
64-Pin, Low-Power, High-Performance Microcontrollers with XLP Technology

Description

PIC18(L)F67K40 microcontrollers feature Analog, Core Independent Peripherals and Communication Peripherals, combined with eXtreme Low-Power (XLP) technology for a wide range of general purpose and low-power applications. These 64-pin devices are equipped with a 10-bit ADC with Computation (ADCC) automating Capacitive Voltage Divider (CVD) techniques for advanced touch sensing, averaging, filtering, oversampling and performing automatic threshold comparisons. They also offer a set of Core Independent Peripherals such as Complementary Waveform Generator (CWG), Windowed Watchdog Timer (WWDT), Cyclic Redundancy Check (CRC)/Memory Scan, Zero-Cross Detect (ZCD) and Peripheral Pin Select (PPS), providing for increased design flexibility and lower system cost.

Core Features

- C Compiler Optimized RISC Architecture
- Only 83 Instructions
- Operating Speed:
 - DC – 64 MHz clock input
 - 62.5 ns minimum instruction cycle
- Programmable 2-Level Interrupt Priority
- 31-Level Deep Hardware Stack
- Four 8-Bit Timers (TMR2/4/6/8) with Hardware Limit Timer (HLT)
- Five 16-Bit Timers (TMR0/1/3/5/7)
- Low-Current Power-on Reset (POR)
- Power-up Timer (PWRT)
- Brown-out Reset (BOR)
- Low-Power BOR (LPBOR) Option
- Programmable Code Protection
- Windowed Watchdog Timer (WWDT):
 - Timer monitoring of overflow and underflow events
 - Variable prescaler selection
 - Variable window size selection
 - All sources configurable in hardware or software

Memory

- 128K bytes Program Flash Memory
- 3568 Bytes Data SRAM Memory
- 1024 Bytes Data EEPROM
- Direct, Indirect and Relative Addressing modes

Operating Characteristics

- Operating Voltage Ranges:
 - 1.8V to 3.6V (PIC18LF6xK40)
 - 2.3V to 5.5V (PIC18F6xK40)
- Temperature Range:
 - Industrial: -40°C to 85°C
 - Extended: -40°C to 125°C

Power-Saving Operation Modes

- 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

eXtreme Low-Power (XLP) Features

- Sleep mode: 50 nA @ 1.8V, typical
- Windowed Watchdog Timer: 500 nA @ 1.8V, typical
- Secondary Oscillator: 500 nA @ 32 kHz
- Operating Current:
 - 8 uA @ 32 kHz, 1.8V, typical
 - 32 uA/MHz @ 1.8V, typical

Digital Peripherals

- Complementary Waveform Generator (CWG):
 - Rising and falling edge dead-band control
 - Full-bridge, half-bridge, 1-channel drive
 - Multiple signal sources
- Capture/Compare/PWM (CCP) modules:
 - Five CCPs
 - 16-bit resolution for Capture/Compare modes
 - 10-bit resolution for PWM mode
- 10-Bit Pulse-Width Modulators (PWM):
 - Two 10-bit PWMs
- Serial Communications:
 - Five Enhanced USART (EUSART) with Auto-Baud Detect, Auto-wake-up on Start. RS-232, RS-485, LIN compatible
 - SPI
 - I²C, SMBus and PMBus™ compatible
- 59 I/O Pins and One Input Pin:
 - Individually programmable pull-ups
 - Slew rate control
 - Interrupt-on-change
 - Input level selection control

Digital Peripherals (Continued)

- Programmable CRC with Memory Scan:
 - Reliable data/program memory monitoring for Fail-Safe operation (e.g., Class B)
 - Calculate CRC over any portion of Flash or EEPROM
 - High-speed or background operation
- Hardware Limit Timer (TMR2/4/6/8+HLT):
 - Hardware monitoring and Fault detection
- Peripheral Pin Select (PPS):
 - Enables pin mapping of digital I/O
- Data Signal Modulator (DSM)
- Two Signal Measurement Timer (SMT1/2):
 - 24-bit timer/counter with prescaler
 - Multiple gate and clock inputs

Analog Peripherals

- 10-Bit Analog-to-Digital Converter with Computation (ADC²):
 - 47 external channels
 - Conversion available during Sleep
 - Four internal analog channels
 - Internal and external trigger options
 - Automated math functions on input signals:
 - averaging, filter calculations, oversampling and threshold comparison
- Hardware Capacitive Voltage Divider (CVD) Support:
 - 8-bit precharge timer
 - Adjustable sample and hold capacitor array
 - Guard ring digital output drive
- Zero-Cross Detect (ZCD):
 - Detect when AC signal on pin crosses ground
- 5-Bit Digital-to-Analog Converter (DAC):
 - Output available externally
 - Programmable 5-bit voltage (% of V_{DD})
 - Internal connections to comparators, Fixed Voltage Reference and ADC
- Three Comparators (CMP):
 - Five external inputs
 - External output via PPS
- Fixed Voltage Reference (FVR) module:
 - 1.024V, 2.048V and 4.096V output levels

Clocking Structure

- High-Precision Internal Oscillator Block (HFINTOSC):
 - Selectable frequency range up to 64 MHz
 - ±1% at calibration
- 32 kHz Low-Power Internal Oscillator (LFINTOSC)
- External 32 kHz Crystal Oscillator (SOSC)
- External Oscillator Block:
 - Three crystal/resonator modes
 - 4x PLL with external sources
- Fail-Safe Clock Monitor:
 - Allows for safe shutdown if peripheral clock stops
- Oscillator Start-up Timer (OST)

Programming/Debug Features

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

PIC18(L)F67K40

PIC18(L)F6xK40 Family Types

| Device | Data Sheet Index | Program Memory Flash (words) | Data SRAM (bytes) | Data EEPROM (bytes) | I/O Pins | 16-bit Timers | Comparators | 10-bit ADC ² with Computation (ch) | 5-bit DAC | Zero-Cross Detect | CCP/10-bit PWM | CWG | Signal Measurement Timer (SMT) | 8-bit TMR with HLT | Windowed Watchdog Timer | CRC with Memory Scan | EUSART | I ² C/SPI | PPS | Peripheral Module Disable | Temperature Indicator | Debug ⁽¹⁾ |
|----------------|------------------|------------------------------|-------------------|---------------------|----------|---------------|-------------|---|-----------|-------------------|----------------|-----|--------------------------------|--------------------|-------------------------|----------------------|--------|----------------------|-----|---------------------------|-----------------------|----------------------|
| PIC18(L)F65K40 | (1) | 32k | 2048 | 1024 | 60 | 5 | 3 | 47 | 1 | 1 | 5/2 | 1 | 2 | 4 | Y | Y | 5 | 2 | Y | Y | Y | I |
| PIC18(L)F66K40 | (1) | 64k | 3568 | 1024 | 60 | 5 | 3 | 47 | 1 | 1 | 5/2 | 1 | 2 | 4 | Y | Y | 5 | 2 | Y | Y | Y | I |
| PIC18(L)F67K40 | (2) | 128k | 3568 | 1024 | 60 | 5 | 3 | 47 | 1 | 1 | 5/2 | 1 | 2 | 4 | Y | Y | 5 | 2 | Y | Y | Y | I |

Note 1: Debugging Methods: (I) – Integrated on Chip.

Data Sheet Index: (Unshaded devices are described in this document.)

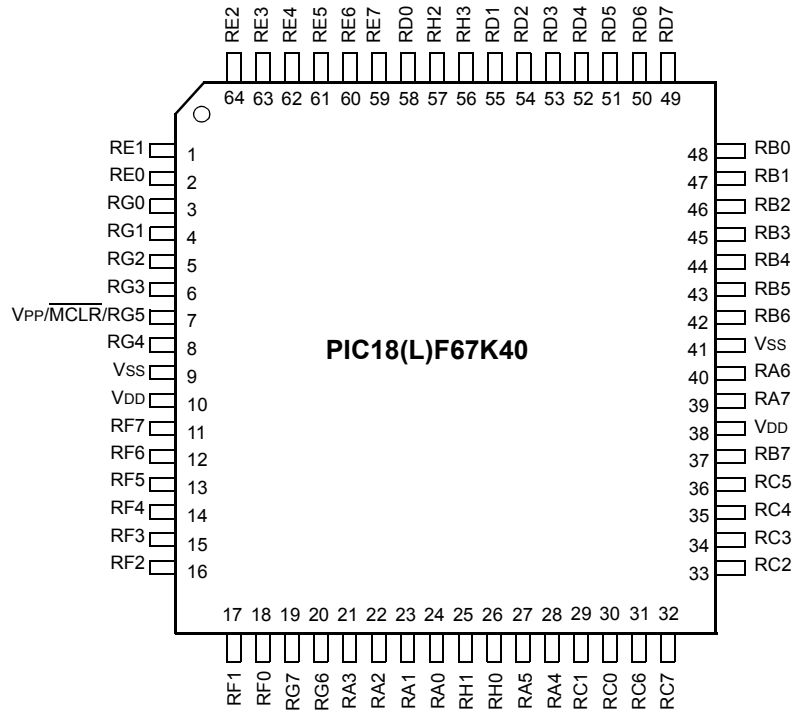
1. DS40001842 [PIC18\(L\)F65/66K40 Data Sheet, 64-Pin, 8-bit Flash Microcontrollers](#)
2. DS40001841 [PIC18\(L\)F67K40 Data Sheet, 64-Pin, 8-bit Flash Microcontrollers](#)

Note: For other small form-factor package availability and marking information, please visit <http://www.microchip.com/packaging> or contact your local sales office.

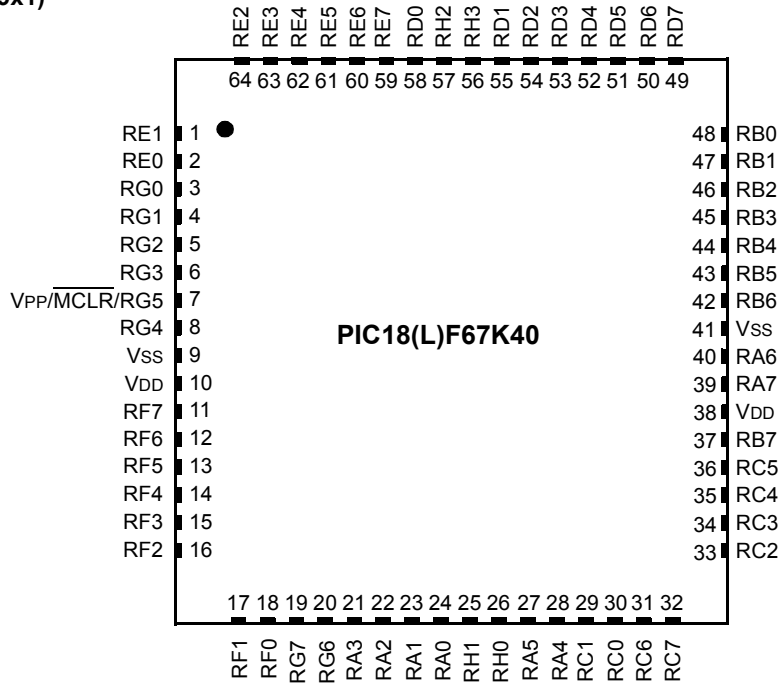
PIC18(L)F67K40

Pin Diagrams

64-Pin TQFP (10x10x1)



64-Pin QFN (9x9x1)



Note: It is recommended that the exposed bottom pad be connected to Vss. However, it must not be the only Vss connection to the device.

Pin Allocation Tables

TABLE 1: 64-PIN ALLOCATION TABLE (PIC18(L)F67K40)

| I/O ⁽²⁾ | 64-Pin TQFP, QFN | A/D | DAC | Comparator | Timers | CCP and PWM | CWG | ZCD | SMT | Clock Reference (CLKR) | Interrupt | ELUSART | DSM | MSSP | Basic |
|--------------------|------------------|---------------|----------|----------------------------|----------------------|-------------|-----|-------|-----|------------------------|------------------------------|--------------------|-----|------------------|----------------|
| RA0 | 24 | ANA0 | — | C1IN4- C2IN4- C3IN4- | T8IN ⁽¹⁾ | — | — | — | — | — | — | — | — | — | — |
| RA1 | 23 | ANA1 | — | — | T2IN ⁽¹⁾ | — | — | — | — | — | — | — | — | — | — |
| RA2 | 22 | ANA2 VREF- | VREF- | C1IN1+ C2IN1+ C3IN1+ | — | — | — | — | — | — | — | — | — | — | — |
| RA3 | 21 | ANA3 VREF+ | VREF+ | — | — | — | — | — | — | — | — | — | — | — | — |
| RA4 | 28 | ANA4 | — | — | T0CKI ⁽¹⁾ | — | — | — | — | — | — | — | — | — | — |
| RA5 | 27 | ANA5 | — | — | T3G ⁽¹⁾ | — | — | — | — | — | — | — | — | — | — |
| RA6 | 40 | ANA6 | — | — | — | — | — | — | — | — | — | — | — | — | CLKOUT OSC2 |
| RA7 | 39 | ANA7 | — | — | — | — | — | — | — | — | — | — | — | — | OSC1 CLKIN |
| RB0 | 48 | ANB0 | — | — | — | — | — | ZCDIN | — | — | IOCB0 INT0 ⁽⁴⁾ | — | — | — | — |
| RB1 | 47 | ANB1 | — | — | — | — | — | — | — | — | IOCB1 INT1 ⁽⁴⁾ | — | — | — ⁽⁴⁾ | — |
| RB2 | 46 | ANB2 | — | — | — | — | — | — | — | — | IOCB2 INT2 ⁽⁴⁾ | — | — | — ⁽⁴⁾ | — |
| RB3 | 45 | ANB3 | — | — | — | — | — | — | — | — | IOCB3 INT3 ⁽⁴⁾ | — | — | — | — |
| RB4 | 44 | ANB4 | — | — | — | — | — | — | — | — | IOCB4 | — | — | — | — |
| RB5 | 43 | ANB5 | — | — | — | — | — | — | — | — | IOCB5 | — | — | — | — |
| RB6 | 42 | ANB6 | — | — | — | — | — | — | — | — | IOCB6 | — | — | — | ICSPCLK |
| RB7 | 37 | ANB7 | DAC1OUT2 | — | — | — | — | — | — | — | IOCB7 | — | — | — | ICSPDAT |
| RC0 | 30 | — | — | — | T1CKI ⁽¹⁾ | — | — | — | — | — | IOCC0 | CK4 ⁽³⁾ | — | — | SOSCO |

- Note** 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 [Table 17-1](#) 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 [Register 17-2](#)
- 3: This is a bidirectional signal. For normal module operation, the firmware should map this signal to the same pin in both the PPS input and PPS output registers.
- 4: These pins are configured for I²C™ logic levels; The SCLx/SDAx signals may be assigned to any of these pins. PPS assignments to the other pins (e.g., RB1) will operate, but input logic levels will be standard TTL/ST as selected by the INLVL register, instead of the I²C specific or SMBus input buffer thresholds.

TABLE 1: 64-PIN ALLOCATION TABLE (PIC18(L)F67K40) (CONTINUED)

| I/O ⁽²⁾ | 64-Pin TQFP, QFN | A/D | DAC | Comparator | Timers | CCP and PWM | CWG | ZCD | SMT | Clock Reference (CLKR) | Interrupt | EUSART | DSM | MSSP | Basic |
|--------------------|------------------|------|-----|------------|--|---------------------|-----------------------|-----|-----|------------------------|-----------|--|-----------------------|--|-------|
| RC1 | 29 | — | — | — | T6IN ⁽¹⁾ | — | — | — | — | — | IOCC1 | RX4 ⁽¹⁾ DT4 ⁽¹⁾ | — | — | SOSCI |
| RC2 | 33 | — | — | — | — | — | CWG1IN ⁽¹⁾ | — | — | — | IOCC2 | — | — | — | — |
| RC3 | 34 | — | — | — | — | — | — | — | — | — | IOCC3 | — | — | SCL1 ^(3,4) SCK1 ⁽¹⁾ | — |
| RC4 | 35 | — | — | — | — | — | — | — | — | — | IOCC4 | — | — | SDA1 ^(3,4) SDI1 ⁽¹⁾ | — |
| RC5 | 36 | — | — | — | — | — | — | — | — | — | IOCC5 | — | — | — | — |
| RC6 | 31 | — | — | — | — | — | — | — | — | — | IOCC6 | CK1 ⁽³⁾ | — | — | — |
| RC7 | 32 | — | — | — | — | — | — | — | — | — | IOCC7 | RX1 ⁽¹⁾ DT1 ⁽³⁾ | — | — | — |
| RD0 | 58 | AND0 | — | — | — | — | — | — | — | — | — | — | — | — | — |
| RD1 | 55 | AND1 | — | — | T5CK1 ⁽¹⁾ T7G ⁽¹⁾ | — | — | — | — | — | — | — | — | — | — |
| RD2 | 54 | AND2 | — | — | — | — | — | — | — | — | — | — | — | — | — |
| RD3 | 53 | AND3 | — | — | — | — | — | — | — | — | — | — | MDCARL ⁽¹⁾ | — | — |
| RD4 | 52 | AND4 | — | — | — | — | — | — | — | — | — | — | MDCARH ⁽¹⁾ | — | — |
| RD5 | 51 | AND5 | — | — | — | — | — | — | — | — | — | — | MDSRC ⁽¹⁾ | SDA2 ^(3,4) SDI2 ⁽¹⁾ | — |
| RD6 | 50 | AND6 | — | — | — | — | — | — | — | — | — | — | — | SCL2 ^(3,4) SCK2 ⁽¹⁾ | — |
| RD7 | 49 | AND7 | — | — | — | — | — | — | — | — | — | — | — | $\overline{SS}2^{(1)}$ | — |
| RE0 | 2 | ANE0 | — | — | — | — | — | — | — | — | IOCE0 | CK3 ⁽³⁾ | — | — | — |
| RE1 | 1 | ANE1 | — | — | — | — | — | — | — | — | IOCE1 | RX3 ⁽¹⁾ DT3 ⁽³⁾ | — | — | — |
| RE2 | 64 | ANE2 | — | — | — | — | — | — | — | — | IOCE2 | CK5 ⁽³⁾ | — | — | — |
| RE3 | 63 | ANE3 | — | — | — | — | — | — | — | — | IOCE3 | RX5 ⁽¹⁾ DT5 ⁽³⁾ | — | — | — |
| RE4 | 62 | ANE4 | — | — | T4IN ⁽¹⁾ | CCP2 ⁽¹⁾ | — | — | — | — | IOCE4 | — | — | — | — |

- Note**
- 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 [Table 17-1](#) 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 [Register 17-2](#)
 - 3: This is a bidirectional signal. For normal module operation, the firmware should map this signal to the same pin in both the PPS input and PPS output registers.
 - 4: These pins are configured for I²C™ logic levels; The SCLx/SDAx signals may be assigned to any of these pins. PPS assignments to the other pins (e.g., RB1) will operate, but input logic levels will be standard TTL/ST as selected by the INLVL register, instead of the I²C specific or SMBus input buffer thresholds.

TABLE 1: 64-PIN ALLOCATION TABLE (PIC18(L)F67K40) (CONTINUED)

| I/O ⁽²⁾ | 64-Pin TQFP, QFN | A/D | DAC | Comparator | Timers | CCP and PWM | CWG | ZCD | SMT | Clock Reference (CLKR) | Interrupt | EUSART | DSM | MSSP | Basic |
|--------------------|------------------|-----------------------|----------|----------------------------|--|---------------------|-----|-----|-------------------------|------------------------|-----------|--|-----|--------------------|-----------|
| RE5 | 61 | ANE5 | — | — | — | CCP1 ⁽¹⁾ | — | — | — | — | IOCE5 | — | — | — | — |
| RE6 | 60 | ANE6 | — | — | — | CCP3 ⁽¹⁾ | — | — | SMT1WIN1 ⁽¹⁾ | — | IOCE6 | — | — | — | — |
| RE7 | 59 | ANE7 | — | — | — | — | — | — | SMT1SIG1 ⁽¹⁾ | — | IOCE7 | — | — | — | — |
| RF0 | 18 | ANF0— | — | C1IN0- C2IN0- | — | — | — | — | — | — | — | — | — | — | — |
| RF1 | 17 | ANF1 | — | — | — | — | — | — | — | — | — | — | — | — | — |
| RF2 | 16 | ANF2 | — | — | — | — | — | — | — | — | — | — | — | — | — |
| RF3 | 15 | ANF3— | — | C1IN2- C2IN2- C3IN2- | — | — | — | — | — | — | — | — | — | — | — |
| RF4 | 14 | ANF4 | — | C2IN0+ | — | — | — | — | — | — | — | — | — | — | — |
| RF5 | 13 | ANF5 | DAC1OUT1 | C1IN1- C2IN1- | — | — | — | — | — | — | — | — | — | — | — |
| RF6 | 12 | ANF6 | — | C1IN0+ | — | — | — | — | — | — | — | — | — | — | — |
| RF7 | 11 | ANF7— | — | C1IN3- C2IN3- C3IN3- | — | — | — | — | — | — | — | — | — | SS1 ⁽¹⁾ | — |
| RG0 | 3 | ANG0 | — | — | — | — | — | — | — | — | — | — | — | — | — |
| RG1 | 4 | ANG1 | — | — | — | — | — | — | — | — | — | CK2 ⁽¹⁾ | — | — | — |
| RG2 | 5 | ANG2 | — | C3IN0+ | — | — | — | — | — | — | — | RX2 ⁽¹⁾ DT2 ⁽³⁾ | — | — | — |
| RG3 | 6 | ANG3 | — | C3IN0- | — | CCP4 ⁽¹⁾ | — | — | — | — | — | — | — | — | — |
| RG4 | 8 | ANG4 | — | C3IN1- | T5G ⁽¹⁾ T7CKI ⁽¹⁾ | CCP5 ⁽¹⁾ | — | — | — | — | — | — | — | — | — |
| RG5 | 7 | — | — | — | — | — | — | — | — | — | — | IOCG5 | — | — | MCLR, VPP |
| RG6 | 20 | ANG6 | — | — | — | — | — | — | SMT2WIN1 ⁽¹⁾ | — | — | — | — | — | — |
| RG7 | 19 | ANG7 | — | — | — | — | — | — | SMT2SIG1 ⁽¹⁾ | — | — | — | — | — | — |
| RH0 | 26 | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| RH1 | 25 | ADCACT ⁽¹⁾ | — | — | — | — | — | — | — | — | — | — | — | — | — |

- Note** 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 [Table 17-1](#) 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 [Register 17-2](#)
- 3: This is a bidirectional signal. For normal module operation, the firmware should map this signal to the same pin in both the PPS input and PPS output registers.
- 4: These pins are configured for I²C™ logic levels; The SCLx/SDAx signals may be assigned to any of these pins. PPS assignments to the other pins (e.g., RB1) will operate, but input logic levels will be standard TTL/ST as selected by the INLVL register, instead of the I²C specific or SMBus input buffer thresholds.

TABLE 1: 64-PIN ALLOCATION TABLE (PIC18(L)F6XK40) (CONTINUED)

| I/O ⁽²⁾ | 64-Pin TQFP, QFN | A/D | DAC | Comparator | Timers | CCP and PWM | CWG | ZCD | SMT | Clock Reference (CLKR) | Interrupt | EUSART | DSM | MSSP | Basic |
|--------------------|------------------|------------------|-----|-------------------------|--------|--|----------------------------------|-----|-----|------------------------|-----------|--|-----|------------------------------|-------|
| RH2 | 57 | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| RH3 | 56 | — | — | — | — | — | — | — | — | — | — | — | — | — | — |
| VDD | 10, 38 | — | — | — | — | — | — | — | — | — | — | — | — | — | VDD |
| VSS | 9, 41 | — | — | — | — | — | — | — | — | — | — | — | — | — | VSS |
| OUT ⁽²⁾ | — | ADGRDA ADGRDB | — | C1OUT C2OUT C3OUT | TMR0 | CCP1 CCP2 CCP3 CCP4 CCP5 PWM6OUT PWM7OUT | CWG1A CWG1B CWG1C CWG1D | — | — | CLKR | — | TX1/CK1 ⁽³⁾ DT1 ⁽³⁾ TX2/CK2 ⁽³⁾ DT2 ⁽³⁾ TX3/CK3 ⁽³⁾ DT3 ⁽³⁾ TX4/CK4 ⁽³⁾ DT4 ⁽³⁾ TX5/CK5 ⁽³⁾ DT5 ⁽³⁾ | DSM | SDO1 SCK1 SDO2 SCK2 | — |

- Note** 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 [Table 17-1](#) 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 [Register 17-2](#).
- 3: This is a bidirectional signal. For normal module operation, the firmware should map this signal to the same pin in both the PPS input and PPS output registers.
- 4: These pins are configured for I²C™ logic levels; The SCLx/SDAx signals may be assigned to any of these pins. PPS assignments to the other pins (e.g., RB1) will operate, but input logic levels will be standard TTL/ST as selected by the INLVL register, instead of the I²C specific or SMBus input buffer thresholds.

Table of Contents

| | | |
|------|---|-----|
| 1.0 | Device Overview | 11 |
| 2.0 | Guidelines for Getting Started with PIC18(L)F6xK40 Microcontrollers | 16 |
| 3.0 | Device Configuration | 19 |
| 4.0 | Oscillator Module (with Fail-Safe Clock Monitor) | 32 |
| 5.0 | Reference Clock Output Module | 51 |
| 6.0 | Power-Saving Operation Modes | 56 |
| 7.0 | Peripheral Module Disable (PMD) | 64 |
| 8.0 | Resets | 71 |
| 9.0 | Windowed Watchdog Timer (WWDT) | 80 |
| 10.0 | Memory Organization | 89 |
| 11.0 | Nonvolatile Memory (NVM) Control | 126 |
| 12.0 | 8x8 Hardware Multiplier | 150 |
| 13.0 | Cyclic Redundancy Check (CRC) Module with Memory Scanner | 152 |
| 14.0 | Interrupts | 169 |
| 15.0 | I/O Ports | 206 |
| 16.0 | Interrupt-on-Change | 217 |
| 17.0 | Peripheral Pin Select (PPS) Module | 221 |
| 18.0 | Timer0 Module | 231 |
| 19.0 | Timer1/3/5/7 Module with Gate Control | 237 |
| 20.0 | Timer2/4/6/8 Module | 253 |
| 21.0 | Capture/Compare/PWM Module | 274 |
| 22.0 | Pulse-Width Modulation (PWM 6/7) | 289 |
| 23.0 | Zero-Cross Detection (ZCD) Module | 297 |
| 24.0 | Complementary Waveform Generator (CWG) Module | 303 |
| 25.0 | Signal Measurement Timer (SMT) | 330 |
| 26.0 | Data Signal Modulator (DSM) Module | 374 |
| 27.0 | Master Synchronous Serial Port (MSSP1) Module | 385 |
| 28.0 | Enhanced Universal Synchronous Asynchronous Receiver Transmitter (EUSART) | 442 |
| 29.0 | Fixed Voltage Reference (FVR) | 473 |
| 30.0 | Temperature Indicator Module | 475 |
| 31.0 | 5-Bit Digital-to-Analog Converter (DAC) Module | 477 |
| 32.0 | Analog-to-Digital Converter with Computation (ADC2) Module | 481 |
| 33.0 | Comparator Module (C1/2/3) | 517 |
| 34.0 | High/Low-Voltage Detect (HLVD) | 527 |
| 35.0 | In-Circuit Serial Programming™ (ICSP™) | 534 |
| 36.0 | Instruction Set Summary | 536 |
| 37.0 | Development Support | 586 |
| 38.0 | Electrical Specifications | 590 |
| 39.0 | DC and AC Characteristics Graphs and Tables | 620 |
| 40.0 | Packaging Information | 621 |
| | Appendix A: Revision History | 628 |
| | Appendix B: Device Differences | 629 |
| | The Microchip Website | 630 |
| | Customer Change Notification Service | 630 |
| | Customer Support | 630 |
| | Product Identification System | 631 |

TO OUR VALUED CUSTOMERS

It is our intention to provide our valued customers with the best documentation possible to ensure successful use of your Microchip products. To this end, we will continue to improve our publications to better suit your needs. Our publications will be refined and enhanced as new volumes and updates are introduced.

If you have any questions or comments regarding this publication, please contact the Marketing Communications Department via E-mail at docerrors@microchip.com. We welcome your feedback.

Most Current Data Sheet

To obtain the most up-to-date version of this data sheet, please register at our Worldwide Website at:

<http://www.microchip.com>

You can determine the version of a data sheet by examining its literature number found on the bottom outside corner of any page. The last character of the literature number is the version number, (e.g., DS30000000A is version A of document DS30000000).

Errata

An errata sheet, describing minor operational differences from the data sheet and recommended workarounds, may exist for current devices. As device/documentation issues become known to us, we will publish an errata sheet. The errata will specify the revision of silicon and revision of document to which it applies.

To determine if an errata sheet exists for a particular device, please check with one of the following:

- Microchip's Worldwide Website; <http://www.microchip.com>
- Your local Microchip sales office (see last page)

When contacting a sales office, please specify which device, revision of silicon and data sheet (include literature number) you are using.

Customer Notification System

Register on our website at www.microchip.com to receive the most current information on all of our products.

1.0 DEVICE OVERVIEW

This document contains device specific information for the following devices:

- PIC18F67K40
- PIC18LF67K40

This family offers the advantages of all PIC18 microcontrollers – namely, high computational performance at an economical price – with the addition of high-endurance, Program Flash Memory. In addition to these features, the PIC18(L)F6xK40 family introduces design enhancements that make these microcontrollers a logical choice for many high-performance, power sensitive applications.

1.1 New Core Features

1.1.1 XLP TECHNOLOGY

All of the devices in the PIC18(L)F6xK40 family incorporate a range of features that can significantly reduce power consumption during operation. Key items include:

- **Alternate Run Modes:** By clocking the controller from the secondary oscillator or the internal oscillator block, power consumption during code execution can be reduced by as much as 90%.
- **Multiple Idle Modes:** The controller can also run with its CPU core disabled but the peripherals still active. In these states, power consumption can be reduced even further, to as little as 4% of normal operation requirements.
- **On-the-fly Mode Switching:** The power-managed modes are invoked by user code during operation, allowing the user to incorporate power-saving ideas into their application's software design.
- **Peripheral Module Disable:** Modules that are not being used in the code can be selectively disabled using the PMD module. This further reduces the power consumption.

1.1.2 MULTIPLE OSCILLATOR OPTIONS AND FEATURES

All of the devices in the PIC18(L)F6xK40 family offer several different oscillator options. The PIC18(L)F6xK40 family can be clocked from several different sources:

- **HFINTOSC**
 - 1-16 MHz precision digitally controlled internal oscillator
- **LFINTOSC**
 - 31 kHz internal oscillator
- **EXTOSC**
 - External clock (EC)
 - Low-power oscillator (LP)
 - Medium power oscillator (XT)
 - High-power oscillator (HS)
- **SOSC**
 - Secondary oscillator circuit operating at 31 kHz
- A Phase Lock Loop (PLL) frequency multiplier (4x) is available to both the External and Internal Oscillator modes enabling clock speeds of up to 64 MHz

Besides its availability as a clock source, the internal oscillator block provides a stable reference source that gives the family additional features for robust operation:

- **Fail-Safe Clock Monitor:** This option constantly monitors the main clock source against a reference signal provided by the LFINTOSC. If a clock failure occurs, the controller is switched to the internal oscillator block, allowing for continued operation or a safe application shutdown.
- **Two-Speed Start-up:** This option allows the internal oscillator to serve as the clock source from Power-on Reset, or wake-up from Sleep mode, until the primary clock source is available.

1.2 Other Special Features

- **Memory Endurance:** The Flash cells for both program memory and data EEPROM are rated to last for many thousands of erase/write cycles – up to 10K for program memory and 100K for EEPROM. Data retention without refresh is conservatively estimated to be greater than 40 years.
- **Self-programmability:** These devices can write to their own program memory spaces under internal software control. By using a boot loader routine located in the protected Boot Block at the top of program memory, it becomes possible to create an application that can update itself in the field.
- **Extended Instruction Set:** The PIC18(L)F6xK40 family introduces an optional extension to the PIC18 instruction set, which adds eight new instructions and an Indexed Addressing mode. This extension, enabled as a device configuration option, has been specifically designed to optimize re-entrant application code originally developed in high-level languages, such as C.
- **Enhanced Peripheral Pin Select:** 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. All analog inputs and outputs remain fixed to their assigned pins.
- **Enhanced Addressable EUSART:** This serial communication module is capable of standard RS-232 operation and provides support for the LIN bus protocol. Other enhancements include automatic baud rate detection and a 16-bit Baud Rate Generator for improved resolution. When the microcontroller is using the internal oscillator block, the EUSART provides stable operation for applications that talk to the outside world without using an external crystal (or its accompanying power requirement).
- **10-bit A/D Converter with Computation:** This module incorporates programmable acquisition time, allowing for a channel to be selected and a conversion to be initiated without waiting for a sampling period and thus, reduce code overhead. It has a new module called ADC² with computation features, which provides a digital filter and threshold interrupt functions.
- **Windowed Watchdog Timer (WWDT):**
 - Timer monitoring of overflow and underflow events
 - Variable prescaler selection
 - Variable window size selection
 - All sources configurable in hardware or software

1.3 Details on Individual Family Members

Devices in the PIC18(L)F6xK40 family are available in 64-pin packages. The block diagram for this device is shown in [Figure 1-1](#).

The devices have the following differences:

1. Program Flash Memory
2. Data Memory SRAM
3. Input Voltage Range/Power Consumption

All other features for devices in this family are identical. These are summarized in [Table 1-1](#).

The pinouts for all devices are listed in the pin summary tables ([Table 1](#)).

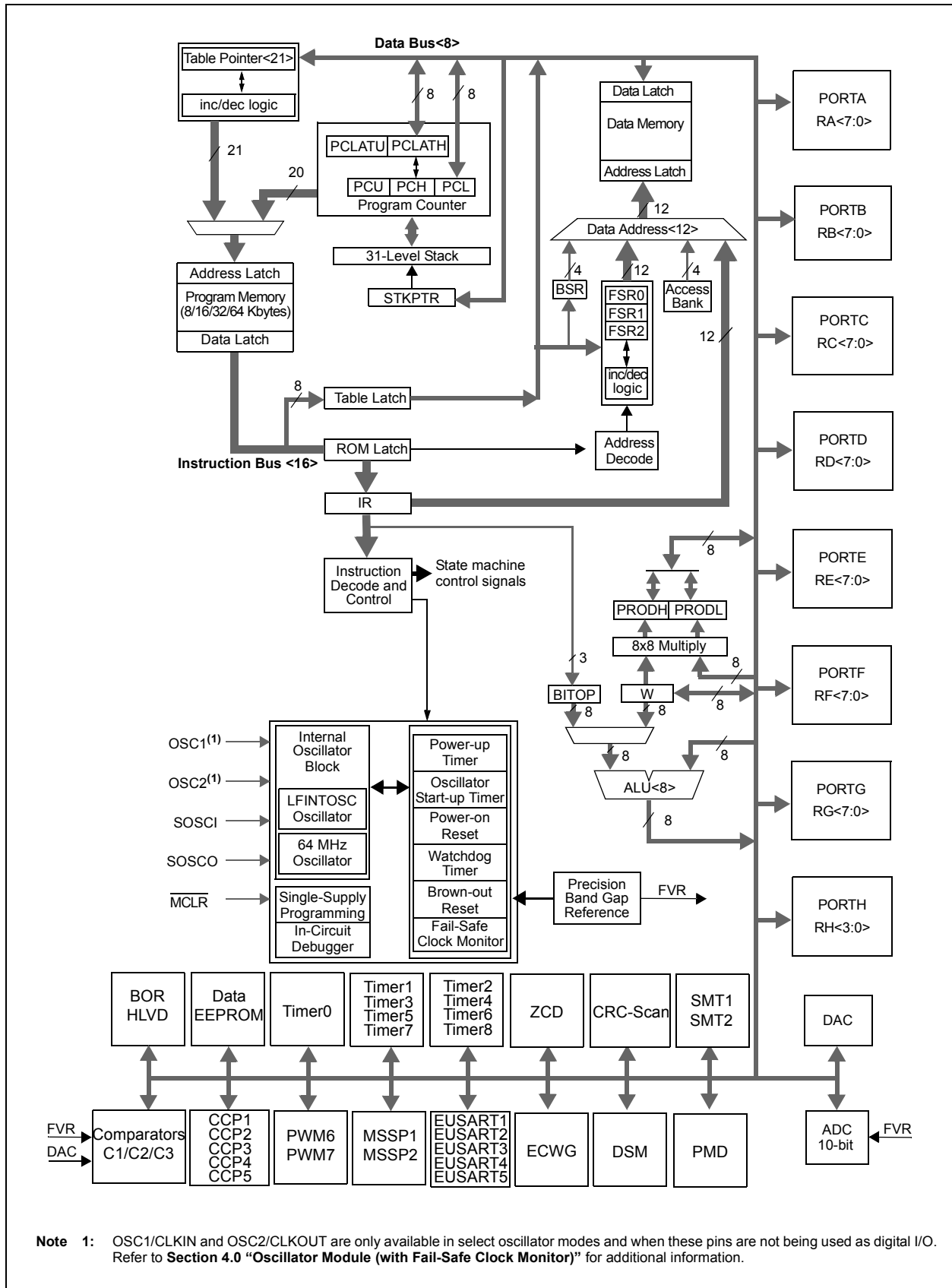
PIC18(L)F67K40

TABLE 1-1: DEVICE FEATURES

| Features | PIC18(L)F65K40 | PIC18(L)F66K40 | PIC18(L)F67K40 |
|--|--|-----------------|------------------|
| Program Memory (Bytes) | 32768 | 65536 | 131072 |
| Program Memory (Instructions) | 16384 | 32768 | 65536 |
| Data Memory (Bytes) | 2048 | 3562 | 3562 |
| Data EEPROM Memory (Bytes) | 1024 | 1024 | 1024 |
| I/O Ports | A,B,C,D,E,F,G,H | A,B,C,D,E,F,G,H | A,B, C,D,E,F,G,H |
| Capture/Compare/PWM Modules (CCP) | 5 | | |
| 10-Bit Pulse-Width Modulator (PWM) | 2 | | |
| 10-Bit Analog-to-Digital Module (ADC ²) with Computation Accelerator | 4 internal 47 external | | |
| Packages | 64-pin TQFP 64-pin QFN | | |
| Interrupt Sources | 56 | | |
| Timers (16-/8-bit) | 5/4 | | |
| Serial Communications | 2 MSSP, 5 EUSART | | |
| Enhanced Complementary Waveform Generator (ECWG) | 1 | | |
| Signal Measurement Timer (SMT) | 2 | | |
| Comparators | 3 | | |
| Zero-Cross Detect (ZCD) | 1 | | |
| Data Signal Modulator (DSM) | 1 | | |
| Peripheral Pin Select (PPS) | Yes | | |
| Peripheral Module Disable (PMD) | Yes | | |
| 16-bit CRC with NVMSCAN | Yes | | |
| Programmable High/Low-Voltage Detect (HLVD) | Yes | | |
| Programmable Brown-out Reset (BOR) | Yes | | |
| Resets (and Delays) | POR, BOR, RESET Instruction, Stack Overflow, Stack Underflow (PWRT, OST), MCLR, WDT | | |
| Instruction Set | 75 Instructions; 83 with Extended Instruction Set enabled | | |
| Operating Frequency | DC – 64 MHz | | |

PIC18(L)F67K40

FIGURE 1-1: PIC18(L)F67K40 FAMILY BLOCK DIAGRAM



1.4 Register and Bit naming conventions

1.4.1 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.

1.4.2 BIT NAMES

There are two variants for bit names:

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

1.4.2.1 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 *RegisterName*bits.*ShortName*. For example, the enable bit, EN, in the COG1CON0 register can be set in C programs with the instruction `COG1CON0bits.EN = 1`.

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

1.4.2.2 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 COG1 enable bit is the COG1 prefix, G1, appended with the enable bit short name, EN, resulting in the unique bit name G1EN.

Long bit names are useful in both C and assembly programs. For example, in C the COG1CON0 enable bit can be set with the `G1EN = 1` instruction. In assembly, this bit can be set with the `BSF COG1CON0,G1EN` instruction.

1.4.2.3 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 COG1CON0 register contain the mode control bits. The short name for this field is MD. There is no long bit name variant. Bit field access is only possible in C programs. The following example demonstrates a C program instruction for setting the COG1 to the Push-Pull mode:

```
COG1CON0bits.MD = 0x5;
```

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 G1MD2. The following two examples demonstrate assembly program sequences for setting the COG1 to Push-Pull mode:

Example 1:

```
MOVLW ~(1<<G1MD1)
ANDWF COG1CON0,F
MOVLW 1<<G1MD2 | 1<<G1MD0
IORWF COG1CON0,F
```

Example 2:

```
BSF COG1CON0,G1MD2
BCF COG1CON0,G1MD1
BSF COG1CON0,G1MD0
```

1.4.3 REGISTER AND BIT NAMING EXCEPTIONS

1.4.3.1 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.

1.4.3.2 Legacy Peripherals

There are some peripherals that do not strictly adhere to these naming conventions. Peripherals that have existed for many years and are present in almost every device are the exceptions. These exceptions were necessary to limit the adverse impact of the new conventions on legacy code. Peripherals that do adhere to the new convention will include a table in the registers section indicating the long name prefix for each peripheral instance. Peripherals that fall into the exception category will not have this table. These peripherals include, but are not limited to, the following:

- EUSART
- MSSP

2.0 GUIDELINES FOR GETTING STARTED WITH PIC18(L)F6XK40 MICROCONTROLLERS

2.1 Basic Connection Requirements

Getting started with the PIC18(L)F6xK40 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 VDD and VSS pins (see [Section 2.2 “Power Supply Pins”](#))
- MCLR pin (see [Section 2.3 “Master Clear \(MCLR\) Pin”](#))

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

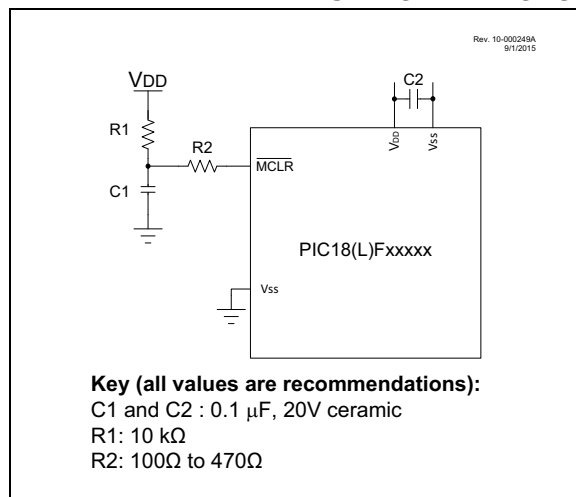
- PGC/PGD pins used for In-Circuit Serial Programming™ (ICSP™) and debugging purposes (see [Section 2.4 “ICSP™ Pins”](#))
- OSCI and OSCO pins when an external oscillator source is used (see [Section 2.5 “External Oscillator Pins”](#))

Additionally, the following pins may be required:

- VREF+/VREF- pins are used when external voltage reference for analog modules is implemented

The minimum mandatory connections are shown in [Figure 2-1](#).

FIGURE 2-1: RECOMMENDED MINIMUM CONNECTIONS



2.2 Power Supply Pins

2.2.1 DECOUPLING CAPACITORS

The use of decoupling capacitors on every pair of power supply pins (VDD and VSS) is required.

Consider the following criteria when using decoupling capacitors:

- **Value and type of capacitor:** A 0.1 μ F (100 nF), 10-20V capacitor is recommended. The capacitor should 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 should 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).
- **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 μ F to 0.001 μ F. 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 μ F in parallel with 0.001 μ F).
- **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.

2.2.2 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 should 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 so that it meets the acceptable voltage sag at the device. Typical values range from 4.7 μ F to 47 μ F.

2.3 Master Clear ($\overline{\text{MCLR}}$) Pin

The $\overline{\text{MCLR}}$ 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 2-1. 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 $\overline{\text{MCLR}}$ 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 $\overline{\text{MCLR}}$ pin during programming and debugging operations by using a jumper (Figure 2-2). The jumper is replaced for normal run-time operations.

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

2.4 ICSP™ Pins

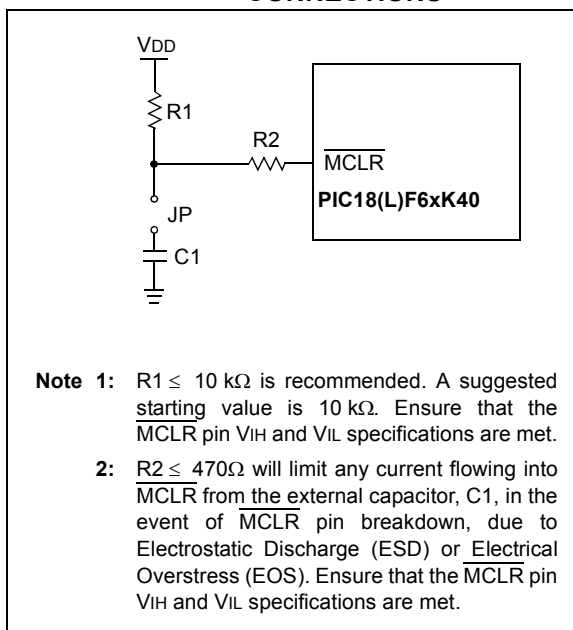
The PGC and PGD pins are used for In-Circuit Serial Programming™ (ICSP™) and debugging purposes. It is recommended to keep the trace length between the ICSP connector and the ICSP pins on the device as short as possible. If the ICSP connector is expected to experience an ESD event, a series resistor is recommended, with the value in the range of a few tens of ohms, not to exceed 100Ω.

Pull-up resistors, series diodes and capacitors on the PGC and PGD pins are not recommended as they will interfere with the programmer/debugger communications to the device. If such discrete components are an application requirement, they should 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., PGCx/PGDx pins), programmed into the device, matches the physical connections for the ICSP to the Microchip debugger/emulator tool.

For more information on available Microchip development tools connection requirements, refer to [Section 37.0 “Development Support”](#).

FIGURE 2-2: EXAMPLE OF $\overline{\text{MCLR}}$ PIN CONNECTIONS



2.5 External Oscillator Pins

Many microcontrollers have options for at least two oscillators: a high-frequency primary oscillator and a low-frequency secondary oscillator (refer to **Section 4.0 “Oscillator Module (with Fail-Safe Clock Monitor)”** for details).

The oscillator circuit should 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 should 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 should 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 [Figure 2-3](#). 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.

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).

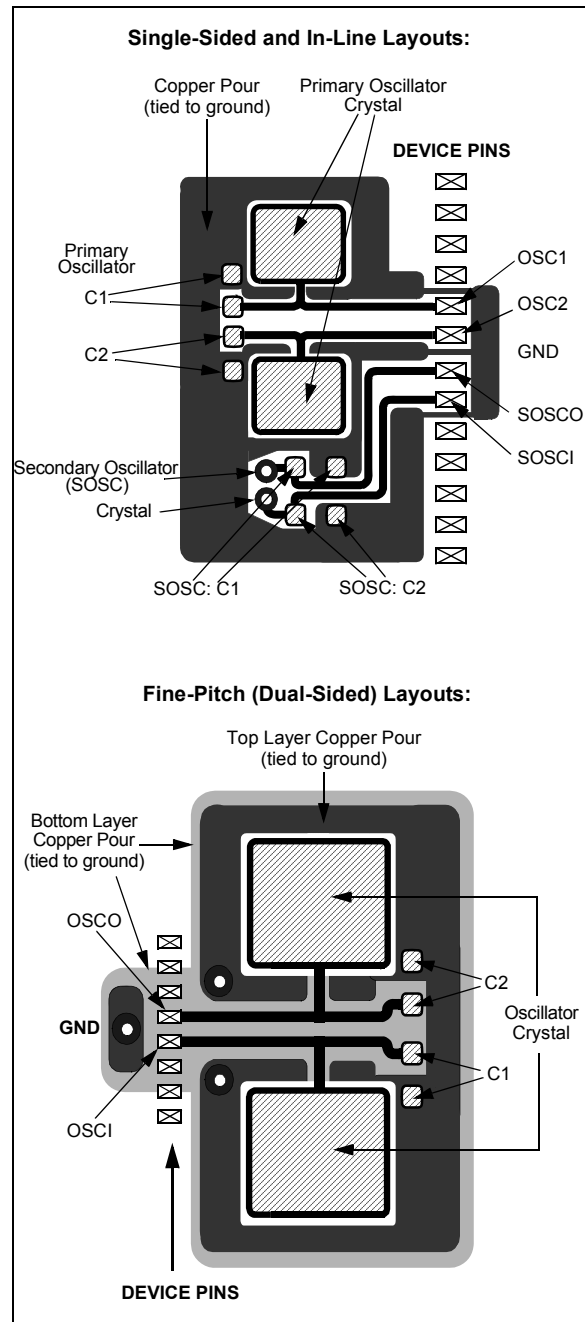
For additional information and design guidance on oscillator circuits, please refer to these Microchip Application Notes, available at the corporate website (www.microchip.com):

- AN826, “Crystal Oscillator Basics and Crystal Selection for rPIC™ and PICmicro® Devices”
- AN849, “Basic PICmicro® Oscillator Design”
- AN943, “Practical PICmicro® Oscillator Analysis and Design”
- AN949, “Making Your Oscillator Work”

2.6 Unused I/Os

Unused I/O pins should be configured as outputs and driven to a logic low state. Alternatively, connect a 1 kΩ to 10 kΩ resistor to Vss on unused pins and drive the output to logic low.

FIGURE 2-3: SUGGESTED PLACEMENT OF THE OSCILLATOR CIRCUIT



3.0 DEVICE CONFIGURATION

Device configuration consists of Configuration Words, Code Protection, Device ID and Rev ID.

3.1 Configuration Words

There are six Configuration Word bits that allow the user to setup the device with several choices of oscillators, Resets and memory protection options. These are implemented as Configuration Word 1 through Configuration Word 6 at 300000h through 30000Bh.

| |
|--|
| <p>Note: The <code>DEBUG</code> bit in Configuration Words is managed automatically by device development tools including debuggers and programmers. For normal device operation, this bit should be maintained as a '1'.</p> |
|--|

3.2 Register Definitions: Configuration Words

REGISTER 3-1: Configuration Word 1L (30 0000h): Oscillators

| | | | | | | | |
|-------|-------------|-------|-------|-------|--------------|-------|-------|
| U-1 | R/W-1 | R/W-1 | R/W-1 | U-1 | R/W-1 | R/W-1 | R/W-1 |
| — | RSTOSC<2:0> | | | — | FEXTOSC<2:0> | | |
| bit 7 | | | | bit 0 | | | |

Legend:

| | | |
|-----------------------------|------------------|------------------------------------|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '1' |
| -n = Value for blank device | '1' = Bit is set | '0' = Bit is cleared |
| | | x = Bit is unknown |

bit 7 **Unimplemented:** Read as '1'

bit 6-4 **RSTOSC<2:0>:** Power-up Default Value for COSC bits
This value is the Reset default value for COSC and selects the oscillator first used by user software. Refer to COSC operation.

111 = EXTOSC operating per FEXTOSC bits (device manufacturing default)

110 = HFINTOSC with HFFRQ = 4 MHz (Register 4-5) and CDIV = 4:1 (Register 4-2)

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 (Register 4-5) and CDIV = 1:1 (Register 4-2). Resets COSC/NOSC to 3'b110.

bit 3 **Unimplemented:** Read as '1'

bit 2-0 **FEXTOSC<2:0>:** FEXTOSC External Oscillator Mode Selection bits

111 = EC (external clock) above 8 MHz; PFM set to high power (device manufacturing default)

110 = EC (external clock) for 500 kHz to 8 MHz; PFM set to medium power

101 = EC (external clock) below 500 kHz; PFM set to low power

100 = Oscillator not enabled

011 = Reserved (do not use)

010 = HS (crystal oscillator) above 8 MHz; PFM set to high power

001 = XT (crystal oscillator) above 500 kHz, below 8 MHz; PFM set to medium power

000 = LP (crystal oscillator) optimized for 32.768 kHz; PFM set to low power

REGISTER 3-2: Configuration Word 1H (30 0001h): Oscillators

| | | | | | | | |
|-------|-----|-------|-----|-------|-----|-----|----------|
| U-1 | U-1 | R/W-1 | U-1 | R/W-1 | U-1 | U-1 | R/W-1 |
| — | — | FCMEN | — | CSWEN | — | — | CLKOUTEN |
| bit 7 | | | | | | | bit 0 |

Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '1'

-n = Value for blank device

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7-6 **Unimplemented:** Read as '1'

bit 5 **FCMEN:** Fail-Safe Clock Monitor Enable bit
 1 = FSCM timer enabled
 0 = FSCM timer disabled

bit 4 **Unimplemented:** Read as '1'

bit 3 **CSWEN:** Clock Switch Enable bit
 1 = Writing to NOSC and NDIV is allowed
 0 = The NOSC and NDIV bits cannot be changed by user software

bit 2-1 **Unimplemented:** Read as '1'

bit 0 **CLKOUTEN:** Clock Out Enable bit
If FEXTOSC = HS, XT, LP, then this bit is ignored
Otherwise:
 1 = CLKOUT function is disabled; I/O or oscillator function on OSC2
 0 = CLKOUT function is enabled; FOSC/4 clock appears at OSC2

REGISTER 3-3: Configuration Word 2L (30 0002h): Supervisor

| | | | | | | | |
|------------|-------|---------|-----|-----|-----|-------|-------|
| R/W-1 | R/W-1 | R/W-1 | U-1 | U-1 | U-1 | R/W-1 | R/W-1 |
| BOREN<1:0> | | LPBOREN | — | — | — | PWRTE | MCLRE |
| bit 7 | | | | | | | bit 0 |

Legend:

R = Readable bit W = Writable bit U = Unimplemented bit, read as '1'
 -n = Value for blank device '1' = Bit is set '0' = Bit is cleared x = Bit is unknown

- bit 7-6 **BOREN<1:0>**: Brown-out Reset Enable bits
 When enabled, Brown-out Reset Voltage (VBOR) is set by BORV bit
 11 = Brown-out Reset enabled, 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 **LPBOREN**: Low-Power BOR Enable bit
 1 = Low-Power Brown-out Reset is disabled
 0 = Low-Power Brown-out Reset is enabled
- bit 4-2 **Unimplemented**: Read as '1'
- bit 1 **PWRTE**: Power-up Timer Enable bit
 1 = PWRT disabled
 0 = PWRT enabled
- bit 0 **MCLRE**: Master Clear ($\overline{\text{MCLR}}$) Enable bit
 If $\text{LVP} = 1$
 RE3 pin function is $\overline{\text{MCLR}}$
 If $\text{LVP} = 0$
 1 = $\overline{\text{MCLR}}$ pin is $\overline{\text{MCLR}}$
 0 = $\overline{\text{MCLR}}$ pin function is port defined function

REGISTER 3-4: Configuration Word 2H (30 0003h): Supervisor

| R/W-1 | U-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 |
|---------------------------|-----|---------------------------|--------|---------|-------------------------|-----------|-------|
| $\overline{\text{XINST}}$ | — | $\overline{\text{DEBUG}}$ | STVREN | PPS1WAY | $\overline{\text{ZCD}}$ | BORV<1:0> | |
| bit 7 | | | | | | | bit 0 |

Legend:

R = Readable bit W = Writable bit U = Unimplemented bit, read as '1'
 -n = Value for blank device '1' = Bit is set '0' = Bit is cleared x = Bit is unknown

- bit 7 **$\overline{\text{XINST}}$** : Extended Instruction Set Enable bit
 1 = Extended Instruction Set and Indexed Addressing mode disabled (Legacy mode)
 0 = Extended Instruction Set and Indexed Addressing mode enabled
- bit 6 **Unimplemented**: Read as '1'
- bit 5 **$\overline{\text{DEBUG}}$** : Debugger Enable bit
 1 = Background debugger disabled
 0 = Background debugger enabled
- bit 4 **STVREN**: Stack Overflow/Underflow Reset Enable bit
 1 = Stack Overflow or Underflow will cause a Reset
 0 = Stack Overflow or Underflow will not cause a Reset
- bit 3 **PPS1WAY**: PPSLOCKED bit One-Way Set Enable bit
 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 (provided an unlocking sequence is executed)
- bit 2 **$\overline{\text{ZCD}}$** : ZCD Disable bit
 1 = ZCD disabled. ZCD can be enabled by setting the ZCDSEN bit of ZCDCON
 0 = ZCD always enabled, ZCDMD bit is ignored
- bit 1-0 **BORV<1:0>**: Brown-out Reset Voltage Selection bit⁽¹⁾
 PIC18F6xK40 device:
 11 = Brown-out Reset Voltage (VBOR) set to 2.45V
 10 = Brown-out Reset Voltage (VBOR) set to 2.45V
 01 = Brown-out Reset Voltage (VBOR) set to 2.7V
 00 = Brown-out Reset Voltage (VBOR) set to 2.85V
 PIC18LF6xK40 device:
 11 = Brown-out Reset Voltage (VBOR) set to 1.90V
 10 = Brown-out Reset Voltage (VBOR) set to 2.45V
 01 = Brown-out Reset Voltage (VBOR) set to 2.7V
 00 = Brown-out Reset Voltage (VBOR) set to 2.85V

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

PIC18(L)F67K40

REGISTER 3-5: CONFIGURATION WORD 3L (30 0004h): WINDOWED WATCHDOG TIMER

| | | | | | | | |
|-------|-----------|-------|-------------|-------|-------|-------|-------|
| U-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 |
| — | WDTE<1:0> | | WDTCP5<4:0> | | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

R = Readable bit W = Writable bit U = Unimplemented bit, read as '1'
 -n = Value for blank device '1' = Bit is set '0' = Bit is cleared x = Bit is unknown

bit 7 **Unimplemented:** Read as '1'

bit 6-5 **WDTE<1:0>:** WDT Operating Mode bits
 11 = WDT enabled regardless of Sleep; SEN bit in WDTCON0 is ignored
 10 = WDT enabled while Sleep = 0, suspended when Sleep = 1; SEN bit in WDTCON0 is ignored
 01 = WDT enabled/disabled by SEN bit in WDTCON0
 00 = WDT disabled, SEN bit in WDTCON0 is ignored

bit 4-0 **WDTCP5<4:0>:** WDT Period Select bits

| WDTCP5 | 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 |
| 10011 | 10011 | 1:32 | 2 ⁵ | 1 ms | No |
| ... | ... | | | | |
| 11110 | 11110 | | | | |
| 10010 | 10010 | 1:8388608 | 2 ²³ | 256s | No |
| 10001 | 10001 | 1:4194304 | 2 ²² | 128s | |
| 10000 | 10000 | 1:2097152 | 2 ²¹ | 64s | |
| 01111 | 01111 | 1:1048576 | 2 ²⁰ | 32s | |
| 01110 | 01110 | 1:524299 | 2 ¹⁹ | 16s | |
| 01101 | 01101 | 1:262144 | 2 ¹⁸ | 8s | |
| 01100 | 01100 | 1:131072 | 2 ¹⁷ | 4s | |
| 01011 | 01011 | 1:65536 | 2 ¹⁶ | 2s | |
| 01010 | 01010 | 1:32768 | 2 ¹⁵ | 1s | |
| 01001 | 01001 | 1:16384 | 2 ¹⁴ | 512 ms | |
| 01000 | 01000 | 1:8192 | 2 ¹³ | 256 ms | |
| 00111 | 00111 | 1:4096 | 2 ¹² | 128 ms | |
| 00110 | 00110 | 1:2048 | 2 ¹¹ | 64 ms | |
| 00101 | 00101 | 1:1024 | 2 ¹⁰ | 32 ms | |
| 00100 | 00100 | 1:512 | 2 ⁹ | 16 ms | |
| 00011 | 00011 | 1:256 | 2 ⁸ | 8 ms | |
| 00010 | 00010 | 1:128 | 2 ⁷ | 4 ms | |
| 00001 | 00001 | 1:64 | 2 ⁶ | 2 ms | |
| 00000 | 00000 | 1:32 | 2 ⁵ | 1 ms | |

REGISTER 3-6: CONFIGURATION WORD 3H (30 0005h): WINDOWED WATCHDOG TIMER

| | | | | | | | | |
|-------|-----|-------------|-------|-------|-------------|-------|-------|-------|
| U-1 | U-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | |
| — | — | WDTCCS<2:0> | | | WDTCWS<2:0> | | | |
| bit 7 | | | | | | | | bit 0 |

Legend:

R = Readable bit W = Writable bit U = Unimplemented bit, read as '1'
 -n = Value for blank device '1' = Bit is set '0' = Bit is cleared x = Bit is unknown

bit 7-6 **Unimplemented:** Read as '1'

bit 5-3 **WDTCCS<2:0>:** WDT Input Clock Selector bits

If WDTE<1:0> fuses = 2'b00

This bit is ignored.

Otherwise:

111 = Software Control

110 = Reserved (Default to LFINTOSC)

.

.

.

010 = Reserved (Default to LFINTOSC)

001 = WDT reference clock is the 31.25 kHz HFINTOSC (MFINTOSC)

000 = WDT reference clock is the 31.0 kHz LFINTOSC (default value)

bit 2-0 **WDTCWS<2:0>:** WDT Window Select bits

| WDTCWS | 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 | 111 | 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 | | |

Register 3-7: Configuration Word 4L (30 0006h): Memory Write Protection

| R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 |
|-------|-------|-------|-------|-------|-------|-------|-------|
| WRT7 | WRT6 | WRT5 | WRT4 | WRT3 | WRT2 | WRT1 | WRT0 |
| bit 7 | | | | | | | bit 0 |

Legend:

R = Readable bit W = Writable bit U = Unimplemented bit, read as '1'
 -n = Value for blank device '1' = Bit is set '0' = Bit is cleared x = Bit is unknown

bit 7-0 **WRT<7:0>**: User NVM Self-Write Protection bits⁽¹⁾
 1 = Corresponding Memory Block NOT write-protected
 0 = Corresponding Memory Block write-protected

Note 1: Refer to [Table 10-2](#) for details on implementation of the individual WRT bits.

Register 3-8: Configuration Word 4H (30 0007h): Memory Write Protection

| U-1 | U-1 | R/W-1 | R/W-1 | U-1 | R/W-1 | R/W-1 | R/W-1 |
|-------|-----|-------|-------|-----|-------|-------|-------|
| — | — | LVP | SCANE | — | WRTD | WRTB | WRTC |
| bit 7 | | | | | | | bit 0 |

Legend:

R = Readable bit W = Writable bit U = Unimplemented bit, read as '1'
 -n = Value for blank device '1' = Bit is set '0' = Bit is cleared x = Bit is unknown

bit 7-6 **Unimplemented:** Read as '1'

bit 5 **LVP:** Low-Voltage Programming Enable bit
 1 = Low-voltage programming enabled. $\overline{\text{MCLR}}/\text{VPP}$ pin function is $\overline{\text{MCLR}}$. MCLRE Configuration bit is ignored.
 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.
 0 = HV on $\overline{\text{MCLR}}/\text{VPP}$ must be used for programming

bit 4 **SCANE:** Scanner Enable bit
 1 = Scanner module is available for use, SCANMD bit enables the module
 0 = Scanner module is NOT available for use, SCANMD bit is ignored

bit 3 **Unimplemented:** Read as '1'

bit 2 **WRTD:** Data EEPROM Write Protection bit
 1 = Data EEPROM NOT write-protected
 0 = Data EEPROM write-protected

bit 1 **WRTB:** Boot Block Write Protection bit
 1 = Boot Block NOT write-protected
 0 = Boot Block write-protected

bit 0 **WRTC:** Configuration Register Write Protection bit
 1 = Configuration Register NOT write-protected
 0 = Configuration Register write-protected

PIC18(L)F67K40

REGISTER 3-9: Configuration Word 5L (30 0008h): Code Protection

| | | | | | | | |
|-------|-----|-----|-----|-----|-----|-------------------------|------------------------|
| U-1 | U-1 | U-1 | U-1 | U-1 | U-1 | R/W-1 | R/W-1 |
| — | — | — | — | — | — | $\overline{\text{CPD}}$ | $\overline{\text{CP}}$ |
| bit 7 | | | | | | bit 0 | |

Legend:

R = Readable bit W = Writable bit U = Unimplemented bit, read as '1'
 -n = Value for blank device '1' = Bit is set '0' = Bit is cleared x = Bit is unknown

bit 7-2 **Unimplemented:** Read as '1'
 bit 1 **CPD:** Data NVM Memory Code Protection bit
 1 = Data NVM code protection disabled
 0 = Data NVM code protection enabled
 bit 0 **CP:** User NVM Program Memory Code Protection bit
 1 = User NVM code protection disabled
 0 = User NVM code protection enabled

REGISTER 3-10: Configuration Word 6L (30 000Ah): Memory Read Protection

| | | | | | | | |
|-------|-------|-------|-------|-------|-------|-------|-------|
| R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 |
| EBTR7 | EBTR6 | EBTR5 | EBTR4 | EBTR3 | EBTR2 | EBTR1 | EBTR0 |
| bit 7 | | | | | | bit 0 | |

Legend:

R = Readable bit W = Writable bit U = Unimplemented bit, read as '1'
 -n = Value for blank device '1' = Bit is set '0' = Bit is cleared x = Bit is unknown

bit 7-0 **EBTR<7:0>:** Table Read Protection bits⁽¹⁾
 1 = Corresponding Memory Block NOT protected from table reads executed in other blocks
 0 = Corresponding Memory Block protected from table reads executed in other blocks

Note 1: Refer to [Table 10-2](#) for details on implementation of the individual EBTR bits.

REGISTER 3-11: Configuration Word 6H (30 000Bh): Memory Read Protection

| | | | | | | | |
|-------|-----|-----|-----|-----|-----|-------|-----|
| U-1 | U-1 | U-1 | U-1 | U-1 | U-1 | R/W-1 | U-1 |
| — | — | — | — | — | — | EBTRB | — |
| bit 7 | | | | | | bit 0 | |

Legend:

R = Readable bit W = Writable bit U = Unimplemented bit, read as '1'
 -n = Value for blank device '1' = Bit is set '0' = Bit is cleared x = Bit is unknown

bit 7-2 **Unimplemented:** Read as '1'
 bit 1 **EBTRB:** Table Read Protection bit
 1 = Memory Boot Block NOT protected from table reads executed in other blocks
 0 = Memory Boot Block protected from table reads executed in other blocks
 bit 0 **Unimplemented:** Read as '1'

TABLE 3-1: SUMMARY OF CONFIGURATION WORDS

| Address | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Default/ Unprogrammed Value |
|----------|----------|--------|-----------|-------------|--------------|---------|-------------|----------|----------|-----------------------------------|
| 30 0000h | CONFIG1L | — | RSTOSC2 | RSTOSC1 | RSTOSC0 | — | FEXTOSC2 | FEXTOSC1 | FEXTOSC0 | 1111 1111 |
| 30 0001h | CONFIG1H | — | — | FCMEN | — | CSWEN | — | — | CLKOUTEN | 1111 1111 |
| 30 0002h | CONFIG2L | BOREN1 | BOREN0 | LPBOREN | — | — | — | PWRTE | MCLRE | 1111 1111 |
| 30 0003h | CONFIG2H | XINST | — | DEBUG | STVREN | PPS1WAY | ZCD | BORV1 | BORV0 | 1111 1111 |
| 30 0004h | CONFIG3L | — | WDTE<1:0> | | WDTCCPS<4:0> | | | | | 1111 1111 |
| 30 0005h | CONFIG3H | — | — | WDTCCS<2:0> | | | WDTCWS<2:0> | | | 1111 1111 |
| 30 0006h | CONFIG4L | WRT7 | WRT6 | WRT5 | WRT4 | WRT3 | WRT2 | WRT1 | WRT0 | 1111 1111 |
| 30 0007h | CONFIG4H | — | — | LVP | SCANE | — | WRTD | WRTB | WRTC | 1111 1111 |
| 30 0008h | CONFIG5L | — | — | — | — | — | — | CPD | CP | 1111 1111 |
| 30 000Ah | CONFIG6L | EBTR7 | EBTR6 | EBTR5 | EBTR4 | EBTR3 | EBTR2 | EBTR1 | EBTR0 | 1111 1111 |
| 30 000Bh | CONFIG6H | — | — | — | — | — | — | EBTRB | — | 1111 1111 |

3.3 Code Protection

Code protection allows the device to be protected from unauthorized access. Program memory protection and data memory are controlled independently. Internal access to the program memory is unaffected by any code protection setting.

3.3.1 PROGRAM MEMORY PROTECTION

The entire program memory space is protected from external reads and writes by the CP bit in Configuration Words. When $\overline{CP} = 0$, external reads and writes of program memory are inhibited and a read will return all '0's. The CPU can continue to read program memory, regardless of the protection bit settings. Self-writing the program memory is dependent upon the write protection setting. See [Section 3.4 "Write Protection"](#) for more information.

3.3.2 DATA MEMORY PROTECTION

The entire Data EEPROM Memory space is protected from external reads and writes by the CPD bit in the Configuration Words. When $\overline{CPD} = 0$, external reads and writes of Data EEPROM Memory are inhibited and a read will return all '0's. The CPU can continue to read Data EEPROM Memory regardless of the protection bit settings.

3.4 Write Protection

Write protection allows the device to be protected from unintended self-writes. Applications, such as boot loader software, can be protected while allowing other regions of the program memory to be modified.

The WRT<1:0> bits in Configuration Words define the size of the program memory block that is protected.

3.5 User ID

Eight words in the memory space (200000h-200000Fh) 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 [Section 11.2 "User ID, Device ID and Configuration Word Access"](#) for more information on accessing these memory locations. For more information on checksum calculation, see the "PIC18(L)F67K40 Memory Programming Specification" (DS40001822).

3.6 Device ID and Revision ID

The 16-bit device ID word is located at 3F FFEh and the 16-bit revision ID is located at 3F FFFCh. 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 Words. Refer to [11.0 “Nonvolatile Memory \(NVM\) Control”](#) for more information on accessing these locations.

3.7 Register Definitions: Device and Revision

REGISTER 3-12: DEVICE ID: DEVICE ID REGISTER

| | | | | | | | |
|--------|-------|-------|-------|-------|-------|------|-------|
| R | R | R | R | R | R | R | R |
| DEV15 | DEV14 | DEV13 | DEV12 | DEV11 | DEV10 | DEV9 | DEV8 |
| bit 15 | | | | | | | bit 8 |

| | | | | | | | |
|-------|------|------|------|------|------|------|-------|
| R | R | R | R | R | R | R | R |
| DEV7 | DEV6 | DEV5 | DEV4 | DEV3 | DEV2 | DEV1 | DEV0 |
| bit 7 | | | | | | | bit 0 |

Legend:

R = Readable bit '1' = Bit is set 0' = Bit is cleared x = Bit is unknown

bit 15-0

DEV<15:0>: Device ID bits

| Device | Device ID |
|--------------|-----------|
| PIC18F67K40 | 6AC0h |
| PIC18LF67K40 | 6B20h |

REGISTER 3-13: REVISION ID: REVISION ID REGISTER

| | | | | | | | |
|--------|---|---|---|-------------|---|---|---|
| R | R | R | R | R | R | R | R |
| 1 | 0 | 1 | 0 | MJRREV<5:2> | | | |
| bit 15 | | | | bit 8 | | | |

| | | | | | | | |
|-------------|---|-------------|---|---|---|---|---|
| R | R | R | R | R | R | R | R |
| MJRREV<1:0> | | MNRREV<5:0> | | | | | |
| bit 7 | | bit 0 | | | | | |

Legend:

R = Readable bit '1' = Bit is set 0' = Bit is cleared x = Bit is unknown

- bit 15-12 **Read as '1010'**
These bits are fixed with value '1010' for all devices in this family.
- bit 11-6 **MJRREV<5:0>**: Major Revision ID bits
These bits are used to identify a major revision. A major revision is indicated by an all-layer revision (A0, B0, C0, etc.).
Revision A = 6'b00_0000
- bit 5-0 **MNRREV<5:0>**: Minor Revision ID bits
These bits are used to identify a minor revision.

4.0 OSCILLATOR MODULE (WITH FAIL-SAFE CLOCK MONITOR)

4.1 Overview

The oscillator module has 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. [Figure 4-1](#) illustrates a block diagram of the oscillator module.

Clock sources can be supplied from external oscillators, quartz-crystal resonators and ceramic resonators. In addition, the system clock source can be supplied from one of two internal oscillators and PLL circuits, with a choice of speeds selectable via software. Additional clock features include:

- Selectable system clock source between external or internal sources via software.
- Fail-Safe Clock Monitor (FSCM) designed to detect a failure of the external clock source (LP, XT, HS, ECH, ECM, ECL) and switch automatically to the internal oscillator.
- Oscillator Start-up Timer (OST) ensures stability of crystal oscillator sources.

The RSTOSC bits of Configuration Word 1 ([Register 3-1](#)) determine the type of oscillator that will be used when the device runs after Reset, including when it is first powered up.

If an external clock source is selected, the FEXTOSC bits of Configuration Word 1 must be used in conjunction with the RSTOSC bits to select the External Clock mode.

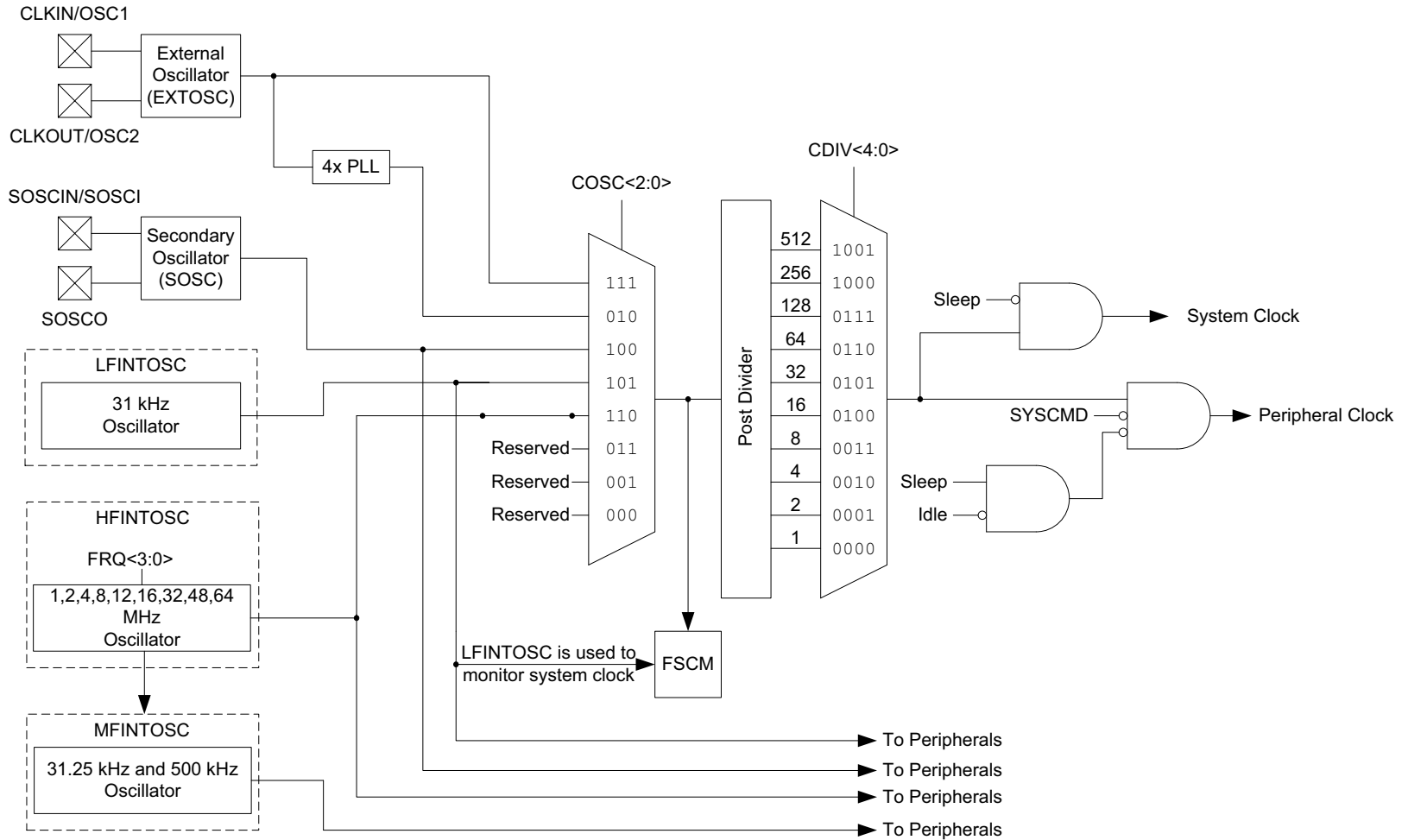
The external oscillator module can be configured in one of the following clock modes, by setting the FEXTOSC<2:0> bits of Configuration Word 1:

1. ECL – External Clock Low-Power mode (below 100 kHz)
2. ECM – External Clock Medium Power mode (100 kHz to 8 MHz)
3. ECH – External Clock High-Power mode (above 8 MHz)
4. LP – 32 kHz Low-Power Crystal mode.
5. XT – Medium Gain Crystal or Ceramic Resonator Oscillator mode (between 100 kHz and 8 MHz)
6. HS – High Gain Crystal or Ceramic Resonator mode (above 4 MHz)

The ECH, ECM, and ECL Clock modes rely on an external logic level signal as the device clock source. The LP, XT, and HS Clock modes require an external crystal or resonator to be connected to the device. Each mode is optimized for a different frequency range. The internal oscillator block produces low and high-frequency clock sources, designated LFINTOSC and HFINTOSC. (see Internal Oscillator Block, [Figure 4-1](#)). Multiple device clock frequencies may be derived from these clock sources.

FIGURE 4-1: SIMPLIFIED PIC® MCU CLOCK SOURCE BLOCK DIAGRAM

Rev. 10-000208D
5/10/2016



4.2 Register Definitions: Oscillator Control

REGISTER 4-1: OSCCON1: OSCILLATOR CONTROL REGISTER1

| | | | | | | | |
|-------|-----------|---------|---------|-----------|---------|---------|---------|
| U-0 | R/W-f/f | R/W-f/f | R/W-f/f | R/W-q/q | R/W-q/q | R/W-q/q | R/W-q/q |
| — | NOSC<2:0> | | | NDIV<3:0> | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | f = determined by fuse setting |
| | | q = Reset value is determined by hardware |

bit 7 **Unimplemented:** Read as '0'

bit 6-4 **NOSC<2:0>:** New Oscillator Source Request bits^(1,2,3)
 The setting requests a source oscillator and PLL combination per [Table 4-2](#).
 POR value = RSTOSC ([Register 3-1](#)).

bit 3-0 **NDIV<3:0>:** New Divider Selection Request bits^(2,3)
 The setting determines the new postscaler division ratio per [Table 4-2](#).

Note 1: The default value (f/f) is determined by the RSTOSC Configuration bits. See [Table 4-1](#) below.

2: If NOSC is written with a reserved value ([Table 4-2](#)), the operation is ignored and neither NOSC nor NDIV is written.

3: When CSWEN = 0, this register is read-only and cannot be changed from the POR value.

TABLE 4-1: DEFAULT OSCILLATOR SETTINGS

| RSTOSC | SFR Reset Values | | | Initial Fosc Frequency |
|--------|------------------|------|--------|------------------------|
| | NOSC/COSC | CDIV | OSCFRQ | |
| 111 | 111 | 1:1 | 4 MHz | EXTOSC per FEXTOSC |
| 110 | 110 | 4:1 | | Fosc = 1 MHz (4 MHz/4) |
| 101 | 101 | 1:1 | | LFINTOSC |
| 100 | 100 | 1:1 | | SOSC |
| 011 | Reserved | | | |
| 010 | 010 | 1:1 | 4 MHz | EXTOSC + 4xPLL (1) |
| 001 | Reserved | | | |
| 000 | 110 | 1:1 | 64 MHz | Fosc = 64 MHz |

Note 1: EXTOSC must meet the PLL specifications ([Table 37-9](#)).

REGISTER 4-2: OSCCON2: OSCILLATOR CONTROL REGISTER 2

| | | | | | | | |
|-------|-----------|-------|-------|-----------|-------|-------|-------|
| U-0 | R-q/q | R-q/q | R-q/q | R-q/q | R-q/q | R-q/q | R-q/q |
| — | COSC<2:0> | | | CDIV<3:0> | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | q = Reset value is determined by hardware |

- bit 7 **Unimplemented:** Read as '0'
- bit 6-4 **COSC<2:0>:** Current Oscillator Source Select bits (read-only)^(1,2)
Indicates the current source oscillator and PLL combination per [Table 4-2](#).
- bit 3-0 **CDIV<3:0>:** Current Divider Select bits (read-only)^(1,2)
Indicates the current postscaler division ratio per [Table 4-2](#).

Note 1: The POR value is the value present when user code execution begins.

2: The Reset value (q/q) is the same as the NOSC/NDIV bits.

TABLE 4-2: NOSC/COSC AND NDIV/CDIV BIT SETTINGS

| NOSC<2:0> COSC<2:0> | Clock Source | NDIV<3:0> CDIV<3:0> | Clock Divider |
|------------------------|--------------------------------|------------------------|---------------|
| 111 | EXTOSC ⁽¹⁾ | 1111-1010 | Reserved |
| 110 | HFINTOSC ⁽²⁾ | 1001 | 512 |
| 101 | LFINTOSC | 1000 | 256 |
| 100 | SOSC | 0111 | 128 |
| 011 | Reserved | 0110 | 64 |
| 010 | EXTOSC + 4x PLL ⁽³⁾ | 0101 | 32 |
| 001 | Reserved | 0100 | 16 |
| 000 | Reserved | 0011 | 8 |
| | | 0010 | 4 |
| | | 0001 | 2 |
| | | 0000 | 1 |

Note 1: EXTOSC configured by the FEXTOSC bits of Configuration Word 1 ([Register 3-1](#)).

2: HFINTOSC frequency is set with the HFFRQ bits of the OSCFRQ register ([Register 4-5](#)).

3: EXTOSC must meet the PLL specifications ([Table 37-9](#)).

REGISTER 4-3: OSCCON3: OSCILLATOR CONTROL REGISTER 3

| | | | | | | | |
|------------|---------|-----|-------|-------|-----|-----|-------|
| R/W/HC-0/0 | R/W-0/0 | U-0 | R-0/0 | R-0/0 | U-0 | U-0 | U-0 |
| CSWHOLD | SOSCPWR | — | ORDY | NOSCR | — | — | — |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | HC = Bit is cleared by hardware |

- bit 7 **CSWHOLD:** Clock Switch Hold bit
 1 = Clock switch will hold (with interrupt) when the oscillator selected by NOSC is ready
 0 = Clock switch may proceed when the oscillator selected by NOSC is ready; NOSCR becomes '1', the switch will occur
- bit 6 **SOSCPWR:** Secondary Oscillator Power Mode Select bit
 1 = Secondary oscillator operating in High-Power mode
 0 = Secondary oscillator operating in Low-Power mode
- bit 5 **Unimplemented:** Read as '0'
- bit 4 **ORDY:** Oscillator Ready bit (read-only)
 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 bit (read-only)⁽¹⁾
 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 yet ready
- bit 2-0 **Unimplemented:** Read as '0'

Note 1: If CSWHOLD = 0, the user may not see this bit set because, when the oscillator becomes ready there may be a delay of one instruction clock before this bit is set. The clock switch occurs in the next instruction cycle and this bit is cleared.

REGISTER 4-4: OSCSTAT: OSCILLATOR STATUS REGISTER 1

| | | | | | | | |
|-------|-------|-------|-------|-------|-------|-------|-------|
| R-q/q | R-q/q | R-q/q | R-q/q | R-q/q | R-q/q | U-0 | R-q/q |
| EXTOR | HFOR | MFOR | LFOR | SOR | ADOR | — | PLL R |
| bit 7 | | | | | | bit 0 | |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | q = Reset value is determined by hardware |

- bit 7 **EXTOR:** EXTOSC (external) Oscillator Ready bit
 1 = The oscillator is ready to be used
 0 = The oscillator is not enabled, or is not yet ready to be used
- bit 6 **HFOR:** HFINTOSC Oscillator Ready bit
 1 = The oscillator is ready to be used
 0 = The oscillator is not enabled, or is not yet ready to be used
- bit 5 **MFOR:** MFINTOSC Oscillator Ready
 1 = The oscillator is ready to be used
 0 = The oscillator is not enabled, or is not yet ready to be used
- bit 4 **LFOR:** LFINTOSC Oscillator Ready bit
 1 = The oscillator is ready to be used
 0 = The oscillator is not enabled, or is not yet ready to be used
- bit 3 **SOR:** Secondary (Timer1) Oscillator Ready bit
 1 = The oscillator is ready to be used
 0 = The oscillator is not enabled, or is not yet ready to be used
- bit 2 **ADOR:** ADC Oscillator Ready bit
 1 = The oscillator is ready to be used
 0 = The oscillator is not enabled, or is not yet ready to be used
- bit 1 **Unimplemented:** Read as '0'
- bit 0 **PLL R:** PLL is Ready bit
 1 = The PLL is ready to be used
 0 = The PLL is not enabled, the required input source is not ready, or the PLL is not locked.

REGISTER 4-5: OSCFRQ: HFINTOSC FREQUENCY SELECTION REGISTER

| | | | | | | | |
|-------|-----|-----|-----|------------|---------|---------|---------|
| U-0 | U-0 | U-0 | U-0 | R/W-q/q | R/W-q/q | R/W-q/q | R/W-q/q |
| — | — | — | — | HFFRQ<3:0> | | | |
| bit 7 | | | | bit 0 | | | |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | q = Reset value is determined by hardware |

bit 7-4

Unimplemented: Read as '0'

bit 3-0

HFFRQ<3:0>: HFINTOSC Frequency Selection bits

| HFFRQ<3:0> | Nominal Freq (MHz) |
|-----------------------|--------------------|
| 1001 | Reserved |
| 1010 | |
| 1111 | |
| 1110 | |
| 1101 | |
| 1100 | |
| 1011 | |
| 1000 ⁽³⁾ | |
| 0111 | 48 |
| 0110 | 32 |
| 0101 ⁽⁴⁾ | 16 |
| 0100 | 12 |
| 0011 | 8 |
| 0010 ^(1,2) | 4 |
| 0001 | 2 |
| 0000 | 1 |

Note 1: Refer to [Table 4-1](#) for more information.

REGISTER 4-6: OSCTUNE: HFINTOSC TUNING REGISTER

| | | | | | | | | |
|-------|-----|------------|---------|---------|---------|---------|---------|-------|
| U-0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | |
| — | — | HFTUN<5:0> | | | | | | |
| bit 7 | | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-6 **Unimplemented:** Read as '0'

bit 5-0 **HFTUN<5:0>:** HFINTOSC Frequency Tuning bits

01 1111 = Maximum frequency

•

•

•

00 0000 = Center frequency. Oscillator module is running at the calibrated frequency (default value).

•

•

•

10 0000 = Minimum frequency

REGISTER 4-7: OSCEN: OSCILLATOR MANUAL ENABLE REGISTER

| | | | | | | | |
|---------|---------|---------|---------|---------|---------|-----|-----|
| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | U-0 | U-0 |
| EXTOEN | HFOEN | MFOEN | LFOEN | SOSCEN | ADOEN | — | — |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

- bit 7 **EXTOEN:** External Oscillator Manual Request Enable bit
 1 = EXTOSC is explicitly enabled, operating as specified by FEXTOSC
 0 = EXTOSC could be enabled by requesting peripheral
- bit 6 **HFOEN:** HFINTOSC Oscillator Manual Request Enable bit
 1 = HFINTOSC is explicitly enabled, operating as specified by OSCFRQ ([Register 4-5](#))
 0 = HFINTOSC could be enabled by requesting peripheral
- bit 5 **MFOEN:** MFINTOSC (500 kHz/31.25 kHz) Oscillator Manual Request Enable bit (Derived from HFINTOSC)
 1 = MFINTOSC is explicitly enabled
 0 = MFINTOSC could be enabled by requesting peripheral
- bit 4 **LFOEN:** LFINTOSC (31 kHz) Oscillator Manual Request Enable bit
 1 = LFINTOSC is explicitly enabled
 0 = LFINTOSC could be enabled by requesting peripheral
- bit 3 **SOSCEN:** Secondary Oscillator Manual Request Enable bit
 1 = Secondary Oscillator is explicitly enabled, operating as specified by SOSCPWR
 0 = Secondary Oscillator could be enabled by requesting peripheral
- bit 2 **ADOEN:** ADC Oscillator Manual Request Enable bit
 1 = ADC oscillator is explicitly enabled
 0 = ADC oscillator could be enabled by requesting peripheral
- bit 1-0 **Unimplemented:** Read as '0'

4.3 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 are: oscillator modules (ECH, ECM, ECL mode), quartz crystal resonators or ceramic resonators (LP, XT and HS modes).

Internal clock sources are contained within the oscillator module. The internal oscillator block has two internal oscillators that are used to generate internal system clock sources. The High-Frequency Internal Oscillator (HFINTOSC) can produce 1, 2, 4, 8, 12, 16, 32, 48 and 64 MHz clock. The frequency can be controlled through the OSCFRQ register (Register 4-5). The Low-Frequency Internal Oscillator (LFINTOSC) generates a fixed 31 kHz frequency.

A 4x PLL is provided that can be used in conjunction with the external clock. See Section 4.3.1.4 “4x PLL” for more details.

The system clock can be selected between external or internal clock sources via the NOSC bits in the OSCCON1 register. See Section 4.4 “Clock Switching” for additional information. The system clock can be made available on the OSC2/CLKOUT pin for any of the modes that do not use the OSC2 pin. The clock out functionality is governed by the CLKOUTEN bit in the CONFIG1H register (Register 3-2). If enabled, the clock out signal is always at a frequency of $F_{osc}/4$.

4.3.1 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<2:0> and FEXTOSC<2:0> bits in the Configuration Words to select an external clock source that will be used as the default system clock upon a device Reset.
- Write the NOSC<2:0> and NDIV<3:0> bits in the OSCCON1 register to switch the system clock source.

See Section 4.4 “Clock Switching” for more information.

4.3.1.1 EC Mode

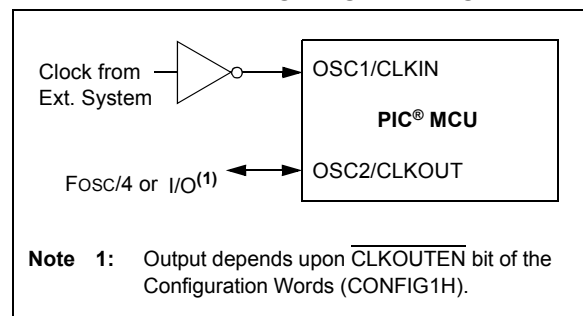
The External Clock (EC) mode allows an externally generated logic level signal to be the system clock source. When operating in this mode, an external clock source is connected to the OSC1 input. OSC2/CLKOUT is available for general purpose I/O or CLKOUT. Figure 4-2 shows the pin connections for EC mode.

EC mode has three power modes to select from through Configuration Words:

- ECH – High power, above 8 MHz
- ECM – Medium power, 100 kHz-8 MHz
- ECL – Low power, below 100 MHz

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.

FIGURE 4-2: EXTERNAL CLOCK (EC) MODE OPERATION



4.3.1.2 LP, XT, HS Modes

The LP, XT and HS modes support the use of quartz crystal resonators or ceramic resonators connected to OSC1 and OSC2 (Figure 4-3). The three modes select a low, medium or high gain setting of the internal inverter-amplifier to support various resonator types and speed.

LP Oscillator mode selects the lowest gain setting of the internal inverter-amplifier. LP mode current consumption is the least of the three modes. This mode is designed to drive only 32.768 kHz tuning-fork type crystals (watch crystals).

XT Oscillator mode selects the intermediate gain setting of the internal inverter-amplifier. XT mode current consumption is the medium of the three modes. This mode is best suited to drive resonators with a medium drive level specification (above 100 kHz - 8 MHz).

HS Oscillator mode selects the highest gain setting of the internal inverter-amplifier. HS mode current consumption is the highest of the three modes. This mode is best suited for resonators that require a high drive setting (above 8 MHz).

Figure 4-3 and Figure 4-4 show typical circuits for quartz crystal and ceramic resonators, respectively.

FIGURE 4-3: QUARTZ CRYSTAL OPERATION (LP, XT OR HS MODE)

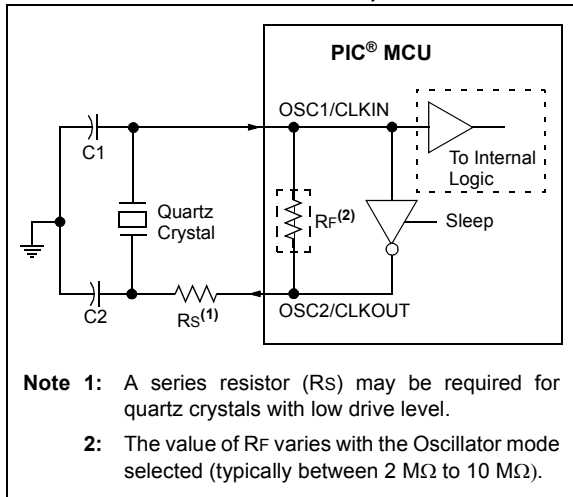
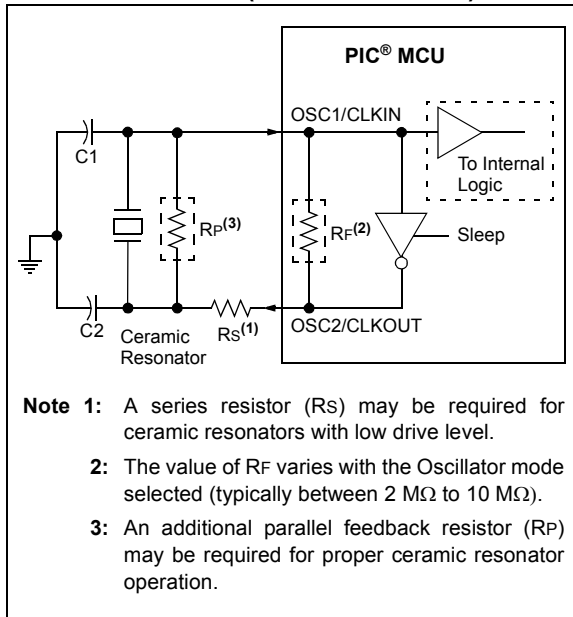


FIGURE 4-4: CERAMIC RESONATOR OPERATION (XT OR HS MODE)



4.3.1.3 Oscillator Start-up Timer (OST)

If the oscillator module is configured for LP, XT or HS modes, the Oscillator Start-up Timer (OST) counts 1024 oscillations from OSC1. This occurs following a Power-on Reset (POR), or a wake-up from Sleep. The OST ensures that the oscillator circuit, using a quartz crystal resonator or ceramic resonator, has started and is providing a stable system clock to the oscillator module.

4.3.1.4 4x PLL

The oscillator module contains a 4x PLL that can be used with the external clock sources to provide a system clock source. The input frequency for the PLL must fall within specifications. See the PLL Clock Timing Specifications in [Table 37-9](#).

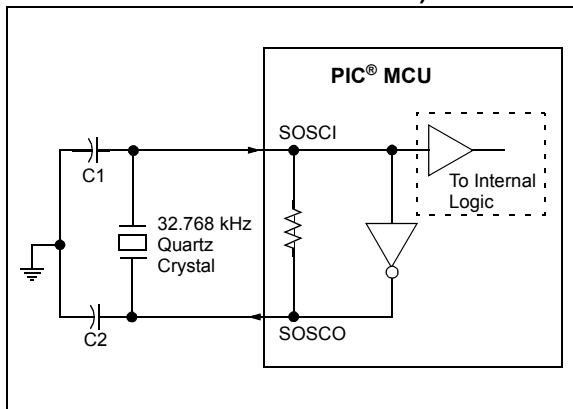
The PLL can be enabled for use by one of two methods:

1. Program the RSTOSC bits in the Configuration Word 1 to 010 (enable EXTOSC with 4x PLL).
2. Write the NOSC bits in the OSCCON1 register to 010 (enable EXTOSC with 4x PLL).

4.3.1.5 Secondary Oscillator

The secondary oscillator is a separate oscillator block that can be used as an alternate system clock source. The secondary oscillator is optimized for 32.768 kHz, and can be used with an external crystal oscillator connected to the SOSC1 and SOSCO device pins, or an external clock source connected to the SOSCIN pin. The secondary oscillator can be selected during run-time using clock switching. Refer to [Section 4.4 “Clock Switching”](#) for more information.

FIGURE 4-5: QUARTZ CRYSTAL OPERATION (SECONDARY OSCILLATOR)



Note 1: Quartz crystal characteristics vary according to type, package and manufacturer. The user should consult the manufacturer data sheets for specifications and recommended application.

- 2: Always verify oscillator performance over the VDD and temperature range that is expected for the application.
- 3: For oscillator design assistance, reference the following Microchip Application Notes:

- AN826, “Crystal Oscillator Basics and Crystal Selection for PIC® and PIC® Devices” (DS00826)
- AN849, “Basic PIC® Oscillator Design” (DS00849)
- AN943, “Practical PIC® Oscillator Analysis and Design” (DS00943)
- AN949, “Making Your Oscillator Work” (DS00949)
- TB097, “Interfacing a Micro Crystal MS1V-T1K 32.768 kHz Tuning Fork Crystal to a PIC16F690/SS” (DS91097)
- AN1288, “Design Practices for Low-Power External Oscillators” (DS01288)

4.3.2 INTERNAL CLOCK SOURCES

The device may be configured to use the internal oscillator block as the system clock by performing one of the following actions:

- Program the RSTOSC<2:0> bits in Configuration Words to select the INTOSC clock source, which will be used as the default system clock upon a device Reset.
- Write the NOSC<2:0> bits in the OSCCON1 register to switch the system clock source to the internal oscillator during run-time. See [Section 4.4 “Clock Switching”](#) for more information.

In INTOSC mode, OSC1/CLKIN is available for general purpose I/O. OSC2/CLKOUT is available for general purpose I/O or CLKOUT.

The function of the OSC2/CLKOUT pin is determined by the CLKOUTEN bit in Configuration Words.

The internal oscillator block has two independent oscillators that can produce two internal system clock sources.

1. The **HFINTOSC** (High-Frequency Internal Oscillator) is factory-calibrated and operates from 1 to 64 MHz. The frequency of HFINTOSC can be selected through the OSCFRQ Frequency Selection register, and fine-tuning can be done via the OSCTUNE register.
2. The **LFINTOSC** (Low-Frequency Internal Oscillator) is factory-calibrated and operates at 31 kHz.

4.3.2.1 HFINTOSC

The High-Frequency Internal Oscillator (HFINTOSC) is a precision digitally-controlled internal clock source that produces a stable clock up to 64 MHz. The HFINTOSC can be enabled through one of the following methods:

- Programming the RSTOSC<2:0> bits in Configuration Word 1 to '110' (Fosc = 1 MHz) or '000' (Fosc = 64 MHz) to set the oscillator upon device Power-up or Reset.
- Write to the NOSC<2:0> bits of the OSCCON1 register during run-time. See [Section 4.4 “Clock Switching”](#) for more information.

The HFINTOSC frequency can be selected by setting the HFFRQ<3:0> bits of the OSCFRQ register.

The NDIV<3:0> bits of the OSCCON1 register allow for division of the HFINTOSC output from a range between 1:1 and 1:512.

4.3.2.2 MFINTOSC

The module provides two (500 kHz and 31.25 kHz) constant clock outputs. These clocks are digital divisors of the HFINTOSC clock. Dynamic divider logic is used to provide constant MFINTOSC clock rates for all settings of HFINTOSC.

The MFINTOSC cannot be used to drive the system but it is used to clock certain modules such as the Timers and WWDT.

4.3.2.3 Internal Oscillator Frequency Adjustment

The internal oscillator is factory-calibrated. This internal oscillator can be adjusted in software by writing to the OSCTUNE register ([Register 4-3](#)).

The default value of the OSCTUNE register is 00h. The value is a 6-bit two's complement number. A value of 1Fh will provide an adjustment to the maximum frequency. A value of 20h will provide an adjustment to the minimum frequency.

When the OSCTUNE register is modified, the oscillator frequency will begin shifting to the new frequency. Code execution continues during this shift. There is no indication that the shift has occurred.

OSCTUNE **does not affect** the LFINTOSC frequency. Operation of features that depend on the LFINTOSC clock source frequency, such as the Power-up Timer (PWRT), WWDT, Fail-Safe Clock Monitor (FSCM) and peripherals, are *not* affected by the change in frequency.

4.3.2.4 LFINTOSC

The Low-Frequency Internal Oscillator (LFINTOSC) is a factory-calibrated 31 kHz internal clock source.

The LFINTOSC is the frequency for the Power-up Timer (PWRT), Windowed Watchdog Timer (WWDT) and Fail-Safe Clock Monitor (FSCM).

The LFINTOSC is enabled through one of the following methods:

- Programming the RSTOSC<2:0> bits of Configuration Word 1 to enable LFINTOSC.
- Write to the NOSC<2:0> bits of the OSCCON1 register during run-time. See [Section 4.4, Clock Switching](#) for more information.

4.3.2.5 ADCRC

The ADCRC is an oscillator dedicated to the ADC² module. The ADCRC oscillator can be manually enabled using the ADOEN bit of the OSCEN register. The ADCRC runs at a fixed frequency of 600 kHz. ADCRC is automatically enabled if it is selected as the clock source for the ADC² module.

4.3.2.6 Oscillator Status and Manual Enable

The Ready status of each oscillator (including the ADCRC oscillator) is displayed in OSCSTAT (Register 4-4). The oscillators (but not the PLL) may be explicitly enabled through OSCEN (Register 4-7).

4.3.2.7 HFOR and MFOR Bits

The HFOR and MFOR bits indicate that the HFINTOSC and MFINTOSC is ready. These clocks are always valid for use at all times, but only accurate after they are ready.

When a new value is loaded into the OSCFRQ register, the HFOR and MFOR bits will clear, and set again when the oscillator is ready. During pending OSCFRQ changes the MFINTOSC clock will stall at a high or a low state, until the HFINTOSC resumes operation.

4.4 Clock Switching

The system clock source can be switched between external and internal clock sources via software using the New Oscillator Source (NOSC) bits of the OSCCON1 register. The following clock sources can be selected using the following:

- External oscillator
- Internal Oscillator Block (INTOSC)

Note: The Clock Switch Enable bit in Configuration Word 1 can be used to enable or disable the clock switching capability. When cleared, the NOSC and NDIV bits cannot be changed by user software. When set, writing to NOSC and NDIV is allowed and would switch the clock frequency.

4.4.1 NEW OSCILLATOR SOURCE (NOSC) AND NEW DIVIDER SELECTION REQUEST (NDIV) BITS

The New Oscillator Source (NOSC) and New Divider Selection Request (NDIV) bits of the OSCCON1 register select the system clock source and frequency that are used for the CPU and peripherals.

When new values of NOSC and NDIV are written to OSCCON1, the current oscillator selection will continue to operate while waiting for the new clock source to indicate that it is stable and ready. In some cases, the newly requested source may already be in use, and is ready immediately. In the case of a divider-only change, the new and old sources are the same, so the old source will be ready immediately. The device may enter Sleep while waiting for the switch as described in Section 4.4.3 “Clock Switch and Sleep”.

When the new oscillator is ready, the New Oscillator Ready (NOSCR) bit of OSCCON3 is set and also the Clock Switch Interrupt Flag (CSWIF) bit of PIR1 sets. If Clock Switch Interrupts are enabled (CSWIE = 1), an interrupt will be generated at that time. The Oscillator Ready (ORDY) bit of OSCCON3 can also be polled to determine when the oscillator is ready in lieu of an interrupt.

Note: The CSWIF interrupt will not wake the system from Sleep.

If the Clock Switch Hold (CSWHOLD) bit of OSCCON3 is clear, the oscillator switch will occur when the New Oscillator is Ready bit (NOSCR) is set, and the interrupt (if enabled) will be serviced at the new oscillator setting.

If CSWHOLD is set, the oscillator switch is suspended, while execution continues using the current (old) clock source. When the NOSCR bit is set, software should:

- Set CSWHOLD = 0 so the switch can complete, or
- Copy COSC into NOSC to abandon the switch.

If DOZE is in effect, the switch occurs on the next clock cycle, whether or not the CPU is operating during that cycle.

Changing the clock post-divider without changing the clock source (i.e., changing Fosc from 1 MHz to 2 MHz) is handled in the same manner as a clock source change, as described previously. The clock source will already be active, so the switch is relatively quick. CSWHOLD must be clear (CSWHOLD = 0) for the switch to complete.

The current COSC and CDIV are indicated in the OSCCON2 register up to the moment when the switch actually occurs, at which time OSCCON2 is updated and ORDY is set. NOSCR is cleared by hardware to indicate that the switch is complete.

4.4.2 PLL INPUT SWITCH

Switching between the PLL and any non-PLL source is managed as described above. The input to the PLL is established when NOSC selects the PLL, and maintained by the COSC setting.

When NOSC and COSC select the PLL with different input sources, the system continues to run using the COSC setting, and the new source is enabled per NOSC. When the new oscillator is ready (and CSWHOLD = 0), system operation is suspended while the PLL input is switched and the PLL acquires lock. This provides a truly glitch-free clock switch operation.

Note: If the PLL fails to lock, the FSCM will trigger.

4.4.3 CLOCK SWITCH AND SLEEP

If OSCCON1 is written with a new value and the device is put to Sleep before the switch completes, the switch will not take place and the device will enter Sleep mode.

When the device wakes from Sleep and the CSWHOLD bit is clear, the device will wake with the 'new' clock active, and the clock switch interrupt flag bit (CSWIF) will be set.

When the device wakes from Sleep and the CSWHOLD bit is set, the device will wake with the 'old' clock active and the new clock will be requested again.

FIGURE 4-6: CLOCK SWITCH (CSWHOLD = 0)

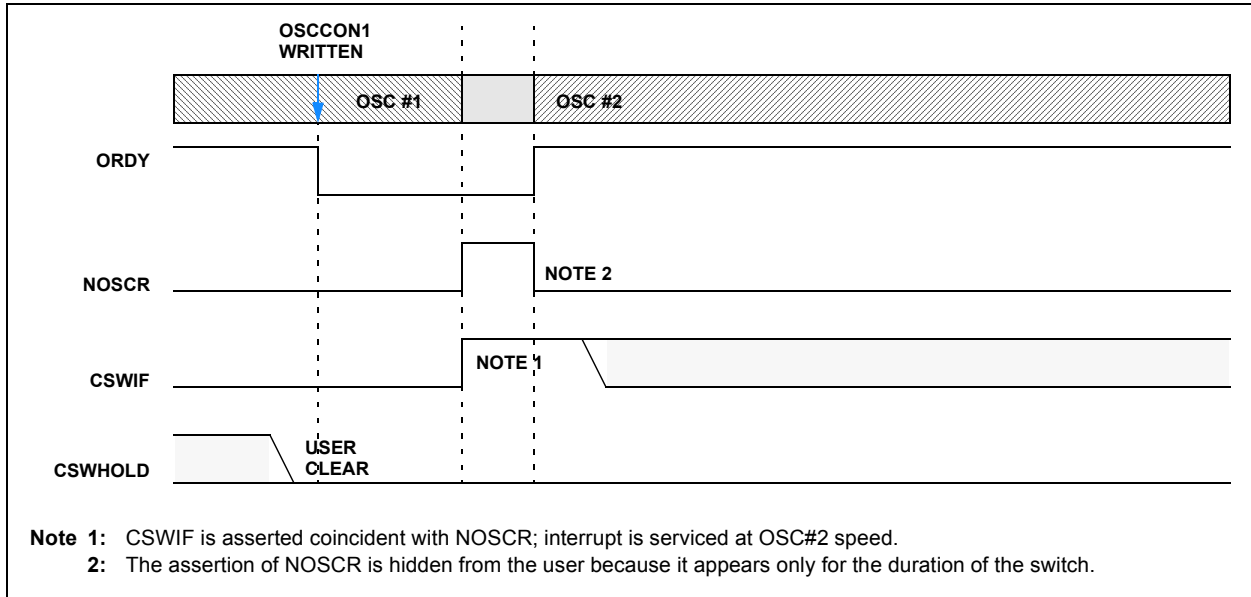


FIGURE 4-7: CLOCK SWITCH (CSWHOLD = 1)

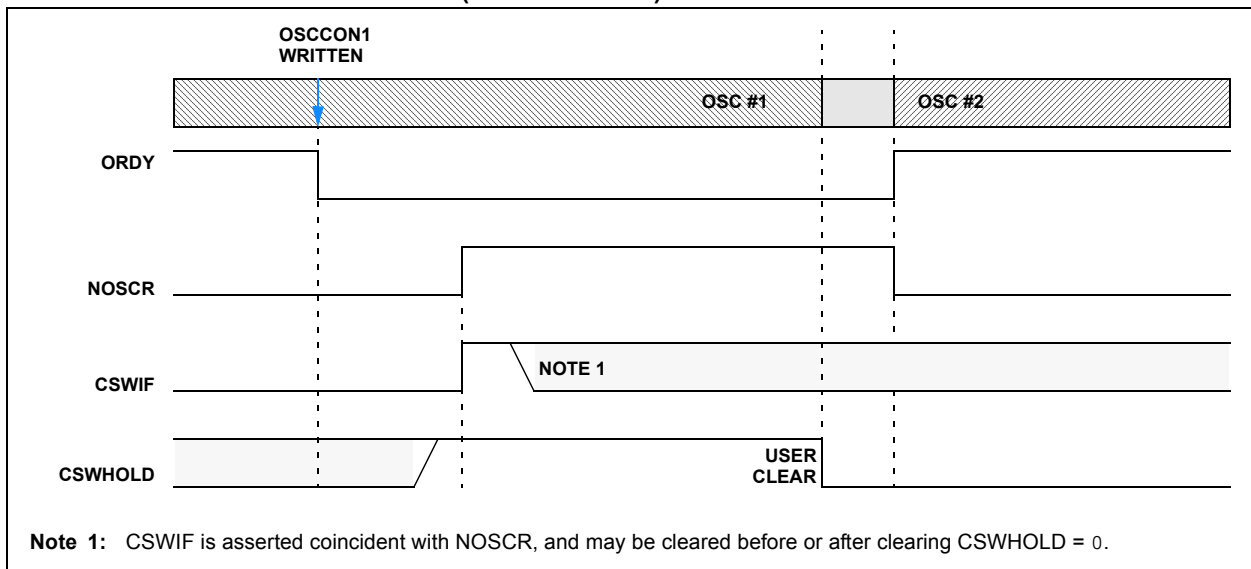
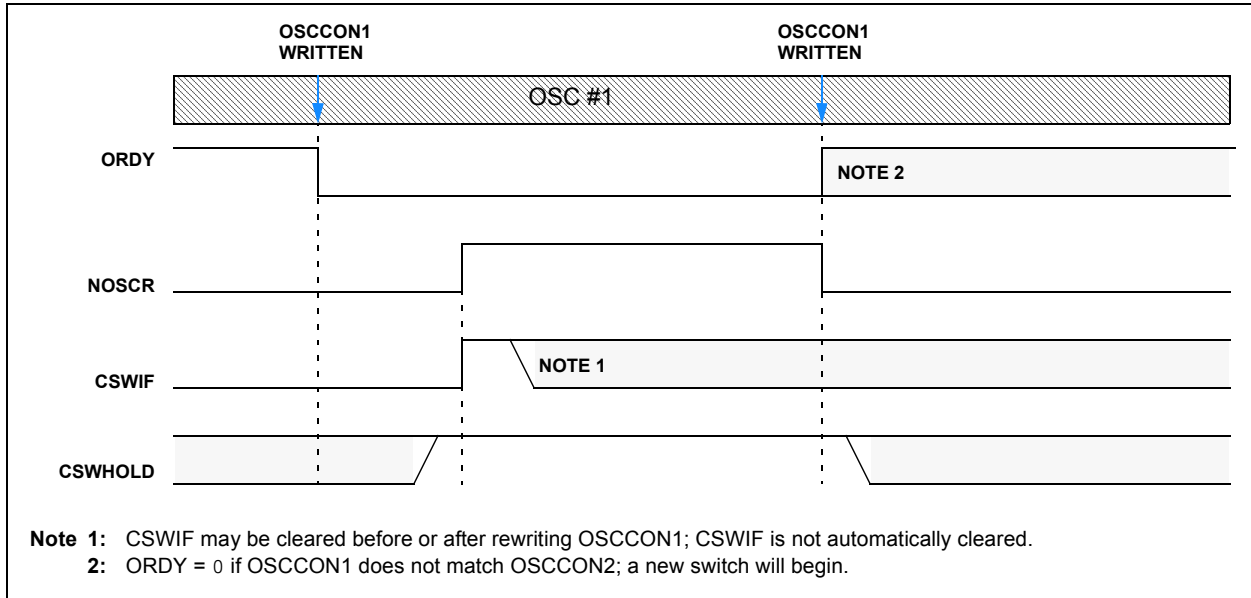


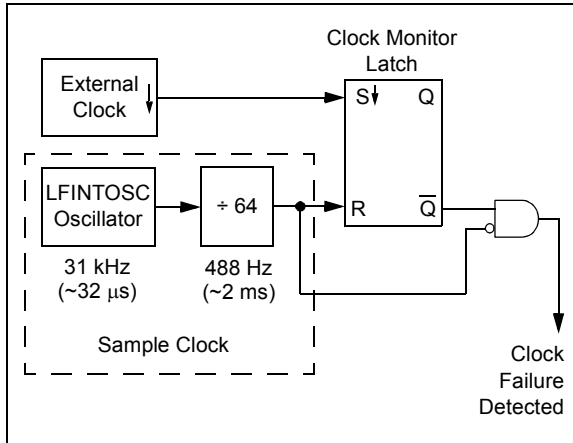
FIGURE 4-8: CLOCK SWITCH ABANDONED



4.5 Fail-Safe Clock Monitor

The Fail-Safe Clock Monitor (FSCM) allows the device to continue operating should the external oscillator fail. The FSCM is enabled by setting the FCMEN bit in the Configuration Words. The FSCM is applicable to all external Oscillator modes (LP, XT, HS, ECL/M/H and Secondary Oscillator).

FIGURE 4-9: FSCM BLOCK DIAGRAM



4.5.1 FAIL-SAFE DETECTION

The FSCM module 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. See Figure 4-9. Inside the fail detector block is a latch. The external clock sets the latch on each falling edge of the external clock. The sample clock clears the latch on each rising edge of the sample clock. A failure is detected when an entire half-cycle of the sample clock elapses before the external clock goes low.

4.5.2 FAIL-SAFE OPERATION

When the external clock fails, the FSCM overwrites the COSC bits to select HFINTOSC (3'b110). The frequency of HFINTOSC would be determined by the previous state of the HFFRQ bits and the NDIV/CDIV bits. The bit flag OSCFIF of the PIR1 register is set. Setting this flag will generate an interrupt if the OSCFIE bit of the PIE1 register is also set. The device firmware can then take steps to mitigate the problems that may arise from a failed clock. The system clock will continue to be sourced from the internal clock source until the device firmware successfully restarts the external oscillator and switches back to external operation, by writing to the NOSC and NDIV bits of the OSCCON1 register.

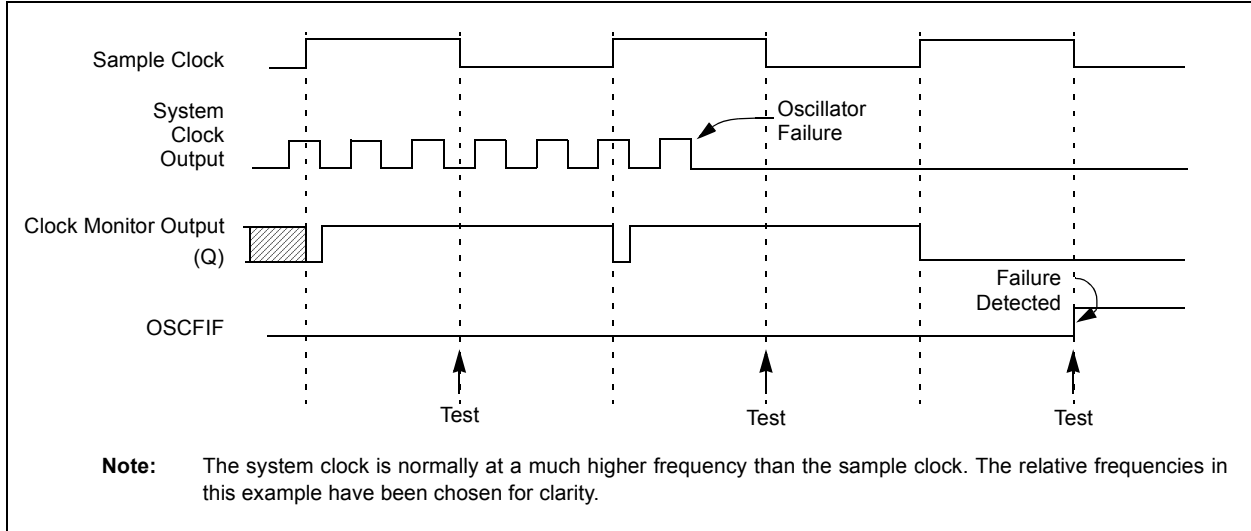
4.5.3 FAIL-SAFE CONDITION CLEARING

The Fail-Safe condition is cleared after a Reset, executing a SLEEP instruction or changing the NOSC and NDIV bits of the OSCCON1 register. When switching to the external oscillator or PLL, the OST is restarted. While the OST is running, the device continues to operate from the INTOSC selected in OSCCON1. When the OST times out, the Fail-Safe condition is cleared after successfully switching to the external clock source. The OSCFIF bit should be cleared prior to switching to the external clock source. If the Fail-Safe condition still exists, the OSCFIF flag will again become set by hardware.

4.5.4 RESET OR WAKE-UP FROM SLEEP

The FSCM is designed to detect an oscillator failure after the Oscillator Start-up Timer (OST) has expired. The OST is used after waking up from Sleep and after any type of Reset. The OST is not used with the EC Clock modes so that the FSCM will be active as soon as the Reset or wake-up has completed.

FIGURE 4-10: FSCM TIMING DIAGRAM



PIC18(L)F67K40

TABLE 4-3: SUMMARY OF REGISTERS ASSOCIATED WITH CLOCK SOURCES

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on Page |
|---------|---------|-----------|------------|-------|------------|-------|-------|-------|------------------|
| OSCCON1 | — | NOSC<2:0> | | | NDIV<3:0> | | | | 34 |
| OSCCON2 | — | COSC<2:0> | | | CDIV<3:0> | | | | 35 |
| OSCCON3 | CSWHOLD | SOSCPWR | — | ORDY | NOSCR | — | — | — | 36 |
| OSCSTAT | EXTOR | HFOR | MFOR | LFOR | SOR | ADOR | — | PLLR | 37 |
| OSCTUNE | — | — | HFTUN<5:0> | | | | | | 39 |
| OSCFRQ | — | — | — | — | HFFRQ<3:0> | | | | 38 |
| OSCCON | EXTOEN | HFOEN | MFOEN | LFOEN | SOSCEN | ADOEN | — | — | 40 |
| PIE1 | OSCFIE | CSWIE | — | — | — | — | ADTIE | ADIE | 185 |
| PIR1 | OSCFIF | CSWIF | — | — | — | — | ADTIF | ADIF | 174 |
| IPR1 | OSCFIP | CSWIP | — | — | — | — | ADTIP | ADIP | 195 |

Legend: — = unimplemented location, read as '0'. Shaded cells are not used by clock sources.

TABLE 4-4: SUMMARY OF CONFIGURATION WORD WITH CLOCK SOURCES

| Name | Bits | Bit -/7 | Bit -/6 | Bit 13/5 | Bit 12/4 | Bit 11/3 | Bit 10/2 | Bit 9/1 | Bit 8/0 | Register on Page |
|---------|------|---------|-------------|----------|----------|----------|--------------|---------|----------|------------------|
| CONFIG1 | 13:8 | — | — | FCMEN | — | CSWEN | — | — | CLKOUTEN | 21 |
| | 7:0 | — | RSTOSC<2:0> | | | — | FEXTOSC<2:0> | | | |

Legend: — = unimplemented location, read as '0'. Shaded cells are not used by clock sources.

5.0 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 also be used as a signal for other peripherals, such as the Data Signal Modulator (DSM), Memory Scanner and Timer module.

The reference clock output module has the following features:

- Selectable clock source using the CLKRCLK register
- Programmable clock divider
- Selectable duty cycle

FIGURE 5-1: CLOCK REFERENCE BLOCK DIAGRAM

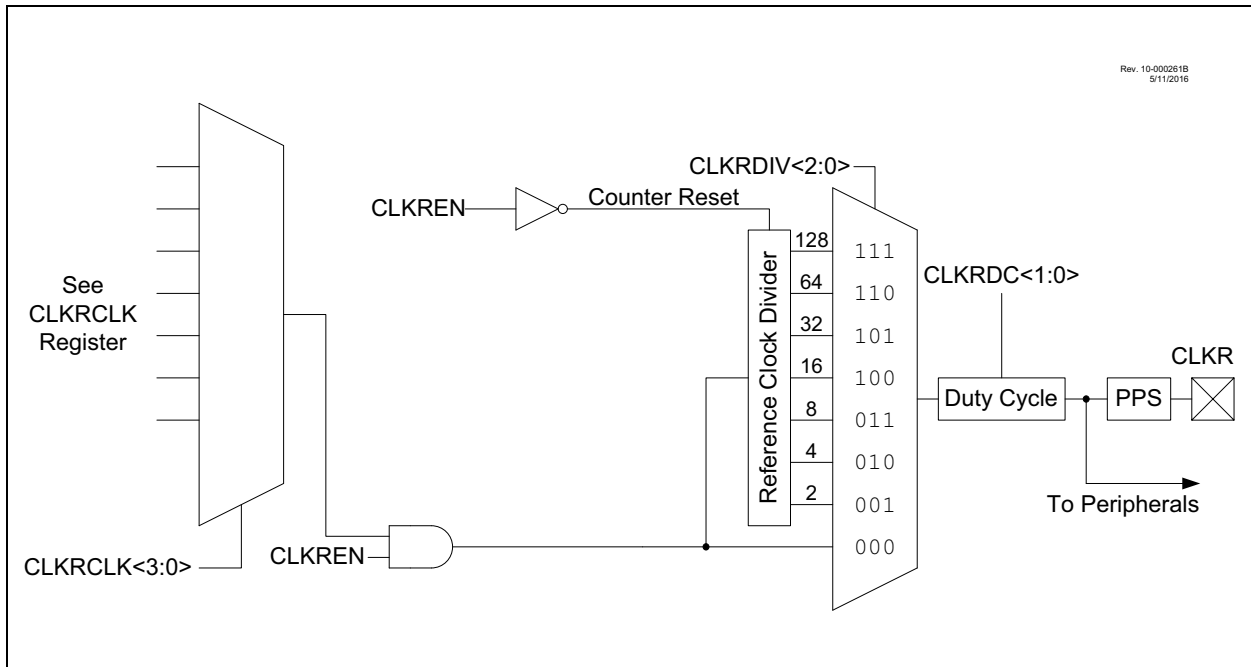
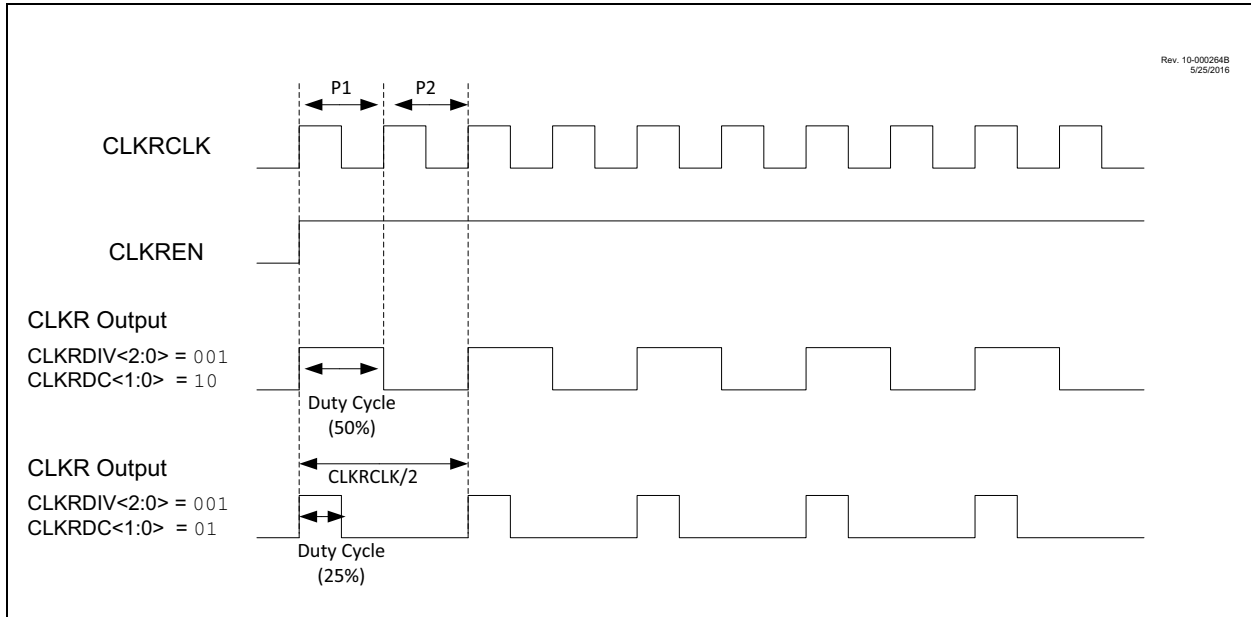


FIGURE 5-2: CLOCK REFERENCE TIMING



5.1 Clock Source

The input to the reference clock output can be selected using the CLKRCLK register.

5.1.1 CLOCK SYNCHRONIZATION

Once the reference clock enable (EN) is set, the module is ensured to be glitch-free at start-up.

When the reference clock output is disabled, the output signal will be disabled immediately.

Clock dividers and clock duty cycles can be changed while the module is enabled, but glitches may occur on the output. To avoid possible glitches, clock dividers and clock duty cycles should be changed only when the CLKREN is clear.

5.2 Programmable Clock Divider

The module takes the clock input and divides it based on the value of the DIV<2:0> bits of the CLKRCON register ([Register 5-1](#)).

The following configurations can be made based on the DIV<2:0> bits:

- Base FOSC value
- FOSC divided by 2
- FOSC divided by 4
- FOSC divided by 8
- FOSC divided by 16
- FOSC divided by 32
- FOSC divided by 64
- FOSC divided by 128

The clock divider values can be changed while the module is enabled; however, in order to prevent glitches on the output, the DIV<2:0> bits should only be changed when the module is disabled (EN = 0).

5.3 Selectable Duty Cycle

The DC<1:0> bits of the CLKRCON register can be used to modify the duty cycle of the output clock. A duty cycle of 25%, 50%, or 75% can be selected for all clock rates, with the exception of the undivided base Fosc value.

The duty cycle can be changed while the module is enabled; however, in order to prevent glitches on the output, the DC<1:0> bits should only be changed when the module is disabled (EN = 0).

Note: The DC1 bit is reset to '1'. This makes the default duty cycle 50% and not 0%.

5.4 Operation in Sleep Mode

The reference clock output module clock is based on the system clock. When the device goes to Sleep, the module outputs will remain in their current state. This will have a direct effect on peripherals using the reference clock output as an input signal. No change should occur in the module from entering or exiting from Sleep.

5.5 Register Definitions: Reference Clock

Long bit name prefixes for the Reference Clock peripherals are shown in [Table 5-1](#). Refer to [Section 1.4.2.2 “Long Bit Names”](#) for more information.

TABLE 5-1:

| Peripheral | Bit Name Prefix |
|------------|-----------------|
| CLKR | CLKR |

REGISTER 5-1: CLKRCON: REFERENCE CLOCK CONTROL REGISTER

| | | | | | | | |
|---------|-----|-----|---------|----------|---------|---------|---------|
| R/W-0/0 | U-0 | U-0 | R/W-1/1 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| EN | — | — | DC<1:0> | DIV<2:0> | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7 **EN:** Reference Clock Module Enable bit
 1 = Reference clock module enabled
 0 = Reference clock module is disabled

bit 6-5 **Unimplemented:** Read as '0'

bit 4-3 **DC<1:0>:** Reference Clock Duty Cycle bits⁽¹⁾
 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%

bit 2-0 **DIV<2:0>:** Reference Clock Divider bits
 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 reference clock divider values of two or larger, the base clock cannot be further divided.

PIC18(L)F67K40

REGISTER 5-2: CLKRCLK: CLOCK REFERENCE CLOCK SELECTION MUX

| | | | | | | | |
|-------|-----|-----|-----|-----|----------|---------|---------|
| U-0 | U-0 | U-0 | U-0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| — | — | — | — | — | CLK<2:0> | | |
| bit 7 | | | | | | | bit 0 |

Legend:

R = Readable bit W = Writable bit U = Unimplemented bit, read as '0'
u = Bit is unchanged x = Bit is unknown -n/n = Value at POR and BOR/Value at all other Resets
'1' = Bit is set '0' = Bit is cleared

bit 7-3 **Unimplemented:** Read as '0'
bit 2-0 **CLK<2:0>:** CLKR Clock Selection bits
111 = Unimplemented
110 = Unimplemented
101 = Unimplemented
100 = SOSC
011 = MFINTOSC (500 kHz)
010 = LFINTOSC (31 kHz)
001 = HFINTOSC
000 = FOSC

TABLE 5-2: SUMMARY OF REGISTERS ASSOCIATED WITH CLOCK REFERENCE OUTPUT

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on Page |
|---------|--------|-------|-------------|---------|--------|----------|--------|-------|------------------|
| CLKRCON | EN | — | — | DC<1:0> | | DIV<2:0> | | | 54 |
| CLKRCLK | — | — | — | — | — | CLK<2:0> | | | 55 |
| PMD0 | SYSCMD | FVRMD | HLVDMD | CRCMD | SCANMD | NVMMD | CLKRMD | IOCMD | 65 |
| RxyPPS | — | — | RxyPPS<5:0> | | | | | | 227 |

Legend: — = unimplemented, read as '0'. Shaded cells are not used by the CLKR module.

6.0 POWER-SAVING OPERATION MODES

The purpose of the Power-Down modes is to reduce power consumption. There are three Power-Down modes:

- Doze mode
- Sleep mode
- Idle mode

6.1 Doze Mode

Doze mode allows for power saving by reducing CPU operation and program memory (PFM) access, without affecting peripheral operation. Doze mode differs from Sleep mode because the bandgap 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 (DOZEN) bit is set (DOZEN = 1), the CPU executes only one instruction cycle out of every N cycles as defined by the DOZE<2:0> bits of the CPUDOZE register. For example, if DOZE<2:0> = 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.

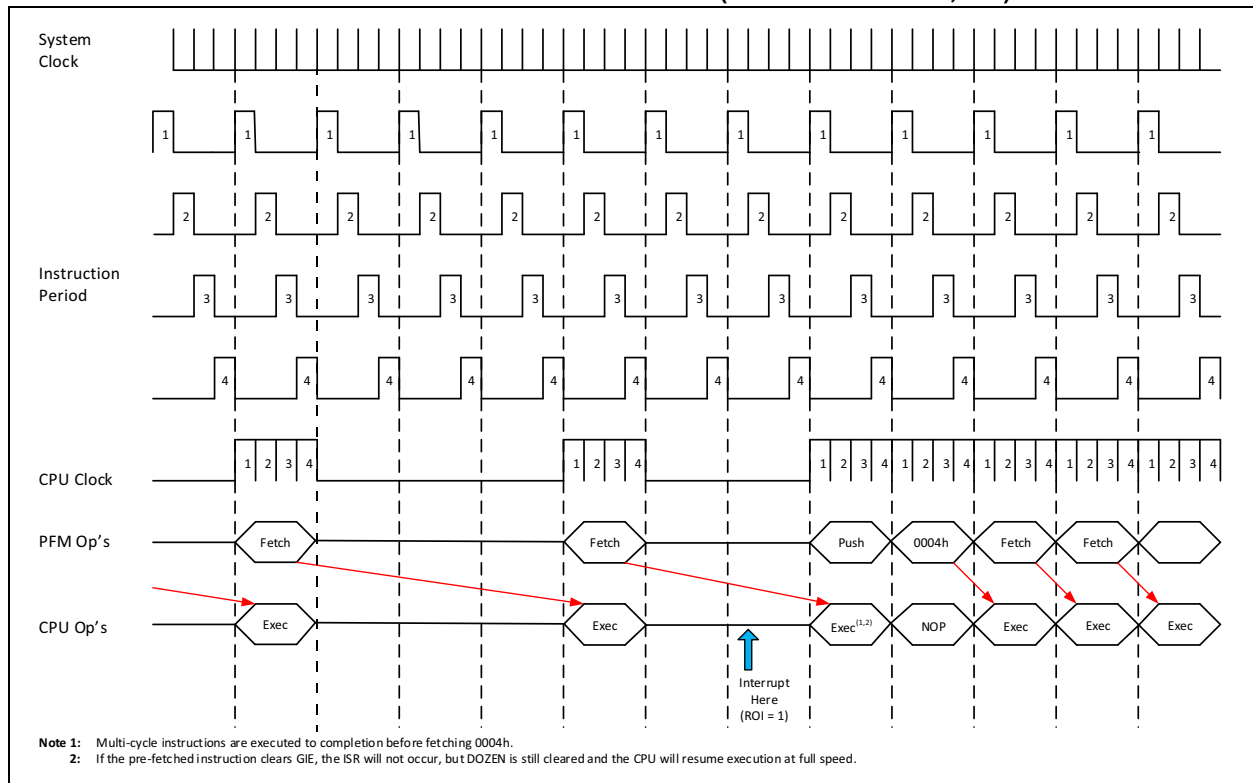
6.1.1 DOZE OPERATION

The Doze operation is illustrated in Figure 6-1. For this example:

- Doze enable (DOZEN) bit set (DOZEN = 1)
- DOZE<2:0> = 001 (1:4) ratio
- Recover-on-Interrupt (ROI) bit set (ROI = 1)

As with normal operation, the PFM fetches for the next instruction cycle. The Q-clocks to the peripherals continue throughout.

FIGURE 6-1: DOZE MODE OPERATION EXAMPLE (DOZE<2:0> = 001, 1:4)



6.1.2 INTERRUPTS DURING DOZE

If an interrupt occurs and the Recover-On-Interrupt bit is clear (ROI = 0) at the time of the interrupt, the Interrupt Service Routine (ISR) continues to execute at the rate selected by DOZE<2:0>. Interrupt latency is extended by the DOZE<2:0> ratio.

If an interrupt occurs and the ROI bit is set (ROI = 1) at the time of the interrupt, the DOZEN bit is cleared and the CPU executes at full speed. The prefetched instruction is executed and then the interrupt vector sequence is executed. In [Figure 6-1](#), the interrupt occurs during the 2nd instruction cycle of the Doze period, and immediately brings the CPU out of Doze. If the Doze-On-Exit (DOE) bit is set (DOE = 1) when the RETFIE operation is executed, DOZEN is set, and the CPU executes at the reduced rate based on the DOZE<2:0> ratio.

