f_sf_sf_sf_s$ | $f_sf_sf_sf_s$ | None | 1, 3, 4 |
| | | | | 1111 | $f_df_df_df_d$ | $f_df_df_df_d$ | $f_df_df_df_d$ | | |
| MOVFFL | $f_s$, $f_d$ | Move $f_s$ (14-bit source) to $f_d$ (14-bit destination) | 3 | 0000 | 0000 | 0110 | $f_sf_sf_sf_s$ | None | 1, 3 |
| | | | | 1111 | $f_sf_sf_sf_s$ | $f_sf_sf_sf_s$ | $f_sf_sf_df_d$ | | |
| | | | | 1111 | $f_df_df_df_d$ | $f_df_df_df_d$ | $f_df_df_df_d$ | | |
| MOVWF | f, a | Move WREG to f | 1 | 0110 | 111a | ffff | ffff | None | |
| MULWF | f, a | Multiply WREG with f | 1 | 0000 | 001a | ffff | ffff | None | 1 |
| NEGF | f, a | Negate f | 1 | 0110 | 110a | ffff | ffff | C, DC, Z, OV, N | 1 |
| RLCF | f, d, a | Rotate Left f through Carry | 1 | 0011 | 01da | ffff | ffff | C, Z, N | 1 |
| RLNCF | f, d, a | Rotate Left f (No Carry) | 1 | 0100 | 01da | ffff | ffff | Z, N | 1 |
| RRCF | f, d, a | Rotate Right f through Carry | 1 | 0011 | 00da | ffff | ffff | C, Z, N | 1 |
| RRNCF | f, d, a | Rotate Right f (No Carry) | 1 | 0100 | 00da | ffff | ffff | Z, N | 1 |
| SETF | f, a | Set f | 1 | 0110 | 100a | ffff | ffff | None | |

© 2019-2023 Microchip Technology Inc. and its subsidiaries

.....continued

| Mnemonic, Operands | | Description | Cycles | 16-Bit Instruction Word | | | | Status Affected | Notes |
|---|---|---|---|---|---|---|---|---|---|
| | | | | MSb | | | LSb | | |
| SUBFWB | f, d, a | Subtract f from WREG with Borrow | 1 | 0101 | 01da | ffff | ffff | C, DC, Z, OV, N | 1 |
| SUBWF | f, d, a | Subtract WREG from f | 1 | 0101 | 11da | ffff | ffff | C, DC, Z, OV, N | 1 |
| SUBWFB | f, d, a | Subtract WREG from f with Borrow | 1 | 0101 | 10da | ffff | ffff | C, DC, Z, OV, N | 1 |
| SWAPF | f, d, a | Swap nibbles in f | 1 | 0011 | 10da | ffff | ffff | None | 1 |
| XORWF | f, d, a | Exclusive OR WREG with f | 1 | 0001 | 10da | ffff | ffff | Z, N | 1 |
| **BYTE-ORIENTED SKIP INSTRUCTIONS** | | | | | | | | | |
| CPFSEQ | f, a | Compare f with WREG, skip if = | 1 – 4 | 0110 | 001a | ffff | ffff | None | 1, 2 |
| CPFSGT | f, a | Compare f with WREG, skip if > | 1 – 4 | 0110 | 010a | ffff | ffff | None | 1, 2 |
| CPFSLT | f, a | Compare f with WREG, skip if < | 1 – 4 | 0110 | 000a | ffff | ffff | None | 1, 2 |
| DECFSZ | f, d, a | Decrement f, Skip if 0 | 1 – 4 | 0010 | 11da | ffff | ffff | None | 1, 2 |
| DCFSNZ | f, d, a | Decrement f, Skip if Not 0 | 1 – 4 | 0100 | 11da | ffff | ffff | None | 1, 2 |
| INCFSZ | f, d, a | Increment f, Skip if 0 | 1 – 4 | 0011 | 11da | ffff | ffff | None | 1, 2 |
| INFSNZ | f, d, a | Increment f, Skip if Not 0 | 1 – 4 | 0100 | 10da | ffff | ffff | None | 1, 2 |
| TSTFSZ | f, a | Test f, skip if 0 | 1 – 4 | 0110 | 011a | ffff | ffff | None | 1, 2 |
| **BIT-ORIENTED FILE REGISTER INSTRUCTIONS** | | | | | | | | | |
| BCF | f, b, a | Bit Clear f | 1 | 1001 | bbba | ffff | ffff | None | 1 |
| BSF | f, b, a | Bit Set f | 1 | 1000 | bbba | ffff | ffff | None | 1 |
| BTG | f, b, a | Bit Toggle f | 1 | 0111 | bbba | ffff | ffff | None | 1 |
| **BIT-ORIENTED SKIP INSTRUCTIONS** | | | | | | | | | |
| BTFSC | f, b, a | Bit Test f, Skip if Clear | 1 – 4 | 1011 | bbba | ffff | ffff | None | 1, 2 |

© 2019-2023 Microchip Technology Inc. and its subsidiaries

.....continued

| Mnemonic, Operands | | Description | Cycles | 16-Bit Instruction Word | | | | Status Affected | Notes |
|---|---|---|---|---|---|---|---|---|---|
| | | | | MSb | | | LSb | | |
| BTFSS | f, b, a | Bit Test f, Skip if Set | 1 – 4 | 1010 | bbba | ffff | ffff | None | 1, 2 |
| **CONTROL INSTRUCTIONS** | | | | | | | | | |
| BC | n | Branch if Carry | 1 – 2 | 1110 | 0010 | nnnn | nnnn | None | 2 |
| BN | n | Branch if Negative | 1 – 2 | 1110 | 0110 | nnnn | nnnn | None | 2 |
| BNC | n | Branch if Not Carry | 1 – 2 | 1110 | 0011 | nnnn | nnnn | None | 2 |
| BNN | n | Branch if Not Negative | 1 – 2 | 1110 | 0111 | nnnn | nnnn | None | 2 |
| BNOV | n | Branch if Not Overflow | 1 – 2 | 1110 | 0101 | nnnn | nnnn | None | 2 |
| BNZ | n | Branch if Not Zero | 1 – 2 | 1110 | 0001 | nnnn | nnnn | None | 2 |
| BOV | n | Branch if Overflow | 1 – 2 | 1110 | 0100 | nnnn | nnnn | None | 2 |
| BRA | n | Branch Unconditionally | 2 | 1101 | 0nnn | nnnn | nnnn | None | 2 |
| BZ | n | Branch if Zero | 1 – 2 | 1110 | 0000 | nnnn | nnnn | None | 2 |
| CALL | k, s | Call subroutine | 2 | 1110 | 110s | kkkk | kkkk | None | 2, 3 |
| | | | | 1111 | kkkk | kkkk | kkkk | | |
| CALLW | — | Call subroutine using WREG | 2 | 0000 | 0000 | 0001 | 0100 | None | 2 |
| GOTO | k | Go to address | 2 | 1110 | 1111 | kkkk | kkkk | None | 3 |
| | | | | 1111 | kkkk | kkkk | kkkk | | |
| RCALL | n | Relative Call | 2 | 1101 | 1nnn | nnnn | nnnn | None | 2 |
| RETFIE | s | Return from interrupt enable | 2 | 0000 | 0000 | 0001 | 000s | INTCONx STAT bits | 2 |
| RETLW | k | Return with literal in WREG | 2 | 0000 | 1100 | kkkk | kkkk | None | 2 |
| RETURN | s | Return from Subroutine | 2 | 0000 | 0000 | 0001 | 001s | None | 2 |
| **INHERENT INSTRUCTIONS** | | | | | | | | | |
| CLRWDT | — | Clear Watchdog Timer | 1 | 0000 | 0000 | 0000 | 0100 | $\overline{TO}$, $\overline{PD}$ | |

© 2019-2023 Microchip Technology Inc. and its subsidiaries

.....continued

| Mnemonic, Operands | | Description | Cycles | 16-Bit Instruction Word | | | | Status Affected | Notes |
|---|---|---|---|---|---|---|---|---|---|
| | | | | MSb | | | LSb | | |
| DAW | — | Decimal Adjust WREG | 1 | 0000 | 0000 | 0000 | 0111 | C | |
| NOP | — | No Operation | 1 | 0000 | 0000 | 0000 | 0000 | None | |
| NOP | — | No Operation | 1 | 1111 | xxxx | xxxx | xxxx | None | 3 |
| POP | — | Pop top of return stack (TOS) | 1 | 0000 | 0000 | 0000 | 0110 | None | |
| PUSH | — | Push top of return stack (TOS) | 1 | 0000 | 0000 | 0000 | 0101 | None | |
| RESET | — | Software device Reset | 1 | 0000 | 0000 | 1111 | 1111 | All | |
| SLEEP | — | Go into Standby mode | 1 | 0000 | 0000 | 0000 | 0011 | $\overline{TO}$, $\overline{PD}$ | |
| **LITERAL INSTRUCTIONS** | | | | | | | | | |
| ADDFSR | $f_n$, k | Add FSR ($f_n$) with literal (k) | 1 | 1110 | 1000 | $f_nf_nkk$ | kkkk | None | |
| ADDLW | k | Add literal and WREG | 1 | 0000 | 1111 | kkkk | kkkk | C, DC, Z, OV, N | |
| ANDLW | k | AND literal with WREG | 1 | 0000 | 1011 | kkkk | kkkk | Z, N | |
| IORLW | k | Inclusive OR literal with WREG | 1 | 0000 | 1001 | kkkk | kkkk | Z, N | |
| LFSR | $f_n$, k | Load FSR($f_n$) with a 14-bit literal (k) | 2 | 1110 | 1110 | $00f_nf_n$ | kkkk | None | 3 |
| | | | | 1111 | 00kk | kkkk | kkkk | | |
| MOVLB | k | Move literal to BSR<5:0> | 1 | 0000 | 0001 | 00kk | kkkk | None | |
| MOVLW | k | Move literal to WREG | 1 | 0000 | 1110 | kkkk | kkkk | None | |
| MULLW | k | Multiply literal with WREG | 1 | 0000 | 1101 | kkkk | kkkk | None | |
| RETLW | k | Return with literal in WREG | 2 | 0000 | 1100 | kkkk | kkkk | None | |
| SUBFSR | $f_n$, k | Subtract literal (k) from FSR ($f_n$) | 1 | 1110 | 1001 | $f_nf_nkk$ | kkkk | None | |
| SUBLW | k | Subtract WREG from literal | 1 | 0000 | 1000 | kkkk | kkkk | C, DC, Z, OV, N | |

© 2019-2023 Microchip Technology Inc. and its subsidiaries

.....continued

| Mnemonic, Operands | | Description | Cycles | 16-Bit Instruction Word | | | | Status Affected | Notes |
|---|---|---|---|---|---|---|---|---|---|
| | | | | MSb | | | LSb | | |
| XORLW | k | Exclusive OR literal with WREG | 1 | 0000 | 1010 | kkkk | kkkk | Z, N | |
| **DATA MEMORY – PROGRAM MEMORY INSTRUCTIONS** | | | | | | | | | |
| TBLRD* | — | Table Read | 2 | 0000 | 0000 | 0000 | 1000 | None | |
| TBLRD*+ | — | Table Read with post-increment | 2 | 0000 | 0000 | 0000 | 1001 | None | |
| TBLRD*- | — | Table Read with post-decrement | 2 | 0000 | 0000 | 0000 | 1010 | None | |
| TBLRD+* | — | Table Read with pre-increment | 2 | 0000 | 0000 | 0000 | 1011 | None | |
| TBLWT* | — | Table Write | 2 | 0000 | 0000 | 0000 | 1100 | None | |
| TBLWT*+ | — | Table Write with post-increment | 2 | 0000 | 0000 | 0000 | 1101 | None | |
| TBLWT*- | — | Table Write with post-decrement | 2 | 0000 | 0000 | 0000 | 1110 | None | |
| TBLWT+* | — | Table Write with pre-increment | 2 | 0000 | 0000 | 0000 | 1111 | None | |

**Notes:**

1. When a PORT register is modified as a function of itself (e.g., `MOVF PORTB, 1, 0`), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.
2. If Program Counter (PC) is modified or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a `NOP`.
3. Some instructions are multi-word instructions. The extra words of these instructions will be executed as a `NOP` unless the first word of the instruction retrieves the information embedded in these 16 bits. This ensures that all program memory locations have a valid instruction.
4. $f_s$ and $f_d$ do not cover the full memory range. 2 MSbs of bank selection are forced to `0b00` to limit the range of these instructions to the lower 4k addressing space.

© 2019-2023 Microchip Technology Inc. and its subsidiaries

### 44.1.1 Standard Instruction Set

**Important:** All PIC18 instructions may take an optional label argument preceding the instruction mnemonic for use in symbolic addressing. If a label is used, the instruction format then becomes: `{label} instruction argument(s).`

| ADDFSR | Add Literal to FSR | | | |
|---|---|---|---|---|
| **Syntax** | **`ADDFSR fn, k`** | | | |
| Operands | $0 \le k \le 63$<br>$f_n \in [0, 1, 2]$ | | | |
| Operation | $(FSRf_n) + k \rightarrow FSRf_n$ | | | |
| Status Affected | None | | | |
| Encoding | 1110 | 1000 | $f_nf_nkk$ | kkkk |
| Description | The 6-bit literal ‘k’ is added to the contents of the FSR specified by ‘$f_n$’. | | | |
| Words | 1 | | | |
| Cycles | 1 | | | |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal ‘k’ | Process Data | Write to FSR |

Example: `ADDFSR 2, 23h`

Before Instruction
FSR2 = `03FFh`

After Instruction
FSR2 = `0422h`

| ADDLW | Add Literal to W | | | |
|---|---|---|---|---|
| **Syntax** | **`ADDLW k`** | | | |
| Operands | $0 \le k \le 255$ | | | |
| Operation | $(W) + k \rightarrow W$ | | | |
| Status Affected | N, OV, C, DC, Z | | | |
| Encoding | 0000 | 1111 | kkkk | kkkk |
| Description | The contents of W are added to the 8-bit literal ‘k’ and the result is placed in W. | | | |
| Words | 1 | | | |
| Cycles | 1 | | | |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|

| Decode | Read literal 'k' | Process Data | Write to W |
|---|---|---|---|

Example: `ADDLW 15h`

Before Instruction
W = `10h`

After Instruction
W = `25h`

| ADDWF | Add W to f | | | |
|---|---|---|---|---|
| **Syntax** | **`ADDWF f {,d {,a}}`** | | | |
| Operands | 0 ≤ f ≤ 255<br>d ∈ [0, 1]<br>a ∈ [0, 1] | | | |
| Operation | (W) + (f) → dest | | | |
| Status Affected | N, OV, C, DC, Z | | | |
| Encoding | `0010` | `01da` | `ffff` | `ffff` |
| Description | Add W to register 'f'. If 'd' is '`0`', the result is stored in W. If 'd' is '`1`', the result is stored back in register 'f' (default).<br>If 'a' is '`0`', the Access Bank is selected. If 'a' is '`1`', the BSR is used to select the GPR bank.<br>If 'a' is '`0`' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ `95` (`5Fh`). See Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode for details. | | | |
| Words | 1 | | | |
| Cycles | 1 | | | |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example: `ADDWF REG, 0, 0`

Before Instruction
W = `17h`
REG = `0C2h`

After Instruction
W = `0D9h`
REG = `0C2h`

| ADDWFC | Add W and Carry Bit to f |
|---|---|
| **Syntax** | **`ADDWFC f {,d {,a}}`** |
| Operands | 0 ≤ f ≤ 255<br>d ∈ [0, 1]<br>a ∈ [0, 1] |

| .....continued | | | | |
|---|---|---|---|---|
| **ADDWFC** | **Add W and Carry Bit to f** | | | |
| **Syntax** | **ADDWFC f {,d {,a}}** | | | |
| Operation | (W) + (f) + (C) → dest | | | |
| Status Affected | N, OV, C, DC, Z | | | |
| Encoding | 0010 | 00da | ffff | ffff |
| Description | Add W, the Carry flag and data memory location 'f'. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is placed in data memory location 'f'.<br>If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.<br>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode for details. | | | |
| Words | 1 | | | |
| Cycles | 1 | | | |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example: `ADDWFC REG, 0, 1`

Before Instruction
Carry bit = 1
REG = 02h
W = 4Dh

After Instruction
Carry bit = 0
REG = 02h
W = 50h

| **ANDLW** | **AND Literal with W** | | | |
|---|---|---|---|---|
| **Syntax** | **ANDLW k** | | | |
| Operands | 0 ≤ k ≤ 255 | | | |
| Operation | (W) .AND. k → W | | | |
| Status Affected | N, Z | | | |
| Encoding | 0000 | 1011 | kkkk | kkkk |
| Description | The contents of W are ANDed with the 8-bit literal 'k'. The result is placed in W. | | | |
| Words | 1 | | | |
| Cycles | 1 | | | |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|

| Decode | Read literal 'k' | Process Data | Write to W |
|---|---|---|---|

Example: `ANDLW 05Fh`

Before Instruction
W = `A3h`

After Instruction
W = `03h`

| ANDWF | AND W with f | | | |
|---|---|---|---|---|
| **Syntax** | **`ANDWF f {,d {,a}}`** | | | |
| Operands | 0 ≤ f ≤ 255<br>d ∈ [0, 1]<br>a ∈ [0, 1] | | | |
| Operation | (W) .AND. (f) → dest | | | |
| Status Affected | N, Z | | | |
| Encoding | `0001` | `01da` | `ffff` | `ffff` |
| Description | The contents of W are ANDed with register 'f'. If 'd' is '`0`', the result is stored in W. If 'd' is '`1`', the result is stored back in register 'f' (default).<br>If 'a' is '`0`', the Access Bank is selected. If 'a' is '`1`', the BSR is used to select the GPR bank.<br>If 'a' is '`0`' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ `95` (`5Fh`). See Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode for details. | | | |
| Words | 1 | | | |
| Cycles | 1 | | | |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example: `ANDWF REG, 0, 0`

Before Instruction
W = `17h`
REG = `C2h`

After Instruction
W = `02h`
REG = `C2h`

| BC | Branch if Carry |
|---|---|
| **Syntax** | **`BC n`** |
| Operands | -128 ≤ n ≤ 127 |

| .....continued | | | | |
|---|---|---|---|---|
| **BC** | **Branch if Carry** | | | |
| **Syntax** | **BC n** | | | |
| Operation | If the Carry bit is '1'<br>(PC) + 2 + 2n → PC | | | |
| Status Affected | None | | | |
| Encoding | 1110 | 0010 | nnnn | nnnn |
| Description | If the Carry bit is '1', then the program will branch. The two's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a two-cycle instruction. | | | |
| Words | 1 | | | |
| Cycles | 1 (2) | | | |

Q Cycle Activity:

If Jump:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'n' | Process Data | Write to PC |
| No operation | No operation | No operation | No operation |

If No Jump:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'n' | Process Data | No operation |

Example: `HERE BC 5`

Before Instruction
PC = address `(HERE)`

After Instruction
If Carry = `1`; PC = address `(HERE + 12)`
If Carry = `0`; PC = address `(HERE + 2)`

| **BCF** | **Bit Clear f** | | | |
|---|---|---|---|---|
| **Syntax** | **BCF f, b {,a}** | | | |
| Operands | 0 ≤ f ≤ 255<br>0 ≤ b ≤ 7<br>a ∈ [0, 1] | | | |
| Operation | 0 → f<b> | | | |
| Status Affected | None | | | |
| Encoding | 1001 | bbba | ffff | ffff |

| .....continued | |
|---|---|
| **BCF** | **Bit Clear f** |
| **Syntax** | `BCF f, b {,a}` |
| Description | Bit 'b' in register 'f' is cleared.<br>If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.<br>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode for details. |
| Words | 1 |
| Cycles | 1 |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write register 'f' |

Example: `BCF FLAG_REG, 7, 0`

Before Instruction
FLAG_REG = C7h

After Instruction
FLAG_REG = 47h

| **BN** | **Branch if Negative** | | | |
|---|---|---|---|---|
| **Syntax** | `BN n` | | | |
| Operands | -128 ≤ n ≤ 127 | | | |
| Operation | If NEGATIVE bit is '1'<br>(PC) + 2 + 2n → PC | | | |
| Status Affected | None | | | |
| Encoding | 1110 | 0110 | nnnn | nnnn |
| Description | If the NEGATIVE bit is '1', then the program will branch. The two's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a two-cycle instruction. | | | |
| Words | 1 | | | |
| Cycles | 1 (2) | | | |

Q Cycle Activity:

If Jump:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'n' | Process Data | Write to PC |
| No operation | No operation | No operation | No operation |

If No Jump:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal ‘n’ | Process Data | No operation |

Example: `HERE BN Jump`

Before Instruction
PC = address `(HERE)`

After Instruction
If NEGATIVE = `1`; PC = address `(Jump)`
If NEGATIVE = `0`; PC = address `(HERE + 2)`

| **BNC** | **Branch if Not Carry** | | | |
|---|---|---|---|---|
| **Syntax** | **`BNC n`** | | | |
| Operands | -128 ≤ n ≤ 127 | | | |
| Operation | If the Carry bit is ‘0’<br>(PC) + 2 + 2n → PC | | | |
| Status Affected | None | | | |
| Encoding | 1110 | 0011 | nnnn | nnnn |
| Description | If the Carry bit is ‘0’, then the program will branch. The two’s complement number ‘2n’ is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a two-cycle instruction. | | | |
| Words | 1 | | | |
| Cycles | 1 (2) | | | |

Q Cycle Activity:

If Jump:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal ‘n’ | Process Data | Write to PC |
| No operation | No operation | No operation | No operation |

If No Jump:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal ‘n’ | Process Data | No operation |

Example: `HERE BNC Jump`

Before Instruction
PC = address `(HERE)`

After Instruction
If Carry = `0`; PC = address `(Jump)`
If Carry = `1`; PC = address `(HERE + 2)`

| BNN | Branch if Not Negative | | | |
|---|---|---|---|---|
| **Syntax** | **BNN n** | | | |
| Operands | -128 ≤ n ≤ 127 | | | |
| Operation | If NEGATIVE bit is '0'<br>(PC) + 2 + 2n → PC | | | |
| Status Affected | None | | | |
| Encoding | 1110 | 0111 | nnnn | nnnn |
| Description | If the NEGATIVE bit is '0', then the program will branch. The two's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a two-cycle instruction. | | | |
| Words | 1 | | | |
| Cycles | 1 (2) | | | |

Q Cycle Activity:

If Jump:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'n' | Process Data | Write to PC |
| No operation | No operation | No operation | No operation |

If No Jump:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'n' | Process Data | No operation |

Example: `HERE BNN Jump`

Before Instruction
PC = address `(HERE)`

After Instruction
If NEGATIVE = `0`; PC = address `(Jump)`
If NEGATIVE = `1`; PC = address `(HERE + 2)`

| BNOV | Branch if Not Overflow | | | |
|---|---|---|---|---|
| **Syntax** | **BNOV n** | | | |
| Operands | -128 ≤ n ≤ 127 | | | |
| Operation | If OVERFLOW bit is '0'<br>(PC) + 2 + 2n → PC | | | |
| Status Affected | None | | | |
| Encoding | 1110 | 0101 | nnnn | nnnn |
| Description | If the OVERFLOW bit is '0', then the program will branch. The two's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a two-cycle instruction. | | | |

| .....continued | |
|---|---|
| **BNOV** | **Branch if Not Overflow** |
| **Syntax** | **BNOV n** |
| Words | 1 |
| Cycles | 1 (2) |

Q Cycle Activity:

If Jump:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'n' | Process Data | Write to PC |
| No operation | No operation | No operation | No operation |

If No Jump:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'n' | Process Data | No operation |

Example: `HERE BNOV Jump`

Before Instruction
PC = address `(HERE)`

After Instruction
If OVERFLOW = `0`; PC = address `(Jump)`
If OVERFLOW = `1`; PC = address `(HERE + 2)`

| **BNZ** | **Branch if Not Zero** | | | |
|---|---|---|---|---|
| **Syntax** | **BNZ n** | | | |
| Operands | -128 ≤ n ≤ 127 | | | |
| Operation | If ZERO bit is '0'<br>(PC) + 2 + 2n → PC | | | |
| Status Affected | None | | | |
| Encoding | 1110 | 0001 | nnnn | nnnn |
| Description | If the ZERO bit is '0', then the program will branch. The two's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a two-cycle instruction. | | | |
| Words | 1 | | | |
| Cycles | 1 (2) | | | |

Q Cycle Activity:

If Jump:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'n' | Process Data | Write to PC |

| | | | |
|---|---|---|---|
| No operation | No operation | No operation | No operation |

If No Jump:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal ‘n’ | Process Data | No operation |

Example: `HERE BNZ Jump`

Before Instruction
PC = address `(HERE)`

After Instruction
If ZERO = `0`; PC = address `(Jump)`
If ZERO = `1`; PC = address `(HERE + 2)`

| **BOV** | **Branch if Overflow** | | | |
|---|---|---|---|---|
| **Syntax** | **`BOV n`** | | | |
| Operands | -128 ≤ n ≤ 127 | | | |
| Operation | If OVERFLOW bit is ‘1’<br>(PC) + 2 + 2n → PC | | | |
| Status Affected | None | | | |
| Encoding | 1110 | 0100 | nnnn | nnnn |
| Description | If the OVERFLOW bit is ‘1’, then the program will branch. The two’s complement number ‘2n’ is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a two-cycle instruction. | | | |
| Words | 1 | | | |
| Cycles | 1 (2) | | | |

Q Cycle Activity:

If Jump:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal ‘n’ | Process Data | Write to PC |
| No operation | No operation | No operation | No operation |

If No Jump:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal ‘n’ | Process Data | No operation |

Example: `HERE BOV Jump`

Before Instruction
PC = address `(HERE)`

After Instruction
If OVERFLOW = 1; PC = address (Jump)
If OVERFLOW = 0; PC = address (HERE + 2)

| BRA | Unconditional Branch | | | |
|---|---|---|---|---|
| **Syntax** | **BRA n** | | | |
| Operands | -1024 ≤ n ≤ 1023 | | | |
| Operation | (PC) + 2 + 2n → PC | | | |
| Status Affected | None | | | |
| Encoding | 1101 | 0nnn | nnnn | nnnn |
| Description | The two's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is a two-cycle instruction. | | | |
| Words | 1 | | | |
| Cycles | 2 | | | |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'n' | Process Data | Write to PC |
| No operation | No operation | No operation | No operation |

Example: HERE BRA Jump

Before Instruction
PC = address (HERE)

After Instruction
PC = address (Jump)

| BSF | Bit Set f | | | |
|---|---|---|---|---|
| **Syntax** | **BSF f, b {,a}** | | | |
| Operands | 0 ≤ f ≤ 255<br>0 ≤ b ≤ 7<br>a ∈ [0, 1] | | | |
| Operation | 1 → f<b> | | | |
| Status Affected | None | | | |
| Encoding | 1000 | bbba | ffff | ffff |
| Description | Bit 'b' in register 'f' is set.<br>If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.<br>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode for details. | | | |
| Words | 1 | | | |

| .....continued | |
|---|---|
| **BSF** | **Bit Set f** |
| **Syntax** | `BSF f, b {,a}` |
| Cycles | 1 |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register ‘f’ | Process Data | Write register ‘f’ |

Example: `BSF FLAG_REG, 7, 1`

Before Instruction
FLAG_REG = `0Ah`

After Instruction
FLAG_REG = `8Ah`

| **BTFSC** | **Bit Test File, Skip if Clear** | | | |
|---|---|---|---|---|
| **Syntax** | `BTFSC f, b {,a}` | | | |
| Operands | 0 ≤ f ≤ 255<br>0 ≤ b ≤ 7<br>a ∈ [0, 1] | | | |
| Operation | Skip if (f<b>) = `0` | | | |
| Status Affected | None | | | |
| Encoding | `1011` | `bbba` | `ffff` | `ffff` |
| Description | If bit ‘b’ in register ‘f’ is ‘`0`’, then the next instruction is skipped. If bit ‘b’ is ‘`0`’, then the next instruction fetched during the current instruction execution is discarded and a `NOP` is executed instead, making this a two-cycle instruction.<br>If ‘a’ is ‘`0`’, the Access Bank is selected. If ‘a’ is ‘`1`’, the BSR is used to select the GPR bank.<br>If ‘a’ is ‘`0`’ and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ `95` (`5Fh`). See Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode for details. | | | |
| Words | 1 | | | |
| Cycles | 1 (2)<br>**Note:** Three cycles if skip and followed by a two-word instruction. Four cycles if skip and followed by a three-word instruction. | | | |

Q Cycle Activity:

If no skip:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register ‘f’ | Process Data | No operation |

If skip:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register ‘f’ | Process Data | No operation |
| No operation | No operation | No operation | No operation |

If skip and followed by two-word instruction:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register ‘f’ | Process Data | No operation |
| No operation | No operation | No operation | No operation |
| No operation | No operation | No operation | No operation |

If skip and followed by three-word instruction:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register ‘f’ | Process Data | No operation |
| No operation | No operation | No operation | No operation |
| No operation | No operation | No operation | No operation |
| No operation | No operation | No operation | No operation |

Example:

```
HERE   BTFSC   FLAG, 1, 0
FALSE:
TRUE:
```

Before Instruction
PC = address `(HERE)`

After Instruction
If FLAG<1> = `0`; PC = address `(TRUE)`
If FLAG<1> = `1`; PC = address `(FALSE)`

| **BTFSS** | **Bit Test File, Skip if Set** | | | |
|---|---|---|---|---|
| **Syntax** | **`BTFSS f, b {,a}`** | | | |
| Operands | 0 ≤ f ≤ 255<br>0 ≤ b ≤ 7<br>a ∈ [0, 1] | | | |
| Operation | Skip if (f<b>) = `1` | | | |
| Status Affected | None | | | |
| Encoding | `1010` | `bbba` | `ffff` | `ffff` |
| Description | If bit ‘b’ in register ‘f’ is ‘`1`’, then the next instruction is skipped. If bit ‘b’ is ‘`1`’, then the next instruction fetched during the current instruction execution is discarded and a `NOP` is executed instead, making this a two-cycle instruction.<br>If ‘a’ is ‘`0`’, the Access Bank is selected. If ‘a’ is ‘`1`’, the BSR is used to select the GPR bank.<br>If ‘a’ is ‘`0`’ and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ `95` (`5Fh`). See Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode for details. | | | |

| .....continued | |
|---|---|
| **BTFSS** | **Bit Test File, Skip if Set** |
| **Syntax** | `BTFSS f, b {,a}` |
| Words | 1 |
| Cycles | 1 (2)<br>**Note:** Three cycles if skip and followed by a two-word instruction. Four cycles if skip and followed by a three-word instruction. |

Q Cycle Activity:

If no skip:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | No operation |

If skip:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | No operation |
| No operation | No operation | No operation | No operation |

If skip and followed by two-word instruction:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | No operation |
| No operation | No operation | No operation | No operation |
| No operation | No operation | No operation | No operation |

If skip and followed by three-word instruction:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | No operation |
| No operation | No operation | No operation | No operation |
| No operation | No operation | No operation | No operation |
| No operation | No operation | No operation | No operation |

Example:

```
HERE   BTFSS   FLAG, 1, 0
FALSE:
TRUE:
```

Before Instruction
PC = address `(HERE)`

After Instruction
If FLAG<1> = `0`; PC = address `(FALSE)`
If FLAG<1> = `1`; PC = address `(TRUE)`

| BTG | Bit Toggle f | | | |
|---|---|---|---|---|
| **Syntax** | `BTG f, b {,a}` | | | |
| Operands | 0 ≤ f ≤ 255<br>0 ≤ b ≤ 7<br>a ∈ [0, 1] | | | |
| Operation | (f<b>) → f<b> | | | |
| Status Affected | None | | | |
| Encoding | 0111 | bbba | ffff | ffff |
| Description | Bit 'b' in data memory location 'f' is inverted.<br>If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.<br>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode for details. | | | |
| Words | 1 | | | |
| Cycles | 1 | | | |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write register 'f' |

Example: `BTG PORTC, 4, 0`

Before Instruction
PORTC = 0111 0101 [75h]

After Instruction
PORTC = 0110 0101 [65h]

| BZ | Branch if Zero | | | |
|---|---|---|---|---|
| **Syntax** | `BZ n` | | | |
| Operands | -128 ≤ n ≤ 127 | | | |
| Operation | If ZERO bit is '1'<br>(PC) + 2 + 2n → PC | | | |
| Status Affected | None | | | |
| Encoding | 1110 | 0000 | nnnn | nnnn |
| Description | If the ZERO bit is '1', then the program will branch. The two's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a two-cycle instruction. | | | |
| Words | 1 | | | |
| Cycles | 1 (2) | | | |

Q Cycle Activity:

If Jump:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'n' | Process Data | Write to PC |
| No operation | No operation | No operation | No operation |

If No Jump:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'n' | Process Data | No operation |

Example: `HERE BOV Jump`

Before Instruction
PC = address `(HERE)`

After Instruction
If ZERO = `1`; PC = address `(Jump)`
If ZERO = `0`; PC = address `(HERE + 2)`

| **CALL** | **Subroutine Call** | | | |
|---|---|---|---|---|
| **Syntax** | **`CALL k {,s}`** | | | |
| Operands | 0 ≤ k ≤ 1048575<br>s ∈ [0, 1] | | | |
| Operation | (PC) + 4 → TOS<br>k → PC<20:1><br><br>If s = `1`<br>(W) → WREG_CSHAD<br>(STATUS) → STATUS_CSHAD<br>(BSR) → BSR_CSHAD | | | |
| Status Affected | None | | | |
| Encoding<br>1st word (k<7:0>)<br>2nd word (k<19:8>) | `1110` | `110s` | $k_7kkk$ | $kkkk_0$ |
| | `1111` | $k_{19}kkk$ | `kkkk` | $kkkk_8$ |
| Description | Subroutine call of entire 2-Mbyte memory range. First, return address (PC + 4) is pushed onto the return stack. If 's' = `1`, the WREG, STATUS and BSR registers are also pushed into their respective shadow registers WREG_CSHAD, STATUS_CSHAD and BSR_CSHAD. If 's' = `0`, no update occurs (default). Then, the 20-bit value 'k' is loaded into PC<20:1>. CALL is a two-cycle instruction. | | | |
| Words | 2 | | | |
| Cycles | 2 | | | |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'k'<7:0> | PUSH PC to stack | Read literal 'k'<19:8><br>Write to PC |

| | | | |
|---|---|---|---|
| No operation | No operation | No operation | No operation |

Example: `HERE CALL THERE, 1`

Before Instruction

PC = address `(HERE)`

After Instruction

PC = address `(THERE)`
TOS = address `(HERE + 4)`
WREG_CSHAD = (WREG)
BSR_CSHAD = (BSR)
STATUS_CSHAD = (STATUS)

| **CALLW** | **Subroutine Call using WREG** | | | |
|---|---|---|---|---|
| **Syntax** | **`CALLW`** | | | |
| Operands | None | | | |
| Operation | (PC) + 2 → TOS<br>(W) → PCL<br>(PCLATH) → PCH<br>(PCLATU) → PCU | | | |
| Status Affected | None | | | |
| Encoding | `0000` | `0000` | `0001` | `0100` |
| Description | First, the return address (PC + 2) is pushed onto the return stack. Next, the contents of W are written to PCL; the existing value is discarded. Then, the contents of PCLATH and PCLATU are latched onto PCH and PCU respectively. The second cycle is executed as a `NOP` instruction while the new next instruction is fetched. Unlike `CALL`, there is no option to update W, STATUS or BSR. | | | |
| Words | 1 | | | |
| Cycles | 2 | | | |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read WREG | PUSH PC to stack | No operation |
| No operation | No operation | No operation | No operation |

Example: `HERE CALLW`

Before Instruction

PC = address `(HERE)`
PCLATH = `10h`
PCLATU = `00h`
W = `06h`

After Instruction

PC = address `001006h`
TOS = address `(HERE + 2)`

PCLATH = 10h
PCLATU = 00h
W = 06h

| CLRF | Clear f | | | |
|---|---|---|---|---|
| **Syntax** | **CLRF f {,a}** | | | |
| Operands | 0 ≤ f ≤ 255<br>a ∈ [0, 1] | | | |
| Operation | 000h → f<br>1 → Z | | | |
| Status Affected | Z | | | |
| Encoding | 0110 | 101a | ffff | ffff |
| Description | Clears the contents of the specified register 'f'.<br>If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.<br>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode for details. | | | |
| Words | 1 | | | |
| Cycles | 1 | | | |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write register 'f' |

Example: CLRF FLAG_REG, 1

Before Instruction
FLAG_REG = 5Ah

After Instruction
FLAG_REG = 00h

| CLRWDT | Clear Watchdog Timer | | | |
|---|---|---|---|---|
| **Syntax** | **CLRWDT** | | | |
| Operands | None | | | |
| Operation | 000h → WDT<br>1 → $\overline{TO}$<br>1 → $\overline{PD}$ | | | |
| Status Affected | $\overline{TO}$, $\overline{PD}$ | | | |
| Encoding | 0000 | 0000 | 0000 | 0100 |
| Description | CLRWDT instruction resets the Watchdog Timer. It also resets the STATUS bits, and $\overline{TO}$ and $\overline{PD}$ are set. | | | |
| Words | 1 | | | |

© 2019-2023 Microchip Technology Inc. and its subsidiaries

| .....continued | |
|---|---|
| **CLRWDT** | **Clear Watchdog Timer** |
| **Syntax** | `CLRWDT` |
| Cycles | 1 |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | No operation | Process Data | No operation |

Example: `CLRWDT`

Before Instruction
WDT Counter = `?`

After Instruction
WDT Counter = `00h`
$\overline{TO}$ = `1`
$\overline{PD}$ = `1`

| **COMF** | **Complement f** | | | |
|---|---|---|---|---|
| **Syntax** | `COMF f {,d {,a}}` | | | |
| Operands | 0 ≤ f ≤ 255<br>d ∈ [0, 1]<br>a ∈ [0, 1] | | | |
| Operation | (f̄) → dest | | | |
| Status Affected | N, Z | | | |
| Encoding | `0001` | `11da` | `ffff` | `ffff` |
| Description | The contents of register ‘f’ are complemented. If ‘d’ is ‘`0`’, the result is stored in W. If ‘d’ is ‘`1`’, the result is stored back in the register ‘f’ (default).<br>If ‘a’ is ‘`0`’, the Access Bank is selected. If ‘a’ is ‘`1`’, the BSR is used to select the GPR bank.<br>If ‘a’ is ‘`0`’ and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ `95` (`5Fh`). See Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode for details. | | | |
| Words | 1 | | | |
| Cycles | 1 | | | |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register ‘f’ | Process Data | Write to destination |

Example: `COMF REG0, 0, 0`

Before Instruction
REG = `13h`

After Instruction
REG = 13h
W = ECh

| CPFSEQ | Compare f with W, Skip if f = W | | | |
|---|---|---|---|---|
| **Syntax** | **CPFSEQ f {,a}** | | | |
| Operands | 0 ≤ f ≤ 255<br>a ∈ [0, 1] | | | |
| Operation | (f) – (W), skip if (f) = (W)<br>(unsigned comparison) | | | |
| Status Affected | None | | | |
| Encoding | 0110 | 001a | ffff | ffff |
| Description | Compares the contents of data memory location 'f' to the contents of W by performing an unsigned subtraction. If the contents of 'f' are equal to the contents of WREG, then the fetched instruction is discarded and a NOP is executed instead, making this a two-cycle instruction.<br>If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.<br>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode for details. | | | |
| Words | 1 | | | |
| Cycles | 1 (2)<br>**Note:** Three cycles if skip and followed by a two-word instruction. Four cycles if skip and followed by a three-word instruction. | | | |

Q Cycle Activity:

If no skip:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | No operation |

If skip:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | No operation |
| No operation | No operation | No operation | No operation |

If skip and followed by two-word instruction:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | No operation |
| No operation | No operation | No operation | No operation |
| No operation | No operation | No operation | No operation |

If skip and followed by three-word instruction:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register ‘f’ | Process Data | No operation |
| No operation | No operation | No operation | No operation |
| No operation | No operation | No operation | No operation |
| No operation | No operation | No operation | No operation |

Example:

```
HERE   CPFSEQ   REG, 0
NEQUAL:
EQUAL:
```

Before Instruction
PC = address (HERE)
W = ?
REG = ?

After Instruction
If REG = W; PC = address (EQUAL)
If REG ≠ W; PC = address (NEQUAL)

| **CPFSGT** | **Compare f with W, Skip if f > W** | | | |
|---|---|---|---|---|
| **Syntax** | **CPFSGT f {,a}** | | | |
| Operands | 0 ≤ f ≤ 255<br>a ∈ [0, 1] | | | |
| Operation | (f) – (W), skip if (f) > (W)<br>(unsigned comparison) | | | |
| Status Affected | None | | | |
| Encoding | 0110 | 010a | ffff | ffff |
| Description | Compares the contents of data memory location ‘f’ to the contents of W by performing an unsigned subtraction. If the contents of ‘f’ are greater than the contents of WREG, then the fetched instruction is discarded and a NOP is executed instead, making this a two-cycle instruction.<br>If ‘a’ is ‘0’, the Access Bank is selected. If ‘a’ is ‘1’, the BSR is used to select the GPR bank.<br>If ‘a’ is ‘0’ and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode for details. | | | |
| Words | 1 | | | |
| Cycles | 1 (2)<br>**Note:** Three cycles if skip and followed by a two-word instruction. Four cycles if skip and followed by a three-word instruction. | | | |

Q Cycle Activity:

If no skip:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register ‘f’ | Process Data | No operation |

If skip:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register ‘f’ | Process Data | No operation |
| No operation | No operation | No operation | No operation |

If skip and followed by two-word instruction:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register ‘f’ | Process Data | No operation |
| No operation | No operation | No operation | No operation |
| No operation | No operation | No operation | No operation |

If skip and followed by three-word instruction:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register ‘f’ | Process Data | No operation |
| No operation | No operation | No operation | No operation |
| No operation | No operation | No operation | No operation |
| No operation | No operation | No operation | No operation |

Example:

```
HERE   CPFSGT   REG, 0
NGREATER:
GREATER:
```

Before Instruction
PC = address `(HERE)`
W = ?
REG = ?

After Instruction
If REG > W; PC = address `(GREATER)`
If REG ≤ W; PC = address `(NGREATER)`

| **CPFSLT** | **Compare f with W, Skip if f < W** | | | |
|---|---|---|---|---|
| **Syntax** | `CPFSLT f {,a}` | | | |
| Operands | 0 ≤ f ≤ 255<br>a ∈ [0, 1] | | | |
| Operation | (f) – (W), skip if (f) < (W)<br>(unsigned comparison) | | | |
| Status Affected | None | | | |
| Encoding | 0110 | 000a | ffff | ffff |

| .....continued | |
|---|---|
| **CPFSLT** | **Compare f with W, Skip if f < W** |
| **Syntax** | `CPFSLT f {,a}` |
| Description | Compares the contents of data memory location 'f' to the contents of W by performing an unsigned subtraction. If the contents of 'f' are less than the contents of WREG, then the fetched instruction is discarded and a `NOP` is executed instead, making this a two-cycle instruction.<br>If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.<br>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode for details. |
| Words | 1 |
| Cycles | 1 (2)<br>**Note:** Three cycles if skip and followed by a two-word instruction. Four cycles if skip and followed by a three-word instruction. |

Q Cycle Activity:

If no skip:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | No operation |

If skip:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | No operation |
| No operation | No operation | No operation | No operation |

If skip and followed by two-word instruction:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | No operation |
| No operation | No operation | No operation | No operation |
| No operation | No operation | No operation | No operation |

If skip and followed by three-word instruction:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | No operation |
| No operation | No operation | No operation | No operation |
| No operation | No operation | No operation | No operation |
| No operation | No operation | No operation | No operation |

© 2019-2023 Microchip Technology Inc. and its subsidiaries

Example:

```
HERE   CPFSLT   REG, 1
NLESS:
LESS:
```

Before Instruction
PC = address (HERE)
W = ?
REG = ?

After Instruction
If REG < W; PC = address (LESS)
If REG ≥ W; PC = address (NLESS)

| DAW | Decimal Adjust W Register | | | |
|---|---|---|---|---|
| **Syntax** | **DAW** | | | |
| Operands | None | | | |
| Operation | If [(W<3:0>) > 9] or [DC = 1] then<br>(W<3:0>) + 6 → W<3:0>;<br>else<br>(W<3:0>) → W<3:0>;<br>If [(W<7:4>) + DC > 9] or [C = 1] then<br>(W<7:4>) + 6 + DC → W<7:4>;<br>else<br>(W<7:4>) + DC → W<7:4> | | | |
| Status Affected | C | | | |
| Encoding | 0000 | 0000 | 0000 | 0111 |
| Description | DAW adjusts the 8-bit value in W, resulting from the earlier addition of two variables (each in packed BCD format) and produces a correct packed BCD result. | | | |
| Words | 1 | | | |
| Cycles | 1 | | | |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register W | Process Data | Write register W |

Example 1: DAW

Before Instruction
W = A5h
C = 0
DC = 0

After Instruction
W = 05h
C = 1
DC = 0

Example 2: DAW

Before Instruction
W = CEh
C = 0
DC = 0

After Instruction
W = 34h
C = 1
DC = 0

| DECF | Decrement f | | | |
|---|---|---|---|---|
| **Syntax** | **DECF f {,d {,a}}** | | | |
| Operands | 0 ≤ f ≤ 255<br>d ∈ [0, 1]<br>a ∈ [0, 1] | | | |
| Operation | (f) – 1 → dest | | | |
| Status Affected | C, DC, N, OV, Z | | | |
| Encoding | 0000 | 01da | ffff | ffff |
| Description | Decrement register ‘f’. If ‘d’ is ‘0’, the result is stored in W. If ‘d’ is ‘1’, the result is stored back in the register ‘f’ (default).<br>If ‘a’ is ‘0’, the Access Bank is selected. If ‘a’ is ‘1’, the BSR is used to select the GPR bank.<br>If ‘a’ is ‘0’ and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode for details. | | | |
| Words | 1 | | | |
| Cycles | 1 | | | |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register ‘f’ | Process Data | Write to destination |

Example: DECF CNT, 1, 0

Before Instruction
CNT = 01h
Z = 0

After Instruction
CNT = 00h
Z = 1

| DECFSZ | Decrement f, Skip if 0 |
|---|---|
| **Syntax** | **DECFSZ f {,d {,a}}** |
| Operands | 0 ≤ f ≤ 255<br>d ∈ [0, 1]<br>a ∈ [0, 1] |

| .....continued | | | | |
|---|---|---|---|---|
| **DECFSZ** | **Decrement f, Skip if 0** | | | |
| **Syntax** | `DECFSZ f {,d {,a}}` | | | |
| Operation | (f) – 1 → dest, skip if result = 0 | | | |
| Status Affected | None | | | |
| Encoding | 0010 | 11da | ffff | ffff |
| Description | The contents of register 'f' are decremented. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default).<br>If the result is '0', the next instruction, which is already fetched, is discarded and a NOP is executed instead, making it a two-cycle instruction.<br>If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.<br>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode for details. | | | |
| Words | 1 | | | |
| Cycles | 1 (2)<br>**Note:** Three cycles if skip and followed by a two-word instruction. Four cycles if skip and followed by a three-word instruction. | | | |

Q Cycle Activity:

If no skip:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

If skip:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |
| No operation | No operation | No operation | No operation |

If skip and followed by two-word instruction:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |
| No operation | No operation | No operation | No operation |
| No operation | No operation | No operation | No operation |

If skip and followed by three-word instruction:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |
| No operation | No operation | No operation | No operation |
| No operation | No operation | No operation | No operation |

© 2019-2023 Microchip Technology Inc. and its subsidiaries

| No operation | No operation | No operation | No operation |
|---|---|---|---|

Example:

```
HERE    DECFSZ    CNT, 1, 1
        GOTO      LOOP
CONTINUE
```

Before Instruction
CNT = ?
PC = address `(HERE)`

After Instruction
CNT = CNT – 1
If CNT = `0`; PC = address `(CONTINUE)`
If CNT ≠ `0`; PC = address `(HERE + 2)`

| **DCFSNZ** | **Decrement f, Skip if not 0** | | | |
|---|---|---|---|---|
| **Syntax** | `DCFSNZ f {,d {,a}}` | | | |
| Operands | 0 ≤ f ≤ 255<br>d ∈ [0, 1]<br>a ∈ [0, 1] | | | |
| Operation | (f) – 1 → dest, skip if result ≠ `0` | | | |
| Status Affected | None | | | |
| Encoding | `0100` | `11da` | `ffff` | `ffff` |
| Description | The contents of register ‘f’ are decremented. If ‘d’ is ‘`0`’, the result is placed in W. If ‘d’ is ‘`1`’, the result is placed back in register ‘f’ (default).<br>If the result is not ‘`0`’, the next instruction, which is already fetched, is discarded and a `NOP` is executed instead, making it a two-cycle instruction.<br>If ‘a’ is ‘`0`’, the Access Bank is selected. If ‘a’ is ‘`1`’, the BSR is used to select the GPR bank.<br>If ‘a’ is ‘`0`’ and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ `95` (`5Fh`). See Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode for details. | | | |
| Words | 1 | | | |
| Cycles | 1 (2)<br>**Note:** Three cycles if skip and followed by a two-word instruction. Four cycles if skip and followed by a three-word instruction. | | | |

Q Cycle Activity:

If no skip:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register ‘f’ | Process Data | Write to destination |

If skip:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register ‘f’ | Process Data | Write to destination |

| | | | |
|---|---|---|---|
| No operation | No operation | No operation | No operation |

If skip and followed by two-word instruction:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register ‘f’ | Process Data | Write to destination |
| No operation | No operation | No operation | No operation |
| No operation | No operation | No operation | No operation |

If skip and followed by three-word instruction:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register ‘f’ | Process Data | Write to destination |
| No operation | No operation | No operation | No operation |
| No operation | No operation | No operation | No operation |
| No operation | No operation | No operation | No operation |

Example:

```
HERE   DCFSNZ   TEMP, 1, 0
ZERO:
NZERO:
```

Before Instruction
TEMP = ?
PC = address `(HERE)`

After Instruction
TEMP = TEMP – 1
If TEMP = `0`; PC = address `(ZER0)`
If TEMP ≠ `0`; PC = address `(NZERO)`

| **GOTO** | **Unconditional Branch** | | | |
|---|---|---|---|---|
| **Syntax** | **`GOTO k`** | | | |
| Operands | 0 ≤ k ≤ 1048575 | | | |
| Operation | k → PC<20:1> | | | |
| Status Affected | None | | | |
| Encoding<br>1st word (k<7:0>)<br>2nd word (k<19:8>) | `1110` | `1111` | $k_7kkk$ | $kkkk_0$ |
| | `1111` | $k_{19}kkk$ | `kkkk` | $kkkk_8$ |
| Description | `GOTO` allows an unconditional branch anywhere within entire 2-Mbyte memory range. The 20-bit value ‘k’ is loaded into PC<20:1>. `GOTO` is always a two-cycle instruction. | | | |
| Words | 2 | | | |
| Cycles | 2 | | | |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'k'<7:0> | No operation | Read literal 'k'<19:8> Write to PC |
| No operation | No operation | No operation | No operation |

Example: `HERE GOTO THERE`

Before Instruction

PC = address `(HERE)`

After Instruction

PC = address `(THERE)`

| INCF | Increment f | | | |
|---|---|---|---|---|
| **Syntax** | `INCF f {,d {,a}}` | | | |
| Operands | 0 ≤ f ≤ 255<br>d ∈ [0, 1]<br>a ∈ [0, 1] | | | |
| Operation | (f) + 1 → dest | | | |
| Status Affected | C, DC, N, OV, Z | | | |
| Encoding | 0010 | 10da | ffff | ffff |
| Description | The contents of register 'f' are incremented. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in the register 'f' (default).<br>If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.<br>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode for details. | | | |
| Words | 1 | | | |
| Cycles | 1 | | | |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example: `INCF CNT, 1, 0`

Before Instruction
CNT = FFh
Z = 0
C = ?
DC = ?

After Instruction
CNT = 00h
Z = 1

C = 1
DC = 1

| INCFSZ | Increment f, Skip if 0 | | | |
|---|---|---|---|---|
| **Syntax** | `INCFSZ f {,d {,a}}` | | | |
| Operands | 0 ≤ f ≤ 255<br>d ∈ [0, 1]<br>a ∈ [0, 1] | | | |
| Operation | (f) + 1 → dest, skip if result = 0 | | | |
| Status Affected | None | | | |
| Encoding | 0011 | 11da | ffff | ffff |
| Description | The contents of register 'f' are incremented. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default).<br>If the result is '0', the next instruction, which is already fetched, is discarded and a NOP is executed instead, making it a two-cycle instruction.<br>If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.<br>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode for details. | | | |
| Words | 1 | | | |
| Cycles | 1 (2)<br>**Note:** Three cycles if skip and followed by a two-word instruction. Four cycles if skip and followed by a three-word instruction. | | | |

Q Cycle Activity:

If no skip:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

If skip:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |
| No operation | No operation | No operation | No operation |

If skip and followed by two-word instruction:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |
| No operation | No operation | No operation | No operation |
| No operation | No operation | No operation | No operation |

If skip and followed by three-word instruction:

© 2019-2023 Microchip Technology Inc. and its subsidiaries

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |
| No operation | No operation | No operation | No operation |
| No operation | No operation | No operation | No operation |
| No operation | No operation | No operation | No operation |

Example:

```
HERE   INCFSZ   CNT, 1, 0
NZERO:
ZERO:
```

Before Instruction
CNT = ?
PC = address (HERE)

After Instruction
CNT = CNT + 1
If CNT = 0; PC = address (ZERO)
If CNT ≠ 0; PC = address (NZERO)

| **INFSNZ** | **Increment f, Skip if not 0** | | | |
|---|---|---|---|---|
| **Syntax** | **INFSNZ f {,d {,a}}** | | | |
| Operands | 0 ≤ f ≤ 255<br>d ∈ [0, 1]<br>a ∈ [0, 1] | | | |
| Operation | (f) + 1 → dest, skip if result ≠ 0 | | | |
| Status Affected | None | | | |
| Encoding | 0100 | 10da | ffff | ffff |
| Description | The contents of register 'f' are incremented. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default).<br>If the result is not '0', the next instruction, which is already fetched, is discarded and a NOP is executed instead, making it a two-cycle instruction.<br>If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.<br>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode for details. | | | |
| Words | 1 | | | |
| Cycles | 1 (2)<br>**Note:** Three cycles if skip and followed by a two-word instruction. Four cycles if skip and followed by a three-word instruction. | | | |

Q Cycle Activity:

If no skip:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|

| Decode | Read register ‘f’ | Process Data | Write to destination |
|---|---|---|---|

If skip:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register ‘f’ | Process Data | Write to destination |
| No operation | No operation | No operation | No operation |

If skip and followed by two-word instruction:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register ‘f’ | Process Data | Write to destination |
| No operation | No operation | No operation | No operation |
| No operation | No operation | No operation | No operation |

If skip and followed by three-word instruction:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register ‘f’ | Process Data | Write to destination |
| No operation | No operation | No operation | No operation |
| No operation | No operation | No operation | No operation |
| No operation | No operation | No operation | No operation |

Example:

```
HERE   INFSNZ   REG, 1, 0
ZERO:
NZERO:
```

Before Instruction
REG = ?
PC = address `(HERE)`

After Instruction
REG = REG + 1
If REG = `0`; PC = address `(ZER0)`
If REG ≠ `0`; PC = address `(NZERO)`

| **IORLW** | **Inclusive OR Literal with W** | | | |
|---|---|---|---|---|
| **Syntax** | **`IORLW k`** | | | |
| Operands | 0 ≤ k ≤ 255 | | | |
| Operation | (W) .OR. k → W | | | |
| Status Affected | N, Z | | | |
| Encoding | `0000` | `1001` | `kkkk` | `kkkk` |
| Description | The contents of W are ORed with the 8-bit literal ‘k’. The result is placed in W. | | | |
| Words | 1 | | | |
| Cycles | 1 | | | |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'k' | Process Data | Write to W |

Example: `IORLW 35h`

Before Instruction
W = `9Ah`

After Instruction
W = `BFh`

| IORWF | Inclusive OR W with f | | | |
|---|---|---|---|---|
| **Syntax** | `IORWF f {,d {,a}}` | | | |
| Operands | 0 ≤ f ≤ 255<br>d ∈ [0, 1]<br>a ∈ [0, 1] | | | |
| Operation | (W) .OR. (f) → dest | | | |
| Status Affected | N, Z | | | |
| Encoding | `0001` | `00da` | `ffff` | `ffff` |
| Description | Inclusive OR W with register 'f'. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in the register 'f' (default).<br>If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.<br>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode for details. | | | |
| Words | 1 | | | |
| Cycles | 1 | | | |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example: `IORWF RESULT, 0, 1`

Before Instruction
RESULT = `13h`
W = `91h`

After Instruction
RESULT = `13h`
W = `93h`

| LFSR | Load FSR | | | |
|---|---|---|---|---|
| **Syntax** | `LFSR` $f_n$, k | | | |
| Operands | $0 \le f_n \le 2$<br>$0 \le k \le 16383$ | | | |
| Operation | $k \rightarrow FSRf_n$ | | | |
| Status Affected | None | | | |
| Encoding | 1110 | 1110 | $00f_nf_n$ | $k_{13}kkk_{10}$ |
| | 1111 | $00k_9k$ | kkkk | $kkkk_0$ |
| Description | The 14-bit literal 'k' is loaded into the File Select Register '$f_n$'. | | | |
| Words | 2 | | | |
| Cycles | 2 | | | |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'k'<13:10> | Process Data | Write literal 'k'<13:10> to $FSRf_n$<13:10> |
| No operation | Read literal 'k'<9:0> | No operation | Write literal 'k'<9:0> to $FSRf_n$<9:0> |

Example: `LFSR 2, 3ABh`

Before Instruction
FSR2H = ?
FSR2L = ?