EXAMPLE 6-1: 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;
    }
}

```

6.2 Sleep Mode

Sleep mode is entered by executing the SLEEP instruction, while the Idle Enable (IDLEN) bit of the CPUDOZE register is clear (IDLEN = 0).

Upon entering Sleep mode, the following conditions exist:

1. WDT will be cleared but keeps running if enabled for operation during Sleep
2. The \overline{PD} bit of the STATUS register is cleared ([Register 10-2](#))
3. The \overline{TO} bit of the STATUS register is set ([Register 10-2](#))
4. The CPU clock is disabled
5. LFINTOSC, SOSC, HFINTOSC and ADCRC are unaffected and peripherals using them may continue operation in 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

Refer to individual chapters for more details on peripheral operation during Sleep.

To minimize current consumption, the following conditions should be considered:

- I/O pins should not be floating
- External circuitry sinking current from I/O pins
- Internal circuitry sourcing current from I/O pins
- Current draw from pins with internal weak pull-ups
- Modules using any oscillator

I/O pins that are high-impedance inputs should be pulled to VDD or VSS externally to avoid switching currents caused by floating inputs.

Examples of internal circuitry that might be sourcing current include modules such as the DAC and FVR modules. See [Section 31.0 “5-Bit Digital-to-Analog Converter \(DAC\) Module”](#) and [Section 29.0 “Fixed Voltage Reference \(FVR\)”](#) for more information on these modules.

6.2.1 WAKE-UP FROM SLEEP

The device can wake-up from Sleep through one of the following events:

1. External Reset input on $\overline{\text{MCLR}}$ 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.

The first five events will cause a device Reset. The last one event is considered a continuation of program execution. To determine whether a device Reset or wake-up event occurred, refer to [Section 8.13 “Determining the Cause of a Reset”](#).

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, as well as the Peripheral Interrupt Enable bit (`PEIE = 1`), for every interrupt not in `PIR0`. Wake-up will occur regardless of the state of the `GIE` bit. If the `GIE` bit is disabled, the device continues execution at the instruction after the `SLEEP` instruction. If the `GIE` bit is enabled, the device executes the instruction after the `SLEEP` instruction, the device will then call the Interrupt Service Routine. In cases where the execution of the instruction following `SLEEP` is not desirable, the user should have a `NOP` 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 a Sleep event, the core will wait for a combination of three conditions before beginning execution. The conditions are:

- PFM Ready
- `COSC`-Selected Oscillator Ready
- BOR Ready (unless BOR is disabled)

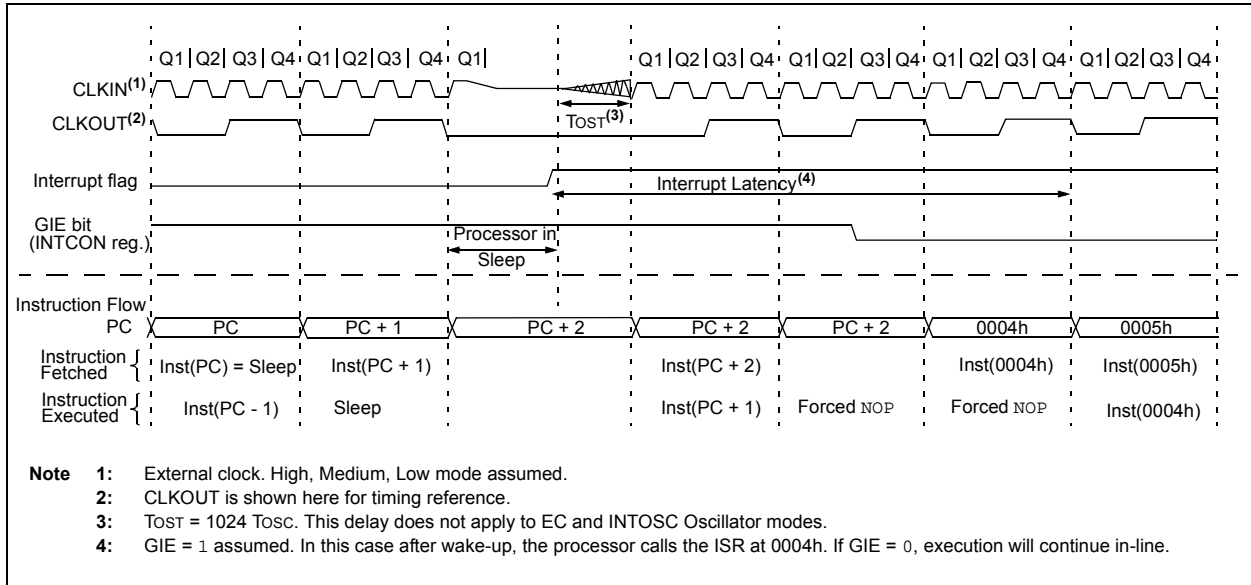
6.2.2 WAKE-UP USING INTERRUPTS

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
 - `SLEEP` instruction will execute as a `NOP`
 - `WDT` and `WDT` prescaler will not be cleared
 - $\overline{\text{TO}}$ bit of the `STATUS` register will not be set
 - $\overline{\text{PD}}$ bit of the `STATUS` register will not be cleared
- If the interrupt occurs **during or after** the execution of a `SLEEP` instruction
 - `SLEEP` instruction will be completely executed
 - Device will immediately wake-up from Sleep
 - `WDT` and `WDT` prescaler will be cleared
 - $\overline{\text{TO}}$ bit of the `STATUS` register will be set
 - $\overline{\text{PD}}$ bit of the `STATUS` register will be cleared

Even if the 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 $\overline{\text{PD}}$ bit. If the $\overline{\text{PD}}$ bit is set, the `SLEEP` instruction was executed as a `NOP`.

FIGURE 6-2: WAKE-UP FROM SLEEP THROUGH INTERRUPT



6.2.3 LOW-POWER SLEEP MODE

The PIC18F6xK40 device family contains an internal Low Dropout (LDO) voltage regulator, which allows the device I/O pins to operate at voltages up to 5.5V while the internal device logic operates at a lower voltage. The LDO and its associated reference circuitry must remain active when the device is in Sleep mode.

The PIC18F6xK40 devices allows the user to optimize the operating current in Sleep, depending on the application requirements.

Low-Power Sleep mode can be selected by setting the VREGPM bit of the VREGCON register.

6.2.3.1 Sleep Current vs. Wake-up Time

In the default operating mode, the LDO and reference circuitry remain in the normal configuration while in Sleep. The device is able to exit Sleep mode quickly since all circuits remain active. In Low-Power Sleep mode, when waking-up from Sleep, an extra delay time is required for these circuits to return to the normal configuration and stabilize.

The Low-Power Sleep mode is beneficial for applications that stay in Sleep mode for long periods of time. The Normal mode is beneficial for applications that need to wake from Sleep quickly and frequently.

6.2.3.2 Peripheral Usage in Sleep

Some peripherals that can operate in Sleep mode will not operate properly with the Low-Power Sleep mode selected. The Low-Power Sleep mode is intended for use with these peripherals:

- Brown-out Reset (BOR)
- Windowed Watchdog Timer (WWDT)
- External interrupt pin/Interrupt-On-Change pins
- Peripherals that run off external secondary clock source

It is the responsibility of the end user to determine what is acceptable for their application when setting the VREGPM settings in order to ensure operation in Sleep.

Note: The PIC18LF6xK40 devices do not have a configurable Low-Power Sleep mode. PIC18LF6xK40 devices are unregulated and are always in the lowest power state when in Sleep, with no wake-up time penalty. These devices have a lower maximum V_{DD} and I/O voltage than the PIC18F6xK40 devices. See [Section 37.0 “Electrical Specifications”](#) for more information.

6.2.4 IDLE MODE

When IDLEN is set (IDLEN = 1), 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 PFM are shut off.

Note: If CLKOUTEN is enabled (CLKOUTEN = 0, Configuration Word 1H), the output will continue operating while in Idle.

6.2.4.1 Idle and Interrupts

IDLE mode ends when an interrupt occurs (even if GIE = 0), 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.

6.2.4.2 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.

Note: The WDT can bring the device out of Idle, in the same way it brings the device out of Sleep. The DOZEN bit is not affected.

6.3 Peripheral Operation in Power Saving Modes

All selected clock sources and the peripherals running off them are active in both IDLE and DOZE mode. Only in Sleep mode, both the Fosc and Fosc/4 clocks are unavailable. All the other clock sources are active, if enabled manually or through peripheral clock selection before the part enters Sleep.

6.4 Register Definitions: Voltage Regulator Control

REGISTER 6-1: VREGCON: VOLTAGE REGULATOR CONTROL REGISTER⁽¹⁾

| | | | | | | | |
|-------|-----|-----|-----|-----|-----|---------|----------|
| U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | R/W-0/0 | R/W-1/1 |
| — | — | — | — | — | — | VREGPM | Reserved |
| bit 7 | | | | | | bit 0 | |

Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

bit 7-2 **Unimplemented:** Read as '0'

bit 1 **VREGPM:** Voltage Regulator Power Mode Selection bit

1 = Low-Power Sleep mode enabled in Sleep⁽²⁾

 Draws lowest current in Sleep, slower wake-up

0 = Normal Power mode enabled in Sleep⁽²⁾

 Draws higher current in Sleep, faster wake-up

bit 0 **Reserved:** Read as '1'. Maintain this bit set.

Note 1: PIC18F6xK40 only.

2: See [Section 37.0 "Electrical Specifications"](#).

PIC18(L)F67K40

REGISTER 6-2: CPUDOZE: DOZE AND IDLE REGISTER

| | | | | | | | |
|---------|---------------|---------|---------|-----|-----------|---------|---------|
| R/W-0/u | R/W/HC/HS-0/0 | R/W-0/0 | R/W-0/0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| IDLEN | DOZEN | ROI | DOE | — | DOZE<2:0> | | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | HC = Bit is cleared by hardware |

- bit 7 **IDLEN:** Idle Enable bit
 1 = A *SLEEP* instruction inhibits the CPU clock, but not the peripheral clock(s)
 0 = A *SLEEP* instruction places the device into full Sleep mode
- bit 6 **DOZEN:** Doze Enable bit^(1,2)
 1 = The CPU executes instruction cycles according to DOZE setting
 0 = The CPU executes all instruction cycles (fastest, highest power operation)
- bit 5 **ROI:** Recover-On-Interrupt bit
 1 = Entering the Interrupt Service Routine (ISR) makes DOZEN = 0 bit, bringing the CPU to full-speed operation
 0 = Interrupt entry does not change DOZEN
- bit 4 **DOE:** Doze-On-Exit bit
 1 = Executing RETFIE makes DOZEN = 1, bringing the CPU to reduced speed operation
 0 = RETFIE does not change DOZEN
- bit 3 **Unimplemented:** Read as '0'
- bit 2-0 **DOZE<2:0>:** Ratio of CPU Instruction Cycles to Peripheral Instruction Cycles
 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, DOZEN is changed by hardware interrupt entry and/or exit.
Note 2: Entering ICD overrides DOZEN, returning the CPU to full execution speed; this bit is not affected.

TABLE 6-1: SUMMARY OF REGISTERS ASSOCIATED WITH POWER-DOWN MODE

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on Page |
|---------|--------|------------|------------|-----------------|-----------------|-------------|-----------|----------|------------------|
| INTCON | GIE | PEIE | — | — | — | — | — | INTEDG | 172 |
| PIE0 | — | — | TMR0IE | IOCFIE | INT3IE | INT2IE | INT1IE | INT0IE | 184 |
| PIE1 | OSCFIE | CSWIE | — | — | — | — | ADTIE | ADIE | 185 |
| PIE2 | HLVDIE | ZCDIE | — | — | — | — | C2IE | C1IE | 186 |
| PIE3 | RC2IE | TX2IE | RC1IE | TX1IE | BCL2IE | SSP2IE | BCL1IE | SSP1IE | 187 |
| PIE4 | — | — | RC5IE | TX5IE | RC4IE | TX4IE | RC3IE | TX3IE | 188 |
| PIE5 | — | — | — | — | — | TMR3IE | TMR2IE | TMR1IE | 189 |
| PIE6 | — | — | — | — | TMR7GIE | TMR5GIE | TMR3GIE | TMR1GIE | 190 |
| PIE7 | — | — | — | CCP5IE | CCP4IE | CCP3IE | CCP2IE | CCP1IE | 191 |
| PIE8 | SCANIE | CRCIE | NVMIE | — | — | — | — | CWG1IE | 192 |
| PIE9 | — | — | SMT2PWAIE | SMT2PRAIE | SMT2IE | SMT1PWAIE | SMT1PRAIE | SMT1IE | 192 |
| STATUS | — | — | — | \overline{TO} | \overline{PD} | Z | DC | C | 120 |
| VREGCON | — | — | — | — | — | — | VREGPM | Reserved | 61 |
| CPUDOZE | IDLEN | DOZEN | ROI | DOE | — | DOZE<2:0> | | | 62 |
| WDTCON0 | — | — | WDTPS<4:0> | | | | — | SEN | 82 |
| WDTCON1 | — | WDTPS<2:0> | | | — | WINDOW<2:0> | | | 83 |

Note 1: — = unimplemented location, read as '0'. Shaded cells are not used in Power-Down mode.

7.0 PERIPHERAL MODULE DISABLE (PMD)

Sleep, Idle and Doze modes allow users to substantially reduce power consumption by slowing or stopping the CPU clock. Even so, peripheral modules still remain clocked, and thus, consume some amount of power. There may be cases where the application needs what these modes do not provide: the ability to allocate limited power resources to the CPU while eliminating power consumption from the peripherals.

The PIC18(L)F6xK40 family addresses this requirement by allowing peripheral modules to be selectively enabled or disabled, placing them into the lowest possible power mode.

For legacy reasons, all modules are ON by default following any Reset.

7.1 Disabling a Module

Disabling a module has the following effects:

- All clock and control inputs to the module are suspended; there are no logic transitions, and the module will not function.
- The module is held in Reset.
- Any SFR becomes “unimplemented”
 - Writing is disabled
 - Reading returns 00h
- I/O functionality is prioritized as per [Section 15.1, I/O Priorities](#)
- All associated Input Selection registers are also disabled

7.2 Enabling a Module

When the PMD register bit is cleared, the module is re-enabled and will be in its Reset state (Power-on Reset). SFR data will reflect the POR Reset values.

Depending on the module, it may take up to one full instruction cycle for the module to become active. There should be no interaction with the module (e.g., writing to registers) for at least one instruction after it has been re-enabled.

7.3 Effects of a Reset

Following any Reset, each control bit is set to ‘0’, enabling all modules.

7.4 System Clock Disable

Setting SYSCMD (PMD0, [Register 7-1](#)) disables the system clock (FOSC) distribution network to the peripherals. Not all peripherals make use of SYSCLK, so not all peripherals are affected. Refer to the specific peripheral description to see if it will be affected by this bit.

7.5 Register Definitions: Peripheral Module Disable

REGISTER 7-1: PMD0: PMD CONTROL REGISTER 0

| | | | | | | | |
|---------|---------|---------|---------|---------|---------|---------|---------|
| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| SYSCMD | FVRMD | HLVDMD | CRCMD | SCANMD | NVMMD | CLKRMD | IOCMD |
| 7 | | | | | | | 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | q = Value depends on condition |

- bit 7 **SYSCMD:** Disable Peripheral System Clock Network bit⁽¹⁾
 See description in [Section 7.4 “System Clock Disable”](#).
 1 = System clock network disabled (Fosc)
 0 = System clock network enabled
- bit 6 **FVRMD:** Disable Fixed Voltage Reference bit
 1 = FVR module disabled
 0 = FVR module enabled
- bit 5 **HLVDMD:** Disable Low-Voltage Detect bit
 1 = HLVD module disabled
 0 = HLVD module enabled
- bit 4 **CRCMD:** Disable CRC Engine bit
 1 = CRC module disabled
 0 = CRC module enabled
- bit 3 **SCANMD:** Disable NVM Memory Scanner bit⁽²⁾
 1 = NVM Memory Scan module disabled
 0 = NVM Memory Scan module enabled
- bit 2 **NVMMD:** NVM Module Disable bit⁽³⁾
 1 = All Memory reading and writing is disabled; NVMCON registers cannot be written
 0 = NVM module enabled
- bit 1 **CLKRMD:** Disable Clock Reference bit
 1 = CLKR module disabled
 0 = CLKR module enabled
- bit 0 **IOCMD:** Disable Interrupt-on-Change bit, All Ports
 1 = IOC module(s) disabled
 0 = IOC module(s) enabled

- Note 1:** Clearing the SYSCMD bit disables the system clock (Fosc) to peripherals, however peripherals clocked by Fosc/4 are not affected.
- 2:** Subject to SCANE bit in CONFIG4H.
- 3:** When enabling NVM, a delay of up to 1 μ s may be required before accessing data.

REGISTER 7-2: PMD1: PMD CONTROL REGISTER 1

| | | | | | | | |
|---------|---------|---------|---------|---------|---------|---------|---------|
| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| TMR7MD | TMR6MD | TMR5MD | TMR4MD | TMR3MD | TMR2MD | TMR1MD | TMR0MD |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | q = Value depends on condition |

| | |
|-------|--|
| bit 7 | TMR7MD: Disable Timer TMR7 bit 1 = TMR7 module disabled 0 = TMR7 module enabled |
| bit 6 | TMR6MD: Disable Timer TMR6 bit 1 = TMR6 module disabled 0 = TMR6 module enabled |
| bit 5 | TMR5MD: Disable Timer TMR5 bit 1 = TMR5 module disabled 0 = TMR5 module enabled |
| bit 4 | TMR4MD: Disable Timer TMR4 bit 1 = TMR4 module disabled 0 = TMR4 module enabled |
| bit 3 | TMR3MD: Disable Timer TMR3 bit 1 = TMR3 module disabled 0 = TMR3 module enabled |
| bit 2 | TMR2MD: Disable Timer TMR2 bit 1 = TMR2 module disabled 0 = TMR2 module enabled |
| bit 1 | TMR1MD: Disable Timer TMR1 bit 1 = TMR1 module disabled 0 = TMR1 module enabled |
| bit 0 | TMR0MD: Disable Timer TMR0 bit 1 = TMR0 module disabled 0 = TMR0 module enabled |

REGISTER 7-3: PMD2: PMD CONTROL REGISTER 2

| | | | | | | | |
|-------|-----|---------|-----|---------|---------|---------|---------|
| U-0 | U-0 | R/W-0/0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| — | — | CWGMD | — | DSMMD | SMT2MD | SMT1MD | TMR8MD |
| bit 7 | | | | | | | bit 0 |

Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

q = Value depends on condition

- bit 7-6 **Unimplemented:** Read as '0'
- bit 5 **CWGMD:** Disable CWG bit
1 = CWG module disabled
0 = CWG module enabled
- bit 4 **Unimplemented:** Read as '0'
- bit 3 **DSMMD:** Disable Comparator DSM bit
1 = DSM module disabled
0 = DSM module enabled
- bit 2 **SMT2MD:** Disable Comparator SMT2 bit
1 = SMT2 module disabled
0 = SMT2 module enabled
- bit 1 **SMT1MD:** Disable Comparator SMT1 bit
1 = SMT1 module disabled
0 = SMT1 module enabled
- bit 0 **TMR8MD:** Disable Comparator TMR8 bit
1 = TMR8 module disabled
0 = TMR8 module enabled

REGISTER 7-4: PMD3: PMD CONTROL REGISTER 3

| U-0 | R/W-0/0 | R/W-0/0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|-------|---------|---------|-----|---------|---------|---------|----------------------|
| — | DACMD | ADCMD | — | CMP3MD | CMP2MD | CMP1MD | ZCDMD ⁽¹⁾ |
| bit 7 | | | | | | | bit 0 |

Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

q = Value depends on condition

bit 7 **Unimplemented:** Read as '0'

bit 6 **DACMD:** Disable DAC bit
 1 = DAC module disabled
 0 = DAC module enabled

bit 5 **ADCMD:** Disable ADC bit
 1 = ADC module disabled
 0 = ADC module enabled

bit 4 **Unimplemented:** Read as '0'

bit 3 **CMP3MD:** Disable Comparator C3 bit
 1 = C3 module disabled
 0 = C3 module enabled

bit 2 **CMP2MD:** Disable Comparator C2 bit
 1 = C2 module disabled
 0 = C2 module enabled

bit 1 **CMP1MD:** Disable Comparator C1 bit
 1 = C1 module disabled
 0 = C1 module enabled

bit 0 **ZCDMD:** Disable ZCD bit⁽¹⁾
 1 = ZCD module disabled
 0 = ZCD module enabled

Note 1: Subject to ZCD bit in CONFIG2H.

REGISTER 7-5: PMD4: PMD CONTROL REGISTER 4

| | | | | | | | |
|-------|---------|---------|---------|---------|---------|---------|---------|
| U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| — | PWM7MD | PWM6MD | CCP5MD | CCP4MD | CCP3MD | CCP2MD | CCP1MD |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | q = Value depends on condition |

| | |
|-------|---|
| bit 7 | Unimplemented: Read as '0' |
| bit 6 | PWM7MD: Disable PWM7 Module bit 1 = PWM7 module disabled 0 = PWM7 module enabled |
| bit 5 | PWM6MD: Disable PWM6 Module bit 1 = PWM6 module disabled 0 = PWM6 module enabled |
| bit 4 | CCP5MD: Disable CCP5 Module bit 1 = CCP5 module disabled 0 = CCP5 module enabled |
| bit 3 | CCP4MD: Disable CCP4 Module bit 1 = CCP4 module disabled 0 = CCP4 module enabled |
| bit 2 | CCP3MD: Disable CCP3 Module bit 1 = CCP3 module disabled 0 = CCP3 module enabled |
| bit 1 | CCP2MD: Disable CCP2 Module bit 1 = CCP2 module disabled 0 = CCP2 module enabled |
| bit 0 | CCP1MD: Disable CCP1 Module bit 1 = CCP1 module disabled 0 = CCP1 module enabled |

PIC18(L)F67K40

REGISTER 7-6: PMD5: PMD CONTROL REGISTER 5

| | | | | | | | |
|-------|----------|----------|----------|----------|----------|---------|---------|
| U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| — | USART5MD | USART4MD | USART3MD | USART2MD | USART1MD | MSSP2MD | MSSP1MD |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | q = Value depends on condition |

| | |
|-------|---|
| bit 7 | Unimplemented: Read as '0' |
| bit 6 | USART5MD: Disable USART5 Module bit 1 = USART5 module disabled 0 = USART5 module enabled |
| bit 5 | USART4MD: Disable USART4 Module bit 1 = USART4 module disabled 0 = USART4 module enabled |
| bit 4 | USART3MD: Disable USART3 Module bit 1 = USART3 module disabled 0 = USART3 module enabled |
| bit 3 | USART2MD: Disable USART2 Module bit 1 = USART2 module disabled 0 = USART2 module enabled |
| bit 2 | USART1MD: Disable USART4 Module bit 1 = USART1 module disabled 0 = USART1 module enabled |
| bit 1 | MSSP2MD: Disable MSSP2 Module bit 1 = MSSP2 module disabled 0 = MSSP2 module enabled |
| bit 0 | MSSP1MD: Disable MSSP1 Module bit 1 = MSSP1 module disabled 0 = MSSP1 module enabled |

TABLE 7-1: SUMMARY OF REGISTERS ASSOCIATED WITH PMD

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on page |
|------|--------|----------|----------|----------|----------|----------|---------|---------|------------------|
| PMD0 | SYSCMD | FVRMD | HLVDMD | CRCMD | SCANMD | NVMMD | CLKRMD | IOCMD | 65 |
| PMD1 | TMR7MD | TMR6MD | TMR5MD | TMR4MD | TMR3MD | TMR2MD | TMR1MD | TMR0MD | 66 |
| PMD2 | — | — | CWGMD | — | DSMMD | SMT2MD | SMT1MD | TMR8MD | 67 |
| PMD3 | — | DACMD | ADCMD | — | CMP3MD | CMP2MD | CMP1MD | ZCDMD | 68 |
| PMD4 | — | PWM7MD | PWM6MD | CCP5MD | CCP4MD | CCP3MD | CCP2MD | CCP1MD | 69 |
| PMD5 | — | USART5MD | USART4MD | USART3MD | USART2MD | USART1MD | MSSP2MD | MSSP1MD | 70 |

Legend: — = unimplemented, read as '0'. Shaded cells are not used by the PMD.

8.0 RESETS

There are multiple ways to reset this device:

- Power-on Reset (POR)
- Brown-out Reset (BOR)
- Low-Power Brown-Out Reset (LPBOR)
- MCLR Reset
- WDT Reset
- RESET instruction
- Stack Overflow
- Stack Underflow
- Programming mode exit

To allow VDD to stabilize, an optional Power-up Timer can be enabled to extend the Reset time after a BOR or POR event.

A simplified block diagram of the On-Chip Reset Circuit is shown in [Figure 8-1](#).

FIGURE 8-1: SIMPLIFIED BLOCK DIAGRAM OF ON-CHIP RESET CIRCUIT

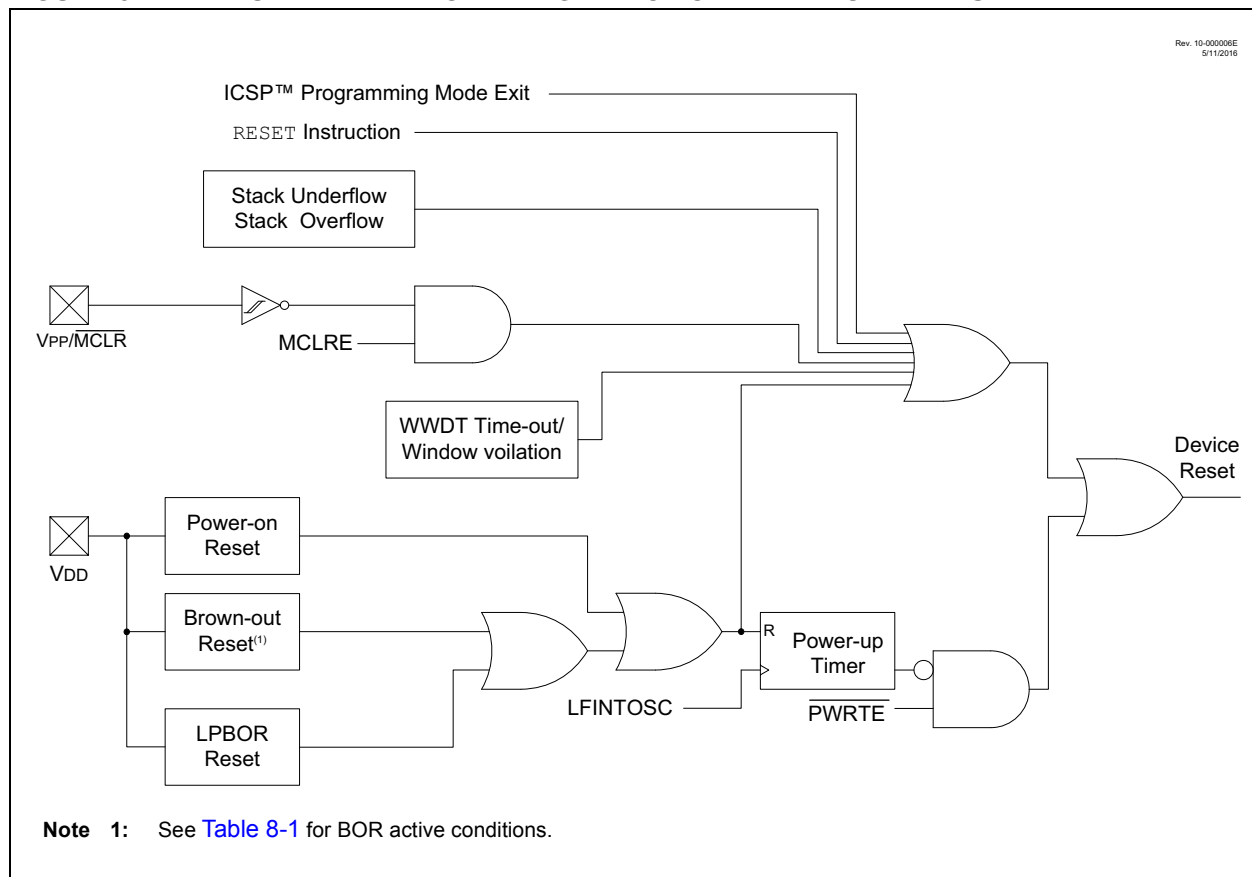
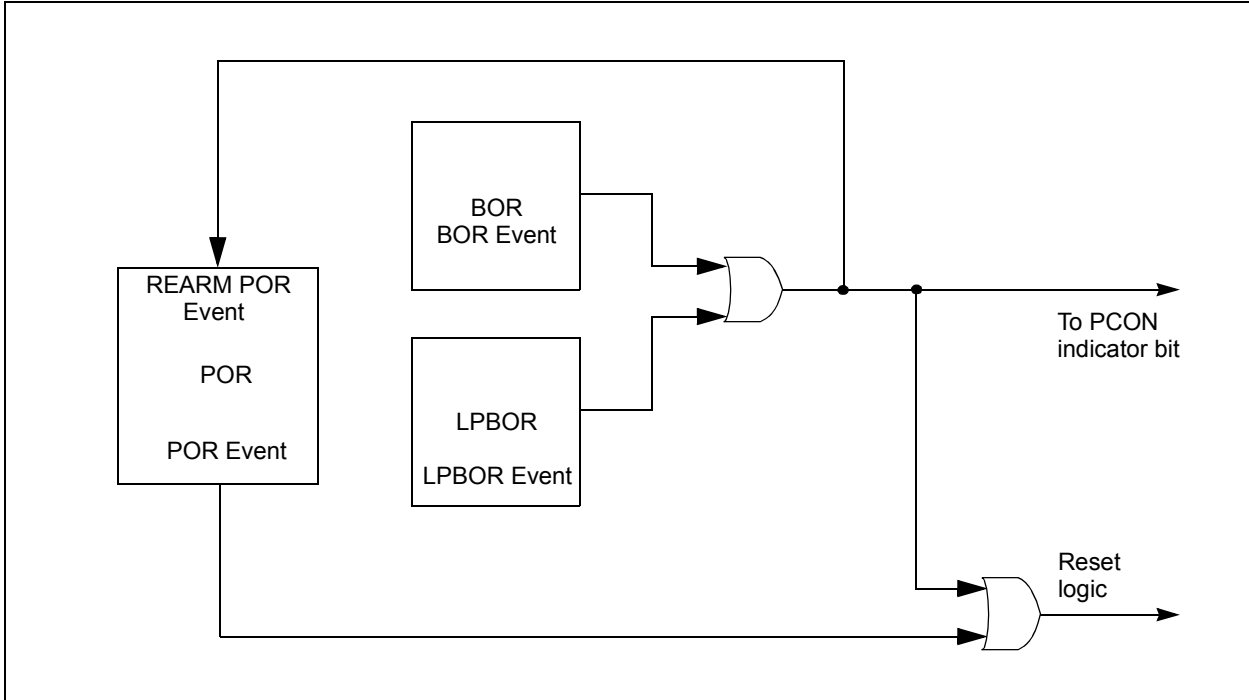


FIGURE 8-2: LPBOR, BOR, POR RELATIONSHIP



8.1 Register Definitions: BOR Control

REGISTER 8-1: BORCON: BROWN-OUT RESET CONTROL REGISTER

| | | | | | | | |
|---------|-----|-----|-----|-----|-----|-----|--------|
| R/W-1/u | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | R-q/u |
| SBOREN | — | — | — | — | — | — | BORRDY |
| bit 7 | | | | | | | bit 0 |

Legend:

R = Readable bit W = Writable bit U = Unimplemented bit, read as '0'
u = Bit is unchanged x = Bit is unknown -n/n = Value at POR and BOR/Value at all other Resets
'1' = Bit is set '0' = Bit is cleared q = Value depends on condition

bit 7 **SBOREN:** Software Brown-out Reset Enable bit

If BOREN ≠ 01:

SBOREN is read/write, but has no effect on the BOR.

If BOREN = 01:

1 = BOR Enabled

0 = BOR Disabled

bit 6-1 **Unimplemented:** Read as '0'

bit 0 **BORRDY:** Brown-out Reset Circuit Ready Status bit

1 = The Brown-out Reset Circuit is active and armed

0 = The Brown-out Reset Circuit is disabled or is warming up

8.2 Register Definitions: Power Control

REGISTER 8-2: PCON0: POWER CONTROL REGISTER 0

| | | | | | | | |
|------------|------------|---------------------------|--------------------------|---------------------------|------------------------|-------------------------|-------------------------|
| R/W/HS-0/q | R/W/HS-0/q | R/W/HC-1/q | R/W/HC-1/q | R/W/HC-1/q | R/W/HC-1/q | R/W/HC-0/u | R/W/HC-q/u |
| STKOVF | STKUNF | $\overline{\text{WDTWV}}$ | $\overline{\text{RWDT}}$ | $\overline{\text{RMCLR}}$ | $\overline{\text{RI}}$ | $\overline{\text{POR}}$ | $\overline{\text{BOR}}$ |
| bit 7 | | | | | | | bit 0 |

Legend:

| | |
|---------------------------------|---|
| HC = Bit is cleared by hardware | HS = Bit is set by hardware |
| R = Readable bit | W = Writable bit |
| u = Bit is unchanged | x = Bit is unknown |
| '1' = Bit is set | '0' = Bit is cleared |
| | U = Unimplemented bit, read as '0' |
| | -m/n = Value at POR and BOR/Value at all other Resets |
| | q = Value depends on condition |

| | |
|-------|--|
| bit 7 | <p>STKOVF: Stack Overflow Flag bit</p> <p>1 = A Stack Overflow occurred (more CALLS than fit on the stack)</p> <p>0 = A Stack Overflow has not occurred or set to '0' by firmware</p> |
| bit 6 | <p>STKUNF: Stack Underflow Flag bit</p> <p>1 = A Stack Underflow occurred (more RETURNS than CALLS)</p> <p>0 = A Stack Underflow has not occurred or set to '0' by firmware</p> |
| bit 5 | <p>$\overline{\text{WDTWV}}$: Watchdog Window Violation bit</p> <p>1 = A WDT window violation has not occurred or set to '1' by firmware</p> <p>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)</p> |
| bit 4 | <p>$\overline{\text{RWDT}}$: WDT Reset Flag bit</p> <p>1 = A WDT overflow/time-out Reset has not occurred or set to '1' by firmware</p> <p>0 = A WDT overflow/time-out Reset has occurred (set to '0' in hardware when a WDT Reset occurs)</p> |
| bit 3 | <p>$\overline{\text{RMCLR}}$: MCLR Reset Flag bit</p> <p>1 = A MCLR Reset has not occurred or set to '1' by firmware</p> <p>0 = A MCLR Reset has occurred (set to '0' in hardware when a MCLR Reset occurs)</p> |
| bit 2 | <p>$\overline{\text{RI}}$: RESET Instruction Flag bit</p> <p>1 = A RESET instruction has not been executed or set to '1' by firmware</p> <p>0 = A RESET instruction has been executed (set to '0' in hardware upon executing a RESET instruction)</p> |
| bit 1 | <p>$\overline{\text{POR}}$: Power-on Reset Status bit</p> <p>1 = No Power-on Reset occurred or set to '1' by firmware</p> <p>0 = A Power-on Reset occurred (set to '0' in hardware when a Power-on Reset occurs)</p> |
| bit 0 | <p>$\overline{\text{BOR}}$: Brown-out Reset Status bit</p> <p>1 = No Brown-out Reset occurred or set to '1' by firmware</p> <p>0 = A Brown-out Reset occurred (set to '0' in hardware when a Brown-out Reset occurs)</p> |

8.3 Power-on Reset (POR)

The POR circuit holds the device in Reset until VDD has reached an acceptable level for minimum operation. Slow rising VDD, fast operating speeds or analog performance may require greater than minimum VDD. The PWRT, BOR or MCLR features can be used to extend the start-up period until all device operation conditions have been met.

8.4 Brown-out Reset (BOR)

The BOR circuit holds the device in Reset when VDD reaches a selectable minimum level. Between the POR and BOR, complete voltage range coverage for execution protection can be implemented.

The Brown-out Reset module has four operating modes controlled by the BOREN<1:0> bits in Configuration Words. 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 [Table 8-1](#) for more information.

The Brown-out Reset voltage level is selectable by configuring the BORV<1:0> bits in Configuration Words.

A VDD noise rejection filter prevents the BOR from triggering on small events. If VDD falls below VBOR for a duration greater than parameter TBORDC, the device will reset. See [Table 37-11](#) for more information.

8.4.1 BOR IS ALWAYS ON

When the BOREN bits of Configuration Words are programmed to '11', the BOR is always on. The device start-up will be delayed until the BOR is ready and VDD is higher than the BOR threshold.

BOR protection is active during Sleep. The BOR does not delay wake-up from Sleep.

8.4.2 BOR IS OFF IN SLEEP

When the BOREN bits of Configuration Words are programmed to '10', the BOR is on, except in Sleep. The device start-up will be delayed until the BOR is ready and VDD 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.

8.4.3 BOR CONTROLLED BY SOFTWARE

When the BOREN bits of Configuration Words are programmed to '01', the BOR is controlled by the SBOREN bit of the BORCON register. The device start-up is not delayed by the BOR ready condition or the VDD level.

BOR protection begins as soon as the BOR circuit is ready. The status of the BOR circuit is reflected in the BORRDY bit of the BORCON register.

BOR protection is unchanged by Sleep.

8.4.4 BOR AND BULK ERASE

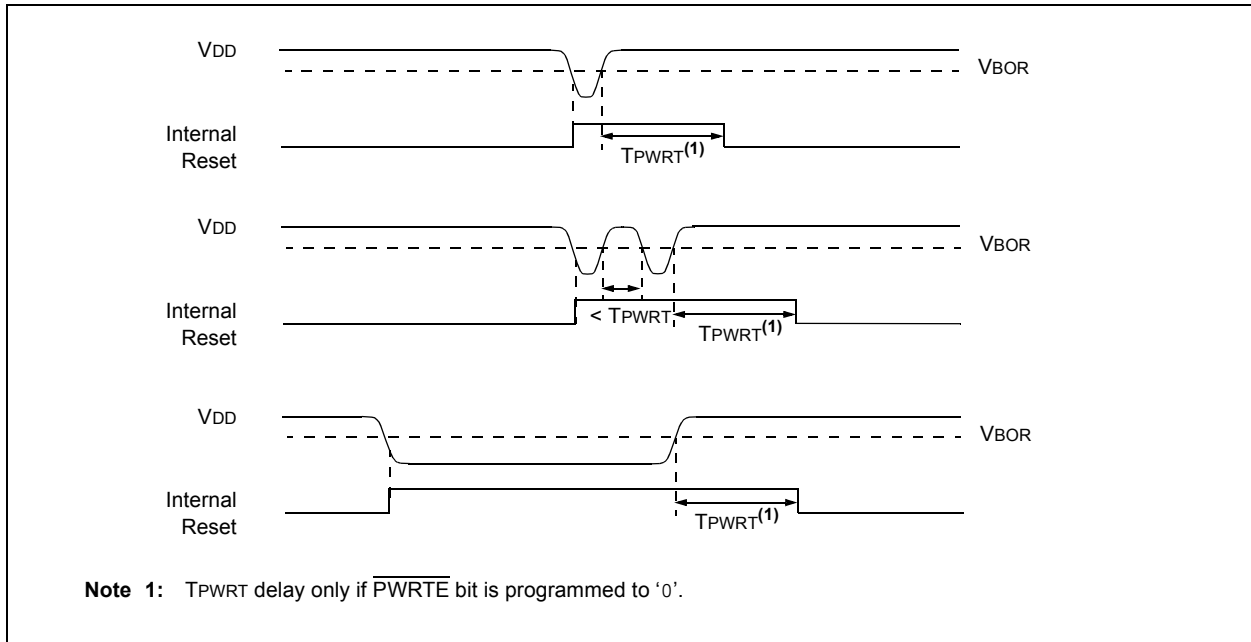
BOR is forced ON during PFM Bulk Erase operations to make sure that the system code protection cannot be compromised by reducing VDD.

During Bulk Erase, the BOR is enabled at 2.45V for F and LF devices, even if it is configured to some other value. If VDD falls, the erase cycle will be aborted, but the device will not be reset.

TABLE 8-1: BOR OPERATING MODES

| BOREN<1:0> | SBOREN | Device Mode | BOR Mode | Instruction Execution upon: | |
|------------|--------|-------------|-----------|--------------------------------------|--------------------------------------|
| | | | | Release of POR | Wake-up from Sleep |
| 11 | 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 | |

FIGURE 8-3: BROWN-OUT SITUATIONS



8.5 Low-Power Brown-out Reset (LPBOR)

The Low-Power Brown-out Reset (LPBOR) provides an additional BOR circuit for low power operation. Refer to [Figure 8-2](#) to see how the BOR interacts with other modules.

The LPBOR is used to monitor the external VDD pin. When too low of a voltage is detected, the device is held in Reset.

8.5.1 ENABLING LPBOR

The LPBOR is controlled by the $\overline{\text{LPBOREN}}$ bit of Configuration Word 2L. When the device is erased, the LPBOR module defaults to disabled.

8.5.1.1 LPBOR Module Output

The output of the LPBOR module is a signal indicating whether or not a Reset is to be asserted. This signal is OR'd together with the $\overline{\text{Reset}}$ 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.

8.6 $\overline{\text{MCLR}}$

The $\overline{\text{MCLR}}$ is an optional external input that can reset the device. The $\overline{\text{MCLR}}$ function is controlled by the MCLRE bit of Configuration Words and the LVP bit of Configuration Words ([Table 8-2](#)). The $\overline{\text{RMCLR}}$ bit in the PCON0 register will be set to '0' if a MCLR has occurred.

TABLE 8-2: $\overline{\text{MCLR}}$ CONFIGURATION

| MCLRE | LVP | $\overline{\text{MCLR}}$ |
|-------|-----|--------------------------|
| x | 1 | Enabled |
| 1 | 0 | Enabled |
| 0 | 0 | Disabled |

8.6.1 $\overline{\text{MCLR}}$ 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 VDD 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.

Note: An internal Reset event (RESET, instr, BOR, WWDT, POR STK), does not drive the $\overline{\text{MCLR}}$ pin low.

8.6.2 $\overline{\text{MCLR}}$ DISABLED

When $\overline{\text{MCLR}}$ is disabled, the $\overline{\text{MCLR}}$ becomes input-only and pin functions such as internal weak pull-ups are under software control. See [Section 15.1 "I/O Priorities"](#) for more information.

8.7 Windowed Watchdog Timer (WWDT) Reset

The Windowed Watchdog Timer generates a Reset if the firmware does not issue a $\overline{\text{CLRWDT}}$ instruction within the time-out period or window set. The $\overline{\text{TO}}$ and $\overline{\text{PD}}$ bits in the STATUS register and the $\overline{\text{RWDT}}$ bit in the PCON0 register are changed to indicate a WDT Reset. The $\overline{\text{WDTWV}}$ bit in the PCON0 register indicates if the WDT Reset has occurred due to a time out or a window violation. See [Section 9.0 "Windowed Watchdog Timer \(WWDT\)"](#) for more information.

8.8 RESET Instruction

A RESET instruction will cause a device Reset. The $\overline{\text{RI}}$ bit in the PCON0 register will be set to '0'. See [Table 8-3](#) for default conditions after a RESET instruction has occurred.

8.9 Stack Overflow/Underflow Reset

The device can reset when the Stack Overflows or Underflows. The STKOVF or STKUNF bits of the PCON0 register indicate the Reset condition. These Resets are enabled by setting the STVREN bit in Configuration Words. See [Section 10.2.1 "Stack Overflow and Underflow Resets"](#) for more information.

8.10 Programming Mode Exit

Upon exit of Programming mode, the device will behave as if a POR had just occurred.

8.11 Power-up Timer (PWRT)

The Power-up Timer provides a nominal 66 ms (2048 cycles of LFINTOSC) time out 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 the VDD to rise to an acceptable level. The Power-up Timer is enabled by clearing the $\overline{\text{PWRTTE}}$ bit in Configuration Words.

The Power-up Timer starts after the release of the POR and BOR.

For additional information, refer to Application Note AN607, "Power-up Trouble Shooting" (DS00000607).

8.12 Start-up Sequence

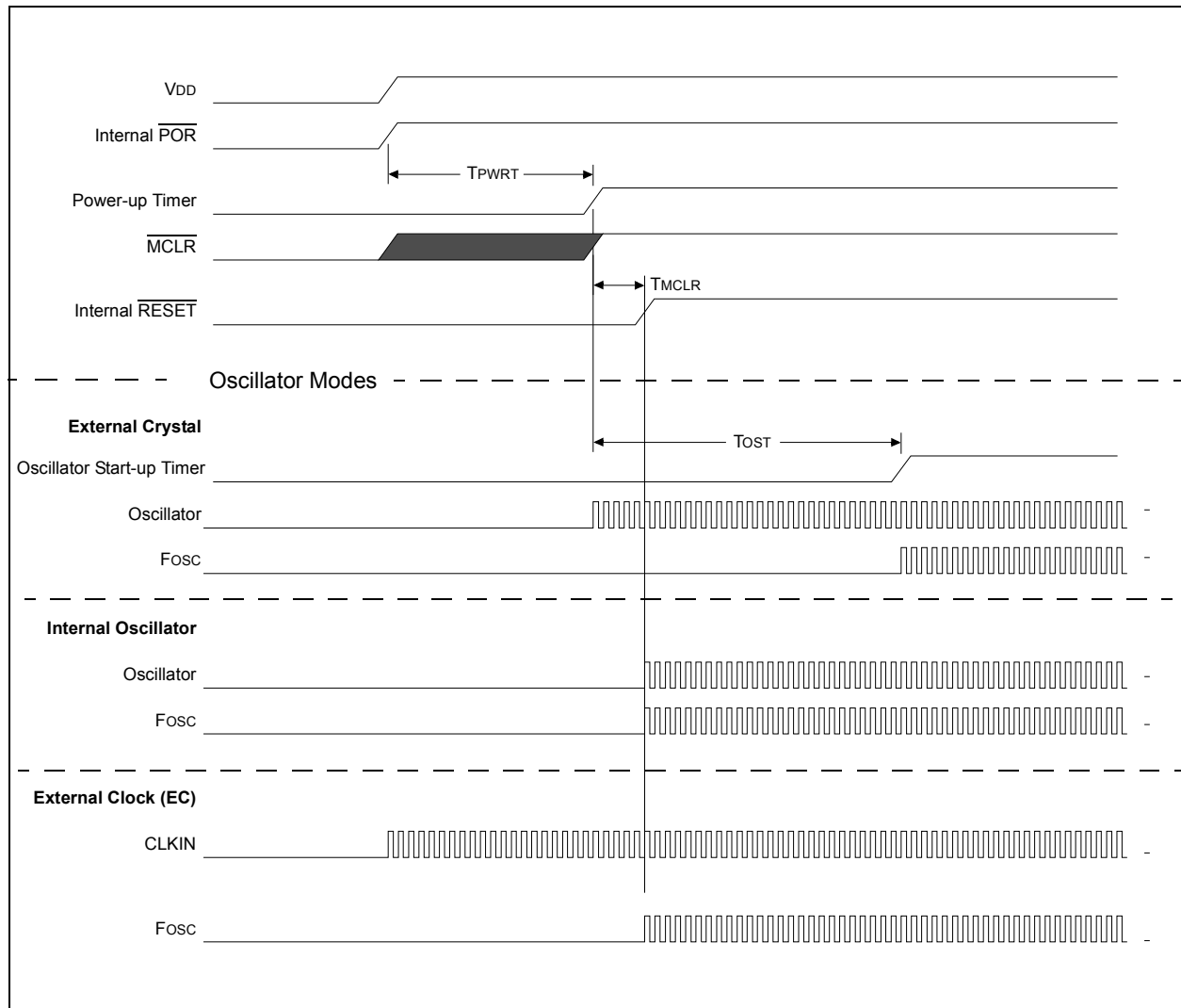
Upon the release of a POR or BOR, the following must occur before the device will begin executing:

1. Power-up Timer runs to completion (if enabled).
2. Oscillator start-up timer runs to completion (if required for selected oscillator source).
3. $\overline{\text{MCLR}}$ must be released (if enabled).

The total time out will vary based on oscillator configuration and Power-up Timer configuration. See [Section 4.0 “Oscillator Module \(with Fail-Safe Clock Monitor\)”](#) for more information.

The Power-up Timer and oscillator start-up timer run independently of $\overline{\text{MCLR}}$ Reset. If $\overline{\text{MCLR}}$ is kept low long enough, the Power-up Timer and oscillator Start-up Timer will expire. Upon bringing $\overline{\text{MCLR}}$ high, the device will begin execution after 10 Fosc cycles (see [Figure 8-4](#)). This is useful for testing purposes or to synchronize more than one device operating in parallel.

FIGURE 8-4: RESET START-UP SEQUENCE



8.13 Determining the Cause of a Reset

Upon any Reset, multiple bits in the STATUS and PCON0 registers are updated to indicate the cause of the Reset. [Table 8-3](#) shows the Reset conditions of these registers.

TABLE 8-3: RESET CONDITION FOR SPECIAL REGISTERS

| Condition | Program Counter | STATUS Register ^(2,3) | PCON0 Register |
|------------------------------------|-----------------------|----------------------------------|----------------|
| Power-on Reset | 0 | -110 0000 | 0011 110x |
| Brown-out Reset | 0 | -110 0000 | 0011 11u0 |
| MCLR Reset during normal operation | 0 | -uuu uuuu | uuuu 0uuu |
| MCLR Reset during Sleep | 0 | -10u uuuu | uuuu 0uuu |
| WDT Time-out Reset | 0 | -0uu uuuu | uuu0 uuuu |
| WDT Wake-up from Sleep | PC + 2 | -00u uuuu | uuuu uuuu |
| WWDT Window Violation Reset | 0 | -uuu uuuu | uu0u uuuu |
| Interrupt Wake-up from Sleep | PC + 2 ⁽¹⁾ | -10u 0uuu | uuuu uuuu |
| RESET Instruction Executed | 0 | -uuu uuuu | uuuu u0uu |
| Stack Overflow Reset (STVREN = 1) | 0 | -uuu uuuu | 1uuu uuuu |
| Stack Underflow Reset (STVREN = 1) | 0 | -uuu uuuu | u1uu uuuu |

Legend: u = unchanged, x = unknown, – = unimplemented bit, reads as '0'.

Note 1: When the wake-up is due to an interrupt and Global Interrupt Enable bit (GIE) 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.

2: If a Status bit is not implemented, that bit will be read as '0'.

3: Status bits Z, C, DC are reset by POR/BOR ([Register 10-2](#)).

8.14 Power Control (PCON0) Register

The Power Control (PCON0) register contains flag bits to differentiate between a:

- Brown-out Reset ($\overline{\text{BOR}}$)
- Power-on Reset ($\overline{\text{POR}}$)
- Reset Instruction Reset ($\overline{\text{RI}}$)
- MCLR Reset ($\overline{\text{RMCLR}}$)
- Watchdog Timer Reset ($\overline{\text{RWDT}}$)
- Watchdog Window Violation ($\overline{\text{WDTWV}}$)
- Stack Underflow Reset (STKUNF)
- Stack Overflow Reset (STKOVF)

The PCON0 register bits are shown in [Register 8-2](#).

Hardware will change the corresponding register bit during the Reset process; if the Reset was not caused by the condition, the bit remains unchanged ([Table 8-3](#)).

Software should 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.

TABLE 8-4: SUMMARY OF REGISTERS ASSOCIATED WITH RESETS

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on Page |
|---------|--------|------------------------|---------------------------|--------------------------|---------------------------|------------------------|-------------------------|-------------------------|---------------------|
| BORCON | SBOREN | — | — | — | — | — | — | BORRDY | 72 |
| PCON0 | STKOVF | STKUNF | $\overline{\text{WDTWV}}$ | $\overline{\text{RWDT}}$ | $\overline{\text{RMCLR}}$ | $\overline{\text{RI}}$ | $\overline{\text{POR}}$ | $\overline{\text{BOR}}$ | 73 |
| STATUS | — | $\overline{\text{TO}}$ | $\overline{\text{PD}}$ | N | OV | Z | DC | C | 120 |
| WDTCON0 | — | — | WDTPS<4:0> | | | | | SEN | 82 |
| WDTCON1 | — | WDTCS<2:0> | | | — | WINDOW<2:0> | | | 83 |

Legend: — = unimplemented location, read as '0'. Shaded cells are not used by Resets.

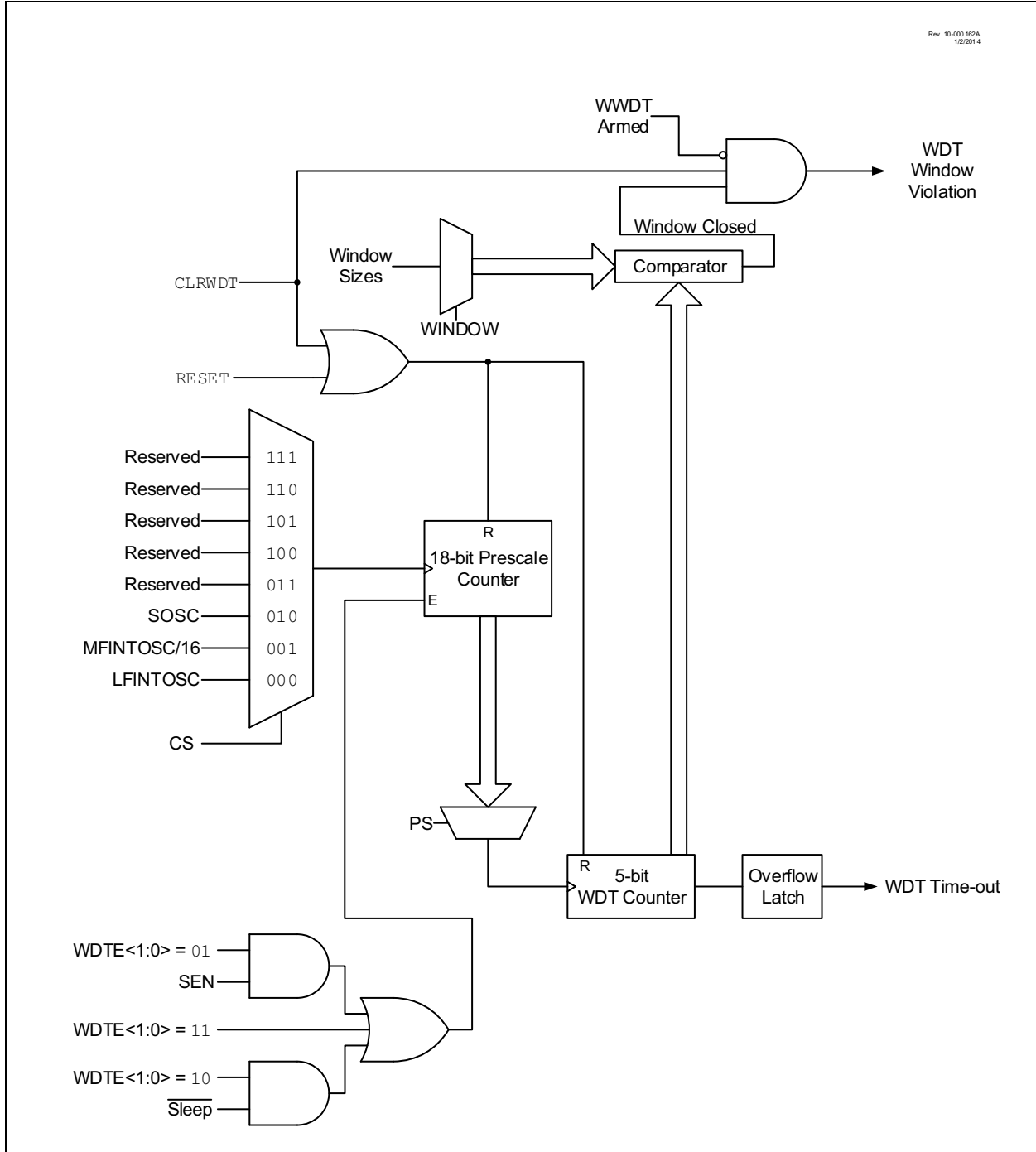
9.0 WINDOWED WATCHDOG TIMER (WWDT)

The 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. The Watchdog Timer is typically used to recover the system from unexpected events. The Windowed Watchdog Timer (WWDT) differs 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 is from 1 ms to 256s (nominal)
- Configurable window size from 12.5% to 100% of the time-out period
- Multiple Reset conditions

FIGURE 9-1: WINDOWED WATCHDOG TIMER BLOCK DIAGRAM



9.1 Register Definitions: Windowed Watchdog Timer Control

REGISTER 9-1: WDTCON0: WATCHDOG TIMER CONTROL REGISTER 0

| U-0 | U-0 | R/W ⁽³⁾ -q/q ⁽²⁾ | R/W ⁽³⁾ -q/q ⁽²⁾ | R/W ⁽³⁾ -q/q ⁽²⁾ | R/W ⁽³⁾ -q/q ⁽²⁾ | R/W ⁽³⁾ -q/q ⁽²⁾ | R/W-0/0 |
|-------|-----|--|--|--|--|--|---------|
| — | — | WDTPS<4:0> | | | | | SEN |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | q = Value depends on condition |

bit 7-6 **Unimplemented:** Read as '0'

bit 5-1 **WDTPS<4:0>:** Watchdog Timer Prescale Select bits⁽¹⁾

Bit Value = Prescale Rate

11111 = Reserved. Results in minimum interval (1:32)

•
•
•

10011 = Reserved. Results in minimum interval (1:32)

10010 = 1:8388608 (2²³) (Interval 256s nominal)

10001 = 1:4194304 (2²²) (Interval 128s nominal)

10000 = 1:2097152 (2²¹) (Interval 64s nominal)

01111 = 1:1048576 (2²⁰) (Interval 32s nominal)

01110 = 1:524288 (2¹⁹) (Interval 16s nominal)

01101 = 1:262144 (2¹⁸) (Interval 8s nominal)

01100 = 1:131072 (2¹⁷) (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 bit

If WDTE<1:0> = 1x:

This bit is ignored.

If WDTE<1:0> = 01:

1 = WDT is turned on

0 = WDT is turned off

If WDTE<1:0> = 00:

This bit is ignored.

Note 1: Times are approximate. WDT time is based on 31 kHz LFINTOSC.

2: When WDTCPS <4:0> in CONFIG3L = 11111, the Reset value of WDTPS<4:0> is 01011. Otherwise, the Reset value of WDTPS<4:0> is equal to WDTCPS<4:0> in CONFIG3L.

3: When WDTCPS <4:0> in CONFIG3L ≠ 11111, these bits are read-only.

REGISTER 9-2: WDTCON1: WATCHDOG TIMER CONTROL REGISTER 1

| | | | | | | | |
|-------|--|--|--|-----|--|--|--|
| U-0 | R/W ⁽³⁾ -q/q ⁽¹⁾ | R/W ⁽³⁾ -q/q ⁽¹⁾ | R/W ⁽³⁾ -q/q ⁽¹⁾ | U-0 | R/W ⁽⁴⁾ -q/q ⁽²⁾ | R/W ⁽⁴⁾ -q/q ⁽²⁾ | R/W ⁽⁴⁾ -q/q ⁽²⁾ |
| — | WDTCS<2:0> | | | — | WINDOW<2:0> | | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | q = Value depends on condition |

bit 7 **Unimplemented:** Read as '0'

bit 6-4 **WDTCS<2:0>:** Watchdog Timer Clock Select bits

111 = Reserved

•

•

•

010 = Reserved

001 = MFINTOSC 31.25 kHz

000 = LFINTOSC 31 kHz

bit 3 **Unimplemented:** Read as '0'

bit 2-0 **WINDOW<2:0>:** Watchdog Timer Window Select bits

| WINDOW<2:0> | 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 |

- Note 1:** If WDTCCS <2:0> in CONFIG3H = 111, the Reset value of WDTCS<2:0> is 000.
- 2:** The Reset value of WINDOW<2:0> is determined by the value of WDTCCS<2:0> in the CONFIG3H register.
- 3:** If WDTCCS<2:0> in CONFIG3H ≠ 111, these bits are read-only.
- 4:** If WDTCCS<2:0> in CONFIG3H ≠ 111, these bits are read-only.

REGISTER 9-3: WDTPSL: WWDT PRESCALE SELECT LOW BYTE REGISTER (READ-ONLY)

| | | | | | | | |
|------------|-------|-------|-------|-------|-------|-------|-------|
| R-0/0 | R-0/0 | R-0/0 | R-0/0 | R-0/0 | R-0/0 | R-0/0 | R-0/0 |
| PSCNT<7:0> | | | | | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-0 **PSCNT<7:0>**: Prescale Select Low Byte bits⁽¹⁾

Note 1: The 18-bit WDT prescale 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 should be read during normal operation.

REGISTER 9-4: WDTPSH: WWDT PRESCALE SELECT HIGH BYTE REGISTER (READ-ONLY)

| | | | | | | | |
|-------------|-------|-------|-------|-------|-------|-------|-------|
| R-0/0 | R-0/0 | R-0/0 | R-0/0 | R-0/0 | R-0/0 | R-0/0 | R-0/0 |
| PSCNT<15:8> | | | | | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-0 **PSCNT<15:8>**: Prescale Select High Byte bits⁽¹⁾

Note 1: The 18-bit WDT prescale 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 should be read during normal operation.

REGISTER 9-5: WDTTMR: WDT TIMER REGISTER (READ-ONLY)

| | | | | | | | |
|-------------|-------|-------|-------|-------|-------|--------------|-------|
| R-0/0 | R-0/0 | R-0/0 | R-0/0 | R-0/0 | R-0/0 | R-0/0 | R-0/0 |
| WDTTMR<4:0> | | | | | STATE | PSCNT<17:16> | |
| bit 7 | | | | | bit 0 | | |

Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

 bit 7-3 **WDTTMR<4:0>**: Watchdog Window Value bits

| 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 bit

1 = WDT is armed

0 = WDT is not armed

 bit 1-0 **PSCNT<17:16>**: Prescale Select Upper Byte bits⁽¹⁾

Note 1: The 18-bit WDT prescale 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 should be read during normal operation.

9.2 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 WDTE<1:0> Configuration bits.

If WDTE = 2'b1x, then the clock source will be enabled depending on the WDTCCS<2:0> Configuration bits.

If WDTE = 2'b01, the SEN bit should be set by software to enable WWDT, and the clock source is enabled by the WDTCS bits in the WDTCON1 register.

Time intervals in this chapter are based on a minimum nominal interval of 1 ms. See [Section 37.0 “Electrical Specifications”](#) for LFINTOSC and MFINTOSC tolerances.

9.3 WWDT Operating Modes

The Windowed Watchdog Timer module has four operating modes controlled by the WDTE<1:0> bits in Configuration Words. See [Table 9-1](#).

9.3.1 WWDT IS ALWAYS ON

When the WDTE bits of Configuration Words are set to '11', the WWDT is always on.

WWDT protection is active during Sleep.

9.3.2 WWDT IS OFF IN SLEEP

When the WDTE bits of Configuration Words are set to '10', the WWDT is on, except in Sleep.

WWDT protection is not active during Sleep.

9.3.3 WWDT CONTROLLED BY SOFTWARE

When the WDTE bits of Configuration Words are set to '01', the WWDT is controlled by the SEN bit of the WDTCON0 register.

WWDT protection is unchanged by Sleep. See [Table 9-1](#) for more details.

TABLE 9-1: WWDT OPERATING MODES

| WDTE<1:0> | 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 |

9.4 Time-out Period

If the WDTCCPS<4:0> Configuration bits default to 5'b11111, then the WDTPS bits of the WDTCON0 register set the time-out period from 1 ms to 256 seconds (nominal). If any value other than the default value is assigned to WDTCCPS<4:0> Configuration bits, then the timer period will be based on the WDTCCPS<4:0> bits in the CONFIG3L register. After a Reset, the default time-out period is 2s.

9.5 Watchdog Window

The Windowed Watchdog Timer has an optional Windowed mode that is controlled by the WDTCCWS<2:0> Configuration bits and WINDOW<2:0> bits of the WDTCON1 register. In the Windowed mode, 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 9-2](#) for an example.

The window size is controlled by the WINDOW<2:0> Configuration bits, or the WINDOW<2:0> bits of WDTCON1, if WDTCCWS<2:0> = 111.

The five Most Significant bits of the WDTTMR register are used to determine whether the window is open, as defined by the WINDOW<2:0> bits of the WDTCON1 register.

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 or can be set in firmware.

9.6 Clearing the WWDT

The WWDT is cleared when any of the following conditions occur:

- Any Reset
- Valid CLRWDT instruction is executed
- Device enters Sleep
- Exit Sleep by Interrupt
- WWDT is disabled
- Oscillator Start-up Timer (OST) is running
- Any write to the WDTCON0 or WDTCON1 registers

9.6.1 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 9-2](#) for more information.

9.7 Operation During Sleep

When the device enters Sleep, the WWDT is cleared. If the WWDT is enabled during Sleep, the WWDT resumes counting. When the device exits Sleep, the WWDT is cleared again.

The WWDT remains clear until the Oscillator Start-up Timer (OST) completes, if enabled. See [Section 4.3.1.3 “Oscillator Start-up Timer \(OST\)”](#) for more information on the OST.

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 RWDT bit in the PCON0 register can also be used. See [Section 10.0 “Memory Organization”](#) for more information.

TABLE 9-2: WWDT CLEARING CONDITIONS

| Conditions | WWDT |
|---|------------------------------|
| WDTE<1:0> = 00 | Cleared |
| WDTE<1:0> = 01 and SEN = 0 | |
| WDTE<1:0> = 10 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 |

FIGURE 9-2: WINDOW PERIOD AND DELAY

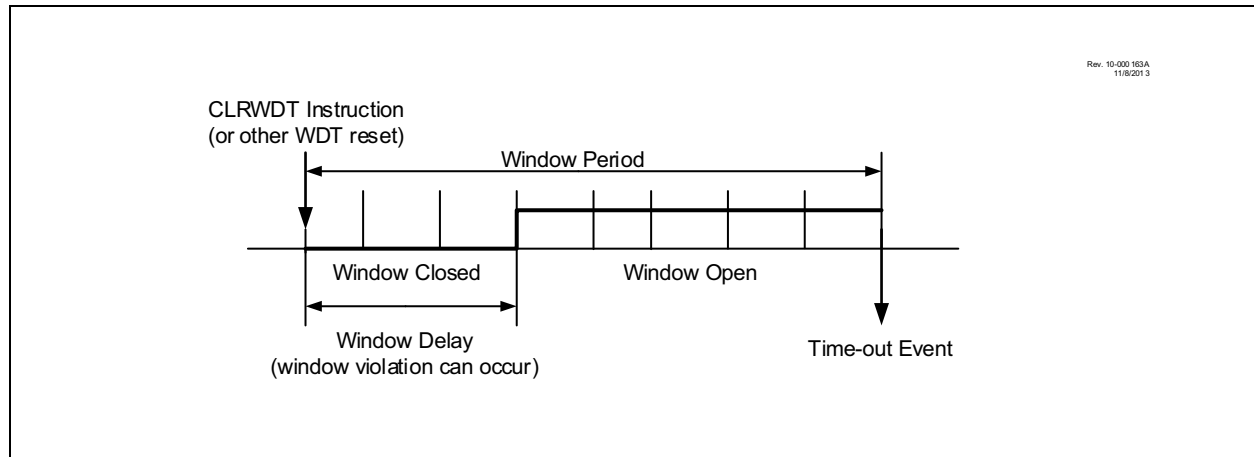


TABLE 9-3: SUMMARY OF REGISTERS ASSOCIATED WITH WINDOWED WATCHDOG TIMER

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on Page |
|---------|-------------|------------|---------------------------|-------------------------------------|---------------------------|------------------------|-------------------------|-------------------------|---------------------|
| PCON0 | STKOVF | STKUNF | $\overline{\text{WDTWV}}$ | $\overline{\text{RWD\overline{T}}}$ | $\overline{\text{RMCLR}}$ | $\overline{\text{RI}}$ | $\overline{\text{POR}}$ | $\overline{\text{BOR}}$ | 73 |
| STATUS | — | — | — | $\overline{\text{TO}}$ | $\overline{\text{PD}}$ | Z | DC | C | 120 |
| WDTCON0 | — | — | WDTPS<4:0> | | | | | SEN | 82 |
| WDTCON1 | — | WDTCS<2:0> | | | — | WINDOW<2:0> | | | 83 |
| WDTPSL | PSCNT<7:0> | | | | | | | | 84 |
| WDTPSH | PSCNT<15:8> | | | | | | | | 84 |
| WDTTMR | WDTTMR<4:0> | | | | | STATE | PSCNT<17:16> | | 85 |

Legend: x = unknown, u = unchanged, — = unimplemented locations read as '0'. Shaded cells are not used by Windowed Watchdog Timer.

10.0 MEMORY ORGANIZATION

There are three types of memory in PIC18 enhanced microcontroller devices:

- Program Memory
- Data RAM
- Data EEPROM

As Harvard architecture devices, the data and program memories use separate buses; this allows 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 [Section 11.0 “Nonvolatile Memory \(NVM\) Control”](#).

10.1 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).

These devices contains the following:

- PIC18(L)F67K40: 128 Kbytes of Flash memory, up to 65,536 single-word instructions

PIC18 devices have two interrupt vectors. The Reset vector address is at 0000h and the interrupt vector addresses are at 0008h and 0018h.

| |
|--|
| Note: For memory information on this family of devices, see Table 10-1 and Table 10-2 . |
|--|

PIC18(L)F67K40

TABLE 10-1: PROGRAM AND DATA EEPROM MEMORY MAP

| | PIC18(L)F65K40 | PIC18(L)F66K40 | PIC18(L)F67K40 | |
|---------------|--|---------------------------------|---------------------------------|---------------|
| | PC<21:0> | PC<21:0> | PC<21:0> | |
| Note 1 | Stack (31 levels) | Stack (31 levels) | Stack (31 levels) | Note 1 |
| 00 0000h | Reset Vector | Reset Vector | Reset Vector | 00 0000h |
| ... | ... | ... | ... | ... |
| 00 0008h | Interrupt Vector High | Interrupt Vector High | Interrupt Vector High | 00 0008h |
| ... | ... | ... | ... | ... |
| 00 0018h | Interrupt Vector Low | Interrupt Vector Low | Interrupt Vector Low | 00 0018h |
| 00 001Ah | Program Flash Memory (16 KW) | Program Flash Memory (32 KW) | Program Flash Memory (64 KW) | 00 001Ah |
| 00 3FFFh | | | | 00 3FFFh |
| 00 4000h | | | | 00 4000h |
| 00 7FFFh | | | | 00 7FFFh |
| 00 8000h | Not present ⁽²⁾ | Not present ⁽²⁾ | Not present ⁽²⁾ | 00 8000h |
| 00 FFFFh | | | | 00 FFFFh |
| 01 0000h | | | | 01 0000h |
| 01 FFFFh | | | | 01 FFFFh |
| 02 0000h | | | | 02 0000h |
| 1F FFFFh | | | | 1F FFFFh |
| 20 0000h | User IDs (8 Words) ⁽³⁾ | | | 20 0000h |
| 20 000Fh | | | | 20 000Fh |
| 20 0010h | Reserved | | | 20 0010h |
| 2F FFFFh | | | | 2F FFFFh |
| 30 0000h | Configuration Words (6 Words) ⁽³⁾ | | | 30 0000h |
| 30 000Bh | | | | 30 000Bh |
| 30 000Ch | Reserved | | | 30 000Ch |
| 30 FFFFh | | | | 30 FFFFh |
| 31 0000h | DataEEByte0 ... DataEEByte1023 | | | 31 0000h |
| ... | | | | ... |
| 31 03FFFh | | | | 31 03FFFh |
| 31 0400h | | | | 31 0400h |
| 3F FFFBh | Reserved | | | 3F FFFBh |
| 3F FFFCh | Revision ID (1 Word) ⁽⁴⁾ | | | 3F FFFCh |
| 3F FFFDh | | | | 3F FFFDh |
| 3F FFFEh | Device ID (1 Word) ⁽⁴⁾ | | | 3F FFFEh |
| 3F FFFFh | | | | 3F FFFFh |

- Note 1:** The stack is a separate SRAM panel, apart from all user memory panels.
Note 2: The addresses do not roll over. The region is read as '0'.
Note 3: Not code-protected.
Note 4: Device/Revision IDs are hard-coded in silicon.

PIC18(L)F67K40

TABLE 10-2: MEMORY MAP AND CODE PROTECTION CONTROL

| Reg. | Address (from/to) | Device | | | |
|------|----------------------|--|------------------------------------|------------------------------------|------------------------------------|
| | | PIC18(L)F65K40 | PIC18(L)F66K40 | PIC18(L)F67K40 | |
| PFM | 00 0000h 00 07FFh | Boot Block 1 KW CP, WRTB, EBTRB | Boot Block 1 KW CP WRTB, EBTRB | Boot Block 1 KW CP WRTB, EBTRB | |
| | 00 0800h 00 1FFFh | Block 0 3 KW CP, WRT0, EBTR0 | Block 0 7 KW CP, WRT0, EBTR0 | Block 0 7 KW CP, WRT0, EBTR0 | |
| | 00 2000h 00 3FFFh | Block 1 4 KW CP, WRT1, EBTR1 | | | |
| | 00 4000h 00 5FFFh | Block 2 4 KW CP, WRT2, EBTR2 | Block 1 8 KW CP, WRT1, EBTR1 | Block 1 8 KW CP, WRT1, EBTR1 | |
| | 00 6000h 00 7FFFh | Block 3 4 KW CP, WRT3, EBTR3 | | | |
| | 00 8000h 00 BFFFh | Not present | Block 2 8 KW CP, WRT2, EBTR2 | Block 2 8 KW CP, WRT2, EBTR2 | |
| | 00 C000h 00 FFFFh | | Block 3 8 KW CP, WRT3, EBTR3 | Block 3 8 KW CP, WRT3, EBTR3 | |
| | 01 0000h 01 3FFFh | | Not present | Not present | Block 4 8 KW CP, WRT4, EBTR4 |
| | 01 4000h 01 7FFFh | | | | Block 5 8 KW CP, WRT5, EBTR5 |
| | 01 8000h 01 BFFFh | | | | Block 6 8 KW CP, WRT6, EBTR6 |
| | 01 C000h 01 FFFFh | | | | Block 7 8 KW CP, WRT7, EBTR7 |
| | CONFIG | 30 0000h 30 000Bh | 6 Words WRTC | | |
| | Data EEPROM | 31 0000h 31 00FFh 31 0100h 31 03FFh | 1 KW CPD, WRTD | | |

10.1.1 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 [Section 10.2.3.1 “Computed GOTO”](#)).

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.

10.1.2 RETURN ADDRESS STACK

The return address stack allows any combination of up to 31 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 operates as a 31-word by 21-bit RAM and a 5-bit Stack Pointer, or as a 35-word by 21-bit RAM with a 6-bit Stack Pointer in ICD mode. The stack space is not part of either program or data space. 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 '00000' after all Resets. There is no RAM associated with the location corresponding to a Stack Pointer value of '00000'; this is only a Reset value. Status bits in the PCON0 register indicate if the stack is full or has overflowed or has underflowed.

10.1.2.1 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 ([Figure 10-1](#)). 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.

10.2 Register Definitions: Stack Pointer

REGISTER 10-1: STKPTR: STACK POINTER REGISTER

| | | | | | | | |
|-------|-----|-----|-------------|-------|-------|-------|-------|
| U-0 | U-0 | U-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
| — | — | — | STKPTR<4:0> | | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | | |
|-------------------|------------------|----------------------|------------------------|
| R = Readable bit | W = Writable bit | U = Unimplemented | C = Clearable only bit |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared | x = Bit is unknown |

bit 7-5 **Unimplemented:** Read as '0'
bit 4-0 **STKPTR<4:0>:** Stack Pointer Location bits

10.2.1 STACK OVERFLOW AND UNDERFLOW RESETS

Device Resets on Stack Overflow and Stack Underflow conditions are enabled by setting the STVREN bit in Configuration Register 4L. When STVREN is set, a Full or Underflow condition will set the appropriate STKOVF or STKUNF bit and then cause a device Reset. When STVREN is cleared, a Full or Underflow condition will set the appropriate STKOVF or STKUNF bit but not cause a device Reset. The STKOVF or STKUNF bits are cleared by the user software or a Power-on Reset.

10.2.2 FAST REGISTER STACK

A fast register stack is provided for the Status, WREG and BSR registers, to provide a "fast return" option for interrupts. The stack for each register is only one level deep and is neither readable nor writable. 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.

Note: The TO and PD bits of the STATUS register are not copied over in this operation.

If both low and high priority interrupts are enabled, the stack registers cannot be used reliably to return from low priority interrupts. If a high priority interrupt occurs while servicing a low priority interrupt, the stack register values stored by the low priority interrupt will be overwritten. In these cases, users must save the key registers by software during a low priority interrupt.

If interrupt priority is not used, all interrupts may use the fast register stack for returns from interrupt. If no interrupts are used, the fast register stack can be used to restore the STATUS, WREG and BSR registers at the end of a subroutine call. To use the fast register stack for a subroutine call, a CALL label, FAST instruction must be executed to save the STATUS,

WREG and BSR registers to the fast register stack. A RETURN, FAST instruction is then executed to restore these registers from the fast register stack.

Example 10-1 shows a source code example that uses the fast register stack during a subroutine call and return.

EXAMPLE 10-1: 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
```

10.2.3 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

10.2.3.1 Computed GOTO

A computed GOTO is accomplished by adding an offset to the program counter. An example is shown in [Example 10-2](#).

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 should advance and should 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 10-2: COMPUTED GOTO USING AN OFFSET VALUE

```

MOVWF  OFFSET, W
CALL   TABLE
ORG    nn00h
TABLE  ADDWF  PCL
        RETLW nnh
        RETLW nnh
        RETLW nnh
        .
        .
        .
    
```

10.2.3.2 Table Reads and Table Writes

A better 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 [Section 11.1.1 “Table Reads and Table Writes”](#).

10.3 PIC18 Instruction Cycle

10.3.1 CLOCKING SCHEME

The microcontroller clock input, whether from an internal or external source, is internally divided by four to generate four non-overlapping quadrature clocks (Q1, Q2, Q3 and Q4). Internally, the program counter is incremented on every Q1; the instruction is fetched from the program memory and latched into the instruction register during Q4. The instruction is decoded and executed during the following Q1 through Q4. The clocks and instruction execution flow are shown in Figure 10-2.

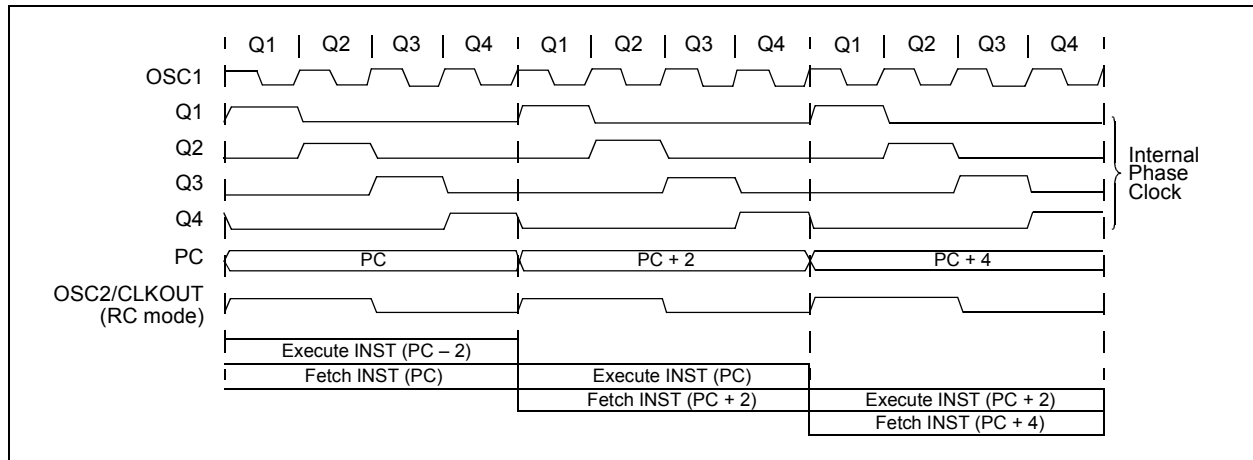
10.3.2 INSTRUCTION FLOW/PIPELINING

An "Instruction Cycle" consists of four Q cycles: Q1 through Q4. 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 to change (e.g., GOTO), then two cycles are required to complete the instruction (Example 10-3).

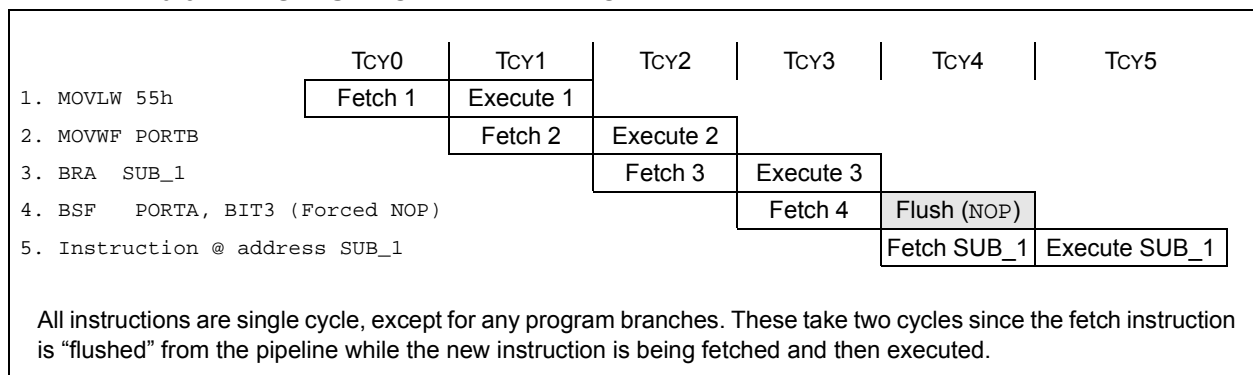
A fetch cycle begins with the Program Counter (PC) incrementing in Q1.

In the execution cycle, the fetched instruction is latched into the Instruction Register (IR) in cycle Q1. This instruction is then decoded and executed during the Q2, Q3 and Q4 cycles. Data memory is read during Q2 (operand read) and written during Q4 (destination write).

FIGURE 10-2: CLOCK/INSTRUCTION CYCLE



EXAMPLE 10-3: INSTRUCTION PIPELINE FLOW



10.3.3 INSTRUCTIONS IN PROGRAM MEMORY

The program memory is addressed in bytes. Instructions are stored as either two bytes or four 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 [Section 10.1.1 “Program Counter”](#)).

[Figure 10-3](#) shows an example of 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 `PC<20:1>`, which accesses the desired byte address in program memory. Instruction #2 in [Figure 10-3](#) 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. [Section 36.0 “Instruction Set Summary”](#) provides further details of the instruction set.

10.3.4 TWO-WORD INSTRUCTIONS

The standard PIC18 instruction set has four two-word instructions: `CALL`, `MOVFF`, `GOTO` and `LFSR`. In all cases, the second 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 4 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. [Example 10-4](#) shows how this works.

Note: See [Section 10.8 “PIC18 Instruction Execution and the Extended Instruction Set”](#) for information on two-word instructions in the extended instruction set.

FIGURE 10-3: INSTRUCTIONS IN PROGRAM MEMORY

| Program Memory Byte Locations → | | | Word Address | | |
|------------------------------------|-------|------------|--------------|---------|---------|
| | | | LSB = 1 | LSB = 0 | |
| | | | | 000000h | |
| | | | | 000002h | |
| | | | | 000004h | |
| | | | | 000006h | |
| Instruction 1: | MOVLW | 055h | 0Fh | 55h | 000008h |
| Instruction 2: | GOTO | 0006h | EFh | 03h | 00000Ah |
| | | | F0h | 00h | 00000Ch |
| Instruction 3: | MOVFF | 123h, 456h | C1h | 23h | 00000Eh |
| | | | F4h | 56h | 000010h |
| | | | | | 000012h |
| | | | | | 000014h |

EXAMPLE 10-4: TWO-WORD INSTRUCTIONS

| CASE 1: | | |
|---------------------|-------------|-------------------------------------|
| Object Code | Source Code | |
| 0110 0110 0000 0000 | TSTFSZ | REG1 ; is RAM location 0? |
| 1100 0001 0010 0011 | MOVFF | REG1, REG2 ; No, skip this word |
| 1111 0100 0101 0110 | | ; Execute this word as a NOP |
| 0010 0100 0000 0000 | ADDWF | REG3 ; continue code |
| CASE 2: | | |
| Object Code | Source Code | |
| 0110 0110 0000 0000 | TSTFSZ | REG1 ; is RAM location 0? |
| 1100 0001 0010 0011 | MOVFF | REG1, REG2 ; Yes, execute this word |
| 1111 0100 0101 0110 | | ; 2nd word of instruction |
| 0010 0100 0000 0000 | ADDWF | REG3 ; continue code |

10.4 Data Memory Organization

Note: The operation of some aspects of data memory are changed when the PIC18 extended instruction set is enabled. See [Section 10.7 “Data Memory and the Extended Instruction Set”](#) for more information.

The data memory in PIC18 devices is implemented as static RAM. Each register in the data memory has a 12-bit address, allowing up to 4096 bytes of data memory. The memory space is divided into as many as 16 banks that contain 256 bytes each. [Figure 10-4](#) shows the data memory organization for the PIC18(L)F6xK40 devices.

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's.

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 256-byte memory space that provides fast access to SFRs and the lower portion of GPR Bank 0 without using the Bank Select Register (BSR). [Section 10.4.2 “Access Bank”](#) provides a detailed description of the Access RAM.

10.4.1 BANK SELECT REGISTER (BSR)

Large areas of data memory require an efficient addressing scheme to make rapid access to any address possible. Ideally, this means that an entire address does not need to be provided for each read or write operation. For PIC18 devices, this is accomplished with a RAM banking scheme. This divides the memory space into 16 contiguous banks of 256 bytes. Depending on the instruction, each location can be addressed directly by its full 12-bit address, or an 8-bit low-order address and a 4-bit 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 four Most Significant bits of a location's address; the instruction itself includes the eight Least Significant bits. Only the four lower bits of the BSR are implemented (BSR<3:0>). The upper four bits are unused; they will always read '0' and cannot be written to. The BSR can be loaded directly by using the `MOVLB` instruction.

The value of the BSR indicates the bank in data memory; 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 10-4](#).

Since up to 16 registers may share the same low-order address, the user must always be careful to ensure that the proper bank is selected before performing a data read or write. For example, writing what should be program data to an 8-bit address of F9h while the BSR is 0Fh will end up resetting the program counter.

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's. Even so, the STATUS register will still be affected as if the operation was successful. The data memory maps in [Figure 10-4](#) indicate which banks are implemented.

In the core PIC18 instruction set, only the `MOVFF` instruction fully specifies the 12-bit address of the source and target registers. This instruction ignores the BSR completely when it executes. All other instructions include only the low-order address as an operand and must use either the BSR or the Access Bank to locate their target registers.

FIGURE 10-4: DATA MEMORY MAP FOR PIC18(L)F6XK40 DEVICES

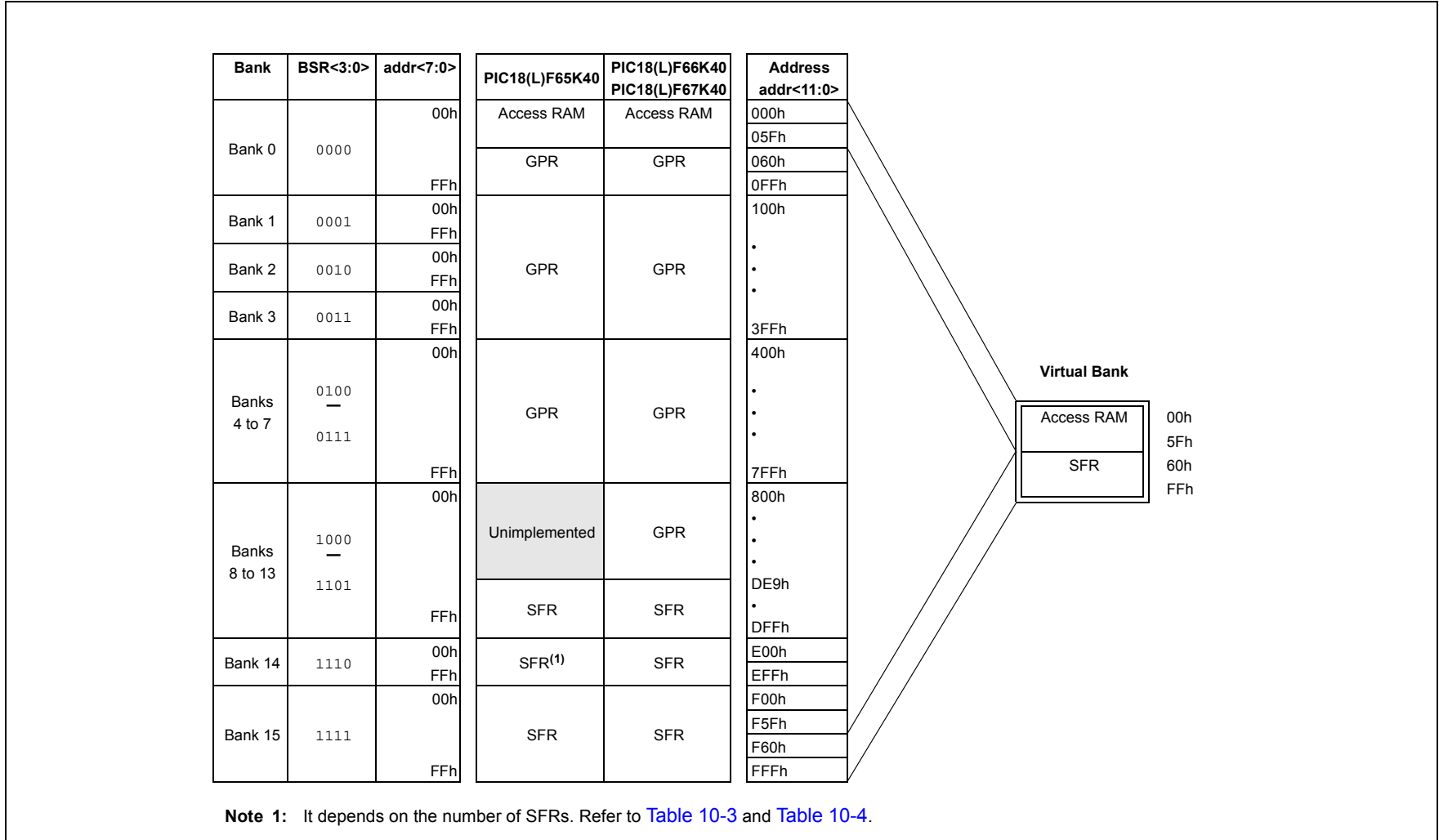
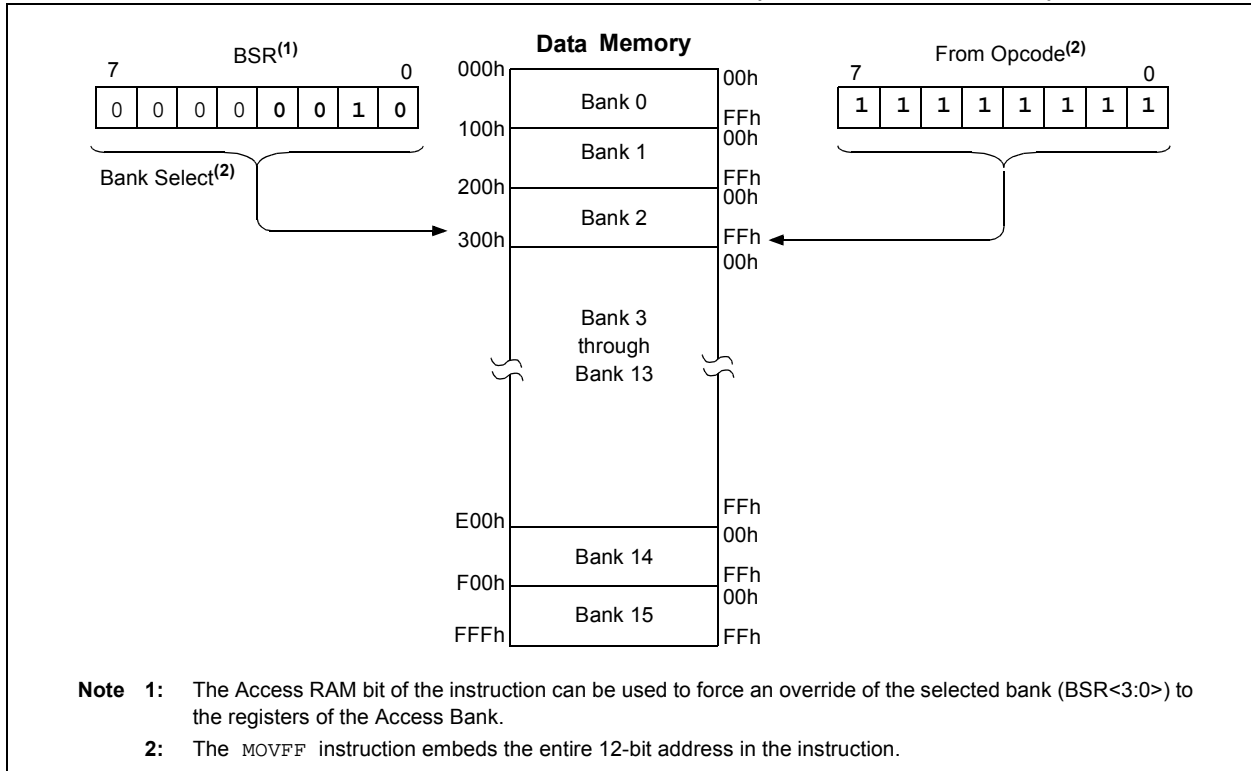


FIGURE 10-5: USE OF THE BANK SELECT REGISTER (DIRECT ADDRESSING)



10.4.2 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. This can be disastrous if a GPR is the intended target of an operation, but an SFR is written to instead. 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 an 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 (00h-5Fh) in Bank 0 and the last 160 bytes of memory (60h-FFh) in Bank 15. The lower half is known as the “Access RAM” and is composed of GPRs. This upper half is also where the device’s SFRs are mapped. These two areas are mapped contiguously in the Access Bank and can be addressed in a linear fashion by an 8-bit address ([Figure 10-4](#)).

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’, however, the instruction is forced to use the Access Bank address map; the current value of the BSR is ignored entirely.

Using this “forced” addressing allows the instruction to operate on a data address in a single cycle, without updating the BSR first. For 8-bit addresses of 60h and above, this means that users can evaluate and operate on SFRs more efficiently. The Access RAM below 60h is a good place for data values that the user might need to access rapidly, such as immediate computational results or common program variables. 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 (XINST Configuration bit = 1). This is discussed in more detail in [Section 10.7.3 “Mapping the Access Bank in Indexed Literal Offset Mode”](#).

10.4.3 GENERAL PURPOSE REGISTER FILE

PIC18 devices may have banked memory in the GPR area. This is data RAM, which is available for use by all instructions. GPRs start at the bottom of Bank 0 (address 000h) and grow upwards towards the bottom of the SFR area. GPRs are not initialized by a Power-on Reset and are unchanged on all other Resets.

10.4.4 SPECIAL FUNCTION REGISTERS

The Special Function Registers (SFRs) are registers used by the CPU and peripheral modules for controlling the desired operation of the device. These registers are implemented as static RAM. SFRs start at the top of data memory (FFFh) and extend downward to occupy the top portion of Bank 15 (F38h to FFFh). A list of these registers is given in [Table 10-3](#) and [Table 10-4](#).

The SFRs can be classified into two sets: those associated with the “core” device functionality (ALU, Resets and interrupts) and those related to the peripheral functions. The Reset and Interrupt registers are described in their respective chapters, while the ALU’s STATUS register is described later in this section. Registers related to the operation of a peripheral feature are described in the chapter for that peripheral.

The SFRs are typically distributed among the peripherals whose functions they control. Unused SFR locations are unimplemented and read as ‘0’s.

PIC18(L)F67K40

TABLE 10-3: SPECIAL FUNCTION REGISTER MAP FOR PIC18(L)F67K40 DEVICES

| Address | Name | Address | Name | Address | Name | Address | Name |
|---------|-------------------------|---------|--------|---------|----------|---------|---------|
| FFFh | TOSU | FD7h | PCON0 | FAFh | TMR6 | F87h | TRISG |
| FFEh | TOSH | FD6h | T0CON1 | FAEh | CCPTMRS1 | F86h | TRISF |
| FFDh | TOSL | FD5h | T0CON0 | FADh | CCPTMRS0 | F85h | TRISE |
| FFCh | STKPTR | FD4h | PR0 | FACH | CCP1CAP | F84h | TRISD |
| FFBh | PCLATU | FD3h | TMR0 | FABh | CCP1CON | F83h | TRISC |
| FFAh | PCLATH | FD2h | T1CLK | FAAh | CCP1H | F82h | TRISB |
| FF9h | PCL | FD1h | T1GATE | FA9h | CCP1L | F81h | TRISA |
| FF8h | TBLPTRU | FD0h | T1GCON | FA8h | CCP2CAP | F80h | LATH |
| FF7h | TBLPTRH | FCFh | T1CON | FA7h | CCP2CON | F7Fh | LATG |
| FF6h | TBLPTRL | FCEh | TMR1H | FA6h | CCP2H | F7Eh | LATF |
| FF5h | TABLAT | FCDh | TMR1L | FA5h | CCP2L | F7Dh | LATE |
| FF4h | PRODH | FCCh | T3CLK | FA4h | PWM6CON | F7Ch | LATD |
| FF3h | PRODL | FCBh | T3GATE | FA3h | PWM6PWH | F7Bh | LATC |
| FF2h | INTCON | FCAh | T3GCON | FA2h | PWM6PWL | F7Ah | LATB |
| FF1h | — | FC9h | T3CON | FA1h | PWM7CON | F79h | LATA |
| FF0h | — | FC8h | TMR3H | FA0h | PWM7PWH | F78h | NVMCON2 |
| FEFh | INDF0 ⁽¹⁾ | FC7h | TMR3L | F9Fh | PWM7PWL | F77h | NVMCON1 |
| FEEh | POSTINC0 ⁽¹⁾ | FC6h | T5CLK | F9Eh | BAUDCON1 | F76h | NVMDAT |
| FEDh | POSTDEC0 ⁽¹⁾ | FC5h | T5GATE | F9Dh | TXSTA1 | F75h | NVMADRH |
| FECh | PREINC0 ⁽¹⁾ | FC4h | T5GCON | F9Ch | RCSTA1 | F74h | NVMADRL |
| FEBh | PLUSW0 ⁽¹⁾ | FC3h | T5CON | F9Bh | SPBRG1H | F73h | ADFLTRH |
| FEAh | FSR0H | FC2h | TMR5H | F9Ah | SPBRG1L | F72h | ADFLTRL |
| FE9h | FSR0L | FC1h | TMR5L | F99h | TXREG1 | F71h | ADACCH |
| FE8h | WREG | FC0h | T2ERS | F98h | RCREG1 | F70h | ADACCL |
| FE7h | INDF1 ⁽¹⁾ | FBFh | T2CLK | F97h | SSP1CON3 | F6Fh | ADERRH |
| FE6h | POSTINC1 ⁽¹⁾ | FBEh | T2HLT | F96h | SSP1CON2 | F6Eh | ADERRL |
| FE5h | POSTDEC1 ⁽¹⁾ | FBDh | T2CON | F95h | SSP1CON1 | F6Dh | ADUTHH |
| FE4h | PREINC1 ⁽¹⁾ | FBCh | T2PR | F94h | SSP1STAT | F6Ch | ADUTHL |
| FE3h | PLUSW1 ⁽¹⁾ | FBBh | T2TMR | F93h | SSP1MASK | F6Bh | ADLTHH |
| FE2h | FSR1H | FBAh | T4ERS | F92h | SSP1ADD | F6Ah | ADLTHL |
| FE1h | FSR1L | FB9h | T4CLK | F91h | SSP1BUF | F69h | ADSTPTH |
| FE0h | BSR | FB8h | T4HLT | F90h | PORTH | F68h | ADSTPTL |
| FDFh | INDF2 ⁽¹⁾ | FB7h | T4CON | F8Fh | PORTG | F67h | ADCNT |
| FDEh | POSTINC2 ⁽¹⁾ | FB6h | T4PR | F8Eh | PORTF | F66h | ADRPT |
| FDDh | POSTDEC2 ⁽¹⁾ | FB5h | T4TMR | F8Dh | PORTE | F65h | ADSTAT |
| FDCh | PREINC2 ⁽¹⁾ | FB4h | T6ERS | F8Ch | PORTD | F64h | ADRESH |
| FDBh | PLUSW2 ⁽¹⁾ | FB3h | T6CLK | F8Bh | PORTC | F63h | ADRESL |
| FDAh | FSR2H | FB2h | T6HLT | F8Ah | PORTB | F62h | ADPREVH |
| FD9h | FSR2L | FB1h | T6CON | F89h | PORTA | F61h | ADPREVL |
| FD8h | STATUS | FB0h | T6PR | F88h | TRISH | F60h | ADCON0 |

Note 1: This is not a physical register.

PIC18(L)F67K40

TABLE 10-4: SPECIAL FUNCTION REGISTER MAP FOR PIC18(L)F67K40 DEVICES

| Address | Name | Address | Name | Address | Name | Address | Name | Address | Name |
|---------|------------|---------|----------|---------|----------|---------|----------|---------|---------|
| F5Fh | ADPCH | F37h | CWG1DAT | F0Fh | SMT1CPRU | EE7h | RCSTA4 | EBFh | ODCONH |
| F5Eh | ADPRE | F36h | CWG1CLK | F0Eh | SMT1CPRH | EE6h | SPBRGH4 | EBEh | SLRCONH |
| F5Dh | ADCAP | F35h | CLKRCLK | F0Dh | SMT1CPRL | EE5h | SPBRGL4 | EBDh | INLV LH |
| F5Ch | ADACQ | F34h | CLKRCON | F0Ch | SMT1TMRU | EE4h | TXREG4 | EBCh | ANSELG |
| F5Bh | ADCON3 | F33h | T7CLK | F0Bh | SMT1TMRH | EE3h | RCREG4 | EBBh | WPUG |
| F5Ah | ADCON2 | F32h | T7GATE | F0Ah | SMT1TMRL | EE2h | BAUDCON5 | EBAh | ODCONG |
| F59h | ADCON1 | F31h | T7GCON | F09h | SMT2WIN | EE1h | TXSTA5 | EB9h | SLRCONG |
| F58h | ADREF | F30h | T7CON | F08h | SMT2SIG | EE0h | RCSTA5 | EB8h | INLV LG |
| F57h | ADCLK | F2Fh | TMR7H | F07h | SMT2CLK | EDFh | SPBRGH5 | EB7h | IOCGP |
| F56h | ADACT | F2Eh | TMR7L | F06h | SMT2STAT | EDEh | SPBRGL5 | EB6h | IOCGN |
| F55h | MDMDCARH | F2Dh | T8ERS | F05h | SMT2CON1 | EDDh | TXREG5 | EB5h | IOCGF |
| F54h | MDCARL | F2Ch | T8CLK | F04h | SMT2CON0 | EDCh | RCREG5 | EB4h | ANSELF |
| F53h | MDSRC | F2Bh | T8HLT | F03h | SMT2PRU | EDBh | SSP2CON3 | EB3h | WPUF |
| F52h | MDCON1 | F2Ah | T8CON | F02h | SMT2PRH | EDAh | SSP2CON2 | EB2h | ODCONF |
| F51h | MDCON0 | F29h | T8PR | F01h | SMT2PRL | ED9h | SSP2CON1 | EB1h | SLRCONF |
| F50h | SCANDTI | F28h | T8TMR | F00h | SMT2CPWU | ED8h | SSP2STAT | EB0h | INLV LF |
| F4Fh | SCANCON0 | F27h | CCP3CAP | EFFh | SMT2CPWH | ED7h | SSP2MASK | EAFh | ANSELE |
| F4Eh | SCANLADDRU | F26h | CCP3CON | EFEh | SMT2CPWL | ED6h | SSP2ADD | EAEh | WPUE |
| F4Dh | SCANLADDRH | F25h | CCP3H | EDFh | SMT2CPRU | ED5h | SSP2BUF | EADh | ODCONE |
| F4Ch | SCANLADDRL | F24h | CCP3L | EFCh | SMT2CPRH | ED4h | CMOUT | EACH | SLRCONE |
| F4Bh | SCANNADDRU | F23h | CCP4CAP | EFBh | SMT2CPRL | ED3h | CM1PCH | EABh | INLV LE |
| F4Ah | SCANNADDRH | F22h | CCP4CON | EFAh | SMT2TMRU | ED2h | CM1NCH | EAAh | IOCEP |
| F49h | SCANNADDRL | F21h | CCP4H | EF9h | SMT2TMRH | ED1h | CM1CON1 | EA9h | IOCEN |
| F48h | CRCCON1 | F20h | CCP4L | EF8h | SMT2TMRL | ED0h | CM1CON0 | EA8h | IOCEF |
| F47h | CRCCON0 | F1Fh | CCP5CAP | EF7h | BAUDCON2 | ECFh | CM2PCH | EA7h | ANSELD |
| F46h | CRCXORH | F1Eh | CCP5CON | EF6h | TXSTA2 | ECEh | CM2NCH | EA6h | WPUD |
| F45h | CRCXORL | F1Dh | CCP5H | EF5h | RCSTA2 | ECDh | CM2CON1 | EA5h | ODCOND |
| F44h | CRCSHFTH | F1Ch | CCP5L | EF4h | SPBRGH2 | ECCh | CM2CON0 | EA4h | SLRCOND |
| F43h | CRCSHFTL | F1Bh | SMT1WIN | EF3h | SPBRGL2 | ECBh | CM3PCH | EA3h | INLV LD |
| F42h | CRCACCH | F1Ah | SMT1SIG | EF2h | TXREG2 | ECAh | CM3NCH | EA2h | ANSELC |
| F41h | CRCACCL | F19h | SMT1CLK | EF1h | RCREG2 | EC9h | CM3CON1 | EA1h | WPUC |
| F40h | CRCDATAH | F18h | SMT1STAT | EF0h | BAUDCON3 | EC8h | CM3CON0 | EA0h | ODCONC |
| F3Fh | CRCDATA | F17h | SMT1CON1 | EEFh | TXSTA3 | EC7h | DAC1CON1 | E9Fh | SLRCONC |
| F3Eh | CWG1STR | F16h | SMT1CON0 | EEEh | RCSTA3 | EC6h | DAC1CON0 | E9Eh | INLV LC |
| F3Dh | CWG1ASD1 | F15h | SMT1PRU | EEDh | SPBRGH3 | EC5h | ZCDCON | E9Dh | IOCCP |
| F3Ch | CWG1ASD0 | F14h | SMT1PRH | EECh | SPBRGL3 | EC4h | FVRCON | E9Ch | IOCCN |
| F3Bh | CWG1CON1 | F13h | SMT1PRL | EEBh | TXREG3 | EC3h | LVDCON1 | E9Bh | IOCCF |
| F3Ah | CWG1CON0 | F12h | SMT1CPWU | EEAh | RCREG3 | EC2h | LVDCON0 | E9Ah | ANSELB |
| F39h | CWG1DBF | F11h | SMT1CPWH | EE9h | BAUDCON4 | EC1h | ANSELH | E99h | WPUB |
| F38h | CWG1DBR | F10h | SMT1CPWL | EE8h | TXSTA4 | EC0h | WPUH | E98h | ODCONB |

PIC18(L)F67K40

TABLE 10-4: SPECIAL FUNCTION REGISTER MAP FOR PIC18(L)F6XK40 DEVICES (CONTINUED)

| Address | Name | Address | Name | Address | Name | Address | Name | Address | Name |
|---------|---------|---------|------------------------|---------|----------|---------|------------|---------|----------|
| E97h | SLRCONB | E6Fh | RD5PPS | E47h | OSCEN | E1Fh | IPR0 | DF7h | T3CKIPPS |
| E96h | INLVLB | E6Eh | RD4PPS | E46h | OSCSTAT1 | E1Eh | SSP2SSPPS | DF6h | T1GPPS |
| E95h | IOCBP | E6Dh | RD3PPS | E45h | OSCCON3 | E1Dh | SSP2DATPPS | DF5h | T1CKIPPS |
| E94h | IOCBN | E6Ch | RD2PPS | E44h | OSCCON2 | E1Ch | SSP2CLKPPS | DF4h | TOCKIPPS |
| E93h | IOCBF | E6Bh | RD1PPS | E43h | OSCCON1 | E1Bh | SSP1SSPPS | DF3h | INT3PPS |
| E92h | ANSELA | E6Ah | RD0PPS | E42h | CPUDOZE | E1Ah | SSP1DATPPS | DF2h | INT2PPS |
| E91h | WPUA | E69h | RC7PPS | E41h | WDTU | E19h | SSP1CLKPPS | DF1h | INT1PPS |
| E90h | ODCONA | E68h | RC6PPS | E40h | WDTH | E18h | TXCK5PPS | DF0h | INT0PPS |
| E8Fh | SLRCONA | E67h | RC5PPS | E3Fh | WDTL | E17h | RXDT5PPS | DEFh | PPSLOCK |
| E8Eh | INLVLA | E66h | RC4PPS | E3Eh | WDTCON1 | E16h | TXCK4PPS | | |
| E8Dh | RH3PPS | E65h | RC3PPS | E3Dh | WDTCON0 | E15h | RXDT4PPS | | |
| E8Ch | RH2PPS | E64h | RC2PPS | E3Ch | PIR9 | E14h | TXCK3PPS | | |
| E8Bh | RH1PPS | E63h | RC1PPS | E3Bh | PIR8 | E13h | RXDT3PPS | | |
| E8Ah | RH0PPS | E62h | RC0PPS | E3Ah | PIR7 | E12h | TXCK2PPS | | |
| E89h | RG7PPS | E61h | RB7PPS | E39h | PIR6 | E11h | RXDT2PPS | | |
| E88h | RG6PPS | E60h | RB6PPS | E38h | PIR5 | E10h | TXCK1PPS | | |
| E87h | — | E5Fh | RB5PPS | E37h | PIR4 | E0Fh | RXDT1PPS | | |
| E86h | RG4PPS | E5Eh | RB4PPS | E36h | PIR3 | E0Eh | MDSRCPPS | | |
| E85h | RG3PPS | E5Dh | RB3PPS | E35h | PIR2 | E0Dh | MDCARHPPS | | |
| E84h | RG2PPS | E5Ch | RB2PPS | E34h | PIR1 | E0Ch | MDCARLPPS | | |
| E83h | RG1PPS | E5Bh | RB1PPS | E33h | PIR0 | E0Bh | CWG1INPPS | | |
| E82h | RG0PPS | E5Ah | RB0PPS | E32h | PIE9 | E0Ah | SMT2SIGPPS | | |
| E81h | RF7PPS | E59h | RA7PPS | E31h | PIE8 | E09h | SMT2WINPPS | | |
| E80h | RF6PPS | E58h | RA6PPS | E30h | PIE7 | E08h | SMT1SIGPPS | | |
| E7Fh | RF5PPS | E57h | RA5PPS | E2Fh | PIE6 | E07h | SMT1WINPPS | | |
| E7Eh | RF4PPS | E56h | RA4PPS | E2Eh | PIE5 | E06h | CCP5PPS | | |
| E7Dh | RF3PPS | E55h | RA3PPS | E2Dh | PIE4 | E05h | CCP4PPS | | |
| E7Ch | RF2PPS | E54h | RA2PPS | E2Ch | PIE3 | E04h | CCP3PPS | | |
| E7Bh | RF1PPS | E53h | RA1PPS | E2Bh | PIE2 | E03h | CCP2PPS | | |
| E7Ah | RF0PPS | E52h | RA0PPS | E2Ah | PIE1 | E02h | CCP1PPS | | |
| E79h | RE7PPS | E51h | PMD5 | E29h | PIE0 | E01h | ADACTPPS | | |
| E78h | RE6PPS | E50h | PMD4 | E28h | IPR9 | E00h | T8INPPS | | |
| E77h | RE5PPS | E4Fh | PMD3 | E27h | IPR8 | DFFh | T6INPPS | | |
| E76h | RE4PPS | E4Eh | PMD2 | E26h | IPR7 | DFEh | T4INPPS | | |
| E75h | RE3PPS | E4Dh | PMD1 | E25h | IPR6 | DFDh | T2INPPS | | |
| E74h | RE2PPS | E4Ch | PMD0 | E24h | IPR5 | DFCh | T7GPPS | | |
| E73h | RE1PPS | E4Bh | BORCON | E23h | IPR4 | DFBh | T7CKIPPS | | |
| E72h | RE0PPS | E4Ah | VREGCON ⁽¹⁾ | E22h | IPR3 | DFAh | T5GPPS | | |
| E71h | RD7PPS | E49h | OSCFREQ | E21h | IPR2 | DF9h | T5CKIPPS | | |

Note 1: Not available on LF parts.

PIC18(L)F67K40

TABLE 10-5: REGISTER FILE SUMMARY FOR PIC18(L)F67K40 DEVICES

| Address | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | |
|---------|----------|--|-----------|--|--------------------------------------|---|---------|---------|---------|-------------------|---------|
| FFFh | TOSU | — | — | — | Top of Stack Upper byte (TOS<20:16>) | | | | | ---xxxxx | |
| FFEh | TOSH | Top of Stack High byte (TOS<15:8>) | | | | | | | | xxxxxxxx | |
| FFDh | TOSL | Top of Stack Low byte (TOS<7:0>) | | | | | | | | xxxxxxxx | |
| FFCh | STKPTR | — | — | — | STKPTR<4:0> | | | | | --00000 | |
| FFBh | PCLATU | — | — | — | Holding Register for PC<20:16> | | | | | ---00000 | |
| FFAh | PCLATH | Holding Register for PC<15:8> | | | | | | | | 0000000 | |
| FF9h | PCL | PC Low byte (PC<7:0>) | | | | | | | | 00000000 | |
| FF8h | TBLPTRU | — | — | Program Memory Table Pointer (TBLPTR<21:16>) | | | | | | | --00000 |
| FF7h | TBLPTRH | Program Memory Table Pointer (TBLPTR<15:8>) | | | | | | | | 0000000 | |
| FF6h | TBLPTRL | Program Memory Table Pointer (TBLPTR<7:0>) | | | | | | | | 00000000 | |
| FF5h | TABLAT | TABLAT | | | | | | | | 0000000 | |
| FF4h | PRODH | Product Register High byte | | | | | | | | xxxxxxxx | |
| FF3h | PRODL | Product Register Low byte | | | | | | | | xxxxxxxx | |
| FF2h | INTCON | GIE/GIEH | PEIE/GIEL | IPEN | — | INT3EDG | INT2EDG | INT1EDG | INT0EDG | 000-1111 | |
| FF1h | — | Unimplemented | | | | | | | | — | |
| FF0h | — | Unimplemented | | | | | | | | — | |
| FEFh | INDF0 | Uses contents of FSR0 to address data memory – value of FSR0 not changed (not a physical register) | | | | | | | | ----- | |
| FEEh | POSTINC0 | Uses contents of FSR0 to address data memory – value of FSR0 post-incremented (not a physical register) | | | | | | | | ----- | |
| FEDh | POSTDEC0 | Uses contents of FSR0 to address data memory – value of FSR0 post-decremented (not a physical register) | | | | | | | | ----- | |
| FECh | PREINC0 | Uses contents of FSR0 to address data memory – value of FSR0 pre-incremented (not a physical register) | | | | | | | | ----- | |
| FEBh | PLUSW0 | Uses contents of FSR0 to address data memory – value of FSR0 pre-incremented (not a physical register) – value of FSR0 offset by W | | | | | | | | ----- | |
| FEAh | FSR0H | — | — | — | — | Indirect Data Memory Address Pointer 0 High | | | | ----xxxx | |
| FE9h | FSR0L | Indirect Data Memory Address Pointer 0 Low | | | | | | | | xxxxxxxx | |
| FE8h | WREG | Working Register | | | | | | | | xxxxxxxx | |
| FE7h | INDF1 | Uses contents of FSR0 to address data memory – value of FSR1 not changed (not a physical register) | | | | | | | | ----- | |
| FE6h | POSTINC1 | Uses contents of FSR0 to address data memory – value of FSR1 post-incremented (not a physical register) | | | | | | | | ----- | |
| FE5h | POSTDEC1 | Uses contents of FSR0 to address data memory – value of FSR1 post-decremented (not a physical register) | | | | | | | | ----- | |
| FE4h | PREINC1 | Uses contents of FSR0 to address data memory – value of FSR1 pre-incremented (not a physical register) | | | | | | | | ----- | |
| FE3h | PLUSW1 | Uses contents of FSR0 to address data memory – value of FSR1 pre-incremented (not a physical register) – value of FSR0 offset by W | | | | | | | | ----- | |

Legend: x = unknown, u = unchanged, — = unimplemented, q = value depends on condition

Note 1: Not available on LF devices.

PIC18(L)F67K40

TABLE 10-5: REGISTER FILE SUMMARY FOR PIC18(L)F67K40 DEVICES (CONTINUED)

| Address | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR |
|---------|----------|--|-----------------|-------------------|-------------------|---|------------------|------------------|------------------|-------------------|
| FE2h | FSR1H | — | — | — | — | Indirect Data Memory Address Pointer 1 High | | | | ----xxxx |
| FE1h | FSR1L | Indirect Data Memory Address Pointer 1 Low | | | | | | | | xxxxxxxx |
| FE0h | BSR | — | — | — | — | Bank Select Register | | | | ----0000 |
| FDFh | INDF2 | Uses contents of FSR0 to address data memory – value of FSR2 not changed (not a physical register) | | | | | | | | ----- |
| FDEh | POSTINC2 | Uses contents of FSR0 to address data memory – value of FSR2 post-incremented (not a physical register) | | | | | | | | ----- |
| FDDh | POSTDEC2 | Uses contents of FSR0 to address data memory – value of FSR2 post-decremented (not a physical register) | | | | | | | | ----- |
| FDCh | PREINC2 | Uses contents of FSR0 to address data memory – value of FSR2 pre-incremented (not a physical register) | | | | | | | | ----- |
| FDBh | PLUSW2 | Uses contents of FSR0 to address data memory – value of FSR2 pre-incremented (not a physical register) – value of FSR0 offset by W | | | | | | | | ----- |
| FDAh | FSR2H | — | — | — | — | Indirect Data Memory Address Pointer 2 High | | | | ----xxxx |
| FD9h | FSR2L | Indirect Data Memory Address Pointer 2 Low | | | | | | | | xxxxxxxx |
| FD8h | STATUS | — | \overline{TO} | \overline{PD} | N | OV | Z | DC | C | -1100000 |
| FD7h | PCON0 | STKOVF | STKUNF | \overline{WDTW} | \overline{RWD} | \overline{RMCLR} | \overline{RI} | \overline{POR} | \overline{BOR} | 0011110q |
| FD6h | T0CON1 | T0CS<2:0> | | | $\overline{T0AS}$ | T0CKPS<3:0> | | | | 00000000 |
| FD5h | T0CON0 | T0EN | — | T0OUT | T016BIT | T0OUTPS<3:0> | | | | 0-000000 |
| FD4h | TMR0H | TMR0H<15:8> | | | | | | | | 11111111 |
| FD3h | TMR0L | TMR0L<7:0> | | | | | | | | 00000000 |
| FD2h | T1CLK | — | — | — | — | CS<3:0> | | | | ----0000 |
| FD1h | T1GATE | — | — | — | — | GSS<3:0> | | | | ----0000 |
| FD0h | T1GCON | GE | GPOL | GTM | GSPM | GO/DONE | GVAL | — | — | 00000x-- |
| FCFh | T1CON | — | — | T1CKPS<1:0> | | — | $\overline{T1S}$ | T1RD16 | TMR1ON | --00-000 |
| FCEh | TMR1H | TMR1H<7:0> | | | | | | | | 00000000 |
| FCDh | TMR1L | TMR1L<7:0> | | | | | | | | 00000000 |
| FCCh | T3CLK | — | — | — | — | CS<3:0> | | | | ----0000 |
| FCBh | T3GATE | — | — | — | — | GSS<3:0> | | | | ----0000 |
| FCAh | T3GCON | GE | GPOL | GTM | GSPM | GO/DONE | GVAL | — | — | 00000x-- |
| FC9h | T3CON | — | — | CKPS<1:0> | | — | \overline{S} | RD16 | ON | --00-000 |
| FC8h | TMR3H | TMR3H<7:0> | | | | | | | | 00000000 |
| FC7h | TMR3L | TMR3L<7:0> | | | | | | | | 00000000 |
| FC6h | T5CLK | — | — | — | — | CS<3:0> | | | | ----0000 |
| FC5h | T5GATE | — | — | — | — | GSS<3:0> | | | | ----0000 |
| FC4h | T5GCON | GE | GPOL | GTM | GSPM | GO/DONE | GVAL | — | — | 00000x-- |
| FC3h | T5CON | — | — | CKPS<1:0> | | — | \overline{S} | RD16 | ON | --00-000 |
| FC2h | TMR5H | TMR5H<7:0> | | | | | | | | 00000000 |
| FC1h | TMR5L | TMR5L<7:0> | | | | | | | | 00000000 |
| FC0h | T2RST | — | — | — | — | RSEL<3:0> | | | | ----0000 |
| FBFh | T2CLKCON | — | — | — | — | CS<3:0> | | | | ----0000 |
| FBEh | T2HLT | \overline{P} | CPOL | \overline{C} | MODE<4:0> | | | | | 00000000 |

Legend: x = unknown, u = unchanged, — = unimplemented, q = value depends on condition
Note 1: Not available on LF devices.

PIC18(L)F67K40

TABLE 10-5: REGISTER FILE SUMMARY FOR PIC18(L)F67K40 DEVICES (CONTINUED)

| Address | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR |
|---------|----------|--|-----------|-------------|-----------|-------------|-------|-------------|----------|-------------------|
| FBDh | T2CON | ON | CKPS<2:0> | | | OUTPS<3:0> | | | | 00000000 |
| FBCh | T2PR | TMR2 Period Register | | | | | | | | 11111111 |
| FBBh | T2TMR | Holding Register for the 8-bit TMR2 Register | | | | | | | | 00000000 |
| FBAh | T4RST | — | — | — | — | RSEL<3:0> | | | | ----0000 |
| FB9h | T4CLKCON | — | — | — | — | CS<3:0> | | | | ----0000 |
| FB8h | T4HLT | PSYNC | CPOL | CSYNC | MODE<4:0> | | | | 00000000 | |
| FB7h | T4CON | ON | CKPS<2:0> | | | OUTPS<3:0> | | | | 00000000 |
| FB6h | T4PR | TMR4 Period Register | | | | | | | | 11111111 |
| FB5h | T4TMR | Holding Register for the 8-bit TMR4 Register | | | | | | | | 00000000 |
| FB4h | T6RST | — | — | — | — | RSEL<3:0> | | | | ----0000 |
| FB3h | T6CLKCON | — | — | — | — | CS<3:0> | | | | ----0000 |
| FB2h | T6HLT | PSYNC | CPOL | CSYNC | MODE<4:0> | | | | 00000000 | |
| FB1h | T6CON | ON | CKPS<2:0> | | | OUTPS<3:0> | | | | 00000000 |
| FB0h | T6PR | TMR6 Period Register | | | | | | | | 11111111 |
| FAFh | T6TMR | Holding Register for the 8-bit TMR6 Register | | | | | | | | 00000000 |
| FAEh | CCPTMRS1 | — | — | P7TSEL<1:0> | | P6TSEL<1:0> | | C5TSEL<1:0> | | --000000 |
| FADh | CCPTMRS0 | C4TSEL<1:0> | | C3TSEL<1:0> | | C2TSEL<1:0> | | C1TSEL<1:0> | | 00000000 |
| FACh | CCP1CAP | — | — | — | — | — | — | CTS<1:0> | | -----00 |
| FABh | CCP1CON | EN | — | OUT | FMT | MODE<3:0> | | | | 0-000000 |
| FAAh | CCPR1H | Capture/Compare/PWM Register 1 (MSB) | | | | | | | | xxxxxxxx |
| FA9h | CCPR1L | Capture/Compare/PWM Register 1 (LSB) | | | | | | | | xxxxxxxx |
| FA8h | CCP2CAP | — | — | — | — | — | — | CTS<1:0> | | -----00 |
| FA7h | CCP2CON | EN | — | OUT | FMT | MODE<3:0> | | | | 0-000000 |
| FA6h | CCPR2H | Capture/Compare/PWM Register 2 (LSB) | | | | | | | | xxxxxxxx |
| FA5h | CCPR2L | Capture/Compare/PWM Register 2 (MSB) | | | | | | | | xxxxxxxx |
| FA4h | PWM6CON | EN | — | OUT | POL | — | — | — | — | 0-00---- |
| FA3h | PWM6DCH | DC<7:0> | | | | | | | | xxxxxxxx |
| FA2h | PWM6DCL | DC<9:8> | | — | — | — | — | — | — | xx----- |
| FA1h | PWM7CON | EN | — | OUT | POL | — | — | — | — | 0-00---- |
| FA0h | PWM7DCH | DC<7:0> | | | | | | | | xxxxxxxx |
| F9Fh | PWM7DCL | DC<9:8> | | — | — | — | — | — | — | xx----- |
| F9Eh | BAUD1CON | ABDOVF | RCIDL | — | SCKP | BRG16 | — | WUE | ABDEN | 01-00-00 |
| F9Dh | TX1STA | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D | 00000010 |
| F9Ch | RC1STA | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 00000000 |
| F9Bh | SP1BRGH | EUSART1 Baud Rate Generator, High Byte | | | | | | | | 00000000 |
| F9Ah | SP1BRGL | EUSART1 Baud Rate Generator, Low Byte | | | | | | | | 00000000 |
| F99h | TX1REG | EUSART1 Transmit Register | | | | | | | | 00000000 |
| F98h | RC1REG | EUSART1 Receive Register | | | | | | | | 00000000 |
| F97h | SSP1CON3 | ACKTIM | PCIE | SCIE | BOEN | SDAHT | SBCDE | AHEN | DHEN | 00000000 |
| F96h | SSP1CON2 | GCEN | ACKSTAT | ACKDT | ACKEN | RCEN | PEN | RSEN | SEN | 00000000 |
| F95h | SSP1CON1 | WCOL | SSPOV | SSPEN | CKP | SSPM<3:0> | | | | 00000000 |

Legend: x = unknown, u = unchanged, — = unimplemented, q = value depends on condition

Note 1: Not available on LF devices.

PIC18(L)F67K40

TABLE 10-5: REGISTER FILE SUMMARY FOR PIC18(L)F67K40 DEVICES (CONTINUED)

| Address | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR |
|---------|----------|---------------|--------|--------|--------|--------|--------|-------------|--------|-------------------|
| F94h | SSP1STAT | SMP | CKE | D/A | P | S | R/W | UA | BF | 00000000 |
| F93h | SSP1MSK | MSK<7:0> | | | | | | | | 11111111 |
| F92h | SSP1ADD | ADD<7:0> | | | | | | | | 00000000 |
| F91h | SSP1BUF | BUF<7:0> | | | | | | | | xxxxxxxx |
| F90h | PORTH | — | — | — | — | RH3 | RH2 | RH1 | RH0 | ----xxxx |
| F8Fh | PORTG | RG7 | RG6 | RG5 | RG4 | RG3 | RG2 | RG1 | RG0 | xxxxxxxx |
| F8Eh | PORTF | RF7 | RF6 | RF5 | RF4 | RF3 | RF2 | RF1 | RF0 | xxxxxxxx |
| F8Dh | PORTE | RE7 | RE6 | RE5 | RE4 | RE3 | RE2 | RE1 | RE0 | xxxxxxxx |
| F8Ch | PORTD | RD7 | RD6 | RD5 | RD4 | RD3 | RD2 | RD1 | RD0 | xxxxxxxx |
| F8Bh | PORTC | RC7 | RC6 | RC5 | RC4 | RC3 | RC2 | RC1 | RC0 | xxxxxxxx |
| F8Ah | PORTB | RB7 | RB6 | RB5 | RB4 | RB3 | RB2 | RB1 | RB0 | xxxxxxxx |
| F89h | PORTA | RA7 | RA6 | RA5 | RA4 | RA3 | RA2 | RA1 | RA0 | xxxxxxxx |
| F88h | TRISH | — | — | — | — | TRISH3 | TRISH2 | TRISH1 | TRISH0 | 11111111 |
| F87h | TRISG | TRISG7 | TRISG6 | — | TRISG4 | TRISG3 | TRISG2 | TRISG1 | TRISG0 | 11-11111 |
| F86h | TRISF | TRISF7 | TRISF6 | TRISF5 | TRISF4 | TRISF3 | TRISF2 | TRISF1 | TRISF0 | 11111111 |
| F85h | TRISE | TRISE7 | TRISE6 | TRISE5 | TRISE4 | TRISE3 | TRISE2 | TRISE1 | TRISE0 | 11111111 |
| F84h | TRISD | TRISD7 | TRISD6 | TRISD5 | TRISD4 | TRISD3 | TRISD2 | TRISD1 | TRISD0 | 11111111 |
| F83h | TRISC | TRISC7 | TRISC6 | TRISC5 | TRISC4 | TRISC3 | TRISC2 | TRISC1 | TRISC0 | 11111111 |
| F82h | TRISB | TRISB7 | TRISB6 | TRISB5 | TRISB4 | TRISB3 | TRISB2 | TRISB1 | TRISB0 | 11111111 |
| F81h | TRISA | TRISA7 | TRISA6 | TRISA5 | TRISA4 | TRISA3 | TRISA2 | TRISA1 | TRISA0 | 11111111 |
| F80h | LATH | — | — | — | — | LATH3 | LATH2 | LATH1 | LATH0 | ----xxxx |
| F7Fh | LATG | LATG7 | LATG6 | — | LATG4 | LATG3 | LATG2 | LATG1 | LATG0 | xx-xxxxx |
| F7Eh | LATF | LATF7 | LATF6 | LATF5 | LATF4 | LATF3 | LATF2 | LATF1 | LATF0 | xxxxxxxx |
| F7Dh | LATE | LATE7 | LATE6 | LATE5 | LATE4 | LATE3 | LATE2 | LATE1 | LATE0 | xxxxxxxx |
| F7Ch | LATD | LATD7 | LATD6 | LATD5 | LATD4 | LATD3 | LATD2 | LATD1 | LATD0 | xxxxxxxx |
| F7Bh | LATC | LATC7 | LATC6 | LATC5 | LATC4 | LATC3 | LATC2 | LATC1 | LATC0 | xxxxxxxx |
| F7Ah | LATB | LATB7 | LATB6 | LATB5 | LATB4 | LATB3 | LATB2 | LATB1 | LATB0 | xxxxxxxx |
| F79h | LATA | LATA7 | LATA6 | LATA5 | LATA4 | LATA3 | LATA2 | LATA1 | LATA0 | xxxxxxxx |
| F78h | NVMCON2 | NVMCON2<7:0> | | | | | | | | 00000000 |
| F77h | NVMCON1 | NVMREG<1:0> | — | FREE | WRERR | WREN | WR | RD | | 00-0x000 |
| F76h | NVMDAT | NVMDAT<7:0> | | | | | | | | 00000000 |
| F75h | NVMADRH | — | — | — | — | — | — | NVMADR<9:8> | | -----xx |
| F74h | NVMADRL | NVMADR<7:0> | | | | | | | | xxxxxxxx |
| F73h | ADFLTRH | ADFLTRH<15:8> | | | | | | | | xxxxxxxx |
| F72h | ADFLTRL | ADFLTRL<7:0> | | | | | | | | xxxxxxxx |
| F71h | ADACCH | ADACCH<15:8> | | | | | | | | xxxxxxxx |
| F70h | ADACCL | ADACCL<7:0> | | | | | | | | xxxxxxxx |
| F6Fh | ADERRH | ADERRH<15:8> | | | | | | | | 00000000 |
| F6Eh | ADERRL | ADERRL<7:0> | | | | | | | | 00000000 |
| F6Dh | ADUTHH | ADUTHH<15:8> | | | | | | | | 00000000 |
| F6Ch | ADUTHL | ADUTHL<7:0> | | | | | | | | 00000000 |

Legend: x = unknown, u = unchanged, — = unimplemented, q = value depends on condition
Note 1: Not available on LF devices.

PIC18(L)F67K40

TABLE 10-5: REGISTER FILE SUMMARY FOR PIC18(L)F67K40 DEVICES (CONTINUED)

| Address | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | |
|---------|-----------|---------------|-------------|-------------|------------|-----------|-------------|-------------|---------|-------------------|-----------|
| F6Bh | ADLTHH | ADLTHH<15:8> | | | | | | | | 00000000 | |
| F6Ah | ADLTHL | ADLTHL<7:0> | | | | | | | | 00000000 | |
| F69h | ADSTPTH | ADSTPTH<15:8> | | | | | | | | 00000000 | |
| F68h | ADSTPTL | ADSTPTL<7:0> | | | | | | | | 00000000 | |
| F67h | ADCNT | ADCNT<7:0> | | | | | | | | 00000000 | |
| F66h | ADRPT | ADRPT<7:0> | | | | | | | | 00000000 | |
| F65h | ADSTAT | ADAOV | ADUTHR | ADLTHR | ADMATH | — | ADSTAT<2:0> | | | 0000-000 | |
| F64h | ADRESH | ADRESH<7:0> | | | | | | | | 00000000 | |
| F63h | ADRESL | ADRESL<7:0> | | | | | | | | 00000000 | |
| F62h | ADPREVH | ADPREVH<15:8> | | | | | | | | 00000000 | |
| F61h | ADPREVL | ADPREVL<7:0> | | | | | | | | 00000000 | |
| F60h | ADCON0 | ADON | ADCONT | — | ADSC | — | ADFM | — | ADGO | 00-0-0-0 | |
| F5Fh | ADPCH | — | — | ADPCH<5:0> | | | | | — | 000000 | |
| F5Eh | ADPRE | ADPRE<7:0> | | | | | | | | 00000000 | |
| F5Dh | ADCAP | — | — | — | ADCAP<4:0> | | | | | ---00000 | |
| F5Ch | ADACQ | ADACQ<7:0> | | | | | | | | 00000000 | |
| F5Bh | ADCON3 | — | ADCALC<2:0> | | | ADSOI | ADTMD<2:0> | | | -0000000 | |
| F5Ah | ADCON2 | ADPSIS | ADCRS<2:0> | | | ADACL | ADMMD<2:0> | | | 00000000 | |
| F59h | ADCON1 | ADPPOL | ADIPEN | ADGPOL | — | — | — | — | ADDSSEN | 000----0 | |
| F58h | ADREF | — | — | — | ADNREF | — | — | ADPREF<1:0> | | ---0--00 | |
| F57h | ADCLK | — | — | ADCS<5:0> | | | | | — | --000000 | |
| F56h | ADACT | — | — | — | ADACT<4:0> | | | | | ---00000 | |
| F55h | MDCARH | — | — | — | — | CHS<3:0> | | | | ----0000 | |
| F54h | MDCARL | — | — | — | — | CLS<3:0> | | | | ----0000 | |
| F53h | MDSRC | — | — | — | SRCS<4:0> | | | | | ---00000 | |
| F52h | MDCON1 | — | — | CHPOL | CHSYNC | — | — | CLPOL | CLSYNC | --00--00 | |
| F51h | MDCON0 | EN | — | OUT | OPOL | — | — | — | BIT | 0-00---0 | |
| F50h | SCANTRIG | — | — | — | — | TSEL<3:0> | | | | ----0000 | |
| F4Fh | SCANCON0 | EN | GO | BUSY | INVALID | INTM | — | MODE<1:0> | | 00000-00 | |
| F4Eh | SCANHADRU | — | — | HADR<21:16> | | | | | | — | --111111 |
| F4Dh | SCANHADRH | HADR<15:8> | | | | | | | | 11111111 | |
| F4Ch | SCANHADRL | HADR<7:0> | | | | | | | | 11111111 | |
| F4Bh | SCANLADRU | — | — | LADR<21:16> | | | | | | — | --000000 |
| F4Ah | SCANLADRH | LADR<15:8> | | | | | | | | 00000000 | |
| F49h | SCANLADRL | LADR<7:0> | | | | | | | | 00000000 | |
| F48h | CRCCON1 | DLEN<3:0> | | | | PLEN<3:0> | | | | 00000000 | |
| F47h | CRCCON0 | EN | GO | BUSY | ACCM | — | — | SHIFTM | FULL | 0000--00 | |
| F46h | CRCXORH | X<15:8> | | | | | | | | xxxxxxxx | |
| F45h | CRCXORL | X<7:1> | | | | | | | | — | xxxxxxxx0 |
| F44h | CRCSHIFTH | SHIFT<15:8> | | | | | | | | 00000000 | |
| F43h | CRCSHIFTL | SHIFT<7:0> | | | | | | | | 00000000 | |

Legend: x = unknown, u = unchanged, — = unimplemented, q = value depends on condition

Note 1: Not available on LF devices.

PIC18(L)F67K40

TABLE 10-5: REGISTER FILE SUMMARY FOR PIC18(L)F67K40 DEVICES (CONTINUED)

| Address | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | |
|---------|------------|--|-----------|-----------|-------------|------------|---------------|----------|-------|-------------------|----------|
| F42h | CRCACCH | ACC<15:8> | | | | | | | | 00000000 | |
| F41h | CRCACCL | ACC<7:0> | | | | | | | | 00000000 | |
| F40h | CRCDATH | DATA<15:8> | | | | | | | | xxxxxxxx | |
| F3Fh | CRCDATL | DATA<7:0> | | | | | | | | xxxxxxxx | |
| F3Eh | CWG1STR | OVRD | OVRC | OVRB | OVRA | STRD | STRC | STRB | STRA | 00000000 | |
| F3Dh | CWG1AS1 | AS7E | AS6E | AS5E | AS4E | AS3E | AS2E | AS1E | AS0E | 00000000 | |
| F3Ch | CWG1AS0 | SHUTDOWN | REN | LSBD<1:0> | | LSAC<1:0> | | — | — | 000101-- | |
| F3Bh | CWG1CON1 | — | — | IN | — | POLD | POLC | POLB | POLA | --x-0000 | |
| F3Ah | CWG1CON0 | EN | LD | — | — | — | MODE<2:0> | | | 00---000 | |
| F39h | CWG1DBF | — | — | DBF<5:0> | | | | | — | — | --000000 |
| F38h | CWG1DBR | — | — | DBR<5:0> | | | | | — | — | --000000 |
| F37h | CWG1ISM | — | — | — | — | ISM<3:0> | | | | ----0000 | |
| F36h | CWG1CLKCON | — | — | — | — | — | — | — | CS | -----0 | |
| F35h | CLKRCLK | — | — | — | — | — | CLKRxCLK<2:0> | | | -----000 | |
| F34h | CLKRCON | EN | — | — | CLKRDC<1:0> | | CLKRDIV<2:0> | | | 0--10000 | |
| F33h | T7CLK | — | — | — | — | CS<3:0> | | | | ----0000 | |
| F32h | T7GATE | — | — | — | — | GSS<3:0> | | | | ----0000 | |
| F31h | T7GCON | GE | GPOL | GTM | GSPM | GO/DONE | GVAL | — | — | 00000x-- | |
| F30h | T7CON | — | — | CKPS<1:0> | | — | SYNC | RD16 | ON | --00-000 | |
| F2Fh | TMR7H | TMR5H<7:0> | | | | | | | | 00000000 | |
| F2Eh | TMR7L | TMR5L<7:0> | | | | | | | | 00000000 | |
| F2Dh | T8RST | — | — | — | — | RSEL<3:0> | | | | ----0000 | |
| F2Ch | T8CLKCON | — | — | — | — | CS<3:0> | | | | ----0000 | |
| F2Bh | T8HLT | PSYNC | CPOL | CSYNC | MODE<4:0> | | | | | 00000000 | |
| F2Ah | T8CON | ON | CKPS<2:0> | | | OUTPS<3:0> | | | | 00000000 | |
| F29h | T8PR | TMR2 Period Register | | | | | | | | 11111111 | |
| F28h | T8TMR | Holding Register for the 8-bit TMR2 Register | | | | | | | | 00000000 | |
| F27h | CCP3CAP | — | — | — | — | — | — | CTS<1:0> | | -----00 | |
| F26h | CCP3CON | EN | — | OUT | FMT | MODE<3:0> | | | | 0-000000 | |
| F25h | CCP3H | Capture/Compare/PWM Register 3 (MSB) | | | | | | | | xxxxxxxx | |
| F24h | CCP3L | Capture/Compare/PWM Register 3 (LSB) | | | | | | | | xxxxxxxx | |
| F23h | CCP4CAP | — | — | — | — | — | — | CTS<1:0> | | -----00 | |
| F22h | CCP4CON | EN | — | OUT | FMT | MODE<3:0> | | | | 0-000000 | |
| F21h | CCPR4H | Capture/Compare/PWM Register 4 (MSB) | | | | | | | | xxxxxxxx | |
| F20h | CCPR4L | Capture/Compare/PWM Register 4 (LSB) | | | | | | | | xxxxxxxx | |
| F1Fh | CCP5CAP | — | — | — | — | — | — | CTS<1:0> | | -----00 | |
| F1Eh | CCP5CON | EN | — | OUT | FMT | MODE<3:0> | | | | 0-000000 | |
| F1Dh | CCPR5H | Capture/Compare/PWM Register 5 (MSB) | | | | | | | | xxxxxxxx | |
| F1Ch | CCPR5L | Capture/Compare/PWM Register 5 (LSB) | | | | | | | | xxxxxxxx | |
| F1Bh | SMT1WIN | — | — | — | WSEL<4:0> | | | | | --00000 | |
| F1Ah | SMT1SIG | — | — | — | SSEL<4:0> | | | | | --00000 | |

Legend: x = unknown, u = unchanged, — = unimplemented, q = value depends on condition

Note 1: Not available on LF devices.

PIC18(L)F67K40

TABLE 10-5: REGISTER FILE SUMMARY FOR PIC18(L)F67K40 DEVICES (CONTINUED)

| Address | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR |
|---------|----------|--|--------|-------|-----------|-----------|-----------|-------------|----------|-------------------|
| F19h | SMT1CLK | — | — | — | — | — | CSEL<2:0> | | | ----000 |
| F18h | SMT1STAT | CPRUP | CPWUP | RST | — | — | TS | WS | AS | 000--000 |
| F17h | SMT1CON1 | SMT2GO | REPEAT | — | — | MODE<3:0> | | | 00--0000 | |
| F16h | SMT1CON0 | EN | — | STP | WPOL | SPOL | CPOL | SMT2PS<1:0> | | 0-000000 |
| F15h | SMT1PRU | SMT1PR<23:16> | | | | | | | | 11111111 |
| F14h | SMT1PRH | SMT1PR<15:8> | | | | | | | | 11111111 |
| F13h | SMT1PRL | SMT1PR<7:0> | | | | | | | | 11111111 |
| F12h | SMT1CPWU | SMT1CPW<23:16> | | | | | | | | xxxxxxxx |
| F11h | SMT1CPWH | SMT1CPW<15:8> | | | | | | | | xxxxxxxx |
| F10h | SMT1CPWL | SMT1CPW<7:0> | | | | | | | | xxxxxxxx |
| F0Fh | SMT1CPRU | SMT1CPR<23:16> | | | | | | | | xxxxxxxx |
| F0Eh | SMT1CPRH | SMT1CPR<15:8> | | | | | | | | xxxxxxxx |
| F0Dh | SMT1CPRL | SMT1CPR<7:0> | | | | | | | | xxxxxxxx |
| F0Ch | SMT1TMRU | SMT1TMR<23:16> | | | | | | | | 00000000 |
| F0Bh | SMT1TMRH | SMT1TMR<15:8> | | | | | | | | 00000000 |
| F0Ah | SMT1TMRL | SMT1TMR<7:0> | | | | | | | | 00000000 |
| F09h | SMT2WIN | — | — | — | WSEL<4:0> | | | --00000 | | |
| F08h | SMT2SIG | — | — | — | SSEL<4:0> | | | --00000 | | |
| F07h | SMT2CLK | — | — | — | — | — | CSEL<2:0> | | | ----000 |
| F06h | SMT2STAT | CPRUP | CPWUP | RST | — | — | TS | WS | AS | 000--000 |
| F05h | SMT2CON1 | SMT2GO | REPEAT | — | — | MODE<3:0> | | | 00--0000 | |
| F04h | SMT2CON0 | EN | — | STP | WPOL | SPOL | CPOL | SMT2PS<1:0> | | 0-000000 |
| F03h | SMT2PRU | SMT2PR<23:16> | | | | | | | | 11111111 |
| F02h | SMT2PRH | SMT2PR<15:8> | | | | | | | | 11111111 |
| F01h | SMT2PRL | SMT2PR<7:0> | | | | | | | | 11111111 |
| F00h | SMT2CPWU | SMT2CPW<23:16> | | | | | | | | xxxxxxxx |
| EFh | SMT2CPWH | SMT2CPW<15:8> | | | | | | | | xxxxxxxx |
| EFEh | SMT2CPWL | SMT2CPW<7:0> | | | | | | | | xxxxxxxx |
| EFDh | SMT2CPRU | SMT2CPR<23:16> | | | | | | | | xxxxxxxx |
| EFCh | SMT2CPRH | SMT2CPR<15:8> | | | | | | | | xxxxxxxx |
| EFBh | SMT2CPRL | SMT2CPR<7:0> | | | | | | | | xxxxxxxx |
| EFAh | SMT2TMRU | SMT2TMR<23:16> | | | | | | | | 00000000 |
| EF9h | SMT2TMRH | SMT2TMR<15:8> | | | | | | | | 00000000 |
| EF8h | SMT2TMRL | SMT2TMR<7:0> | | | | | | | | 00000000 |
| EF7h | BAUD2CON | ABDOVF | RCIDL | — | SCKP | BRG16 | — | WUE | ABDEN | 01-00-00 |
| EF6h | TX2STA | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D | 00000010 |
| EF5h | RC2STA | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 00000000 |
| EF4h | SP2BRGH | EUSART2 Baud Rate Generator, High Byte | | | | | | | | 00000000 |
| EF3h | SP2BRGL | EUSART2 Baud Rate Generator, Low Byte | | | | | | | | 00000000 |
| EF2h | TX2REG | EUSART2 Transmit Register | | | | | | | | 00000000 |
| EF1h | RC2REG | EUSART2 Receive Register | | | | | | | | 00000000 |

Legend: x = unknown, u = unchanged, — = unimplemented, q = value depends on condition

Note 1: Not available on LF devices.