After Instruction
FSR2H = `03h`
FSR2L = `ABh`

| MOVF | Move f | | | |
|---|---|---|---|---|
| **Syntax** | `MOVF f {,d {,a}}` | | | |
| Operands | $0 \le f \le 255$<br>$d \in [0, 1]$<br>$a \in [0, 1]$ | | | |
| Operation | (f) → dest | | | |
| Status Affected | N, Z | | | |
| Encoding | 0101 | 00da | ffff | ffff |
| Description | The contents of register 'f' are moved to a destination. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in the register 'f' (default).<br>If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.<br>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode for details. | | | |

| .....continued | |
|---|---|
| **MOVF** | **Move f** |
| **Syntax** | `MOVF f {,d {,a}}` |
| Words | 1 |
| Cycles | 1 |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example: `MOVF REG, 0, 0`

Before Instruction
REG = `22h`
W = `FFh`

After Instruction
REG = `22h`
W = `22h`

| **MOVFF** | **Move f to f** | | | |
|---|---|---|---|---|
| **Syntax** | `MOVFF` $f_s$, $f_d$ | | | |
| Operands | $0 \le f_s \le 4095$<br>$0 \le f_d \le 4095$ | | | |
| Operation | $(f_s) \rightarrow f_d$ | | | |
| Status Affected | None | | | |
| Encoding | 1100 | $f_s f_s f_s f_s$ | $f_s f_s f_s f_s$ | $f_s f_s f_s f_s$ |
| | 1111 | $f_d f_d f_d f_d$ | $f_d f_d f_d f_d$ | $f_d f_d f_d f_d$ |
| Description | The contents of source register '$f_s$' are moved to destination register '$f_d$'. Location of source '$f_s$' can be anywhere in the 4096-byte data space (`000h` to `FFFh`) and location of destination '$f_d$' can also be anywhere from `000h` to `FFFh`.<br>`MOVFF` is particularly useful for transferring a data memory location to a peripheral register (such as the transmit buffer or an I/O port).<br>The `MOVFF` instruction cannot use the PCL, TOSU, TOSH or TOSL as the destination register.<br>**Note:**<br>`MOVFF` has curtailed the source and destination range to the lower 4 Kbyte space of memory (Banks 1 through 15). For everything else, use `MOVFFL`. | | | |
| Words | 2 | | | |
| Cycles | 2 | | | |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|

| Decode | Read register '$f_s$' | Process Data | No operation |
|---|---|---|---|
| Decode | No operation<br>No dummy read | No operation | Write register '$f_d$' |

Example: `MOVFF REG1, REG2`

Before Instruction
Address of REG1 = `100h`
Address of REG2 = `200h`
REG1 = `33h`
REG2 = `11h`

After Instruction
Address of REG1 = `100h`
Address of REG2 = `200h`
REG1 = `33h`
REG2 = `33h`

| **MOVFFL** | **Move f to f (Long Range)** | | | |
|---|---|---|---|---|
| **Syntax** | **`MOVFFL` $f_s$, $f_d$** | | | |
| Operands | $0 \le f_s \le 16383$<br>$0 \le f_d \le 16383$ | | | |
| Operation | $(f_s) \rightarrow f_d$ | | | |
| Status Affected | None | | | |
| Encoding | `0000` | `0000` | `0110` | $f_s f_s f_s f_s$ |
| | `1111` | $f_s f_s f_s f_s$ | $f_s f_s f_s f_s$ | $f_s f_s f_d f_d$ |
| | `1111` | $f_d f_d f_d f_d$ | $f_d f_d f_d f_d$ | $f_d f_d f_d f_d$ |
| Description | The contents of source register '$f_s$' are moved to destination register '$f_d$'. Location of source '$f_s$' can be anywhere in the 16 Kbyte data space (`0000h` to `3FFFh`) and location of destination '$f_d$' can also be anywhere from `0000h` to `3FFFh`. Either source or destination can be W (a useful special situation).<br>`MOVFFL` is particularly useful for transferring a data memory location to a peripheral register (such as the transmit buffer or an I/O port).<br>The `MOVFFL` instruction cannot use the PCL, TOSU, TOSH or TOSL as the destination register. | | | |
| Words | 3 | | | |
| Cycles | 3 | | | |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | No operation | No operation | No operation |
| Decode | Read register '$f_s$' | Process Data | No operation |
| Decode | No operation<br>No dummy read | No operation | Write register '$f_d$' |

Example: `MOVFFL 2000h, 200Ah`

Before Instruction
Contents of `2000h` = `33h`
Contents of `200Ah` = `11h`

After Instruction
Contents of `2000h` = `33h`
Contents of `200Ah` = `33h`

| MOVLB | Move Literal to BSR | | | |
|---|---|---|---|---|
| **Syntax** | **`MOVLB k`** | | | |
| Operands | 0 ≤ k ≤ 63 | | | |
| Operation | k → BSR | | | |
| Status Affected | None | | | |
| Encoding | `0000` | `0001` | `00kk` | `kkkk` |
| Description | The 6-bit literal ‘k’ is loaded into the Bank Select Register (BSR<5:0>). The value of BSR<7:6> always remains ‘`0`’. | | | |
| Words | 1 | | | |
| Cycles | 1 | | | |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal ‘k’ | Process Data | Write to BSR |

Example: `MOVLB 5`

Before Instruction
BSR = `02h`

After Instruction
BSR = `05h`

| MOVLW | Move Literal to W | | | |
|---|---|---|---|---|
| **Syntax** | **`MOVLW k`** | | | |
| Operands | 0 ≤ k ≤ 255 | | | |
| Operation | k → W | | | |
| Status Affected | None | | | |
| Encoding | `0000` | `1110` | `kkkk` | `kkkk` |
| Description | The 8-bit literal ‘k’ is loaded into W. | | | |
| Words | 1 | | | |
| Cycles | 1 | | | |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'k' | Process Data | Write to W |

Example: `MOVLW 5Ah`

Before Instruction
W = ?

After Instruction
W = `5Ah`

| MOVWF | Move W to f | | | |
|---|---|---|---|---|
| **Syntax** | **`MOVWF f {,a}`** | | | |
| Operands | 0 ≤ f ≤ 255<br>a ∈ [0, 1] | | | |
| Operation | (W) → f | | | |
| Status Affected | None | | | |
| Encoding | `0110` | `111a` | `ffff` | `ffff` |
| Description | Move data from W to register 'f'. Location 'f' can be anywhere in the 256-byte bank.<br>If 'a' is '`0`', the Access Bank is selected. If 'a' is '`1`', the BSR is used to select the GPR bank.<br>If 'a' is '`0`' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ `95` (`5Fh`). See Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode for details. | | | |
| Words | 1 | | | |
| Cycles | 1 | | | |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read W | Process Data | Write register 'f' |

Example: `MOVWF REG, 0`

Before Instruction
W = `4Fh`
REG = `FFh`

After Instruction
W = `4Fh`
REG = `4Fh`

| MULLW | Multiply literal with W |
|---|---|
| **Syntax** | **`MULLW k`** |
| Operands | 0 ≤ k ≤ 255 |
| Operation | (W) x k → PRODH:PRODL |

| .....continued | | | | |
|---|---|---|---|---|
| **MULLW** | **Multiply literal with W** | | | |
| **Syntax** | **MULLW k** | | | |
| Status Affected | None | | | |
| Encoding | 0000 | 1101 | kkkk | kkkk |
| Description | An unsigned multiplication is carried out between the contents of W and the 8-bit literal ‘k’. The 16-bit result is placed in the PRODH:PRODL register pair. PRODH contains the high byte. W is unchanged.<br>None of the Status flags are affected. Note that neither overflow nor carry is possible in this operation. A zero result is possible but not detected. | | | |
| Words | 1 | | | |
| Cycles | 1 | | | |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal ‘k’ | Process Data | Write registers PRODH:PRODL |

Example: `MULLW 0C4h`

Before Instruction
W = `E2h`
PRODH = ?
PRODL = ?

After Instruction
W = `E2h`
PRODH = `ADh`
PRODL = `08h`

| **MULWF** | **Multiply W with f** | | | |
|---|---|---|---|---|
| **Syntax** | **MULWF f {,a}** | | | |
| Operands | 0 ≤ f ≤ 255<br>a ∈ [0, 1] | | | |
| Operation | (W) x (f) → PRODH:PRODL | | | |
| Status Affected | None | | | |
| Encoding | 0000 | 001a | ffff | ffff |

| .....continued | |
|---|---|
| **MULWF** | **Multiply W with f** |
| **Syntax** | **MULWF f {,a}** |
| Description | An unsigned multiplication is carried out between the contents of W and the register file location ‘f’. The 16-bit result is placed in the PRODH:PRODL register pair. PRODH contains the high byte. Both W and ‘f’ are unchanged.<br>None of the Status flags are affected. Note that neither overflow nor carry is possible in this operation. A zero result is possible but not detected.<br>If ‘a’ is ‘0’, the Access Bank is selected. If ‘a’ is ‘1’, the BSR is used to select the GPR bank.<br>If ‘a’ is ‘0’ and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode for details. |
| Words | 1 |
| Cycles | 1 |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register ‘f’ | Process Data | Write registers PRODH:PRODL |

Example: `MULWF REG, 1`

Before Instruction
W = C4h
REG = B5h
PRODH = ?
PRODL = ?

After Instruction
W = C4h
REG = B5h
PRODH = 8Ah
PRODL = 94h

| **NEGF** | **Negate f** | | | |
|---|---|---|---|---|
| **Syntax** | **NEGF f {,a}** | | | |
| Operands | 0 ≤ f ≤ 255<br>a ∈ [0, 1] | | | |
| Operation | $(\bar{f}) + 1 \rightarrow f$ | | | |
| Status Affected | N, OV, C, DC, Z | | | |
| Encoding | 0110 | 110a | ffff | ffff |

| .....continued | |
|---|---|
| **NEGF** | **Negate f** |
| **Syntax** | `NEGF f {,a}` |
| Description | Location ‘f’ is negated using two’s complement. The result is placed in the data memory location ‘f’.<br>If ‘a’ is ‘0’, the Access Bank is selected. If ‘a’ is ‘1’, the BSR is used to select the GPR bank.<br>If ‘a’ is ‘0’ and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode for details. |
| Words | 1 |
| Cycles | 1 |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register ‘f’ | Process Data | Write register ‘f’ |

Example: `NEGF REG, 1`

Before Instruction
REG = `0011 1010` [`3Ah`]

After Instruction
REG = `1100 0110` [`C6h`]

| **NOP** | **No Operation** | | | |
|---|---|---|---|---|
| **Syntax** | `NOP` | | | |
| Operands | None | | | |
| Operation | No operation | | | |
| Status Affected | None | | | |
| Encoding | 0000 | 0000 | 0000 | 0000 |
| | 1111 | xxxx | xxxx | xxxx |
| Description | No operation. | | | |
| Words | 1 | | | |
| Cycles | 1 | | | |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | No operation | No operation | No operation |

Example: None.

| POP | Pop Top of Return Stack | | | |
|---|---|---|---|---|
| **Syntax** | **POP** | | | |
| Operands | None | | | |
| Operation | (TOS) → bit bucket | | | |
| Status Affected | None | | | |
| Encoding | 0000 | 0000 | 0000 | 0110 |
| Description | The TOS value is pulled off the return stack and is discarded. The TOS value then becomes the previous value that was pushed onto the return stack. This instruction is provided to enable the user to properly manage the return stack to incorporate a software stack. (See the PUSH instruction description). | | | |
| Words | 1 | | | |
| Cycles | 1 | | | |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | No operation | POP TOS value | No operation |

Example:

```
    POP
    GOTO    NEW
```

Before Instruction
TOS = 0031A2h
Stack (1 level down) = 014332h

After Instruction
TOS = 014332h
PC = address (NEW)

| PUSH | Push Top of Return Stack | | | |
|---|---|---|---|---|
| **Syntax** | **PUSH** | | | |
| Operands | None | | | |
| Operation | (PC) + 2 → TOS | | | |
| Status Affected | None | | | |
| Encoding | 0000 | 0000 | 0000 | 0101 |
| Description | The PC + 2 is pushed onto the top of the return stack. The previous TOS value is pushed down on the stack. This instruction allows implementing a software stack by modifying TOS and then pushing it onto the return stack. (See the POP instruction description). | | | |
| Words | 1 | | | |
| Cycles | 1 | | | |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|

| | | | |
|---|---|---|---|
| Decode | PUSH PC + 2 onto return stack | No operation | No operation |

Example: `PUSH`

Before Instruction
TOS = `00345Ah`
PC = `000124h`

After Instruction
TOS = `000126h`
PC = `000126h`
Stack (1 level down) = `00345Ah`

| RCALL | Relative Call | | | |
|---|---|---|---|---|
| **Syntax** | **`RCALL n`** | | | |
| Operands | -1024 ≤ n ≤ 1023 | | | |
| Operation | (PC) + 2 → TOS<br>(PC) + 2 + 2n → PC | | | |
| Status Affected | None | | | |
| Encoding | `1101` | `1nnn` | `nnnn` | `nnnn` |
| Description | Subroutine call with a jump up to 1K from the current location. First, return address (PC + 2) is pushed onto the stack. Then, add the two's complement number '2n' to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is a two-cycle instruction. | | | |
| Words | 1 | | | |
| Cycles | 2 | | | |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'n'<br>PUSH PC to stack | Process Data | Write to PC |
| No operation | No operation | No operation | No operation |

Example: `HERE RCALL Jump`

Before Instruction
PC = address (`HERE`)

After Instruction
PC = address (`Jump`)
TOS = address (`HERE + 2`)

| RESET | Reset |
|---|---|
| **Syntax** | **`RESET`** |
| Operands | None |

.....continued

| RESET | Reset | | | |
|---|---|---|---|---|
| **Syntax** | **RESET** | | | |
| Operation | Reset all registers and flags that are affected by a $\overline{\text{MCLR}}$ Reset. | | | |
| Status Affected | All | | | |
| Encoding | 0000 | 0000 | 1111 | 1111 |
| Description | This instruction provides a way to execute a $\overline{\text{MCLR}}$ Reset by software. | | | |
| Words | 1 | | | |
| Cycles | 1 | | | |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Start Reset | No operation | No operation |

Example: `RESET`

Before Instruction
All Registers = ?
All Flags = ?

After Instruction
All Registers = Reset Value
All Flags = Reset Value

| RETFIE | Return from Interrupt | | | |
|---|---|---|---|---|
| **Syntax** | **RETFIE {s}** | | | |
| Operands | s ∈ [0, 1] | | | |
| Operation | (TOS) → PC<br>If s = 1, context is restored into WREG, STATUS, BSR, FSR0H, FSR0L, FSR1H, FSR1L, FSR2H, FSR2L, PRODH, PRODL, PCLATH and PCLATU registers from the corresponding shadow registers.<br>If s = 0, there is no change in status of any register.<br>PCLATU, PCLATH are unchanged. | | | |
| Status Affected | STAT bits in INTCONx register | | | |
| Encoding | 0000 | 0000 | 0001 | 000s |

| .....continued | |
|---|---|
| **RETFIE** | **Return from Interrupt** |
| **Syntax** | `RETFIE {s}` |
| Description | Return from interrupt. Stack is popped and Top-of-Stack (TOS) is loaded into the PC. Interrupts are enabled by setting either the high- or low-priority Global Interrupt Enable bit.<br>If 's' = `1`, the contents of the shadow registers WREG_SHAD, STATUS_SHAD, BSR_SHAD, FSR0H_SHAD, FSR0L_SHAD, FSR1H_SHAD, FSR1L_SHAD, FSR2H_SHAD, FSR2L_SHAD, PRODH_SHAD, PRODL_SHAD, PCLATH_SHAD and PCLATU_SHAD are loaded into corresponding registers. There are two sets of shadow registers, main context and low context. The set retrieved on `RETFIE` instruction execution depends on what the state of operation of the CPU was when `RETFIE` was executed.<br>If 's' = `0`, no update of these registers occurs (default).<br>The upper and high address latches (PCLATU/H) remain unchanged. |
| Words | 1 |
| Cycles | 2 |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | No operation | Process Data | POP PC from stack |
| No operation | No operation | No operation | No operation |

Example: `RETFIE 1`

After Instruction
PC = (TOS)
WREG = (WREG_SHAD)
BSR = (BSR_SHAD)
STATUS = (STATUS_SHAD)
FSR0H/L = (FSR0H/L_SHAD)
FSR1H/L = (FSR1H/L_SHAD)
FSR2H/L = (FSR2H/L_SHAD)
PRODH/L = (PRODH/L_SHAD)
PCLATH/U = (PCLATH/U_SHAD)

| **RETLW** | **Return Literal to W** | | | |
|---|---|---|---|---|
| **Syntax** | `RETLW k` | | | |
| Operands | 0 ≤ k ≤ 255 | | | |
| Operation | k → W<br>(TOS) → PC<br>PCLATU, PCLATH are unchanged | | | |
| Status Affected | None | | | |
| Encoding | `0000` | `1100` | `kkkk` | `kkkk` |
| Description | W is loaded with the 8-bit literal 'k'. The Program Counter is loaded from the top of the stack (the return address). The upper and high address latches (PCLATU/H) remain unchanged. | | | |
| Words | 1 | | | |

| .....continued | |
|---|---|
| **RETLW** | **Return Literal to W** |
| **Syntax** | **RETLW k** |
| Cycles | 2 |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'k' | Process Data | POP PC from stack<br>Write to W |
| No operation | No operation | No operation | No operation |

Example:

```
    CALL   TABLE  ; W contains table offset value
BACK              ; W now has table value (after RETLW)
    :
    :
TABLE
    ADDWF  PCL    ; W = offset
    RETLW  k0     ; Begin table
    RETLW  k1     ;
    :
    :
    RETLW  kn     ; End of table
```

Before Instruction
W = 07h

After Instruction
W = value of kn

| **RETURN** | **Return from Subroutine** | | | |
|---|---|---|---|---|
| **Syntax** | **RETURN {s}** | | | |
| Operands | s ∈ [0, 1] | | | |
| Operation | (TOS) → PC<br><br>If s = 1<br>(WREG_CSHAD) → WREG<br>(STATUS_CSHAD) → STATUS<br>(BSR_CSHAD) → BSR<br>PCLATU, PCLATH are unchanged | | | |
| Status Affected | None | | | |
| Encoding | 0000 | 0000 | 0001 | 001s |
| Description | Return from subroutine. The stack is popped and the top of the stack (TOS) is loaded into the Program Counter. If 's' = 1, the contents of the shadow registers WREG_CSHAD, STATUS_CSHAD and BSR_CSHAD, are loaded into their corresponding registers. If 's' = 0, no update of these registers occurs (default). The upper and high address latches (PCLATU/H) remain unchanged. | | | |
| Words | 1 | | | |
| Cycles | 2 | | | |

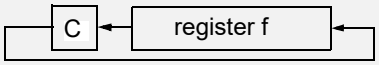
© 2019-2023 Microchip Technology Inc. and its subsidiaries

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | No operation | Process Data | POP PC from stack |
| No operation | No operation | No operation | No operation |

Example: `RETURN 1`

After Instruction
PC = (TOS)
WREG = (WREG_CSHAD)
BSR = (BSR_CSHAD)
STATUS = (STATUS_CSHAD)

| **RLCF** | **Rotate Left f through Carry** | | | |
|---|---|---|---|---|
| **Syntax** | `RLCF f {,d {,a}}` | | | |
| Operands | 0 ≤ f ≤ 255<br>d ∈ [0, 1]<br>a ∈ [0, 1] | | | |
| Operation | (f<n>) → dest<n+1><br>(f<7>) → C<br>(C) → dest<0> | | | |
| Status Affected | C, N, Z | | | |
| Encoding | 0011 | 01da | ffff | ffff |
| Description | The contents of register 'f' are rotated one bit to the left through the Carry flag. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in the register 'f' (default).<br>If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.<br>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode for details.<br>  | | | |
| Words | 1 | | | |
| Cycles | 1 | | | |

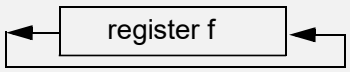
Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example: `RLCF REG, 0, 0`

Before Instruction
REG = 1110 0110 [E6h]
W = ?
C = 0

After Instruction
REG = 1110 0110 [E6h]
W = 1100 1100 [CCh]
C = 1

| RLNCF | Rotate Left f (No Carry) | | | |
|---|---|---|---|---|
| **Syntax** | **RLNCF f {,d {,a}}** | | | |
| Operands | 0 ≤ f ≤ 255<br>d ∈ [0, 1]<br>a ∈ [0, 1] | | | |
| Operation | (f<n>) → dest<n+1><br>(f<7>) → dest<0> | | | |
| Status Affected | N, Z | | | |
| Encoding | 0100 | 01da | ffff | ffff |
| Description | The contents of register 'f' are rotated one bit to the left. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in the register 'f' (default).<br>If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.<br>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode for details.<br>  | | | |
| Words | 1 | | | |
| Cycles | 1 | | | |

Q Cycle Activity:

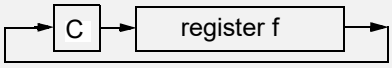
| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example: RLNCF REG, 1, 0

Before Instruction
REG = 1010 1011 [ABh]

After Instruction
REG = 0101 0111 [57h]

| RRCF | Rotate Right f through Carry |
|---|---|
| **Syntax** | **RRCF f {,d {,a}}** |
| Operands | 0 ≤ f ≤ 255<br>d ∈ [0, 1]<br>a ∈ [0, 1] |
| Operation | (f<n>) → dest<n-1><br>(f<0>) → C<br>(C) → dest<7> |

| .....continued | | | | |
|---|---|---|---|---|
| **RRCF** | **Rotate Right f through Carry** | | | |
| **Syntax** | **RRCF f {,d {,a}}** | | | |
| Status Affected | C, N, Z | | | |
| Encoding | 0011 | 00da | ffff | ffff |
| Description | The contents of register 'f' are rotated one bit to the right through the Carry flag. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in the register 'f' (default).<br>If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.<br>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode for details.<br>  | | | |
| Words | 1 | | | |
| Cycles | 1 | | | |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example: `RRCF REG, 0, 0`

Before Instruction
REG = 1110 0110 [E6h]
W = ?
C = 0

After Instruction
REG = 1110 0110 [E6h]
W = 0111 0011 [73h]
C = 0

| **RRNCF** | **Rotate Right f (No Carry)** | | | |
|---|---|---|---|---|
| **Syntax** | **RRNCF f {,d {,a}}** | | | |
| Operands | 0 ≤ f ≤ 255<br>d ∈ [0, 1]<br>a ∈ [0, 1] | | | |
| Operation | (f<n>) → dest<n-1><br>(f<0>) → dest<7> | | | |
| Status Affected | N, Z | | | |
| Encoding | 0100 | 00da | ffff | ffff |

| .....continued | |
|---|---|
| **RRNCF** | **Rotate Right f (No Carry)** |
| **Syntax** | `RRNCF f {,d {,a}}` |
| Description | The contents of register 'f' are rotated one bit to the right. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in the register 'f' (default).<br>If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.<br>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode for details.<br> |
| Words | 1 |
| Cycles | 1 |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example 1: `RRNCF REG, 1, 0`

Before Instruction
REG = 1101 0111 [D7h]

After Instruction
REG = 1110 1011 [EBh]

Example 2: `RRNCF REG, 0, 0`

Before Instruction
REG = 1101 0111 [D7h]
W = ?

After Instruction
REG = 1101 0111 [D7h]
W = 1110 1011 [EBh]

| **SETF** | **Set f** | | | |
|---|---|---|---|---|
| **Syntax** | `SETF f {,a}` | | | |
| Operands | 0 ≤ f ≤ 255<br>a ∈ [0, 1] | | | |
| Operation | FFh → f | | | |
| Status Affected | None | | | |
| Encoding | 0110 | 100a | ffff | ffff |

| .....continued | |
|---|---|
| **SETF** | **Set f** |
| **Syntax** | **SETF f {,a}** |
| Description | The contents of the specified register 'f' are set to FFh.<br>If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.<br>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode for details. |
| Words | 1 |
| Cycles | 1 |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write register 'f' |

Example: SETF REG, 1

Before Instruction
REG = 5Ah

After Instruction
REG = FFh

| **SLEEP** | **Enter Sleep Mode** | | | |
|---|---|---|---|---|
| **Syntax** | **SLEEP** | | | |
| Operands | None | | | |
| Operation | 00h → WDT<br>1 → $\overline{TO}$<br>0 → $\overline{PD}$ | | | |
| Status Affected | $\overline{TO}$, $\overline{PD}$ | | | |
| Encoding | 0000 | 0000 | 0000 | 0011 |
| Description | The Power-down Status ($\overline{PD}$) bit is cleared. The Time-Out Status $\overline{TO}$) bit is set. Watchdog Timer is cleared. The processor is put into Sleep mode with the oscillator stopped. | | | |
| Words | 1 | | | |
| Cycles | 1 | | | |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | No operation | Process Data | Go to Sleep |

Example: SLEEP

Before Instruction
$\overline{TO}$ = ?
$\overline{PD}$ = ?

After Instruction
$\overline{TO}$ = 1 †
$\overline{PD}$ = 0

† If WDT causes wake-up, this bit is cleared.

| SUBFSR | Subtract Literal from FSR | | | |
|---|---|---|---|---|
| **Syntax** | **SUBFSR $f_n$, k** | | | |
| Operands | 0 ≤ k ≤ 63<br>$f_n \in [0, 1, 2]$ | | | |
| Operation | $(FSRf_n) - k \rightarrow FSRf_n$ | | | |
| Status Affected | None | | | |
| Encoding | 1110 | 1001 | $f_nf_nkk$ | kkkk |
| Description | The 6-bit literal ‘k’ is subtracted from the contents of the FSR specified by ‘$f_n$’. | | | |
| Words | 1 | | | |
| Cycles | 1 | | | |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal ‘k’ | Process Data | Write to FSR |

Example: SUBFSR 2, 23h

Before Instruction
FSR2 = 03FFh

After Instruction
FSR2 = 03DCh

| SUBFWB | Subtract f from W with Borrow | | | |
|---|---|---|---|---|
| **Syntax** | **SUBFWB f {,d {,a}}** | | | |
| Operands | 0 ≤ f ≤ 255<br>d ∈ [0, 1]<br>a ∈ [0, 1] | | | |
| Operation | $(W) - (f) - (\overline{C}) \rightarrow dest$ | | | |
| Status Affected | N, OV, C, DC, Z | | | |
| Encoding | 0101 | 01da | ffff | ffff |

| .....continued | |
|---|---|
| **SUBFWB** | **Subtract f from W with Borrow** |
| **Syntax** | `SUBFWB f {,d {,a}}` |
| Description | Subtract register 'f' and Carry flag (Borrow) from W (two's complement method). If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in the register 'f' (default).<br>If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.<br>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode for details. |
| Words | 1 |
| Cycles | 1 |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example 1: `SUBFWB REG, 1, 0`

Before Instruction
REG = 03h
W = 02h
C = 1

After Instruction
REG = FFh (two's complement)
W = 02h
C = 0
Z = 0
N = 1 (result is negative)

Example 2: `SUBFWB REG, 0, 0`

Before Instruction
REG = 02h
W = 05h
C = 1

After Instruction
REG = 02h
W = 03h
C = 1
Z = 0
N = 0 (result is positive)

Example 3: `SUBFWB REG, 1, 0`

Before Instruction
REG = 01h
W = 02h
C = 0

After Instruction
REG = 00h
W = 02h
C = 1
Z = 1 (result is zero)
N = 0

| SUBLW | Subtract W from Literal | | | |
|---|---|---|---|---|
| **Syntax** | **SUBLW k** | | | |
| Operands | 0 ≤ k ≤ 255 | | | |
| Operation | k – (W) → W | | | |
| Status Affected | N, OV, C, DC, Z | | | |
| Encoding | 0000 | 1000 | kkkk | kkkk |
| Description | W is subtracted from the 8-bit literal ‘k’. The result is placed in W. | | | |
| Words | 1 | | | |
| Cycles | 1 | | | |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal ‘k’ | Process Data | Write to W |

Example 1: SUBLW 02h

Before Instruction
W = 01h
C = ?

After Instruction
W = 01h
C = 1 (result is positive)
Z = 0
N = 0

Example 2: SUBLW 02h

Before Instruction
W = 02h
C = ?

After Instruction
W = 00h
C = 1
Z = 1 (result is zero)
N = 0

Example 3: SUBLW 02h

Before Instruction
W = 03h
C = ?

After Instruction
W = FFh (two's complement)
C = 0
Z = 0
N = 1 (result is negative)

| SUBWF | Subtract W from f | | | |
|---|---|---|---|---|
| **Syntax** | **SUBWF f {,d {,a}}** | | | |
| Operands | 0 ≤ f ≤ 255<br>d ∈ [0, 1]<br>a ∈ [0, 1] | | | |
| Operation | (f) – (W) → dest | | | |
| Status Affected | N, OV, C, DC, Z | | | |
| Encoding | 0101 | 11da | ffff | ffff |
| Description | Subtract W from register 'f' (two's complement method). If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in the register 'f' (default).<br>If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.<br>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode for details. | | | |
| Words | 1 | | | |
| Cycles | 1 | | | |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example 1: SUBWF REG, 1, 0

Before Instruction
REG = 03h
W = 02h
C = ?

After Instruction
REG = 01h (two's complement)
W = 02h
C = 1 (result is positive)
Z = 0
N = 0

Example 2: SUBWF REG, 0, 0

Before Instruction
REG = 02h
W = 02h
C = ?

© 2019-2023 Microchip Technology Inc. and its subsidiaries

After Instruction
REG = 02h
W = 00h
C = 1
Z = 1 (result is zero)
N = 0

Example 3: SUBWF REG, 1, 0

Before Instruction
REG = 01h
W = 02h
C = ?

After Instruction
REG = FFh (two's complement)
W = 02h
C = 0
Z = 0
N = 1 (result is negative)

| SUBWFB | Subtract W from f with Borrow | | | |
|---|---|---|---|---|
| **Syntax** | **SUBWFB f {,d {,a}}** | | | |
| Operands | 0 ≤ f ≤ 255<br>d ∈ [0, 1]<br>a ∈ [0, 1] | | | |
| Operation | (f) – (W) – ($\overline{C}$) → dest | | | |
| Status Affected | N, OV, C, DC, Z | | | |
| Encoding | 0101 | 10da | ffff | ffff |
| Description | Subtract W and the Carry flag (Borrow) from register 'f' (two's complement method). If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in the register 'f' (default).<br>If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.<br>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode for details. | | | |
| Words | 1 | | | |
| Cycles | 1 | | | |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example 1: SUBWFB REG, 1, 0

Before Instruction
REG = 19h (0001 1001)
W = 0Dh (0000 1101)
C = 1

After Instruction
REG = 0Ch (0000 1100)
W = 0Dh (0000 1101)
C = 1 (result is positive)
Z = 0
N = 0

Example 2: SUBWFB REG, 0, 0

Before Instruction
REG = 1Bh (0001 1011)
W = 1Ah (0001 1010)
C = 0

After Instruction
REG = 1Bh (0001 1011)
W = 00h
C = 1
Z = 1 (result is zero)
N = 0

Example 3: SUBWFB REG, 1, 0

Before Instruction
REG = 03h (0000 0011)
W = 0Eh (0000 1110)
C = 1

After Instruction
REG = F5h (1111 0101) (two's complement)
W = 0Eh (0000 1110)
C = 0
Z = 0
N = 1 (result is negative)

| SWAPF | Swap f | | | |
|---|---|---|---|---|
| **Syntax** | **SWAPF f {,d {,a}}** | | | |
| Operands | 0 ≤ f ≤ 255<br>d ∈ [0, 1]<br>a ∈ [0, 1] | | | |
| Operation | (f<3:0>) → dest<7:4><br>(f<7:4>) → dest<3:0> | | | |
| Status Affected | None | | | |
| Encoding | 0011 | 10da | ffff | ffff |
| Description | The upper and lower nibbles of register 'f' are exchanged. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in the register 'f' (default).<br>If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.<br>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode for details. | | | |
| Words | 1 | | | |

| .....continued | |
|---|---|
| **SWAPF** | **Swap f** |
| **Syntax** | **SWAPF f {,d {,a}}** |
| Cycles | 1 |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example: `SWAPF REG, 1, 0`

Before Instruction
REG = `53h`

After Instruction
REG = `35h`

| **TBLRD** | **Table Read** | | | |
|---|---|---|---|---|
| **Syntax** | **TBLRD *<br>TBLRD *+<br>TBLRD *-<br>TBLRD +*** | | | |
| Operands | None | | | |
| Operation | If `TBLRD *`<br>(Prog Mem (TBLPTR)) → TABLAT<br>TBLPTR – No Change<br><br>If `TBLRD *+`<br>(Prog Mem (TBLPTR)) → TABLAT<br>(TBLPTR) + 1 → TBLPTR<br><br>If `TBLRD *-`<br>(Prog Mem (TBLPTR)) → TABLAT<br>(TBLPTR) – 1 → TBLPTR<br><br>If `TBLRD +*`<br>(TBLPTR) + 1 → TBLPTR<br>(Prog Mem (TBLPTR)) → TABLAT | | | |
| Status Affected | None | | | |
| Encoding | `0000` | `0000` | `0000` | `10mm`<br><br>`mm=0 *`<br>`mm=1 *+`<br>`mm=2 *-`<br>`mm=3 +*` |

| .....continued | |
|---|---|
| **TBLRD** | **Table Read** |
| **Syntax** | `TBLRD *`<br>`TBLRD *+`<br>`TBLRD *-`<br>`TBLRD +*` |
| Description | This instruction is used to read the contents of Program Memory. To address the program memory, a pointer called Table Pointer (TBLPTR) is used.<br>The TBLPTR (a 21-bit pointer) points to each byte in the program memory. TBLPTR has a 2-Mbyte address range.<br>TBLPTR[0] = `0`: Least Significant Byte of Program Memory Word<br>TBLPTR[0] = `1`: Most Significant Byte of Program Memory Word<br>The `TBLRD` instruction can modify the value of TBLPTR as follows:<br>• no change (`TBLRD *`)<br>• post-increment (`TBLRD *+`)<br>• post-decrement (`TBLRD *-`)<br>• pre-increment (`TBLRD +*`) |
| Words | 1 |
| Cycles | 2 |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | No operation | No operation | No operation |
| No operation | No operation<br>(Read Program Memory) | No operation | No operation<br>(Write TABLAT) |

Example 1: `TBLRD *+`

Before Instruction
TABLAT = `55h`
TBLPTR = `00A356h`
MEMORY (`00A356h`) = `34h`

After Instruction
TABLAT = `34h`
TBLPTR = `00A357h`

Example 2: `TBLRD +*`

Before Instruction
TABLAT = `AAh`
TBLPTR = `01A357h`
MEMORY (`01A357h`) = `12h`
MEMORY (`01A358h`) = `34h`

After Instruction
TABLAT = `34h`
TBLPTR = `01A358h`

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

| TBLWT | Table Write | | | |
|---|---|---|---|---|
| Syntax | `TBLWT *`<br>`TBLWT *+`<br>`TBLWT *-`<br>`TBLWT +*` | | | |
| Operands | None | | | |
| Operation | If `TBLWT *`<br>(TABLAT) → Holding Register<br>TBLPTR – No Change<br><br>If `TBLWT *+`<br>(TABLAT) → Holding Register<br>(TBLPTR) + 1 → TBLPTR<br><br>If `TBLWT *-`<br>(TABLAT) → Holding Register<br>(TBLPTR) – 1 → TBLPTR<br><br>If `TBLWT +*`<br>(TBLPTR) + 1 → TBLPTR<br>(TABLAT) → Holding Register | | | |
| Status Affected | None | | | |
| Encoding | `0000` | `0000` | `0000` | `11mm`<br><br>`mm=0 *`<br>`mm=1 *+`<br>`mm=2 *-`<br>`mm=3 +*` |
| Description | This instruction uses the three LSBs of TBLPTR to determine which of the eight holding registers the TABLAT is written to. The holding registers are used to program the contents of Program Memory. (Refer to the *"Program Flash Memory"* section for additional details on programming Flash memory.)<br><br>The TBLPTR (a 21-bit pointer) points to each byte in the program memory. TBLPTR has a 2-Mbyte address range. The LSb of the TBLPTR selects which byte of the program memory location to access.<br><br>TBLPTR[0] = `0`: Least Significant Byte of Program Memory Word<br><br>TBLPTR[0] = `1`: Most Significant Byte of Program Memory Word<br><br>The `TBLWT` instruction can modify the value of TBLPTR as follows:<br>• no change (`TBLWT *`)<br>• post-increment (`TBLWT *+`)<br>• post-decrement (`TBLWT *-`)<br>• pre-increment (`TBLWT +*`) | | | |
| Words | 1 | | | |
| Cycles | 2 | | | |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | No operation | No operation | No operation |

| | | | |
|---|---|---|---|
| No operation | No operation (Read TABLAT) | No operation | No operation (Write to Holding Register) |

Example 1: `TBLWT *+`

Before Instruction
TABLAT = `55h`
TBLPTR = `00A356h`
HOLDING REGISTER (`00A356h`) = `FFh`

After Instruction (table write completion)
TABLAT = `55h`
TBLPTR = `00A357h`
HOLDING REGISTER (`00A356h`) = `55h`

Example 2: `TBLWT +*`

Before Instruction
TABLAT = `34h`
TBLPTR = `01389Ah`
HOLDING REGISTER (`01389Ah`) = `FFh`
HOLDING REGISTER (`01389Bh`) = `FFh`

After Instruction (table write completion)
TABLAT = `34h`
TBLPTR = `01389Bh`
HOLDING REGISTER (`01389Ah`) = `FFh`
HOLDING REGISTER (`01389Bh`) = `34h`

| **TSTFSZ** | **Test f, Skip if 0** | | | |
|---|---|---|---|---|
| **Syntax** | **`TSTFSZ f {,a}`** | | | |
| Operands | $0 \le f \le 255$<br>$a \in [0, 1]$ | | | |
| Operation | Skip if f = `0` | | | |
| Status Affected | None | | | |
| Encoding | `0110` | `011a` | `ffff` | `ffff` |
| Description | If ‘f’ = `0`, the next instruction fetched during the current instruction execution is discarded and a `NOP` is executed, making this a two-cycle instruction.<br>If ‘a’ is ‘`0`’, the Access Bank is selected. If ‘a’ is ‘`1`’, the BSR is used to select the GPR bank.<br>If ‘a’ is ‘`0`’ and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ `95` (`5Fh`). See Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode for details. | | | |
| Words | 1 | | | |
| Cycles | 1 (2)<br>**Note:** Three cycles if skip and followed by a two-word instruction. Four cycles if skip and followed by a three-word instruction. | | | |

Q Cycle Activity:

If no skip:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register ‘f’ | Process Data | No operation |

If skip:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register ‘f’ | Process Data | No operation |
| No operation | No operation | No operation | No operation |

If skip and followed by two-word instruction:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register ‘f’ | Process Data | No operation |
| No operation | No operation | No operation | No operation |
| No operation | No operation | No operation | No operation |

If skip and followed by three-word instruction:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register ‘f’ | Process Data | No operation |
| No operation | No operation | No operation | No operation |
| No operation | No operation | No operation | No operation |
| No operation | No operation | No operation | No operation |

Example:

```
HERE   TSTFSZ   CNT, 1
NZERO:
ZERO:
```

Before Instruction
PC = address `(HERE)`

After Instruction
If CNT = `0`; PC = address `(ZERO)`
If CNT ≠ `0`; PC = address `(NZERO)`

| **XORLW** | **Exclusive OR Literal with W** | | | |
|---|---|---|---|---|
| **Syntax** | **`XORLW k`** | | | |
| Operands | 0 ≤ k ≤ 255 | | | |
| Operation | (W) .XOR. k → W | | | |
| Status Affected | N, Z | | | |
| Encoding | `0000` | `1010` | `kkkk` | `kkkk` |
| Description | The contents of W are XORed with the 8-bit literal ‘k’. The result is placed in W. | | | |
| Words | 1 | | | |
| Cycles | 1 | | | |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'k' | Process Data | Write to W |

Example: `XORLW 0AFh`

Before Instruction
W = `B5h`

After Instruction
W = `1Ah`

| XORWF | Exclusive OR W with f | | | |
|---|---|---|---|---|
| **Syntax** | **`XORWF f {,d {,a}}`** | | | |
| Operands | 0 ≤ f ≤ 255<br>d ∈ [0, 1]<br>a ∈ [0, 1] | | | |
| Operation | (W) .XOR. (f) → dest | | | |
| Status Affected | N, Z | | | |
| Encoding | `0001` | `10da` | `ffff` | `ffff` |
| Description | Exclusive OR the contents of W with register 'f'. If 'd' is '`0`', the result is stored in W. If 'd' is '`1`', the result is stored back in the register 'f' (default).<br>If 'a' is '`0`', the Access Bank is selected. If 'a' is '`1`', the BSR is used to select the GPR bank.<br>If 'a' is '`0`' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ `95` (`5Fh`). See Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode for details. | | | |
| Words | 1 | | | |
| Cycles | 1 | | | |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read register 'f' | Process Data | Write to destination |

Example: `XORWF REG, 1, 0`

Before Instruction
REG = `AFh`
W = `B5h`

After Instruction
REG = `1Ah`
W = `B5h`

© 2019-2023 Microchip Technology Inc. and its subsidiaries

## 44.2 Extended Instruction Set

In addition to the standard instruction set, PIC18 devices also provide an optional extension to the core CPU functionality. The added features include additional instructions that augment Indirect and Indexed Addressing operations and the implementation of Indexed Literal Offset Addressing mode for many of the standard PIC18 instructions.

The additional features of the extended instruction set are disabled by default. To enable them, users must set the XINST Configuration bit.

The instructions in the extended set can all be classified as literal operations, which either manipulate the File Select registers, or use them for Indexed Addressing. Two of the standard instructions, `ADDFSR` and `SUBFSR`, each have an additional special instantiation for using FSR2 as extended instructions. These versions (`ADDULNK` and `SUBULNK`) allow for automatic return after execution.

The extended instructions are specifically implemented to optimize re-entrant program code (that is, code that is recursive or that uses a software stack) written in high-level languages, particularly C. Among other things, they allow users working in high-level languages to perform certain operations on data structures more efficiently. These include:

- Dynamic allocation and deallocation of software stack space when entering and leaving subroutines
- Function pointer invocation
- Software Stack Pointer manipulation
- Manipulation of variables located in a software stack

A summary of the instructions in the extended instruction set is provided in 44.2.1. Extended Instruction Syntax. Detailed descriptions are provided in 44.2.2. Extended Instruction Set. The opcode field descriptions in Table 44-1 apply to both the standard and extended PIC18 instruction sets.

**Important:** The instruction set extension and the Indexed Literal Offset Addressing mode were designed for optimizing applications written in C; the user may likely never use these instructions directly in assembler. The syntax for these commands is provided as a reference for users who may be reviewing code that has been generated by a compiler.

**Important:** Enabling the PIC18 instruction set extension may cause legacy applications to behave erratically or fail entirely. Refer to 44.2.3. Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode for details.

### 44.2.1 Extended Instruction Syntax

Most of the extended instructions use indexed arguments, using one of the File Select registers and some offset to specify a source or destination register. When an argument for an instruction serves as part of Indexed Addressing, it is enclosed in square brackets (“`[ ]`”). This is done to indicate that the argument is used as an index or offset. MPASM™ Assembler will flag an error if it determines that an index or offset value is not bracketed.

When the extended instruction set is enabled, brackets are also used to indicate index arguments in byte-oriented and bit-oriented instructions. This is in addition to other changes in their syntax. For more details, see 44.2.3.1. Extended Instruction Syntax with Standard PIC18 Commands.

**Table 44-3. Extensions to the PIC18 Instruction Set**

| Mnemonic, Operands | | Description | Cycles | 16-Bit Instruction Word | | | | Status Affected | Notes |
|---|---|---|---|---|---|---|---|---|---|
| | | | | MSb | | | LSb | | |
| ADDULNK | k | Add literal to FSR2 and return | 2 | 1110 | 1000 | 11kk | kkkk | None | 1, 3 |
| MOVSF | $z_s$, $f_d$ | Move $z_s$ (12-bit source) to $f_d$ (12-bit destination) | 2 | 1110 | 1011 | $0z_sz_sz_s$ | $z_sz_sz_sz_s$ | None | 2, 3, 4 |
| | | | | 1111 | $f_df_df_df_d$ | $f_df_df_df_d$ | $f_df_df_df_d$ | | |
| MOVSFL | $z_s$, $f_d$ | Move $z_s$ (14-bit source) to $f_d$ (14-bit destination) | 3 | 0000 | 0000 | 0000 | 0010 | None | 2, 3 |
| | | | | 1111 | $xxxz_s$ | $z_sz_sz_sz_s$ | $z_sz_sf_df_d$ | | |
| | | | | 1111 | $f_df_df_df_d$ | $f_df_df_df_d$ | $f_df_df_df_d$ | | |
| MOVSS | $z_s$, $z_d$ | Move $z_s$ (source) to $z_d$ (destination) | 2 | 1110 | 1011 | $1z_sz_sz_s$ | $z_sz_sz_sz_s$ | None | 2, 3 |
| | | | | 1111 | xxxx | $xz_dz_dz_d$ | $z_dz_dz_dz_d$ | | |
| PUSHL | k | Store literal at FSR2, decrement FSR2 | 1 | 1110 | 1010 | kkkk | kkkk | None | 3 |
| SUBULNK | k | Subtract literal from FSR2 and return | 2 | 1110 | 1001 | 11kk | kkkk | None | 1, 3 |

**Notes:**

1. If Program Counter (PC) is modified or a conditional test is true, the instruction requires an additional cycle. The extra cycle is executed as a NOP.
2. Some instructions are multi-word instructions. The extra words of these instructions will be decoded as a NOP, unless the first word of the instruction retrieves the information embedded in these 16 bits. This ensures that all program memory locations have a valid instruction.
3. Only available when extended instruction set is enabled.
4. $f_s$ and $f_d$ do not cover the full memory range. 2 MSbs of bank selection are forced to 0b00 to limit the range of these instructions to lower 4k addressing space.

© 2019-2023 Microchip Technology Inc. and its subsidiaries

### 44.2.2 Extended Instruction Set

**Important:** All PIC18 instructions may take an optional label argument preceding the instruction mnemonic for use in symbolic addressing. If a label is used, the instruction format then becomes:
`{label} instruction argument(s)`

| ADDULNK | Add Literal to FSR2 and Return | | | |
|---|---|---|---|---|
| **Syntax** | **ADDULNK k** | | | |
| Operands | 0 ≤ k ≤ 63 | | | |
| Operation | (FSR2) + k → FSR2<br>(TOS) → PC | | | |
| Status Affected | None | | | |
| Encoding | 1110 | 1000 | 11kk | kkkk |
| Description | The 6-bit literal 'k' is added to the contents of FSR2. A RETURN is then executed by loading the PC with the TOS. The instruction takes two cycles to execute; a NOP is performed during the second cycle. This may be thought of as a special case of the ADDFSR instruction, where $f_n$ = 3 (binary '11'); it operates only on FSR2. | | | |
| Words | 1 | | | |
| Cycles | 2 | | | |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'k' | Process Data | Write to destination |
| No operation | No operation | No operation | No operation |

Example: `ADDULNK 23h`

Before Instruction
FSR2 = 03FFh
PC = 0100h

After Instruction
FSR2 = 0422h
PC = (TOS)

| MOVSF | Move Indexed to f | | | |
|---|---|---|---|---|
| **Syntax** | **MOVSF [$z_s$], $f_d$** | | | |
| Operands | 0 ≤ $z_s$ ≤ 127<br>0 ≤ $f_d$ ≤ 4095 | | | |
| Operation | ((FSR2) + $z_s$) → $f_d$ | | | |
| Status Affected | None | | | |
| Encoding | 1110 | 1011 | $0z_sz_sz_s$ | $z_sz_sz_sz_s$ |
| | 1111 | $f_df_df_df_d$ | $f_df_df_df_d$ | $f_df_df_df_d$ |

| .....continued | |
|---|---|
| **MOVSF** | **Move Indexed to f** |
| **Syntax** | **MOVSF [$z_s$], $f_d$** |
| Description | The contents of the source register are moved to destination register '$f_d$'. The actual address of the source register is determined by adding the 7-bit literal offset '$z_s$' in the first word to the value of FSR2. The address of the destination register is specified by the 12-bit literal '$f_d$' in the second word. Both addresses can be anywhere in the 4096-byte data space (000h to FFFh).<br>**Note:**<br>MOVSF has curtailed the destination range to the lower 4 Kbyte space in memory (Banks 1 through 15). For everything else, use MOVSFL. |
| Words | 2 |
| Cycles | 2 |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Determine source address | Determine source address | Read source register |
| Decode | No operation<br>No dummy read | No operation | Write register '$f_d$' |

Example: MOVSF [05h], REG2

Before Instruction
FSR2 = 80h
Contents of 85h = 33h
REG2 = 11h
Address of REG2 = 100h

After Instruction
FSR2 = 80h
Contents of 85h = 33h
REG2 = 33h
Address of REG2 = 100h

| **MOVSFL** | **Move Indexed to f (Long Range)** | | | |
|---|---|---|---|---|
| **Syntax** | **MOVSFL [$z_s$], $f_d$** | | | |
| Operands | $0 \le z_s \le 127$<br>$0 \le f_d \le 16383$ | | | |
| Operation | $((FSR2) + z_s) \rightarrow f_d$ | | | |
| Status Affected | None | | | |
| Encoding | 0000 | 0000 | 0110 | 0010 |
| | 1111 | $xxxz_s$ | $z_sz_sz_sz_s$ | $z_sz_sf_df_d$ |
| | 1111 | $f_df_df_df_d$ | $f_df_df_df_d$ | $f_df_df_df_d$ |

| .....continued | |
|---|---|
| **MOVSFL** | **Move Indexed to f (Long Range)** |
| **Syntax** | **MOVSFL [$z_s$], $f_d$** |
| Description | The contents of the source register are moved to destination register '$f_d$'. The actual address of the source register is determined by adding the 7-bit literal offset '$z_s$' in the first word to the value of FSR2 (14 bits). The address of the destination register is specified by the 14-bit literal '$f_d$' in the second word. Both addresses can be anywhere in the 16 Kbyte data space (0000h to 3FFFh). The MOVSFL instruction cannot use the PCL, TOSU, TOSH or TOSL as the destination register. If the resultant source address points to an indirect addressing register, the value returned will be 00h. |
| Words | 3 |
| Cycles | 3 |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | No operation | No operation | No operation |
| Decode | Read source register | Process Data | No operation |
| Decode | No operation<br>No dummy read | No operation | Write register '$f_d$' |

Example: MOVSFL [05h], REG2

Before Instruction
FSR2 = 2080h
Contents of 2085h = 33h
REG2 = 11h
Address of REG2 = 2000h

After Instruction
FSR2 = 2080h
Contents of 2085h = 33h
REG2 = 33h
Address of REG2 = 2000h

| **MOVSS** | **Move Indexed to Indexed** | | | |
|---|---|---|---|---|
| **Syntax** | **MOVSS [$z_s$], [$z_d$]** | | | |
| Operands | $0 \le z_s \le 127$<br>$0 \le z_d \le 127$ | | | |
| Operation | ((FSR2) + $z_s$) → ((FSR2) + $z_d$) | | | |
| Status Affected | None | | | |
| Encoding | 1110 | 1011 | $1z_sz_sz_s$ | $z_sz_sz_sz_s$ |
| | 1111 | xxxx | $xz_dz_dz_d$ | $z_dz_dz_dz_d$ |

| .....continued | |
|---|---|
| **MOVSS** | **Move Indexed to Indexed** |
| **Syntax** | **MOVSS [$z_s$], [$z_d$]** |
| Description | The contents of the source register are moved to the destination register. The addresses of the source and destination registers are determined by adding the 7-bit literal offsets '$z_s$' or '$z_d$' respectively to the value of FSR2. Both registers can be located anywhere in the 16 Kbyte data memory space (0000h to 3FFFh).<br>The MOVSS instruction cannot use the PCL, TOSU, TOSH or TOSL as the destination register.<br>If the resultant source address points to an indirect addressing register, the value returned will be 00h. If the resultant destination address points to an indirect addressing register, the instruction will execute as a NOP. |
| Words | 2 |
| Cycles | 2 |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Determine source address | Determine source address | Read source register |
| Decode | Determine destination address | Determine destination address | Write to destination register |

Example: MOVSS [05h], [06h]

Before Instruction
FSR2 = 80h
Contents of 85h = 33h
Contents of 86h = 11h

After Instruction
FSR2 = 80h
Contents of 85h = 33h
Contents of 86h = 33h

| **PUSHL** | **Store Literal at FSR2, Decrement FSR2** | | | |
|---|---|---|---|---|
| **Syntax** | **PUSHL k** | | | |
| Operands | 0 ≤ k ≤ 255 | | | |
| Operation | k → FSR2<br>(FSR2) – 1 → FSR2 | | | |
| Status Affected | None | | | |
| Encoding | 1111 | 1010 | kkkk | kkkk |
| Description | The 8-bit literal 'k' is written to the data memory address specified by FSR2. FSR2 is decremented by 1 after the operation. This instruction allows users to push values onto a software stack. | | | |
| Words | 1 | | | |
| Cycles | 1 | | | |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'k' | Process Data | Write to destination |

Example: `PUSHL 08h`

Before Instruction
FSR2 = `01ECh`
Contents of `01ECh` = `00h`

After Instruction
FSR2 = `01EBh`
Contents of `01ECh` = `08h`

| **SUBULNK** | **Subtract Literal from FSR2 and Return** | | | |
|---|---|---|---|---|
| **Syntax** | **`SUBULNK k`** | | | |
| Operands | 0 ≤ k ≤ 63 | | | |
| Operation | (FSR2) – k → FSR2<br>(TOS) → PC | | | |
| Status Affected | None | | | |
| Encoding | `1110` | `1001` | `11kk` | `kkkk` |
| Description | The 6-bit literal 'k' is subtracted from the contents of FSR2. A `RETURN` is then executed by loading the PC with the TOS. The instruction takes two cycles to execute; a `NOP` is performed during the second cycle. This may be thought of as a special case of the `SUBFSR` instruction, where $f_n$ = 3 (binary '`11`'); it operates only on FSR2. | | | |
| Words | 1 | | | |
| Cycles | 2 | | | |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'k' | Process Data | Write to destination |
| No operation | No operation | No operation | No operation |

Example: `SUBULNK 23h`

Before Instruction
FSR2 = `03FFh`
PC = `0100h`

After Instruction
FSR2 = `03DCh`
PC = (TOS)

© 2019-2023 Microchip Technology Inc. and its subsidiaries

### 44.2.3 Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode

**Important:** Enabling the PIC18 instruction set extension may cause legacy applications to behave erratically or fail entirely.

In addition to the new commands in the extended set, enabling the extended instruction set also enables Indexed Literal Offset Addressing mode (the **"Indexed Addressing with Literal Offset"** section in the **"Memory Organization"** chapter). This has a significant impact on the way many commands of the standard PIC18 instruction set are interpreted.

When the extended set is disabled, addresses embedded in opcodes are treated as literal memory locations: Either as a location in the Access Bank ('a' = 0), or in a GPR bank designated by the BSR ('a' = 1). When the extended instruction set is enabled and 'a' = 0, however, a file register argument of 5Fh or less is interpreted as an offset from the pointer value in FSR2 and not as a literal address. For practical purposes, this means that all instructions using the Access RAM bit as an argument – that is, all byte-oriented and bit-oriented instructions, or almost half of the core PIC18 instructions – may behave differently when the extended instruction set is enabled.

When the content of FSR2 is 00h, the boundaries of the Access RAM are essentially remapped to their original values. This may be useful in creating backward compatible code. If this technique is used, it may be necessary to save the value of FSR2 and restore it when moving back and forth between C and assembly routines to preserve the Stack Pointer. Users must also keep in mind the syntax requirements of the extended instruction set (see 44.2.3.1. Extended Instruction Syntax with Standard PIC18 Commands).

Although the Indexed Literal Offset Addressing mode can be very useful for dynamic stack and pointer manipulation, it can also be very annoying if a simple arithmetic operation is carried out on the wrong register. Users who are accustomed to the PIC18 programming must keep in mind that, when the extended instruction set is enabled, register addresses of 5Fh or less are used for Indexed Literal Offset Addressing.

Representative examples of typical byte-oriented and bit-oriented instructions in the Indexed Literal Offset Addressing mode are provided in 44.2.4. Considerations when Enabling the Extended Instruction Set to show how execution is affected. The operand conditions shown in the examples are applicable to all instructions of these types.

**Related Links**

9.6. Data Memory and the Extended Instruction Set

#### 44.2.3.1 Extended Instruction Syntax with Standard PIC18 Commands

When the extended instruction set is enabled, the file register argument, 'f', in the standard byte-oriented and bit-oriented commands is replaced with the literal offset value, 'k'. As already noted, this occurs only when 'f' is less than or equal to 5Fh. When an offset value is used, it must be indicated by square brackets ("[ ]"). As with the extended instructions, the use of brackets indicates to the compiler that the value is to be interpreted as an index or an offset. Omitting the brackets, or using a value greater than 5Fh within brackets, will generate an error in the MPASM Assembler.

If the index argument is properly bracketed for Indexed Literal Offset Addressing, the Access RAM argument is never specified; it will automatically be assumed to be '0'. This is in contrast to standard operation (extended instruction set disabled) when 'a' is set on the basis of the target address. Declaring the Access RAM bit in this mode will also generate an error in the MPASM Assembler.

The destination argument, 'd', functions as before.

In the latest versions of the MPASM Assembler, language support for the extended instruction set must be explicitly invoked. This is done with either the command-line option, /y, or the PE directive in the source listing.

**Related Links**

9.6. Data Memory and the Extended Instruction Set

### 44.2.4 Considerations when Enabling the Extended Instruction Set

It is important to note that the extensions to the instruction set may not be beneficial to all users. In particular, users who are not writing code that uses a software stack may not benefit from using the extensions to the instruction set.

© 2019-2023 Microchip Technology Inc. and its subsidiaries

Additionally, the Indexed Literal Offset Addressing mode may create issues with legacy applications written to the PIC18 assembler. This is because instructions in the legacy code may attempt to address registers in the Access Bank below 5Fh. Since these addresses are interpreted as literal offsets to FSR2 when the instruction set extension is enabled, the application may read or write to the wrong data addresses.

When porting an application to a PIC18 device supporting extensions to the instruction set, it is very important to consider the type of code. A large, re-entrant application that is written in 'C' and benefits from efficient compilation will do well when using the instruction set extensions. Legacy applications that heavily use the Access Bank will most likely not benefit from using the extended instruction set.

| ADDWF | Add W to Indexed (Indexed Literal Offset Mode) | | | |
|---|---|---|---|---|
| **Syntax** | **ADDWF [k] {,d}** | | | |
| Operands | 0 ≤ k ≤ 95<br>d ∈ [0, 1] | | | |
| Operation | (W) + ((FSR2) + k) → dest | | | |
| Status Affected | N, OV, C, DC, Z | | | |
| Encoding | 0010 | 01d0 | kkkk | kkkk |
| Description | The contents of W are added to the contents of the register indicated by FSR2, offset by the value 'k'. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in the register 'f' (default). | | | |
| Words | 1 | | | |
| Cycles | 1 | | | |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'k' | Process Data | Write to destination |

Example: ADDWF [OFST] , 0

Before Instruction
W = 17h
OFST = 2Ch
FSR2 = 0A00h
Contents of 0A2Ch = 20h

After Instruction
W = 37h
Contents of 0A2Ch = 20h

| BSF | Bit Set Indexed (Indexed Literal Offset Mode) | | | |
|---|---|---|---|---|
| **Syntax** | **BSF [k], b** | | | |
| Operands | 0 ≤ k ≤ 95<br>0 ≤ b ≤ 7 | | | |
| Operation | 1 → ((FSR2) + k)<b> | | | |
| Status Affected | None | | | |
| Encoding | 1000 | bbb0 | kkkk | kkkk |
| Description | Bit 'b' of the register indicated by FSR2, offset by the value 'k', is set. | | | |

| .....continued | |
|---|---|
| **BSF** | **Bit Set Indexed (Indexed Literal Offset Mode)** |
| **Syntax** | **BSF [k], b** |
| Words | 1 |
| Cycles | 1 |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'k' | Process Data | Write to destination |

Example: BSF [FLAG_OFST], 7

Before Instruction
FLAG_OFST = 0Ah
FSR2 = 0A00h
Contents of 0A0Ah = 55h

After Instruction
Contents of 0A0Ah = D5h

| **SETF** | **Set Indexed (Indexed Literal Offset Mode)** | | | |
|---|---|---|---|---|
| **Syntax** | **SETF [k]** | | | |
| Operands | 0 ≤ k ≤ 95 | | | |
| Operation | FFh → ((FSR2) + k) | | | |
| Status Affected | None | | | |
| Encoding | 0110 | 1000 | kkkk | kkkk |
| Description | The contents of the register indicated by FSR2, offset by the value 'k', are set to FFh. | | | |
| Words | 1 | | | |
| Cycles | 1 | | | |

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|---|---|---|---|
| Decode | Read literal 'k' | Process Data | Write to destination |

Example: SETF [OFST]

Before Instruction
OFST = 2Ch
FSR2 = 0A00h
Contents of 0A2Ch = 00h

After Instruction
Contents of 0A2Ch = FFh

© 2019-2023 Microchip Technology Inc. and its subsidiaries

### 44.2.5 Special Considerations with Microchip MPLAB® IDE Tools

The latest versions of Microchip's software tools have been designed to fully support the extended instruction set on the PIC18 devices. This includes the MPLAB XC8 C compiler, MPASM Assembler and MPLAB X Integrated Development Environment (IDE).

When selecting a target device for software development, MPLAB X IDE will automatically set default Configuration bits for that device. The default setting for the XINST Configuration bit is '0', disabling the extended instruction set and Indexed Literal Offset Addressing mode. For proper execution of applications developed to take advantage of the extended instruction set, XINST must be set during programming.

To develop software for the extended instruction set, the user must enable support for the instructions and the Indexed Addressing mode in their language tool(s). Depending on the environment being used, this may be done in several ways:

- A menu option, or dialog box within the environment, that allows the user to configure the language tool and its settings for the project
- A command-line option
- A directive in the source code

These options vary between different compilers, assemblers and development environments. Users are encouraged to review the documentation accompanying their development systems for the appropriate information.

© 2019-2023 Microchip Technology Inc. and its subsidiaries

# 45. ICSP™ - In-Circuit Serial Programming™

ICSP programming allows customers to manufacture circuit boards with unprogrammed devices. Programming can be done after the assembly process, allowing the device to be programmed with the most recent firmware or a custom firmware. Five pins are needed for ICSP programming:

- ICSPCLK
- ICSPDAT
- $\overline{MCLR}/V_{PP}$
- $V_{DD}$
- $V_{SS}$

In Program/Verify mode, the program memory, User IDs and the Configuration bits are programmed through serial communications. The ICSPDAT pin is a bidirectional I/O used for transferring the serial data and the ICSPCLK pin is the clock input. For more information on ICSP, refer to the **"Family Programming Specification"**.

## 45.1 High-Voltage Programming Entry Mode

The device is placed into High-Voltage Programming Entry mode by holding the ICSPCLK and ICSPDAT pins low, then raising the voltage on $\overline{MCLR}/V_{PP}$ to $V_{IH}$.

## 45.2 Low-Voltage Programming Entry Mode

The Low-Voltage Programming Entry mode allows the PIC® Flash MCUs to be programmed using $V_{DD}$ only, without high voltage. When the LVP Configuration bit is set to '1', the low-voltage ICSP programming entry is enabled. To disable the Low-Voltage ICSP mode, the LVP bit must be programmed to '0'.

Entry into the Low-Voltage Programming Entry mode requires the following steps:

1. $\overline{MCLR}$ is brought to $V_{IL}$.
2. A 32-bit key sequence is presented on ICSPDAT, while clocking ICSPCLK.

Once the key sequence is complete, $\overline{MCLR}$ must be held at $V_{IL}$ for as long as Program/Verify mode is to be maintained.

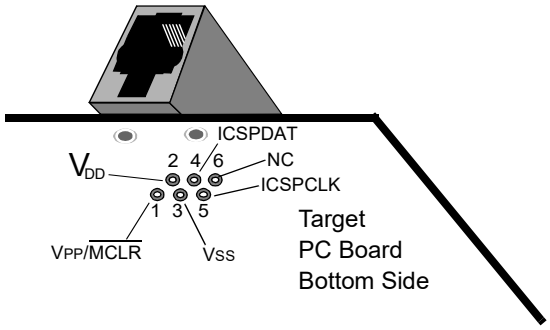
If low-voltage programming is enabled (LVP = 1), the $\overline{MCLR}$ Reset function is automatically enabled and cannot be disabled. See the *$\overline{MCLR}$* section for more information.

The LVP bit can only be reprogrammed to '0' by using the High-Voltage Programming mode.

## 45.3 Common Programming Interfaces

Connection to a target device is typically done through an ICSP header. A commonly found connector on development tools is the RJ-11 in the 6P6C (6-pin, 6-connector) configuration. See Figure 45-1.

**Figure 45-1. ICD RJ-11 Style Connector Interface**




Pin Description

1 = $V_{PP}/\overline{MCLR}$

2 = $V_{DD}$ Target

3 = $V_{SS}$ (ground)

4 = ICSPDAT

5 = ICSPCLK

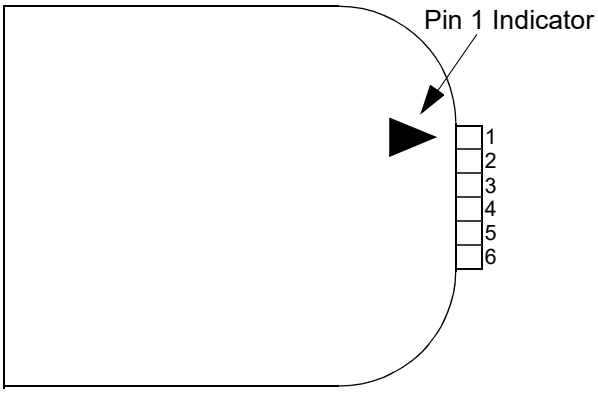
6 = No Connect

Another connector often found in use with the PICkit™ programmers is a standard 6-pin header with 0.1 inch spacing. Refer to Figure 45-2.

For additional interface recommendations, refer to the specific device programmer manual prior to PCB design.

It is recommended that isolation devices be used to separate the programming pins from other circuitry. The type of isolation is highly dependent on the specific application and may include devices such as resistors, diodes, or even jumpers. See Figure 45-3 for more information.

**Figure 45-2. PICkit™ Programmer Style Connector Interface**



Pin Description[1]:

1 = $V_{PP}$/$\overline{MCLR}$

2 = $V_{DD}$ Target

3 = $V_{SS}$ (ground)

4 = ICSPDAT

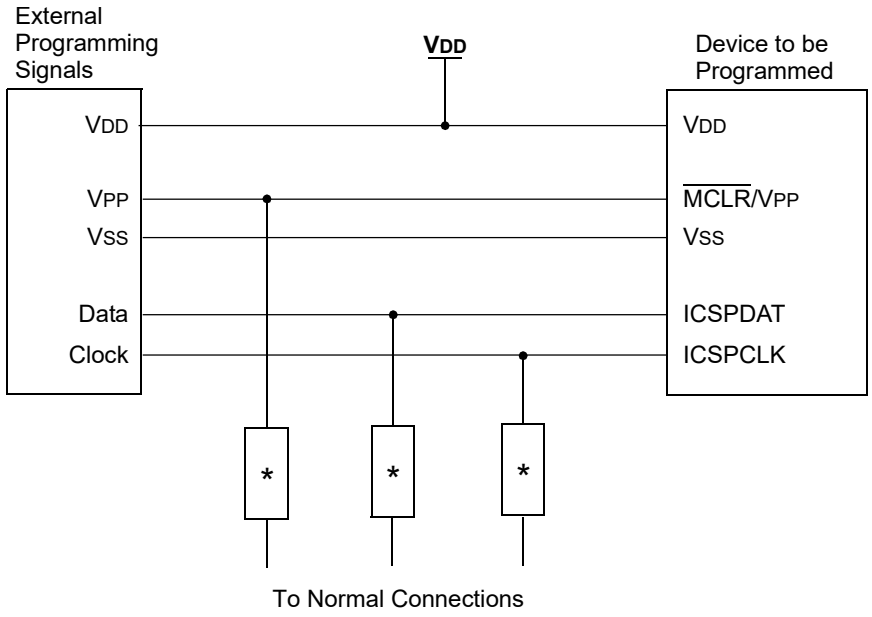
5 = ICSPCLK

6 = No Connect

**Note:**

1. The 6-pin header (0.100" spacing) accepts 0.025" square pins.

**Figure 45-3. Typical Connection for ICSP™ Programming**




* Isolation devices (as required).

# 46. Register Summary

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x39 | CLKRCON | 7:0 | EN | | | DC[1:0] | | DIV[2:0] | | |
| 0x3A | CLKRCLK | 7:0 | | | | CLK[4:0] | | | | |
| 0x3B ... 0x3F | Reserved | | | | | | | | | |
| 0x40 | NVMCON0 | 7:0 | | | | | | | | GO |
| 0x41 | NVMCON1 | 7:0 | WRERR | | | | | NVMCMD[2:0] | | |
| 0x42 | NVMLOCK | 7:0 | NVMLOCK[7:0] | | | | | | | |
| 0x43 | NVMADR | 7:0 | NVMADR[7:0] | | | | | | | |
| | | 15:8 | NVMADR[15:8] | | | | | | | |
| | | 23:16 | | | NVMADR[21:16] | | | | | |
| 0x46 | NVMDAT | 7:0 | NVMDAT[7:0] | | | | | | | |
| | | 15:8 | NVMDAT[15:8] | | | | | | | |
| 0x48 | VREGCON | 7:0 | | | PMSYS[1:0] | | | | VREGPM[1:0] | |
| 0x49 | BORCON | 7:0 | SBOREN | | | | | | | BORRDY |
| 0x4A | HLVDCON0 | 7:0 | EN | | OUT | RDY | | | INTH | INTL |
| 0x4B | HLVDCON1 | 7:0 | | | | | SEL[3:0] | | | |
| 0x4C | ZCDCON | 7:0 | SEN | | OUT | POL | | | INTP | INTN |
| 0x4D ... 0x5F | Reserved | | | | | | | | | |
| 0x60 | PMD0 | 7:0 | SYSCMD | FVRMD | HLVDMD | CRCMD | SCANMD | | CLKRMD | IOCMD |
| 0x61 | PMD1 | 7:0 | SMT1MD | TMR6MD | TMR5MD | TMR4MD | TMR3MD | TMR2MD | TMR1MD | TMR0MD |
| 0x62 | Reserved | | | | | | | | | |
| 0x63 | PMD3 | 7:0 | ACTMD | DAC1MD | ADCMD | | | C2MD | C1MD | ZCDMD |
| 0x64 | PMD4 | 7:0 | | CWG3MD | CWG2MD | CWG1MD | DSM1MD | NCO3MD | NCO2MD | NCO1MD |
| 0x65 | PMD5 | 7:0 | | PWM3MD | PWM2MD | PWM1MD | | CCP3MD | CCP2MD | CCP1MD |
| 0x66 | PMD6 | 7:0 | U5MD | U4MD | U3MD | U2MD | U1MD | SPI2MD | SPI1MD | I2C1MD |
| 0x67 | PMD7 | 7:0 | CLC8MD | CLC7MD | CLC6MD | CLC5MD | CLC4MD | CLC3MD | CLC2MD | CLC1MD |
| 0x68 | PMD8 | 7:0 | | | DMA6MD | DMA5MD | DMA4MD | DMA3MD | DMA2MD | DMA1MD |
| 0x69 | Reserved | | | | | | | | | |
| 0x6A | MD1CON0 | 7:0 | EN | | OUT | OPOL | | | | BIT |
| 0x6B | MD1CON1 | 7:0 | | | CHPOL | CHSYNC | | | CLPOL | CLSYNC |
| 0x6C | MD1SRC | 7:0 | | | MS[5:0] | | | | | |
| 0x6D | MD1CARL | 7:0 | | | | CL[4:0] | | | | |
| 0x6E | MD1CARH | 7:0 | | | | CH[4:0] | | | | |
| 0x6F | CMOUT | 7:0 | | | | | | | C2OUT | C1OUT |
| 0x70 | CM1CON0 | 7:0 | EN | OUT | | POL | | | HYS | SYNC |
| 0x71 | CM1CON1 | 7:0 | | | | | | | INTP | INTN |
| 0x72 | CM1NCH | 7:0 | | | | | | NCH[2:0] | | |
| 0x73 | CM1PCH | 7:0 | | | | | | PCH[2:0] | | |
| 0x74 | CM2CON0 | 7:0 | EN | OUT | | POL | | | HYS | SYNC |
| 0x75 | CM2CON1 | 7:0 | | | | | | | INTP | INTN |
| 0x76 | CM2NCH | 7:0 | | | | | | NCH[2:0] | | |
| 0x77 | CM2PCH | 7:0 | | | | | | PCH[2:0] | | |
| 0x78 | WDTCON0 | 7:0 | | | PS[4:0] | | | | | SEN |
| 0x79 | WDTCON1 | 7:0 | | CS[2:0] | | | | WINDOW[2:0] | | |
| 0x7A | WDTPSL | 7:0 | PSCNTL[7:0] | | | | | | | |
| 0x7B | WDTPSH | 7:0 | PSCNTH[7:0] | | | | | | | |
| 0x7C | WDTTMR | 7:0 | TMR[4:0] | | | | | STATE | PSCNT[17:16] | |
| 0x7D | DAC1DATL | 7:0 | DAC1R[7:0] | | | | | | | |
| 0x7E | Reserved | | | | | | | | | |
| 0x7F | DAC1CON | 7:0 | EN | | OE[1:0] | | PSS[1:0] | | | NSS |
| 0x80 | SPI1RXB | 7:0 | RXB[7:0] | | | | | | | |
| 0x81 | SPI1TXB | 7:0 | TXB[7:0] | | | | | | | |

.....continued

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x82 | SPI1TCNT | 7:0 | TCNTL[7:0] | | | | | | | |
| | | 15:8 | | | | | | TCNTH[2:0] | | |
| 0x84 | SPI1CON0 | 7:0 | EN | | | | | LSBF | MST | BMODE |
| 0x85 | SPI1CON1 | 7:0 | SMP | CKE | CKP | FST | | SSP | SDIP | SDOP |
| 0x86 | SPI1CON2 | 7:0 | BUSY | SSFLT | | | | SSET | TXR | RXR |
| 0x87 | SPI1STATUS | 7:0 | TXWE | | TXBE | | RXRE | CLB | | RXBF |
| 0x88 | SPI1TWIDTH | 7:0 | | | | | | TWIDTH[2:0] | | |
| 0x89 | SPI1BAUD | 7:0 | BAUD[7:0] | | | | | | | |
| 0x8A | SPI1INTF | 7:0 | SRMTIF | TCZIF | SOSIF | EOSIF | | RXOIF | TXUIF | |
| 0x8B | SPI1INTE | 7:0 | SRMTIE | TCZIE | SOSIE | EOSIE | | RXOIE | TXUIE | |
| 0x8C | SPI1CLK | 7:0 | | | | CLKSEL[4:0] | | | | |
| 0x8D | SPI2RXB | 7:0 | RXB[7:0] | | | | | | | |
| 0x8E | SPI2TXB | 7:0 | TXB[7:0] | | | | | | | |
| 0x8F | SPI2TCNT | 7:0 | TCNTL[7:0] | | | | | | | |
| | | 15:8 | | | | | | TCNTH[2:0] | | |
| 0x91 | SPI2CON0 | 7:0 | EN | | | | | LSBF | MST | BMODE |
| 0x92 | SPI2CON1 | 7:0 | SMP | CKE | CKP | FST | | SSP | SDIP | SDOP |
| 0x93 | SPI2CON2 | 7:0 | BUSY | SSFLT | | | | SSET | TXR | RXR |
| 0x94 | SPI2STATUS | 7:0 | TXWE | | TXBE | | RXRE | CLB | | RXBF |
| 0x95 | SPI2TWIDTH | 7:0 | | | | | | TWIDTH[2:0] | | |
| 0x96 | SPI2BAUD | 7:0 | BAUD[7:0] | | | | | | | |
| 0x97 | SPI2INTF | 7:0 | SRMTIF | TCZIF | SOSIF | EOSIF | | RXOIF | TXUIF | |
| 0x98 | SPI2INTE | 7:0 | SRMTIE | TCZIE | SOSIE | EOSIE | | RXOIE | TXUIE | |
| 0x99 | SPI2CLK | 7:0 | | | | CLKSEL[4:0] | | | | |
| 0x9A ... 0xAB | Reserved | | | | | | | | | |
| 0xAC | ACTCON | 7:0 | ACTEN | ACTUD | | | ACTLOCK | | ACTORS | |
| 0xAD | OSCCON1 | 7:0 | | NOSC[2:0] | | | NDIV[3:0] | | | |
| 0xAE | OSCCON2 | 7:0 | | COSC[2:0] | | | CDIV[3:0] | | | |
| 0xAF | OSCCON3 | 7:0 | CSWHOLD | SOSCPWR | | ORDY | NOSCR | | | |
| 0xB0 | OSCTUNE | 7:0 | | | TUN[5:0] | | | | | |
| 0xB1 | OSCFRQ | 7:0 | | | | | FRQ[3:0] | | | |
| 0xB2 | OSCSTAT | 7:0 | EXTOR | HFOR | MFOR | LFOR | SOR | ADOR | | PLLR |
| 0xB3 | OSCEN | 7:0 | EXTOEN | HFOEN | MFOEN | LFOEN | SOSCEN | ADOEN | | PLLEN |
| 0xB4 | PRLOCK | 7:0 | | | | | | | | PRLOCKED |
| 0xB5 | SCANPR | 7:0 | | | | | | PR[2:0] | | |
| 0xB6 | DMA1PR | 7:0 | | | | | | PR[2:0] | | |
| 0xB7 | DMA2PR | 7:0 | | | | | | PR[2:0] | | |
| 0xB8 | DMA3PR | 7:0 | | | | | | PR[2:0] | | |
| 0xB9 | DMA4PR | 7:0 | | | | | | PR[2:0] | | |
| 0xBA | DMA5PR | 7:0 | | | | | | PR[2:0] | | |
| 0xBB | DMA6PR | 7:0 | | | | | | PR[2:0] | | |
| 0xBC ... 0xBD | Reserved | | | | | | | | | |
| 0xBE | MAINPR | 7:0 | | | | | | PR[2:0] | | |
| 0xBF | ISRPR | 7:0 | | | | | | PR[2:0] | | |
| 0xC0 ... 0xD3 | Reserved | | | | | | | | | |
| 0xD4 | CLCDATA | 7:0 | CLC8OUT | CLC7OUT | CLC6OUT | CLC5OUT | CLC4OUT | CLC3OUT | CLC2OUT | CLC1OUT |
| 0xD5 | CLCSELECT | 7:0 | | | | | | SLCT[2:0] | | |
| 0xD6 | CLCnCON | 7:0 | EN | | OUT | INTP | INTN | MODE[2:0] | | |
| 0xD7 | CLCnPOL | 7:0 | POL | | | | G4POL | G3POL | G2POL | G1POL |
| 0xD8 | CLCnSEL0 | 7:0 | D1S[7:0] | | | | | | | |
| 0xD9 | CLCnSEL1 | 7:0 | D2S[7:0] | | | | | | | |
| 0xDA | CLCnSEL2 | 7:0 | D3S[7:0] | | | | | | | |
| 0xDB | CLCnSEL3 | 7:0 | D4S[7:0] | | | | | | | |

.....continued

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0xDC | CLCnGLS0 | 7:0 | G1D4T | G1D4N | G1D3T | G1D3N | G1D2T | G1D2N | G1D1T | G1D1N |
| 0xDD | CLCnGLS1 | 7:0 | G2D4T | G2D4N | G2D3T | G2D3N | G2D2T | G2D2N | G2D1T | G2D1N |
| 0xDE | CLCnGLS2 | 7:0 | G3D4T | G3D4N | G3D3T | G3D3N | G3D2T | G3D2N | G3D1T | G3D1N |
| 0xDF | CLCnGLS3 | 7:0 | G4D4T | G4D4N | G4D3T | G4D3N | G4D2T | G4D2N | G4D1T | G4D1N |
| 0xE0 ... 0xE7 | Reserved | | | | | | | | | |
| 0xE8 | DMASELECT | 7:0 | | | | | | SLCT[2:0] | | |
| 0xE9 | DMAnBUF | 7:0 | BUF[7:0] | | | | | | | |
| 0xEA | DMAnDCNT | 7:0 | DCNT[7:0] | | | | | | | |
| | | 15:8 | | | | | DCNT[11:8] | | | |
| 0xEC | DMAnDPTR | 7:0 | DPTR[7:0] | | | | | | | |
| | | 15:8 | DPTR[15:8] | | | | | | | |
| 0xEE | DMAnDSZ | 7:0 | DSZ[7:0] | | | | | | | |
| | | 15:8 | | | | | DSZ[11:8] | | | |
| 0xF0 | DMAnDSA | 7:0 | DSA[7:0] | | | | | | | |
| | | 15:8 | DSA[15:8] | | | | | | | |
| 0xF2 | DMAnSCNT | 7:0 | SCNT[7:0] | | | | | | | |
| | | 15:8 | | | | | SCNT[11:8] | | | |
| 0xF4 | DMAnSPTR | 7:0 | SPTR[7:0] | | | | | | | |
| | | 15:8 | SPTR[15:8] | | | | | | | |
| | | 23:16 | | | SPTR[21:16] | | | | | |
| 0xF7 | DMAnSSZ | 7:0 | SSZ[7:0] | | | | | | | |
| | | 15:8 | | | | | SSZ[11:8] | | | |
| 0xF9 | DMAnSSA | 7:0 | SSA[7:0] | | | | | | | |
| | | 15:8 | SSA[15:8] | | | | | | | |
| | | 23:16 | | | SSA[21:16] | | | | | |
| 0xFC | DMAnCON0 | 7:0 | EN | SIRQEN | DGO | | | AIRQEN | | XIP |
| 0xFD | DMAnCON1 | 7:0 | DMODE[1:0] | | DSTP | SMR[1:0] | | SMODE[1:0] | | SSTP |
| 0xFE | DMAnAIRQ | 7:0 | AIRQ[7:0] | | | | | | | |
| 0xFF | DMAnSIRQ | 7:0 | SIRQ[7:0] | | | | | | | |
| 0x0100 ... 0x01FF | Reserved | | | | | | | | | |
| 0x0200 | PPSLOCK | 7:0 | | | | | | | | PPSLOCKED |
| 0x0201 | RA0PPS | 7:0 | | RA0PPS[6:0] | | | | | | |
| 0x0202 | RA1PPS | 7:0 | | RA1PPS[6:0] | | | | | | |
| 0x0203 | RA2PPS | 7:0 | | RA2PPS[6:0] | | | | | | |
| 0x0204 | RA3PPS | 7:0 | | RA3PPS[6:0] | | | | | | |
| 0x0205 | RA4PPS | 7:0 | | RA4PPS[6:0] | | | | | | |
| 0x0206 | RA5PPS | 7:0 | | RA5PPS[6:0] | | | | | | |
| 0x0207 | RA6PPS | 7:0 | | RA6PPS[6:0] | | | | | | |
| 0x0208 | RA7PPS | 7:0 | | RA7PPS[6:0] | | | | | | |
| 0x0209 | RB0PPS | 7:0 | | RB0PPS[6:0] | | | | | | |
| 0x020A | RB1PPS | 7:0 | | RB1PPS[6:0] | | | | | | |
| 0x020B | RB2PPS | 7:0 | | RB2PPS[6:0] | | | | | | |
| 0x020C | RB3PPS | 7:0 | | RB3PPS[6:0] | | | | | | |
| 0x020D | RB4PPS | 7:0 | | RB4PPS[6:0] | | | | | | |
| 0x020E | RB5PPS | 7:0 | | RB5PPS[6:0] | | | | | | |
| 0x020F | RB6PPS | 7:0 | | RB6PPS[6:0] | | | | | | |
| 0x0210 | RB7PPS | 7:0 | | RB7PPS[6:0] | | | | | | |
| 0x0211 | RC0PPS | 7:0 | | RC0PPS[6:0] | | | | | | |
| 0x0212 | RC1PPS | 7:0 | | RC1PPS[6:0] | | | | | | |
| 0x0213 | RC2PPS | 7:0 | | RC2PPS[6:0] | | | | | | |
| 0x0214 | RC3PPS | 7:0 | | RC3PPS[6:0] | | | | | | |
| 0x0215 | RC4PPS | 7:0 | | RC4PPS[6:0] | | | | | | |
| 0x0216 | RC5PPS | 7:0 | | RC5PPS[6:0] | | | | | | |
| 0x0217 | RC6PPS | 7:0 | | RC6PPS[6:0] | | | | | | |
| 0x0218 | RC7PPS | 7:0 | | RC7PPS[6:0] | | | | | | |

.....continued

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0219 | RD0PPS | 7:0 | | | | | RD0PPS[6:0] | | | |
| 0x021A | RD1PPS | 7:0 | | | | | RD1PPS[6:0] | | | |
| 0x021B | RD2PPS | 7:0 | | | | | RD2PPS[6:0] | | | |
| 0x021C | RD3PPS | 7:0 | | | | | RD3PPS[6:0] | | | |
| 0x021D | RD4PPS | 7:0 | | | | | RD4PPS[6:0] | | | |
| 0x021E | RD5PPS | 7:0 | | | | | RD5PPS[6:0] | | | |
| 0x021F | RD6PPS | 7:0 | | | | | RD6PPS[6:0] | | | |
| 0x0220 | RD7PPS | 7:0 | | | | | RD7PPS[6:0] | | | |
| 0x0221 | RE0PPS | 7:0 | | | | | RE0PPS[6:0] | | | |
| 0x0222 | RE1PPS | 7:0 | | | | | RE1PPS[6:0] | | | |
| 0x0223 | RE2PPS | 7:0 | | | | | RE2PPS[6:0] | | | |
| 0x0224 ... 0x0228 | Reserved | | | | | | | | | |
| 0x0229 | RF0PPS | 7:0 | | | | | RF0PPS[6:0] | | | |
| 0x022A | RF1PPS | 7:0 | | | | | RF1PPS[6:0] | | | |
| 0x022B | RF2PPS | 7:0 | | | | | RF2PPS[6:0] | | | |
| 0x022C | RF3PPS | 7:0 | | | | | RF3PPS[6:0] | | | |
| 0x022D | RF4PPS | 7:0 | | | | | RF4PPS[6:0] | | | |
| 0x022E | RF5PPS | 7:0 | | | | | RF5PPS[6:0] | | | |
| 0x022F | RF6PPS | 7:0 | | | | | RF6PPS[6:0] | | | |
| 0x0230 | RF7PPS | 7:0 | | | | | RF7PPS[6:0] | | | |
| 0x0231 ... 0x023D | Reserved | | | | | | | | | |
| 0x023E | INT0PPS | 7:0 | | | | | PORT | PIN[2:0] | | |
| 0x023F | INT1PPS | 7:0 | | | | PORT[1:0] | | PIN[2:0] | | |
| 0x0240 | INT2PPS | 7:0 | | | PORT[2:0] | | | PIN[2:0] | | |
| 0x0241 | T0CKIPPS | 7:0 | | | PORT[2:0] | | | PIN[2:0] | | |
| 0x0242 | T1CKIPPS | 7:0 | | | PORT[2:0] | | | PIN[2:0] | | |
| 0x0243 | T1GPPS | 7:0 | | | | PORT[1:0] | | PIN[2:0] | | |
| 0x0244 | T3CKIPPS | 7:0 | | | PORT[2:0] | | | PIN[2:0] | | |
| 0x0245 | T3GPPS | 7:0 | | | | PORT[1:0] | | PIN[2:0] | | |
| 0x0246 | T5CKIPPS | 7:0 | | | PORT[2:0] | | | PIN[2:0] | | |
| 0x0247 | T5GPPS | 7:0 | | | | PORT[1:0] | | PIN[2:0] | | |
| 0x0248 | T2INPPS | 7:0 | | | | PORT[1:0] | | PIN[2:0] | | |
| 0x0249 | T4INPPS | 7:0 | | | | PORT[1:0] | | PIN[2:0] | | |
| 0x024A | T6INPPS | 7:0 | | | | PORT[1:0] | | PIN[2:0] | | |
| 0x024B ... 0x024E | Reserved | | | | | | | | | |
| 0x024F | CCP1PPS | 7:0 | | | PORT[2:0] | | | PIN[2:0] | | |
| 0x0250 | CCP2PPS | 7:0 | | | PORT[2:0] | | | PIN[2:0] | | |
| 0x0251 | CCP3PPS | 7:0 | | | | PORT[1:0] | | PIN[2:0] | | |
| 0x0252 | Reserved | | | | | | | | | |
| 0x0253 | PWM1ERSPPS | 7:0 | | | | PORT[1:0] | | PIN[2:0] | | |
| 0x0254 | PWM2ERSPPS | 7:0 | | | PORT[2:0] | | | PIN[2:0] | | |
| 0x0255 | PWM3ERSPPS | 7:0 | | | | PORT[1:0] | | PIN[2:0] | | |
| 0x0256 | Reserved | | | | | | | | | |
| 0x0257 | PWMIN0PPS | 7:0 | | | PORT[2:0] | | | PIN[2:0] | | |
| 0x0258 | PWMIN1PPS | 7:0 | | | PORT[2:0] | | | PIN[2:0] | | |
| 0x0259 | SMT1WINPPS | 7:0 | | | PORT[2:0] | | | PIN[2:0] | | |
| 0x025A | SMT1SIGPPS | 7:0 | | | PORT[2:0] | | | PIN[2:0] | | |
| 0x025B | CWG1PPS | 7:0 | | | | PORT[1:0] | | PIN[2:0] | | |
| 0x025C | CWG2PPS | 7:0 | | | | PORT[1:0] | | PIN[2:0] | | |
| 0x025D | CWG3PPS | 7:0 | | | | PORT[1:0] | | PIN[2:0] | | |
| 0x025E | MD1CARLPPS | 7:0 | | | | PORT[1:0] | | PIN[2:0] | | |
| 0x025F | MD1CARHPPS | 7:0 | | | | PORT[1:0] | | PIN[2:0] | | |
| 0x0260 | MD1SRCPPS | 7:0 | | | | PORT[1:0] | | PIN[2:0] | | |

© 2019-2023 Microchip Technology Inc. and its subsidiaries

.....continued

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0261 | CLCIN0PPS | 7:0 | | | | PORT[1:0] | | | PIN[2:0] | |
| 0x0262 | CLCIN1PPS | 7:0 | | | | PORT[1:0] | | | PIN[2:0] | |
| 0x0263 | CLCIN2PPS | 7:0 | | | | PORT[1:0] | | | PIN[2:0] | |
| 0x0264 | CLCIN3PPS | 7:0 | | | | PORT[1:0] | | | PIN[2:0] | |
| 0x0265 | CLCIN4PPS | 7:0 | | | | PORT[1:0] | | | PIN[2:0] | |
| 0x0266 | CLCIN5PPS | 7:0 | | | | PORT[1:0] | | | PIN[2:0] | |
| 0x0267 | CLCIN6PPS | 7:0 | | | | PORT[1:0] | | | PIN[2:0] | |
| 0x0268 | CLCIN7PPS | 7:0 | | | | PORT[1:0] | | | PIN[2:0] | |
| 0x0269 | ADACTPPS | 7:0 | | | | PORT[1:0] | | | PIN[2:0] | |
| 0x026A | SPI1SCKPPS | 7:0 | | | | PORT[1:0] | | | PIN[2:0] | |
| 0x026B | SPI1SDIPPS | 7:0 | | | | PORT[1:0] | | | PIN[2:0] | |
| 0x026C | SPI1SSPPS | 7:0 | | | | PORT[1:0] | | | PIN[2:0] | |
| 0x026D | SPI2SCKPPS | 7:0 | | | | PORT[1:0] | | | PIN[2:0] | |
| 0x026E | SPI2SDIPPS | 7:0 | | | | PORT[1:0] | | | PIN[2:0] | |
| 0x026F | SPI2SSPPS | 7:0 | | | | PORT[1:0] | | | PIN[2:0] | |
| 0x0270 | I2C1SDAPPS | 7:0 | | | | PORT[1:0] | | | PIN[2:0] | |
| 0x0271 | I2C1SCLPPS | 7:0 | | | | PORT[1:0] | | | PIN[2:0] | |
| 0x0272 | U1RXPPS | 7:0 | | | PORT[2:0] | | | | PIN[2:0] | |
| 0x0273 | U1CTSPPS | 7:0 | | | PORT[2:0] | | | | PIN[2:0] | |
| 0x0274 | U2RXPPS | 7:0 | | | | PORT[1:0] | | | PIN[2:0] | |
| 0x0275 | U2CTSPPS | 7:0 | | | | PORT[1:0] | | | PIN[2:0] | |
| 0x0276 | U3RXPPS | 7:0 | | | PORT[2:0] | | | | PIN[2:0] | |
| 0x0277 | U3CTSPPS | 7:0 | | | PORT[2:0] | | | | PIN[2:0] | |
| 0x0278 | U4RXPPS | 7:0 | | | | PORT[1:0] | | | PIN[2:0] | |
| 0x0279 | U4CTSPPS | 7:0 | | | | PORT[1:0] | | | PIN[2:0] | |
| 0x027A | U5RXPPS | 7:0 | | | PORT[2:0] | | | | PIN[2:0] | |
| 0x027B | U5CTSPPS | 7:0 | | | PORT[2:0] | | | | PIN[2:0] | |
| 0x027C ... 0x0285 | Reserved | | | | | | | | | |
| 0x0286 | RC4I2C | 7:0 | SLEW[1:0] | | PU[1:0] | | | | TH[1:0] | |
| 0x0287 | RC3I2C | 7:0 | SLEW[1:0] | | PU[1:0] | | | | TH[1:0] | |
| 0x0288 | RB2I2C | 7:0 | SLEW[1:0] | | PU[1:0] | | | | TH[1:0] | |
| 0x0289 | RB1I2C | 7:0 | SLEW[1:0] | | PU[1:0] | | | | TH[1:0] | |
| 0x028A | Reserved | | | | | | | | | |
| 0x028B | I2C1RXB | 7:0 | RXB[7:0] | | | | | | | |
| 0x028C | I2C1TXB | 7:0 | TXB[7:0] | | | | | | | |
| 0x028D | I2C1CNT | 7:0 | CNT[7:0] | | | | | | | |
| 0x028E | I2C1ADB0 | 7:0 | ADB[7:0] | | | | | | | |
| 0x028F | I2C1ADB1 | 7:0 | ADB[7:0] | | | | | | | |
| 0x0290 | I2C1ADR0 | 7:0 | ADR[7:0] | | | | | | | |
| 0x0291 | I2C1ADR1 | 7:0 | ADR[6:0] | | | | | | | |
| 0x0292 | I2C1ADR2 | 7:0 | ADR[7:0] | | | | | | | |
| 0x0293 | I2C1ADR3 | 7:0 | ADR[6:0] | | | | | | | |
| 0x0294 | I2C1CON0 | 7:0 | EN | RSEN | S | CSTR | MDR | MODE[2:0] | | |
| 0x0295 | I2C1CON1 | 7:0 | ACKCNT | ACKDT | ACKSTAT | ACKT | | RXO | TXU | CSD |
| 0x0296 | I2C1CON2 | 7:0 | ACNT | GCEN | FME | ABD | SDAHT[1:0] | | BFRET[1:0] | |
| 0x0297 | I2C1ERR | 7:0 | | BTOIF | BCLIF | NACKIF | | BTOIE | BLCIE | NACKIE |
| 0x0298 | I2C1STAT0 | 7:0 | BFRE | SMA | MMA | R | D | | | |
| 0x0299 | I2C1STAT1 | 7:0 | TXWE | | TXBE | | RXRE | CLRBF | | RXBF |
| 0x029A | I2C1PIR | 7:0 | CNTIF | ACKTIF | | WRIF | ADRIF | PCIF | RSCIF | SCIF |
| 0x029B | I2C1PIE | 7:0 | CNTIE | ACKTIE | | WRIE | ADRIE | PCIE | RSCIE | SCIE |
| 0x029C | I2C1CLK | 7:0 | | | | CLK[4:0] | | | | |
| 0x029D | I2C1BTO | 7:0 | | | | | BTO[3:0] | | | |
| 0x029E ... 0x02A0 | Reserved | | | | | | | | | |
| 0x02A1 | U1RXB | 7:0 | RXB[7:0] | | | | | | | |
| 0x02A2 | U1RXCHK | 7:0 | RXCHK[7:0] | | | | | | | |

.....continued

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x02A3 | U1TXB | 7:0 | TXB[7:0] | | | | | | | |
| 0x02A4 | U1TXCHK | 7:0 | TXCHK[7:0] | | | | | | | |
| 0x02A5 | U1P1 | 7:0 | P1[7:0] | | | | | | | |
| | | 15:8 | | | | | | | | P1[8] |
| 0x02A7 | U1P2 | 7:0 | P2[7:0] | | | | | | | |
| | | 15:8 | | | | | | | | P2[8] |
| 0x02A9 | U1P3 | 7:0 | P3[7:0] | | | | | | | |
| | | 15:8 | | | | | | | | P3[8] |
| 0x02AB | U1CON0 | 7:0 | BRGS | ABDEN | TXEN | RXEN | MODE[3:0] | | | |
| 0x02AC | U1CON1 | 7:0 | ON | | | WUE | RXBIMD | | BRKOVR | SENDB |
| 0x02AD | U1CON2 | 7:0 | RUNOVF | RXPOL | STP[1:0] | | C0EN | TXPOL | FLO[1:0] | |
| 0x02AE | U1BRG | 7:0 | BRG[7:0] | | | | | | | |
| | | 15:8 | BRG[15:8] | | | | | | | |
| 0x02B0 | U1FIFO | 7:0 | TXWRE | STPMD | TXBE | TXBF | RXIDL | XON | RXBE | RXBF |
| 0x02B1 | U1UIR | 7:0 | WUIF | ABDIF | | | | ABDIE | | |
| 0x02B2 | U1ERRIR | 7:0 | TXMTIF | PERIF | ABDOVF | CERIF | FERIF | RXBKIF | RXFOIF | TXCIF |
| 0x02B3 | U1ERRIE | 7:0 | TXMTIE | PERIE | ABDOVE | CERIE | FERIE | RXBKIE | RXFOIE | TXCIE |
| 0x02B4 | U2RXB | 7:0 | RXB[7:0] | | | | | | | |
| 0x02B5 | Reserved | | | | | | | | | |
| 0x02B6 | U2TXB | 7:0 | TXB[7:0] | | | | | | | |
| 0x02B7 | Reserved | | | | | | | | | |
| 0x02B8 | U2P1 | 7:0 | P1[7:0] | | | | | | | |
| | | 15:8 | | | | | | | | |
| 0x02BA | U2P2 | 7:0 | P2[7:0] | | | | | | | |
| | | 15:8 | | | | | | | | |
| 0x02BC | U2P3 | 7:0 | P3[7:0] | | | | | | | |
| | | 15:8 | | | | | | | | |
| 0x02BE | U2CON0 | 7:0 | BRGS | ABDEN | TXEN | RXEN | MODE[3:0] | | | |
| 0x02BF | U2CON1 | 7:0 | ON | | | WUE | RXBIMD | | BRKOVR | SENDB |
| 0x02C0 | U2CON2 | 7:0 | RUNOVF | RXPOL | STP[1:0] | | | TXPOL | FLO[1:0] | |
| 0x02C1 | U2BRG | 7:0 | BRG[7:0] | | | | | | | |
| | | 15:8 | BRG[15:8] | | | | | | | |
| 0x02C3 | U2FIFO | 7:0 | TXWRE | STPMD | TXBE | TXBF | RXIDL | XON | RXBE | RXBF |
| 0x02C4 | U2UIR | 7:0 | WUIF | ABDIF | | | | ABDIE | | |
| 0x02C5 | U2ERRIR | 7:0 | TXMTIF | PERIF | ABDOVF | CERIF | FERIF | RXBKIF | RXFOIF | |
| 0x02C6 | U2ERRIE | 7:0 | TXMTIE | PERIE | ABDOVE | CERIE | FERIE | RXBKIE | RXFOIE | |
| 0x02C7 | U3RXB | 7:0 | RXB[7:0] | | | | | | | |
| 0x02C8 | Reserved | | | | | | | | | |
| 0x02C9 | U3TXB | 7:0 | TXB[7:0] | | | | | | | |
| 0x02CA | Reserved | | | | | | | | | |
| 0x02CB | U3P1 | 7:0 | P1[7:0] | | | | | | | |
| | | 15:8 | | | | | | | | |
| 0x02CD | U3P2 | 7:0 | P2[7:0] | | | | | | | |
| | | 15:8 | | | | | | | | |
| 0x02CF | U3P3 | 7:0 | P3[7:0] | | | | | | | |
| | | 15:8 | | | | | | | | |
| 0x02D1 | U3CON0 | 7:0 | BRGS | ABDEN | TXEN | RXEN | MODE[3:0] | | | |
| 0x02D2 | U3CON1 | 7:0 | ON | | | WUE | RXBIMD | | BRKOVR | SENDB |
| 0x02D3 | U3CON2 | 7:0 | RUNOVF | RXPOL | STP[1:0] | | | TXPOL | FLO[1:0] | |
| 0x02D4 | U3BRG | 7:0 | BRG[7:0] | | | | | | | |
| | | 15:8 | BRG[15:8] | | | | | | | |
| 0x02D6 | U3FIFO | 7:0 | TXWRE | STPMD | TXBE | TXBF | RXIDL | XON | RXBE | RXBF |
| 0x02D7 | U3UIR | 7:0 | WUIF | ABDIF | | | | ABDIE | | |
| 0x02D8 | U3ERRIR | 7:0 | TXMTIF | PERIF | ABDOVF | CERIF | FERIF | RXBKIF | RXFOIF | |
| 0x02D9 | U3ERRIE | 7:0 | TXMTIE | PERIE | ABDOVE | CERIE | FERIE | RXBKIE | RXFOIE | |
| 0x02DA | U4RXB | 7:0 | RXB[7:0] | | | | | | | |
| 0x02DB | Reserved | | | | | | | | | |
| 0x02DC | U4TXB | 7:0 | TXB[7:0] | | | | | | | |
| 0x02DD | Reserved | | | | | | | | | |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

.....continued

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x02DE | U4P1 | 7:0 | P1[7:0] | | | | | | | |
| | | 15:8 | | | | | | | | |
| 0x02E0 | U4P2 | 7:0 | P2[7:0] | | | | | | | |
| | | 15:8 | | | | | | | | |
| 0x02E2 | U4P3 | 7:0 | P3[7:0] | | | | | | | |
| | | 15:8 | | | | | | | | |
| 0x02E4 | U4CON0 | 7:0 | BRGS | ABDEN | TXEN | RXEN | MODE[3:0] | | | |
| 0x02E5 | U4CON1 | 7:0 | ON | | | WUE | RXBIMD | | BRKOVR | SENDB |
| 0x02E6 | U4CON2 | 7:0 | RUNOVF | RXPOL | STP[1:0] | | | TXPOL | FLO[1:0] | |
| 0x02E7 | U4BRG | 7:0 | BRG[7:0] | | | | | | | |
| | | 15:8 | BRG[15:8] | | | | | | | |
| 0x02E9 | U4FIFO | 7:0 | TXWRE | STPMD | TXBE | TXBF | RXIDL | XON | RXBE | RXBF |
| 0x02EA | U4UIR | 7:0 | WUIF | ABDIF | | | | ABDIE | | |
| 0x02EB | U4ERRIR | 7:0 | TXMTIF | PERIF | ABDOVF | CERIF | FERIF | RXBKIF | RXFOIF | |
| 0x02EC | U4ERRIE | 7:0 | TXMTIE | PERIE | ABDOVE | CERIE | FERIE | RXBKIE | RXFOIE | |
| 0x02ED | U5RXB | 7:0 | RXB[7:0] | | | | | | | |
| 0x02EE | Reserved | | | | | | | | | |
| 0x02EF | U5TXB | 7:0 | TXB[7:0] | | | | | | | |
| 0x02F0 | Reserved | | | | | | | | | |
| 0x02F1 | U5P1 | 7:0 | P1[7:0] | | | | | | | |
| | | 15:8 | | | | | | | | |
| 0x02F3 | U5P2 | 7:0 | P2[7:0] | | | | | | | |
| | | 15:8 | | | | | | | | |
| 0x02F5 | U5P3 | 7:0 | P3[7:0] | | | | | | | |
| | | 15:8 | | | | | | | | |
| 0x02F7 | U5CON0 | 7:0 | BRGS | ABDEN | TXEN | RXEN | MODE[3:0] | | | |
| 0x02F8 | U5CON1 | 7:0 | ON | | | WUE | RXBIMD | | BRKOVR | SENDB |
| 0x02F9 | U5CON2 | 7:0 | RUNOVF | RXPOL | STP[1:0] | | | TXPOL | FLO[1:0] | |
| 0x02FA | U5BRG | 7:0 | BRG[7:0] | | | | | | | |
| | | 15:8 | BRG[15:8] | | | | | | | |
| 0x02FC | U5FIFO | 7:0 | TXWRE | STPMD | TXBE | TXBF | RXIDL | XON | RXBE | RXBF |
| 0x02FD | U5UIR | 7:0 | WUIF | ABDIF | | | | ABDIE | | |
| 0x02FE | U5ERRIR | 7:0 | TXMTIF | PERIF | ABDOVF | CERIF | FERIF | RXBKIF | RXFOIF | |
| 0x02FF | U5ERRIE | 7:0 | TXMTIE | PERIE | ABDOVE | CERIE | FERIE | RXBKIE | RXFOIE | |
| 0x0300 | SMT1TMR | 7:0 | TMR[7:0] | | | | | | | |
| | | 15:8 | TMR[15:8] | | | | | | | |
| | | 23:16 | TMR[23:16] | | | | | | | |
| 0x0303 | SMT1CPR | 7:0 | CPR[7:0] | | | | | | | |
| | | 15:8 | CPR[15:8] | | | | | | | |
| | | 23:16 | CPR[23:16] | | | | | | | |
| 0x0306 | SMT1CPW | 7:0 | CPW[7:0] | | | | | | | |
| | | 15:8 | CPW[15:8] | | | | | | | |
| | | 23:16 | CPW[23:16] | | | | | | | |
| 0x0309 | SMT1PR | 7:0 | PR[7:0] | | | | | | | |
| | | 15:8 | PR[15:8] | | | | | | | |
| | | 23:16 | PR[23:16] | | | | | | | |
| 0x030C | SMT1CON0 | 7:0 | EN | | STP | WPOL | SPOL | CPOL | PS[1:0] | |
| 0x030D | SMT1CON1 | 7:0 | GO | REPEAT | | | MODE[3:0] | | | |
| 0x030E | SMT1STAT | 7:0 | CPRUP | CPWUP | RST | | | TS | WS | AS |
| 0x030F | SMT1CLK | 7:0 | | | | | CSEL[3:0] | | | |
| 0x0310 | SMT1SIG | 7:0 | | | SSEL[5:0] | | | | | |
| 0x0311 | SMT1WIN | 7:0 | | | WSEL[5:0] | | | | | |
| 0x0312 ... 0x0317 | Reserved | | | | | | | | | |
| 0x0318 | TMR0L | 7:0 | TMR0L[7:0] | | | | | | | |
| 0x0319 | TMR0H | 7:0 | TMR0H[7:0] | | | | | | | |
| 0x031A | T0CON0 | 7:0 | EN | | OUT | MD16 | OUTPS[3:0] | | | |
| 0x031B | T0CON1 | 7:0 | CS[2:0] | | | ASYNC | CKPS[3:0] | | | |

.....continued

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x031C | TMR1 | 7:0 | TMR1[7:0] | | | | | | | |
| | | 15:8 | TMR1[15:8] | | | | | | | |
| 0x031E | T1CON | 7:0 | | | CKPS[1:0] | | | SYNC | RD16 | ON |
| 0x031F | T1GCON | 7:0 | GE | GPOL | GTM | GSPM | GGO/DONE | GVAL | | |
| 0x0320 | T1GATE | 7:0 | | | GSS[5:0] | | | | | |
| 0x0321 | T1CLK | 7:0 | | | | CS[4:0] | | | | |
| 0x0322 | T2TMR | 7:0 | T2TMR[7:0] | | | | | | | |
| 0x0323 | T2PR | 7:0 | T2PR[7:0] | | | | | | | |
| 0x0324 | T2CON | 7:0 | ON | | CKPS[2:0] | | OUTPS[3:0] | | | |
| 0x0325 | T2HLT | 7:0 | PSYNC | CPOL | CSYNC | MODE[4:0] | | | | |
| 0x0326 | T2CLKCON | 7:0 | | | | CS[4:0] | | | | |
| 0x0327 | T2RST | 7:0 | | | RSEL[5:0] | | | | | |
| 0x0328 | TMR3 | 7:0 | TMR3[7:0] | | | | | | | |
| | | 15:8 | TMR3[15:8] | | | | | | | |
| 0x032A | T3CON | 7:0 | | | CKPS[1:0] | | | SYNC | RD16 | ON |
| 0x032B | T3GCON | 7:0 | GE | GPOL | GTM | GSPM | GGO/DONE | GVAL | | |
| 0x032C | T3GATE | 7:0 | | | GSS[5:0] | | | | | |
| 0x032D | T3CLK | 7:0 | | | | CS[4:0] | | | | |
| 0x032E | T4TMR | 7:0 | T4TMR[7:0] | | | | | | | |
| 0x032F | T4PR | 7:0 | T4PR[7:0] | | | | | | | |
| 0x0330 | T4CON | 7:0 | ON | | CKPS[2:0] | | OUTPS[3:0] | | | |
| 0x0331 | T4HLT | 7:0 | PSYNC | CPOL | CSYNC | MODE[4:0] | | | | |
| 0x0332 | T4CLKCON | 7:0 | | | | CS[4:0] | | | | |
| 0x0333 | T4RST | 7:0 | | | RSEL[5:0] | | | | | |
| 0x0334 | TMR5 | 7:0 | TMR5[7:0] | | | | | | | |
| | | 15:8 | TMR5[15:8] | | | | | | | |
| 0x0336 | T5CON | 7:0 | | | CKPS[1:0] | | | SYNC | RD16 | ON |
| 0x0337 | T5GCON | 7:0 | GE | GPOL | GTM | GSPM | GGO/DONE | GVAL | | |
| 0x0338 | T5GATE | 7:0 | | | GSS[5:0] | | | | | |
| 0x0339 | T5CLK | 7:0 | | | | CS[4:0] | | | | |
| 0x033A | T6TMR | 7:0 | T6TMR[7:0] | | | | | | | |
| 0x033B | T6PR | 7:0 | T6PR[7:0] | | | | | | | |
| 0x033C | T6CON | 7:0 | ON | | CKPS[2:0] | | OUTPS[3:0] | | | |
| 0x033D | T6HLT | 7:0 | PSYNC | CPOL | CSYNC | MODE[4:0] | | | | |
| 0x033E | T6CLKCON | 7:0 | | | | CS[4:0] | | | | |
| 0x033F | T6RST | 7:0 | | | RSEL[5:0] | | | | | |
| 0x0340 | CCPR1 | 7:0 | CCPR[7:0] | | | | | | | |
| | | 15:8 | CCPR[15:8] | | | | | | | |
| 0x0342 | CCP1CON | 7:0 | EN | | OUT | FMT | MODE[3:0] | | | |
| 0x0343 | CCP1CAP | 7:0 | | | | | CTS[3:0] | | | |
| 0x0344 | CCPR2 | 7:0 | CCPR[7:0] | | | | | | | |
| | | 15:8 | CCPR[15:8] | | | | | | | |
| 0x0346 | CCP2CON | 7:0 | EN | | OUT | FMT | MODE[3:0] | | | |
| 0x0347 | CCP2CAP | 7:0 | | | | | CTS[3:0] | | | |
| 0x0348 | CCPR3 | 7:0 | CCPR[7:0] | | | | | | | |
| | | 15:8 | CCPR[15:8] | | | | | | | |
| 0x034A | CCP3CON | 7:0 | EN | | OUT | FMT | MODE[3:0] | | | |
| 0x034B | CCP3CAP | 7:0 | | | | | CTS[3:0] | | | |
| 0x034C | CCPTMRS0 | 7:0 | | | C3TSEL[1:0] | | C2TSEL[1:0] | | C1TSEL[1:0] | |
| 0x034D ... 0x034E | Reserved | | | | | | | | | |
| 0x034F | CRCDATA | 7:0 | DATA[7:0] | | | | | | | |
| | | 15:8 | DATA[15:8] | | | | | | | |
| 0x0351 | CRCACC | 7:0 | ACC[7:0] | | | | | | | |
| | | 15:8 | ACC[15:8] | | | | | | | |
| 0x0353 | CRCSHFT | 7:0 | SHFT[7:0] | | | | | | | |
| | | 15:8 | SHFT[15:8] | | | | | | | |

.....continued

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0355 | CRCXOR | 7:0 | X[7:0] | | | | | | | |
| | | 15:8 | X[15:8] | | | | | | | |
| 0x0357 | CRCCON0 | 7:0 | EN | GO | BUSY | ACCM | | | SHIFTM | FULL |
| 0x0358 | CRCCON1 | 7:0 | DLEN[3:0] | | | | PLEN[3:0] | | | |
| 0x0359 | Reserved | | | | | | | | | |
| 0x035A | SCANLADR | 7:0 | SCANLADRL[7:0] | | | | | | | |
| | | 15:8 | SCANLADRH[7:0] | | | | | | | |
| | | 23:16 | | | SCANLADRU[5:0] | | | | | |
| 0x035D | SCANHADR | 7:0 | SCANHADRL[7:0] | | | | | | | |
| | | 15:8 | SCANHADRH[7:0] | | | | | | | |
| | | 23:16 | | | SCANHADRU[5:0] | | | | | |
| 0x0360 | SCANCON0 | 7:0 | EN | TRIGEN | SGO | | | MREG | BURSTMD | BUSY |
| 0x0361 | SCANTRIG | 7:0 | | | | TSEL[4:0] | | | | |
| 0x0362 | IPR0 | 7:0 | IOCIP | | CLC1IP | | CSWIP | OSFIP | HLVDIP | SWIP |
| 0x0363 | IPR1 | 7:0 | SMT1PWAIP | SMT1PRAIP | SMT1IP | CM1IP | ACTIP | ADIP | ZCDIP | INT0IP |
| 0x0364 | IPR2 | 7:0 | DMA1AIP | DMA1ORIP | DMA1DCNTIP | DMA1SCNTIP | | | | ADTIP |
| 0x0365 | IPR3 | 7:0 | TMR0IP | CCP1IP | TMR1GIP | TMR1IP | TMR2IP | SPI1IP | SPI1TXIP | SPI1RXIP |
| 0x0366 | IPR4 | 7:0 | PWM1IP | PWM1PIP | | | U1IP | U1EIP | U1TXIP | U1RXIP |
| 0x0367 | IPR5 | 7:0 | PWM2IP | PWM2PIP | TMR3GIP | TMR3IP | | SPI2IP | SPI2TXIP | SPI2RXIP |
| 0x0368 | IPR6 | 7:0 | DMA2AIP | DMA2ORIP | DMA2DCNTIP | DMA2SCNTIP | NCO1IP | CWG1IP | CLC2IP | INT1IP |
| 0x0369 | IPR7 | 7:0 | PWM3IP | PWM3PIP | CLC3IP | | I2C1EIP | I2C1IP | I2C1TXIP | I2C1RXIP |
| 0x036A | IPR8 | 7:0 | SCANIP | CCP2IP | TMR5GIP | TMR5IP | U2IP | U2EIP | U2TXIP | U2RXIP |
| 0x036B | IPR9 | 7:0 | | | CLC4IP | | U3IP | U3EIP | U3TXIP | U3RXIP |
| 0x036C | IPR10 | 7:0 | DMA3AIP | DMA3ORIP | DMA3DCNTIP | DMA3SCNTIP | NCO2IP | CWG2IP | CLC5IP | INT2IP |
| 0x036D | IPR11 | 7:0 | DMA4AIP | DMA4ORIP | DMA4DCNTIP | DMA4SCNTIP | TMR4IP | CWG3IP | CLC6IP | CCP3IP |
| 0x036E | IPR12 | 7:0 | DMA5AIP | DMA5ORIP | DMA5DCNTIP | DMA5SCNTIP | U4IP | U4EIP | U4TXIP | U4RXIP |
| 0x036F | IPR13 | 7:0 | DMA6AIP | DMA6ORIP | DMA6DCNTIP | DMA6SCNTIP | U5IP | U5EIP | U5TXIP | U5RXIP |
| 0x0370 | IPR14 | 7:0 | | | | | NCO3IP | CM2IP | CLC7IP | |
| 0x0371 | IPR15 | 7:0 | | | | | TMR6IP | CRCIP | CLC8IP | NVMIP |
| 0x0372 | Reserved | | | | | | | | | |
| 0x0373 | STATUS_CSHAD | 7:0 | | $\overline{TO}$ | $\overline{PD}$ | N | OV | Z | DC | C |
| 0x0374 | WREG_CSHAD | 7:0 | WREG[7:0] | | | | | | | |
| 0x0375 | BSR_CSHAD | 7:0 | | | BSR[5:0] | | | | | |
| 0x0376 | SHADCON | 7:0 | | | | | | | | SHADLO |
| 0x0377 | STATUS_SHAD | 7:0 | | $\overline{TO}$ | $\overline{PD}$ | N | OV | Z | DC | C |
| 0x0378 | WREG_SHAD | 7:0 | WREG[7:0] | | | | | | | |
| 0x0379 | BSR_SHAD | 7:0 | | | BSR[5:0] | | | | | |
| 0x037A | PCLAT_SHAD | 7:0 | PCLATH[7:0] | | | | | | | |
| | | 15:8 | | | | PCLATU[4:0] | | | | |
| 0x037C | FSR0_SHAD | 7:0 | FSRL[7:0] | | | | | | | |
| | | 15:8 | | | FSRH[5:0] | | | | | |
| 0x037E | FSR1_SHAD | 7:0 | FSRL[7:0] | | | | | | | |
| | | 15:8 | | | FSRH[5:0] | | | | | |
| 0x0380 | FSR2_SHAD | 7:0 | FSRL[7:0] | | | | | | | |
| | | 15:8 | | | FSRH[5:0] | | | | | |
| 0x0382 | PROD_SHAD | 7:0 | PROD[7:0] | | | | | | | |
| | | 15:8 | PROD[15:8] | | | | | | | |
| 0x0384 ... 0x03BB | Reserved | | | | | | | | | |
| 0x03BC | CWG1CLK | 7:0 | | | | | | | | CS |
| 0x03BD | CWG1ISM | 7:0 | | | | ISM[4:0] | | | | |
| 0x03BE | CWG1DBR | 7:0 | | | DBR[5:0] | | | | | |
| 0x03BF | CWG1DBF | 7:0 | | | DBF[5:0] | | | | | |
| 0x03C0 | CWG1CON0 | 7:0 | EN | LD | | | | MODE[2:0] | | |
| 0x03C1 | CWG1CON1 | 7:0 | | | IN | | POLD | POLC | POLB | POLA |
| 0x03C2 | CWG1AS0 | 7:0 | SHUTDOWN | REN | LSBD[1:0] | | LSAC[1:0] | | | |
| 0x03C3 | CWG1AS1 | 7:0 | AS7E | AS6E | AS5E | AS4E | AS3E | AS2E | AS1E | AS0E |
| 0x03C4 | CWG1STR | 7:0 | OVRD | OVRC | OVRB | OVRA | STRD | STRC | STRB | STRA |