PIC18(L)F67K40

TABLE 10-5: REGISTER FILE SUMMARY FOR PIC18(L)F67K40 DEVICES (CONTINUED)

| Address | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR |
|---------|----------|--|---------|-------|-------|-----------|----------|--------|--------|-------------------|
| EF0h | BAUD3CON | ABDOVF | RCIDL | — | SCKP | BRG16 | — | WUE | ABDEN | 01-00-00 |
| EEFh | TX3STA | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D | 00000010 |
| EEEh | RC3STA | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 00000000 |
| EEDh | SP3BRGH | EUSART3 Baud Rate Generator, High Byte | | | | | | | | 00000000 |
| EECh | SP3BRGL | EUSART3 Baud Rate Generator, Low Byte | | | | | | | | 00000000 |
| EEBh | TX3REG | EUSART3 Transmit Register | | | | | | | | 00000000 |
| EEAh | RC3REG | EUSART3 Receive Register | | | | | | | | 00000000 |
| EE9h | BAUD4CON | ABDOVF | RCIDL | — | SCKP | BRG16 | — | WUE | ABDEN | 01-00-00 |
| EE8h | TX4STA | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D | 00000010 |
| EE7h | RC4STA | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 00000000 |
| EE6h | SP4BRGH | EUSART4 Baud Rate Generator, High Byte | | | | | | | | 00000000 |
| EE5h | SP4BRGL | EUSART4 Baud Rate Generator, Low Byte | | | | | | | | 00000000 |
| EE4h | TX4REG | EUSART4 Transmit Register | | | | | | | | 00000000 |
| EE3h | RC4REG | EUSART4 Receive Register | | | | | | | | 00000000 |
| EE2h | BAUD5CON | ABDOVF | RCIDL | — | SCKP | BRG16 | — | WUE | ABDEN | 01-00-00 |
| EE1h | TX5STA | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D | 00000010 |
| EE0h | RC5STA | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 00000000 |
| EDFh | SP5BRGH | EUSART5 Baud Rate Generator, High Byte | | | | | | | | 00000000 |
| EDEh | SP5BRGL | EUSART5 Baud Rate Generator, Low Byte | | | | | | | | 00000000 |
| EDDh | TX5REG | EUSART5 Transmit Register | | | | | | | | 00000000 |
| EDCh | RC5REG | EUSART5 Receive Register | | | | | | | | 00000000 |
| EDBh | SSP2CON3 | ACKTIM | PCIE | SCIE | BOEN | SDAHT | SBCDE | AHEN | DHEN | 00000000 |
| EDAh | SSP2CON2 | GCEN | ACKSTAT | ACKDT | ACKEN | RCEN | PEN | RSEN | SEN | 00000000 |
| ED9h | SSP2CON1 | WCOL | SSPOV | SSPEN | CKP | SSPM<3:0> | | | | 00000000 |
| ED8h | SSP2STAT | SMP | CKE | D/A | P | S | R/W | UA | BF | 00000000 |
| ED7h | SSP2MSK | MSK<7:0> | | | | | | | | 11111111 |
| ED6h | SSP2ADD | ADD<7:0> | | | | | | | | 00000000 |
| ED5h | SSP2BUF | BUF<7:0> | | | | | | | | xxxxxxxxxx |
| ED4h | CMOUT | — | — | — | — | — | MC3OUT | MC2OUT | MC1OUT | -----000 |
| ED3h | CM1PCH | — | — | — | — | — | PCH<2:0> | | | -----000 |
| ED2h | CM1NCH | — | — | — | — | — | NCH<2:0> | | | -----000 |
| ED1h | CM1CON1 | — | — | — | — | — | — | INTP | INTN | -----100 |
| ED0h | CM1CON0 | EN | OUT | — | POL | — | — | HYS | SYNC | 00-0--00 |
| ECFh | CM2PCH | — | — | — | — | — | PCH<2:0> | | | -----000 |
| ECEh | CM2NCH | — | — | — | — | — | NCH<2:0> | | | -----000 |
| ECDh | CM2CON1 | — | — | — | — | — | — | INTP | INTN | -----100 |
| ECCh | CM2CON0 | EN | OUT | — | POL | — | — | HYS | SYNC | 00-0--00 |
| ECBh | CM3PCH | — | — | — | — | — | PCH<2:0> | | | -----000 |
| ECAh | CM3NCH | — | — | — | — | — | NCH<2:0> | | | -----000 |
| EC9h | CM3CON1 | — | — | — | — | — | — | INTP | INTN | -----100 |
| EC8h | CM3CON0 | EN | OUT | — | POL | — | — | HYS | SYNC | 00-0--00 |

Legend: x = unknown, u = unchanged, — = unimplemented, q = value depends on condition
Note 1: Not available on LF devices.

PIC18(L)F67K40

TABLE 10-5: REGISTER FILE SUMMARY FOR PIC18(L)F67K40 DEVICES (CONTINUED)

| Address | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR |
|---------|----------|---------|---------|---------|------------|-------------|---------|------------|---------|-------------------|
| EC7h | DAC1CON1 | — | — | — | DAC1R<4:0> | | | | | ---xxxxx |
| EC6h | DAC1CON0 | EN | — | OE1 | OE2 | PSS<1:0> | | — | NSS | 0-0000-0 |
| EC5h | ZCDCON | ZCDSEN | — | ZCDOUT | ZCDPOL | — | — | ZCDINTP | ZCDINTN | 0-x0--00 |
| EC4h | FVRCON | FVREN | FVRRDY | TSEN | TSRNG | CDAFVR<1:0> | | ADFVR<1:0> | | 0x000000 |
| EC3h | HLVDCON1 | — | — | — | — | SEL<3:0> | | | | ----0000 |
| EC2h | HLVDCON0 | EN | — | OUT | RDY | — | — | INTH | INTL | 0-xx--00 |
| EC1h | ANSELH | — | — | — | — | ANSELH3 | ANSELH2 | ANSELH1 | ANSELH0 | ----1111 |
| EC0h | WPUH | — | — | — | — | WPUH3 | WPUH2 | WPUH1 | WPUH0 | ----1111 |
| EBFh | ODCONH | — | — | — | — | ODCH3 | ODCH2 | ODCH1 | ODCH0 | ----0000 |
| EBEh | SLRCONH | — | — | — | — | SLRH3 | SLRH2 | SLRH1 | SLRH0 | ----1111 |
| EBDh | INLVLH | — | — | — | — | INLVLH3 | INLVLH2 | INLVLH1 | INLVLH0 | ----1111 |
| EBCh | ANSELG | ANSELG7 | ANSELG6 | — | ANSELG4 | ANSELG3 | ANSELG2 | ANSELG1 | ANSELG0 | 11-11111 |
| EBBh | WPUG | WPUG7 | WPUG6 | WPUG5 | WPUG4 | WPUG3 | WPUG2 | WPUG1 | WPUG0 | 1111111 |
| EBAh | ODCONG | ODCONG7 | ODCONG6 | — | ODCONG4 | ODCONG3 | ODCONG2 | ODCONG1 | ODCONG0 | 00-00000 |
| EB9h | SLRCONG | SLRCG7 | SLRCG6 | — | SLRCG4 | SLRCG3 | SLRCG2 | SLRCG1 | SLRCG0 | 11-11111 |
| EB8h | INLVLG | INLVLG7 | INLVLG6 | INLVLG5 | INLVLG4 | INLVLG3 | INLVLG2 | INLVLG1 | INLVLG0 | 1111111 |
| EB7h | IOCGP | — | — | IOCGP5 | — | — | — | — | — | --0----- |
| EB6h | IOCGN | — | — | IOCGN5 | — | — | — | — | — | --0----- |
| EB5h | IOCGF | — | — | IOCGF5 | — | — | — | — | — | --0----- |
| EB4h | ANSELF | ANSELF7 | ANSELF6 | ANSELF5 | ANSELF4 | ANSELF3 | ANSELF2 | ANSELF1 | ANSELF0 | 1111111 |
| EB3h | WPUF | WPUF7 | WPUF6 | WPUF5 | WPUF4 | WPUF3 | WPUF2 | WPUF1 | WPUF0 | 1111111 |
| EB2h | ODCONF | ODCONF7 | ODCONF6 | ODCONF5 | ODCONF4 | ODCONF3 | ODCONF2 | ODCONF1 | ODCONF0 | 0000000 |
| EB1h | SLRCONF | SLRCF7 | SLRCF6 | SLRCF5 | SLRCF4 | SLRCF3 | SLRCF2 | SLRCF1 | SLRCF0 | 1111111 |
| EB0h | INLVLF | INLVLF7 | INLVLF6 | INLVLF5 | INLVLF4 | INLVLF3 | INLVLF2 | INLVLF1 | INLVLF0 | 1111111 |
| EAFh | ANSELE | ANSELE7 | ANSELE6 | ANSELE5 | ANSELE4 | ANSELE3 | ANSELE2 | ANSELE1 | ANSELE0 | 1111111 |
| EAEh | WPUE | WPUE7 | WPUE6 | WPUE5 | WPUE4 | WPUE3 | WPUE2 | WPUE1 | WPUE0 | 1111111 |
| EADh | ODCONE | ODCONE7 | ODCONE6 | ODCONE5 | ODCONE4 | ODCONE3 | ODCONE2 | ODCONE1 | ODCONE0 | 0000000 |
| EACH | SLRCONC | SLRCE7 | SLRCE6 | SLRCE5 | SLRCE4 | SLRCE3 | SLRCE2 | SLRCE1 | SLRCE0 | 1111111 |
| EABh | INLVLE | INLVLE7 | INLVLE6 | INLVLE5 | INLVLE4 | INLVLE3 | INLVLE2 | INLVLE1 | INLVLE0 | 1111111 |
| EAAh | IOCEP | IOCEP7 | IOCEP6 | IOCEP5 | IOCEP4 | IOCEP3 | IOCEP2 | IOCEP1 | IOCEP0 | 0000000 |
| EA9h | IOCEN | IOCEN7 | IOCEN6 | IOCEN5 | IOCEN4 | IOCEN3 | IOCEN2 | IOCEN1 | IOCEN0 | 0000000 |
| EA8h | IOCEF | IOCEF7 | IOCEF6 | IOCEF5 | IOCEF4 | IOCEF3 | IOCEF2 | IOCEF1 | IOCEF0 | 0000000 |
| EA7h | ANSELD | ANSELD7 | ANSELD6 | ANSELD5 | ANSELD4 | ANSELD3 | ANSELD2 | ANSELD1 | ANSELD0 | 1111111 |
| EA6h | WPUD | WPUD7 | WPUD6 | WPUD5 | WPUD4 | WPUD3 | WPUD2 | WPUD1 | WPUD0 | 0000000 |
| EA5h | ODCOND | ODCD7 | ODCD6 | ODCD5 | ODCD4 | ODCD3 | ODCD2 | ODCD1 | ODCD0 | 0000000 |
| EA4h | SLRCOND | SLRD7 | SLRD6 | SLRD5 | SLRD4 | SLRD3 | SLRD2 | SLRD1 | SLRD0 | 1111111 |
| EA3h | INLVLD | INLVLD7 | INLVLD6 | INLVLD5 | INLVLD4 | INLVLD3 | INLVLD2 | INLVLD1 | INLVLD0 | 1000000 |
| EA2h | ANSELC | ANSELC7 | ANSELC6 | ANSELC5 | ANSELC4 | ANSELC3 | ANSELC2 | ANSELC1 | ANSELC0 | 1111111 |
| EA1h | WPUC | WPUC7 | WPUC6 | WPUC5 | WPUC4 | WPUC3 | WPUC2 | WPUC1 | WPUC0 | 0000000 |
| EA0h | ODCONC | ODCC7 | ODCC6 | ODCC5 | ODCC4 | ODCC3 | ODCC2 | ODCC1 | ODCC0 | 0000000 |
| E9Fh | SLRCONC | SLRC7 | SLRC6 | SLRC5 | SLRC4 | SLRC3 | SLRC2 | SLRC1 | SLRC0 | 1111111 |

Legend: x = unknown, u = unchanged, — = unimplemented, □ = value depends on condition
Note 1: Not available on LF devices.

PIC18(L)F67K40

TABLE 10-5: REGISTER FILE SUMMARY FOR PIC18(L)F67K40 DEVICES (CONTINUED)

| Address | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR |
|---------|---------|---------------|---------|-------------|---------|---------|---------|---------|---------|-------------------|
| E9Eh | INLVLC | INLVLC7 | INLVLC6 | INLVLC5 | INLVLC4 | INLVLC3 | INLVLC2 | INLVLC1 | INLVLC0 | 11111111 |
| E9Dh | IOCCP | IOCCP7 | IOCCP6 | IOCCP5 | IOCCP4 | IOCCP3 | IOCCP2 | IOCCP1 | IOCCP0 | 00000000 |
| E9Ch | IOCCN | IOCCN7 | IOCCN6 | IOCCN5 | IOCCN4 | IOCCN3 | IOCCN2 | IOCCN1 | IOCCN0 | 00000000 |
| E9Bh | IOCCF | IOCCF7 | IOCCF6 | IOCCF5 | IOCCF4 | IOCCF3 | IOCCF2 | IOCCF1 | IOCCF0 | 00000000 |
| E9Ah | ANSELB | ANSELB7 | ANSELB6 | ANSELB5 | ANSELB4 | ANSELB3 | ANSELB2 | ANSELB1 | ANSELB0 | 11111111 |
| E99h | WPUB | WPUB7 | WPUB6 | WPUB5 | WPUB4 | WPUB3 | WPUB2 | WPUB1 | WPUB0 | 00000000 |
| E98h | ODCONB | ODCB7 | ODCB6 | ODCB5 | ODCB4 | ODCB3 | ODCB2 | ODCB1 | ODCB0 | 00000000 |
| E97h | SLRCONB | SLRB7 | SLRB6 | SLRB5 | SLRB4 | SLRB3 | SLRB2 | SLRB1 | SLRB0 | 11111111 |
| E96h | INVLVB | INVLVB7 | INVLVB6 | INVLVB5 | INVLVB4 | INVLVB3 | INVLVB2 | INVLVB1 | INVLVB0 | 11111111 |
| E95h | IOCBP | IOCBP7 | IOCBP6 | IOCBP5 | IOCBP4 | IOCBP3 | IOCBP2 | IOCBP1 | IOCBP0 | 00000000 |
| E94h | IOCBN | IOCBN7 | IOCBN6 | IOCBN5 | IOCBN4 | IOCBN3 | IOCBN2 | IOCBN1 | IOCBN0 | 00000000 |
| E93h | IOCBF | IOCBF7 | IOCBF6 | IOCBF5 | IOCBF4 | IOCBF3 | IOCBF2 | IOCBF1 | IOCBF0 | 00000000 |
| E92h | ANSELA | ANSELA7 | ANSELA6 | ANSELA5 | ANSELA4 | ANSELA3 | ANSELA2 | ANSELA1 | ANSELA0 | 11111111 |
| E91h | WPUA | WPUA7 | WPUA6 | WPUA5 | WPUA4 | WPUA3 | WPUA2 | WPUA1 | WPUA0 | 00000000 |
| E90h | ODCONA | ODCA7 | ODCA6 | ODCA5 | ODCA4 | ODCA3 | ODCA2 | ODCA1 | ODCA0 | 00000000 |
| E8Fh | SLRCONA | SLRA7 | SLRA6 | SLRA5 | SLRA4 | SLRA3 | SLRA2 | SLRA1 | SLRA0 | 11111111 |
| E8Eh | INLVLA | INLVLA7 | INLVLA6 | INLVLA5 | INLVLA4 | INLVLA3 | INLVLA2 | INLVLA1 | INLVLA0 | 11111111 |
| E8Dh | RH3PPS | — | — | RH3PPS<5:0> | | | | | | --000000 |
| E8Ch | RH2PPS | — | — | RH2PPS<5:0> | | | | | | --000000 |
| E8Bh | RH1PPS | — | — | RH1PPS<5:0> | | | | | | --000000 |
| E8Ah | RH0PPS | — | — | RH0PPS<5:0> | | | | | | --000000 |
| E89h | RG7PPS | — | — | RG7PPS<5:0> | | | | | | --000000 |
| E88h | RG6PPS | — | — | RG6PPS<5:0> | | | | | | --000000 |
| E87h | — | Unimplemented | | | | | | | | — |
| E86h | RG4PPS | — | — | RG4PPS<5:0> | | | | | | --000000 |
| E85h | RG3PPS | — | — | RG3PPS<5:0> | | | | | | --000000 |
| E84h | RG2PPS | — | — | RG2PPS<5:0> | | | | | | --000000 |
| E83h | RG1PPS | — | — | RG1PPS<5:0> | | | | | | --000000 |
| E82h | RG0PPS | — | — | RG0PPS<5:0> | | | | | | --000000 |
| E81h | RF7PPS | — | — | RF7PPS<5:0> | | | | | | --000000 |
| E80h | RF6PPS | — | — | RF6PPS<5:0> | | | | | | --000000 |
| E7Fh | RF5PPS | — | — | RF5PPS<5:0> | | | | | | --000000 |
| E7Eh | RF4PPS | — | — | RF4PPS<5:0> | | | | | | --000000 |
| E7Dh | RF3PPS | — | — | RF3PPS<5:0> | | | | | | --000000 |
| E7Ch | RF2PPS | — | — | RF2PPS<5:0> | | | | | | --000000 |
| E7Bh | RF1PPS | — | — | RF1PPS<5:0> | | | | | | --000000 |
| E7Ah | RF0PPS | — | — | RF0PPS<5:0> | | | | | | --000000 |
| E79h | RE7PPS | — | — | RE7PPS<5:0> | | | | | | --000000 |
| E78h | RE6PPS | — | — | RE6PPS<5:0> | | | | | | --000000 |
| E77h | RE5PPS | — | — | RE5PPS<5:0> | | | | | | --000000 |
| E76h | RE4PPS | — | — | RE4PPS<5:0> | | | | | | --000000 |

Legend: x = unknown, u = unchanged, — = unimplemented, q = value depends on condition

Note 1: Not available on LF devices.

PIC18(L)F67K40

TABLE 10-5: REGISTER FILE SUMMARY FOR PIC18(L)F67K40 DEVICES (CONTINUED)

| Address | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR |
|---------|--------|--------|---------|-------------|---------|---------|---------|---------|---------|-------------------|
| E75h | RE3PPS | — | — | RE3PPS<5:0> | | | | | | --000000 |
| E74h | RE2PPS | — | — | RE2PPS<5:0> | | | | | | --000000 |
| E73h | RE1PPS | — | — | RE1PPS<5:0> | | | | | | --000000 |
| E72h | RE0PPS | — | — | RE0PPS<5:0> | | | | | | --000000 |
| E71h | RD7PPS | — | — | RD7PPS<5:0> | | | | | | --000000 |
| E70h | RD6PPS | — | — | RD6PPS<5:0> | | | | | | --000000 |
| E6Fh | RD5PPS | — | — | RD5PPS<5:0> | | | | | | --000000 |
| E6Eh | RD4PPS | — | — | RD4PPS<5:0> | | | | | | --000000 |
| E6Dh | RD3PPS | — | — | RD3PPS<5:0> | | | | | | --000000 |
| E6Ch | RD2PPS | — | — | RD2PPS<5:0> | | | | | | --000000 |
| E6Bh | RD1PPS | — | — | RD1PPS<5:0> | | | | | | --000000 |
| E6Ah | RD0PPS | — | — | RD0PPS<5:0> | | | | | | --000000 |
| E69h | RC7PPS | — | — | RC7PPS<5:0> | | | | | | --000000 |
| E68h | RC6PPS | — | — | RC6PPS<5:0> | | | | | | --000000 |
| E67h | RC5PPS | — | — | RC5PPS<5:0> | | | | | | --000000 |
| E66h | RC4PPS | — | — | RC4PPS<5:0> | | | | | | --000000 |
| E65h | RC3PPS | — | — | RC3PPS<5:0> | | | | | | --000000 |
| E64h | RC2PPS | — | — | RC2PPS<5:0> | | | | | | --000000 |
| E63h | RC1PPS | — | — | RC1PPS<5:0> | | | | | | --000000 |
| E62h | RC0PPS | — | — | RC0PPS<5:0> | | | | | | --000000 |
| E61h | RB7PPS | — | — | RB7PPS<5:0> | | | | | | --000000 |
| E60h | RB6PPS | — | — | RB6PPS<5:0> | | | | | | --000000 |
| E5Fh | RB5PPS | — | — | RB5PPS<5:0> | | | | | | --000000 |
| E5Eh | RB4PPS | — | — | RB4PPS<5:0> | | | | | | --000000 |
| E5Dh | RB3PPS | — | — | RB3PPS<5:0> | | | | | | --000000 |
| E5Ch | RB2PPS | — | — | RB2PPS<5:0> | | | | | | --000000 |
| E5Bh | RB1PPS | — | — | RB1PPS<5:0> | | | | | | --000000 |
| E5Ah | RB0PPS | — | — | RB0PPS<5:0> | | | | | | --000000 |
| E59h | RA7PPS | — | — | RA7PPS<5:0> | | | | | | --000000 |
| E58h | RA6PPS | — | — | RA6PPS<5:0> | | | | | | --000000 |
| E57h | RA5PPS | — | — | RA5PPS<5:0> | | | | | | --000000 |
| E56h | RA4PPS | — | — | RA4PPS<5:0> | | | | | | --000000 |
| E55h | RA3PPS | — | — | RA3PPS<5:0> | | | | | | --000000 |
| E54h | RA2PPS | — | — | RA2PPS<5:0> | | | | | | --000000 |
| E53h | RA1PPS | — | — | RA1PPS<5:0> | | | | | | --000000 |
| E52h | RA0PPS | — | — | RA0PPS<5:0> | | | | | | --000000 |
| E51h | PMD5 | — | UART5MD | UART4MD | UART3MD | UART2MD | UART1MD | MSSP2MD | MSSP1MD | -0000000 |
| E50h | PMD4 | — | PWM7MD | PWM6MD | CCP5MD | CCP4MD | CCP3MD | CCP2MD | CCP1MD | -00000000 |
| E4Fh | PMD3 | — | DACMD | ADCMD | — | CMP3MD | CMP2MD | CMP1MD | ZCDMD | -00-0000 |
| E4Eh | PMD2 | — | — | CWGMMD | — | DSMMD | SMT2MD | SMT1MD | TMR8MD | --0-0000 |
| E4Dh | PMD1 | TMR7MD | TMR6MD | TMR5MD | TMR4MD | TMR3MD | TMR2MD | TMR1MD | TMR0MD | 00000000 |

Legend: x = unknown, u = unchanged, — = unimplemented, q = value depends on condition

Note 1: Not available on LF devices.

PIC18(L)F67K40

TABLE 10-5: REGISTER FILE SUMMARY FOR PIC18(L)F67K40 DEVICES (CONTINUED)

| Address | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | |
|---------|------------------------|-------------|------------|------------|-----------|------------|-------------|--------------|----------|-------------------|----------|
| E4Ch | PMD0 | SYSCMD | FVRMD | HLVDM | CRCMD | SCANMD | NVMM | CLKRMD | IOCMD | 00x00000 | |
| E4Bh | BORCON | SBOREN | — | — | — | — | — | — | BORRDY | 1-----q | |
| E4Ah | VREGCON ⁽¹⁾ | — | — | — | — | — | — | VREGPM | Reserved | -----01 | |
| E49h | OSCFRQ | — | — | — | — | HFFRQ<3:0> | | | | ----1111 | |
| E48h | OSCTUNE | — | — | HFTUN<5:0> | | | | | | — | --100000 |
| E47h | OSCCN | EXTOEN | HFOEN | MFOEN | LFOEN | SOSCEN | ADOEN | — | — | 000000-- | |
| E46h | OSCSTAT | EXTOR | HFOR | MFOR | LFOR | SOR | ADOR | — | PLL | qqqqqq-q | |
| E45h | OSCCON3 | CSWHOLD | SOSCPWR | — | ORDY | NOSCR | — | — | — | 00-00--- | |
| E44h | OSCCON2 | — | COSC<2:0> | | | CDIV<3:0> | | | | — | -qqqqqqq |
| E43h | OSCCON1 | — | NOSC<2:0> | | | NDIV<3:0> | | | | — | -qqqqqqq |
| E42h | CPUDOZE | IDLEN | DOZEN | ROI | DOE | — | DOZE<2:0> | | | 0000-000 | |
| E41h | WDTTMR | WDTTMR<4:0> | | | | | STATE | PSCNT<17:16> | | | xxxxx000 |
| E40h | WDTPSH | PSCNT<7:0> | | | | | | | | — | 00000000 |
| E3Fh | WDTPSL | PSCNT<15:8> | | | | | | | | — | 00000000 |
| E3Eh | WDTCON1 | — | WDTCS<2:0> | | | — | WINDOW<2:0> | | | — | -qqq-qqq |
| E3Dh | WDTCON0 | — | — | WDTPS<4:0> | | | | | SEN | — | --qqqqq0 |
| E3Ch | PIR9 | — | — | SMT2PWAIF | SMT2PRAIF | SMT2IF | SMT1PWAIF | SMT1PRAIF | SMT1IF | --000000 | |
| E3Bh | PIR8 | DMAIF | CRCIF | NVMIF | — | — | — | — | CWG1IF | 000----0 | |
| E3Ah | PIR7 | — | — | — | CCP5IF | CCP4IF | CCP3IF | CCP2IF | CCP1IF | ---00000 | |
| E39h | PIR6 | — | — | — | — | TMR7GIF | TMR5GIF | TMR3GIF | TMR1GIF | ----0000 | |
| E38h | PIR5 | TMR8IF | TMR7IF | TMR6IF | TMR5IF | TMR4IF | TMR3IF | TMR2IF | TMR1IF | 00000000 | |
| E37h | PIR4 | — | — | RC5IF | TX5IF | RC4IF | TX4IF | RC3IF | TX3IF | --000000 | |
| E36h | PIR3 | RC2IF | TX2IF | RC1IF | TX1IF | BCL2IF | SSP2IF | BCL1IF | SSP1IF | 00000000 | |
| E35h | PIR2 | HLVDIF | ZCDIF | — | — | — | C3IF | C2IF | C1IF | 00---000 | |
| E34h | PIR1 | OSCFIF | CSWIF | — | — | — | — | ADTIF | ADIF | 00----00 | |
| E33h | PIR0 | — | — | TMR0IF | IOCIF | INT3IF | INT2IF | INT1IF | INT0IF | --000000 | |
| E32h | PIE9 | — | — | SMT2PWAIE | SMT2PRAIE | SMT2IE | SMT1PWAIE | SMT1PRAIE | SMT1IE | --000000 | |
| E31h | PIE8 | DMAIE | CRCIE | NVMIE | — | — | — | — | CWG1IE | 000----0 | |
| E30h | PIE7 | — | — | — | CCP5IE | CCP4IE | CCP3IE | CCP2IE | CCP1IE | ---00000 | |
| E2Fh | PIE6 | — | — | — | — | TMR7GIE | TMR5GIE | TMR3GIE | TMR1GIE | ----0000 | |
| E2Eh | PIE5 | TMR8IE | TMR7IE | TMR6IE | TMR5IE | TMR4IE | TMR3IE | TMR2IE | TMR1IE | 00000000 | |
| E2Dh | PIE4 | — | — | RC5IE | TX5IE | RC4IE | TX4IE | RC3IE | TX3IE | --000000 | |
| E2Ch | PIE3 | RC2IE | TX2IE | RC1IE | TX1IE | BCL2IE | SSP2IE | BCL1IE | SSP1IE | 00000000 | |
| E2Bh | PIE2 | HLVDIE | ZCDIE | — | — | — | C3IE | C2IE | C1IE | 00---000 | |
| E2Ah | PIE1 | OSCFIE | CSWIE | — | — | — | — | ADTIE | ADIE | 00----00 | |
| E29h | PIE0 | — | — | TMR0IE | OCIE | INT3IE | INT2IE | INT1IE | INT0IE | --000000 | |
| E28h | IPR9 | — | — | SMT2PWAIP | SMT2PRAIP | SMT2IP | SMT1PWAIP | SMT1PRAIP | SMT1IP | --111111 | |
| E27h | IPR8 | DMAIP | CRCIP | NVMIP | — | — | — | — | CWG1IP | 111----1 | |
| E26h | IPR7 | — | — | — | CCP5IP | CCP4IP | CCP3IP | CCP2IP | CCP1IP | ---11111 | |
| E25h | IPR6 | — | — | — | — | TMR7GIP | TMR5GIP | TMR3GIP | TMR1GIP | ----1111 | |
| E24h | IPR5 | TMR8IP | TMR7IP | TMR6IP | TMR5IP | TMR4IP | TMR3IP | TMR2IP | TMR1IP | 11111111 | |

Legend: x = unknown, u = unchanged, — = unimplemented, q = value depends on condition

Note 1: Not available on LF devices.

PIC18(L)F67K40

TABLE 10-5: REGISTER FILE SUMMARY FOR PIC18(L)F67K40 DEVICES (CONTINUED)

| Address | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR |
|---------|------------|--------|-------|-----------------|-------|--------|--------|--------|--------|-------------------|
| E23h | IPR4 | — | — | RC5IP | TX5IP | RC4IP | TX4IP | RC3IP | TX3IP | --111111 |
| E22h | IPR3 | RC2IP | TX2IP | RC1IP | TX1IP | BCL2IP | SSP2IP | BCL1IP | SSP1IP | 11111111 |
| E21h | IPR2 | HLVDIP | ZCDIP | — | — | — | C3IP | C2IP | C1IP | 11---111 |
| E20h | IPR1 | OSCFIP | CSWIP | — | — | — | — | ADTIP | ADIP | 11----11 |
| E1Fh | IPR0 | — | — | TMR0IP | IOCIP | INT3IP | INT2IP | INT1IP | INT0IP | 11----11 |
| E1Eh | SSP2SSPPS | — | — | SSP2SSPPS<5:0> | | | | | | --011111 |
| E1Dh | SSP2DATPPS | — | — | SSP2DATPPS<5:0> | | | | | | --011101 |
| E1Ch | SSP2CLKPPS | — | — | SSP2CLKPPS<5:0> | | | | | | --011110 |
| E1Bh | SSP1SSPPS | — | — | SSP1SSPPS<5:0> | | | | | | --101111 |
| E1Ah | SSP1DATPPS | — | — | SSP1DATPPS<5:0> | | | | | | --010100 |
| E19h | SSP1CLKPPS | — | — | SSP1CLKPPS<5:0> | | | | | | --0101011 |
| E18h | TX5PPS | — | — | TX5PPS<5:0> | | | | | | --100010 |
| E17h | RX5PPS | — | — | RX5PPS<5:0> | | | | | | --100011 |
| E16h | TX4PPS | — | — | TX4PPS<5:0> | | | | | | --010000 |
| E15h | RX4PPS | — | — | RX4PPS<5:0> | | | | | | --010001 |
| E14h | TX3PPS | — | — | TX3PPS<5:0> | | | | | | --100000 |
| E13h | RX3PPS | — | — | RX3PPS<5:0> | | | | | | --100001 |
| E12h | TX2PPS | — | — | TX2PPS<5:0> | | | | | | --110001 |
| E11h | RX2PPS | — | — | RX2PPS<5:0> | | | | | | --110010 |
| E10h | TX1PPS | — | — | TX1PPS<5:0> | | | | | | --010110 |
| E0Fh | RX1PPS | — | — | RX1PPS<5:0> | | | | | | --010111 |
| E0Eh | MDSRCPPS | — | — | MDSRCPPS<5:0> | | | | | | --011101 |
| E0Dh | MDCARHPPS | — | — | MDCARHPPS<5:0> | | | | | | --011100 |
| E0Ch | MDCARLPPS | — | — | MDCARLPPS<5:0> | | | | | | --011011 |
| E0Bh | CWGINPPS | — | — | CWGINPPS<5:0> | | | | | | --010010 |
| E0Ah | SMT2SIGPPS | — | — | SMT2SIGPPS<5:0> | | | | | | --110111 |
| E09h | SMT2WINPPS | — | — | SMT2WINPPS<5:0> | | | | | | --110110 |
| E08h | SMT1SIGPPS | — | — | SMT1SIGPPS<5:0> | | | | | | --100111 |
| E07h | SMT1WINPPS | — | — | SMT1WINPPS<5:0> | | | | | | --100110 |
| E06h | CCP5PPS | — | — | CCP5PPS<5:0> | | | | | | --110100 |
| E05h | CCP4PPS | — | — | CCP4PPS<5:0> | | | | | | --110011 |
| E04h | CCP3PPS | — | — | CCP3PPS<5:0> | | | | | | --100110 |
| E03h | CCP2PPS | — | — | CCP2PPS<5:0> | | | | | | --100100 |
| E02h | CCP1PPS | — | — | CCP1PPS<5:0> | | | | | | --100101 |
| E01h | ADACTPPS | — | — | ADACTPPS<5:0> | | | | | | --111001 |
| E00h | T8INPPS | — | — | T8INPPS<5:0> | | | | | | --000000 |
| DFFh | T6INPPS | — | — | T6INPPS<5:0> | | | | | | --010001 |
| DFEh | T4INPPS | — | — | T4INPPS<5:0> | | | | | | --100100 |
| DFDh | T2INPPS | — | — | T2INPPS<5:0> | | | | | | --000001 |
| DFCh | T7GPPS | — | — | T7GPPS<5:0> | | | | | | --011001 |
| DFBh | T7CKIPPS | — | — | T7CKIPPS<5:0> | | | | | | --110100 |

Legend: x = unknown, u = unchanged, — = unimplemented, q = value depends on condition

Note 1: Not available on LF devices.

PIC18(L)F67K40

TABLE 10-5: REGISTER FILE SUMMARY FOR PIC18(L)F67K40 DEVICES (CONTINUED)

| Address | Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR |
|--------------|----------|---------------|-------|---------------|-------|-------|-------|-------|-----------|-------------------|
| DFAh | T5GPPS | — | — | T5GPPS<5:0> | | | | | | --110100 |
| DF9h | T5CKIPPS | — | — | T5CKIPPS<5:0> | | | | | | --011001 |
| DF8h | T3GPPS | — | — | T3GPPS<5:0> | | | | | | --000101 |
| DF7h | T3CKIPPS | — | — | T3CKIPPS<5:0> | | | | | | --001101 |
| DF6h | T1GPPS | — | — | T1GPPS<5:0> | | | | | | --001101 |
| DF5h | T1CKIPPS | — | — | T1CKIPPS<5:0> | | | | | | --010000 |
| DF4h | T0CKIPPS | — | — | T0CKIPPS<5:0> | | | | | | --000100 |
| DF3h | INT3PPS | — | — | INT3PPS<5:0> | | | | | | --000100 |
| DF2h | INT2PPS | — | — | INT2PPS<5:0> | | | | | | --001010 |
| DF1h | INT1PPS | — | — | INT1PPS<5:0> | | | | | | --001001 |
| DF0h | INT0PPS | — | — | INT0PPS<5:0> | | | | | | --001000 |
| DE0h | PPSLOCK | — | — | — | — | — | — | — | PPSLOCKED | -----0 |
| DD0h to E7Eh | — | Unimplemented | | | | | | | | — |

Legend: x = unknown, u = unchanged, — = unimplemented, α = value depends on condition

Note 1: Not available on LF devices.

10.4.5 STATUS REGISTER

The STATUS register, shown in [Register 10-2](#), 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 in [Section 36.0 "Instruction Set Summary"](#) and [Table 36-3](#).

| |
|--|
| <p>Note: The C and DC bits operate as the borrow and digit borrow bits, respectively, in subtraction.</p> |
|--|

10.5 Register Definitions: Status

REGISTER 10-2: STATUS: STATUS REGISTER

| | | | | | | | |
|-------|------------------------|------------------------|---------|---------|---------|---------|---------|
| U-0 | R-1/q | R-1/q | R/W-0/u | R/W-0/u | R/W-0/u | R/W-0/u | R/W-0/u |
| — | $\overline{\text{TO}}$ | $\overline{\text{PD}}$ | N | OV | Z | DC | C |
| bit 7 | | | | | | | bit 0 |

Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7 **Unimplemented:** Read as '0'

bit 6 **$\overline{\text{TO}}$:** Time-Out bit

1 = Set at power-up or by execution of CLRWDT or SLEEP instruction

0 = A WDT time-out occurred

bit 5 **$\overline{\text{PD}}$:** Power-Down bit

1 = Set at power-up or by execution of CLRWDT instruction

0 = Set by execution of the SLEEP instruction

bit 4 **N:** Negative bit used for signed arithmetic (2's complement); indicates if the result is negative, (ALU MSb = 1).

1 = The result is negative

0 = The result is positive

bit 3 **OV:** Overflow bit used for signed arithmetic (2's complement); indicates an overflow of the 7-bit magnitude, which causes the sign bit (bit 7) to change state.

1 = Overflow occurred for current signed arithmetic operation

0 = No overflow occurred

bit 2 **Z:** Zero bit

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/Borrow bit (ADDWF, ADDLW, SUBLW, SUBWF instructions)⁽¹⁾

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/Borrow bit (ADDWF, ADDLW, SUBLW, SUBWF instructions)^(1,2)

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

Note 1: For Borrow, the polarity is reversed. A subtraction is executed by adding the two's complement of the second operand.

2: For Rotate (RRF, RLF) instructions, this bit is loaded with either the high or low-order bit of the Source register.

10.6 Data Addressing Modes

Note: The execution of some instructions in the core PIC18 instruction set are changed when the PIC18 extended instruction set is enabled. See [Section 10.7 “Data Memory and the Extended Instruction Set”](#) for more information.

While the program memory can be addressed in only one way – through the program counter – 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 [Section 10.7.1 “Indexed Addressing with Literal Offset”](#).

10.6.1 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 20-bit program memory address.

10.6.2 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 ([Section 10.4.3 “General Purpose Register File”](#)) or a location in the Access Bank ([Section 10.4.2 “Access Bank”](#)) 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 ([Section 10.4.1 “Bank Select Register \(BSR\)”](#)) 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. Addressing that uses the Access RAM is sometimes also known as Direct Forced Addressing mode.

A few instructions, such as MOVFF, include the entire 12-bit address (either source or destination) in their opcodes. In these cases, the BSR is ignored entirely.

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. 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.

10.6.3 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 example of clearing an entire RAM bank in [Example 10-5](#).

EXAMPLE 10-5: HOW TO CLEAR RAM (BANK 1) USING INDIRECT ADDRESSING

```

LFSR   FSR0, 100h ;
NEXT   CLRF POSTINCO ; Clear INDF
                                ; register then
                                ; inc pointer
        BTFSS FSR0H, 1 ; All done with
                                ; Bank1?
        BRA   NEXT    ; NO, clear next
CONTINUE                                ; YES, continue
    
```

10.6.3.1 FSR Registers and the INDF Operand

At the core of indirect addressing are three sets of registers: FSR0, FSR1 and FSR2. Each represents a pair of 8-bit registers, FSRnH and FSRnL. Each FSR pair holds a 12-bit value, therefore, the four upper bits of the FSRnH register are not used. The 12-bit 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 12-bit address, data RAM banking is not necessary. Thus, the current contents of the BSR and the Access RAM bit have no effect on determining the target address.

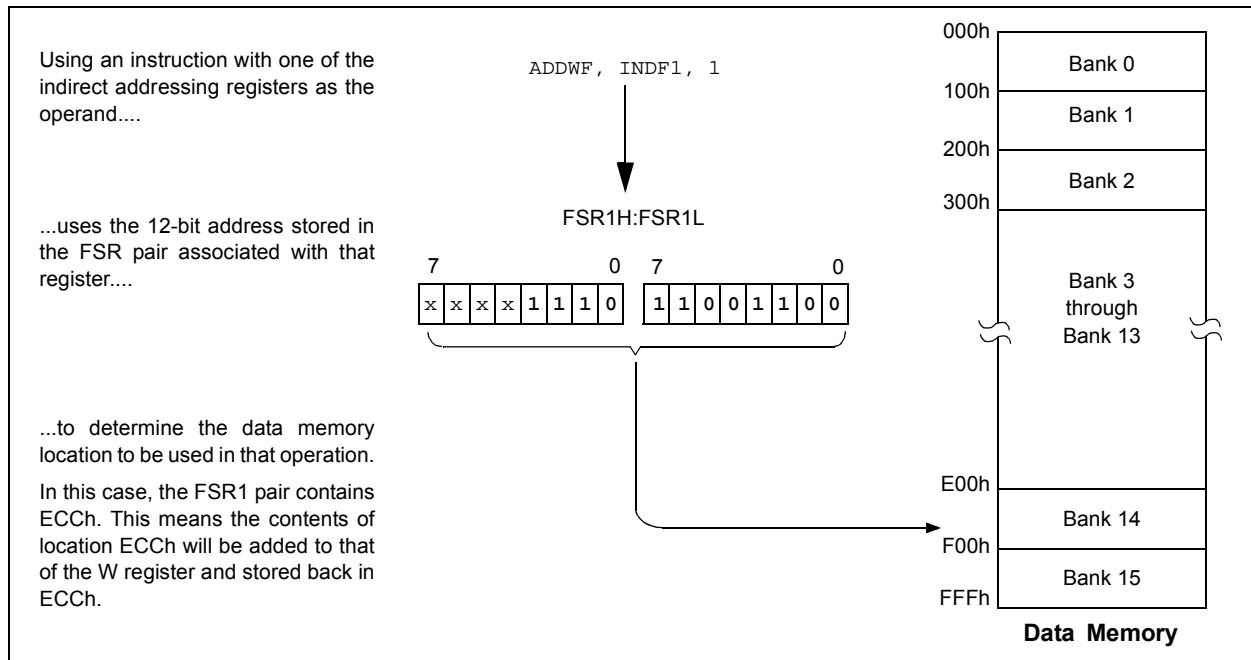
10.6.3.2 FSR Registers and POSTINC, POSTDEC, PREINC and PLUSW

In addition to the INDF operand, each FSR register pair also has four additional indirect operands. Like INDF, these are “virtual” registers which 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 by that 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.

FIGURE 10-6: INDIRECT ADDRESSING



Operations on the FSRs with POSTDEC, POSTINC and PREINC affect the entire register pair; that is, roll-overs 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.

10.6.3.3 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 FE7h, 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 the FSR2H:FSR2L.

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

Similarly, operations by indirect addressing are generally permitted on all other SFRs. Users should exercise the appropriate caution that they do not inadvertently change settings that might affect the operation of the device.

10.7 Data Memory and the Extended Instruction Set

Enabling the PIC18 extended instruction set (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.

10.7.1 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.

10.7.2 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 [Figure 10-7](#).

Those who desire to use byte-oriented or bit-oriented instructions in the Indexed Literal Offset mode should note the changes to assembler syntax for this mode. This is described in more detail in [Section 36.2.1 “Extended Instruction Syntax”](#).

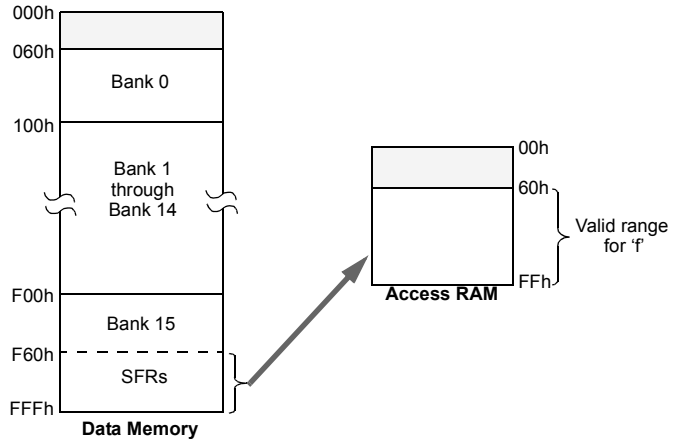
FIGURE 10-7: COMPARING ADDRESSING OPTIONS FOR BIT-ORIENTED AND BYTE-ORIENTED INSTRUCTIONS (EXTENDED INSTRUCTION SET ENABLED)

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

When 'a' = 0 and $f \geq 60h$:

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 F60h to FFFh (Bank 15) of data memory.

Locations below 60h are not available in this addressing mode.

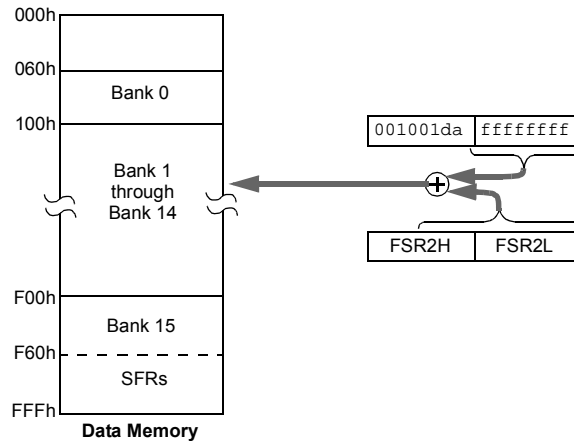


When 'a' = 0 and $f \leq 5Fh$:

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.

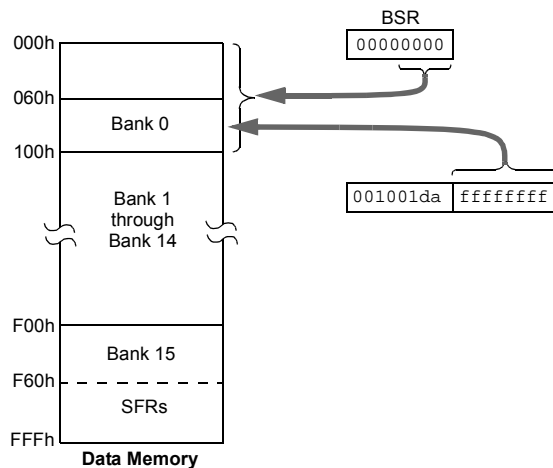
Note that in this mode, the correct syntax is now:

`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 16 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.



10.7.3 MAPPING THE ACCESS BANK IN INDEXED LITERAL OFFSET MODE

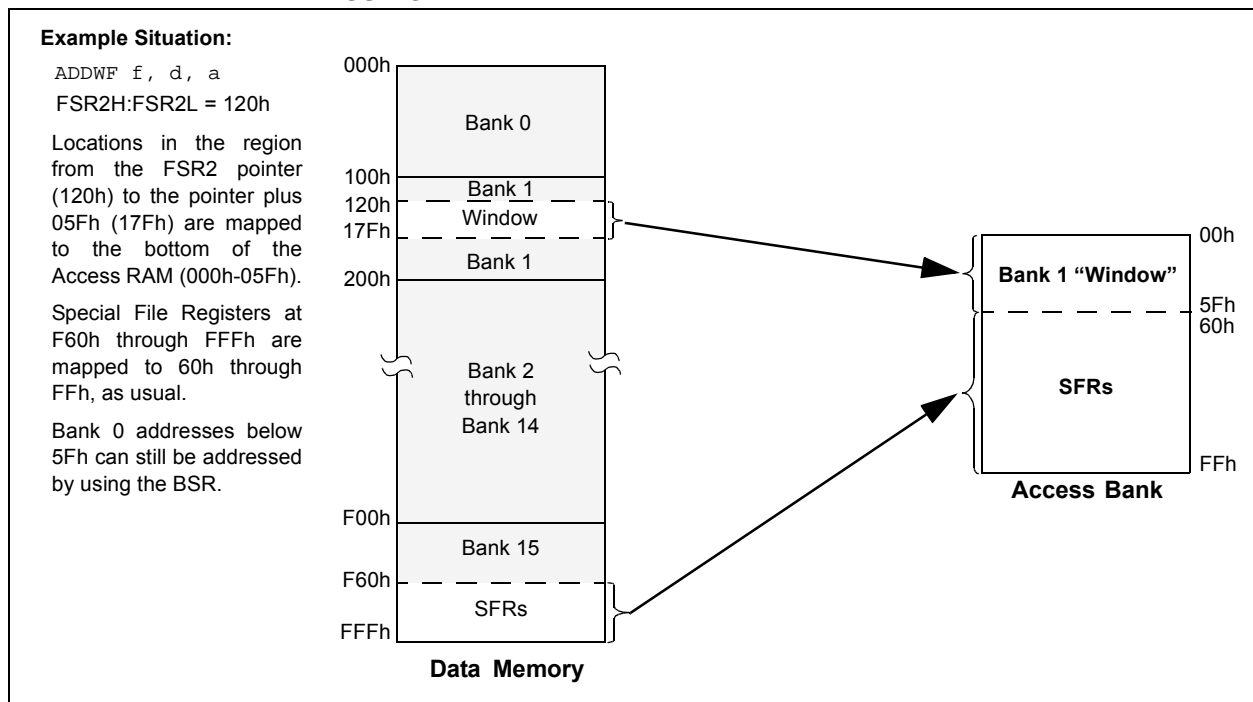
The use of Indexed Literal Offset Addressing mode effectively changes how the first 96 locations of Access RAM (00h to 5Fh) are mapped. Rather than containing just the contents of the bottom section of Bank 0, this mode maps the contents from a user defined “window” that can be located anywhere in the 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 [Section 10.4.2 “Access Bank”](#)). An example of Access Bank remapping in this addressing mode is shown in [Figure 10-8](#).

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.

10.8 PIC18 Instruction Execution and the Extended Instruction Set

Enabling the extended instruction set adds eight additional commands to the existing PIC18 instruction set. These instructions are executed as described in [Section 36.2 “Extended Instruction Set”](#).

FIGURE 10-8: REMAPPING THE ACCESS BANK WITH INDEXED LITERAL OFFSET ADDRESSING



11.0 NONVOLATILE MEMORY (NVM) CONTROL

Nonvolatile Memory (NVM) is separated into two types: Program Flash Memory (PFM) and Data EEPROM Memory.

PFM, Data EEPROM, User IDs and Configuration bits can all be accessed using the NVMREG<1:0> bits of the NVMCON1 register.

The write time is controlled by an on-chip timer. The write/erase voltages are generated by an on-chip charge pump rated to operate over the operating voltage range of the device.

NVM can be protected in two ways, by either code protection or write protection. Code protection (\overline{CP} and \overline{CPD} bits in Configuration Word 5L) disables access, reading and writing to both PFM and Data EEPROM Memory via external device programmers. Code protection does not affect the self-write and erase functionality. Code protection can only be reset by a device programmer performing a Bulk Erase to the device, clearing all nonvolatile memory, Configuration bits and User IDs.

Write protection prohibits self-write and erase to a portion or all of the PFM, as defined by the WRT bits of Configuration Word 4H. Write protection does not affect a device programmer's ability to read, write or erase the device.

TABLE 11-1: NVM ORGANIZATION AND ACCESS INFORMATION

| Memory | PC<20:0> ICSP™ Addr<21:0> TBLPTR<21:0> NVMADDR<9:0> | Execution | User Access | | |
|--------------------------------|--|---------------|-------------------|---------------------------|-------------------|
| | | CPU Execution | NVMREG | TABLAT | NVMDAT |
| User Flash Memory (PFM) | 00 0000h ... 01 FFFFh | Read | 10 | Read/Write ⁽¹⁾ | __ ⁽³⁾ |
| User IDs ⁽²⁾ | 20 0000h ... 20 000Fh | No Access | x1 | Read/Write | __ ⁽³⁾ |
| Reserved | 20 0010h 2F FFFFh | No Access | __ ⁽³⁾ | | |
| Configuration | 30 0000h ... 30 000Bh | No Access | x1 | Read/Write | __ ⁽³⁾ |
| Reserved | 30 000Ch 30 FFFFh | No Access | __ ⁽³⁾ | | |
| User Data Memory (Data EEPROM) | 31 0000h ... 31 03FFh | No Access | 00 | __ ⁽³⁾ | Read/Write |
| Reserved | 32 0000h 3F FFFBh | No Access | __ ⁽³⁾ | | |
| Revision ID/ Device ID | 3F FFFCh ... 3F FFFFh | No Access | x1 | Read | __ ⁽³⁾ |

Note 1: Subject to Memory Write Protection settings.

2: User IDs are eight words ONLY. There is no code protection, table read protection or write protection implemented for this region.

3: Reads as '0', writes clear the WR bit and WRERR bit is set.

11.1 Program Flash Memory

The Program Flash Memory is readable, writable and erasable during normal operation over the entire VDD range.

A read from program memory is executed one byte at a time. A write to program memory or program memory erase is executed on blocks of n bytes at a time. Refer to [Table 11-3](#) for write and erase block sizes. A Bulk Erase operation cannot be issued from user code.

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, therefore, code cannot execute. An internal programming timer terminates 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. PFM is arranged in

rows. A row is the minimum size that can be erased by user software. Refer to [Table 11-3](#) for the row sizes for the these devices.

After a row has been erased, all or a portion of this row can be programmed. Data to be written into the program memory row is written to 6-bit wide data write latches. These latches are not directly accessible, but may be loaded via sequential writes to the TABLAT register.

Note: To modify only a portion of a previously programmed row, then the contents of the entire row must be read and saved in RAM prior to the erase. Then, the new data and retained data can be written into the write latches to reprogram the row of PFM. However, any unprogrammed locations can be written without first erasing the row. In this case, it is not necessary to save and rewrite the other previously programmed locations

TABLE 11-2: FLASH MEMORY ORGANIZATION BY DEVICE

| Device | Row Erase Size (Words) | Write Latches (Bytes) | Program Flash Memory (Words) | Data Memory (Bytes) |
|----------------|------------------------|-----------------------|------------------------------|---------------------|
| PIC18(L)F65K40 | 32 | 64 | 16384 | 1024 |
| PIC18(L)F66K40 | | | 32768 | |
| PIC18(L)F67K40 | 64 | 128 | 65536 | |

11.1.1 TABLE READS AND TABLE WRITES

In order 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 program memory space is 16 bits wide, while the data RAM space is eight bits wide. Table reads and table writes move data between these two memory spaces through an 8-bit register (TABLAT).

The table read operation retrieves one byte of data directly from program memory and places it into the TABLAT register. Figure 11-1 shows the operation of a table read.

The table write operation stores one byte of data from the TABLAT register into a write block holding register. The procedure to write the contents of the holding registers into program memory is detailed in Section 11.1.6 “Writing to Program Flash Memory”. Figure 11-2 shows the operation of a table write with program memory and data RAM.

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.

FIGURE 11-1: TABLE READ OPERATION

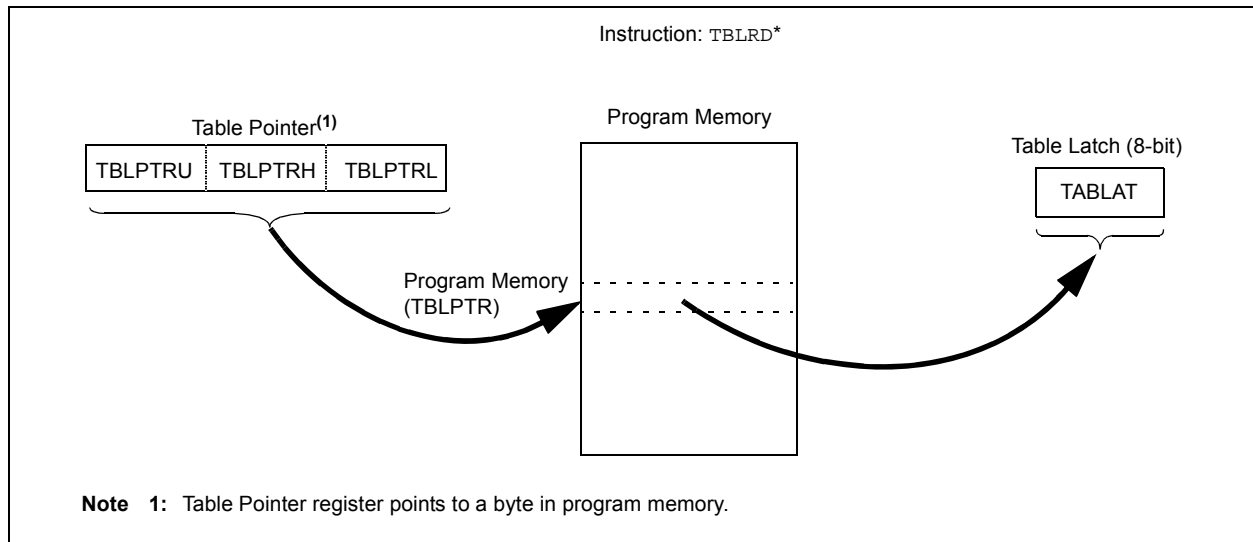
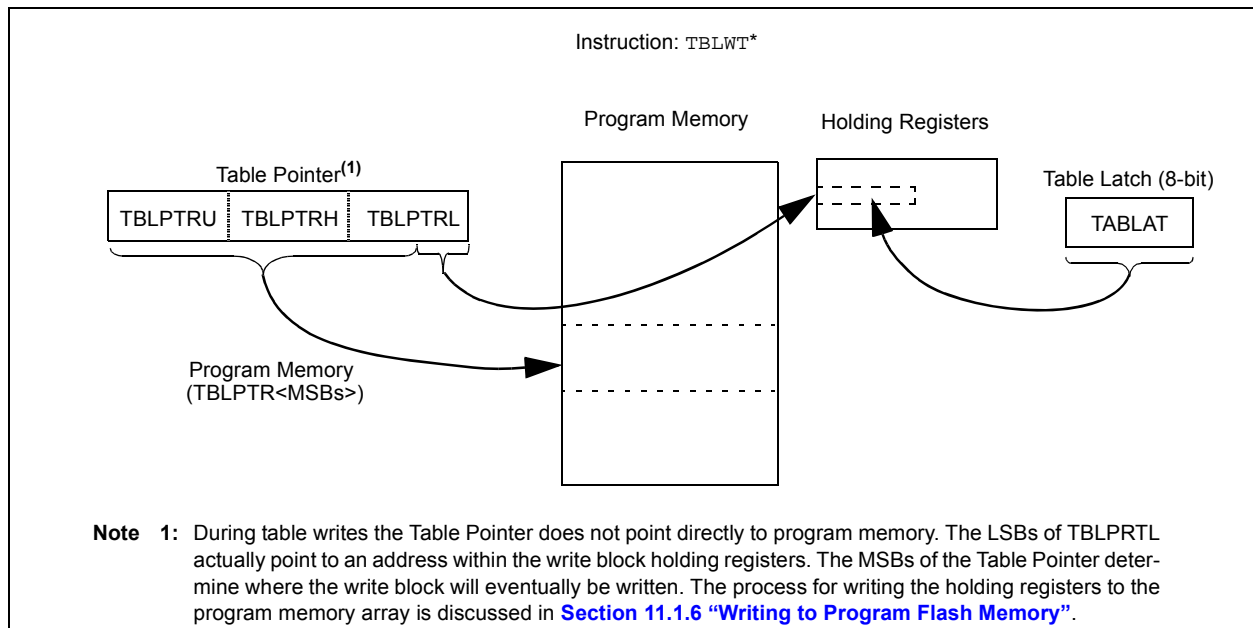


FIGURE 11-2: TABLE WRITE OPERATION



11.1.2 CONTROL REGISTERS

Several control registers are used in conjunction with the `TBLRD` and `TBLWT` instructions. These include the following registers:

- NVMCON1 register
- NVMCON2 register
- TABLAT register
- TBLPTR registers

11.1.2.1 NVMCON1 and NVMCON2 Registers

The NVMCON1 register ([Register 11-1](#)) is the control register for memory accesses. The NVMCON2 register is not a physical register; it is used exclusively in the memory write and erase sequences. Reading NVMCON2 will read all '0's.

The NVMREG<1:0> control bits determine if the access will be to Data EEPROM Memory locations, PFM locations or User IDs, Configuration bits, Rev ID and Device ID. When NVMREG<1:0> = 00, any subsequent operations will operate on the Data EEPROM Memory. When NVMREG<1:0> = 10, any subsequent operations will operate on the program memory. When NVMREG<1:0> = x1, any subsequent operations will operate on the Configuration bits, User IDs, Rev ID and Device ID.

The FREE bit allows the program memory erase operation. When the FREE bit is set, an erase operation is initiated on the next WR command. When FREE is clear, only writes are enabled. This bit is applicable only to the PFM and not to data EEPROM.

When set, the WREN bit will allow a program/erase operation. The WREN bit is cleared on power-up.

The WRERR bit is set by hardware when the WR bit is set and cleared when the internal programming timer expires and the write operation is successfully complete.

The WR control bit initiates erase/write cycle operation when the NVMREG<1:0> bits point to the Data EEPROM Memory location, and it initiates a write operation when the NVMREG<1:0> bits point to the PFM location. The WR bit cannot be cleared by firmware; it can only be set by firmware. Then the WR bit is cleared by hardware at the completion of the write operation.

The NVMIF Interrupt Flag bit of the PIR7 register is set when the write is complete. The NVMIF flag stays set until cleared by firmware.

11.1.2.2 TABLAT – Table Latch Register

The Table Latch (TABLAT) is an 8-bit register mapped into the SFR space. The Table Latch register is used to hold 8-bit data during data transfers between program memory and data RAM.

11.1.2.3 TBLPTR – Table Pointer Register

The Table Pointer (TBLPTR) register addresses a byte within the program memory. The TBLPTR is comprised of 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. The low-order 21 bits allow the device to address up to 2 Mbytes of program memory space. The 22nd bit allows access to the Device ID, the User ID and the Configuration bits.

The Table Pointer register, TBLPTR, is used by the `TBLRD` and `TBLWT` instructions. These instructions can update the TBLPTR in one of four ways based on the table operation. These operations on the TBLPTR affect only the low-order 21 bits.

11.1.2.4 Table Pointer Boundaries

TBLPTR is used in reads, writes and erases of the Program Flash Memory.

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

When a `TBLWT` is executed the byte in the TABLAT register is written, not to Flash memory but, to a holding register in preparation for a program memory write. The holding registers constitute a write block which varies depending on the device (see [Table 11-3](#)). The 3, 4, or 5 LSBs of the TBLPTRL register determine which specific address within the holding register block is written to. The MSBs of the Table Pointer have no effect during `TBLWT` operations.

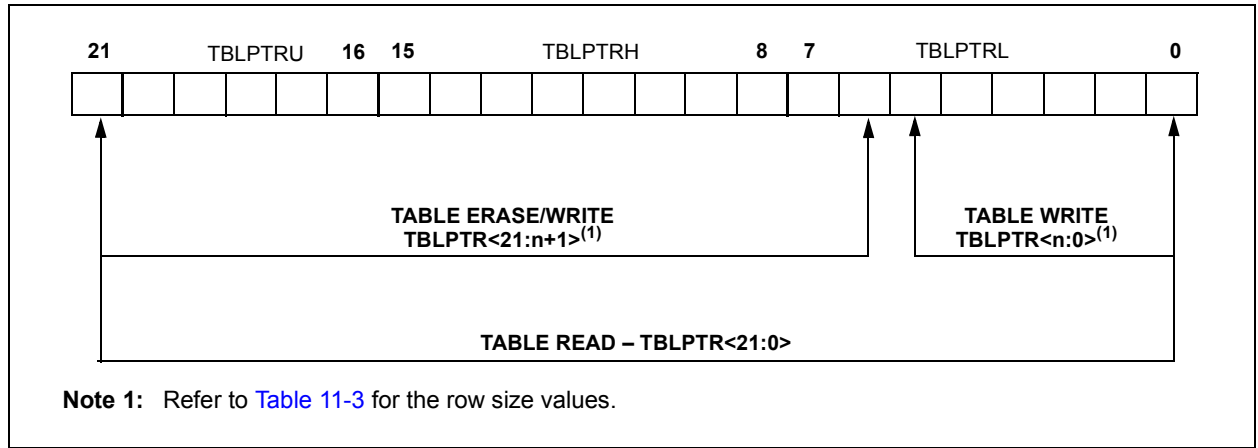
When a program memory write is executed the entire holding register block is written to the Flash memory at the address determined by the MSBs of the TBLPTR. The 3, 4, or 5 LSBs are ignored during Flash memory writes. For more detail, see [Section 11.1.6 “Writing to Program Flash Memory”](#).

[Figure 11-3](#) describes the relevant boundaries of TBLPTR based on Program Flash Memory operations.

TABLE 11-3: TABLE POINTER OPERATIONS WITH TBLRD AND TBLWT INSTRUCTIONS

| Example | Operation on Table Pointer |
|--------------------|---|
| TBLRD* TBLWT* | TBLPTR is not modified |
| TBLRD*+ TBLWT*+ | TBLPTR is incremented after the read/write |
| TBLRD*- TBLWT*- | TBLPTR is decremented after the read/write |
| TBLRD+* TBLWT+* | TBLPTR is incremented before the read/write |

FIGURE 11-3: TABLE POINTER BOUNDARIES BASED ON OPERATION



11.1.3 READING THE PROGRAM FLASH MEMORY

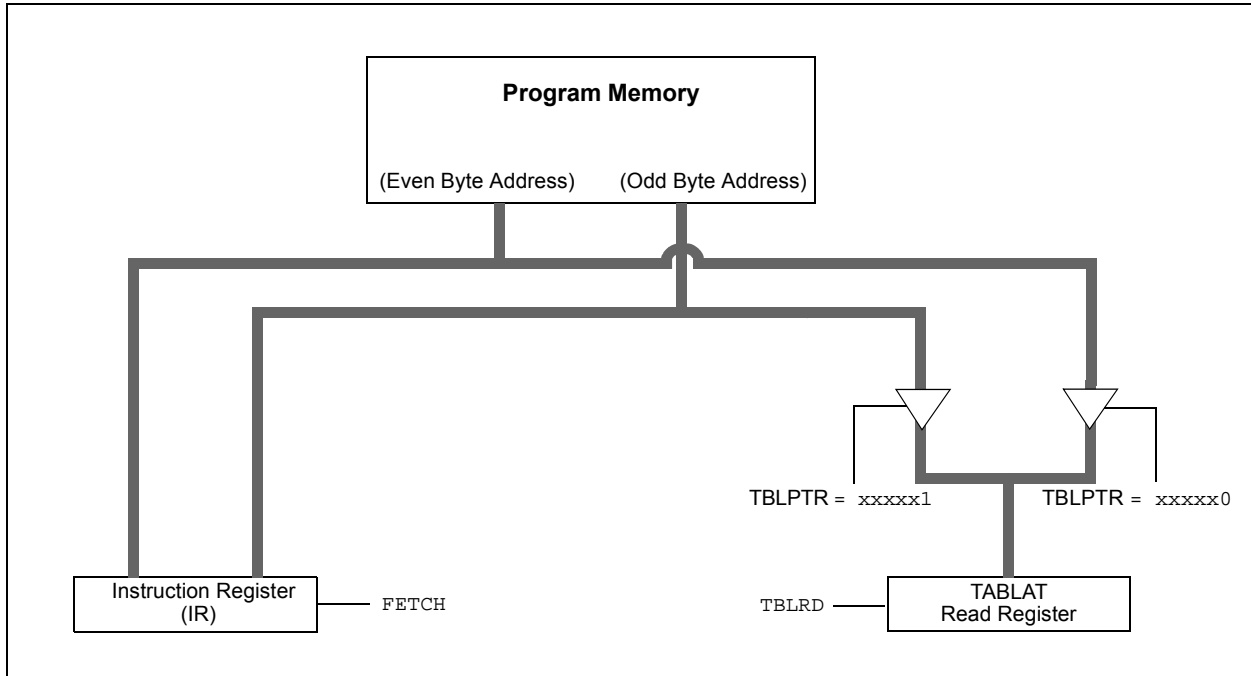
The `TBLRD` instruction retrieves data from program memory and places it into data RAM. Table reads from program memory are performed one byte at a time.

`TBLPTR` points to a byte address in program space. Executing `TBLRD` places the byte pointed to into `TABLAT`. In addition, `TBLPTR` can be modified automatically for the next table read operation.

The CPU operation is suspended during the read, and it resumes immediately after. From the user point of view, `TABLAT` 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. [Figure 11-4](#) shows the interface between the internal program memory and the `TABLAT`.

FIGURE 11-4: READS FROM PROGRAM FLASH MEMORY



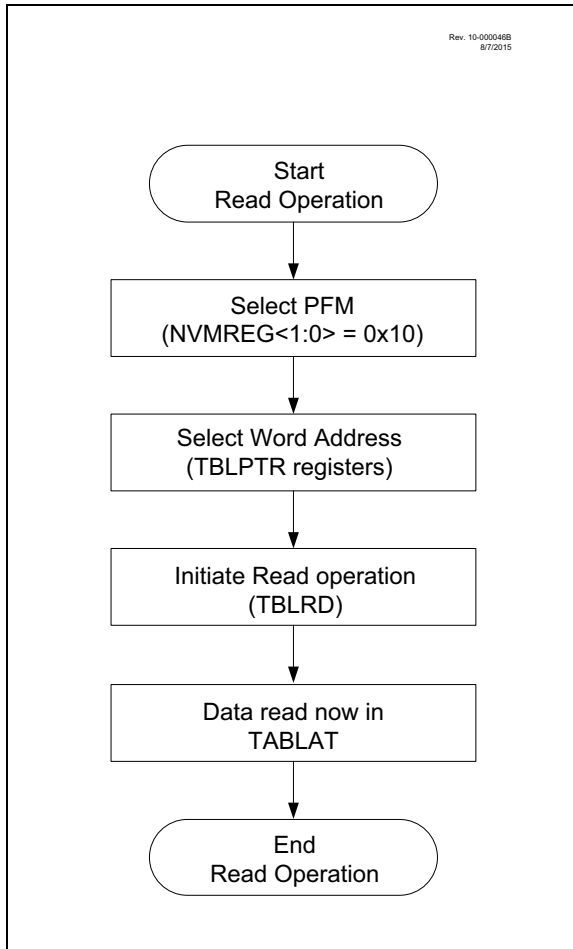
EXAMPLE 11-1: READING A PROGRAM FLASH MEMORY WORD

```

MOV LW    CODE_ADDR_UPPER    ; Load TBLPTR with the base
MOV WF    TBLPTRU             ; address of the word
MOV LW    CODE_ADDR_HIGH
MOV WF    TBLPTRH
MOV LW    CODE_ADDR_LOW
MOV WF    TBLPTRL

READ_WORD
TBLRD*+           ; read into TABLAT and increment
MOV F    TABLAT, W ; get data
MOV WF    WORD_EVEN
TBLRD*+           ; read into TABLAT and increment
MOV FW    TABLAT, W ; get data
MOV F    WORD_ODD
    
```

**FIGURE 11-5: PROGRAM FLASH
MEMORY READ
FLOWCHART**



11.1.4 NVM UNLOCK SEQUENCE

The unlock sequence is a mechanism that protects the NVM from unintended self-write programming or erasing. The sequence must be executed and completed without interruption to successfully complete any of the following operations:

- PFM Row Erase
- Write of PFM write latches to PFM memory
- Write of PFM write latches to User IDs
- Write to Data EEPROM Memory
- Write to Configuration Words

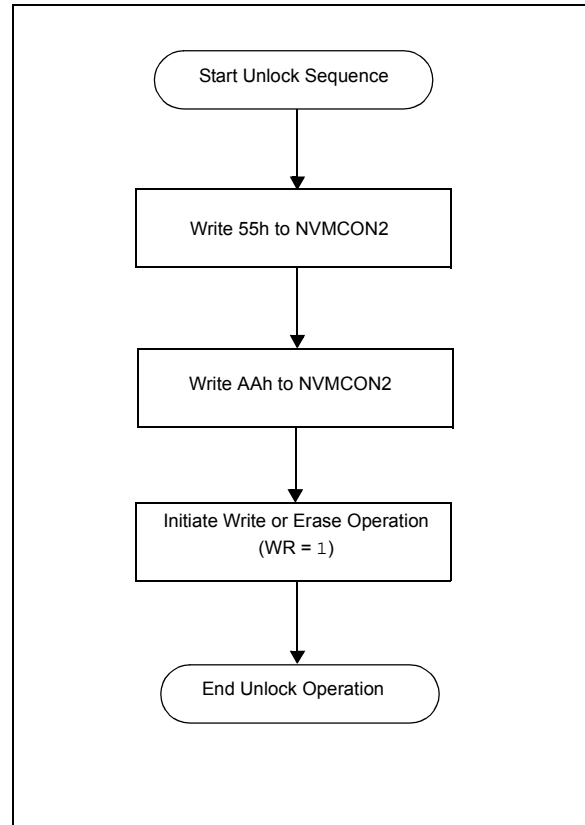
The unlock sequence consists of the following steps and must be completed in order:

- Write 55h to NVMCON2
- Write AAh to NVMCON2
- Set the WR bit of NVMCON1

Once the WR bit is set, the processor will stall internal operations until the operation is complete and then resume with the next instruction.

Since the unlock sequence must not be interrupted, global interrupts should be disabled prior to the unlock sequence and re-enabled after the unlock sequence is completed.

FIGURE 11-6: NVM UNLOCK SEQUENCE FLOWCHART



EXAMPLE 11-2: NVM UNLOCK SEQUENCE

```

BCF          INTCON,GIE      ; Recommended so sequence is not interrupted
BANKSEL     NVMCON1
BSF         NVMCON1,WREN     ; Enable write/erase
MOVLW      55h              ; Load 55h

MOVWF      NVMCON2          ; Step 1: Load 55h into NVMCON2
MOVLW      AAh              ; Step 2: Load W with AAh
MOVWF      NVMCON2          ; Step 3: Load AAh into NVMCON2
BSF         INTCON1,WR      ; Step 4: Set WR bit to begin write/erase

BSF         INTCON,GIE      ; Re-enable interrupts
  
```

Note 1: Sequence begins when NVMCON2 is written; steps 1-4 must occur in the cycle-accurate order shown. If the timing of the steps 1 to 4 is corrupted by an interrupt or a debugger Halt, the action will not take place.

2: Opcodes shown are illustrative; any instruction that has the indicated effect may be used.

11.1.5 ERASING PROGRAM FLASH MEMORY

The minimum erase block is 32 or 64 words (refer to [Table 11-3](#)). Only through the use of an external programmer, or through ICSP™ control, can larger blocks of program memory be bulk erased. Word erase in the Flash array is not supported.

For example, when initiating an erase sequence from a microcontroller with erase row size of 32 words, a block of 32 words (64 bytes) of program memory is erased. The Most Significant 16 bits of the TBLPTR<21:6> point to the block being erased. The TBLPTR<5:0> bits are ignored.

The NVMCON1 register commands the erase operation. The NVMREG<1:0> bits must be set to point to the Program Flash Memory. The WREN bit must be set to enable write operations. The FREE bit is set to select an erase operation.

The NVM unlock sequence described in [Section 11.1.4 “NVM Unlock Sequence”](#) should be used to guard against accidental writes. This is sometimes referred to as a long write.

A long write is necessary for erasing the internal Flash. Instruction execution is halted during the long write cycle. The long write is terminated by the internal programming timer.

11.1.5.1 Program Flash Memory Erase Sequence

The sequence of events for erasing a block of internal program memory is:

1. NVMREG bits of the NVMCON1 register point to PFM
2. Set the FREE and WREN bits of the NVMCON1 register
3. Perform the unlock sequence as described in [Section 11.1.4 “NVM Unlock Sequence”](#)

If the PFM address is write-protected, the WR bit will be cleared and the erase operation will not take place, WRERR is signaled in this scenario.

The operation erases the memory row indicated by masking the LSBs of the current TBLPTR.

While erasing PFM, CPU operation is suspended and it resumes when the operation is complete. Upon completion the WR bit is cleared in hardware, the NVMIF is set and an interrupt will occur if the NVMIE bit is also set.

Write latch data is not affected by erase operations and WREN will remain unchanged.

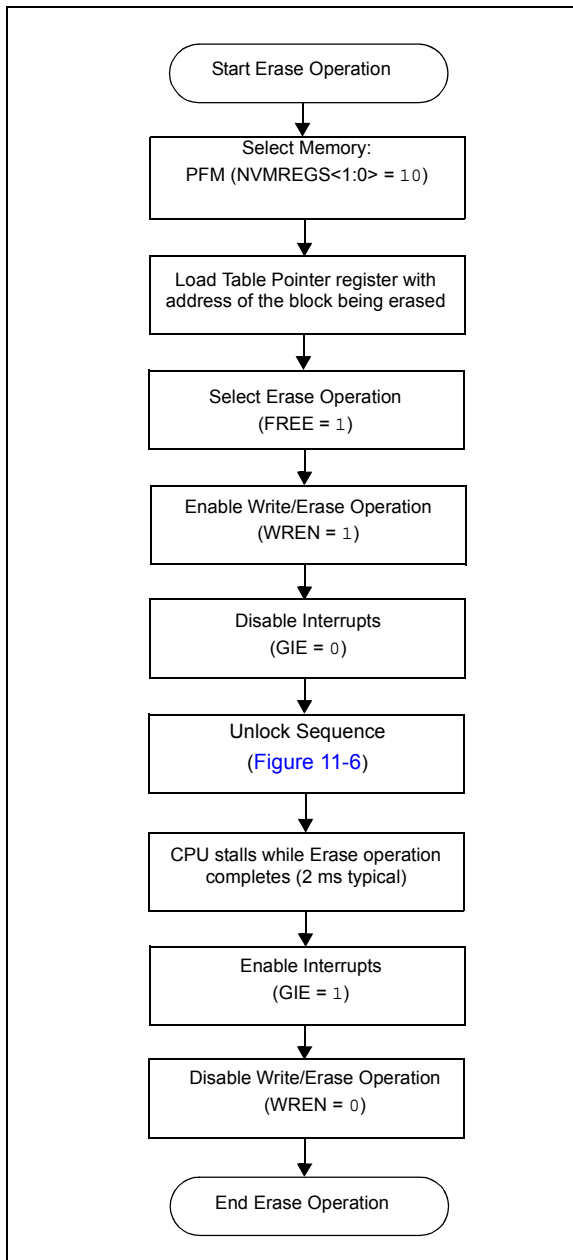
- | |
|--|
| <p>Note 1: If a write or erase operation is terminated by an unexpected event, WRERR bit will be set which the user can check to decide whether a rewrite of the location(s) is needed.</p> <p>2: WRERR is set if WR is written to '1' while TBLPTR points to a write-protected address.</p> <p>3: WRERR is set if WR is written to '1' while TBLPTR points to an invalid address location (Table 10-1 and Table 11-1).</p> |
|--|

EXAMPLE 11-3: ERASING A PROGRAM FLASH MEMORY BLOCK

```
; This sample row erase routine assumes the following:
; 1. A valid address within the erase row is loaded in variables TBLPTR register
; 2. ADDRH and ADDRL are located in common RAM (locations 0x70 - 0x7F)

        MOVLW    CODE_ADDR_UPPER    ; load TBLPTR with the base
        MOVWF   TBLPTRU             ; address of the memory block
        MOVLW    CODE_ADDR_HIGH
        MOVWF   TBLPTRH
        MOVLW    CODE_ADDR_LOW
        MOVWF   TBLPTRL
ERASE_BLOCK
        BCF     NVMCON1, NVMREG0    ; point to Program Flash Memory
        BSF     NVMCON1, NVMREG1    ; access Program Flash Memory
        BSF     NVMCON1, WREN       ; enable write to memory
        BSF     NVMCON1, FREE       ; enable block Erase operation
        BCF     INTCON, GIE         ; disable interrupts
Required MOVLW    55h
Sequence MOVWF   NVMCON2             ; write 55h
        MOVLW    AAh
        MOVWF   NVMCON2             ; write AAh
        BSF     NVMCON1, WR         ; start erase (CPU stalls)
        BSF     INTCON, GIE         ; re-enable interrupts
```

FIGURE 11-7: PFM ROW ERASE FLOWCHART



11.1.6 WRITING TO PROGRAM FLASH MEMORY

The programming write block size is described in [Table 11-3](#). Word or byte programming is not supported. Table writes are used internally to load the holding registers needed to program the Flash memory. There are only as many holding registers as there are bytes in a write block. Refer to [Table 11-3](#) for write latch size.

Since the table latch (TABLAT) is only a single byte, the TBLWT instruction needs to be executed multiple times for each programming operation. The write protection state is ignored for this operation. All of the table write operations will essentially be short writes because only the holding registers are written. NVMIF is not affected while writing to the holding registers.

After all the holding registers have been written, the programming operation of that block of memory is started by configuring the NVMCON1 register for a program memory write and performing the long write sequence.

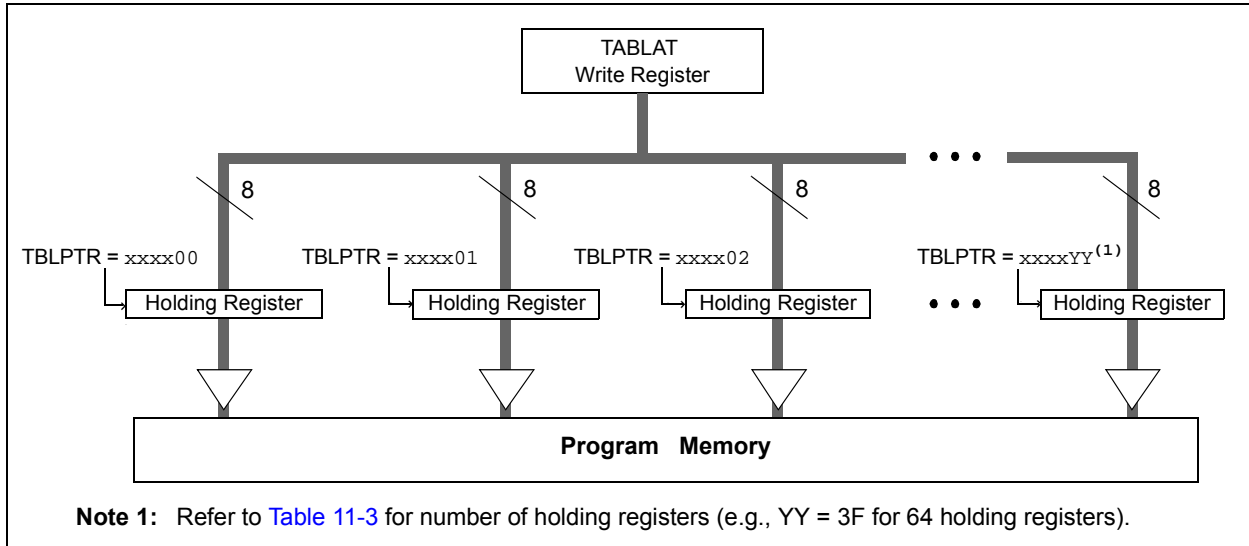
If the PFM address in the TBLPTR is write-protected or if TBLPTR points to an invalid location, the WR bit is cleared without any effect and the WREER is signaled.

The long write is necessary for programming the internal Flash. CPU operation is suspended during a long write cycle and resumes when the operation is complete. The long write operation completes in one instruction cycle. When complete, WR is cleared in hardware and NVMIF is set and an interrupt will occur if NVMIE is also set. The latched data is reset to all '1s'. WREN is not changed.

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

Note: The default value of the holding registers on device Resets and after write operations is FFh. A write of FFh to a holding register does not modify that byte. This means that individual bytes of program memory may be modified, provided that the change does not attempt to change any bit from a '0' to a '1'. When modifying individual bytes, it is not necessary to load all holding registers before executing a long write operation.

FIGURE 11-8: TABLE WRITES TO PROGRAM FLASH MEMORY



11.1.6.1 Program Flash Memory Write Sequence

The sequence of events for programming an internal program memory location should be:

1. Read appropriate number of bytes into RAM. Refer to [Table 11-2](#) for Write latch size.
2. Update data values in RAM as necessary.
3. Load Table Pointer register with address being erased.
4. Execute the block erase procedure.
5. Load Table Pointer register with address of first byte being written.
6. Write the n-byte block into the holding registers with auto-increment. Refer to [Table 11-2](#) for Write latch size.
7. Set NVMREG<1:0> bits to point to program memory.
8. Clear FREE bit and set WREN bit in NVMCON1 register.
9. Disable interrupts.
10. Execute the unlock sequence (see [Section 11.1.4 "NVM Unlock Sequence"](#)).
11. WR bit is set in NVMCON1 register.
12. The CPU will stall for the duration of the write (about 2 ms using internal timer).
13. Re-enable interrupts.
14. Verify the memory (table read).

This procedure will require about 6 ms to update each write block of memory. An example of the required code is given in [Example 11-4](#).

Note: Before setting the WR bit, the Table Pointer address needs to be within the intended address range of the bytes in the holding registers.

EXAMPLE 11-4: WRITING TO PROGRAM FLASH MEMORY

```

        MOVLW    D'64'                ; number of bytes in erase block
        MOVWF   COUNTER
        MOVLW   BUFFER_ADDR_HIGH     ; point to buffer
        MOVWF   FSR0H
        MOVLW   BUFFER_ADDR_LOW
        MOVWF   FSR0L
        MOVLW   CODE_ADDR_UPPER      ; Load TBLPTR with the base
        MOVWF   TBLPTRU              ; address of the memory block
        MOVLW   CODE_ADDR_HIGH
        MOVWF   TBLPTRH
        MOVLW   CODE_ADDR_LOW
        MOVWF   TBLPTRL

READ_BLOCK
        TBLRD*+                       ; read into TABLAT, and inc
        MOVF    TABLAT, W             ; get data
        MOVWF   POSTINC0             ; store data
        DECFSZ  COUNTER              ; done?
        BRA     READ_BLOCK           ; repeat

MODIFY_WORD
        MOVLW   BUFFER_ADDR_HIGH     ; point to buffer
        MOVWF   FSR0H
        MOVLW   BUFFER_ADDR_LOW
        MOVWF   FSR0L
        MOVLW   NEW_DATA_LOW         ; update buffer word
        MOVWF   POSTINC0
        MOVLW   NEW_DATA_HIGH
        MOVWF   INDF0

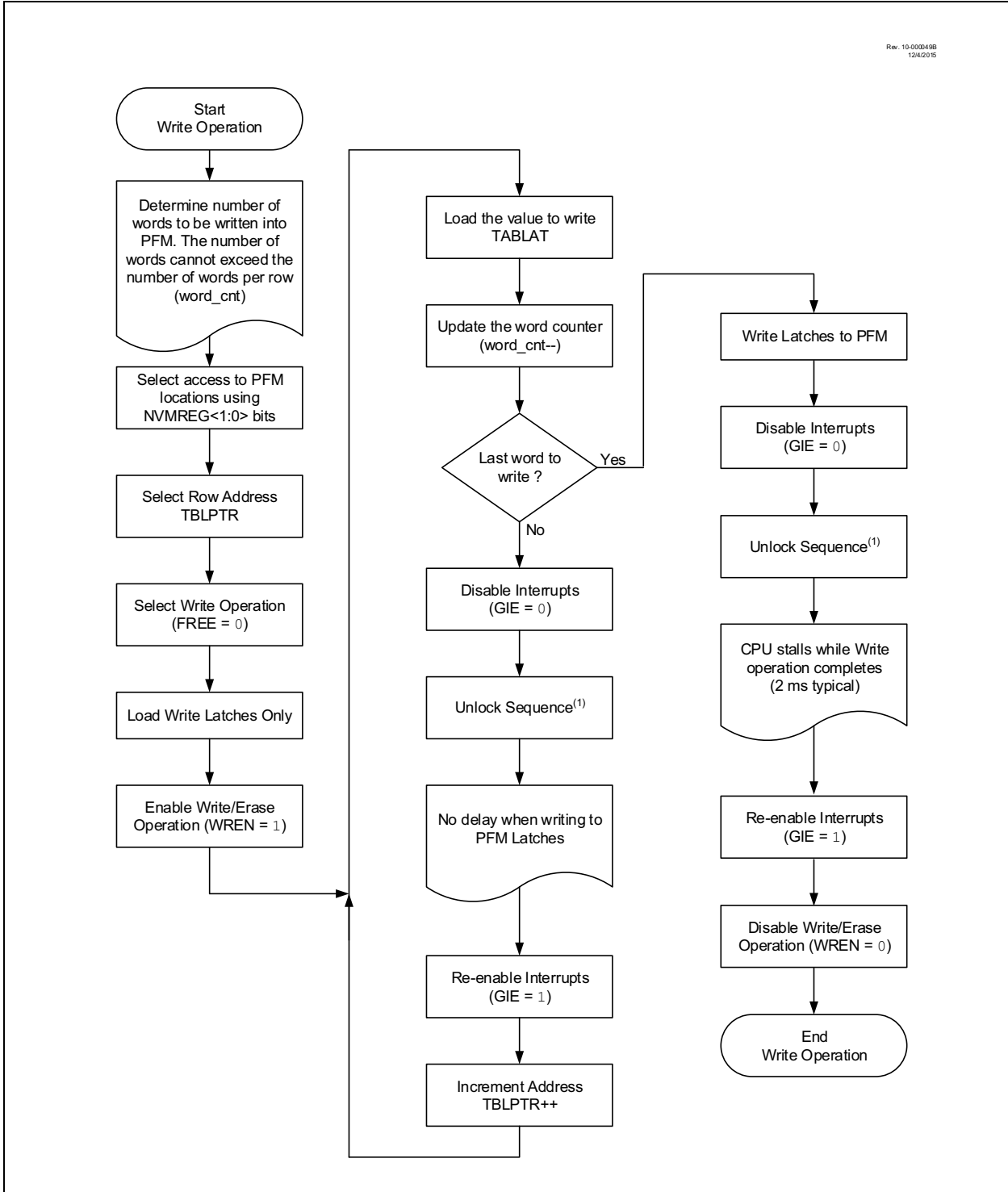
ERASE_BLOCK
        MOVLW   CODE_ADDR_UPPER      ; load TBLPTR with the base
        MOVWF   TBLPTRU              ; address of the memory block
        MOVLW   CODE_ADDR_HIGH
        MOVWF   TBLPTRH
        MOVLW   CODE_ADDR_LOW
        MOVWF   TBLPTRL
        BCF    NVMCON1, NVMREG0      ; point to Program Flash Memory
        BSF    NVMCON1, NVMREG1      ; point to Program Flash Memory
        BSF    NVMCON1, WREN         ; enable write to memory
        BSF    NVMCON1, FREE         ; enable Erase operation
        BCF    INTCON, GIE           ; disable interrupts
        MOVLW   55h
        MOVWF   NVMCON2              ; write 55h
        MOVLW   AAh
        MOVWF   NVMCON2              ; write 0AAh
        BSF    NVMCON1, WR           ; start erase (CPU stall)
        BSF    INTCON, GIE           ; re-enable interrupts
        TBLRD*-                       ; dummy read decrement
        MOVLW   BUFFER_ADDR_HIGH     ; point to buffer
        MOVWF   FSR0H
        MOVLW   BUFFER_ADDR_LOW
        MOVWF   FSR0L
WRITE_BUFFER_BACK
        MOVLW   BlockSize            ; number of bytes in holding register
        MOVWF   COUNTER
        MOVLW   D'64'/BlockSize     ; number of write blocks in 64 bytes
        MOVWF   COUNTER2
    
```

EXAMPLE 11-4: WRITING TO PROGRAM FLASH MEMORY (CONTINUED)

```
WRITE_BYTE_TO_HREGS
    MOVF    POSTINC0, W           ; get low byte of buffer data
    MOVWF   TABLAT               ; present data to table latch
    TBLWT+*                       ; write data, perform a short write
                                   ; to internal TBLWT holding register.
    DECFSZ  COUNTER              ; loop until holding registers are full
    BRA     WRITE_WORD_TO_HREGS

PROGRAM_MEMORY
    BCF     NVMCON1, NVMREG0      ; point to Program Flash Memory
    BSF     NVMCON1, NVMREG1      ; point to Program Flash Memory
    BSF     NVMCON1, WREN         ; enable write to memory
    BCF     NVMCON1, FREE        ; enable write to memory
    BCF     INTCON, GIE          ; disable interrupts
    MOVLW   55h
Required MOVWF   NVMCON2           ; write 55h
Sequence MOVLW   0AAh
    MOVWF   NVMCON2             ; write 0AAh
    BSF     NVMCON1, WR          ; start program (CPU stall)
    DCFSZ  COUNTER2             ; repeat for remaining write blocks
    BRA     WRITE_BYTE_TO_HREGS
    BSF     INTCON, GIE          ; re-enable interrupts
    BCF     NVMCON1, WREN        ; disable write to memory
```

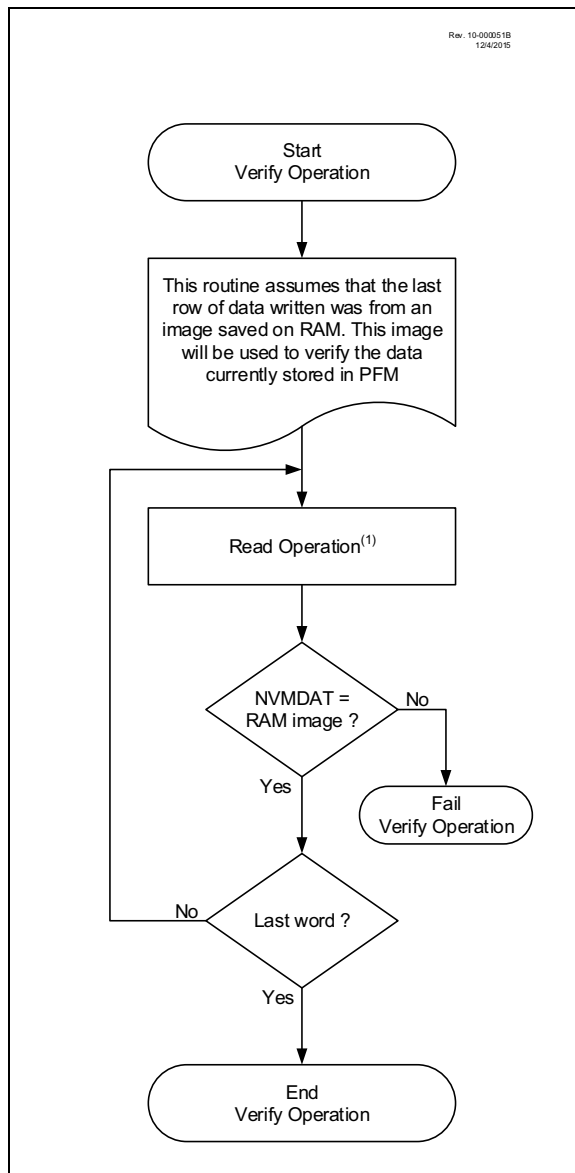
FIGURE 11-9: PROGRAM FLASH MEMORY (PFM) WRITE FLOWCHART



11.1.6.2 Write Verify

Depending on the application, good programming practice may dictate that the value written to the memory should be verified against the original value. This should 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 RAM after the last write is complete.

FIGURE 11-10: PROGRAM FLASH MEMORY VERIFY FLOWCHART



11.1.6.3 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 should be verified and reprogrammed if needed. If the write operation is interrupted by a 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.

11.1.6.4 Protection Against Spurious Writes

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

1. The WR bit is gated through the WREN bit. It is suggested to have the WREN bit cleared at all times except during memory writes. This prevents memory writes if the WR bit gets set accidentally.
2. The NVM unlock sequence must be performed each time before a write operation.

11.2 User ID, Device ID and Configuration Word Access

When $NVMREG<1:0> = 0x01$ or $0x11$ in the NVMCON1 register, the User ID's, Device ID/Revision ID and Configuration Words can be accessed. Different access may exist for reads and writes (see [Table 11-3](#)).

11.2.1 Reading Access

The user can read from these blocks by setting the NVMREG bits to $0x01$ or $0x11$. The user needs to load the address into the TBLPTR registers. Executing a TBLRD after that moves the byte pointed to the TABLAT register. The CPU operation is suspended during the read and resumes after. When read access is initiated on an address outside the parameters listed in [Table 11-3](#), the TABLAT register is cleared, reading back '0's.

11.2.2 Writing Access

The WREN bit in NVMCON1 must be set to enable writes. This prevents accidental writes to the CONFIG words due to errant (unexpected) code execution. The WREN bit should be kept clear at all times, except when updating the CONFIG words. The WREN bit is not cleared by hardware. The WR bit will be inhibited from being set unless the WREN bit is set.

The user needs to load the TBLPTR and TABLAT register with the address and data byte respectively before executing the write command. An unlock sequence needs to be followed for writing to the USER IDs/DEVICE IDs/CONFIG words ([Section 11.1.4, NVM Unlock Sequence](#)). If WRTC = 0 or if TBLPTR points an invalid address location (see [Table 11-3](#)), WR bit is cleared without any effect and WRERR is set.

A single CONFIG word byte is written at once and the operation includes an implicit erase cycle for that byte (it is not necessary to set FREE). CPU execution is stalled and at the completion of the write cycle, the WR bit is cleared in hardware and the NVM Interrupt Flag bit (NVMIF) is set. The new CONFIG value takes effect when the CPU resumes operation.

TABLE 11-4: USER ID, DEV/REV ID AND CONFIGURATION WORD ACCESS (NVMREG<1:0> = x1)

| Address | Function | Read Access | Write Access |
|-------------------|-------------------------|-------------|--------------|
| 20 0000h-20 000Fh | User IDs | Yes | Yes |
| 3F FFFCh-3F FFFFh | Revision ID/Device ID | Yes | No |
| 30 0000h-30 000Bh | Configuration Words 1-6 | Yes | Yes |

11.3 Data EEPROM Memory

The data EEPROM is a nonvolatile memory array, separate from the data RAM and program memory, which is used for long-term storage of program data. It is not directly mapped in either the register file or program memory space but is indirectly addressed through the Special Function Registers (SFRs). The EEPROM is readable and writable during normal operation over the entire VDD range.

Four SFRs are used to read and write to the data EEPROM as well as the program memory. They are:

- NVMCON1
- NVMCON2
- NVMDAT
- NVMADRL
- NVMADRH

The data EEPROM allows byte read and write. When interfacing to the data memory block, NVMDAT holds the 8-bit data for read/write and the NVMADRH:NVMADRL register pair hold the address of the EEPROM location being accessed.

The EEPROM data memory 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 chip-to-chip. Please refer to the Data EEPROM Memory parameters in [Section 37.0 “Electrical Specifications”](#) for limits.

11.3.1 NVMADRL AND NVMADRH REGISTERS

The NVMADRH:NVMADRL registers are used to address the data EEPROM for read and write operations.

11.3.2 NVMCON1 AND NVMCON2 REGISTERS

Access to the data EEPROM is controlled by two registers: NVMCON1 and NVMCON2. These are the same registers which control access to the program memory and are used in a similar manner for the data EEPROM.

The NVMCON1 register ([Register 11-1](#)) is the control register for data and program memory access. Control bits NVMREG<1:0> determine if the access will be to program, Data EEPROM Memory or the User IDs, Configuration bits, Revision ID and Device ID.

The WREN bit, when set, will allow a write operation. On power-up, the WREN bit is clear.

The WRERR bit is set by hardware when the WR bit is set and cleared when the internal programming timer expires and the write operation is complete.

The WR control bit initiates write operations. The bit can be set but not cleared by software. It is cleared only by hardware at the completion of the write operation.

The NVMIF interrupt flag bit of the PIR7 register is set when the write is complete. It must be cleared by software.

Control bits, RD and WR, start read and erase/write operations, respectively. These bits are set by firmware and cleared by hardware at the completion of the operation.

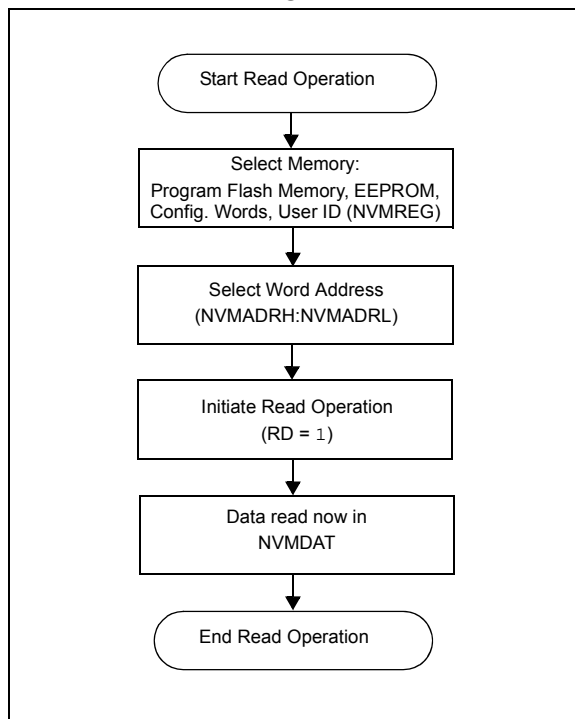
The RD bit cannot be set when accessing program memory (NVMREG<1:0> = 0x10). Program memory is read using table read instructions. See [Section 11.1.1 “Table Reads and Table Writes”](#) regarding table reads.

11.3.3 READING THE DATA EEPROM MEMORY

To read a data memory location, the user must write the address to the NVMADRL and NVMADRH register pair, clear NVMREG<1:0> control bit in NVMCON1 register to access Data EEPROM locations and then set control bit, RD. The data is available on the very next instruction cycle; therefore, the NVMDAT register can be read by the next instruction. NVMDAT will hold this value until another read operation, or until it is written to by the user (during a write operation).

The basic process is shown in [Example 11-5](#).

FIGURE 11-11: PROGRAM FLASH MEMORY READ FLOWCHART



11.3.4 WRITING TO THE DATA EEPROM MEMORY

To write an EEPROM data location, the address must first be written to the NVMADRL and NVMADRH register pair and the data written to the NVMDAT register. The sequence in [Example 11-6](#) must be followed to initiate the write cycle.

The write will not begin if NVM Unlock sequence, described in [Section 11.1.4 “NVM Unlock Sequence”](#), is not exactly followed for each byte. It is strongly recommended that interrupts be disabled during this code segment.

Additionally, the WREN bit in NVMCON1 must be set to enable writes. This mechanism prevents accidental writes to data EEPROM due to unexpected code execution (i.e., runaway programs). The WREN bit should be kept clear at all times, except when updating the EEPROM. The WREN bit is not cleared by hardware.

After a write sequence has been initiated, NVMCON1, NVMADRL, NVMADRH and NVMDAT cannot be modified. The WR bit will be inhibited from being set unless the WREN bit is set. Both WR and WREN cannot be set with the same instruction.

After a write sequence has been initiated, clearing the WREN bit will not affect this write cycle. A single Data EEPROM word is written and the operation includes an implicit erase cycle for that word (it is not necessary to set FREE). CPU execution continues in parallel and at the completion of the write cycle, the WR bit is cleared in hardware and the NVM Interrupt Flag bit (NVMIF) is set. The user can either enable this interrupt or poll this bit. NVMIF must be cleared by software.

11.3.5 WRITE VERIFY

Depending on the application, good programming practice may dictate that the value written to the memory should be verified against the original value. This should be used in applications where excessive writes can stress bits near the specification limit.

EXAMPLE 11-5: DATA EEPROM READ

```
; Data Memory Address to read
      BCF NVMCON1, NVMREG0 ; Setup Data EEPROM Access
      BCF NVMCON1, NVMREG1 ; Setup Data EEPROM Access
      MOVF EE_ADDRL, W      ;
      MOVWF NVMADRL        ; Setup Address low byte
      MOVF EE_ADDRH, W      ;
      MOVWF NVMADRH        ; Setup Address high byte (if applicable)
      BSF NVMCON1, RD      ; Issue EE Read
      MOVF NVMDAT, W       ; W = EE_DATA
```

EXAMPLE 11-6: DATA EEPROM WRITE

```
; Data Memory Address to write
      BCF NVMCON1, NVMREG0 ; Setup Data EEPROM access
      BCF NVMCON1, NVMREG1 ; Setup Data EEPROM access
      MOVF EE_ADDRL, W      ;
      MOVWF NVMADRL        ; Setup Address low byte
      MOVF EE_ADDRH, W      ;
      MOVWF NVMADRH        ; Setup Address high byte (if applicable)
; Data Memory Value to write
      MOVF EE_DATA, W       ;
      MOVWF NVMDAT         ;
; Enable writes
      BSF NVMCON1, WREN    ;
; Disable interrupts
      BCF INTCON, GIE      ;
; Required unlock sequence
      MOVLW 55h            ;
      MOVWF NVMCON2        ;
      MOVLW AAh            ;
      MOVWF NVMCON2        ;
; Set WR bit to begin write
      BSF NVMCON1, WR      ;
; Wait for write to complete
      BTFSC NVMCON1, WR    ;
      BRA $-2              ;
; Enable INT
      BSF INTCON, GIE      ;
; Disable writes
      BCF NVMCON1, WREN    ;
```

11.3.6 OPERATION DURING CODE-PROTECT

Data EEPROM Memory has its own code-protect bits in Configuration Words. External read and write operations are disabled if code protection is enabled.

If the Data EEPROM is write-protected or if NVMADR points an invalid address location, the WR bit is cleared without any effect. WRERR is signaled in this scenario.

11.3.7 PROTECTION AGAINST SPURIOUS WRITE

There are conditions when the user may not want to write to the Data EEPROM Memory. To protect against spurious EEPROM writes, various mechanisms have been implemented. On power-up, the WREN bit is cleared. In addition, writes to the EEPROM are blocked during the Power-up Timer period (TPWRT).

The unlock sequence and the WREN bit together help prevent an accidental write during brown-out, power glitch or software malfunction.

11.3.8 ERASING THE DATA EEPROM MEMORY

Data EEPROM Memory can be erased by writing 0xFF to all locations in the Data EEPROM Memory that needs to be erased.

EXAMPLE 11-7: DATA EEPROM REFRESH ROUTINE

```

        CLRF    NVMADRL           ; Clear address low byte register
        CLRF    NVMADRH           ; Clear address high byte register (if applicable)
        BCF     NVMCON1, NVMREG0   ; Set access for EEPROM
        BCF     NVMCON1, NVMREG1   ; Set access for EEPROM
        SETF    NVMDAT            ; Load 0xFF to data register
        BCF     INTCON, GIE        ; Disable interrupts
        BSF     NVMCON1, WREN      ; Enable writes
Loop
        MOVLW   0x55              ; Initiate unlock sequence
        MOVWF   NVMCON2           ;
        MOVLW   0xAA              ;
        MOVWF   NVMCON2           ;
        BSF     NVMCON1, WR       ; Set WR bit to begin write
        BTFSC  NVMCON1, WR       ; Wait for write to complete
        BRA     $-2
        INCFSZ  NVMADRL, F        ; Increment address low byte
        BRA     Loop              ; Not zero, do it again

//The following 4 lines of code are not needed if the part doesn't have NVMADRH register
        INCF    NVMADRH, F        ; Decrement address high byte
        MOVLW   0x03              ; Move 0x03 to working register
        CPFSGT  NVMADRH          ; Compare address high byte with working register
        BRA     Loop              ; Skip if greater than working register
                                   ; Else go back to erase loop

        BCF     NVMCON1, WREN     ; Disable writes
        BSF     INTCON, GIE       ; Enable interrupts
    
```

11.4 Register Definitions: Nonvolatile Memory

REGISTER 11-1: NVMCON1: NONVOLATILE MEMORY CONTROL 1 REGISTER

| | | | | | | | |
|-------------|---------|-----|------------|------------|---------|------------|------------|
| R/W-0/0 | R/W-0/0 | U-0 | R/S/HC-0/0 | R/W/HS-x/q | R/W-0/0 | R/S/HC-0/0 | R/S/HC-0/0 |
| NVMREG<1:0> | | — | FREE | WRERR | WREN | WR | RD |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|-------------------|---|
| R = Readable bit | W = Writable bit | HC = Bit is cleared by hardware |
| x = Bit is unknown | -n = Value at POR | S = Bit can be set by software, but not cleared |
| '0' = Bit is cleared | '1' = Bit is set | U = Unimplemented bit, read as '0' |

- bit 7-6 **NVMREG<1:0>**: NVM Region Selection bit
 10 = Access PFM Locations
 x1 = Access User IDs, Configuration Bits, Rev ID and Device ID
 00 = Access Data EEPROM Memory Locations
- bit 5 **Unimplemented**: Read as '0'
- bit 4 **FREE**: Program Flash Memory Erase Enable bit⁽¹⁾
 1 = Performs an erase operation on the next WR command
 0 = The next WR command performs a write operation
- bit 3 **WRERR**: Write-Reset Error Flag bit^(2,3,4)
 1 = A write operation was interrupted by a Reset (hardware set),
 or WR was written to 1'b1 when an invalid address is accessed (Table 10-1, Table 11-1)
 or WR was written to 1'b1 when NVMREG<1:0> and address do not point to the same region
 or WR was written to 1'b1 when a write-protected address is accessed (Table 10-2).
 0 = All write operations have completed normally
- bit 2 **WREN**: Program/Erase Enable bit
 1 = Allows program/erase and refresh cycles
 0 = Inhibits programming/erasing and user refresh of NVM
- bit 1 **WR**: Write Control bit^(5,6,7)
When NVMREG points to a Data EEPROM Memory location:
 1 = Initiates an erase/program cycle at the corresponding Data EEPROM Memory location
When NVMREG points to a PFM location:
 1 = Initiates the PFM write operation with data from the holding registers
 0 = NVM program/erase operation is complete and inactive
- bit 0 **RD**: Read Control bit⁽⁸⁾
 1 = Initiates a read at address pointed by NVMREG and NVMDR, and loads data into NVMDAT
 0 = NVM read operation is complete and inactive

- Note 1:** This can only be used with PFM.
2: This bit is set when WR = 1 and clears when the internal programming timer expires or the write is completed successfully.
3: Bit must be cleared by the user; hardware will not clear this bit.
4: Bit may be written to '1' by the user in order to implement test sequences.
5: This bit can only be set by following the unlock sequence of Section 11.1.4 "NVM Unlock Sequence".
6: Operations are self-timed and the WR bit is cleared by hardware when complete.
7: Once a write operation is initiated, setting this bit to zero will have no effect.
8: The bit can only be set in software. The bit is cleared by hardware when the operation is complete.

REGISTER 11-2: NVMCON2: NONVOLATILE MEMORY CONTROL 2 REGISTER

| | | | | | | | |
|--------------|-------|-------|-------|-------|-------|-------|-------|
| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
| NVMCON2<7:0> | | | | | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

R = Readable bit W = Writable bit U = Unimplemented bit, read as '0'
x = Bit is unknown '0' = Bit is cleared '1' = Bit is set
-n = Value at POR

bit 7-0 **NVMCON2<7:0>**:
Refer to [Section 11.1.4 “NVM Unlock Sequence”](#).

Note 1: This register always reads zeros, regardless of data written.

Register 11-3: NVMADRL: Data EEPROM Memory Address Low

| | | | | | | | |
|-------------|---------|---------|---------|---------|---------|---------|---------|
| R/W-x/0 | R/W-x/0 | R/W-x/0 | R/W-x/0 | R/W-x/0 | R/W-x/0 | R/W-x/0 | R/W-x/0 |
| NVMADR<7:0> | | | | | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

R = Readable bit W = Writable bit U = Unimplemented bit, read as '0'
x = Bit is unknown '0' = Bit is cleared '1' = Bit is set
-n = Value at POR

bit 7-0 **NVMADR<7:0>**: EEPROM Read Address bits

REGISTER 11-4: NVMADRH: DATA EEPROM MEMORY ADDRESS HIGH

| | | | | | | | |
|-------|-----|-----|-----|-----|-----|-------------|---------|
| U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | R/W-x/u | R/W-x/u |
| — | — | — | — | — | — | NVMADR<9:8> | |
| bit 7 | | | | | | | bit 0 |

Legend:

R = Readable bit W = Writable bit U = Unimplemented bit, read as '0'
x = Bit is unknown '0' = Bit is cleared '1' = Bit is set
-n = Value at POR

bit 7-2 **Unimplemented:** Read as '0'
bit 1-0 **NVMADR<9:8>**: EEPROM Read Address bits

REGISTER 11-5: NVMDAT: DATA EEPROM MEMORY DATA

| | | | | | | | |
|-------------|---------|---------|---------|---------|---------|---------|---------|
| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| NVMDAT<7:0> | | | | | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

R = Readable bit W = Writable bit U = Unimplemented bit, read as '0'
x = Bit is unknown '0' = Bit is cleared '1' = Bit is set
-n = Value at POR

bit 7-0 **NVMDAT<7:0>**: The value of the data memory word returned from NVMADR after a Read command, or the data written by a Write command.

TABLE 11-5: SUMMARY OF REGISTERS ASSOCIATED WITH DATA EEPROM MEMORY

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on Page | |
|---------|---|-----------|--|-------|-------|---------|-------------|---------|------------------|------|
| NVMCON1 | NVMREG<1:0> | | — | FREE | WRERR | WREN | WR | RD | 147 | |
| NVMCON2 | Unlock Pattern | | | | | | | | 148 | |
| NVMADRL | NVMADR<7:0> | | | | | | | | 148 | |
| NVMADRH | — | — | — | — | — | — | NVMADR<9:8> | | 148 | |
| NVMDAT | NVMDAT<7:0> | | | | | | | | 149 | |
| TBLPTRU | — | — | Program Memory Table Pointer (TBLPTR<21:16>) | | | | | | | 129* |
| TBLPTRH | Program Memory Table Pointer (TBLPTR<15:8>) | | | | | | | | 129* | |
| TBLPTRL | Program Memory Table Pointer (TBLPTR<7:0>) | | | | | | | | 129* | |
| TABLAT | TABLAT | | | | | | | | 128* | |
| INTCON | GIE/GIEH | PEIE/GIEL | IPEN | — | — | INT2EDG | INT1EDG | INT0EDG | 172 | |
| PIE8 | SCANIE | CRCIE | NVMIE | — | — | — | — | CWG1IE | 192 | |
| PIR8 | SCANIF | CRCIF | NVMIF | — | — | — | — | CWG1IF | 182 | |
| IPR8 | SCANIP | CRCIP | NVMIP | — | — | — | — | CWG1IP | 202 | |

Legend: — = unimplemented, read as '0'. Shaded bits are not used during EEPROM access.

*Page provides register information.

12.0 8x8 HARDWARE MULTIPLIER

12.1 Introduction

All PIC18 devices include an 8x8 hardware multiplier as part of the ALU. The multiplier performs an unsigned operation and yields a 16-bit result that is stored in the product register pair, PRODH:PRODL. 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 PIC18 devices 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 12-1](#).

12.2 Operation

[Example 12-1](#) 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 12-2](#) 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 12-1: 8x8 UNSIGNED MULTIPLY ROUTINE

```
MOVWF ARG1, W ;
MULWF ARG2 ; ARG1 * ARG2 ->
; PRODH:PRODL
```

EXAMPLE 12-2: 8x8 SIGNED MULTIPLY ROUTINE

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

MOVWF ARG2, W
BTFSC ARG1, SB ; Test Sign Bit
SUBWF PRODH, F ; PRODH = PRODH
; - ARG2
```

TABLE 12-1: 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 |

Example 12-3 shows the sequence to do a 16 x 16 unsigned multiplication. Equation 12-1 shows the algorithm that is used. The 32-bit result is stored in four registers (RES<3:0>).

EQUATION 12-1: 16 x 16 UNSIGNED MULTIPLICATION ALGORITHM

$$\begin{aligned} \text{RES3:RES0} &= \text{ARG1H:ARG1L} \cdot \text{ARG2H:ARG2L} \\ &= (\text{ARG1H} \cdot \text{ARG2H} \cdot 2^{16}) + \\ &\quad (\text{ARG1H} \cdot \text{ARG2L} \cdot 2^8) + \\ &\quad (\text{ARG1L} \cdot \text{ARG2H} \cdot 2^8) + \\ &\quad (\text{ARG1L} \cdot \text{ARG2L}) \end{aligned}$$

EXAMPLE 12-3: 16 x 16 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
MOVF PRODH, W         ; products
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
MOVF PRODH, W         ; products
ADDWFC RES2, F
CLRF WREG
ADDWFC RES3, F
;

```

Example 12-4 shows the sequence to do a 16 x 16 signed multiply. Equation 12-2 shows the algorithm used. The 32-bit result is stored in four registers (RES<3:0>). To account for the sign bits of the arguments, the MSb for each argument pair is tested and the appropriate subtractions are done.

EQUATION 12-2: 16 x 16 SIGNED MULTIPLICATION ALGORITHM

$$\begin{aligned} \text{RES3:RES0} &= \text{ARG1H:ARG1L} \cdot \text{ARG2H:ARG2L} \\ &= (\text{ARG1H} \cdot \text{ARG2H} \cdot 2^{16}) + \\ &\quad (\text{ARG1H} \cdot \text{ARG2L} \cdot 2^8) + \\ &\quad (\text{ARG1L} \cdot \text{ARG2H} \cdot 2^8) + \\ &\quad (\text{ARG1L} \cdot \text{ARG2L}) + \\ &\quad (-1 \cdot \text{ARG2H} \cdot \text{ARG1H:ARG1L} \cdot 2^{16}) + \\ &\quad (-1 \cdot \text{ARG1H} \cdot \text{ARG2H:ARG2L} \cdot 2^{16}) \end{aligned}$$

EXAMPLE 12-4: 16 x 16 SIGNED 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
MOVF PRODH, W         ; products
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
MOVF PRODH, W         ; products
ADDWFC RES2, F
CLRF WREG
ADDWFC RES3, F
;

BTFS ARG2H, 7         ; ARG2H:ARG2L neg?
BRA SIGN_ARG1         ; no, check ARG1
MOVF ARG1L, W
SUBWF RES2
MOVF ARG1H, W
SUBWFB RES3
;