.....continued

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x03C5 | CWG2CLK | 7:0 | | | | | | | | CS |
| 0x03C6 | CWG2ISM | 7:0 | | | | ISM[4:0] | | | | |
| 0x03C7 | CWG2DBR | 7:0 | | | DBR[5:0] | | | | | |
| 0x03C8 | CWG2DBF | 7:0 | | | DBF[5:0] | | | | | |
| 0x03C9 | CWG2CON0 | 7:0 | EN | LD | | | | MODE[2:0] | | |
| 0x03CA | CWG2CON1 | 7:0 | | | IN | | POLD | POLC | POLB | POLA |
| 0x03CB | CWG2AS0 | 7:0 | SHUTDOWN | REN | LSBD[1:0] | | LSAC[1:0] | | | |
| 0x03CC | CWG2AS1 | 7:0 | AS7E | AS6E | AS5E | AS4E | AS3E | AS2E | AS1E | AS0E |
| 0x03CD | CWG2STR | 7:0 | OVRD | OVRC | OVRB | OVRA | STRD | STRC | STRB | STRA |
| 0x03CE | CWG3CLK | 7:0 | | | | | | | | CS |
| 0x03CF | CWG3ISM | 7:0 | | | | ISM[4:0] | | | | |
| 0x03D0 | CWG3DBR | 7:0 | | | DBR[5:0] | | | | | |
| 0x03D1 | CWG3DBF | 7:0 | | | DBF[5:0] | | | | | |
| 0x03D2 | CWG3CON0 | 7:0 | EN | LD | | | | MODE[2:0] | | |
| 0x03D3 | CWG3CON1 | 7:0 | | | IN | | POLD | POLC | POLB | POLA |
| 0x03D4 | CWG3AS0 | 7:0 | SHUTDOWN | REN | LSBD[1:0] | | LSAC[1:0] | | | |
| 0x03D5 | CWG3AS1 | 7:0 | AS7E | AS6E | AS5E | AS4E | AS3E | AS2E | AS1E | AS0E |
| 0x03D6 | CWG3STR | 7:0 | OVRD | OVRC | OVRB | OVRA | STRD | STRC | STRB | STRA |
| 0x03D7 | FVRCON | 7:0 | EN | RDY | TSEN | TSRNG | CDAFVR[1:0] | | ADFVR[1:0] | |
| 0x03D8 | ADCP | 7:0 | CPON | | | | | | | CPRDY |
| 0x03D9 | ADLTH | 7:0 | LTH[7:0] | | | | | | | |
| | | 15:8 | LTH[15:8] | | | | | | | |
| 0x03DB | ADUTH | 7:0 | UTH[7:0] | | | | | | | |
| | | 15:8 | UTH[15:8] | | | | | | | |
| 0x03DD | ADERR | 7:0 | ERR[7:0] | | | | | | | |
| | | 15:8 | ERR[15:8] | | | | | | | |
| 0x03DF | ADSTPT | 7:0 | STPT[7:0] | | | | | | | |
| | | 15:8 | STPT[15:8] | | | | | | | |
| 0x03E1 | ADFLTR | 7:0 | FLTR[7:0] | | | | | | | |
| | | 15:8 | FLTR[15:8] | | | | | | | |
| 0x03E3 | ADACC | 7:0 | ACC[7:0] | | | | | | | |
| | | 15:8 | ACC[15:8] | | | | | | | |
| | | 23:16 | | | | | | | ACC[17:16] | |
| 0x03E6 | ADCNT | 7:0 | CNT[7:0] | | | | | | | |
| 0x03E7 | ADRPT | 7:0 | RPT[7:0] | | | | | | | |
| 0x03E8 | ADPREV | 7:0 | PREV[7:0] | | | | | | | |
| | | 15:8 | PREV[15:8] | | | | | | | |
| 0x03EA | ADRES | 7:0 | RES[7:0] | | | | | | | |
| | | 15:8 | RES[15:8] | | | | | | | |
| 0x03EC | ADPCH | 7:0 | | | PCH[5:0] | | | | | |
| 0x03ED | Reserved | | | | | | | | | |
| 0x03EE | ADACQ | 7:0 | ACQ[7:0] | | | | | | | |
| | | 15:8 | | | | ACQ[12:8] | | | | |
| 0x03F0 | ADCAP | 7:0 | | | | CAP[4:0] | | | | |
| 0x03F1 | ADPRE | 7:0 | PRE[7:0] | | | | | | | |
| | | 15:8 | | | | PRE[12:8] | | | | |
| 0x03F3 | ADCON0 | 7:0 | ON | CONT | | CS | | FM | | GO |
| 0x03F4 | ADCON1 | 7:0 | PPOL | IPEN | GPOL | | | | | DSEN |
| 0x03F5 | ADCON2 | 7:0 | PSIS | CRS[2:0] | | | ACLR | MD[2:0] | | |
| 0x03F6 | ADCON3 | 7:0 | | CALC[2:0] | | | SOI | TMD[2:0] | | |
| 0x03F7 | ADSTAT | 7:0 | AOV | UTHR | LTHR | MATH | | STAT[2:0] | | |
| 0x03F8 | ADREF | 7:0 | | | | NREF | | | PREF[1:0] | |
| 0x03F9 | ADACT | 7:0 | | | ACT[5:0] | | | | | |
| 0x03FA | ADCLK | 7:0 | | | CS[5:0] | | | | | |
| 0x03FB ... 0x03FF | Reserved | | | | | | | | | |
| 0x0400 | ANSELA | 7:0 | ANSELA7 | ANSELA6 | ANSELA5 | ANSELA4 | ANSELA3 | ANSELA2 | ANSELA1 | ANSELA0 |
| 0x0401 | WPUA | 7:0 | WPUA7 | WPUA6 | WPUA5 | WPUA4 | WPUA3 | WPUA2 | WPUA1 | WPUA0 |

.....continued

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0402 | ODCONA | 7:0 | ODCA7 | ODCA6 | ODCA5 | ODCA4 | ODCA3 | ODCA2 | ODCA1 | ODCA0 |
| 0x0403 | SLRCONA | 7:0 | SLRA7 | SLRA6 | SLRA5 | SLRA4 | SLRA3 | SLRA2 | SLRA1 | SLRA0 |
| 0x0404 | INLVLA | 7:0 | INLVLA7 | INLVLA6 | INLVLA5 | INLVLA4 | INLVLA3 | INLVLA2 | INLVLA1 | INLVLA0 |
| 0x0405 | IOCAP | 7:0 | IOCAP7 | IOCAP6 | IOCAP5 | IOCAP4 | IOCAP3 | IOCAP2 | IOCAP1 | IOCAP0 |
| 0x0406 | IOCAN | 7:0 | IOCAN7 | IOCAN6 | IOCAN5 | IOCAN4 | IOCAN3 | IOCAN2 | IOCAN1 | IOCAN0 |
| 0x0407 | IOCAF | 7:0 | IOCAF7 | IOCAF6 | IOCAF5 | IOCAF4 | IOCAF3 | IOCAF2 | IOCAF1 | IOCAF0 |
| 0x0408 | ANSELB | 7:0 | ANSELB7 | ANSELB6 | ANSELB5 | ANSELB4 | ANSELB3 | ANSELB2 | ANSELB1 | ANSELB0 |
| 0x0409 | WPUB | 7:0 | WPUB7 | WPUB6 | WPUB5 | WPUB4 | WPUB3 | WPUB2 | WPUB1 | WPUB0 |
| 0x040A | ODCONB | 7:0 | ODCB7 | ODCB6 | ODCB5 | ODCB4 | ODCB3 | ODCB2 | ODCB1 | ODCB0 |
| 0x040B | SLRCONB | 7:0 | SLRB7 | SLRB6 | SLRB5 | SLRB4 | SLRB3 | SLRB2 | SLRB1 | SLRB0 |
| 0x040C | INLVLB | 7:0 | INLVLB7 | INLVLB6 | INLVLB5 | INLVLB4 | INLVLB3 | INLVLB2 | INLVLB1 | INLVLB0 |
| 0x040D | IOCBP | 7:0 | IOCBP7 | IOCBP6 | IOCBP5 | IOCBP4 | IOCBP3 | IOCBP2 | IOCBP1 | IOCBP0 |
| 0x040E | IOCBN | 7:0 | IOCBN7 | IOCBN6 | IOCBN5 | IOCBN4 | IOCBN3 | IOCBN2 | IOCBN1 | IOCBN0 |
| 0x040F | IOCBF | 7:0 | IOCBF7 | IOCBF6 | IOCBF5 | IOCBF4 | IOCBF3 | IOCBF2 | IOCBF1 | IOCBF0 |
| 0x0410 | ANSELC | 7:0 | ANSELC7 | ANSELC6 | ANSELC5 | ANSELC4 | ANSELC3 | ANSELC2 | ANSELC1 | ANSELC0 |
| 0x0411 | WPUC | 7:0 | WPUC7 | WPUC6 | WPUC5 | WPUC4 | WPUC3 | WPUC2 | WPUC1 | WPUC0 |
| 0x0412 | ODCONC | 7:0 | ODCC7 | ODCC6 | ODCC5 | ODCC4 | ODCC3 | ODCC2 | ODCC1 | ODCC0 |
| 0x0413 | SLRCONC | 7:0 | SLRC7 | SLRC6 | SLRC5 | SLRC4 | SLRC3 | SLRC2 | SLRC1 | SLRC0 |
| 0x0414 | INLVLC | 7:0 | INLVLC7 | INLVLC6 | INLVLC5 | INLVLC4 | INLVLC3 | INLVLC2 | INLVLC1 | INLVLC0 |
| 0x0415 | IOCCP | 7:0 | IOCCP7 | IOCCP6 | IOCCP5 | IOCCP4 | IOCCP3 | IOCCP2 | IOCCP1 | IOCCP0 |
| 0x0416 | IOCCN | 7:0 | IOCCN7 | IOCCN6 | IOCCN5 | IOCCN4 | IOCCN3 | IOCCN2 | IOCCN1 | IOCCN0 |
| 0x0417 | IOCCF | 7:0 | IOCCF7 | IOCCF6 | IOCCF5 | IOCCF4 | IOCCF3 | IOCCF2 | IOCCF1 | IOCCF0 |
| 0x0418 | ANSELD | 7:0 | ANSELD7 | ANSELD6 | ANSELD5 | ANSELD4 | ANSELD3 | ANSELD2 | ANSELD1 | ANSELD0 |
| 0x0419 | WPUD | 7:0 | WPUD7 | WPUD6 | WPUD5 | WPUD4 | WPUD3 | WPUD2 | WPUD1 | WPUD0 |
| 0x041A | ODCOND | 7:0 | ODCD7 | ODCD6 | ODCD5 | ODCD4 | ODCD3 | ODCD2 | ODCD1 | ODCD0 |
| 0x041B | SLRCOND | 7:0 | SLRD7 | SLRD6 | SLRD5 | SLRD4 | SLRD3 | SLRD2 | SLRD1 | SLRD0 |
| 0x041C | INLVLD | 7:0 | INLVLD7 | INLVLD6 | INLVLD5 | INLVLD4 | INLVLD3 | INLVLD2 | INLVLD1 | INLVLD0 |
| 0x041D ... 0x041F | Reserved | | | | | | | | | |
| 0x0420 | ANSELE | 7:0 | | | | | | ANSELE2 | ANSELE1 | ANSELE0 |
| 0x0421 | WPUE | 7:0 | | | | | WPUE3 | WPUE2 | WPUE1 | WPUE0 |
| 0x0422 | ODCONE | 7:0 | | | | | | ODCE2 | ODCE1 | ODCE0 |
| 0x0423 | SLRCONE | 7:0 | | | | | | SLRE2 | SLRE1 | SLRE0 |
| 0x0424 | INLVLE | 7:0 | | | | | INLVLE3 | INLVLE2 | INLVLE1 | INLVLE0 |
| 0x0425 | IOCEP | 7:0 | | | | | IOCEP3 | | | |
| 0x0426 | IOCEN | 7:0 | | | | | IOCEN3 | | | |
| 0x0427 | IOCEF | 7:0 | | | | | IOCEF3 | | | |
| 0x0428 | ANSELF | 7:0 | ANSELF7 | ANSELF6 | ANSELF5 | ANSELF4 | ANSELF3 | ANSELF2 | ANSELF1 | ANSELF0 |
| 0x0429 | WPUF | 7:0 | WPUF7 | WPUF6 | WPUF5 | WPUF4 | WPUF3 | WPUF2 | WPUF1 | WPUF0 |
| 0x042A | ODCONF | 7:0 | ODCF7 | ODCF6 | ODCF5 | ODCF4 | ODCF3 | ODCF2 | ODCF1 | ODCF0 |
| 0x042B | SLRCONF | 7:0 | SLRF7 | SLRF6 | SLRF5 | SLRF4 | SLRF3 | SLRF2 | SLRF1 | SLRF0 |
| 0x042C | INLVLF | 7:0 | INLVLF7 | INLVLF6 | INLVLF5 | INLVLF4 | INLVLF3 | INLVLF2 | INLVLF1 | INLVLF0 |
| 0x042D ... 0x043F | Reserved | | | | | | | | | |
| 0x0440 | NCO1ACC | 7:0 | ACC[7:0] | | | | | | | |
| | | 15:8 | ACC[15:8] | | | | | | | |
| | | 23:16 | | | | | ACC[19:16] | | | |
| 0x0443 | NCO1INC | 7:0 | INC[7:0] | | | | | | | |
| | | 15:8 | INC[15:8] | | | | | | | |
| | | 23:16 | | | | | INC[19:16] | | | |
| 0x0446 | NCO1CON | 7:0 | EN | | OUT | POL | | | | PFM |
| 0x0447 | NCO1CLK | 7:0 | | PWS[2:0] | | CKS[4:0] | | | | |
| 0x0448 | NCO2ACC | 7:0 | ACC[7:0] | | | | | | | |
| | | 15:8 | ACC[15:8] | | | | | | | |
| | | 23:16 | | | | | ACC[19:16] | | | |

.....continued

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x044B | NCO2INC | 7:0 | INC[7:0] | | | | | | | |
| | | 15:8 | INC[15:8] | | | | | | | |
| | | 23:16 | | | | | INC[19:16] | | | |
| 0x044E | NCO2CON | 7:0 | EN | | OUT | POL | | | | PFM |
| 0x044F | NCO2CLK | 7:0 | | PWS[2:0] | | | CKS[4:0] | | | |
| 0x0450 | NCO3ACC | 7:0 | ACC[7:0] | | | | | | | |
| | | 15:8 | ACC[15:8] | | | | | | | |
| | | 23:16 | | | | | ACC[19:16] | | | |
| 0x0453 | NCO3INC | 7:0 | INC[7:0] | | | | | | | |
| | | 15:8 | INC[15:8] | | | | | | | |
| | | 23:16 | | | | | INC[19:16] | | | |
| 0x0456 | NCO3CON | 7:0 | EN | | OUT | POL | | | | PFM |
| 0x0457 | NCO3CLK | 7:0 | | PWS[2:0] | | | CKS[4:0] | | | |
| 0x0458 | Reserved | | | | | | | | | |
| 0x0459 | IVTLOCK | 7:0 | | | | | | | | IVTLOCKED |
| 0x045A | IVTAD | 7:0 | IVTADL[7:0] | | | | | | | |
| | | 15:8 | IVTADH[7:0] | | | | | | | |
| | | 23:16 | | | | IVTADU[4:0] | | | | |
| 0x045D | IVTBASE | 7:0 | IVTBASEL[7:0] | | | | | | | |
| | | 15:8 | IVTBASEH[7:0] | | | | | | | |
| | | 23:16 | | | | IVTBASEU[4:0] | | | | |
| 0x0460 | PWM1ERS | 7:0 | | | | ERS[4:0] | | | | |
| 0x0461 | PWM1CLK | 7:0 | | | | CLK[4:0] | | | | |
| 0x0462 | PWM1LDS | 7:0 | | | | LDS[4:0] | | | | |
| 0x0463 | PWM1PR | 7:0 | PR[7:0] | | | | | | | |
| | | 15:8 | PR[15:8] | | | | | | | |
| 0x0465 | PWM1CPRE | 7:0 | CPRE[7:0] | | | | | | | |
| 0x0466 | PWM1PIPOS | 7:0 | PIPOS[7:0] | | | | | | | |
| 0x0467 | PWM1GIR | 7:0 | | | | | | | S1P2 | S1P1 |
| 0x0468 | PWM1GIE | 7:0 | | | | | | | S1P2 | S1P1 |
| 0x0469 | PWM1CON | 7:0 | EN | | | | | LD | ERSPOL | ERSNOW |
| 0x046A | PWM1S1CFG | 7:0 | POL2 | POL1 | | | PPEN | | MODE[2:0] | |
| 0x046B | PWM1S1P1 | 7:0 | P1[7:0] | | | | | | | |
| | | 15:8 | P1[15:8] | | | | | | | |
| 0x046D | PWM1S1P2 | 7:0 | P2[7:0] | | | | | | | |
| | | 15:8 | P2[15:8] | | | | | | | |
| 0x046F | PWM2ERS | 7:0 | | | | ERS[4:0] | | | | |
| 0x0470 | PWM2CLK | 7:0 | | | | CLK[4:0] | | | | |
| 0x0471 | PWM2LDS | 7:0 | | | | LDS[4:0] | | | | |
| 0x0472 | PWM2PR | 7:0 | PR[7:0] | | | | | | | |
| | | 15:8 | PR[15:8] | | | | | | | |
| 0x0474 | PWM2CPRE | 7:0 | CPRE[7:0] | | | | | | | |
| 0x0475 | PWM2PIPOS | 7:0 | PIPOS[7:0] | | | | | | | |
| 0x0476 | PWM2GIR | 7:0 | | | | | | | S1P2 | S1P1 |
| 0x0477 | PWM2GIE | 7:0 | | | | | | | S1P2 | S1P1 |
| 0x0478 | PWM2CON | 7:0 | EN | | | | | LD | ERSPOL | ERSNOW |
| 0x0479 | PWM2S1CFG | 7:0 | POL2 | POL1 | | | PPEN | | MODE[2:0] | |
| 0x047A | PWM2S1P1 | 7:0 | P1[7:0] | | | | | | | |
| | | 15:8 | P1[15:8] | | | | | | | |
| 0x047C | PWM2S1P2 | 7:0 | P2[7:0] | | | | | | | |
| | | 15:8 | P2[15:8] | | | | | | | |
| 0x047E | PWM3ERS | 7:0 | | | | ERS[4:0] | | | | |
| 0x047F | PWM3CLK | 7:0 | | | | CLK[4:0] | | | | |
| 0x0480 | PWM3LDS | 7:0 | | | | LDS[4:0] | | | | |
| 0x0481 | PWM3PR | 7:0 | PR[7:0] | | | | | | | |
| | | 15:8 | PR[15:8] | | | | | | | |
| 0x0483 | PWM3CPRE | 7:0 | CPRE[7:0] | | | | | | | |
| 0x0484 | PWM3PIPOS | 7:0 | PIPOS[7:0] | | | | | | | |
| 0x0485 | PWM3GIR | 7:0 | | | | | | | S1P2 | S1P1 |

.....continued

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x0486 | PWM3GIE | 7:0 | | | | | | | S1P2 | S1P1 |
| 0x0487 | PWM3CON | 7:0 | EN | | | | | LD | ERSPOL | ERSNOW |
| 0x0488 | PWM3S1CFG | 7:0 | POL2 | POL1 | | | PPEN | | MODE[2:0] | |
| 0x0489 | PWM3S1P1 | 7:0 | P1[7:0] | | | | | | | |
| | | 15:8 | P1[15:8] | | | | | | | |
| 0x048B | PWM3S1P2 | 7:0 | P2[7:0] | | | | | | | |
| | | 15:8 | P2[15:8] | | | | | | | |
| 0x048D ... 0x049B | Reserved | | | | | | | | | |
| 0x049C | PWMLOAD | 7:0 | | | | | | MPWM3LD | MPWM2LD | MPWM1LD |
| 0x049D | PWMEN | 7:0 | | | | | | MPWM3EN | MPWM2EN | MPWM1EN |
| 0x049E | PIE0 | 7:0 | IOCIE | | CLC1IE | | CSWIE | OSFIE | HLVDIE | SWIE |
| 0x049F | PIE1 | 7:0 | SMT1PWAIE | SMT1PRAIE | SMT1IE | CM1IE | ACTIE | ADIE | ZCDIE | INT0IE |
| 0x04A0 | PIE2 | 7:0 | DMA1AIE | DMA1ORIE | DMA1DCNTIE | DMA1SCNTIE | | | | ADTIE |
| 0x04A1 | PIE3 | 7:0 | TMR0IE | CCP1IE | TMR1GIE | TMR1IE | TMR2IE | SPI1IE | SPI1TXIE | SPI1RXIE |
| 0x04A2 | PIE4 | 7:0 | PWM1IE | PWM1PIE | | | U1IE | U1EIE | U1TXIE | U1RXIE |
| 0x04A3 | PIE5 | 7:0 | PWM2IE | PWM2PIE | TMR3GIE | TMR3IE | | SPI2IE | SPI2TXIE | SPI2RXIE |
| 0x04A4 | PIE6 | 7:0 | DMA2AIE | DMA2ORIE | DMA2DCNTIE | DMA2SCNTIE | NCO1IE | CWG1IE | CLC2IE | INT1IE |
| 0x04A5 | PIE7 | 7:0 | PWM3IE | PWM3PIE | CLC3IE | | I2C1EIE | I2C1IE | I2C1TXIE | I2C1RXIE |
| 0x04A6 | PIE8 | 7:0 | SCANIE | CCP2IE | TMR5GIE | TMR5IE | U2IE | U2EIE | U2TXIE | U2RXIE |
| 0x04A7 | PIE9 | 7:0 | | | CLC4IE | | U3IE | U3EIE | U3TXIE | U3RXIE |
| 0x04A8 | PIE10 | 7:0 | DMA3AIE | DMA3ORIE | DMA3DCNTIE | DMA3SCNTIE | NCO2IE | CWG2IE | CLC5IE | INT2IE |
| 0x04A9 | PIE11 | 7:0 | DMA4AIE | DMA4ORIE | DMA4DCNTIE | DMA4SCNTIE | TMR4IE | CWG3IE | CLC6IE | CCP3IE |
| 0x04AA | PIE12 | 7:0 | DMA5AIE | DMA5ORIE | DMA5DCNTIE | DMA5SCNTIE | U4IE | U4EIE | U4TXIE | U4RXIE |
| 0x04AB | PIE13 | 7:0 | DMA6AIE | DMA6ORIE | DMA6DCNTIE | DMA6SCNTIE | U5IE | U5EIE | U5TXIE | U5RXIE |
| 0x04AC | PIE14 | 7:0 | | | | | NCO3IE | CM2IE | CLC7IE | |
| 0x04AD | PIE15 | 7:0 | | | | | TMR6IE | CRCIE | CLC8IE | NVMIE |
| 0x04AE | PIR0 | 7:0 | IOCIF | | CLC1IF | | CSWIF | OSFIF | HLVDIF | SWIF |
| 0x04AF | PIR1 | 7:0 | SMT1PWAIF | SMT1PRAIF | SMT1IF | CM1IF | ACTIF | ADIF | ZCDIF | INT0IF |
| 0x04B0 | PIR2 | 7:0 | DMA1AIF | DMA1ORIF | DMA1DCNTIF | DMA1SCNTIF | | | | ADTIF |
| 0x04B1 | PIR3 | 7:0 | TMR0IF | CCP1IF | TMR1GIF | TMR1IF | TMR2IF | SPI1IF | SPI1TXIF | SPI1RXIF |
| 0x04B2 | PIR4 | 7:0 | PWM1IF | PWM1PIF | | | U1IF | U1EIF | U1TXIF | U1RXIF |
| 0x04B3 | PIR5 | 7:0 | PWM2IF | PWM2PIF | TMR3GIF | TMR3IF | | SPI2IF | SPI2TXIF | SPI2RXIF |
| 0x04B4 | PIR6 | 7:0 | DMA2AIF | DMA2ORIF | DMA2DCNTIF | DMA2SCNTIF | NCO1IF | CWG1IF | CLC2IF | INT1IF |
| 0x04B5 | PIR7 | 7:0 | PWM3IF | PWM3PIF | CLC3IF | | I2C1EIF | I2C1IF | I2C1TXIF | I2C1RXIF |
| 0x04B6 | PIR8 | 7:0 | SCANIF | CCP2IF | TMR5GIF | TMR5IF | U2IF | U2EIF | U2TXIF | U2RXIF |
| 0x04B7 | PIR9 | 7:0 | | | CLC4IF | | U3IF | U3EIF | U3TXIF | U3RXIF |
| 0x04B8 | PIR10 | 7:0 | DMA3AIF | DMA3ORIF | DMA3DCNTIF | DMA3SCNTIF | NCO2IF | CWG2IF | CLC5IF | INT2IF |
| 0x04B9 | PIR11 | 7:0 | DMA4AIF | DMA4ORIF | DMA4DCNTIF | DMA4SCNTIF | TMR4IF | CWG3IF | CLC6IF | CCP3IF |
| 0x04BA | PIR12 | 7:0 | DMA5AIF | DMA5ORIF | DMA5DCNTIF | DMA5SCNTIF | U4IF | U4EIF | U4TXIF | U4RXIF |
| 0x04BB | PIR13 | 7:0 | DMA6AIF | DMA6ORIF | DMA6DCNTIF | DMA6SCNTIF | U5IF | U5EIF | U5TXIF | U5RXIF |
| 0x04BC | PIR14 | 7:0 | | | | | NCO3IF | CM2IF | CLC7IF | |
| 0x04BD | PIR15 | 7:0 | | | | | TMR6IF | CRCIF | CLC8IF | NVMIF |
| 0x04BE | LATA | 7:0 | LATA7 | LATA6 | LATA5 | LATA4 | LATA3 | LATA2 | LATA1 | LATA0 |
| 0x04BF | LATB | 7:0 | LATB7 | LATB6 | LATB5 | LATB4 | LATB3 | LATB2 | LATB1 | LATB0 |
| 0x04C0 | LATC | 7:0 | LATC7 | LATC6 | LATC5 | LATC4 | LATC3 | LATC2 | LATC1 | LATC0 |
| 0x04C1 | LATD | 7:0 | LATD7 | LATD6 | LATD5 | LATD4 | LATD3 | LATD2 | LATD1 | LATD0 |
| 0x04C2 | LATE | 7:0 | | | | | | LATE2 | LATE1 | LATE0 |
| 0x04C3 | LATF | 7:0 | LATF7 | LATF6 | LATF5 | LATF4 | LATF3 | LATF2 | LATF1 | LATF0 |
| 0x04C4 ... 0x04C5 | Reserved | | | | | | | | | |
| 0x04C6 | TRISA | 7:0 | TRISA7 | TRISA6 | TRISA5 | TRISA4 | TRISA3 | TRISA2 | TRISA1 | TRISA0 |
| 0x04C7 | TRISB | 7:0 | TRISB7 | TRISB6 | TRISB5 | TRISB4 | TRISB3 | TRISB2 | TRISB1 | TRISB0 |
| 0x04C8 | TRISC | 7:0 | TRISC7 | TRISC6 | TRISC5 | TRISC4 | TRISC3 | TRISC2 | TRISC1 | TRISC0 |
| 0x04C9 | TRISD | 7:0 | TRISD7 | TRISD6 | TRISD5 | TRISD4 | TRISD3 | TRISD2 | TRISD1 | TRISD0 |
| 0x04CA | TRISE | 7:0 | | | | | Reserved | TRISE2 | TRISE1 | TRISE0 |
| 0x04CB | TRISF | 7:0 | TRISF7 | TRISF6 | TRISF5 | TRISF4 | TRISF3 | TRISF2 | TRISF1 | TRISF0 |

© 2019-2023 Microchip Technology Inc. and its subsidiaries

.....continued

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x04CC ... 0x04CD | Reserved | | | | | | | | | |
| 0x04CE | PORTA | 7:0 | RA7 | RA6 | RA5 | RA4 | RA3 | RA2 | RA1 | RA0 |
| 0x04CF | PORTB | 7:0 | RB7 | RB6 | RB5 | RB4 | RB3 | RB2 | RB1 | RB0 |
| 0x04D0 | PORTC | 7:0 | RC7 | RC6 | RC5 | RC4 | RC3 | RC2 | RC1 | RC0 |
| 0x04D1 | PORTD | 7:0 | RD7 | RD6 | RD5 | RD4 | RD3 | RD2 | RD1 | RD0 |
| 0x04D2 | PORTE | 7:0 | | | | | RE3 | RE2 | RE1 | RE0 |
| 0x04D3 | PORTF | 7:0 | RF7 | RF6 | RF5 | RF4 | RF3 | RF2 | RF1 | RF0 |
| 0x04D4 ... 0x04D5 | Reserved | | | | | | | | | |
| 0x04D6 | INTCON0 | 7:0 | GIE/GIEH | GIEL | IPEN | | | INT2EDG | INT1EDG | INT0EDG |
| 0x04D7 | INTCON1 | 7:0 | STAT[1:0] | | | | | | | |
| 0x04D8 | STATUS | 7:0 | | $\overline{TO}$ | $\overline{PD}$ | N | OV | Z | DC | C |
| 0x04D9 | FSR2 | 7:0 | FSRL[7:0] | | | | | | | |
| | | 15:8 | | | FSRH[5:0] | | | | | |
| 0x04DB | PLUSW2 | 7:0 | PLUSW[7:0] | | | | | | | |
| 0x04DC | PREINC2 | 7:0 | PREINC[7:0] | | | | | | | |
| 0x04DD | POSTDEC2 | 7:0 | POSTDEC[7:0] | | | | | | | |
| 0x04DE | POSTINC2 | 7:0 | POSTINC[7:0] | | | | | | | |
| 0x04DF | INDF2 | 7:0 | INDF[7:0] | | | | | | | |
| 0x04E0 | BSR | 7:0 | | | BSR[5:0] | | | | | |
| 0x04E1 | FSR1 | 7:0 | FSRL[7:0] | | | | | | | |
| | | 15:8 | | | FSRH[5:0] | | | | | |
| 0x04E3 | PLUSW1 | 7:0 | PLUSW[7:0] | | | | | | | |
| 0x04E4 | PREINC1 | 7:0 | PREINC[7:0] | | | | | | | |
| 0x04E5 | POSTDEC1 | 7:0 | POSTDEC[7:0] | | | | | | | |
| 0x04E6 | POSTINC1 | 7:0 | POSTINC[7:0] | | | | | | | |
| 0x04E7 | INDF1 | 7:0 | INDF[7:0] | | | | | | | |
| 0x04E8 | WREG | 7:0 | WREG[7:0] | | | | | | | |
| 0x04E9 | FSR0 | 7:0 | FSRL[7:0] | | | | | | | |
| | | 15:8 | | | FSRH[5:0] | | | | | |
| 0x04EB | PLUSW0 | 7:0 | PLUSW[7:0] | | | | | | | |
| 0x04EC | PREINC0 | 7:0 | PREINC[7:0] | | | | | | | |
| 0x04ED | POSTDEC0 | 7:0 | POSTDEC[7:0] | | | | | | | |
| 0x04EE | POSTINC0 | 7:0 | POSTINC[7:0] | | | | | | | |
| 0x04EF | INDF0 | 7:0 | INDF[7:0] | | | | | | | |
| 0x04F0 | PCON0 | 7:0 | STKOVF | STKUNF | $\overline{WDTWV}$ | $\overline{RWDT}$ | $\overline{RMCLR}$ | $\overline{RI}$ | $\overline{POR}$ | $\overline{BOR}$ |
| 0x04F1 | PCON1 | 7:0 | | | | | | $\overline{RVREG}$ | $\overline{MEMV}$ | $\overline{RCM}$ |
| 0x04F2 | CPUDOZE | 7:0 | IDLEN | DOZEN | ROI | DOE | | DOZE[2:0] | | |
| 0x04F3 | PROD | 7:0 | PROD[7:0] | | | | | | | |
| | | 15:8 | PROD[15:8] | | | | | | | |
| 0x04F5 | TABLAT | 7:0 | TABLAT[7:0] | | | | | | | |
| 0x04F6 | TBLPTR | 7:0 | TBLPTR[7:0] | | | | | | | |
| | | 15:8 | TBLPTR[15:8] | | | | | | | |
| | | 23:16 | | | TBLPTR21 | TBLPTR[20:16] | | | | |
| 0x04F9 | PCL | 7:0 | PCL[7:0] | | | | | | | |
| 0x04FA | PCLAT | 7:0 | PCLATH[7:0] | | | | | | | |
| | | 15:8 | | | | PCLATU[4:0] | | | | |
| 0x04FC | STKPTR | 7:0 | | STKPTR[6:0] | | | | | | |
| 0x04FD | TOS | 7:0 | TOS[7:0] | | | | | | | |
| | | 15:8 | TOS[15:8] | | | | | | | |
| | | 23:16 | | | | TOS[20:16] | | | | |
| 0x0500 ... 0x2FFFFF | Reserved | | | | | | | | | |
| 0x300000 | CONFIG1 | 7:0 | | RSTOSC[2:0] | | | | FEXTOSC[2:0] | | |
| 0x300001 | CONFIG2 | 7:0 | | | FCMEN | | CSWEN | | PR1WAY | $\overline{CLKOUTEN}$ |

.....continued

| Address | Name | Bit Pos. | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|---|---|---|---|
| 0x300002 | CONFIG3 | 7:0 | BOREN[1:0] | | LPBOREN | IVT1WAY | MVECEN | PWRTS[1:0] | | MCLRE |
| 0x300003 | CONFIG4 | 7:0 | XINST | | LVP | STVREN | PPS1WAY | ZCD | BORV[1:0] | |
| 0x300004 | CONFIG5 | 7:0 | | WDTE[1:0] | | WDTCPS[4:0] | | | | |
| 0x300005 | CONFIG6 | 7:0 | | | WDTCCS[2:0] | | | WDTCWS[2:0] | | |
| 0x300006 | CONFIG7 | 7:0 | | | DEBUG | SAFEN | BBEN | BBSIZE[2:0] | | |
| 0x300007 | CONFIG8 | 7:0 | WRTAPP | | | | WRTSAF | WRTD | WRTC | WRTB |
| 0x300008 | CONFIG9 | 7:0 | | | | | | | | |
| 0x300009 | CONFIG10 | 7:0 | | | | | | | | CP |
| 0x30000A ... 0x3FFFFB | Reserved | | | | | | | | | |
| 0x3FFFFC | REVISIONID | 7:0 | MJRREV[1:0] | | MNRREV[5:0] | | | | | |
| | | 15:8 | 1010[3:0] | | | | MJRREV[5:2] | | | |
| 0x3FFFFE | DEVICEID | 7:0 | DEV[7:0] | | | | | | | |
| | | 15:8 | DEV[15:8] | | | | | | | |

# 47. Electrical Specifications

## 47.1 Absolute Maximum Ratings(†)

| Parameter | | Rating |
|---|---|---|
| **Ambient temperature under bias** | | -40°C to +125°C |
| **Storage temperature** | | -65°C to +150°C |
| **Voltage on pins with respect to $V_{SS}$** | | |
| • **on $V_{DD}$ pin:** | | -0.3V to +6.5V |
| • **on $\overline{MCLR}$ pin:** | | -0.3V to +9.0V |
| • **on all other pins:** | | -0.3V to ($V_{DD}$ + 0.3V) |
| **Maximum current(1)** | | |
| • **on $V_{SS}$ pin** | -40°C ≤ $T_A$ ≤ +85°C | 350 mA |
| | 85°C < $T_A$ ≤ +125°C | 120 mA |
| • **on $V_{DD}$ pin (28-pin devices)** | -40°C ≤ $T_A$ ≤ +85°C | 250 mA |
| | 85°C < $T_A$ ≤ +125°C | 85 mA |
| • **on $V_{DD}$ pin (40-pin devices)** | -40°C ≤ $T_A$ ≤ +85°C | 350 mA |
| | 85°C < $T_A$ ≤ +125°C | 120 mA |
| • **on any standard I/O pin** | | ±50 mA |
| **Clamp current, $I_K$ ($V_{PIN}$ < 0 or $V_{PIN}$ > $V_{DD}$)** | | ±20 mA |
| **Total power dissipation(2)** | | 800 mW |

**Notes:**

1. Maximum current rating requires even load distribution across I/O pins. Maximum current rating may be limited by the device package power dissipation characterizations, see the Thermal Characteristics section to calculate device specifications.
2. Power dissipation is calculated as follows:
   $P_{DIS} = V_{DD} \times \{I_{DD} - \Sigma I_{OH}\} + \Sigma \{(V_{DD} - V_{OH}) \times I_{OH}\} + \Sigma (V_{OI} \times I_{OL})$.

† NOTICE: Stresses above those listed under **"Absolute Maximum Ratings"** may cause permanent damage to the device. This is a stress rating only and functional operation of the device at those or any other conditions above those indicated in the operation listings of this specification is not implied. Exposure above maximum rating conditions for extended periods may affect device reliability.

## 47.2 Standard Operating Conditions

The standard operating conditions for any device are defined as:

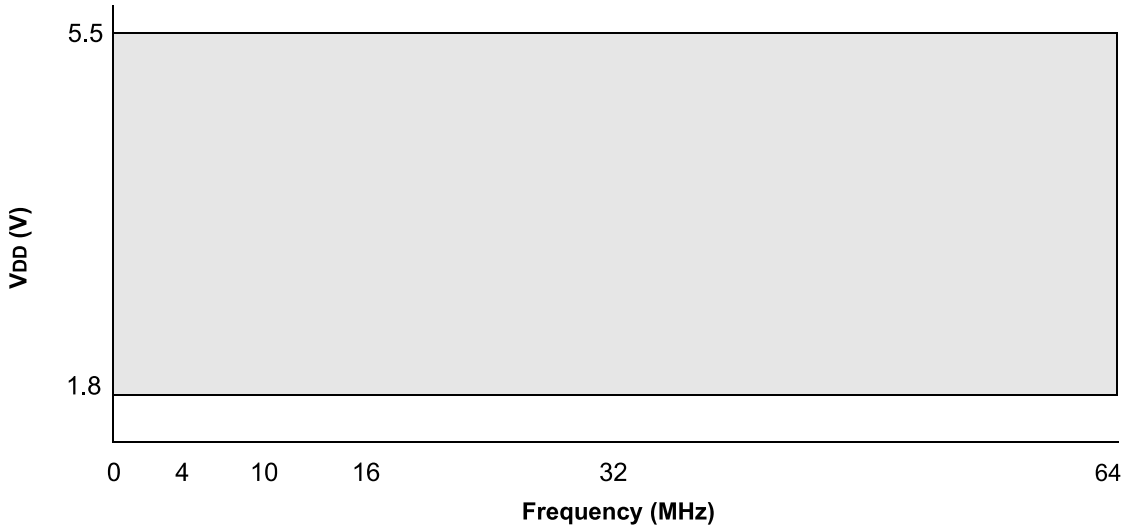
| | |
|---|---|
| **Operating Voltage:** | $V_{DDMIN} \le V_{DD} \le V_{DDMAX}$ |
| **Operating Temperature:** | $T_{A_MIN} \le T_A \le T_{A_MAX}$ |

| Parameter | | Ratings |
|---|---|---|
| **$V_{DD}$ — Operating Supply Voltage(1)** | $V_{DDMIN}$ | +1.8V |
| | $V_{DDMAX}$ | +5.5V |
| **$T_A$ — Operating Ambient Temperature Range** | | |
| Industrial Temperature | $T_{A_MIN}$ | -40°C |
| | $T_{A_MAX}$ | +85°C |
| Extended Temperature | $T_{A_MIN}$ | -40°C |
| | $T_{A_MAX}$ | +125°C |

**Note:**

1. See the Parameter Supply Voltage in the **"DC Characteristics"** chapter for more details.

**Figure 47-1. Voltage Frequency Graph, -40°C ≤ $T_A$ ≤ +125°C**




**Notes:**

- The shaded region indicates the permissible combinations of voltage and frequency.
- Refer to the **"External Clock/Oscillator Timing Requirements"** table in the **"AC Characteristics"** chapter for each Oscillator mode's supported frequencies.

## 47.3 DC Characteristics

### 47.3.1 Supply Voltage

**Table 47-1. Supply Voltage**

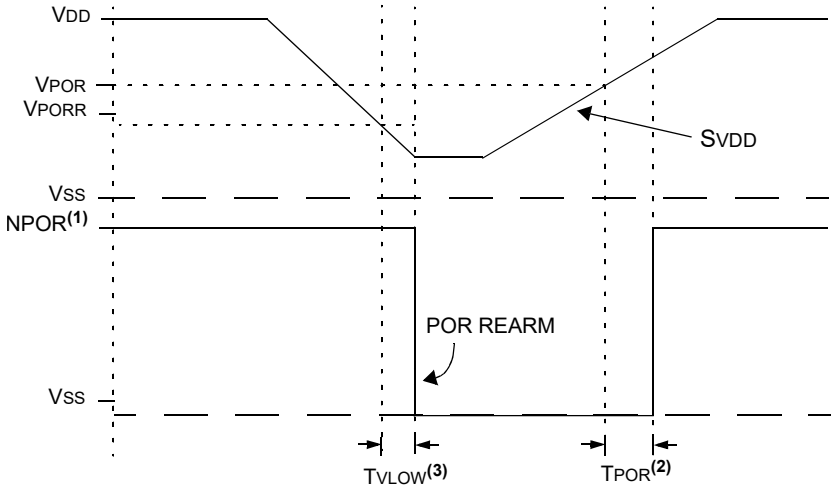
| Standard Operating Conditions (unless otherwise stated) | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Param. No.** | **Sym.** | **Characteristic** | **Min.** | **Typ.†** | **Max.** | **Units** | **Conditions** |
| **Supply Voltage** | | | | | | | |
| D002 | $V_{DD}$ | | 1.8 | — | 5.5 | V | |
| **RAM Data Retention(1)** | | | | | | | |
| D003 | $V_{DR}$ | | 1.7 | — | — | V | Device in Sleep mode |
| **Power-on Reset Release Voltage(2)** | | | | | | | |
| D004 | $V_{POR}$ | | — | 1.6 | — | V | BOR and LPBOR disabled(3) |
| **Power-on Reset Rearm Voltage(2)** | | | | | | | |
| D005 | $V_{PORR}$ | | — | 1 | — | V | BOR and LPBOR disabled(3) |
| **$V_{DD}$ Rise Rate to ensure internal Power-on Reset signal(2)** | | | | | | | |
| D006 | $S_{VDD}$ | | 0.05 | — | — | V/ms | BOR and LPBOR disabled(3) |

† Data in “Typ.” column is at 3.0V, 25℃ unless otherwise stated. These parameters are for design guidance only and are not tested.

**Notes:**

1. This is the limit to which $V_{DD}$ can be lowered in Sleep mode without losing RAM data.
2. See the following figure, POR and POR REARM with Slow Rising $V_{DD}$.
3. See Reset, WDT, Oscillator Start-up Timer, Brown-Out Reset and Low-Power Brown-Out Reset Specifications for BOR and LPBOR trip point information.

**Figure 47-2. POR and POR Rearm with Slow Rising $V_{DD}$**




**Notes:**

1. When NPOR is low, the device is held in Reset.
2. $T_{POR}$ 1 µs typical.
3. $T_{VLOW}$ 2.7 µs typical.

### 47.3.2 Supply Current ($I_{DD}$)(1,2,4)

**Table 47-2. Supply Current**

| Standard Operating Conditions (unless otherwise stated) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Param. No. | Sym. | Device Characteristics | Min. | Typ.† | Max. | Units | Conditions | |
| | | | | | | | $V_{DD}$ | Note |
| D100 | $I_{DD_{XT4}}$ | XT = 4 MHz | — | 640 | 870 | µA | 3.0V | |
| D100A | $I_{DD_{XT4}}$ | XT = 4 MHz | — | 490 | 700 | µA | 3.0V | PMD's all 1's |
| D101 | $I_{DD_{HFO16}}$ | HFINTOSC = 16 MHz | — | 2 | 2.5 | mA | 3.0V | |
| D101A | $I_{DD_{HFO16}}$ | HFINTOSC = 16 MHz | — | 1.5 | 1.9 | mA | 3.0V | PMD's all 1's |
| D102 | $I_{DD_{HFO64}}$ | HFINTOSC = 64 MHz | — | 6.7 | 8.2 | mA | 3.0V | |
| D102A | $I_{DD_{HFO64}}$ | HFINTOSC = 64 MHz | — | 4.5 | 5.4 | mA | 3.0V | PMD's all 1's |
| D103 | $I_{DD_{HSPLL64}}$ | HS+PLL = 64 MHz | — | 5.6 | 13.8 | mA | 3.0V | |
| D103A | $I_{DD_{HSPLL64}}$ | HS+PLL = 64 MHz | — | 3.8 | 11.5 | mA | 3.0V | PMD's all 1's |
| D104 | $I_{DD_{IDLE}}$ | Idle mode, HFINTOSC = 16 MHz | — | 1.4 | 1.8 | mA | 3.0V | |
| D105 | $I_{DD_{DOZE}}$(3) | Doze mode, HFINTOSC = 16 MHz, Doze Ratio = 16 | — | 1.5 | 1.9 | mA | 3.0V | |

**.....continued**

| Standard Operating Conditions (unless otherwise stated) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Param. No. | Sym. | Device Characteristics | Min. | Typ.† | Max. | Units | Conditions | |
| | | | | | | | $V_{DD}$ | Note |

† Data in “Typ.” column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Notes:**

1. The test conditions for all $I_{DD}$ measurements in Active Operation mode are: OSC1 = external square wave, from rail-to-rail; all I/O pins are outputs driven low; $\overline{MCLR}$ = $V_{DD}$; WDT disabled.
2. The supply current is mainly a function of the operating voltage and frequency. Other factors, such as I/O pin loading and switching rate, oscillator type, internal code execution pattern and temperature, also have an impact on the current consumption.
3. $I_{DD_{DOZE}} = [I_{DD_{IDLE}}*(N-1)/N] + I_{DD_{HFO}}16/N$ where N = Doze Ratio (see **CPUDOZE** register).
4. PMD bits are all in the Default state, no modules are disabled.

### 47.3.3 Power-Down Current ($I_{PD}$)(1,2)

**Table 47-3. Power Down Current**

| Standard Operating Conditions (unless otherwise stated) | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Param. No. | Sym. | Device Characteristics | Min. | Typ.† | Max. +85°C | Max. +125°C | Units | Conditions | | |
| | | | | | | | | $V_{DD}$ | VREGPM | Note |
| D200 | $I_{PD}$ | $I_{PD}$ Base | — | 1.1 | 3.3 | 4.6 | µA | 3.0V | 'b11 | |
| | | | — | 0.9 | 12.1 | 43 | µA | 3.0V | 'b10 | |
| | | | — | 29.5 | 45.5 | 68.9 | µA | 3.0V | 'b01 | |
| | | | — | 152 | 210 | 225 | µA | 3.0V | 'b00 | |
| D201 | $I_{PD_WDT}$ | Low-Frequency Internal Oscillator/WDT | — | 1.5 | 3.8 | 5.1 | µA | 3.0V | 'b11 | |
| D202 | $I_{PD_SOSC}$ | Secondary Oscillator ($S_{OSC}$) | — | 2.1 | 4.6 | 7.9 | µA | 3.0V | 'b11 | |
| D203 | $I_{PD_LPBOR}$ | Low-Power Brown-out Reset (LPBOR) | — | 1.2 | 4 | 6 | µA | 3.0V | 'b11 | |
| D204 | $I_{PD_FVR_BUF1}$ | FVR Buffer 1 (ADC) | — | 144.7 | 220 | 225 | µA | 3.0V | 'b11 | |
| D204A | $I_{PD_FVR_BUF2}$ | FVR Buffer 2 (DAC/CMP) | — | 49.4 | 74.2 | 93 | µA | 3.0V | 'bx1 or 'b10 | |
| D205 | $I_{PD_BOR}$ | Brown-out Reset (BOR) | — | 16.6 | 21 | 21.4 | µA | 3.0V | 'b11 | |
| D206 | $I_{PD_HLVD}$ | High/Low Voltage Detect (HLVD) | — | 13.9 | 20.8 | 22.5 | µA | 3.0V | 'b11 | |
| D207 | $I_{PD_ADCA}$ | ADC - Active | — | 483 | 789 | 795 | µA | 3.0V | 'bx1 or 'b10 | ADC is converting **(Note 4)** |

**.....continued**

| Standard Operating Conditions (unless otherwise stated) | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Param. No. | Sym. | Device Characteristics | Min. | Typ.† | Max. +85°C | Max. +125°C | Units | Conditions | | |
| | | | | | | | | $V_{DD}$ | VREGPM | Note |
| D208 | $I_{PD_CMP}$ | Comparator | — | 52.5 | 95 | 105 | µA | 3.0V | 'b11 | |

* These parameters are characterized but not tested.

† Data in "Typ." column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Notes:**

1. The peripheral current is the sum of the base $I_{DD}$ and the additional current consumed when this peripheral is enabled. The peripheral Δ current can be determined by subtracting the base $I_{DD}$ or $I_{PD}$current from this limit. Max. values will be used when calculating total current consumption.
2. The power-down current in Sleep mode does not depend on the oscillator type. Power-down current is measured with the part in Sleep mode with all I/O pins in high-impedance state and tied to $V_{SS}$.
3. All peripheral currents listed are on a per-peripheral basis if more than one instance of a peripheral is available.
4. ADC clock source is ADCRC.

### 47.3.4 I/O Ports

**Table 47-4. I/O Ports**

| Standard Operating Conditions (unless otherwise stated) | | | | | | | |
|---|---|---|---|---|---|---|---|
| Param. No. | Sym. | Device Characteristics | Min. | Typ.† | Max. | Units | Conditions |
| **Input Low Voltage** | | | | | | | |
| | $V_{IL}$ | I/O PORT: | | | | | |
| D300 | | • with TTL buffer | — | — | 0.8 | V | $4.5V \le V_{DD} \le 5.5V$ |
| D301 | | | — | — | $0.15\ V_{DD}$ | V | $1.8V \le V_{DD} < 4.5V$ |
| D302 | | • with Schmitt Trigger buffer | — | — | $0.2\ V_{DD}$ | V | $2.0V \le V_{DD} \le 5.5V$ |
| D303 | | • with $I^2C$ levels | — | — | $0.3\ V_{DD}$ | V | $2.0V \le V_{DD} \le 5.5V$ |
| D304 | | • with SMBus 2.0 | — | — | 0.8 | V | $2.7V \le V_{DD} \le 5.5V$ |
| D305 | | • with SMBus 3.0 | — | — | 0.8 | V | |
| D306 | | $\overline{MCLR}$ | — | — | $0.2\ V_{DD}$ | V | |
| **High Low Voltage** | | | | | | | |

| .....continued | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Standard Operating Conditions (unless otherwise stated)** | | | | | | | |
| **Param. No.** | **Sym.** | **Device Characteristics** | **Min.** | **Typ.†** | **Max.** | **Units** | **Conditions** |
| | $V_{IH}$ | I/O PORT: | | | | | |
| D320 | | • with TTL buffer | 2.0 | — | — | V | $4.5V \le V_{DD} \le 5.5V$ |
| D321 | | | 0.25 $V_{DD}$ + 0.8 | — | — | V | $1.8V \le V_{DD} < 4.5V$ |
| D322 | | • with Schmitt Trigger buffer | 0.8 $V_{DD}$ | — | — | V | $2.0V \le V_{DD} \le 5.5V$ |
| D323 | | • with $I^2C$ levels | 0.7 $V_{DD}$ | — | — | V | |
| D324 | | • with SMBus 2.0 | 2.1 | — | — | V | $2.7V \le V_{DD} \le 5.5V$ |
| D325 | | • with SMBus 3.0 | 1.35 | — | — | V | 25°C $\le T_A \le$ +125°C $1.8V \le V_{DD} \le 5.5V$ |
| | | | 1.45 | — | — | V | -40°C $\le T_A \le$ +25°C $1.8V \le V_{DD} \le 5.5V$ |
| D326 | | $\overline{MCLR}$ | 0.7 $V_{DD}$ | — | — | V | |
| **Input Leakage Current(1)** | | | | | | | |
| D340 | $I_{IL}$ | I/O PORTS | — | ±5 | ±125 | nA | $V_{SS} \le V_{PIN} \le V_{DD}$, Pin at high-impedance, 85°C |
| D341 | | | — | ±5 | ±1000 | nA | $V_{SS} \le V_{PIN} \le V_{DD}$, Pin at high-impedance, 125°C |
| D342 | | $\overline{MCLR}$(2) | — | ±50 | ±200 | nA | $V_{SS} \le V_{PIN} \le V_{DD}$, Pin at high-impedance, 85°C |
| **Weak Pull-up Current** | | | | | | | |
| D350 | $I_{PUR}$ | | 80 | 140 | 200 | µA | $V_{DD}$ = 3.0V, $V_{PIN} = V_{SS}$ |
| **Output Low Voltage** | | | | | | | |
| D360 | $V_{OL}$ | I/O PORTS | — | — | 0.6 | V | $I_{OL}$ = 10.0 mA, $V_{PIN}$ = 3.0V |
| **Output High Voltage** | | | | | | | |
| D370 | $V_{OH}$ | I/O PORTS | $V_{DD}$ - 0.7 | — | — | V | $I_{OH}$ = 6.0 mA, $V_{PIN}$ = 3.0V |
| **All I/O Pins** | | | | | | | |
| D380 | $C_{IO}$ | | — | 5 | 50 | pF | |

† Data in “Typ” column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Notes:**

1. Negative current is defined as current sourced by the pin.
2. The leakage current on the $\overline{MCLR}$ pin is strongly dependent on the applied voltage level. The specified levels represent normal operating conditions. Higher leakage current may be measured at different input voltages.

### 47.3.5 Memory Programming Specifications

**Table 47-5. Memory Programming**

| Standard Operating Conditions (unless otherwise stated) | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Param No.** | **Sym.** | **Device Characteristics** | **Min.** | **Typ†** | **Max.** | **Units** | **Conditions** |
| **Data EEPROM Memory Specifications** | | | | | | | |
| MEM20 | $E_D$ | DataEE Byte Endurance | 100k | — | — | E/W | -40°C ≤ $T_A$ ≤ +85°C |
| MEM21 | $T_{D_RET}$ | Characteristic Retention | — | 40 | — | Year | Provided no other specifications are violated |
| MEM22 | $N_{D_REF}$ | Total Erase/Write Cycles before Refresh | 1M | 4M | — | E/W | -40°C ≤ $T_A$ ≤ +85°C |
| MEM23 | $V_{D_RW}$ | $V_{DD}$ for Read or Erase/Write operation | $V_{DDMIN}$ | — | $V_{DDMAX}$ | V | |
| MEM24 | $T_{D_BEW}$ | Byte Erase and Write Cycle Time | — | — | 11 | ms | |
| **Program Flash Memory Specifications** | | | | | | | |
| MEM30 | $E_P$ | Flash Memory Cell Endurance | 1k | — | — | E/W | -40°C ≤ $T_A$ ≤ +85°C **(Note 1)** |
| MEM32 | $T_{P_RET}$ | Characteristic Retention | — | 40 | — | Year | Provided no other specifications are violated |
| MEM33 | $V_{P_RD}$ | $V_{DD}$ for Read operation | $V_{DDMIN}$ | — | $V_{DDMAX}$ | V | |
| MEM34 | $V_{P_REW}$ | $V_{DD}$ for Row Erase or Write operation | $V_{DDMIN}$ | — | $V_{DDMAX}$ | V | |
| MEM35 | $T_{P_REW}$ | Self-Timed Page Write | — | — | 10 | ms | |
| MEM36 | $T_{SE}$ | Self-Timed Page Erase | — | — | 11 | ms | |
| MEM37 | $T_{P_WRD}$ | Self-Timed Word Write | — | — | 75 | µs | |

† Data in “Typ” column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Note:**

1. Flash Memory Cell Endurance for the Flash memory is defined as: One Row Erase operation and one Self-Timed Write.

### 47.3.6 Thermal Characteristics

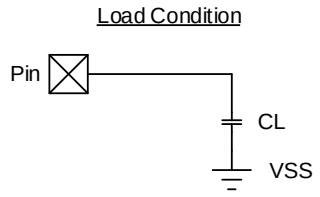
**Table 47-6. Thermal Considerations**

| Standard Operating Conditions (unless otherwise stated) | | | | | |
|---|---|---|---|---|---|
| **Param No.** | **Sym.** | **Characteristic** | **Typ.** | **Units** | **Conditions** |
| TH01 | $\theta_{JA}$ | Thermal Resistance Junction to Ambient | 60 | °C/W | 28-pin SPDIP package |
| | | | 80 | °C/W | 28-pin SOIC package |
| | | | 90 | °C/W | 28-pin SSOP package |
| | | | 27.5 | °C/W | 28-pin VQFN 4x4 mm package |
| | | | 47.2 | °C/W | 40-pin PDIP package |
| | | | 29 | °C/W | 40-pin QFN package |
| | | | 46 | °C/W | 44-pin TQFP package |
| | | | 62.8 | °C/W | 48-pin TQFP package |
| | | | 24.8 | °C/W | 48-pin VQFN package |
| TH02 | $T_{JMAX}$ | Maximum Junction Temperature | 150 | °C | |

**Note:**
1. See “Absolute Maximum Ratings” for total power dissipation.

## 47.4 AC Characteristics

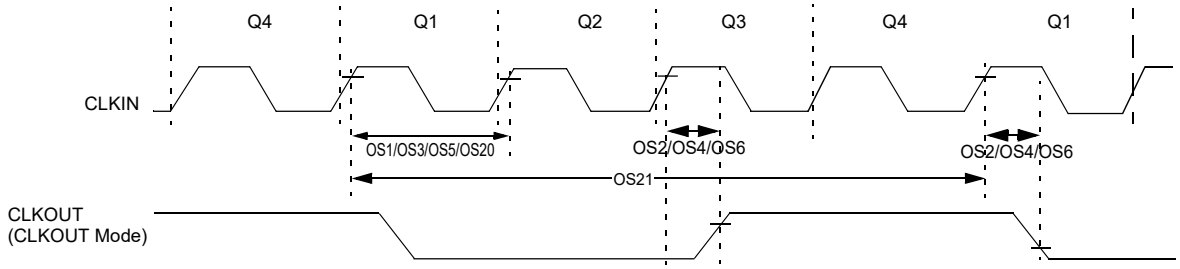
**Figure 47-3. Load Conditions**




**Legend:** CL = 50 pF for all pins

### 47.4.1 External Clock/Oscillator Timing Requirements

**Figure 47-4. Clock Timing**




**Note:** See the table below.

**Table 47-7. External Clock Oscillator**

| Standard Operating Conditions (unless otherwise stated) | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Param No.** | **Sym.** | **Characteristic** | **Min.** | **Typ. †** | **Max.** | **Units** | **Conditions** |
| **ECL Oscillator** | | | | | | | |
| OS1 | $F_{ECL}$ | Clock Frequency | — | — | 1 | MHz | |
| OS2 | $T_{ECL_DC}$ | Clock Duty Cycle | 40 | — | 60 | % | |
| **ECM Oscillator** | | | | | | | |
| OS3 | $F_{ECM}$ | Clock Frequency | — | — | 16 | MHz | |
| OS4 | $T_{ECM_DC}$ | Clock Duty Cycle | 40 | — | 60 | % | |
| **ECH Oscillator** | | | | | | | |
| OS5 | $F_{ECH}$ | Clock Frequency | — | — | 64 | MHz | $V_{DD} \geq 2.7V$ |
| | | | — | — | 32 | MHz | $V_{DD} < 2.7V$ |
| OS6 | $T_{ECH_DC}$ | Clock Duty Cycle | 40 | — | 60 | % | |
| **LP Oscillator** | | | | | | | |
| OS7 | $F_{LP}$ | Clock Frequency | — | — | 100 | kHz | **(Note 4)** |
| **XT Oscillator** | | | | | | | |
| OS8 | $F_{XT}$ | Clock Frequency | — | — | 4 | MHz | **(Note 4)** |
| **HS Oscillator** | | | | | | | |

| .....continued | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Standard Operating Conditions (unless otherwise stated)** | | | | | | | |
| **Param No.** | **Sym.** | **Characteristic** | **Min.** | **Typ. †** | **Max.** | **Units** | **Conditions** |
| OS9 | $F_{HS}$ | Clock Frequency | — | — | 20 | MHz | $V_{DD}$ > 2.5V **(Note 4)** |
| **Secondary Oscillator** | | | | | | | |
| OS10 | $F_{SEC}$ | Clock Frequency | 32.4 | 32.768 | 33.1 | kHz | **(Note 4)** |
| **System Oscillator** | | | | | | | |
| OS20 | $F_{OSC}$ | System Clock Frequency | — | — | 64 | MHz | **(Note 2, Note 3)** |
| OS21 | $F_{CY}$ | Instruction Frequency | — | $F_{OSC}/4$ | — | MHz | |
| OS22 | $T_{CY}$ | Instruction Period | 62.5 | $1/F_{CY}$ | — | ns | |

**Notes:**

1. Instruction cycle period (TCY) equals four times the input oscillator time base period. All specified values are based on characterization data for that particular oscillator type under standard operating conditions with the device executing code. Exceeding these specified limits may result in an unstable oscillator operation and/or higher than expected current consumption. All devices are tested to operate at “min” values with an external clock applied to OSC1 pin. When an external clock input is used, the “max” cycle time limit is “DC” (no clock) for all devices.
2. The system clock frequency ($F_{OSC}$) is selected by the “main clock switch controls” as described in the **“Power Saving Operation Modes”** section.
3. The system clock frequency ($F_{OSC}$) must meet the voltage requirements defined in the **“Standard Operating Conditions”** section.
4. LP, XT and HS oscillator modes require an appropriate crystal or resonator to be connected to the device. For clocking the device with the external square wave, one of the EC mode selections must be used.