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

CONT_CODE
:

```

13.0 CYCLIC REDUNDANCY CHECK (CRC) 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 CRC's

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 Register Definitions: CRC and Scanner Control

Long bit name prefixes for the CRC and Scanner peripherals are shown in [Table 13-1](#). Refer to [Section 1.4.2.2 “Long Bit Names”](#) for more information.

TABLE 13-1:

| Peripheral | Bit Name Prefix |
|------------|-----------------|
| CRC | CRC |

REGISTER 13-1: CRCCON0: CRC CONTROL REGISTER 0

| R/W-0/0 | R/W-0/0 | R-0 | R/W-0/0 | U-0 | U-0 | R/W-0/0 | R-0 |
|---------|---------|------|---------|-----|-----|---------|-------|
| EN | GO | BUSY | ACCM | — | — | SHIFTM | FULL |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

- bit 7 **EN:** CRC Enable bit
1 = CRC module is released from Reset
0 = CRC is disabled and consumes no operating current
- bit 6 **GO:** CRC Start bit
1 = Start CRC serial shifter
0 = CRC serial shifter turned off
- bit 5 **BUSY:** CRC Busy bit
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 bit
1 = Data is augmented with zeros
0 = Data is not augmented with zeros
- bit 3-2 **Unimplemented:** Read as '0'
- bit 1 **SHIFTM:** Shift Mode bit
1 = Shift right (LSb)
0 = Shift left (MSb)
- bit 0 **FULL:** Data Path Full Indicator bit
1 = CRCDATA/L registers are full
0 = CRCDATA/L registers have shifted their data into the shifter

REGISTER 13-2: CRCCON1: CRC CONTROL REGISTER 1

| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|-----------|---------|---------|---------|-----------|---------|---------|---------|
| DLEN<3:0> | | | | PLEN<3:0> | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

- bit 7-4 **DLEN<3:0>:** Data Length bits
Denotes the length of the data word -1 (See [Example 13-1](#))
- bit 3-0 **PLEN<3:0>:** Polynomial Length bits
Denotes the length of the polynomial -1 (See [Example 13-1](#))

REGISTER 13-3: CRCDATA: CRC DATA HIGH BYTE REGISTER

| | | | | | | | |
|------------|---------|---------|---------|---------|---------|---------|---------|
| R/W-xx | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x |
| DATA<15:8> | | | | | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-0 **DATA<15:8>**: CRC Input/Output Data bits

REGISTER 13-4: CRCDATL: CRC DATA LOW BYTE REGISTER

| | | | | | | | |
|-----------|---------|---------|---------|---------|---------|---------|---------|
| R/W-xx | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x |
| DATA<7:0> | | | | | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-0 **DATA<7:0>**: CRC Input/Output Data bits
Writing to this register fills the shifter.

REGISTER 13-5: CRCACCH: CRC ACCUMULATOR HIGH BYTE REGISTER

| | | | | | | | |
|-----------|---------|---------|---------|---------|---------|---------|---------|
| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| ACC<15:8> | | | | | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-0 **ACC<15:8>**: CRC Accumulator Register bits
Writing to this register writes to the CRC accumulator register. Reading from this register reads the CRC accumulator.

REGISTER 13-6: CRCACCL: CRC ACCUMULATOR LOW BYTE REGISTER

| | | | | | | | |
|----------|---------|---------|---------|---------|---------|---------|---------|
| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| ACC<7:0> | | | | | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-0 **ACC<7:0>**: CRC Accumulator Register bits
 Writing to this register writes to the CRC accumulator register through the CRC write bus. Reading from this register reads the CRC accumulator.

REGISTER 13-7: CRCSHIFTH: CRC SHIFT HIGH BYTE REGISTER

| | | | | | | | |
|-------------|-----|-----|-----|-----|-----|-----|-------|
| R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |
| SHIFT<15:8> | | | | | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-0 **SHIFT<15:8>**: CRC Shifter Register bits
 Reading from this register reads the CRC Shifter.

REGISTER 13-8: CRCSHIFTL: CRC SHIFT LOW BYTE REGISTER

| | | | | | | | |
|------------|-----|-----|-----|-----|-----|-----|-------|
| R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |
| SHIFT<7:0> | | | | | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-0 **SHIFT<7:0>**: CRC Shifter Register bits
 Reading from this register reads the CRC Shifter.

REGISTER 13-9: CRCXORH: CRC XOR HIGH BYTE REGISTER

| | | | | | | | |
|---------|---------|---------|---------|---------|---------|---------|---------|
| R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x |
| X<15:8> | | | | | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-0 **X<15:8>**: XOR of Polynomial Term XN Enable bits

REGISTER 13-10: CRCXORL: CRC XOR LOW BYTE REGISTER

| | | | | | | | |
|---------|---------|---------|---------|---------|---------|---------|-------|
| R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | U-1 |
| X<7:1> | | | | | | | — |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-1 **X<7:1>**: XOR of Polynomial Term XN Enable bits

bit 0 **Unimplemented**: Read as '1'

REGISTER 13-11: SCANCON0: SCANNER ACCESS CONTROL REGISTER 0

| | | | | | | | |
|---------|------------|------|---------|---------|-----|-----------|---------|
| R/W-0/0 | R/W/HC-0/0 | R-0 | R-1 | R/W-0/0 | U-0 | R/W-0/0 | R/W-0/0 |
| SCANEN | SCANGO | BUSY | INVALID | INTM | — | MODE<1:0> | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | HC = Bit is cleared by hardware |

| | |
|---------|---|
| bit 7 | SCANEN: Scanner Enable bit ⁽¹⁾ 1 = Scanner is enabled 0 = Scanner is disabled, internal states are reset |
| bit 6 | SCANGO: Scanner GO bit ^(2, 3) 1 = When the CRC sends a ready signal, NVM will be accessed according to MDx and data passed to the client peripheral. 0 = Scanner operations will not occur |
| bit 5 | BUSY: Scanner Busy Indicator bit ⁽⁴⁾ 1 = Scanner cycle is in process 0 = Scanner cycle is complete (or never started) |
| bit 4 | INVALID: Scanner Abort Signal bit 1 = SCANLADRL/H/U has incremented to an invalid address ⁽⁶⁾ or the scanner was not setup correctly ⁽⁷⁾ 0 = SCANLADRL/H/U points to a valid address |
| bit 3 | INTM: NVM Scanner Interrupt Management Mode Select bit <u>If MODE = 10:</u> This bit is ignored <u>If MODE = 01 (CPU is stalled until all data is transferred):</u> 1 = SCANGO is overridden (to zero) during interrupt operation; scanner resumes after returning from interrupt 0 = SCANGO is not affected by interrupts, the interrupt response will be affected <u>If MODE = 00 or 11:</u> 1 = SCANGO is overridden (to zero) during interrupt operation; scan operations resume after returning from interrupt 0 = Interrupts do not prevent NVM access |
| bit 2 | Unimplemented: Read as '0' |
| bit 1-0 | MODE<1:0>: Memory Access Mode bits ⁽⁵⁾ 11 = Triggered mode 10 = Peek mode 01 = Burst mode 00 = Concurrent mode |

- Note**
- Setting SCANEN = 0 (SCANCON0 register) does not affect any other register content.
 - This bit is cleared when LADR > HADR (and a data cycle is not occurring).
 - If INTM = 1, this bit is overridden (to zero, but not cleared) during an interrupt response.
 - BUSY = 1 when the NVM is being accessed, or when the CRC sends a ready signal.
 - See [Table 13-2](#) for more detailed information.
 - An invalid address can occur when the entire range of PFM is scanned and the value of LADR rolls over. An invalid address can also occur if the value in the Scan Low address registers points to a location that is not mapped in the memory map of the device.
 - CRCEN and CRCGO bits must be set before setting SCANGO bit. Refer to [Section 13.9 “Program Memory Scan Configuration”](#).

REGISTER 13-12: SCANLADRU: SCAN LOW ADDRESS UPPER BYTE REGISTER

| | | | | | | | |
|-------|-----|------------------------------|---------|---------|---------|---------|---------|
| U-0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| — | — | LADR<21:16> ^(1,2) | | | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-6 **Unimplemented:** Read as '0'

bit 5-0 **LADR<21:16>:** Scan Start/Current Address bits^(1,2)
Upper bits of the current address to be fetched from, value increments on each fetch of memory.

Note 1: Registers SCANLADRU/H/L form a 22-bit value, but are not guarded for atomic or asynchronous access; registers should only be read or written while SCANGO = 0 (SCANCON0 register).

2: While SCANGO = 1 (SCANCON0 register), writing to this register is ignored.

REGISTER 13-13: SCANLADRH: SCAN LOW ADDRESS HIGH BYTE REGISTER

| | | | | | | | |
|-----------------------------|---------|---------|---------|---------|---------|---------|---------|
| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| LADR<15:8> ^(1,2) | | | | | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-0 **LADR<15:8>:** Scan Start/Current Address bits^(1,2)

Most Significant bits of the current address to be fetched from, value increments on each fetch of memory.

Note 1: Registers SCANLADRU/H/L form a 22-bit value, but are not guarded for atomic or asynchronous access; registers should only be read or written while SCANGO = 0 (SCANCON0 register).

2: While SCANGO = 1 (SCANCON0 register), writing to this register is ignored.

REGISTER 13-14: SCANLADRL: SCAN LOW ADDRESS LOW BYTE REGISTER

| | | | | | | | |
|-----------------------------|---------|---------|---------|---------|---------|---------|---------|
| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| LADR<7:0> ^(1, 2) | | | | | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-0 **LADR<7:0>**: Scan Start/Current Address bits^(1, 2)
 Least Significant bits of the current address to be fetched from, value increments on each fetch of memory

- Note 1:** Registers SCANLADRU/H/L form a 22-bit value, but are not guarded for atomic or asynchronous access; registers should only be read or written while SCANGO = 0 (SCANCON0 register).
2: While SCANGO = 1 (SCANCON0 register), writing to this register is ignored.

REGISTER 13-15: SCANHADRU: SCAN HIGH ADDRESS UPPER BYTE REGISTER

| | | | | | | | |
|-------|-----|-------------|---------|---------|---------|---------|---------|
| U-0 | U-0 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 |
| — | — | HADR<21:16> | | | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-6 **Unimplemented:** Read as '0'
 bit 5-0 **HADR<21:16>**: Scan End Address bits^(1, 2)
 Upper bits of the address at the end of the designated scan

- Note 1:** Registers SCANHADRU/H/L form a 22-bit value but are not guarded for atomic or asynchronous access; registers should only be read or written while SCANGO = 0 (SCANCON0 register).
2: While SCANGO = 1 (SCANCON0 register), writing to this register is ignored.

REGISTER 13-16: SCANHADR_H: SCAN HIGH ADDRESS HIGH BYTE REGISTER

| | | | | | | | |
|------------------------------|---------|---------|---------|---------|---------|---------|---------|
| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| HADR<15:8> ^(1, 2) | | | | | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-0 **HADR<15:8>**: Scan End Address bits^(1, 2)
 Most Significant bits of the address at the end of the designated scan

- Note 1:** Registers SCANHADRU/H/L form a 22-bit value, but are not guarded for atomic or asynchronous access; registers should only be read or written while SCANGO = 0 (SCANCON0 register).
- 2:** While SCANGO = 1 (SCANCON0 register), writing to this register is ignored.

REGISTER 13-17: SCANHADRL: SCAN HIGH ADDRESS LOW BYTE REGISTER

| | | | | | | | |
|-----------------------------|---------|---------|---------|---------|---------|---------|---------|
| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| HADR<7:0> ^(1, 2) | | | | | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-0 **HADR<7:0>**: Scan End Address bits^(1, 2)
 Least Significant bits of the address at the end of the designated scan

- Note 1:** Registers SCANHADRU/H/L form a 22-bit value, but are not guarded for atomic or asynchronous access; registers should only be read or written while SCANGO = 0 (SCANCON0 register).
- 2:** While SCANGO = 1 (SCANCON0 register), writing to this register is ignored.

REGISTER 13-18: SCANTRIG: SCAN TRIGGER SELECTION REGISTER

| | | | | | | | |
|-------|-----|-----|-----|-----------|---------|---------|---------|
| U-0 | U-0 | U-0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| — | — | — | — | TSEL<3:0> | | | |
| bit 7 | | | | bit 0 | | | |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-4 **Unimplemented:** Read as '0'

bit 3-0 **TSEL<3:0>:** Scanner Data Trigger Input Selection bits

1111-1101 = Reserved
 1100 = SMT2OUT
 1011 = SMT1OUT
 1010 = TMR8_postscaled
 1001 = TMR7_output
 1000 = TMR6_postscaled
 0111 = TMR5_output
 0110 = TMR4_postscaled
 0101 = TMR3_output
 0100 = TMR2_postscaled
 0011 = TMR1_output
 0010 = TMR0_output
 0001 = CLKREF_output
 0000 = LFINTOSC

13.3 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<15:0> registers for user storage. The CRC module uses an XOR shift register implementation to perform the polynomial division required for the CRC calculation.

EXAMPLE 13-1: CRC EXAMPLE

Rev. 10-000206A
1/8/2014

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

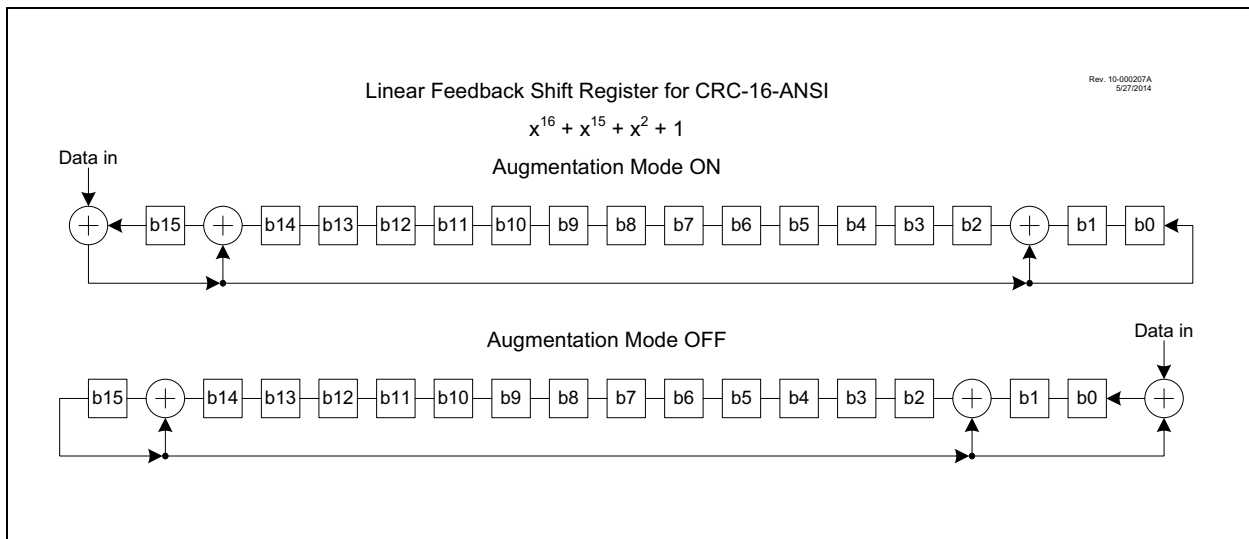
Note 1: Bit 0 is unimplemented. The LSb of any CRC polynomial is always '1' and will always be treated as a '1' by the CRC for calculating the CRC check value. This bit will be read in software as a '0'.

13.4 CRC Polynomial Implementation

Any polynomial can be used. The polynomial and accumulator sizes are determined by the PLEN<3:0> 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. 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. The actual value is 0x8005 because the hardware sets the LSb to 1. However, the LSb of the CRCXORL register is unimplemented and always reads as '0'. Refer to [Example 13-1](#).

EXAMPLE 13-2: CRC LFSR EXAMPLE



13.5 CRC Data Sources

Data can be input to the CRC module in two ways:

- User data using the CRCDATA registers (CRCDAT and CRCDATL)
- Flash using the Program Memory Scanner

To set the number of bits of data, up to 16 bits, the DLEN bits of CRCCON1 must be set accordingly. Only data bits in CRCDATA registers up to 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. If SHIFTM is not set, the data will be shifted in MSb first (Big Endian). The value of DLEN will determine the MSb. If SHIFTM bit is set, the data will be shifted into the accumulator in reversed order, LSb first (Little Endian).

The CRC module can be seeded with an initial value by setting the CRCACC<15:0> registers to the appropriate value before beginning the CRC.

13.5.1 CRC FROM USER DATA

To use the CRC module on data input from the user, the user must write the data to the CRCDAT registers. The data from the CRCDAT registers will be latched into the shift registers on any write to the CRCDATL register.

13.5.2 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 [Section 13.9, Program Memory Scan Configuration](#).

13.6 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 two mode settings of the CRCCON register: ACCM and SHIFTM.

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 CRCDAT 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 0.

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. This is the expected check value in bit reversed form. If you are 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. You can use the CRC to do this reversal by the following method:

- Save CRCACC value in user RAM space
- Clear the CRCACC registers
- Clear the CRCXOR registers
- Write the saved CRCACC value to the CRCDAT input

The properly oriented check value will be in the CRCACC registers as the result.

13.7 CRC Interrupt

The CRC will generate an interrupt when the BUSY bit transitions from 1 to 0. The CRCIF Interrupt Flag bit of the PIR7 register 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 in software. The CRC interrupt enable is the CRCIE bit of the PIE7 register.

13.8 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 [Section 13.5 “CRC Data Sources”](#), depending on which decision was made.
2. If desired, seed a starting CRC value into the CRCACCH/L registers.
3. Program the CRCXORH/L registers with the desired generator polynomial.
4. Program the DLEN<3:0> bits of the CRCCON1 register with the length of the data word - 1 (refer to [Example 13-1](#)). This determines how many times the shifter will shift into the accumulator for each data word.
5. Program the PLEN<3:0> bits of the CRCCON1 register with the length of the polynomial - 2 (refer to [Example 13-1](#)).
6. Determine whether shifting in trailing zeros is desired and set the ACCM bit of the CRCCON0 register appropriately.
7. Likewise, determine whether the MSb or LSb should be shifted first and write the SHIFTM bit of the CRCCON0 register appropriately.
8. Write the CRCGO bit of the CRCCON0 register to begin the shifting process.
- 9a. If manual SFR entry is used, monitor the FULL bit of the CRCCON0 register. When FULL = 0, another word of data can be written to the CRCDATH/L registers, keeping in mind that CRCDATH should be written first if the data has more than eight bits, as the shifter will begin upon the CRCDATL register being written.
- 9b. If the scanner is used, the scanner will automatically stuff words into the CRCDATH/L registers as needed, as long as the SCANGO bit is set.
- 10a. If using the Flash memory scanner, monitor the SCANIF (or the SCANGO bit) for the scanner to finish pushing information into the CRCDATA registers. After the scanner is completed, monitor the CRCIF (or the BUSY bit) to determine that the CRC has been completed and the check value can be read from the CRCACC registers. If both the interrupt flags are set (or both BUSY and SCANGO bits are cleared), the completed CRC calculation can be read from the CRCACCH/L registers.
- 10b. If manual entry is used, monitor the CRCIF (or BUSY bit) to determine when the CRCACC registers will hold the check value.

13.9 Program Memory Scan Configuration

If desired, the program memory scan module may be used in conjunction with the CRC module to perform a CRC calculation over a range of program memory addresses. In order to set up the scanner to work with the CRC you need to perform the following steps:

1. Set the Enable bit in both the CRCCON0 and SCANCON0 registers. If they get disabled, all internal states of the scanner and the CRC are reset (registers are unaffected).
2. Choose which memory access mode is to be used (see [Section 13.11 “Scanning Modes”](#)) and set the MODE bits of the SCANCON0 register appropriately.
3. Based on the memory access mode, set the INTM bits of the SCANCON0 register to the appropriate interrupt mode (see [Section 13.11.5 “Interrupt Interaction”](#))
4. Set the SCANLADRL/H/U and SCANHADRL/H/U registers with the beginning and ending locations in memory that are to be scanned.
5. The CRCGO bit must be set before setting the SCANGO bit. Setting the SCANGO bit starts the scan. Both CRCEN and CRCGO bits must be enabled to use the scanner. When either of these bits are disabled, the scan aborts and the INVALID bit SCANCON0 is set. The scanner will wait for the signal from the CRC that it is ready for the first Flash memory location, then begin loading data into the CRC. It will continue to do so until it either hits the configured end address or an address that is unimplemented on the device, at which point the SCANGO bit will clear, Scanner functions will cease, and the SCANIF interrupt will be triggered. Alternately, the SCANGO bit can be cleared in software if desired.

13.10 Scanner Interrupt

The scanner will trigger an interrupt when the SCANGO bit transitions from '1' to '0'. The SCANIF interrupt flag of PIR7 is set when the last memory location is reached and the data is entered into the CRCDATA registers. The SCANIF bit can only be cleared in software. The SCAN interrupt enable is the SCANIE bit of the PIE7 register.

13.11 Scanning Modes

The memory scanner can scan in four modes: Burst, Peek, Concurrent, and Triggered. These modes are controlled by the MODE bits of the SCANCON0 register. The four modes are summarized in [Table 13-2](#).

13.11.1 BURST MODE

When MODE = 01, the scanner is in Burst mode. In Burst mode, CPU operation is stalled beginning with the operation after the one that sets the SCANGO bit, and the scan begins, using the instruction clock to execute. The CPU is held in its current state until the scan stops. Note that because the CPU is not executing instructions, the SCANGO bit cannot be cleared in software, so the CPU will remain stalled until one of the hardware end-conditions occurs. Burst mode has the highest throughput for the scanner, but has the cost of stalling other execution while it occurs.

13.11.2 CONCURRENT MODE

When MODE = 00, the scanner is in Concurrent mode. Concurrent mode, like Burst mode, stalls the CPU while performing accesses of memory. However, while Burst mode stalls until all accesses are complete, Concurrent mode allows the CPU to execute in between access cycles.

13.11.3 TRIGGERED MODE

When MODE = 11, the scanner is in Triggered mode. Triggered mode behaves identically to Concurrent mode, except instead of beginning the scan immediately upon the SCANGO bit being set, it waits for a rising edge from a separate trigger clock, the source of which is determined by the SCANTRIG register.

13.11.4 PEEK MODE

When MODE = 10, the scanner is in Peek mode. Peek mode waits for an instruction cycle in which the CPU does not need to access the NVM (such as a branch instruction) and uses that cycle to do its own NVM access. This results in the lowest throughput for the NVM access (and can take a much longer time to complete a scan than the other modes), but does so without any impact on execution times, unlike the other modes.

TABLE 13-2: SUMMARY OF SCANNER MODES

| MODE<1:0> | | Description | | |
|-----------|------------|---|---------------------------|---|
| | | First Scan Access | CPU Operation | |
| 11 | Triggered | As soon as possible following a trigger | Stalled during NVM access | CPU resumes execution following each access |
| 10 | Peek | At the first dead cycle | Timing is unaffected | CPU continues execution following each access |
| 01 | Burst | As soon as possible | Stalled during NVM access | CPU suspended until scan completes |
| 00 | Concurrent | | | CPU resumes execution following each access |

13.11.5 INTERRUPT INTERACTION

The INTM bit of the SCANCON0 register controls the scanner's response to interrupts depending on which mode the NVM scanner is in, as described in [Table 13-3](#).

TABLE 13-3: SCAN INTERRUPT MODES

| INTM | MODE<1:0> | | |
|------|---|---|---------------------|
| | MODE == Burst | MODE == CONCURRENT or TRIGGERED | MODE == PEEK |
| 1 | Interrupt overrides SCANGO (to zero) to pause the burst and the interrupt handler executes at full speed; Scanner Burst resumes when interrupt completes. | Scanner suspended during interrupt response (SCANGO = 0); interrupt executes at full speed and scan resumes when the interrupt is complete. | This bit is ignored |
| 0 | Interrupts do not override SCANGO, and the scan (burst) operation will continue; interrupt response will be delayed until scan completes (latency will be increased). | Scanner accesses NVM during interrupt response. | This bit is ignored |

In general, if INTM = 0, the scanner will take precedence over the interrupt, resulting in decreased interrupt processing speed and/or increased interrupt response latency. If INTM = 1, the interrupt will take precedence and have a better speed, delaying the memory scan.

13.11.6 WWDT INTERACTION

Operation of the WWDT is not affected by scanner activity. Hence, 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 should be considered when performing memory scans with an application that also utilizes WWDT.

13.11.7 IN-CIRCUIT DEBUG (ICD) INTERACTION

The scanner freezes when an ICD halt occurs, and remains frozen until user-mode operation resumes. The debugger may inspect the SCANCON0 and SCANLADR registers to determine the state of the scan.

The ICD interaction with each operating mode is summarized in [Table 13-4](#).

TABLE 13-4: ICD AND SCANNER INTERACTIONS

| ICD Halt | Scanner Operating Mode | | |
|---|--|--|--|
| | Peek | Concurrent Triggered | Burst |
| External Halt | If scanner would peek an instruction that is not executed (because of ICD entry), the peek will occur after ICD exit, when the instruction executes. | If external halt is asserted during a scan cycle, the instruction (delayed by scan) may or may not execute before ICD entry, depending on external halt timing. | If external halt is asserted during the BSF (SCANCON.GO), ICD entry occurs, and the burst is delayed until ICD exit. Otherwise, the current NVM-access cycle will complete, and then the scanner will be interrupted for ICD entry. |
| | | If external halt is asserted during the cycle immediately prior to the scan cycle, both scan and instruction execution happen after the ICD exits. | If external halt is asserted during the burst, the burst is suspended and will resume with ICD exit. |
| Scan cycle occurs before ICD entry and instruction execution happens after the ICD exits. | | If PCPB (or single step) is on BSF (SCANCON.GO), the ICD is entered before execution; execution of the burst will occur at ICD exit, and the burst will run to completion. | |
| The instruction with the dataBP executes and ICD entry occurs immediately after. If scan is requested during that cycle, the scan cycle is postponed until the ICD exits. | | | |
| PC Breakpoint | | If a scan cycle is ready after the debug instruction is executed, the scan will read PFM and then the ICD is re-entered. | Note that the burst can be interrupted by an external halt. |
| Data Breakpoint | If scan would stall a SWBP, the scan cycle occurs and the ICD is entered. | If SWBP replaces BSF (SCANCON.GO), the ICD will be entered; instruction execution will occur at ICD exit (from ICDINSTR register), and the burst will run to completion. | |
| Single Step | | | |
| SWBP and ICDINST | | | |

13.11.8 PERIPHERAL MODULE DISABLE

Both the CRC and scanner module can be disabled individually by setting the CRCMD and SCANMD bits of the PMD0 register ([Register 7-1](#)). The SCANMD can be used to enable or disable to the scanner module only if the SCANE bit of Configuration Word 4 is set. If the SCANE bit is cleared, then the scanner module is not available for use and the SCANMD bit is ignored.

TABLE 13-5: SUMMARY OF REGISTERS ASSOCIATED WITH CRC

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on Page |
|-----------|-------------|-----------|-------------|---------|-----------|---------|-----------|---------|------------------|
| CRCACCH | ACC<15:8> | | | | | | | | 154 |
| CRCACCL | ACC<7:0> | | | | | | | | 155 |
| CRCCON0 | EN | GO | BUSY | ACCM | — | — | SHIFTM | FULL | 153 |
| CRCCON1 | DLEN<3:0> | | | | PLEN<3:0> | | | | 153 |
| CRCDATH | DATA<15:8> | | | | | | | | 154 |
| CRCDATL | DATA<7:0> | | | | | | | | 154 |
| CRCSHIFTH | SHIFT<15:8> | | | | | | | | 155 |
| CRCSHIFTL | SHIFT<7:0> | | | | | | | | 155 |
| CRCXORH | X<15:8> | | | | | | | | 156 |
| CRCXORL | X<7:1> | | | | | | | — | 156 |
| PMD0 | SYSCMD | FVRMD | HLVDMD | CRCMD | SCANMD | NVMMD | CLKRMD | IOCMD | 65 |
| SCANCON0 | SCANEN | SCANGO | BUSY | INVALID | INTM | — | MODE<1:0> | | 157 |
| SCANHADRU | — | — | HADR<21:16> | | | | | | 159 |
| SCANHADRH | HADR<15:8> | | | | | | | | 160 |
| SCANHADRL | HADR<7:0> | | | | | | | | 160 |
| SCANLADRU | — | — | LADR<21:16> | | | | | | 158 |
| SCANLADRH | LADR<15:8> | | | | | | | | 158 |
| SCANLADRL | LADR<7:0> | | | | | | | | 159 |
| SCANTRIG | — | — | — | — | TSEL<3:0> | | | | 161 |
| INTCON | GIE/GIEH | PEIE/GIEL | IPEN | — | INT3EDG | INT2EDG | INT1EDG | INT0EDG | 172 |
| PIR8 | SCANIF | CRCIF | NVMIF | — | — | — | — | CWG1IF | 182 |
| PIE8 | SCANIE | CRCIE | NVMIE | — | — | — | — | CWG1IE | 192 |
| IPR8 | SCANIP | CRCIP | NVMIP | — | — | — | — | CWG1IP | 202 |

Legend: — = unimplemented location, read as '0'. Shaded cells are not used for the CRC module.

14.0 INTERRUPTS

The PIC18(L)F6xK40 devices have multiple interrupt sources and an interrupt priority feature that allows most interrupt sources to be assigned a high or low priority level. The high priority interrupt vector is at 0008h and the low priority interrupt vector is at 0018h. A high priority interrupt event will interrupt a low priority interrupt that may be in progress.

The registers for controlling interrupt operation are:

- INTCON
- PIR1, PIR2, PIR3, PIR4, PIR5, PIR6, PIR7, PIR8, PIR9
- PIE1, PIE2, PIE3, PIE4, PIE5, PIE6, PIE7, PIE8, PIE9
- IPR1, IPR2, IPR3, IPR4, IPR5, IPR6, IPR7, IPR8, IPR9

It is recommended that the Microchip header files supplied with MPLAB® IDE be used for the symbolic bit names in these registers. This allows the assembler/compiler to automatically take care of the placement of these bits within the specified register.

In general, interrupt sources have three bits to control their operation. They are:

- **Flag bit** to indicate that an interrupt event occurred
- **Enable bit** that allows program execution to branch to the interrupt vector address when the flag bit is set
- **Priority bit** to select high priority or low priority

14.1 Mid-Range Compatibility

When the IPEN bit is cleared (default state), the interrupt priority feature is disabled and interrupts are compatible with PIC® microcontroller mid-range devices. In Compatibility mode, the interrupt priority bits of the IPRx registers have no effect. The PEIE/GIEL bit of the INTCON register is the global interrupt enable for the peripherals. The PEIE/GIEL bit disables only the peripheral interrupt sources and enables the peripheral interrupt sources when the GIE/GIEH bit is also set. The GIE/GIEH bit of the INTCON register is the global interrupt enable which enables all non-peripheral interrupt sources and disables all interrupt sources, including the peripherals. All interrupts branch to address 0008h in Compatibility mode.

14.2 Interrupt Priority

The interrupt priority feature is enabled by setting the IPEN bit of the INTCON register. When interrupt priority is enabled the GIE/GIEH and PEIE/GIEL Global Interrupt Enable bits of Compatibility mode are replaced by the GIEH high priority, and GIEL low priority, global interrupt enables. When the IPEN bit is set, the GEIH bit of the INTCON register enables all interrupts which have their associated bit in the IPRx register set. When the GEIH bit is cleared, then all interrupt sources including those selected as low priority in the IPRx register are disabled.

When both GIEH and GIEL bits are set, all interrupts selected as low priority sources are enabled.

A high priority interrupt will vector immediately to address 00 0008h and a low priority interrupt will vector to address 00 0018h.

14.3 Interrupt Response

When an interrupt is responded to, the Global Interrupt Enable bit is cleared to disable further interrupts. The GIE/GIEH bit is the Global Interrupt Enable when the IPEN bit is cleared. When the IPEN bit is set, enabling interrupt priority levels, the GIEH bit is the high priority Global Interrupt Enable and the GIEL bit is the low priority Global Interrupt Enable. High priority interrupt sources can interrupt a low priority interrupt. Low priority interrupts are not processed while high priority interrupts are in progress.

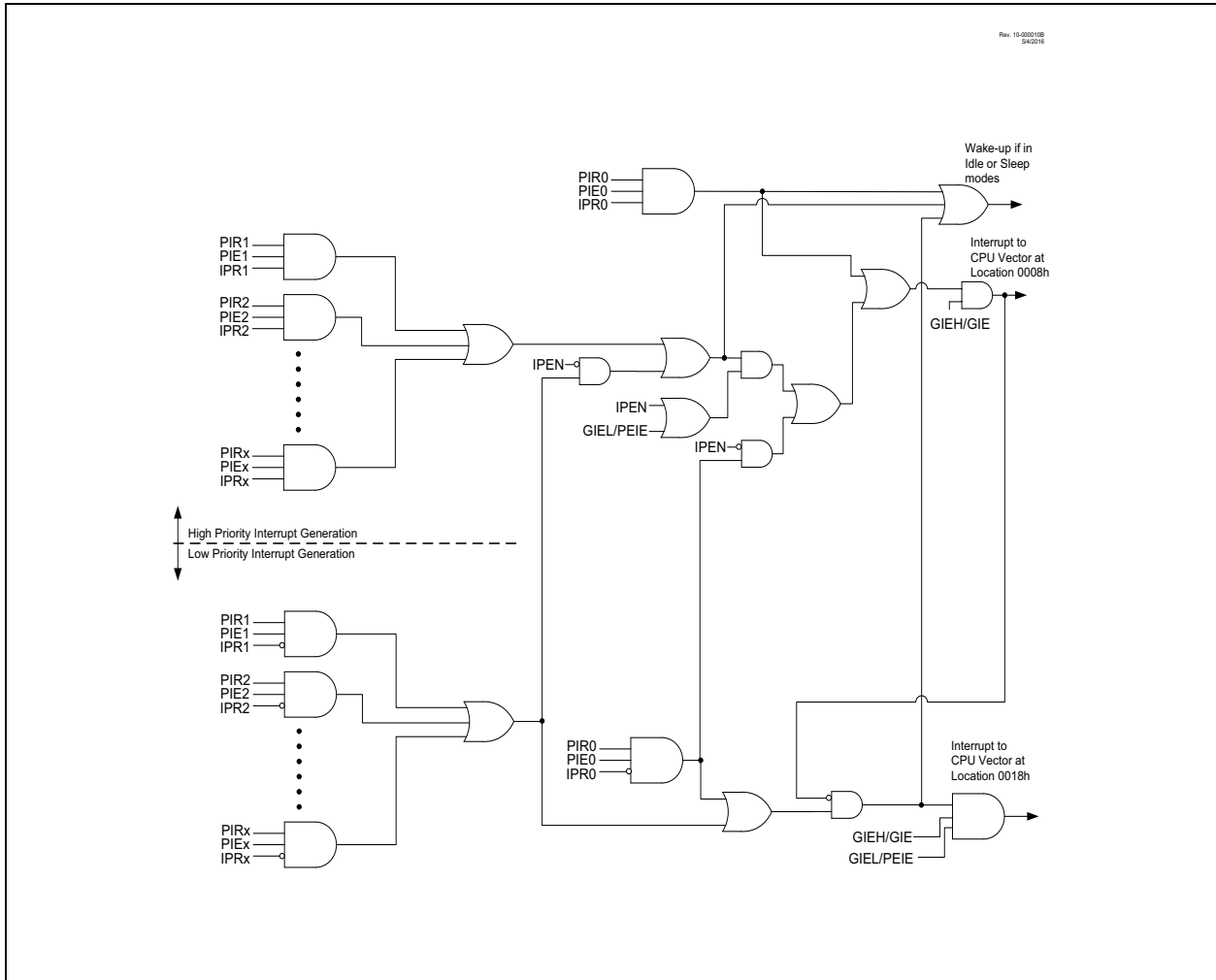
The return address is pushed onto the stack and the PC is loaded with the interrupt vector address (0008h or 0018h). Once in the Interrupt Service Routine, the source(s) of the interrupt can be determined by polling the interrupt flag bits in the INTCONx and PIRx registers. The interrupt flag bits must be cleared by software before re-enabling interrupts to avoid repeating the same interrupt.

The “return from interrupt” instruction, `RETFIE`, exits the interrupt routine and sets the GIE/GIEH bit (GIEH or GIEL if priority levels are used), which re-enables interrupts.

For external interrupt events, such as the INT pins or the Interrupt-on-change pins, the interrupt latency will be three to four instruction cycles. The exact latency is the same for one-cycle or two-cycle instructions. Individual interrupt flag bits are set, regardless of the status of their corresponding enable bits or the Global Interrupt Enable bit.

Note: Do not use the `MOVFF` instruction to modify any of the interrupt control registers while **any** interrupt is enabled. Doing so may cause erratic microcontroller behavior.

FIGURE 14-1: PIC18 INTERRUPT LOGIC



14.4 INTCON Registers

The INTCON registers are readable and writable registers, which contain various enable and priority bits.

14.5 PIR Registers

The PIR registers contain the individual flag bits for the peripheral interrupts. Due to the number of peripheral interrupt sources, there are 10 Peripheral Interrupt Request Flag registers (PIR0, PIR1, PIR2, PIR3, PIR4, PIR5, PIR6, PIR7, PIR8, and PIR9).

14.6 PIE Registers

The PIE registers contain the individual enable bits for the peripheral interrupts. Due to the number of peripheral interrupt sources, there are 10 Peripheral Interrupt Enable registers (PIE0, PIE1, PIE2, PIE3, PIE4, PIE5, PIE6, PIE7, PIE8 and PIE9). When IPEN = 0, the PEIE/GIEL bit must be set to enable any of these peripheral interrupts.

14.7 IPR Registers

The IPR registers contain the individual priority bits for the peripheral interrupts. Due to the number of peripheral interrupt sources, there are 10 Peripheral Interrupt Priority registers (IPR0, IPR1, IPR2, IPR3, IPR4 and IPR5, IPR6, IPR7, IPR8 and IPR9). Using the priority bits requires that the Interrupt Priority Enable (IPEN) bit be set.

14.8 Register Definitions: Interrupt Control

REGISTER 14-1: INTCON: INTERRUPT CONTROL REGISTER

| | | | | | | | |
|----------|-----------|---------|-----|---------|---------|---------|---------|
| R/W-0/0 | R/W-0/0 | R/W-0/0 | U-0 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 |
| GIE/GIEH | PEIE/GIEL | IPEN | — | INT3EDG | INT2EDG | INT1EDG | INT0EDG |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|-------------------|------------------|------------------------------------|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared |
| | | x = Bit is unknown |

- bit 7 **GIE/GIEH:** Global Interrupt Enable bit
If IPEN = 1:
 1 = Enables all unmasked interrupts and cleared by hardware for high-priority interrupts only
 0 = Disables all interrupts
If IPEN = 0:
 1 = Enables all unmasked interrupts and cleared by hardware for all interrupts
 0 = Disables all interrupts
- bit 6 **PEIE/GIEL:** Peripheral Interrupt Enable bit
If IPEN = 1:
 1 = Enables all low-priority interrupts and cleared by hardware for low-priority interrupts only
 0 = Disables all low-priority interrupts
If IPEN = 0:
 1 = Enables all unmasked peripheral interrupts
 0 = Disables all peripheral interrupts
- bit 5 **IPEN:** Interrupt Priority Enable bit
 1 = Enable priority levels on interrupts
 0 = Disable priority levels on interrupts
- bit 4 **Unimplemented:** Read as '0'
- bit 3 **INT3EDG:** External Interrupt 3 Edge Select bit
 1 = Interrupt on rising edge of INT3 pin
 0 = Interrupt on falling edge of INT3 pin
- bit 2 **INT2EDG:** External Interrupt 2 Edge Select bit
 1 = Interrupt on rising edge of INT2 pin
 0 = Interrupt on falling edge of INT2 pin
- bit 1 **INT1EDG:** External Interrupt 1 Edge Select bit
 1 = Interrupt on rising edge of INT1 pin
 0 = Interrupt on falling edge of INT1 pin
- bit 0 **INT0EDG:** External Interrupt 0 Edge Select bit
 1 = Interrupt on rising edge of INT0 pin
 0 = Interrupt on falling edge of INT0 pin

Note: Interrupt flag bits are set when an interrupt condition occurs, regardless of the state of its corresponding enable bit or the global enable bit. User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt. This feature allows for software polling.

REGISTER 14-2: PIR0: PERIPHERAL INTERRUPT REQUEST (FLAG) REGISTER 0

| U-0 | U-0 | R/W-0/0 | R-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|-------|-----|-----------------------|------------------------|-------------------------|-------------------------|-------------------------|-------------------------|
| — | — | TMR0IF ⁽¹⁾ | IOCIF ^(1,2) | INT3IF ^(1,3) | INT2IF ^(1,3) | INT1IF ^(1,3) | INT0IF ^(1,3) |
| bit 7 | | | | | | | bit 0 |

Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7-6 **Unimplemented:** Read as '0'

bit 5 **TMR0IF:** Timer0 Interrupt Flag bit⁽¹⁾

1 = TMR0 register has overflowed (must be cleared by software)

0 = TMR0 register has not overflowed

bit 4 **IOCIF:** Interrupt-on-Change Flag bit^(1,2)

1 = IOC event has occurred (must be cleared by software)

0 = IOC event has not occurred

bit 3 **INT3IF:** External Interrupt 3 Flag bit^(1,3)

1 = External Interrupt 3 has occurred

0 = External Interrupt 3 has not occurred

bit 2 **INT2IF:** External Interrupt 2 Flag bit^(1,3)

1 = External Interrupt 2 has occurred

0 = External Interrupt 2 has not occurred

bit 1 **INT1IF:** External Interrupt 1 Flag bit^(1,3)

1 = External Interrupt 1 has occurred

0 = External Interrupt 1 has not occurred

bit 0 **INT0IF:** External Interrupt 0 Flag bit^(1,3)

1 = External Interrupt 0 has occurred

0 = External Interrupt 0 has not occurred

Note 1: Interrupts are not disabled by the PEIE bit in the INTCON register.

Note 2: IOCIF is a read-only bit, to clear the interrupt condition, all bits in the IOCF register must be cleared.

Note 3: The external interrupt GPIO pin is selected by the INTPPS register.

REGISTER 14-3: PIR1: PERIPHERAL INTERRUPT REQUEST (FLAG) REGISTER 1

| | | | | | | | |
|---------|----------------------|-----|-----|-----|-----|---------|---------|
| R/W-0/0 | R/W-0/0 | U-0 | U-0 | U-0 | U-0 | R/W-0/0 | R/W-0/0 |
| OSCFIF | CSWIF ⁽¹⁾ | — | — | — | — | ADTIF | ADIF |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|-------------------|------------------|--|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared x = Bit is unknown |

- bit 7 **OSCFIF:** Oscillator Fail Interrupt Flag bit
 1 = Device oscillator failed, clock input has changed to HFINTOSC (must be cleared by software)
 0 = Device clock operating
- bit 6 **CSWIF:** Clock-Switch Interrupt Flag bit⁽¹⁾
 1 = New oscillator is ready for switch (must be cleared by software) (see [Figure 4-6](#) and [Figure 4-7](#))
 0 = New oscillator is not ready for switch or has not been started
- bit 5-2 **Unimplemented:** Read as '0'
- bit 1 **ADTIF:** ADC Threshold Interrupt Flag bit
 1 = ADC Threshold interrupt has occurred (must be cleared by software)
 0 = ADC Threshold event is not complete or has not been started
- bit 0 **ADIF:** ADC Interrupt Flag bit
 1 = An A/D conversion completed (must be cleared by software)
 0 = The A/D conversion is not complete or has not been started

Note 1: The CSWIF interrupt will not wake the system from Sleep. The system will sleep until another interrupt causes the wake-up.

REGISTER 14-4: PIR2: PERIPHERAL INTERRUPT REQUEST (FLAG) REGISTER 2

| | | | | | | | |
|---------|---------|-----|-----|-----|---------|---------|---------|
| R/W-0/0 | R/W-0/0 | U-0 | U-0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| HLVDIF | ZCDIF | — | — | — | C3IF | C2IF | C1IF |
| bit 7 | | | | | | bit 0 | |

Legend:

| | | |
|-------------------|------------------|------------------------------------|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared |
| | | x = Bit is unknown |

- bit 7 **HLVDIF:** HLVD Interrupt Flag bit
1 = HLVD interrupt event has occurred
0 = HLVD interrupt event has not occurred or has not been set up
- bit 6 **ZCDIF:** Zero-Cross Detect Interrupt Flag bit
1 = ZCD Output has changed (must be cleared in software)
0 = ZCD Output has not changed
- bit 5-3 **Unimplemented:** Read as '0'
- bit 2 **C3IF:** Comparator 3 Interrupt Flag bit
1 = Comparator C3 output has changed (must be cleared by software)
0 = Comparator C3 output has not changed
- bit 1 **C2IF:** Comparator 2 Interrupt Flag bit
1 = Comparator C2 output has changed (must be cleared by software)
0 = Comparator C2 output has not changed
- bit 0 **C1IF:** Comparator 1 Interrupt Flag bit
1 = Comparator C1 output has changed (must be cleared by software)
0 = Comparator C1 output has not changed

REGISTER 14-5: PIR3: PERIPHERAL INTERRUPT REQUEST (FLAG) REGISTER 3

| | | | | | | | |
|-------|-------|-------|-------|---------|---------|---------|---------|
| R-0/0 | R-0/0 | R-0/0 | R-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| RC2IF | TX2IF | RC1IF | TX1IF | BCL2IF | SSP2IF | BCL1IF | SSP1IF |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|-------------------|------------------|------------------------------------|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared |
| | | x = Bit is unknown |

- bit 7 **RC2IF:** EUSART2 Receive Interrupt Flag bit
 - 1 = The EUSART2 receive buffer, RC1REG, is full (cleared by reading RC2REG)
 - 0 = The EUSART2 receive buffer is empty
- bit 6 **TX2IF:** EUSART2 Transmit Interrupt Flag bit
 - 1 = The EUSART2 transmit buffer, TX2REG, is empty (cleared by writing TX2REG)
 - 0 = The EUSART2 transmit buffer is full
- bit 5 **RC1IF:** EUSART1 Receive Interrupt Flag bit
 - 1 = The EUSART1 receive buffer, RC1REG, is full (cleared by reading RC1REG)
 - 0 = The EUSART1 receive buffer is empty
- bit 4 **TX1IF:** EUSART1 Transmit Interrupt Flag bit
 - 1 = The EUSART1 transmit buffer, TX1REG, is empty (cleared by writing TX1REG)
 - 0 = The EUSART1 transmit buffer is full
- bit 3 **BCL2IF:** MSSP2 Bus Collision Interrupt Flag bit
 - 1 = A bus collision has occurred while the MSSP2 module configured in I²C master was transmitting (must be cleared in software)
 - 0 = No bus collision occurred
- bit 2 **SSP2IF:** Synchronous Serial Port 2 Interrupt Flag bit
 - 1 = The transmission/reception is complete (must be cleared in software)
 - 0 = Waiting to transmit/receive
- bit 1 **BCL1IF:** MSSP1 Bus Collision Interrupt Flag bit
 - 1 = A bus collision has occurred while the MSSP1 module configured in I²C master was transmitting (must be cleared in software)
 - 0 = No bus collision occurred
- bit 0 **SSP1IF:** Synchronous Serial Port 1 Interrupt Flag bit
 - 1 = The transmission/reception is complete (must be cleared in software)
 - 0 = Waiting to transmit/receive

REGISTER 14-6: PIR4: PERIPHERAL INTERRUPT REQUEST (FLAG) REGISTER 4

| | | | | | | | |
|-------|-----|-------|-------|-------|-------|-------|-------|
| U-0 | U-0 | R-0/0 | R-0/0 | R-0/0 | R-0/0 | R-0/0 | R-0/0 |
| — | — | RC5IF | TX5IF | RC4IF | TX4IF | RC3IF | TX3IF |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|-------------------|------------------|--|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared x = Bit is unknown |

bit 7-6 **Unimplemented:** Read as '0'

bit 5 **RC5IF:** EUSART5 Receive Interrupt Flag bit
 1 = The EUSART5 receive buffer, RC5REG, is full (cleared by reading RC5REG)
 0 = The EUSART5 receive buffer is empty

bit 4 **TX5IF:** EUSART5 Transmit Interrupt Flag bit
 1 = The EUSART5 transmit buffer, TX5REG, is empty (cleared by writing TX5REG)
 0 = The EUSART5 transmit buffer is full

bit 3 **RC4IF:** EUSART4 Receive Interrupt Flag bit
 1 = The EUSART4 receive buffer, RC4REG, is full (cleared by reading RC4REG)
 0 = The EUSART4 receive buffer is empty

bit 2 **TX4IF:** EUSART4 Transmit Interrupt Flag bit
 1 = The EUSART4 transmit buffer, TX4REG, is empty (cleared by writing TX4REG)
 0 = The EUSART4 transmit buffer is full

bit 1 **RC3IF:** EUSART3 Receive Interrupt Flag bit
 1 = The EUSART3 receive buffer, RC3REG, is full (cleared by reading RC3REG)
 0 = The EUSART3 receive buffer is empty

bit 0 **TX3IF:** EUSART3 Transmit Interrupt Flag bit
 1 = The EUSART3 transmit buffer, TX3REG, is empty (cleared by writing TX3REG)
 0 = The EUSART3 transmit buffer is full

REGISTER 14-7: PIR5: PERIPHERAL INTERRUPT REQUEST (FLAG) REGISTER 5

| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|---------|---------|---------|---------|---------|---------|---------|---------|
| TMR8IF | TMR7IF | TMR6IF | TMR5IF | TMR4IF | TMR3IF | TMR2IF | TMR1IF |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|-------------------|------------------|--|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared x = Bit is unknown |

- bit 7 **TMR8IF:** TMR8 to PR8 Match Interrupt Flag bit
1 = TMR8 to PR8 match occurred (must be cleared in software)
0 = No TMR8 to PR8 match occurred
- bit 6 **TMR7IF:** TMR7 to PR7 Match Interrupt Flag bit
1 = TMR7 to PR7 match occurred (must be cleared in software)
0 = No TMR7 to PR7 match occurred
- bit 5 **TMR6IF:** TMR6 to PR6 Match Interrupt Flag bit
1 = TMR6 to PR6 match occurred (must be cleared in software)
0 = No TMR6 to PR6 match occurred
- bit 4 **TMR5IF:** TMR5 to PR5 Match Interrupt Flag bit
1 = TMR5 to PR5 match occurred (must be cleared in software)
0 = No TMR5 to PR5 match occurred
- bit 3 **TMR4IF:** TMR4 to PR4 Match Interrupt Flag bit
1 = TMR4 to PR4 match occurred (must be cleared in software)
0 = No TMR4 to PR4 match occurred
- bit 2 **TMR3IF:** TMR3 to PR3 Match Interrupt Flag bit
1 = TMR3 to PR3 match occurred (must be cleared in software)
0 = No TMR3 to PR3 match occurred
- bit 1 **TMR2IF:** TMR2 to PR2 Match Interrupt Flag bit
1 = TMR2 to PR2 match occurred (must be cleared in software)
0 = No TMR2 to PR2 match occurred
- bit 0 **TMR1IF:** TMR1 to PR1 Match Interrupt Flag bit
1 = TMR1 to PR1 match occurred (must be cleared in software)
0 = No TMR1 to PR1 match occurred

REGISTER 14-8: PIR6: PERIPHERAL INTERRUPT REQUEST (FLAG) REGISTER 6

| | | | | | | | |
|-------|-----|-----|-----|---------|---------|---------|---------|
| U-0 | U-0 | U-0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| — | — | — | — | TMR7GIF | TMR5GIF | TMR3GIF | TMR1GIF |
| bit 7 | | | | | | | bit 0 |

Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7-4 **Unimplemented:** Read as '0'

bit 3 **TMR7GIF:** TMR7 Gate Interrupt Flag bit
 1 = TMR7 gate interrupt occurred (must be cleared in software)
 0 = No TMR7 gate occurred

bit 2 **TMR5GIF:** TMR5 Gate Interrupt Flag bit
 1 = TMR5 gate interrupt occurred (must be cleared in software)
 0 = No TMR5 gate occurred

bit 1 **TMR3GIF:** TMR3 Gate Interrupt Flag bit
 1 = TMR3 gate interrupt occurred (must be cleared in software)
 0 = No TMR3 gate occurred

bit 0 **TMR1GIF:** TMR1 Gate Interrupt Flag bit
 1 = TMR1 gate interrupt occurred (must be cleared in software)
 0 = No TMR1 gate occurred

REGISTER 14-9: PIR7: PERIPHERAL INTERRUPT REQUEST (FLAG) REGISTER 7

| | | | | | | | |
|-------|-----|-----|---------|---------|---------|---------|---------|
| U-0 | U-0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| — | — | — | CCP5IF | CCP4IF | CCP3IF | CCP2IF | CCP1IF |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|-------------------|------------------|------------------------------------|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared |
| | | x = Bit is unknown |

bit 7-5 **Unimplemented:** Read as '0'

bit 4 **CCP5IF:** ECCP5 Interrupt Flag bit

Capture mode:

- 1 = A TMR register capture occurred (must be cleared in software)
- 0 = No TMR register capture occurred

Compare mode:

- 1 = A TMR register compare match occurred (must be cleared in software)
- 0 = No TMR register compare match occurred

PWM mode:

Unused in PWM mode.

bit 3 **CCP4IF:** ECCP4 Interrupt Flag bit

Capture mode:

- 1 = A TMR register capture occurred (must be cleared in software)
- 0 = No TMR register capture occurred

Compare mode:

- 1 = A TMR register compare match occurred (must be cleared in software)
- 0 = No TMR register compare match occurred

PWM mode:

Unused in PWM mode.

bit 2 **CCP3IF:** ECCP3 Interrupt Flag bit

Capture mode:

- 1 = A TMR register capture occurred (must be cleared in software)
- 0 = No TMR register capture occurred

Compare mode:

- 1 = A TMR register compare match occurred (must be cleared in software)
- 0 = No TMR register compare match occurred

PWM mode:

Unused in PWM mode.

bit 1 **CCP2IF:** ECCP2 Interrupt Flag bit

Capture mode:

- 1 = A TMR register capture occurred (must be cleared in software)
- 0 = No TMR register capture occurred

Compare mode:

- 1 = A TMR register compare match occurred (must be cleared in software)
- 0 = No TMR register compare match occurred

PWM mode:

Unused in PWM mode.

REGISTER 14-9: PIR7: PERIPHERAL INTERRUPT REQUEST (FLAG) REGISTER 7 (CONTINUED)

bit 0

CCP1IF: ECCP1 Interrupt Flag bit

Capture mode:

1 = A TMR register capture occurred (must be cleared in software)

0 = No TMR register capture occurred

Compare mode:

1 = A TMR register compare match occurred (must be cleared in software)

0 = No TMR register compare match occurred

PWM mode:

Unused in PWM mode.

REGISTER 14-10: PIR8: PERIPHERAL INTERRUPT REQUEST (FLAG) REGISTER 8

| | | | | | | | |
|---------|---------|---------|-----|-----|-----|-----|---------|
| R/W-0/0 | R/W-0/0 | R/W-0/0 | U-0 | U-0 | U-0 | U-0 | R/W-0/0 |
| SCANIF | CRCIF | NVMIF | — | — | — | — | CWG1IF |
| bit 7 | | | | | | | bit 0 |

Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

- bit 7 **SCANIF:** Memory Scanner Interrupt Flag bit
 1 = Interrupt has occurred (must be cleared by software)
 0 = Interrupt event has not occurred
- bit 6 **CRCIF:** CRC Interrupt Flag bit
 1 = Interrupt has occurred (must be cleared by software)
 0 = Interrupt event has not occurred
- bit 5 **NVMIF:** NVM Interrupt Flag bit
 1 = Interrupt has occurred (must be cleared by software)
 0 = Interrupt event has not occurred
- bit 4-1 **Unimplemented:** Read as '0'
- bit 0 **CWG1IF:** CWG1 Interrupt Flag bit
 1 = Interrupt has occurred (must be cleared by software)
 0 = Interrupt event has not occurred

REGISTER 14-11: PIR9: PERIPHERAL INTERRUPT REQUEST (FLAG) REGISTER 9

| | | | | | | | |
|-------|-----|------------|------------|------------|------------|------------|------------|
| U-0 | U-0 | R/W/HS-0/0 | R/W/HS-0/0 | R/W/HS-0/0 | R/W/HS-0/0 | R/W/HS-0/0 | R/W/HS-0/0 |
| — | — | SMT2PWAIF | SMT2PRAIF | SMT2IF | SMT1PWAIF | SMT1PRAIF | SMT1IF |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | HS = Bit is set in hardware |

| | |
|---------|---|
| bit 7-6 | Unimplemented: Read as '0' |
| bit 5 | SMT2PWAIF: SMT2 Pulse Width Acquisition Interrupt Flag bit 1 = Interrupt has occurred (must be cleared by software) 0 = Interrupt event has not occurred |
| bit 4 | SMT2PRAIF: SMT2 Period Acquisition Interrupt Flag bit 1 = Interrupt has occurred (must be cleared by software) 0 = Interrupt event has not occurred |
| bit 3 | SMT2IF: SMT2 Interrupt Flag bit 1 = Interrupt has occurred (must be cleared by software) 0 = Interrupt event has not occurred |
| bit 2 | SMT1PWAIF: SMT1 Pulse Width Acquisition Interrupt Flag bit 1 = Interrupt has occurred (must be cleared by software) 0 = Interrupt event has not occurred |
| bit 1 | SMT1PRAIF: SMT1 Period Acquisition Interrupt Flag bit 1 = Interrupt has occurred (must be cleared by software) 0 = Interrupt event has not occurred |
| bit 0 | SMT1IF: SMT1 Interrupt Flag bit 1 = Interrupt has occurred (must be cleared by software) 0 = Interrupt event has not occurred |

REGISTER 14-12: PIE0: PERIPHERAL INTERRUPT ENABLE REGISTER 0

| | | | | | | | |
|-------|-----|-----------------------|---------------------|-----------------------|-----------------------|-----------------------|-----------------------|
| U-0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| — | — | TMR0IE ⁽¹⁾ | IOCE ⁽¹⁾ | INT3IE ⁽¹⁾ | INT2IE ⁽¹⁾ | INT1IE ⁽¹⁾ | INT0IE ⁽¹⁾ |
| bit 7 | | | | | | | bit 0 |

Legend: IE

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7-6 **Unimplemented:** Read as '0'

bit 5 **TMR0IE:** Timer0 Interrupt Enable bit⁽¹⁾

1 = Enabled

0 = Disabled

bit 4 **IOCE:** Interrupt-on-Change Enable bit⁽¹⁾

1 = Enabled

0 = Disabled

bit 3 **INT3IE:** External Interrupt 3 Enable bit⁽¹⁾

1 = Enabled

0 = Disabled

bit 2 **INT2IE:** External Interrupt 2 Enable bit⁽¹⁾

1 = Enabled

0 = Disabled

bit 1 **INT1IE:** External Interrupt 1 Enable bit⁽¹⁾

1 = Enabled

0 = Disabled

bit 0 **INT0IE:** External Interrupt 0 Enable bit⁽¹⁾

1 = Enabled

0 = Disabled

Note 1: PIR0 interrupts are not disabled by the PEIE bit in the INTCON register. are not disabled by the PEIE bit in the INTCON register.

PIC18(L)F67K40

REGISTER 14-13: PIE1: PERIPHERAL INTERRUPT ENABLE REGISTER 1

| | | | | | | | |
|---------|---------|-----|-----|-----|-----|---------|---------|
| R/W-0/0 | R/W-0/0 | U-0 | U-0 | U-0 | U-0 | R/W-0/0 | R/W-0/0 |
| OSCFIE | CSWIE | — | — | — | — | ADTIE | ADIE |
| bit 7 | | | | | | | bit 0 |

Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

- bit 7 **OSCFIE:** Oscillator Fail Interrupt Enable bit
 1 = Enabled
 0 = Disabled
- bit 6 **CSWIE:** Clock-Switch Interrupt Enable bit
 1 = Enabled
 0 = Disabled
- bit 5-2 **Unimplemented:** Read as '0'
- bit 1 **ADTIE:** ADC Threshold Interrupt Enable bit
 1 = Enabled
 0 = Disabled
- bit 0 **ADIE:** ADC Interrupt Enable bit
 1 = Enabled
 0 = Disabled

REGISTER 14-14: PIE2: PERIPHERAL INTERRUPT ENABLE REGISTER 2

| | | | | | | | |
|---------|---------|-----|-----|-----|---------|---------|---------|
| R/W-0/0 | R/W-0/0 | U-0 | U-0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| HLVDIE | ZCDIE | — | — | — | C3IE | C2IE | C1IE |
| bit 7 | | | | | | bit 0 | |

Legend:

| | | |
|-------------------|------------------|------------------------------------|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared |
| | | x = Bit is unknown |

- bit 7 **HLVDIE:** HLVD Interrupt Enable bit
1 = Enabled
0 = Disabled
- bit 6 **ZCDIE:** Zero-Cross Detect Interrupt Enable bit
1 = Enabled
0 = Disabled
- bit 5-3 **Unimplemented:** Read as '0'
- bit 2 **C3IE:** Comparator 3 Interrupt Enable bit
1 = Enabled
0 = Disabled
- bit 1 **C2IE:** Comparator 2 Interrupt Enable bit
1 = Enabled
0 = Disabled
- bit 0 **C1IE:** Comparator 1 Interrupt Enable bit
1 = Enabled
0 = Disabled

REGISTER 14-15: PIE3: PERIPHERAL INTERRUPT ENABLE REGISTER 3

| | | | | | | | |
|---------|---------|---------|---------|---------|---------|---------|---------|
| R/W-0/0 | R/W-0/0 | R/W-0/0 | R-/W0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| RC2IE | TX2IE | RC1IE | TX1IE | BCL2IE | SSP2IE | BCL1IE | SSP1IE |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|-------------------|------------------|--|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared x = Bit is unknown |

- bit 7 **RC2IE:** EUSART2 Receive Interrupt Enable bit
1 = Enabled
0 = Disabled
- bit 6 **TX2IE:** EUSART2 Transmit Interrupt Enable bit
1 = Enabled
0 = Disabled
- bit 5 **RC1IE:** EUSART1 Receive Interrupt Enable bit
1 = Enabled
0 = Disabled
- bit 4 **TX1IE:** EUSART1 Transmit Interrupt Enable bit
1 = Enabled
0 = Disabled
- bit 3 **BCL2IE:** MSSP2 Bus Collision Interrupt Enable bit
1 = Enabled
0 = Disabled
- bit 2 **SSP2IE:** Synchronous Serial Port 2 Interrupt Enable bit
1 = Enabled
0 = Disabled
- bit 1 **BCL1IE:** MSSP1 Bus Collision Interrupt Enable bit
1 = Enabled
0 = Disabled
- bit 0 **SSP1IE:** Synchronous Serial Port 1 Interrupt Enable bit
1 = Enabled
0 = Disabled

REGISTER 14-16: PIE4: PERIPHERAL INTERRUPT ENABLE REGISTER 4

| | | | | | | | |
|-------|-----|---------|---------|---------|---------|---------|---------|
| U-0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| — | — | RC5IE | TX5IE | RC4IE | TX4IE | RC3IE | TX3IE |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|-------------------|------------------|------------------------------------|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared |
| | | x = Bit is unknown |

- bit 7-6 **Unimplemented:** Read as '0'
- bit 5 **RC5IE:** EUSART5 Receive Interrupt Enable bit
1 = Enabled
0 = Disabled
- bit 4 **TX5IE:** EUSART5 Transmit Interrupt Enable bit
1 = Enabled
0 = Disabled
- bit 3 **RC4IE:** EUSART4 Receive Interrupt Enable bit
1 = Enabled
0 = Disabled
- bit 2 **TX4IE:** EUSART4 Transmit Interrupt Enable bit
1 = Enabled
0 = Disabled
- bit 1 **RC3IE:** EUSART3 Receive Interrupt Enable bit
1 = Enabled
0 = Disabled
- bit 0 **TX3IE:** EUSART3 Transmit Interrupt Enable bit
1 = Enabled
0 = Disabled

REGISTER 14-17: PIE5: PERIPHERAL INTERRUPT ENABLE REGISTER 5

| | | | | | | | |
|---------|---------|---------|---------|---------|---------|---------|---------|
| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| TMR8IE | TMR7IE | TMR6IE | TMR5IE | TMR4IE | TMR3IE | TMR2IE | TMR1IE |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|-------------------|------------------|--|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared x = Bit is unknown |

- bit 7 **TMR8IE:** TMR8 to PR8 Match Interrupt Enable bit
1 = Enabled
0 = Disabled
- bit 6 **TMR7IE:** TMR7 to PR7 Match Interrupt Enable bit
1 = Enabled
0 = Disabled
- bit 5 **TMR6IE:** TMR6 to PR6 Match Interrupt Enable bit
1 = Enabled
0 = Disabled
- bit 4 **TMR5IE:** TMR5 to PR5 Match Interrupt Enable bit
1 = Enabled
0 = Disabled
- bit 3 **TMR4IE:** TMR4 to PR4 Match Interrupt Enable bit
1 = Enabled
0 = Disabled
- bit 2 **TMR3IE:** TMR3 to PR3 Match Interrupt Enable bit
1 = Enabled
0 = Disabled
- bit 1 **TMR2IE:** TMR2 to PR2 Match Interrupt Enable bit
1 = Enabled
0 = Disabled
- bit 0 **TMR1IE:** TMR1 to PR1 Match Interrupt Enable bit
1 = Enabled
0 = Disabled

REGISTER 14-18: PIE6: PERIPHERAL INTERRUPT ENABLE REGISTER 6

| | | | | | | | |
|-------|-----|-----|-----|---------|---------|---------|---------|
| U-0 | U-0 | U-0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| — | — | — | — | TMR7GIE | TMR5GIE | TMR3GIE | TMR1GIE |
| bit 7 | | | | | | | bit 0 |

Legend:

R = Readable bit W = Writable bit U = Unimplemented bit, read as '0'
 -n = Value at POR '1' = Bit is set '0' = Bit is cleared x = Bit is unknown

- bit 7-4 **Unimplemented:** Read as '0'
- bit 3 **TMR7GIE:** TMR7 Gate Interrupt Enable bit
1 = Enabled
0 = Disabled
- bit 2 **TMR5GIE:** TMR5 Gate Interrupt Enable bit
1 = Enabled
0 = Disabled
- bit 1 **TMR3GIE:** TMR3 Gate Interrupt Enable bit
1 = Enabled
0 = Disabled
- bit 0 **TMR1GIE:** TMR1 Gate Interrupt Enable bit
1 = Enabled
0 = Disabled

REGISTER 14-19: PIE7: PERIPHERAL INTERRUPT ENABLE REGISTER 7

| | | | | | | | |
|-------|-----|-----|---------|---------|---------|---------|---------|
| U-0 | U-0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| — | — | — | CCP5IE | CCP4IE | CCP3IE | CCP2IE | CCP1IE |
| bit 7 | | | | | | | bit 0 |

Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

| | |
|---------|--|
| bit 7-5 | Unimplemented: Read as '0' |
| bit 4 | CCP5IE: ECCP5 Interrupt Enable bit 1 = Enabled 0 = Disabled |
| bit 3 | CCP4IE: ECCP4 Interrupt Enable bit 1 = Enabled 0 = Disabled |
| bit 2 | CCP3IE: ECCP3 Interrupt Enable bit 1 = Enabled 0 = Disabled |
| bit 1 | CCP2IE: ECCP2 Interrupt Enable bit 1 = Enabled 0 = Disabled |
| bit 0 | CCP1IE: ECCP1 Interrupt Enable bit 1 = Enabled 0 = Disabled |

REGISTER 14-20: PIE8: PERIPHERAL INTERRUPT ENABLE REGISTER 8

| | | | | | | | |
|---------|---------|---------|-----|-----|-----|-----|---------|
| R/W-0/0 | R/W-0/0 | R/W-0/0 | U-0 | U-0 | U-0 | U-0 | R/W-0/0 |
| SCANIE | CRCIE | NVMIE | — | — | — | — | CWG1IE |
| bit 7 | | | | | | | bit 0 |

Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7 **SCANIE:** Memory Scanner Interrupt Enable bit

1 = Enabled

0 = Disabled

bit 6 **CRCIE:** CRC Interrupt Enable bit

1 = Enabled

0 = Disabled

bit 5 **NVMIE:** NVM Interrupt Enable bit

1 = Enabled

0 = Disabled

bit 4-1 **Unimplemented:** Read as '0'

bit 0 **CWG1IE:** CWG1 Interrupt Enable bit

1 = Enabled

0 = Disabled

REGISTER 14-21: PIE9: PERIPHERAL INTERRUPT ENABLE REGISTER 9

| | | | | | | | |
|-------|-----|-----------|-----------|---------|-----------|-----------|---------|
| U-0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| — | — | SMT2PWAIE | SMT2PRAIE | SMT2IE | SMT1PWAIE | SMT1PRAIE | SMT1IE |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

- bit 7-6 **Unimplemented:** Read as '0'
- bit 5 **SMT2PWAIE:** SMT2 Pulse Width Acquisition Interrupt Enable bit
1 = Enabled
0 = Disabled
- bit 4 **SMT2PRAIE:** SMT2 Period Acquisition Interrupt Enable bit
1 = Enabled
0 = Disabled
- bit 3 **SMT2IE:** SMT2 Interrupt Enable bit
1 = Enabled
0 = Disabled
- bit 2 **SMT1PWAIE:** SMT1 Pulse Width Acquisition Interrupt Enable bit
1 = Enabled
0 = Disabled
- bit 1 **SMT1PRAIE:** SMT1 Period Acquisition Interrupt Enable bit
1 = Enabled
0 = Disabled
- bit 0 **SMT1IE:** SMT1 Interrupt Enable bit
1 = Enabled
0 = Disabled

REGISTER 14-22: IPR0: PERIPHERAL INTERRUPT PRIORITY REGISTER 0

| | | | | | | | |
|-------|-----|---------|---------|---------|---------|---------|---------|
| U-0 | U-0 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 |
| — | — | TMR0IP | IOCIP | INT3IP | INT2IP | INT1IP | INT0IP |
| bit 7 | | | | | | | bit 0 |

Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

| | |
|---------|---|
| bit 7-6 | Unimplemented: Read as '0' |
| bit 5 | TMR0IP: Timer0 Interrupt Priority bit 1 = High priority 0 = Low priority |
| bit 4 | IOCIP: Interrupt-on-Change Priority bit 1 = High priority 0 = Low priority |
| bit 3 | INT3IP: External Interrupt 3 Priority bit 1 = High priority 0 = Low priority |
| bit 2 | INT2IP: External Interrupt 2 Priority bit 1 = High priority 0 = Low priority |
| bit 1 | INT1IP: External Interrupt 1 Priority bit 1 = High priority 0 = Low priority |
| bit 0 | INT0IP: External Interrupt 0 Priority bit 1 = High priority 0 = Low priority |

REGISTER 14-23: IPR1: PERIPHERAL INTERRUPT PRIORITY REGISTER 1

| | | | | | | | |
|---------|---------|-----|-----|-----|-----|---------|---------|
| R/W-1/1 | R/W-1/1 | U-0 | U-0 | U-0 | U-0 | R/W-1/1 | R/W-1/1 |
| OSCFIP | CSWIP | — | — | — | — | ADTIP | ADIP |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|-------------------|------------------|------------------------------------|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared |
| | | x = Bit is unknown |

- bit 7 **OSCFIP:** Oscillator Fail Interrupt Priority bit
 1 = High priority
 0 = Low priority
- bit 6 **CSWIP:** Clock-Switch Interrupt Priority bit
 1 = High priority
 0 = Low priority
- bit 5-2 **Unimplemented:** Read as '0'
- bit 1 **ADTIP:** ADC Threshold Interrupt Priority bit
 1 = High priority
 0 = Low priority
- bit 0 **ADIP:** ADC Interrupt Priority bit
 1 = High priority
 0 = Low priority

REGISTER 14-24: IPR2: PERIPHERAL INTERRUPT PRIORITY REGISTER 2

| | | | | | | | |
|---------|---------|-----|-----|-----|---------|---------|---------|
| R/W-1/1 | R/W-1/1 | U-0 | U-0 | U-0 | R/W-1/1 | R/W-1/1 | R/W-1/1 |
| HLVDIP | ZCDIP | — | — | — | C3IP | C2IP | C1IP |
| bit 7 | | | | | | bit 0 | |

Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

- bit 7 **HLVDIP:** HLVD Interrupt Priority bit
 1 = High priority
 0 = Low priority
- bit 6 **ZCDIP:** Zero-Cross Detect Interrupt Priority bit
 1 = High priority
 0 = Low priority
- bit 5-3 **Unimplemented:** Read as '0'
- bit 2 **C3IP:** Comparator 3 Interrupt Priority bit
 1 = High priority
 0 = Low priority
- bit 1 **C2IP:** Comparator 2 Interrupt Priority bit
 1 = High priority
 0 = Low priority
- bit 0 **C1IP:** Comparator 1 Interrupt Priority bit
 1 = High priority
 0 = Low priority

REGISTER 14-25: IPR3: PERIPHERAL INTERRUPT PRIORITY REGISTER 3

| R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 |
|---------|---------|---------|---------|---------|---------|---------|---------|
| RC2IP | TX2IP | RC1IP | TX1IP | BCL2IP | SSP2IP | BCL1IP | SSP1IP |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|-------------------|------------------|--|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared x = Bit is unknown |

- bit 7 **RC2IP:** EUSART2 Receive Interrupt Priority bit
1 = High priority
0 = Low priority
- bit 6 **TX2IP:** EUSART2 Transmit Interrupt Priority bit
1 = High priority
0 = Low priority
- bit 5 **RC1IP:** EUSART1 Receive Interrupt Priority bit
1 = High priority
0 = Low priority
- bit 4 **TX1IP:** EUSART1 Transmit Interrupt Priority bit
1 = High priority
0 = Low priority
- bit 3 **BCL2IP:** MSSP2 Bus Collision Interrupt Priority bit
1 = High priority
0 = Low priority
- bit 2 **SSP2IP:** Synchronous Serial Port 2 Interrupt Priority bit
1 = High priority
0 = Low priority
- bit 1 **BCL1IP:** MSSP1 Bus Collision Interrupt Priority bit
1 = High priority
0 = Low priority
- bit 0 **SSP1IP:** Synchronous Serial Port 1 Interrupt Priority bit
1 = High priority
0 = Low priority

REGISTER 14-26: IPR4: PERIPHERAL INTERRUPT PRIORITY REGISTER 4

| | | | | | | | |
|-------|-----|---------|---------|---------|---------|---------|---------|
| U-0 | U-0 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 |
| — | — | RC5IP | TX5IP | RC4IP | TX4IP | RC3IP | TX3IP |
| bit 7 | | | | | | | bit 0 |

Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

- bit 7-6 **Unimplemented:** Read as '0'
- bit 5 **RC5IP:** EUSART5 Receive Interrupt Priority bit
 1 = High priority
 0 = Low priority
- bit 4 **TX5IP:** EUSART5 Transmit Interrupt Priority bit
 1 = High priority
 0 = Low priority
- bit 3 **RC4IP:** EUSART4 Receive Interrupt Priority bit
 1 = High priority
 0 = Low priority
- bit 2 **TX4IP:** EUSART4 Transmit Interrupt Priority bit
 1 = High priority
 0 = Low priority
- bit 1 **RC3IP:** EUSART3 Receive Interrupt Priority bit
 1 = High priority
 0 = Low priority
- bit 0 **TX3IP:** EUSART3 Transmit Interrupt Priority bit
 1 = High priority
 0 = Low priority

REGISTER 14-27: IPR5: PERIPHERAL INTERRUPT PRIORITY REGISTER 5

| R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 |
|---------|---------|---------|---------|---------|---------|---------|---------|
| TMR8IP | TMR7IP | TMR6IP | TMR5IP | TMR4IP | TMR3IP | TMR2IP | TMR1IP |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|-------------------|------------------|--|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared x = Bit is unknown |

- bit 7 **TMR8IP:** TMR8 to PR8 Match Interrupt Priority bit
1 = High priority
0 = Low priority
- bit 6 **TMR7IP:** TMR7 to PR7 Match Interrupt Priority bit
1 = High priority
0 = Low priority
- bit 5 **TMR6IP:** TMR6 to PR6 Match Interrupt Priority bit
1 = High priority
0 = Low priority
- bit 4 **TMR5IP:** TMR5 to PR5 Match Interrupt Priority bit
1 = High priority
0 = Low priority
- bit 3 **TMR4IP:** TMR4 to PR4 Match Interrupt Priority bit
1 = High priority
0 = Low priority
- bit 2 **TMR3IP:** TMR3 to PR3 Match Interrupt Priority bit
1 = High priority
0 = Low priority
- bit 1 **TMR2IP:** TMR2 to PR2 Match Interrupt Priority bit
1 = High priority
0 = Low priority
- bit 0 **TMR1IP:** TMR1 to PR1 Match Interrupt Priority bit
1 = High priority
0 = Low priority

REGISTER 14-28: IPR6: PERIPHERAL INTERRUPT PRIORITY REGISTER 6

| | | | | | | | |
|-------|-----|-----|-----|---------|---------|---------|---------|
| U-0 | U-0 | U-0 | U-0 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 |
| — | — | — | — | TMR7GIP | TMR5GIP | TMR3GIP | TMR1GIP |
| bit 7 | | | | | | | bit 0 |

Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

| | |
|---------|---|
| bit 7-4 | Unimplemented: Read as '0' |
| bit 3 | TMR7GIP: TMR7 Gate Interrupt Priority bit 1 = High priority 0 = Low priority |
| bit 2 | TMR5GIP: TMR5 Gate Interrupt Priority bit 1 = High priority 0 = Low priority |
| bit 1 | TMR3GIP: TMR3 Gate Interrupt Priority bit 1 = High priority 0 = Low priority |
| bit 0 | TMR1GIP: TMR1 Gate Interrupt Priority bit 1 = High priority 0 = Low priority |

REGISTER 14-29: IPR7: PERIPHERAL INTERRUPT PRIORITY REGISTER 7

| | | | | | | | |
|-------|-----|-----|---------|---------|---------|---------|---------|
| U-0 | U-0 | U-0 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 |
| — | — | — | CCP5IP | CCP4IP | CCP3IP | CCP2IP | CCP1IP |
| bit 7 | | | | | | | bit 0 |

Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

| | |
|---------|--|
| bit 7-5 | Unimplemented: Read as '0' |
| bit 4 | CCP5IP: ECCP5 Interrupt Priority bit 1 = High priority 0 = Low priority |
| bit 3 | CCP4IP: ECCP4 Interrupt Priority bit 1 = High priority 0 = Low priority |
| bit 2 | CCP3IP: ECCP3 Interrupt Priority bit 1 = High priority 0 = Low priority |
| bit 1 | CCP2IP: ECCP2 Interrupt Priority bit 1 = High priority 0 = Low priority |
| bit 0 | CCP1IP: ECCP1 Interrupt Priority bit 1 = High priority 0 = Low priority |

REGISTER 14-30: IPR8: PERIPHERAL INTERRUPT PRIORITY REGISTER 8

| | | | | | | | |
|---------|---------|---------|-----|-----|-----|-----|---------|
| R/W-1/1 | R/W-1/1 | R/W-1/1 | U-0 | U-0 | U-0 | U-0 | R/W-1/1 |
| SCANIP | CRCIP | NVMIP | — | — | — | — | CWG1IP |
| bit 7 | | | | | | | bit 0 |

Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7 **SCANIP:** Memory Scanner Interrupt Priority bit

1 = High priority

0 = Low priority

bit 6 **CRCIP:** CRC Interrupt Priority bit

1 = High priority

0 = Low priority

bit 5 **NVMIP:** NVM Interrupt Priority bit

1 = High priority

0 = Low priority

bit 4-1 **Unimplemented:** Read as '0'

bit 0 **CWG1IP:** CWG1 Interrupt Priority bit

1 = High priority

0 = Low priority

REGISTER 14-31: IPR9: PERIPHERAL INTERRUPT PRIORITY REGISTER 9

| U-0 | U-0 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 |
|-------|-----|-----------|-----------|---------|-----------|-----------|---------|
| — | — | SMT2PWAIP | SMT2PRAIP | SMT2IP | SMT1PWAIP | SMT1PRAIP | SMT1IP |
| bit 7 | | | | | | | bit 0 |

Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

| | |
|---------|--|
| bit 7-6 | Unimplemented: Read as '0' |
| bit 5 | SMT2PWAIP: SMT2 Pulse Width Acquisition Interrupt Priority bit 1 = High priority 0 = Low priority |
| bit 4 | SMT2PRAIP: SMT2 Period Acquisition Interrupt Priority bit 1 = High priority 0 = Low priority |
| bit 3 | SMT2IP: SMT2 Interrupt Priority bit 1 = High priority 0 = Low priority |
| bit 2 | SMT1PWAIP: SMT1 Pulse Width Acquisition Interrupt Priority bit 1 = High priority 0 = Low priority |
| bit 1 | SMT1PRAIP: SMT1 Period Acquisition Interrupt Priority bit 1 = High priority 0 = Low priority |
| bit 0 | SMT1IP: SMT1 Interrupt Priority bit 1 = High priority 0 = Low priority |

14.9 INTn Pin Interrupts

PIC18(L)F6xK40 devices have four external interrupt sources which can be assigned to any pin on PORTA and PORTB using PPS. The external interrupt sources are edge-triggered. If the corresponding INTxEDG bit in the INTCON0 register is set (= 1), the interrupt is triggered by a rising edge. If the bit is clear, the trigger is on the falling edge.

All external interrupts (INT0, INT1, INT2 and INT3) can wake-up the processor from Idle or Sleep modes if bit INTxE was set prior to going into those modes. If the Global Interrupt Enable bit, GIE/GIEH, is set, the processor will branch to the interrupt vector following wake-up.

Interrupt priority is determined by the value contained in the interrupt priority bits, INT0IP, INT1IP, INT2IP and INT3 of the IPR0 register.

14.10 TMR0 Interrupt

In 8-bit mode (which is the default), an overflow in the TMR0 register (FFh → 00h) will set flag bit, TMR0IF. In 16-bit mode, an overflow in the TMR0H:TMR0L register pair (FFFFh → 0000h) will set TMR0IF. The interrupt can be enabled/disabled by setting/clearing enable bit, TMR0IE of the PIE0 register. Interrupt priority for Timer0 is determined by the value contained in the interrupt priority bit, TMR0IP of the IPR0 register. See **Section 18.0 “Timer0 Module”** for further details on the Timer0 module.

14.11 Interrupt-on-Change

An input change on any port pins that support IOC sets Flag bit, IOCIF of the PIR0 register. The interrupt can be enabled/disabled by setting/clearing the enable bit, IOCIE of the PIE0 register. Pins must also be individually enabled in the IOCxP and IOCxN register. IOCIF is a read-only bit and the flag can be cleared by clearing the corresponding IOCxF registers. For more information refer to **Section 16.0 “Interrupt-on-Change”**.

14.12 Context Saving During Interrupts

During interrupts, the return PC address is saved on the stack. Additionally, the WREG, STATUS and BSR registers are saved on the fast return stack. If a fast return from interrupt is not used (see **Section 10.2.2 “Fast Register Stack”**), the user may need to save the WREG, STATUS and BSR registers on entry to the Interrupt Service Routine. Depending on the user’s application, other registers may also need to be saved. **Example 14-1** saves and restores the WREG, STATUS and BSR registers during an Interrupt Service Routine.

EXAMPLE 14-1: SAVING STATUS, WREG AND BSR REGISTERS IN RAM

```

MOVWF    W_TEMP                ; W_TEMP is in virtual bank
MOVFF    STATUS, STATUS_TEMP    ; STATUS_TEMP located anywhere
MOVFF    BSR, BSR_TEMP          ; BSR_TEMP located anywhere
;
; USER ISR CODE
;
MOVFF    BSR_TEMP, BSR          ; Restore BSR
MOVF     W_TEMP, W              ; Restore WREG
MOVFF    STATUS_TEMP, STATUS    ; Restore STATUS
    
```

TABLE 14-1: SUMMARY OF REGISTERS ASSOCIATED WITH INTERRUPTS

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on Page |
|--------|----------|-----------|-----------|-----------|---------|-----------|-----------|---------|------------------|
| INTCON | GIE/GIEH | PEIE/GIEL | IPEN | — | INT3EDG | INT2EDG | INT1EDG | INT0EDG | 172 |
| PIE0 | — | — | TMR0IE | IOCFIE | INT3IE | INT2IE | INT1IE | INT0IE | 184 |
| PIE1 | OSCFIE | CSWIE | — | — | — | — | ADTIE | ADIE | 185 |
| PIE2 | HLVDIE | ZCDIE | — | — | — | C3IE | C2IE | C1IE | 186 |
| PIE3 | RC2IE | TX2IE | RC1IE | TX1IE | BCL2IE | SSP2IE | BCL1IE | SSP1IE | 187 |
| PIE4 | — | — | RC5IE | TX5IE | RC4IE | TX4IE | RC3IE | TX3IE | 188 |
| PIE5 | TMR8IE | TMR7IE | TMR6IE | TMR5IE | TMR4IE | TMR3IE | TMR2IE | TMR1IE | 189 |
| PIE6 | — | — | — | — | TMR7GIE | TMR5GIE | TMR3GIE | TMR1GIE | 190 |
| PIE7 | — | — | — | CCP5IE | CCP4IE | CCP3IE | CCP2IE | CCP1IE | 191 |
| PIE8 | SCANIE | CRCIE | NVMIE | — | — | — | — | CWG1IE | 192 |
| PIE9 | — | — | SMT2PWAIE | SMT2PRAIE | SMT2IE | SMT1PWAIE | SMT1PRAIE | SMT1IE | 193 |
| PIR0 | — | — | TMR0IF | IOCFIF | INT3IF | INT2IF | INT1IF | INT0IF | 173 |
| PIR1 | OSCFIF | CSWIF | — | — | — | — | ADTIF | ADIF | 174 |
| PIR2 | HLVDIF | ZCDIF | — | — | — | C3IF | C2IF | C1IF | 175 |
| PIR3 | RC2IF | TX2IF | RC1IF | TX1IF | BCL2IF | SSP2IF | BCL1IF | SSP1IF | 176 |
| PIR4 | — | — | RC5IF | TX5IF | RC4IF | TX4IF | RC3IF | TX3IF | 177 |
| PIR5 | TMR8IF | TMR7IF | TMR6IF | TMR5IF | TMR4IF | TMR3IF | TMR2IF | TMR1IF | 178 |
| PIR6 | — | — | — | — | TMR7GIF | TMR5GIF | TMR3GIF | TMR1GIF | 179 |
| PIR7 | — | — | — | CCP5IF | CCP4IF | CCP3IF | CCP2IF | CCP1IF | 180 |
| PIR8 | SCANIF | CRCIF | NVMIF | — | — | — | — | CWG1IF | 182 |
| PIR9 | — | — | SMT2PWAIF | SMT2PRAIF | SMT2IF | SMT1PWAIF | SMT1PRAIF | SMT1IF | 183 |
| IPR0 | — | — | TMR0IP | IOCIP | INT3IP | INT2IP | INT1IP | INT0IP | 194 |
| IPR1 | OSCFIP | CSWIP | — | — | — | — | ADTIP | ADIP | 195 |
| IPR2 | HLVDIP | ZCDIP | — | — | — | C3IP | C2IP | C1IP | 196 |
| IPR3 | RC2IP | TX2IP | RC1IP | TX1IP | BCL2IP | SSP2IP | BCL1IP | SSP1IP | 197 |
| IPR4 | — | — | RC5IP | TX5IP | RC4IP | TX4IP | RC3IP | TX3IP | 198 |
| IPR5 | TMR8IP | TMR7IP | TMR6IP | TMR5IP | TMR4IP | TMR3IP | TMR2IP | TMR1IP | 199 |
| IPR6 | — | — | — | — | TMR7GIP | TMR5GIP | TMR3GIP | TMR1GIP | 200 |
| IPR7 | — | — | — | CCP5IP | CCP4IP | CCP3IP | CCP2IP | CCP1IP | 201 |
| IPR8 | SCANIP | CRCIP | NVMIP | — | — | — | — | CWG1IP | 202 |
| IPR9 | — | — | SMT2PWAIP | SMT2PRAIP | SMT2IP | SMT1PWAIP | SMT1PRAIP | SMT1IP | 203 |

Legend: — = unimplemented locations, read as '0'. Shaded bits are not used for Interrupts.

15.1 I/O Priorities

Each pin defaults to the PORT data latch after Reset. Other functions are selected with the peripheral pin select logic. See **Section 17.0 “Peripheral Pin Select (PPS) Module”** for more information.

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. Configuration bits
2. Analog outputs (disable the input buffers)
3. Analog inputs
4. Port inputs and outputs from PPS

15.2 PORTx Registers

In this section the generic names such as PORTx, LATx, TRISx, etc. can be associated with all ports.

15.2.1 DATA REGISTER

PORTx is an 8-bit wide, bidirectional port. The corresponding data direction register is TRISx ([Register 15-2](#)). Setting a TRISx bit ('1') will make the corresponding PORTx pin an input (i.e., disable the output driver). Clearing a TRISx bit ('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). [Example 15-1](#) shows how to initialize PORTx.

Reading the PORTx register ([Register 15-1](#)) 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, this value is modified and then written to the PORT data latch (LATx).

The PORT data latch LATx ([Register 15-3](#)) holds the output port data and contains the latest value of a LATx or PORTx write.

EXAMPLE 15-1: INITIALIZING PORTA

```

; This code example illustrates
; initializing the PORTA register. The
; other ports are initialized in the same
; manner.

BANKSEL PORTA      ;
CLRF   PORTA      ;Init PORTA
BANKSEL LATA       ;Data Latch
CLRF   LATA       ;
BANKSEL ANSELA    ;
CLRF   ANSELA     ;digital I/O
BANKSEL TRISA     ;
MOVLW  B'11111000' ;Set RA<7:3> as inputs
MOVWF  TRISA      ;and set RA<2:0> as
                          ;outputs
    
```

15.2.2 DIRECTION CONTROL

The TRISx register ([Register 15-2](#)) controls the PORTx pin output drivers, even when they are being used as analog inputs. The user should ensure the bits in the TRISx register are maintained set when using them as analog inputs. I/O pins configured as analog inputs always read '0'.

15.2.3 ANALOG CONTROL

The ANSELx register ([Register 15-4](#)) is used to configure the Input mode of an I/O pin to analog. Setting the appropriate ANSELx bit high will cause all digital reads on the pin to be read as '0' and allow analog functions on the pin to operate correctly.

The state of the ANSELx bits has no effect on digital 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 affected port.

Note: 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 initialized to '0' by user software.

15.2.4 OPEN-DRAIN CONTROL

The ODCONx register ([Register 15-6](#)) controls the open-drain feature of the port. Open-drain operation is independently selected for each pin. When an ODCONx bit is set, the corresponding port output becomes an open-drain driver capable of sinking current only. When an ODCONx bit is cleared, the corresponding port output pin is the standard push-pull drive capable of sourcing and sinking current.

Note: It is not necessary to set open-drain control when using the pin for I²C; the I²C module controls the pin and makes the pin open-drain.

15.2.5 SLEW RATE CONTROL

The SLRCONx register ([Register 15-7](#)) controls the slew rate option for each port pin. Slew rate for each port pin can be controlled independently. When an SLRCONx bit is set, the corresponding port pin drive is slew rate limited. When an SLRCONx bit is cleared, The corresponding port pin drive slews at the maximum rate possible.

15.2.6 INPUT THRESHOLD CONTROL

The INLVLx register ([Register 15-8](#)) 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. See [Table 37-8](#) for more information on threshold levels.

Note: Changing the input threshold selection should 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.

15.2.7 WEAK PULL-UP CONTROL

The WPUx register ([Register 15-5](#)) controls the individual weak pull-ups for each port pin.

15.2.8 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. Any individual pin can be configured to generate an interrupt. The interrupt-on-change module is present on all the pins of Ports B, C, E and on pin RG5. For further details about the IOC module refer to [Section 16.0 “Interrupt-on-Change”](#).

EXAMPLE 15-2: INITIALIZING PORTE

```
CLRF   PORTE   ; Initialize PORTE by
           ; clearing output
           ; data latches
CLRF   LATE    ; Alternate method
           ; to clear output
           ; data latches
CLRF   ANSELE  ; Configure analog pins
           ; for digital only
MOVLW  05h    ; Value used to
           ; initialize data
           ; direction
MOVWF  TRISE   ; Set RE<0> as input
           ; RE<1> as output
           ; RE<2> as input
```


15.3 Register Definitions: Port Control

REGISTER 15-1: PORTx: PORTx REGISTER⁽¹⁾

| | | | | | | | |
|---------|---------|---------|---------|---------|---------|---------|---------|
| R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u |
| Rx7 | Rx6 | Rx5 | Rx4 | Rx3 | Rx2 | Rx1 | Rx0 |
| bit 7 | | | | | | | bit 0 |

Legend:

R = Readable bit W = Writable bit U = Unimplemented bit, read as '0'
 '1' = Bit is set '0' = Bit is cleared x = Bit is unknown
 -n/n = Value at POR and BOR/Value at all other Resets

bit 7-0 **Rx<7:0>**: Rx7:Rx0 Port I/O Value bits
 1 = Port pin is $\geq V_{IH}$
 0 = Port pin is $\leq V_{IL}$

Note 1: Writes to PORTx are actually written to the corresponding LATx register.
 Reads from PORTx register return actual I/O pin values.

TABLE 15-1: PORT REGISTERS

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|-------|--------------------|--------------------|--------------------|-------|-------|-------|-------|-------|
| PORTA | RA7 | RA6 | RA5 | RA4 | RA3 | RA2 | RA1 | RA0 |
| PORTB | RB7 ⁽¹⁾ | RB6 ⁽¹⁾ | RB5 | RB4 | RB3 | RB2 | RB1 | RB0 |
| PORTC | RC7 | RC6 | RC5 | RC4 | RC3 | RC2 | RC1 | RC0 |
| PORTD | RD7 | RD6 | RD5 | RD4 | RD3 | RD2 | RD1 | RD0 |
| PORTE | RE7 | RE6 | RE5 | RE4 | RE3 | RE2 | RE1 | RE0 |
| PORTF | RF7 | RF6 | RF5 | RF4 | RF3 | RF2 | RF1 | RF0 |
| PORTG | RG7 | RG6 | RG5 ⁽²⁾ | RG4 | RG3 | RG2 | RG1 | RG0 |
| PORTH | — | — | — | — | RH3 | RH2 | RH1 | RH0 |

Note 1: Bits RB6 and RB7 read '1' while in Debug mode.
2: Bit PORTG5 is read-only, and will read '1' when MCLR = 1 (Master Clear enabled).

REGISTER 15-2: TRISx: TRI-STATE CONTROL REGISTER

| | | | | | | | |
|---------|---------|---------|---------|---------|---------|---------|---------|
| R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 |
| TRISx7 | TRISx6 | TRISx5 | TRISx4 | TRISx3 | TRISx2 | TRISx1 | TRISx0 |
| bit 7 | | | | | | | bit 0 |

Legend:

R = Readable bit W = Writable bit U = Unimplemented bit, read as '0'
 '1' = Bit is set '0' = Bit is cleared x = Bit is unknown
 -n/n = Value at POR and BOR/Value at all other Resets

bit 7-0 **TRISx<7:0>**: TRISx Port I/O Tri-state Control bits
 1 = Port output driver is disabled
 0 = Port output driver is enabled

TABLE 15-2: TRIS REGISTERS

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|-------|-----------------------|-----------------------|------------------|--------|--------|--------|--------|--------|
| TRISA | TRISA7 | TRISA6 | TRISA5 | TRISA4 | TRISA3 | TRISA2 | TRISA1 | TRISA0 |
| TRISB | TRISB7 ⁽¹⁾ | TRISB6 ⁽¹⁾ | TRISB5 | TRISB4 | TRISB3 | TRISB2 | TRISB1 | TRISB0 |
| TRISC | TRISC7 | TRISC6 | TRISC5 | TRISC4 | TRISC3 | TRISC2 | TRISC1 | TRISC0 |
| TRISD | TRISD7 | TRISD6 | TRISD5 | TRISD4 | TRISD3 | TRISD2 | TRISD1 | TRISD0 |
| TRISE | TRISE7 | TRISE6 | TRISE5 | TRISE4 | TRISE3 | TRISE2 | TRISE1 | TRISE0 |
| TRISF | TRISF7 | TRISF6 | TRISF5 | TRISF4 | TRISF3 | TRISF2 | TRISF1 | TRISF0 |
| TRISG | TRISG7 | TRISG6 | — ⁽²⁾ | TRISG4 | TRISG3 | TRISG2 | TRISG1 | TRISG0 |
| TRISH | — | — | — | — | TRISH3 | TRISH2 | TRISH1 | TRISH0 |

Note 1: Bits RB6 and RB7 read '1' while in Debug mode.

Note 2: Bit TRISG5 is read-only, and will read '1' always.

REGISTER 15-3: LATx: LATx REGISTER⁽¹⁾

| | | | | | | | |
|---------|---------|---------|---------|---------|---------|---------|---------|
| R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u |
| LATx7 | LATx6 | LATx5 | LATx4 | LATx3 | LATx2 | LATx1 | LATx0 |
| bit 7 | | | | | | | bit 0 |

Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

bit 7-0 **LATx<7:0>**: Rx7:Rx0 Output Latch Value bits

Note 1: Writes to LATx are equivalent with writes to the corresponding PORTx register. Reads from LATx register return register values, not I/O pin values.

TABLE 15-3: LAT REGISTERS

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|------|-------|-------|-------|-------|-------|-------|-------|-------|
| LATA | LATA7 | LATA6 | LATA5 | LATA4 | LATA3 | LATA2 | LATA1 | LATA0 |
| LATB | LATB7 | LATB6 | LATB5 | LATB4 | LATB3 | LATB2 | LATB1 | LATB0 |
| LATC | LATC7 | LATC6 | LATC5 | LATC4 | LATC3 | LATC2 | LATC1 | LATC0 |
| LATD | LATD7 | LATD6 | LATD5 | LATD4 | LATD3 | LATD2 | LATD1 | LATD0 |
| LATE | LATE7 | LATE6 | LATE5 | LATE4 | LATE3 | LATE2 | LATE1 | LATE0 |
| LATF | LATF7 | LATF6 | LATF5 | LATF4 | LATF3 | LATF2 | LATF1 | LATF0 |
| LATG | LATG7 | LATG6 | — | LATG4 | LATG3 | LATG2 | LATG1 | LATG0 |
| LATH | — | — | — | — | LATH3 | LATH2 | LATH1 | LATH0 |

REGISTER 15-4: ANSELx: ANALOG SELECT REGISTER

| | | | | | | | |
|---------|---------|---------|---------|---------|---------|---------|---------|
| R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 |
| ANSELx7 | ANSELx6 | ANSELx5 | ANSELx4 | ANSELx3 | ANSELx2 | ANSELx1 | ANSELx0 |
| bit 7 | | | | | | | bit 0 |

Legend:

R = Readable bit W = Writable bit U = Unimplemented bit, read as '0'
 '1' = Bit is set '0' = Bit is cleared x = Bit is unknown
 -n/n = Value at POR and BOR/Value at all other Resets

bit 7-0 **ANSELx<7:0>**: Analog Select on Pins Rx<7:0>
 1 = Digital Input buffers are disabled.
 0 = ST and TTL input devices are enabled

TABLE 15-4: ANALOG SELECT PORT REGISTERS

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|--------|---------|---------|---------|---------|---------|---------|---------|---------|
| ANSELA | ANSELA7 | ANSELA6 | ANSELA5 | ANSELA4 | ANSELA3 | ANSELA2 | ANSELA1 | ANSELA0 |
| ANSELB | ANSELB7 | ANSELB6 | ANSELB5 | ANSELB4 | ANSELB3 | ANSELB2 | ANSELB1 | ANSELB0 |
| ANSELD | ANSELD7 | ANSELD6 | ANSELD5 | ANSELD4 | ANSELD3 | ANSELD2 | ANSELD1 | ANSELD0 |
| ANSELE | ANSELE7 | ANSELE6 | ANSELE5 | ANSELE4 | ANSELE3 | ANSELE2 | ANSELE1 | ANSELE0 |
| ANSELF | ANSELF7 | ANSELF6 | ANSELF5 | ANSELF4 | ANSELF3 | ANSELF2 | ANSELF1 | ANSELF0 |
| ANSELG | ANSELG7 | ANSELG6 | — | ANSELG4 | ANSELG3 | ANSELG2 | ANSELG1 | ANSELG0 |

REGISTER 15-5: WPUx: WEAK PULL-UP REGISTER

| | | | | | | | |
|---------|---------|---------|---------|---------|---------|---------|---------|
| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| WPUx7 | WPUx6 | WPUx5 | WPUx4 | WPUx3 | WPUx2 | WPUx1 | WPUx0 |
| bit 7 | | | | | | | bit 0 |

Legend:

R = Readable bit W = Writable bit U = Unimplemented bit, read as '0'
 '1' = Bit is set '0' = Bit is cleared x = Bit is unknown
 -n/n = Value at POR and BOR/Value at all other Resets

bit 7-0 **WPUx<7:0>**: Weak Pull-up PORTx Control bits
 1 = Weak Pull-up enabled
 0 = Weak Pull-up disabled

TABLE 15-5: WEAK PULL-UP PORT REGISTERS

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|------|-------|-------|----------------------|-------|-------|-------|-------|-------|
| WPUA | WPUA7 | WPUA6 | WPUA5 | WPUA4 | WPUA3 | WPUA2 | WPUA1 | WPUA0 |
| WPUB | WPUB7 | WPUB6 | WPUB5 | WPUB4 | WPUB3 | WPUB2 | WPUB1 | WPUB0 |
| WPUC | WPUC7 | WPUC6 | WPUC5 | WPUC4 | WPUC3 | WPUC2 | WPUC1 | WPUC0 |
| WPUD | WPUD7 | WPUD6 | WPUD5 | WPUD4 | WPUD3 | WPUD2 | WPUD1 | WPUD0 |
| WPUE | WPUE7 | WPUE6 | WPUE5 | WPUE4 | WPUE3 | WPUE2 | WPUE1 | WPUE0 |
| WPUF | WPUF7 | WPUF6 | WPUF5 | WPUF4 | WPUF3 | WPUF2 | WPUF1 | WPUF0 |
| WPUG | WPUG7 | WPUG6 | WPUG5 ⁽¹⁾ | WPUG4 | WPUG3 | WPUG2 | WPUG1 | WPUG0 |
| WPUH | — | — | — | — | WPUH3 | WPUH2 | WPUH1 | WPUH0 |

Note 1: If MCLRE = 1, the weak pull-up in RG5 is always enabled; bit WPUG5 is not affected and ignored.

REGISTER 15-6: ODCONx: OPEN-DRAIN CONTROL REGISTER

| | | | | | | | |
|---------|---------|---------|---------|---------|---------|---------|---------|
| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| ODCx7 | ODCx6 | ODCx5 | ODCx4 | ODCx3 | ODCx2 | ODCx1 | ODCx0 |
| bit 7 | | | | | | | bit 0 |

Legend:

R = Readable bit W = Writable bit U = Unimplemented bit, read as '0'
 '1' = Bit is set '0' = Bit is cleared x = Bit is unknown
 -n/n = Value at POR and BOR/Value at all other Resets

bit 7-0 **ODCx<7:0>**: Open-Drain Configuration on Pins Rx<7:0>
 1 = Output drives only low-going signals (sink current only)
 0 = Output drives both high-going and low-going signals (source and sink current)

TABLE 15-6: OPEN-DRAIN CONTROL REGISTERS

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|--------|---------|---------|---------|---------|---------|---------|---------|---------|
| ODCONA | ODCONA7 | ODCONA6 | ODCONA5 | ODCONA4 | ODCONA3 | ODCONA2 | ODCONA1 | ODCONA0 |
| ODCONB | ODCONB7 | ODCONB6 | ODCONB5 | ODCONB4 | ODCONB3 | ODCONB2 | ODCONB1 | ODCONB0 |
| ODCONC | ODCONC7 | ODCONC6 | ODCONC5 | ODCONC4 | ODCONC3 | ODCONC2 | ODCONC1 | ODCONC0 |
| ODCOND | ODCOND7 | ODCOND6 | ODCOND5 | ODCOND4 | ODCOND3 | ODCOND2 | ODCOND1 | ODCOND0 |
| ODCONE | ODCONE7 | ODCONE6 | ODCONE5 | ODCONE4 | ODCONE3 | ODCONE2 | ODCONE1 | ODCONE0 |
| ODCONF | ODCONF7 | ODCONF6 | ODCONF5 | ODCONF4 | ODCONF3 | ODCONF2 | ODCONF1 | ODCONF0 |
| ODCONG | ODCONG7 | ODCONG6 | — | ODCONG4 | ODCONG3 | ODCONG2 | ODCONG1 | ODCONG0 |
| ODCONH | — | — | — | — | ODCONH3 | ODCONH2 | ODCONH1 | ODCONH0 |

REGISTER 15-7: SLRCONx: SLEW RATE CONTROL REGISTER

| | | | | | | | |
|---------|---------|---------|---------|---------|---------|---------|---------|
| R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 |
| SLRx7 | SLRx6 | SLRx5 | SLRx4 | SLRx3 | SLRx2 | SLRx1 | SLRx0 |
| bit 7 | | | | | | | bit 0 |

Legend:

R = Readable bit W = Writable bit U = Unimplemented bit, read as '0'
 '1' = Bit is set '0' = Bit is cleared x = Bit is unknown
 -n/n = Value at POR and BOR/Value at all other Resets

bit 7-0 **SLRx<7:0>**: Slew Rate Control on Pins Rx<7:0>, respectively
 1 = Port pin slew rate is limited
 0 = Port pin slews at maximum rate

TABLE 15-7: SLEW RATE CONTROL REGISTERS

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---------|----------|----------|----------|----------|----------|----------|----------|----------|
| SLRCONA | SLRCONA7 | SLRCONA6 | SLRCONA5 | SLRCONA4 | SLRCONA3 | SLRCONA2 | SLRCONA1 | SLRCONA0 |
| SLRCONB | SLRCONB7 | SLRCONB6 | SLRCONB5 | SLRCONB4 | SLRCONB3 | SLRCONB2 | SLRCONB1 | SLRCONB0 |
| SLRCONC | SLRCONC7 | SLRCONC6 | SLRCONC5 | SLRCONC4 | SLRCONC3 | SLRCONC2 | SLRCONC1 | SLRCONC0 |
| SLRCOND | SLRCOND7 | SLRCOND6 | SLRCOND5 | SLRCOND4 | SLRCOND3 | SLRCOND2 | SLRCOND1 | SLRCOND0 |
| SLRCONE | SLRCONE7 | SLRCONE6 | SLRCONE5 | SLRCONE4 | SLRCONE3 | SLRCONE2 | SLRCONE1 | SLRCONE0 |
| SLRCONF | SLRCONF7 | SLRCONF6 | SLRCONF5 | SLRCONF4 | SLRCONF3 | SLRCONF2 | SLRCONF1 | SLRCONF0 |
| SLRCONG | SLRCONG7 | SLRCONG6 | — | SLRCONG4 | SLRCONG3 | SLRCONG2 | SLRCONG1 | SLRCONG0 |
| SLRCONH | — | — | — | — | SLRCONH3 | SLRCONH2 | SLRCONH1 | SLRCONH0 |

REGISTER 15-8: INLVLx: INPUT LEVEL CONTROL REGISTER

| | | | | | | | |
|---------|---------|---------|---------|---------|---------|---------|---------|
| R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 |
| INLVLx7 | INLVLx6 | INLVLx5 | INLVLx4 | INLVLx3 | INLVLx2 | INLVLx1 | INLVLx0 |
| bit 7 | | | | | | | bit 0 |

Legend:

R = Readable bit W = Writable bit U = Unimplemented bit, read as '0'
 '1' = Bit is set '0' = Bit is cleared x = Bit is unknown
 -n/n = Value at POR and BOR/Value at all other Resets

bit 7-0 **INLVLx<7:0>**: Input Level Select on Pins Rx<7:0>, respectively
 1 = ST input used for port reads and interrupt-on-change
 0 = TTL input used for port reads and interrupt-on-change

TABLE 15-8: INPUT LEVEL PORT REGISTERS

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|--------|---------|---------|------------------------|------------------------|------------------------|------------------------|------------------------|---------|
| INLVLA | INLVLA7 | INLVLA6 | INLVLA5 | INLVLA4 | INLVLA3 | INLVLA2 | INLVLA1 | INLVLA0 |
| INVLVB | INVLVB7 | INVLVB6 | INVLVB5 | INVLVB4 | INVLVB3 | INVLVB2 ⁽¹⁾ | INVLVB1 ⁽¹⁾ | INVLVB0 |
| INLVLC | INLVLC7 | INLVLC6 | INLVLC5 | INLVLC4 ⁽¹⁾ | INLVLC3 ⁽¹⁾ | INLVLC2 | INLVLC1 | INLVLC0 |
| INLVLD | INLVLD7 | INLVLD6 | INLVLD5 | INLVLD4 | INLVLD3 | INLVLD2 | INLVLD1 | INLVLD0 |
| INLVLE | INLVLE7 | INLVLE6 | INLVLE5 | INLVLE4 | INLVLE3 | INLVLE2 | INLVLE1 | INLVLE0 |
| INLVLF | INLVLF7 | INLVLF6 | INLVLF5 | INLVLF4 | INLVLF3 | INLVLF2 | INLVLF1 | INLVLF0 |
| INVLVG | INVLVG7 | INVLVG6 | INVLVG5 ⁽²⁾ | INVLVG4 | INVLVG3 | INVLVG2 | INVLVG1 | INVLVG0 |
| INLVLH | — | — | — | — | INLVLH3 | INLVLH2 | INLVLH1 | INLVLH0 |

Note 1: Pins read the I²C ST inputs when MSSP inputs select these pins, and I²C mode is enabled.
Note 2: The state of this bit is ignored when MCLRE = 1.

16.0 INTERRUPT-ON-CHANGE

All the pins of PORTB, PORTC, PORTE and pin RG5 of PORTG can be configured to operate as Interrupt-on-Change (IOC) pins on PIC18(L)F6xK40 family devices. 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. The interrupt-on-change module has the following features:

- Interrupt-on-Change enable (Master Switch)
- Individual pin configuration
- Rising and falling edge detection
- Individual pin interrupt flags

Figure 16-1 is a block diagram of the IOC module.

16.1 Enabling the Module

To allow individual port pins to generate an interrupt, the IOCIE bit of the PIE0 register must be set. If the IOCIE bit is disabled, the edge detection on the pin will still occur, but an interrupt will not be generated.

16.2 Individual Pin Configuration

For each port pin, a rising edge detector and a falling edge detector are present. To enable a pin to detect a rising edge, the associated bit of the IOCxP register is set. To enable a pin to detect a falling edge, the associated bit of the IOCxN register is set.

A pin can be configured to detect rising and falling edges simultaneously by setting both associated bits of the IOCxP and IOCxN registers, respectively.

16.3 Interrupt Flags

The IOCBFx, IOCCFx, IOCEFx and IOCGF5 bits located in the IOCBF, IOCCF, IOCEF and IOCGF registers respectively, are status flags that correspond to the interrupt-on-change pins of the associated 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 of the PIR0 register reflects the status of all IOCBFx, IOCCFx, IOCEFx and IOCGF5 bits.

16.4 Clearing Interrupt Flags

The individual status flags,(IOCBFx, IOCCFx, IOCEFx and IOCGF5 bits), can 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.

In order to ensure that no detected edge is lost while clearing flags, only AND operations masking out known changed bits should be performed. The following sequence is an example of what should be performed.

EXAMPLE 16-1: CLEARING INTERRUPT FLAGS (PORTB EXAMPLE)

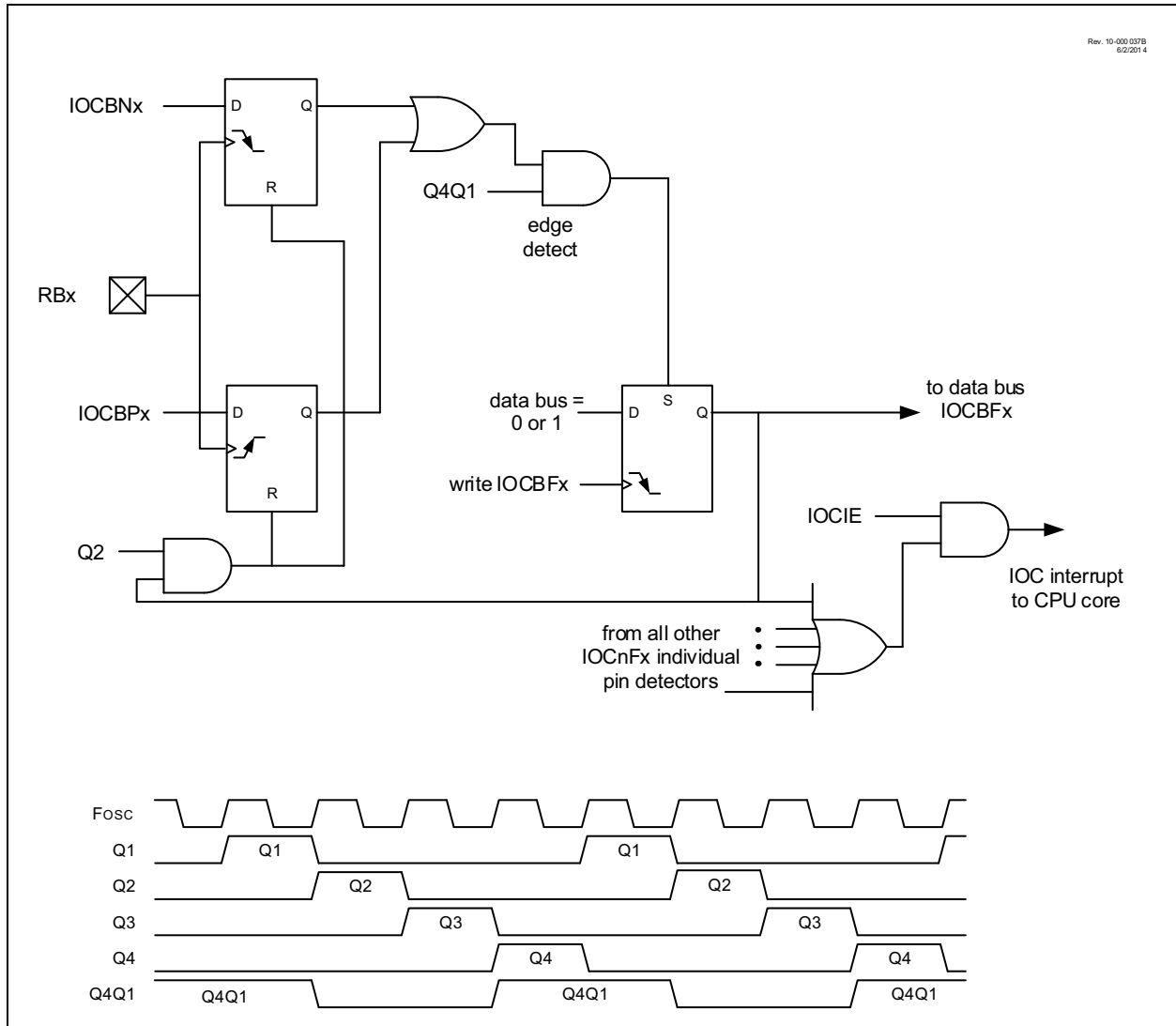
```
MOVLW    0xff
XORWF    IOCBF, W
ANDWF    IOCBF, F
```

16.5 Operation in Sleep

The interrupt-on-change interrupt sequence 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.

FIGURE 16-1: INTERRUPT-ON-CHANGE BLOCK DIAGRAM (PORTB EXAMPLE)



16.6 Register Definitions: Interrupt-on-Change Control

REGISTER 16-1: IOCxP: INTERRUPT-ON-CHANGE POSITIVE EDGE REGISTER EXAMPLE

| | | | | | | | |
|---------|---------|---------|---------|---------|---------|---------|---------|
| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| IOCxP7 | IOCxP6 | IOCxP5 | IOCxP4 | IOCxP3 | IOCxP2 | IOCxP1 | IOCxP0 |
| bit 7 | | | | | | | bit 0 |

Legend:

R = Readable bit W = Writable bit U = Unimplemented bit, read as '0'
u = Bit is unchanged x = Bit is unknown -n/n = Value at POR and BOR/Value at all other Resets
'1' = Bit is set '0' = Bit is cleared

bit 7-0 **IOCxP<7:0>**: Interrupt-on-Change Positive Edge Enable bits
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 = Interrupt-on-Change disabled for the associated pin.

REGISTER 16-2: IOCxN: INTERRUPT-ON-CHANGE NEGATIVE EDGE REGISTER EXAMPLE

| | | | | | | | |
|---------|---------|---------|---------|---------|---------|---------|---------|
| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| IOCxN7 | IOCxN6 | IOCxN5 | IOCxN4 | IOCxN3 | IOCxN2 | IOCxN1 | IOCxN0 |
| bit 7 | | | | | | | bit 0 |

Legend:

R = Readable bit W = Writable bit U = Unimplemented bit, read as '0'
u = Bit is unchanged x = Bit is unknown -n/n = Value at POR and BOR/Value at all other Resets
'1' = Bit is set '0' = Bit is cleared

bit 7-0 **IOCxN<7:0>**: Interrupt-on-Change Negative Edge Enable bits
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 = Interrupt-on-Change disabled for the associated pin

REGISTER 16-3: IOxF: INTERRUPT-ON-CHANGE FLAG REGISTER EXAMPLE

| | | | | | | | |
|------------|------------|------------|------------|------------|------------|------------|------------|
| R/W/HS-0/0 | R/W/HS-0/0 | R/W/HS-0/0 | R/W/HS-0/0 | R/W/HS-0/0 | R/W/HS-0/0 | R/W/HS-0/0 | R/W/HS-0/0 |
| IOxF7 | IOxF6 | IOxF5 | IOxF4 | IOxF3 | IOxF2 | IOxF1 | IOxF0 |
| bit 7 | | | | | | | bit 0 |

Legend:

R = Readable bit W = Writable bit U = Unimplemented bit, read as '0'
u = Bit is unchanged x = Bit is unknown -n/n = Value at POR and BOR/Value at all other Resets
'1' = Bit is set '0' = Bit is cleared HS - Bit is set in hardware

bit 7-0 **IOxF<7:0>**: Interrupt-on-Change Flag bits
1 = A enabled change was detected on the associated pin. Set when IOCP[n] = 1 and a positive edge was detected on the IOCP pin, or when IOCN[n] = 1 and a negative edge was detected on the IOCN pin
0 = No change was detected, or the user cleared the detected change

TABLE 16-1: IOC REGISTERS

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|-------|--------|--------|-----------------------|--------|--------|--------|--------|--------|
| IOCBP | IOCBP7 | IOCBP6 | IOCBP5 | IOCBP4 | IOCBP3 | IOCBP2 | IOCBP1 | IOCBP0 |
| IOCBN | IOCBN7 | IOCBN6 | IOCBN5 | IOCBN4 | IOCBN3 | IOCBN2 | IOCBN1 | IOCBN0 |
| IOCBF | IOCBF7 | IOCBF6 | IOCBF5 | IOCBF4 | IOCBF3 | IOCBF2 | IOCBF1 | IOCBF0 |
| IOCCP | IOCCP7 | IOCCP6 | IOCCP5 | IOCCP4 | IOCCP3 | IOCCP2 | IOCCP1 | IOCCP0 |
| IOCCN | IOCCN7 | IOCCN6 | IOCCN5 | IOCCN4 | IOCCN3 | IOCCN2 | IOCCN1 | IOCCN0 |
| IOCCF | IOCCF7 | IOCCF6 | IOCCF5 | IOCCF4 | IOCCF3 | IOCCF2 | IOCCF1 | IOCCF0 |
| IOCEP | IOCEP7 | IOCEP6 | IOCEP5 | IOCEP4 | IOCEP3 | IOCEP2 | IOCEP1 | IOCEP0 |
| IOCEN | IOCEN7 | IOCEN6 | IOCEN5 | IOCEN4 | IOCEN3 | IOCEN2 | IOCEN1 | IOCEN0 |
| IOCEF | IOCEF7 | IOCEF6 | IOCEF5 | IOCEF4 | IOCEF3 | IOCEF2 | IOCEF1 | IOCEF0 |
| IOCGP | — | — | IOCGP5 ⁽¹⁾ | — | — | — | — | — |
| IOCGN | — | — | IOCGN5 ⁽¹⁾ | — | — | — | — | — |
| IOCGF | — | — | IOCGF5 ⁽¹⁾ | — | — | — | — | — |

Note 1: If MCLRE = 1 or LVP = 1, RG5 port functionality is disabled and IOC on RG5 is not available.

TABLE 16-2: SUMMARY OF REGISTERS ASSOCIATED WITH INTERRUPT-ON-CHANGE

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on Page |
|--------|----------|-----------|-------|-------|---------|---------|---------|---------|------------------|
| INTCON | GIE/GIEH | PEIE/GIEL | IPEN | — | INT3EDG | INT2EDG | INT1EDG | INT0EDG | 172 |
| IOCF | IOCF7 | IOCF6 | IOCF5 | IOCF4 | IOCF3 | IOCF2 | IOCF1 | IOCF0 | 219 |
| IOCN | IOCN7 | IOCN6 | IOCN5 | IOCN4 | IOCN3 | IOCN2 | IOCN1 | IOCN0 | 219 |
| IOCP | IOCP7 | IOCP6 | IOCP5 | IOCP4 | IOCP3 | IOCP2 | IOCP1 | IOCP0 | 219 |

Legend: — = unimplemented location, read as '0'. Shaded cells are not used by interrupt-on-change.

17.0 PERIPHERAL PIN SELECT (PPS) MODULE

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. All analog inputs and outputs remain fixed to their assigned pins. Input and output selections are independent as shown in the simplified block diagram [Figure 17-1](#).

The peripheral input is selected with the peripheral xxxPPS register ([Register 17-1](#)), and the peripheral output is selected with the PORT RxyPPS register ([Register 17-2](#)). For example, to select PORTC<7> as the EUSART1 RX input, set RXxPPS to 6'b01 0111, and to select PORTC<6> as the EUSART1 TX output set RC6PPS to 6'b00 1100.

17.1 PPS Inputs

Each peripheral has a PPS register with which the inputs to the peripheral are selected. Inputs include the device pins.

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.

Although every peripheral has its own PPS input selection register, the selections are identical for every peripheral as shown in [Register 17-1](#).

Note: The notation “xxx” in the register name is a place holder for the peripheral identifier. For example, INT0PPS.

17.2 PPS Outputs

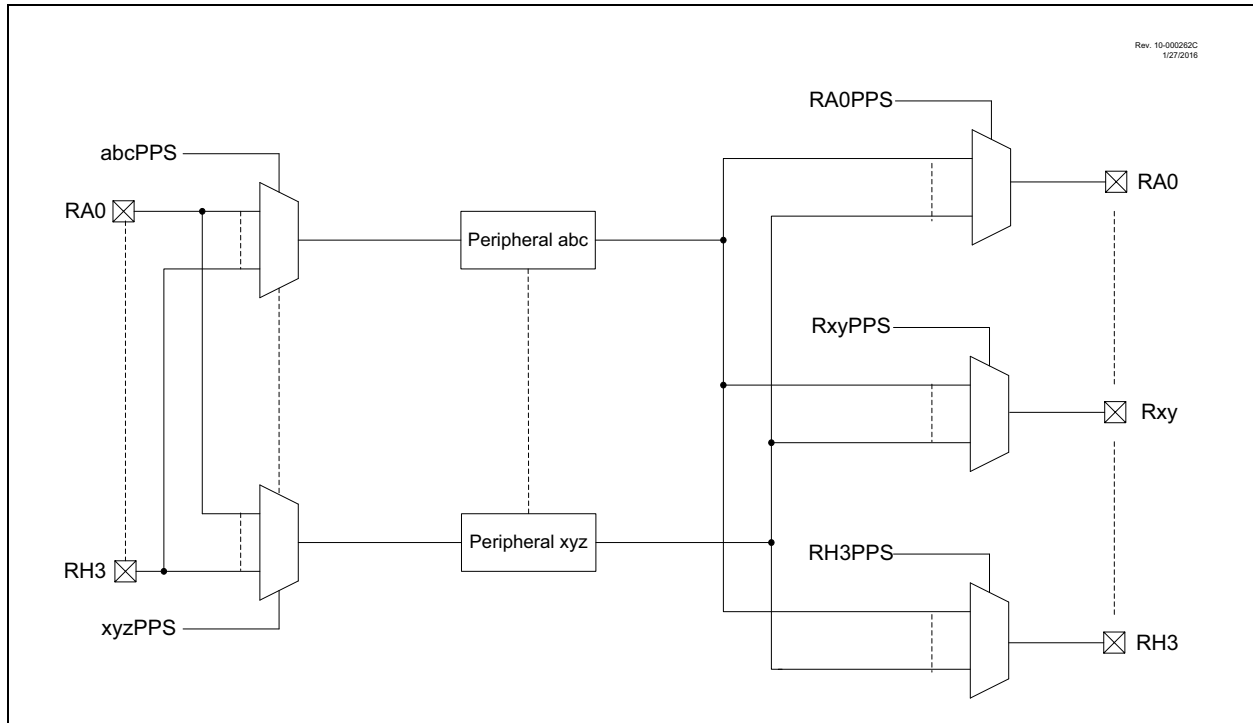
Each I/O pin has a PPS register with which the pin output source is selected. With few exceptions, the port TRIS control associated with that pin retains control over the pin output driver. Peripherals that control the pin output driver as part of the peripheral operation will override the TRIS control as needed. These peripherals include:

- EUSART (synchronous operation)
- MSSP (I²C)

Although every pin has its own PPS peripheral selection register, the selections are identical for every pin as shown in [Register 17-2](#).

Note: The notation “Rxy” is a place holder for the pin identifier. For example, RA0PPS.

FIGURE 17-1: SIMPLIFIED PPS BLOCK DIAGRAM



17.3 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. Peripherals that have bidirectional signals include:

- EUSART (synchronous operation)
- MSSP (I²C)
- CCP module

Note: The I²C default input pins are I²C and SMBus compatible. RB1 and RB2 are additional pins. RC4 and RC3 are default MMP1 pins and are SMBus compatible. Clock and data signals can be routed to any pin, however pins without I²C compatibility will operate at standard TTL/ST logic levels as selected by the INVLV register.

17.4 PPS Lock

The PPS includes a mode in which all input and output selections can be locked to prevent inadvertent changes. PPS selections are locked by setting the PPSLOCKED bit of the PPSLOCK register. Setting and clearing this bit requires a special sequence as an extra precaution against inadvertent changes. Examples of setting and clearing the PPSLOCKED bit are shown in [Example 17-1](#).

EXAMPLE 17-1: PPS LOCK SEQUENCE

```

; Disable interrupts:
BCF    INTCON,GIE

; Bank to PPSLOCK register
BANKSEL PPSLOCK
MOVLB  PPSLOCK
MOVLW  55h

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

; Set PPSLOCKED bit to disable writes
; Only a BSF instruction will work
BSF    PPSLOCK,0

; Enable Interrupts
BSF    INTCON,GIE
    
```

EXAMPLE 17-2: PPS UNLOCK SEQUENCE

```

; Disable interrupts:
BCF    INTCON,GIE

; Bank to PPSLOCK register
BANKSEL PPSLOCK
MOVLB  PPSLOCK
MOVLW  55h

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

; Clear PPSLOCKED bit to enable writes
; Only a BCF instruction will work
BCF    PPSLOCK,0

; Enable Interrupts
BSF    INTCON,GIE
    
```

17.5 PPS One-Way Lock

Using the PPS1WAY Configuration bit, the PPS settings can be locked in. When this bit is set, the PPSLOCKED bit can only be cleared and set one time after a device Reset. This allows for clearing the PPSLOCKED bit so that the input and output selections can be made during initialization. When the PPSLOCKED bit is set after all selections have been made, it will remain set and cannot be cleared until after the next device Reset event.

17.6 Operation During Sleep

PPS input and output selections are unaffected by Sleep.

17.7 Effects of a Reset

A device Power-on-Reset (POR) clears all PPS input and output selections to their default values. All other Resets leave the selections unchanged. Default input selections are shown in the [Section "Pin Allocation Tables"](#). The PPS one-way is also removed.

17.8 Register Definitions: PPS Input Selection

REGISTER 17-1: xxxPPS: PERIPHERAL xxx INPUT SELECTION

| | | | | | | | |
|-------|-----|------------------------|------------------------|------------------------|------------------------|------------------------|------------------------|
| U-0 | U-0 | R/W-m/u ⁽¹⁾ | R/W-m/u ⁽¹⁾ | R/W-m/u ⁽¹⁾ | R/W-m/u ⁽¹⁾ | R/W-m/u ⁽¹⁾ | R/W-m/u ⁽¹⁾ |
| — | — | xxxPPS<5:0> | | | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|---------------------------------------|---|
| R = Readable bit | W = Writable bit | -n/n = Value at POR and BOR/Value at all other Resets |
| u = Bit is unchanged | x = Bit is unknown | q = value depends on peripheral |
| '1' = Bit is set | U = Unimplemented bit, read as '0' | m = value depends on default location for that input |
| '0' = Bit is cleared | | |

bit 7-6 **Unimplemented:** Read as '0'

bit 5-3 **xxxPPS<5:3>:** Peripheral xxx Input PORTx Pin Selection bits
 See [Table 17-1](#) for the list of available ports and default pin locations.
 111 = PORTH
 110 = PORTG
 101 = PORTF
 100 = PORTE
 011 = PORTD
 010 = PORTC
 001 = PORTB
 000 = PORTA

bit 2-0 **xxxPPS<2:0>:** Peripheral xxx Input PORTx Pin Selection bits
 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)

Note 1: The Reset value 'm' of this register is determined by device default locations for that input.

TABLE 17-1: PPS INPUT REGISTER DETAILS

| Peripheral | PPS Input Register | Default Pin Selection at POR | Register Reset Value at POR | Input Available from Selected PORTx | | | | | | | | | |
|------------------------|--------------------|------------------------------|-----------------------------|-------------------------------------|---|---|---|---|---|---|---|---|---|
| | | | | A | B | C | D | E | F | G | H | | |
| Interrupt 0 | INT0PPS | RB0 | 0x08 | A | B | — | — | — | — | — | — | — | — |
| Interrupt 1 | INT1PPS | RB1 | 0x09 | — | B | C | — | — | — | — | — | — | — |
| Interrupt 2 | INT2PPS | RB2 | 0x0A | — | B | — | D | — | — | — | — | — | — |
| Interrupt 3 | INT3PPS | RB3 | 0x0B | — | B | — | — | E | — | — | — | — | — |
| Timer0 Clock | T0CKIPPS | RA4 | 0x04 | A | B | — | — | — | — | — | — | — | — |
| Timer1 Clock | T1CKIPPS | RC0 | 0x10 | — | — | C | D | — | — | — | — | — | — |
| Timer1 Gate | T1GPPS | RB5 | 0x0D | — | B | C | — | — | — | — | — | — | — |
| Timer3 Clock | T3CKIPPS | RB5 | 0x0D | — | B | C | — | — | — | — | — | — | — |
| Timer3 Gate | T3GPPS | RA5 | 0x05 | A | — | C | — | — | — | — | — | — | — |
| Timer5 Clock | T5CKIPPS | RD1 | 0x19 | — | — | — | D | E | — | — | — | — | — |
| Timer5 Gate | T5GPPS | RG4 | 0x34 | — | — | — | — | E | — | G | — | — | — |
| Timer7 Clock | T7CKIPPS | RG4 | 0x34 | — | — | — | — | E | — | G | — | — | — |
| Timer7 Gate | T7GPPS | RD1 | 0x19 | — | — | — | D | E | — | — | — | — | — |
| Timer2 Clock | T2INPPS | RA1 | 0x01 | A | — | C | — | — | — | — | — | — | — |
| Timer4 Clock | T4INPPS | RE4 | 0x24 | — | B | — | — | E | — | — | — | — | — |
| Timer6 Clock | T6INPPS | RC1 | 0x11 | — | B | C | — | — | — | — | — | — | — |
| Timer8 Clock | T8INPPS | RA0 | 0x00 | A | — | — | — | E | — | — | — | — | — |
| ADC Conversion Trigger | ADACTPPS | RH1 | 0x39 | — | B | C | — | — | — | — | — | — | — |
| CCP1 | CCP1PPS | RE5 | 0x25 | — | — | — | — | E | — | G | — | — | — |
| CCP2 | CCP2PPS | RE4 | 0x24 | — | — | — | — | E | — | G | — | — | — |
| CCP3 | CCP3PPS | RE6 | 0x26 | — | — | C | — | E | — | — | — | — | — |
| CCP4 | CCP4PPS | RG3 | 0x33 | — | — | C | — | E | — | — | — | — | — |
| CCP5 | CCP5PPS | RG4 | 0x34 | — | — | C | — | E | — | — | — | — | — |
| SMT1 Window | SMT1WINPPS | RE6 | 0x26 | — | — | C | — | E | — | — | — | — | — |
| SMT1 Signal | SMT1SIGPPS | RE7 | 0x27 | — | — | C | — | E | — | — | — | — | — |
| SMT2 Window | SMT2WINPPS | RG6 | 0x36 | — | — | C | — | — | — | — | G | — | — |
| SMT2 Signal | SMT2SIGPPS | RG7 | 0x37 | — | — | C | — | — | — | — | G | — | — |
| CWG | CWG1PPS | RC2 | 0x12 | A | — | C | — | — | — | — | — | — | — |
| DSM Carrier Low | MDCARLPPS | RD3 | 0x1B | — | — | — | D | — | — | — | — | H | — |
| DSM Carrier High | MDCARHPPS | RD4 | 0x1C | — | — | — | D | — | — | — | — | H | — |
| DSM Source | MDSRCPPS | RD5 | 0x1D | — | — | — | D | — | — | — | — | H | — |
| EUSART1 Receive | RX1PPS | RC7 | 0x17 | — | — | C | D | — | — | — | — | — | — |
| EUSART1 Transmit | TX1PPS | RC6 | 0x16 | — | — | C | D | — | — | — | — | — | — |
| EUSART2 Receive | RX2PPS | RG2 | 0x32 | — | — | — | D | — | — | — | G | — | — |
| EUSART2 Transmit | TX2PPS | RG1 | 0x31 | — | — | — | D | — | — | — | G | — | — |
| EUSART3 Receive | RX3PPS | RE1 | 0x21 | — | B | — | — | E | — | — | — | — | — |

TABLE 17-1: PPS INPUT REGISTER DETAILS (CONTINUED)

| Peripheral | PPS Input Register | Default Pin Selection at POR | Register Reset Value at POR | Input Available from Selected PORTx | | | | | | | |
|--------------------|--------------------|------------------------------|-----------------------------|-------------------------------------|---|---|---|---|---|---|---|
| | | | | — | B | — | — | E | — | — | — |
| EUSART3 Transmit | TX3PPS | RE0 | 0x20 | — | B | — | — | E | — | — | — |
| EUSART4 Receive | RX4PPS | RC1 | 0x11 | — | B | C | — | — | — | — | — |
| EUSART4 Transmit | TX4PPS | RC0 | 0x10 | — | B | C | — | — | — | — | — |
| EUSART5 Receive | RX5PPS | RE3 | 0x23 | — | — | — | — | E | — | G | — |
| EUSART5 Transmit | TX5PPS | RE2 | 0x22 | — | — | — | — | E | — | G | — |
| MSSP1 Clock | SSP1CLKPPS | RC3 | 0x13 | — | B | C | — | — | — | — | — |
| MSSP1 Data | SSP1DATPPS | RC4 | 0x14 | — | B | C | — | — | — | — | — |
| MSSP1 Slave Select | SSP1SSPPS | RF7 | 0x2F | — | B | — | — | — | F | — | — |
| MSSP2 Clock | SSP2CLKPPS | RD6 | 0x1E | — | B | — | D | — | — | — | — |
| MSSP2 Data | SSP2DATPPS | RD5 | 0x1D | — | B | — | D | — | — | — | — |
| MSSP2 Slave Select | SSP2SSPPS | RD7 | 0x1F | — | B | — | D | — | — | — | — |

REGISTER 17-2: RxyPPS: PIN Rxy OUTPUT SOURCE SELECTION REGISTER

| | | | | | | | |
|-------|-----|-------------|---------|---------|---------|---------|---------|
| U-0 | U-0 | R/W-0/u | R/W-0/u | R/W-0/u | R/W-0/u | R/W-0/u | R/W-0/u |
| — | — | RxyPPS<5:0> | | | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-6 **Unimplemented:** Read as '0'

REGISTER 17-2: RxyPPS: PIN Rxy OUTPUT SOURCE SELECTION REGISTER

bit 5-0

RxyPPS<5:0>: Pin Rxy Output Source Selection bits

| RxyPPS<5:0> | Pin Rxy Output Source | Output can be redirected to PORTx | | | | | | | |
|-------------|-----------------------|-----------------------------------|---|---|---|---|---|---|---|
| | | | | | | | | | |
| 0x21 | ADGRDB | — | — | C | — | — | — | — | H |
| 0x20 | ADGRDA | — | — | C | — | — | — | — | H |
| 0x1F | DSM1 | — | — | C | — | — | — | — | H |
| 0x1E | CLKR | — | — | C | — | — | — | — | H |
| 0x1D | TMR0 | — | B | C | — | — | — | — | — |
| 0x1C | MSSP2 (SDO/SDA) | — | B | — | D | — | — | — | — |
| 0x1B | MSSP2 (SCK/SCL) | — | B | — | D | — | — | — | — |
| 0x1A | MSSP1 (SDO/SDA) | — | B | C | — | — | — | — | — |
| 0x19 | MSSP1 (SCK/SCL) | — | B | C | — | — | — | — | — |
| 0x18 | CMP3 | — | — | — | — | — | F | G | — |
| 0x17 | CMP2 | — | — | — | — | — | F | G | — |
| 0x16 | CMP1 | — | — | — | — | — | F | G | — |
| 0x15 | EUSART5 (DT) | — | — | — | — | E | — | G | — |
| 0x14 | EUSART5 (TX/CK) | — | — | — | — | E | — | G | — |
| 0x13 | EUSART4 (DT) | — | B | C | — | — | — | — | — |
| 0x12 | EUSART4 (TX/CK) | — | B | C | — | — | — | — | — |
| 0x11 | EUSART3 (DT) | — | B | — | — | E | — | — | — |
| 0x10 | EUSART3 (TX/CK) | — | B | — | — | E | — | — | — |
| 0xF | EUSART2 (DT) | — | — | — | D | — | — | G | — |
| 0xE | EUSART2 (TX/CK) | — | — | — | D | — | — | G | — |
| 0xD | EUSART1 (DT) | — | — | C | D | — | — | — | — |
| 0xC | EUSART1 (TX/CK) | — | — | C | D | — | — | — | — |
| 0xB | PWM7 | — | — | C | — | E | — | — | — |
| 0xA | PWM6 | — | — | C | — | E | — | — | — |
| 0x9 | CCP5 | — | — | — | — | E | — | G | — |
| 0x8 | CCP4 | — | — | — | — | E | — | G | — |
| 0x7 | CCP3 | — | — | C | — | E | — | — | — |
| 0x6 | CCP2 | — | — | C | — | E | — | — | — |
| 0x5 | CCP1 | — | — | C | — | E | — | — | — |
| 0x4 | CWG1D | — | — | — | — | E | — | G | — |
| 0x3 | CWG1C | — | — | C | — | E | — | — | — |
| 0x2 | CWG1B | — | — | — | — | E | — | G | — |
| 0x1 | CWG1A | — | — | C | — | E | — | — | — |
| 0x0 | LATxy | A | B | C | D | E | F | G | H |

PIC18(L)F67K40

REGISTER 17-3: PPSLOCK: PPS LOCK REGISTER

| | | | | | | | |
|-------|-----|-----|-----|-----|-----|-----|-----------|
| U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | R/W-0/0 |
| — | — | — | — | — | — | — | PPSLOCKED |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

- bit 7-1 **Unimplemented:** Read as '0'
- bit 0 **PPSLOCKED:** PPS Locked bit
- 1 = PPS is locked. PPS selections can not be changed.
- 0 = PPS is not locked. PPS selections can be changed.

TABLE 17-2: SUMMARY OF REGISTERS ASSOCIATED WITH THE PPS MODULE

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on page |
|------------|-------|-------|-----------------|-------|-------|-------|-------|-----------|---------------------|
| PPSLOCK | — | — | — | — | — | — | — | PPSLOCKED | 229 |
| INT0PPS | — | — | INT0PPS<5:0> | | | | | | 224 |
| INT1PPS | — | — | INT1PPS<5:0> | | | | | | 224 |
| INT2PPS | — | — | INT2PPS<5:0> | | | | | | 224 |
| INT3PPS | — | — | INT3PPS<5:0> | | | | | | 224 |
| T0CKIPPS | — | — | T0CKIPPS<5:0> | | | | | | 224 |
| T1CKIPPS | — | — | T1CKIPPS<5:0> | | | | | | 224 |
| T1GPPS | — | — | T1GPPS<5:0> | | | | | | 224 |
| T3CKIPPS | — | — | T3CKIPPS<5:0> | | | | | | 224 |
| T3GPPS | — | — | T3GPPS<5:0> | | | | | | 224 |
| T5CKIPPS | — | — | T5CKIPPS<5:0> | | | | | | 224 |
| T5GPPS | — | — | T5GPPS<4:0> | | | | | | 224 |
| T7CKIPPS | — | — | T7CKIPPS<5:0> | | | | | | 224 |
| T7GPPS | — | — | T7GPPS<5:0> | | | | | | 224 |
| T2INPPS | — | — | T2INPPS<5:0> | | | | | | 224 |
| T4INPPS | — | — | T4INPPS<5:0> | | | | | | 224 |
| T6INPPS | — | — | T6INPPS<5:0> | | | | | | 224 |
| T8INPPS | — | — | T8INPPS<5:0> | | | | | | 224 |
| CCP1PPS | — | — | CCP1PPS<5:0> | | | | | | 224 |
| CCP2PPS | — | — | CCP2PPS<5:0> | | | | | | 224 |
| CCP3PPS | — | — | CCP3PPS<5:0> | | | | | | 224 |
| CCP4PPS | — | — | CCP4PPS<5:0> | | | | | | 224 |
| CCP5PPS | — | — | CCP5PPS<5:0> | | | | | | 224 |
| SMT1WINPPS | — | — | SMT1WINPPS<5:0> | | | | | | 224 |
| SMT1SIGPPS | — | — | SMT1SIGPPS<5:0> | | | | | | 224 |
| SMT2WINPPS | — | — | SMT2WINPPS<5:0> | | | | | | 224 |
| SMT2SIGPPS | — | — | SMT2SIGPPS<5:0> | | | | | | 224 |
| CWG1PPS | — | — | CWG1PPS<5:0> | | | | | | 224 |

PIC18(L)F67K40

TABLE 17-2: SUMMARY OF REGISTERS ASSOCIATED WITH THE PPS MODULE (CONTINUED)

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on page |
|------------|-------|-------|-------|-------|-------|-------|-------|-------|------------------------|
| MDCARLPPS | — | — | | | | | | | MDCARLPPS<5:0> 224 |
| MDCARHPPS | — | — | | | | | | | MDCARHPPS<5:0> 224 |
| MDSRCPPS | — | — | | | | | | | MDSRCPPS<5:0> 224 |
| ADACTPPS | — | — | | | | | | | ADACTPPS<5:0> 224 |
| SSP1CLKPPS | — | — | | | | | | | SSP1CLKPPS<5:0> 224 |
| SSP1DATPPS | — | — | | | | | | | SSP1DATPPS<5:0> 224 |
| SSP1SSPPS | — | — | | | | | | | SSP1SSPPS<5:0> 224 |
| SSP2CLKPPS | — | — | | | | | | | SSP2CLKPPS<5:0> 224 |
| SSP2DATPPS | — | — | | | | | | | SSP2DATPPS<5:0> 224 |
| SSP2SSPPS | — | — | | | | | | | SSP2SSPPS<5:0> 224 |
| RX1PPS | — | — | | | | | | | RX1PPS<5:0> 224 |
| TX1PPS | — | — | | | | | | | TX1PPS<5:0> 224 |
| RX2PPS | — | — | | | | | | | RX2PPS<5:0> 224 |
| TX2PPS | — | — | | | | | | | TX2PPS<5:0> 224 |
| RX3PPS | — | — | | | | | | | RX3PPS<5:0> 224 |
| TX3PPS | — | — | | | | | | | TX3PPS<5:0> 224 |
| RX4PPS | — | — | | | | | | | RX4PPS<5:0> 224 |
| TX4PPS | — | — | | | | | | | TX4PPS<5:0> 224 |
| RX5PPS | — | — | | | | | | | RX5PPS<5:0> 224 |
| TX5PPS | — | — | | | | | | | TX5PPS<5:0> 224 |
| RxyPPS | — | — | | | | | | | RxyPPS<5:0> 224 |

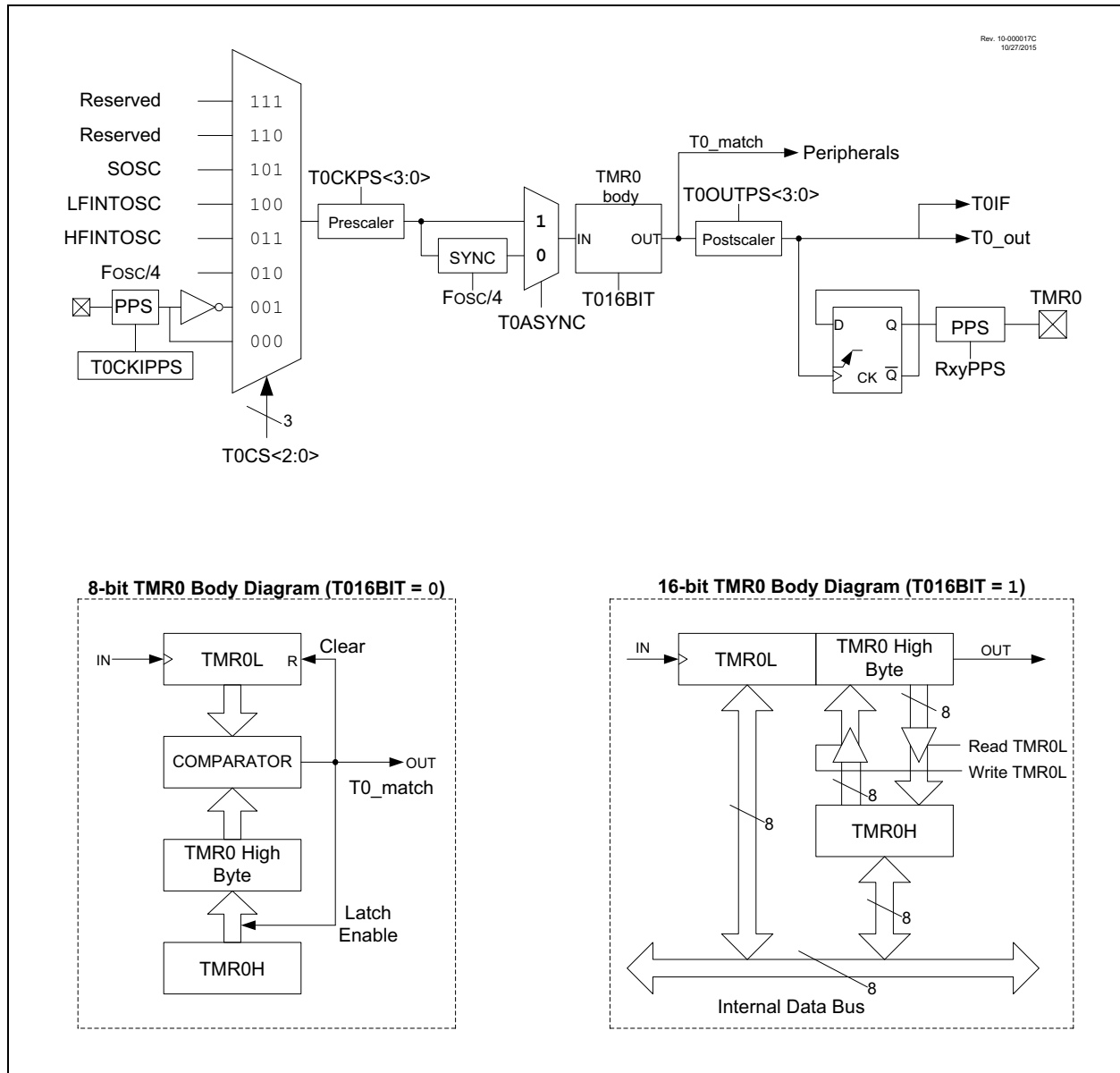
Legend: — = unimplemented, read as '0'. Shaded cells are unused by the PPS module.

18.0 TIMER0 MODULE

Timer0 module is an 8/16-bit timer/counter with the following features:

- 16-bit timer/counter
- 8-bit timer/counter with programmable period
- Synchronous or asynchronous operation
- Selectable clock sources
- Programmable prescaler
- Programmable postscaler
- Operation during Sleep mode
- Interrupt on match or overflow
- Output on I/O pin (via PPS) or to other peripherals

FIGURE 18-1: BLOCK DIAGRAM OF TIMER0



18.1 Timer0 Operation

Timer0 can operate as either an 8-bit timer/counter or a 16-bit timer/counter. The mode is selected with the T016BIT bit of the T0CON register.

18.1.1 16-BIT MODE

The register pair TMR0H:TMR0L, increments on the rising edge of the clock source. A 15-bit prescaler on the clock input gives several prescale options (see prescaler control bits, T0CKPS<3:0> in the T0CON1 register).

18.1.1.1 Timer0 Reads and Writes in 16-Bit Mode

In 16-bit mode, to avoid rollover between reading high and low registers, the TMR0H register is a buffered copy of the actual high byte of Timer0, which is neither directly readable nor writable (see [Figure 18-1](#)). TMR0H is updated with the contents of the high byte of Timer0 during a read of TMR0L. This provides the ability to read all 16 bits of Timer0 without having to verify that the read of the high and low byte was valid, due to a rollover between successive reads of the high and low byte.

Similarly, a write to the high byte of Timer0 must also take place through the TMR0H Buffer register. The high byte is updated with the contents of TMR0H when a write occurs to TMR0L. This allows all 16 bits of Timer0 to be updated at once.

18.1.2 8-BIT MODE

In normal operation, TMR0 increments on the rising edge of the clock source. A 15-bit prescaler on the clock input gives several prescale options (see prescaler control bits, T0CKPS<3:0> in the T0CON1 register).

In 8-bit mode, the value of TMR0L is compared to that of the Period buffer, a copy of TMR0H, on each clock cycle. When the two values match, the following events happen:

- TMR0_out goes high for one prescaled clock period
- TMR0L is reset
- The contents of TMR0H are copied to the period buffer

In 8-bit mode, the TMR0L and TMR0H registers are both directly readable and writable. The TMR0L register is cleared on any device Reset, while the TMR0H register initializes at FFh.

Both the prescaler and postscaler counters are cleared on the following events:

- A write to the TMR0L register
- A write to either the T0CON0 or T0CON1 registers
- Any device Reset – Power-on Reset (POR), MCLR Reset, Watchdog Timer Reset (WDTR) or
- Brown-out Reset (BOR)

18.1.3 COUNTER MODE

In Counter mode, the prescaler is normally disabled by setting the T0CKPS bits of the T0CON1 register to '0000'. Each rising edge of the clock input (or the output of the prescaler if the prescaler is used) increments the counter by '1'.

18.1.4 TIMER MODE

In Timer mode, the Timer0 module will increment every instruction cycle as long as there is a valid clock signal and the T0CKPS bits of the T0CON1 register ([Register 18-2](#)) are set to '0000'. When a prescaler is added, the timer will increment at the rate based on the prescaler value.

18.1.5 ASYNCHRONOUS MODE

When the T0ASYNC bit of the T0CON1 register is set (T0ASYNC = '1'), the counter increments with each rising edge of the input source (or output of the prescaler, if used). Asynchronous mode allows the counter to continue operation during Sleep mode provided that the clock also continues to operate during Sleep.

18.1.6 SYNCHRONOUS MODE

When the T0ASYNC bit of the T0CON1 register is clear (T0ASYNC = 0), the counter clock is synchronized to the system clock (FOSC/4). When operating in Synchronous mode, the counter clock frequency cannot exceed FOSC/4.

18.2 Clock Source Selection

The T0CS<2:0> bits of the T0CON1 register are used to select the clock source for Timer0. [Register 18-2](#) displays the clock source selections.

18.2.1 INTERNAL CLOCK SOURCE

When the internal clock source is selected, Timer0 operates as a timer and will increment on multiples of the clock source, as determined by the Timer0 prescaler.

18.2.2 EXTERNAL CLOCK SOURCE

When an external clock source is selected, Timer0 can operate as either a timer or a counter. Timer0 will increment on multiples of the rising edge of the external clock source, as determined by the Timer0 prescaler.

18.3 Programmable Prescaler

A software programmable prescaler is available for exclusive use with Timer0. There are 16 prescaler options for Timer0 ranging in powers of two from 1:1 to 1:32768. The prescaler values are selected using the T0CKPS<3:0> bits of the T0CON1 register.

The prescaler is not directly readable or writable. Clearing the prescaler register can be done by writing to the TMR0L register or the T0CON0, T0CON1 registers or by any Reset.

18.4 Programmable Postscaler

A software programmable postscaler (output divider) is available for exclusive use with Timer0. There are 16 postscaler options for Timer0 ranging from 1:1 to 1:16. The postscaler values are selected using the T0OUTPS<3:0> bits of the T0CON0 register.

The postscaler is not directly readable or writable. Clearing the postscaler register can be done by writing to the TMR0L register or the T0CON0, T0CON1 registers or by any Reset.

18.5 Operation During Sleep

When operating synchronously, Timer0 will halt. When operating asynchronously, Timer0 will continue to increment and wake the device from Sleep (if Timer0 interrupts are enabled) provided that the input clock source is active.

18.6 Timer0 Interrupts

The Timer0 interrupt flag bit (TMR0IF) is set when either of the following conditions occur:

- 8-bit TMR0L matches the TMR0H value
- 16-bit TMR0 rolls over from 'FFFFh'

When the postscaler bits (T0OUTPS<3:0>) 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.

If Timer0 interrupts are enabled (TMR0IE bit of the PIE0 register = '1'), the CPU will be interrupted and the device may wake from Sleep (see [Section 18.2 "Clock Source Selection"](#) for more details).

18.7 Timer0 Output

The Timer0 output can be routed to any I/O pin via the RxyPPS output selection register (see [Section 17.0 "Peripheral Pin Select \(PPS\) Module"](#) for additional information). The Timer0 output can also be used by other peripherals, such as the auto-conversion trigger of the Analog-to-Digital Converter. Finally, the Timer0 output can be monitored through software via the Timer0 output bit (T0OUT) of the T0CON0 register ([Register 18-1](#)).

TMR0_out will be a pulse of one postscaled clock period when a match occurs between TMR0L and PR0 (Period register for TMR0) in 8-bit mode, or when TMR0 rolls over in 16-bit mode. The Timer0 output is a 50% duty cycle that toggles on each TMR0_out rising clock edge.

18.8 Register Definitions: Timer0 Control

REGISTER 18-1: T0CON0: TIMER0 CONTROL REGISTER 0

| | | | | | | | | |
|---------|-----|-------|---------|--------------|---------|---------|---------|-------|
| R/W-0/0 | U-0 | R-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | |
| T0EN | — | T0OUT | T016BIT | T0OUTPS<3:0> | | | | |
| bit 7 | | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

- bit 7 **T0EN:** TMR0 Enable bit
 1 = The module is enabled and operating
 0 = The module is disabled and in the lowest power mode
- bit 6 **Unimplemented:** Read as '0'
- bit 5 **T0OUT:** TMR0 Output bit (read-only)
 TMR0 output bit
- bit 4 **T016BIT:** TMR0 Operating as 16-Bit Timer Select bit
 1 = TMR0 is a 16-bit timer
 0 = TMR0 is an 8-bit timer
- bit 3-0 **T0OUTPS<3:0>:** TMR0 Output Postscaler (Divider) Select bits
 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

REGISTER 18-2: T0CON1: TIMER0 CONTROL REGISTER 1

| | | | | | | | |
|-----------|---------|---------|-------------|---------|---------|---------|---------|
| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| T0CS<2:0> | | T0ASYNC | T0CKPS<3:0> | | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-5 **T0CS<2:0>**: Timer0 Clock Source Select bits

- 111 = Reserved
- 110 = Reserved
- 101 = SOSC
- 100 = LFINTOSC
- 011 = HFINTOSC
- 010 = Fosc/4
- 001 = Pin selected by T0CKIPPS (Inverted)
- 000 = Pin selected by T0CKIPPS (Non-inverted)

bit 4 **T0ASYNC**: TMR0 Input Asynchronization Enable bit

- 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

bit 3-0 **T0CKPS<3:0>**: Prescaler Rate Select bit

- 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

REGISTER 18-3: TMR0L: TIMER0 COUNT REGISTER

| | | | | | | | |
|-----------|---------|---------|---------|---------|---------|---------|---------|
| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| TMR0<7:0> | | | | | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-0 **TMR0<7:0>**:TMR0 Counter bits <7:0>

REGISTER 18-4: TMR0H: TIMER0 PERIOD REGISTER

| | | | | | | | |
|------------|---------|---------|---------|---------|---------|---------|---------|
| R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 |
| TMR0<15:8> | | | | | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-0 When T016BIT = 0
PR0<7:0>:TMR0 Period Register Bits <7:0>
 When T016BIT = 1
TMR0<15:8>: TMR0 Counter bits <15:8>

TABLE 18-1: SUMMARY OF REGISTERS ASSOCIATED WITH TIMER0

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on Page |
|----------|------------|-----------|---------------|---------|--------------|---------|---------|---------|------------------|
| TMR0L | TMR0<7:0> | | | | | | | | 236 |
| TMR0H | TMR0<15:8> | | | | | | | | 236 |
| T0CON0 | T0EN | — | T0OUT | T016BIT | T0OUTPS<3:0> | | | | 234 |
| T0CON1 | T0CS<2:0> | | | T0ASYNC | T0CKPS<3:0> | | | | 235 |
| T0CKIPPS | — | — | T0CKIPPS<5:0> | | | | | | 224 |
| TMR0PPS | — | — | TMR0PPS<5:0> | | | | | | 224 |
| INTCON | GIE/GIEH | PEIE/GIEL | IPEN | — | INT3EDG | INT2EDG | INT1EDG | INT0EDG | 172 |
| PIR0 | — | — | TMR0IF | IOCIF | INT3IF | INT2IF | INT1IF | INT0IF | 173 |
| PIE0 | — | — | TMR0IE | IOCIE | INT3IE | INT2IE | INT1IE | INT0IE | 184 |
| IPR0 | — | — | TMR0IP | IOCIP | INT3IP | INT2IP | INT1IP | INT0IP | 194 |
| PMD1 | — | TMR6MD | TMR5MD | TMR4MD | TMR3MD | TMR2MD | TMR1MD | TMR0MD | 66 |

Legend: — = Unimplemented location, read as '0'. Shaded cells are not used by the Timer0 module.

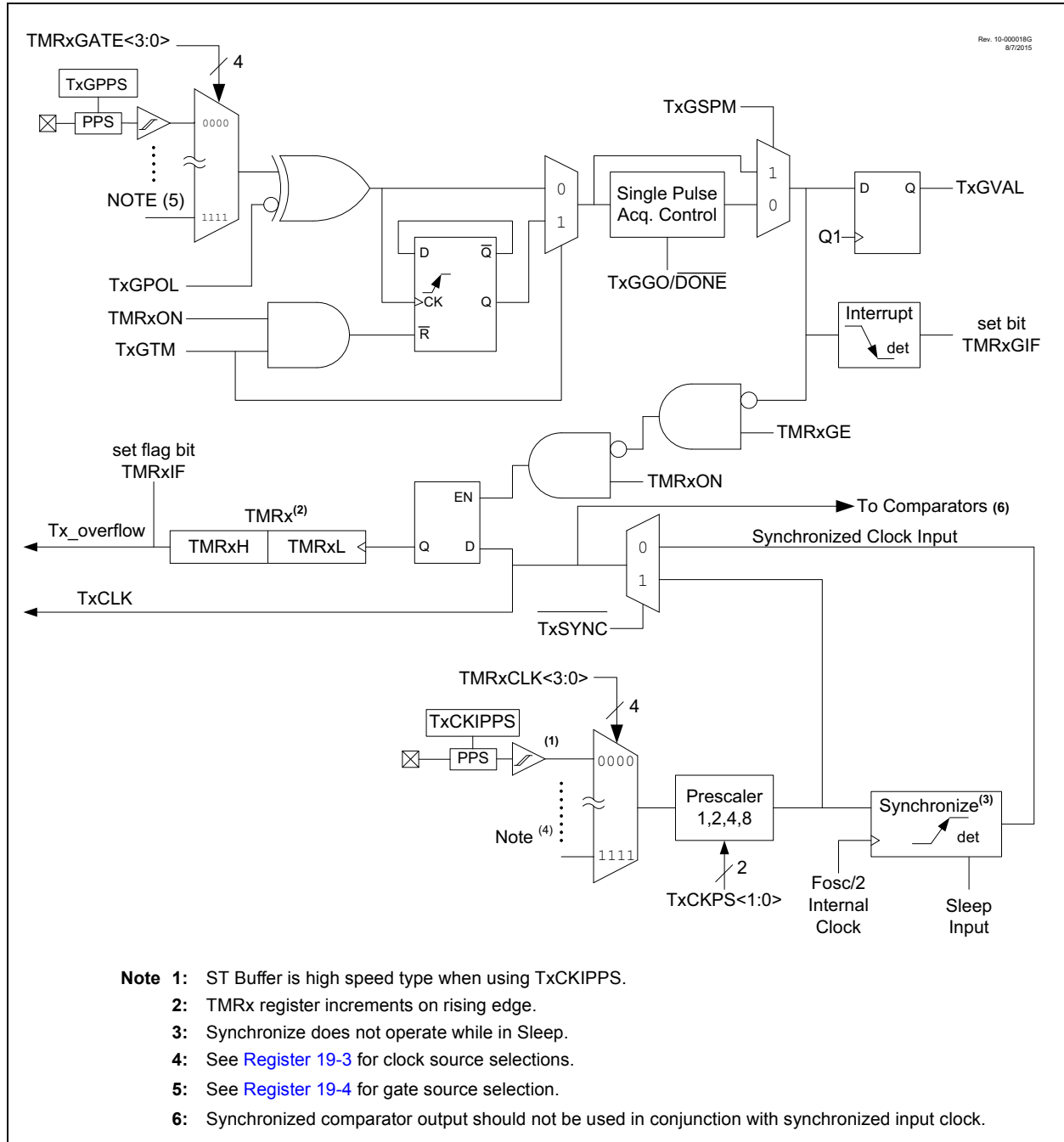
19.0 TIMER1/3/5/7 MODULE WITH GATE CONTROL

Timer1/3/5/7 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
- Dedicated Secondary 32 kHz oscillator circuit
- Optionally synchronized comparator out
- Multiple Timer1/3/5/7 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

Figure 19-1 is a block diagram of the Timer1/3/5/7 module.

FIGURE 19-1: TIMER1/3/5/7 BLOCK DIAGRAM



19.1 Register Definitions: Timer1/3/5/7

Long bit name prefixes for the Timer1/3/5/7 are shown in Table 20-1. Refer to Section 1.4.2.2 “Long Bit Names” for more information.

TABLE 19-1:

| Peripheral | Bit Name Prefix |
|------------|-----------------|
| Timer1 | T1 |
| Timer3 | T3 |
| Timer5 | T5 |
| Timer7 | T7 |

REGISTER 19-1: T_xCON: TIMER_x CONTROL REGISTER

| U-0 | U-0 | R/W-0/u | R/W-0/u | U-0 | R/W-0/u | R/W-0/0 | R/W-0/u |
|-------|-----|-----------|---------|-----|--------------------------|---------|---------|
| — | — | CKPS<1:0> | | — | $\overline{\text{SYNC}}$ | RD16 | ON |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|-------------------|------------------|------------------------------------|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared |
| | | u = unchanged |

- bit 7-6 **Unimplemented:** Read as '0'
- bit 5-4 **CKPS<1:0>:** Timer_x Input Clock Prescale Select bits
 - 11 = 1:8 Prescale value
 - 10 = 1:4 Prescale value
 - 01 = 1:2 Prescale value
 - 00 = 1:1 Prescale value
- bit 3 **Unimplemented:** Read as '0'
- bit 2 **SYNC:** Timer_x External Clock Input Synchronization Control bit
 TMR_xCLK = Fosc/4 or Fosc:
 This bit is ignored. Timer1 uses the incoming clock as is.
 Else:
 1 = Do not synchronize external clock input
 0 = Synchronize external clock input with system clock
- bit 1 **RD16:** 16-Bit Read/Write Mode Enable bit
 - 1 = All 16 bits of Timer1 can be read simultaneously (TMR1H is buffered)
 - 0 = 16-bit reads of Timer1 are disabled (TMR1H not buffered)
- bit 0 **ON:** Timer_x On bit
 - 1 = Enables Timer_x
 - 0 = Disables Timer_x

REGISTER 19-2: TxGCON: TIMERx GATE CONTROL REGISTER

| | | | | | | | |
|---------|---------|---------|---------|----------|------|-------|-----|
| R/W-0/u | R/W-0/u | R/W-0/u | R/W-0/u | R/W-0/u | R-x | U-0 | U-0 |
| GE | GPOL | GTM | GSPM | GGO/DONE | GVAL | — | — |
| bit 7 | | | | | | bit 0 | |

Legend:

| | | |
|-------------------|------------------|------------------------------------|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared |
| | | x = Bit is unknown |

bit 7 **GE:** Timerx Gate Enable bit

If TMRxON = 1:

1 = Timerx counting is controlled by the Timerx gate function

0 = Timerx is always counting

If TMRxON = 0:

This bit is ignored

bit 6 **GPOL:** Timerx Gate Polarity bit

1 = Timerx gate is active-high (Timerx counts when gate is high)

0 = Timerx gate is active-low (Timerx counts when gate is low)

bit 5 **GTM:** Timerx Gate Toggle Mode bit

1 = Timerx Gate Toggle mode is enabled

0 = Timerx Gate Toggle mode is disabled and Toggle flip-flop is cleared

Timerx Gate Flip Flop Toggles on every rising edge

bit 4 **GSPM:** Timerx Gate Single Pulse Mode bit

1 = Timerx Gate Single Pulse mode is enabled and is controlling Timerx gate)

0 = Timerx Gate Single Pulse mode is disabled

bit 3 **GGO/DONE:** Timerx Gate Single Pulse Acquisition Status bit

1 = Timerx Gate Single Pulse Acquisition is ready, waiting for an edge

0 = Timerx Gate Single Pulse Acquisition has completed or has not been started.

This bit is automatically cleared when TxGSPM is cleared.

bit 2 **GVAL:** Timerx Gate Current State bit

Indicates the current state of the Timerx gate that could be provided to TMRxH:TMRxL

Unaffected by Timerx Gate Enable (TMRxGE)

bit 1-0 **Unimplemented:** Read as '0'

REGISTER 19-3: TMRxCLK: TIMERx CLOCK REGISTER

| | | | | | | | |
|-------|-----|-----|-----|---------|---------|---------|---------|
| U-0 | U-0 | U-0 | U-0 | R/W-0/u | R/W-0/u | R/W-0/u | R/W-0/u |
| — | — | — | — | CS<3:0> | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

R = Readable bit W = Writable bit U = Unimplemented bit, read as '0'
 -n = Value at POR '1' = Bit is set '0' = Bit is cleared u = unchanged

bit 7-4 **Unimplemented:** Read as '0'
 bit 3-0 **CS<3:0>:** Timerx Clock Source Selection bits

| CS | Timer1 | Timer3 | Timer5 | Timer7 |
|-----------|--------------------|--------------------|--------------------|--------------------|
| | Clock Source | Clock Source | Clock Source | Clock Source |
| 1111-1101 | Reserved | Reserved | Reserved | Reserved |
| 1100 | TMR7 overflow | TMR7 overflow | TMR7 overflow | Reserved |
| 1011 | TMR5 overflow | TMR5 overflow | Reserved | TMR5 overflow |
| 1010 | TMR3 overflow | Reserved | TMR3 overflow | TMR3 overflow |
| 1001 | Reserved | TMR1 overflow | TMR1 overflow | TMR1 overflow |
| 1000 | TMR0 overflow | TMR0 overflow | TMR0 overflow | TMR0 overflow |
| 0111 | CLKREF | CLKREF | CLKREF | CLKREF |
| 0110 | SOSC | SOSC | SOSC | SOSC |
| 0101 | MFINTOSC (500 kHz) | MFINTOSC (500 kHz) | MFINTOSC (500 kHz) | MFINTOSC (500 kHz) |
| 0100 | LFINTOSC | LFINTOSC | LFINTOSC | LFINTOSC |
| 0011 | HFINTOSC | HFINTOSC | HFINTOSC | HFINTOSC |
| 0010 | Fosc | Fosc | Fosc | Fosc |
| 0001 | Fosc/4 | Fosc/4 | Fosc/4 | Fosc/4 |
| 0000 | T1CKIPPS | T3CKIPPS | T5CKIPPS | T7CKIPPS |

REGISTER 19-4: TMRxGATE: TIMERx GATE ISM REGISTER

| | | | | | | | |
|-------|-----|-----|----------|---------|---------|---------|---------|
| U-0 | U-0 | U-0 | R/W-0/u | R/W-0/u | R/W-0/u | R/W-0/u | R/W-0/u |
| — | — | — | GSS<4:0> | | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

R = Readable bit W = Writable bit U = Unimplemented bit, read as '0'
 -n = Value at POR '1' = Bit is set '0' = Bit is cleared u = unchanged

bit 7-5 **Unimplemented:** Read as '0'
 bit 4-0 **GSS<4:0>:** Timerx Gate Source Selection bits

| GSS | Timer1 | Timer3 | Timer5 | Timer7 |
|-------------|------------------------|------------------------|------------------------|------------------------|
| | Gate Source | Gate Source | Gate Source | Gate Source |
| 11111-10111 | Reserved | Reserved | Reserved | Reserved |
| 10110 | ZCDOUT | ZCDOUT | ZCDOUT | ZCDOUT |
| 10101 | CMP3OUT | CMP3OUT | CMP3OUT | CMP3OUT |
| 10100 | CMP2OUT | CMP2OUT | CMP2OUT | CMP2OUT |
| 10011 | CMP1OUT | CMP1OUT | CMP1OUT | CMP1OUT |
| 10010 | PWM7OUT | PWM7OUT | PWM7OUT | PWM7OUT |
| 10001 | PWM6OUT | PWM6OUT | PWM6OUT | PWM6OUT |
| 10000 | CCP5OUT | CCP5OUT | CCP5OUT | CCP5OUT |
| 01111 | CCP4OUT | CCP4OUT | CCP4OUT | CCP4OUT |
| 01110 | CCP3OUT | CCP3OUT | CCP3OUT | CCP3OUT |
| 01101 | CCP2OUT | CCP2OUT | CCP2OUT | CCP2OUT |
| 01100 | CCP1OUT | CCP1OUT | CCP1OUT | CCP1OUT |
| 01101 | SMT2 overflow | SMT2 overflow | SMT2 overflow | SMT2 overflow |
| 01010 | SMT1 overflow | SMT1 overflow | SMT1 overflow | SMT1 overflow |
| 01001 | TMR8OUT (post-scaled) | TMR8OUT (post-scaled) | TMR8OUT (post-scaled) | TMR8OUT (post-scaled) |
| 01000 | TMR7 overflow | TMR7 overflow | TMR7 overflow | Reserved |
| 00111 | TMR6OUT (post-scaled) | TMR6OUT (post-scaled) | TMR6OUT (post-scaled) | TMR6OUT (post-scaled) |
| 00110 | TMR5 overflow | TMR5 overflow | Reserved | TMR5 overflow |
| 00101 | TMR4OUT (post-scaled) | TMR4OUT (post-scaled) | TMR4OUT (post-scaled) | TMR4OUT (post-scaled) |
| 00100 | TMR3 overflow | Reserved | TMR3 overflow | TMR3 overflow |
| 00011 | TMR2OUT (post-scaled) | TMR2OUT (post-scaled) | TMR2OUT (post-scaled) | TMR2OUT (post-scaled) |
| 00010 | Reserved | TMR1 overflow | TMR1 overflow | TMR1 overflow |
| 00001 | TMR0 overflow | TMR0 overflow | TMR0 overflow | TMR0 overflow |
| 00000 | Pin selected by T1GPPS | Pin selected by T3GPPS | Pin selected by T5GPPS | Pin selected by T7GPPS |

REGISTER 19-5: TMRxL: TIMERx LOW BYTE REGISTER

| | | | | | | | |
|------------|---------|---------|---------|---------|---------|---------|---------|
| R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x |
| TMRxL<7:0> | | | | | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-0 **TMRxL<7:0>**:Timerx Low Byte bits

REGISTER 19-6: TMRxH: TIMERx HIGH BYTE REGISTER

| | | | | | | | |
|------------|---------|---------|---------|---------|---------|---------|---------|
| R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x |
| TMRxH<7:0> | | | | | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-0 **TMRxH<7:0>**:Timerx High Byte bits

19.2 Timer1/3/5/7 Operation

The Timer1/3/5/7 module is a 16-bit incrementing counter which is accessed through the TMRxH:TMRxL register pair. Writes to TMRxH or TMRxL directly update the counter.

When used with an internal clock source, the module is a timer and 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.

Timer1/3/5/7 is enabled by configuring the ON and GE bits in the TxCON and TxGCON registers, respectively. [Table 19-2](#) displays the Timer1/3/5/7 enable selections.

TABLE 19-2: TIMER1/3/5/7 ENABLE SELECTIONS

| ON | GE | Timer1/3/5/7 Operation |
|----|----|------------------------|
| 1 | 1 | Count Enabled |
| 1 | 0 | Always On |
| 0 | 1 | Off |
| 0 | 0 | Off |

19.3 Clock Source Selection

The CS<3:0> bits of the TMRxCLK register ([Register 19-3](#)) are used to select the clock source for Timer1/3/5/7. The four TMRxCLK bits allow the selection of several possible synchronous and asynchronous clock sources. [Register 19-3](#) displays the clock source selections.

19.3.1 INTERNAL CLOCK SOURCE

When the internal clock source is selected the TMRxH:TMRxL register pair will increment on multiples of FOSC as determined by the Timer1/3/5/7 prescaler.

When the FOSC internal clock source is selected, the Timer1/3/5/7 register value will increment by four counts every instruction clock cycle. Due to this condition, a 2 LSB error in resolution will occur when reading the Timer1/3/5/7 value. To utilize the full resolution of Timer1/3/5/7, an asynchronous input signal must be used to gate the Timer1/3/5/7 clock input.

The following asynchronous sources may be used at the Timer1/3/5/7 gate:

- Asynchronous event on the TxGPPS pin
- TMR0OUT
- TMR1/3/5/7OUT (excluding the TMR for which it is being used)
- TMR 2/4/6/8OUT (post-scaled)
- CCP1/2/3/4/5OUT
- PWM6/7OUT
- CMP1/2/3OUT
- ZCDOUT
- SMT1/2 overflow

Note: 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:

- Timer1/3/5/7 enabled after POR
- Write to TMRxH or TMRxL
- Timer1/3/5/7 is disabled
- Timer1/3/5/7 is disabled (TMRxON = 0) when TxCKI is high then Timer1/3/5/7 is enabled (TMRxON = 1) when TxCKI is low.

19.3.2 EXTERNAL CLOCK SOURCE

When the external clock source is selected, the Timer1/3/5/7 module may work as a timer or a counter.

When enabled to count, Timer1/3/5/7 is incremented on the rising edge of the external clock input of the TxCKIPPS pin. This external clock source can be synchronized to the microcontroller system clock or it can run asynchronously.

When used as a timer with a clock oscillator, an external 32.768 kHz crystal can be used in conjunction with the dedicated secondary internal oscillator circuit.

19.4 Timer1/3/5/7 Prescaler

Timer1/3/5/7 has four prescaler options allowing 1, 2, 4 or 8 divisions of the clock input. The CKPS bits of the TxCON register control the prescale counter. The prescale counter is not directly readable or writable; however, the prescaler counter is cleared upon a write to TMRxH or TMRxL.

19.5 Secondary Oscillator

A secondary low-power 32.768 kHz oscillator circuit is built-in between pins SOSC1 (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 Timer1/3/5/7; it can also be used by other modules.

The oscillator circuit is enabled by setting the SOSSEN bit of the OSCEN register (Register 4-7). This can be used as the clock source to the Timer using the TMRxCLK bits. The oscillator will continue to run during Sleep.

Note: The oscillator requires a start-up and stabilization time before use. Thus, the SOSSEN bit of the OSCEN register should be set and a suitable delay observed prior to enabling Timer1/3/5/7. A software check can be performed to confirm if the secondary oscillator is enabled and ready to use. This is done by polling the SOR bit of the OSCSTAT (Register 4-4).

19.6 Timer1/3/5/7 Operation in Asynchronous Counter Mode

If control bit $\overline{\text{SYNC}}$ of the TxCON register is set, the external clock input is not synchronized. The timer increments asynchronously to the internal phase clocks. If 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 (see Section 19.6.1 “Reading and Writing Timer1/3/5/7 in Asynchronous Counter Mode”).

Note: 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.

19.6.1 READING AND WRITING TIMER1/3/5/7 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 should keep in mind that reading the 16-bit timer in two 8-bit values itself, poses certain problems, since the timer may overflow 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.

19.8.2 TIMER1/3/5/7 GATE SOURCE SELECTION

The gate source for Timer1/3/5/7 can be selected using the GSS<4:0> bits of the TMRxGATE register (Register 19-4). The polarity selection for the gate source is controlled by the TxGPOL bit of the TxGCON register (Register 19-2).

Any of the above mentioned signals can be used to trigger the gate. The output of the CMPx can be synchronized to the Timer1/3/5/7 clock or left asynchronous. For more information see Section 33.5.1 “Comparator Output Synchronization”.

19.8.3 TIMER1/3/5/7 GATE TOGGLE MODE

When Timer1/3/5/7 Gate Toggle mode is enabled, it is possible to measure the full-cycle length of a Timer1/3/5/7 gate signal, as opposed to the duration of a single level pulse.

The Timer1/3/5/7 gate source is routed through a flip-flop that changes state on every incrementing edge of the signal. See Figure 19-5 for timing details.

Timer1/3/5/7 Gate Toggle mode is enabled by setting the GTM bit of the TxGCON register. When the GTM bit is cleared, the flip-flop is cleared and held clear. This is necessary in order to control which edge is measured.

Note: Enabling Toggle mode at the same time as changing the gate polarity may result in indeterminate operation.

19.8.4 TIMER1/3/5/7 GATE SINGLE-PULSE MODE

When Timer1/3/5/7 Gate Single-Pulse mode is enabled, it is possible to capture a single-pulse gate event. Timer1/3/5/7 Gate Single-Pulse mode is first enabled by setting the GSPM bit in the TxGCON register. Next, the GGO/DONE bit in the TxGCON register must be set. The Timer1/3/5/7 will be fully enabled on the next incrementing edge. On the next trailing edge of the pulse, the GGO/DONE bit will automatically be cleared. No other gate events will be allowed to increment Timer1/3/5/7 until the GGO/DONE bit is once again set in software.

Clearing the TxGSPM bit of the TxGCON register will also clear the GGO/DONE bit. See Figure 19-6 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 Timer1/3/5/7 gate source to be measured. See Figure 19-7 for timing details.

19.8.5 TIMER1/3/5/7 GATE VALUE STATUS

When Timer1/3/5/7 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 Timer1/3/5/7 gate is not enabled (GE bit is cleared).

19.8.6 TIMER1/3/5/7 GATE EVENT INTERRUPT

When Timer1/3/5/7 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 the PIR5 register will be set. If the TMRxGIE bit in the PIE5 register is set, then an interrupt will be recognized.

The TMRxGIF flag bit operates even when the Timer1/3/5/7 gate is not enabled (GE bit is cleared).

For more information on selecting high or low priority status for the Timer1/3/5/7 Gate Event Interrupt see Section 14.0 “Interrupts”.

19.9 Timer1/3/5/7 Interrupt

The Timer1/3/5/7 register pair (TMRxH:TMRxL) increments to FFFFh and rolls over to 0000h. When Timer1/3/5/7 rolls over, the Timer1/3/5/7 interrupt flag bit of the PIR5 register is set. To enable the interrupt-on-rollover, you must set these bits:

- TMRxON bit of the TxCON register
- TMRxIE bits of the PIE5 register
- PEIE/GIEL bit of the INTCON register
- GIE/GIEH bit of the INTCON register

The interrupt is cleared by clearing the TMRxIF bit in the Interrupt Service Routine.

For more information on selecting high or low priority status for the Timer1/3/5/7 Overflow Interrupt, see [Section 14.0 “Interrupts”](#).

| |
|---|
| Note: The TMRxH:TMRxL register pair and the TMRxIF bit should be cleared before enabling interrupts. |
|---|

19.10 Timer1/3/5/7 Operation During Sleep

Timer1/3/5/7 can only operate during Sleep when set up in Asynchronous Counter mode. In this mode, an external crystal or clock source can be used to increment the counter. To set up the timer to wake the device:

- TMRxON bit of the TxCON register must be set
- TMRxIE bit of the PIE5 register must be set
- PEIE/GIEL bit of the INTCON register must be set
- TxSYNC bit of the TxCON register must be set
- Configure the TMRxCLK register for using secondary oscillator as the clock source
- Enable the SOSSEN bit of the OSCEN register ([Register 4-7](#))

The device will wake-up on an overflow and execute the next instruction. If the GIE/GIEH bit of the INTCON register is set, the device will call the Interrupt Service Routine.

The secondary oscillator will continue to operate in Sleep regardless of the TxSYNC bit setting.

19.11 CCP Capture/Compare Time Base

The CCP modules use the TMRxH:TMRxL register pair as the time base when operating in Capture or Compare mode.

In Capture mode, the value in the TMRxH:TMRxL register pair is copied into the CCPRxH:CCPRxL register pair on a configured event.

In Compare mode, an event is triggered when the value in the CCPRxH:CCPRxL register pair matches the value in the TMRxH:TMRxL register pair. This event can be a Special Event Trigger.

For more information, see [Section 21.0 “Capture/Compare/PWM Module”](#).

19.12 CCP Special Event Trigger

When any of the CCP's are configured to trigger a special event, the trigger will clear the TMRxH:TMRxL register pair. This special event does not cause a Timer1/3/5/7 interrupt. The CCP module may still be configured to generate a CCP interrupt.

In this mode of operation, the CCPRxH:CCPRxL register pair becomes the period register for Timer1/3/5/7.

Timer1/3/5/7 should be synchronized and FOSC/4 should be selected as the clock source in order to utilize the Special Event Trigger. Asynchronous operation of Timer1/3/5/7 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.

FIGURE 19-3: TIMER1/3/5/7 INCREMENTING EDGE

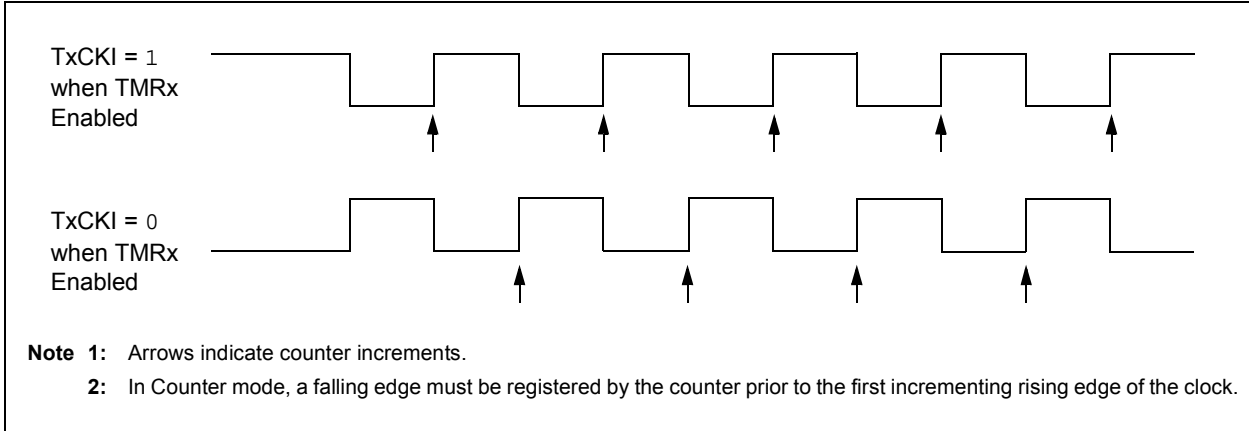


FIGURE 19-4: TIMER1/3/5/7 GATE ENABLE MODE

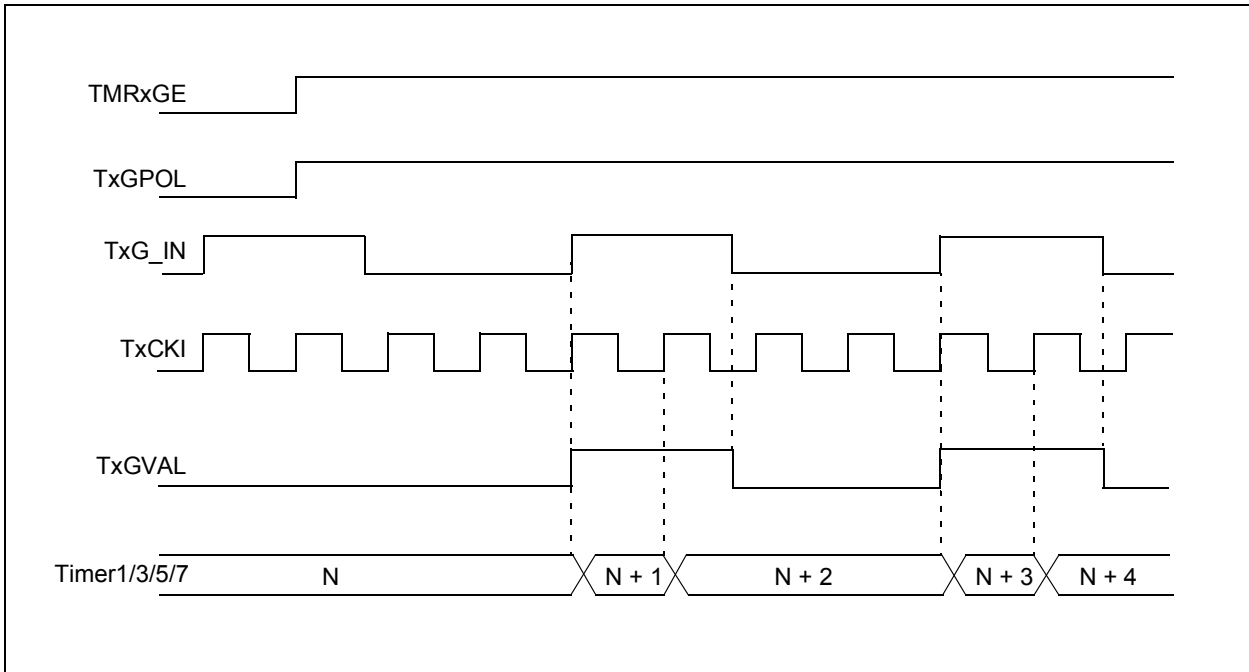


FIGURE 19-5: TIMER1/3/5/7 GATE TOGGLE MODE

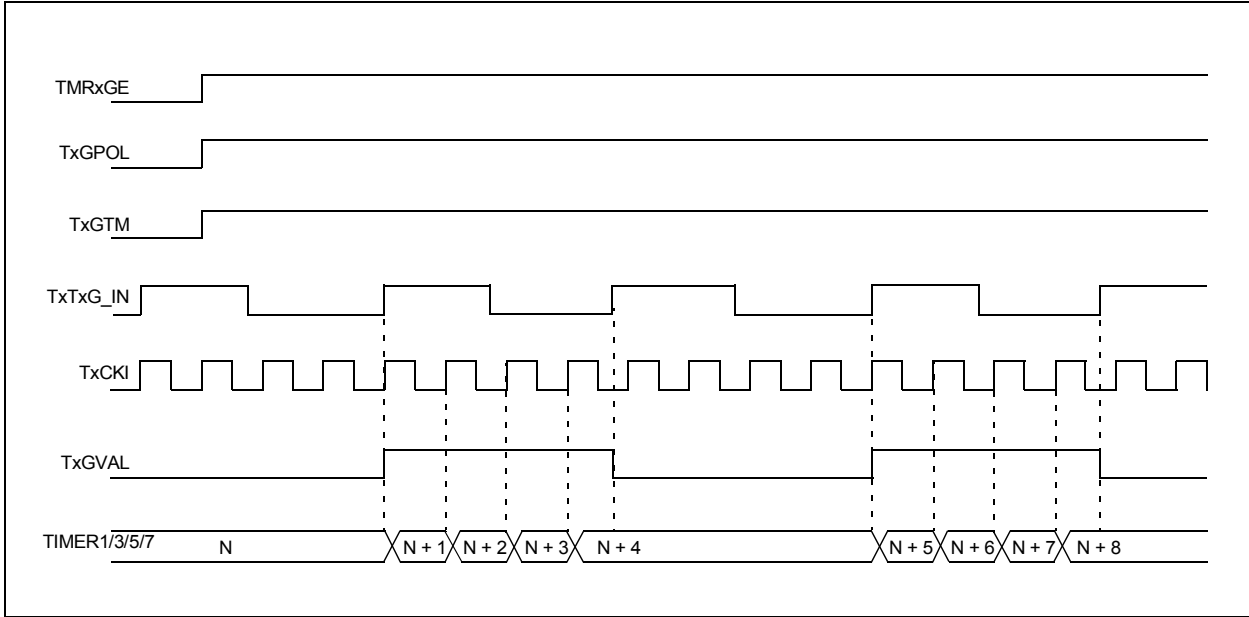


FIGURE 19-6: TIMER1/3/5/7 GATE SINGLE-PULSE MODE

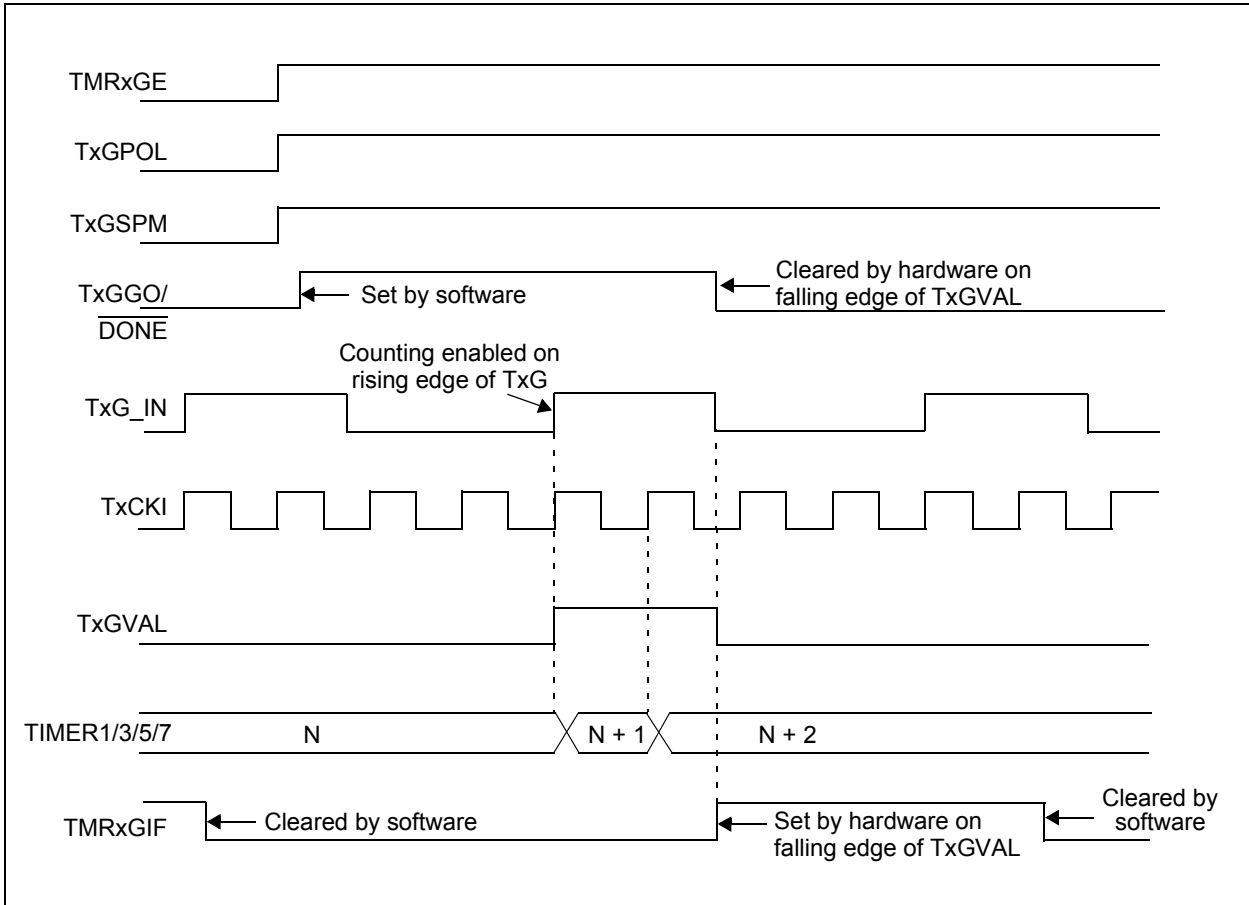
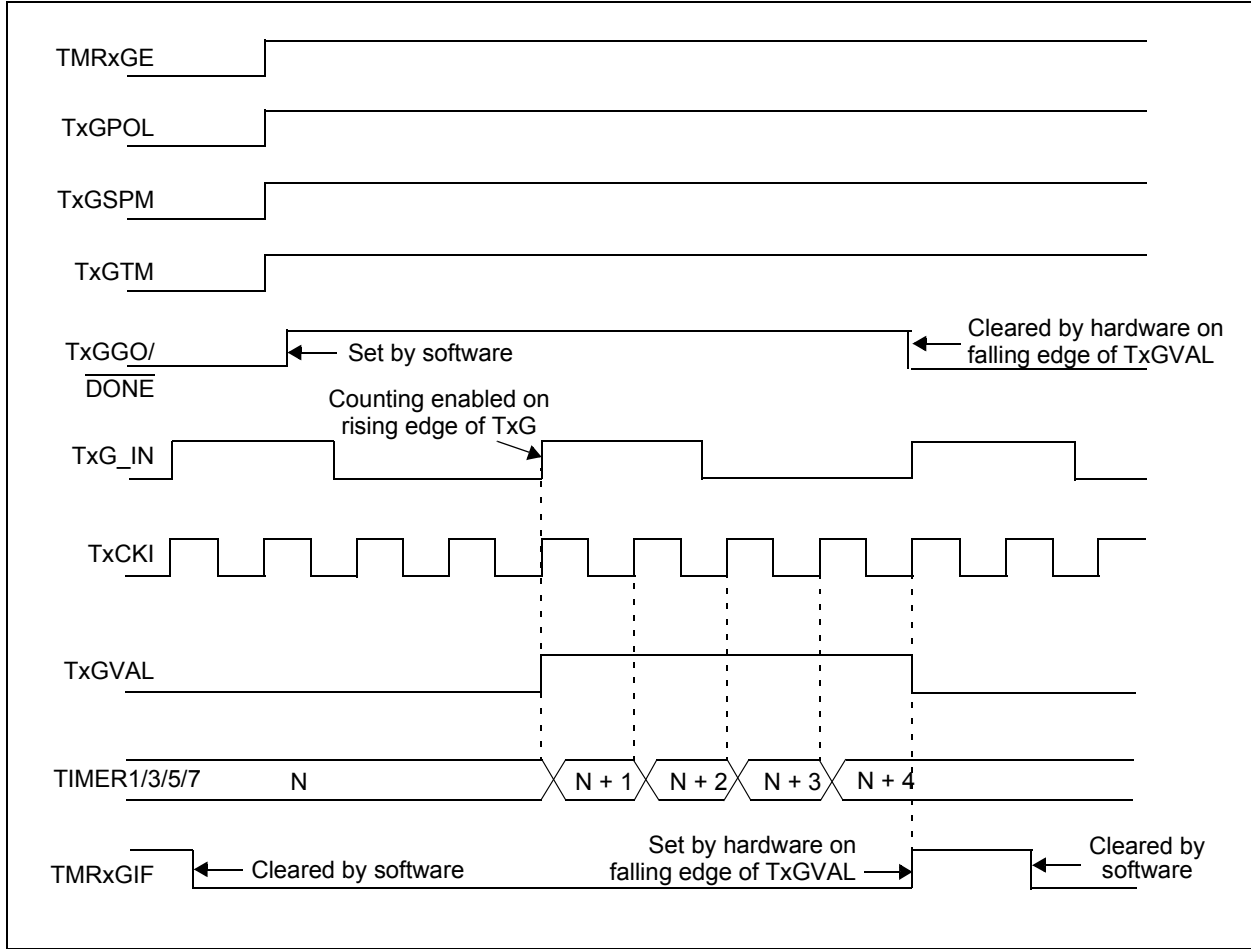


FIGURE 19-7: TIMER1/3/5/7 GATE SINGLE-PULSE AND TOGGLE COMBINED MODE



19.13 Peripheral Module Disable

When a peripheral module 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), Timer3 (TMR3MD), Timer5 (TMR5MD) and Timer7 (TMR7MD) are in the PMD1 register. See [Section 7.0 "Peripheral Module Disable \(PMD\)"](#) for more information.

TABLE 19-4: SUMMARY OF REGISTERS ASSOCIATED WITH TIMER1/3/5/7 AS A TIMER/COUNTER

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset Values on Page |
|----------|--|-----------|---------------|--------|---------|---------|---------|---------|----------------------|
| INTCON | GIE/GIEH | PEIE/GIEL | IPEN | — | INT3EDG | INT2EDG | INT1EDG | INT0EDG | 172 |
| PIE5 | TMR8IE | TMR7IE | TMR6IE | TMR5IE | TMR4IE | TMR3IE | TMR2IE | TMR1IE | 189 |
| PIE6 | — | — | — | — | TMR7GIE | TMR5GIE | TMR3GIE | TMR1GIE | 190 |
| PIR5 | TMR8IF | TMR7IF | TMR6IF | TMR5IF | TMR4IF | TMR3IF | TMR2IF | TMR1IF | 177 |
| PIR6 | — | — | — | — | TMR7GIF | TMR5GIF | TMR3GIF | TMR1GIF | 179 |
| IPR5 | TMR8IP | TMR7IP | TMR6IP | TMR5IP | TMR4IP | TMR3IP | TMR2IP | TMR1IP | 199 |
| IPR6 | — | — | — | — | TMR7GIP | TMR5GIP | TMR3GIP | TMR1GIP | 200 |
| PMD1 | TMR7MD | TMR6MD | TMR5MD | TMR4MD | TMR3MD | TMR2MD | TMR1MD | TMR0MD | 66 |
| T1CON | — | — | CKPS<1:0> | | — | SYNC | RD16 | ON | 239 |
| T1GCON | GE | GPOL | GTM | GSPM | GO/DONE | GVAL | — | — | 240 |
| T3CON | — | — | CKPS<1:0> | | — | SYNC | RD16 | ON | 239 |
| T3GCON | GE | GPOL | GTM | GSPM | GO/DONE | GVAL | — | — | 240 |
| T5CON | — | — | CKPS<1:0> | | — | SYNC | RD16 | ON | 239 |
| T5GCON | GE | GPOL | GTM | GSPM | GO/DONE | GVAL | — | — | 240 |
| T7CON | — | — | CKPS<1:0> | | — | SYNC | RD16 | ON | 239 |
| T7GCON | GE | GPOL | GTM | GSPM | GO/DONE | GVAL | — | — | 240 |
| TMR1H | Holding Register for the Most Significant Byte of the 16-bit TMR1 Register | | | | | | | | 243 |
| TMR1L | Least Significant Byte of the 16-bit TMR1 Register | | | | | | | | 243 |
| TMR3H | Holding Register for the Most Significant Byte of the 16-bit TMR3 Register | | | | | | | | 243 |
| TMR3L | Least Significant Byte of the 16-bit TMR3 Register | | | | | | | | 243 |
| TMR5H | Holding Register for the Most Significant Byte of the 16-bit TMR5 Register | | | | | | | | 243 |
| TMR5L | Least Significant Byte of the 16-bit TMR5 Register | | | | | | | | 243 |
| TMR7H | Holding Register for the Most Significant Byte of the 16-bit TMR7 Register | | | | | | | | 243 |
| TMR7L | Least Significant Byte of the 16-bit TMR7 Register | | | | | | | | 243 |
| T1CKIPPS | — | — | T1CKIPPS<5:0> | | | | | | 224 |
| T1GPPS | — | — | T1GPPS<5:0> | | | | | | 224 |
| T3CKIPPS | — | — | T3CKIPPS<5:0> | | | | | | 224 |
| T3GPPS | — | — | T3GPPS<5:0> | | | | | | 224 |
| T5CKIPPS | — | — | T5CKIPPS<5:0> | | | | | | 224 |
| T5GPPS | — | — | T5GPPS<5:0> | | | | | | 224 |
| T7CKIPPS | — | — | T7CKIPPS<5:0> | | | | | | 224 |
| T7GPPS | — | — | T7GPPS<5:0> | | | | | | 224 |

Legend: — = Unimplemented location, read as '0'. Shaded cells are not used by TIMER1/3/5/7.

20.0 TIMER2/4/6/8 MODULE

The Timer2/4/6/8 modules are 8-bit timers that incorporate the following features:

- 8-bit Timer and Period registers (TMR2 and PR2, respectively)
- Readable and writable (both registers)
- Software programmable prescaler (1:1 to 1:128)
- Software programmable postscaler (1:1 to 1:16)
- Interrupt on TMR2 match with PR2
- 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 20-1](#) for Timer2 clock sources. See [Figure 20-2](#) for a block diagram of Timer2 with HLT.

Note: Four identical Timer2 modules are implemented on this device. The timers are named Timer2, Timer4, Timer6 and Timer8. All references to Timer2 apply as well to Timer4, Timer6 and Timer8. All references to PR2 apply as well to PR4, PR6 and PR8.

FIGURE 20-1: TIMER2 CLOCK SOURCE BLOCK DIAGRAM

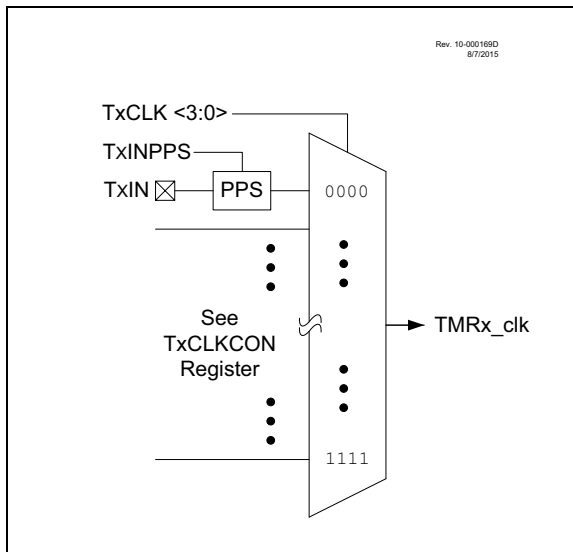
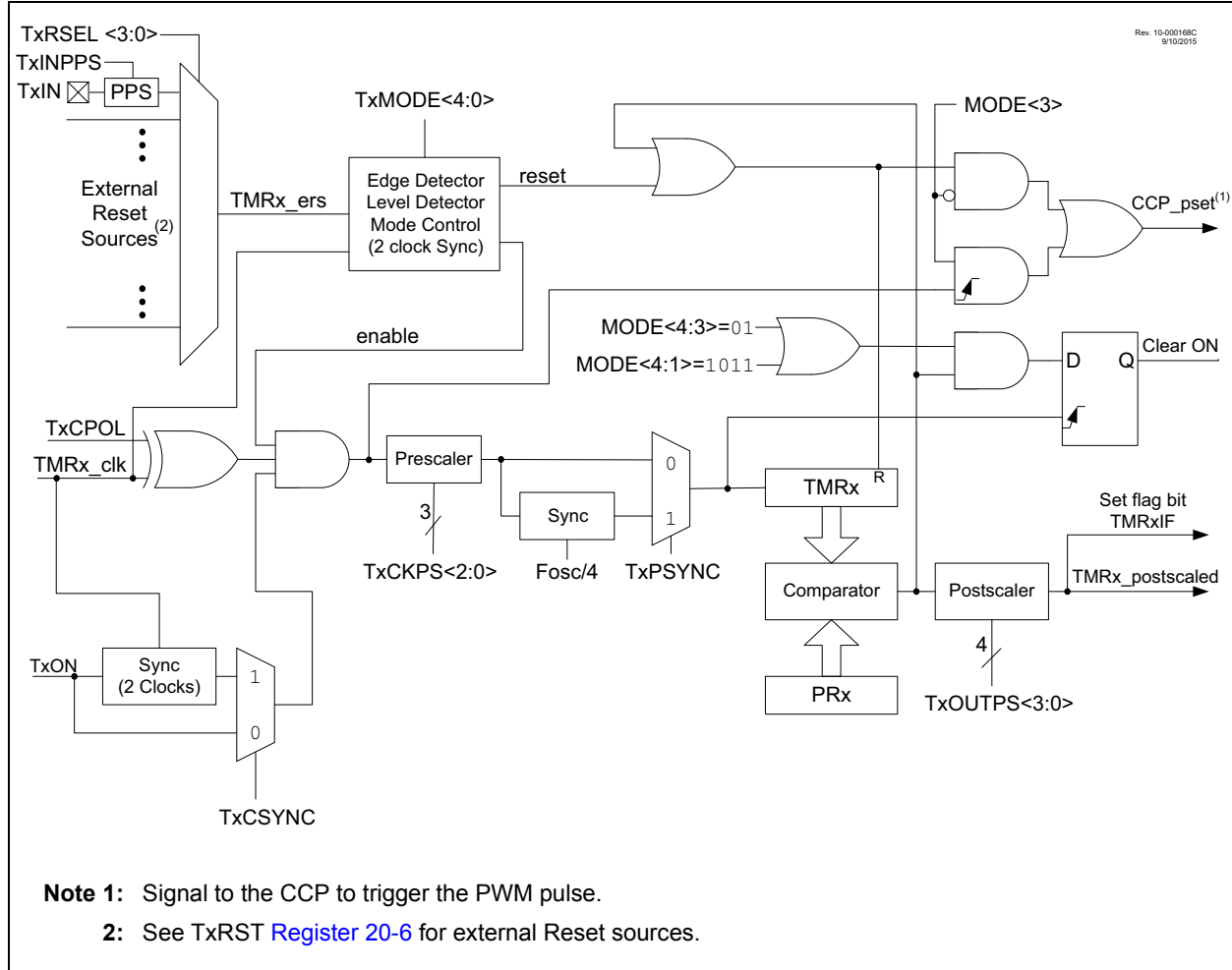


FIGURE 20-2: TIMER2 WITH HARDWARE LIMIT TIMER (HLT) BLOCK DIAGRAM



20.1 Register Definitions: Timer2/4/6/8 Control

Long bit name prefixes for the Timer2/4/6/8 peripherals are shown in [Table 20-1](#). Refer to [Section 1.4.2.2 “Long Bit Names”](#) for more information.

TABLE 20-1:

| Peripheral | Bit Name Prefix |
|------------|-----------------|
| Timer2 | T2 |
| Timer4 | T4 |
| Timer6 | T6 |
| Timer8 | T8 |

REGISTER 20-1: TMRx: TIMERx COUNTER REGISTER

| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|------------|---------|---------|---------|---------|---------|---------|---------|
| TxTMR<7:0> | | | | | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

bit 7-0 **TxTMR<7:0>**: Timerx Counter bits

REGISTER 20-2: TxPR: TIMERx PERIOD REGISTER

| R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 | R/W-1/1 |
|-----------|---------|---------|---------|---------|---------|---------|---------|
| TxPR<7:0> | | | | | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

bit 7-0 **TxPR<7:0>**: Timerx Period Register bits

REGISTER 20-3: TxCON: TIMERx CONTROL REGISTER

| | | | | | | | |
|------------|-----------|---------|---------|------------|---------|---------|---------|
| R/W/HC-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| TxON | CKPS<2:0> | | | OUTPS<3:0> | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | HC = Bit is cleared by hardware |

| | |
|---------|--|
| bit 7 | ON: Timerx On bit ⁽¹⁾ 1 = Timerx is on 0 = Timerx is off: all counters and state machines are reset |
| bit 6-4 | CKPS<2:0>: Timerx-type Clock Prescale Select bits 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 |
| bit 3-0 | OUTPS<3:0>: Timerx Output Postscaler Select bits 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 TxON bit will be auto-cleared by hardware. See [Section 20.5.1 “One-Shot Mode”](#).

REGISTER 20-4: TxHLT: TIMERx HARDWARE LIMIT CONTROL REGISTER

| | | | | | | | |
|---------|---------|---------|-----------|---------|---------|---------|---------|
| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| PSYNC | CPOL | CSYNC | MODE<4:0> | | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

- bit 7 **PSYNC:** Timerx Prescaler Synchronization Enable bit^(1, 2)
 1 = TMRx Prescaler Output is synchronized to Fosc/4
 0 = TMRx Prescaler Output is not synchronized to Fosc/4
- bit 6 **CPOL:** Timerx Clock Polarity Selection bit⁽³⁾
 1 = Falling edge of input clock clocks timer/prescaler
 0 = Rising edge of input clock clocks timer/prescaler
- bit 5 **CSYNC:** Timerx Clock Synchronization Enable bit^(4, 5)
 1 = ON register bit is synchronized to TMR2_clk input
 0 = ON register bit is not synchronized to TMR2_clk input
- bit 4-0 **MODE<4:0>:** Timerx Control Mode Selection bits^(6, 7)
 See [Table 20-2](#) for all operating modes.

- Note 1:** Setting this bit ensures that reading TMRx will return a valid data value.
- 2:** When this bit is '1', Timer2 cannot operate in Sleep mode.
- 3:** CKPOL should 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 TMRx 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 TMRx).
- 7:** When TMRx = PRx, the next clock clears TMRx, regardless of the operating mode.

TABLE 20-2: TxMODE CONFIGURATION OPTIONS^(2,6)

| MODE<4:0> | | | Output Operation | Operation | Timer Control | | |
|-----------|-------|-------------------------------|--|--|----------------------------|----------------------|--|
| <4:3> | <2:0> | <4:0> | | | Start | Reset ⁽¹⁾ | Stop |
| 11 | xxx | 24-31 | Reserved | | | | |
| 10 | 000 | 16 | Reserved | | | | |
| | 001 | 17 | Monostable edge-triggered start ⁽³⁾ | Input rising edge start timer | TMRx_ers RE and TxON = 1 | — | TxON = 0 or Next clock after TMRx = PR2 ⁽⁵⁾ |
| | 010 | 18 | | Input falling edge start timer | TMRx_ers FE and TxON = 1 | — | |
| | 011 | 19 | | Input either edge start timer | TMRx_ers edge and TxON = 1 | — | |
| | 100 | 20 | Reserved | | | | |
| | 101 | 21 | Reserved | | | | |
| | 110 | 22 | One-shot level triggered start with HLT action | Input high start timer, Input active-low Reset timer | TMRx_ers = 1 and TxON = 1 | TMRx_ers = 0 | TxON = 0 or held in Reset ⁽⁴⁾ |
| | 111 | 23 | | Input low start timer, Input active-high Reset timer | TMRx_ers = 0 and TxON = 1 | TMRx_ers = 1 | |
| 01 | 000 | 08 | One-shot SW start | SW start (no HW control) | TxON = 1 | — | TxON = 0 or Next clock after TMRx = PR2 ⁽⁴⁾ |
| | 001 | 09 | One-shot edge-triggered start ⁽³⁾ | Input rising edge start timer | TMRx_ers RE and TxON = 1 | — | |
| | 010 | 10 | | Input falling edge start timer | TMRx_ers FE and TxON = 1 | — | |
| | 011 | 11 | | Input either edge start timer | TMRx_ers edge and TxON = 1 | — | |
| | 100 | 12 | One-shot edge-triggered start with HLT action ⁽³⁾ | Input rising edge start timer; Input rising edge Reset timer | TMRx_ers RE and TxON = 1 | TMRx_ers RE | |
| | 101 | 13 | | Input falling edge start timer, Input falling edge Reset timer | TMRx_ers FE and TxON = 1 | TMRx_ers FE | |
| | 110 | 14 | | Input rising edge start timer, Input active-low Reset timer | TMRx_ers RE and TxON = 1 | TMRx_ers = 0 | |
| | 111 | 15 | | Input falling edge start timer, Input active-high Reset timer | TMRx_ers FE and TxON = 1 | TMRx_ers = 1 | |
| 00 | 000 | 00 | Roll-over pulse | SW gate (no HW control) | TxON = 1 | — | TxON = 0 |
| | 001 | 01 | | HW gate, active-high | TMRx_ers = 1 and TxON = 1 | — | TxON = 0 or TMRx_ers = 0 |
| | 010 | 02 | | HW gate, active-low | TMRx_ers = 0 and TxON = 1 | — | TxON = 0 or TMRx_ers = 1 |
| | 011 | 03 | Roll-over pulse with HLT action | Input either edge Reset timer | TxON = 1 | TMRx_ers edge | TxON = 0 |
| | 100 | 04 | | Input rising edge Reset timer | | TMRx_ers RE | |
| | 101 | 05 | | Input falling edge Reset timer | | TMRx_ers FE | |
| | 110 | 06 | | Input active-low Reset timer | | TMRx_ers = 0 | |
| 111 | 07 | Input active-high Reset timer | TMRx_ers = 1 | | | | |

- Note** 1: When TMRx = PRx, the next clock clears TMRx, regardless of the operating mode.
 2: In all operating modes, TxON = 0 stops the counter without affecting the value of TMRx
 3: In edge-triggered one-shot and monostable modes (not "SW Start" mode), the triggered-start mechanism is reset and rearmed (prepared for next start) if TxON becomes '0' (zero); the counter will not restart until an input edge occurs.
 4: When TMRx = PRx, the next clock clears TxON.
 5: When TMRx = PRx, TxON is not cleared.
 6: Both TxON and TMRx_ers are subject to clock sync delays.

REGISTER 20-5: TxCLKCON: TIMERx CLOCK SELECTION REGISTER

| | | | | | | | |
|-------|-----|-----|-----|---------|---------|---------|---------|
| U-0 | U-0 | U-0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| — | — | — | — | CS<3:0> | | | |
| bit 7 | | | | bit 0 | | | |

Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

bit 7-4

Unimplemented: Read as '0'

bit 3-0

CS<3:0>: Timerx Clock Selection bits

| CS<3:0> | TMR2 | TMR4 | TMR6 | TMR8 |
|-----------|-------------------------|-------------------------|-------------------------|-------------------------|
| | Clock Source | Clock Source | Clock Source | Clock Source |
| 1111-1001 | Reserved | Reserved | Reserved | Reserved |
| 1000 | ZCD_OUT | ZCD_OUT | ZCD_OUT | ZCD_OUT |
| 0111 | CLKREF_OUT | CLKREF_OUT | CLKREF_OUT | CLKREF_OUT |
| 0110 | SOSC | SOSC | SOSC | SOSC |
| 0101 | MFINTOSC (31 kHz) | MFINTOSC (31 kHz) | MFINTOSC (31 kHz) | MFINTOSC (31 kHz) |
| 0100 | LFINTOSC | LFINTOSC | LFINTOSC | LFINTOSC |
| 0011 | HFINTOSC | HFINTOSC | HFINTOSC | HFINTOSC |
| 0010 | Fosc | Fosc | Fosc | Fosc |
| 0001 | Fosc/4 | Fosc/4 | Fosc/4 | Fosc/4 |
| 0000 | Pin selected by T2INPPS | Pin selected by T4INPPS | Pin selected by T6INPPS | Pin selected by T8INPPS |

REGISTER 20-6: TxRST: TIMER2 EXTERNAL RESET SIGNAL SELECTION REGISTER

| | | | | | | | |
|-------|-----|-----|-----|-----------|---------|---------|---------|
| U-0 | U-0 | U-0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| — | — | — | — | RSEL<3:0> | | | |
| bit 7 | | | | bit 0 | | | |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-4 **Unimplemented:** Read as '0'

bit 3-0 **RSEL<3:0>:** Timer2 External Reset Signal Source Selection bits

| RSEL<3:0> | TMR2 | TMR4 | TMR6 | TMR8 |
|-----------|-------------------------|-------------------------|-------------------------|-------------------------|
| | Reset Source | Reset Source | Reset Source | Reset Source |
| 1111 | ZCD_OUT | ZCD_OUT | ZCD_OUT | ZCD_OUT |
| 1110 | CMP3OUT | CMP3OUT | CMP3OUT | CMP3OUT |
| 1101 | CMP2OUT | CMP2OUT | CMP2OUT | CMP2OUT |
| 1100 | CMP1OUT | CMP1OUT | CMP1OUT | CMP1OUT |
| 1011 | PWM7OUT | PWM7OUT | PWM7OUT | PWM7OUT |
| 1010 | PWM6OUT | PWM6OUT | PWM6OUT | PWM6OUT |
| 1001 | CCP5OUT | CCP5OUT | CCP5OUT | CCP5OUT |
| 1000 | CCP4OUT | CCP4OUT | CCP4OUT | CCP4OUT |
| 0111 | CCP3OUT | CCP3OUT | CCP3OUT | CCP3OUT |
| 0110 | CCP2OUT | CCP2OUT | CCP2OUT | CCP2OUT |
| 0101 | CCP1OUT | CCP1OUT | CCP1OUT | CCP1OUT |
| 0100 | TMR8 post-scaled | TMR8 post-scaled | TMR8 post-scaled | Reserved |
| 0011 | TMR6 post-scaled | TMR6 post-scaled | Reserved | TMR6 post-scaled |
| 0010 | TMR4 post-scaled | Reserved | TMR4 post-scaled | TMR4 post-scaled |
| 0001 | Reserved | TMR2 post-scaled | TMR2 post-scaled | TMR2 post-scaled |
| 0000 | Pin selected by T2INPPS | Pin selected by T4INPPS | Pin selected by T6INPPS | Pin selected by T8INPPS |

20.2 Timer2 Operation

The 3-bit prescaler on the clock input allows for several prescaler options, from direct input to divide-by-128. These options are selected by the prescaler control bits CKPS<2:0> of the T2CON register.

The value of TMR2 is compared to that of the Period register, PR2, on each clock cycle. When the two values match, the comparator resets the value of TMR2 to 00h on the next cycle and drives the 4-bit output counter/postscaler (see [Section 20.3 “Timer2 Interrupt”](#)). In addition, the Timer can be Reset through the use of an external Reset signal as outlined in [Section 20.5 “External Reset Sources”](#).

The TMR2 and PR2 registers are both directly readable and writable. The TMR2 register is cleared on any device Reset, whereas the PR2 register initializes to FFh. Both the prescaler and postscaler counters are cleared on the following events:

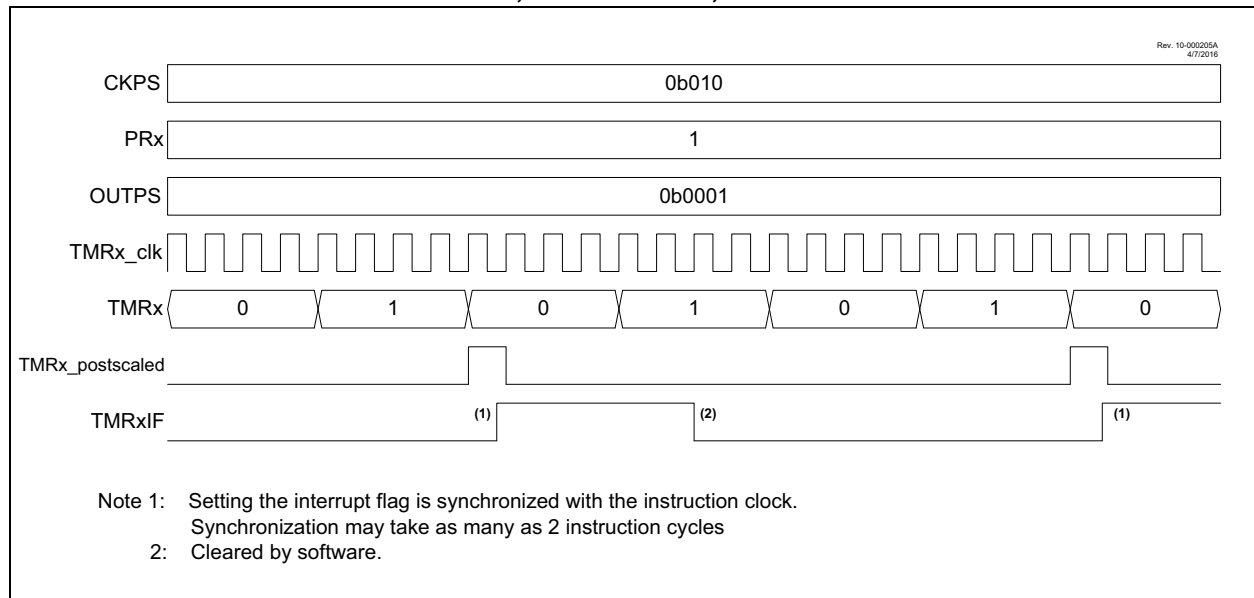
- a write to the TMR2 register
- a write to the T2CON register
- Power-on Reset (POR)
- Brown-out Reset (BOR)
- MCLR Reset
- Windowed Watchdog Timer (WWDT) Reset
- Stack Overflow Reset
- Stack Underflow Reset
- RESET Instruction
- External Reset Source events, which resets the timer.

Note: TMR2 is not cleared when T2CON is written.

20.3 Timer2 Interrupt

Timer2 can also generate a device interrupt. The interrupt is generated when the postscaler counter matches one of 16 postscale options (from 1:1 through 1:16), which is selected with the postscaler control bits, OUTPS<3:0> of the T2CON register. The interrupt is enabled by setting the TMR2 Interrupt Enable bit, TMR2IE, of the PIE4 register. The interrupt timing is illustrated in [Figure 20-3](#).

FIGURE 20-3: TIMER2 PRESCALER, POSTSCALER, AND INTERRUPT TIMING DIAGRAM



20.4 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<3:0> bits of the T2CON register. The PR2 postscaler is incremented each time the TMR2 value matches the PR2 value. This signal can be selected as an input to several other input modules:

- The CRC memory scanner, as a trigger for triggered mode
- The ADC module, as an auto-conversion trigger
- Gate source for Timer1/3/5/7
- CWG, as an auto-shutdown source
- Alternate SPI clock
- Reset signals for other instances of itself (Timer2/4/6/8)

In addition, the Timer2 is also used by the CCP module for pulse generation in PWM mode. Both the actual TMR2 value as well as other internal signals are sent to the CCP module to properly clock both the period and pulse width of the PWM signal. See **Section 21.5 “PWM Overview”** and **Section 22.0 “Pulse-Width Modulation (PWM 6/7)”** for more details on setting up Timer2 for use with the CCP and PWM.

20.5 External Reset Sources

In addition to the clock source, Timer2 also takes in an external Reset source. This external Reset source is selected for Timer2, Timer4, Timer6 and Timer8 with the T2RST, T4RST, T6RST and T8RST registers, respectively. This source can control the starting and stopping of the timer, as well as resetting the timer, depending on which mode the timer is in. The mode of the timer is controlled by the MODE<4:0> bits of the TxHLT register.

20.5.1 ONE-SHOT MODE

The MODE<3> bit of the TxHLT register controls whether the timer is in either the One-Shot mode or the original Normal Period mode. When this bit is set, the timer acts in the One-Shot mode, meaning that upon the timer register matching the PRx period register, the timer will stop incrementing until the timer is manually started again.

20.6 Operation Examples

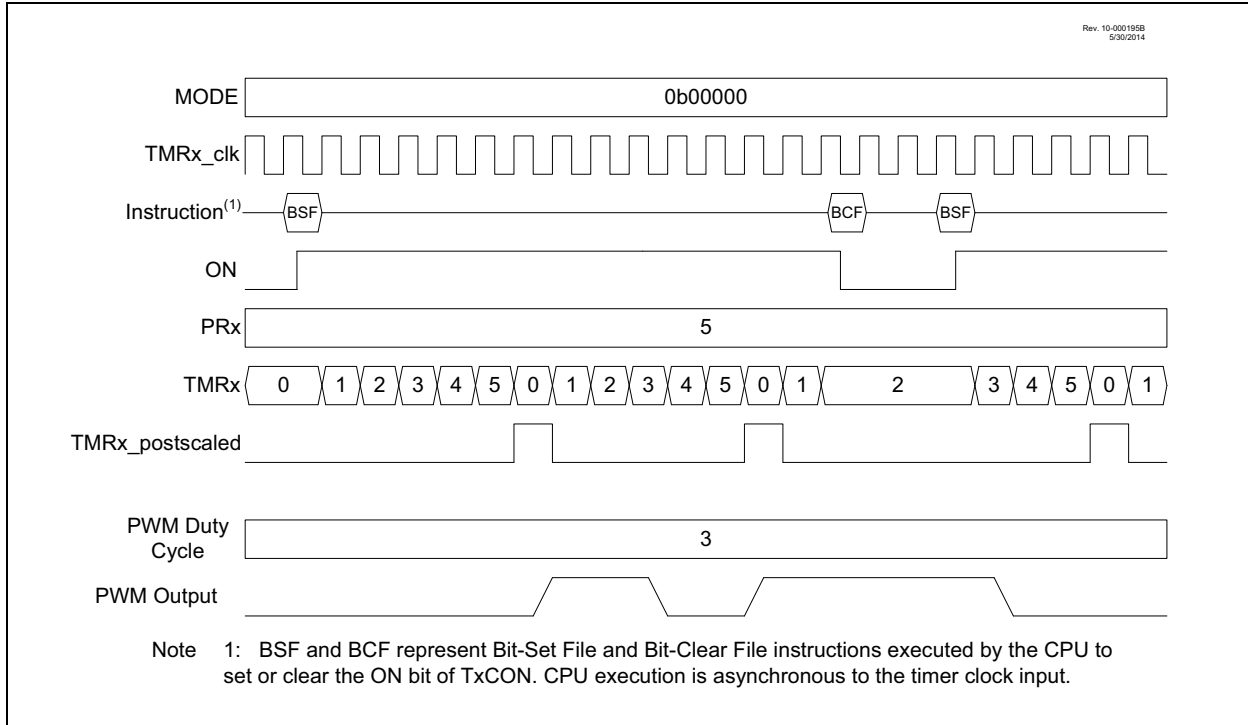
Unless otherwise specified, the following notes apply to the following timing diagrams:

- Both the prescaler and postscaler are set to 1:1 (both the CKPS and OUTPS bits in the TxCON register are cleared).
- The diagrams illustrate any clock except FOSC/4 and show clock-sync delays of at least two full cycles for both ON and Timer2_ers. When using FOSC/4, the clock-sync delay is at least one instruction period for Timer2_ers; ON applies in the next instruction period.
- ON and Timer2_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 **Section 21.5 “PWM Overview”** and **Section 22.0 “Pulse-Width Modulation (PWM 6/7)”**. The signals are not a part of the Timer2 module.

20.6.1 SOFTWARE GATE MODE

The Software Gate 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 TMRx count equals the PRx period count the timer resets on the next clock and continues counting from 0. Operation with the ON bit software controlled is illustrated in Figure 20-4. With $PRx = 5$, the counter advances until $TMRx = 5$, and goes to zero with the next clock.

FIGURE 20-4: SOFTWARE GATE MODE TIMING DIAGRAM



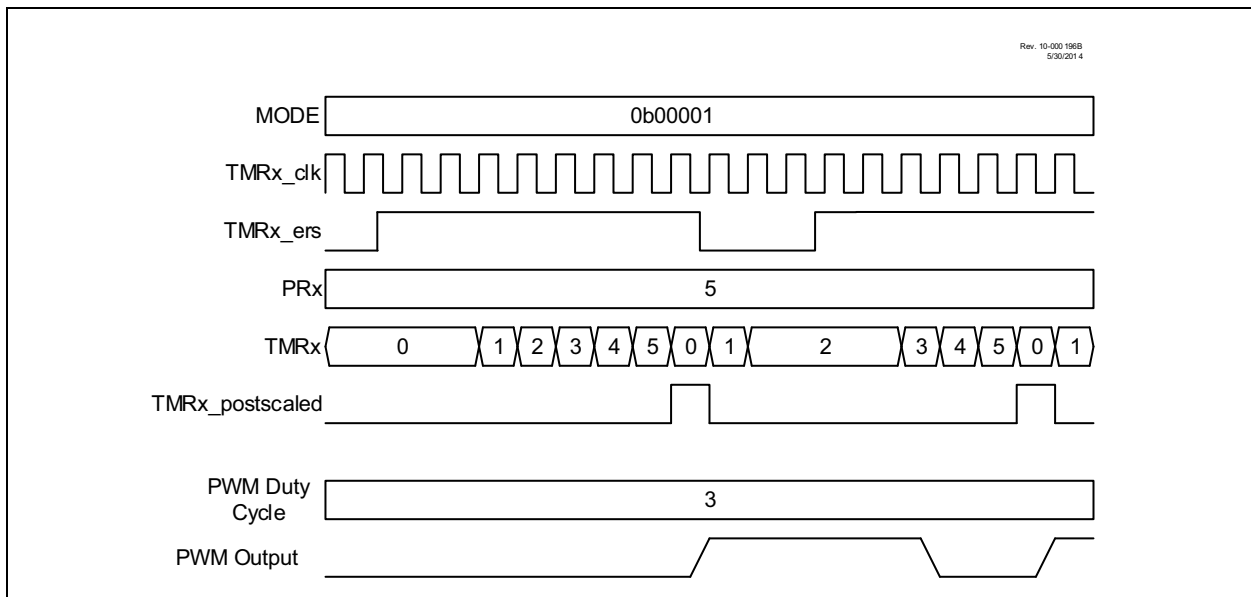
20.6.2 HARDWARE GATE MODE

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<4:0> = 00001 then the timer is stopped when the external signal is high. When MODE<4:0> = 00010, the timer is stopped when the external signal is low.

Figure 20-5 illustrates the Hardware Gating mode for MODE<4:0> = 00001 in which a high input level stops the counter.

FIGURE 20-5: HARDWARE GATE MODE TIMING DIAGRAM



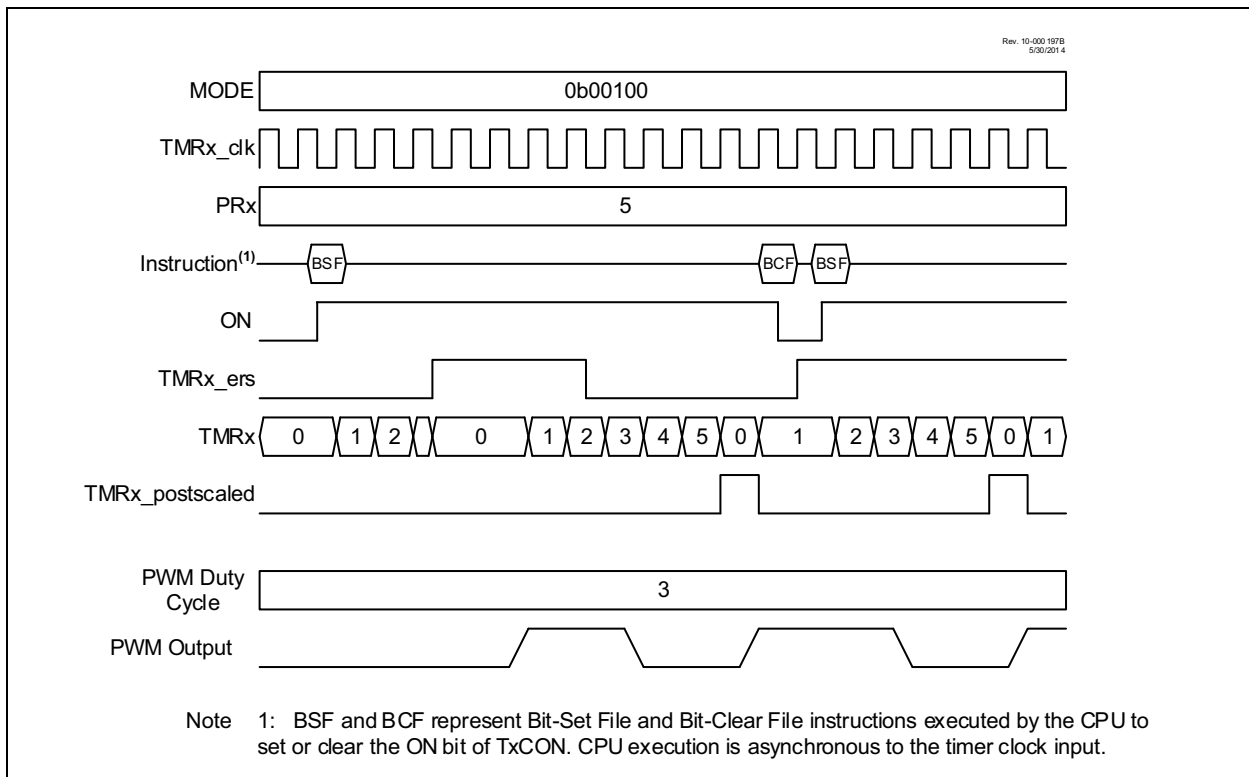
20.6.3 EDGE-TRIGGERED HARDWARE LIMIT MODE

In Edge-Triggered 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<4:0> = 00011)
- Reset on rising edge (MODE<4:0> = 00100)
- Reset on falling edge (MODE<4:0> = 00101)

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 20-6](#).

FIGURE 20-6: EDGE-TRIGGERED HARDWARE LIMIT MODE TIMING DIAGRAM

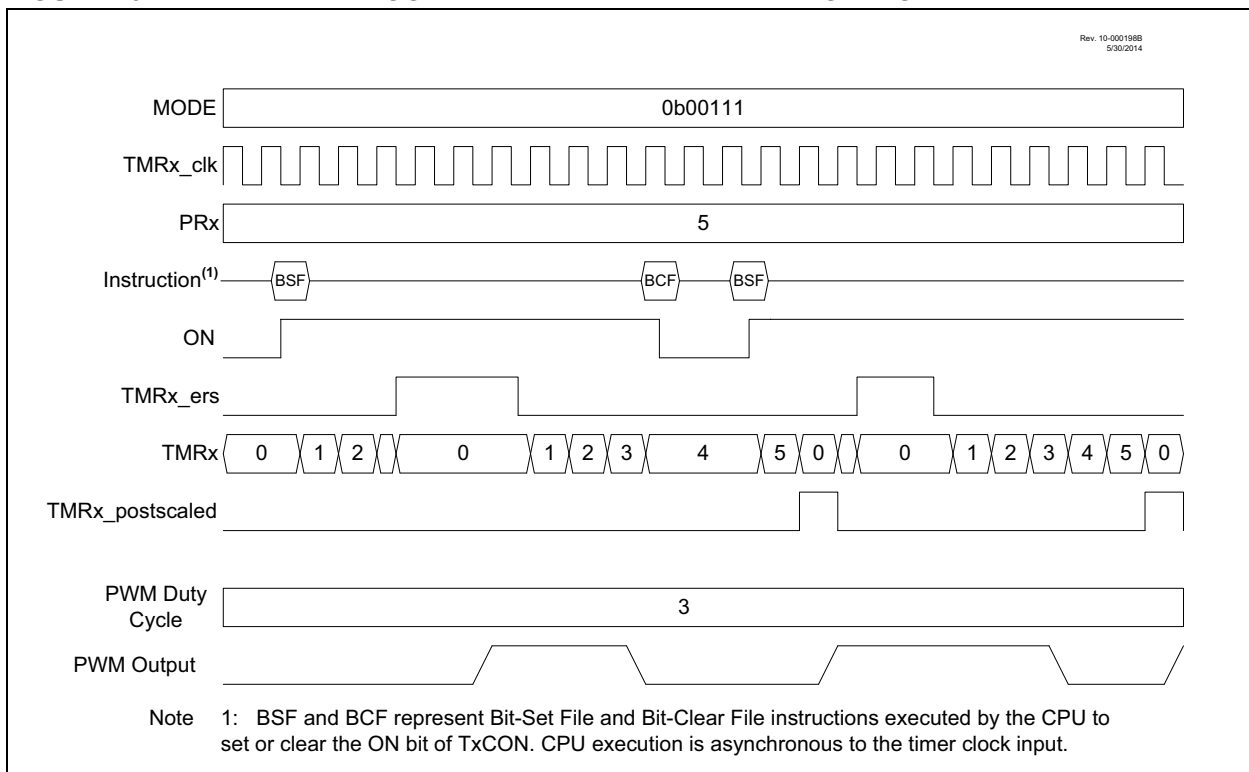


20.6.4 LEVEL-TRIGGERED HARDWARE LIMIT MODE

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 20-7. Selecting MODE<4:0> = 00110 will cause the timer to reset on a low level external signal. Selecting MODE<4:0> = 00111 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 an external signal Reset will set the PWM output high after a two clock synchronization delay or the timer matches the PRx period value. The PWM output will remain high until the external signal is released and the timer counts up to match the CCPRx pulse-width value.

FIGURE 20-7: LEVEL-TRIGGERED HARDWARE LIMIT TIMING DIAGRAM

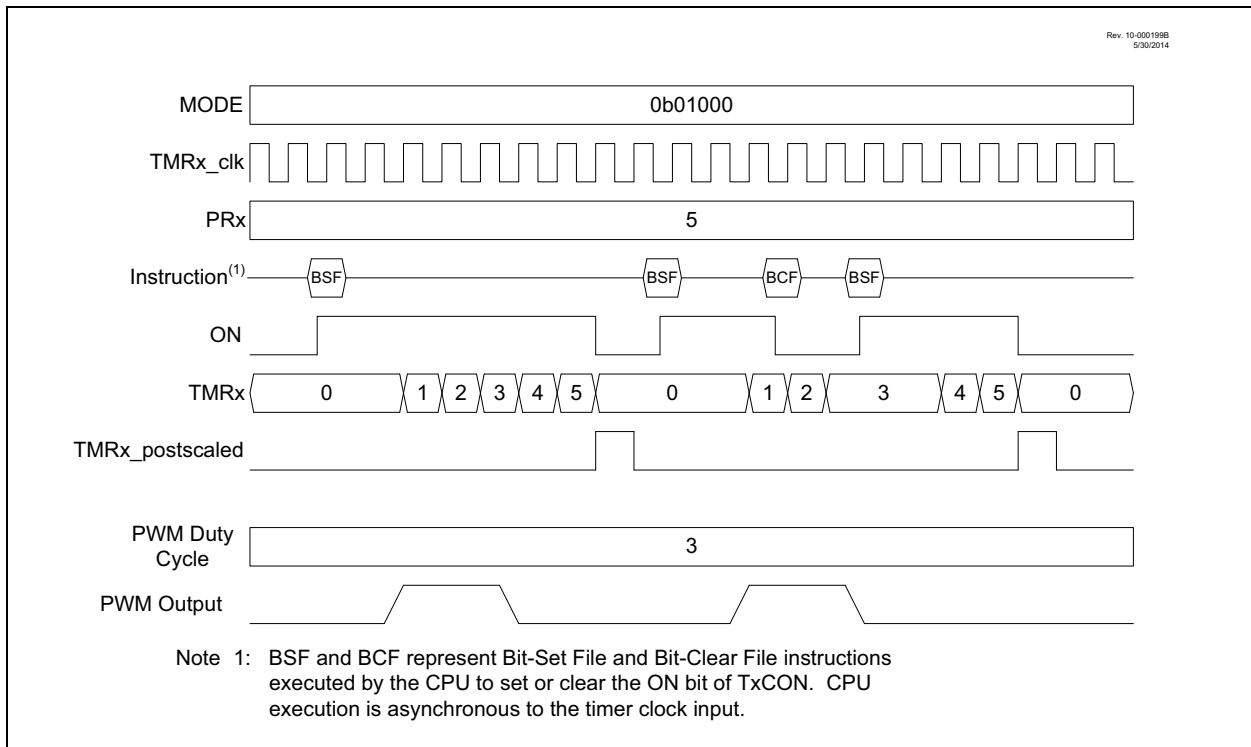


20.6.5 ONE-SHOT MODE

In One-Shot mode, the timer resets and the ON bit is cleared when the timer value matches the PRx period value. The ON bit must be set by software to start another timer cycle. Setting MODE<4:0> = 01000 selects One-Shot mode which is illustrated in Figure 20-8. 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, 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 software sets the ON bit to start another cycle. If software clears the ON bit after the CCPRx match but before the PRx 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 PRx period count match.

FIGURE 20-8: ONE-SHOT MODE TIMING DIAGRAM



20.6.6 EDGE-TRIGGERED ONE-SHOT MODE

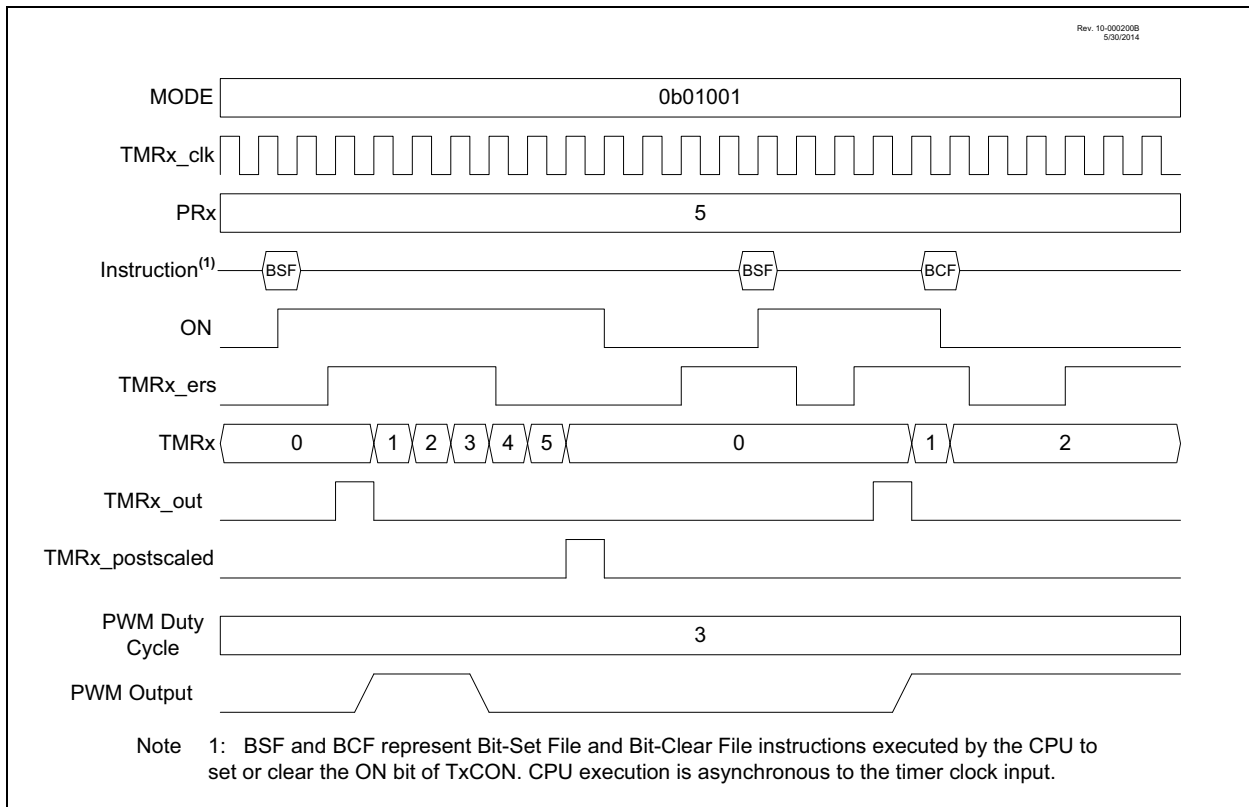
The Edge-Triggered One-Shot modes start the timer on an edge from the external signal input, after the ON bit is set, and clear the ON bit when the timer matches the PRx period value. The following edges will start the timer:

- Rising edge (MODE<4:0> = 01001)
- Falling edge (MODE<4:0> = 01010)
- Rising or Falling edge (MODE<4:0> = 01011)

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 20-9](#) illustrates operation in the rising edge One-Shot mode.

When this 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 PRx period count match.

FIGURE 20-9: EDGE-TRIGGERED ONE-SHOT MODE TIMING DIAGRAM



20.6.7 EDGE-TRIGGERED HARDWARE LIMIT ONE-SHOT MODE

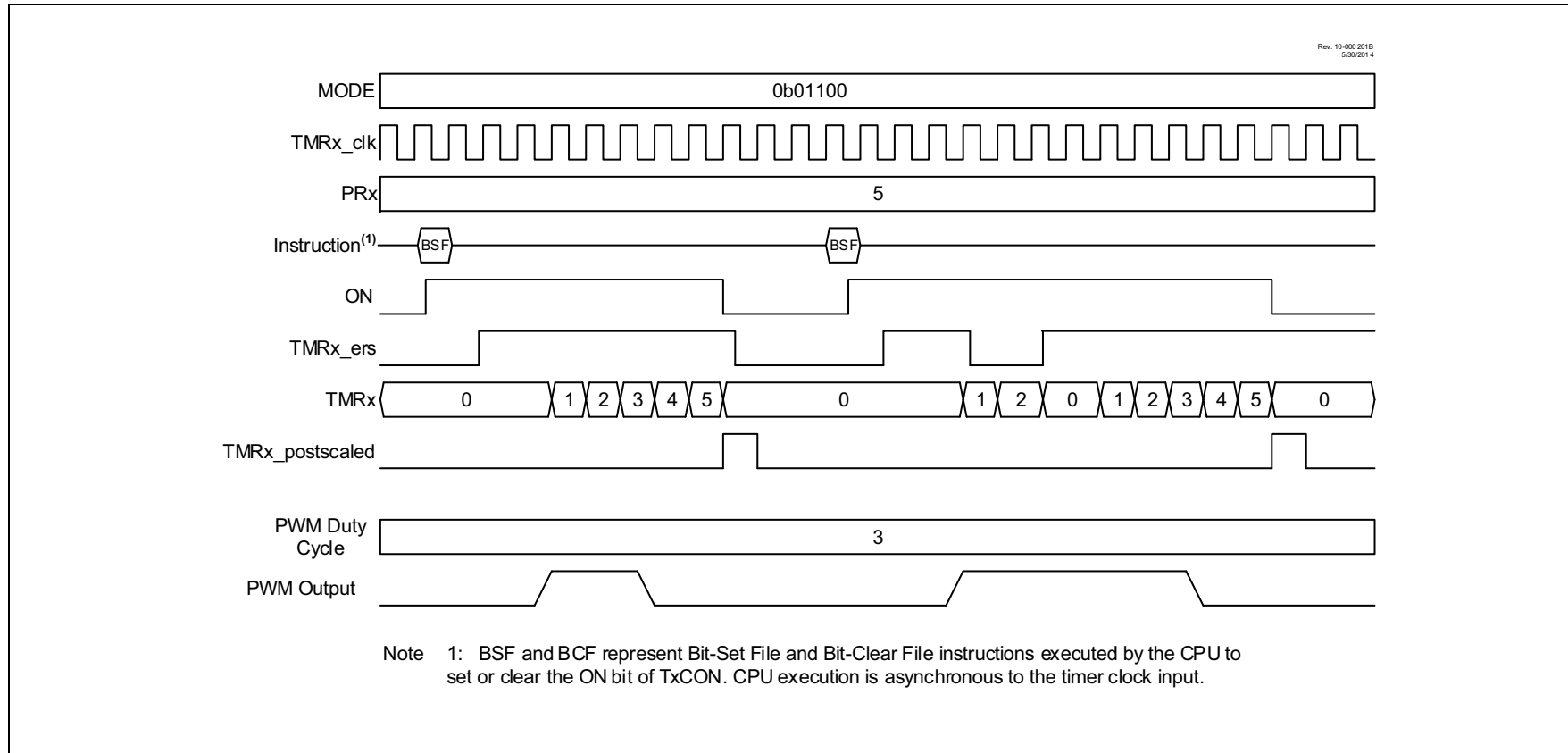
In Edge-Triggered Hardware Limit One-Shot modes the timer starts on the first external signal edge after the ON bit is set and resets on all subsequent edges. Only the first edge after the ON bit is set is needed to start the timer. The counter will resume counting automatically two clocks after all subsequent external Reset edges. Edge triggers are as follows:

- Rising edge Start and Reset
(MODE<4:0> = 01100)
- Falling edge Start and Reset
(MODE<4:0> = 01101)

The timer resets and clears the ON bit when the timer value matches the PRx period value. External signal edges will have no effect until after software sets the ON bit. [Figure 20-10](#) 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 PRx period match unless an external signal edge resets the timer before the match occurs.

FIGURE 20-10: EDGE-TRIGGERED HARDWARE LIMIT ONE-SHOT TIMING DIAGRAM



20.6.8 LEVEL 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 when the external signal level relinquishes the Reset. Reset levels are selected as follows:

- High Reset level (MODE<4:0> = 01110)
- Low Reset level (MODE<4:0> = 01111)

When the timer count matches the PRx period count then the timer is reset and the ON bit is cleared. When the ON bit is cleared by either a PRx match or by software control a new external signal edge is required after the ON bit is set to start the counter.

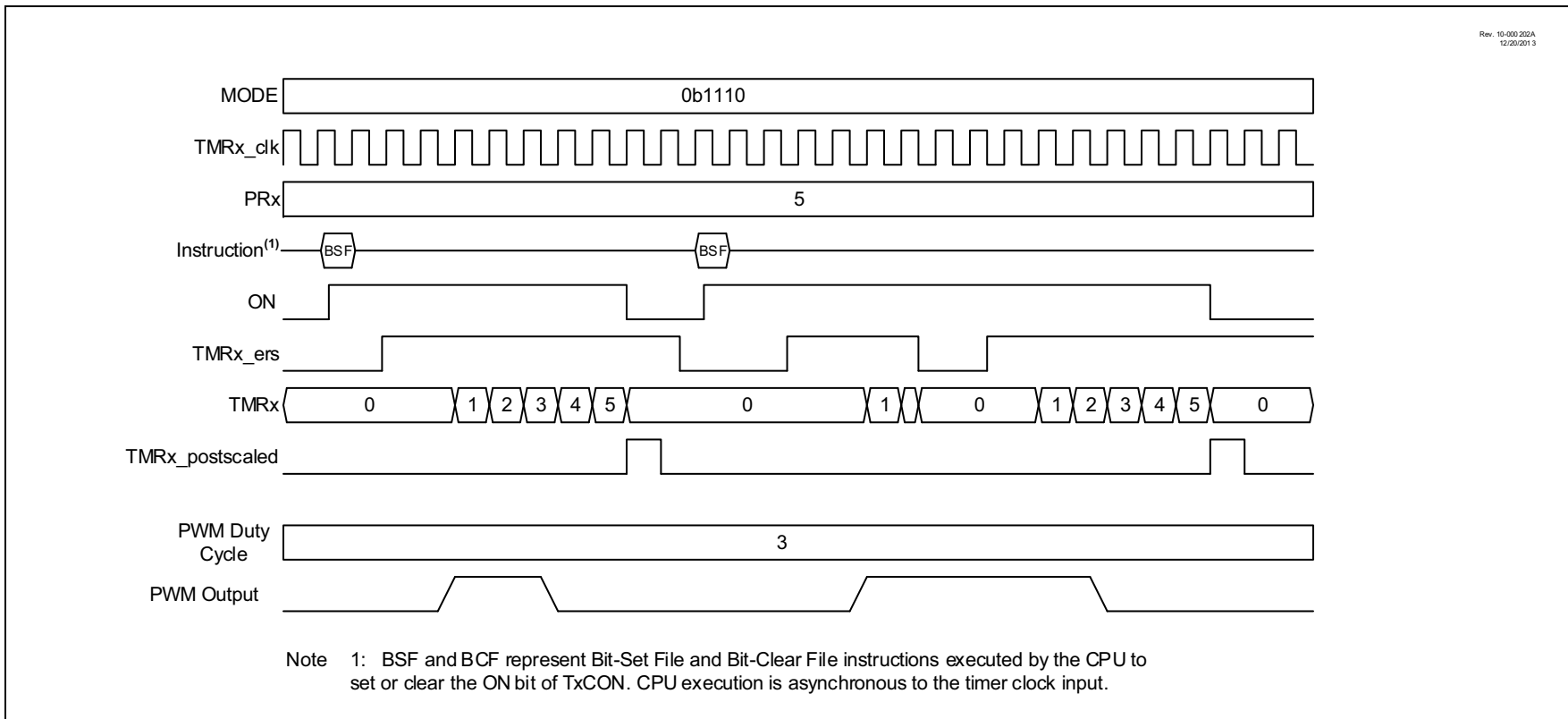
When Level Triggered Reset One-Shot mode is used in conjunction with the CCP PWM operation the PWM drive goes active with the external signal edge that starts the timer. The PWM drive goes inactive when the timer count equals the CCPRx pulse-width count. The PWM drive does not go active when the timer count clears at the PRx period count match.

20.7 Timer2 Operation During Sleep

When $\overline{\text{TxPSYNC}} = 1$, Timer2 cannot be operated while the processor is in Sleep mode. The contents of the TMR2 and PR2 registers will remain unchanged while processor is in Sleep mode.

When $\overline{\text{TxPSYNC}} = 0$, Timer2 will operate in Sleep as long as the clock source selected is also still running. Selecting the LFINTOSC, MFINTOSC, or HFINTOSC oscillator as the timer clock source will keep the selected oscillator running during Sleep.

FIGURE 20-11: LEVEL-TRIGGERED HARDWARE LIMIT ONE-SHOT MODE TIMING DIAGRAM



PIC18(L)F67K40

TABLE 20-3: SUMMARY OF REGISTERS ASSOCIATED WITH TIMER2

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on Page |
|----------|--|-----------|---------------------------|-----------|-------------|---------|-------------|---------|------------------|
| CCPTMRS1 | — | — | P7TSEL<1:0> | | P6TSEL<1:0> | | C5TSEL<1:0> | | 278 |
| CCPTMRS0 | C4TSEL<1:0> | | C3TSEL<1:0> | | C2TSEL<1:0> | | C1TSEL<1:0> | | 277 |
| INTCON | GIE/GIEH | PEIE/GIEL | IPEN | — | INT3EDG | INT2EDG | INT1EDG | INT0EDG | 172 |
| PIE5 | TMR8IE | TMR7IE | TMR6IE | TMR5IE | TMR4IE | TMR3IE | TMR2IE | TMR1IE | 189 |
| PIR5 | TMR8IF | TMR7IF | TMR6IF | TMR5IF | TMR4IF | TMR3IF | TMR2IF | TMR1IF | 178 |
| IPR5 | TMR8IP | TMR7IP | TMR6IP | TMR5IP | TMR4IP | TMR3IP | TMR2IP | TMR1IP | 199 |
| PMD1 | TMR7MD | TMR6MD | TMR5MD | TMR4MD | TMR3MD | TMR2MD | TMR1MD | TMR0MD | 66 |
| PMD2 | — | — | CWGMD | — | DSMMD | SMT2MD | SMT1MD | TMR8MD | 67 |
| PR2 | Timer2 Module Period Register | | | | | | | | 261* |
| TMR2 | Holding Register for the 8-bit TMR2 Register | | | | | | | | 261* |
| T2CON | ON | CKPS<2:0> | | | OUTPS<3:0> | | | | 256 |
| T2CLKCON | — | — | — | — | — | CS<2:0> | | | 259 |
| T2RST | — | — | — | — | RSEL<3:0> | | | | 260 |
| T2HLT | PSYNC | CPOL | CSYNC | MODE<4:0> | | | | | 257 |
| PR4 | Timer4 Module Period Register | | | | | | | | 261* |
| TMR4 | Holding Register for the 8-bit TMR4 Register | | | | | | | | 261* |
| T4CON | ON | CKPS<2:0> | | | OUTPS<3:0> | | | | 256 |
| T4CLKCON | — | — | — | — | — | CS<2:0> | | | 259 |
| T4RST | — | — | — | — | RSEL<3:0> | | | | 260 |
| T4HLT | PSYNC | CPOL | CSYNC | MODE<4:0> | | | | | 257 |
| PR6 | Timer6 Module Period Register | | | | | | | | 261* |
| TMR6 | Holding Register for the 8-bit TMR6 Register | | | | | | | | 261* |
| T6CON | ON | CKPS<2:0> | | | OUTPS<3:0> | | | | 256 |
| T6CLKCON | — | — | — | — | — | CS<2:0> | | | 259 |
| T6RST | — | — | — | — | RSEL<3:0> | | | | 260 |
| T6HLT | PSYNC | CPOL | CSYNC | MODE<4:0> | | | | | 257 |
| PR8 | Timer8 Module Period Register | | | | | | | | 261* |
| TMR8 | Holding Register for the 8-bit TMR8 Register | | | | | | | | 261* |
| T8CON | ON | CKPS<2:0> | | | OUTPS<3:0> | | | | 256 |
| T8CLKCON | — | — | — | — | — | CS<2:0> | | | 259 |
| T8RST | — | — | — | — | RSEL<3:0> | | | | 260 |
| T8HLT | $\overline{\text{PSYNC}}$ | CPOL | $\overline{\text{CSYNC}}$ | MODE<4:0> | | | | | 257 |
| T2INPPS | — | — | T2INPPS<5:0> | | | | | 224 | |
| T4INPPS | — | — | T4INPPS<5:0> | | | | | 224 | |
| T6INPPS | — | — | T6INPPS<5:0> | | | | | 224 | |
| T8INPPS | — | — | T8INPPS<5:0> | | | | | 224 | |

Legend: — = unimplemented location, read as '0'. Shaded cells are not used for Timer2 module.

* Page provides register information.

21.0 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.

This family of devices contains five standard Capture/Compare/PWM modules (CCP1, CCP2, CCP3, CCP4, and CCP5). Each individual CCP module can select the timer source that controls the module. Each module has an independent timer selection which can be accessed using the CxTSEL bits in the CCPTMRS0 register ([Register 21-2](#)). The default timer selection is TMR1 when using Capture/Compare mode and TMR2 when using PWM mode in the CCPx module.

Please note that the Capture/Compare mode operation is described with respect to TMR1 and the PWM mode operation is described with respect to TMR2 in the following sections.

The Capture and Compare functions are identical for all CCP modules.

Note 1: In devices with more than one CCP module, it is very important to pay close attention to the register names used. A number placed after the module acronym is used to distinguish between separate modules. For example, the CCP1CON and CCP2CON control the same operational aspects of two completely different CCP modules.

2: Throughout this section, generic references to a CCP module in any of its operating modes may be interpreted as being equally applicable to CCPx module. Register names, module signals, I/O pins, and bit names may use the generic designator 'x' to indicate the use of a numeral to distinguish a particular module, when required.

21.1 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).

21.1.1 CCP MODULES AND TIMER RESOURCES

The CCP modules utilize Timers 1 through 8 that vary with the selected mode. Various timers are available to the CCP modules in Capture, Compare or PWM modes, as shown in [Table 21-1](#).

TABLE 21-1: CCP MODE – TIMER RESOURCE

| CCP Mode | Timer Resource |
|----------|-----------------------------------|
| Capture | Timer1, Timer3, Timer5, or Timer7 |
| Compare | |
| PWM | Timer2, Timer4, Timer6, or Timer8 |

The assignment of a particular timer to a module is determined by the timer to CCP enable bits in the CCPTMRS0/1 registers (see [Register 21-2](#) and [Register 21-3](#)) 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.

21.1.2 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.

21.2 Register Definitions: CCP Control

Long bit name prefixes for the CCP peripherals are shown in Table 21-2. Refer to Section 1.4.2.2 “Long Bit Names” for more information.

TABLE 21-2:

| Peripheral | Bit Name Prefix |
|------------|-----------------|
| CCP1 | CCP1 |
| CCP2 | CCP2 |
| CCP3 | CCP3 |
| CCP4 | CCP4 |
| CCP5 | CCP5 |

REGISTER 21-1: CCPxCON: CCPx CONTROL REGISTER

| | | | | | | | |
|---------|-----|-----|---------|-----------|---------|---------|---------|
| R/W-0/0 | U-0 | R-x | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| EN | — | OUT | FMT | MODE<3:0> | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|-------------------|------------------|------------------------------------|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared |
| | | x = Bit is unknown |

bit 7 **EN:** CCP Module Enable bit
 1 = CCP is enabled
 0 = CCP is disabled

bit 6 **Unimplemented:** Read as '0'

bit 5 **OUT:** CCPx Output Data bit (read-only)

bit 4 **FMT:** CCPW (pulse-width) Alignment bit
 MODE = Capture mode:
 Unused
 MODE = Compare mode:
 Unused
 MODE = PWM mode:
 1 = Left-aligned format
 0 = Right-aligned format

- Note 1:** The set and clear operations of the Compare mode are reset by setting MODE = 4'b0000 or EN = 0.
- Note 2:** When MODE = 0001 or 1011, 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 purpose only.

REGISTER 21-1: CCPxCON: CCPx CONTROL REGISTER (CONTINUED)

bit 3-0

MODE<3:0>: CCPx Mode Select bits

| MODE | 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 | | — |

- Note 1:** The set and clear operations of the Compare mode are reset by setting MODE = 4'b0000 or EN = 0.
- Note 2:** When MODE = 0001 or 1011, 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 purpose only.

REGISTER 21-2: CCPTMRS0: CCP TIMERS CONTROL REGISTER 0

| | | | | | | | |
|-------------|---------|-------------|---------|-------------|---------|-------------|---------|
| R/W-0/0 | R/W-1/1 | R/W-0/0 | R/W-1/1 | R/W-0/0 | R/W-1/1 | R/W-0/0 | R/W-1/1 |
| C4TSEL<1:0> | | C3TSEL<1:0> | | C2TSEL<1:0> | | C1TSEL<1:0> | |
| bit 7 | | | | | | bit 0 | |

Legend:

| | | |
|-------------------|------------------|------------------------------------|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared |
| | | x = Bit is unknown |

- bit 7-6 **C4TSEL<1:0>**: CCP4 Timer Selection bits
 11 = CCP4 is based off Timer7 in Capture/Compare mode and Timer8 in PWM mode
 10 = CCP4 is based off Timer5 in Capture/Compare mode and Timer6 in PWM mode
 01 = CCP4 is based off Timer3 in Capture/Compare mode and Timer4 in PWM mode
 00 = CCP4 is based off Timer1 in Capture/Compare mode and Timer2 in PWM mode
- bit 5-4 **C3TSEL<1:0>**: CCP3 Timer Selection bits
 11 = CCP3 is based off Timer7 in Capture/Compare mode and Timer8 in PWM mode
 10 = CCP3 is based off Timer5 in Capture/Compare mode and Timer6 in PWM mode
 01 = CCP3 is based off Timer3 in Capture/Compare mode and Timer4 in PWM mode
 00 = CCP3 is based off Timer1 in Capture/Compare mode and Timer2 in PWM mode
- bit 3-2 **C2TSEL<1:0>**: CCP2 Timer Selection bits
 11 = CCP2 is based off Timer7 in Capture/Compare mode and Timer8 in PWM mode
 10 = CCP2 is based off Timer5 in Capture/Compare mode and Timer6 in PWM mode
 01 = CCP2 is based off Timer3 in Capture/Compare mode and Timer4 in PWM mode
 00 = CCP2 is based off Timer1 in Capture/Compare mode and Timer2 in PWM mode
- bit 1-0 **C1TSEL<1:0>**: CCP1 Timer Selection bits
 11 = CCP1 is based off Timer7 in Capture/Compare mode and Timer8 in PWM mode
 10 = CCP1 is based off Timer5 in Capture/Compare mode and Timer6 in PWM mode
 01 = CCP1 is based off Timer3 in Capture/Compare mode and Timer4 in PWM mode
 00 = CCP1 is based off Timer1 in Capture/Compare mode and Timer2 in PWM mode

REGISTER 21-3: CCPTMRS1: CCP TIMERS CONTROL REGISTER 1

| | | | | | | | |
|-------|-----|-------------|---------|-------------|---------|-------------|---------|
| U-0 | U-0 | R/W-0/0 | R/W-1/1 | R/W-0/0 | R/W-1/1 | R/W-0/0 | R/W-1/1 |
| — | — | P7TSEL<1:0> | | P6TSEL<1:0> | | C5TSEL<1:0> | |
| bit 7 | | | | | | bit 0 | |

Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7-6 **Unimplemented:** Read as '0'

bit 5-4 **P7TSEL<1:0>:** PWM7 Timer Selection bits

11 = PWM7 based on TMR8

10 = PWM7 based on TMR6

01 = PWM7 based on TMR4

00 = PWM7 based on TMR2

bit 3-2 **P6TSEL<1:0>:** PWM6 Timer Selection bits

11 = PWM6 based on TMR8

10 = PWM6 based on TMR6

01 = PWM6 based on TMR4

00 = PWM6 based on TMR2

bit 1-0 **C5TSEL<1:0>:** CCP5 Timer Selection bits

11 = CCP5 is based off Timer7 in Capture/Compare mode and Timer8 in PWM mode

10 = CCP5 is based off Timer5 in Capture/Compare mode and Timer6 in PWM mode

01 = CCP5 is based off Timer3 in Capture/Compare mode and Timer4 in PWM mode

00 = CCP5 is based off Timer1 in Capture/Compare mode and Timer2 in PWM mode

REGISTER 21-4: CCPxCAP: CAPTURE INPUT SELECTION MULTIPLEXER REGISTER

| | | | | | | | |
|-------|-----|-----|-----|-----|----------|---------|---------|
| U-0 | U-0 | U-0 | U-0 | U-0 | R/W-0/x | R/W-0/x | R/W-0/x |
| — | — | — | — | — | CTS<2:0> | | |
| bit 7 | | | | | bit 0 | | |

Legend:

R = Readable bit W = Writable bit U = Unimplemented bit, read as '0'
-n = Value at POR '1' = Bit is set '0' = Bit is cleared x = Bit is unknown

bit 7-3 **Unimplemented:** Read as '0'
bit 2-0 **CTS<2:0>:** Capture Trigger Input Selection bits

| CTS<2:0> | Connection | | | | |
|----------|----------------------------|----------------------------|----------------------------|----------------------------|----------------------------|
| | CCP1 | CCP2 | CCP3 | CCP4 | CCP5 |
| 111-101 | Reserved | | | | |
| 100 | IOC_Interrupt | | | | |
| 011 | CMP3_output | | | | |
| 010 | CMP2_output | | | | |
| 001 | CMP1_output | | | | |
| 000 | Pin selected by CCP1PPS | Pin selected by CCP2PPS | Pin selected by CCP3PPS | Pin selected by CCP4PPS | Pin selected by CCP5PPS |

REGISTER 21-5: CCPRxL: CCPx REGISTER LOW BYTE

| | | | | | | | |
|------------|---------|---------|---------|---------|---------|---------|---------|
| R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x |
| CCPRx<7:0> | | | | | | | |
| bit 7 | | | | | bit 0 | | |

Legend:

R = Readable bit W = Writable bit U = Unimplemented bit, read as '0'
-n = Value at POR '1' = Bit is set '0' = Bit is cleared x = Bit is unknown

bit 7-0 MODE = Capture Mode:
CCPRxL<7:0>: LSB of captured TMR1 value
MODE = Compare Mode:
CCPRxL<7:0>: LSB compared to TMR1 value
MODE = PWM Mode && FMT = 0:
CCPRxL<7:0>: CCPW<7:0> – Pulse-Width LS 8 bits
MODE = PWM Mode && FMT = 1:
CCPRxL<7:6>: CCPW<1:0> – Pulse-Width LS 2 bits
CCPRxL<5:0>: Not used

REGISTER 21-6: CCPRxH: CCPx REGISTER HIGH BYTE

| | | | | | | | |
|-------------|---------|---------|---------|---------|---------|---------|---------|
| R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x |
| CCPRx<15:8> | | | | | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|-------------------|------------------|--|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared x = Bit is unknown |

bit 7-0

MODE = Capture Mode:

CCPRxH<7:0>: MSB of captured TMR1 value

MODE = Compare Mode:

CCPRxH<7:0>: MSB compared to TMR1 value

MODE = PWM Mode && FMT = 0:

CCPRxH<7:2>: Not used

CCPRxH<1:0>: CCPW<9:8> – Pulse-Width MS 2 bits

MODE = PWM Mode && FMT = 1:

CCPRxH<7:0>: CCPW<9:2> – Pulse-Width MS 8 bits

21.3 Capture Mode

Capture mode makes use of the 16-bit Timer1 resource. When an event occurs on the capture source, the 16-bit CCPRxH:CCPRxL register pair captures and stores the 16-bit value of the TMRxH:TMRxL register pair, respectively. An event is defined as one of the following and is configured by the MODE<3:0> bits of the CCPxCON register:

- 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 PIR7 register is set. The interrupt flag must be cleared in software. If another capture occurs before the value in the CCPRxH:CCPRxL register pair is read, the old captured value is overwritten by the new captured value.

Note: 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 CCPRxH:CCPRxL register pair to either disable the module or read the register pair twice for data integrity.

Figure 21-1 shows a simplified diagram of the capture operation.

21.3.1 CAPTURE SOURCES

In Capture mode, the CCPx pin should be configured as an input by setting the associated TRIS control bit.

Note: If the CCPx pin is configured as an output, a write to the port can cause a capture condition.

The capture source is selected by configuring the CTS<2:0> bits of the CCPxCAP register. The following sources can be selected:

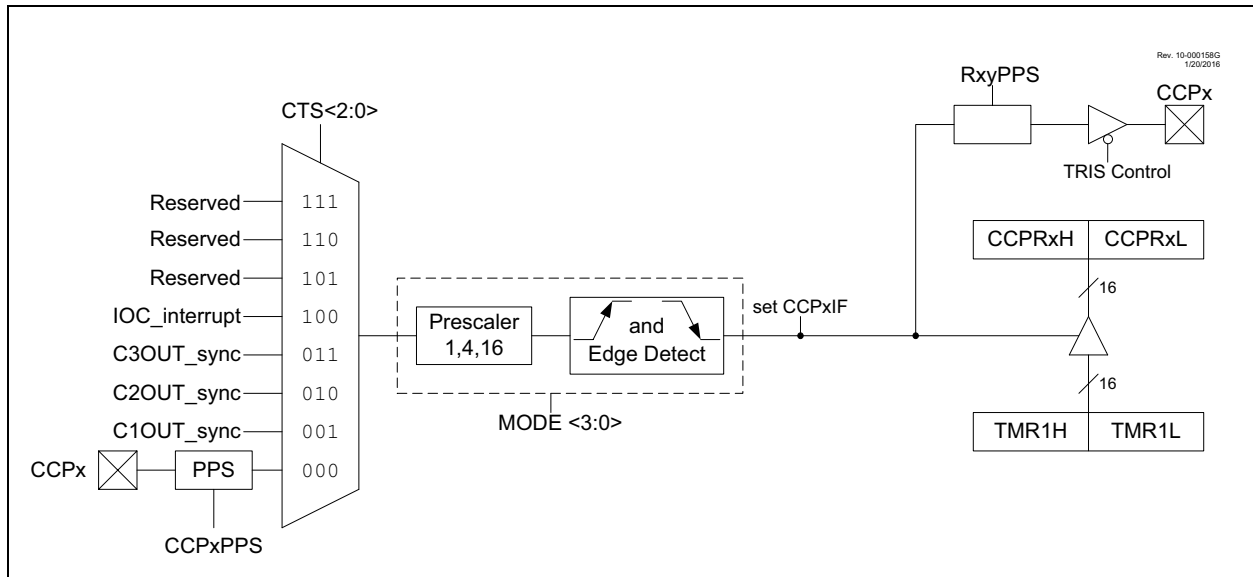
- Pin selected by CCPxPPS
- C1_output
- C2_output
- C3_output
- IOC_interrupt

21.3.2 TIMER1 MODE RESOURCE

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 [Section 19.0 “Timer1/3/5/7 Module with Gate Control”](#) for more information on configuring Timer1.

FIGURE 21-1: CAPTURE MODE OPERATION BLOCK DIAGRAM



21.3.3 SOFTWARE INTERRUPT MODE

When the Capture mode is changed, a false capture interrupt may be generated. The user should keep the CCPxIE Interrupt Priority bit of the PIE7 register clear to avoid false interrupts. Additionally, the user should clear the CCPxIF interrupt flag bit of the PIR7 register following any change in Operating mode.

Note: Clocking Timer1 from the system clock (FOSC) should not be used in Capture mode. In order for Capture mode to recognize the trigger event on the CCPx pin, Timer1 must be clocked from the instruction clock (FOSC/4) or from an external clock source.

21.3.4 CCP PRESCALER

There are four prescaler settings specified by the MODE<3:0> bits of the CCPxCON register. 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. [Example 21-1](#) demonstrates the code to perform this function.

EXAMPLE 21-1: CHANGING BETWEEN CAPTURE PRESCALERS