### 47.4.2 Internal Oscillator Parameters(1)

**Table 47-8. Internal Oscillator**

| **Standard Operating Conditions (unless otherwise stated)** | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Param No.** | **Sym.** | **Characteristic** | **Min.** | **Typ. †** | **Max.** | **Units** | **Conditions** |
| OS50 | $F_{HFOSC}$ | Precision Calibrated HFINTOSC Frequency | — | 4<br>8<br>12<br>16<br>32<br>48<br>64 | — | MHz | (**Note 2**) |
| OS51 | $F_{HFOSCLP}$ | Low-Power Optimized HFINTOSC Frequency | 0.92<br>1.84<br>0.88<br>1.76 | 1<br>2<br>1<br>2 | 1.08<br>2.16<br>1.12<br>2.24 | MHz<br>MHz<br>MHz<br>MHz | -40°C ≤ $T_A$ ≤ 85°C<br>-40°C ≤ $T_A$ ≤ 85°C<br>85°C ≤ $T_A$ ≤ 125°C<br>85°C ≤ $T_A$ ≤ 125°C |

.....continued

| Standard Operating Conditions (unless otherwise stated) | | | | | | | |
|---|---|---|---|---|---|---|---|
| Param No. | Sym. | Characteristic | Min. | Typ. † | Max. | Units | Conditions |
| OS52 | $F_{MFOSC}$ | Internal Calibrated MFINTOSC Frequency | — | 500 | — | kHz | |
| OS53 | $F_{LFOSC}$ | Internal LFINTOSC Frequency | 24.8 | 31 | 37.2 | kHz | |
| OS54 | $T_{HFOSCST}$ | HFINTOSC Wake-up from Sleep Start-up Time | — | 13 | 20 | µs | VREGPM = 00<br>System Clock at 4 MHz<br>$V_{DD}$ = 3.0V |
| | | | — | 30 | 48 | µs | VREGPM = 01<br>System Clock at 4 MHz<br>$V_{DD}$ = 3.0V |
| | | | — | 115 | 210 | µs | VREGPM = 10<br>System Clock at 4 MHz<br>$V_{DD}$ = 3.0V |
| | | | — | 120 | 220 | µs | VREGPM = 11<br>System Clock at 4 MHz<br>$V_{DD}$ = 3.0V |
| OS56 | $T_{LFOSCST}$ | LFINTOSC Wake-up from Sleep Start-up Time | — | 292 | 420 | µs | 25ºC ≤ $T_A$ ≤ 125ºC<br>$V_{DD}$ = 3.0V<br>VREGPM = xx |

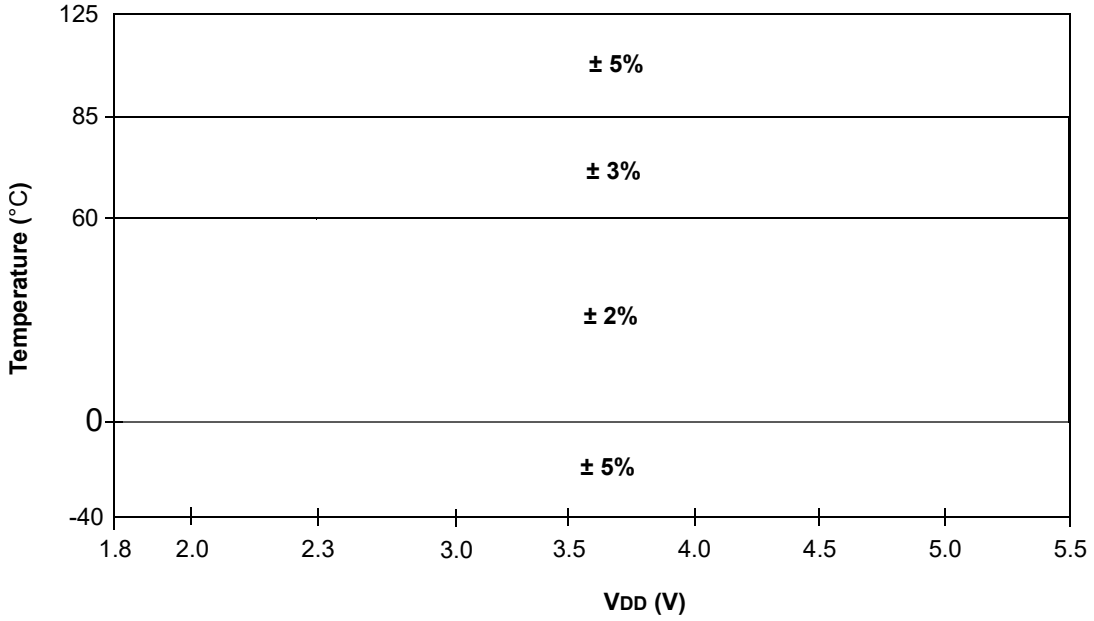
* These parameters are characterized but not tested.

† Data in “Typ” column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Notes:**

1. To ensure these oscillator frequency tolerances, $V_{DD}$ and $V_{SS}$ must be capacitively decoupled as close to the device as possible. 0.1 µF and 0.01 µF values in parallel are recommended.
2. See the figure below.

**Figure 47-5. Precision Calibrated HFINTOSC Frequency Accuracy Over Device $V_{DD}$ and Temperature**




### 47.4.3 PLL Specifications

**Table 47-9. PLL Specifications**

| Standard Operating Conditions (unless otherwise stated) | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Param No.** | **Sym.** | **Characteristic** | **Min.** | **Typ. †** | **Max.** | **Units** | **Conditions** |
| PLL01 | $F_{PLLIN}$ | PLL Input Frequency Range | 4 | — | 16 | MHz | |
| PLL02 | $F_{PLLOUT}$ | PLL Output Frequency Range | 16 | — | 64 | MHz | (**Note 1**) |
| PLL03* | $F_{PLLST}$ | PLL Lock Time | — | 200 | — | µs | |
| PLL04* | $F_{PLLJIT}$ | PLL Output Frequency Stability | -0.25 | — | 0.25 | % | |

* These parameters are characterized but not tested.

† Data in “Typ” column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Note:**

1. The output frequency of the PLL must meet the $F_{OSC}$ requirements listed in Parameter D002.

### 47.4.4 I/O and CLKOUT Timing Specifications

**Figure 47-6. CLKOUT and I/O Timing**

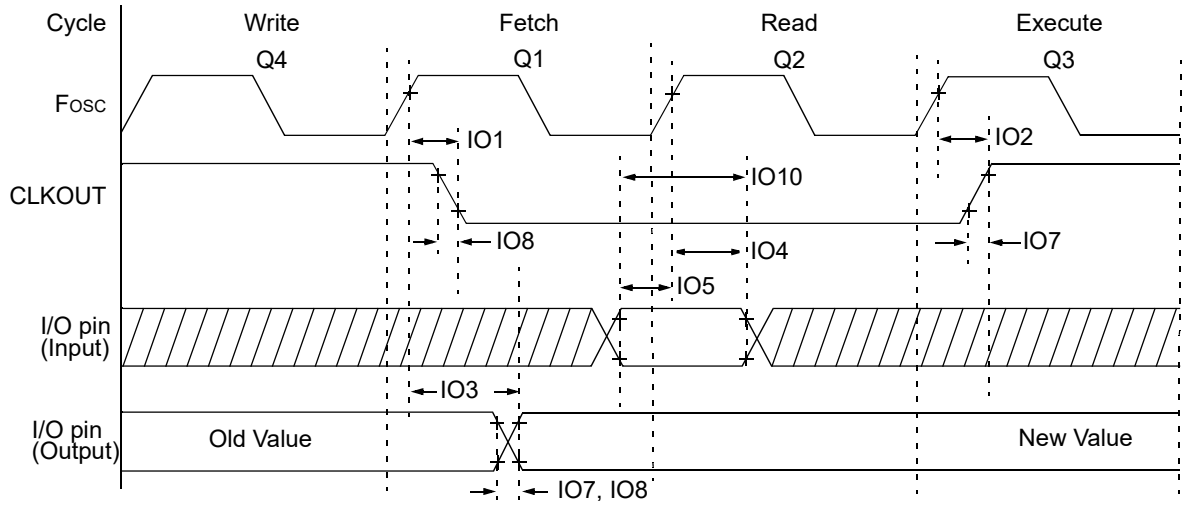


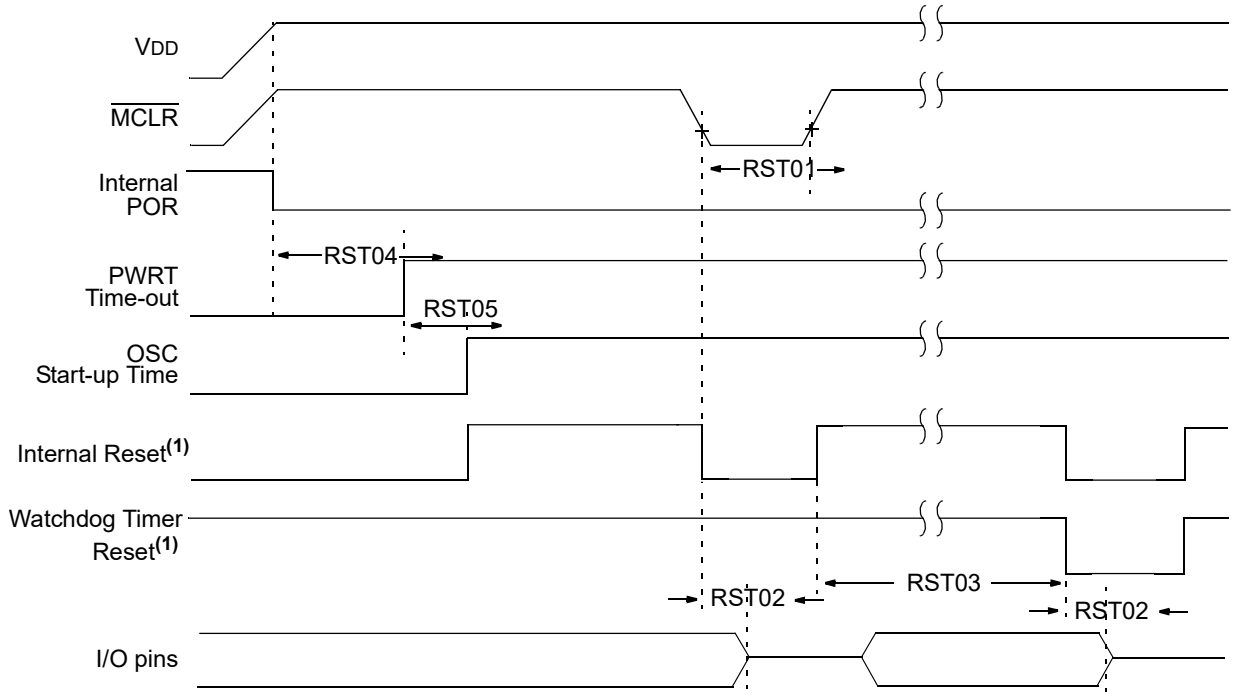

**Table 47-10. I/O and CLKOUT Timing Specifications**

| Standard Operating Conditions (unless otherwise stated) | | | | | | | |
|---|---|---|---|---|---|---|---|
| Param No. | Sym. | Characteristic | Min. | Typ. † | Max. | Units | Conditions |
| IO1* | $T_{CLKOUTH}$ | CLKOUT rising edge delay (rising edge $F_{OSC}$ (Q1 cycle) to falling edge CLKOUT | — | — | 70 | ns | |
| IO2* | $T_{CLKOUTL}$ | CLKOUT falling edge delay (rising edge $F_{OSC}$ (Q3 cycle) to rising edge CLKOUT | — | — | 72 | ns | |
| IO3* | $T_{IO_VALID}$ | Port output valid time (rising edge $F_{OSC}$ (Q1 cycle) to port valid) | — | 50 | 70 | ns | |
| IO4* | $T_{IO_SETUP}$ | Port input setup time (Setup time before rising edge $F_{OSC}$ – Q2 cycle) | 20 | — | — | ns | |
| IO5* | $T_{IO_HOLD}$ | Port input hold time (Hold time after rising edge $F_{OSC}$ – Q2 cycle) | 50 | — | — | ns | |
| IO6* | $T_{IOR_SLREN}$ | Port I/O rise time, slew rate enabled | — | 25 | — | ns | $V_{DD}$ = 3.0V |
| IO7* | $T_{IOR_SLRDIS}$ | Port I/O rise time, slew rate disabled | — | 5 | — | ns | $V_{DD}$ = 3.0V |
| IO8* | $T_{IOF_SLREN}$ | Port I/O fall time, slew rate enabled | — | 25 | — | ns | $V_{DD}$ = 3.0V |
| IO9* | $T_{IOF_SLRDIS}$ | Port I/O fall time, slew rate disabled | — | 5 | — | ns | $V_{DD}$ = 3.0V |
| IO10* | $T_{INT}$ | INT pin high or low time to trigger an interrupt | 25 | — | — | ns | |
| IO11* | $T_{IOC}$ | Interrupt-on-Change minimum high or low time to trigger interrupt | 25 | — | — | ns | |

* These parameters are characterized but not tested.

### 47.4.5 Reset, WDT, Oscillator Start-Up Timer, Power-Up Timer, Brown-Out Reset and Low-Power Brown-Out Reset Specifications

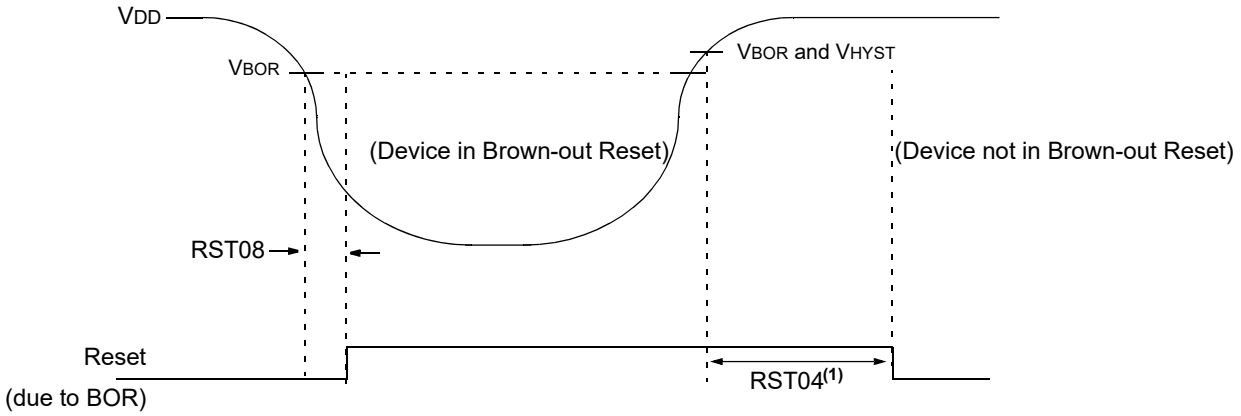
**Figure 47-7. Reset, Watchdog Timer, Oscillator Start-Up Timer and Power-Up Timer Timing**




**Note:**

1. Asserted low.

**Figure 47-8. Brown-out Reset Timing and Characteristics**




**Note:**

1. Only if the $\overline{\text{PWRTE}}$ Configuration bit is programmed to '1'; 2 ms delay if $\overline{\text{PWRTE}}$ = 0.

**Table 47-11. Resets**

| Standard Operating Conditions (unless otherwise stated) | | | | | | | |
|---|---|---|---|---|---|---|---|
| Param No. | Sym. | Characteristic | Min. | Typ. † | Max. | Units | Conditions |
| RST01* | $T_{MCLR}$ | $\overline{\text{MCLR}}$ Pulse Width Low to ensure Reset | — | — | — | µs | |
| RST02* | $T_{IOZ}$ | I/O high-impedance from Reset detection | — | — | 2 | µs | |

| .....continued | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Standard Operating Conditions (unless otherwise stated)** | | | | | | | |
| **Param No.** | **Sym.** | **Characteristic** | **Min.** | **Typ. †** | **Max.** | **Units** | **Conditions** |
| RST03 | $T_{WDT}$ | Watchdog Timer Time-out Period | — | 16 | — | ms | WDTCPS = 00100 |
| RST04* | $T_{PWRT}$ | Power-up Timer Period | — | 65 | — | ms | |
| RST05 | $T_{OST}$ | Oscillator Start-up Timer Period(1,2) | — | 1024 | — | $T_{OSC}$ | |
| RST06 | $V_{BOR}$ | Brown-out Reset Voltage | 2.7 | 2.85 | 3.0 | V | BORV = 00 |
| | | | 2.55 | 2.7 | 2.85 | V | BORV = 01 |
| | | | 2.3 | 2.45 | 2.6 | V | BORV = 10 |
| | | | 1.8 | 1.9 | 2.1 | V | BORV = 11 |
| RST07 | $V_{BORHYS}$ | Brown-out Reset Hysteresis | — | 60 | — | mV | BORV = 00 |
| RST08 | $T_{BORDC}$ | Brown-out Reset Response Time | — | 3 | — | µs | |
| RST09 | $V_{LPBOR}$ | Low-Power Brown-out Reset Voltage | 1.8 | 1.9 | 2.2 | V | |

* These parameters are characterized but not tested.

† Data in “Typ” column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Notes:**

1. By design, the Oscillator Start-up Timer (OST) counts the first 1024 cycles, independent of frequency.
2. To ensure these voltage tolerances, $V_{DD}$ and $V_{SS}$ must be capacitively decoupled as close to the device as possible. 0.1 µF and 0.01 µF values in parallel are recommended.

### 47.4.6 High/Low-Voltage Detect Characteristics

**Table 47-12. High/Low-Voltage Detect**

| **Standard Operating Conditions (unless otherwise stated)** | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Param No.** | **Sym.** | **Characteristic** | **Min.** | **Typ.** | **Max.** | **Units** | **Conditions** |
| HLVD01 | $V_{DET}$ | Voltage Detect | 1.73(1) | 1.90 | 2.07 | V | HLVDSEL = 0000 |
| | | | 1.91 | 2.10 | 2.29 | V | HLVDSEL = 0001 |
| | | | 2.05 | 2.25 | 2.45 | V | HLVDSEL = 0010 |
| | | | 2.28 | 2.50 | 2.73 | V | HLVDSEL = 0011 |
| | | | 2.37 | 2.60 | 2.83 | V | HLVDSEL = 0100 |
| | | | 2.5 | 2.75 | 3.00 | V | HLVDSEL = 0101 |
| | | | 2.64 | 2.90 | 3.16 | V | HLVDSEL = 0110 |
| | | | 2.87 | 3.15 | 3.43 | V | HLVDSEL = 0111 |
| | | | 3.05 | 3.35 | 3.65 | V | HLVDSEL = 1000 |
| | | | 3.28 | 3.60 | 3.92 | V | HLVDSEL = 1001 |
| | | | 3.41 | 3.75 | 4.09 | V | HLVDSEL = 1010 |
| | | | 3.64 | 4.00 | 4.36 | V | HLVDSEL = 1011 |
| | | | 3.82 | 4.20 | 4.58 | V | HLVDSEL = 1100 |
| | | | 3.96 | 4.35 | 4.74 | V | HLVDSEL = 1101 |
| | | | 4.23 | 4.65 | 5.07 | V | HLVDSEL = 1110 |

© 2019-2023 Microchip Technology Inc. and its subsidiaries

| .....continued | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Standard Operating Conditions (unless otherwise stated)** | | | | | | | |
| **Param No.** | **Sym.** | **Characteristic** | **Min.** | **Typ.** | **Max.** | **Units** | **Conditions** |

* These parameters are characterized but not tested.

† Data in "Typ" column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Note:**

1. Device operation below $V_{DD}$ = 1.8 V is not recommended.

### 47.4.7 Analog-to-Digital Converter (ADC) Accuracy Specifications(1,2)

**Table 47-13. ADC Accuracy Specifications**

| **Standard Operating Conditions (unless otherwise stated)**<br>**$V_{DD}$ = 3.0V, $T_A$ = 25°C, $T_{AD}$ = 500 ns** | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Param No.** | **Sym.** | **Characteristic** | **Min.** | **Typ. †** | **Max.** | **Units** | **Conditions** |
| AD01 | $N_R$ | Resolution | — | — | 12 | bit | |
| AD02 | $E_{IL}$ | Integral Nonlinearity Error | — | ±0.1 | ±2.0 | LSb | $ADC_{REF}$+ = 3.0V, $ADC_{REF}$- = 0V |
| AD03 | $E_{DL}$ | Differential Nonlinearity Error | -0.5 | ±0.1 | +1.0 | LSb | $ADC_{REF}$+ = 3.0V, $ADC_{REF}$- = 0V |
| AD04 | $E_{OFF}$ | Offset Error | — | 0.5 | 6.0 | LSb | $ADC_{REF}$+ = 3.0V, $ADC_{REF}$- = 0V |
| AD05 | $E_{GN}$ | Gain Error | — | ±0.2 | ±6.0 | LSb | $ADC_{REF}$+ = 3.0V, $ADC_{REF}$- = 0V |
| AD06 | $V_{ADREF}$ | ADC Reference Voltage ($AD_{REF}$+ - $AD_{REF}$-) | 1.8 | — | $V_{DD}$ | V | |
| AD07 | $V_{AIN}$ | Full-Scale Range | $AD_{REF}$- | — | $AD_{REF}$+ | V | |
| AD08 | $Z_{AIN}$ | Recommended Impedance of Analog Voltage Source | — | 1 | — | kΩ | |
| AD09 | $R_{VREF}$ | ADC Voltage Reference Ladder Impedance | — | 50 | — | kΩ | **(Note 3)** |

* These parameters are characterized but not tested.

† Data in "Typ" column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Notes:**

1. Total Absolute Error is the sum of the offset, gain and integral nonlinearity (INL) errors.
2. The ADC conversion result never decreases with an increase in the input and has no missing codes.
3. This is the impedance seen by the $V_{REF}$ pads when the external reference pads are selected.

© 2019-2023 Microchip Technology Inc. and its subsidiaries

### 47.4.8 Analog-to-Digital Converter (ADC) Conversion Timing Specifications

**Table 47-14. ADC Conversion Timing**

| Standard Operating Conditions (unless otherwise stated) | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Param No.** | **Sym.** | **Characteristic** | **Min.** | **Typ. †** | **Max.** | **Units** | **Conditions** |
| AD20 | $T_{AD}$ | ADC Clock Period | 0.5 | — | 9 | µs | Using $F_{OSC}$ as the ADC clock source ADOCS = 0 |
| | | | — | 2 | — | µs | Using ADCRC as the ADC clock source ADOCS = 1 |
| AD21 | $T_{CNV}$ | Conversion Time | — | 14 $T_{AD}$+2$T_{CY}$ | — | — | Using $F_{OSC}$ as the ADC clock source ADOCS = 0 |
| | | | — | 16 $T_{AD}$+2$T_{CY}$ | — | — | Using ADCRC as the ADC clock source ADOCS = 1 |
| AD22 | $T_{HCD}$ | Sample-and-Hold Capacitor Disconnect Time | — | 2 $T_{AD}$+1$T_{CY}$ | — | — | Using $F_{OSC}$ as the ADC clock source ADOCS = 0 |
| | | | — | 3 $T_{AD}$+2$T_{CY}$ | — | — | Using ADCRC as the ADC clock source ADOCS = 1 |

* These parameters are characterized but not tested.

† Data in “Typ” column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Figure 47-9. ADC Conversion Timing (ADC Clock $F_{OSC}$-Based)**

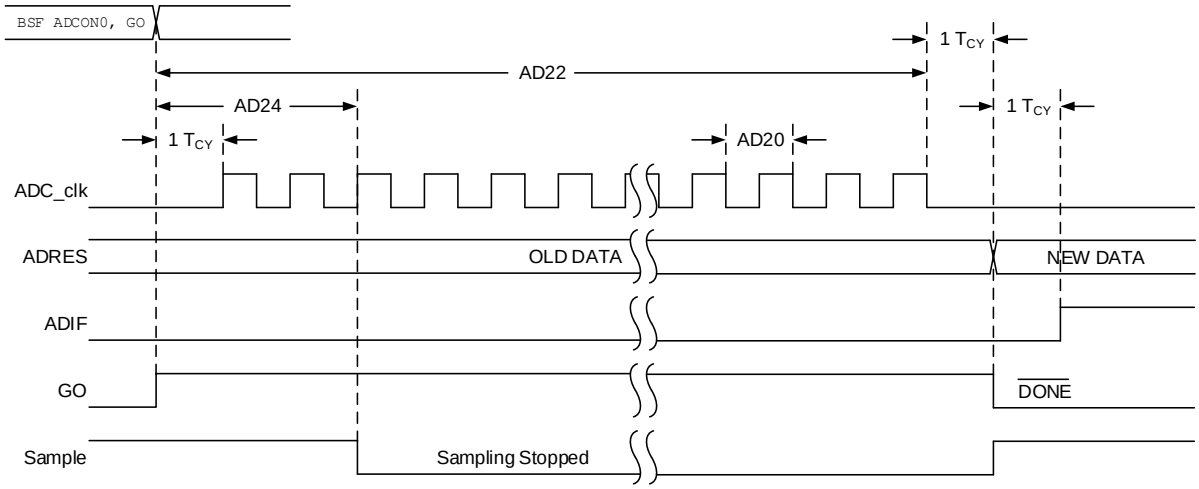
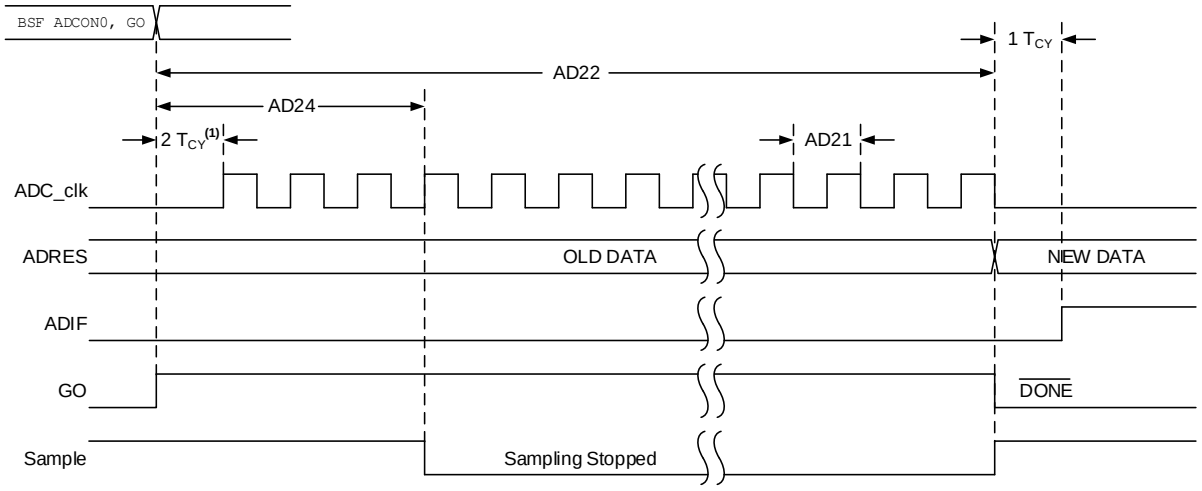


**Figure 47-10. ADC Conversion Timing (ADC Clock from ADCRC)**




**Note 1:** If the ADC clock source is selected as ADCRC, a time of 1 TCY is added before the ADC clock starts. This allows the SLEEP instruction to be executed, if any.

## 47.4.9 Comparator Specifications

**Table 47-15. Comparator Specifications**

| Standard Operating Conditions (unless otherwise stated)<br>$V_{DD}$ = 3.0V, $T_A$ = 25°C | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Param No.** | **Sym.** | **Characteristic** | **Min.** | **Typ. †** | **Max.** | **Units** | **Conditions** |
| CM01 | $V_{IOFF}$ | Input Offset Voltage | — | — | ±50 | mV | $V_{ICM} = V_{DD}/2$ |
| CM02 | $V_{ICM}$ | Input Common Mode Range | GND | — | $V_{DD}$ | V | |
| CM03 | CMRR | Common Mode Input Rejection Ratio | — | 50 | — | dB | |
| CM04 | $V_{HYST}$ | Comparator Hysteresis | 10 | 25 | 50 | mV | |
| CM05 | $T_{RESP}$(1) | Response Time, Rising Edge | — | 300 | 600 | ns | |
| | | Response Time, Falling Edge | — | 220 | 500 | ns | |

* These parameters are characterized but not tested.

† Data in “Typ” column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Note:**

1. Response time measured with one comparator input at $V_{DD}/2$, while the other input transitions from $V_{SS}$ to $V_{DD}$.

### 47.4.10 8-Bit DAC Specifications

**Table 47-16. 8-bit Digital-to-Analog Converter**

| Standard Operating Conditions (unless otherwise stated)<br>$V_{DD}$ = 3.0V, $T_A$ = 25°C | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Param No.** | **Sym.** | **Characteristic** | **Min.** | **Typ. †** | **Max.** | **Units** | **Conditions** |
| DSB01 | $V_{LSB}$ | Step Size | — | ($V_{DACREF^+}$- $V_{DACREF^-}$)/256 | — | V | |
| DSB02 | $V_{ACC}$ | Absolute Accuracy(1) | -2.5 | 1.9 | 7 | LSb | |
| DSB03* | $R_{UNIT}$ | Unit Resistor Value | — | 20 | — | kΩ | |
| DSB04* | $T_{ST}$ | Settling Time(2) | — | 10 | — | µs | |
| DSB05 | $V_{DBO}$ | DAC Buffer Offset | — | 20 | 45 | mV | |
| DSB06 | INL | Integral Nonlinearity | -1.7 | 1 | 1.9 | LSb | 0x09 ≤ DACxDATL ≥ 0x246 |
| DSB07 | DNL | Differential Nonlinearity | -0.5 | 0.4 | 1.2 | LSb | 0x09 ≤ DACxDATL ≥ 0x246 |
| DSB08 | $E_{OFF}$ | Offset Error | -0.8 | 1.4 | 2.5 | LSb | 0x09 ≤ DACxDATL ≥ 0x246 |
| DSB09 | $E_{GN}$ | Gain Error | -1.7 | -1.2 | 0.8 | LSb | 0x09 ≤ DACxDATL ≥ 0x246 |

* These parameters are characterized but not tested.

† Data in “Typ” column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Notes:**

1. Absolute accuracy = Offset Error + Gain Error + DAC Buffer Error ($E_{OFF}$ + $E_{GN}$ + $V_{DBO}$)
2. Settling time measured while DACR[7:0] transitions from `'b00000000` to `'b11111111`

### 47.4.11 Fixed Voltage Reference (FVR) Specifications

**Table 47-17. FVR Specifications**

| Standard Operating Conditions (unless otherwise stated) | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Param No.** | **Sym.** | **Characteristic** | **Min.** | **Typ. †** | **Max.** | **Units** | **Conditions** |
| FVR01 | $V_{FVR}1$ | 1x Gain (1.024V) | -4 | — | +4 | % | $V_{DD}$ ≥ 2.5V, -40°C to 85°C |
| FVR02 | $V_{FVR}2$ | 2x Gain (2.048V) | -4 | — | +4 | % | $V_{DD}$ ≥ 2.5V, -40°C to 85°C |
| FVR03 | $V_{FVR}4$ | 4x Gain (4.096V) | -5 | — | +5 | % | $V_{DD}$ ≥ 4.75V, -40°C to 85°C |
| FVR04 | $T_{FVRST}$ | FVR Start-up Time | — | 260 | — | µs | |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

### 47.4.12 Zero-Cross Detect (ZCD) Specifications

**Table 47-18. ZCD Specifications**

| Standard Operating Conditions (unless otherwise stated)<br>$V_{DD}$ = 3.0V, $T_A$ = 25°C | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Param No.** | **Sym.** | **Characteristic** | **Min.** | **Typ. †** | **Max.** | **Units** | **Conditions** |
| ZC01 | $V_{PINZC}$ | Voltage on Zero Cross Pin | — | 0.9 | — | V | |
| ZC02 | $I_{ZCD_MAX}$ | Maximum source or sink current | — | — | 600 | µA | |
| ZC03 | $T_{RESPH}$ | Response Time, Rising Edge | — | 1 | — | µs | |
| | $T_{RESPL}$ | Response Time, Falling Edge | — | 1 | — | µs | |

† Data in “Typ” column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

### 47.4.13 Timer0 and Timer1 External Clock Requirements

**Table 47-19. TMR0 and TMR1 Requirements**

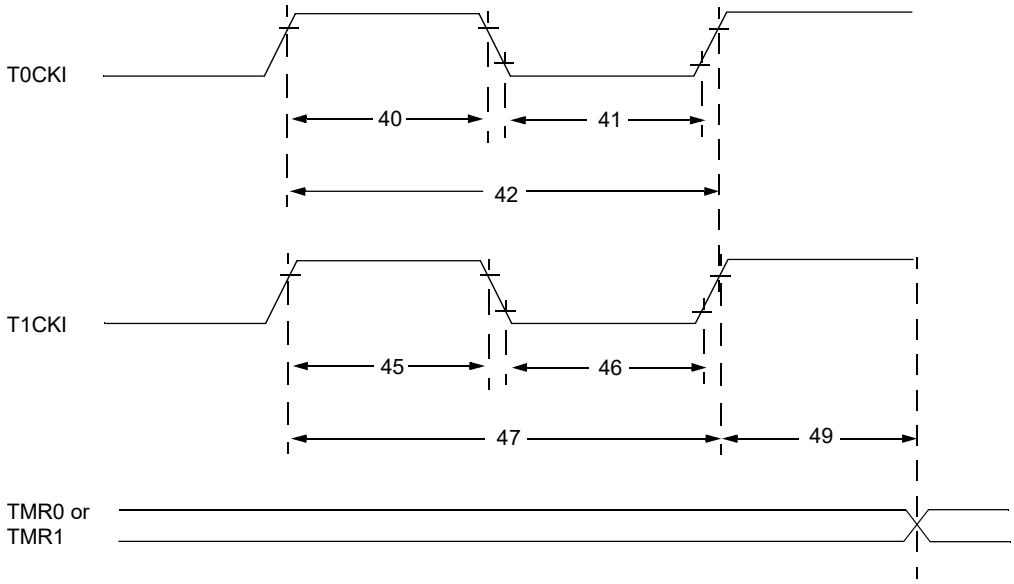
| Standard Operating Conditions (unless otherwise stated)<br>Operating Temperature: -40°C≤$T_A$≤+125°C | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Param No.** | **Sym.** | **Characteristic** | | **Min.** | **Typ. †** | **Max.** | **Units** | **Conditions** |
| 40* | $T_T0H$ | T0CKI High Pulse Width | No Prescaler | 0.5$T_{CY}$+20 | — | — | ns | |
| | | | With Prescaler | 10 | — | — | ns | |
| 41* | $T_T0L$ | T0CKI Low Pulse Width | No Prescaler | 0.5$T_{CY}$+20 | — | — | ns | |
| | | | With Prescaler | 10 | — | — | ns | |
| 42* | $T_T0P$ | T0CKI Period | | Greater of: 20 or ($T_{CY}$+40)/N | — | — | ns | N = Prescale value |
| 45* | $T_T1H$ | T1CKI High Time | Synchronous, No Prescaler | 0.5$T_{CY}$+20 | — | — | ns | |
| | | | Synchronous, with Prescaler | 15 | — | — | ns | |
| | | | Asynchronous | 30 | — | — | ns | |
| 46* | $T_T1L$ | T1CKI Low Time | Synchronous, No Prescaler | 0.5$T_{CY}$+20 | — | — | ns | |
| | | | Synchronous, with Prescaler | 15 | — | — | ns | |
| | | | Asynchronous | 30 | — | — | ns | |
| 47* | $T_T1P$ | T1CKI Input Period | Synchronous | Greater of: 30 or ($T_{CY}$+40)/N | — | — | ns | N = Prescale value |
| | | | Asynchronous | 60 | — | — | ns | |

| .....continued | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Standard Operating Conditions (unless otherwise stated)**<br>**Operating Temperature: -40°C≤$T_A$≤+125°C** | | | | | | | |
| **Param No.** | **Sym.** | **Characteristic** | **Min.** | **Typ. †** | **Max.** | **Units** | **Conditions** |
| 49* | $TCKEZ_{TMR}1$ | Delay from External Clock Edge to Timer Increment | 2 $T_{OSC}$ | — | 7 $T_{OSC}$ | — | Timers in Sync mode |

* These parameters are characterized but not tested.

† Data in "Typ" column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Figure 47-11. Timer0 and Timing1 External Clock Timings**




### 47.4.14 Capture/Compare/PWM Requirements (CCP)

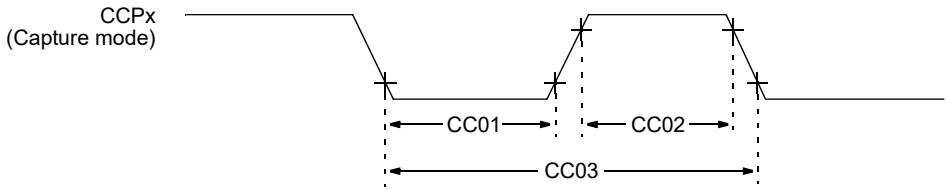
**Table 47-20. CCP Requirements**

| **Standard Operating Conditions (unless otherwise stated)**<br>**Operating Temperature: -40°C ≤ $T_A$ ≤ +125°C** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Param No.** | **Sym.** | **Characteristic** | | **Min.** | **Typ. †** | **Max.** | **Units** | **Conditions** |
| CC01* | $T_{CC}L$ | CCPx Input Low Time | No Prescaler | 0.5$T_{CY}$+20 | — | — | ns | |
| | | | With Prescaler | 20 | — | — | ns | |
| CC02* | $T_{CC}H$ | CCPx Input High Time | No Prescaler | 0.5$T_{CY}$+20 | — | — | ns | |
| | | | With Prescaler | 20 | — | — | ns | |
| CC03* | $T_{CC}P$ | CCPx Input Period | | (3$T_{CY}$+40)/N | — | — | ns | N = Prescale value |

* These parameters are characterized but not tested.

† Data in "Typ" column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Figure 47-12. Capture/Compare/PWM Timings (CCP)**




**Note:** Refer to the Load Conditions figure for more details.

### 47.4.15 SPI Mode Requirements

**Table 47-21. SPI Host Mode**

| Standard Operating Conditions (unless otherwise stated) | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Param No.** | **Sym.** | **Characteristic** | **Min.** | **Typ. †** | **Max.** | **Units** | **Conditions** |
| | $T_{SCK}$ | SCK Cycle Time (2x Prescaled) | 61 | — | — | ns | Transmit only mode |
| | | | — | 16(1) | — | MHz | |
| | | | 95 | — | — | ns | Full-Duplex mode |
| | | | — | 10(1) | — | MHz | |
| SP70* | $T_{SS}L2_{SC}H$, $T_{SS}L2_{SC}L$ | SDO to SCK↓ or SCK↑ input | $T_{SCK}$ | — | — | ns | FST = 0 |
| | | | 0 | — | — | ns | FST = 1 |
| SP71* | $T_{SC}H$ | SCK output high time | 0.5 $T_{SCK}$ - 12 | — | 0.5 $T_{SCK}$ + 12 | ns | |
| SP72* | $T_{SC}L$ | SCK output low time | 0.5 $T_{SCK}$ - 12 | — | 0.5 $T_{SCK}$ + 12 | ns | |
| SP73* | $T_{DI}V2_{SC}H$, $T_{DI}V2_{SC}L$ | Setup time of SDI data input to SCK edge | 85 | — | — | ns | |
| SP74* | $T_{SC}H2_{DI}L$, $T_{SC}L2_{DI}L$ | Hold time of SDI data input to SCK edge | 0 | — | — | ns | |
| | | Hold time of SDI data input to final SCK | 0.5 $T_{SCK}$ | | | ns | CKE = 0, SMP = 1 |
| SP75* | $T_{DO}R$ | SDO data output rise time | — | 10 | 25 | ns | $C_L$ = 50 pF |
| SP76* | $T_{DO}F$ | SDO data output fall time | — | 10 | 25 | ns | $C_L$ = 50 pF |
| SP78* | $T_{SC}R$ | SCK output rise time | — | 10 | 25 | ns | $C_L$ = 50 pF |
| SP79* | $T_{SC}F$ | SCK output fall time | — | 10 | 25 | ns | $C_L$ = 50 pF |
| SP80* | $T_{SC}H2_{DO}V$, $T_{SC}L2_{DO}V$ | SDO data output valid after SCK edge | -15 | — | 15 | ns | $C_L$ = 50 pF |
| SP81* | $T_{DO}V2_{SC}H$, $T_{DO}V2_{SC}L$ | SDO data output valid to first SCK edge | $T_{SCK}$ - 10 | — | — | ns | $C_L$ = 50 pF<br>CKE = 1 |
| SP82* | $T_{SS}L2_{DO}V$ | SDO data output valid after $\overline{SS}$↓ edge | — | — | 50 | ns | $C_L$ = 20 pF |

.....continued

| Standard Operating Conditions (unless otherwise stated) | | | | | | | |
|---|---|---|---|---|---|---|---|
| Param No. | Sym. | Characteristic | Min. | Typ. † | Max. | Units | Conditions |
| SP83* | $T_{SC}H2\overline{SS}H$, $T_{SC}L2\overline{SS}H$ | $\overline{SS}$ ↑ after last SCK edge | $T_{SCK}$ - 10 | — | — | ns | |
| SP84* | $T_{SS}H2\overline{SS}L$ | $\overline{SS}$ ↑ to $\overline{SS}$↓ edge | $T_{SCK}$ - 10 | — | — | ns | |

* These parameters are characterized but not tested.

† Data in "Typ" column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Note:**

1. SMP bit in the SPIxCON1 register must be set and the slew rate control must be disabled on the clock and data pins (clear the corresponding bits in SLRCONx register) for SPI to operate over 4 MHz.

**Table 47-22. SPI Client Mode**

| Standard Operating Conditions (unless otherwise stated) | | | | | | | |
|---|---|---|---|---|---|---|---|
| Param No. | Sym. | Characteristic | Min. | Typ. † | Max. | Units | Conditions |
| | $T_{SCK}$ | SCK Total Cycle Time | 47 | — | — | ns | Receive Only mode |
| | | | — | 20(1) | — | MHz | |
| | | | 95 | — | — | ns | Full-Duplex mode |
| | | | — | 10(1) | — | MHz | |
| SP70* | $T_{SS}L2_{SC}H$, $T_{SS}L2_{SC}L$ | $\overline{SS}$↓ to SCK↓ or SCK↑ input | 0 | — | — | ns | CKE = 0 |
| | | | 25 | — | — | ns | CKE = 1 |
| SP71* | $T_{SC}H$ | SCK input high time | 20 | — | — | ns | |
| SP72* | $T_{SC}L$ | SCK input low time | 20 | — | — | ns | |
| SP73* | $T_{DI}V2_{SC}H$, $T_{DI}V2_{SC}L$ | Setup time of SDI data input to SCK edge | 10 | — | — | ns | |
| SP74* | $T_{SC}H2_{DI}L$, $T_{SC}L2_{DI}L$ | Hold time of SDI data input to SCK edge | 0 | — | — | ns | |
| SP75* | $T_{DO}R$ | SDO data output rise time | — | 10 | 25 | ns | $C_L$ = 50 pF |
| SP76* | $T_{DO}F$ | SDO data output fall time | — | 10 | 25 | ns | $C_L$ = 50 pF |
| SP77* | $T_{SS}H2_{DO}Z$ | $\overline{SS}$↑ to SDO output high-impedance | — | — | 85 | ns | |
| SP80* | $T_{SC}H2_{DO}V$, $T_{SC}L2_{DO}V$ | SDO data output valid after SCK edge | — | — | 85 | ns | |
| SP82* | $T_{SS}L2_{DO}V$ | SDO data output valid after $\overline{SS}$↓ edge | — | — | 85 | ns | |

.....continued

| Standard Operating Conditions (unless otherwise stated) | | | | | | | |
|---|---|---|---|---|---|---|---|
| Param No. | Sym. | Characteristic | Min. | Typ. † | Max. | Units | Conditions |
| SP83* | $T_{SC}H2_{SS}H$, $T_{SC}L2_{SS}H$ | $\overline{SS}$ ↑ after SCK edge | 20 | — | — | ns | |
| SP84* | $T_{SS}H2_{SSL}$ | $\overline{SS}$ ↑ to $\overline{SS}$↓ edge | 47 | — | — | ns | |

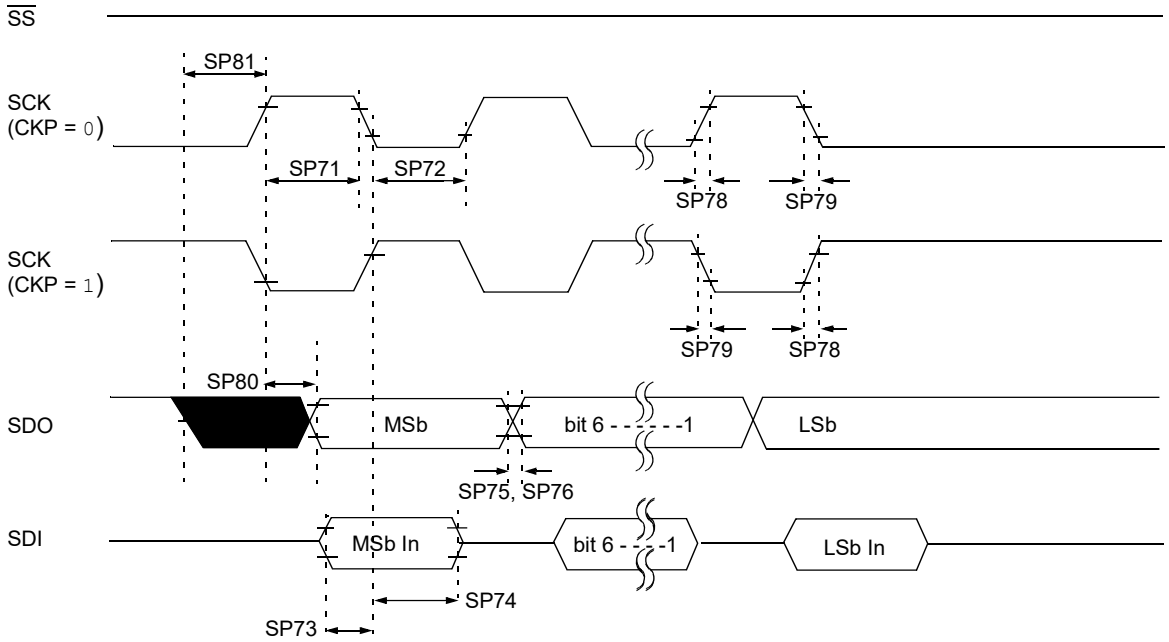
* These parameters are characterized but not tested.

† Data in "Typ" column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Note:**

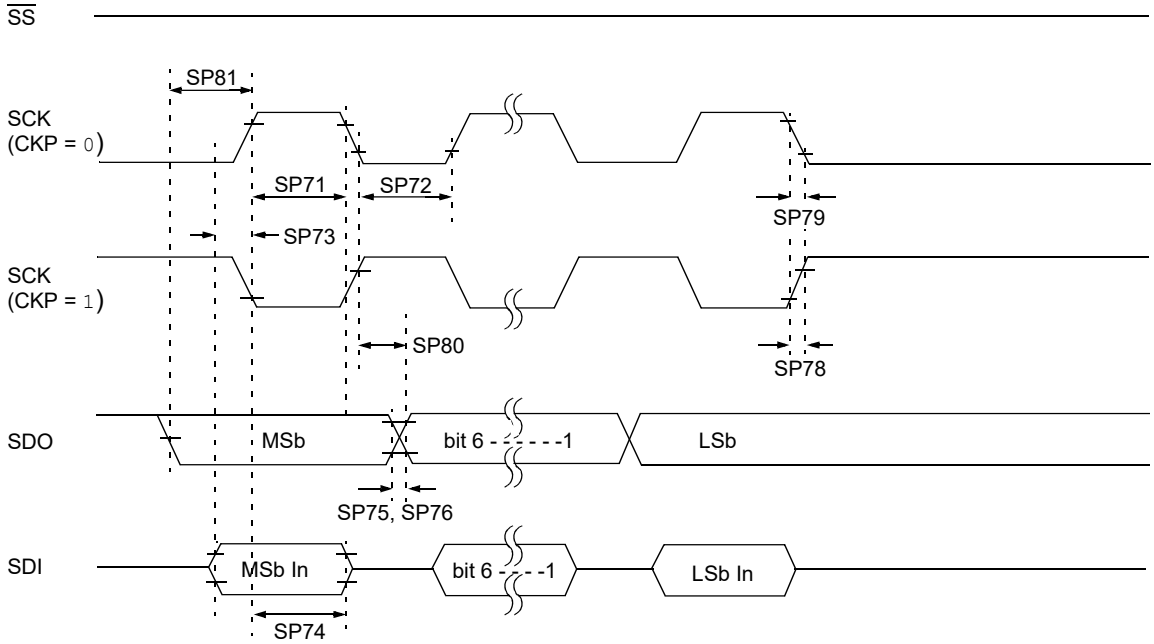
1. SMP bit in the SPIxCON1 register must be set and the slew rate control must be disabled on the clock and data pins (clear the corresponding bits in SLRCONx register) for SPI to operate over 4 MHz.

**Figure 47-13. SPI Host Mode Timing (CKE = 0, SMP = 0)**



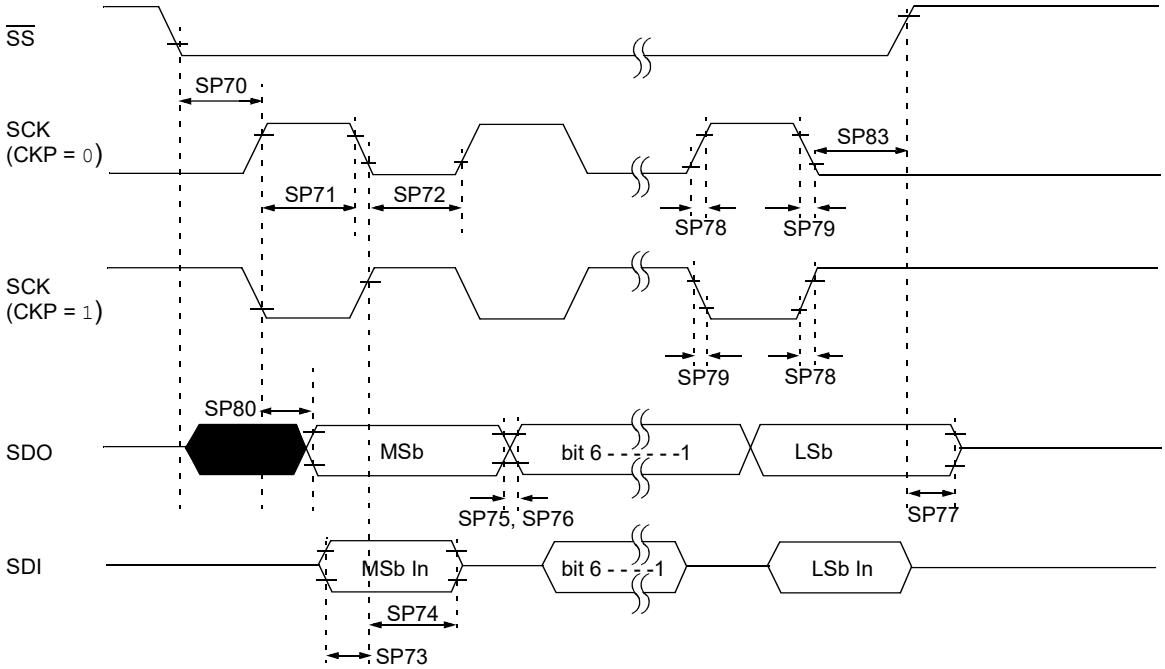

**Note:** Refer to the Load Conditions figure for more details.

**Figure 47-14. SPI Host Mode Timing (CKE = 1, SMP = 1)**



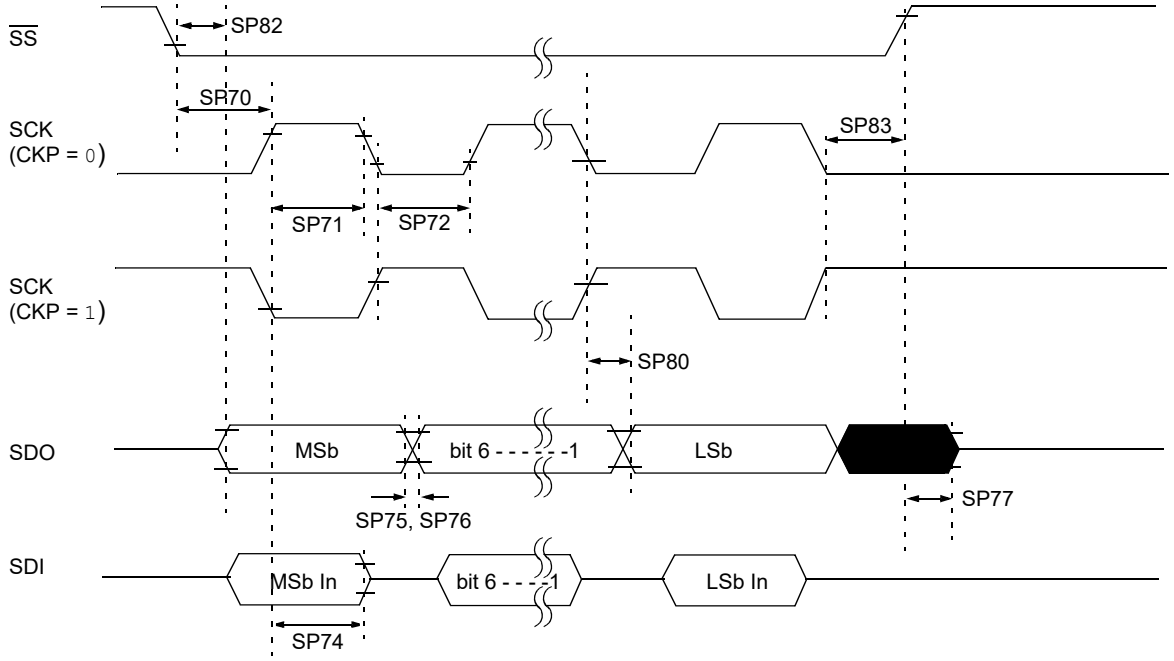

**Note:** Refer to the Load Conditions figure for more details.

**Figure 47-15. SPI Client Mode Timing (CKE = 0)**




**Note:** Refer to the Load Conditions figure for more details.

**Figure 47-16. SPI Client Mode Timing (CKE = 1)**



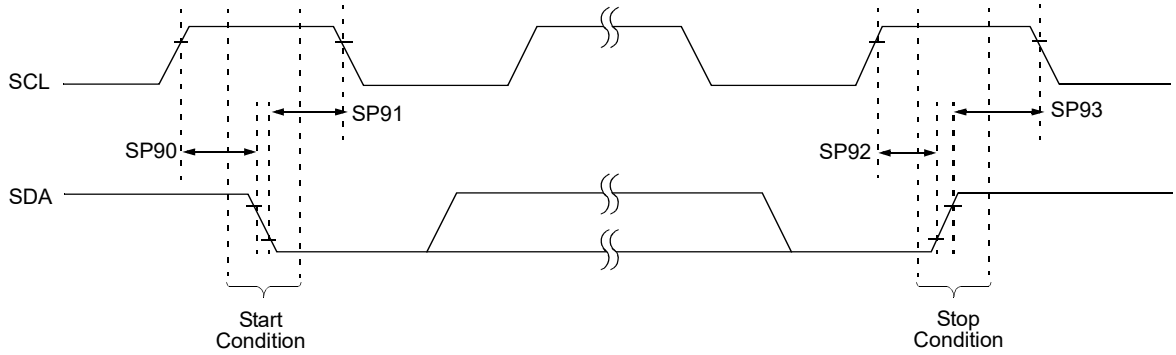

**Note:** Refer to the Load Conditions figure for more details.

### 47.4.16 I²C Bus Start/Stop Bits Requirements

**Table 47-23. I²C Start/Stop Requirements**

| Standard Operating Conditions (unless otherwise stated) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Param. No.** | **Sym.** | **Characteristic** | | **Min.** | **Typ. †** | **Max.** | **Units** | **Conditions** |
| SP90* | $T_{SU:STA}$ | Start condition Setup time | 100 kHz mode | 4700 | — | — | ns | Only relevant for Repeated Start condition |
| | | | 400 kHz mode | 600 | — | — | | |
| | | | 1 MHz mode | 260 | — | — | | |
| SP91* | $T_{HD:STA}$ | Start condition Hold time | 100 kHz mode | 4000 | — | — | ns | After this period, the first clock pulse is generated |
| | | | 400 kHz mode | 600 | — | — | | |
| | | | 1 MHz mode | 260 | — | — | | |
| SP92* | $T_{SU:STO}$ | Stop condition Setup time | 100 kHz mode | 4000 | — | — | ns | |
| | | | 400 kHz mode | 600 | — | — | | |
| | | | 1 MHz mode | 260 | — | — | | |
| SP93* | $T_{HD:STO}$ | Stop condition Hold time | 100 kHz mode | 4700 | — | — | ns | |
| | | | 400 kHz mode | 1300 | — | — | | |
| | | | 1 MHz mode | 500 | — | — | | |
| * These parameters are characterized but not tested. | | | | | | | | |

**Figure 47-17. I²C Bus Start/Stop Bits Timing**




**Note:** Refer to the Load Conditions figure for more details.

### 47.4.17 I²C Bus Data Requirements

**Table 47-24. I²C Bus Data Requirements**

| Standard Operating Conditions (unless otherwise stated) | | | | | | | |
|---|---|---|---|---|---|---|---|
| **Param. No.** | **Sym.** | **Characteristic** | | **Min.** | **Max.** | **Units** | **Conditions** |
| SP100* | $T_{HIGH}$ | Clock high time | 100 kHz mode | 4000 | — | ns | Device must operate at a minimum of 1.5 MHz |
| | | | 400 kHz mode | 600 | — | ns | Device must operate at a minimum of 10 MHz |
| | | | 1 MHz mode | 260 | — | ns | Device must operate at a minimum of 10 MHz |
| SP101* | $T_{LOW}$ | Clock low time | 100 kHz mode | 4700 | — | ns | Device must operate at a minimum of 1.5 MHz |
| | | | 400 kHz mode | 1300 | — | ns | Device must operate at a minimum of 10 MHz |
| | | | 1 MHz mode | 500 | — | ns | Device must operate at a minimum of 10 MHz |
| SP102* | $T_R$ | SDA and SCL rise time | 100 kHz mode | — | 1000 | ns | |
| | | | 400 kHz mode | 20 | 300 | ns | $C_B$ is specified to be from 10-400 pF |
| | | | 1 MHz mode | — | 120 | | |

.....continued

| Standard Operating Conditions (unless otherwise stated) | | | | | | | |
|---|---|---|---|---|---|---|---|
| Param. No. | Sym. | Characteristic | | Min. | Max. | Units | Conditions |
| SP103* | $T_F$ | SDA and SCL fall time | 100 kHz mode | — | 250 | ns | |
| | | | 400 kHz mode | 20 × ($V_{DD}$/5.5V) | 250 | ns | $C_B$ is specified to be from 10-400 pF |
| | | | 1 MHz mode | 20 × ($V_{DD}$/5.5V) | 120 | ns | |
| SP106* | $T_{HD:DAT}$ | Data input hold time | 100 kHz mode | 0 | — | ns | |
| | | | 400 kHz mode | 0 | — | ns | |
| | | | 1 MHz mode | 0 | — | ns | |
| SP107* | $T_{SU:DAT}$ | Data input setup time | 100 kHz mode | 250 | — | ns | **(Note 2)** |
| | | | 400 kHz mode | 100 | — | ns | |
| | | | 1 MHz mode | 50 | — | ns | |
| SP109* | $T_{AA}$ | Output valid from clock | 100 kHz mode | — | 3450 | ns | **(Note 1)** |
| | | | 400 kHz mode | — | 900 | ns | |
| | | | 1 MHz mode | | 450 | ns | |
| SP110* | $T_{BUF}$ | Bus free time | 100 kHz mode | 4700 | — | ns | Time the bus must be free before a new transmission can start |
| | | | 400 kHz mode | 1300 | — | ns | |
| | | | 1 MHz mode | 500 | — | ns | |
| SP111 | $C_B$ | Bus capacitive loading | 100 kHz mode | — | 400 | pF | |
| | | | 400 kHz mode | — | 400 | pF | |
| | | | 1 MHz mode | — | 26 | pF | **(Note 3)** |

© 2019-2023 Microchip Technology Inc. and its subsidiaries

.....continued

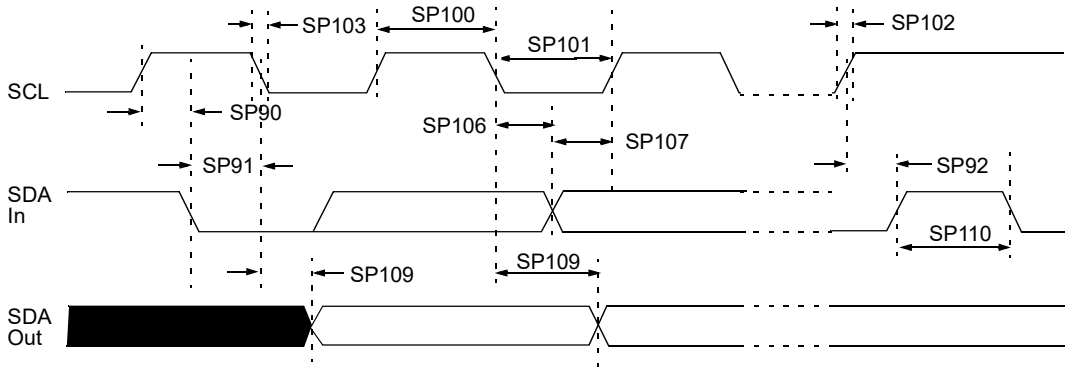
| Standard Operating Conditions (unless otherwise stated) | | | | | | |
|---|---|---|---|---|---|---|
| Param. No. | Sym. | Characteristic | Min. | Max. | Units | Conditions |

* These parameters are characterized but not tested.