```
BANKSEL CCPxCON    ;Set Bank bits to point
                   ;to CCPxCON
CLRF    CCPxCON    ;Turn CCP module off
MOVLW  NEW_CAPT_PS ;Load the W reg with
                   ;the new prescaler
MOVWF  CCPxCON    ;move value and CCP ON
                   ;Load CCPxCON with this
                   ;value
```

21.3.5 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 (FOSC/4), or by an external clock source.

When Timer1 is clocked by FOSC/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.

21.4 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 Timer1 resource. The 16-bit value of the CCPRxH:CCPRxL register pair is constantly compared against the 16-bit value of the TMRxH:TMRxL register pair. When a match occurs, one of the following events can occur:

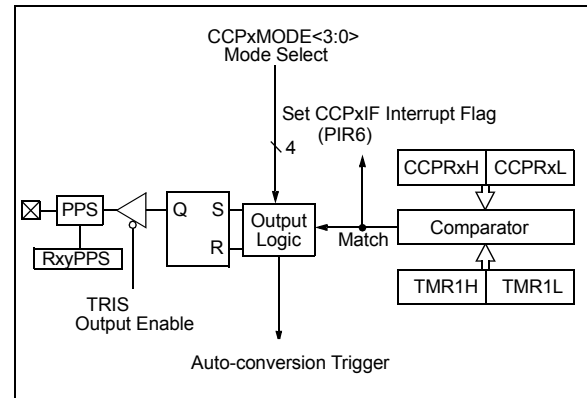
- Toggle the CCPx output, clear TMRx
- Toggle the CCPx output
- Set the CCPx output
- Clear the CCPx output
- Pulse output
- Pulse output, clear TMRx

The action on the pin is based on the value of the MODE<3:0> control bits of the CCPxCON register. At the same time, the interrupt flag CCPxIF bit is set, and an ADC conversion can be triggered, if selected.

All Compare modes can generate an interrupt and trigger an ADC conversion. When MODE = 4'b0001 or 4'b1011, the CCP resets the TMR register pair.

[Figure 21-2](#) shows a simplified diagram of the compare operation.

FIGURE 21-2: COMPARE MODE OPERATION BLOCK DIAGRAM



21.4.1 CCPx PIN CONFIGURATION

The software must configure the CCPx pin as an output by clearing the associated TRIS bit and defining the appropriate output pin through the RxyPPS registers. See [Section 17.0 “Peripheral Pin Select \(PPS\) Module”](#) for more details.

The CCP output can also be used as an input for other peripherals.

Note: 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.

21.4.2 TIMER1 MODE RESOURCE

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 [Section 19.0 “Timer1/3/5/7 Module with Gate Control”](#) for more information on configuring Timer1.

Note: Clocking Timer1 from the system clock (FOSC) should not be used in Compare mode. In order for Compare mode to recognize the trigger event on the CCPx pin, Timer1 must be clocked from the instruction clock (FOSC/4) or from an external clock source.

21.4.3 AUTO-CONVERSION TRIGGER

All CCPx modes set the CCP interrupt flag (CCPxF). When this flag is set and a match occurs, an auto-conversion trigger can take place if the CCP module is selected as the conversion trigger source.

Refer to [Section 32.2.6 “Auto-Conversion Trigger”](#) for more information.

Note: Removing the match condition by changing the contents of the CCPRxH and CCPRxL register pair, between the clock edge that generates the Auto-conversion Trigger and the clock edge that generates the Timer1 Reset, will preclude the Reset from occurring

21.4.4 COMPARE DURING SLEEP

Since FOSC 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).

21.5 PWM Overview

Pulse-Width Modulation (PWM) is a scheme that provides 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.

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 pulse-width time and in turn the power that is 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.

[Figure 21-3](#) shows a typical waveform of the PWM signal.

21.5.1 STANDARD PWM OPERATION

The standard PWM function described in this section is available and identical for all CCP modules.

The standard PWM mode 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:

- PR2 registers
- T2CON registers
- CCPRxL and CCPRxH registers
- CCPxCON registers

It is required to have FOSC/4 as the clock input to TMR2/4/6/8 for correct PWM operation. [Figure 21-4](#) shows a simplified block diagram of PWM operation.

Note: The corresponding TRIS bit must be cleared to enable the PWM output on the CCPx pin.

FIGURE 21-3: CCP PWM OUTPUT SIGNAL

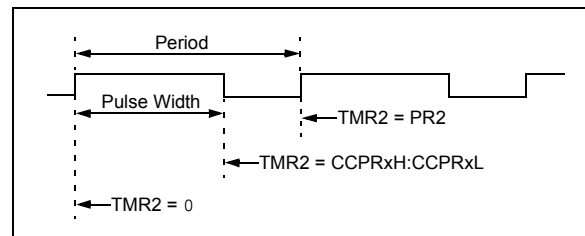
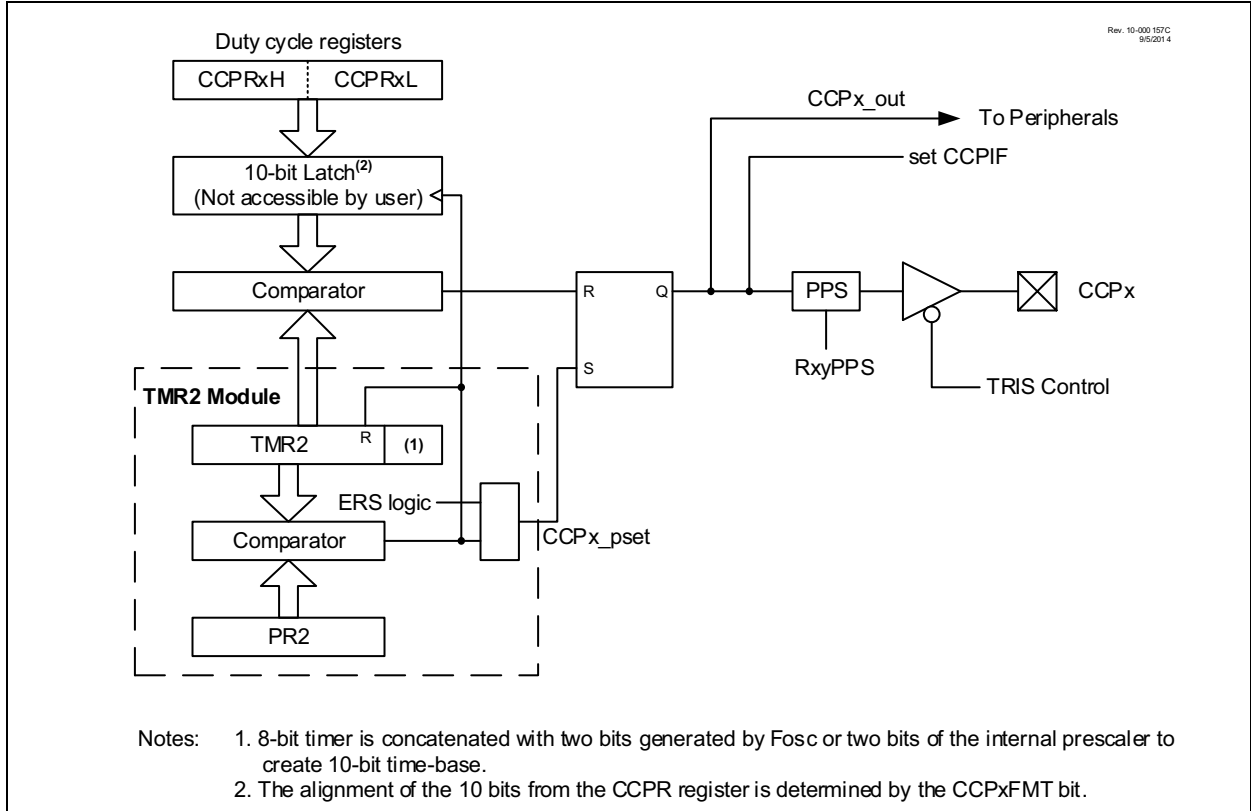


FIGURE 21-4: SIMPLIFIED PWM BLOCK DIAGRAM



21.5.2 SETUP FOR PWM OPERATION

The following steps should be taken when configuring the CCP module for standard PWM operation:

1. Use the desired output pin RxyPPS control to select CCPx as the source and disable the CCPx pin output driver by setting the associated TRIS bit.
2. Load the PR2 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 CCPRxL register, and the CCPRxH register with the PWM duty cycle value and configure the FMT bit of the CCPxCON register to set the proper register alignment.
5. Configure and start Timer2:
 - Clear the TMR2IF interrupt flag bit of the PIR5 register. See Note below.
 - Select the timer clock source to be as $F_{OSC}/4$ using the TxCLKCON register. This is required for correct operation of the PWM module.
 - Configure the T2CKPS bits of the T2CON register with the Timer prescale value.
 - Enable the Timer by setting the ON bit of the T2CON register.
6. Enable PWM output pin:
 - Wait until the Timer overflows and the TMR2IF bit of the PIR5 register is set. See Note below.
 - Enable the CCPx pin output driver by clearing the associated TRIS bit.

Note: In order 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.

21.5.3 TIMER2 TIMER RESOURCE

The PWM standard mode makes use of the 8-bit Timer2 timer resources to specify the PWM period.

21.5.4 PWM PERIOD

The PWM period is specified by the PR2 register of Timer2. The PWM period can be calculated using the formula of [Equation 21-1](#).

EQUATION 21-1: PWM PERIOD

$$PWM\ Period = [(PR2) + 1] \cdot 4 \cdot T_{OSC} \cdot (TMR2\ Prescale\ Value)$$

Note 1: $T_{OSC} = 1/F_{OSC}$

When TMR2 is equal to PR2, the following three events occur on the next increment cycle:

- TMR2 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 CCPRxL/H register pair into a 10-bit buffer.

Note: The Timer postscaler (see [Section 20.3 “Timer2 Interrupt”](#)) is not used in the determination of the PWM frequency.

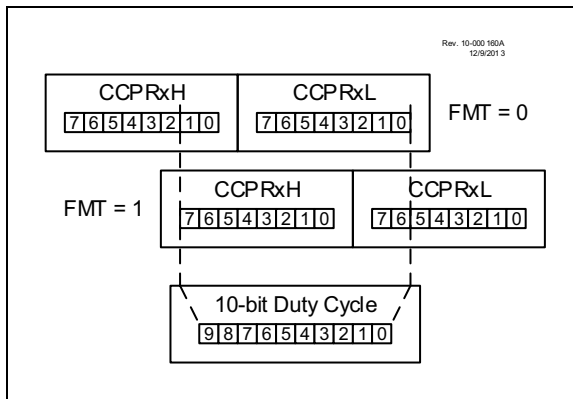
21.5.5 PWM DUTY CYCLE

The PWM duty cycle is specified by writing a 10-bit value to the CCPRxH:CCPRxL register pair. The alignment of the 10-bit value is determined by the FMT bit of the CCPxCON register (see Figure 21-5). The CCPRxH:CCPRxL register pair can be written to at any time; however the duty cycle value is not latched into the 10-bit buffer until after a match between PR2 and TMR2.

Equation 21-2 is used to calculate the PWM pulse width.

Equation 21-3 is used to calculate the PWM duty cycle ratio.

FIGURE 21-5: PWM 10-BIT ALIGNMENT



EQUATION 21-2: PULSE WIDTH

$$\text{Pulse Width} = (\text{CCPRxH:CCPRxL register pair}) \cdot T_{OSC} \cdot (\text{TMR2 Prescale Value})$$

EQUATION 21-3: DUTY CYCLE RATIO

$$\text{Duty Cycle Ratio} = \frac{(\text{CCPRxH:CCPRxL register pair})}{4(\text{PR2} + 1)}$$

CCPRxH:CCPRxL register pair are used to double buffer the PWM duty cycle. This double buffering is essential for glitchless PWM operation.

The 8-bit timer TMR2 register is concatenated with either the 2-bit internal system clock (FOSC), 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 CCPRxH:CCPRxL register pair, then the CCPx pin is cleared (see Figure 21-4).

21.5.6 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 ten bits when PR2 is 255. The resolution is a function of the PR2 register value as shown by Equation 21-4.

EQUATION 21-4: PWM RESOLUTION

$$\text{Resolution} = \frac{\log[4(\text{PR2} + 1)]}{\log(2)} \text{ bits}$$

Note: If the pulse-width value is greater than the period the assigned PWM pin(s) will remain unchanged.

TABLE 21-3: EXAMPLE PWM FREQUENCIES AND RESOLUTIONS (Fosc = 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 |
| PR2 Value | 0xFF | 0xFF | 0xFF | 0x3F | 0x1F | 0x17 |
| Maximum Resolution (bits) | 10 | 10 | 10 | 8 | 7 | 6.6 |

TABLE 21-4: EXAMPLE PWM FREQUENCIES AND RESOLUTIONS (Fosc = 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 |
| PR2 Value | 0x65 | 0x65 | 0x65 | 0x19 | 0x0C | 0x09 |
| Maximum Resolution (bits) | 8 | 8 | 8 | 6 | 5 | 5 |

21.5.7 OPERATION IN SLEEP MODE

In Sleep mode, the TMR2 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, TMR2 will continue from its previous state.

21.5.8 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 [Section 4.0 “Oscillator Module \(with Fail-Safe Clock Monitor\)”](#) for additional details.

21.5.9 EFFECTS OF RESET

Any Reset will force all ports to Input mode and the CCP registers to their Reset states.

TABLE 21-5: SUMMARY OF REGISTERS ASSOCIATED WITH CCPx

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on Page |
|----------|-------------|-----------|--------------|-----------|-------------|---------|-------------|---------|------------------|
| INTCON | GIE/GIEH | PEIE/GIEL | IPEN | — | INT3EDG | INT2EDG | INT1EDG | INT0EDG | 172 |
| PIE7 | — | — | — | CCP5IE | CCP4IE | CCP3IE | CCP2IE | CCP1IE | 191 |
| PIR7 | — | — | — | CCP5IF | CCP4IF | CCP3IF | CCP2IF | CCP1IF | 180 |
| IPR7 | — | — | — | CCP5IP | CCP4IP | CCP3IP | CCP2IP | CCP1IP | 201 |
| PMD4 | — | PWM7MD | PWM6MD | CCP5MD | CCP4MD | CCP3MD | CCP2MD | CCP1MD | 69 |
| CCPxCON | EN | — | OUT | FMT | MODE<3:0> | | | | 275 |
| CCPxCAP | — | — | — | — | — | — | CTS<1:0> | | 279 |
| CCPRxL | CCPRx<7:0> | | | | | | | | 279 |
| CCPRxH | CCPRx<15:8> | | | | | | | | 280 |
| CCPTMRS0 | C4TSEL<1:0> | | C3TSEL<1:0> | | C2TSEL<1:0> | | C1TSEL<1:0> | | 277 |
| CCPTMRS1 | — | — | P7TSEL<1:0> | | P6TSEL<1:0> | | C5TSEL<1:0> | | 278 |
| CCPxPPS | — | — | CCPxPPS<5:0> | | | | | | 224 |
| RxyPPS | — | — | RxyPPS<5:0> | | | | | | 227 |
| T1CON | — | — | T1CKPS<1:0> | | — | T1SYNC | T1RD16 | TMR1ON | 239 |
| T1GCON | TMR1GE | T1GPOL | T1GTM | T1GSPM | T1GO/DONE | T1GVAL | — | — | 240 |
| T1CLK | — | — | — | — | CS<3:0> | | | | 241 |
| T1GATE | — | — | — | — | GSS<3:0> | | | | 242 |
| TMR1L | TMR1L7 | TMR1L6 | TMR1L5 | TMR1L4 | TMR1L3 | TMR1L2 | TMR1L1 | TMR1L0 | 243 |
| TMR1H | TMR1H7 | TMR1H6 | TMR1H5 | TMR1H4 | TMR1H3 | TMR1H2 | TMR1H1 | TMR1H0 | 243 |
| TMR2 | TMR2<7:0> | | | | | | | | 255 |
| T2PR | PR2<7:0> | | | | | | | | 255 |
| T2CON | ON | CKPS<2:0> | | | OUTPS<3:0> | | | | 256 |
| T2HLT | PSYNC | CPOL | CSYNC | MODE<4:0> | | | | | 257 |
| T2CLKCON | — | — | — | — | CS<3:0> | | | | 259 |
| T2RST | — | — | — | — | RSEL<3:0> | | | | 260 |

Legend: — = Unimplemented location, read as '0'. Shaded cells are not used by the CCP module.

22.0 PULSE-WIDTH MODULATION (PWM 6/7)

The PIC18(L)F6xK40 family has two instances of the PWM module (PWM6/7). The PWM module generates a Pulse-Width Modulated signal determined by the duty cycle, period, and resolution that are configured by the following registers:

- PRx
- TxCON
- PWMxDCH
- PWMxDCL
- PWMxCON

Note: The corresponding TRIS bit must be cleared to enable the PWM output on the PWMx pin.

Each PWM module can select the timer source that controls the module. Each module has an independent timer selection which can be accessed using the CCPTMRS1 register ([Register 22-2](#)). Please note that the PWM mode operation is described with respect to TMR2 in the following sections.

[Figure 22-1](#) shows a simplified block diagram of PWM operation.

[Figure 22-2](#) shows a typical waveform of the PWM signal.

FIGURE 22-1: SIMPLIFIED PWM BLOCK DIAGRAM

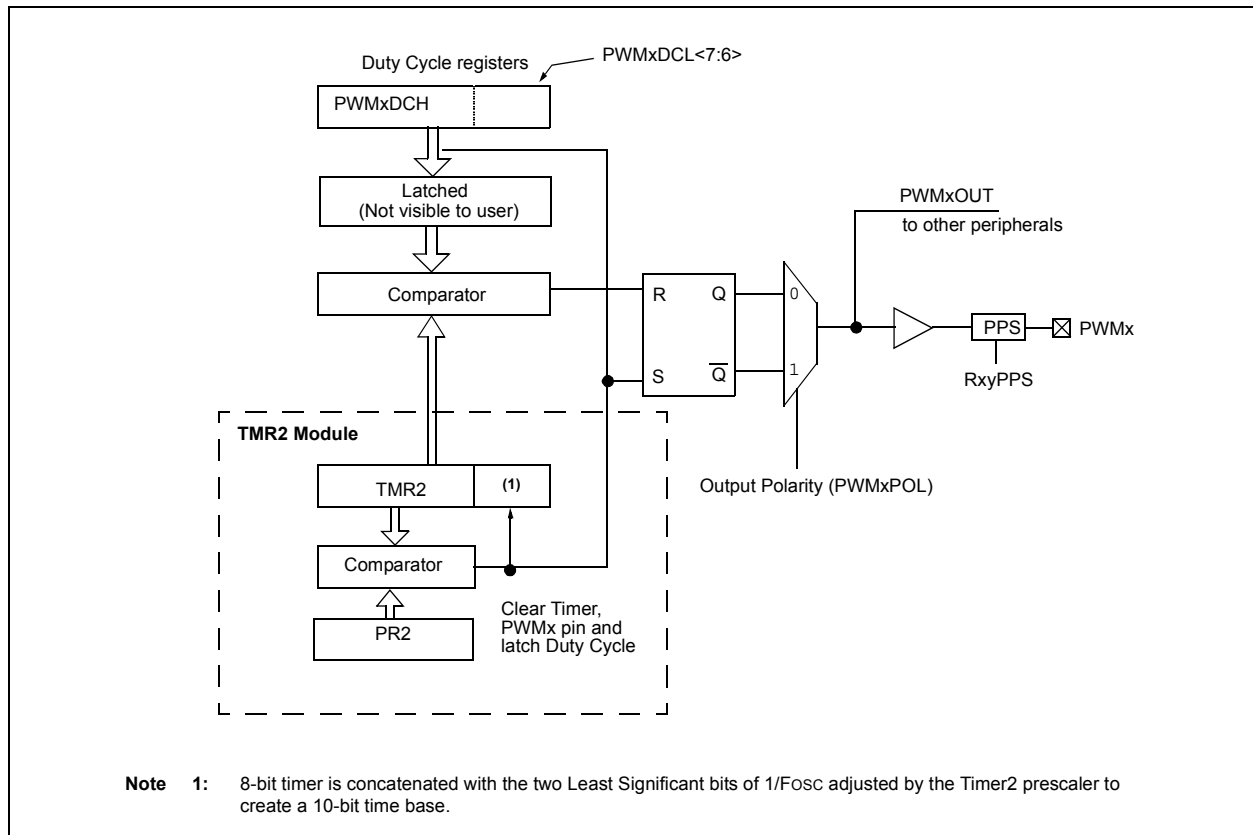
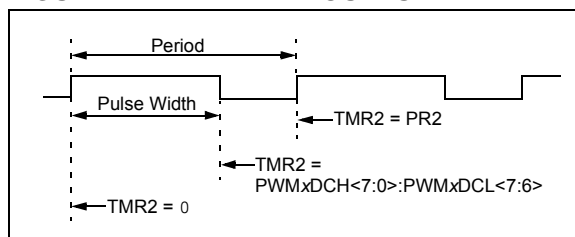


FIGURE 22-2: PWM OUTPUT



For a step-by-step procedure on how to set up this module for PWM operation, refer to [Section 22.1.9 "Setup for PWM Operation using PWMx Pins"](#).

22.1 PWMx Pin Configuration

All PWM outputs are multiplexed with the PORT data latch. The user must configure the pins as outputs by clearing the associated TRIS bits.

22.1.1 FUNDAMENTAL OPERATION

The PWM module produces a 10-bit resolution output. The PWM timer can be selected using the PxTSEL bits in the CCPTMRS1 register. The default selection for PWMx is TMR2. Please note that the PWM module operation in the following sections is described with respect to TMR2. Timer2 and PR2 set the period of the PWM. The PWMxDCL and PWMxDCH registers configure the duty cycle. The period is common to all PWM modules, whereas the duty cycle is independently controlled.

Note: The Timer2 postscaler is not used in the determination of the PWM frequency. The postscaler could be used to have a servo update rate at a different frequency than the PWM output.

All PWM outputs associated with Timer2 are set when TMR2 is cleared. Each PWMx is cleared when TMR2 is equal to the value specified in the corresponding PWMxDCH (8 MSb) and PWMxDCL<7:6> (2 LSb) registers. When the value is greater than or equal to PR2, the PWM output is never cleared (100% duty cycle).

Note: The PWMxDCH and PWMxDCL registers are double buffered. The buffers are updated when Timer2 matches PR2. Care should be taken to update both registers before the timer match occurs.

22.1.2 PWM OUTPUT POLARITY

The output polarity is inverted by setting the PWMxPOL bit of the PWMxCON register.

22.1.3 PWM PERIOD

The PWM period is specified by the PR2 register of Timer2. The PWM period can be calculated using the formula of Equation 22-1. It is required to have FOSC/4 as the clock input to TMR2/4/6/8 for correct PWM operation.

EQUATION 22-1: PWM PERIOD

$$PWM\ Period = [(PR2) + 1] \cdot 4 \cdot TOSC \cdot (TMR2\ Prescale\ Value)$$

Note: TOSC = 1/FOSC

When TMR2 is equal to PR2, the following three events occur on the next increment cycle:

- TMR2 is cleared
- The PWM output is active. (Exception: When the PWM duty cycle = 0%, the PWM output will remain inactive.)
- The PWMxDCH and PWMxDCL register values are latched into the buffers.

Note: The Timer2 postscaler has no effect on the PWM operation.

22.1.4 PWM DUTY CYCLE

The PWM duty cycle is specified by writing a 10-bit value to the PWMxDCH and PWMxDCL register pair. The PWMxDCH register contains the eight MSbs and the PWMxDCL<7:6>, the two LSbs. The PWMxDCH and PWMxDCL registers can be written to at any time.

Equation 22-2 is used to calculate the PWM pulse width.

Equation 22-3 is used to calculate the PWM duty cycle ratio.

EQUATION 22-2: PULSE WIDTH

$$Pulse\ Width = (PWMxDCH:PWMxDCL<7:6>) \cdot TOSC \cdot (TMR2\ Prescale\ Value)$$

Note: TOSC = 1/FOSC

EQUATION 22-3: DUTY CYCLE RATIO

$$Duty\ Cycle\ Ratio = \frac{(PWMxDCH:PWMxDCL<7:6>)}{4(PR2 + 1)}$$

The 8-bit timer TMR2 register is concatenated with the two Least Significant bits of 1/FOSC, adjusted by the Timer2 prescaler to create the 10-bit time base. The system clock is used if the Timer2 prescaler is set to 1:1.

22.1.5 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 ten bits when PR2 is 255. The resolution is a function of the PR2 register value as shown by [Equation 22-4](#).

EQUATION 22-4: PWM RESOLUTION

$$Resolution = \frac{\log[4(PR2 + 1)]}{\log(2)} \text{ bits}$$

Note: If the pulse-width value is greater than the period the assigned PWM pin(s) will remain unchanged.

TABLE 22-1: EXAMPLE PWM FREQUENCIES AND RESOLUTIONS (Fosc = 20 MHz)

| PWM Frequency | 0.31 kHz | 4.88 kHz | 19.53 kHz | 78.12 kHz | 156.3 kHz | 208.3 kHz |
|---------------------------|----------|----------|-----------|-----------|-----------|-----------|
| Timer Prescale | 64 | 4 | 1 | 1 | 1 | 1 |
| PR2 Value | 0xFF | 0xFF | 0xFF | 0x3F | 0x1F | 0x17 |
| Maximum Resolution (bits) | 10 | 10 | 10 | 8 | 7 | 6.6 |

TABLE 22-2: EXAMPLE PWM FREQUENCIES AND RESOLUTIONS (Fosc = 8 MHz)

| PWM Frequency | 0.31 kHz | 4.90 kHz | 19.61 kHz | 76.92 kHz | 153.85 kHz | 200.0 kHz |
|---------------------------|----------|----------|-----------|-----------|------------|-----------|
| Timer Prescale | 64 | 4 | 1 | 1 | 1 | 1 |
| PR2 Value | 0x65 | 0x65 | 0x65 | 0x19 | 0x0C | 0x09 |
| Maximum Resolution (bits) | 8 | 8 | 8 | 6 | 5 | 5 |

22.1.6 OPERATION IN SLEEP MODE

In Sleep mode, the TMR2 register will not increment and the state of the module will not change. If the PWMx pin is driving a value, it will continue to drive that value. When the device wakes up, TMR2 will continue from its previous state.

22.1.7 CHANGES IN SYSTEM CLOCK FREQUENCY

The PWM frequency is derived from the system clock frequency (Fosc). Any changes in the system clock frequency will result in changes to the PWM frequency. Refer to [Section 4.0 “Oscillator Module \(with Fail-Safe Clock Monitor\)”](#) for additional details.

22.1.8 EFFECTS OF RESET

Any Reset will force all ports to Input mode and the PWM registers to their Reset states.

22.1.9 SETUP FOR PWM OPERATION USING PWMx PINS

The following steps should be taken when configuring the module for PWM operation using the PWMx pins:

1. Disable the PWMx pin output driver(s) by setting the associated TRIS bit(s).
2. Clear the PWMxCON register.
3. Load the PR2 register with the PWM period value.
4. Load the PWMxDCH register and bits <7:6> of the PWMxDCL register with the PWM duty cycle value.
5. Configure and start Timer2:
 - Clear the TMR2IF interrupt flag bit of the PIR4 register. See Note 1 below.
 - Select the timer clock source to be as $F_{OSC}/4$ using the TxCLKCON register. This is required for correct operation of the PWM module.
 - Configure the T2CKPS bits of the T2CON register with the Timer2 prescale value.
 - Enable Timer2 by setting the T2ON bit of the T2CON register.
6. Enable PWM output pin and wait until Timer2 overflows, TMR2IF bit of the PIR4 register is set. See note below.
7. Enable the PWMx pin output driver(s) by clearing the associated TRIS bit(s) and setting the desired pin PPS control bits.
8. Configure the PWM module by loading the PWMxCON register with the appropriate values.

Note 1: In order to send a complete duty cycle and period on the first PWM output, the above steps must be followed in the order given. If it is not critical to start with a complete PWM signal, then move Step 8 to replace Step 4.

2: For operation with other peripherals only, disable PWMx pin outputs.

22.1.10 SETUP FOR PWM OPERATION TO OTHER DEVICE PERIPHERALS

The following steps should be taken when configuring the module for PWM operation to be used by other device peripherals:

1. Disable the PWMx pin output driver(s) by setting the associated TRIS bit(s).
2. Clear the PWMxCON register.
3. Load the PR2 register with the PWM period value.
4. Load the PWMxDCH register and bits <7:6> of the PWMxDCL register with the PWM duty cycle value.
5. Configure and start Timer2:
 - Clear the TMR2IF interrupt flag bit of the PIR4 register. See Note 1 below.
 - Select the timer clock source to be as $F_{OSC}/4$ using the TxCLKCON register. This is required for correct operation of the PWM module.
 - Configure the T2CKPS bits of the T2CON register with the Timer2 prescale value.
 - Enable Timer2 by setting the T2ON bit of the T2CON register.
6. Enable PWM output pin:
 - Wait until Timer2 overflows, TMR2IF bit of the PIR4 register is set. See Note 1 below.
7. Configure the PWM module by loading the PWMxCON register with the appropriate values.

Note 1: In order 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.

22.2 Register Definitions: PWM Control

Long bit name prefixes for the PWM peripherals are shown in [Table 22-3](#). Refer to [Section 1.4.2.2 “Long Bit Names”](#) for more information.

TABLE 22-3:

| Peripheral | Bit Name Prefix |
|------------|-----------------|
| PWM6 | PWM6 |
| PWM7 | PWM7 |

REGISTER 22-1: PWMxCON: PWM CONTROL REGISTER

| R/W-0/0 | U-0 | R-0/0 | R/W-0/0 | U-0 | U-0 | U-0 | U-0 |
|---------|-----|-------|---------|-----|-----|-----|-------|
| EN | — | OUT | POL | — | — | — | — |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

- bit 7 **EN:** PWM Module Enable bit
1 = PWM module is enabled
0 = PWM module is disabled
- bit 6 **Unimplemented:** Read as '0'
- bit 5 **OUT:** PWM Module Output Level When Bit is Read
- bit 4 **POL:** PWM Output Polarity Select bit
1 = PWM output is inverted
0 = PWM output is normal
- bit 3-0 **Unimplemented:** Read as '0'

REGISTER 22-2: CCPTMRS1: CCP TIMERS CONTROL REGISTER 1

| | | | | | | | |
|-------|-----|-------------|---------|-------------|---------|-------------|---------|
| U-0 | U-0 | R/W-0/0 | R/W-1/1 | R/W-0/0 | R/W-1/1 | R/W-0/0 | R/W-1/1 |
| — | — | P7TSEL<1:0> | | P6TSEL<1:0> | | C5TSEL<1:0> | |
| bit 7 | | | | | | bit 0 | |

Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

bit 7-6 **Unimplemented:** Read as '0'

bit 5-4 **P7TSEL<1:0>:** PWM7 Timer Selection bits

11 = PWM7 based on TMR8

10 = PWM7 based on TMR6

01 = PWM7 based on TMR4

00 = PWM7 based on TMR2

bit 3-2 **P6TSEL<1:0>:** PWM6 Timer Selection bits

11 = PWM6 based on TMR8

10 = PWM6 based on TMR6

01 = PWM6 based on TMR4

00 = PWM6 based on TMR2

bit 1-0 **C5TSEL<1:0>:** CCP5 Timer Selection bits

11 = CCP5 is based off Timer7 in Capture/Compare mode and Timer8 in PWM mode

10 = CCP5 is based off Timer5 in Capture/Compare mode and Timer6 in PWM mode

01 = CCP5 is based off Timer3 in Capture/Compare mode and Timer4 in PWM mode

00 = CCP5 is based off Timer1 in Capture/Compare mode and Timer2 in PWM mode

REGISTER 22-3: PWMxDCH: PWM DUTY CYCLE HIGH BITS

| | | | | | | | |
|----------|---------|---------|---------|---------|---------|---------|---------|
| R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u |
| DCH<7:0> | | | | | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-0 **DC<7:0>**: PWM Duty Cycle Most Significant bits
 These bits are the MSBs of the PWM duty cycle. The two LSBs are found in PWMxDCL Register.

REGISTER 22-4: PWMxDCL: PWM DUTY CYCLE LOW BITS

| | | | | | | | |
|----------|---------|-----|-----|-----|-----|-----|-------|
| R/W-x/u | R/W-x/u | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 |
| DCL<7:6> | | — | — | — | — | — | — |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-6 **DC<8:9>**: PWM Duty Cycle Least Significant bits
 These bits are the LSBs of the PWM duty cycle. The MSBs are found in PWMxDCH Register.

bit 5-0 **Unimplemented**: Read as '0'

TABLE 22-4: SUMMARY OF REGISTERS ASSOCIATED WITH PWM

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on Page | |
|----------|--------------|-------------|-------------|-------------|--------------|---------|-------------|---------|------------------|-----|
| CCPTMRS1 | — | — | P7TSEL<1:0> | | P6TSEL<1:0> | | C5TSEL<1:0> | | 294 | |
| PWM6CON | EN | — | OUT | POL | — | — | — | — | 293 | |
| PWM6DCH | DC<7:0> | | | | | | | | 295 | |
| PWM6DCL | PWM6DC<9:8> | | — | — | — | — | — | — | 295 | |
| PWM7CON | EN | — | PWM7OUT | PWM7POL | — | — | — | — | 293 | |
| PWM7DCH | PWM7DCH<7:0> | | | | | | | | 295 | |
| PWM7DCL | PWM7DCL<7:6> | | — | — | — | — | — | — | 295 | |
| INTCON | GIE/GIEH | PEIE/GIEL | IPEN | — | INT3EDG | INT2EDG | INT1EDG | INT0EDG | 172 | |
| PIE5 | TMR8IE | TMR7IE | TMR6IE | TMR5IE | TMR4IE | TMR3IE | TMR2IE | TMR1IE | 189 | |
| PIR5 | TMR8IF | TMR7IF | TMR6IF | TMR5IF | TMR4IF | TMR3IF | TMR2IF | TMR1IF | 178 | |
| IPR5 | TMR8IP | TMR7IP | TMR6IP | TMR5IP | TMR4IP | TMR3IP | TMR2IP | TMR1IP | 199 | |
| RxyPPS | — | — | RxyPPS<5:0> | | | | | | 227 | |
| TMR2 | TMR2<7:0> | | | | | | | | 255 | |
| PR2 | PR2<7:0> | | | | | | | | 255 | |
| T2CON | T2ON | T2CKPS<2:0> | | | T2OUTPS<3:0> | | | | 256 | |
| T2HLT | T2PSYNC | T2CPOL | T2CSYNC | T2MODE<4:0> | | | | | | 257 |
| T2CLKCON | — | — | — | — | T2CS<3:0> | | | | 259 | |
| T2RST | — | — | — | — | T2RSEL<3:0> | | | | 260 | |
| PMD4 | — | PWM7MD | PWM6MD | CCP5MD | CCP4MD | CCP3MD | CCP2MD | CCP1MD | 69 | |

Legend: — = Unimplemented locations, read as '0', u = unchanged, x = unknown. Shaded cells are not used by the PWM.

23.0 ZERO-CROSS DETECTION (ZCD) 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, V_{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 simplified block diagram [Figure 23-2](#).

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

23.1 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 nominally 300 μ A. Refer to [Equation 23-1](#) and [Figure 23-1](#). 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 23-1: EXTERNAL RESISTOR

$$R_{SERIES} = \frac{V_{PEAK}}{3 \times 10^{-4}}$$

FIGURE 23-1: EXTERNAL VOLTAGE

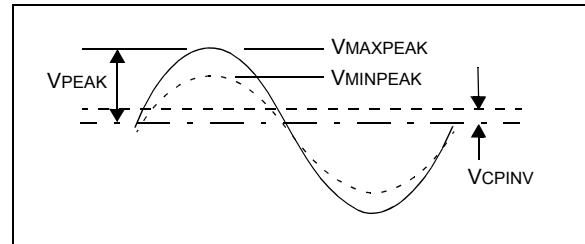
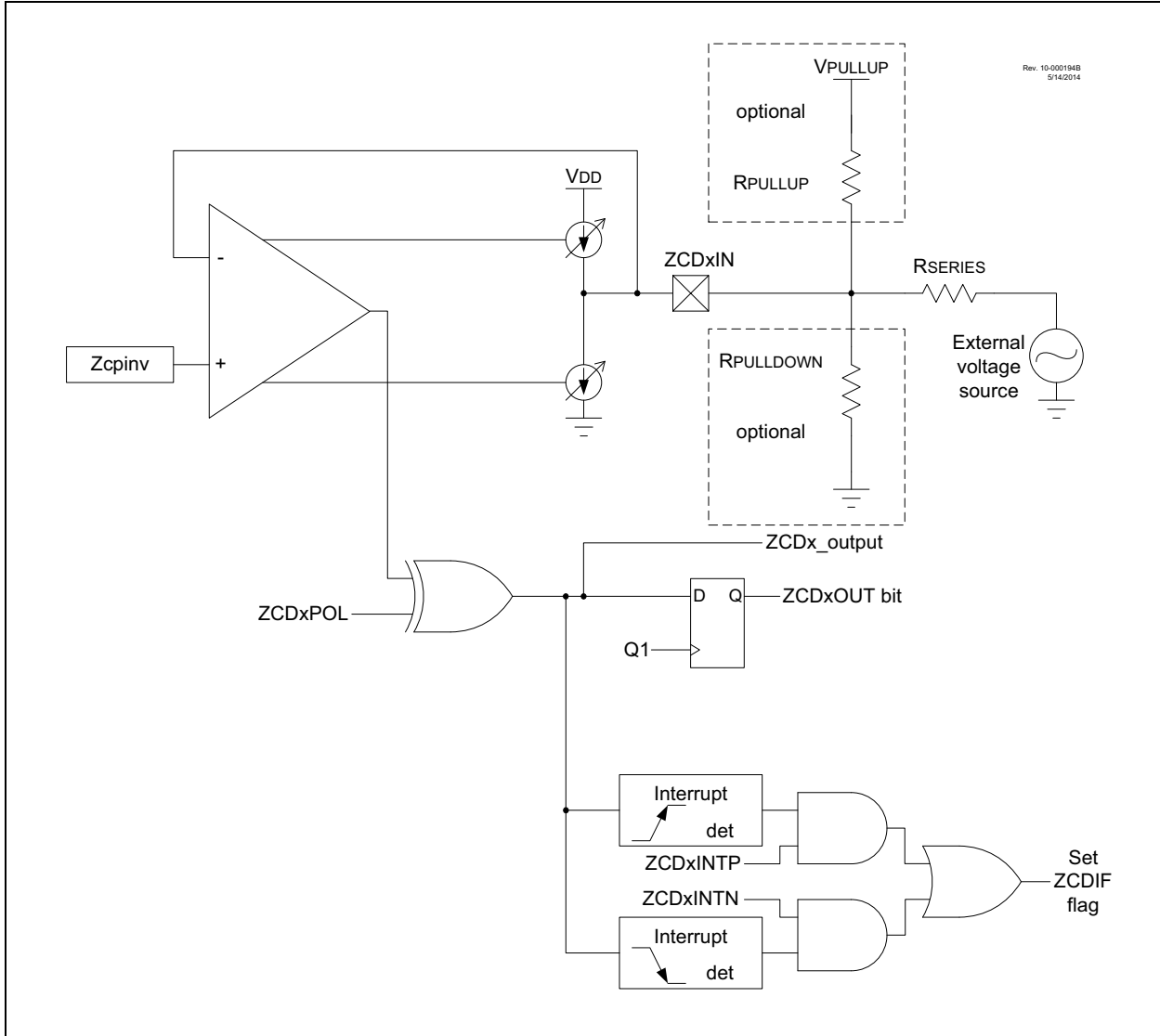


FIGURE 23-2: SIMPLIFIED ZCD BLOCK DIAGRAM



23.2 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 ZCDOUT bit of the ZCDCON register is set when the current sink is active, and cleared when the current source is active. The ZCDOUT bit is affected by the polarity bit.

The ZCDOUT signal can also be used as input to other modules. This is controlled by the registers of the corresponding module. ZCDOUT can be used as follows:

- Gate source for TMR1/3/5/7
- Clock source for TMR2/4/6/8
- Reset source for TMR2/4/6/8

23.3 ZCD Logic Polarity

The ZCDPOL bit of the ZCDCON register inverts the ZCDOUT bit relative to the current source and sink output. When the ZCDPOL bit is set, a ZCDOUT high indicates that the current source is active, and a low output indicates that the current sink is active.

The ZCDPOL bit affects the ZCD interrupts.

23.4 ZCD Interrupts

An interrupt will be generated upon a change in the ZCD logic output when the appropriate interrupt enables are set. A rising edge detector and a falling edge detector are present in the ZCD for this purpose.

The ZCDIF bit of the PIR2 register will be set when either edge detector is triggered and its associated enable bit is set. The ZCDINTP enables rising edge interrupts and the ZCDINTN bit enables falling edge interrupts. Both are located in the ZCDCON register. Priority of the interrupt can be changed if the IPEN bit of the INTCON register is set. The ZCD interrupt can be made high or low priority by setting or clearing the ZCDIP bit of the IPR2 register.

To fully enable the interrupt, the following bits must be set:

- ZCDIE bit of the PIE2 register
- ZCDINTP bit of the ZCDCON register (for a rising edge detection)
- ZCDINTN bit of the ZCDCON register (for a falling edge detection)
- PEIE and GIE bits of the INTCON register

Changing the ZCDPOL bit will cause an interrupt, regardless of the level of the ZCDSEN bit.

The ZCDIF bit of the PIR2 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.

23.5 Correcting for VCPINV offset

The actual voltage at which the ZCD switches is the reference voltage at the non-inverting 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. When the waveform is varying relative to V_{SS}, then 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}, then 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 in [Equation 23-2](#).

EQUATION 23-2: ZCD EVENT OFFSET

When External Voltage Source is relative to V_{SS}:

$$T_{OFFSET} = \frac{\text{asin}\left(\frac{V_{CPINV}}{V_{PEAK}}\right)}{2\pi \cdot Freq}$$

When External Voltage Source is relative to V_{DD}:

$$T_{OFFSET} = \frac{\text{asin}\left(\frac{V_{DD}-V_{CPINV}}{V_{PEAK}}\right)}{2\pi \cdot Freq}$$

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 V_{CPINV} switching voltage. The pull-up or pull-down value can be determined with the equations shown in [Equation 23-3](#) or [Equation 23-4](#).

EQUATION 23-3: ZCD PULL-UP/DOWN

When External Signal is relative to V_{SS}:

$$R_{PULLUP} = \frac{R_{SERIES}(V_{PULLUP} - V_{CPINV})}{V_{CPINV}}$$

When External Signal is relative to V_{DD}:

$$R_{PULLDOWN} = \frac{R_{SERIES}(V_{CPINV})}{(V_{DD} - V_{CPINV})}$$

The pull-up and pull-down resistor values are significantly affected by small variations of V_{CPINV} . Measuring V_{CPINV} can be difficult, especially when the waveform is relative to V_{DD} . However, by combining Equations 23-2 and 23-3, the resistor value can be determined from the time difference between the ZCD_output high and low periods. Note that the time difference, ΔT , is $4 \cdot T_{OFFSET}$. The equation for determining the pull-up and pull-down resistor values from the high and low ZCD_output periods is shown in Equation 23-4.

EQUATION 23-4: PULL-UP/DOWN RESISTOR VALUES

$$R = R_{SERIES} \left(\frac{V_{BIAS}}{V_{PEAK} \left(\sin \left(\pi Freq \frac{(\Delta T)}{2} \right) \right)} - 1 \right)$$

R is pull-up or pull-down resistor.

V_{BIAS} is V_{PULLUP} when R is pull-up or V_{DD} when R is pull-down.

ΔT is the ZCDOUT high and low period difference.

23.6 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 of $\pm 600 \mu A$ and above a reasonable minimum range. A general rule of thumb is that the maximum peak voltage can be no more than six times the minimum peak voltage. To ensure that the maximum current does not exceed $\pm 600 \mu A$ and the minimum is at least $\pm 100 \mu A$, compute the series resistance as shown in Equation 23-5. The compensating pull-up for this series resistance can be determined with Equation 23-3 because the pull-up value is independent from the peak voltage.

EQUATION 23-5: SERIES R FOR V RANGE

$$R_{SERIES} = \frac{V_{MAXPEAK} + V_{MINPEAK}}{7 \times 10^{-4}}$$

23.7 Operation During Sleep

The ZCD current sources and interrupts are unaffected by Sleep.

23.8 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 ZCDSEN bit of the ZCDCON register must be set to enable the ZCD module.

23.9 Disabling the ZCD Module

The ZCD module can be disabled in two ways:

1. Configuration Word 2H has the \overline{ZCD} bit which disables the ZCD module when set, but it can be enabled using the ZCDSEN bit of the ZCDCON register (Register 23-1). If the \overline{ZCD} bit is clear, the ZCD is always enabled.
2. The ZCD can also be disabled using the ZCDMD bit of the PMD3 register (Register 7-4). This is subject to the status of the \overline{ZCD} bit.

23.10 Register Definitions: ZCD Control

REGISTER 23-1: ZCDCON: ZERO-CROSS DETECT CONTROL REGISTER

| | | | | | | | |
|---------|-----|--------|---------|-----|-----|---------|---------|
| R/W-0/0 | U-0 | R-x | R/W-0/0 | U-0 | U-0 | R/W-0/0 | R/W-0/0 |
| ZCDSEN | — | ZCDOUT | ZCDPOL | — | — | ZCDINTP | ZCDINTN |
| bit 7 | | | | | | bit 0 | |

Legend:

| | | |
|-------------------|------------------|------------------------------------|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared |
| | | x = Bit is unknown |

- bit 7 **ZCDSEN:** Zero-Cross Detect Software Enable bit
This bit is ignored when ZCDSEN fuse is set.
1 = Zero-cross detect is enabled. ZCD pin is forced to output to source and sink current.
0 = Zero-cross detect is disabled. ZCD pin operates according to PPS and TRIS controls.
- bit 6 **Unimplemented:** Read as '0'
- bit 5 **ZCDOUT:** Zero-Cross Detect Data Output bit
ZCDPOL bit = 0:
1 = ZCD pin is sinking current
0 = ZCD pin is sourcing current
ZCDPOL bit = 1:
1 = ZCD pin is sourcing current
0 = ZCD pin is sinking current
- bit 4 **ZCDPOL:** Zero-Cross Detect Polarity bit
1 = ZCD logic output is inverted
0 = ZCD logic output is not inverted
- bit 3-2 **Unimplemented:** Read as '0'
- bit 1 **ZCDINTP:** Zero-Cross Detect Positive-Going Edge Interrupt Enable bit
1 = ZCDIF bit is set on low-to-high ZCD_output transition
0 = ZCDIF bit is unaffected by low-to-high ZCD_output transition
- bit 0 **ZCDINTN:** Zero-Cross Detect Negative-Going Edge Interrupt Enable bit
1 = ZCDIF bit is set on high-to-low ZCD_output transition
0 = ZCDIF bit is unaffected by high-to-low ZCD_output transition

TABLE 23-1: SUMMARY OF REGISTERS ASSOCIATED WITH THE ZCD MODULE

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on page |
|--------|--------|-------|--------|--------|--------|--------|---------|---------|---------------------|
| PIE2 | HLVDIE | ZCDIE | — | — | — | C3IE | C2IE | C1IE | 186 |
| PIR2 | HLVDIF | ZCDIF | — | — | — | C3IF | C2IF | C1IF | 175 |
| IPR2 | HLVDIP | ZCDIP | — | — | — | C3IP | C2IP | C1IP | 196 |
| ZCDCON | ZCDSEN | — | ZCDOUT | ZCDPOL | — | — | ZCDINTP | ZCDINTN | 301 |
| PMD3 | — | DACMD | ADCMD | — | CMP3MD | CMP2MD | CMP1MD | ZCDMD | 68 |

Legend: — = unimplemented, read as '0'. Shaded cells are unused by the ZCD module.

TABLE 23-2: SUMMARY OF CONFIGURATION WORD WITH THE ZCD MODULE

| Name | Bits | Bit 15/7 | Bit 14/6 | Bit 13/5 | Bit 12/4 | Bit 11/3 | Bit 10/2 | Bit 9/1 | Bit 8/0 | Register on Page |
|---------|------|--------------------|----------|----------------------|----------|----------|------------------|--------------------|---------|--------------------|
| CONFIG2 | 15:8 | \overline{XINST} | — | \overline{DEBUG} | STVREN | PPS1WAY | \overline{ZCD} | BORV1 | BORV0 | 22 |
| | 7:0 | BOREN1 | BOREN0 | $\overline{LPBOREN}$ | — | — | — | \overline{PWRTE} | MCLRE | |

Legend: — = unimplemented location, read as '0'. Shaded cells are not used by the ZCD module.

24.0 COMPLEMENTARY WAVEFORM GENERATOR (CWG) 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 PIC18(L)F6xK40 family has one instance of the CWG module.

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

24.1 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 [Section 24.6 “Dead-Band Control”](#).

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 [Section 24.10 “Auto-Shutdown”](#).

24.2 Operating Modes

The CWG module can operate in six different modes, as specified by the MODE<2:0> bits of the CWG1CON0 register:

- 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 data input, and provide up to four outputs as described in the following sections.

All modes include auto-shutdown control as described in [Section 24.10 “Auto-Shutdown”](#)

Note: Except as noted for Full-bridge mode ([Section 24.2.3 “Full-Bridge Modes”](#)), mode changes should only be performed while EN = 0 ([Register 24-1](#)).

24.2.1 HALF-BRIDGE MODE

In Half-Bridge mode, two output signals are generated as true and inverted versions of the input as illustrated in [Figure 24-2](#). A non-overlap (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 [Section 24.6 “Dead-Band Control”](#). The output steering feature cannot be used in this mode. A basic block diagram of this mode is shown in [Figure 24-1](#).

The unused outputs CWG1C and CWG1D drive similar signals, with polarity independently controlled by the POLC and POLD bits of the CWG1CON1 register, respectively.

FIGURE 24-1: SIMPLIFIED CWG BLOCK DIAGRAM (HALF-BRIDGE MODE, MODE<2:0> = 100)

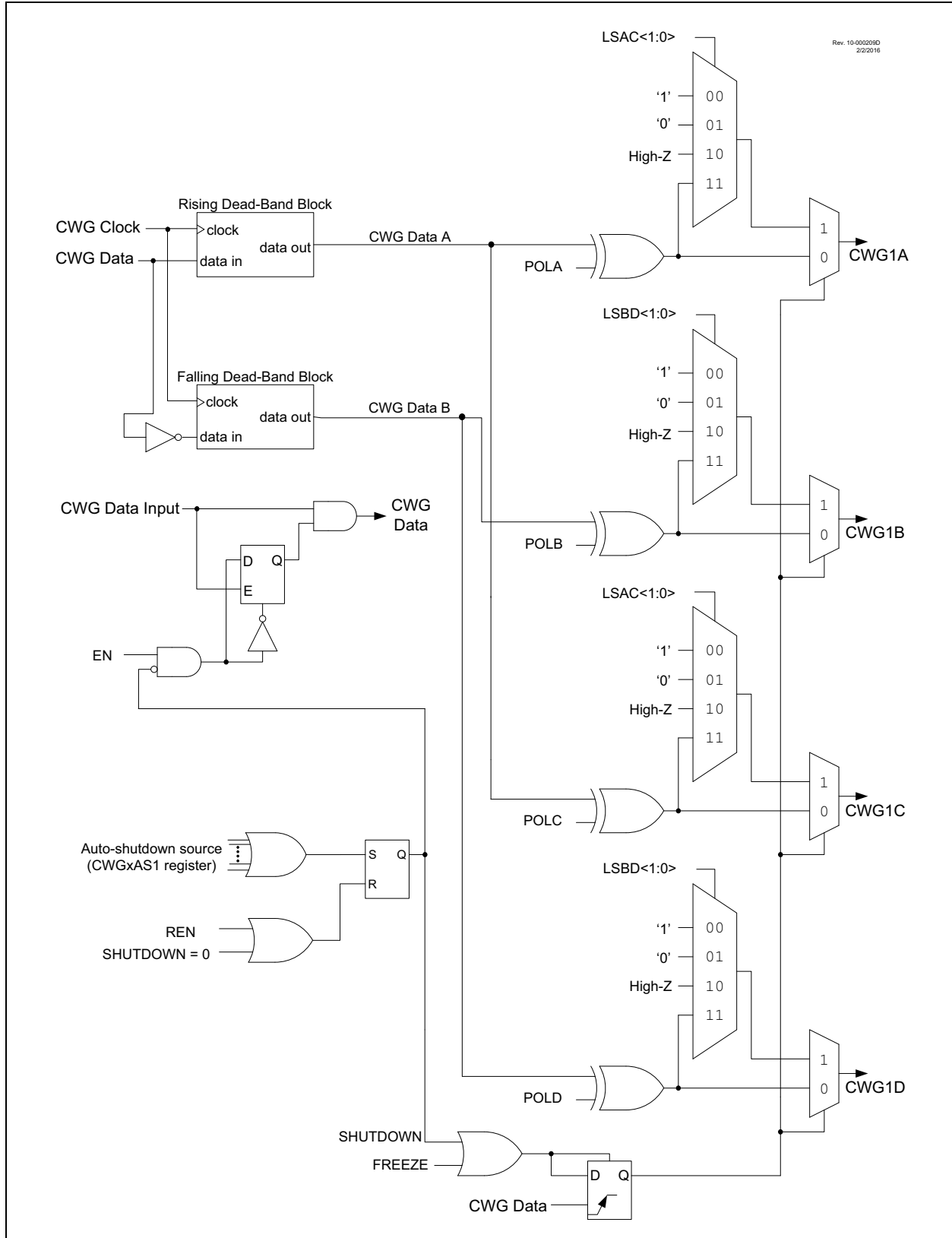
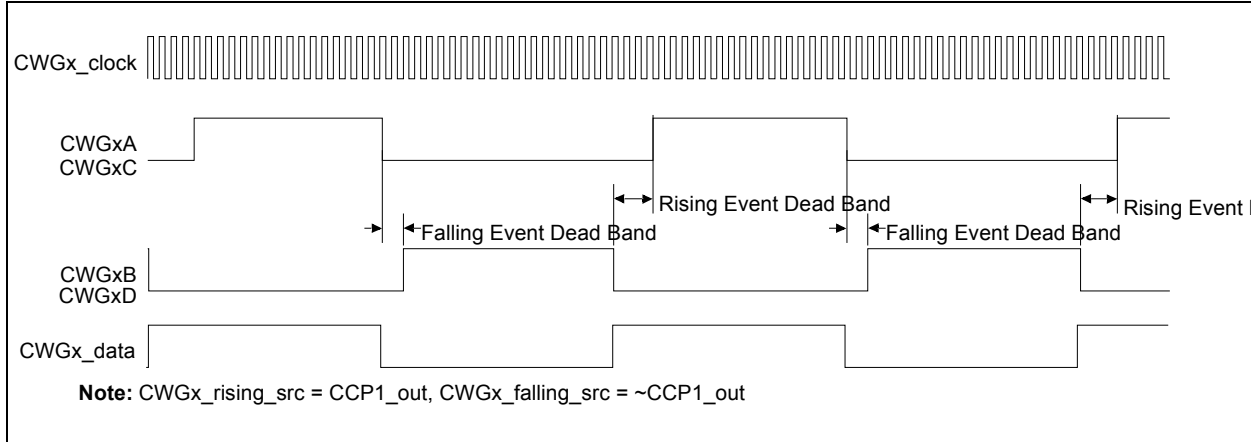


FIGURE 24-2: CWG1 HALF-BRIDGE MODE OPERATION



24.2.2 PUSH-PULL MODE

In Push-Pull mode, two output signals are generated, alternating copies of the input as illustrated in [Figure 24-4](#). 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 24-3](#).

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 CWG1A.

The unused outputs CWG1C and CWG1D drive copies of CWG1A and CWG1B, respectively, but with polarity controlled by the POLC and POLD bits of the CWG1CON1 register, respectively.

FIGURE 24-3: SIMPLIFIED CWG BLOCK DIAGRAM (PUSH-PULL MODE, MODE<2:0> = 101)

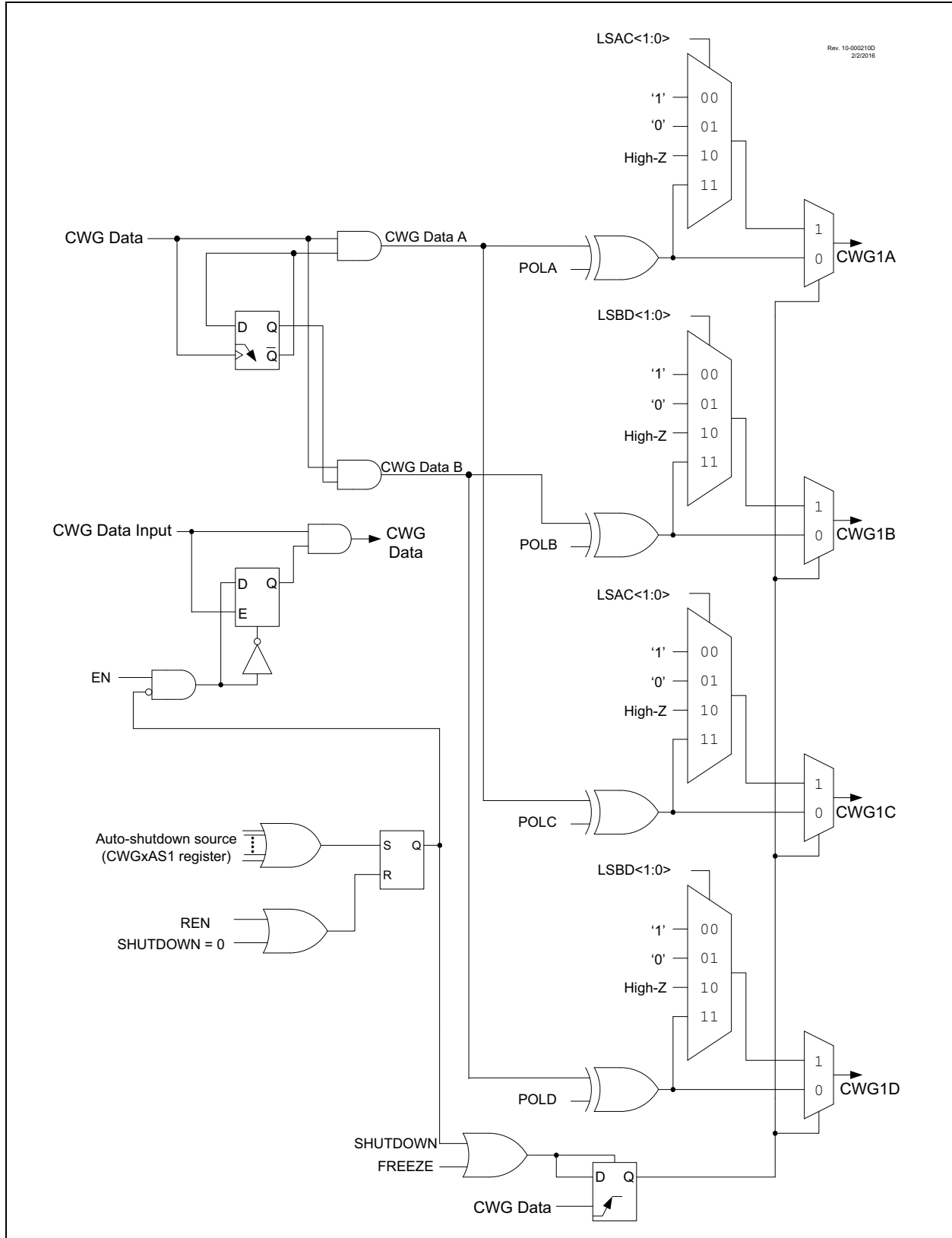
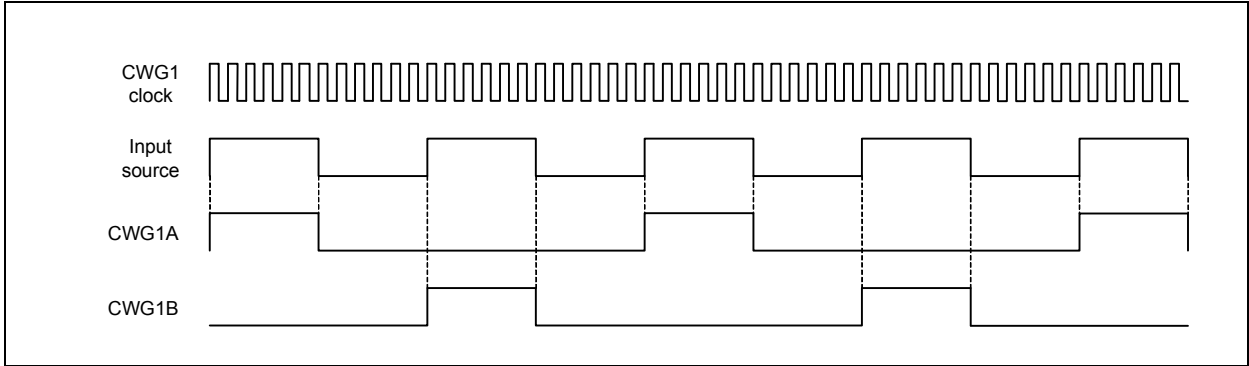


FIGURE 24-4: CWG1 PUSH-PULL MODE OPERATION



24.2.3 FULL-BRIDGE MODES

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 CWG1CON0 while keeping MODE<2:1> static, without disabling the CWG module. When connected as shown in [Figure 24-5](#), 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 24-6](#).

FIGURE 24-5: EXAMPLE OF FULL-BRIDGE APPLICATION

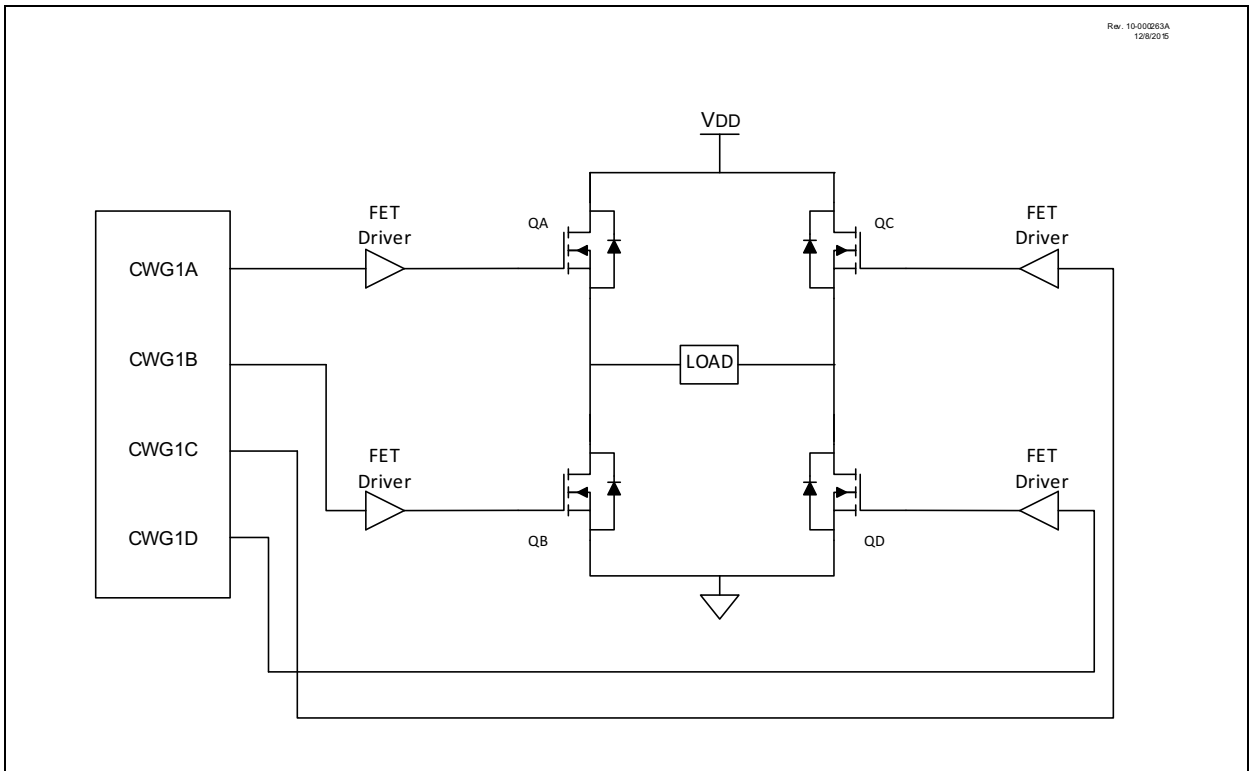
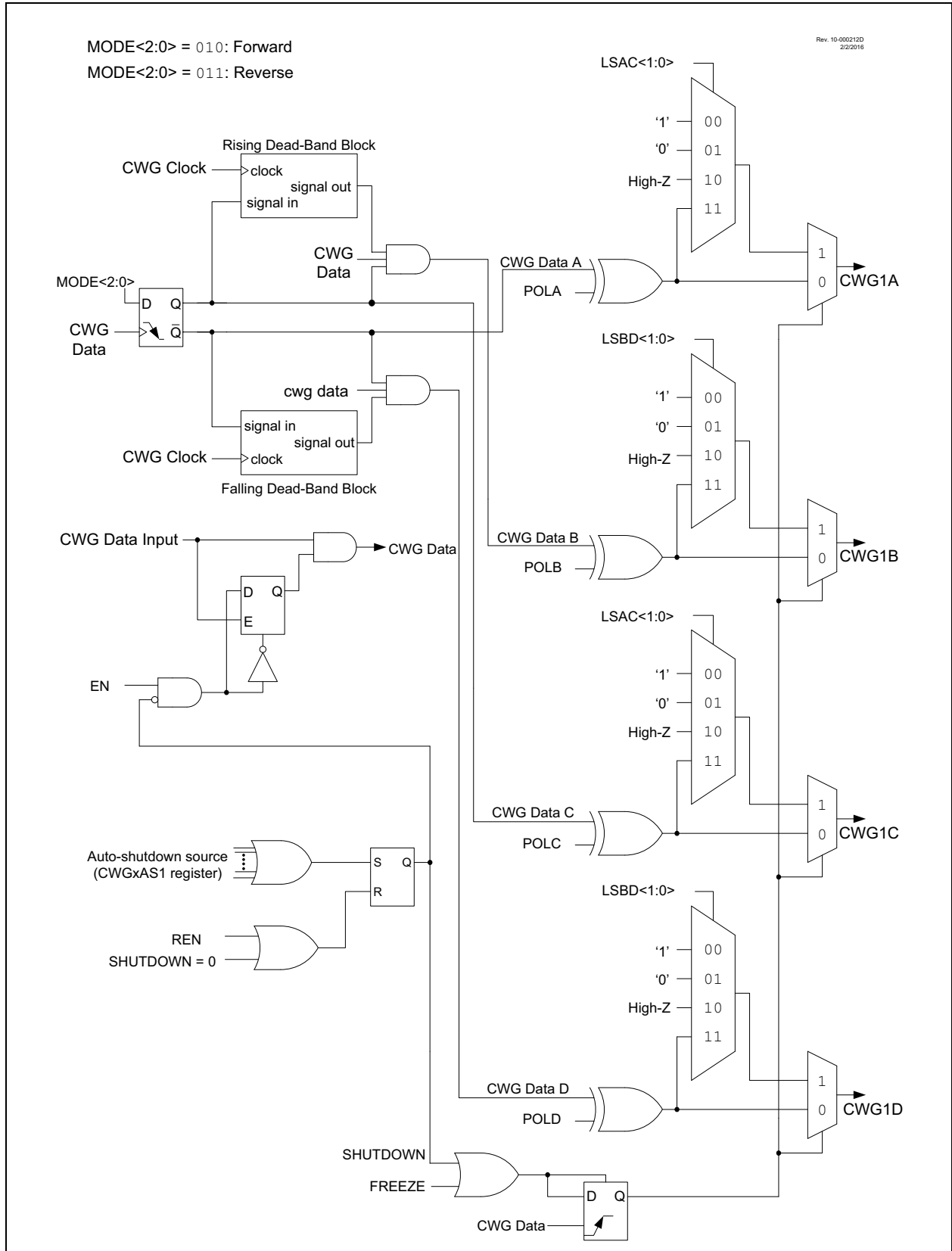


FIGURE 24-6: SIMPLIFIED CWG BLOCK DIAGRAM (FORWARD AND REVERSE FULL-BRIDGE MODES)

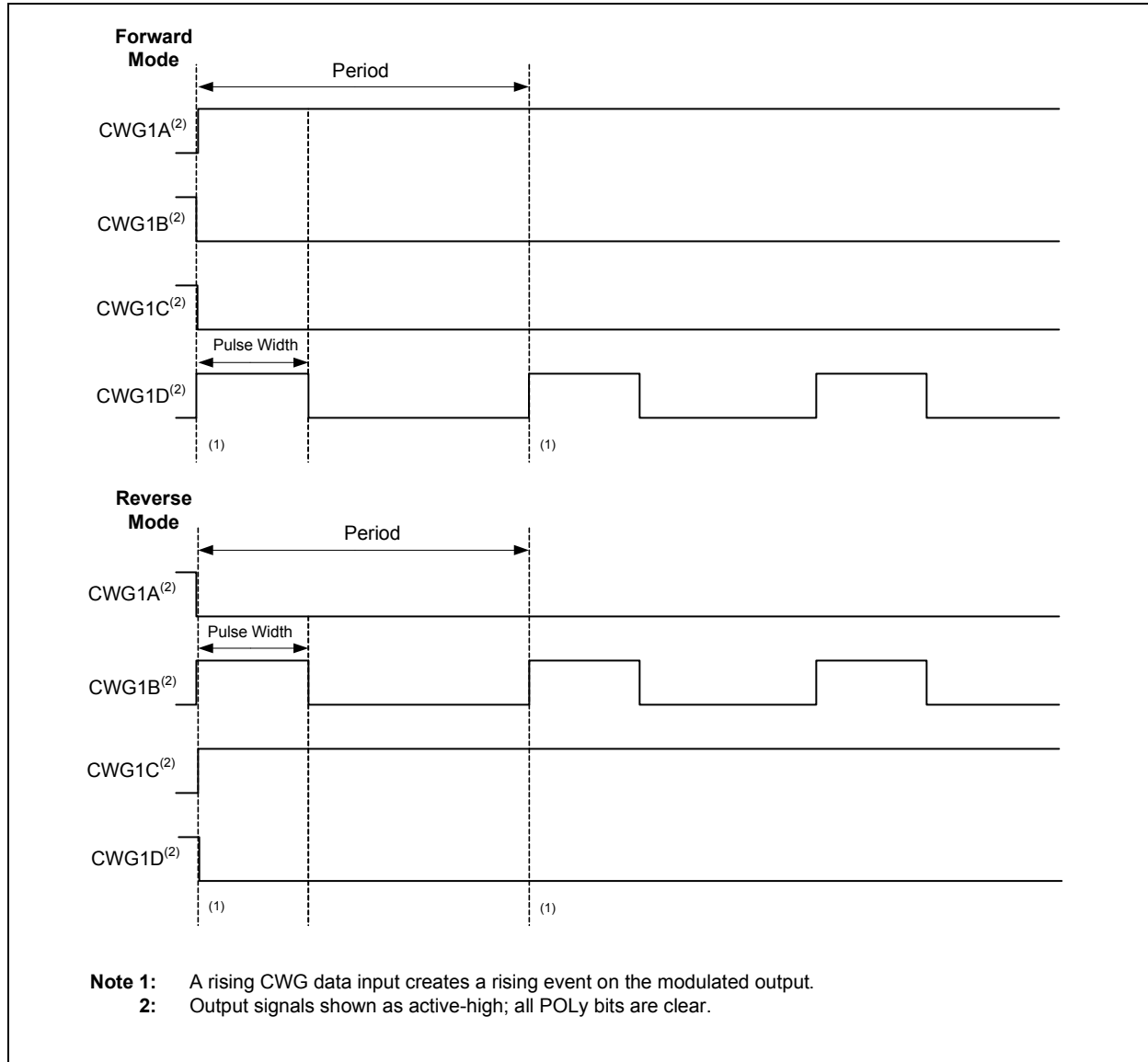


In Forward Full-Bridge mode (MODE<2:0> = 010), CWG1A is driven to its active state, CWG1B and CWG1C are driven to their inactive state, and CWG1D is modulated by the input signal, as shown in Figure 24-7.

In Reverse Full-Bridge mode (MODE<2:0> = 011), CWG1C is driven to its active state, CWG1A and CWG1D are driven to their inactive states, and CWG1B is modulated by the input signal, as shown in Figure 24-7.

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 Section 24.6 “Dead-Band Control”, with additional details in Section 24.7 “Rising Edge and Reverse Dead Band” and Section 24.8 “Falling Edge and Forward Dead Band”. Steering modes are not used with either of the Full-Bridge modes. The mode selection may be toggled between forward and reverse toggling the MODE<0> bit of the CWG1CON0 while keeping MODE<2:1> static, without disabling the CWG module.

FIGURE 24-7: EXAMPLE OF FULL-BRIDGE OUTPUT



24.2.3.1 Direction Change in Full-Bridge Mode

In Full-Bridge mode, changing MODE<2:0> controls the forward/reverse direction. Changes to MODE<2:0> change to the new direction on the next rising edge of the modulated input.

A direction change is initiated in software by changing the MODE<2:0> bits of the CWG1CON0 register. The sequence is illustrated in Figure 24-8.

- The associated active output CWG1A and the inactive output CWG1C are switched to drive in the opposite direction.
- The previously modulated output CWG1D is switched to the inactive state, and the previously inactive output CWG1B begins to modulate.
- CWG modulation resumes after the direction-switch dead band has elapsed.

24.2.3.2 Dead-Band Delay in Full-Bridge Mode

Dead-band delay is important when either of the following conditions is true:

1. The direction of the CWG output changes when the duty cycle of the data input is at or near 100%, or
2. The turn-off time of the power switch, including the power device and driver circuit, is greater than the turn-on time.

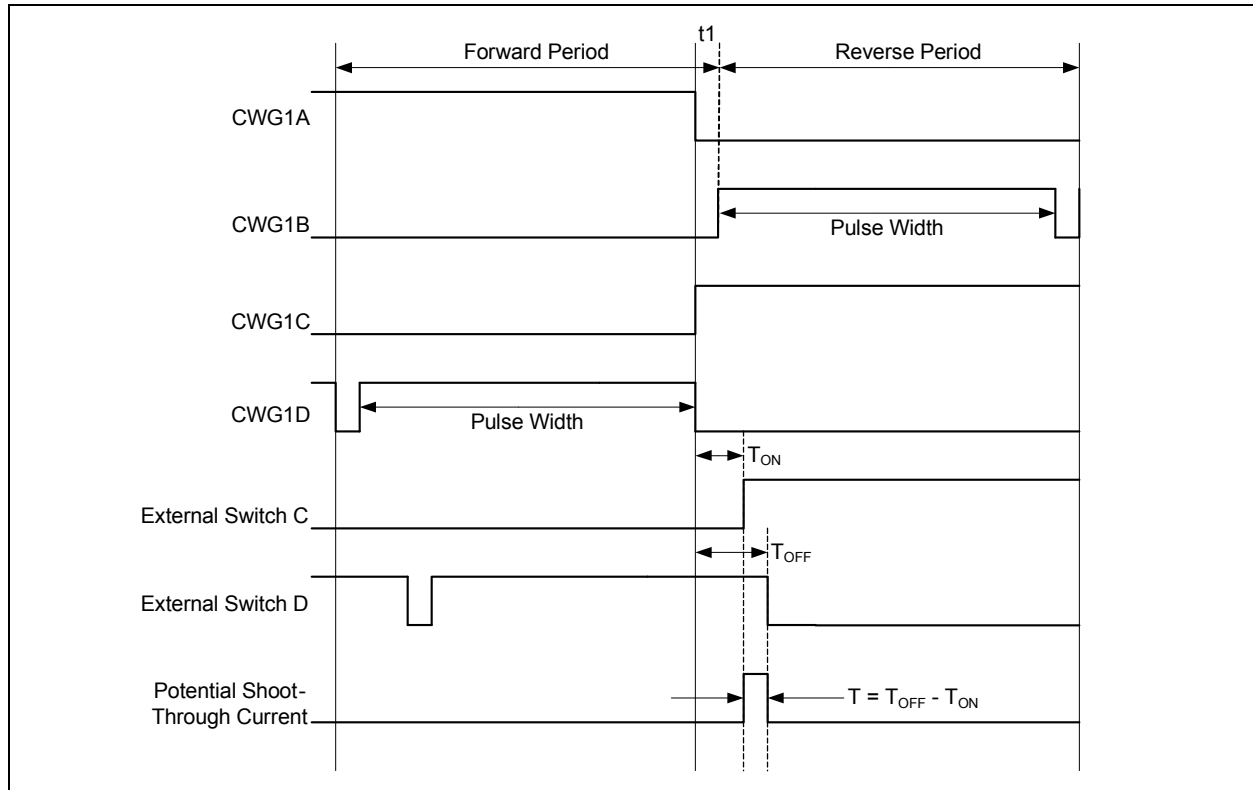
The dead-band delay is inserted only when changing directions, and only the modulated output is affected. The statically-configured outputs (CWG1A and CWG1C) are not afforded dead band, and switch essentially simultaneously.

Figure 24-8 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 CWG1A and CWG1D become inactive, while output CWG1C 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 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.

FIGURE 24-8: EXAMPLE OF PWM DIRECTION CHANGE AT NEAR 100% DUTY CYCLE



24.2.4 STEERING MODES

In both Synchronous and Asynchronous Steering modes, the modulated input signal can be steered to any combination of four CWG outputs and 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.

When $STRx = 0$ (Register 24-5), then the corresponding pin is held at the level defined by $DATx$ (Register 24-5). When $STRx = 1$, then the pin is driven by the modulated input signal.

The $POLx$ bits (Register 24-2) control the signal polarity only when $STRx = 1$.

The CWG auto-shutdown operation also applies to steering modes as described in Section 24.14 “Register Definitions: CWG Control”.

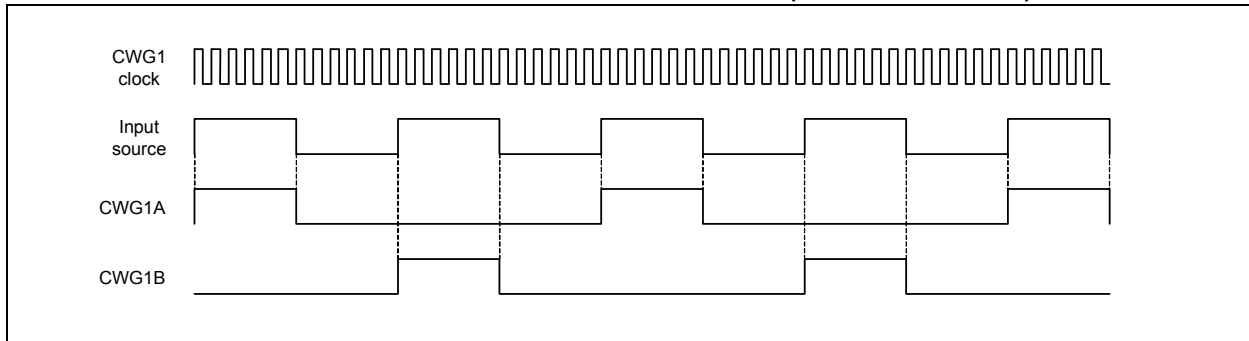
Note: Only the $STRx$ bits are synchronized; the $SDATx$ (data) bits are not synchronized.

The CWG auto-shutdown operation also applies in Steering modes as described in Section 24.10 “Auto-Shutdown”. An auto-shutdown event will only affect pins that have $STRx = 1$.

24.2.4.1 Synchronous Steering Mode

In Synchronous Steering mode ($MODE<2:0>$ bits = 001, Register 24-1), changes to steering selection registers take effect on the next rising edge of the modulated data input (Figure 24-9). In Synchronous Steering mode, the output will always produce a complete waveform.

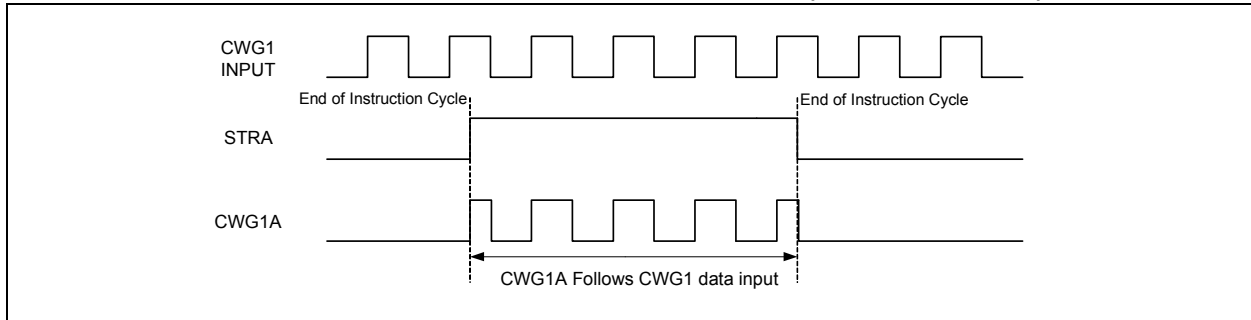
FIGURE 24-9: EXAMPLE OF SYNCHRONOUS STEERING ($MODE<2:0> = 001$)



24.2.4.2 Asynchronous Steering Mode

In Asynchronous mode (MODE<2:0> bits = 000, [Register 24-1](#)), steering takes effect at the end of the instruction cycle that writes to STR. In Asynchronous Steering mode, the output signal may be an incomplete waveform ([Figure 24-10](#)). This operation may be useful when the user firmware needs to immediately remove a signal from the output pin.

FIGURE 24-10: EXAMPLE OF ASYNCHRONOUS STEERING (MODE<2:0>= 000)

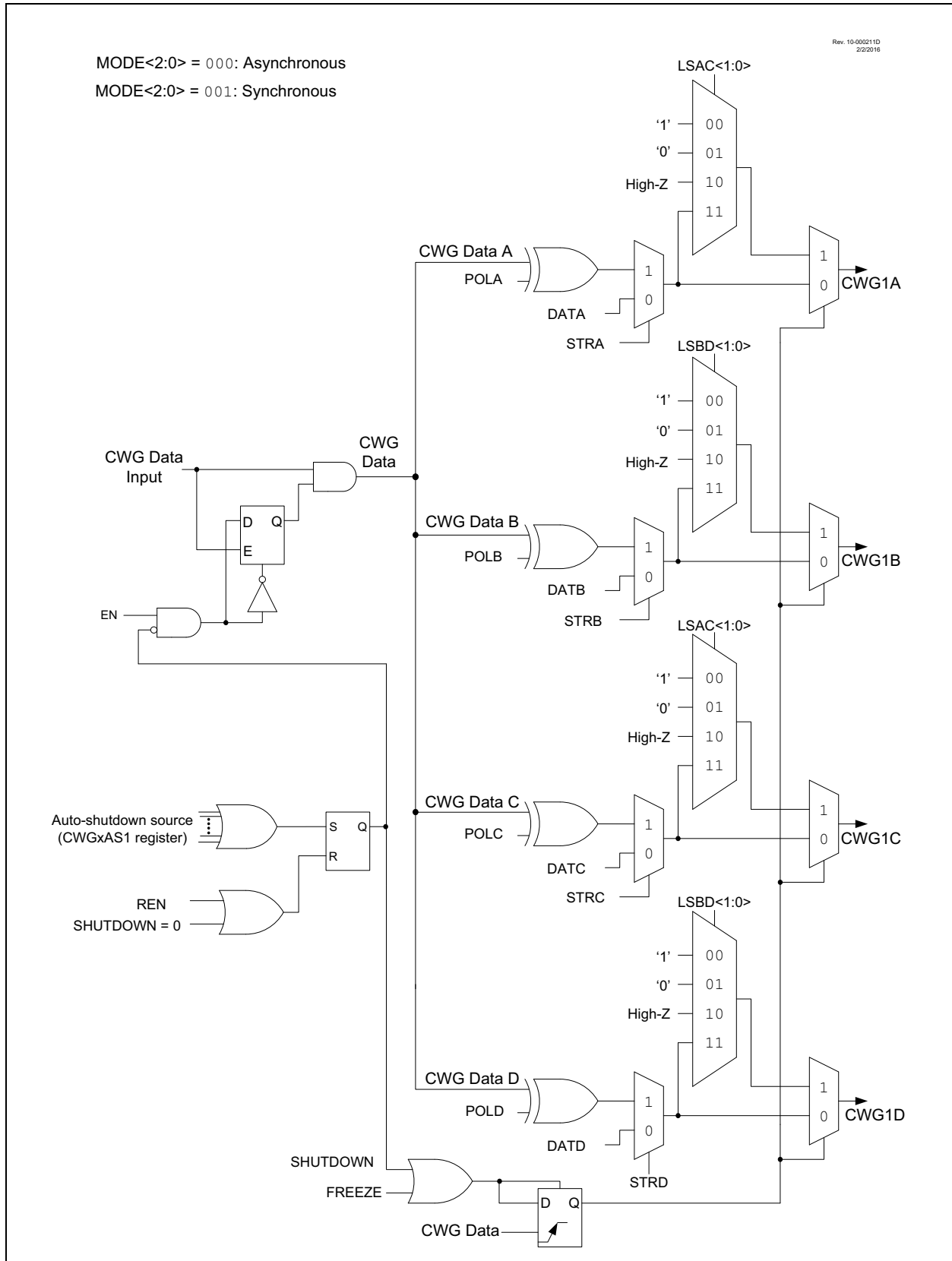


24.2.4.3 Start-up Considerations

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 POLy bits ([Register 24-2](#)) allow the user to choose whether the output signals are active-high or active-low.

FIGURE 24-11: SIMPLIFIED CWG BLOCK DIAGRAM (OUTPUT STEERING MODES)



24.3 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:

- FOSC (system clock)
- HFINTOSC

When the HFINTOSC is selected, the HFINTOSC will be kept running during Sleep. Therefore, 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 of the CWG1CLKCON register (Register 24-3). The system clock FOSC, is disabled in Sleep and thus dead-band control cannot be used.

24.4 Selectable Input Sources

The CWG generates the output waveforms from the input sources in Table 24-1.

TABLE 24-1: SELECTABLE INPUT SOURCES

| Source Peripheral | Signal Name | ISM<3:0> |
|-------------------|------------------------------|---------------|
| CWG1PPS | Pin selected by CWG1PPS | 0000 |
| CCP1 | CCP1 Output | 0001 |
| CCP2 | CCP2 Output | 0010 |
| CCP3 | CCP3 Output | 0011 |
| CCP4 | CCP4 Output | 0100 |
| CCP5 | CCP5 Output | 0101 |
| PWM6 | PWM6 Output | 0110 |
| PWM7 | PWM7 Output | 0111 |
| CMP1 | Comparator 1 Output | 1000 |
| CMP2 | Comparator 2 Output | 1001 |
| CMP3 | Comparator 3 Output | 1010 |
| DSM | Data signal modulator output | 1011 |
| Reserved | | 1100- 1111 |

The input sources are selected using the ISM<3:0> bits in the CWG1ISM register (Register 24-4).

24.5 Output Control

24.5.1 CWG OUTPUTS

Each CWG output can be routed to a Peripheral Pin Select (PPS) output via the RxyPPS register (see Section 17.0 “Peripheral Pin Select (PPS) Module”).

24.5.2 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 of the CWG1CON1. Auto-shutdown and steering options are unaffected by polarity.

24.6 Dead-Band Control

The dead-band control provides non-overlapping PWM signals to prevent shoot-through current in PWM switches. 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 dead-band Full-Bridge mode. The other is used for the falling edge of the input source control in Half-Bridge mode or for forward 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. See CWG1DBR and CWG1DBF registers, respectively.

24.6.1 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 24-2.

24.6.2 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 of the CWG1CON0 register can be set or cleared while the CWG is running, allowing for changes from Forward to Reverse mode. The CWG1A and CWG1C signals will change immediately upon the first rising input edge following a direction change, but the modulated signals (CWG1B or CWG1D, depending on the direction of the change) will experience a delay dictated by the dead-band counters.

24.7 Rising Edge and Reverse Dead Band

In Half-Bridge mode, the rising edge dead band delays the turn-on of the CWG1A 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 CWG1B is affected.

The CWG1DBR 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.

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 CWG1DBR register value is double-buffered. When EN = 0 ([Register 24-1](#)), the buffer is loaded when CWG1DBR is written. If EN = 1, then the buffer will be loaded at the rising edge following the first falling edge of the data input, after the LD bit ([Register 24-1](#)) is set. Refer to [Figure 24-12](#) for an example.

24.8 Falling Edge and Forward Dead Band

In Half-Bridge mode, the falling edge dead band delays the turn-on of the CWG1B 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 CWG1D is affected.

The CWG1DBF register determines the duration of the dead-band interval on the falling edge of the input source signal. This duration is from zero to 64 periods of 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.

The CWG1DBF register value is double-buffered. When EN = 0 ([Register 24-1](#)), the buffer is loaded when CWG1DBF is written. If EN = 1, then the buffer will be loaded at the rising edge following the first falling edge of the data input after the LD ([Register 24-1](#)) is set. Refer to [Figure 24-13](#) for an example.

FIGURE 24-12: DEAD-BAND OPERATION, CWG1DBR = 0x01, CWG1DBF = 0x02

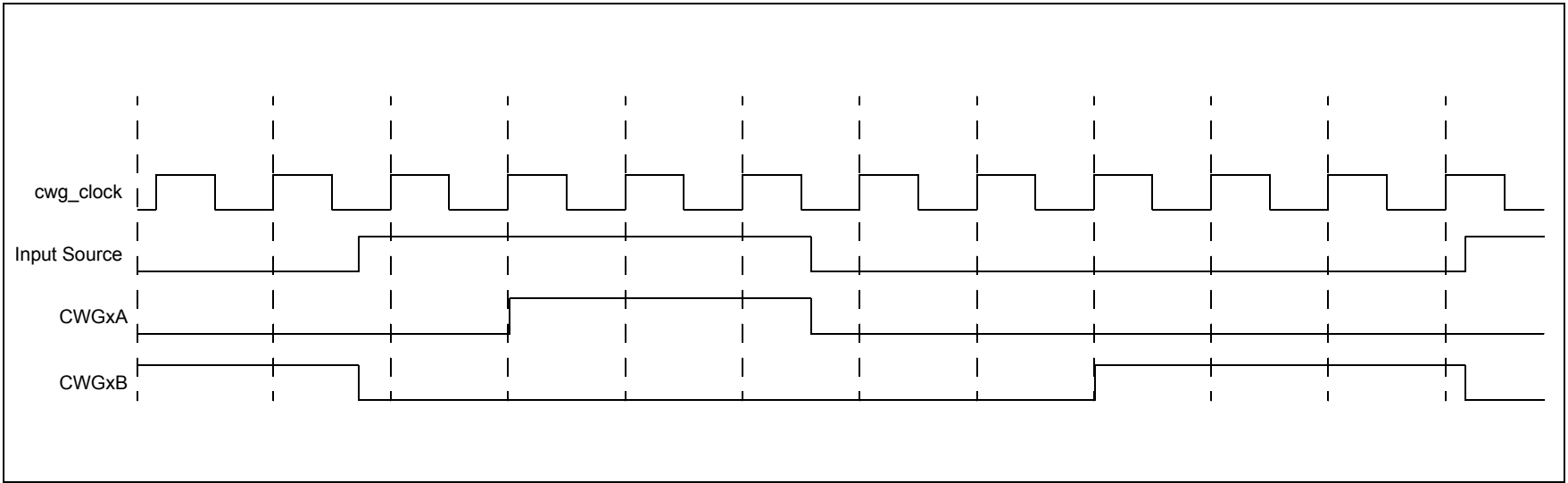
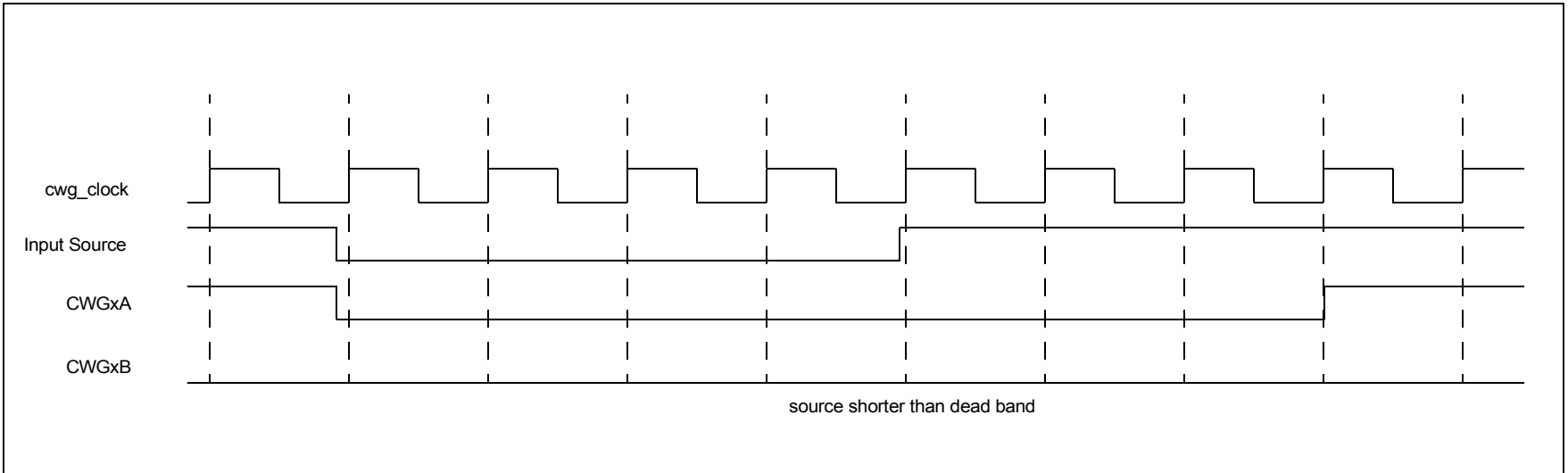


FIGURE 24-13: DEAD-BAND OPERATION, CWG1DBR = 0x03, CWG1DBF = 0x06, SOURCE SHORTER THAN DEAD BAND



24.9 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 [Equation 24-1](#) for more details.

EQUATION 24-1: DEAD-BAND DELAY TIME CALCULATION

$$T_{DEAD-BAND_MIN} = \frac{1}{F_{CWG_CLOCK}} \cdot DBx < 4:0 >$$

$$T_{DEAD-BAND_MAX} = \frac{1}{F_{CWG_CLOCK}} \cdot DBx < 4:0 > + 1$$

$$T_{JITTER} = T_{DEAD-BAND_MAX} - T_{DEAD-BAND_MIN}$$

$$T_{JITTER} = \frac{1}{F_{CWG_CLOCK}}$$

$$T_{DEAD-BAND_MAX} = T_{DEAD-BAND_MIN} + T_{JITTER}$$

EXAMPLE

$$DBR < 4:0 > = 0x0A = 10$$

$$F_{CWG_CLOCK} = 8 \text{ MHz}$$

$$T_{JITTER} = \frac{1}{8 \text{ MHz}} = 125 \text{ ns}$$

$$T_{DEAD-BAND_MIN} = 125 \text{ ns} \cdot 10 = 125 \text{ } \mu\text{s}$$

$$T_{DEAD-BAND_MAX} = 1.25 \text{ } \mu\text{s} + 0.125 \text{ } \mu\text{s} = 1.37 \text{ } \mu\text{s}$$

24.10 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 [Figure 24-14](#).

24.10.1 SHUTDOWN

The shutdown state can be entered by either of the following two methods:

- Software generated
- External Input

24.10.1.1 Software Generated Shutdown

Setting the SHUTDOWN bit of the CWG1AS0 register 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.

24.10.1.2 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 LSB<1:0> and LSAC<1:0> bits of the CWG1AS0 register ([Register 24-6](#)). Several input sources can be selected to cause a shutdown condition. All input sources are active-low. The sources are:

- Pin selected by CWG1PPS
- Timer2 post-scaled output
- Timer4 post-scaled output
- Timer6 post-scaled output
- Comparator 1 output
- Comparator 2 output

Shutdown input sources are individually enabled by the ASxE bits of the CWG1AS1 register ([Register 24-7](#)).

Note: 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.

24.10.1.3 Pin Override Levels

The levels driven to the CWG outputs during an auto-shutdown event are controlled by the LSB<1:0> and LSAC<1:0> bits of the CWG1AS0 register ([Register 24-6](#)). The LSB<1:0> bits control CWG1B/D output levels, while the LSAC<1:0> bits control the CWG1A/C output levels.

24.10.1.4 Auto-Shutdown Interrupts

When an auto-shutdown event occurs, either by software or hardware setting SHUTDOWN, the CWG1IF flag bit of the PIR7 register is set ([Register 14-5](#)).

24.11 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 shut-down source must be cleared before the restart can take place. That is, either the shutdown condition must be removed, or the corresponding ASxE bit must be cleared.

24.11.1 SOFTWARE-CONTROLLED RESTART

If the REN bit of the CWG1AS0 register 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 SHUTDOWN. Once SHUTDOWN is cleared, the CWG module will resume operation upon the first rising edge of the CWG data input.

Note: The SHUTDOWN bit cannot be cleared in software if the auto-shutdown condition is still present.

24.11.2 AUTO-RESTART

If the REN bit of the CWG1AS0 register 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 SHUTDOWN. Once SHUTDOWN is cleared, the CWG module will resume operation upon the first rising edge of the CWG data input.

Note: The SHUTDOWN bit cannot be cleared in software if the auto-shutdown condition is still present.

24.12 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.

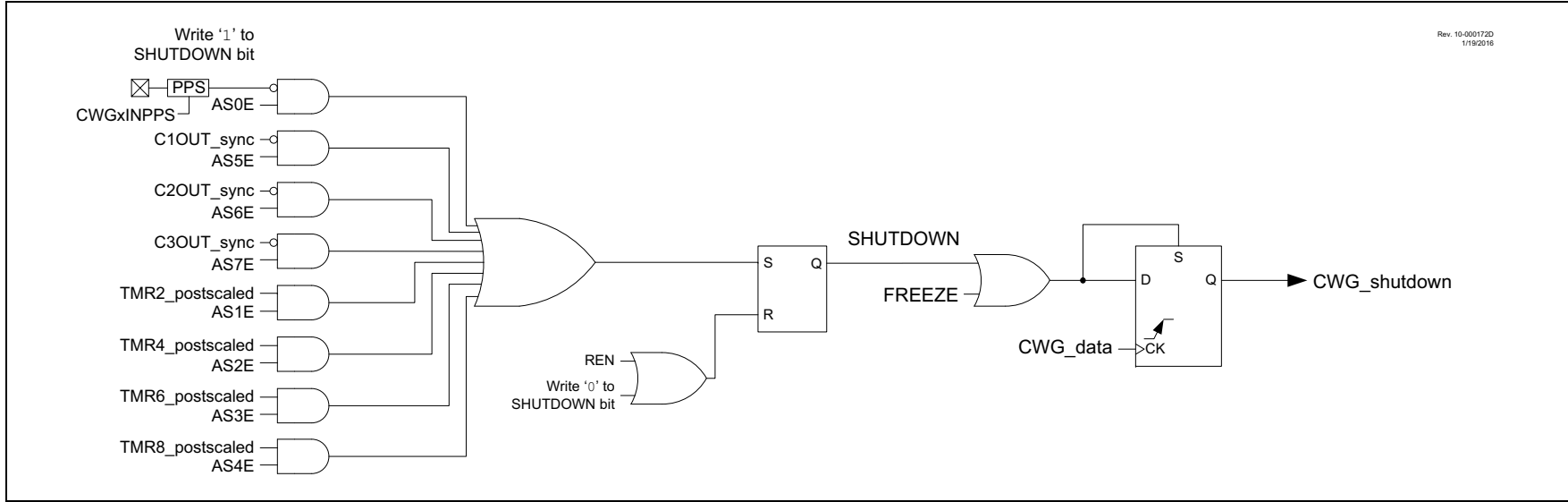
In other words, if the HFINTOSC is simultaneously selected as the system clock and the CWG clock source, when the CWG is enabled and the input source is active, then the CPU will go idle during Sleep, but the HFINTOSC will remain active and the CWG will continue to operate. This will have a direct effect on the Sleep mode current.

24.13 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 should ensure that pin levels are held to safe levels.
2. Clear the EN bit, if not already cleared.
3. Configure the MODE<2:0> bits of the CWG1CON0 register to set the output operating mode.
4. Configure the POLy bits of the CWG1CON1 register to set the output polarities.
5. Configure the ISM<3:0> bits of the CWG1ISM register to select the data input source.
6. If a steering mode is selected, configure the STRx bits to select the desired output on the CWG outputs.
7. Configure the LSBD<1:0> and LSAC<1:0> bits of the CWG1ASD0 register 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 of CWG1AS0.
9. If auto-shutdown is desired, configure the ASxE bits of the CWG1AS1 register to select the shutdown source.
10. Set the desired rising and falling dead-band times with the CWG1DBR and CWG1DBF registers.
11. Select the clock source in the CWG1CLKCON register.
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.

FIGURE 24-14: CWG SHUTDOWN BLOCK DIAGRAM



Rev. 10-000172D
1/19/2016

FIGURE 24-15: SHUTDOWN FUNCTIONALITY, AUTO-RESTART DISABLED (REN = 0, LSAC = 01, LSBD = 01)

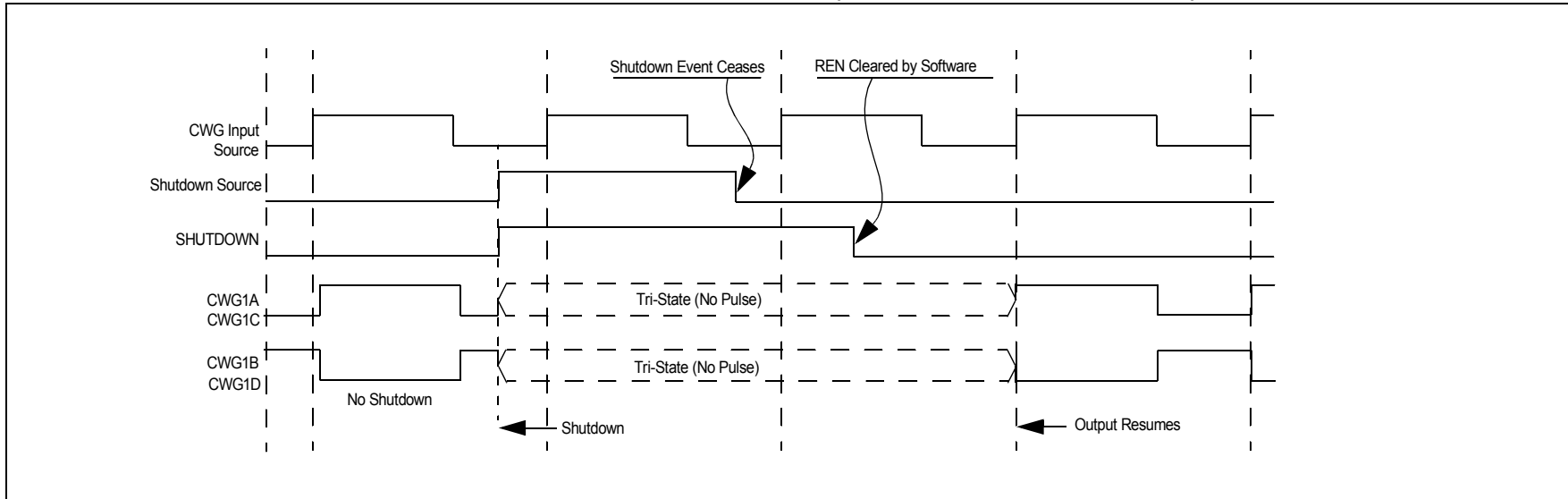
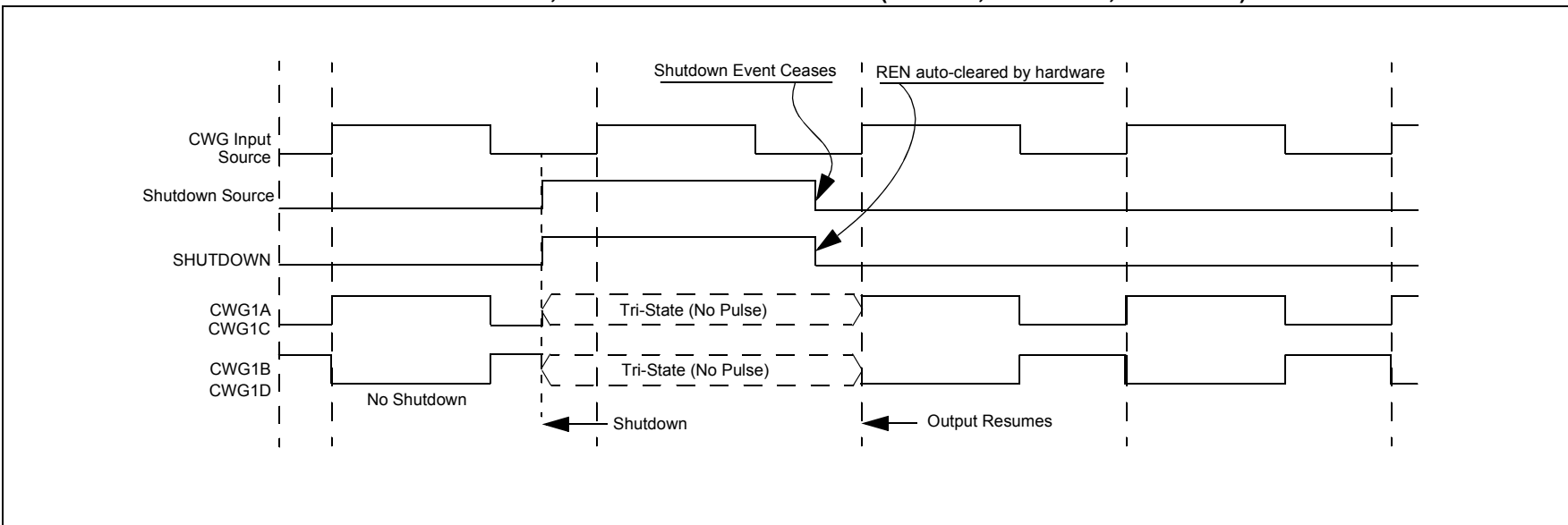


FIGURE 24-16: SHUTDOWN FUNCTIONALITY, AUTO-RESTART ENABLED (REN = 1, LSAC = 01, LSBD = 01)



24.14 Register Definitions: CWG Control

Long bit name prefixes for the CWG peripheral is shown in [Table 24-2](#). Refer to [Section 1.4.2.2 “Long Bit Names”](#) for more information.

TABLE 24-2:

| Peripheral | Bit Name Prefix |
|------------|-----------------|
| CWG | CWG |

REGISTER 24-1: CWG1CON0: CWG CONTROL REGISTER 0

| R/W-0/0 | R/W/HC-0/0 | U-0 | U-0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|---------|-------------------|-----|-----|-----|-----------|---------|---------|
| EN | LD ⁽¹⁾ | — | — | — | MODE<2:0> | | |
| bit 7 | | | | | | bit 0 | |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | HC = Bit is cleared by hardware |

| | |
|---------|---|
| bit 7 | EN: CWG1 Enable bit 1 = Module is enabled 0 = Module is disabled |
| bit 6 | LD: CWG1 Load Buffers bit ⁽¹⁾ 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 |
| bit 5-3 | Unimplemented: Read as '0' |
| bit 2-0 | MODE<2:0>: CWG1 Mode bits 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.

REGISTER 24-2: CWG1CON1: CWG CONTROL REGISTER 1

| | | | | | | | |
|-------|-----|-----|-----|---------|---------|---------|---------|
| U-0 | U-0 | R-x | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| — | — | IN | — | POLD | POLC | POLB | POLA |
| bit 7 | | | | | | | bit 0 |

Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

q = Value depends on condition

| | |
|---------|--|
| bit 7-6 | Unimplemented: Read as '0' |
| bit 5 | IN: CWG Input Value bit (read-only) |
| bit 4 | Unimplemented: Read as '0' |
| bit 3 | POLD: CWG1D Output Polarity bit 1 = Signal output is inverted polarity 0 = Signal output is normal polarity |
| bit 2 | POLC: CWG1C Output Polarity bit 1 = Signal output is inverted polarity 0 = Signal output is normal polarity |
| bit 1 | POLB: CWG1B Output Polarity bit 1 = Signal output is inverted polarity 0 = Signal output is normal polarity |
| bit 0 | POLA: CWG1A Output Polarity bit 1 = Signal output is inverted polarity 0 = Signal output is normal polarity |

REGISTER 24-3: CWG1CLKCON: CWG1 CLOCK INPUT SELECTION REGISTER

| | | | | | | | |
|-------|-----|-----|-----|-----|-----|-----|---------|
| U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | R/W-0/0 |
| — | — | — | — | — | — | — | CS |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | q = Value depends on condition |

bit 7-1 **Unimplemented:** Read as '0'
bit 0 **CS:** CWG Clock Source Selection Select bits

| CS | Clock Source |
|----|---|
| 1 | HFINTOSC (remains operating during Sleep) |
| 0 | Fosc |

REGISTER 24-4: CWG1ISM: CWGx INPUT SELECTION REGISTER

| | | | | | | | |
|-------|-----|-----|-----|----------|---------|---------|---------|
| U-0 | U-0 | U-0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| — | — | — | — | ISM<3:0> | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | q = Value depends on condition |

bit 7-4 **Unimplemented** Read as '0'
bit 3-0 **ISM<3:0>:** CWG Data Input Selection Multiplexer Select bits

| ISM<3:0> | Input Source |
|-----------|-------------------------|
| 1111-1100 | Reserved |
| 1011 | DSM OUT |
| 1010 | CMP3 OUT |
| 1001 | CMP2 OUT |
| 1000 | CMP1 OUT |
| 0111 | PWM7 OUT |
| 0110 | PWM6 OUT |
| 0101 | CCP5 OUT |
| 0100 | CCP4 OUT |
| 0011 | CCP3 OUT |
| 0010 | CCP2 OUT |
| 0001 | CCP1 OUT |
| 0000 | Pin selected by CWG1PPS |

REGISTER 24-5: CWG1STR⁽¹⁾: CWG STEERING CONTROL REGISTER

| | | | | | | | |
|---------|---------|---------|---------|---------------------|---------------------|---------------------|---------------------|
| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| OVRD | OVRC | OVRB | OVRA | STRD ⁽²⁾ | STRC ⁽²⁾ | STRB ⁽²⁾ | STRA ⁽²⁾ |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | q = Value depends on condition |

| | |
|-------|--|
| bit 7 | OVRD: Steering Data D bit |
| bit 6 | OVRC: Steering Data C bit |
| bit 5 | OVRB: Steering Data B bit |
| bit 4 | OVRA: Steering Data A bit |
| bit 3 | STRD: Steering Enable bit D ⁽²⁾ 1 = CWG1D output has the CWG data input waveform with polarity control from POLD bit 0 = CWG1D output is assigned to value of OVRD bit |
| bit 2 | STRC: Steering Enable bit C ⁽²⁾ 1 = CWG1C output has the CWG data input waveform with polarity control from POLC bit 0 = CWG1C output is assigned to value of OVRC bit |
| bit 1 | STRB: Steering Enable bit B ⁽²⁾ 1 = CWG1B output has the CWG data input waveform with polarity control from POLB bit 0 = CWG1B output is assigned to value of OVRB bit |
| bit 0 | STRA: Steering Enable bit A ⁽²⁾ 1 = CWG1A output has the CWG data input waveform with polarity control from POLA bit 0 = CWG1A output is assigned to value of OVRA bit |

- Note 1:** The bits in this register apply only when MODE<2:0> = 00x (Register 24-1, Steering modes).
Note 2: This bit is double-buffered when MODE<2:0> = 001.

REGISTER 24-6: CWG1AS0: CWG AUTO-SHUTDOWN CONTROL REGISTER 0

| | | | | | | | |
|---------------|---------|-----------|---------|-----------|---------|-------|-----|
| R/W/HS/HC-0/0 | R/W-0/0 | R/W-0/0 | R/W-1/1 | R/W-0/0 | R/W-1/1 | U-0 | U-0 |
| SHUTDOWN | REN | LSBD<1:0> | | LSAC<1:0> | | — | — |
| bit 7 | | | | | | bit 0 | |

Legend:

| | | |
|--------------------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | HS/HC = Bit is set/cleared by hardware |
| q = Value depends on condition | | |

bit 7 **SHUTDOWN:** Auto-Shutdown Event Status bit^(1,2)

- 1 = An auto-shutdown state is in effect
- 0 = No auto-shutdown event has occurred

bit 6 **REN:** Auto-Restart Enable bit

- 1 = Auto-restart is enabled
- 0 = Auto-restart is disabled

bit 5-4 **LSBD<1:0>:** CWG1B and CWG1D Auto-Shutdown State Control bits

- 11 = A logic '1' is placed on CWG1B/D when an auto-shutdown event occurs.
- 10 = A logic '0' is placed on CWG1B/D when an auto-shutdown event occurs.
- 01 = Pin is tri-stated on CWG1B/D when an auto-shutdown event occurs.
- 00 = The inactive state of the pin, including polarity, is placed on CWG1B/D after the required dead-band interval when an auto-shutdown event occurs.

bit 3-2 **LSAC<1:0>:** CWG1A and CWG1C Auto-Shutdown State Control bits

- 11 = A logic '1' is placed on CWG1A/C when an auto-shutdown event occurs.
- 10 = A logic '0' is placed on CWG1A/C when an auto-shutdown event occurs.
- 01 = Pin is tri-stated on CWG1A/C when an auto-shutdown event occurs.
- 00 = The inactive state of the pin, including polarity, is placed on CWG1A/C after the required dead-band interval when an auto-shutdown event occurs.

bit 1-0 **Unimplemented:** Read as '0'

- Note 1:** This bit may be written while EN = 0 ([Register 24-1](#)), 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.

REGISTER 24-7: CWG1AS1: CWG AUTO-SHUTDOWN CONTROL REGISTER 1

| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
|---------|---------|---------|---------|---------|---------|---------|---------|
| AS7E | AS6E | AS5E | AS4E | AS3E | AS2E | AS1E | AS0E |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | q = Value depends on condition |

| | |
|-------|---|
| bit 7 | AS7E: CWG Auto-shutdown Source 7 (CMP3 OUT) Enable bit 1 = Auto-shutdown for CMP3 OUT is enabled 0 = Auto-shutdown for CMP3 OUT is disabled |
| bit 6 | AS6E: CWG Auto-shutdown Source 6 (CMP2 OUT) Enable bit 1 = Auto-shutdown for CMP2 OUT is enabled 0 = Auto-shutdown for CMP2 OUT is disabled |
| bit 5 | AS5E: CWG Auto-shutdown Source 5 (CMP1 OUT) Enable bit 1 = Auto-shutdown for CMP1 OUT is enabled 0 = Auto-shutdown for CMP1 OUT is disabled |
| bit 4 | AS4E: CWG Auto-shutdown Source 4 (TMR8_Postscaled) Enable bit 1 = Auto-shutdown for TMR8_Postscaled is enabled 0 = Auto-shutdown for TMR8_Postscaled is disabled |
| bit 3 | AS3E: CWG Auto-shutdown Source 3 (TMR6_Postscaled) Enable bit 1 = Auto-shutdown for TMR6_Postscaled is enabled 0 = Auto-shutdown for TMR6_Postscaled is disabled |
| bit 2 | AS2E: CWG Auto-shutdown Source 2 (TMR4_Postscaled) Enable bit 1 = Auto-shutdown for TMR4_Postscaled is enabled 0 = Auto-shutdown for TMR4_Postscaled is disabled |
| bit 1 | AS1E: CWG Auto-shutdown Source 1 (TMR2_Postscaled) Enable bit 1 = Auto-shutdown for TMR2_Postscaled is enabled 0 = Auto-shutdown for TMR2_Postscaled is disabled |
| bit 0 | AS0E: CWG Auto-shutdown Source 0 (Pin selected by CWG1PPS) Enable bit 1 = Auto-shutdown for CWG1PPS Pin is enabled 0 = Auto-shutdown for CWG1PPS Pin is disabled |

REGISTER 24-8: CWG1DBR: CWG RISING DEAD-BAND COUNT REGISTER

| U-0 | U-0 | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u |
|-------|-----|----------|---------|---------|---------|---------|---------|
| — | — | DBR<5:0> | | | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | q = Value depends on condition |

bit 7-6 **Unimplemented:** Read as '0'

bit 5-0 **DBR<5:0>:** CWG Rising Edge Triggered Dead-Band Count bits

11 1111 = 63-64 CWG clock periods

11 1110 = 62-63 CWG clock periods

.

.

.

00 0010 = 2-3 CWG clock periods

00 0001 = 1-2 CWG clock periods

00 0000 = 0 CWG clock periods. Dead-band generation is bypassed

REGISTER 24-9: CWG1DBF: CWG FALLING DEAD-BAND COUNT REGISTER

| U-0 | U-0 | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u |
|-------|-----|----------|---------|---------|---------|---------|---------|
| — | — | DBF<5:0> | | | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | q = Value depends on condition |

bit 7-6 **Unimplemented:** Read as '0'

bit 5-0 **DBF<5:0>:** CWG Falling Edge Triggered Dead-Band Count bits

11 1111 = 63-64 CWG clock periods

11 1110 = 62-63 CWG clock periods

.

.

.

00 0010 = 2-3 CWG clock periods

00 0001 = 1-2 CWG clock periods

00 0000 = 0 CWG clock periods. Dead-band generation is bypassed.

TABLE 24-3: SUMMARY OF REGISTERS ASSOCIATED WITH CWG

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on Page |
|------------|----------|-------|-----------|-------|-----------|-----------|--------|--------|------------------|
| CWG1CON0 | EN | LD | — | — | — | MODE<2:0> | | | 322 |
| CWG1CON1 | — | — | IN | — | POLD | POLC | POLB | POLA | 323 |
| CWG1CLKCON | — | — | — | — | — | — | — | CS | 324 |
| CWG1ISM | — | — | — | — | ISM<3:0> | | | | 324 |
| CWG1STR | OVRD | OVRC | OVRB | OVRA | STRD | STRC | STRB | STRA | 325 |
| CWG1AS0 | SHUTDOWN | REN | LSBD<1:0> | | LSAC<1:0> | | — | — | 326 |
| CWG1AS1 | AS7E | AS6E | AS5E | AS4E | AS3E | AS2E | AS1E | AS0E | 327 |
| CWG1DBR | — | — | DBR<5:0> | | | | | | 328 |
| CWG1DBF | — | — | DBF<5:0> | | | | | | 328 |
| PIE8 | SCANIE | CRCIE | NVMIE | — | — | — | — | CWG1IE | 192 |
| PIR8 | SCANIF | CRCIF | NVMIF | — | — | — | — | CWG1IF | 182 |
| IPR8 | SCANIP | CRCIP | NVMIP | — | — | — | — | CWG1IP | 202 |
| PMD2 | — | — | CWGMD | — | DSMMD | SMT2MD | SMT1MD | TMR8MD | 67 |

Legend: — = unimplemented locations read as '0'. Shaded cells are not used by CWG.

25.0 SIGNAL MEASUREMENT TIMER (SMT)

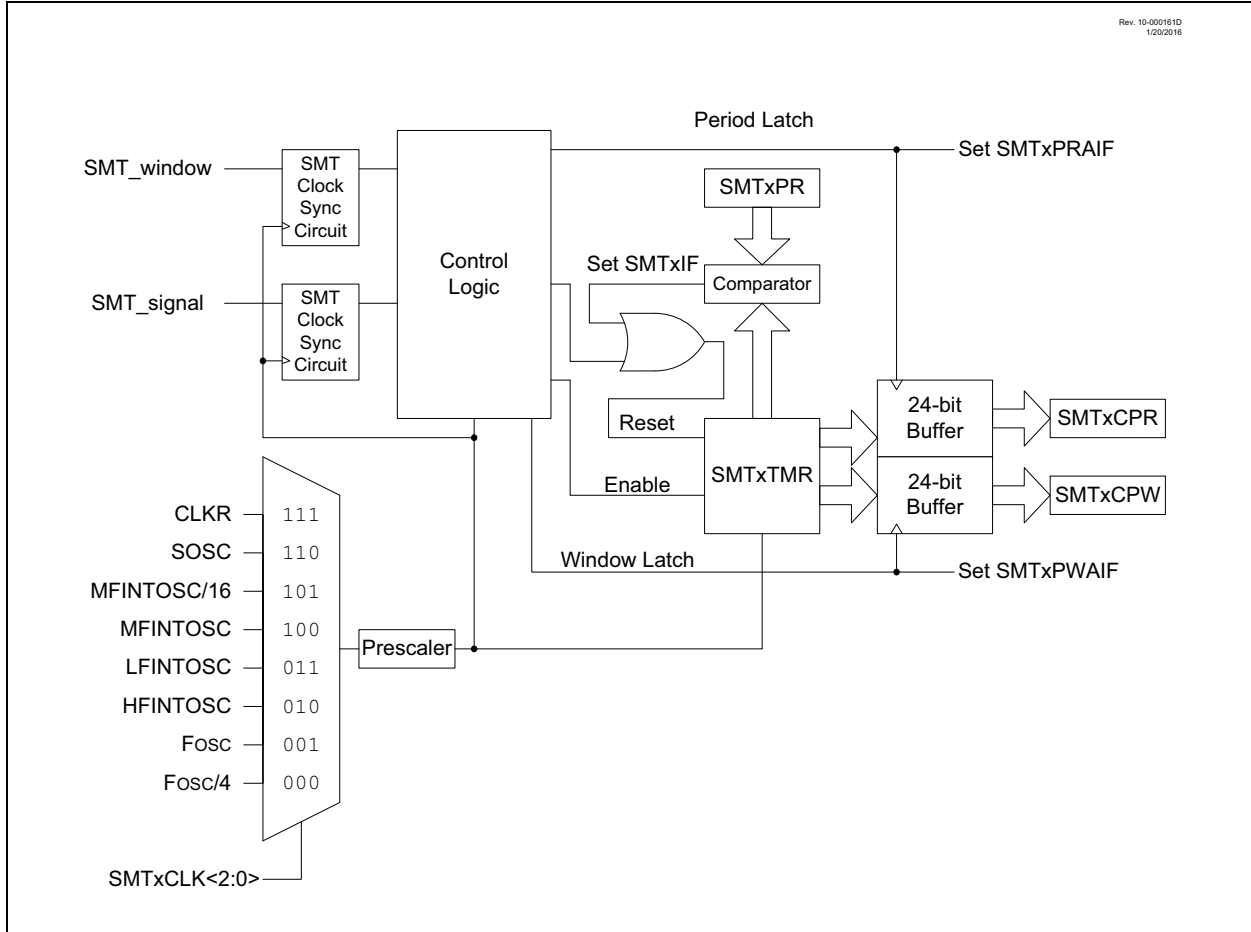
The SMT is a 24-bit counter with advanced clock and gating logic, which can be configured for measuring a variety of digital signal parameters such as pulse width, frequency and duty cycle, and the time difference between edges on two signals.

Features of the SMT include:

- 24-bit timer/counter
 - Three 8-bit registers (SMTxTMRL/H/U)
 - Readable and writable
 - Optional 16-bit operating mode
- Two 24-bit measurement capture registers
- One 24-bit period match register
- Multi-mode operation, including relative timing measurement
- Interrupt on period match
- Multiple clock, gate and signal sources
- Interrupt on acquisition complete

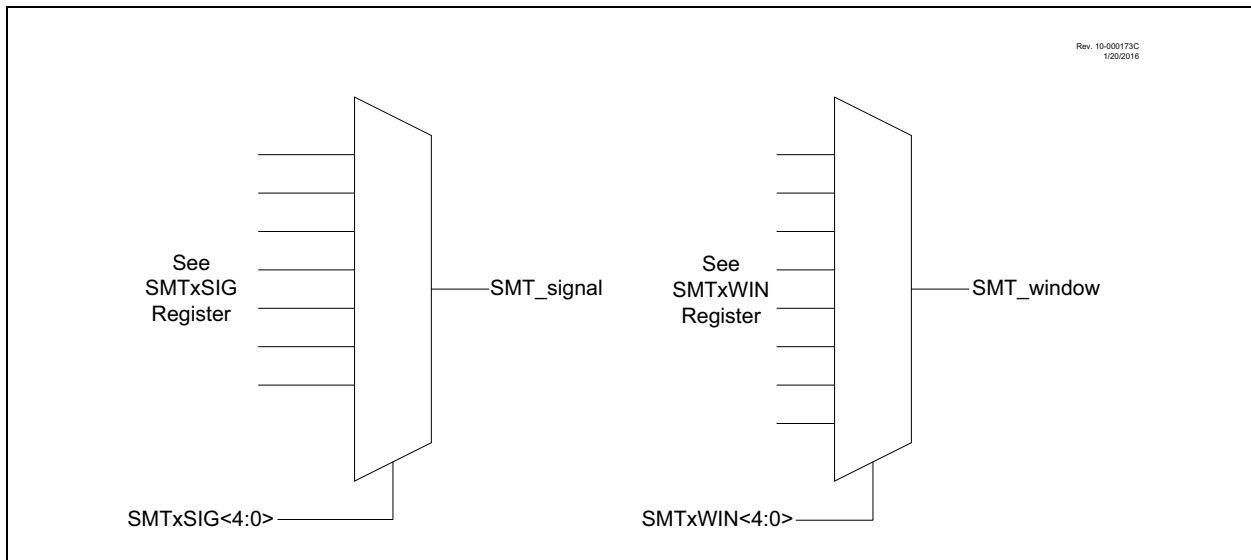
Ability to read current input values

FIGURE 25-1: SMTx BLOCK DIAGRAM



Note: These devices implement two SMT modules. All references to SMTx apply to SMT1 and SMT2.

FIGURE 25-2: SMTx SIGNAL AND WINDOW BLOCK DIAGRAM



25.1 Register Definitions: SMT Control

Long bit name prefixes for the SMT peripherals are shown in [Table 25-1](#). Refer to [Section 1.4.2.2 “Long Bit Names”](#) for more information.

TABLE 25-1:

| Peripheral | Bit Name Prefix |
|------------|-----------------|
| SMT1 | SMT1 |
| SMT2 | SMT2 |

REGISTER 25-1: SMTxCON0: SMT CONTROL REGISTER 0

| | | | | | | | |
|-------------------|-----|---------|---------|---------|---------|-------------|---------|
| R/W-0/0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| EN ⁽¹⁾ | — | STP | WPOL | SPOL | CPOL | SMTxPS<1:0> | |
| bit 7 | | | | | | bit 0 | |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

- bit 7 **EN:** SMT Enable bit⁽¹⁾
 1 = SMT is enabled
 0 = SMT is disabled; internal states are reset, clock requests are disabled
- bit 6 **Unimplemented:** Read as '0'
- bit 5 **STP:** SMT Counter Halt Enable bit
 When SMTxTMR = SMTxPR:
 1 = Counter remains SMTxPR; period match interrupt occurs when clocked
 0 = Counter resets to 24'h000000; period match interrupt occurs when clocked
- bit 4 **WPOL:** SMTxWIN Input Polarity Control bit
 1 = SMTxWIN signal is active-low/falling edge enabled
 0 = SMTxWIN signal is active-high/rising edge enabled
- bit 3 **SPOL:** SMTxSIG Input Polarity Control bit
 1 = SMTx_signal is active-low/falling edge enabled
 0 = SMTx_signal is active-high/rising edge enabled
- bit 2 **CPOL:** SMT Clock Input Polarity Control bit
 1 = SMTxTMR increments on the falling edge of the selected clock signal
 0 = SMTxTMR increments on the rising edge of the selected clock signal
- bit 1-0 **SMTxPS<1:0>:** SMT Prescale Select bits
 11 = Prescaler = 1:8
 10 = Prescaler = 1:4
 01 = Prescaler = 1:2
 00 = Prescaler = 1:1

Note 1: Setting EN to '0' does not affect the register contents.

REGISTER 25-2: SMTxCON1: SMT CONTROL REGISTER 1

| | | | | | | | |
|------------|---------|-----|-----|-----------|---------|---------|---------|
| R/W/HC-0/0 | R/W-0/0 | U-0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| SMTxGO | REPEAT | — | — | MODE<3:0> | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

HC = Bit is cleared by hardware

HS = Bit is set by hardware

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

q = Value depends on condition

- bit 7 **SMTxGO:** SMT GO Data Acquisition bit
 1 = Incrementing, acquiring data is enabled
 0 = Incrementing, acquiring data is disabled
- bit 6 **REPEAT:** SMT Repeat Acquisition Enable bit
 1 = Repeat Data Acquisition mode is enabled
 0 = Single Acquisition mode is enabled
- bit 5-4 **Unimplemented:** Read as '0'
- bit 3-0 **MODE<3:0>** SMT Operation Mode Select bits
 1111 = Reserved
 •
 •
 •
 1011 = Reserved
 1010 = Windowed counter
 1001 = Gated counter
 1000 = Counter
 0111 = Capture
 0110 = Time of flight
 0101 = Gated windowed measure
 0100 = Windowed measure
 0011 = High and low time measurement
 0010 = Period and Duty-Cycle Acquisition
 0001 = Gated Timer
 0000 = Timer

REGISTER 25-3: SMTxSTAT: SMT STATUS REGISTER

| | | | | | | | |
|------------|------------|------------|-----|-----|-------|-------|-------|
| R/W/HC-0/0 | R/W/HC-0/0 | R/W/HC-0/0 | U-0 | U-0 | R-0/0 | R-0/0 | R-0/0 |
| CPRUP | CPWUP | RST | — | — | TS | WS | AS |
| bit 7 | | | | | | | bit 0 |

Legend:

HC = Bit is cleared by hardware

HS = Bit is set by hardware

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

q = Value depends on condition

- bit 7 **CPRUP:** SMT Manual Period Buffer Update bit
 1 = Request update to SMTxCPRx registers
 0 = SMTxCPRx registers update is complete
- bit 6 **CPWUP:** SMT Manual Pulse Width Buffer Update bit
 1 = Request update to SMTxCPW registers
 0 = SMTxCPW registers update is complete
- bit 5 **RST:** SMT Manual Timer Reset bit
 1 = Request Reset to SMTxTMR registers
 0 = SMTxTMR registers update is complete
- bit 4-3 **Unimplemented:** Read as '0'
- bit 2 **TS:** SMT GO Value Status bit
 1 = SMT timer is incrementing
 0 = SMT timer is not incrementing
- bit 1 **WS:** SMTxWIN Value Status bit
 1 = SMT window is open
 0 = SMT window is closed
- bit 0 **AS:** SMT_signal Value Status bit
 1 = SMT acquisition is in progress
 0 = SMT acquisition is not in progress

REGISTER 25-4: SMTxCLK: SMT CLOCK SELECTION REGISTER

| | | | | | | | |
|-------|-----|-----|-----|-----|-----------|---------|---------|
| U-0 | U-0 | U-0 | U-0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| — | — | — | — | — | CSEL<2:0> | | |
| bit 7 | | | | | bit 0 | | |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | q = Value depends on condition |

bit 7-3 **Unimplemented:** Read as '0'

bit 2-0 **CSEL<2:0>:** SMT Clock Selection bits

- 111 = Clock reference out
- 110 = SOSC
- 101 = MFINTOSC/16 (31.25kHz)
- 100 = MFINTOSC (500 kHz)
- 011 = LFINTOSC
- 010 = HFINTOSC
- 001 = FOSC
- 000 = FOSC/4

REGISTER 25-5: SMTxWIN: SMTx WINDOW INPUT SELECT REGISTER

| | | | | | | | |
|-------|-----|-----|-----------|---------|---------|---------|---------|
| U-0 | U-0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| — | — | — | WSEL<4:0> | | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | q = Value depends on condition |

bit 7-5 **Unimplemented:** Read as '0'

bit 4-0 **WSEL<4:0>:** SMTx Window Selection bits

| WSEL | SMT1 Window Source | SMT2 Window Source |
|-------------|----------------------------|----------------------------|
| 11111-10110 | Reserved | Reserved |
| 10101 | ZCDOUT | ZCDOUT |
| 10100 | C3OUT | C3OUT |
| 10011 | C2OUT | C2OUT |
| 10010 | C1OUT | C1OUT |
| 10001 | PWM7OUT | PWM7OUT |
| 10000 | PWM6OUT | PWM6OUT |
| 01111 | CCP5OUT | CCP5OUT |
| 01110 | CCP4OUT | CCP4OUT |
| 01101 | CCP3OUT | CCP3OUT |
| 01100 | CCP2OUT | CCP2OUT |
| 01011 | CCP1OUT | CCP1OUT |
| 01010 | SMT2 overflow | Reserved |
| 01001 | Reserved | SMT1 overflow |
| 01000 | TMR8_postscaler | TMR8_postscaler |
| 00111 | TMR6_postscaler | TMR6_postscaler |
| 00110 | TMR4_postscaler | TMR4_postscaler |
| 00101 | TMR2_postscaler | TMR2_postscaler |
| 00100 | TMR0_overflow | TMR0_overflow |
| 00011 | SOSC | SOSC |
| 00010 | MFINTOSC (31 kHz) | MFINTOSC (31 kHz) |
| 00001 | LFINTOSC (31 kHz) | LFINTOSC (31 kHz) |
| 00000 | Pin selected by SMT1WINPPS | Pin selected by SMT2WINPPS |

REGISTER 25-6: SMTxSIG: SMTx SIGNAL INPUT SELECT REGISTER

| | | | | | | | |
|-------|-----|-----|-----------|---------|---------|---------|---------|
| U-0 | U-0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| — | — | — | SSEL<4:0> | | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | q = Value depends on condition |

bit 7-5 **Unimplemented:** Read as '0'

bit 4-0 **SSEL<4:0>:** SMTx Signal Selection bits

| SSEL | SMT1 Signal Source | SMT2 Signal Source |
|-------------|----------------------------|----------------------------|
| 11111-10111 | Reserved | Reserved |
| 10110 | ZCDOUT | ZCDOUT |
| 10101 | C3OUT | C3OUT |
| 10100 | C2OUT | C2OUT |
| 10011 | C1OUT | C1OUT |
| 10010 | PWM7OUT | PWM7OUT |
| 10001 | PWM6OUT | PWM6OUT |
| 10000 | CCP5OUT | CCP5OUT |
| 01111 | CCP4OUT | CCP4OUT |
| 01110 | CCP3OUT | CCP3OUT |
| 01101 | CCP2OUT | CCP2OUT |
| 01100 | CCP1OUT | CCP1OUT |
| 01011 | SMT2 overflow | Reserved |
| 01010 | Reserved | SMT1 overflow |
| 01001 | TMR8_postscaler | TMR8_postscaler |
| 01000 | TMR7_postscaler | TMR7_postscaler |
| 00111 | TMR6_postscaler | TMR6_postscaler |
| 00110 | TMR5_postscaler | TMR5_postscaler |
| 00101 | TMR4_postscaler | TMR4_postscaler |
| 00100 | TMR3_postscaler | TMR3_postscaler |
| 00011 | TMR2_postscaler | TMR2_postscaler |
| 00010 | TMR1_postscaler | TMR1_postscaler |
| 00001 | TMR0_postscaler | TMR0_postscaler |
| 00000 | Pin selected by SMT1SIGPPS | Pin selected by SMT2SIGPPS |

REGISTER 25-7: SMTxTMRL: SMT TIMER REGISTER – LOW BYTE

| | | | | | | | |
|--------------|---------|---------|---------|---------|---------|---------|---------|
| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| SMTxTMR<7:0> | | | | | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-0 **SMTxTMR<7:0>**: Significant bits of the SMT Counter – Low Byte

REGISTER 25-8: SMTxTMRH: SMT TIMER REGISTER – HIGH BYTE

| | | | | | | | |
|---------------|---------|---------|---------|---------|---------|---------|---------|
| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| SMTxTMR<15:8> | | | | | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-0 **SMTxTMR<15:8>**: Significant bits of the SMT Counter – High Byte

REGISTER 25-9: SMTxTMRU: SMT TIMER REGISTER – UPPER BYTE

| | | | | | | | |
|----------------|---------|---------|---------|---------|---------|---------|---------|
| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| SMTxTMR<23:16> | | | | | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-0 **SMTxTMR<23:16>**: Significant bits of the SMT Counter – Upper Byte

REGISTER 25-10: SMTxCPRL: SMT CAPTURED PERIOD REGISTER – LOW BYTE

| | | | | | | | |
|--------------|-------|-------|-------|-------|-------|-------|-------|
| R-x/x | R-x/x | R-x/x | R-x/x | R-x/x | R-x/x | R-x/x | R-x/x |
| SMTxCPR<7:0> | | | | | | | |
| bit 7 | | | | bit 0 | | | |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-0 **SMTxCPR<7:0>**: Significant bits of the SMT Period Latch – Low Byte

REGISTER 25-11: SMTxCPRH: SMT CAPTURED PERIOD REGISTER – HIGH BYTE

| | | | | | | | |
|---------------|-------|-------|-------|-------|-------|-------|-------|
| R-x/x | R-x/x | R-x/x | R-x/x | R-x/x | R-x/x | R-x/x | R-x/x |
| SMTxCPR<15:8> | | | | | | | |
| bit 7 | | | | bit 0 | | | |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-0 **SMTxCPR<15:8>**: Significant bits of the SMT Period Latch – High Byte

REGISTER 25-12: SMTxCPRU: SMT CAPTURED PERIOD REGISTER – UPPER BYTE

| | | | | | | | |
|----------------|-------|-------|-------|-------|-------|-------|-------|
| R-x/x | R-x/x | R-x/x | R-x/x | R-x/x | R-x/x | R-x/x | R-x/x |
| SMTxCPR<23:16> | | | | | | | |
| bit 7 | | | | bit 0 | | | |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-0 **SMTxCPR<23:16>**: Significant bits of the SMT Period Latch – Upper Byte

REGISTER 25-13: SMTxCPWL: SMT CAPTURED PULSE WIDTH REGISTER – LOW BYTE

| | | | | | | | |
|--------------|-------|-------|-------|-------|-------|-------|-------|
| R-x/x | R-x/x | R-x/x | R-x/x | R-x/x | R-x/x | R-x/x | R-x/x |
| SMTxCPW<7:0> | | | | | | | |
| bit 7 | | | | bit 0 | | | |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-0 **SMTxCPW<7:0>**: Significant bits of the SMT PW Latch – Low Byte

REGISTER 25-14: SMTxCPWH: SMT CAPTURED PULSE WIDTH REGISTER – HIGH BYTE

| | | | | | | | |
|---------------|-------|-------|-------|-------|-------|-------|-------|
| R-x/x | R-x/x | R-x/x | R-x/x | R-x/x | R-x/x | R-x/x | R-x/x |
| SMTxCPW<15:8> | | | | | | | |
| bit 7 | | | | bit 0 | | | |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-0 **SMTxCPW<15:8>**: Significant bits of the SMT PW Latch – High Byte

REGISTER 25-15: SMTxCPWU: SMT CAPTURED PULSE WIDTH REGISTER – UPPER BYTE

| | | | | | | | |
|----------------|-------|-------|-------|-------|-------|-------|-------|
| R-x/x | R-x/x | R-x/x | R-x/x | R-x/x | R-x/x | R-x/x | R-x/x |
| SMTxCPW<23:16> | | | | | | | |
| bit 7 | | | | bit 0 | | | |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-0 **SMTxCPW<23:16>**: Significant bits of the SMT PW Latch – Upper Byte

REGISTER 25-16: SMTxPRL: SMT PERIOD REGISTER – LOW BYTE

| | | | | | | | |
|-------------|---------|---------|---------|---------|---------|---------|---------|
| R/W-x/1 | R/W-x/1 | R/W-x/1 | R/W-x/1 | R/W-x/1 | R/W-x/1 | R/W-x/1 | R/W-x/1 |
| SMTxPR<7:0> | | | | | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-0 **SMTxPR<7:0>**: Significant bits of the SMT Timer Value for Period Match – Low Byte

REGISTER 25-17: SMTxPRH: SMT PERIOD REGISTER – HIGH BYTE

| | | | | | | | |
|--------------|---------|---------|---------|---------|---------|---------|---------|
| R/W-x/1 | R/W-x/1 | R/W-x/1 | R/W-x/1 | R/W-x/1 | R/W-x/1 | R/W-x/1 | R/W-x/1 |
| SMTxPR<15:8> | | | | | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-0 **SMTxPR<15:8>**: Significant bits of the SMT Timer Value for Period Match – High Byte

REGISTER 25-18: SMTxPRU: SMT PERIOD REGISTER – UPPER BYTE

| | | | | | | | |
|---------------|---------|---------|---------|---------|---------|---------|---------|
| R/W-x/1 | R/W-x/1 | R/W-x/1 | R/W-x/1 | R/W-x/1 | R/W-x/1 | R/W-x/1 | R/W-x/1 |
| SMTxPR<23:16> | | | | | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-0 **SMTxPR<23:16>**: Significant bits of the SMT Timer Value for Period Match – Upper Byte

TABLE 25-2: SUMMARY OF REGISTERS ASSOCIATED WITH SMTx

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on Page | |
|----------|----------------|--------|-----------|-----------|-----------|-----------|-------------|--------|------------------|-----|
| PIE9 | — | — | SMT2PWAIE | SMT2PRAIE | SMT2IE | SMT1PWAIE | SMT1PRAIE | SMT1IE | 188 | |
| PIR9 | — | — | SMT2PWAIF | SMT2PRAIF | SMT2IF | SMT1PWAIF | SMT1PRAIF | SMT1IF | 175 | |
| IPR9 | — | — | SMT2PWAIP | SMT2PRAIP | SMT2IP | SMT1PWAIP | SMT1PRAIP | SMT1IP | 200 | |
| SMT1CLK | — | — | — | — | — | CSEL<2:0> | | | 335 | |
| SMT1CON0 | EN | — | STP | WPOL | SPOL | CPOL | SMT1PS<1:0> | | 332 | |
| SMT1CON1 | SMT1GO | REPEAT | — | — | MODE<3:0> | | | | 333 | |
| SMT1CPRH | SMT1CPR<15:8> | | | | | | | | 339 | |
| SMT1CPRL | SMT1CPR<7:0> | | | | | | | | 339 | |
| SMT1CPRU | SMT1CPR<23:16> | | | | | | | | 339 | |
| SMT1CPWH | SMT1CPW<15:8> | | | | | | | | 340 | |
| SMT1CPWL | SMT1CPW<7:0> | | | | | | | | 340 | |
| SMT1CPWU | SMT1CPW<23:16> | | | | | | | | 340 | |
| SMT1PRH | SMT1PR<15:8> | | | | | | | | 341 | |
| SMT1PRL | SMT1PR<7:0> | | | | | | | | 341 | |
| SMT1PRU | SMT1PR<23:16> | | | | | | | | 341 | |
| SMT1SIG | — | — | — | SSEL<4:0> | | | | | | 337 |
| SMT1STAT | CPRUP | CPWUP | RST | — | — | TS | WS | AS | 334 | |
| SMT1TMRH | SMT1TMR<15:8> | | | | | | | | 338 | |
| SMT1TMRL | SMT1TMR<7:0> | | | | | | | | 338 | |
| SMT1TMRU | SMT1TMR<23:16> | | | | | | | | 338 | |
| SMT1WIN | — | — | — | WSEL<4:0> | | | | | | 336 |
| SMT2CLK | — | — | — | — | — | CSEL<2:0> | | | 335 | |
| SMT2CON0 | EN | — | STP | WPOL | SPOL | CPOL | SMT2PS<1:0> | | 332 | |
| SMT2CON1 | SMT2GO | REPEAT | — | — | MODE<3:0> | | | | 333 | |
| SMT2CPRH | SMT2CPR<15:8> | | | | | | | | 339 | |
| SMT2CPRL | SMT2CPR<7:0> | | | | | | | | 339 | |
| SMT2CPRU | SMT2CPR<23:16> | | | | | | | | 339 | |
| SMT2CPWH | SMT2CPW<15:8> | | | | | | | | 340 | |
| SMT2CPWL | SMT2CPW<7:0> | | | | | | | | 340 | |
| SMT2CPWU | SMT2CPW<23:16> | | | | | | | | 340 | |
| SMT2PRH | SMT2PR<15:8> | | | | | | | | 341 | |
| SMT2PRL | SMT2PR<7:0> | | | | | | | | 341 | |
| SMT2PRU | SMT2PR<23:16> | | | | | | | | 341 | |
| SMT2SIG | — | — | — | SSEL<4:0> | | | | | | 337 |
| SMT2STAT | CPRUP | CPWUP | RST | — | — | TS | WS | AS | 334 | |
| SMT2TMRH | SMT2TMR<15:8> | | | | | | | | 338 | |
| SMT2TMRL | SMT2TMR<7:0> | | | | | | | | 338 | |
| SMT2TMRU | SMT2TMR<23:16> | | | | | | | | 338 | |
| SMT2WIN | — | — | — | WSEL<4:0> | | | | | | 335 |

Legend: x = unknown, u = unchanged, — = unimplemented read as '0', q = value depends on condition. Shaded cells are not used for SMTx module.

25.2 SMT Operation

The core of the module is the 24-bit counter, SMTxTMR combined with a complex data acquisition front-end. Depending on the mode of operation selected, the SMT can perform a variety of measurements summarized in [Table 25-3](#).

25.2.1 CLOCK SOURCES

Clock sources available to the SMT include:

- FOSC
- FOSC/4
- HFINTOSC
- MFINTOSC (500 kHz and 31.25 kHz)
- LFINTOSC
- SOSC
- Clock reference out

The SMT clock source is selected by configuring the CSEL<2:0> bits in the SMTxCLK register. The clock source can also be prescaled using the PS<1:0> bits of the SMTxCON0 register. The prescaled clock source is used to clock both the counter and any synchronization logic used by the module.

25.2.2 PERIOD MATCH INTERRUPT

Similar to other timers, the SMT triggers an interrupt when SMTxTMR rolls over to '0'. This happens when SMTxTMR = SMTxPR, regardless of mode. Hence, in any mode that relies on an external signal or a window to reset the timer, proper operation requires that SMTxPR be set to a period larger than that of the expected signal or window.

25.3 Basic Timer Function Registers

The SMTxTMR time base and the SMTxCPW/SMTxPR/SMTxCPR buffer registers serve several functions and can be manually updated using software.

25.3.1 TIME BASE

The SMTxTMR is the 24-bit counter that is the center of the SMT. It is used as the basic counter/timer for measurement in each of the modes of the SMT. It can be reset to a value of 24'h00_0000 by setting the RST bit of the SMTxSTAT register. It can be written to and read from software, but it is not guarded for atomic access, therefore reads and writes to the SMTxTMR should only be made when the GO = 0, or the software should have other measures to ensure integrity of SMTxTMR reads/writes.

25.3.2 PULSE WIDTH LATCH REGISTERS

The SMTxCPW registers are the 24-bit SMT pulse width latch. They are used to latch in the value of the SMTxTMR when triggered by various signals, which are determined by the mode the SMT is currently in.

The SMTxCPW registers can also be updated with the current value of the SMTxTMR value by setting the CPWUP bit of the SMTxSTAT register.

25.3.3 PERIOD LATCH REGISTERS

The SMTxCPR registers are the 24-bit SMT period latch. They are used to latch in other values of the SMTxTMR when triggered by various other signals, which are determined by the mode the SMT is currently in.

The SMTxCPR registers can also be updated with the current value of the SMTxTMR value by setting the CPRUP bit in the SMTxSTAT register.

25.4 Halt Operation

The counter can be prevented from rolling-over using the STP bit in the SMTxCON0 register. When halting is enabled, the period match interrupt persists until the SMTxTMR is reset (either by a manual reset, [Section 25.3.1 "Time Base"](#)) or by clearing the SMTxGO bit of the SMTxCON1 register and writing the SMTxTMR values in software.

25.5 Polarity Control

The three input signals for the SMT have polarity control to determine whether or not they are active high/positive edge or active low/negative edge signals.

The following bits apply to Polarity Control:

- WSEL bit (Window Polarity)
- SSEL bit (Signal Polarity)
- CSEL bit (Clock Polarity)

These bits are located in the SMTxCON0 register.

25.6 Status Information

The SMT provides input status information for the user without requiring the need to deal with the polarity of the incoming signals.

25.6.1 WINDOW STATUS

Window status is determined by the WS bit of the SMTxSTAT register. This bit is only used in Windowed Measure, Gated Counter and Gated Window Measure modes, and is only valid when TS = 1, and will be delayed in time by synchronizer delays in non-Counter modes.

25.6.2 SIGNAL STATUS

Signal status is determined by the AS bit of the SMTxSTAT register. This bit is used in all modes except Window Measure, Time of Flight and Capture modes, and is only valid when TS = 1, and will be delayed in time by synchronizer delays in non-Counter modes.

25.6.3 GO STATUS

Timer run status is determined by the TS bit of the SMTxSTAT register, and will be delayed in time by synchronizer delays in non-Counter modes.

25.7 Modes of Operation

The modes of operation are summarized in [Table 25-3](#). The following sections provide detailed descriptions, examples of how the modes can be used. Note that all waveforms assume WPOL/SPOL/CPOL = 0. When WPOL/SPOL/CPOL = 1, all SMTSIGx, SMTWINx and SMT clock signals will have a polarity opposite to that indicated. For all modes, the REPEAT bit controls whether the acquisition is repeated or single. When REPEAT = 0 (Single Acquisition mode), the timer will stop incrementing and the SMTxGO bit will be reset 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 in software.

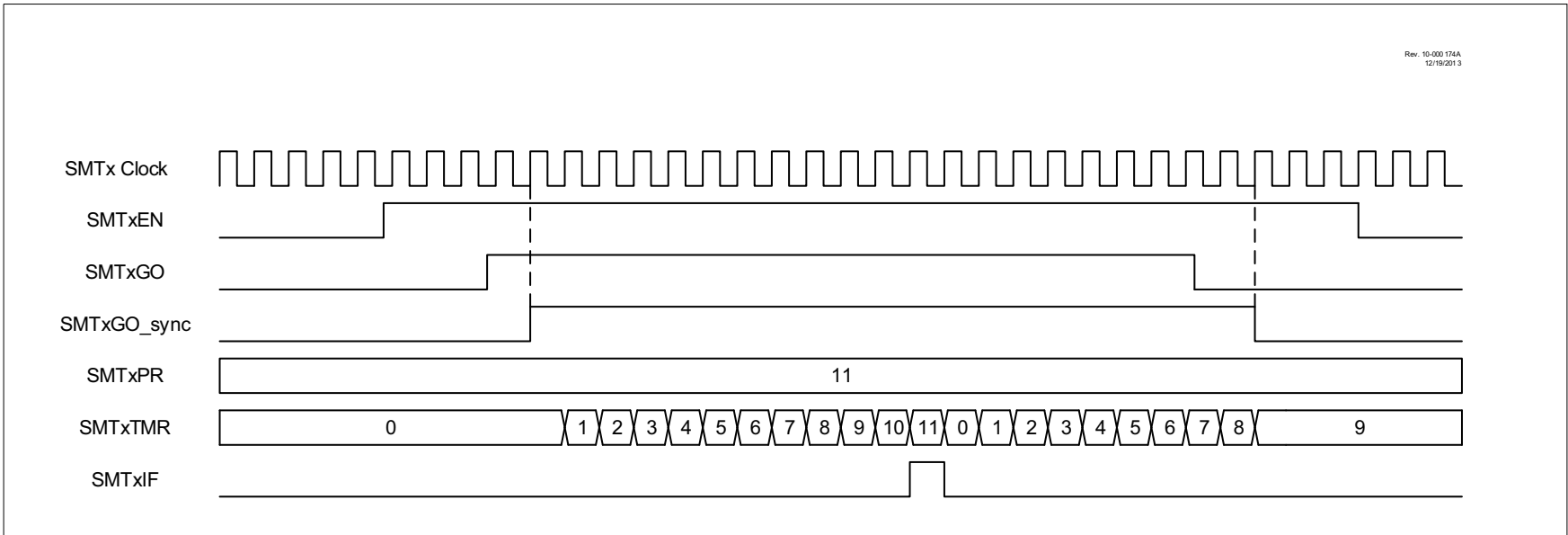
25.7.1 TIMER MODE

Timer mode is the simplest mode of operation where the SMTxTMR is used as a 16/24-bit timer. No data acquisition takes place in this mode. The timer increments as long as the SMTxGO bit has been set by software. No SMT window or SMT signal events affect the SMTxGO bit. Everything is synchronized to the SMT clock source. When the timer experiences a period match (SMTxTMR = SMTxPR), SMTxTMR is reset and the period match interrupt trips. See [Figure 25-3](#).

TABLE 25-3: MODES OF OPERATION

| MODE | Mode of Operation | Synchronous Operation | Reference |
|-------------|-----------------------------------|-----------------------|--|
| 0000 | Timer | Yes | Section 25.7.1 “Timer Mode” |
| 0001 | Gated Timer | Yes | Section 25.7.2 “Gated Timer Mode” |
| 0010 | Period and Duty Cycle Acquisition | Yes | Section 25.7.3 “Period and Duty-Cycle Mode” |
| 0011 | High and Low Time Measurement | Yes | Section 25.7.4 “High and Low Measure Mode” |
| 0100 | Windowed Measurement | Yes | Section 25.7.5 “Windowed Measure Mode” |
| 0101 | Gated Windowed Measurement | Yes | Section 25.7.6 “Gated Window Measure Mode” |
| 0110 | Time of Flight | Yes | Section 25.7.7 “Time of Flight Measure Mode” |
| 0111 | Capture | Yes | Section 25.7.8 “Capture Mode” |
| 1000 | Counter | No | Section 25.7.9 “Counter Mode” |
| 1001 | Gated Counter | No | Section 25.7.10 “Gated Counter Mode” |
| 1010 | Windowed Counter | No | Section 25.7.11 “Windowed Counter Mode” |
| 1011 - 1111 | Reserved | — | — |

FIGURE 25-3: TIMER MODE TIMING DIAGRAM



25.7.2 GATED TIMER MODE

Gated Timer mode uses the SMTSIGx input to control whether or not the SMTxTMR will increment. Upon a falling edge of the external signal, the SMTxCPW register will update to the current value of the SMTxTMR. Example waveforms for both repeated and single acquisitions are provided in [Figure 25-4](#) and [Figure 25-5](#).

FIGURE 25-4: GATED TIMER MODE REPEAT ACQUISITION TIMING DIAGRAM

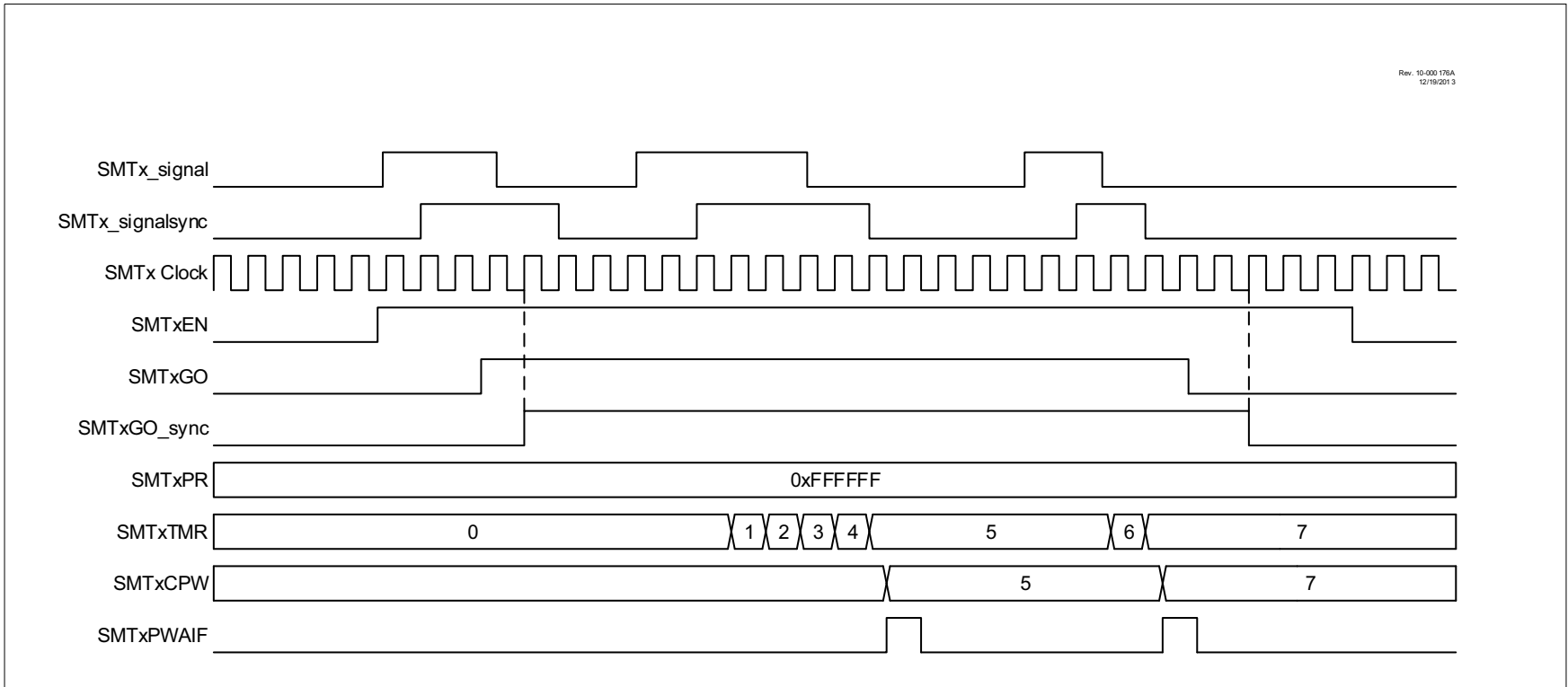
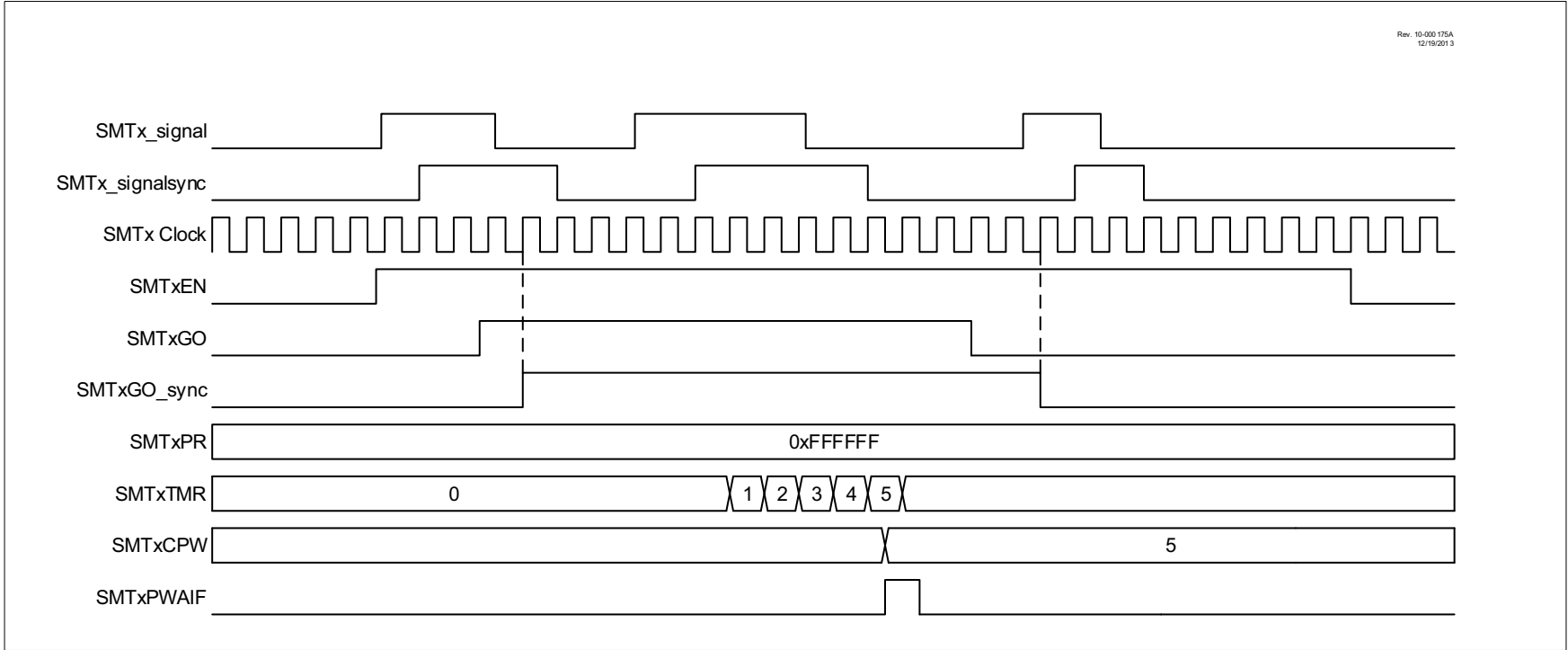


FIGURE 25-5: GATED TIMER MODE SINGLE ACQUISITION TIMING DIAGRAM



25.7.3 PERIOD AND DUTY-CYCLE MODE

In Duty-Cycle mode, either the duty cycle or period (depending on polarity) of the SMTx_signal can be acquired relative to the SMT clock. The CPW register is updated on a falling edge of the signal, and the CPR register is updated on a rising edge of the signal, along with the SMTxTMR resetting to 0x0001. In addition, the SMTxGO bit is reset on a rising edge when the SMT is in Single Acquisition mode. See [Figure 25-6](#) and [Figure 25-7](#).

FIGURE 25-6: PERIOD AND DUTY-CYCLE REPEAT ACQUISITION MODE TIMING DIAGRAM

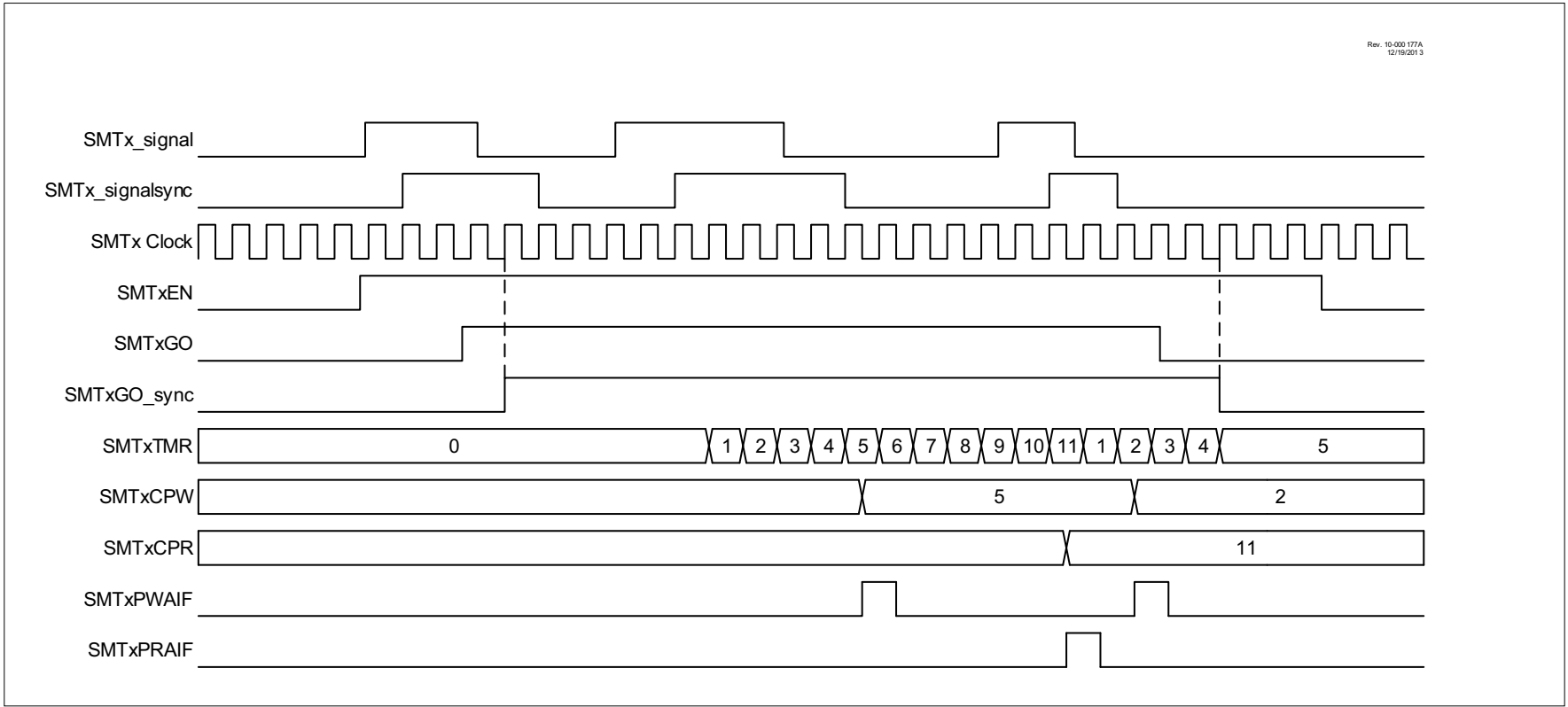
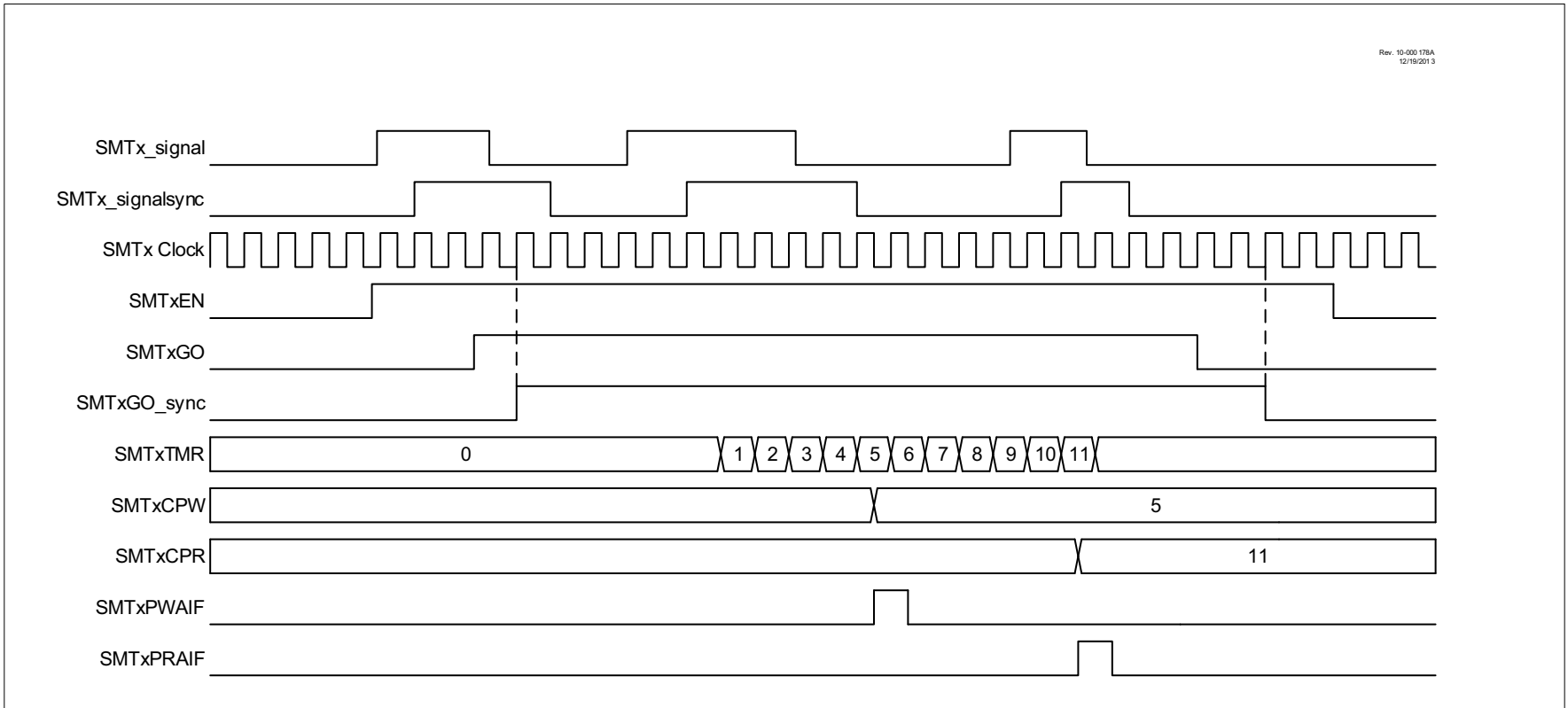


FIGURE 25-7: PERIOD AND DUTY-CYCLE SINGLE ACQUISITION TIMING DIAGRAM



25.7.4 HIGH AND LOW MEASURE MODE

This mode measures the high and low pulse time of the SMTSIGx relative to the SMT clock. It begins incrementing the SMTxTMR on a rising edge on the SMTSIGx input, then updates the SMTxCPW register with the value and resets the SMTxTMR on a falling edge, starting to increment again. Upon observing another rising edge, it updates the SMTxCPR register with its current value and once again resets the SMTxTMR value and begins incrementing again. See [Figure 25-8](#) and [Figure 25-9](#).

FIGURE 25-8: HIGH AND LOW MEASURE MODE REPEAT ACQUISITION TIMING DIAGRAM

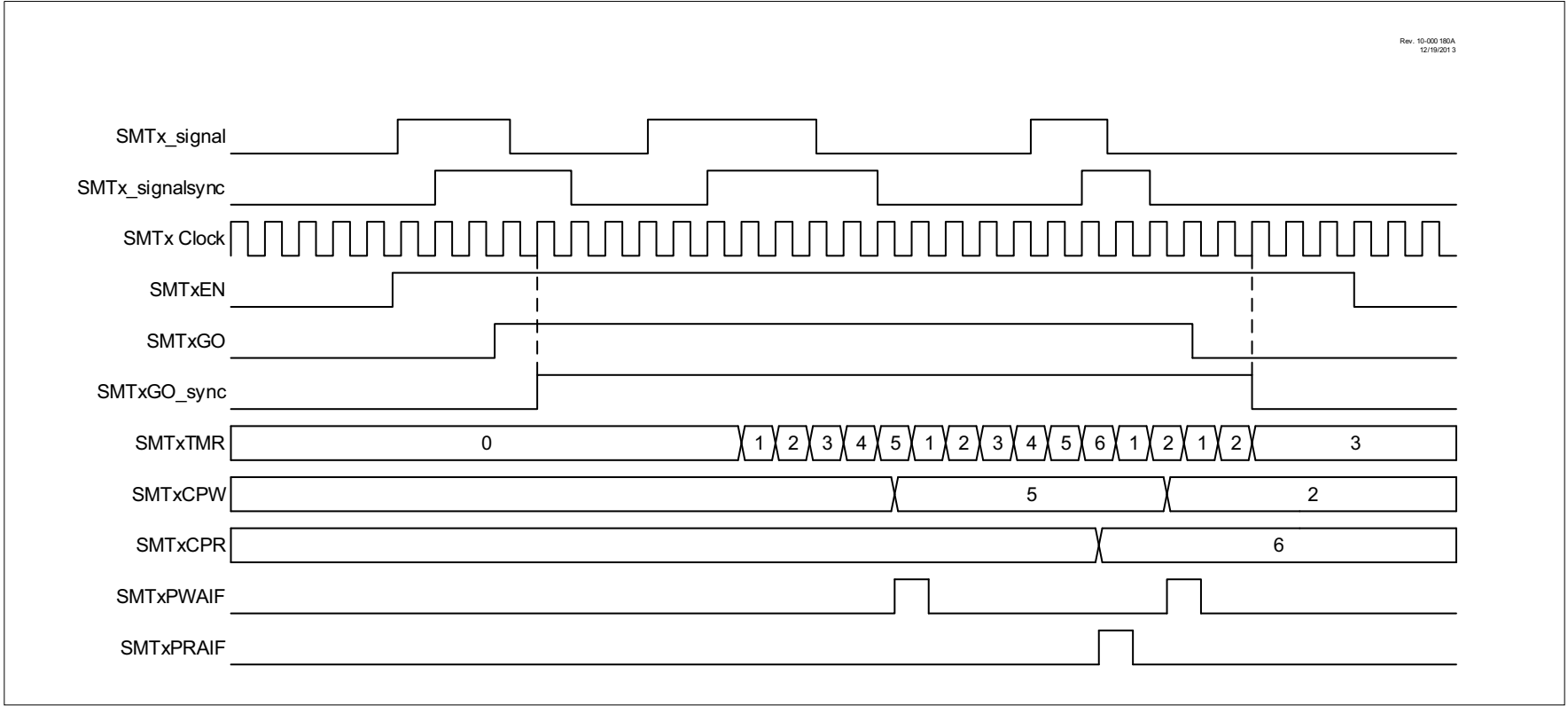
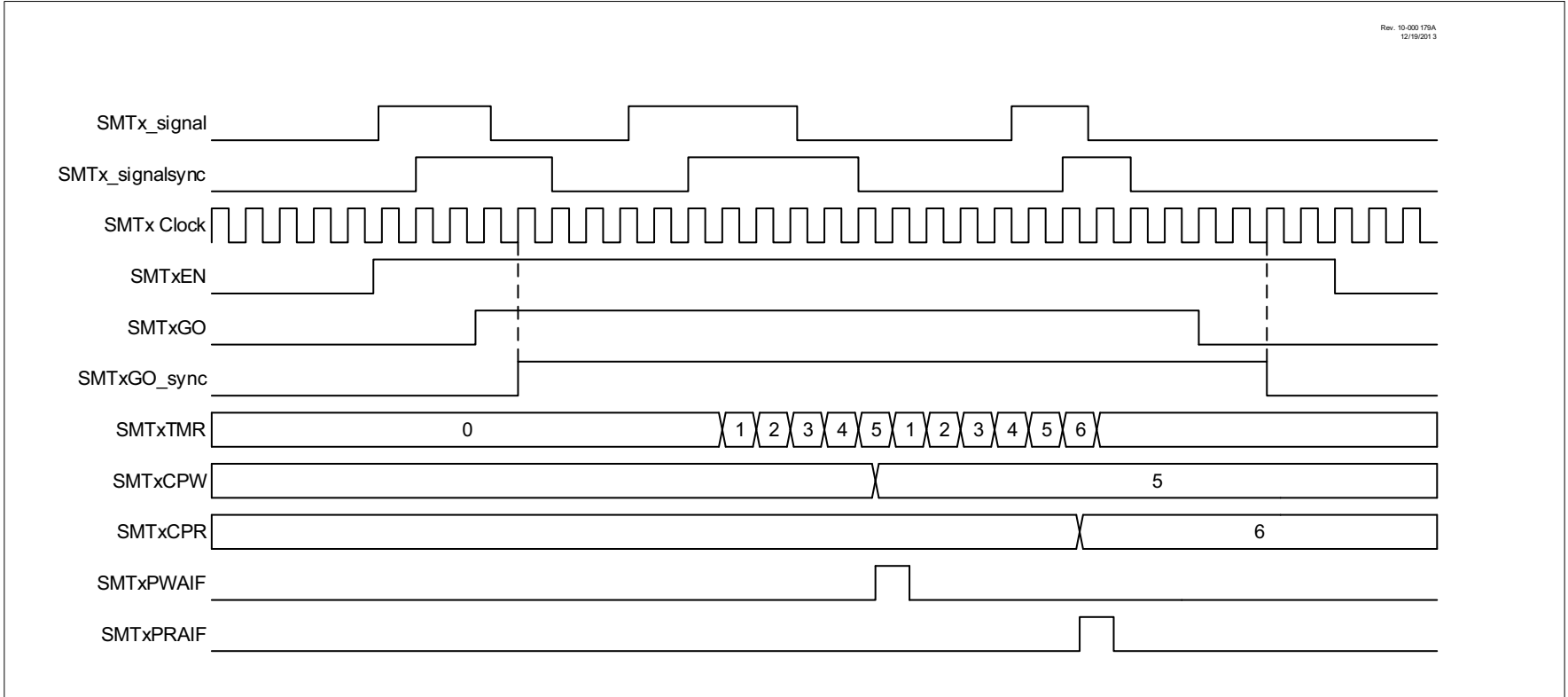


FIGURE 25-9: HIGH AND LOW MEASURE MODE SINGLE ACQUISITION TIMING DIAGRAM



25.7.5 WINDOWED MEASURE MODE

This mode measures the window duration of the SMTWINx input of the SMT. It begins incrementing the timer on a rising edge of the SMTWINx input and updates the SMTxCPR register with the value of the timer and resets the timer on a second rising edge. See [Figure 25-10](#) and [Figure 25-11](#).

FIGURE 25-10: WINDOWED MEASURE MODE REPEAT ACQUISITION TIMING DIAGRAM

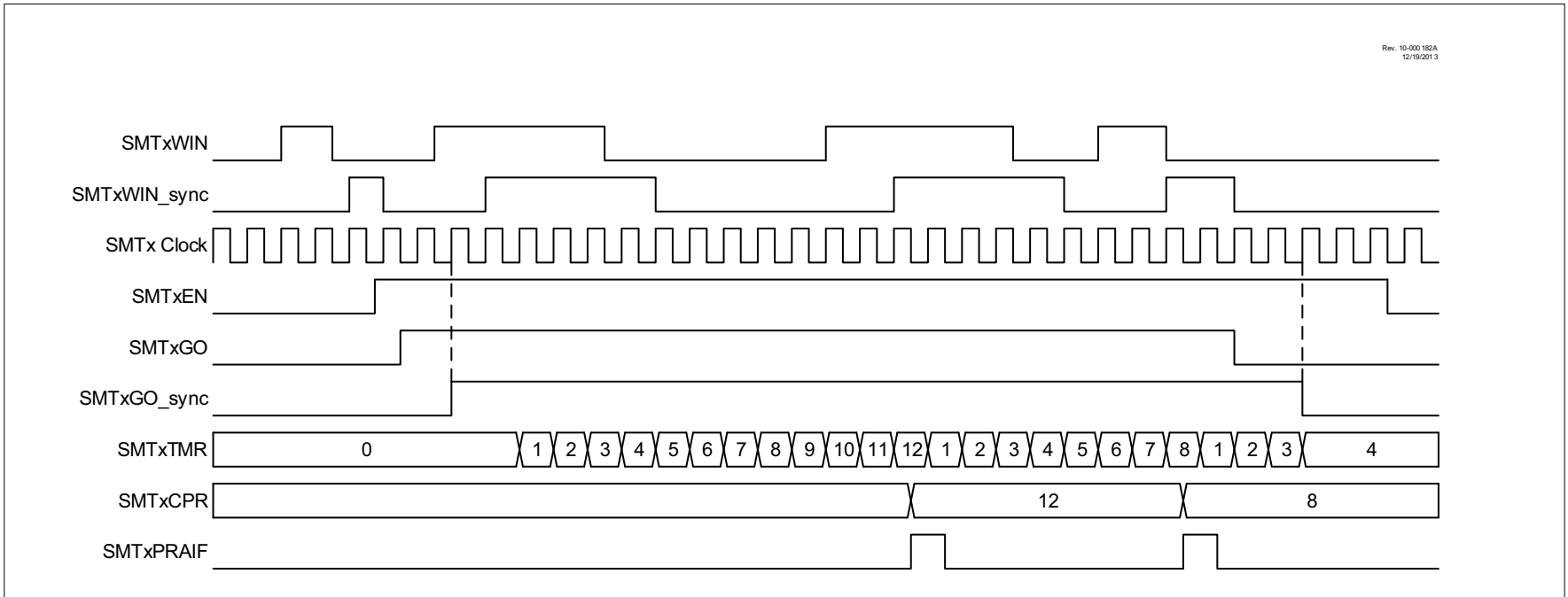
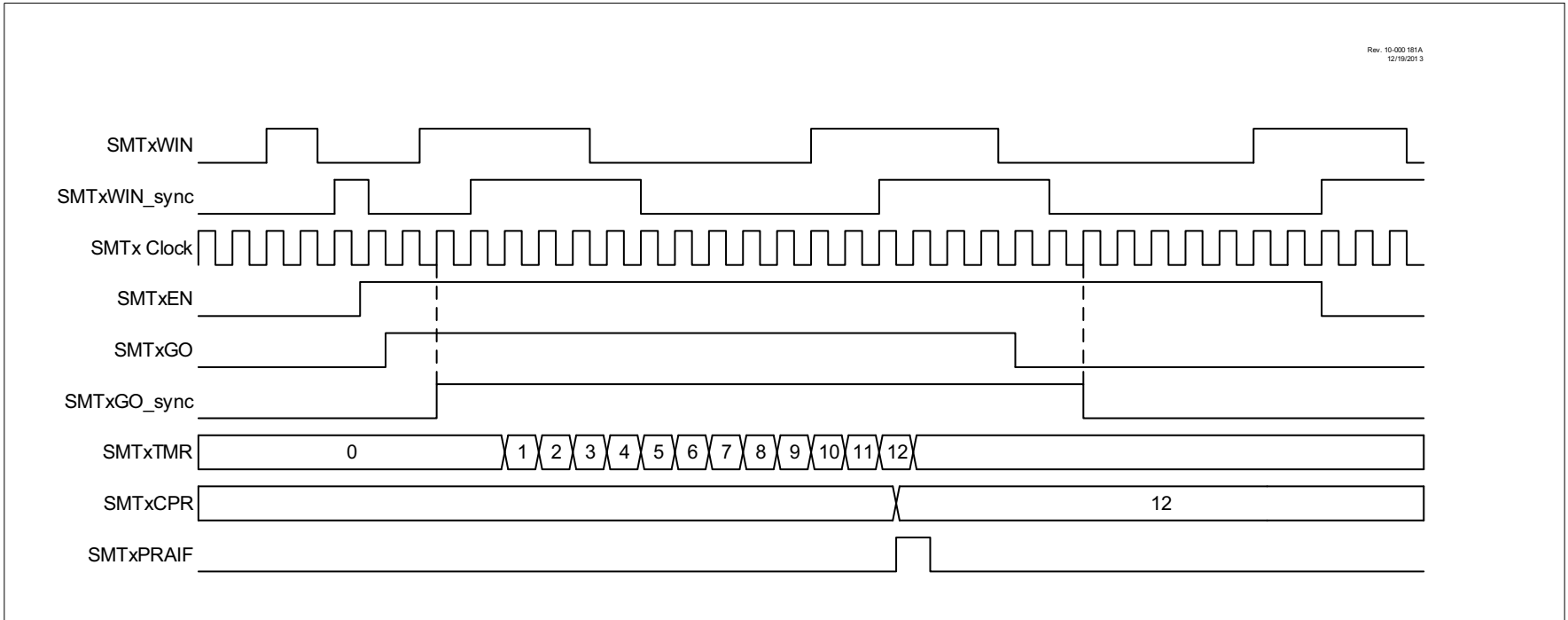


FIGURE 25-11: WINDOWED MEASURE MODE SINGLE ACQUISITION TIMING DIAGRAM



25.7.6 GATED WINDOW MEASURE MODE

This mode measures the duty cycle of the SMTx_signal input over a known input window. It does so by incrementing the timer on each pulse of the clock signal while the SMTx_signal input is high, updating the SMTxCPR register and resetting the timer on every rising edge of the SMTWINx input after the first. See [Figure 25-12](#) and [Figure 25-13](#).

FIGURE 25-12: GATED WINDOWED MEASURE MODE REPEAT ACQUISITION TIMING DIAGRAM

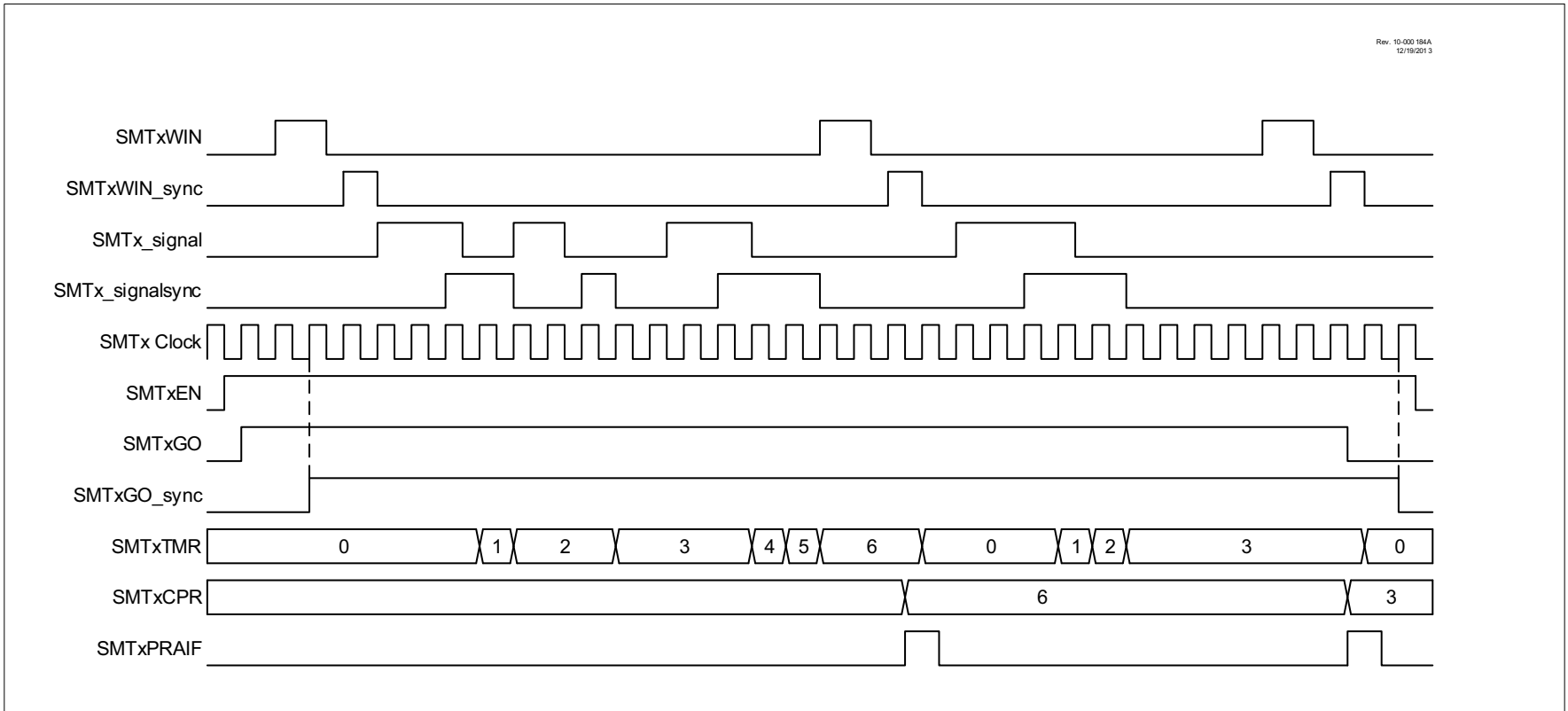
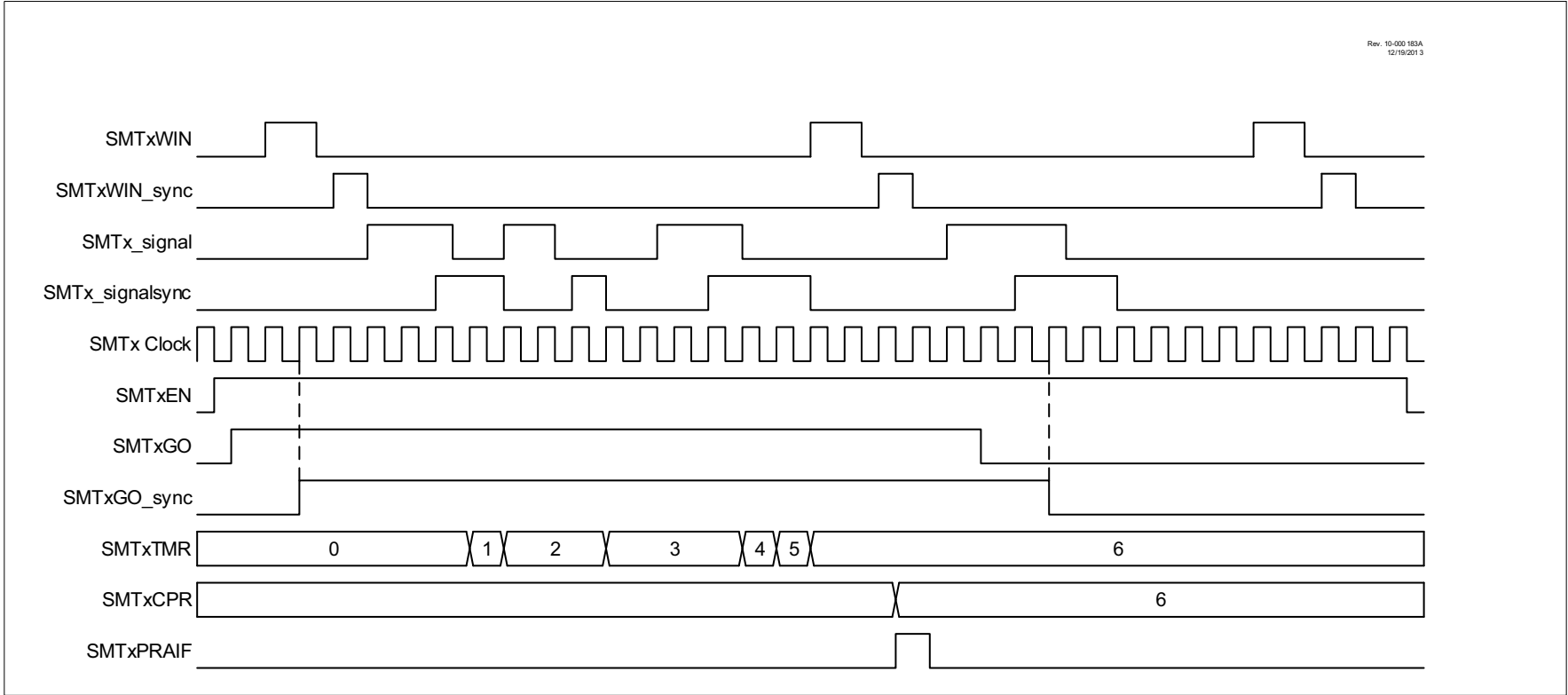


FIGURE 25-13: GATED WINDOWED MEASURE MODE SINGLE ACQUISITION TIMING DIAGRAMS



25.7.7 TIME OF FLIGHT MEASURE MODE

This mode measures the time interval between a rising edge on the SMTWINx input and a rising edge on the SMTx_signal input, beginning to increment the timer upon observing a rising edge on the SMTWINx input, while updating the SMTxCPR register and resetting the timer upon observing a rising edge on the SMTx_signal input. In the event of two SMTWINx rising edges without an SMTx_signal rising edge, it will update the SMTxCPW register with the current value of the timer and reset the timer value. See [Figure 25-14](#) and [Figure 25-15](#).

FIGURE 25-14: TIME OF FLIGHT MODE REPEAT ACQUISITION TIMING DIAGRAM

Rev. 10-000186A
4/22/2016

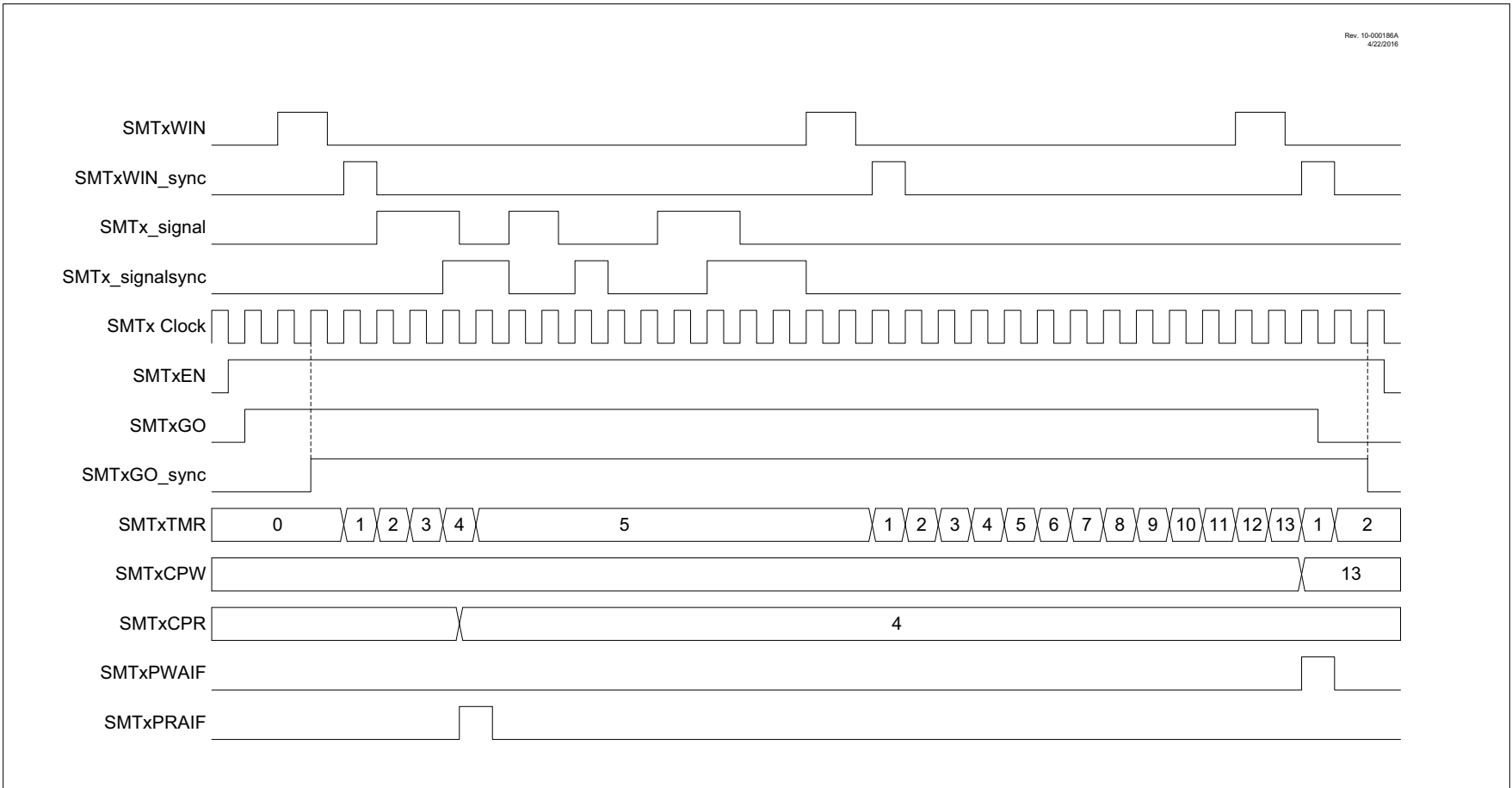
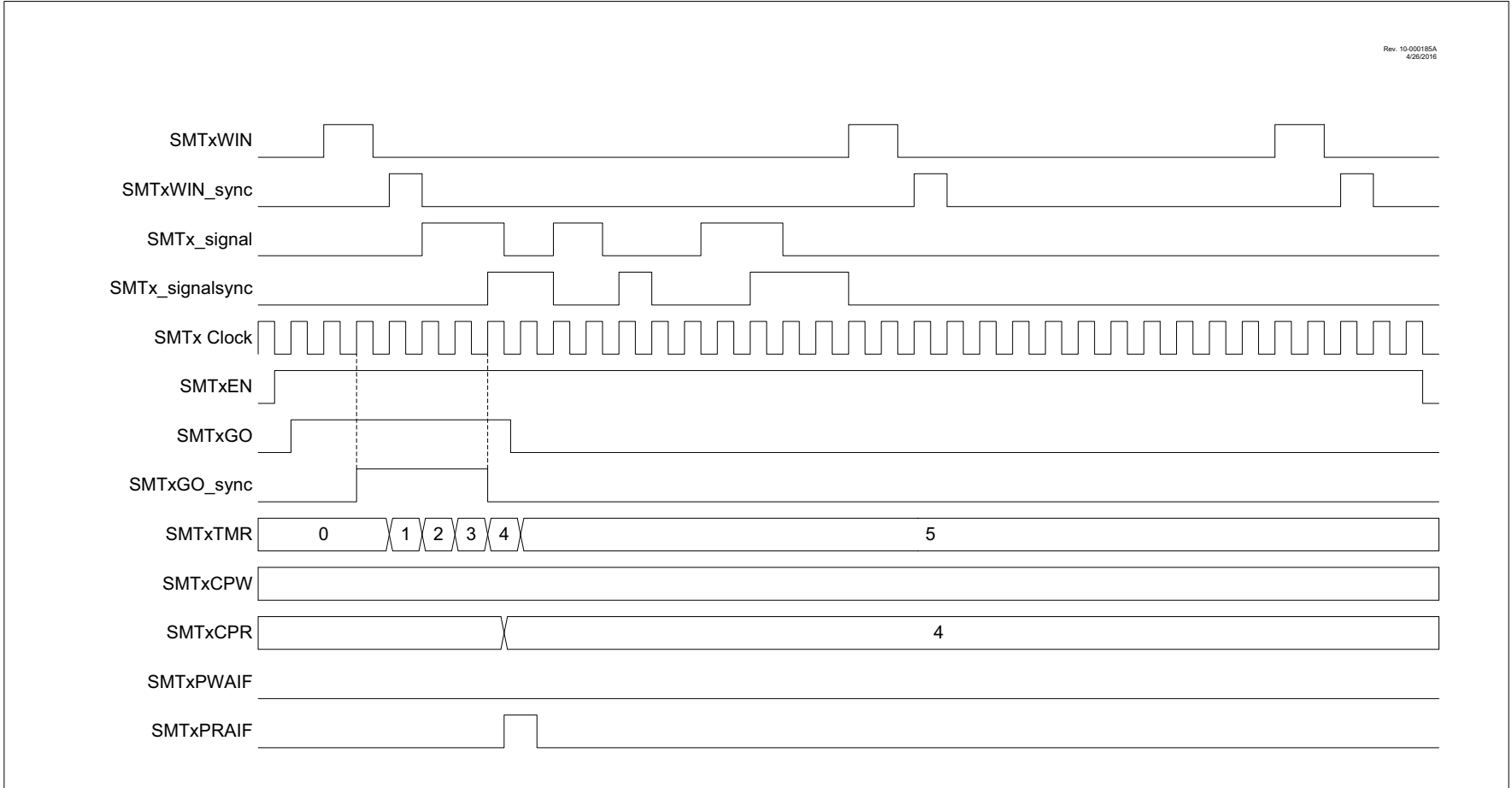


FIGURE 25-15: TIME OF FLIGHT MODE SINGLE ACQUISITION TIMING DIAGRAM



25.7.8 CAPTURE MODE

This mode captures the Timer value based on a rising or falling edge on the SMTWINx input and triggers an interrupt. This mimics the capture feature of a CCP module. The timer begins incrementing upon the SMTxGO bit being set, and updates the value of the SMTxCPR register on each rising edge of SMTWINx, and updates the value of the CPW register on each falling edge of the SMTWINx. The timer is not reset by any hardware conditions in this mode and must be reset by software, if desired. See [Figure 25-16](#) and [Figure 25-17](#).

FIGURE 25-16: CAPTURE MODE REPEAT ACQUISITION TIMING DIAGRAM

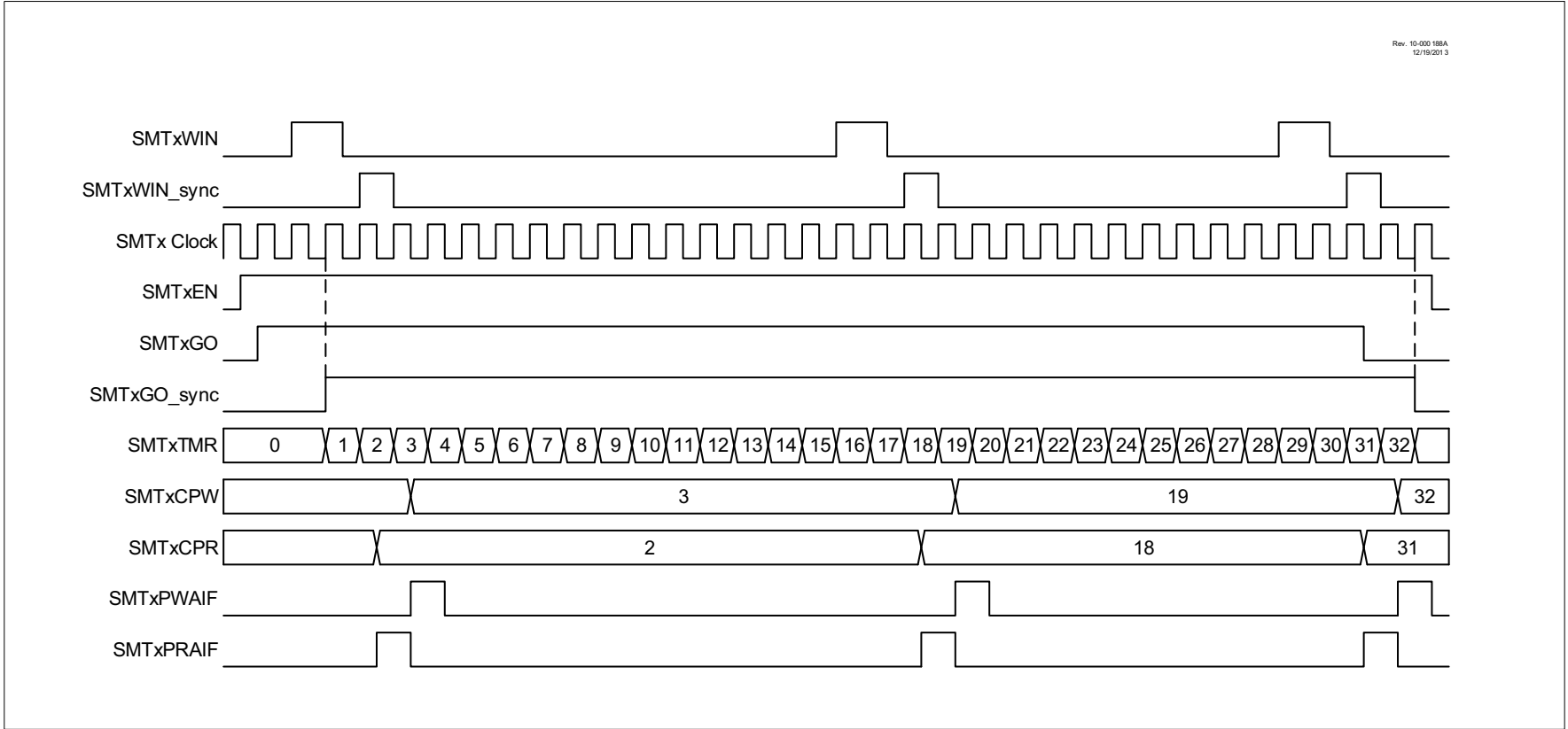
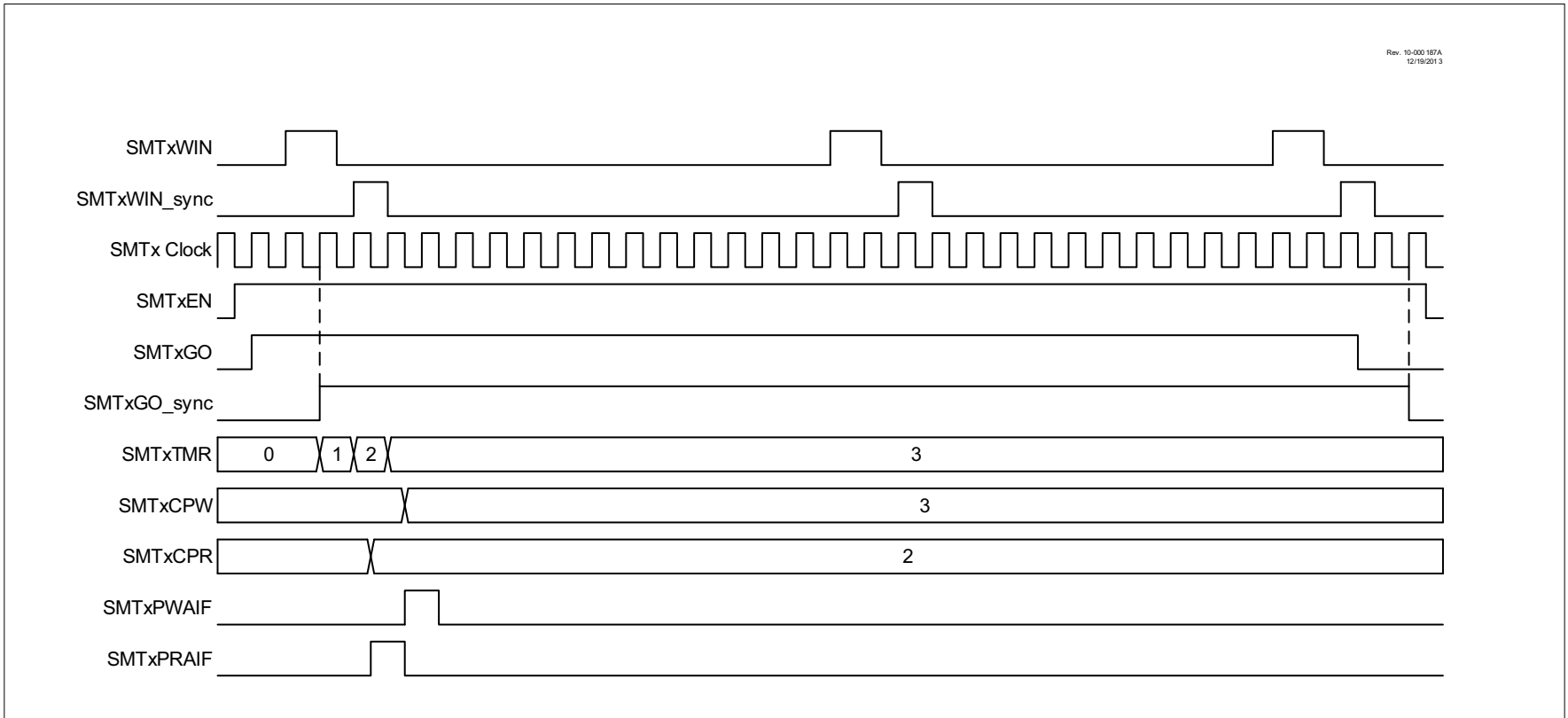


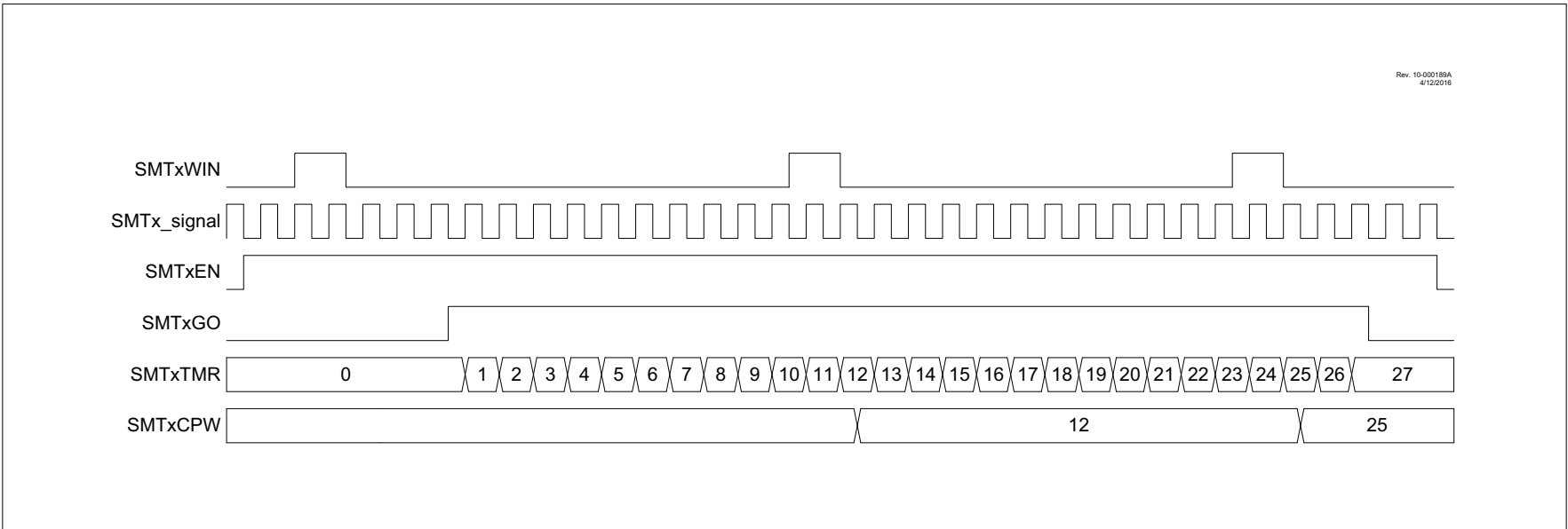
FIGURE 25-17: CAPTURE MODE SINGLE ACQUISITION TIMING DIAGRAM



25.7.9 COUNTER MODE

This mode increments the timer on each pulse of the SMTx_signal input. This mode is asynchronous to the SMT clock and uses the SMTx_signal as a time source. The SMTxCPW register will be updated with the current SMTxTMR value on the falling edge of the SMTxWIN input. See [Figure 25-18](#).

FIGURE 25-18: COUNTER MODE TIMING DIAGRAM



25.7.10 GATED COUNTER MODE

This mode counts pulses on the SMTx_signal input, gated by the SMTxWIN input. It begins incrementing the timer upon seeing a rising edge of the SMTxWIN input and updates the SMTxCPW register upon a falling edge on the SMTxWIN input. See [Figure 25-19](#) and [Figure 25-20](#).

FIGURE 25-19: GATED COUNTER MODE REPEAT ACQUISITION TIMING DIAGRAM

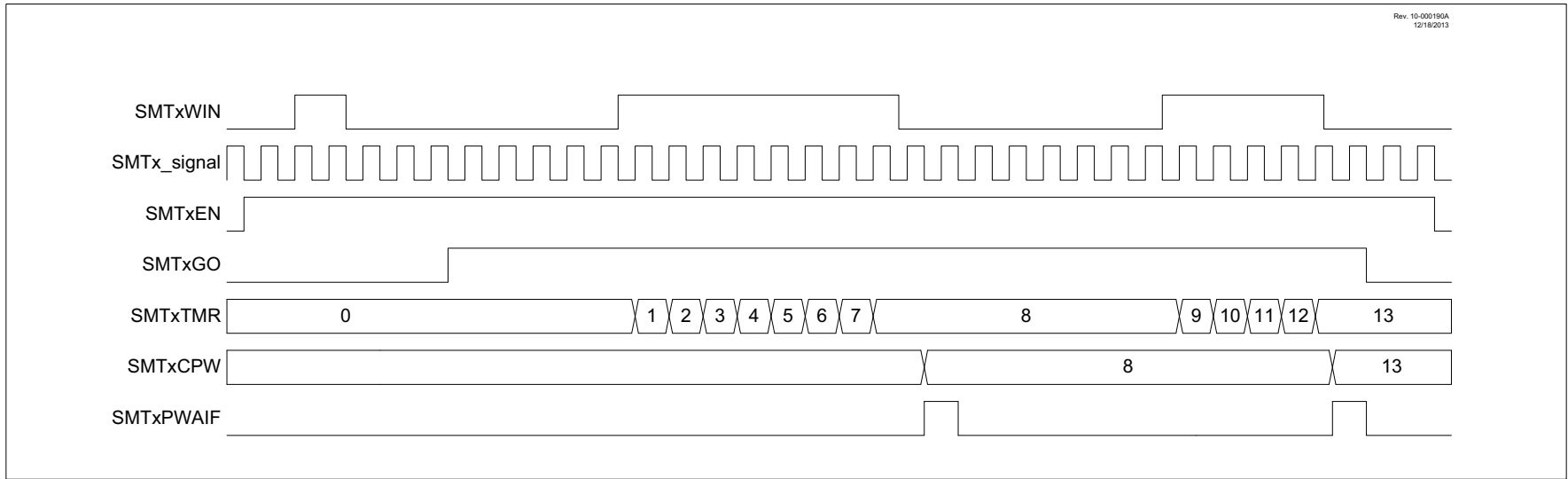
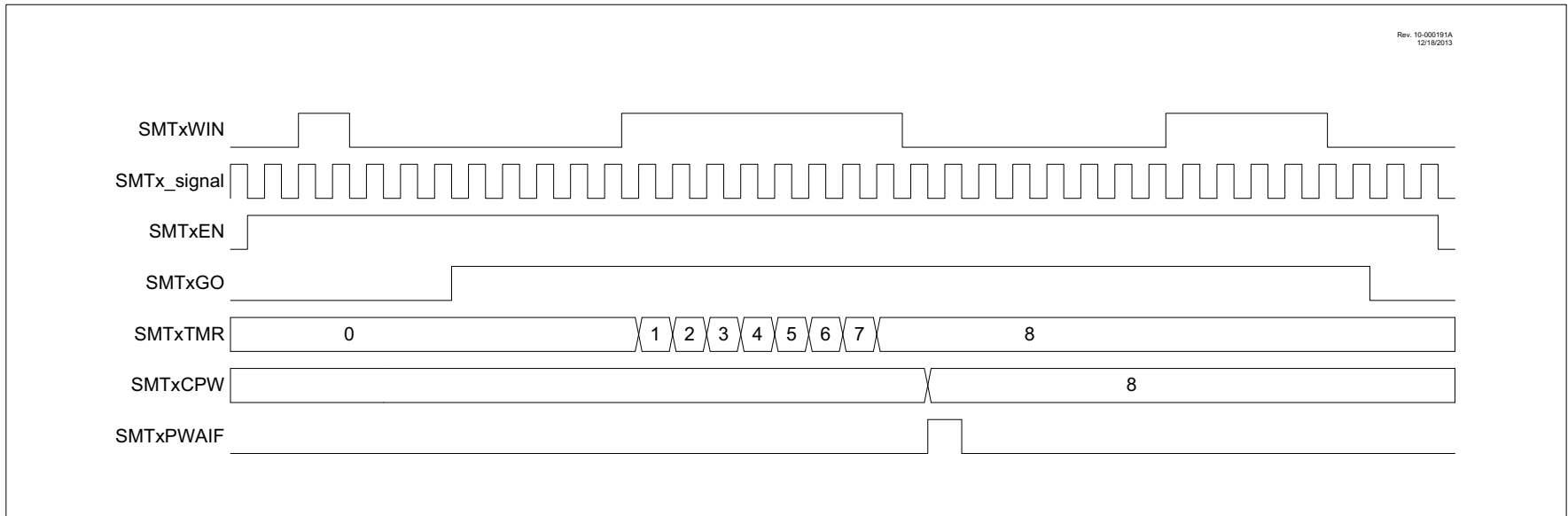


FIGURE 25-20: GATED COUNTER MODE SINGLE ACQUISITION TIMING DIAGRAM



25.7.11 WINDOWED COUNTER MODE

This mode counts pulses on the SMTx_{signal} input, within a window dictated by the SMTxWIN input. It begins counting upon seeing a rising edge of the SMTxWIN input, updates the SMTxCPW register on a falling edge of the SMTxWIN input, and updates the SMTxCPR register on each rising edge of the SMTxWIN input beyond the first. See [Figure 25-21](#) and [Figure 25-22](#).

FIGURE 25-21: WINDOWED COUNTER MODE REPEAT ACQUISITION TIMING DIAGRAM

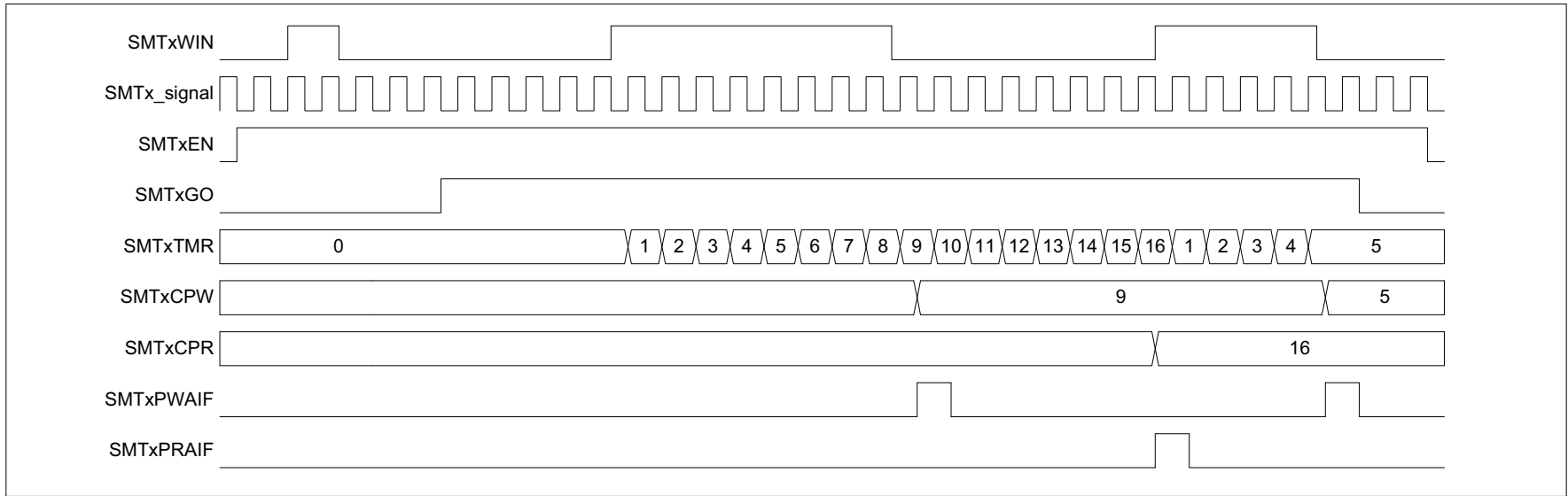
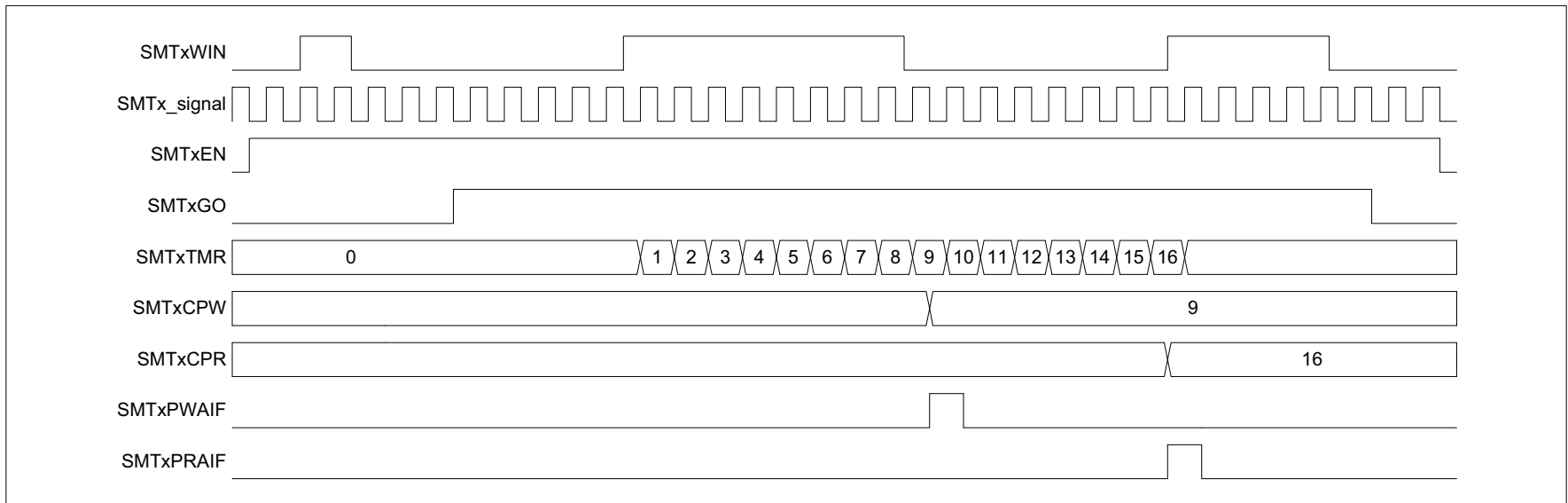


FIGURE 25-22: WINDOWED COUNTER MODE SINGLE ACQUISITION TIMING DIAGRAM



25.8 Interrupts

The SMT can trigger an interrupt under three different conditions:

- PW Acquisition Complete
- PR Acquisition Complete
- Counter Period Match

The interrupts are controlled by the PIR9 and PIE9 registers of the device.

25.8.1 PW AND PR ACQUISITION INTERRUPTS

The SMT can trigger interrupts whenever it updates the SMTxCPW and SMTxCPR registers, the circumstances for which are dependent on the SMT mode, and are discussed in each mode's specific section. The SMTxCPW interrupt is controlled by SMTxPWAIF and SMTxPWAIE bits in registers PIR9 and PIE9, respectively. The SMTxCPR interrupt is controlled by the SMTxPRAIF and SMTxPRAIE bits, also located in registers PIR9 and PIE9, respectively.

In synchronous SMT modes, the interrupt trigger is synchronized to the SMTxCLK. In Asynchronous modes, the interrupt trigger is asynchronous. In either mode, once triggered, the interrupt will be synchronized to the CPU clock.

25.8.2 COUNTER PERIOD MATCH INTERRUPT

As described in [Section 25.2.2 “Period Match interrupt”](#), the SMT will also interrupt upon SMTxTMR, matching SMTxPR with its period match limit functionality described in [Section 25.4 “Halt Operation”](#). The period match interrupt is controlled by SMTxIF and SMTxIE.

26.0 DATA SIGNAL MODULATOR (DSM) MODULE

The Data Signal Modulator (DSM) is a peripheral which 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 MDOUT 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 carrier high signal with the modulator signal. When the modulator signal is in a logic low state, the DSM mixes the carrier low signal with the modulator signal.

Using this method, the DSM can generate the following types of Key Modulation schemes:

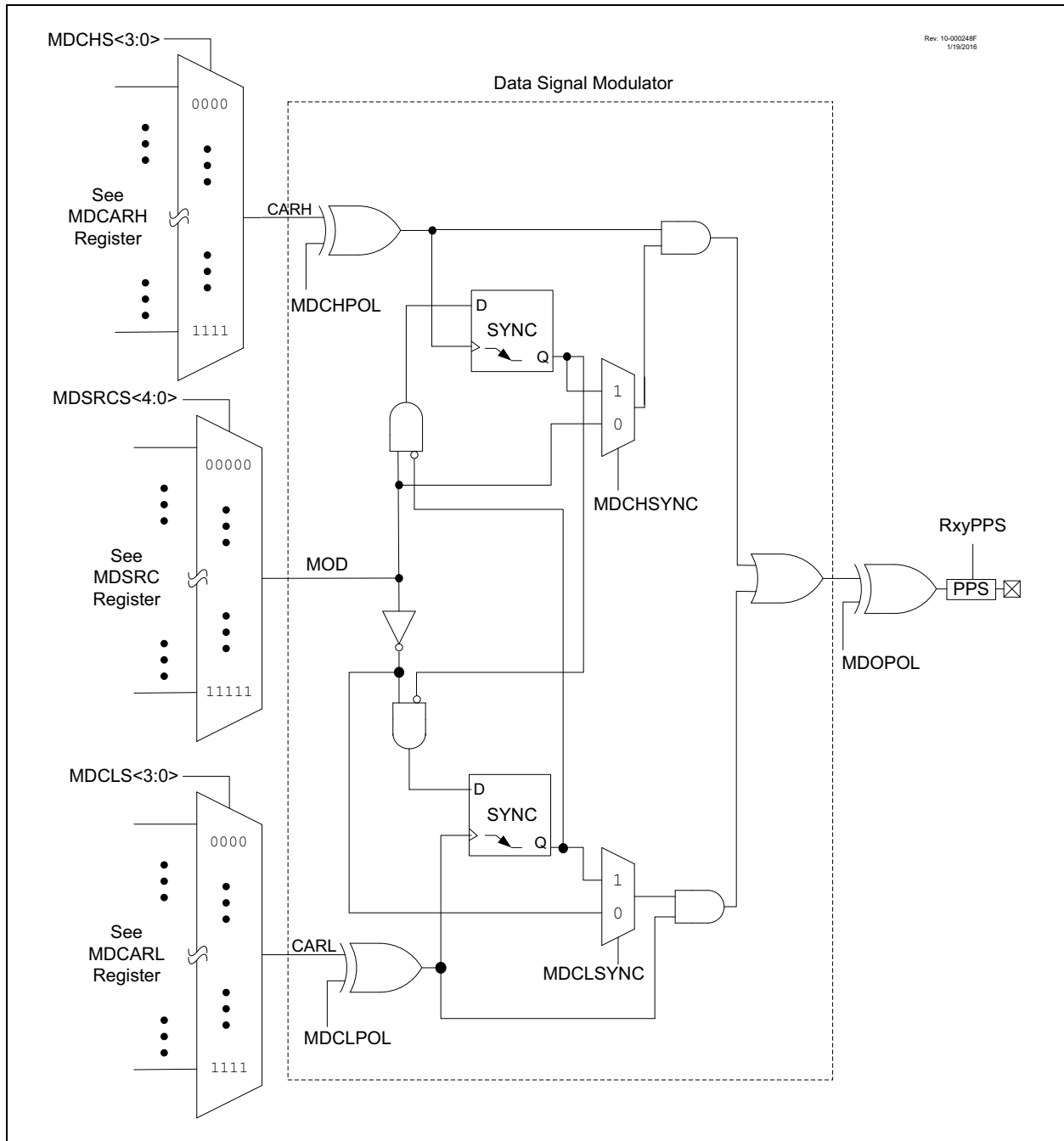
- Frequency-Shift Keying (FSK)
- Phase-Shift Keying (PSK)
- On-Off Keying (OOK)

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

[Figure 26-1](#) shows a Simplified Block Diagram of the Data Signal Modulator peripheral.

FIGURE 26-1: SIMPLIFIED BLOCK DIAGRAM OF THE DATA SIGNAL MODULATOR



26.1 Register Definitions: Modulation Control

Long bit name prefixes for the Modulation peripheral is shown in Table 26-1. Refer to Section 1.4.2.2 “Long Bit Names” for more information.

TABLE 26-1:

| Peripheral | Bit Name Prefix |
|------------|-----------------|
| MD | MD |

REGISTER 26-1: MDCON0: MODULATION CONTROL REGISTER 0

| R/W-0/0 | U-0 | R/W-0/0 | R/W-0/0 | U-0 | U-0 | U-0 | R/W-0/0 |
|---------|-----|---------|---------|-----|-----|-----|---------|
| EN | — | OUT | OPOL | — | — | — | BIT |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

- bit 7 **EN:** Modulator Module Enable bit
 1 = Modulator module is enabled and mixing input signals
 0 = Modulator module is disabled and has no output
- bit 6 **Unimplemented:** Read as '0'
- bit 5 **OUT:** Modulator Output bit
 Displays the current output value of the Modulator module.⁽¹⁾
- bit 4 **OPOL:** Modulator Output Polarity Select bit
 1 = Modulator output signal is inverted; idle high output
 0 = Modulator output signal is not inverted; idle low output
- bit 3-1 **Unimplemented:** Read as '0'
- bit 0 **BIT:** Allows software to manually set modulation source input to module⁽²⁾

- Note 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 MDSRC register for this operation.

REGISTER 26-2: MDCON1: MODULATION CONTROL REGISTER 1

| | | | | | | | |
|-------|-----|---------|---------|-----|-----|---------|---------|
| U-0 | U-0 | R/W-0/0 | R/W-0/0 | U-0 | U-0 | R/W-0/0 | R/W-0/0 |
| — | — | CHPOL | CHSYNC | — | — | CLPOL | CLSYNC |
| bit 7 | | | | | | bit 0 | |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

- bit 7-6 **Unimplemented:** Read as '0'
- bit 5 **CHPOL:** Modulator High Carrier Polarity Select bit
 - 1 = Selected high carrier signal is inverted
 - 0 = Selected high carrier signal is not inverted
- bit 4 **CHSYNC:** Modulator High Carrier Synchronization Enable bit
 - 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 3-2 **Unimplemented:** Read as '0'
- bit 1 **CLPOL:** Modulator Low Carrier Polarity Select bit
 - 1 = Selected low carrier signal is inverted
 - 0 = Selected low carrier signal is not inverted
- bit 0 **CLSYNC:** Modulator Low Carrier Synchronization Enable bit
 - 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.

REGISTER 26-3: MDCARH: MODULATION HIGH CARRIER CONTROL REGISTER

| | | | | | | | |
|-------|-----|-----|-----|----------|---------|---------|---------|
| U-0 | U-0 | U-0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| — | — | — | — | CHS<3:0> | | | |
| bit 7 | | | | bit 0 | | | |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-4 **Unimplemented:** Read as '0'
bit 3-0 **CHS<3:0>:** Modulator Carrier High Selection bits
See [Table 26-2](#) for signal list

REGISTER 26-4: MDCARL: MODULATION LOW CARRIER CONTROL REGISTER

| | | | | | | | |
|-------|-----|-----|-----|----------|---------|---------|---------|
| U-0 | U-0 | U-0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| — | — | — | — | CLS<3:0> | | | |
| bit 7 | | | | bit 0 | | | |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-4 **Unimplemented:** Read as '0'
bit 3-0 **CLS<3:0>:** Modulator Carrier Low Input Selection bits
See [Table 26-2](#) for signal list

TABLE 26-2: MDCARH/MDCARL SELECTION MUX CONNECTIONS

| MDCARH | | | MDCARL | | |
|------------|----|---------------------------|------------|----|---------------------------|
| MDCHS<3:0> | | Connection | MDCLS<3:0> | | Connection |
| 1111-1011 | | Reserved | 1111-1011 | | Reserved |
| 1010 | 10 | PWM7 OUT | 1010 | 10 | PWM7 OUT |
| 1001 | 9 | PWM6OUT | 1001 | 9 | PWM6OUT |
| 1000 | 8 | CCP5 OUT | 1000 | 8 | CCP5 OUT |
| 0111 | 7 | CCP4 OUT | 0111 | 7 | CCP4 OUT |
| 0110 | 6 | CCP3 OUT | 0110 | 6 | CCP3 OUT |
| 0101 | 5 | CCP2 OUT | 0101 | 5 | CCP2 OUT |
| 0100 | 4 | CCP1 OUT | 0100 | 4 | CCP1 OUT |
| 0011 | 3 | CLKREF output | 0011 | 3 | CLKREF output |
| 0010 | 2 | HFINTOSC | 0010 | 2 | HFINTOSC |
| 0001 | 1 | FOSC (system clock) | 0001 | 1 | FOSC (system clock) |
| 0000 | 0 | Pin selected by MDCARHPPS | 0000 | 0 | Pin selected by MDCARHPPS |

REGISTER 26-5: MDSRC: MODULATION SOURCE CONTROL REGISTER

| | | | | | | | | |
|-------|-----|-----|-----------|---------|---------|---------|---------|-------|
| U-0 | U-0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | |
| — | — | — | SRCS<4:0> | | | | | |
| bit 7 | | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-5 **Unimplemented:** Read as '0'
bit 4-0 **SRCS<4:0>:** Modulator Source Selection bits
See [Table 26-3](#) for signal list

TABLE 26-3: MDSRC SELECTION MUX CONNECTIONS

| MDSRCS<4:0> | Connection |
|-------------|------------------------------|
| 11111-11000 | Reserved |
| 10111 | 23 MSSP2 – SDO |
| 10110 | 22 MSSP1 – SDO |
| 10101 | 21 EUSART5 TX (TX/CK output) |
| 10100 | 20 EUSART5 RX (DT output) |
| 10011 | 19 EUSART4 TX (TX/CK output) |
| 10010 | 18 EUSART4 RX (DT output) |
| 10001 | 17 EUSART3 TX (TX/CK output) |
| 10000 | 16 EUSART3 RX (DT output) |
| 01111 | 15 EUSART2 TX (TX/CK output) |
| 01110 | 14 EUSART2 RX (DT output) |
| 01101 | 13 EUSART1 TX (TX/CK output) |
| 01100 | 12 EUSART1 RX (DT output) |
| 01011 | 11 CMP3 OUT |
| 01010 | 10 CMP2 OUT |
| 01001 | 9 CMP1 OUT |
| 01000 | 8 PWM7 OUT |
| 00111 | 7 PWM6 OUT |
| 00110 | 6 CCP5 OUT |
| 00101 | 5 CCP4 OUT |
| 00100 | 4 CCP3 OUT |
| 00011 | 3 CCP2 OUT |
| 00010 | 2 CCP1 OUT |
| 00001 | 1 MDBIT |
| 00000 | 0 Pin selected by MDSRCPPS |

TABLE 26-4: SUMMARY OF REGISTERS ASSOCIATED WITH DATA SIGNAL MODULATOR MODE

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on Page |
|-----------|-------|-------|--------------|-----------|----------|--------|--------|--------|------------------|
| MDCON0 | EN | — | OUT | OPOL | — | — | — | BIT | 376 |
| MDCON1 | — | — | CHPOL | CHSYNC | — | — | CLPOL | CLSYNC | 377 |
| MDCARH | — | — | — | — | CHS<3:0> | | | | 378 |
| MDCARL | — | — | — | — | CLS<3:0> | | | | 378 |
| MDSRC | — | — | — | SRCS<4:0> | | | | 379 | |
| MDCARLPPS | — | — | CARLPPS<5:0> | | | | 224 | | |
| MDCARHPPS | — | — | CARHPPS<5:0> | | | | 224 | | |
| MDSRCPPS | — | — | SRCPPS<5:0> | | | | 224 | | |
| RxyPPS | — | — | RxyPPS<5:0> | | | | 227 | | |
| PMD2 | — | — | CWGMMD | — | DSMMD | SMT2MD | SMT1MD | TMR8MD | 67 |

Legend: — = unimplemented, read as '0'. Shaded cells are not used in the Data Signal Modulator mode.

26.2 DSM Operation

The DSM module can be enabled by setting the MDEN bit in the MDCON0 register. Clearing the MDEN bit in the MDCON0 register, disables the DSM module output and switches the carrier high and carrier low signals to the default option of MDCARHPPS and MDCARLPPS, respectively. The modulator signal source is also switched to the MDBIT in the MDCON0 register. This not only assures that the DSM module is inactive, but that it is also consuming the least amount of current.

The values used to select the carrier high, carrier low, and modulator sources held by the Modulation Source, Modulation High Carrier, and Modulation Low Carrier control registers are not affected when the MDEN bit is cleared and the DSM module is disabled. The values inside these registers remain unchanged while the DSM is inactive. The sources for the carrier high, carrier low and modulator signals will once again be selected when the MDEN bit is set and the DSM module is again enabled and active.

The modulated output signal can be disabled without shutting down the DSM module. The DSM module will remain active and continue to mix signals, but the output value will not be sent to the DSM pin. During the time that the output is disabled, the DSM pin will remain low. The modulated output can be disabled by clearing the MDEN bit in the MDCON register.

26.3 Modulator Signal Sources

The modulator signal can be supplied from the following sources:

- External signal on pin selected by MDSRCPPS
- MDBIT bit in the MDCON0 register
- CCP1/2/3/4/5 Output
- PWM6/7 Output
- Comparator C1/C2/C3 Output
- EUSART1/2/3/4/5 RX Signal
- EUSART1/2/3/4/5 TX Signal
- MSSP1/2 SDO Signal (SPI Mode Only)

The modulator signal is selected by configuring the MDSRCS<4:0> bits in the MDSRC register.

26.4 Carrier Signal Sources

The carrier high signal and carrier low signal can be supplied from the following sources:

- External signal on pin selected by MDCARHPPS/ MDCARLPPS
- FOSC (system clock)
- HFINTOSC
- Reference Clock Module Signal
- CCP1/2/3/4/5 Output Signal
- PWM6/7 Output

The carrier high signal is selected by configuring the MDCHS<3:0> bits in the MDCARH register. The carrier low signal is selected by configuring the MDCLS<3:0> bits in the MDCARL register.

26.5 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 MDCHSYNC bit in the MDCON1 register. Synchronization for the carrier low signal is enabled by setting the MDCLSYNC bit in the MDCON1 register.

[Figure 26-2](#) through [Figure 26-6](#) show timing diagrams of using various synchronization methods.

FIGURE 26-2: On Off Keying (OOK) Synchronization

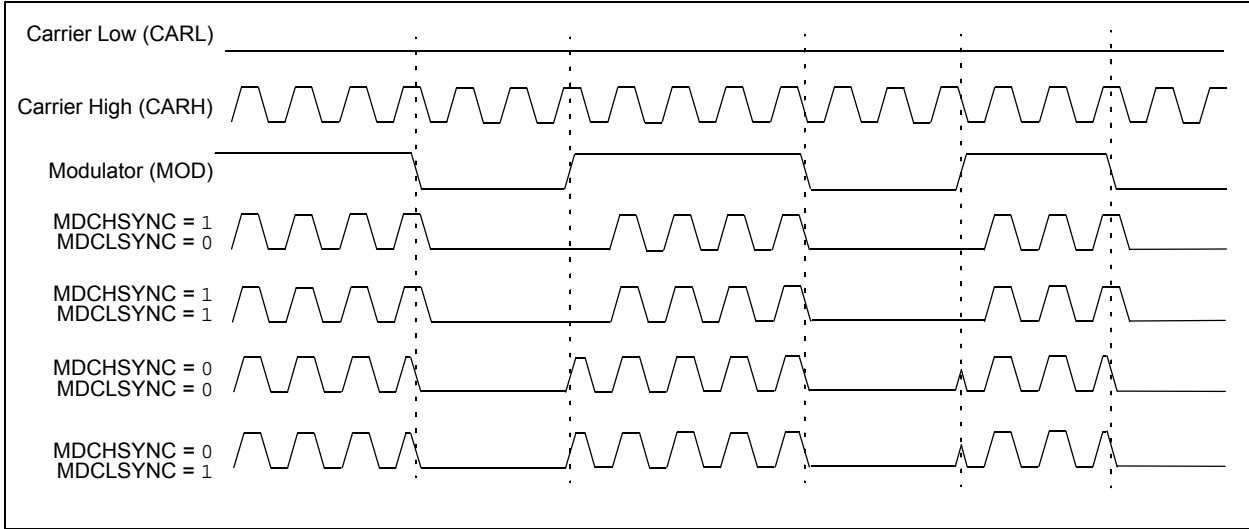


FIGURE 26-3: No Synchronization (MDSHSYNC = 0, MDCLSYNC = 0)

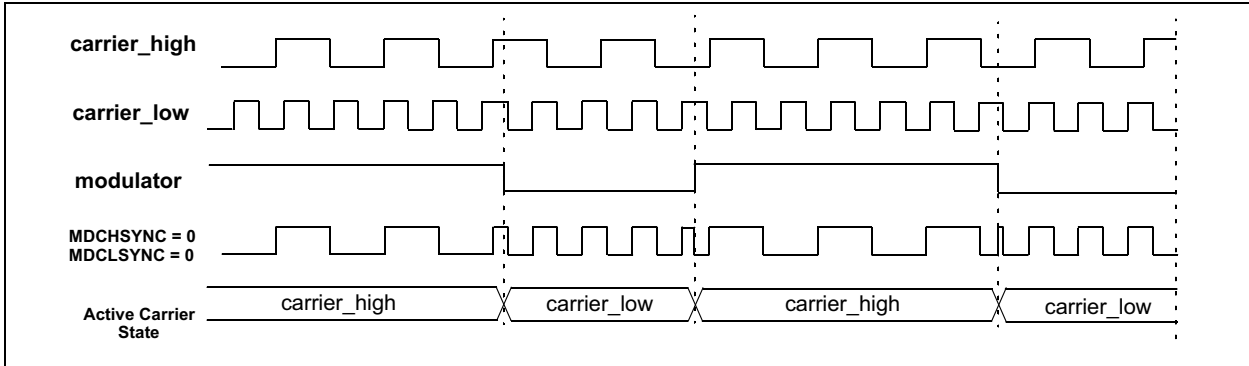


FIGURE 26-4: Carrier High Synchronization (MDSHSYNC = 1, MDCLSYNC = 0)

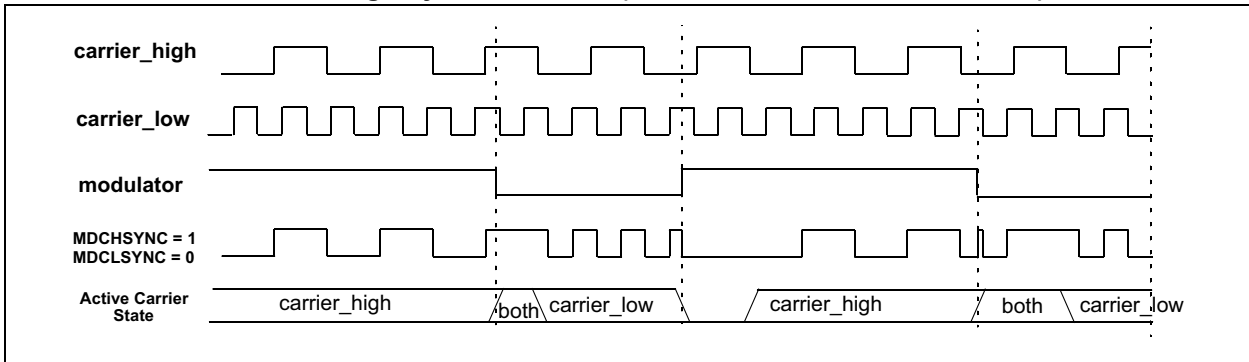


FIGURE 26-5: Carrier Low Synchronization (MDSHSYNC = 0, MDCLSYNC = 1)

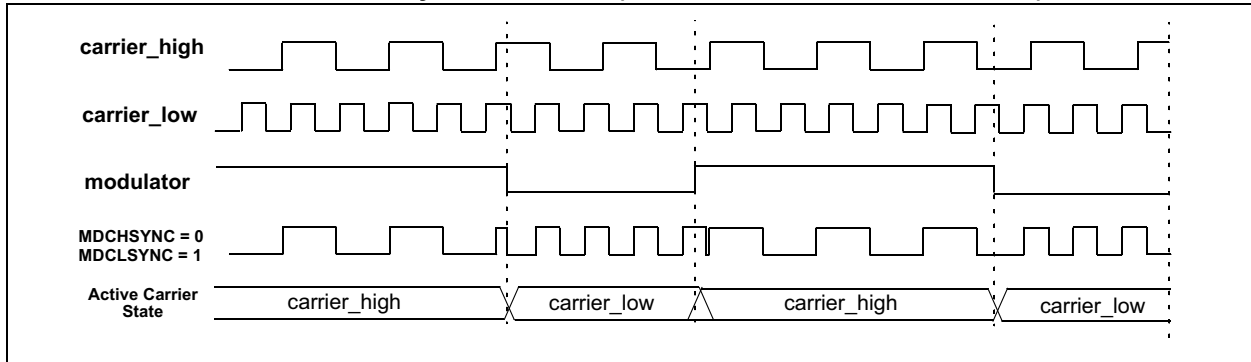
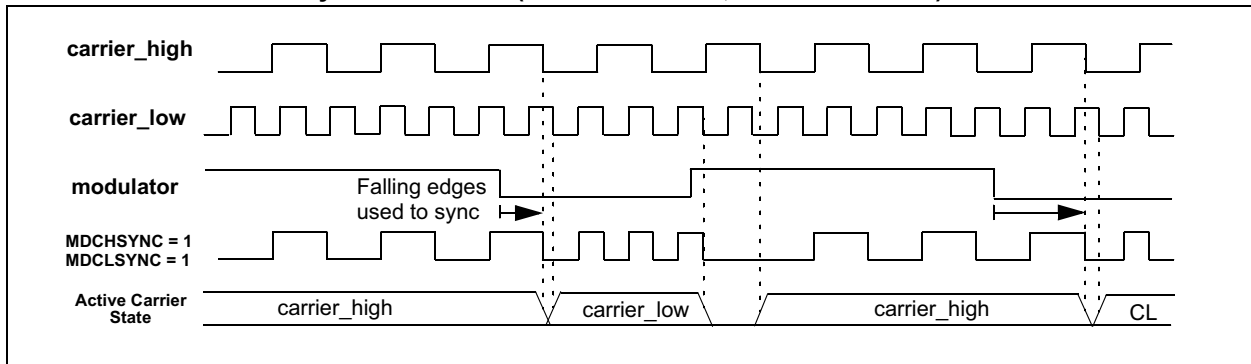


FIGURE 26-6: Full Synchronization (MDSHSYNC = 1, MDCLSYNC = 1)



26.6 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 source is enabled by setting the MDCHPOL bit of the MDCON1 register. Inverting the signal for the carrier low source is enabled by setting the MDCLPOL bit of the MDCON1 register.

26.7 Programmable Modulator Data

The MDBIT of the MDCON0 register can be selected as the source for the modulator signal. This gives the user the ability to program the value used for modulation.

26.8 Modulated Output Polarity

The modulated output signal provided on the DSM pin can also be inverted. Inverting the modulated output signal is enabled by setting the MDOPOL bit of the MDCON0 register.

26.9 Operation in Sleep Mode

The DSM module is not affected by Sleep mode. The DSM can still operate during Sleep, if the Carrier and Modulator input sources are also still operable during Sleep. Refer to [Section 6.0 “Power-Saving Operation Modes”](#) for more details.

26.10 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. The registers are reset to their default values.

26.11 Peripheral Module Disable

The DSM module can be completely disabled using the PMD module to achieve maximum power saving. The DSMMMD bit of PMD2 ([Register 7-3](#)) when set disables the DSM module completely. When enabled again all the registers of the DSM module default to POR status.

27.0 MASTER SYNCHRONOUS SERIAL PORT (MSSP1)MODULE

27.1 MSSP Module Overview

The Master Synchronous Serial Port (MSSP) module is a serial interface useful for communicating with other peripheral or microcontroller devices. These peripheral devices may be serial EEPROMs, shift registers, display drivers, A/D converters, etc. The PIC18(L)F6xK40 devices have two MSSP modules that can operate in one of two modes:

- Serial Peripheral Interface (SPI)
- Inter-Integrated Circuit (I²C)

The SPI interface supports the following modes and features:

- Master mode
- Slave mode
- Clock Parity
- Slave Select Synchronization (Slave mode only)
- Daisy-chain connection of slave devices

The I²C interface supports the following modes and features:

- Master mode
- Slave mode
- Byte NACKing (Slave mode)
- Limited multi-master support
- 7-bit and 10-bit addressing
- Start and Stop interrupts
- Interrupt masking
- Clock stretching
- Bus collision detection
- General call address matching
- Address masking
- Address Hold and Data Hold modes
- Selectable SDA hold times

27.2 SPI Mode Overview

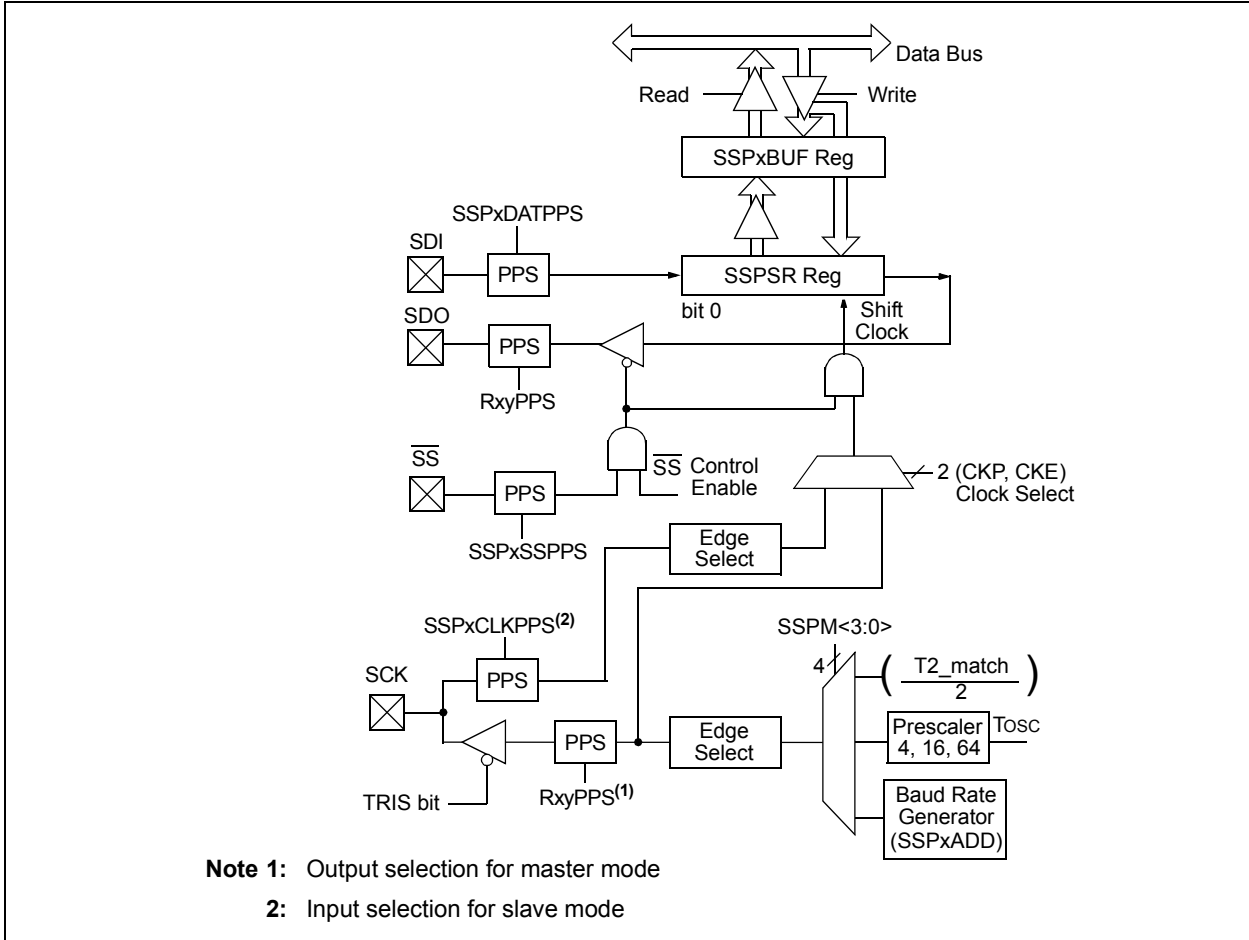
The Serial Peripheral Interface (SPI) bus is a synchronous serial data communication bus that operates in Full-Duplex mode. Devices communicate in a master/slave environment where the master device initiates the communication. A slave device is controlled through a Chip Select known as Slave Select.

The SPI bus specifies four signal connections:

- Serial Clock (SCK)
- Serial Data Out (SDO)
- Serial Data In (SDI)
- Slave Select (\overline{SS})

Figure 27-1 shows the block diagram of the MSSP module when operating in SPI mode.

FIGURE 27-1: MSSP BLOCK DIAGRAM (SPI MODE)



The SPI bus operates with a single master device and one or more slave devices. When multiple slave devices are used, an independent Slave Select connection is required from the master device to each slave device.

Figure 27-2 shows a typical connection between a master device and multiple slave devices.

The master selects only one slave at a time. Most slave devices have tri-state outputs so their output signal appears disconnected from the bus when they are not selected.

FIGURE 27-2: SPI MASTER AND MULTIPLE SLAVE CONNECTION



27.3 SPI Mode Registers

The MSSP module has five registers for SPI mode operation. These are:

- MSSP STATUS register (SSPxSTAT)
- MSSP Control register 1 (SSPxCON1)
- MSSP Control register 3 (SSPxCON3)
- MSSP Data Buffer register (SSPxBUF)
- MSSP Address register (SSPxADD)
- MSSP Shift register (SSPSR)
(Not directly accessible)

SSPxCON1 and SSPxSTAT are the control and STATUS registers in SPI mode operation. The SSPxCON1 register is readable and writable. The lower six bits of the SSPxSTAT are read-only. The upper two bits of the SSPxSTAT are read/write.

In one SPI master mode, SSPxADD can be loaded with a value used in the Baud Rate Generator. More information on the Baud Rate Generator is available in [Section 27.11 “Baud Rate Generator”](#).

SSPSR is the shift register used for shifting data in and out. SSPxBUF provides indirect access to the SSPSR register. SSPxBUF is the buffer register to which data bytes are written, and from which data bytes are read.

In receive operations, SSPSR and SSPxBUF together create a buffered receiver. When SSPSR receives a complete byte, it is transferred to SSPxBUF and the SSPxIF interrupt is set.

During transmission, the SSPxBUF is not buffered. A write to SSPxBUF will write to both SSPxBUF and SSPSR.

27.4 Register Definitions: MSSP Control

REGISTER 27-1: SSPxSTAT: MSSPx STATUS REGISTER (SPI MODE)

| | | | | | | | |
|-------|--------------------|-----|-----|-----|-----|-----|-------|
| R/W-0 | R/W-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |
| SMP | CKE ⁽¹⁾ | D/A | P | S | R/W | UA | BF |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|-------------------|------------------|------------------------------------|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared |
| | | x = Bit is unknown |

- bit 7 **SMP:** Sample bit
SPI Master mode:
 1 = Input data is sampled at the end of data output time
 0 = Input data is sampled at the middle of data output time
SPI Slave mode:
 SMP must be cleared when SPI is used in Slave mode.
- bit 6 **CKE:** SPI Clock Select bit⁽¹⁾
 1 = Transmit occurs on the transition from active to Idle clock state
 0 = Transmit occurs on the transition from Idle to active clock state
- bit 5 **D/A:** Data/Address bit
 Used in I²C mode only.
- bit 4 **P:** Stop bit
 Used in I²C mode only. This bit is cleared when the MSSPx module is disabled; SSPEN is cleared.
- bit 3 **S:** Start bit
 Used in I²C mode only.
- bit 2 **R/W:** Read/Write Information bit
 Used in I²C mode only.
- bit 1 **UA:** Update Address bit
 Used in I²C mode only.
- bit 0 **BF:** Buffer Full Status bit (Receive mode only)
 1 = Receive is complete, SSPxBUF is full
 0 = Receive is not complete, SSPxBUF is empty

Note 1: Polarity of clock state is set by the CKP bit (SSPxCON1<4>).

REGISTER 27-2: SSPxCON1: MSSPx CONTROL REGISTER 1 (SPI MODE)

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|-------|----------------------|----------------------|-------|----------------------|----------------------|----------------------|----------------------|
| WCOL | SSPOV ⁽¹⁾ | SSPEN ⁽²⁾ | CKP | SSPM3 ⁽⁴⁾ | SSPM2 ⁽⁴⁾ | SSPM1 ⁽⁴⁾ | SSPM0 ⁽⁴⁾ |
| bit 7 | | | | | | | bit 0 |

Legend:

R = Readable bit
-n = Value at POR

W = Writable bit
'1' = Bit is set

U = Unimplemented bit, read as '0'
'0' = Bit is cleared

x = Bit is unknown

- bit 7 **WCOL:** Write Collision Detect bit
1 = The SSPxBUF register is written while it is still transmitting the previous word (must be cleared in software)
0 = No collision
- bit 6 **SSPOV:** Receive Overflow Indicator bit⁽¹⁾
SPI Slave mode:
1 = A new byte is received while the SSPxBUF register is still holding the previous data. In case of overflow, the data in SSPxSR is lost. Overflow can only occur in Slave mode. The user must read the SSPxBUF, even if only transmitting data, to avoid setting overflow (must be cleared in software).
0 = No overflow
- bit 5 **SSPEN:** Master Synchronous Serial Port Enable bit⁽²⁾
1 = Enables serial port and configures SCKx, SDOx, SDIx and \overline{SSx} as serial port pins
0 = Disables serial port and configures these pins as I/O port pins
- bit 4 **CKP:** Clock Polarity Select bit
1 = Idle state for the clock is a high level
0 = Idle state for the clock is a low level
- bit 3-0 **SSPM<3:0>:** Master Synchronous Serial Port Mode Select bits⁽⁴⁾
1010 = SPI Master mode: Clock = $F_{OSC}/(4 * (SSPxADD + 1))$ ⁽³⁾
0101 = SPI Slave mode: Clock = SCKx pin; \overline{SSx} pin control is disabled; \overline{SSx} can be used as I/O pin
0100 = SPI Slave mode: Clock = SCKx pin; \overline{SSx} pin control is enabled
0011 = SPI Master mode: Clock = TMR2 output/2
0010 = SPI Master mode: Clock = $F_{OSC}/64$
0001 = SPI Master mode: Clock = $F_{OSC}/16$
0000 = SPI Master mode: Clock = $F_{OSC}/4$

- Note 1:** In Master mode, the overflow bit is not set since each new reception (and transmission) is initiated by writing to the SSPxBUF register.
- 2:** When enabled, these pins must be properly configured as inputs or outputs.
- 3:** SSPxADD = 0 is not supported.
- 4:** Bit combinations not specifically listed here are either reserved or implemented in I²C mode only.

REGISTER 27-3: SSPxCON3: MSSPx CONTROL REGISTER 3 (SPI MODE)

| | | | | | | | |
|-----------|---------------------|---------------------|---------------------|-------|-------|-------|-------|
| R/HS/HC-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
| ACKTIM | PCIE ⁽¹⁾ | SCIE ⁽¹⁾ | BOEN ⁽²⁾ | SDAHT | SBCDE | AHEN | DHEN |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|--------------------|----------------------|--|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| -n = Value at POR | '1' = Bit is set | HS/HC = Bit is set/cleared by hardware |
| x = Bit is unknown | '0' = Bit is cleared | |

| | |
|-------|--|
| bit 7 | ACKTIM: Acknowledge Time Status bit Unused in SPI. |
| bit 6 | PCIE: Stop Condition Interrupt Enable bit ⁽¹⁾ 1 = Enable interrupt on detection of Stop condition 0 = Stop detection interrupts are disabled |
| bit 5 | SCIE: Start Condition Interrupt Enable bit ⁽¹⁾ 1 = Enable interrupt on detection of Start or Restart conditions 0 = Start detection interrupts are disabled |
| bit 4 | BOEN: Buffer Overwrite Enable bit ⁽²⁾ 1 = SSPxBUF updates every time a new data byte is shifted in, ignoring the BF bit 0 = If a new byte is received with BF bit already set, SSPOV is set, and the buffer is not updated |
| bit 3 | SDAHT: SDA Hold Time Selection bit Unused in SPI. |
| bit 2 | SBCDE: Slave Mode Bus Collision Detect Enable bit Unused in SPI. |
| bit 1 | AHEN: Address Hold Enable bit Unused in SPI. |
| bit 0 | DHEN: Data Hold Enable bit Unused in SPI. |

- Note 1:** This bit has no effect in Slave modes that Start and Stop condition detection is explicitly listed as enabled.
- Note 2:** For daisy-chained SPI operation; allows the user to ignore all but the last received byte. SSPOV is still set when a new byte is received and BF = 1, but hardware continues to write the most recent byte to SSPxBUF.

REGISTER 27-4: SSPxBUF: MSSP DATA BUFFER REGISTER

| | | | | | | | |
|----------|-------|-------|-------|-------|-------|-------|-------|
| R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x |
| BUF<7:0> | | | | | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-0 **BUF<7:0>**: MSSP Buffer bits

REGISTER 27-5: SSPxADD: MSSP ADDRESS REGISTER (SPI MODE)

| | | | | | | | |
|----------|-------|-------|-------|-------|-------|-------|-------|
| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
| ADD<7:0> | | | | | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

Master mode: SPI mode

bit 7-0 Baud Rate Clock Divider bits
 SCK/SCL pin clock period = ((SSPxADD<7:0> + 1) * 4) / Fosc

27.5 SPI Mode Operation

Transmissions involve two shift registers, eight bits in size, one in the master and one in the slave. With either the master or the slave 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 Least Significant bit (LSb) is shifted into the same register.

Figure 27-3 shows a typical connection between two processors configured as master and slave devices.

Data is shifted out of both shift registers on the programmed clock edge and latched on the opposite edge of the clock.

The master device transmits information out on its SDO output pin which is connected to, and received by, the slave's SDI input pin. The slave device transmits information out on its SDO output pin, which is connected to, and received by, the master's SDI input pin.

To begin communication, the master device first sends out the clock signal. Both the master and the slave devices should be configured for the same clock polarity.

The master device starts a transmission by sending out the MSb from its shift register. The slave device reads this bit from that same line and saves it into the LSb position of its shift register.

During each SPI clock cycle, a full-duplex data transmission occurs. This means that while the master device is sending out the MSb from its shift register (on its SDO pin) and the slave device is reading this bit and saving it as the LSb of its shift register, that the slave device is also sending out the MSb from its shift register (on its SDO pin) and the master device is reading this bit and saving it as the LSb of its shift register.

After eight bits have been shifted out, the master and slave have exchanged register values.

If there is more data to exchange, the shift registers are loaded with new data and the process repeats itself.

Whether the data is meaningful or not (dummy data), depends on the application software. This leads to three scenarios for data transmission:

- Master sends useful data and slave sends dummy data.
- Master sends useful data and slave sends useful data.
- Master sends dummy data and slave sends useful data.

Transmissions may involve any number of clock cycles. When there is no more data to be transmitted, the master stops sending the clock signal and it deselects the slave.

Every slave device connected to the bus that has not been selected through its slave select line must disregard the clock and transmission signals and must not transmit out any data of its own.

When initializing the SPI, several options need to be specified. This is done by programming the appropriate control bits (SSPxCON1<5:0> and SSPxSTAT<7:6>). These control bits allow the following to be specified:

- Master mode (SCK is the clock output)
- Slave mode (SCK is the clock input)
- Clock Polarity (Idle state of SCK)
- Data Input Sample Phase (middle or end of data output time)
- Clock Edge (output data on rising/falling edge of SCK)
- Clock Rate (Master mode only)
- Slave Select mode (Slave mode only)

To enable the serial port, SSP Enable bit, SSPEN of the SSPxCON1 register, must be set. To reset or reconfigure SPI mode, clear the SSPEN bit, re-initialize the SSPxCONx registers and then set the SSPEN bit. This configures the SDI, SDO, SCK and \overline{SS} pins as serial port pins. For the pins to behave as the serial port function, some must have their data direction bits (in the TRIS register) appropriately programmed as follows:

- SDI must have corresponding TRIS bit set
- SDO must have corresponding TRIS bit cleared
- SCK (Master mode) must have corresponding TRIS bit cleared
- SCK (Slave mode) must have corresponding TRIS bit set
- \overline{SS} must have corresponding TRIS bit set

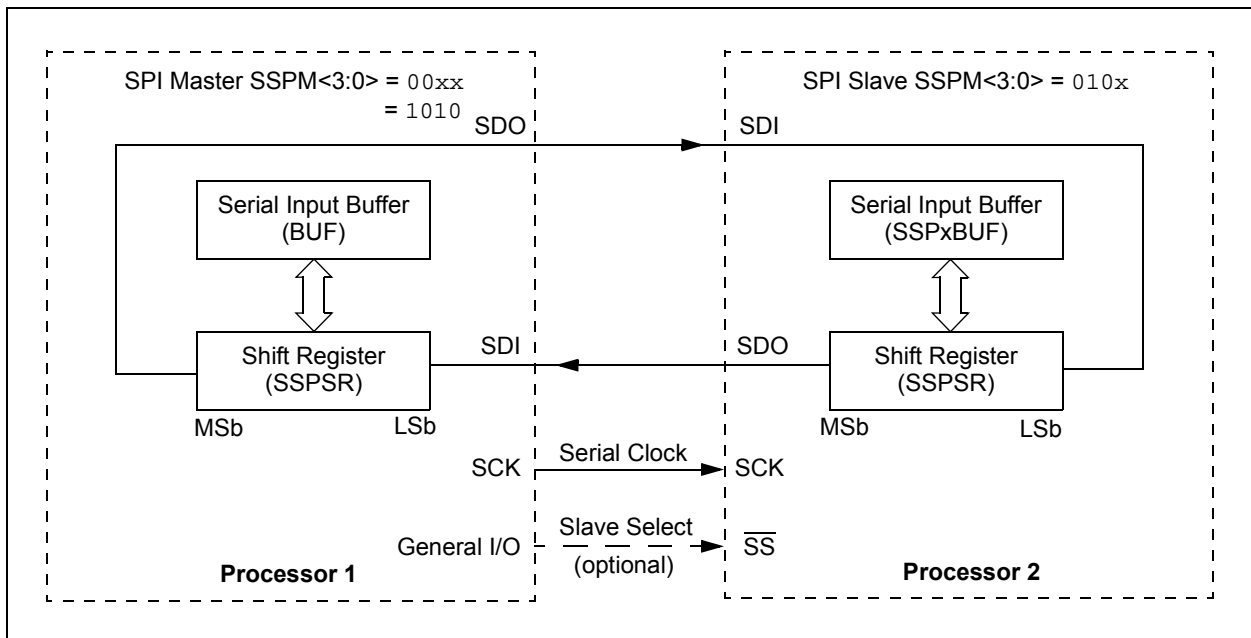
Any serial port function that is not desired may be overridden by programming the corresponding data direction (TRIS) register to the opposite value.

The MSSP consists of a transmit/receive shift register (SSPSR) and a buffer register (SSPxBUF). The SSPSR shifts the data in and out of the device, MSb first. The SSPxBUF holds the data that was written to the SSPSR until the received data is ready. Once the eight bits of data have been received, that byte is moved to the SSPxBUF register. Then, the Buffer Full Detect bit, BF of the SSPxSTAT register, and the interrupt flag bit, SSPxIF, are set. This double-buffering of the received data (SSPxBUF) allows the next byte to start reception before reading the data that was just received. Any write to the SSPxBUF register during transmission/reception of data will be ignored and the write collision detect bit, WCOL of the SSPxCON1 register, will be set. User software must clear the WCOL bit to allow the following write(s) to the SSPxBUF register to complete successfully.

When the application software is expecting to receive valid data, the SSPxBUF should be read before the next byte of data to transfer is written to the SSPxBUF. The Buffer Full bit, BF of the SSPxSTAT register, indicates when SSPxBUF has been loaded with the received data (transmission is complete). When the SSPxBUF is read, the BF bit is cleared. This data may be irrelevant if the SPI is only a transmitter. Generally, the MSSP interrupt is used to determine when the transmission/reception has completed. If the interrupt method is not going to be used, then software polling can be done to ensure that a write collision does not occur.

The SSPSR is not directly readable or writable and can only be accessed by addressing the SSPxBUF register. Additionally, the SSPxSTAT register indicates the various Status conditions.

FIGURE 27-3: SPI MASTER/SLAVE CONNECTION



27.5.1 SPI MASTER MODE

The master can initiate the data transfer at any time because it controls the SCK line. The master determines when the slave (Processor 2, [Figure 27-3](#)) is to broadcast data by the software protocol.

In Master mode, the data is transmitted/received as soon as the SSPxBUF register is written to. If the SPI is only going to receive, the SDO output could be disabled (programmed as an input). The SSPSR register will continue to shift in the signal present on the SDI pin at the programmed clock rate. As each byte is received, it will be loaded into the SSPxBUF register as if a normal received byte (interrupts and Status bits appropriately set).

The clock polarity is selected by appropriately programming the CKP bit of the SSPxCON1 register and the CKE bit of the SSPxSTAT register. This then, would give waveforms for SPI communication as shown in [Figure 27-4](#), [Figure 27-6](#), [Figure 27-7](#) and [Figure 27-8](#), where the MSB is transmitted first. In Master mode, the SPI clock rate (bit rate) is user programmable to be one of the following:

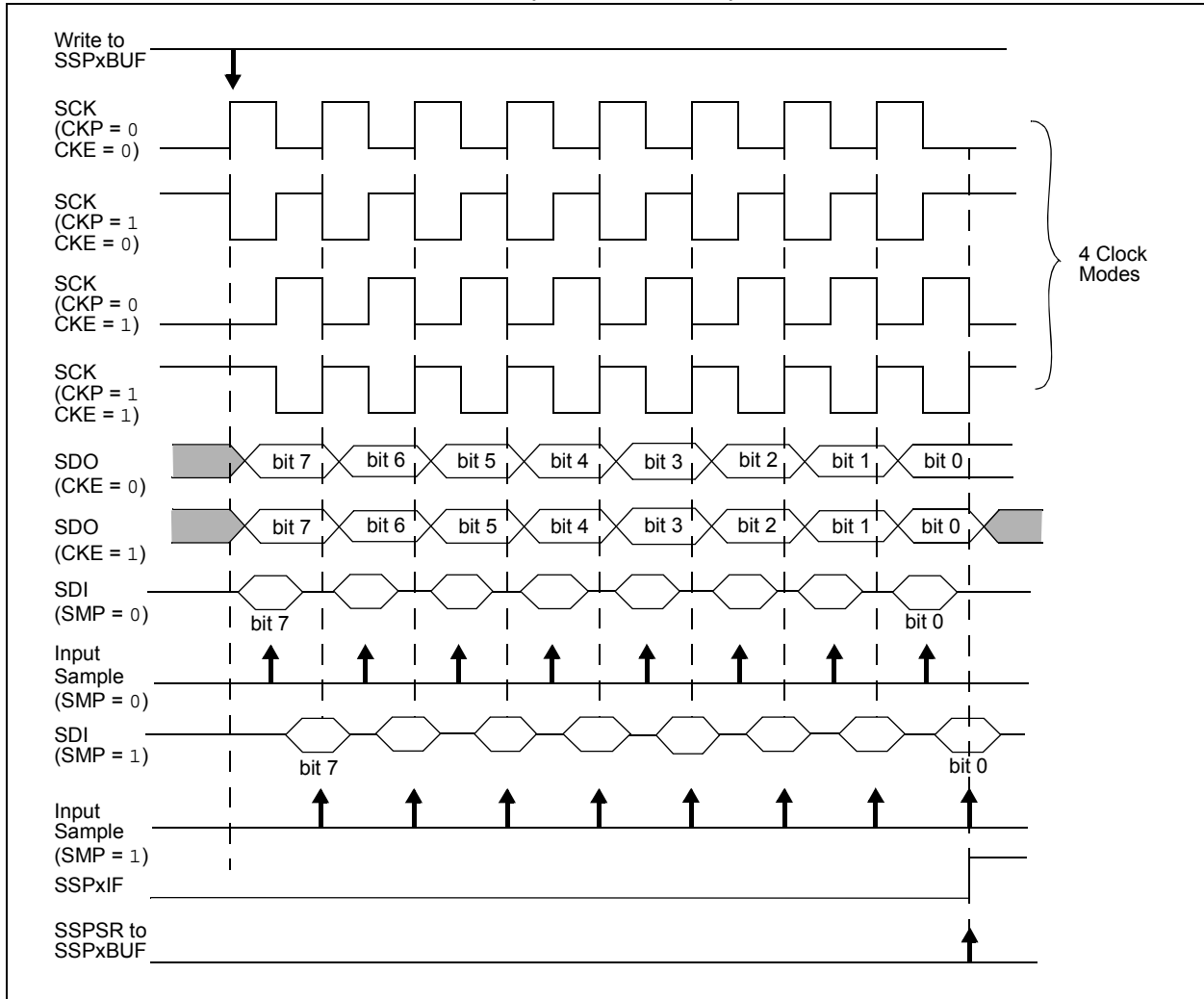
- $F_{osc}/4$ (or T_{CY})
- $F_{osc}/16$ (or $4 * T_{CY}$)
- $F_{osc}/64$ (or $16 * T_{CY}$)
- Timer2 output/2
- $F_{osc}/(4 * (SSPxADD + 1))$

[Figure 27-4](#) shows the waveforms for Master mode.

When the CKE bit is set, the SDO data is valid before there is a clock edge on SCK. The change of the input sample is shown based on the state of the SMP bit. The time when the SSPxBUF is loaded with the received data is shown.

Note: In Master mode the clock signal output to the SCK pin is also the clock signal input to the peripheral. The pin selected for output with the RxyPPS register must also be selected as the peripheral input with the SSPxCLKPPS register.

FIGURE 27-4: SPI MODE WAVEFORM (MASTER MODE)



27.5.2 SPI SLAVE MODE

In Slave mode, the data is transmitted and received as external clock pulses appear on SCK. When the last bit is latched, the SSPxIF interrupt flag bit is set.

Before enabling the module in SPI Slave mode, the clock line must match the proper Idle state. The clock line can be observed by reading the SCK pin. The Idle state is determined by the CKP bit of the SSPxCON1 register.

While in Slave mode, the external clock is supplied by the external clock source on the SCK pin. This external clock must meet the minimum high and low times as specified in the electrical specifications.

While in Sleep mode, the slave can transmit/receive data. The shift register is clocked from the SCK pin input and when a byte is received, the device will generate an interrupt. If enabled, the device will wake-up from Sleep.

27.5.3 DAISY-CHAIN CONFIGURATION

The SPI bus can sometimes be connected in a daisy-chain configuration. The first slave output is connected to the second slave input, the second slave output is connected to the third slave input, and so on. The final slave output is connected to the master input. Each slave 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 Slave Select line from the master device.

Figure 27-5 shows the block diagram of a typical daisy-chain connection when operating in SPI mode.

In a daisy-chain configuration, only the most recent byte on the bus is required by the slave. Setting the BOEN bit of the SSPxCON3 register will enable writes to the SSPxBUF register, even if the previous byte has not been read. This allows the software to ignore data that may not apply to it.

27.5.4 SLAVE SELECT SYNCHRONIZATION

The Slave Select can also be used to synchronize communication. The Slave Select line is held high until the master device is ready to communicate. When the Slave Select line is pulled low, the slave knows that a new transmission is starting.

If the slave fails to receive the communication properly, it will be reset at the end of the transmission, when the Slave Select line returns to a high state. The slave is then ready to receive a new transmission when the Slave Select line is pulled low again. If the Slave Select line is not used, there is a risk that the slave will eventually become out of sync with the master. If the slave misses a bit, it will always be one bit off in future transmissions. Use of the Slave Select line allows the slave and master to align themselves at the beginning of each transmission.

The \overline{SS} pin allows a Synchronous Slave mode. The SPI must be in Slave mode with \overline{SS} pin control enabled ($SSPxCON1<3:0> = 0100$).

When the \overline{SS} pin is low, transmission and reception are enabled and the SDO pin is driven.

When the \overline{SS} pin goes high, the SDO pin is no longer driven, even if in the middle of a transmitted byte and becomes a floating output. External pull-up/pull-down resistors may be desirable depending on the application.

- Note 1:** When the SPI is in Slave mode with \overline{SS} pin control enabled ($SSPxCON1<3:0> = 0100$), the SPI module will reset if the \overline{SS} pin is set to VDD.
- 2:** When the SPI is used in Slave mode with CKE set; the user must enable \overline{SS} pin control.
- 3:** While operated in SPI Slave mode the SMP bit of the SSPxSTAT register must remain clear.

When the SPI module resets, the bit counter is forced to '0'. This can be done by either forcing the \overline{SS} pin to a high level or clearing the SSPEN bit.

FIGURE 27-5: SPI DAISY-CHAIN CONNECTION

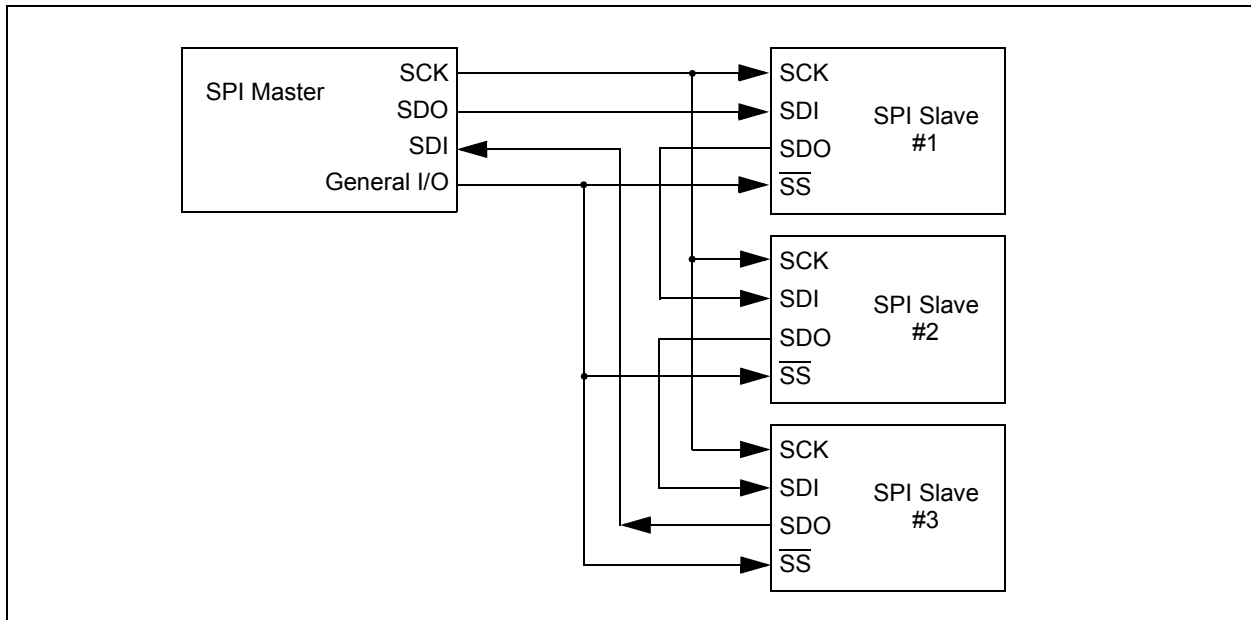


FIGURE 27-6: SLAVE SELECT SYNCHRONOUS WAVEFORM

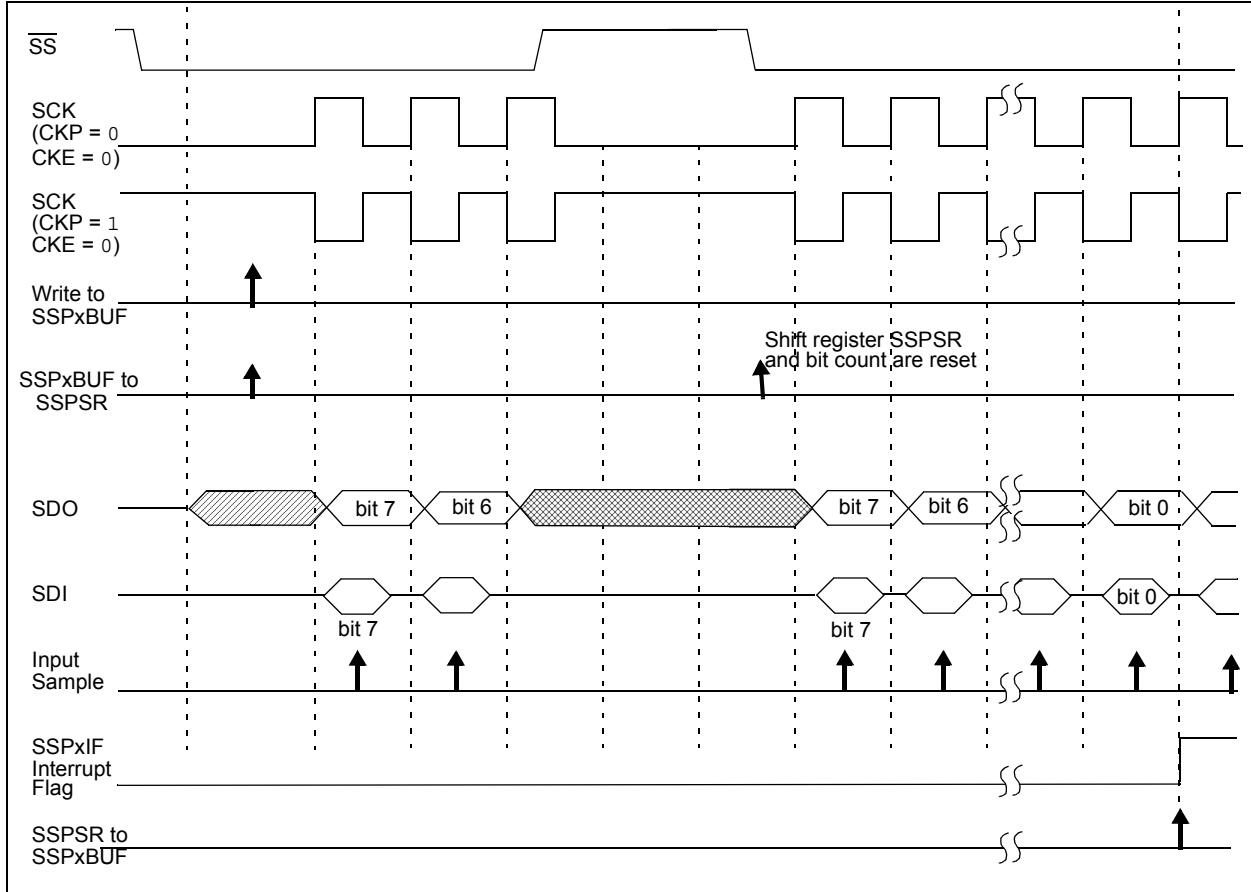


FIGURE 27-7: SPI MODE WAVEFORM (SLAVE MODE WITH CKE = 0)

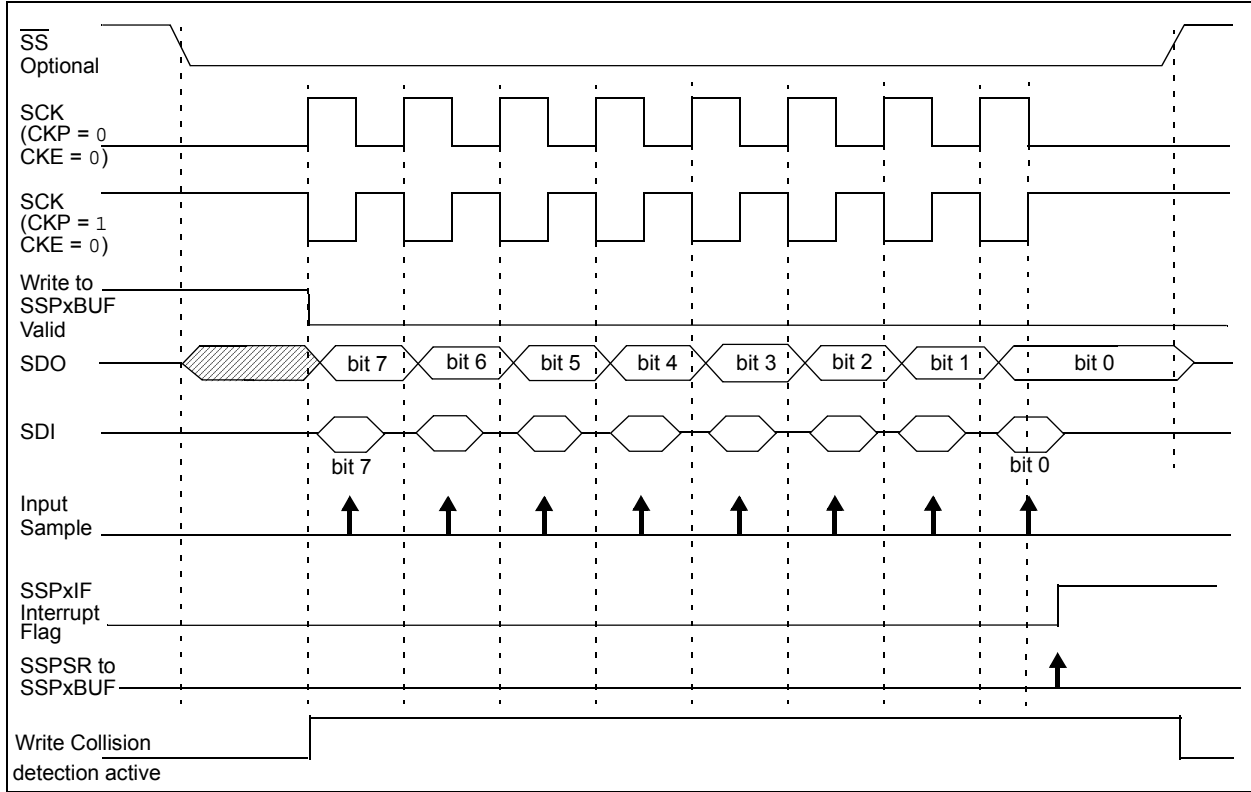
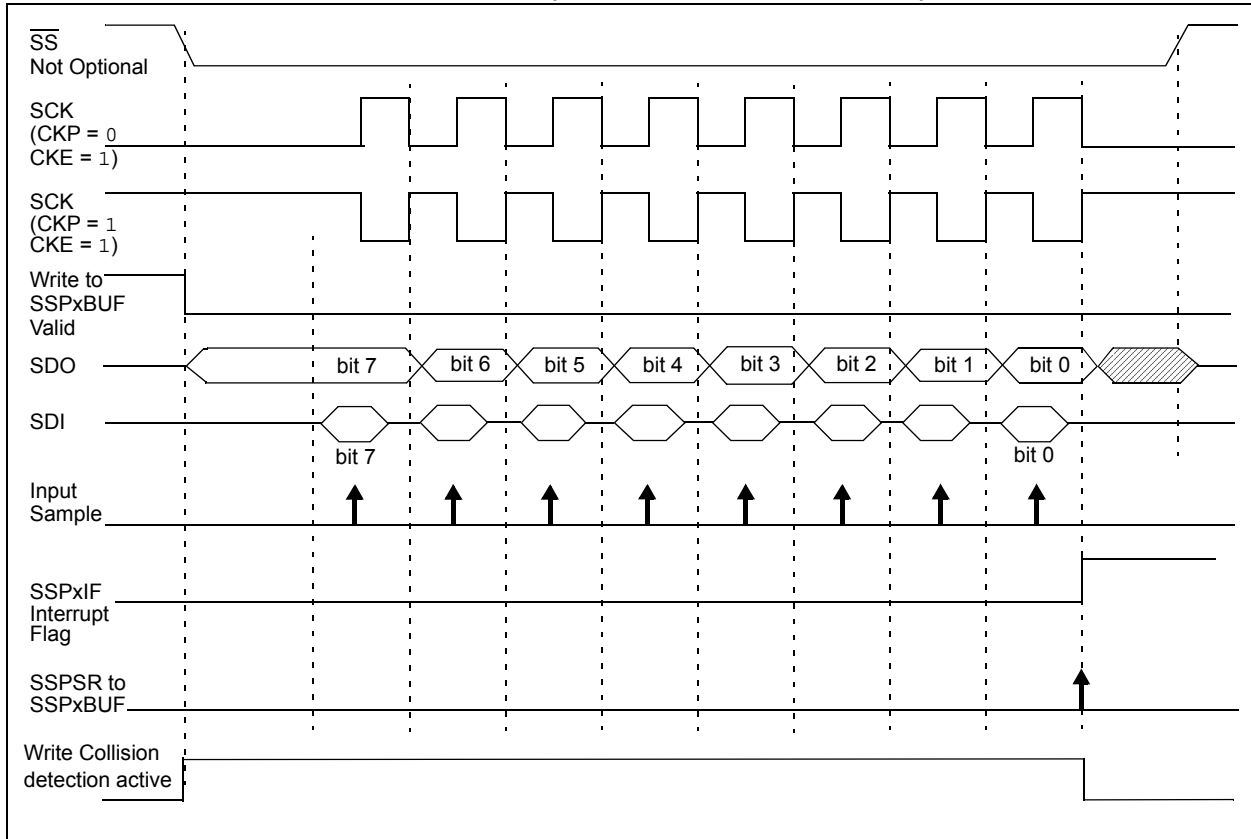


FIGURE 27-8: SPI MODE WAVEFORM (SLAVE MODE WITH CKE = 1)



27.5.5 SPI OPERATION IN SLEEP MODE

In SPI Master mode, module clocks may be operating at a different speed than when in Full-Power mode; in the case of the Sleep mode, all clocks are halted.

Special care must be taken by the user when the MSSP clock is much faster than the system clock.

In Slave mode, when MSSP interrupts are enabled, after the master completes sending data, an MSSP interrupt will wake the controller from Sleep.

If an exit from Sleep mode is not desired, MSSP interrupts should be disabled.

In SPI Master mode, when the Sleep mode is selected, all module clocks are halted and the transmission/reception will remain in that state until the device wakes. After the device returns to Run mode, the module will resume transmitting and receiving data.

In SPI Slave mode, the SPI Transmit/Receive Shift register operates asynchronously to the device. This allows the device to be placed in Sleep mode and data to be shifted into the SPI Transmit/Receive Shift register. When all eight bits have been received, the MSSP interrupt flag bit will be set and if enabled, will wake the device.

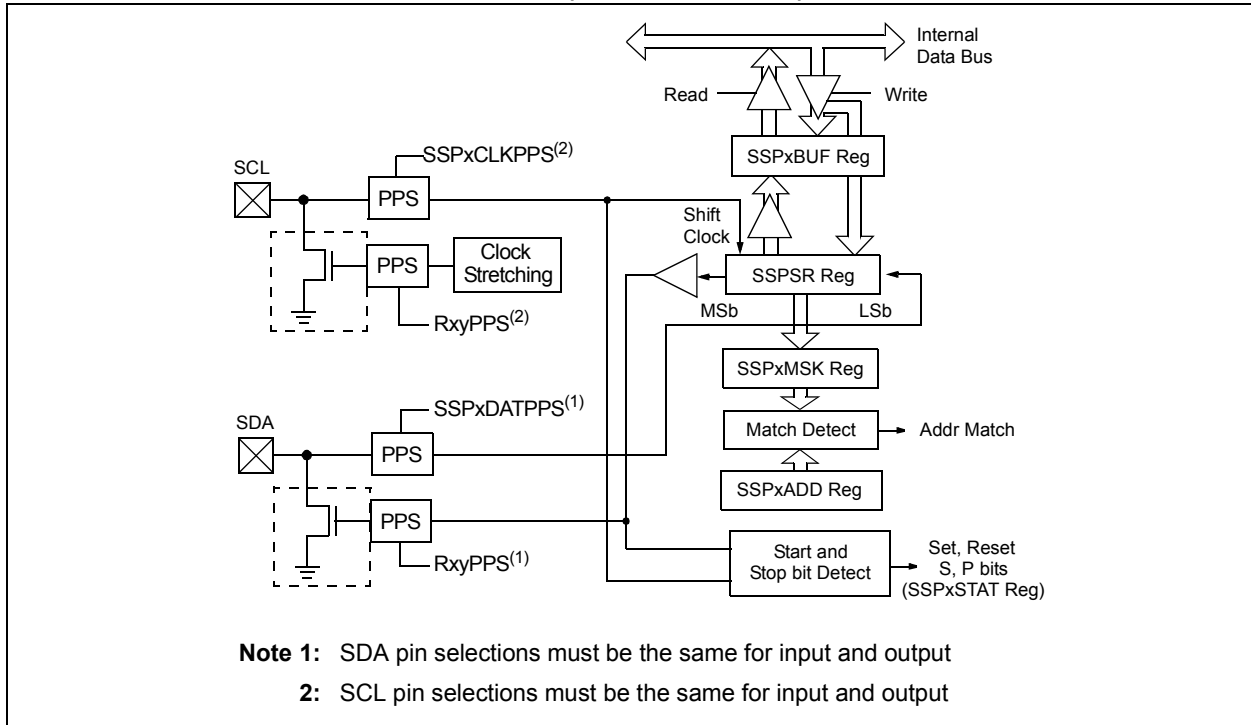
TABLE 27-1: SUMMARY OF REGISTERS ASSOCIATED WITH SPI OPERATION

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on Page |
|------------|----------|-----------|------------------|-------|-----------|---------|---------|---------|------------------|
| INTCON | GIE/GIEH | PEIE/GIEL | IPEN | — | INT3EDG | INT2EDG | INT1EDG | INT0EDG | 172 |
| PIE3 | RC2IE | TX2IE | RC1IE | TX1IE | BCL2IE | SSP2IE | BCL1IE | SSP1IE | 187 |
| PIR3 | RC2IF | TX2IF | RC1IF | TX1IF | BCL2IF | SSP2IF | BCL1IF | SSP1IF | 176 |
| IPR3 | RC2IP | TX2IP | RC1IP | TX1IP | BCL2IP | SSP2IP | BCL1IP | SSP1IP | 197 |
| RxyPPS | — | — | RxyPPS<5:0> | | | | | | 227 |
| SSPxBUF | BUF<7:0> | | | | | | | | 387* |
| SSPxCLKPPS | — | — | SSPxCLKPPS<<5:0> | | | | | | 224 |
| SSPxCON1 | WCOL | SSPOV | SSPEN | CKP | SSPM<3:0> | | | | 389 |
| SSPxCON3 | ACKTIM | PCIE | SCIE | BOEN | SDAHT | SBCDE | AHEN | DHEN | 390 |
| SSPxDATPPS | — | — | SSPDATPPS<5:0> | | | | | | 224 |
| SSPxSSPPS | — | — | SSPSSPPS<5:0> | | | | | | 224 |
| SSPxSTAT | SMP | CKE | D/Ā | P | S | R/Ā | UA | BF | 404 |

Legend: — = Unimplemented location, read as '0'. Shaded cells are not used by the MSSP in SPI mode.

* Page provides register information.

FIGURE 27-10: MSSP BLOCK DIAGRAM (I²C SLAVE MODE)



The I²C bus specifies two signal connections:

- Serial Clock (SCL)
- Serial Data (SDA)

Both the SCL and SDA connections are bidirectional open-drain lines, each requiring pull-up resistors for the supply voltage. Pulling the line to ground is considered a logical zero and letting the line float is considered a logical one.

Figure 27-11 shows a typical connection between two processors configured as master and slave devices.

The I²C bus can operate with one or more master devices and one or more slave devices.

There are four potential modes of operation for a given device:

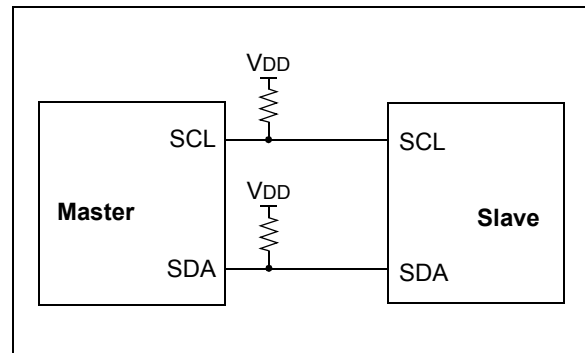
- Master Transmit mode (master is transmitting data to a slave)
- Master Receive mode (master is receiving data from a slave)
- Slave Transmit mode (slave is transmitting data to a master)
- Slave Receive mode (slave is receiving data from the master)

To begin communication, a master device starts out in Master Transmit mode. The master device sends out a Start bit followed by the address byte of the slave it intends to communicate with. This is followed by a single Read/Write bit, which determines whether the master intends to transmit to or receive data from the slave device.

If the requested slave exists on the bus, it will respond with an Acknowledge bit, otherwise known as an $\overline{\text{ACK}}$. The master then continues in either Transmit mode or Receive mode and the slave continues in the complement, either in Receive mode or Transmit mode, respectively.

A Start bit is indicated by a high-to-low transition of the SDA line while the SCL line is held high. Address and data bytes are sent out, Most Significant bit (MSb) first. The Read/Write bit is sent out as a logical one when the master intends to read data from the slave, and is sent out as a logical zero when it intends to write data to the slave.

FIGURE 27-11: I²C MASTER/SLAVE CONNECTION



The Acknowledge bit ($\overline{\text{ACK}}$) is an active-low signal, which holds the SDA line low to indicate to the transmitter that the slave device has received the transmitted data and is ready to receive more.

The transition of a data bit is always performed while the SCL line is held low. Transitions that occur while the SCL line is held high are used to indicate Start and Stop bits.

If the master intends to write to the slave, then it repeatedly sends out a byte of data, with the slave responding after each byte with an $\overline{\text{ACK}}$ bit. In this example, the master device is in Master Transmit mode and the slave is in Slave Receive mode.

If the master intends to read from the slave, then it repeatedly receives a byte of data from the slave, and responds after each byte with an $\overline{\text{ACK}}$ bit. In this example, the master device is in Master Receive mode and the slave is Slave Transmit mode.

On the last byte of data communicated, the master device may end the transmission by sending a Stop bit. If the master device is in Receive mode, it sends the Stop bit in place of the last $\overline{\text{ACK}}$ bit. A Stop bit is indicated by a low-to-high transition of the SDA line while the SCL line is held high.

In some cases, the master may want to maintain control of the bus and re-initiate another transmission. If so, the master device may send another Start bit in place of the Stop bit or last $\overline{\text{ACK}}$ bit when it is in receive mode.

The I²C bus specifies three message protocols;

- Single message where a master writes data to a slave.
- Single message where a master reads data from a slave.
- Combined message where a master initiates a minimum of two writes, or two reads, or a combination of writes and reads, to one or more slaves.

When one device is transmitting a logical one, or letting the line float, and a second device is transmitting a logical zero, or holding the line low, the first device can detect that the line is not a logical one. This detection, when used on the SCL line, is called clock stretching. Clock stretching gives slave devices a mechanism to control the flow of data. When this detection is used on the SDA line, it is called arbitration. Arbitration ensures that there is only one master device communicating at any single time.

27.6.1 CLOCK STRETCHING

When a slave device has not completed processing data, it can delay the transfer of more data through the process of clock stretching. An addressed slave device may hold the SCL clock line low after receiving or sending a bit, indicating that it is not yet ready to continue. The master that is communicating with the slave will attempt to raise the SCL line in order to transfer the next bit, but will detect that the clock line has not yet been released. Because the SCL connection is open-drain, the slave has the ability to hold that line low until it is ready to continue communicating.

Clock stretching allows receivers that cannot keep up with a transmitter to control the flow of incoming data.

27.6.2 ARBITRATION

Each master device must monitor the bus for Start and Stop bits. If the device detects that the bus is busy, it cannot begin a new message until the bus returns to an Idle state.

However, two master devices may try to initiate a transmission on or about the same time. When this occurs, the process of arbitration begins. Each transmitter checks the level of the SDA data line and compares it to the level that it expects to find. The first transmitter to observe that the two levels do not match, loses arbitration, and must stop transmitting on the SDA line.

For example, if one transmitter holds the SDA line to a logical one (lets it float) and a second transmitter holds it to a logical zero (pulls it low), the result is that the SDA line will be low. The first transmitter then observes that the level of the line is different than expected and concludes that another transmitter is communicating.

The first transmitter to notice this difference is the one that loses arbitration and must stop driving the SDA line. If this transmitter is also a master device, it also must stop driving the SCL line. It then can monitor the lines for a Stop condition before trying to reissue its transmission. In the meantime, the other device that has not noticed any difference between the expected and actual levels on the SDA line continues with its original transmission. It can do so without any complications, because so far, the transmission appears exactly as expected with no other transmitter disturbing the message.

Slave Transmit mode can also be arbitrated, when a master addresses multiple slaves, but this is less common.

If two master devices are sending a message to two different slave devices at the address stage, the master sending the lower slave address always wins arbitration. When two master devices send messages to the same slave address, and addresses can sometimes refer to multiple slaves, the arbitration process must continue into the data stage.

Arbitration usually occurs very rarely, but it is a necessary process for proper multi-master support.

27.7 Register Definitions: I²C Mode

The MSSPx module has seven registers for I²C operation.

These are:

- MSSP Status Register (SSPxSTAT)
- MSSP Control Register 1 (SSPxCON1)
- MSSP Control Register 2 (SSPxCON2)
- MSSP Control Register 3 (SSPxCON3)
- Serial Receive/Transmit Buffer Register (SSPxBUF)
- MSSP Address Register (SSPxADD)
- I²C Slave Address Mask Register (SSPxMSK)
- MSSP Shift Register (SSPSR) – not directly accessible

SSPxCON1, SSPxCON2, SSPxCON3 and SSPxSTAT are the Control and Status registers in I²C mode operation. The SSPxCON1, SSPxCON2, and SSPxCON3 registers are readable and writable. The lower six bits of the SSPxSTAT are read-only. The upper two bits of the SSPxSTAT are read/write. SSPSR is the Shift register used for shifting data in or out. SSPxBUF is the buffer register to which data bytes are written to or read from. SSPxADD contains the slave device address when the MSSP is configured in I²C Slave mode. When the MSSP is configured in Master mode, the lower seven bits of SSPxADD act as the Baud Rate Generator reload value.

SSPxMSK holds the slave address mask value when the module is configured for 7-Bit Address Masking mode. While it is a separate register, it shares the same SFR address as SSPxADD; it is only accessible when the SSPM<3:0> bits are specifically set to permit access. In receive operations, SSPSR and SSPxBUF together, create a double-buffered receiver. When SSPSR receives a complete byte, it is transferred to SSPxBUF and the SSPxIF interrupt is set. During transmission, the SSPxBUF is not double-buffered. A write to SSPxBUF will write to both SSPxBUF and SSPSR.

REGISTER 27-6: SSPxSTAT: MSSPx STATUS REGISTER (I²C MASTER MODE)

| | | | | | | | |
|-------|-------|-----|------------------|------------------|----------------------|-----|-------|
| R/W-0 | R/W-0 | R-0 | R-0 | R-0 | R-0 | R-0 | R-0 |
| SMP | CKE | D/A | P ⁽¹⁾ | S ⁽¹⁾ | R/W ^(2,3) | UA | BF |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|-------------------|------------------|------------------------------------|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared |
| | | x = Bit is unknown |

- bit 7 **SMP:** Slew Rate Control bit
In Master or Slave mode:
 1 = Slew rate control is disabled for Standard Speed mode (100 kHz and 1 MHz)
 0 = Slew rate control is enabled for High-Speed mode (400 kHz)
- bit 6 **CKE:** SMBus Select bit
In Master or Slave mode:
 1 = Enables SMBus-specific inputs
 0 = Disables SMBus-specific inputs
- bit 5 **D/A:** Data/Address bit
In Master mode:
 Reserved.
In Slave mode:
 1 = Indicates that the last byte received or transmitted was data
 0 = Indicates that the last byte received or transmitted was address
- bit 4 **P:** Stop bit⁽¹⁾
 1 = Indicates that a Stop bit has been detected last
 0 = Stop bit was not detected last
- bit 3 **S:** Start bit⁽¹⁾
 1 = Indicates that a Start bit has been detected last
 0 = Start bit was not detected last
- bit 2 **R/W:** Read/Write Information bit^(2,3)
In Slave mode:
 1 = Read
 0 = Write
In Master mode:
 1 = Transmit is in progress
 0 = Transmit is not in progress
- bit 1 **UA:** Update Address bit (10-Bit Slave mode only)
 1 = Indicates that the user needs to update the address in the SSPxADD register
 0 = Address does not need to be updated
- bit 0 **BF:** Buffer Full Status bit
In Transmit mode:
 1 = SSPxBUF is full
 0 = SSPxBUF is empty
In Receive mode:
 1 = SSPxBUF is full (does not include the $\overline{\text{ACK}}$ and Stop bits)
 0 = SSPxBUF is empty (does not include the $\overline{\text{ACK}}$ and Stop bits)

- Note 1:** This bit is cleared on Reset and when SSPEN is cleared.
- 2:** This bit holds the R/W bit information following the last address match. This bit is only valid from the address match to the next Start bit, Stop bit or not $\overline{\text{ACK}}$ bit.
- 3:** ORing this bit with SEN, RSEN, PEN, RCEN or ACKEN will indicate if the MSSPx is in Active mode.

REGISTER 27-7: SSPxCON1: MSSPx CONTROL REGISTER 1 (I²C MASTER MODE)

| R/W-0 | R/W/HC-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|-------|----------|----------------------|-------|----------------------|----------------------|----------------------|----------------------|
| WCOL | SSPOV | SSPEN ⁽¹⁾ | CKP | SSPM3 ⁽²⁾ | SSPM2 ⁽²⁾ | SSPM1 ⁽²⁾ | SSPM0 ⁽²⁾ |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|-------------------|------------------|---------------------------------|
| R = Readable bit | W = Writable bit | HC = Bit is cleared by hardware |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared |
| | | x = Bit is unknown |

- bit 7 **WCOL:** Write Collision Detect bit
In Master Transmit mode:
 1 = A write to the SSPxBUF register was attempted while the I²C conditions were not valid for a transmission to be started (must be cleared in software)
 0 = No collision
In Slave Transmit mode:
 1 = The SSPxBUF register is written while it is still transmitting the previous word (must be cleared in software)
 0 = No collision
In Receive mode (Master or Slave modes):
 This is a "don't care" bit.
- bit 6 **SSPOV:** Receive Overflow Indicator bit
In Receive mode:
 1 = A byte is received while the SSPxBUF register is still holding the previous byte (must be cleared in software)
 0 = No overflow
In Transmit mode:
 This is a "don't care" bit in Transmit mode.
- bit 5 **SSPEN:** Master Synchronous Serial Port Enable bit⁽¹⁾
 1 = Enables the serial port and configures the SDAx and SCLx pins as the serial port pins
 0 = Disables serial port and configures these pins as I/O port pins
- bit 4 **CKP:** SCKx Release Control bit
In Slave mode:
 1 = Releases clock
 0 = Holds clock low (clock stretch), used to ensure data setup time
In Master mode:
 Unused in this mode.
- bit 3-0 **SSPM<3:0>:** Master Synchronous Serial Port Mode Select bits⁽²⁾
 1111 = I²C Slave mode: 10-bit address with Start and Stop bit interrupts enabled
 1110 = I²C Slave mode: 7-bit address with Start and Stop bit interrupts enabled
 1011 = I²C Firmware Controlled Master mode (slave Idle)
 1001 = Load SSPxMSK register at SSPxADD SFR address^(3,4)
 1000 = I²C Master mode: Clock = FOSC/(4 * (SSPxADD + 1))
 0111 = I²C Slave mode: 10-bit address^(3,4)
 0110 = I²C Slave mode: 7-bit address

- Note 1:** When enabled, the SDAx and SCLx pins must be configured as inputs.
2: Bit combinations not specifically listed here are either reserved or implemented in SPI mode only.
3: When SSPM<3:0> = 1001, any reads or writes to the SSPxADD SFR address actually accesses the SSPxMSK register.
4: This mode is only available when 7-Bit Address Masking mode is selected (MSSPMASK Configuration bit is '1').