**Notes:**

1. As a transmitter, the device must provide this internal minimum delay time to bridge the undefined region (min. 300 ns) of the falling edge of SCL to avoid unintended generation of Start or Stop conditions.
2. A Fast mode (400 kHz) $I^2C$ bus device can be used in a Standard mode (100 kHz) $I^2C$ bus system, but the requirement $T_{SU:DAT} \geq 250$ ns must then be met. This will automatically be the case if the device does not stretch the low period of the SCL signal. If such a device does stretch the low period of the SCL signal, it must output the next data bit to the SDA line TR max. + $T_{SU:DAT}$ = 1000 + 250 = 1250 ns (according to the Standard mode $I^2C$ bus specification), before the SCL line is released.
3. Using internal $I^2C$ pull-ups. For greater bus capacitance use external pull-ups.

**Figure 47-18. $I^2C$ Bus Data Timing**




**Note:** Refer to the Load Conditions figure for more details.

### 47.4.18 Configurable Logic Cell (CLC) Characteristics

**Table 47-25. CLC Characteristics**

| Standard Operating Conditions (unless otherwise stated)<br>Operating Temperature: -40°C≤$T_A$≤+125°C | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Param No. | Sym. | Characteristic | | Min. | Typ. † | Max. | Units | Conditions |
| CLC01* | $T_{CLCIN}$ | CLC input time | | — | 7 | IO5 | ns | **(Note 1)** |
| CLC02* | $T_{CLC}$ | CLC module input to output propagation time | | — | 24 | — | ns | $V_{DD}$ = 1.8V |
| | | | | — | 12 | — | ns | $V_{DD}$ > 3.6V |
| CLC03* | $T_{CLCOUT}$ | CLC output time | Rise Time | — | IO6 | — | — | **(Note 1)** |
| | | | Fall Time | — | IO8 | — | — | **(Note 1)** |
| CLC04* | $F_{CLCMAX}$ | CLC maximum switching frequency | | — | — | OS20 | — | |

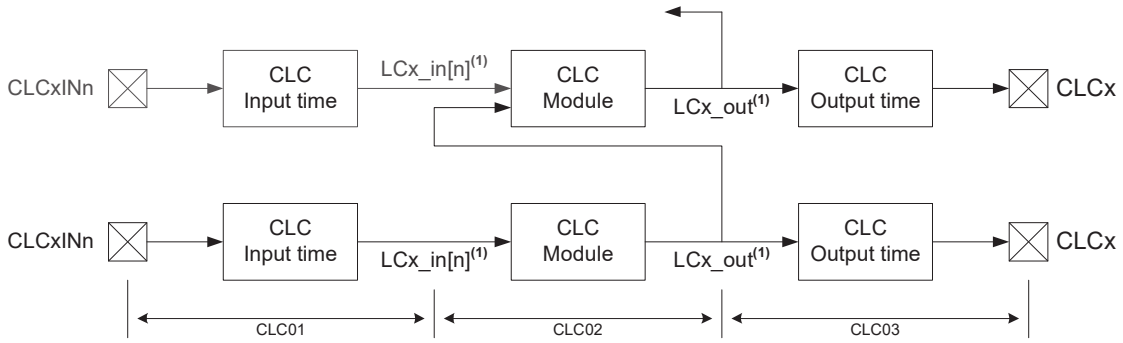
* These parameters are characterized but not tested.

† Data in "Typ" column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

**Note:**

1. See the **"I/O and CLKOUT Timing Specifications"** section for IO5, IO6 and IO8 rise and fall times.

**Figure 47-19. CLC Propagation Timing**




### 47.4.19 Temperature Indicator Requirements

**Table 47-26. Temperature Indicator Requirements**

| Standard Operating Conditions (unless otherwise stated) | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **Param No.** | **Sym.** | **Characteristic** | | **Min.** | **Typ. †** | **Max.** | **Units** | **Conditions** |
| TS01* | $T_{ACQMIN}$ | Minimum ADC Acquisition Time Delay | | — | 25 | — | µs | |
| TS02* | $M_V$ | Voltage Sensitivity | High Range | — | -3.75 | — | mV/℃ | TSRNG = 1 |
| | | | Low Range | — | -2.75 | — | mV/℃ | TSRNG = 0 |

* These parameters are characterized but not tested.

† Data in “Typ” column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

# 48. DC and AC Characteristics Graphs and Tables

The graphs and tables provided in this section are for design guidance and are not tested. In some graphs or tables, the data presented are outside specified operating range (i.e., outside specified $V_{DD}$ range). This is for information only and devices are ensured to operate properly only within the specified range. Unless otherwise noted, all graphs apply to both the L and LF devices.

**Note:** The graphs and tables provided following this note are a statistical summary based on a limited number of samples and are provided for informational purposes only. The performance characteristics listed herein are not tested or guaranteed. In some graphs or tables, the data presented may be outside the specified operating range (e.g., outside specified power supply range) and therefore, outside the warranted range.

**Note:** “Typical” represents the mean of the distribution at 25°C. “Maximum”, “Max.”, “Minimum” or “Min.” represents (mean + 3σ) or (mean - 3σ) respectively, where σ is a standard deviation, over each temperature range.

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

## 48.1 Analog-to-Digital Converter (12-bit) Graphs

**Figure 48-1.**

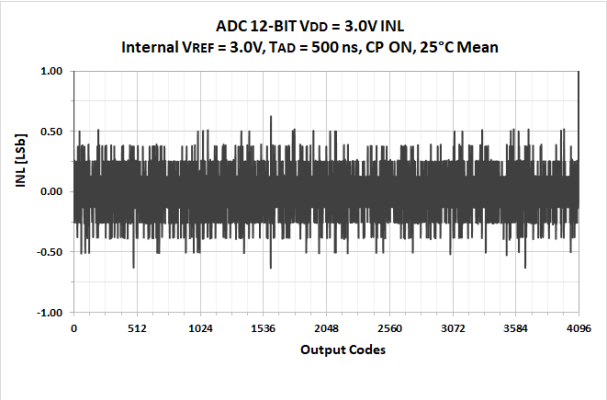



**Figure 48-2.**

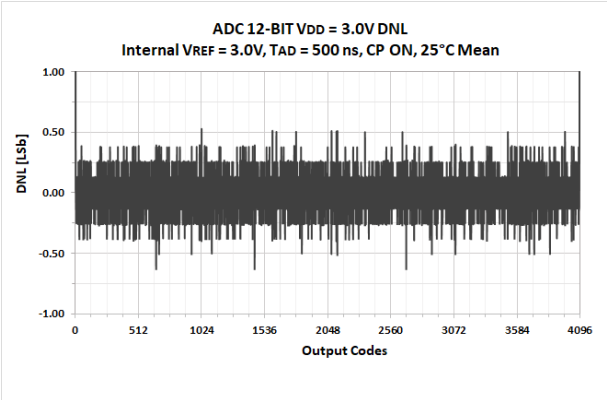



**Figure 48-3.**

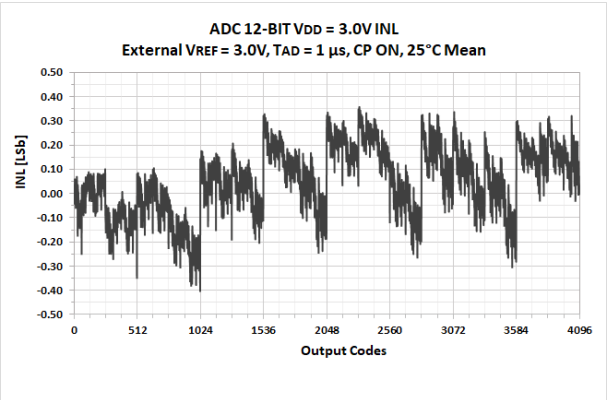



**Figure 48-4.**

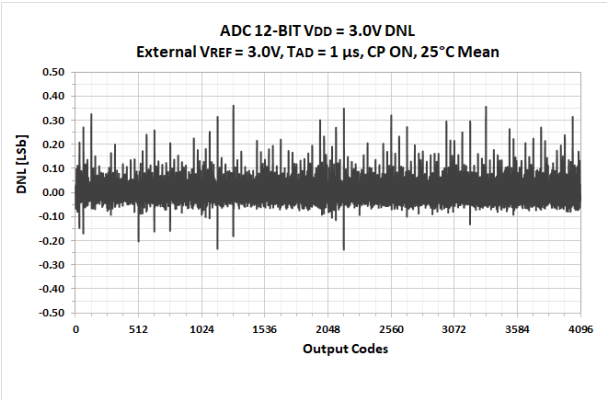



**Figure 48-5.**

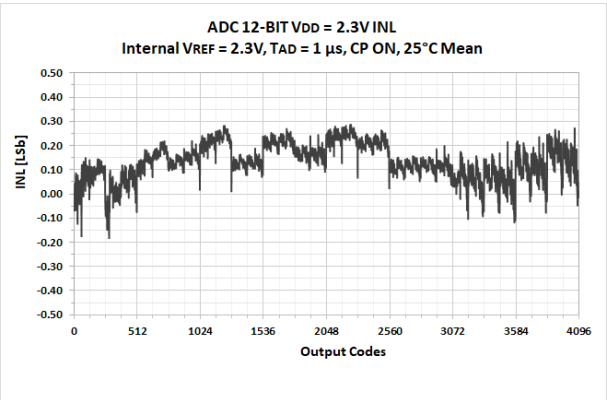



**Figure 48-6.**

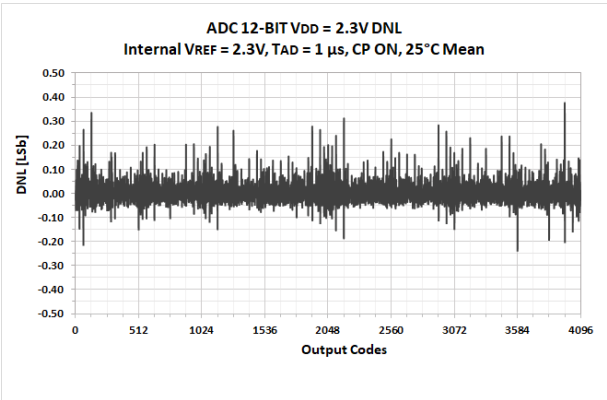


**Figure 48-7.**

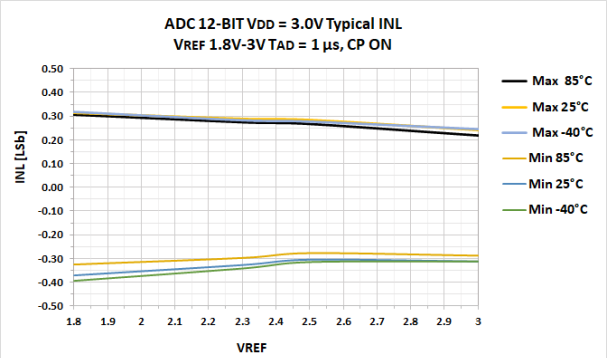
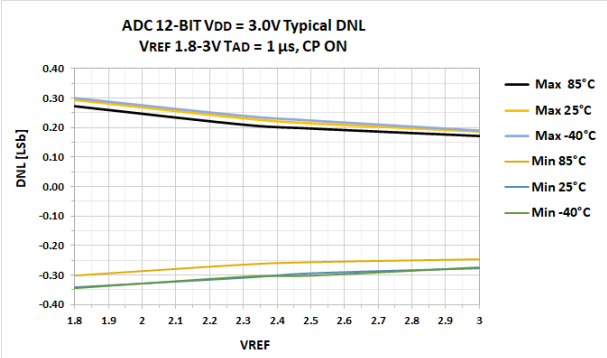



**Figure 48-8.**




## 48.2 Brown-Out Reset Graphs

**Figure 48-9.**

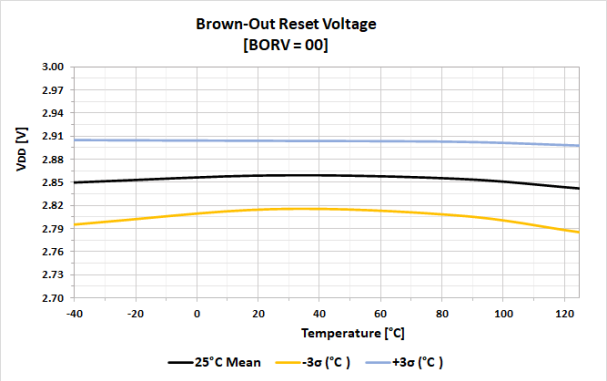



**Figure 48-10.**

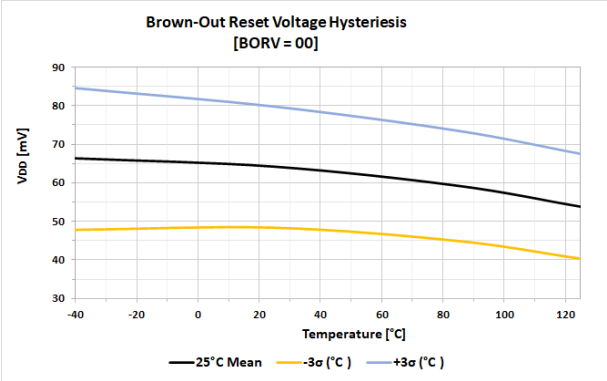



**Figure 48-11.**

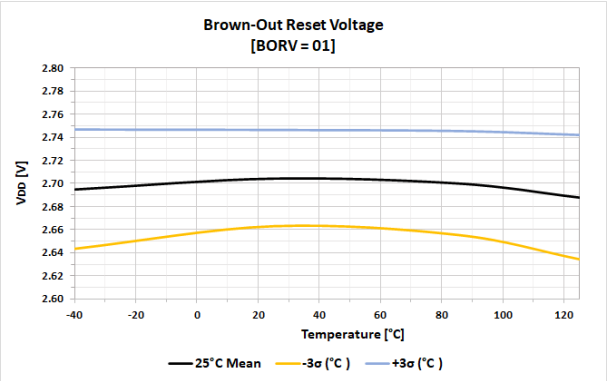



**Figure 48-12.**

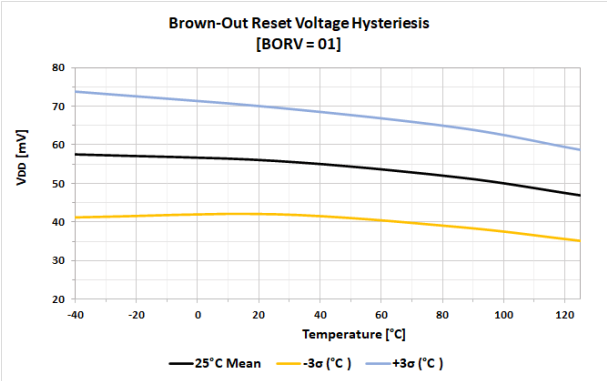


**Figure 48-13.**

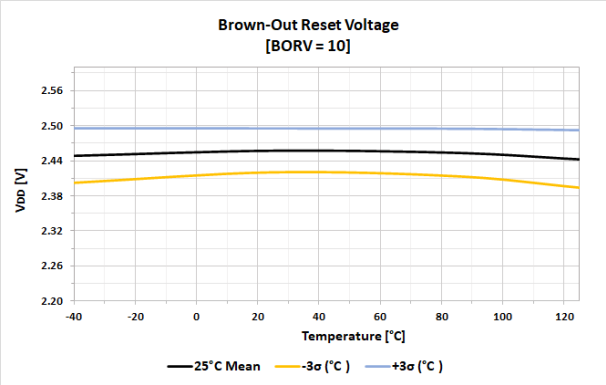



**Figure 48-14.**

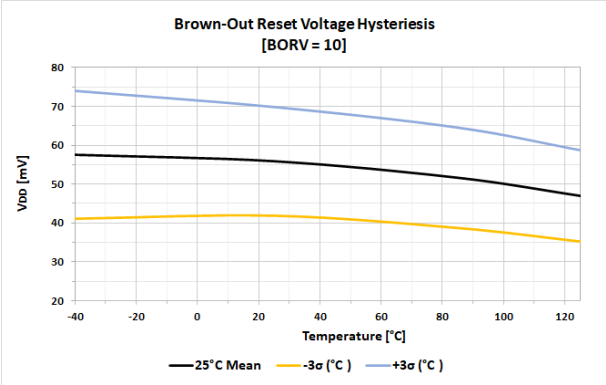



**Figure 48-15.**

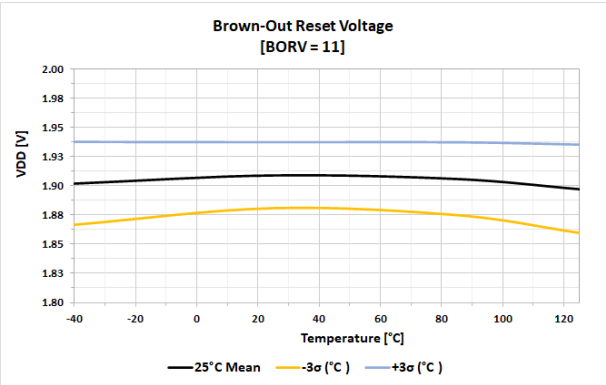
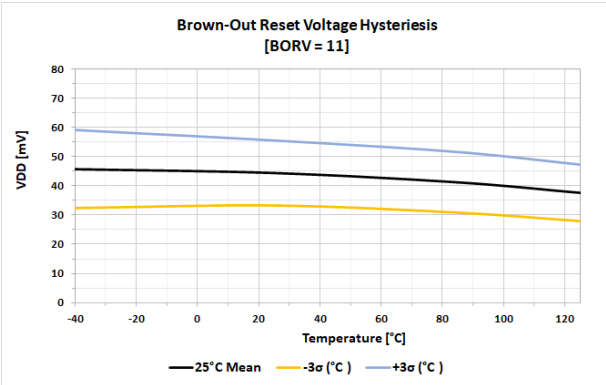



**Figure 48-16.**




## 48.3 Digital-to-Analog Converter Graphs

**Figure 48-17.**

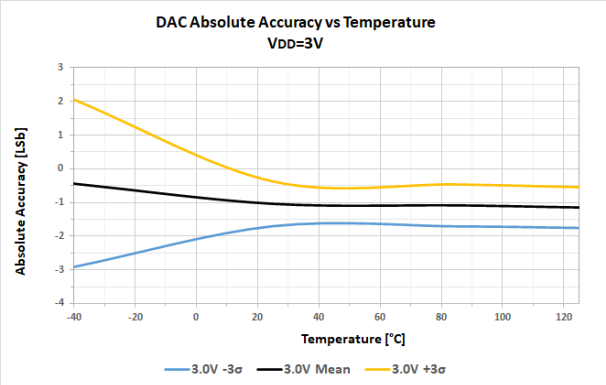



**Figure 48-18.**

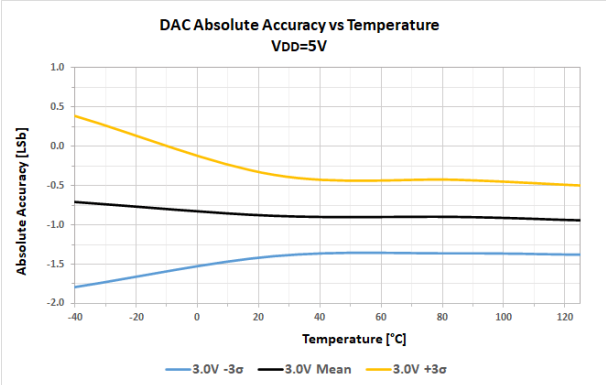


**Figure 48-19.**

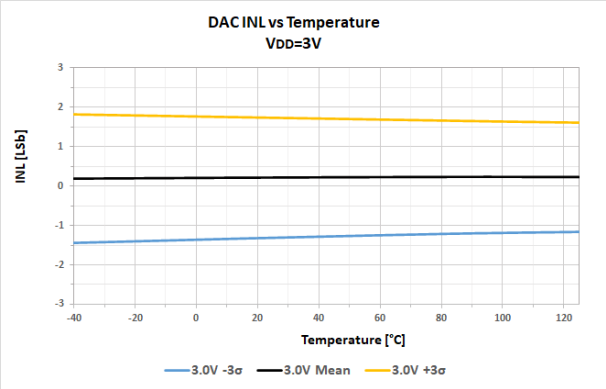



**Figure 48-20.**

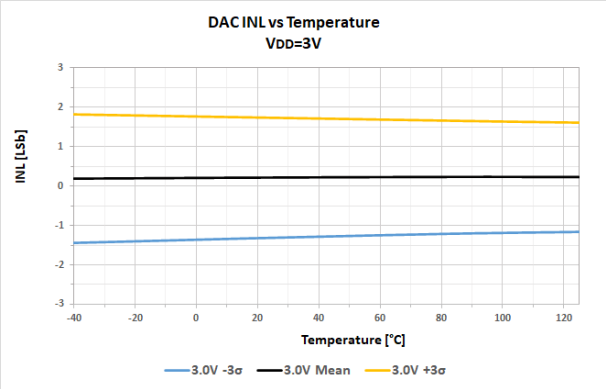



**Figure 48-21.**

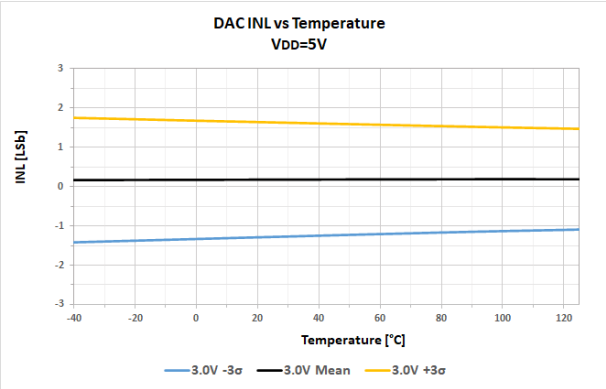



**Figure 48-22.**

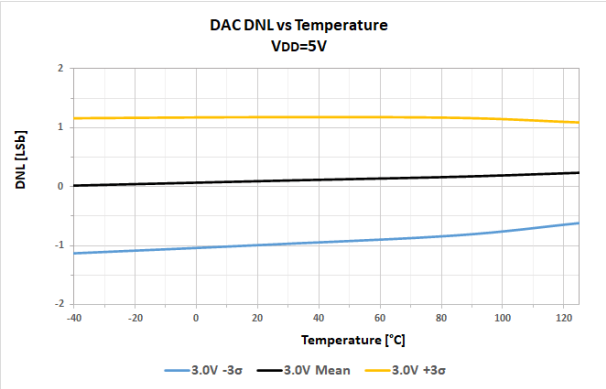



**Figure 48-23.**

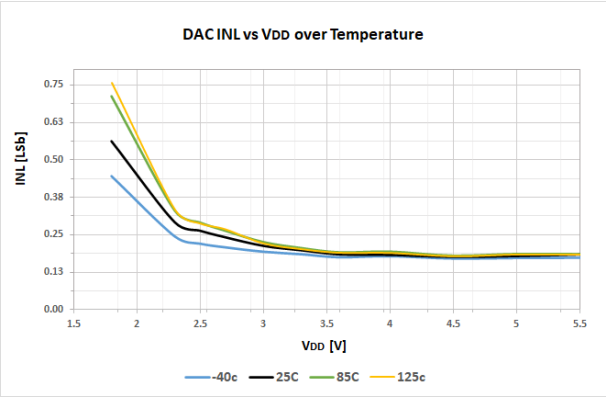



**Figure 48-24.**

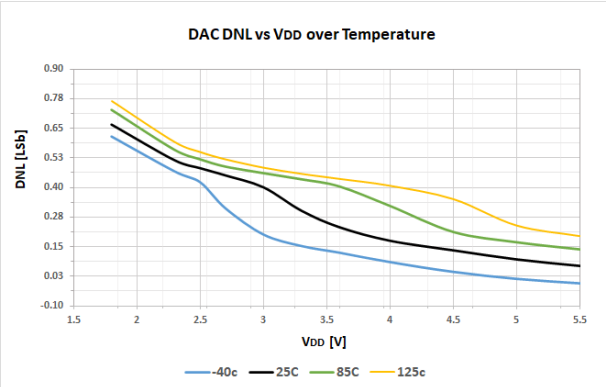


**Figure 48-25.**

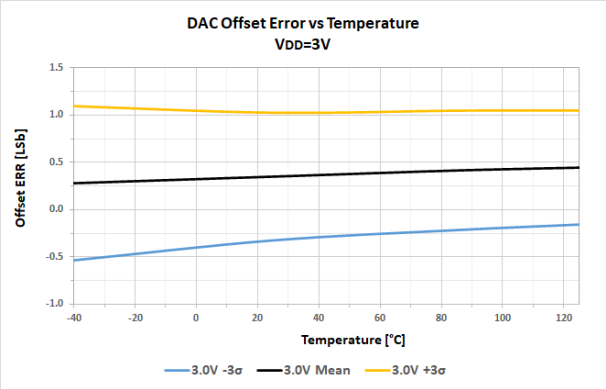



**Figure 48-26.**

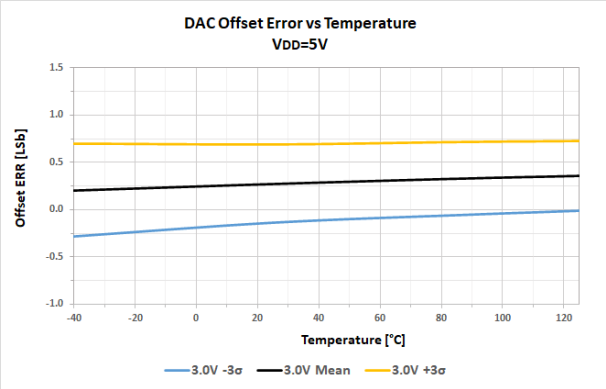



**Figure 48-27.**

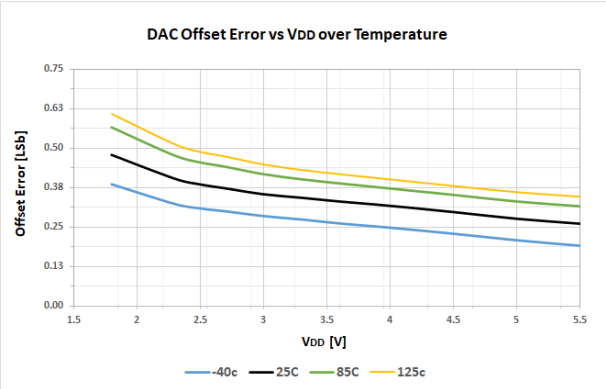



**Figure 48-28.**

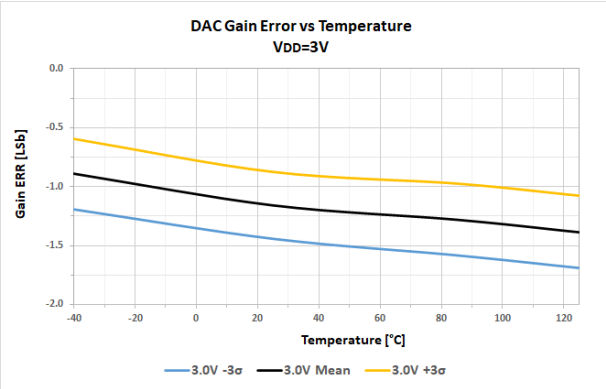



**Figure 48-29.**

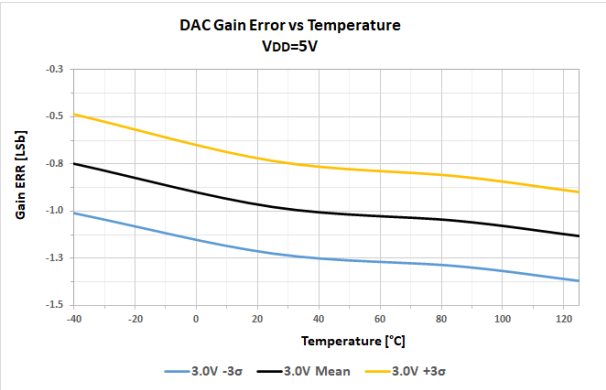
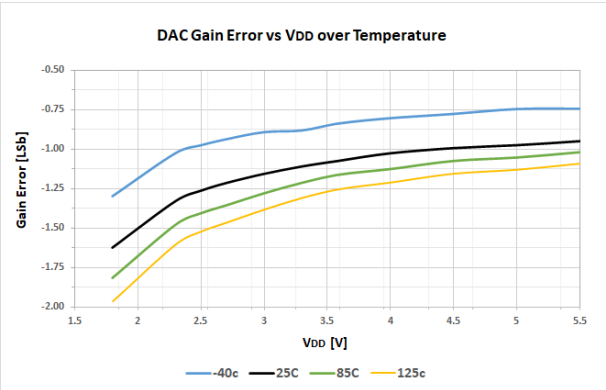



**Figure 48-30.**



## 48.4 Fixed Voltage Reference Graphs

**Figure 48-31.**

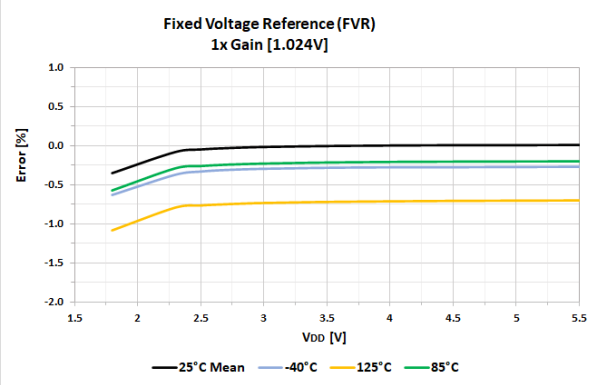



**Figure 48-32.**

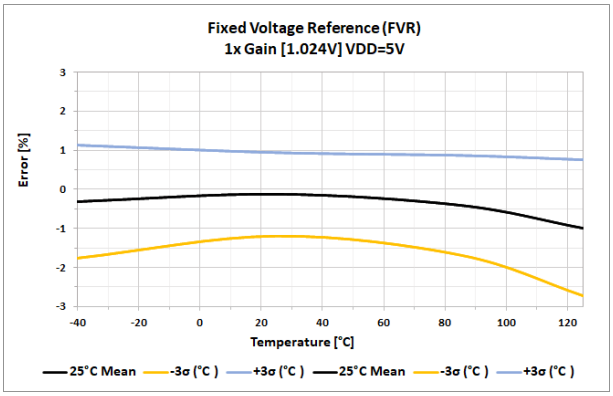



**Figure 48-33.**

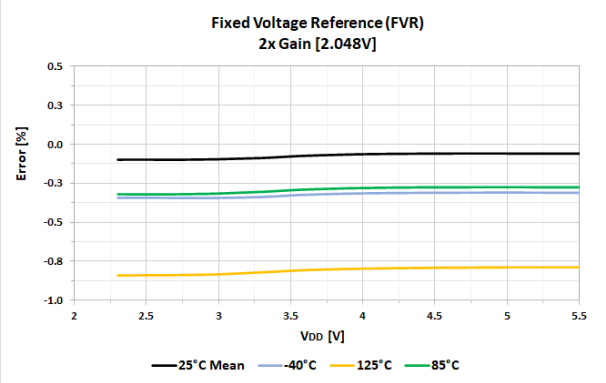



**Figure 48-34.**

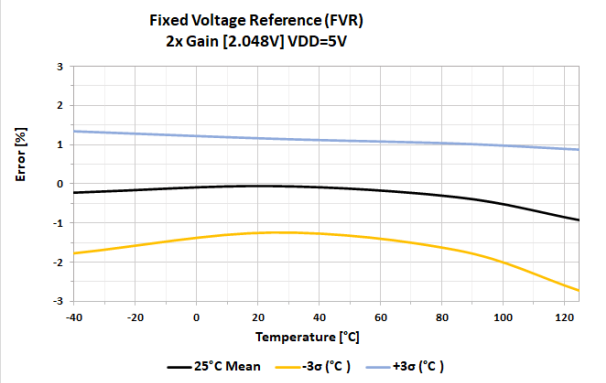



**Figure 48-35.**

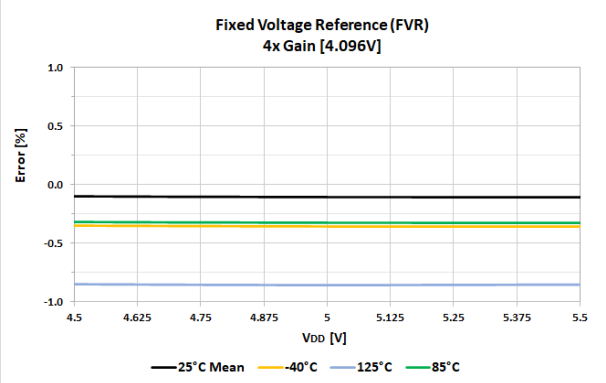
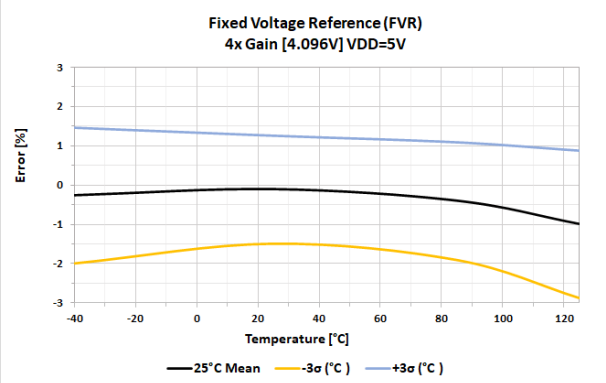



**Figure 48-36.**



## 48.5 HFINTOSC Graphs

**Figure 48-37.**

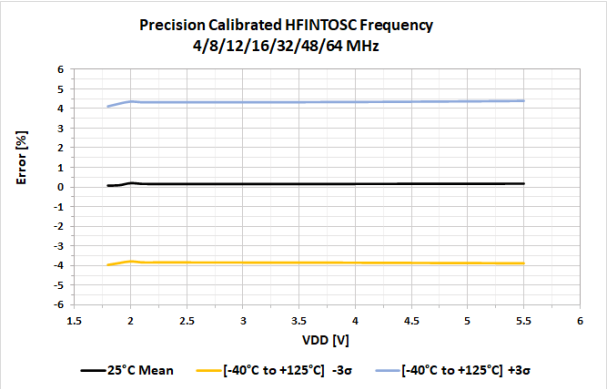



**Figure 48-38.**

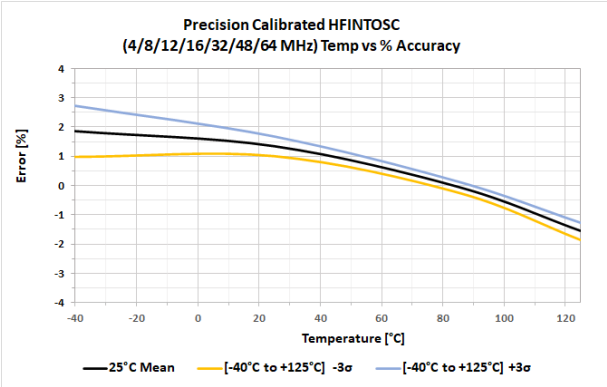



**Figure 48-39.**

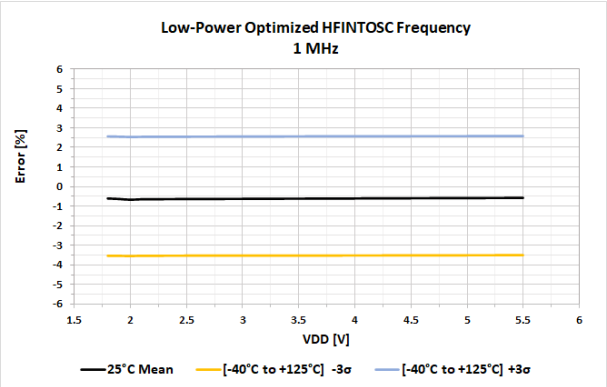



**Figure 48-40.**

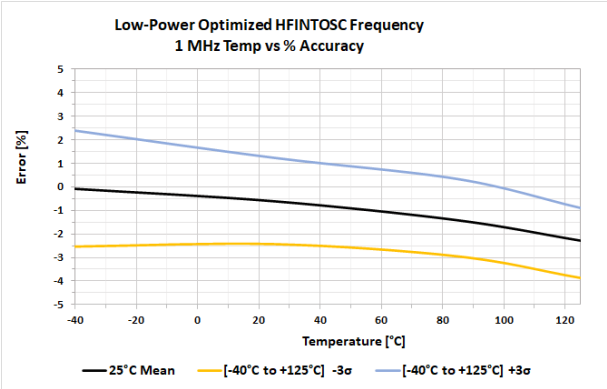



**Figure 48-41.**

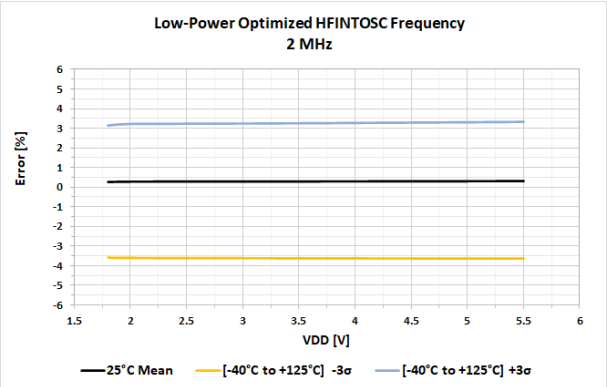
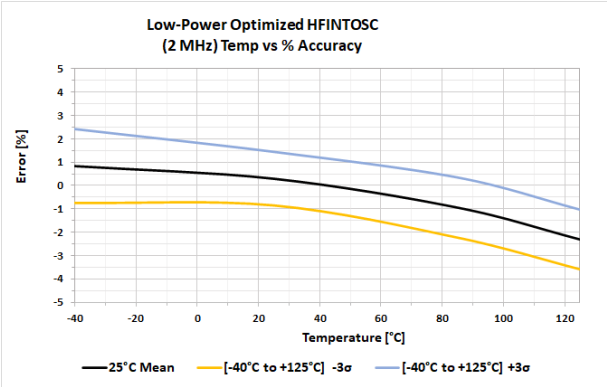



**Figure 48-42.**



## 48.6 HFINTOSC Wake From Sleep Graphs

**Figure 48-43.**

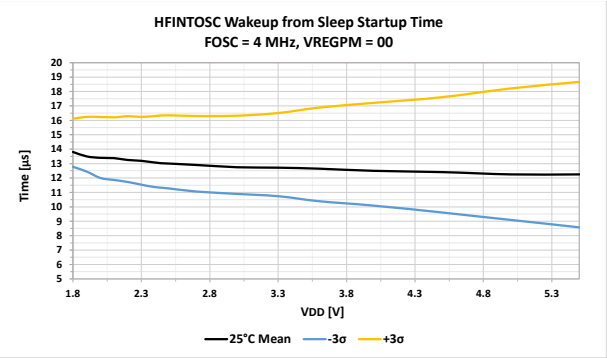



**Figure 48-44.**

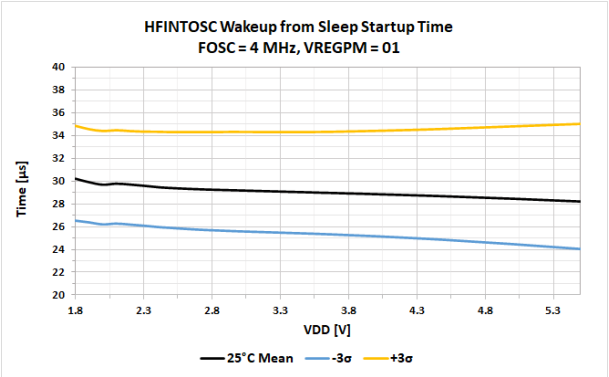



**Figure 48-45.**

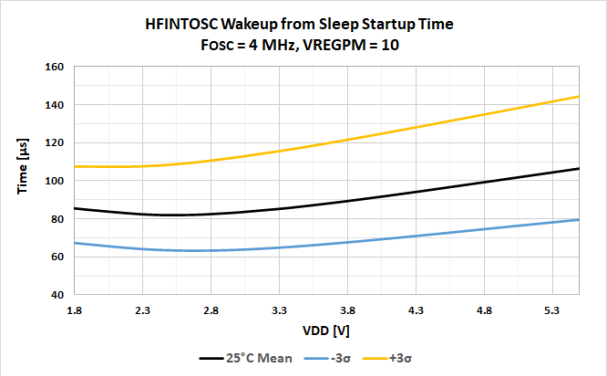
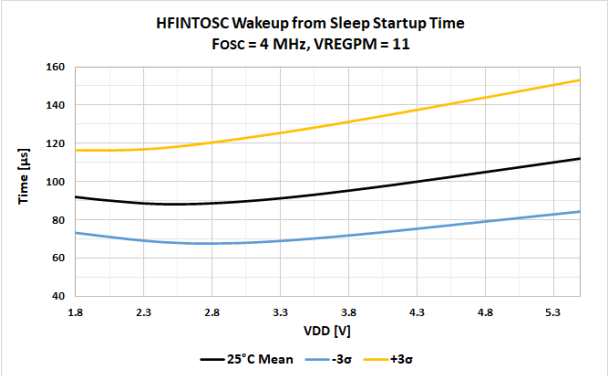



**Figure 48-46.**




## 48.7 High/Low-Voltage Detect Graphs

**Figure 48-47.**

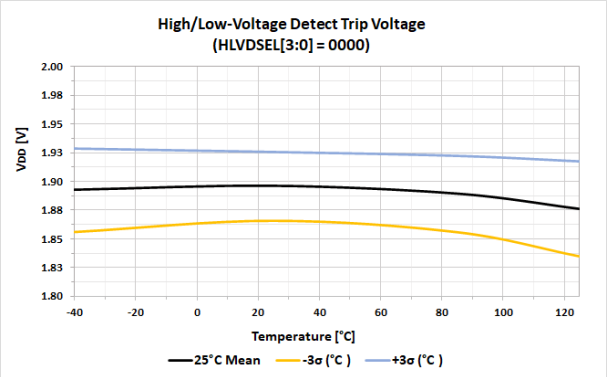



**Figure 48-48.**

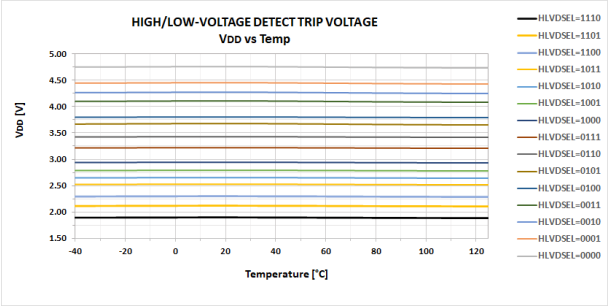
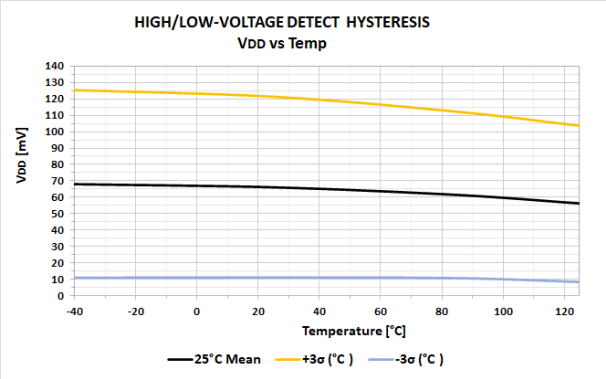


**Figure 48-49.**




## 48.8 I/O Graphs

**Figure 48-50.**

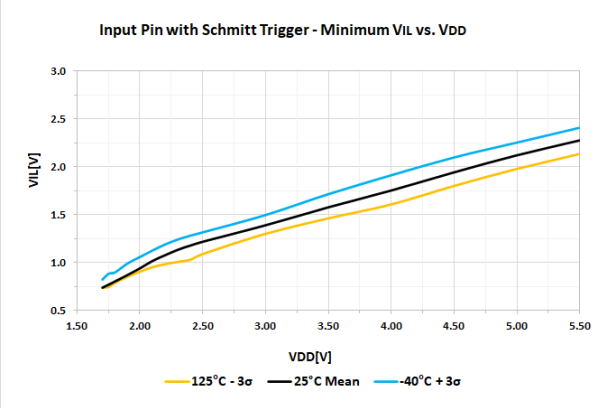



**Figure 48-51.**

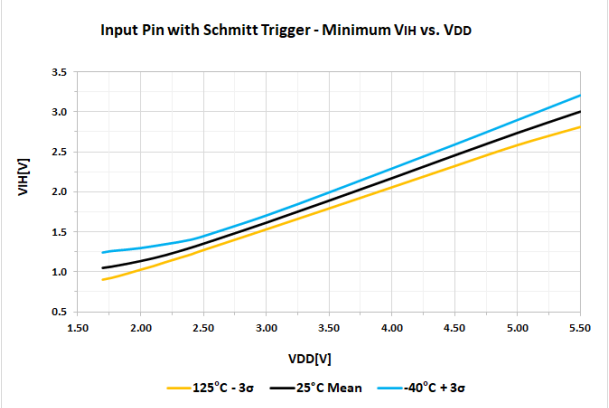



**Figure 48-52.**

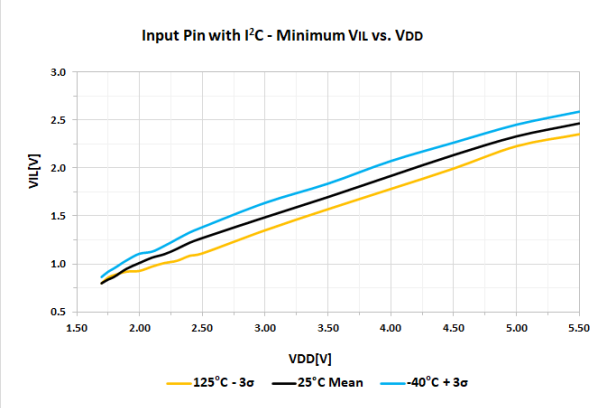



**Figure 48-53.**

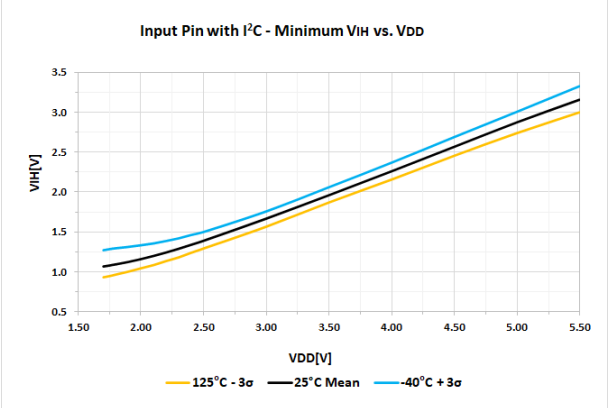


**Figure 48-54.**

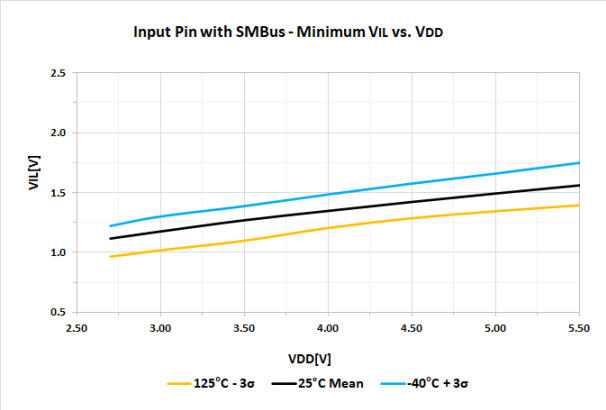



**Figure 48-55.**

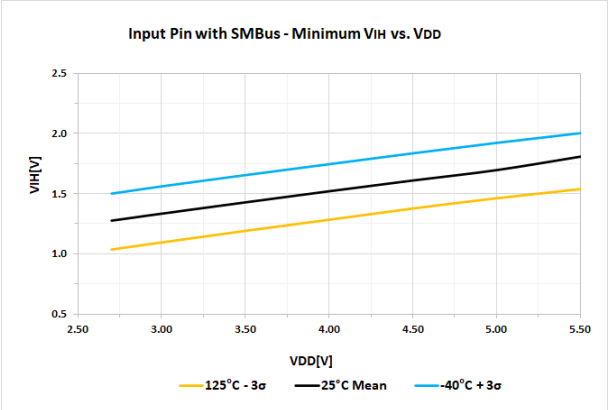



**Figure 48-56.**

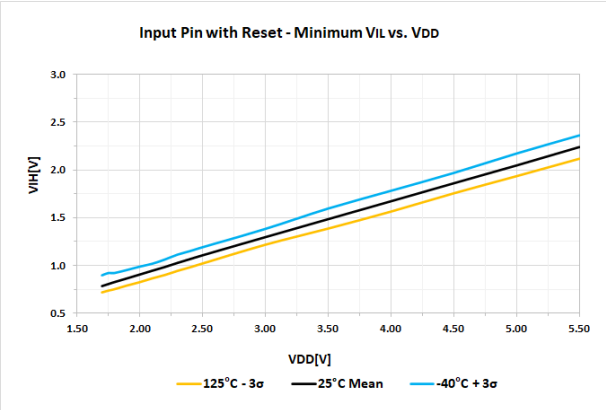



**Figure 48-57.**

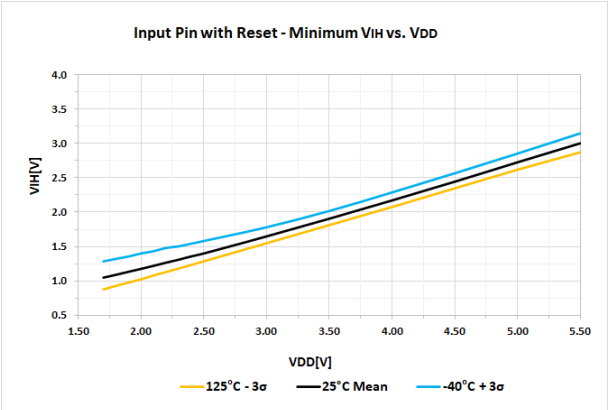



**Figure 48-58.**

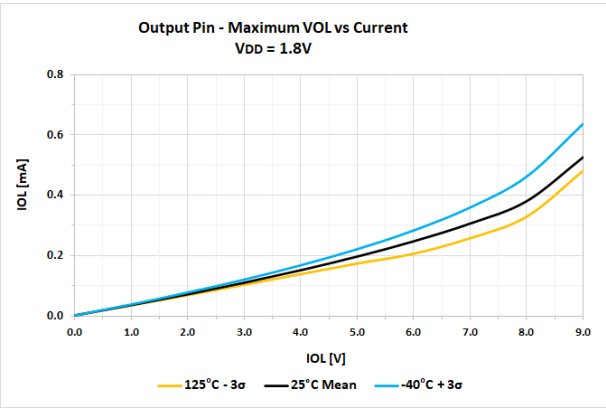



**Figure 48-59.**

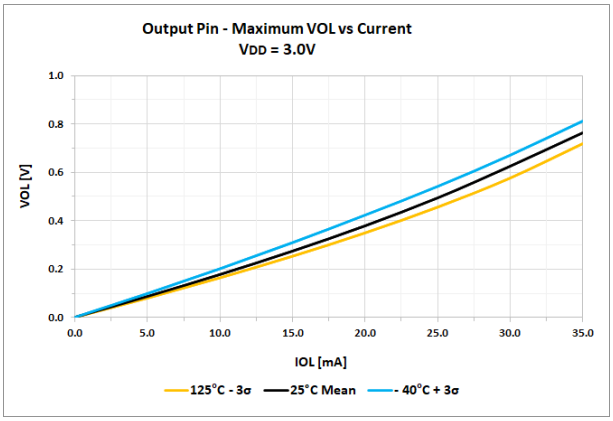


**Figure 48-60.**

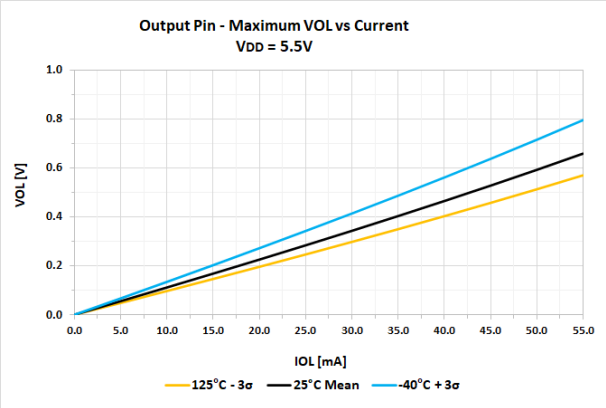



**Figure 48-61.**

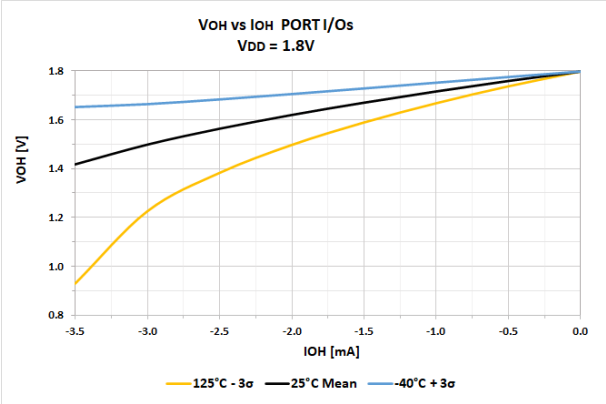



**Figure 48-62.**

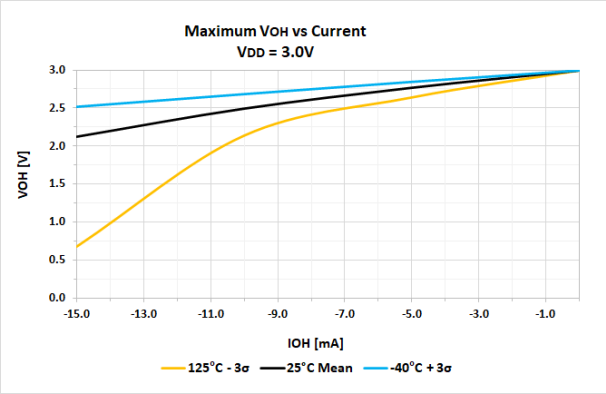
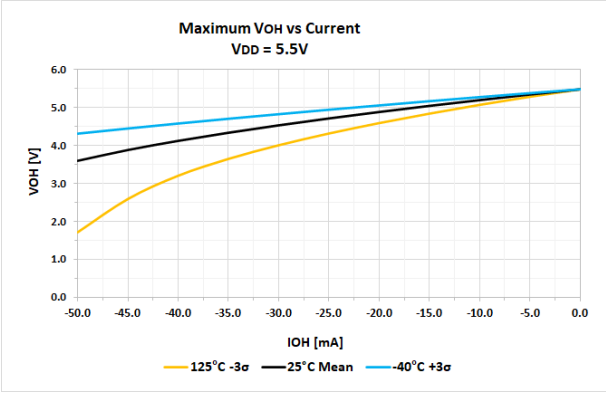



**Figure 48-63.**




## 48.9 $I_{DD}$ Graphs

**Figure 48-64.**

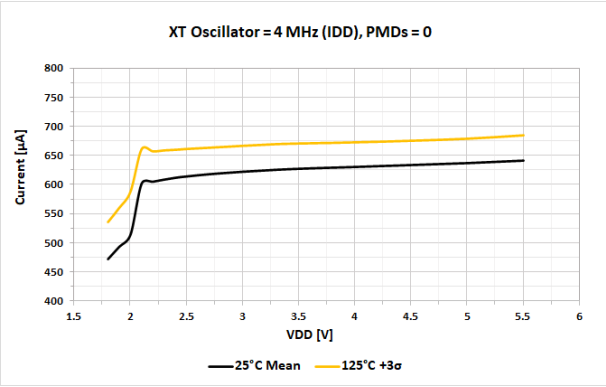



**Figure 48-65.**

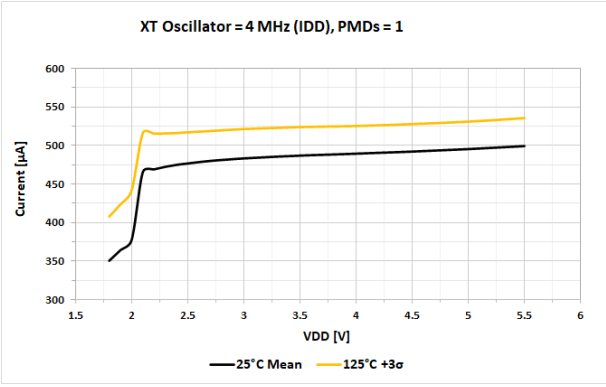


**Figure 48-66.**

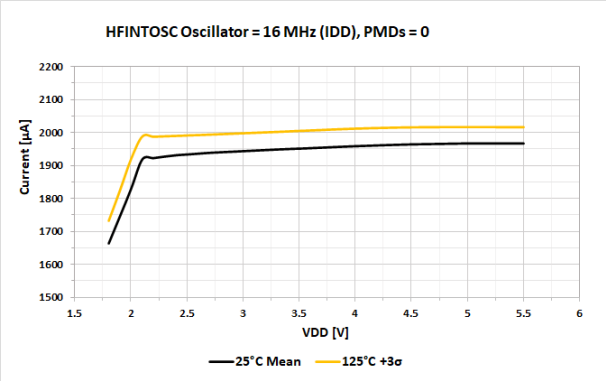



**Figure 48-67.**

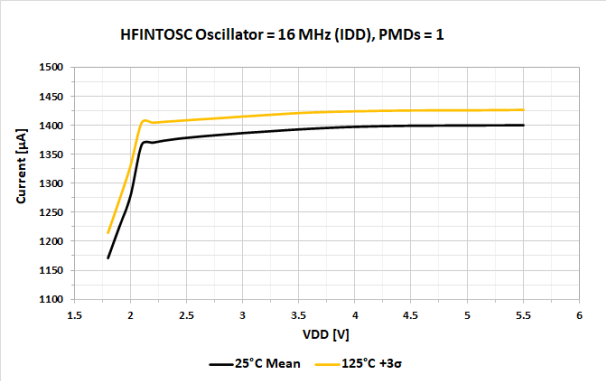



**Figure 48-68.**

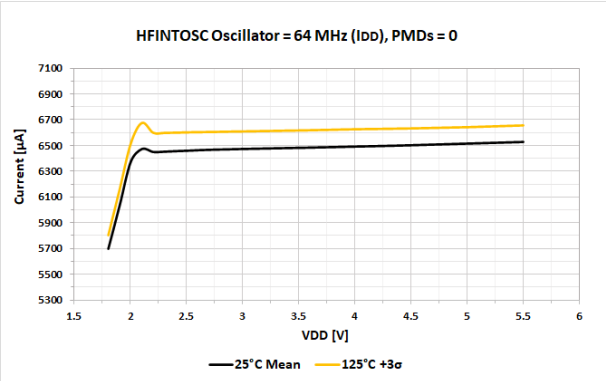



**Figure 48-69.**

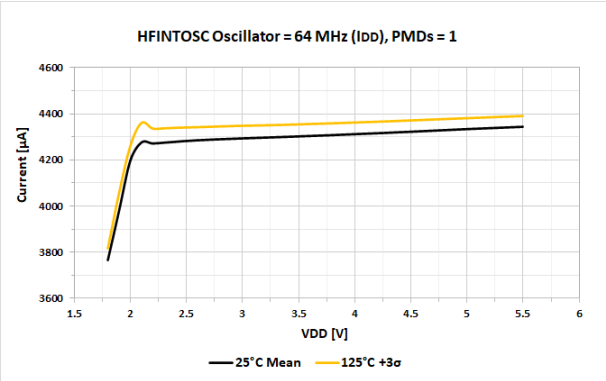



**Figure 48-70.**

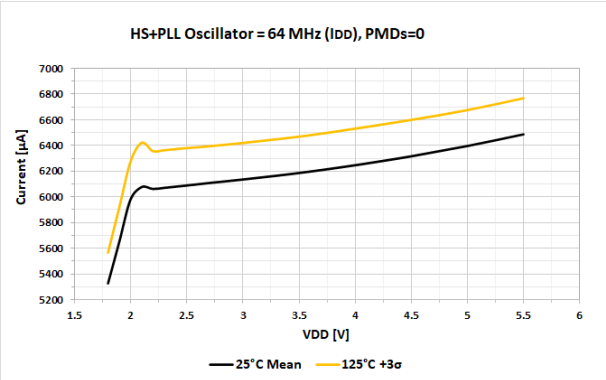



**Figure 48-71.**

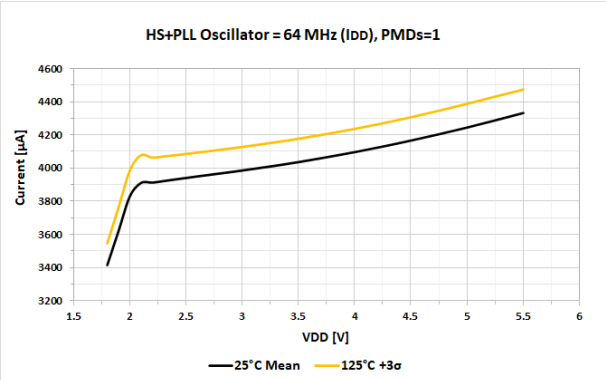


**Figure 48-72.**

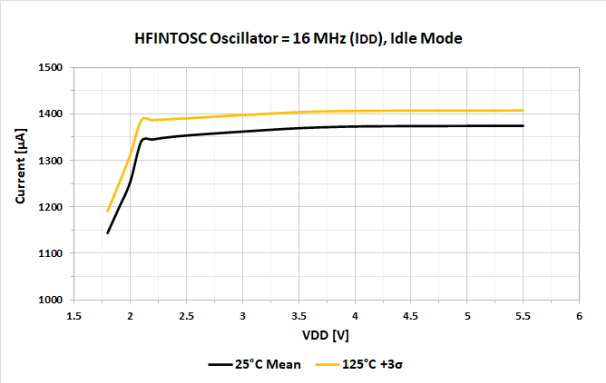
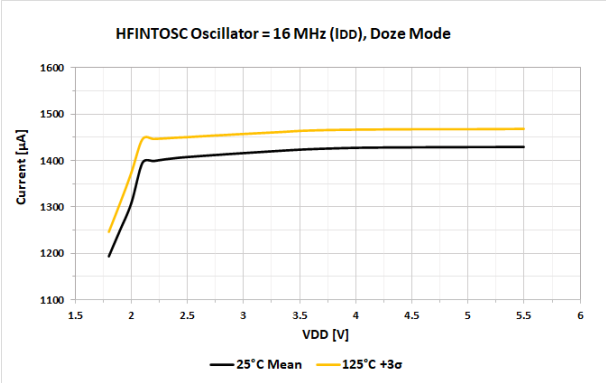



**Figure 48-73.**




## 48.10 $I_{PD}$ Graphs

**Figure 48-74.**

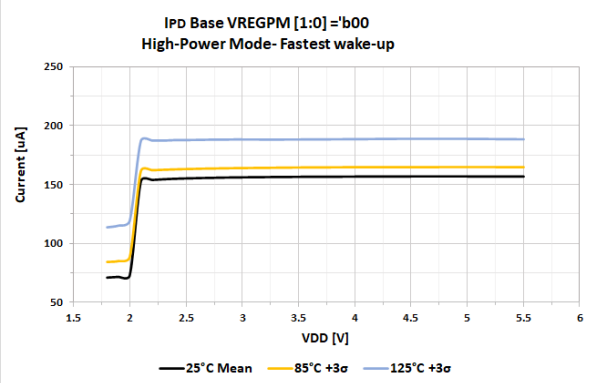



**Figure 48-75.**

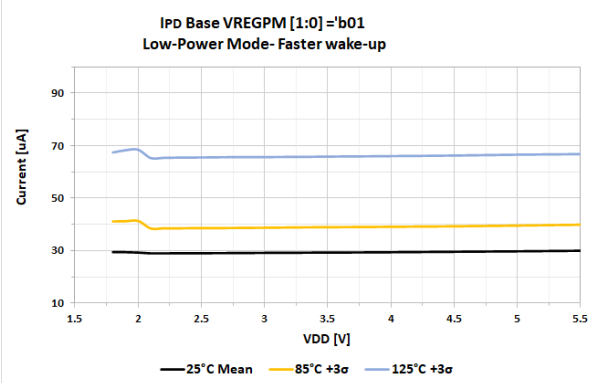



**Figure 48-76.**

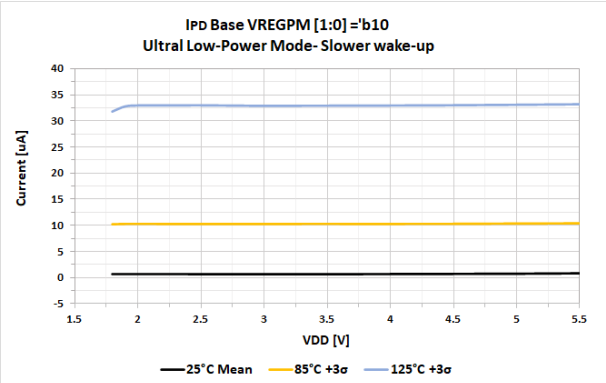



**Figure 48-77.**

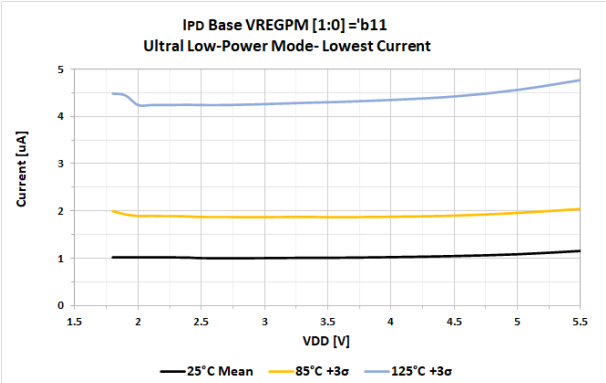


**Figure 48-78.**

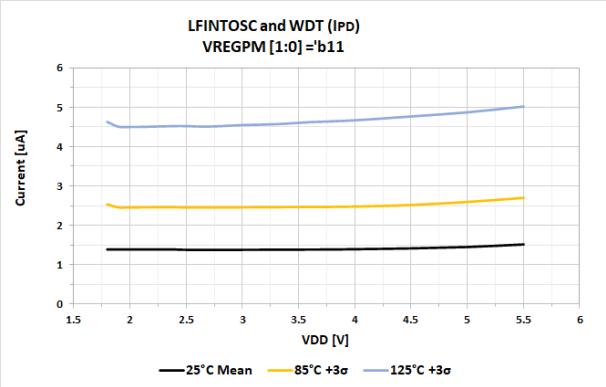



**Figure 48-79.**

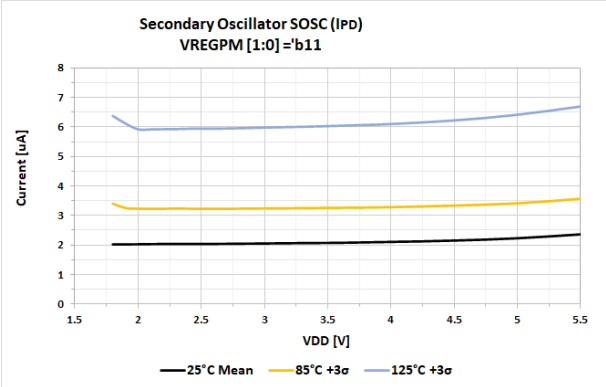



**Figure 48-80.**

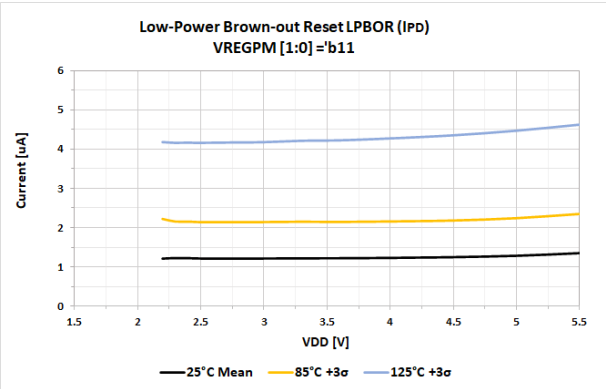



**Figure 48-81.**

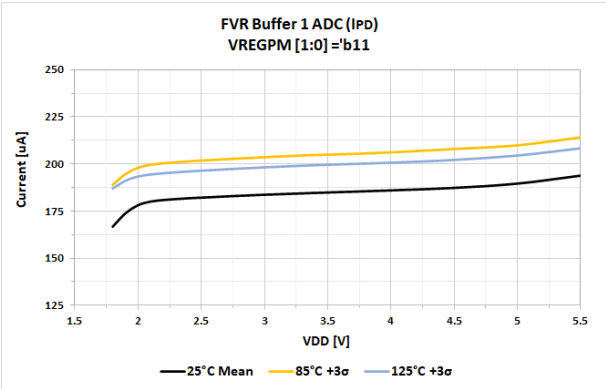



**Figure 48-82.**

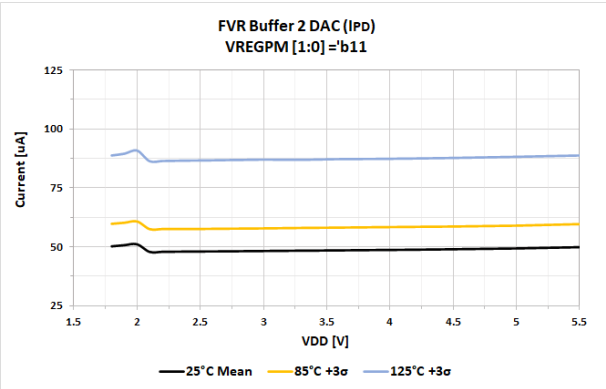



**Figure 48-83.**

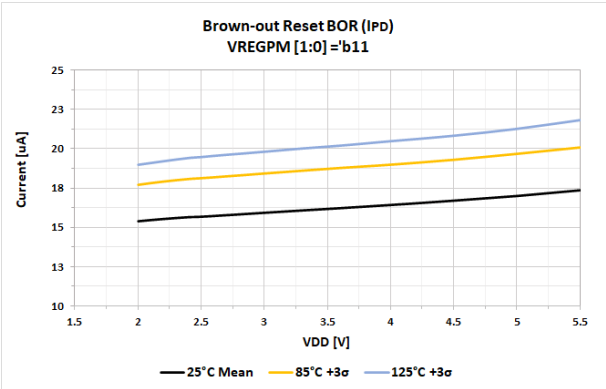


**Figure 48-84.**

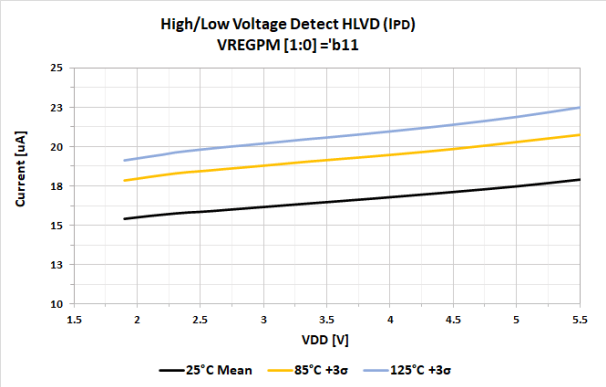



**Figure 48-85.**

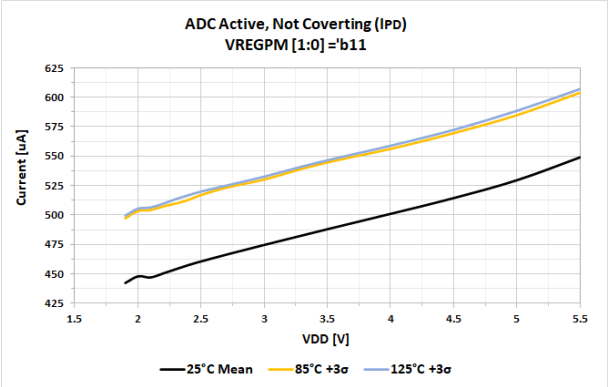
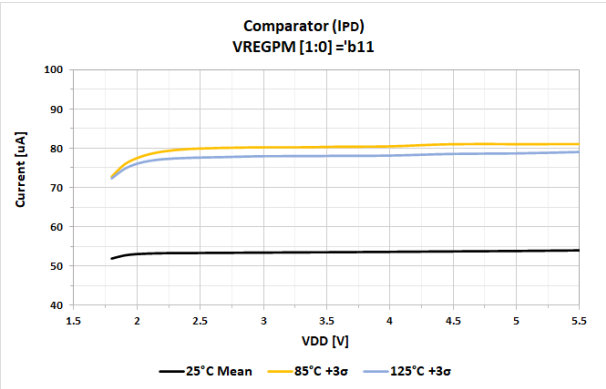



**Figure 48-86.**




## 48.11 LFINTOSC Graphs

**Figure 48-87.**

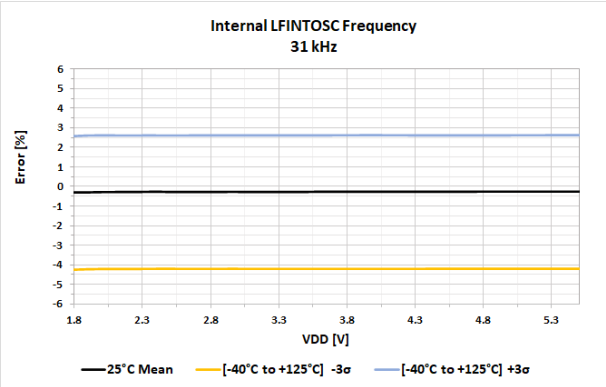
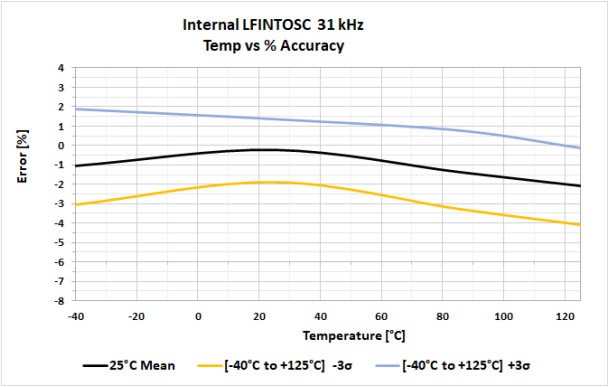



**Figure 48-88.**



## 48.12 LFINTOSC Wake From Sleep Graphs

**Figure 48-89.**

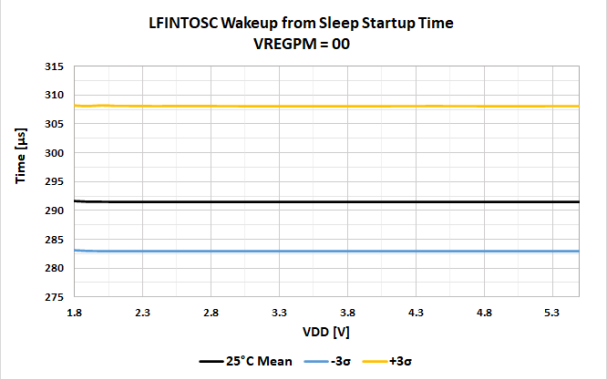



**Figure 48-90.**

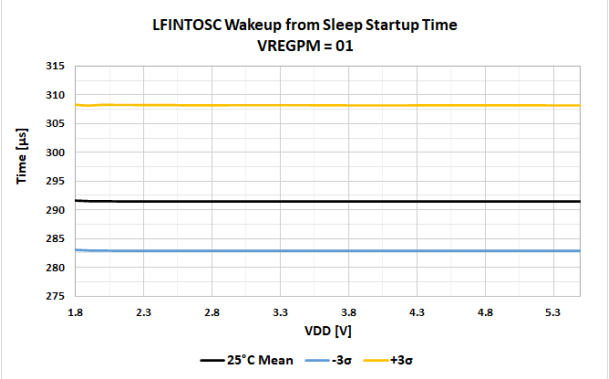



**Figure 48-91.**

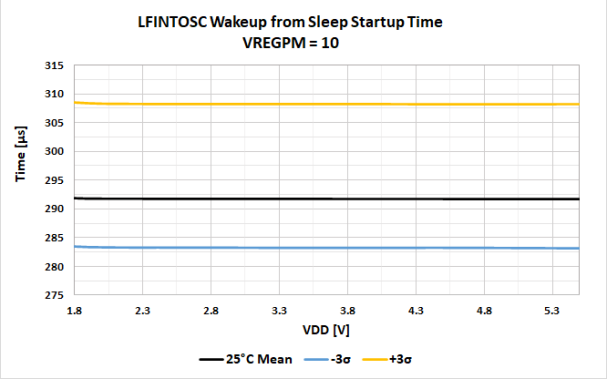
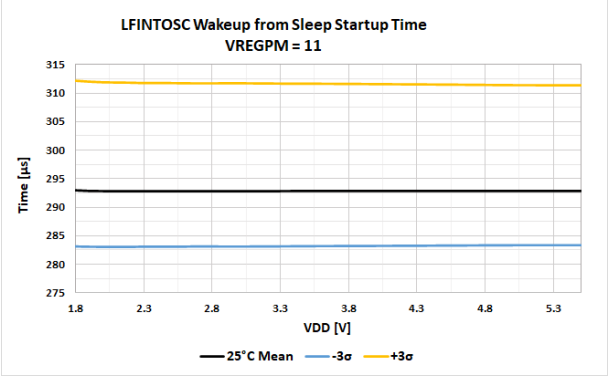



**Figure 48-92.**




## 48.13 Low-Power Brown-Out Reset Graphs

**Figure 48-93.**

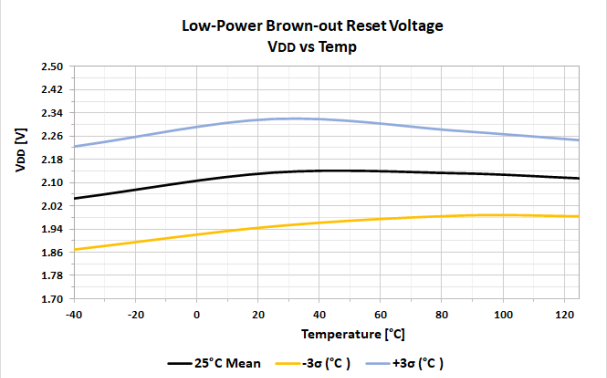
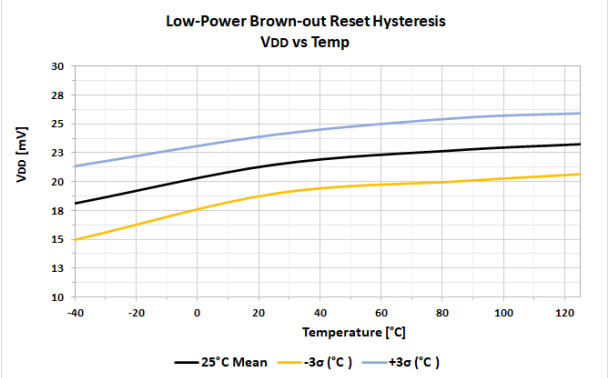



**Figure 48-94.**



## 48.14 MFINTOSC Graphs

**Figure 48-95.**

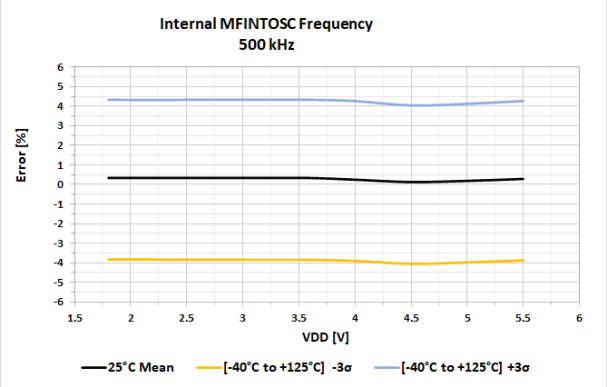
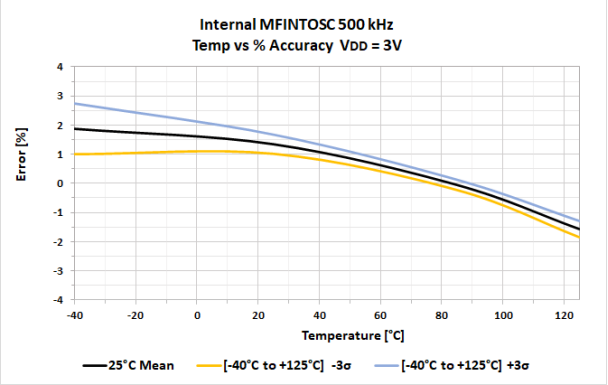



**Figure 48-96.**




## 48.15 Power-On Reset Graphs

**Figure 48-97.**

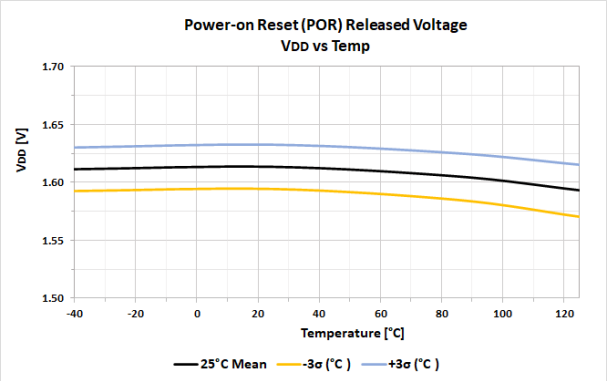



**Figure 48-98.**

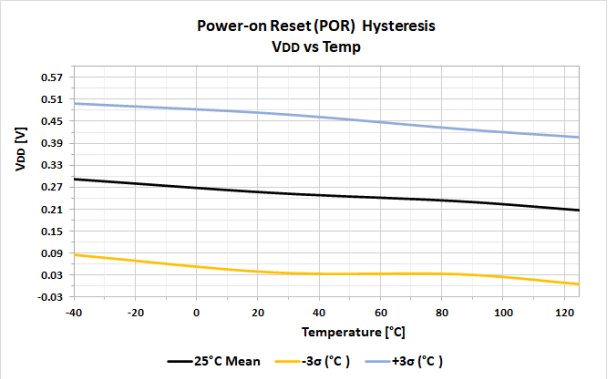
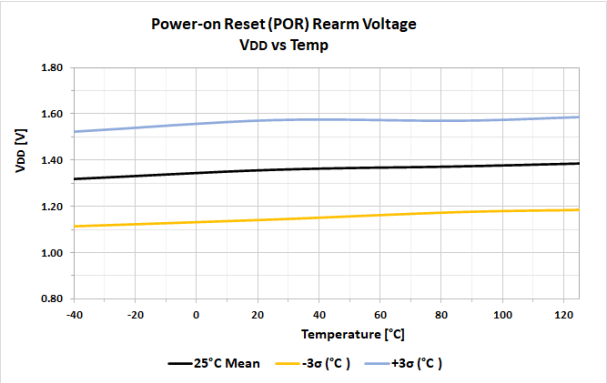



**Figure 48-99.**



## 48.16 Weak Pull-Up Graphs

**Figure 48-100.**

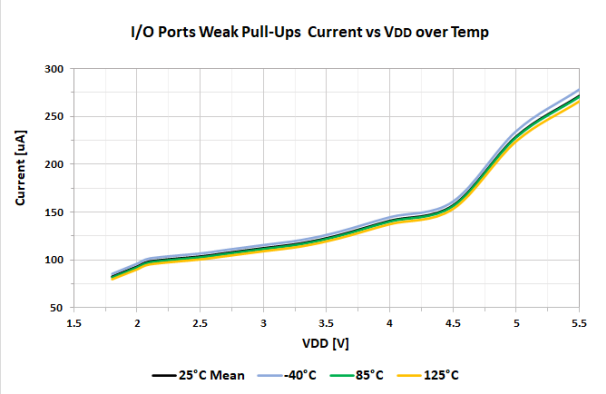



**Figure 48-101.**

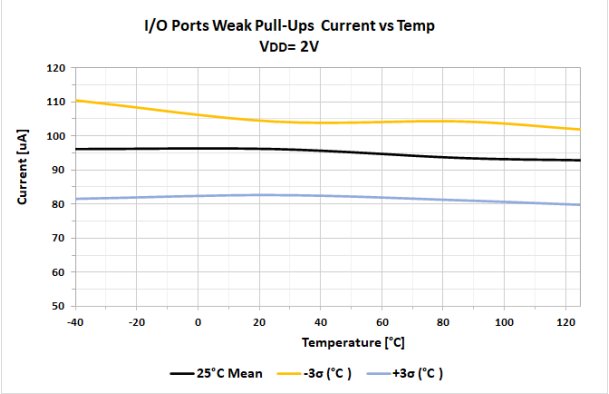



**Figure 48-102.**

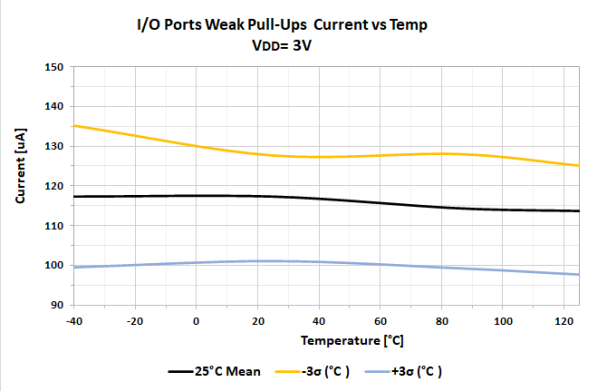
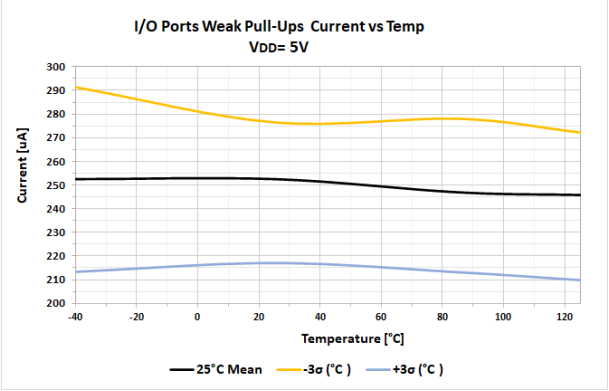



**Figure 48-103.**

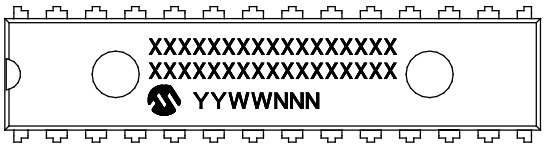


# 49. Packaging Information

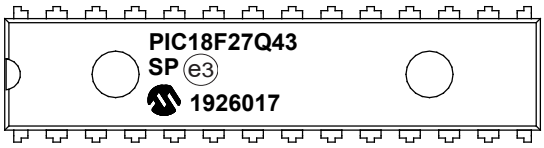
Package Marking Information

| | | |
|---|---|---|
| **Legend:** | XX...X | Customer-specific information or Microchip part number |
| | Y | Year code (last digit of calendar year) |
| | YY | Year code (last 2 digits of calendar year) |
| | WW | Week code (week of January 1 is week ‘01’) |
| | NNN | Alphanumeric traceability code |
| | (e3) | Pb-free JEDEC® designator for Matte Tin (Sn) |
| **Note**: | In the event the full Microchip part number cannot be marked on one line, it will be carried over to the next line, thus limiting the number of available characters for customer-specific information. | |

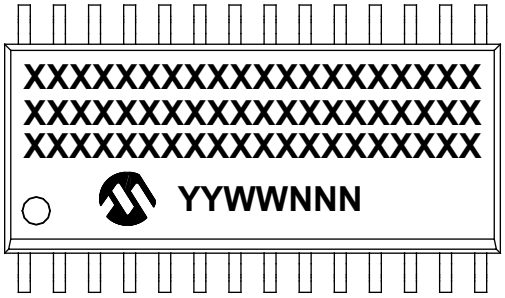
28-Lead SPDIP (.300”)



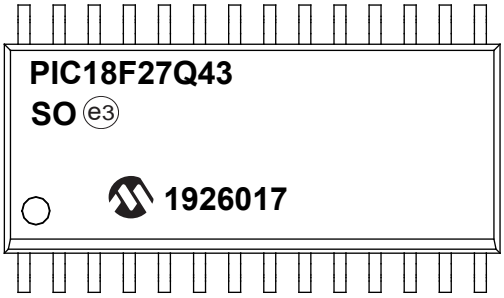

Example



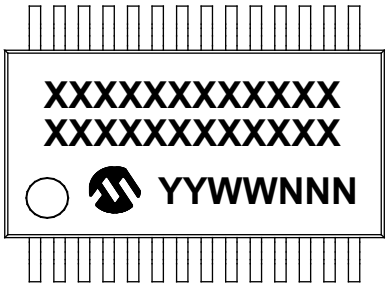

28-Lead SOIC (7.50 mm)



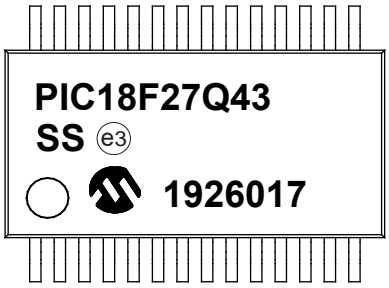

Example



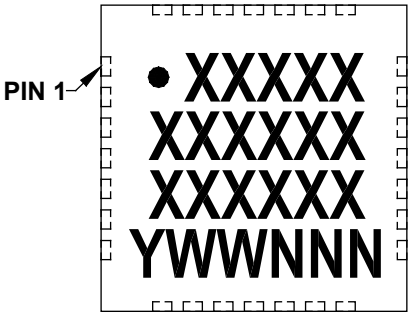
28-Lead SSOP (5.30 mm)



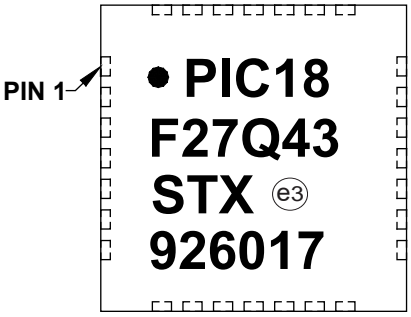

Example



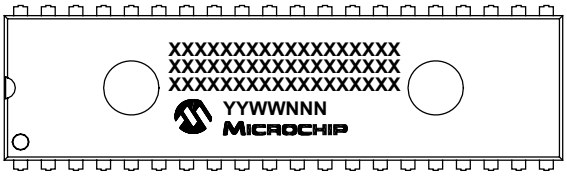

28-Lead VQFN (4x4x1 mm)



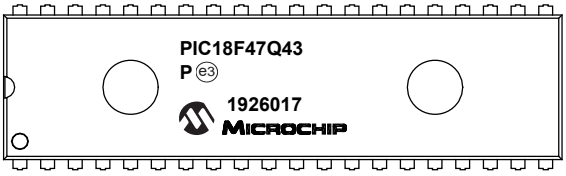

Example



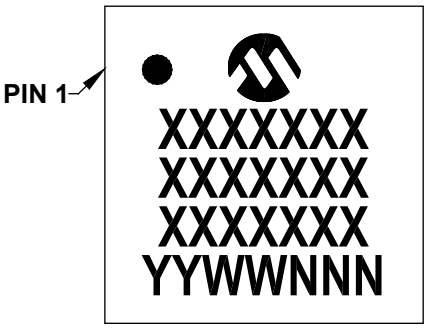

40-Lead PDIP (600 mil)



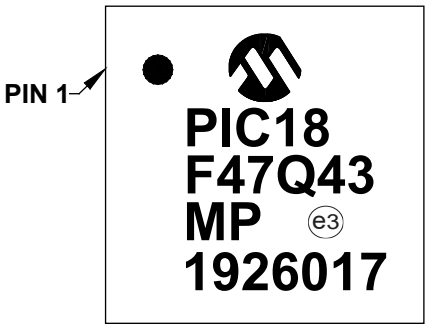

Example



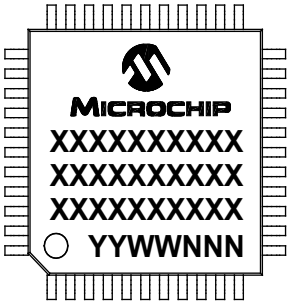
40-Lead QFN (5x5x0.9mm)



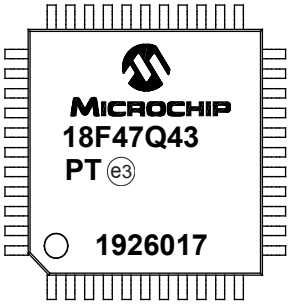

Example



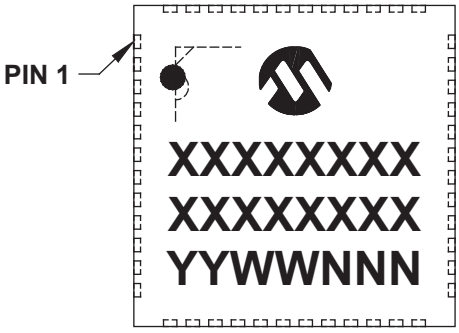

44-Lead TQFP (10x10x1 mm)



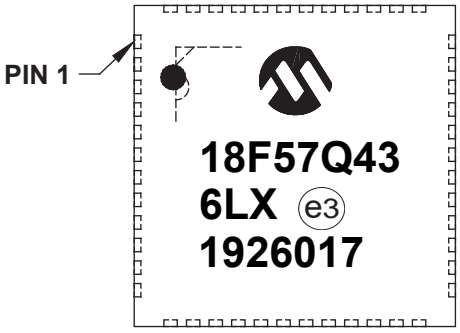

Example




48-Lead VQFN (6x6x0.9 mm)

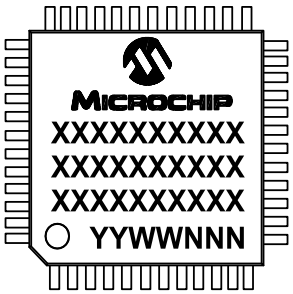



Example

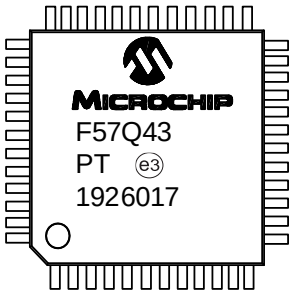



© 2019-2023 Microchip Technology Inc.
and its subsidiaries

48-Lead TQFP (7x7x1 mm)




Example



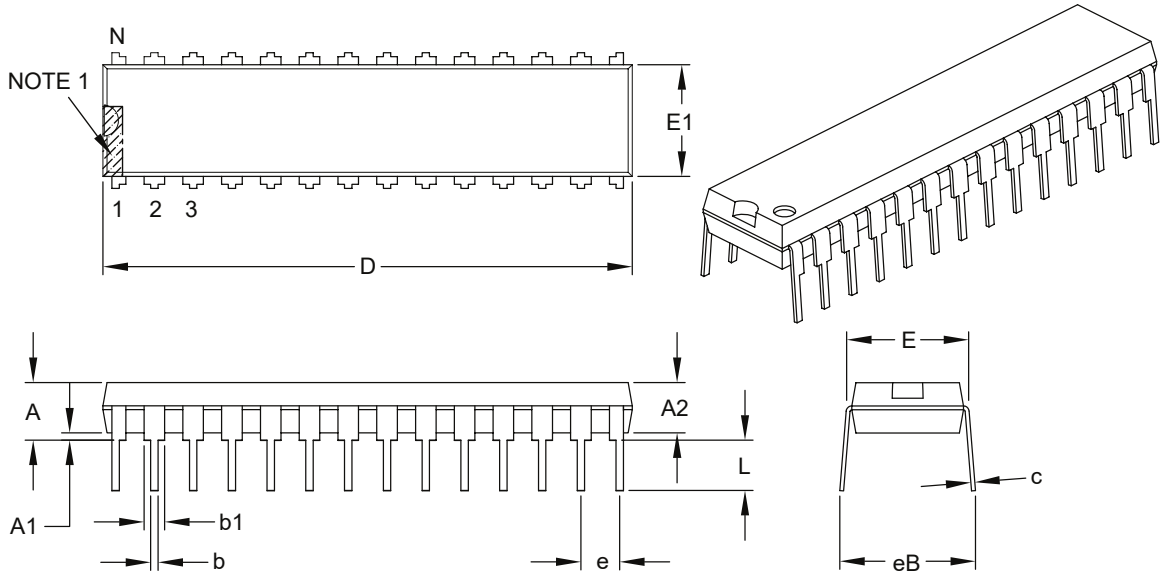

- –

## 49.1 Package Details

The following sections give the technical details of the packages.

### 28-Lead Skinny Plastic Dual In-Line (SP) – 300 mil Body [SPDIP]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging




| Units | | INCHES | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Number of Pins | N | | 28 | |
| Pitch | e | | .100 BSC | |
| Top to Seating Plane | A | – | – | .200 |
| Molded Package Thickness | A2 | .120 | .135 | .150 |
| Base to Seating Plane | A1 | .015 | – | – |
| Shoulder to Shoulder Width | E | .290 | .310 | .335 |
| Molded Package Width | E1 | .240 | .285 | .295 |
| Overall Length | D | 1.345 | 1.365 | 1.400 |
| Tip to Seating Plane | L | .110 | .130 | .150 |
| Lead Thickness | c | .008 | .010 | .015 |
| Upper Lead Width | b1 | .040 | .050 | .070 |
| Lower Lead Width | b | .014 | .018 | .022 |
| Overall Row Spacing § | eB | – | – | .430 |

**Notes:**

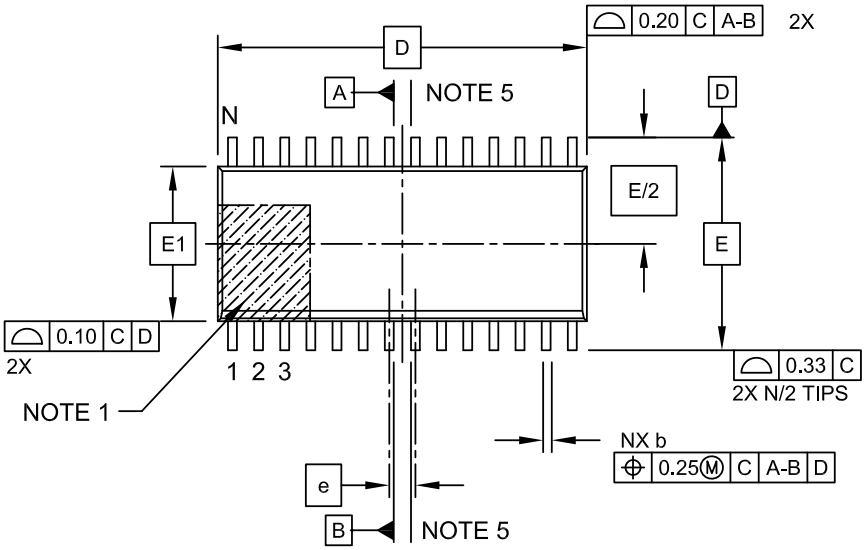
1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. § Significant Characteristic.
3. Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010" per side.
4. Dimensioning and tolerancing per ASME Y14.5M.

   BSC: Basic Dimension. Theoretically exact value shown without tolerances.

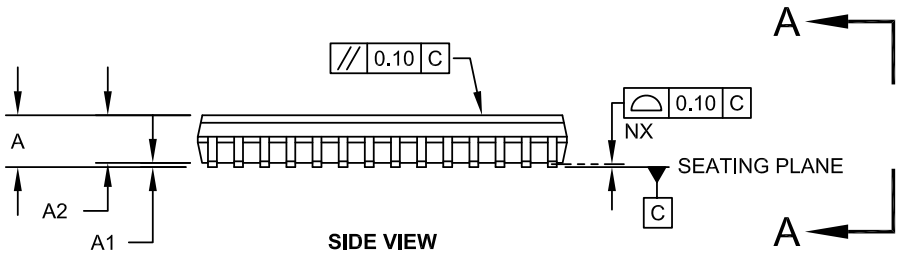
Microchip Technology Drawing C04-070B

### 28-Lead Plastic Small Outline (SO) - Wide, 7.50 mm Body [SOIC]

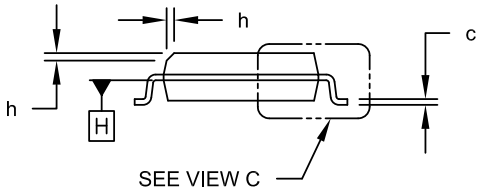
**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging




**TOP VIEW**




**SIDE VIEW**

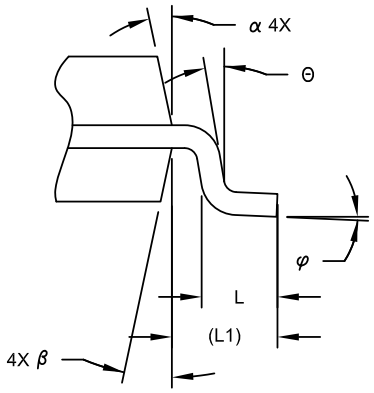



**VIEW A-A**

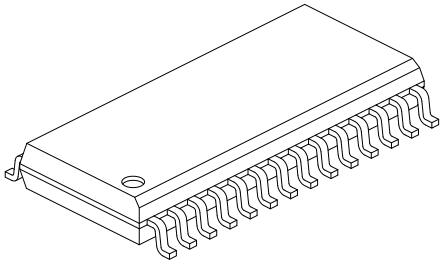
Microchip Technology Drawing C04-052C Sheet 1 of 2

### 28-Lead Plastic Small Outline (SO) - Wide, 7.50 mm Body [SOIC]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging




VIEW C



| Units | | MILLIMETERS | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Number of Pins | N | | 28 | |
| Pitch | e | | 1.27 BSC | |
| Overall Height | A | - | - | 2.65 |
| Molded Package Thickness | A2 | 2.05 | - | - |
| Standoff § | A1 | 0.10 | - | 0.30 |
| Overall Width | E | | 10.30 BSC | |
| Molded Package Width | E1 | | 7.50 BSC | |
| Overall Length | D | | 17.90 BSC | |
| Chamfer (Optional) | h | 0.25 | - | 0.75 |
| Foot Length | L | 0.40 | - | 1.27 |
| Footprint | L1 | | 1.40 REF | |
| Lead Angle | Θ | 0° | - | - |
| Foot Angle | φ | 0° | - | 8° |
| Lead Thickness | c | 0.18 | - | 0.33 |
| Lead Width | b | 0.31 | - | 0.51 |
| Mold Draft Angle Top | α | 5° | - | 15° |
| Mold Draft Angle Bottom | β | 5° | - | 15° |

**Notes:**

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. § Significant Characteristic
3. Dimension D does not include mold flash, protrusions or gate burrs, which shall not exceed 0.15 mm per end. Dimension E1 does not include interlead flash or protrusion, which shall not exceed 0.25 mm per side.
4. Dimensioning and tolerancing per ASME Y14.5M

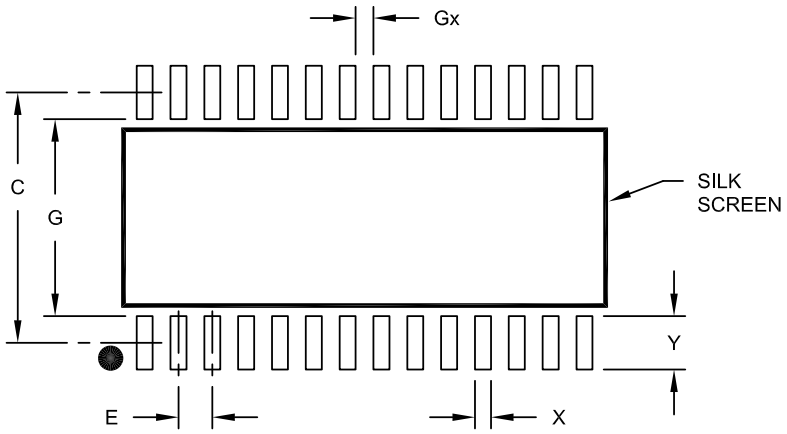
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.

   REF: Reference Dimension, usually without tolerance, for information purposes only.
5. Datums A & B to be determined at Datum H.

Microchip Technology Drawing C04-052C Sheet 2 of 2

28-Lead Plastic Small Outline (SO) - Wide, 7.50 mm Body [SOIC]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging




RECOMMENDED LAND PATTERN

| Units | | MILLIMETERS | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Contact Pitch | E | | 1.27 BSC | |
| Contact Pad Spacing | C | | 9.40 | |
| Contact Pad Width (X28) | X | | | 0.60 |
| Contact Pad Length (X28) | Y | | | 2.00 |
| Distance Between Pads | Gx | 0.67 | | |
| Distance Between Pads | G | 7.40 | | |

Notes:

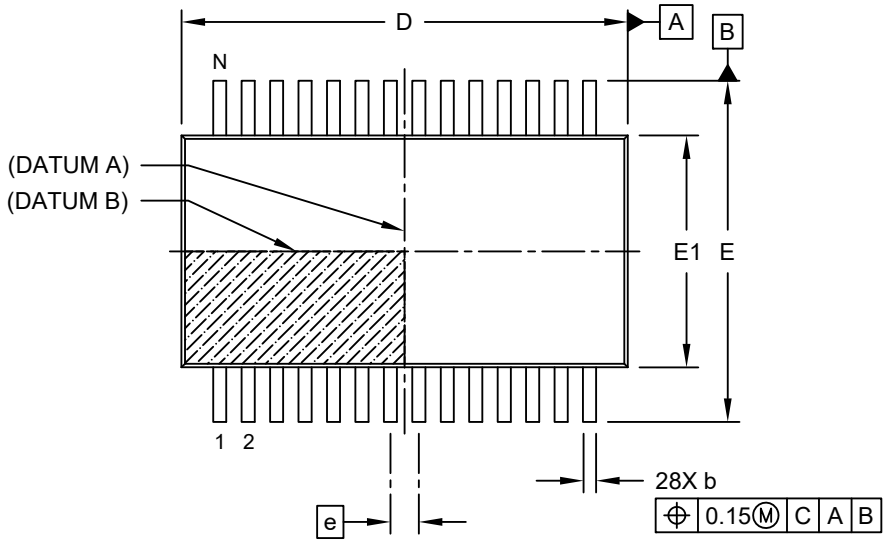
1. Dimensioning and tolerancing per ASME Y14.5M

BSC: Basic Dimension. Theoretically exact value shown without tolerances.

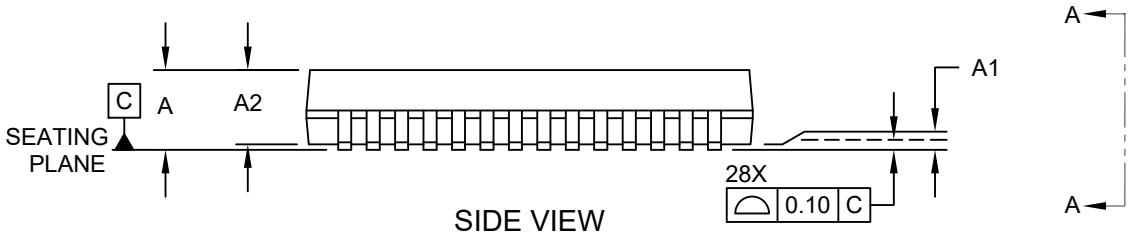
Microchip Technology Drawing No. C04-2052A

### 28-Lead Plastic Shrink Small Outline (SS) - 5.30 mm Body [SSOP]

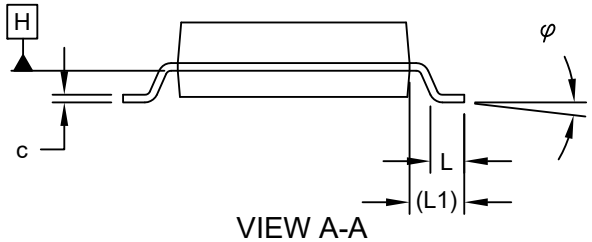
**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging




TOP VIEW




SIDE VIEW

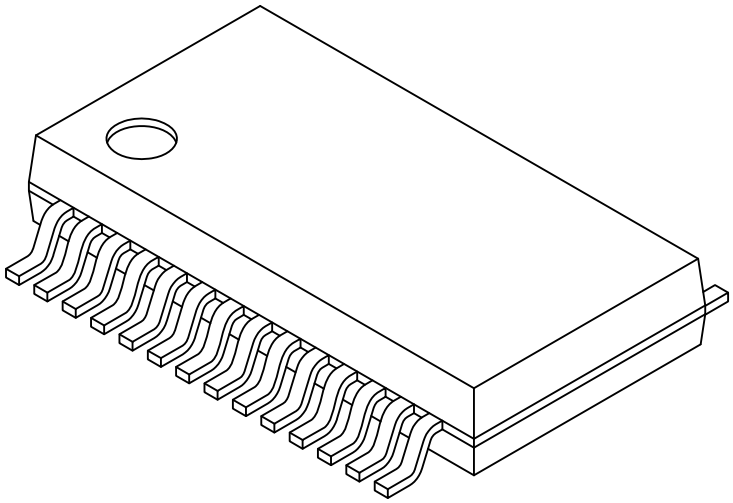



VIEW A-A

Microchip Technology Drawing C04-073 Rev C Sheet 1 of 2

## 28-Lead Plastic Shrink Small Outline (SS) - 5.30 mm Body [SSOP]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



| Units | | MILLIMETERS | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Number of Pins | N | | 28 | |
| Pitch | e | | 0.65 BSC | |
| Overall Height | A | - | - | 2.00 |
| Molded Package Thickness | A2 | 1.65 | 1.75 | 1.85 |
| Standoff | A1 | 0.05 | - | - |
| Overall Width | E | 7.40 | 7.80 | 8.20 |
| Molded Package Width | E1 | 5.00 | 5.30 | 5.60 |
| Overall Length | D | 9.90 | 10.20 | 10.50 |
| Foot Length | L | 0.55 | 0.75 | 0.95 |
| Footprint | L1 | | 1.25 REF | |
| Lead Thickness | c | 0.09 | - | 0.25 |
| Foot Angle | φ | 0° | 4° | 8° |
| Lead Width | b | 0.22 | - | 0.38 |

Notes:

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed 0.20mm per side.
3. Dimensioning and tolerancing per ASME Y14.5M

   BSC: Basic Dimension. Theoretically exact value shown without tolerances.

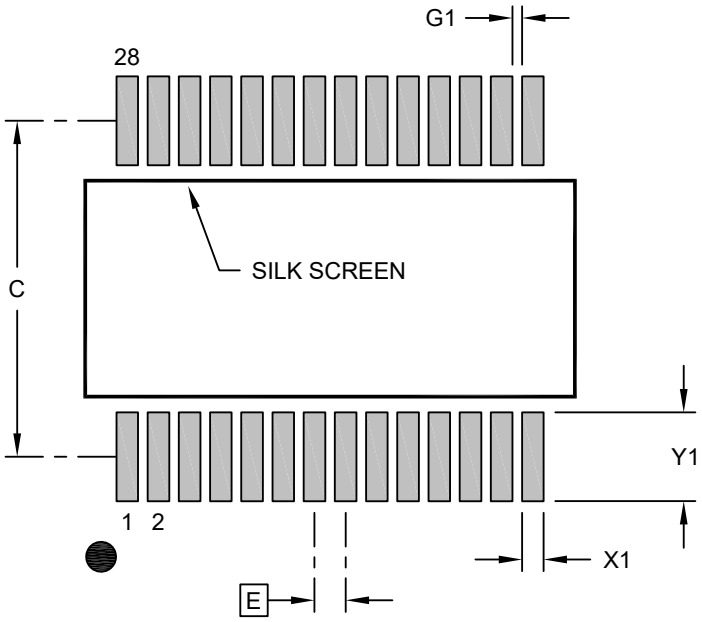
   REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-073 Rev C Sheet 2 of 2

© 2019-2023 Microchip Technology Inc. and its subsidiaries

## 28-Lead Plastic Shrink Small Outline (SS) - 5.30 mm Body [SSOP]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging




RECOMMENDED LAND PATTERN

| Units | | MILLIMETERS | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Contact Pitch | E | 0.65 BSC | | |
| Contact Pad Spacing | C | | 7.00 | |
| Contact Pad Width (X28) | X1 | | | 0.45 |
| Contact Pad Length (X28) | Y1 | | | 1.85 |
| Contact Pad to Center Pad (X26) | G1 | 0.20 | | |

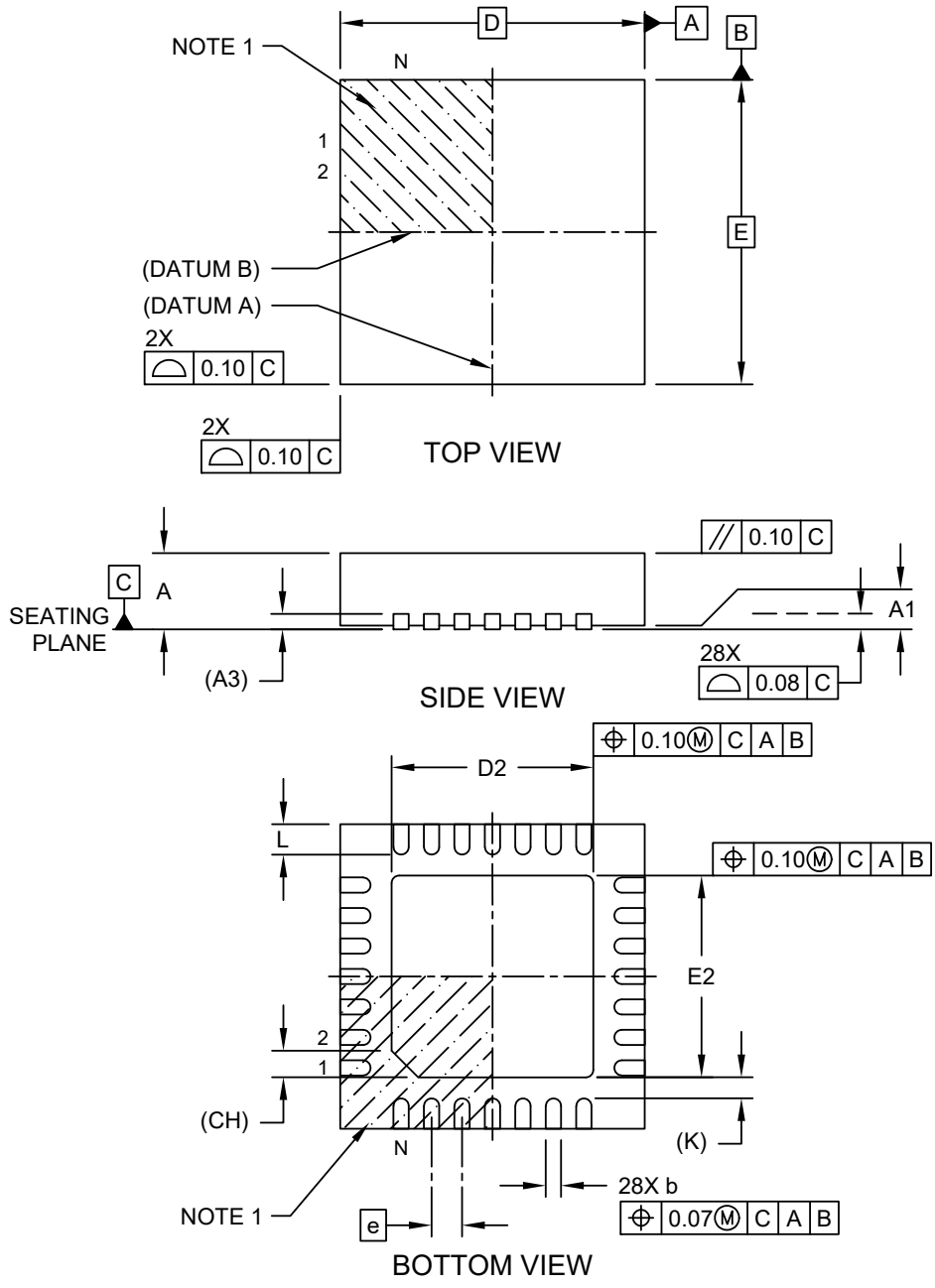
Notes:

1. Dimensioning and tolerancing per ASME Y14.5M
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.
2. For best soldering results, thermal vias, if used, should be filled or tented to avoid solder loss during reflow process

Microchip Technology Drawing C04-2073 Rev B

**28-Lead Very Thin Plastic Quad Flat, No Lead (STX) - 4x4x1.0 mm Body [VQFN]**
**With 2.65x2.65 mm Exposed Pad**

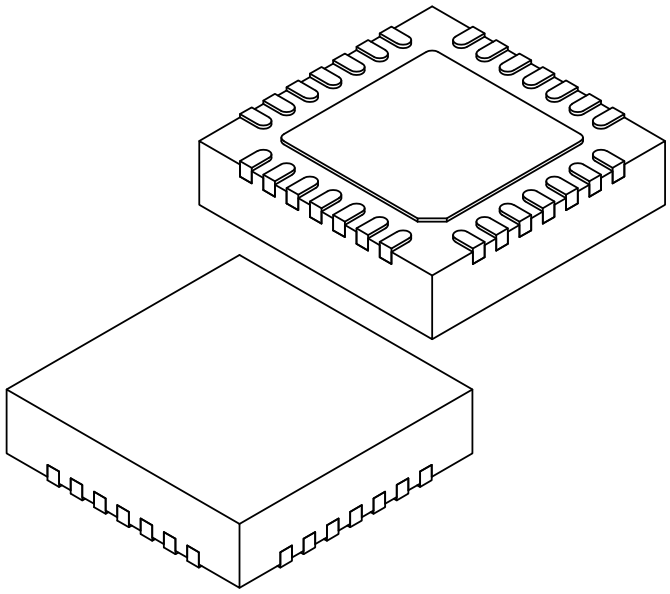
**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging




Microchip Technology Drawing C04-456 Rev C Sheet 1 of 2

### 28-Lead Very Thin Plastic Quad Flat, No Lead (STX) - 4x4x1.0 mm Body [VQFN] With 2.65x2.65 mm Exposed Pad

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



| Units | | MILLIMETERS | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Number of Terminals | N | | 28 | |
| Pitch | e | | 0.40 BSC | |
| Overall Height | A | 0.80 | 0.90 | 1.00 |
| Standoff | A1 | 0.00 | 0.02 | 0.05 |
| Terminal Thickness | A3 | | 0.203 REF | |
| Overall Length | D | | 4.00 BSC | |
| Exposed Pad Length | D2 | 2.55 | 2.65 | 2.75 |
| Overall Width | E | | 4.00 BSC | |
| Exposed Pad Width | E2 | 2.55 | 2.65 | 2.75 |
| Exposed Pad Corner Chamfer | CH | | 0.35 REF | |
| Terminal Width | b | 0.15 | 0.20 | 0.25 |
| Terminal Length | L | 0.30 | 0.40 | 0.50 |
| Terminal-to-Exposed-Pad | K | | 0.275 REF | |

Notes:

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Package is saw singulated
3. Dimensioning and tolerancing per ASME Y14.5M

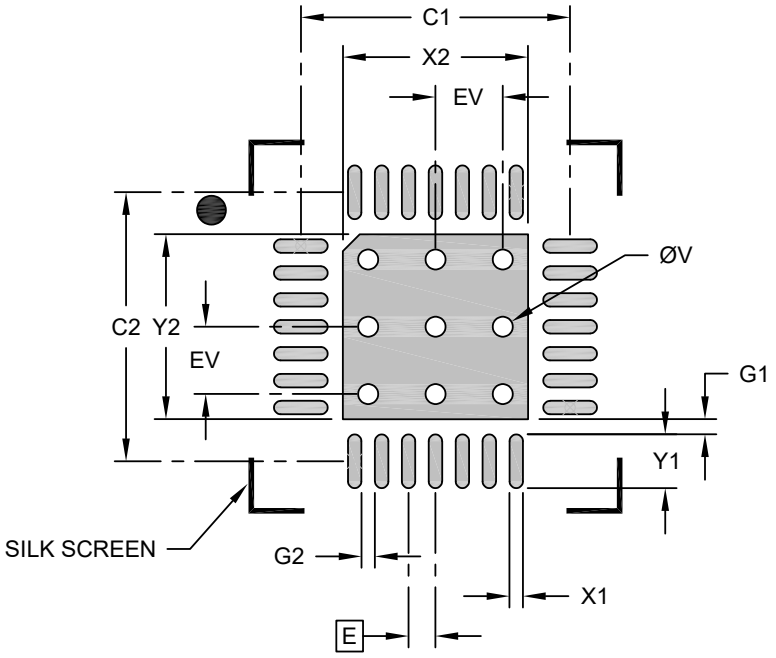
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.

   REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-456 Rev C Sheet 2 of 2

### 28-Lead Very Thin Plastic Quad Flat, No Lead (STX) - 4x4x1.0 mm Body [VQFN] With 2.65x2.65 mm Exposed Pad

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging




RECOMMENDED LAND PATTERN

| Units | | MILLIMETERS | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Contact Pitch | E | 0.40 BSC | | |
| Optional Center Pad Width | X2 | | | 2.75 |
| Optional Center Pad Length | Y2 | | | 2.75 |
| Contact Pad Spacing | C1 | | 4.00 | |
| Contact Pad Spacing | C2 | | 4.00 | |
| Contact Pad Width (X28) | X1 | | | 0.20 |
| Contact Pad Length (X28) | Y1 | | | 0.80 |
| Contact Pad to Center Pad (X28) | G1 | 0.23 | | |
| Contact Pad to Contact Pad (X24) | G2 | 0.20 | | |
| Thermal Via Diameter | V | | 0.30 | |
| Thermal Via Pitch | EV | | 1.00 | |

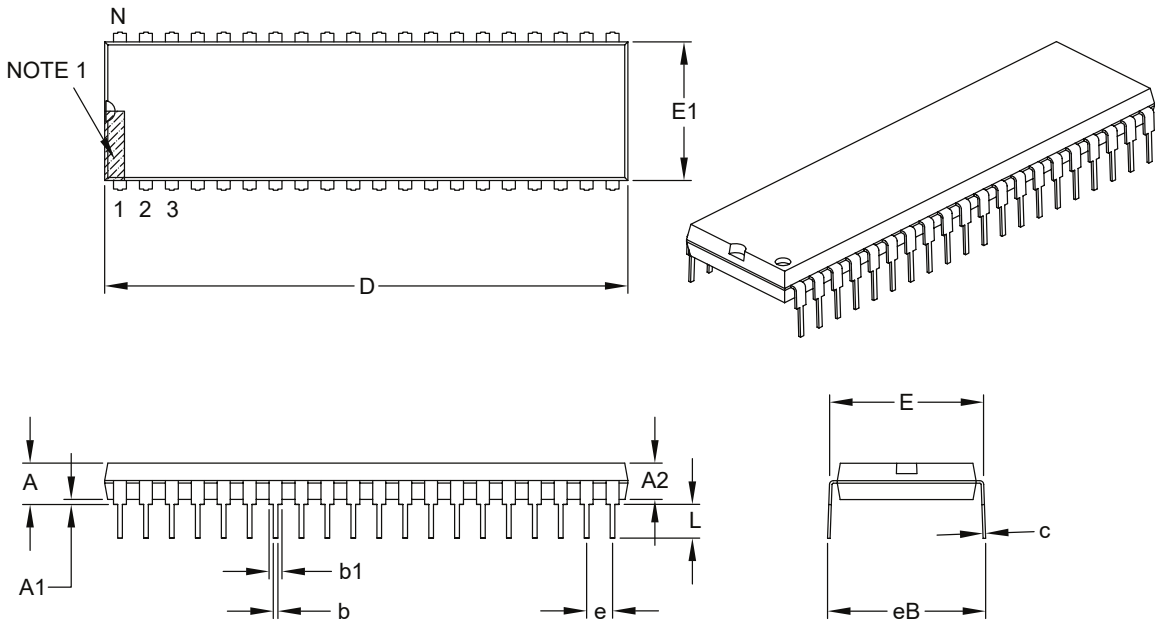
Notes:

1. Dimensioning and tolerancing per ASME Y14.5M
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.
2. For best soldering results, thermal vias, if used, should be filled or tented to avoid solder loss during reflow process

Microchip Technology Drawing C04-2456 Rev C

### 40-Lead Plastic Dual In-Line (P) – 600 mil Body [PDIP]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging




| Units | | INCHES | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Number of Pins | N | | 40 | |
| Pitch | e | | .100 BSC | |
| Top to Seating Plane | A | – | – | .250 |
| Molded Package Thickness | A2 | .125 | – | .195 |
| Base to Seating Plane | A1 | .015 | – | – |
| Shoulder to Shoulder Width | E | .590 | – | .625 |
| Molded Package Width | E1 | .485 | – | .580 |
| Overall Length | D | 1.980 | – | 2.095 |
| Tip to Seating Plane | L | .115 | – | .200 |
| Lead Thickness | c | .008 | – | .015 |
| Upper Lead Width | b1 | .030 | – | .070 |
| Lower Lead Width | b | .014 | – | .023 |
| Overall Row Spacing § | eB | – | – | .700 |

**Notes:**

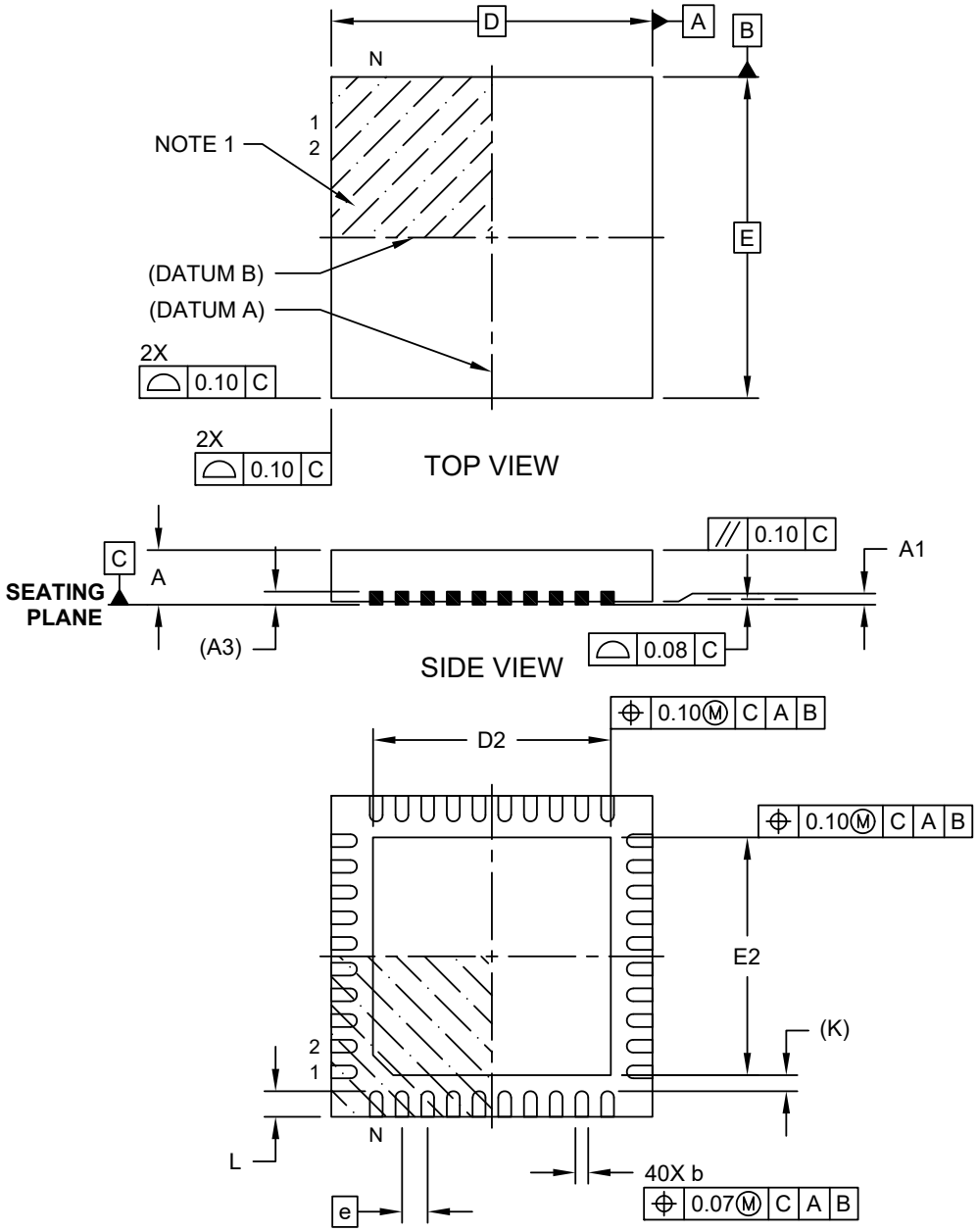
1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. § Significant Characteristic.
3. Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010" per side.
4. Dimensioning and tolerancing per ASME Y14.5M.

BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing C04-016B

**40-Lead Plastic Quad Flat, No Lead Package (MP) - 5x5 mm Body [QFN]**
**With 3.7x3.7 mm Exposed Pad**

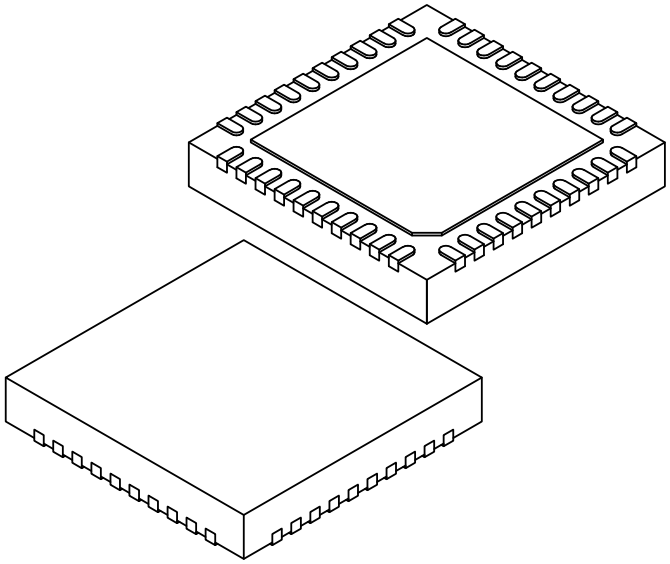
**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging




Microchip Technology Drawing C04-047-MP Rev C Sheet 1 of 2

### 40-Lead Plastic Quad Flat, No Lead Package (MP) - 5x5 mm Body [QFN]
### With 3.7x3.7 mm Exposed Pad

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



| Units | | MILLIMETERS | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Number of Terminals | N | | 40 | |
| Pitch | e | | 0.40 BSC | |
| Overall Height | A | 0.80 | 0.90 | 1.00 |
| Standoff | A1 | 0.00 | 0.02 | 0.05 |
| Terminal Thickness | A3 | | 0.20 REF | |
| Overall Width | E | | 5.00 BSC | |
| Exposed Pad Width | E2 | 3.60 | 3.70 | 3.80 |
| Overall Length | D | | 5.00 BSC | |
| Exposed Pad Length | D2 | 3.60 | 3.70 | 3.80 |
| Terminal Width | b | 0.15 | 0.20 | 0.25 |
| Terminal Length | L | 0.30 | 0.40 | 0.50 |
| Terminal-to-Exposed-Pad | K | | 0.25 REF | |

Notes:

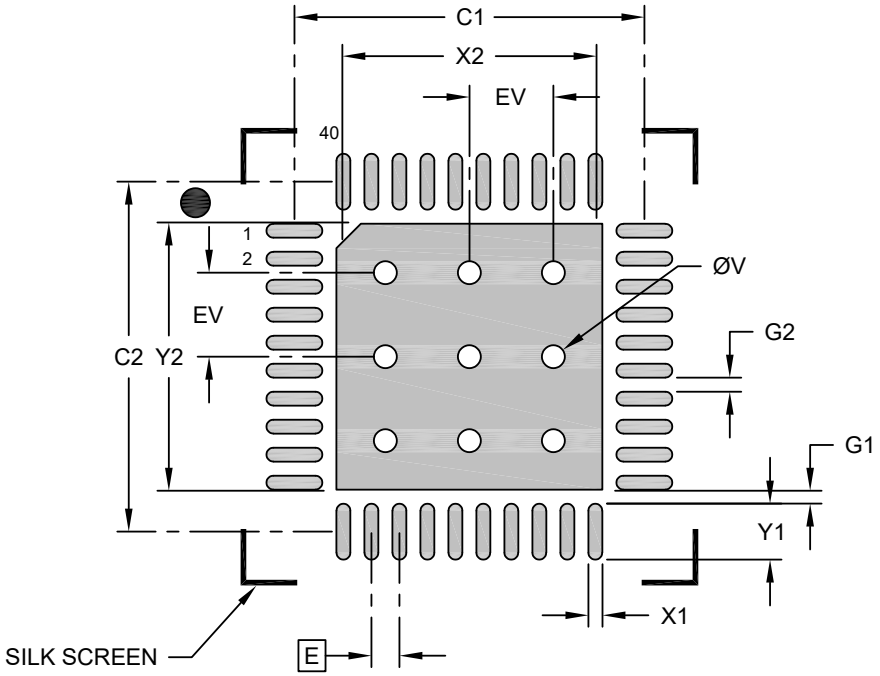
1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Package is saw singulated
3. Dimensioning and tolerancing per ASME Y14.5M

BSC: Basic Dimension. Theoretically exact value shown without tolerances.
REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-047-MP Rev B Sheet 2 of 2

### 40-Lead Plastic Quad Flat, No Lead Package (MP) - 5x5 mm Body [QFN] With 3.7x3.7 mm Exposed Pad

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging




RECOMMENDED LAND PATTERN

| Units | | MILLIMETERS | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Contact Pitch | E | 0.40 BSC | | |
| Optional Center Pad Width | X2 | | | 3.80 |
| Optional Center Pad Length | Y2 | | | 3.80 |
| Contact Pad Spacing | C1 | | 5.00 | |
| Contact Pad Spacing | C2 | | 5.00 | |
| Contact Pad Width (X40) | X1 | | | 0.20 |
| Contact Pad Length (X40) | Y1 | | | 0.80 |
| Contact Pad to Center Pad (X40) | G1 | 0.20 | | |
| Contact Pad to Contact Pad (X36) | G2 | 0.20 | | |
| Thermal Via Diameter | V | | 0.33 | |
| Thermal Via Pitch | EV | | 1.20 | |

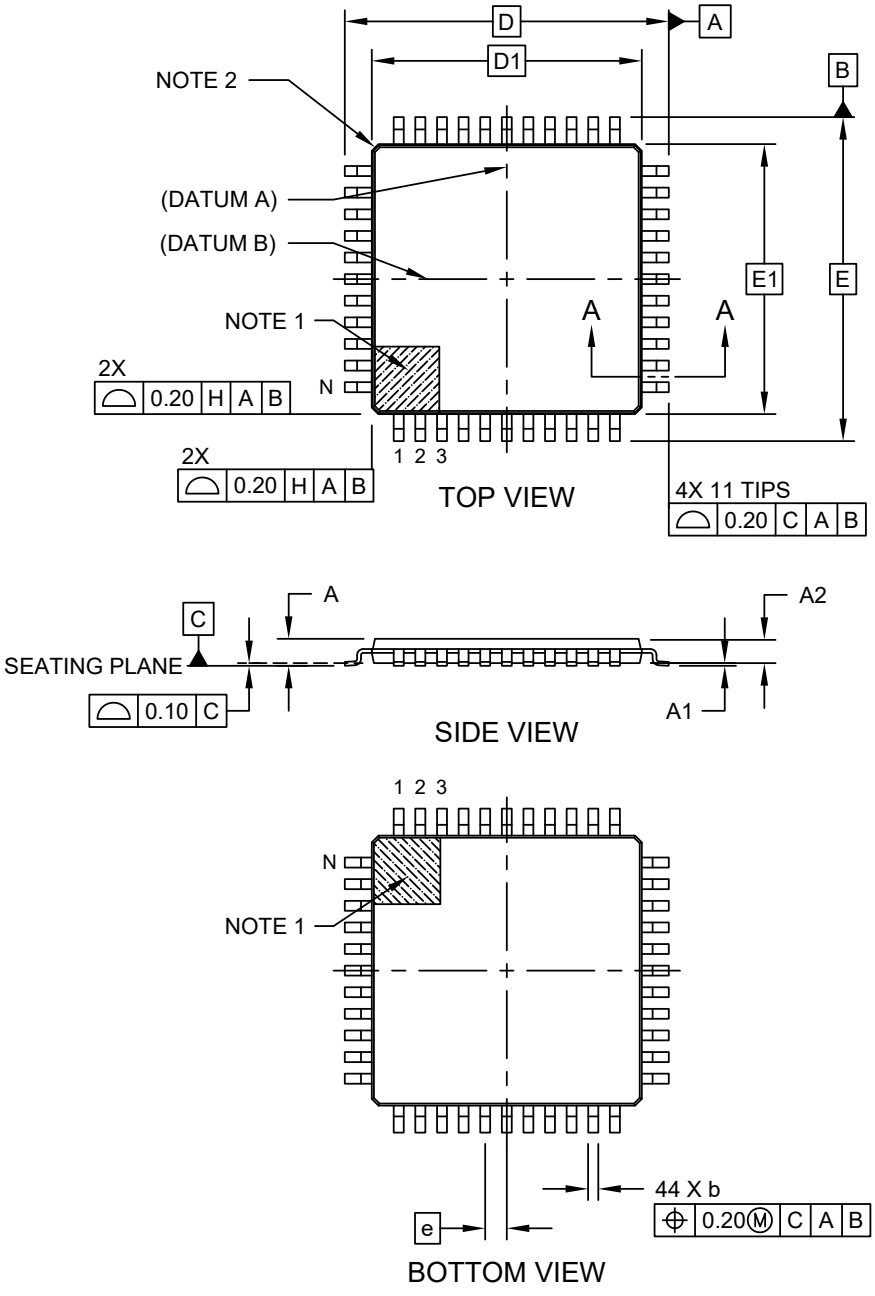
Notes:

1. Dimensioning and tolerancing per ASME Y14.5M
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.
2. For best soldering results, thermal vias, if used, should be filled or tented to avoid solder loss during reflow process

Microchip Technology Drawing C04-2047-MP Rev C

### 44-Lead Plastic Thin Quad Flatpack (PT) - 10x10x1.0 mm Body [TQFP]

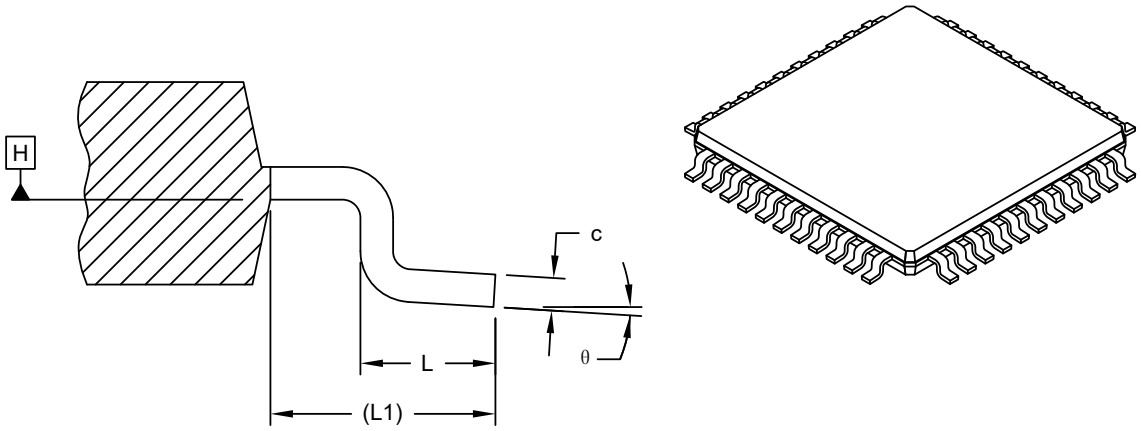
**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging




Microchip Technology Drawing C04-076C Sheet 1 of 2

### 44-Lead Plastic Thin Quad Flatpack (PT) - 10x10x1.0 mm Body [TQFP]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging




SECTION A-A

| Units | | MILLIMETERS | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Number of Leads | N | | 44 | |
| Lead Pitch | e | | 0.80 BSC | |
| Overall Height | A | - | - | 1.20 |
| Standoff | A1 | 0.05 | - | 0.15 |
| Molded Package Thickness | A2 | 0.95 | 1.00 | 1.05 |
| Overall Width | E | | 12.00 BSC | |
| Molded Package Width | E1 | | 10.00 BSC | |
| Overall Length | D | | 12.00 BSC | |
| Molded Package Length | D1 | | 10.00 BSC | |
| Lead Width | b | 0.30 | 0.37 | 0.45 |
| Lead Thickness | c | 0.09 | - | 0.20 |
| Lead Length | L | 0.45 | 0.60 | 0.75 |
| Footprint | L1 | | 1.00 REF | |
| Foot Angle | θ | 0° | 3.5° | 7° |

Notes:

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Exact shape of each corner is optional.
3. Dimensioning and tolerancing per ASME Y14.5M

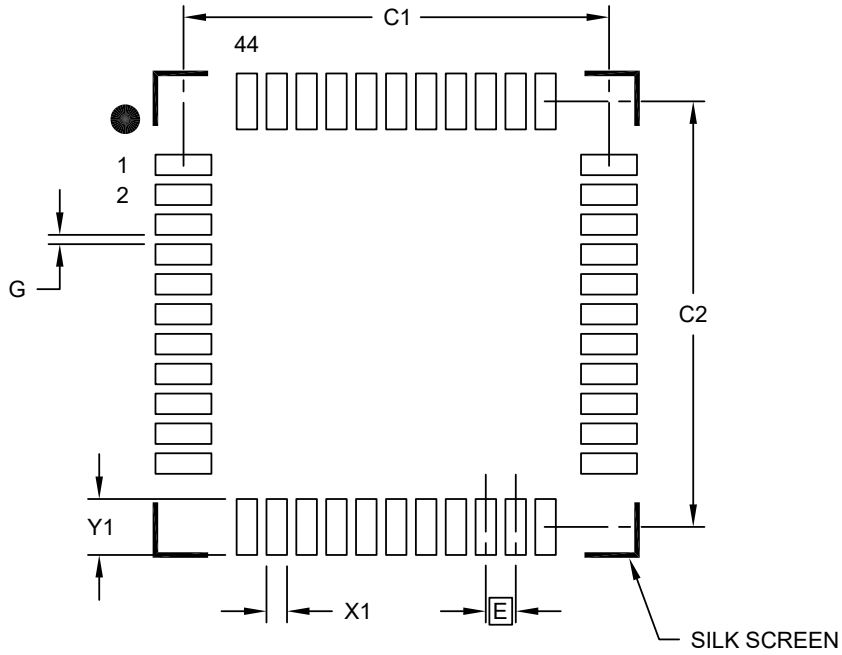
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.

   REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-076C Sheet 2 of 2

### 44-Lead Plastic Thin Quad Flatpack (PT) - 10X10X1 mm Body, 2.00 mm Footprint [TQFP]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging




RECOMMENDED LAND PATTERN

| Units | | MILLIMETERS | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Contact Pitch | E | | 0.80 BSC | |
| Contact Pad Spacing | C1 | | 11.40 | |
| Contact Pad Spacing | C2 | | 11.40 | |
| Contact Pad Width (X44) | X1 | | | 0.55 |
| Contact Pad Length (X44) | Y1 | | | 1.50 |
| Distance Between Pads | G | 0.25 | | |

Notes:

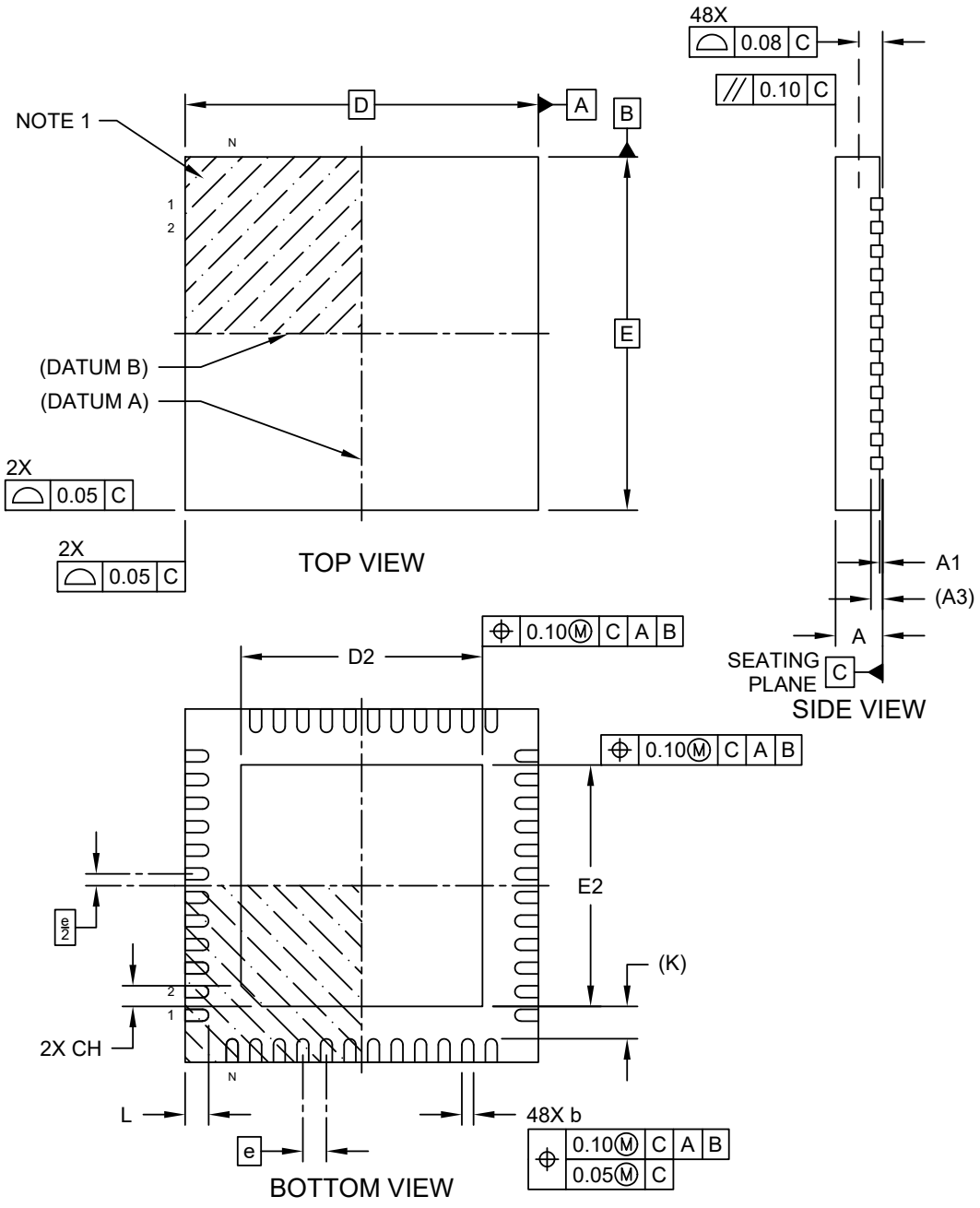
1. Dimensioning and tolerancing per ASME Y14.5M

BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2076B

**48-Lead Very Thin Plastic Quad Flat, No Lead Package (6LX) - 6x6x0.8 mm Body [VQFN] With 4.1x4.1 mm Exposed Pad**

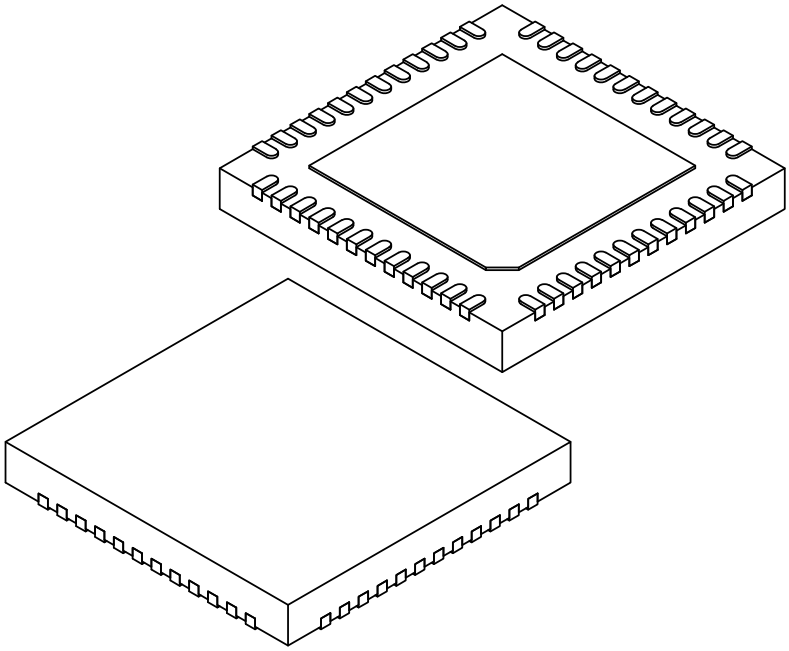
**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging




Microchip Technology Drawing C04-494 Rev B Sheet 1 of 2

### 48-Lead Very Thin Plastic Quad Flat, No Lead Package (6LX) - 6x6x0.8 mm Body [VQFN] With 4.1x4.1 mm Exposed Pad

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



| Units | | MILLIMETERS | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Number of Terminals | N | | 48 | |
| Pitch | e | | 0.40 BSC | |
| Overall Height | A | 0.80 | 0.85 | 0.90 |
| Standoff | A1 | 0.00 | 0.02 | 0.05 |
| Terminal Thickness | A3 | | 0.20 REF | |
| Overall Length | D | | 6.00 BSC | |
| Exposed Pad Length | D2 | 4.00 | 4.10 | 4.20 |
| Overall Width | E | | 6.00 BSC | |
| Exposed Pad Width | E2 | 4.00 | 4.10 | 4.20 |
| Exposed Pad Corner Chamfer | CH | | 0.35 REF | |
| Terminal Width | b | 0.15 | 0.20 | 0.25 |
| Terminal Length | L | 0.30 | 0.40 | 0.50 |
| Terminal-to-Exposed-Pad | K | | 0.55 REF | |

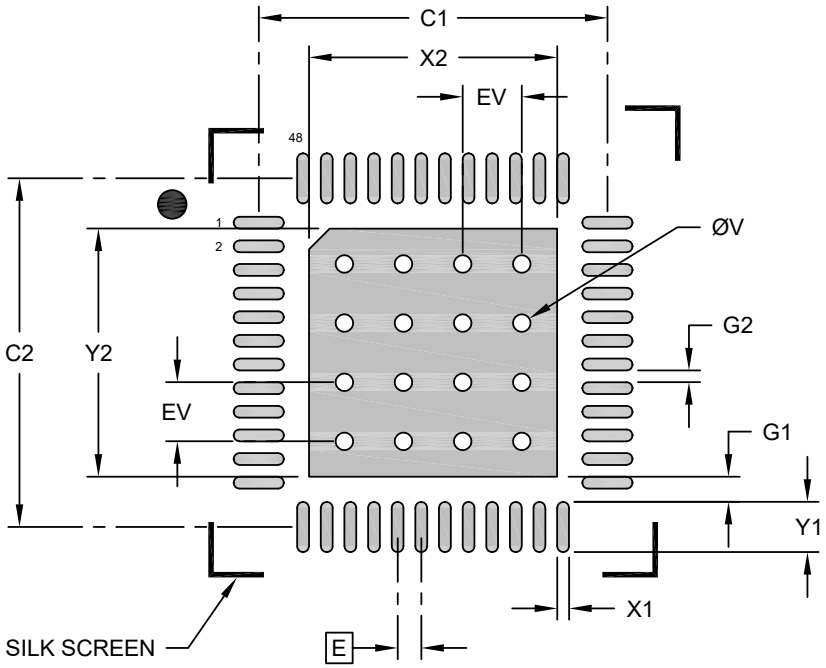
Notes:

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Package is saw singulated
3. Dimensioning and tolerancing per ASME Y14.5M
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.
   REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-494 Rev B Sheet 2 of 2

### 48-Lead Very Thin Plastic Quad Flat, No Lead Package (6LX) - 6x6x0.8 mm Body [VQFN] With 4.1x4.1 mm Exposed Pad

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging




RECOMMENDED LAND PATTERN

| Units | | MILLIMETERS | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Contact Pitch | E | | 0.40 BSC | |
| Optional Center Pad Width | X2 | | | 4.20 |
| Optional Center Pad Length | Y2 | | | 4.20 |
| Contact Pad Spacing | C1 | | 5.90 | |
| Contact Pad Spacing | C2 | | 5.90 | |
| Contact Pad Width (X48) | X1 | | | 0.20 |
| Contact Pad Length (X48) | Y1 | | | 0.85 |
| Contact Pad to Center Pad (X48) | G1 | 0.20 | | |
| Contact Pad to Contact Pad (X44) | G2 | 0.20 | | |
| Thermal Via Diameter | V | | 0.30 | |
| Thermal Via Pitch | EV | | 1.00 | |

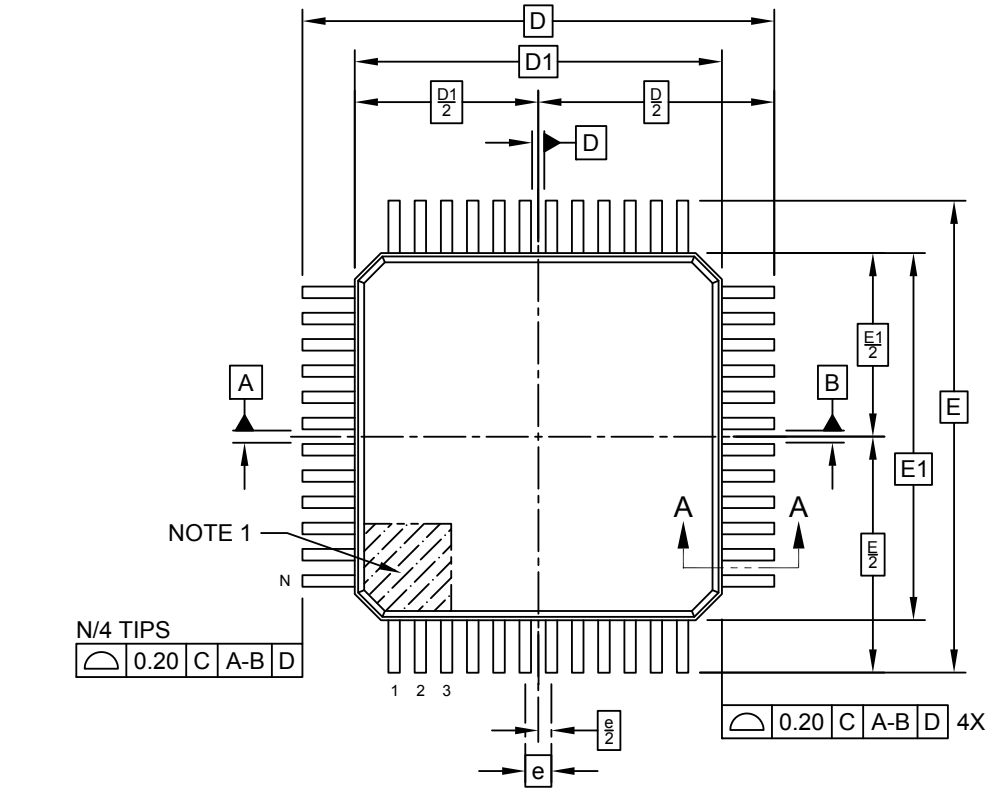
Notes:

1. Dimensioning and tolerancing per ASME Y14.5M
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.
2. For best soldering results, thermal vias, if used, should be filled or tented to avoid solder loss during reflow process

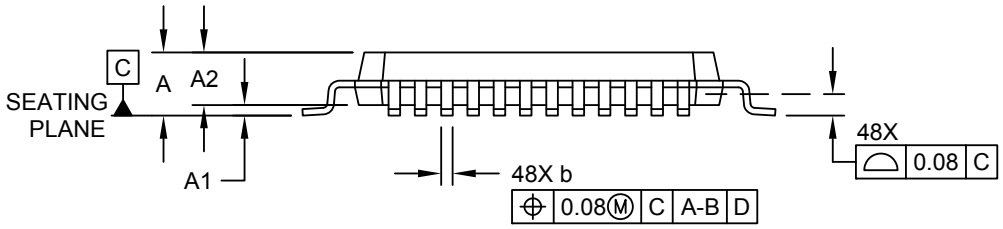
Microchip Technology Drawing C04-2494 Rev B

### 48-Lead Plastic Thin Quad Flatpack (Y8X) - 7x7x1.0 mm Body [TQFP]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging




TOP VIEW

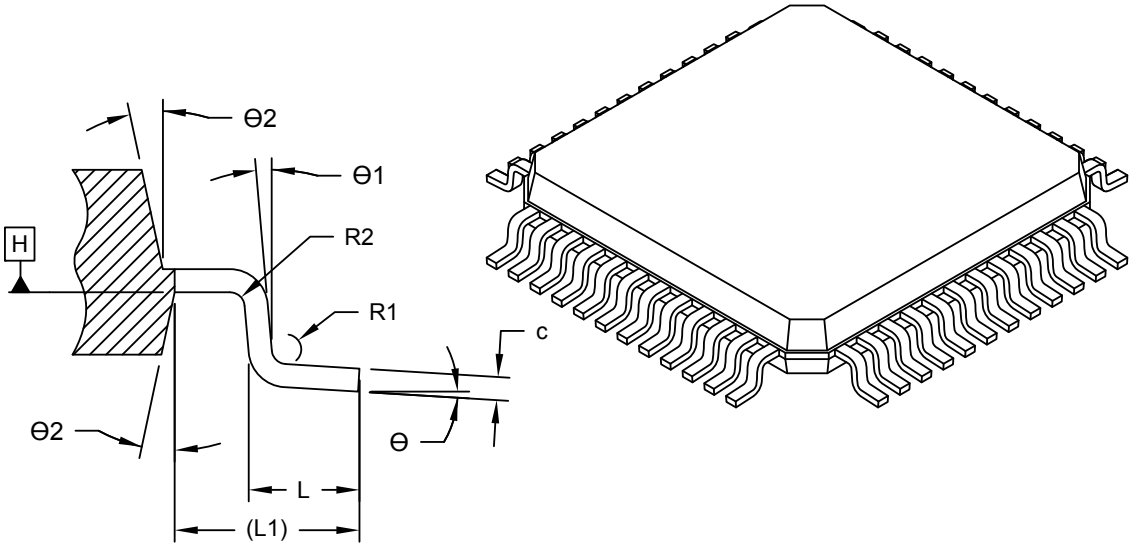



SIDE VIEW

Microchip Technology Drawing C04-300-Y8X Rev D Sheet 1 of 2

### 48-Lead Plastic Thin Quad Flatpack (Y8X) - 7x7x1.0 mm Body [TQFP]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging




SECTION A-A

| Units | | MILLIMETERS | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Number of Terminals | N | | 48 | |
| Pitch | e | | 0.50 BSC | |
| Overall Height | A | - | - | 1.20 |
| Standoff | A1 | 0.05 | - | 0.15 |
| Molded Package Thickness | A2 | 0.95 | 1.00 | 1.05 |
| Overall Length | D | | 9.00 BSC | |
| Molded Package Length | D1 | | 7.00 BSC | |
| Overall Width | E | | 9.00 BSC | |
| Molded Package Width | E1 | | 7.00 BSC | |
| Terminal Width | b | 0.17 | 0.22 | 0.27 |
| Terminal Thickness | c | 0.09 | - | 0.16 |
| Terminal Length | L | 0.45 | 0.60 | 0.75 |
| Footprint | L1 | | 1.00 REF | |
| Lead Bend Radius | R1 | 0.08 | - | - |
| Lead Bend Radius | R2 | 0.08 | - | 0.20 |
| Foot Angle | Θ | 0° | 3.5° | 7° |
| Lead Angle | Θ1 | 0° | - | - |
| Mold Draft Angle | Θ2 | 11° | 12° | 13° |

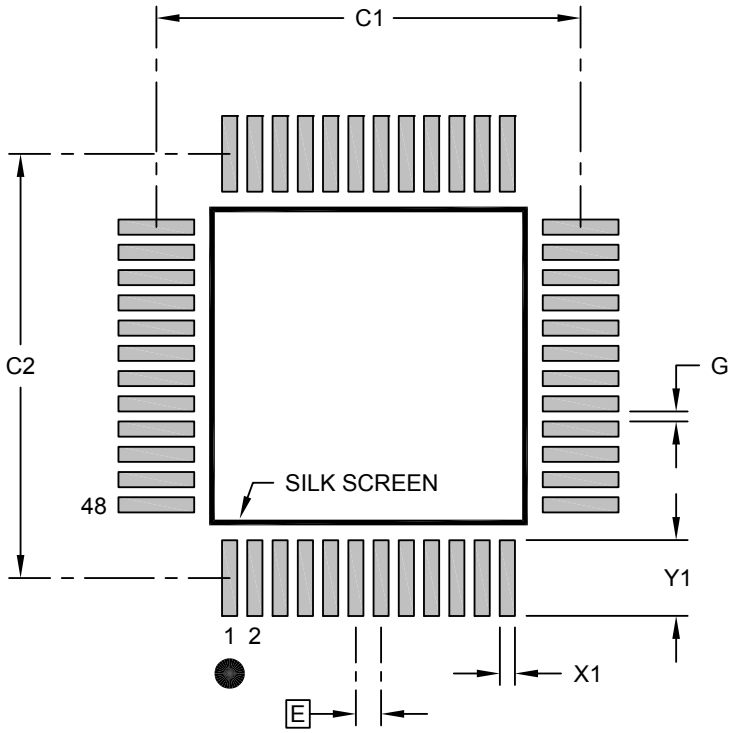
Notes:

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Dimensioning and tolerancing per ASME Y14.5M
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.
   REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-300-Y8X Rev D Sheet 2 of 2

### 48-Lead Plastic Thin Quad Flatpack (Y8X) - 7x7x1.0 mm Body [TQFP]

**Note:** For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging




RECOMMENDED LAND PATTERN

| Units | | MILLIMETERS | | |
|---|---|---|---|---|
| Dimension Limits | | MIN | NOM | MAX |
| Contact Pitch | E | 0.50 BSC | | |
| Contact Pad Spacing | C1 | | 8.40 | |
| Contact Pad Spacing | C2 | | 8.40 | |
| Contact Pad Width (X48) | X1 | | | 0.30 |
| Contact Pad Length (X48) | Y1 | | | 1.50 |
| Distance Between Pads | G | 0.20 | | |

Notes:

1. Dimensioning and tolerancing per ASME Y14.5M
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.
2. For best soldering results, thermal vias, if used, should be filled or tented to avoid solder loss during reflow process

Microchip Technology Drawing C04-2300-Y8X Rev D

# 50. Appendix A: Revision History

| Doc Rev. | Date | Comments |
|---|---|---|
| G | 03/2023 | Added DC and AC Characteristics Graphs and Tables; Updated Tables 47-8, 47-15, 47-16 and 47-17; minor editorial corrections. |
| F | 05/2021 | Updated Package Details section; Updated Peripheral Interrupt registers; Minor editorial corrections |
| E | 12/2020 | Updated Section 17.2.3.1; Minor editorial corrections. |
| D | 07/2020 | Updated Tables: 40-3, 47-7 and 47-15; Minor editorial corrections. |
| C | 02/2020 | Minor editorial corrections. |
| B | 12/2019 | Updated Features: Low-Power Mode Sleep; updated Tables: Electrical Section. |
| A | 10/2019 | Initial document release. |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries

# Microchip Information

## The Microchip Website

Microchip provides online support via our website at www.microchip.com/. This website is used to make files and information easily available to customers. Some of the content available includes:

- **Product Support** – Data sheets and errata, application notes and sample programs, design resources, user's guides and hardware support documents, latest software releases and archived software
- **General Technical Support** – Frequently Asked Questions (FAQs), technical support requests, online discussion groups, Microchip design partner program member listing
- **Business of Microchip** – Product selector and ordering guides, latest Microchip press releases, listing of seminars and events, listings of Microchip sales offices, distributors and factory representatives

## Product Change Notification Service

Microchip's product change notification service helps keep customers current on Microchip products. Subscribers will receive email notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, go to www.microchip.com/pcn and follow the registration instructions.

## Customer Support

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Embedded Solutions Engineer (ESE)
- Technical Support

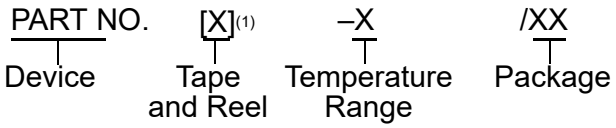
Customers should contact their distributor, representative or ESE for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in this document.

Technical support is available through the website at: www.microchip.com/support

© 2019-2023 Microchip Technology Inc. and its subsidiaries

## Product Identification System

To order or obtain information, e.g., on pricing or delivery, refer to the factory or the listed sales office.




| | | |
|---|---|---|
| Device: | PIC18F25Q43, PIC18F45Q43, PIC18F55Q43, PIC18F26Q43, PIC18F46Q43, PIC18F56Q43, PIC18F27Q43, PIC18F47Q43, PIC18F57Q43 | |
| Tape & Reel Option: | Blank | = Standard Packaging (Tube or Tray) |
| | T | = Tape & Reel |
| Temperature Range: | I | = -40°C to +85°C (Industrial) |
| | E | = -40°C to +125°C (Extended) |
| Package: | SP | = 28-lead SPDIP |
| | SO | = 28-lead SOIC |
| | SS | = 28-lead SSOP |
| | STX | = 28-lead VQFN |
| | P | = 40-lead PDIP |
| | MP | = 40-lead QFN |
| | PT | = 44-lead TQFP |
| | PT | = 48-lead TQFP |
| | 6LX | = 48-lead VQFN |

Examples:

- PIC18F27Q43 T-E/SP: Tape and Reel, Extended temperature, 28-lead SPDIP
- PIC18F46Q43 T-I/PT: Tape and Reel, Industrial temperature, 44-lead TQFP
- PIC18F55Q43 T-I/6LX: Tape and Reel, Industrial temperature, 48-lead VQFN

**Notes:**

1. Tape and Reel identifier only appears in the catalog part number description. This identifier is used for ordering purposes and is not printed on the device package. Check with your Microchip Sales Office for package availability with the Tape and Reel option.
2. Small form-factor packaging options may be available. Please check www.microchip.com/packaging for small-form factor package availability, or contact your local Sales Office.

## Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip products:

- Microchip products meet the specifications contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is secure when used in the intended manner, within operating specifications, and under normal conditions.
- Microchip values and aggressively protects its intellectual property rights. Attempts to breach the code protection features of Microchip product is strictly prohibited and may violate the Digital Millennium Copyright Act.

© 2019-2023 Microchip Technology Inc. and its subsidiaries

- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of its code. Code protection does not mean that we are guaranteeing the product is “unbreakable”. Code protection is constantly evolving. Microchip is committed to continuously improving the code protection features of our products.

## Legal Notice

This publication and the information herein may be used only with Microchip products, including to design, test, and integrate Microchip products with your application. Use of this information in any other manner violates these terms. Information regarding device applications is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. Contact your local Microchip sales office for additional support or, obtain additional support at www.microchip.com/en-us/support/design-help/client-support-services.

THIS INFORMATION IS PROVIDED BY MICROCHIP "AS IS". MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE, OR WARRANTIES RELATED TO ITS CONDITION, QUALITY, OR PERFORMANCE.

IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE, INCIDENTAL, OR CONSEQUENTIAL LOSS, DAMAGE, COST, OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE INFORMATION OR ITS USE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE FULLEST EXTENT ALLOWED BY LAW, MICROCHIP'S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THE INFORMATION OR ITS USE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY, THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THE INFORMATION.

Use of Microchip devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights unless otherwise stated.

## Trademarks

The Microchip name and logo, the Microchip logo, Adaptec, AVR, AVR logo, AVR Freaks, BesTime, BitCloud, CryptoMemory, CryptoRF, dsPIC, flexPWR, HELDO, IGLOO, JukeBlox, KeeLoq, Kleer, LANCheck, LinkMD, maXStylus, maXTouch, MediaLB, megaAVR, Microsemi, Microsemi logo, MOST, MOST logo, MPLAB, OptoLyzer, PIC, picoPower, PICSTART, PIC32 logo, PolarFire, Prochip Designer, QTouch, SAM-BA, SenGenuity, SpyNIC, SST, SST Logo, SuperFlash, Symmetricom, SyncServer, Tachyon, TimeSource, tinyAVR, UNI/O, Vectron, and XMEGA are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

AgileSwitch, APT, ClockWorks, The Embedded Control Solutions Company, EtherSynch, Flashtec, Hyper Speed Control, HyperLight Load, Libero, motorBench, mTouch, Powermite 3, Precision Edge, ProASIC, ProASIC Plus, ProASIC Plus logo, Quiet- Wire, SmartFusion, SyncWorld, Temux, TimeCesium, TimeHub, TimePictra, TimeProvider, TrueTime, and ZL are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Adjacent Key Suppression, AKS, Analog-for-the-Digital Age, Any Capacitor, AnyIn, AnyOut, Augmented Switching, BlueSky, BodyCom, Clockstudio, CodeGuard, CryptoAuthentication, CryptoAutomotive, CryptoCompanion, CryptoController, dsPICDEM, dsPICDEM.net, Dynamic Average Matching, DAM, ECAN, Espresso T1S, EtherGREEN, GridTime, IdealBridge, In-Circuit Serial Programming, ICSP, INICnet, Intelligent Paralleling, IntelliMOS, Inter-Chip Connectivity, JitterBlocker, Knob-on-Display, KoD, maxCrypto, maxView, memBrain, Mindi, MiWi, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, MultiTRAK, NetDetach, Omniscient Code Generation, PICDEM, PICDEM.net, PICkit, PICtail, PowerSmart, PureSilicon, QMatrix, REAL ICE, Ripple Blocker, RTAX, RTG4, SAM-ICE, Serial Quad I/O, simpleMAP, SimpliPHY, SmartBuffer, SmartHLS, SMART-I.S., storClad, SQI, SuperSwitcher, SuperSwitcher II, Switchtec, SynchroPHY, Total Endurance, Trusted Time, TSHARC, USBCheck, VariSense, VectorBlox, VeriPHY, ViewSpan, WiperLock, XpressConnect, and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

The Adaptec logo, Frequency on Demand, Silicon Storage Technology, and Symmcom are registered trademarks of Microchip Technology Inc. in other countries.

GestIC is a registered trademark of Microchip Technology Germany II GmbH & Co. KG, a subsidiary of Microchip Technology Inc., in other countries.

All other trademarks mentioned herein are property of their respective companies.

© 2019-2023, Microchip Technology Incorporated and its subsidiaries. All Rights Reserved.

ISBN: 978-1-6683-2025-9

## Quality Management System

For information regarding Microchip's Quality Management Systems, please visit www.microchip.com/quality.

# Worldwide Sales and Service

| AMERICAS | ASIA/PACIFIC | ASIA/PACIFIC | EUROPE |
|---|---|---|---|
| **Corporate Office**<br>2355 West Chandler Blvd.<br>Chandler, AZ 85224-6199<br>Tel: 480-792-7200<br>Fax: 480-792-7277<br>Technical Support:<br>www.microchip.com/support<br>Web Address:<br>www.microchip.com<br>**Atlanta**<br>Duluth, GA<br>Tel: 678-957-9614<br>Fax: 678-957-1455<br>**Austin, TX**<br>Tel: 512-257-3370<br>**Boston**<br>Westborough, MA<br>Tel: 774-760-0087<br>Fax: 774-760-0088<br>**Chicago**<br>Itasca, IL<br>Tel: 630-285-0071<br>Fax: 630-285-0075<br>**Dallas**<br>Addison, TX<br>Tel: 972-818-7423<br>Fax: 972-818-2924<br>**Detroit**<br>Novi, MI<br>Tel: 248-848-4000<br>**Houston, TX**<br>Tel: 281-894-5983<br>**Indianapolis**<br>Noblesville, IN<br>Tel: 317-773-8323<br>Fax: 317-773-5453<br>Tel: 317-536-2380<br>**Los Angeles**<br>Mission Viejo, CA<br>Tel: 949-462-9523<br>Fax: 949-462-9608<br>Tel: 951-273-7800<br>**Raleigh, NC**<br>Tel: 919-844-7510<br>**New York, NY**<br>Tel: 631-435-6000<br>**San Jose, CA**<br>Tel: 408-735-9110<br>Tel: 408-436-4270<br>**Canada - Toronto**<br>Tel: 905-695-1980<br>Fax: 905-695-2078 | **Australia - Sydney**<br>Tel: 61-2-9868-6733<br>**China - Beijing**<br>Tel: 86-10-8569-7000<br>**China - Chengdu**<br>Tel: 86-28-8665-5511<br>**China - Chongqing**<br>Tel: 86-23-8980-9588<br>**China - Dongguan**<br>Tel: 86-769-8702-9880<br>**China - Guangzhou**<br>Tel: 86-20-8755-8029<br>**China - Hangzhou**<br>Tel: 86-571-8792-8115<br>**China - Hong Kong SAR**<br>Tel: 852-2943-5100<br>**China - Nanjing**<br>Tel: 86-25-8473-2460<br>**China - Qingdao**<br>Tel: 86-532-8502-7355<br>**China - Shanghai**<br>Tel: 86-21-3326-8000<br>**China - Shenyang**<br>Tel: 86-24-2334-2829<br>**China - Shenzhen**<br>Tel: 86-755-8864-2200<br>**China - Suzhou**<br>Tel: 86-186-6233-1526<br>**China - Wuhan**<br>Tel: 86-27-5980-5300<br>**China - Xian**<br>Tel: 86-29-8833-7252<br>**China - Xiamen**<br>Tel: 86-592-2388138<br>**China - Zhuhai**<br>Tel: 86-756-3210040 | **India - Bangalore**<br>Tel: 91-80-3090-4444<br>**India - New Delhi**<br>Tel: 91-11-4160-8631<br>**India - Pune**<br>Tel: 91-20-4121-0141<br>**Japan - Osaka**<br>Tel: 81-6-6152-7160<br>**Japan - Tokyo**<br>Tel: 81-3-6880- 3770<br>**Korea - Daegu**<br>Tel: 82-53-744-4301<br>**Korea - Seoul**<br>Tel: 82-2-554-7200<br>**Malaysia - Kuala Lumpur**<br>Tel: 60-3-7651-7906<br>**Malaysia - Penang**<br>Tel: 60-4-227-8870<br>**Philippines - Manila**<br>Tel: 63-2-634-9065<br>**Singapore**<br>Tel: 65-6334-8870<br>**Taiwan - Hsin Chu**<br>Tel: 886-3-577-8366<br>**Taiwan - Kaohsiung**<br>Tel: 886-7-213-7830<br>**Taiwan - Taipei**<br>Tel: 886-2-2508-8600<br>**Thailand - Bangkok**<br>Tel: 66-2-694-1351<br>**Vietnam - Ho Chi Minh**<br>Tel: 84-28-5448-2100 | **Austria - Wels**<br>Tel: 43-7242-2244-39<br>Fax: 43-7242-2244-393<br>**Denmark - Copenhagen**<br>Tel: 45-4485-5910<br>Fax: 45-4485-2829<br>**Finland - Espoo**<br>Tel: 358-9-4520-820<br>**France - Paris**<br>Tel: 33-1-69-53-63-20<br>Fax: 33-1-69-30-90-79<br>**Germany - Garching**<br>Tel: 49-8931-9700<br>**Germany - Haan**<br>Tel: 49-2129-3766400<br>**Germany - Heilbronn**<br>Tel: 49-7131-72400<br>**Germany - Karlsruhe**<br>Tel: 49-721-625370<br>**Germany - Munich**<br>Tel: 49-89-627-144-0<br>Fax: 49-89-627-144-44<br>**Germany - Rosenheim**<br>Tel: 49-8031-354-560<br>**Israel - Ra'anana**<br>Tel: 972-9-744-7705<br>**Italy - Milan**<br>Tel: 39-0331-742611<br>Fax: 39-0331-466781<br>**Italy - Padova**<br>Tel: 39-049-7625286<br>**Netherlands - Drunen**<br>Tel: 31-416-690399<br>Fax: 31-416-690340<br>**Norway - Trondheim**<br>Tel: 47-72884388<br>**Poland - Warsaw**<br>Tel: 48-22-3325737<br>**Romania - Bucharest**<br>Tel: 40-21-407-87-50<br>**Spain - Madrid**<br>Tel: 34-91-708-08-90<br>Fax: 34-91-708-08-91<br>**Sweden - Gothenberg**<br>Tel: 46-31-704-60-40<br>**Sweden - Stockholm**<br>Tel: 46-8-5090-4654<br>**UK - Wokingham**<br>Tel: 44-118-921-5800<br>Fax: 44-118-921-5820 |

© 2019-2023 Microchip Technology Inc.
and its subsidiaries