REGISTER 27-8: SSPxCON2: MSSPx CONTROL REGISTER 2 (I²C MASTER MODE)

| R/W-0 | R/W/HC-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|-------|----------|----------------------|----------------------|---------------------|--------------------|---------------------|--------------------|
| GCEN | ACKSTAT | ACKDT ⁽¹⁾ | ACKEN ⁽²⁾ | RCEN ⁽²⁾ | PEN ⁽²⁾ | RSEN ⁽²⁾ | SEN ⁽²⁾ |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|-------------------|------------------|---------------------------------|
| R = Readable bit | W = Writable bit | HC = Bit is cleared by hardware |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared |
| | | x = Bit is unknown |

- bit 7 **GCEN:** General Call Enable bit
Unused in Master mode.
- bit 6 **ACKSTAT:** Acknowledge Status bit (Master Transmit mode only)
1 = Acknowledge was not received from slave
0 = Acknowledge was received from slave
- bit 5 **ACKDT:** Acknowledge Data bit (Master Receive mode only)⁽¹⁾
1 = Not Acknowledge
0 = Acknowledge
- bit 4 **ACKEN:** Acknowledge Sequence Enable bit⁽²⁾
1 = Initiates Acknowledge sequence on SDAx and SCLx pins and transmits ACKDT data bit; automatically cleared by hardware
0 = Acknowledge sequence is Idle
- bit 3 **RCEN:** Receive Enable bit (Master Receive mode only)⁽²⁾
1 = Enables Receive mode for I²C
0 = Receive is Idle
- bit 2 **PEN:** Stop Condition Enable bit⁽²⁾
1 = Initiates Stop condition on SDAx and SCLx pins; automatically cleared by hardware
0 = Stop condition is Idle
- bit 1 **RSEN:** Repeated Start Condition Enable bit⁽²⁾
1 = Initiates Repeated Start condition on SDAx and SCLx pins; automatically cleared by hardware
0 = Repeated Start condition is Idle
- bit 0 **SEN:** Start Condition Enable bit⁽²⁾
1 = Initiates Start condition on SDAx and SCLx pins; automatically cleared by hardware
0 = Start condition is Idle

Note 1: The value that will be transmitted when the user initiates an Acknowledge sequence at the end of a receive.

2: If the I²C module is active, these bits may not be set (no spooling) and the SSPxBUF may not be written (or writes to the SSPxBUF are disabled).

PIC18(L)F67K40

REGISTER 27-9: SSPxCON3: MSSPx CONTROL REGISTER 3 (I²C MASTER MODE)

| | | | | | | | |
|-----------|-------|-------|-------|-------|-------|-------|-------|
| R/HS/HC-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
| ACKTIM | PCIE | SCIE | BOEN | SDAHT | SBCDE | AHEN | DHEN |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|--------------------|----------------------|--|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| -n = Value at POR | '1' = Bit is set | HS/HC = Bit is set/cleared by hardware |
| x = Bit is unknown | '0' = Bit is cleared | |

| | |
|-------|---|
| bit 7 | ACKTIM: Acknowledge Time Status bit Unused in Master mode. |
| bit 6 | PCIE: Stop Condition Interrupt Enable bit ⁽¹⁾ 1 = Enable interrupt on detection of Stop condition 0 = Stop detection interrupts are disabled |
| bit 5 | SCIE: Start Condition Interrupt Enable bit ⁽¹⁾ 1 = Enable interrupt on detection of Start or Restart conditions 0 = Start detection interrupts are disabled |
| bit 4 | BOEN: Buffer Overwrite Enable bit 1 = SSPxBUF is updated every time a new data byte is available, ignoring the SSPOV effect on updating the buffer 0 = SSPxBUF is only updated when SSPOV is clear |
| bit 3 | SDAHT: SDA Hold Time Selection bit 1 = Minimum of 300ns hold time on SDA after the falling edge of SCL 0 = Minimum of 100ns hold time on SDA after the falling edge of SCL |
| bit 2 | SBCDE: Slave Mode Bus Collision Detect Enable bit Unused in Master mode. |
| bit 1 | AHEN: Address Hold Enable bit Unused in Master mode. |
| bit 0 | DHEN: Data Hold Enable bit Unused in Master mode. |

Note 1: This bit has no effect in Slave modes that Start and Stop condition detection is explicitly listed as enabled.

REGISTER 27-10: SSPxBUF: MSSP DATA BUFFER REGISTER (I²C MASTER MODE)

| | | | | | | | |
|----------|-------|-------|-------|-------|-------|-------|-------|
| R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x | R/W-x |
| BUF<7:0> | | | | | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-0 **BUF<7:0>:** MSSP Buffer bits

REGISTER 27-11: SSPxADD: MSSP ADDRESS REGISTER (I²C MASTER MODE)

| | | | | | | | |
|----------|-------|-------|-------|-------|-------|-------|-------|
| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
| ADD<7:0> | | | | | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

Master mode: I²C mode

bit 7-0 Baud Rate Clock Divider bits⁽¹⁾
 $SCK/SCL \text{ pin clock period} = ((SSPxADD<7:0> + 1) * 4) / F_{osc}$

10-Bit Slave mode – Most Significant Address Byte:

bit 7-3 **Not used:** Unused for Most Significant Address Byte. Bit state of this register is a don't care. Bit pattern sent by master is fixed by I²C specification and must be equal to, '11110'. However, those bits are compared by hardware and are not affected by the value in this register.

bit 2-1 **ADD<9:8>:** Two Most Significant bits of 10-bit Address

bit 0 **Not used:** Unused in this mode. Bit state is a don't care.

10-Bit Slave mode – Least Significant Address Byte:

bit 7-0 **ADD<7:0>:** Eight Least Significant bits of 10-bit Address

7-Bit Slave mode:

bit 7-1 7-bit Slave Address

bit 0 **Not used:** Unused in this mode. Bit state is a don't care.

Note 1: Values of 0x00, 0x01 and 0x02 are not valid for SSPxADD when used as a Baud Rate Generator for I²C. This is an implementation limitation.

REGISTER 27-12: SSPxMSK: MSSPx ADDRESS MASK REGISTER

| | | | | | | | |
|----------|-------|-------|-------|-------|-------|-------|-------|
| R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 | R/W-1 |
| MSK<7:1> | | | | | | | MSK0 |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-1 **MSK<7:1>:** Mask bits
 1 = The received address bit n is compared to SSPxADDn to detect I²C address match
 0 = The received address bit n is not used to detect I²C address match

bit 0 **MSK0:** Mask bit for I²C Slave mode, 10-bit Address
 I²C Slave mode, 10-bit address (SSPM<3:0> = 0111 or 1111):
 1 = The received address bit 0 is compared to SSPxADD0 to detect I²C address match
 0 = The received address bit 0 is not used to detect I²C address match
 I²C Slave mode, 7-bit address, the bit is ignored.

27.8 I²C Mode Operation

All MSSP I²C communication is byte oriented and shifted out MSb first. Six SFR registers and two interrupt flags interface the module with the PIC[®] microcontroller and user software. Two pins, SDA and SCL, are exercised by the module to communicate with other external I²C devices.

27.8.1 BYTE FORMAT

All communication in I²C is done in 9-bit segments. A byte is sent from a master to a slave or vice-versa, followed by an Acknowledge bit sent back. After the eighth falling edge of the SCL line, the device outputting data on the SDA changes that pin to an input and reads in an acknowledge value on the next clock pulse.

The clock signal, SCL, is provided by the master. Data is valid to change while the SCL signal is low, and sampled on the rising edge of the clock. Changes on the SDA line while the SCL line is high define special conditions on the bus, explained below.

27.8.2 DEFINITION OF I²C TERMINOLOGY

There is language and terminology in the description of I²C communication that have definitions specific to I²C. That word usage is defined below and may be used in the rest of this document without explanation. This table was adapted from the Philips I²C specification.

27.8.3 SDA AND SCL PINS

Selection of any I²C mode with the SSPEN bit set, forces the SCL and SDA pins to be open-drain. These pins should be set by the user to inputs by setting the appropriate TRIS bits.

Note 1: Data is tied to output zero when an I²C mode is enabled.

- 2:** Any device pin can be selected for SDA and SCL functions with the PPS peripheral. These functions are bidirectional. The SDA input is selected with the SSPxDATPPS registers. The SCL input is selected with the SSPxCLKPPS registers. Outputs are selected with the RxyPPS registers. It is the user's responsibility to make the selections so that both the input and the output for each function is on the same pin.

27.8.4 SDA HOLD TIME

The hold time of the SDA pin is selected by the SDAHT bit of the SSPxCON3 register. Hold time is the time SDA is held valid after the falling edge of SCL. Setting the SDAHT bit selects a longer 300 ns minimum hold time and may help on buses with large capacitance.

TABLE 27-2: I²C BUS TERMS

| TERM | Description |
|------------------|---|
| Transmitter | The device which shifts data out onto the bus. |
| Receiver | The device which shifts data in from the bus. |
| Master | The device that initiates a transfer, generates clock signals and terminates a transfer. |
| Slave | The device addressed by the master. |
| Multi-master | A bus with more than one device that can initiate data transfers. |
| Arbitration | Procedure to ensure that only one master at a time controls the bus. Winning arbitration ensures that the message is not corrupted. |
| Synchronization | Procedure to synchronize the clocks of two or more devices on the bus. |
| Idle | No master is controlling the bus, and both SDA and SCL lines are high. |
| Active | Any time one or more master devices are controlling the bus. |
| Addressed Slave | Slave device that has received a matching address and is actively being clocked by a master. |
| Matching Address | Address byte that is clocked into a slave that matches the value stored in SSPxADD. |
| Write Request | Slave receives a matching address with R/W bit clear, and is ready to clock in data. |
| Read Request | Master sends an address byte with the R/W bit set, indicating that it wishes to clock data out of the Slave. This data is the next and all following bytes until a Restart or Stop. |
| Clock Stretching | When a device on the bus hold SCL low to stall communication. |
| Bus Collision | Any time the SDA line is sampled low by the module while it is outputting and expected high state. |

27.8.5 START CONDITION

The I²C specification defines a Start condition as a transition of SDA from a high to a low state while SCL line is high. A Start condition is always generated by the master and signifies the transition of the bus from an Idle to an Active state. Figure 27-12 shows wave forms for Start and Stop conditions.

A bus collision can occur on a Start condition if the module samples the SDA line low before asserting it low. This does not conform to the I²C Specification that states no bus collision can occur on a Start.

27.8.6 STOP CONDITION

A Stop condition is a transition of the SDA line from low-to-high state while the SCL line is high.

Note: At least one SCL low time must appear before a Stop is valid, therefore, if the SDA line goes low then high again while the SCL line stays high, only the Start condition is detected.

27.8.7 RESTART CONDITION

A Restart is valid any time that a Stop would be valid. A master can issue a Restart if it wishes to hold the bus after terminating the current transfer. A Restart has the same effect on the slave that a Start would, resetting all slave logic and preparing it to clock in an address. The master may want to address the same or another slave. Figure 27-13 shows the wave form for a Restart condition.

In 10-bit Addressing Slave mode a Restart is required for the master to clock data out of the addressed slave. Once a slave has been fully addressed, matching both high and low address bytes, the master can issue a Restart and the high address byte with the R/W bit set. The slave logic will then hold the clock and prepare to clock out data.

After a full match with $\overline{R/W}$ clear in 10-bit mode, a prior match flag is set and maintained until a Stop condition, a high address with $\overline{R/W}$ clear, or high address match fails.

27.8.8 START/STOP CONDITION INTERRUPT MASKING

The SCIE and PCIE bits of the SSPxCON3 register can enable the generation of an interrupt in Slave modes that do not typically support this function. Slave modes where interrupt on Start and Stop detect are already enabled, these bits will have no effect.

FIGURE 27-12: I²C START AND STOP CONDITIONS

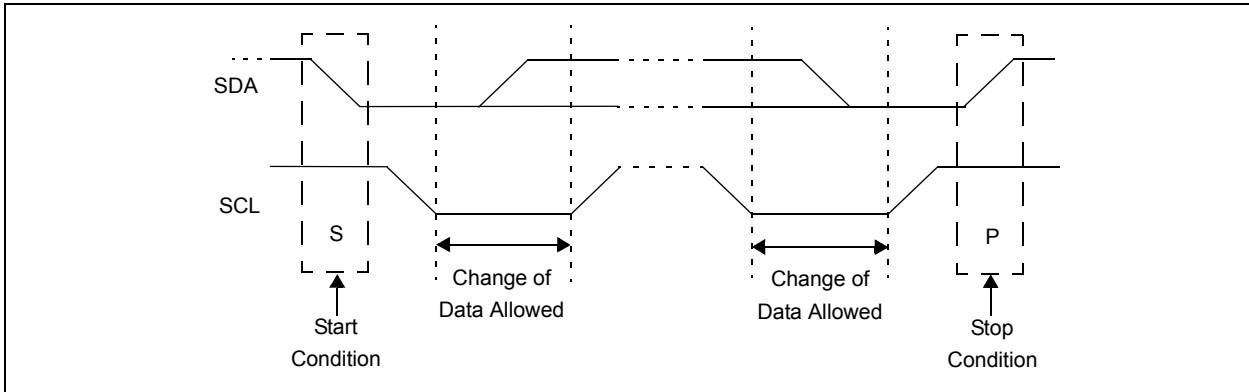
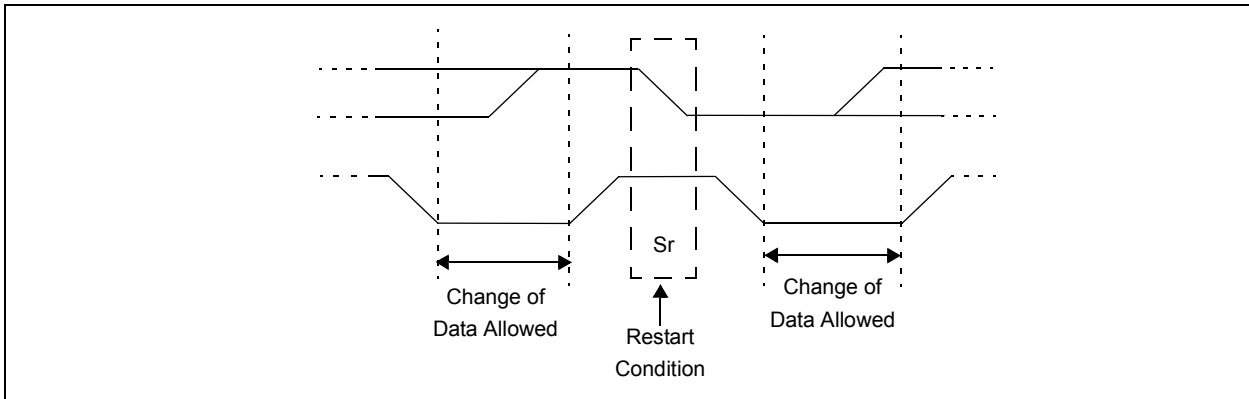


FIGURE 27-13: I²C RESTART CONDITION



27.8.9 ACKNOWLEDGE SEQUENCE

The ninth SCL pulse for any transferred byte in I²C is dedicated as an Acknowledge. It allows receiving devices to respond back to the transmitter by pulling the SDA line low. The transmitter must release control of the line during this time to shift in the response. The Acknowledge ($\overline{\text{ACK}}$) is an active-low signal, pulling the SDA line low indicates to the transmitter that the device has received the transmitted data and is ready to receive more.

The result of an $\overline{\text{ACK}}$ is placed in the ACKSTAT bit of the SSPxCON2 register.

Slave software, when the AHEN and DHEN bits are set, allow the user to set the $\overline{\text{ACK}}$ value sent back to the transmitter. The ACKDT bit of the SSPxCON2 register is set/cleared to determine the response.

Slave hardware will generate an $\overline{\text{ACK}}$ response if the AHEN and DHEN bits of the SSPxCON3 register are clear.

There are certain conditions where an $\overline{\text{ACK}}$ will not be sent by the slave. If the BF bit of the SSPxSTAT register or the SSPOV bit of the SSPxCON1 register are set when a byte is received.

When the module is addressed, after the eighth falling edge of SCL on the bus, the ACKTIM bit of the SSPxCON3 register is set. The ACKTIM bit indicates the acknowledge time of the active bus. The ACKTIM Status bit is only active when the AHEN bit or DHEN bit is enabled.

27.9 I²C Slave Mode Operation

The MSSP Slave mode operates in one of four modes selected by the SSPM bits of the SSPxCON1 register. The modes can be divided into 7-bit and 10-bit Addressing mode. 10-bit Addressing modes operate the same as 7-bit with some additional overhead for handling the larger addresses.

Modes with Start and Stop bit interrupts operate the same as the other modes with SSPxIF additionally getting set upon detection of a Start, Restart, or Stop condition.

27.9.1 SLAVE MODE ADDRESSES

The SSPxADD register ([Register 27-5](#)) contains the Slave mode address. The first byte received after a Start or Restart condition is compared against the value stored in this register. If the byte matches, the value is loaded into the SSPxBUF register and an interrupt is generated. If the value does not match, the module goes idle and no indication is given to the software that anything happened.

The SSP Mask register affects the address matching process. See [Section 27.9.9 “SSP Mask Register”](#) for more information.

27.9.1.1 I²C Slave 7-bit Addressing Mode

In 7-bit Addressing mode, the LSB of the received data byte is ignored when determining if there is an address match.

27.9.1.2 I²C Slave 10-bit Addressing Mode

In 10-bit Addressing mode, the first received byte is compared to the binary value of ‘1 1 1 1 0 A9 A8 0’. A9 and A8 are the two MSB’s of the 10-bit address and stored in bits 2 and 1 of the SSPxADD register.

After the acknowledge of the high byte the UA bit is set and SCL is held low until the user updates SSPxADD with the low address. The low address byte is clocked in and all eight bits are compared to the low address value in SSPxADD. Even if there is not an address match; SSPxIF and UA are set, and SCL is held low until SSPxADD is updated to receive a high byte again. When SSPxADD is updated the UA bit is cleared. This ensures the module is ready to receive the high address byte on the next communication.

A high and low address match as a write request is required at the start of all 10-bit addressing communication. A transmission can be initiated by issuing a Restart once the slave is addressed, and clocking in the high address with the R $\overline{\text{W}}$ bit set. The slave hardware will then acknowledge the read request and prepare to clock out data. This is only valid for a slave after it has received a complete high and low address byte match.

27.9.2 SLAVE RECEPTION

When the R $\overline{\text{W}}$ bit of a matching received address byte is clear, the R $\overline{\text{W}}$ bit of the SSPxSTAT register is cleared. The received address is loaded into the SSPxBUF register and acknowledged.

When the overflow condition exists for a received address, then not Acknowledge is given. An overflow condition is defined as either bit BF of the SSPxSTAT register is set, or bit SSPOV of the SSPxCON1 register is set. The BOEN bit of the SSPxCON3 register modifies this operation. For more information see [Register 27-3](#).

An MSSP interrupt is generated for each transferred data byte. Flag bit, SSPxIF, must be cleared by software.

When the SEN bit of the SSPxCON2 register is set, SCL will be held low (clock stretch) following each received byte. The clock must be released by setting the CKP bit of the SSPxCON1 register, except sometimes in 10-bit mode. See [Section 27.9.6.2 “10-bit Addressing Mode”](#) for more detail.

27.9.2.1 7-bit Addressing Reception

This section describes a standard sequence of events for the MSSP module configured as an I²C slave in 7-bit Addressing mode. Figure 27-14 and Figure 27-15 is used as a visual reference for this description.

This is a step by step process of what typically must be done to accomplish I²C communication.

1. Start bit detected.
2. S bit of SSPxSTAT is set; SSPxIF is set if interrupt on Start detect is enabled.
3. Matching address with R/W bit clear is received.
4. The slave pulls SDA low sending an $\overline{\text{ACK}}$ to the master, and sets SSPxIF bit.
5. Software clears the SSPxIF bit.
6. Software reads received address from SSPxBUF clearing the BF flag.
7. If SEN = 1; Slave software sets CKP bit to release the SCL line.
8. The master clocks out a data byte.
9. Slave drives SDA low sending an $\overline{\text{ACK}}$ to the master, and sets SSPxIF bit.
10. Software clears SSPxIF.
11. Software reads the received byte from SSPxBUF clearing BF.
12. Steps 8-12 are repeated for all received bytes from the master.
13. Master sends Stop condition, setting P bit of SSPxSTAT, and the bus goes idle.

27.9.2.2 7-bit Reception with AHEN and DHEN

Slave device reception with AHEN and DHEN set operate the same as without these options with extra interrupts and clock stretching added after the eighth falling edge of SCL. These additional interrupts allow the slave software to decide whether it wants to $\overline{\text{ACK}}$ the receive address or data byte, rather than the hardware. This functionality adds support for PMBus™ that was not present on previous versions of this module.

This list describes the steps that need to be taken by slave software to use these options for I²C communication. Figure 27-16 displays a module using both address and data holding. Figure 27-17 includes the operation with the SEN bit of the SSPxCON2 register set.

1. S bit of SSPxSTAT is set; SSPxIF is set if interrupt on Start detect is enabled.
2. Matching address with R/W bit clear is clocked in. SSPxIF is set and CKP cleared after the eighth falling edge of SCL.
3. Slave clears the SSPxIF.
4. Slave can look at the ACKTIM bit of the SSPxCON3 register to determine if the SSPxIF was after or before the $\overline{\text{ACK}}$.
5. Slave reads the address value from SSPxBUF, clearing the BF flag.
6. Slave sets $\overline{\text{ACK}}$ value clocked out to the master by setting ACKDT.
7. Slave releases the clock by setting CKP.
8. SSPxIF is set after an $\overline{\text{ACK}}$, not after a NACK.
9. If SEN = 1 the slave hardware will stretch the clock after the $\overline{\text{ACK}}$.
10. Slave clears SSPxIF.

Note: SSPxIF is still set after the ninth falling edge of SCL even if there is no clock stretching and BF has been cleared. Only if NACK is sent to master is SSPxIF not set

11. SSPxIF set and CKP cleared after eighth falling edge of SCL for a received data byte.
12. Slave looks at ACKTIM bit of SSPxCON3 to determine the source of the interrupt.
13. Slave reads the received data from SSPxBUF clearing BF.
14. Steps 7-14 are the same for each received data byte.
15. Communication is ended by either the slave sending an $\overline{\text{ACK}} = 1$, or the master sending a Stop condition. If a Stop is sent and Interrupt on Stop Detect is disabled, the slave will only know by polling the P bit of the SSTSTAT register.

FIGURE 27-14: I²C SLAVE, 7-BIT ADDRESS, RECEPTION (SEN = 0, AHEN = 0, DHEN = 0)

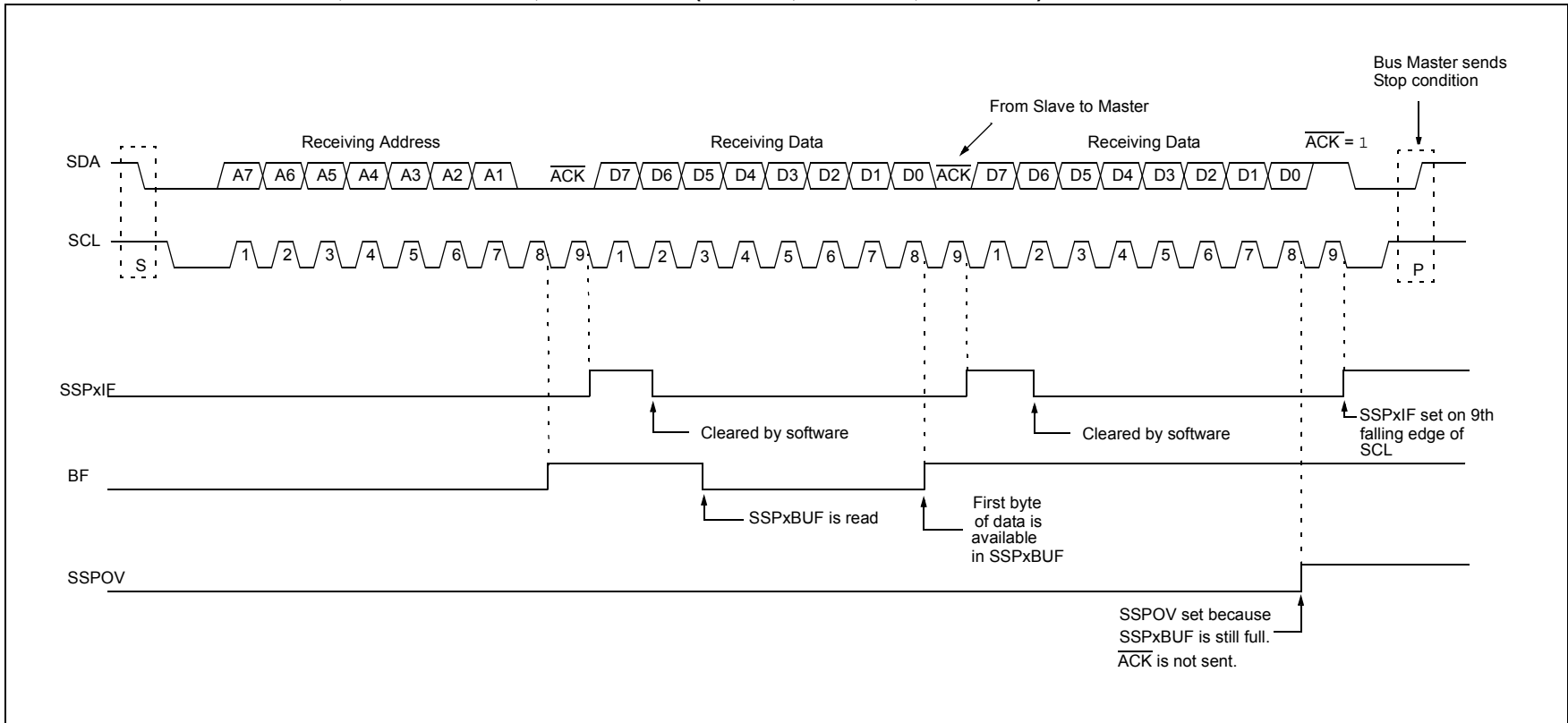


FIGURE 27-15: I²C SLAVE, 7-BIT ADDRESS, RECEPTION (SEN = 1, AHEN = 0, DHEN = 0)

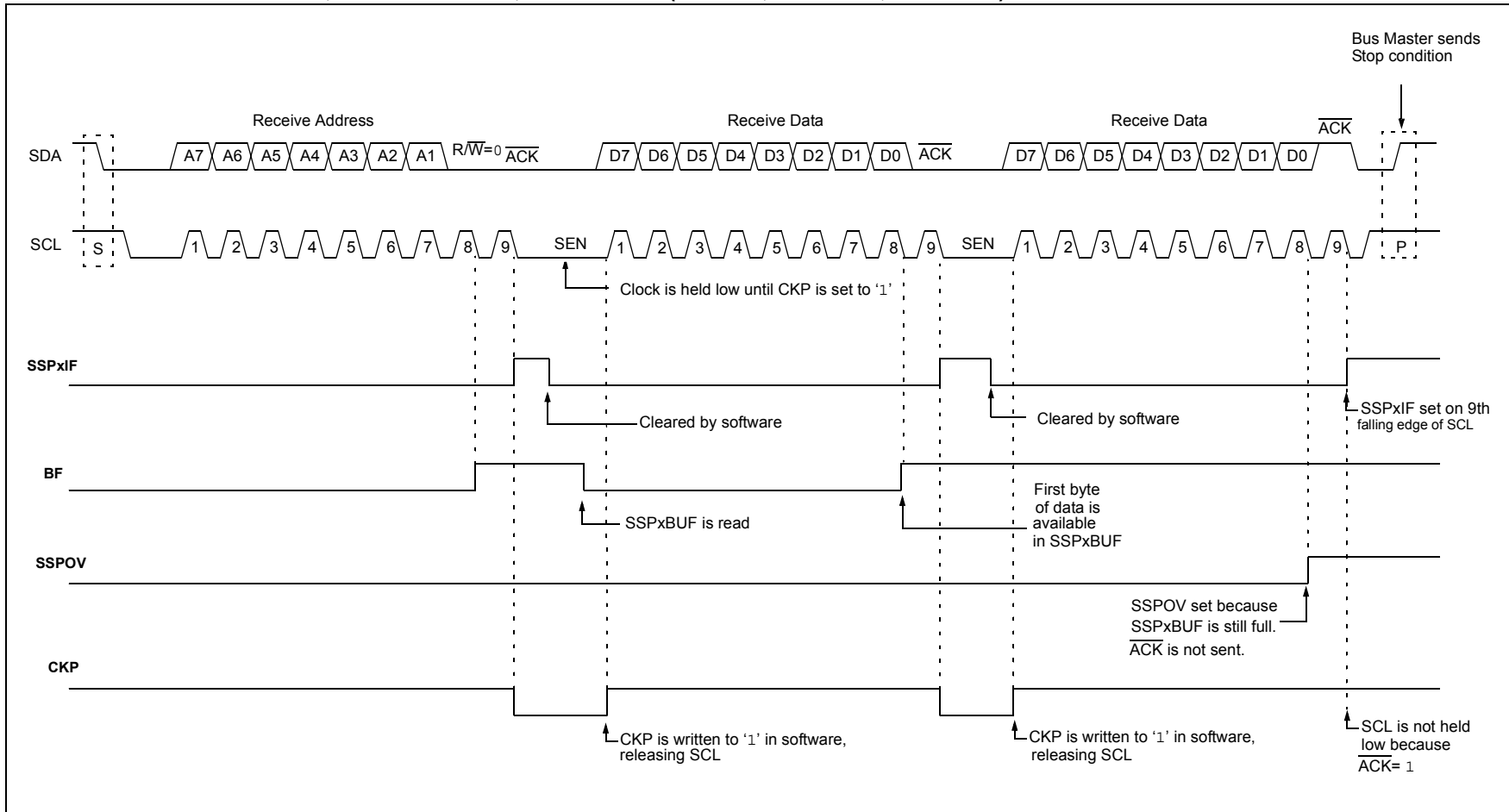


FIGURE 27-16: I²C SLAVE, 7-BIT ADDRESS, RECEPTION (SEN = 0, AHEN = 1, DHEN = 1)

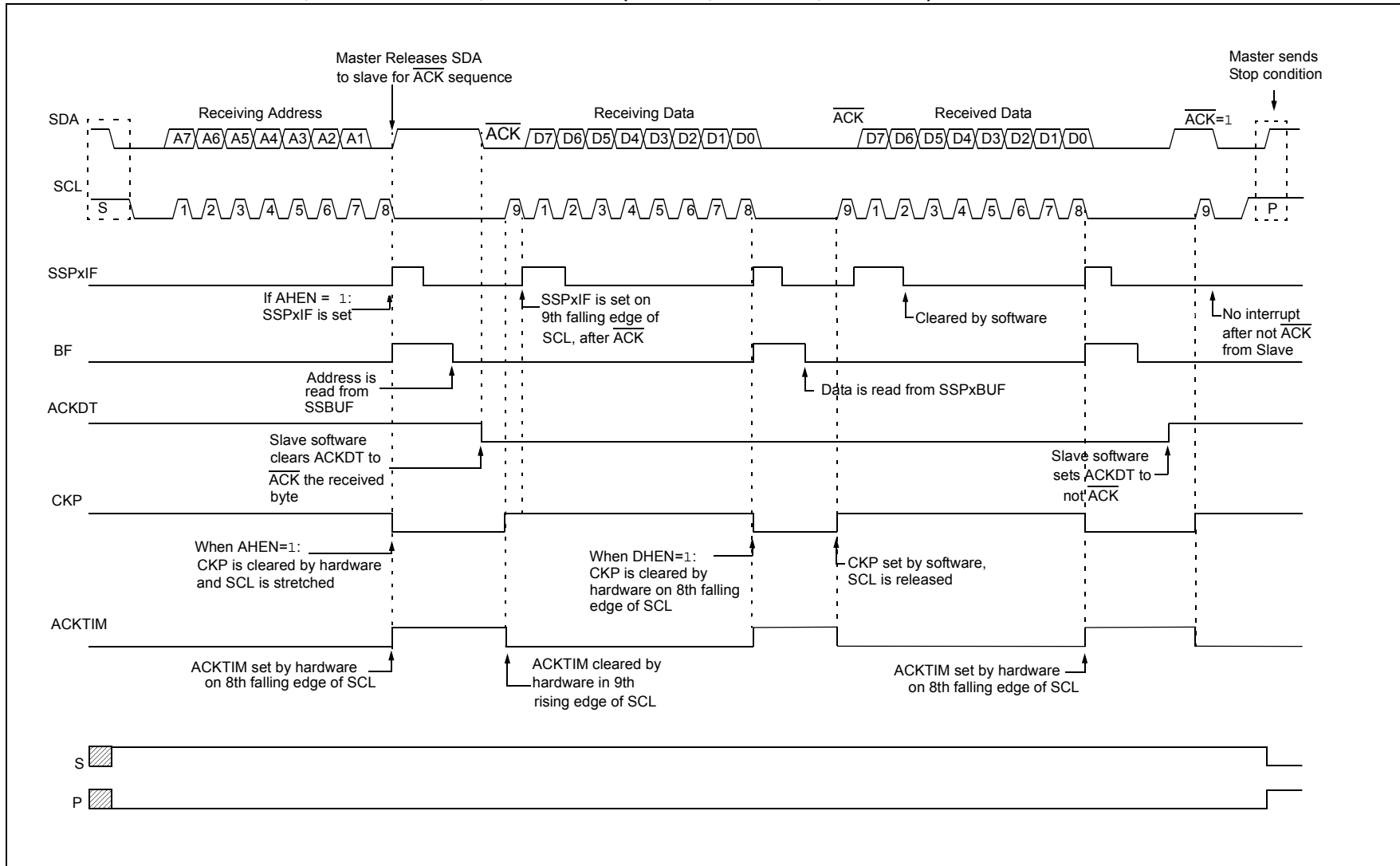
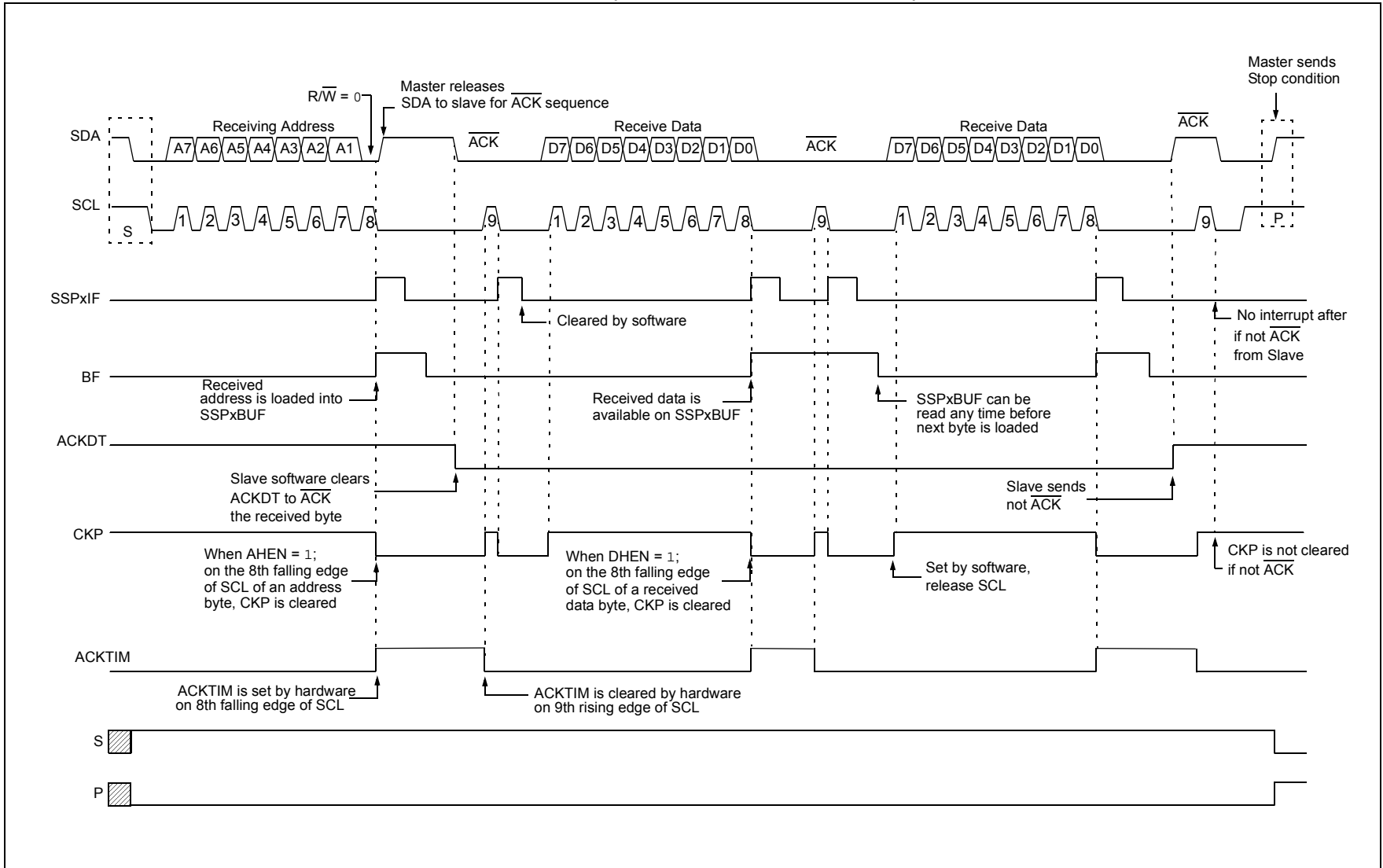


FIGURE 27-17: I²C SLAVE, 7-BIT ADDRESS, RECEPTION (SEN = 1, AHEN = 1, DHEN = 1)



27.9.3 SLAVE TRANSMISSION

When the $\overline{R/W}$ bit of the incoming address byte is set and an address match occurs, the $\overline{R/W}$ bit of the SSPxSTAT register is set. The received address is loaded into the SSPxBUF register, and an \overline{ACK} pulse is sent by the slave on the ninth bit.

Following the \overline{ACK} , slave hardware clears the CKP bit and the SCL pin is held low (see [Section 27.9.6 “Clock Stretching”](#) for more detail). By stretching the clock, the master will be unable to assert another clock pulse until the slave is done preparing the transmit data.

The transmit data must be loaded into the SSPxBUF register which also loads the SSPSR register. Then the SCL pin should be released by setting the CKP bit of the SSPxCON1 register. The eight data bits are shifted out on the falling edge of the SCL input. This ensures that the SDA signal is valid during the SCL high time.

The \overline{ACK} pulse from the master-receiver is latched on the rising edge of the ninth SCL input pulse. This \overline{ACK} value is copied to the ACKSTAT bit of the SSPxCON2 register. If ACKSTAT is set (not \overline{ACK}), then the data transfer is complete. In this case, when the not \overline{ACK} is latched by the slave, the slave goes idle and waits for another occurrence of the Start bit. If the SDA line was low (\overline{ACK}), the next transmit data must be loaded into the SSPxBUF register. Again, the SCL pin must be released by setting bit CKP.

An MSSP interrupt is generated for each data transfer byte. The SSPxIF bit must be cleared by software and the SSPxSTAT register is used to determine the status of the byte. The SSPxIF bit is set on the falling edge of the ninth clock pulse.

27.9.3.1 Slave Mode Bus Collision

A slave receives a Read request and begins shifting data out on the SDA line. If a bus collision is detected and the SBCDE bit of the SSPxCON3 register is set, the BCLxIF bit of the PIR register is set. Once a bus collision is detected, the slave goes idle and waits to be addressed again. User software can use the BCLxIF bit to handle a slave bus collision.

27.9.3.2 7-bit Transmission

A master device can transmit a read request to a slave, and then clock data out of the slave. The list below outlines what software for a slave will need to do to accomplish a standard transmission. [Figure 27-18](#) can be used as a reference to this list.

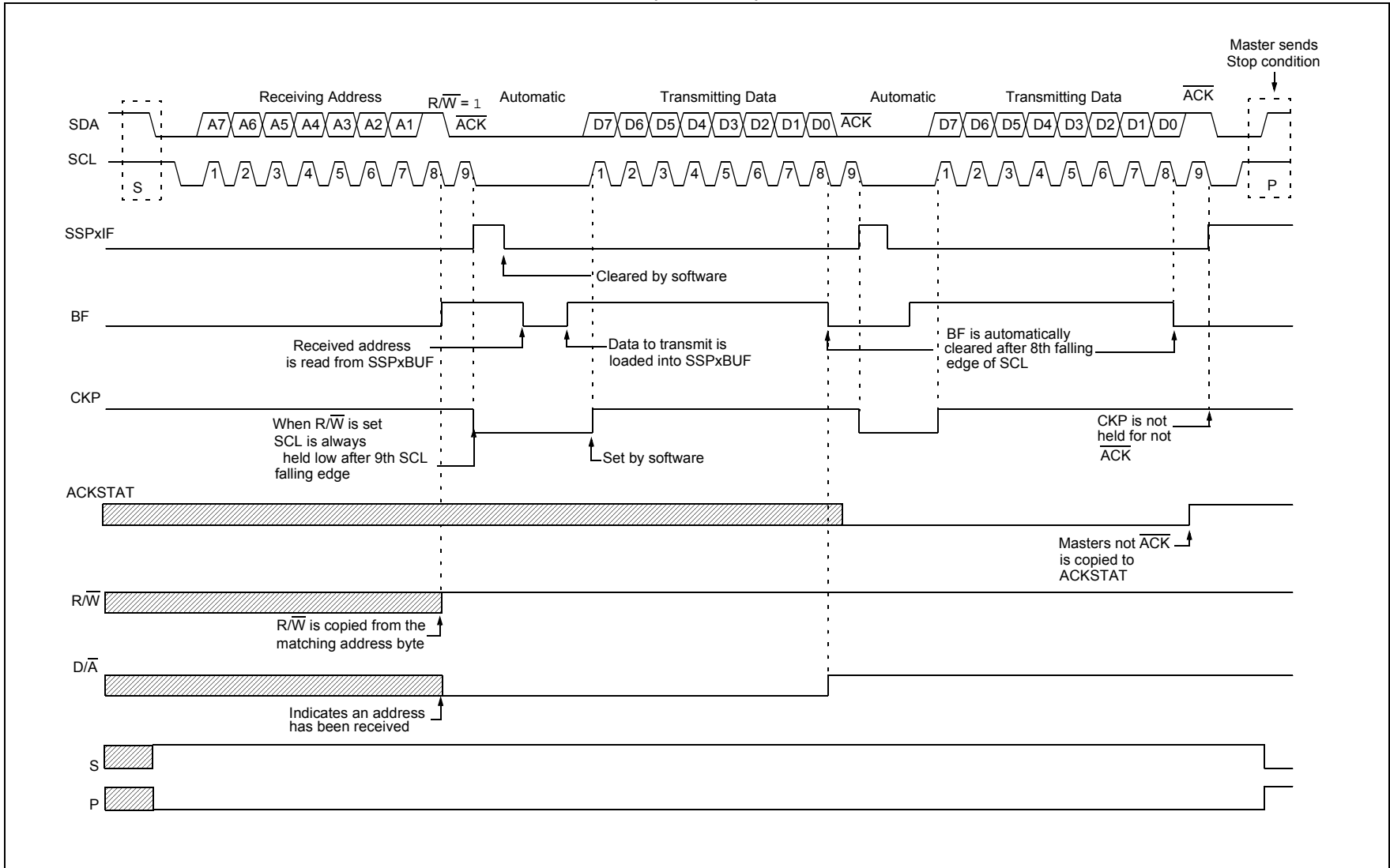
1. Master sends a Start condition on SDA and SCL.
2. S bit of SSPxSTAT is set; SSPxIF is set if interrupt on Start detect is enabled.
3. Matching address with $\overline{R/W}$ bit set is received by the Slave setting SSPxIF bit.
4. Slave hardware generates an \overline{ACK} and sets SSPxIF.
5. SSPxIF bit is cleared by user.
6. Software reads the received address from SSPxBUF, clearing BF.
7. $\overline{R/W}$ is set so CKP was automatically cleared after the \overline{ACK} .
8. The slave software loads the transmit data into SSPxBUF.
9. CKP bit is set releasing SCL, allowing the master to clock the data out of the slave.
10. SSPxIF is set after the \overline{ACK} response from the master is loaded into the ACKSTAT register.
11. SSPxIF bit is cleared.
12. The slave software checks the ACKSTAT bit to see if the master wants to clock out more data.

Note 1: If the master \overline{ACK} s the clock will be stretched.

2: ACKSTAT is the only bit updated on the rising edge of SCL (9th) rather than the falling.

13. Steps 9-13 are repeated for each transmitted byte.
14. If the master sends a not \overline{ACK} ; the clock is not held, but SSPxIF is still set.
15. The master sends a Restart condition or a Stop.
16. The slave is no longer addressed.

FIGURE 27-18: I²C SLAVE, 7-BIT ADDRESS, TRANSMISSION (AHEN = 0)



27.9.3.3 7-bit Transmission with Address Hold Enabled

Setting the AHEN bit of the SSPxCON3 register enables additional clock stretching and interrupt generation after the eighth falling edge of a received matching address. Once a matching address has been clocked in, CKP is cleared and the SSPxIF interrupt is set.

Figure 27-19 displays a standard waveform of a 7-bit address slave transmission with AHEN enabled.

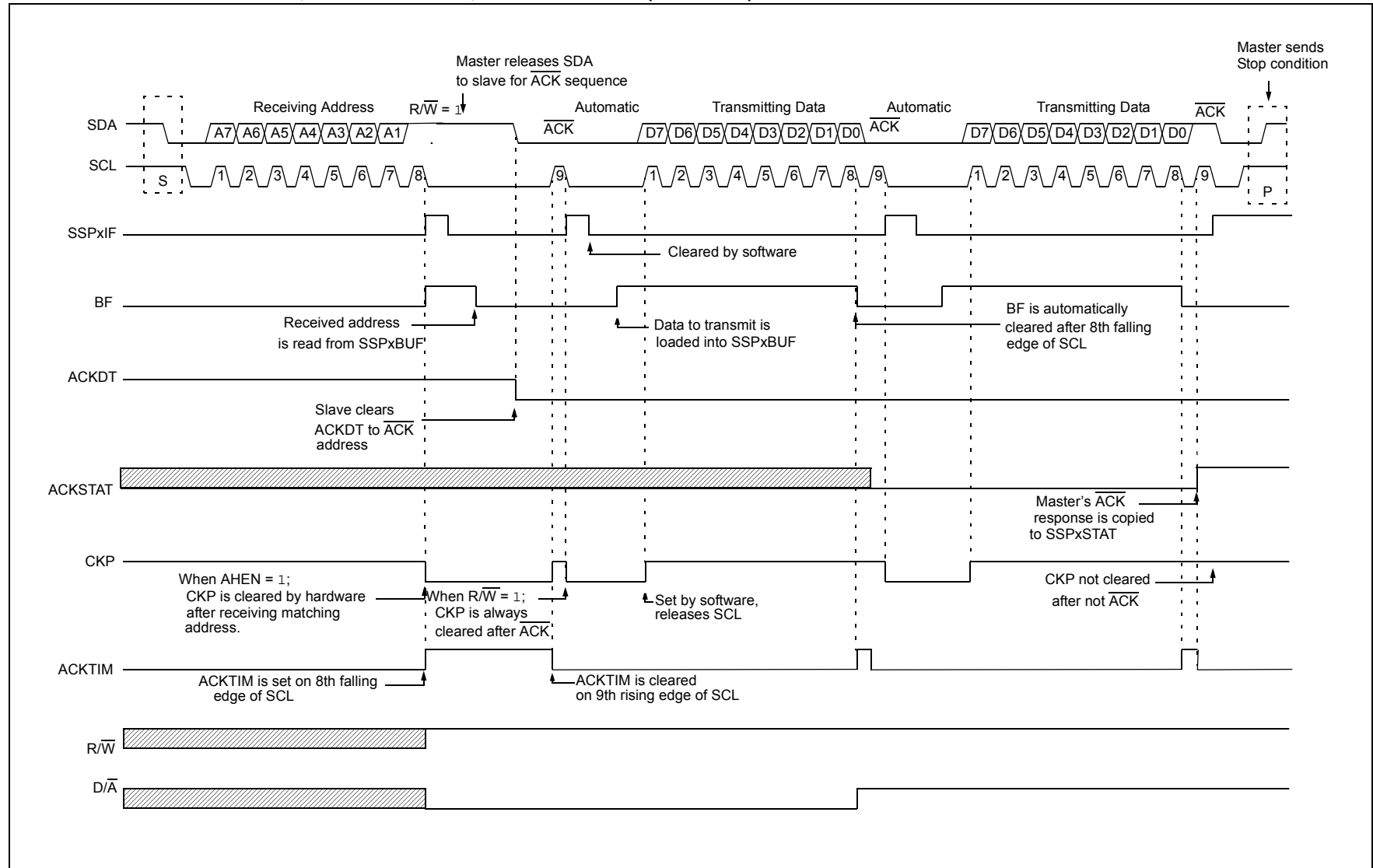
1. Bus starts Idle.
2. Master sends Start condition; the S bit of SSPxSTAT is set; SSPxIF is set if interrupt on Start detect is enabled.
3. Master sends matching address with $\overline{R/W}$ bit set. After the eighth falling edge of the SCL line the CKP bit is cleared and SSPxIF interrupt is generated.
4. Slave software clears SSPxIF.
5. Slave software reads the \overline{ACKTIM} bit of SSPxCON3 register, and $\overline{R/W}$ and $\overline{D/A}$ of the SSPxSTAT register to determine the source of the interrupt.
6. Slave reads the address value from the SSPxBUF register clearing the BF bit.
7. Slave software decides from this information if it wishes to \overline{ACK} or not \overline{ACK} and sets the ACKDT bit of the SSPxCON2 register accordingly.
8. Slave sets the CKP bit releasing SCL.
9. Master clocks in the \overline{ACK} value from the slave.
10. Slave hardware automatically clears the CKP bit and sets SSPxIF after the \overline{ACK} if the $\overline{R/W}$ bit is set.
11. Slave software clears SSPxIF.
12. Slave loads value to transmit to the master into SSPxBUF setting the BF bit.

Note: SSPxBUF cannot be loaded until after the \overline{ACK} .

13. Slave sets the CKP bit releasing the clock.
14. Master clocks out the data from the slave and sends an \overline{ACK} value on the ninth SCL pulse.
15. Slave hardware copies the \overline{ACK} value into the ACKSTAT bit of the SSPxCON2 register.
16. Steps 10-15 are repeated for each byte transmitted to the master from the slave.
17. If the master sends a not \overline{ACK} the slave releases the bus allowing the master to send a Stop and end the communication.

Note: Master must send a not \overline{ACK} on the last byte to ensure that the slave releases the SCL line to receive a Stop.

FIGURE 27-19: I²C SLAVE, 7-BIT ADDRESS, TRANSMISSION (AHEN = 1)



27.9.4 SLAVE MODE 10-BIT ADDRESS RECEPTION

This section describes a standard sequence of events for the MSSP module configured as an I²C slave in 10-bit Addressing mode.

Figure 27-20 is used as a visual reference for this description.

This is a step by step process of what must be done by slave software to accomplish I²C communication.

1. Bus starts Idle.
2. Master sends Start condition; S bit of SSPxSTAT is set; SSPxIF is set if interrupt on Start detect is enabled.
3. Master sends matching high address with R/W bit clear; UA bit of the SSPxSTAT register is set.
4. Slave sends $\overline{\text{ACK}}$ and SSPxIF is set.
5. Software clears the SSPxIF bit.
6. Software reads received address from SSPxBUF clearing the BF flag.
7. Slave loads low address into SSPxADD, releasing SCL.
8. Master sends matching low address byte to the slave; UA bit is set.

Note: Updates to the SSPxADD register are not allowed until after the ACK sequence.

9. Slave sends $\overline{\text{ACK}}$ and SSPxIF is set.

Note: If the low address does not match, SSPxIF and UA are still set so that the slave software can set SSPxADD back to the high address. BF is not set because there is no match. CKP is unaffected.

10. Slave clears SSPxIF.
11. Slave reads the received matching address from SSPxBUF clearing BF.
12. Slave loads high address into SSPxADD.
13. Master clocks a data byte to the slave and clocks out the slaves $\overline{\text{ACK}}$ on the ninth SCL pulse; SSPxIF is set.
14. If SEN bit of SSPxCON2 is set, CKP is cleared by hardware and the clock is stretched.
15. Slave clears SSPxIF.
16. Slave reads the received byte from SSPxBUF clearing BF.
17. If SEN is set the slave sets CKP to release the SCL.
18. Steps 13-17 repeat for each received byte.
19. Master sends Stop to end the transmission.

27.9.5 10-BIT ADDRESSING WITH ADDRESS OR DATA HOLD

Reception using 10-bit addressing with AHEN or DHEN set is the same as with 7-bit modes. The only difference is the need to update the SSPxADD register using the UA bit. All functionality, specifically when the CKP bit is cleared and SCL line is held low are the same. Figure 27-21 can be used as a reference of a slave in 10-bit addressing with AHEN set.

Figure 27-22 shows a standard waveform for a slave transmitter in 10-bit Addressing mode.

FIGURE 27-20: I²C SLAVE, 10-BIT ADDRESS, RECEPTION (SEN = 1, AHEN = 0, DHEN = 0)

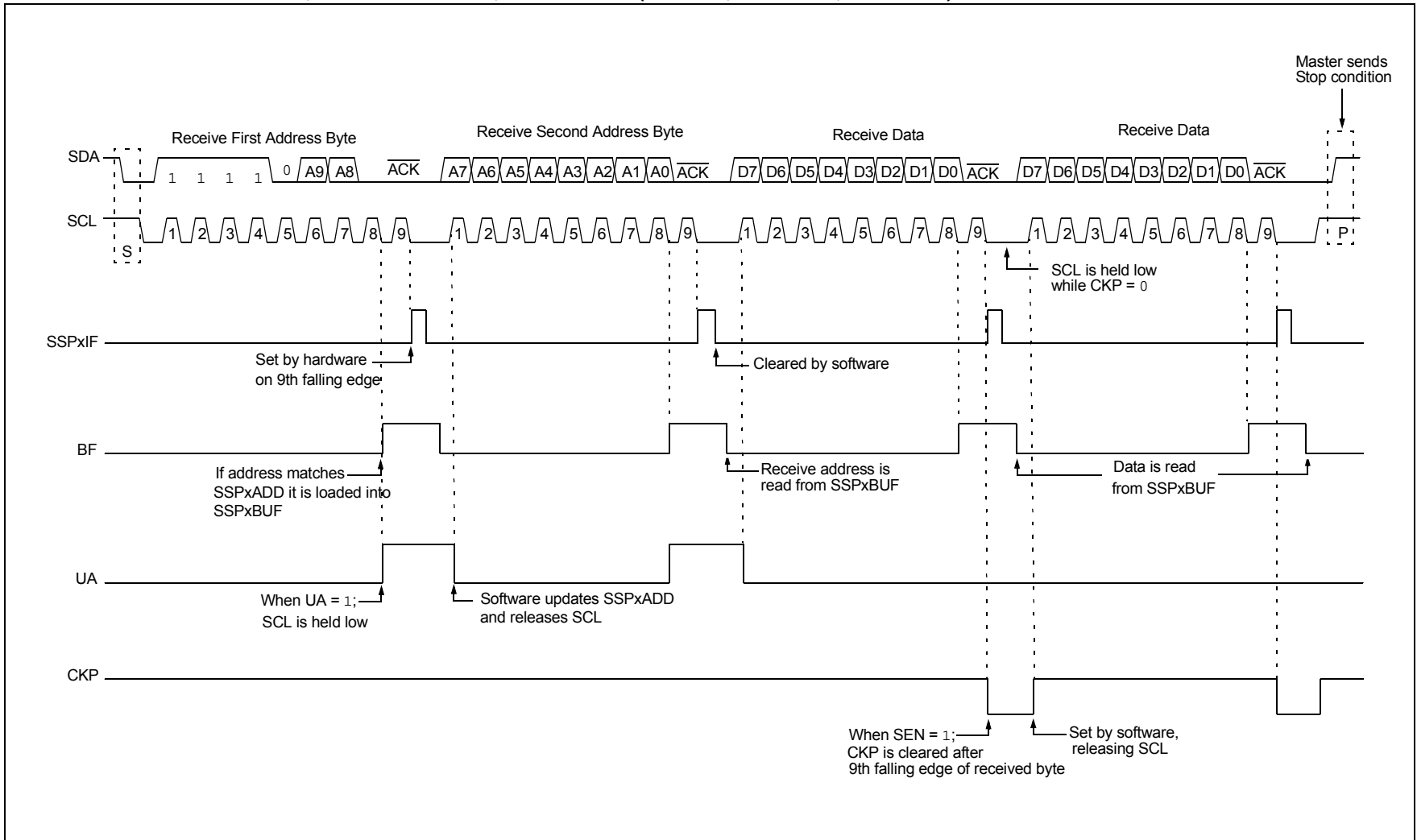


FIGURE 27-21: I²C SLAVE, 10-BIT ADDRESS, RECEPTION (SEN = 0, AHEN = 1, DHEN = 0)

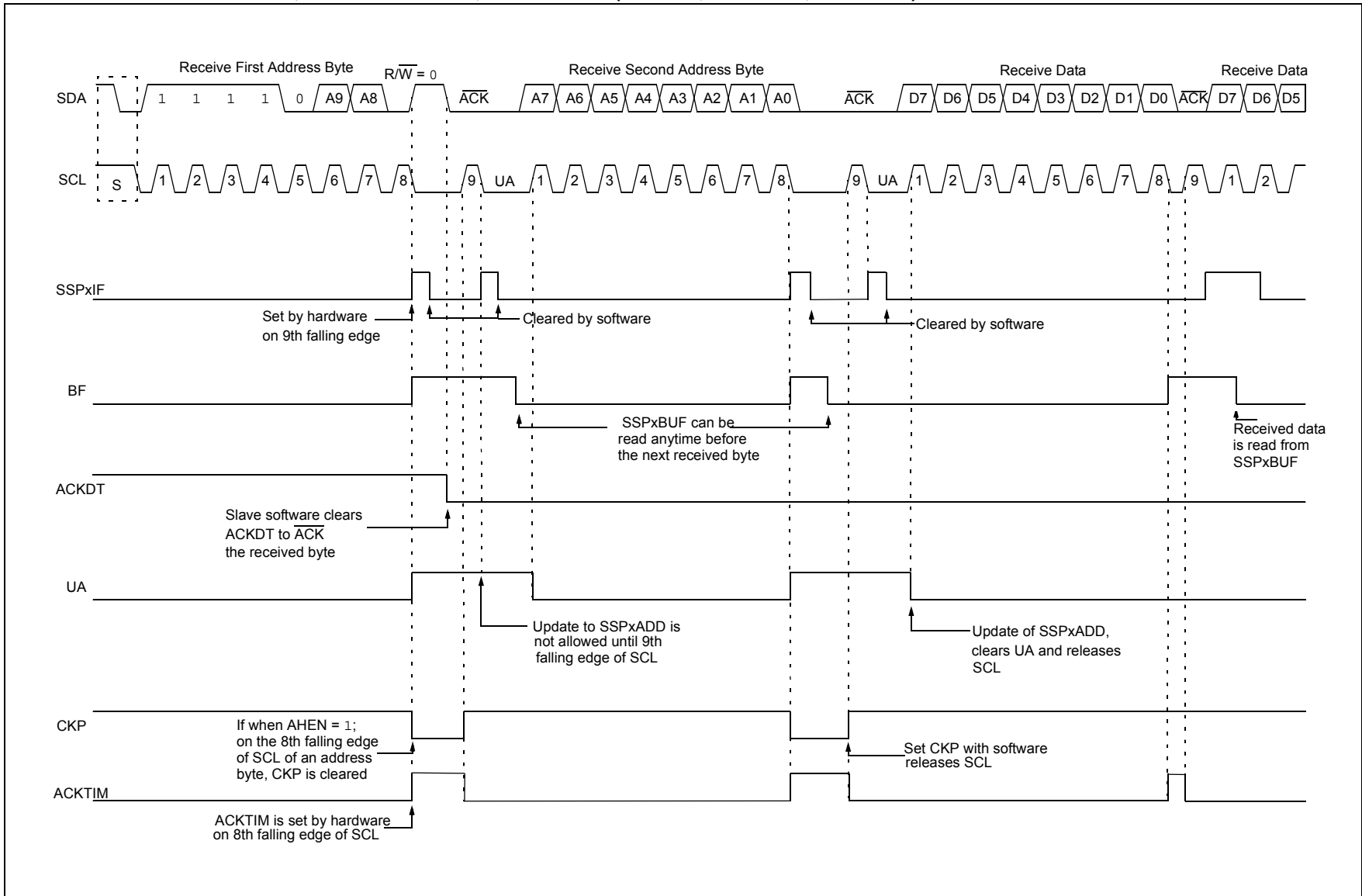
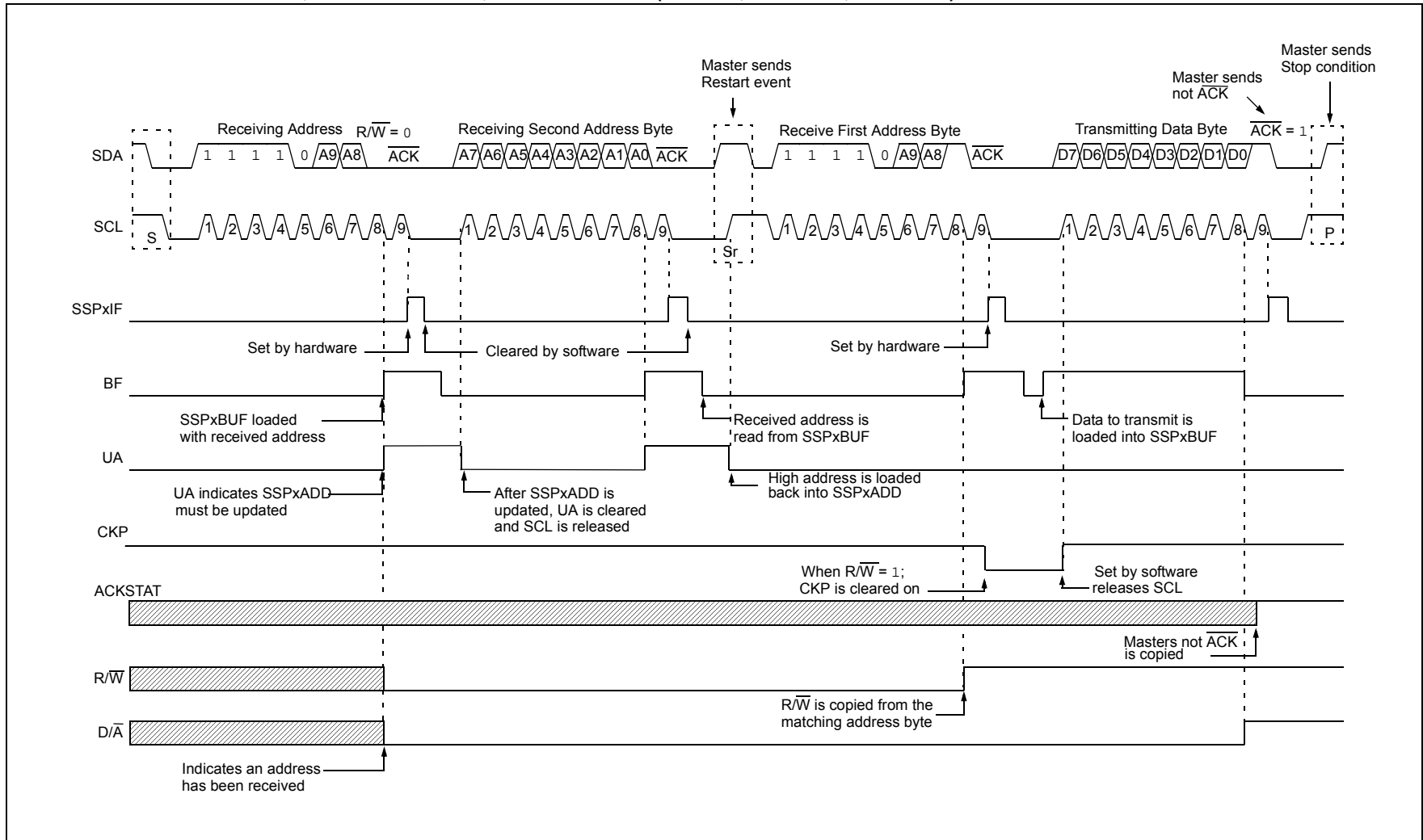


FIGURE 27-22: I²C SLAVE, 10-BIT ADDRESS, TRANSMISSION (SEN = 0, AHEN = 0, DHEN = 0)



27.9.6 CLOCK STRETCHING

Clock stretching occurs when a device on the bus holds the SCL line low, effectively pausing communication. The slave may stretch the clock to allow more time to handle data or prepare a response for the master device. A master device is not concerned with stretching as anytime it is active on the bus and not transferring data it is stretching. Any stretching done by a slave is invisible to the master software and handled by the hardware that generates SCL.

The CKP bit of the SSPxCON1 register is used to control stretching in software. Any time the CKP bit is cleared, the module will wait for the SCL line to go low and then hold it. Setting CKP will release SCL and allow more communication.

27.9.6.1 Normal Clock Stretching

Following an $\overline{\text{ACK}}$ if the $\overline{\text{R/W}}$ bit of SSPxSTAT is set, a read request, the slave hardware will clear CKP. This allows the slave time to update SSPxBUF with data to transfer to the master. If the SEN bit of SSPxCON2 is set, the slave hardware will always stretch the clock after the ACK sequence. Once the slave is ready, CKP is set by software and communication resumes.

Note 1: The BF bit has no effect on if the clock will be stretched or not. This is different than previous versions of the module that would not stretch the clock, clear CKP, if SSPxBUF was read before the ninth falling edge of SCL.

2: Previous versions of the module did not stretch the clock for a transmission if SSPxBUF was loaded before the ninth falling edge of SCL. It is now always cleared for read requests.

27.9.6.2 10-bit Addressing Mode

In 10-bit Addressing mode, when the UA bit is set, the clock is always stretched. This is the only time the SCL is stretched without CKP being cleared. SCL is released immediately after a write to SSPxADD.

Note: Previous versions of the module did not stretch the clock if the second address byte did not match.

27.9.6.3 Byte NACKing

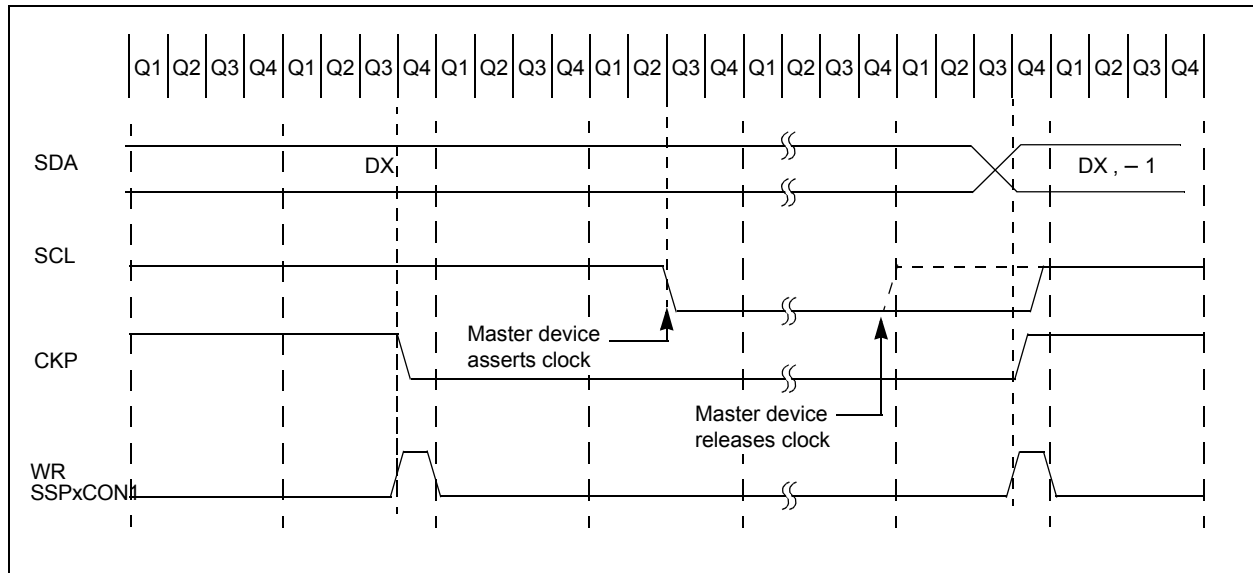
When the AHEN bit of SSPxCON3 is set; CKP is cleared by hardware after the eighth falling edge of SCL for a received matching address byte. When the DHEN bit of SSPxCON3 is set; CKP is cleared after the eighth falling edge of SCL for received data.

Stretching after the eighth falling edge of SCL allows the slave to look at the received address or data and decide if it wants to ACK the received data.

27.9.7 CLOCK SYNCHRONIZATION AND THE CKP BIT

Any time the CKP bit is cleared, the module will wait for the SCL line to go low and then hold it. However, clearing the CKP bit will not assert the SCL output low until the SCL output is already sampled low. Therefore, the CKP bit will not assert the SCL line until an external I²C master device has already asserted the SCL line. The SCL output will remain low until the CKP bit is set and all other devices on the I²C bus have released SCL. This ensures that a write to the CKP bit will not violate the minimum high time requirement for SCL (see Figure 27-23).

FIGURE 27-23: CLOCK SYNCHRONIZATION TIMING



27.9.8 GENERAL CALL ADDRESS SUPPORT

The addressing procedure for the I²C bus is such that the first byte after the Start condition usually determines which device will be the slave addressed by the master device. The exception is the general call address which can address all devices. When this address is used, all devices should, in theory, respond with an acknowledge.

The general call address is a reserved address in the I²C protocol, defined as address 0x00. When the GCEN bit of the SSPxCON2 register is set, the slave module will automatically $\overline{\text{ACK}}$ the reception of this address regardless of the value stored in SSPxADD. After the slave clocks in an address of all zeros with the R/W bit clear, an interrupt is generated and slave software can read SSPxBUF and respond. [Figure 27-24](#) shows a general call reception sequence.

In 10-bit Address mode, the UA bit will not be set on the reception of the general call address. The slave will prepare to receive the second byte as data, just as it would in 7-bit mode.

If the AHEN bit of the SSPxCON3 register is set, just as with any other address reception, the slave hardware will stretch the clock after the eighth falling edge of SCL. The slave must then set its ACKDT value and release the clock with communication progressing as it would normally.

27.9.9 SSP MASK REGISTER

An SSP Mask (SSPxMSK) register ([Register 27-12](#)) is available in I²C Slave mode as a mask for the value held in the SSPSR register during an address comparison operation. A zero ('0') bit in the SSPxMSK register has the effect of making the corresponding bit of the received address a "don't care".

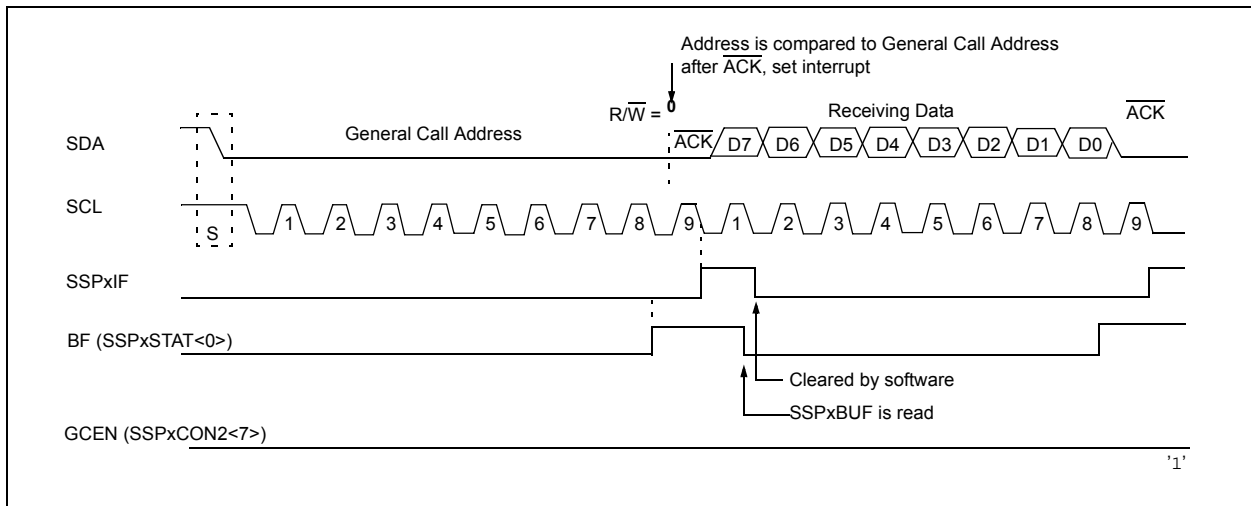
This register is reset to all '1's upon any Reset condition and, therefore, has no effect on standard SSP operation until written with a mask value.

The SSP Mask register is active during:

- 7-bit Address mode: address compare of A<7:1>.

10-bit Address mode: address compare of A<7:0> only. The SSP mask has no effect during the reception of the first (high) byte of the address.

FIGURE 27-24: SLAVE MODE GENERAL CALL ADDRESS SEQUENCE



27.10 I²C Master Mode

Master mode is enabled by setting and clearing the appropriate SSPM bits in the SSPxCON1 register and by setting the SSPEN bit. In Master mode, the SDA and SCK pins must be configured as inputs. The MSSP peripheral hardware will override the output driver TRIS controls when necessary to drive the pins low.

Master mode of operation is supported by interrupt generation on the detection of the Start and Stop conditions. The Stop (P) and Start (S) bits are cleared from a Reset or when the MSSP module is disabled. Control of the I²C bus may be taken when the P bit is set, or the bus is Idle.

In Firmware Controlled Master mode, user code conducts all I²C bus operations based on Start and Stop bit condition detection. Start and Stop condition detection is the only active circuitry in this mode. All other communication is done by the user software directly manipulating the SDA and SCL lines.

The following events will cause the SSP Interrupt Flag bit, SSPxIF, to be set (SSP interrupt, if enabled):

- Start condition detected
- Stop condition detected
- Data transfer byte transmitted/received
- Acknowledge transmitted/received
- Repeated Start generated

Note 1: The MSSP module, when configured in I²C Master mode, does not allow queuing of events. For instance, the user is not allowed to initiate a Start condition and immediately write the SSPxBUF register to initiate transmission before the Start condition is complete. In this case, the SSPxBUF will not be written to and the WCOL bit will be set, indicating that a write to the SSPxBUF did not occur

- 2:** When in Master mode, Start/Stop detection is masked and an interrupt is generated when the SEN/PEN bit is cleared and the generation is complete.

27.10.1 I²C MASTER MODE OPERATION

The master device generates all of the serial clock pulses and the Start and Stop conditions. A transfer is ended with a Stop condition or with a Repeated Start condition. Since the Repeated Start condition is also the beginning of the next serial transfer, the I²C bus will not be released.

In Master Transmitter mode, serial data is output through SDA, while SCL outputs the serial clock. The first byte transmitted contains the slave address of the receiving device (7 bits) and the Read/Write (R/W) bit. In this case, the R/W bit will be logic '0'. Serial data is transmitted eight bits at a time. After each byte is transmitted, an Acknowledge bit is received. Start and Stop conditions are output to indicate the beginning and the end of a serial transfer.

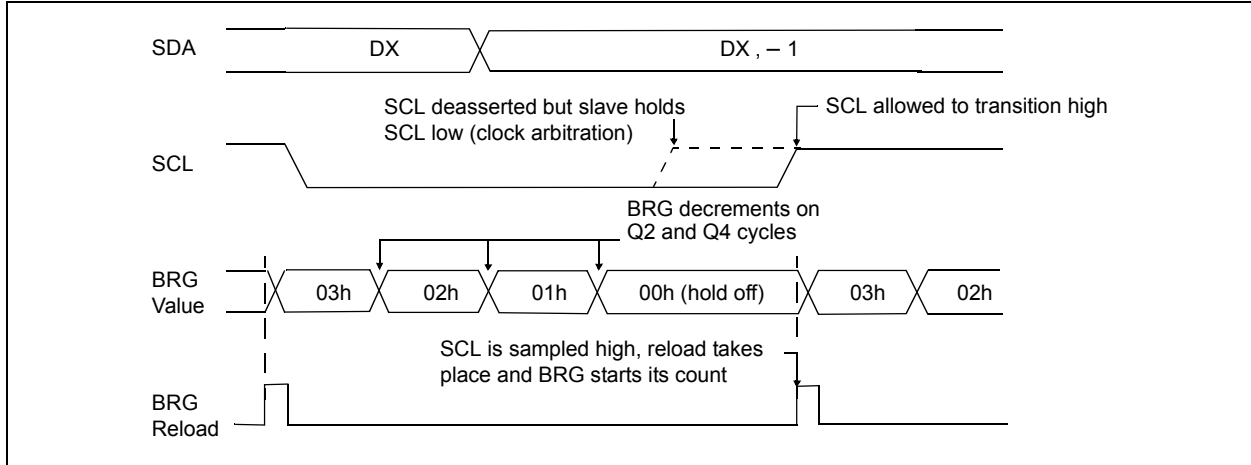
In Master Receive mode, the first byte transmitted contains the slave address of the transmitting device (7 bits) and the R/W bit. In this case, the R/W bit will be logic '1'. Thus, the first byte transmitted is a 7-bit slave address followed by a '1' to indicate the receive bit. Serial data is received via SDA, while SCL outputs the serial clock. Serial data is received eight bits at a time. After each byte is received, an Acknowledge bit is transmitted. Start and Stop conditions indicate the beginning and end of transmission.

A Baud Rate Generator is used to set the clock frequency output on SCL. See [Section 27.11 “Baud Rate Generator”](#) for more detail.

27.10.2 CLOCK ARBITRATION

Clock arbitration occurs when the master, during any receive, transmit or Repeated Start/Stop condition, releases the SCL pin (SCL allowed to float high). When the SCL pin is allowed to float high, the Baud Rate Generator (BRG) is suspended from counting until the SCL pin is actually sampled high. When the SCL pin is sampled high, the Baud Rate Generator is reloaded with the contents of SSPxADD<7:0> and begins counting. This ensures that the SCL high time will always be at least one BRG rollover count in the event that the clock is held low by an external device ([Figure 27-25](#)).

FIGURE 27-25: BAUD RATE GENERATOR TIMING WITH CLOCK ARBITRATION



27.10.3 WCOL STATUS FLAG

If the user writes the SSPxBUF when a Start, Restart, Stop, Receive or Transmit sequence is in progress, the WCOL bit is set and the contents of the buffer are unchanged (the write does not occur). Any time the WCOL bit is set it indicates that an action on SSPxBUF was attempted while the module was not idle.

Note: Because queuing of events is not allowed, writing to the lower five bits of SSPxCON2 is disabled until the Start condition is complete.

the Start condition and causes the S bit of the SSPxSTAT1 register to be set. Following this, the Baud Rate Generator is reloaded with the contents of SSPxADD<7:0> and resumes its count. When the Baud Rate Generator times out (TBRG), the SEN bit of the SSPxCON2 register will be automatically cleared by hardware; the Baud Rate Generator is suspended, leaving the SDA line held low and the Start condition is complete.

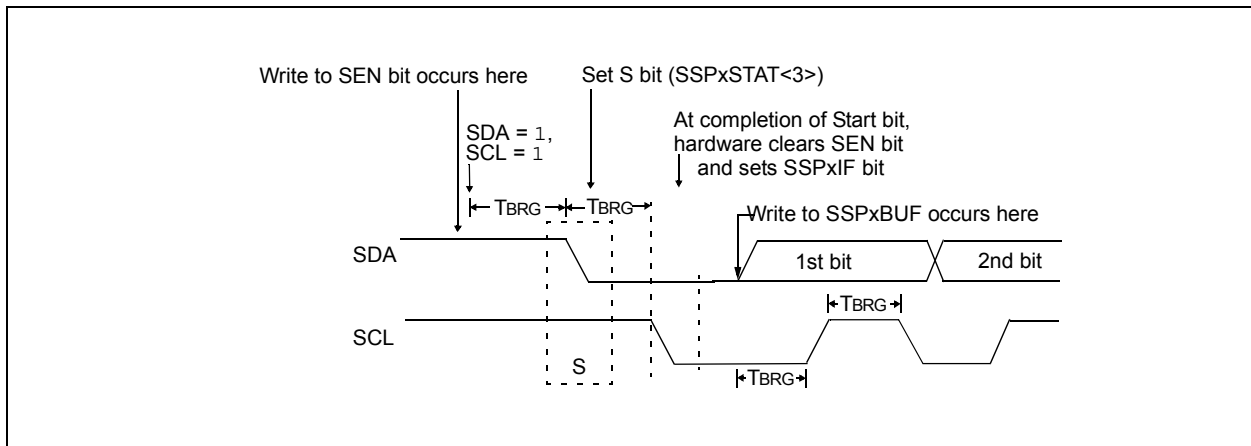
27.10.4 I²C MASTER MODE START CONDITION TIMING

To initiate a Start condition (Figure 27-26), the user sets the Start Enable bit, SEN bit of the SSPxCON2 register. If the SDA and SCL pins are sampled high, the Baud Rate Generator is reloaded with the contents of SSPxADD<7:0> and starts its count. If SCL and SDA are both sampled high when the Baud Rate Generator times out (TBRG), the SDA pin is driven low. The action of the SDA being driven low while SCL is high is

Note 1: If at the beginning of the Start condition, the SDA and SCL pins are already sampled low, or if during the Start condition, the SCL line is sampled low before the SDA line is driven low, a bus collision occurs, the Bus Collision Interrupt Flag, BCLxIF, is set, the Start condition is aborted and the I²C module is reset into its Idle state.

2: The Philips I²C specification states that a bus collision cannot occur on a Start.

FIGURE 27-26: FIRST START BIT TIMING



27.10.5 I²C MASTER MODE REPEATED START CONDITION TIMING

A Repeated Start condition (Figure 27-27) occurs when the RSEN bit of the SSPxCON2 register is programmed high and the master state machine is no longer active. When the RSEN bit is set, the SCL pin is asserted low. When the SCL pin is sampled low, the Baud Rate Generator is loaded and begins counting. The SDA pin is released (brought high) for one Baud Rate Generator count (TBRG). When the Baud Rate Generator times out, if SDA is sampled high, the SCL pin will be deasserted (brought high). When SCL is sampled high, the Baud Rate Generator is reloaded and begins counting. SDA and SCL must be sampled high for one TBRG. This action is then followed by assertion of the SDA pin (SDA = 0) for one TBRG while SCL is high. SCL is asserted low. Following this, the RSEN bit of the SSPxCON2 register will be automatically cleared and the Baud Rate Generator will not be reloaded, leaving the SDA pin held low. As soon as a Start condition is detected on the SDA and SCL pins, the S bit of the SSPxSTAT register will be set. The SSPxIF bit will not be set until the Baud Rate Generator has timed out.

Note 1: If RSEN is programmed while any other event is in progress, it will not take effect.

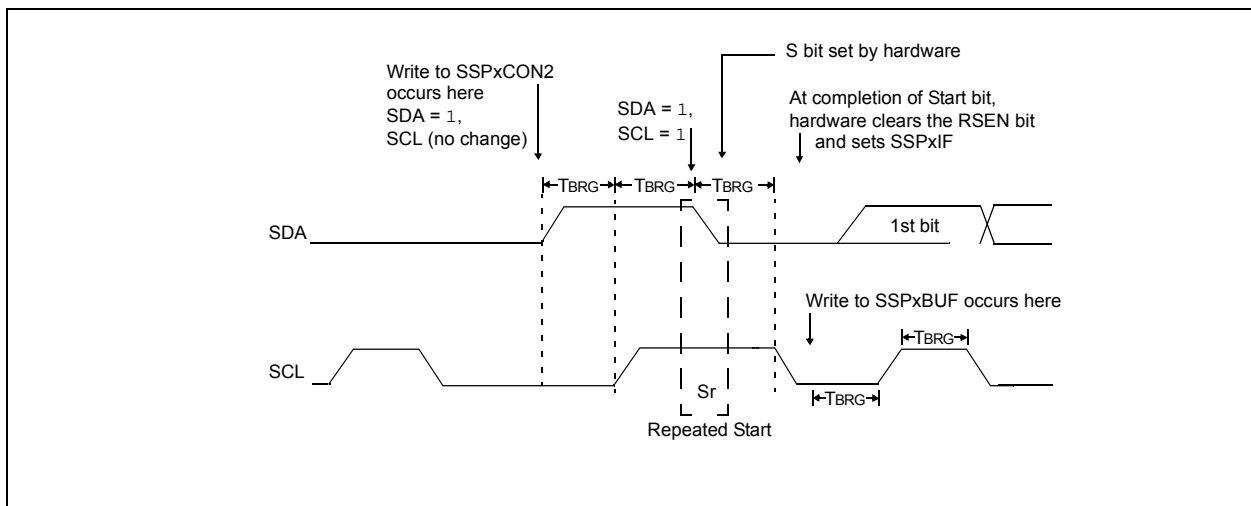
2: A bus collision during the Repeated Start condition occurs if:

- SDA is sampled low when SCL goes from low-to-high.
- SCL goes low before SDA is asserted low. This may indicate that another master is attempting to transmit a data '1'.

27.10.6 I²C MASTER MODE TRANSMISSION

Transmission of a data byte, a 7-bit address or the other half of a 10-bit address is accomplished by simply writing a value to the SSPxBUF register. This action will set the Buffer Full flag bit, BF, and allow the Baud Rate Generator to begin counting and start the next transmission. Each bit of address/data will be shifted out onto the SDA pin after the falling edge of SCL is asserted. SCL is held low for one Baud Rate Generator rollover count (TBRG). Data should be valid before SCL is released high. When the SCL pin is released high, it is held that way for TBRG. The data on the SDA pin must remain stable for that duration and some hold time after the next falling edge of SCL. After the eighth bit is shifted out (the falling edge of the eighth clock), the BF flag is cleared and the master releases SDA. This allows the slave device being addressed to respond with an ACK bit during the ninth bit time if an address match occurred, or if data was received properly. The status of ACK is written into the ACKSTAT bit on the rising edge of the ninth clock. If the master receives an Acknowledge, the Acknowledge Status bit, ACKSTAT, is cleared. If not, the bit is set. After the ninth clock, the SSPxIF bit is set and the master clock (Baud Rate Generator) is suspended until the next data byte is loaded into the SSPxBUF, leaving SCL low and SDA unchanged (Figure 27-28).

FIGURE 27-27: REPEATED START CONDITION WAVEFORM



After the write to the SSPxBUF, each bit of the address will be shifted out on the falling edge of SCL until all seven address bits and the R/W bit are completed. On the falling edge of the eighth clock, the master will release the SDA pin, allowing the slave to respond with an Acknowledge. On the falling edge of the ninth clock, the master will sample the SDA pin to see if the address was recognized by a slave. The status of the $\overline{\text{ACK}}$ bit is loaded into the ACKSTAT Status bit of the SSPxCON2 register. Following the falling edge of the ninth clock transmission of the address, the SSPxIF is set, the BF flag is cleared and the Baud Rate Generator is turned off until another write to the SSPxBUF takes place, holding SCL low and allowing SDA to float.

27.10.6.1 BF Status Flag

In Transmit mode, the BF bit of the SSPxSTAT register is set when the CPU writes to SSPxBUF and is cleared when all eight bits are shifted out.

27.10.6.2 WCOL Status Flag

If the user writes the SSPxBUF when a transmit is already in progress (i.e., SSPSR is still shifting out a data byte), the WCOL bit is set and the contents of the buffer are unchanged (the write does not occur).

The WCOL bit must be cleared by software before the next transmission.

27.10.6.3 ACKSTAT Status Flag

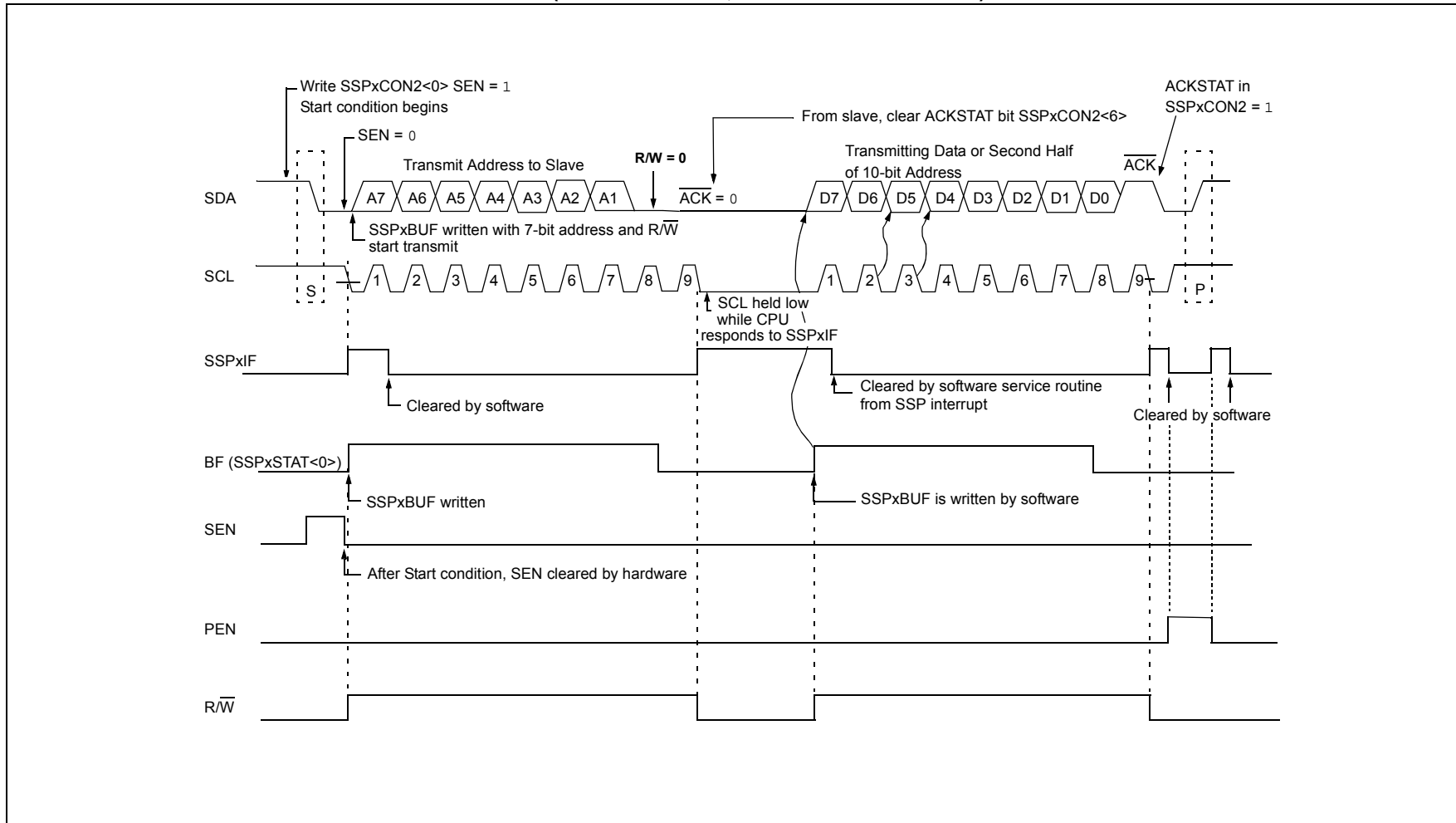
In Transmit mode, the ACKSTAT bit of the SSPxCON2 register is cleared when the slave has sent an Acknowledge ($\overline{\text{ACK}} = 0$) and is set when the slave does not Acknowledge ($\overline{\text{ACK}} = 1$). A slave sends an Acknowledge when it has recognized its address (including a general call), or when the slave has properly received its data.

27.10.6.4 Typical transmit sequence:

1. The user generates a Start condition by setting the SEN bit of the SSPxCON2 register.
2. SSPxIF is set by hardware on completion of the Start.
3. SSPxIF is cleared by software.
4. The MSSP module will wait the required start time before any other operation takes place.
5. The user loads the SSPxBUF with the slave address to transmit.
6. Address is shifted out the SDA pin until all eight bits are transmitted. Transmission begins as soon as SSPxBUF is written to.
7. The MSSP module shifts in the $\overline{\text{ACK}}$ bit from the slave device and writes its value into the ACKSTAT bit of the SSPxCON2 register.
8. The MSSP module generates an interrupt at the end of the ninth clock cycle by setting the SSPxIF bit.

9. The user loads the SSPxBUF with eight bits of data.
10. Data is shifted out the SDA pin until all eight bits are transmitted.
11. The MSSP module shifts in the $\overline{\text{ACK}}$ bit from the slave device and writes its value into the ACKSTAT bit of the SSPxCON2 register.
12. Steps 8-11 are repeated for all transmitted data bytes.
13. The user generates a Stop or Restart condition by setting the PEN or RSEN bits of the SSPxCON2 register. Interrupt is generated once the Stop/Restart condition is complete.

FIGURE 27-28: I²C MASTER MODE WAVEFORM (TRANSMISSION, 7 OR 10-BIT ADDRESS)



27.10.7 I²C Master Mode Reception

Master mode reception (Figure 27-29) is enabled by programming the Receive Enable bit, RCEN bit of the SSP1CON2 register.

Note: The MSSP module must be in an Idle state before the RCEN bit is set or the RCEN bit will be disregarded.

The Baud Rate Generator begins counting and on each rollover, the state of the SCL pin changes (high-to-low/low-to-high) and data is shifted into the SSPSR. After the falling edge of the eighth clock, the receive enable flag is automatically cleared, the contents of the SSPSR are loaded into the SSPxBUF, the BF flag bit is set, the SSPxIF flag bit is set and the Baud Rate Generator is suspended from counting, holding SCL low. The MSSP is now in Idle state awaiting the next command. When the buffer is read by the CPU, the BF flag bit is automatically cleared. The user can then send an Acknowledge bit at the end of reception by setting the Acknowledge Sequence Enable, ACKEN bit of the SSPxCON2 register.

27.10.7.1 BF Status Flag

In receive operation, the BF bit is set when an address or data byte is loaded into SSPxBUF from SSPSR. It is cleared when the SSPxBUF register is read.

27.10.7.2 SSPOV Status Flag

In receive operation, the SSPOV bit is set when eight bits are received into the SSPSR and the BF flag bit is already set from a previous reception.

27.10.7.3 WCOL Status Flag

If the user writes the SSPxBUF when a receive is already in progress (i.e., SSPSR is still shifting in a data byte), the WCOL bit is set and the contents of the buffer are unchanged (the write does not occur).

27.10.7.4 Typical Receive Sequence:

1. The user generates a Start condition by setting the SEN bit of the SSPxCON2 register.
2. SSPxIF is set by hardware on completion of the Start.
3. SSPxIF is cleared by software.
4. User writes SSPxBUF with the slave address to transmit and the R/W bit set.
5. Address is shifted out the SDA pin until all eight bits are transmitted. Transmission begins as soon as SSPxBUF is written to.
6. The MSSP module shifts in the $\overline{\text{ACK}}$ bit from the slave device and writes its value into the ACKSTAT bit of the SSPxCON2 register.
7. The MSSP module generates an interrupt at the end of the ninth clock cycle by setting the SSPxIF bit.
8. User sets the RCEN bit of the SSPxCON2 register and the master clocks in a byte from the slave.
9. After the eighth falling edge of SCL, SSPxIF and BF are set.
10. Master clears SSPxIF and reads the received byte from SSPxIF, clears BF.
11. Master sets the $\overline{\text{ACK}}$ value sent to slave in the ACKDT bit of the SSPxCON2 register and initiates the $\overline{\text{ACK}}$ by setting the ACKEN bit.
12. Master's $\overline{\text{ACK}}$ is clocked out to the slave and SSPxIF is set.
13. User clears SSPxIF.
14. Steps 8-13 are repeated for each received byte from the slave.
15. Master sends a not $\overline{\text{ACK}}$ or Stop to end communication.

27.10.8 ACKNOWLEDGE SEQUENCE TIMING

An Acknowledge sequence is enabled by setting the Acknowledge Sequence Enable bit, ACKEN bit of the SSPxCON2 register. When this bit is set, the SCL pin is pulled low and the contents of the Acknowledge data bit are presented on the SDA pin. If the user wishes to generate an Acknowledge, then the ACKDT bit should be cleared. If not, the user should set the ACKDT bit before starting an Acknowledge sequence. The Baud Rate Generator then counts for one rollover period (TBRG) and the SCL pin is deasserted (pulled high). When the SCL pin is sampled high (clock arbitration), the Baud Rate Generator counts for TBRG. The SCL pin is then pulled low. Following this, the ACKEN bit is automatically cleared, the Baud Rate Generator is turned off and the MSSP module then goes into Idle mode (Figure 27-30).

27.10.8.1 WCOL Status Flag

If the user writes the SSPxBUF when an Acknowledge sequence is in progress, then the WCOL bit is set and the contents of the buffer are unchanged (the write does not occur).

27.10.9 STOP CONDITION TIMING

A Stop bit is asserted on the SDA pin at the end of a receive/transmit by setting the Stop Sequence Enable bit, PEN bit of the SSPxCON2 register. At the end of a receive/transmit, the SCL line is held low after the falling edge of the ninth clock. When the PEN bit is set, the master will assert the SDA line low. When the SDA line is sampled low, the Baud Rate Generator is reloaded and counts down to '0'. When the Baud Rate Generator times out, the SCL pin will be brought high and one TBRG (Baud Rate Generator rollover count) later, the SDA pin will be deasserted. When the SDA pin is sampled high while SCL is high, the P bit of the SSPxSTAT register is set. A TBRG later, the PEN bit is cleared and the SSPxIF bit is set (Figure 27-31).

27.10.9.1 WCOL Status Flag

If the user writes the SSPxBUF when a Stop sequence is in progress, then the WCOL bit is set and the contents of the buffer are unchanged (the write does not occur).

FIGURE 27-30: ACKNOWLEDGE SEQUENCE WAVEFORM

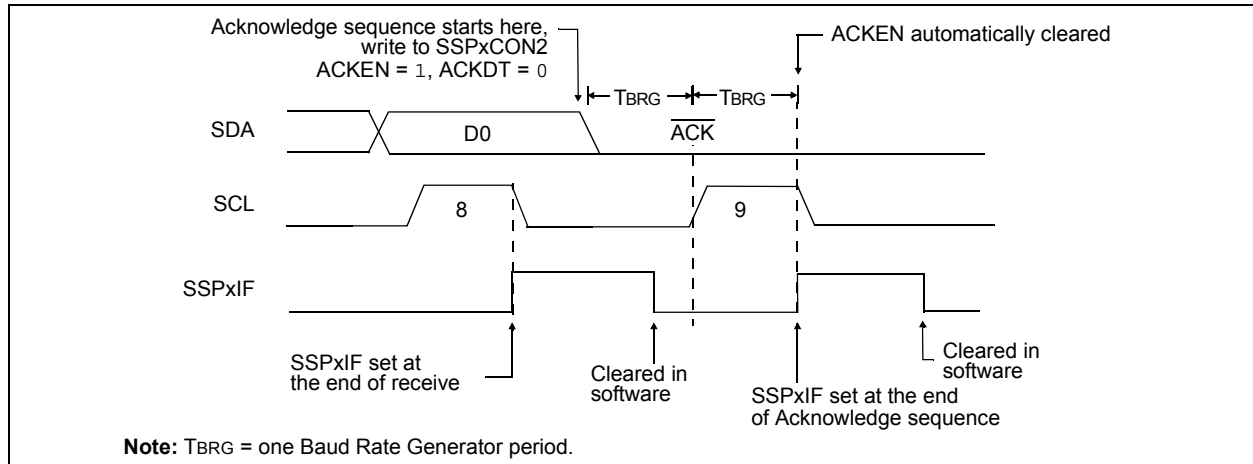
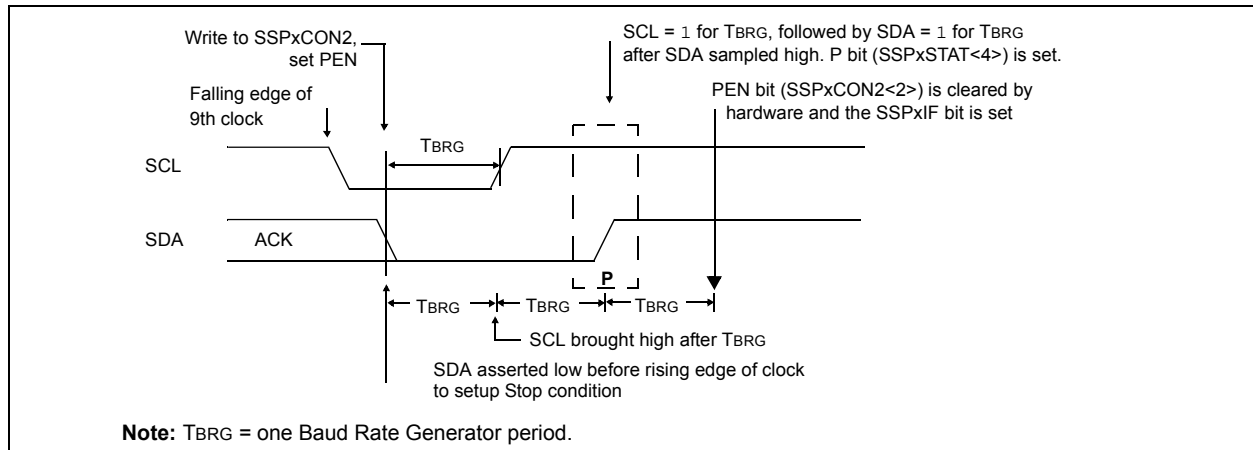


FIGURE 27-31: STOP CONDITION RECEIVE OR TRANSMIT MODE



27.10.10 SLEEP OPERATION

While in Sleep mode, the I²C slave module can receive addresses or data and when an address match or complete byte transfer occurs, wake the processor from Sleep (if the MSSP interrupt is enabled).

27.10.11 EFFECTS OF A RESET

A Reset disables the MSSP module and terminates the current transfer.

27.10.12 MULTI-MASTER MODE

In Multi-Master mode, the interrupt generation on the detection of the Start and Stop conditions allows the determination of when the bus is free. The Stop (P) and Start (S) bits are cleared from a Reset or when the MSSP module is disabled. Control of the I²C bus may be taken when the P bit of the SSPxSTAT register is set, or the bus is Idle, with both the S and P bits clear. When the bus is busy, enabling the SSP interrupt will generate the interrupt when the Stop condition occurs.

In multi-master operation, the SDA line must be monitored for arbitration to see if the signal level is the expected output level. This check is performed by hardware with the result placed in the BCLxIF bit.

The states where arbitration can be lost are:

- Address Transfer
- Data Transfer
- A Start Condition
- A Repeated Start Condition
- An Acknowledge Condition

27.10.13 MULTI-MASTER COMMUNICATION, BUS COLLISION AND BUS ARBITRATION

Multi-Master mode support is achieved by bus arbitration. When the master outputs address/data bits onto the SDA pin, arbitration takes place when the master outputs a '1' on SDA, by letting SDA float high and another master asserts a '0'. When the SCL pin floats high, data should be stable. If the expected data on SDA is a '1' and the data sampled on the SDA pin is '0', then a bus collision has taken place. The master will set the Bus Collision Interrupt Flag, BCLxIF and reset the I²C port to its Idle state (Figure 27-32).

If a transmit was in progress when the bus collision occurred, the transmission is halted, the BF flag is cleared, the SDA and SCL lines are deasserted and the SSPxBUF can be written to. When the user services the bus collision Interrupt Service Routine and if the I²C bus is free, the user can resume communication by asserting a Start condition.

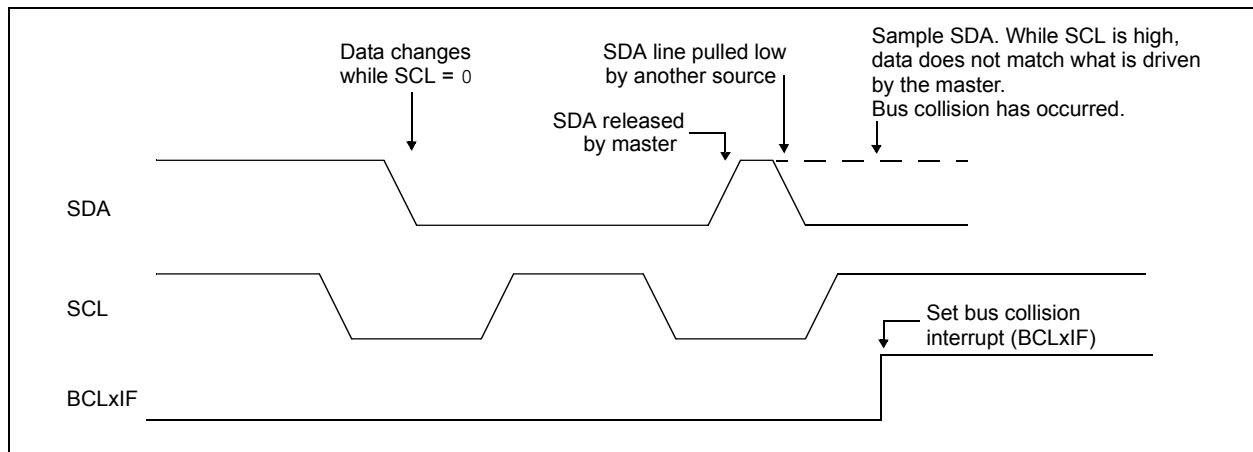
If a Start, Repeated Start, Stop or Acknowledge condition was in progress when the bus collision occurred, the condition is aborted, the SDA and SCL lines are deasserted and the respective control bits in the SSPxCON2 register are cleared. When the user services the bus collision Interrupt Service Routine and if the I²C bus is free, the user can resume communication by asserting a Start condition.

The master will continue to monitor the SDA and SCL pins. If a Stop condition occurs, the SSPxIF bit will be set.

A write to the SSPxBUF will start the transmission of data at the first data bit, regardless of where the transmitter left off when the bus collision occurred.

In Multi-Master mode, the interrupt generation on the detection of Start and Stop conditions allows the determination of when the bus is free. Control of the I²C bus can be taken when the P bit is set in the SSPxSTAT register, or the bus is Idle and the S and P bits are cleared.

FIGURE 27-32: BUS COLLISION TIMING FOR TRANSMIT AND ACKNOWLEDGE



27.10.13.1 Bus Collision During a Start Condition

During a Start condition, a bus collision occurs if:

- SDA or SCL are sampled low at the beginning of the Start condition (Figure 27-33).
- SCL is sampled low before SDA is asserted low (Figure 27-34).

During a Start condition, both the SDA and the SCL pins are monitored.

If the SDA pin is already low, or the SCL pin is already low, then all of the following occur:

- the Start condition is aborted,
- the BCLxIF flag is set and
- the MSSP module is reset to its Idle state (Figure 27-33).

The Start condition begins with the SDA and SCL pins deasserted. When the SDA pin is sampled high, the Baud Rate Generator is loaded and counts down. If the SCL pin is sampled low while SDA is high, a bus collision occurs because it is assumed that another master is attempting to drive a data '1' during the Start condition.

If the SDA pin is sampled low during this count, the BRG is reset and the SDA line is asserted early (Figure 27-35). If, however, a '1' is sampled on the SDA pin, the SDA pin is asserted low at the end of the BRG count. The Baud Rate Generator is then reloaded and counts down to zero; if the SCL pin is sampled as '0' during this time, a bus collision does not occur. At the end of the BRG count, the SCL pin is asserted low.

Note: The reason that bus collision is not a factor during a Start condition is that no two bus masters can assert a Start condition at the exact same time. Therefore, one master will always assert SDA before the other. This condition does not cause a bus collision because the two masters must be allowed to arbitrate the first address following the Start condition. If the address is the same, arbitration must be allowed to continue into the data portion, Repeated Start or Stop conditions.

FIGURE 27-33: BUS COLLISION DURING START CONDITION (SDA ONLY)

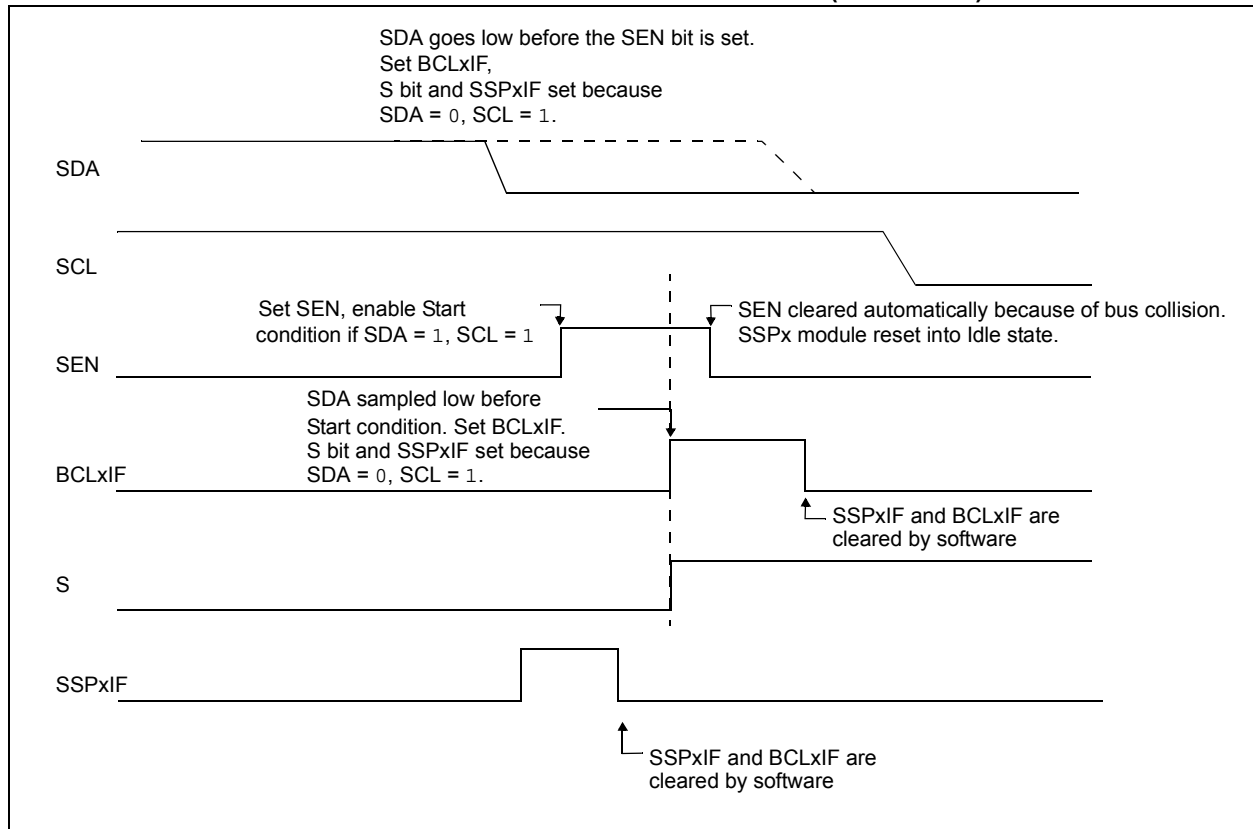


FIGURE 27-34: BUS COLLISION DURING START CONDITION (SCL = 0)

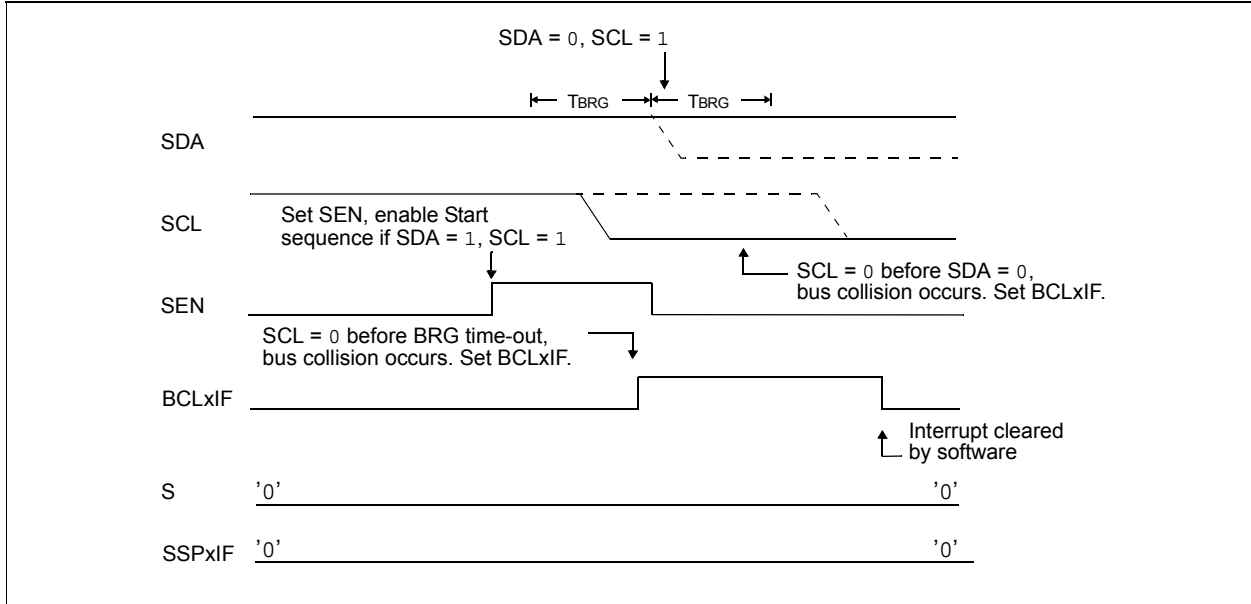
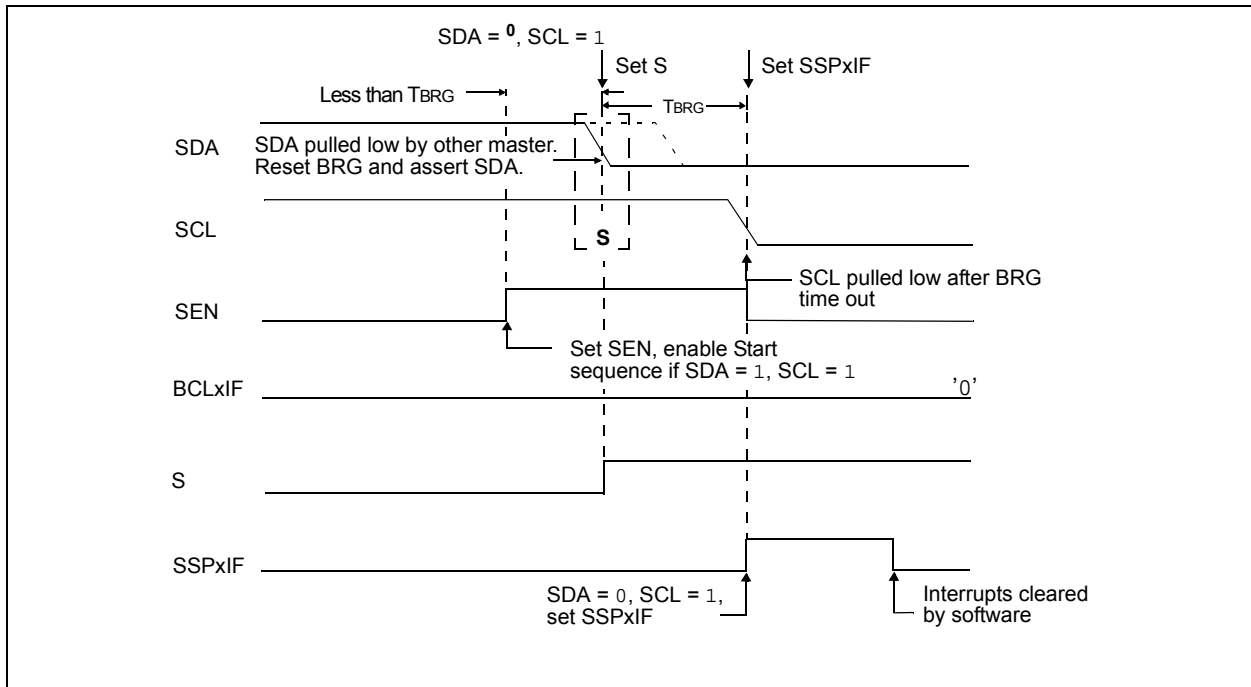


FIGURE 27-35: BRG RESET DUE TO SDA ARBITRATION DURING START CONDITION



27.10.13.2 Bus Collision During a Repeated Start Condition

During a Repeated Start condition, a bus collision occurs if:

- A low level is sampled on SDA when SCL goes from low level to high level (Case 1).
- SCL goes low before SDA is asserted low, indicating that another master is attempting to transmit a data '1' (Case 2).

When the user releases SDA and the pin is allowed to float high, the BRG is loaded with SSPxADD and counts down to zero. The SCL pin is then deasserted and when sampled high, the SDA pin is sampled.

If SDA is low, a bus collision has occurred (i.e., another master is attempting to transmit a data '0', [Figure 27-36](#)). If SDA is sampled high, the BRG is reloaded and begins counting. If SDA goes from high-to-low before the BRG times out, no bus collision occurs because no two masters can assert SDA at exactly the same time.

If SCL goes from high-to-low before the BRG times out and SDA has not already been asserted, a bus collision occurs. In this case, another master is attempting to transmit a data '1' during the Repeated Start condition, see [Figure 27-37](#).

If, at the end of the BRG time out, both SCL and SDA are still high, the SDA pin is driven low and the BRG is reloaded and begins counting. At the end of the count, regardless of the status of the SCL pin, the SCL pin is driven low and the Repeated Start condition is complete.

FIGURE 27-36: BUS COLLISION DURING A REPEATED START CONDITION (CASE 1)

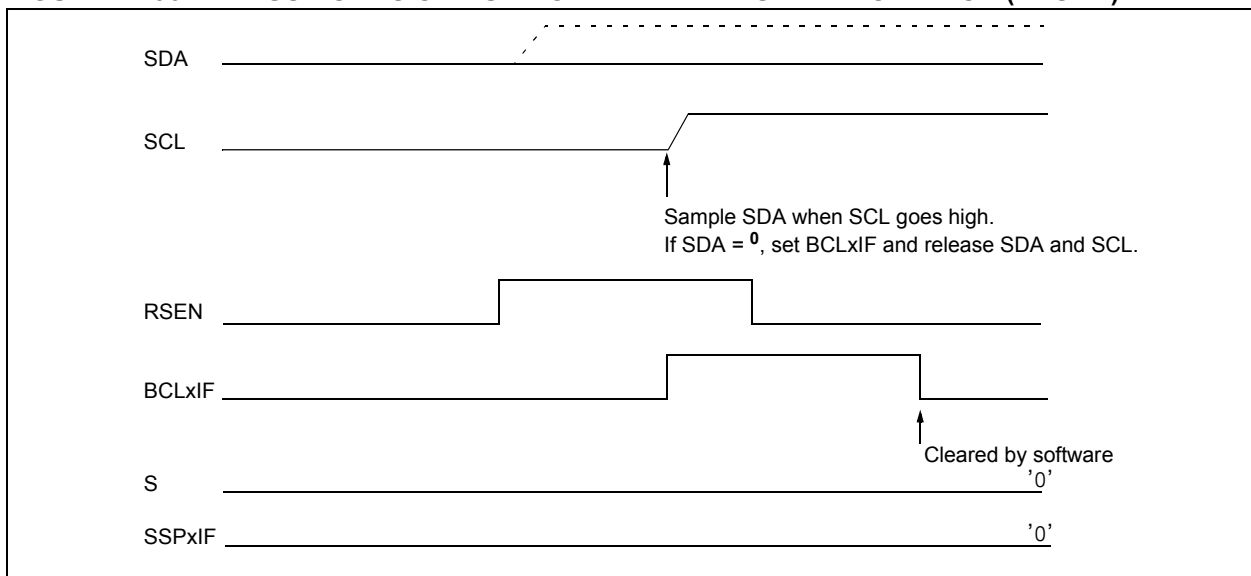
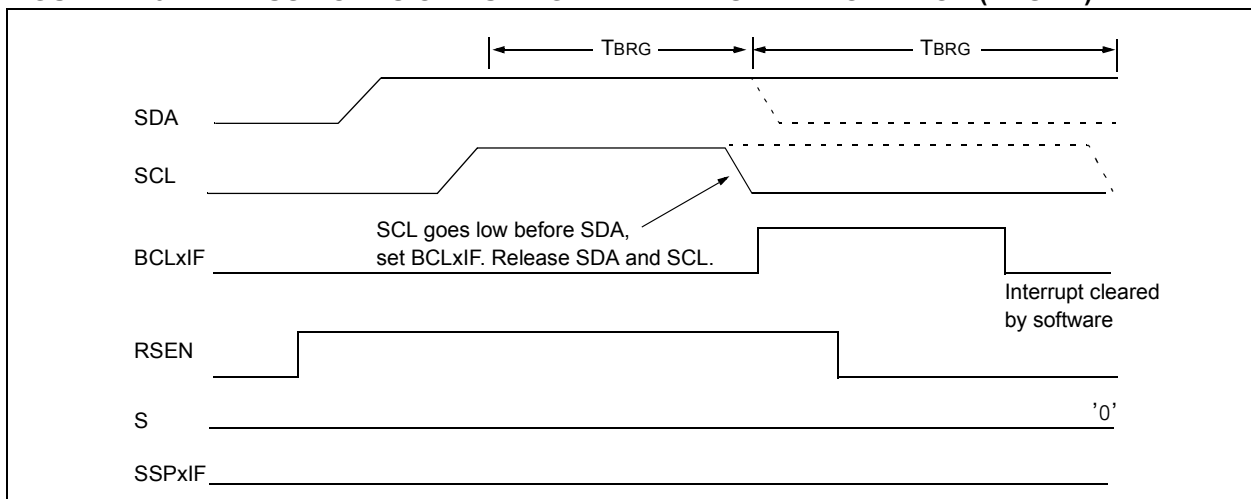


FIGURE 27-37: BUS COLLISION DURING REPEATED START CONDITION (CASE 2)



27.10.13.3 Bus Collision During a Stop Condition

Bus collision occurs during a Stop condition if:

- After the SDA pin has been deasserted and allowed to float high, SDA is sampled low after the BRG has timed out (Case 1).
- After the SCL pin is deasserted, SCL is sampled low before SDA goes high (Case 2).

The Stop condition begins with SDA asserted low. When SDA is sampled low, the SCL pin is allowed to float. When the pin is sampled high (clock arbitration), the Baud Rate Generator is loaded with SSPxADD and counts down to zero. After the BRG times out, SDA is sampled. If SDA is sampled low, a bus collision has occurred. This is due to another master attempting to drive a data '0' (Figure 27-38). If the SCL pin is sampled low before SDA is allowed to float high, a bus collision occurs. This is another case of another master attempting to drive a data '0' (Figure 27-39).

FIGURE 27-38: BUS COLLISION DURING A STOP CONDITION (CASE 1)

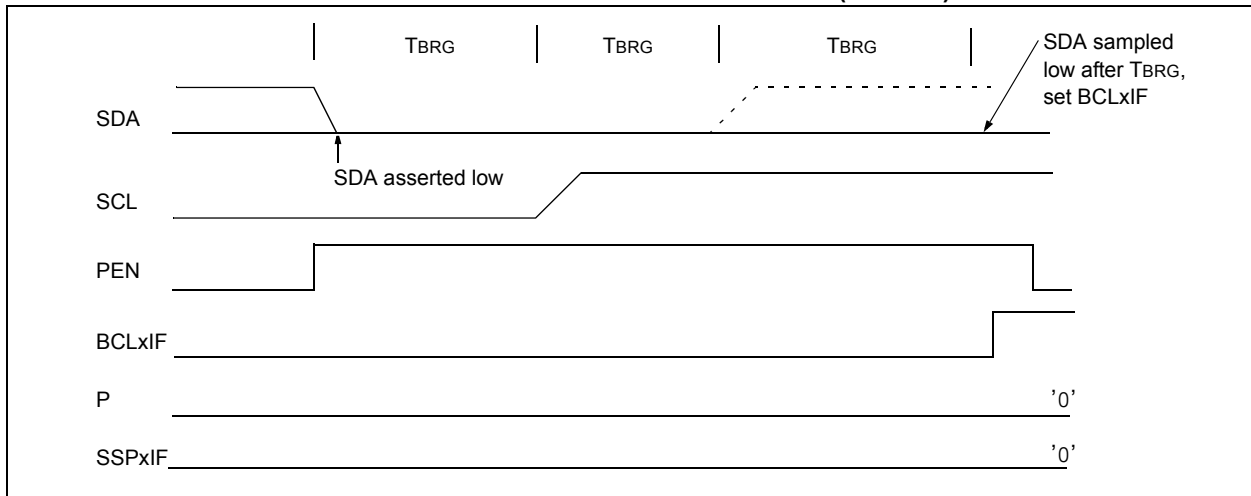
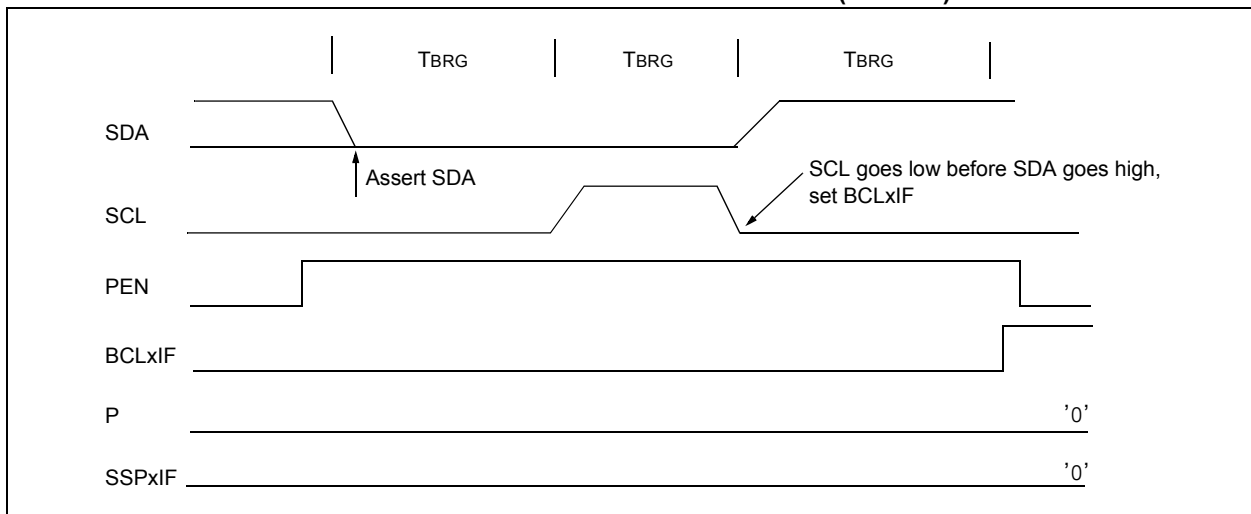


FIGURE 27-39: BUS COLLISION DURING A STOP CONDITION (CASE 2)



27.11 Baud Rate Generator

The MSSP module has a Baud Rate Generator available for clock generation in both I²C and SPI Master modes. The Baud Rate Generator (BRG) reload value is placed in the SSPxADD register (Register 27-5). When a write occurs to SSPxBUF, the Baud Rate Generator will automatically begin counting down.

Once the given operation is complete, the internal clock will automatically stop counting and the clock pin will remain in its last state.

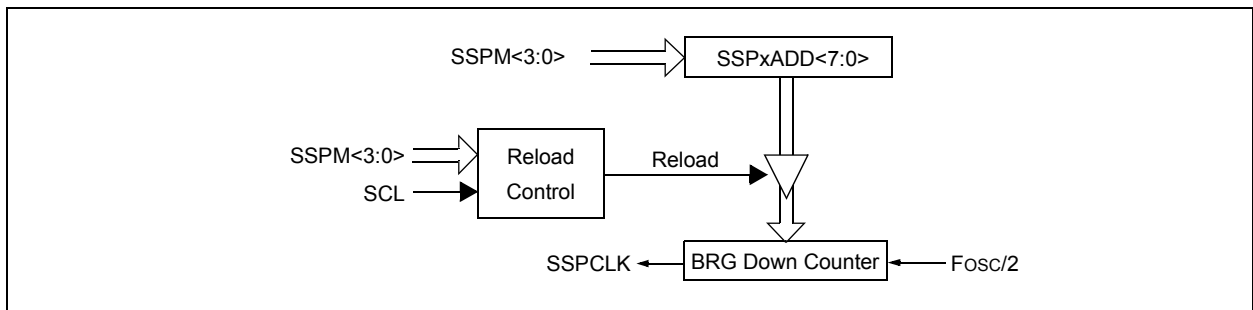
An internal signal “Reload” in Figure 27-40 triggers the value from SSPxADD to be loaded into the BRG counter. This occurs twice for each oscillation of the module clock line. The logic dictating when the reload signal is asserted depends on the mode the MSSP is being operated in.

Table 27-3 demonstrates clock rates based on instruction cycles and the BRG value loaded into SSPxADD.

EQUATION 27-1:

$$F_{CLOCK} = \frac{F_{OSC}}{(SSPADD + 1)(4)}$$

FIGURE 27-40: BAUD RATE GENERATOR BLOCK DIAGRAM



Note: Values of 0x00, 0x01 and 0x02 are not valid for SSPxADD when used as a Baud Rate Generator for I²C. This is an implementation limitation.

TABLE 27-3: MSSP CLOCK RATE W/BRG

| Fosc | Fcy | BRG Value | F _{CLOCK} (2 Rollovers of BRG) |
|--------|-------|-----------|--|
| 32 MHz | 8 MHz | 13h | 400 kHz |
| 32 MHz | 8 MHz | 19h | 308 kHz |
| 32 MHz | 8 MHz | 4Fh | 100 kHz |
| 16 MHz | 4 MHz | 09h | 400 kHz |
| 16 MHz | 4 MHz | 0Ch | 308 kHz |
| 16 MHz | 4 MHz | 27h | 100 kHz |
| 4 MHz | 1 MHz | 09h | 100 kHz |

Note: Refer to the I/O port electrical specifications in Table 37-8: Internal Oscillator Parameters, to ensure the system is designed to support IOL requirements.

TABLE 27-4: SUMMARY OF REGISTERS ASSOCIATED WITH I²C OPERATION

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset Values on Page |
|------------|----------|-----------|----------------|-------|-----------|--------------|---------|---------|----------------------|
| INTCON | GIE/GIEH | PEIE/GIEL | IPEN | — | INT3EDG | INT2EDG | INT1EDG | INT0EDG | 172 |
| PIE3 | RC2IE | TX2IE | RC1IE | TX1IE | BCL2IE | SSP2IE | BCL1IE | SSP1IE | 187 |
| PIR3 | RC2IF | TX2IF | RC1IF | TX1IF | BCL2IF | SSP2IF | BCL1IF | SSP1IF | 176 |
| IPR3 | RC2IP | TX2IP | RC1IP | TX1IP | BCL2IP | SSP2IP | BCL1IP | SSP1IP | 197 |
| RxyPPS | — | — | RxyPPS<5:0> | | | | | | 227 |
| SSPxADD | ADD<7:0> | | | | | | | | 391 |
| SSPxBUF | BUF<7:0> | | | | | | | | 387* |
| SSPxCLKPPS | — | — | SSPCLKPPS<5:0> | | | | | | 224 |
| SSPxCON1 | WCOL | SSPOV | SSPEN | CKP | SSPM<3:0> | | | | 389 |
| SSPxCON2 | GCEN | ACKSTAT | ACKDT | ACKEN | RCEN | PEN | RSEN | SEN | 406 |
| SSPxCON3 | ACKTIM | PCIE | SCIE | BOEN | SDAHT | SBCDE | AHEN | DHEN | 390 |
| SSPxDATPPS | — | — | SSPDATPPS<5:0> | | | | | | 224 |
| SSPxMSK | MSK<7:0> | | | | | | | | 408 |
| SSPxSTAT | SMP | CKE | D/ \bar{A} | P | S | R/ \bar{W} | UA | BF | 388 |

Legend: — = unimplemented location, read as '0'. Shaded cells are not used by the MSSP module in I²C mode.

* Page provides register information.

28.0 ENHANCED UNIVERSAL SYNCHRONOUS ASYNCHRONOUS RECEIVER TRANSMITTER (EUSART)

Note: The PIC18(L)F6xK40 devices have five EUSARTs. Therefore, all information in this section refers to EUSART1/2/3/4/5.

The Enhanced Universal Synchronous Asynchronous Receiver Transmitter (EUSART) 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 EUSART, also known as a Serial Communications Interface (SCI), can be configured as a full-duplex asynchronous system or half-duplex synchronous system. Full-Duplex mode is useful for communications with peripheral systems, such as CRT terminals and personal computers. Half-Duplex Synchronous mode is intended for communications with peripheral devices, such as A/D or D/A integrated circuits, serial EEPROMs or other microcontrollers.

These devices typically do not have internal clocks for baud rate generation and require the external clock signal provided by a master synchronous device.

The EUSART module includes the following capabilities:

- Full-duplex asynchronous transmit and receive
- Two-character input buffer
- One-character output buffer
- Programmable 8-bit or 9-bit character length
- Address detection in 9-bit mode
- Input buffer overrun error detection
- Received character framing error detection
- Half-duplex synchronous master
- Half-duplex synchronous slave
- Programmable clock polarity in synchronous modes
- Sleep operation

The EUSART module implements the following additional features, making it ideally suited for use in Local Interconnect Network (LIN) bus systems:

- Automatic detection and calibration of the baud rate
- Wake-up on Break reception
- 13-bit Break character transmit

Block diagrams of the EUSART transmitter and receiver are shown in [Figure 28-1](#) and [Figure 28-2](#).

FIGURE 28-1: EUSART TRANSMIT BLOCK DIAGRAM

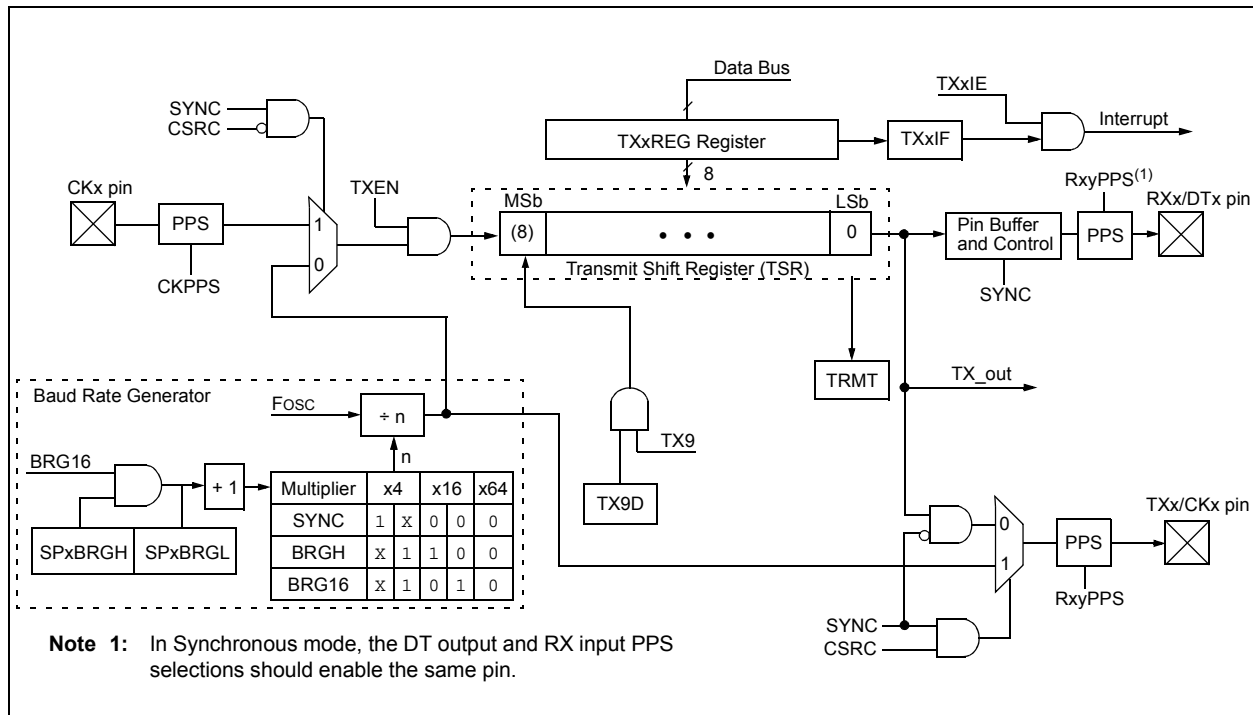
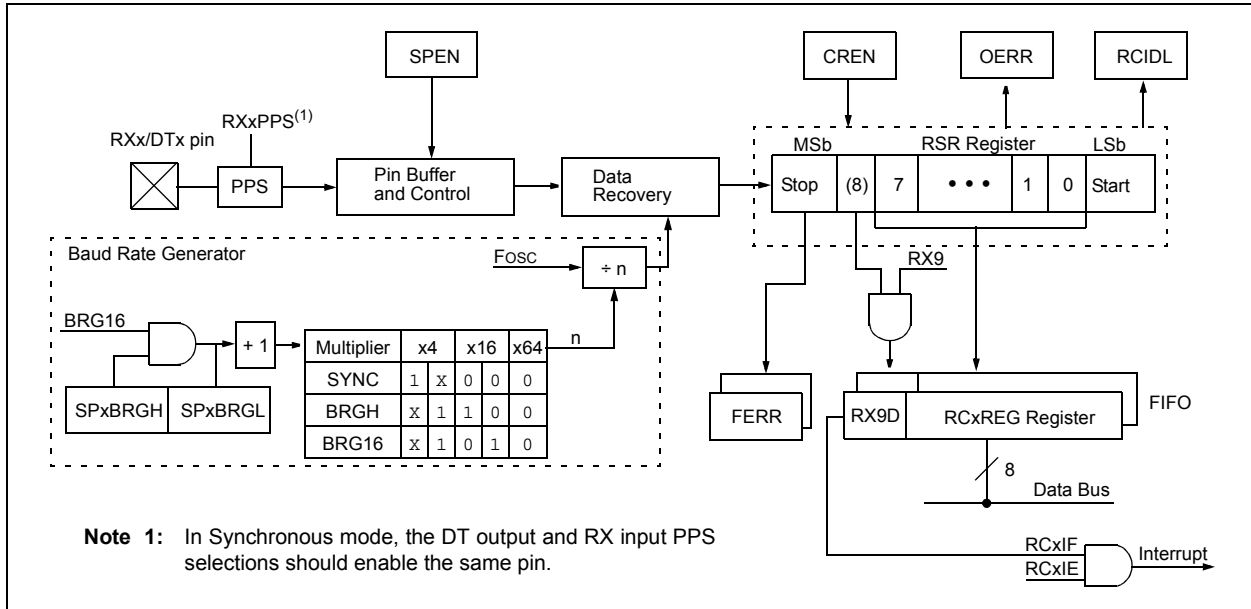


FIGURE 28-2: EUSART RECEIVE BLOCK DIAGRAM



The operation of the EUSART module is controlled through three registers:

- Transmit Status and Control (TXxSTA)
- Receive Status and Control (RCxSTA)
- Baud Rate Control (BAUDxCON)

These registers are detailed in [Register 28-1](#), [Register 28-2](#) and [Register 28-3](#), respectively.

The RXx/DTx and TXx/CKx input pins are selected with the RXxPPS and TXxPPS registers, respectively. TXx, CKx, and DTx output pins are selected with each pin's RxyPPS register. Since the RX input is coupled with the DT output in Synchronous mode, it is the user's responsibility to select the same pin for both of these functions when operating in Synchronous mode. The EUSART control logic will control the data direction drivers automatically.

28.1 Register Definitions: EUSART Control

REGISTER 28-1: TXxSTA: TRANSMIT STATUS AND CONTROL REGISTER

| R/W-/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R-1/1 | R/W-0/0 |
|--------|---------|---------------------|---------|---------|---------|-------|---------|
| CSRC | TX9 | TXEN ⁽¹⁾ | SYNC | SENDB | BRGH | TRMT | TX9D |
| bit 7 | | | | | | bit 0 | |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

- bit 7 **CSRC:** Clock Source Select bit
Asynchronous mode:
 Don't care
Synchronous mode:
 1 = Master mode (clock generated internally from BRG)
 0 = Slave mode (clock from external source)
- bit 6 **TX9:** 9-bit Transmit Enable bit
 1 = Selects 9-bit transmission
 0 = Selects 8-bit transmission
- bit 5 **TXEN:** Transmit Enable bit⁽¹⁾
 1 = Transmit enabled
 0 = Transmit disabled
- bit 4 **SYNC:** EUSART Mode Select bit
 1 = Synchronous mode
 0 = Asynchronous mode
- bit 3 **SENDB:** Send Break Character bit
Asynchronous mode:
 1 = Send Sync Break on next transmission (cleared by hardware upon completion)
 0 = Sync Break transmission disabled or completed
Synchronous mode:
 Don't care
- bit 2 **BRGH:** High Baud Rate Select bit
Asynchronous mode:
 1 = High speed, if BRG16 = 1, baud rate is baudclk/4; else baudclk/16
 0 = Low speed
Synchronous mode:
 Unused in this mode
- bit 1 **TRMT:** Transmit Shift Register Status bit
 1 = TSR empty
 0 = TSR full
- bit 0 **TX9D:** Ninth bit of Transmit Data
 Can be address/data bit or a parity bit.

Note 1: SREN/CREN bits of RCxSTA ([Register 28-2](#)) override TXEN in Sync mode.

REGISTER 28-2: RCxSTA: RECEIVE STATUS AND CONTROL REGISTER

| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R-0/0 | R/HC-0/0 | R/HC-0/0 |
|---------|---------|---------|---------|---------|-------|----------|----------|
| SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D |
| bit 7 | | | | | | bit 0 | |

Legend:

R = Readable bit

W = Writable bit

HC = Bit is cleared by hardware

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

- bit 7 **SPEN:** Serial Port Enable bit
 1 = Serial port enabled
 0 = Serial port disabled (held in Reset)
- bit 6 **RX9:** 9-Bit Receive Enable bit
 1 = Selects 9-bit reception
 0 = Selects 8-bit reception
- bit 5 **SREN:** Single Receive Enable bit
Asynchronous mode:
 Don't care
Synchronous mode – Master:
 1 = Enables single receive
 0 = Disables single receive
 This bit is cleared after reception is complete.
Synchronous mode – Slave
 Don't care
- bit 4 **CREN:** Continuous Receive Enable bit
Asynchronous mode:
 1 = Enables receiver
 0 = Disables receiver
Synchronous mode:
 1 = Enables continuous receive until enable bit CREN is cleared (CREN overrides SREN)
 0 = Disables continuous receive
- bit 3 **ADDEN:** Address Detect Enable bit
Asynchronous mode 9-bit (RX9 = 1):
 1 = Enables address detection, enable interrupt and load the receive buffer when RSR<8> is set
 0 = Disables address detection, all bytes are received and ninth bit can be used as parity bit
Asynchronous mode 8-bit (RX9 = 0):
 Don't care
- bit 2 **FERR:** Framing Error bit
 1 = Framing error (can be updated by reading RCxREG register and receive next valid byte)
 0 = No framing error
- bit 1 **OERR:** Overrun Error bit
 1 = Overrun error (can be cleared by clearing bit CREN)
 0 = No overrun error
- bit 0 **RX9D:** Ninth bit of Received Data
 This can be address/data bit or a parity bit and must be calculated by user firmware.

REGISTER 28-3: BAUDxCON: BAUD RATE CONTROL REGISTER

| R-0/0 | R-1/1 | U-0 | R/W-0/0 | R/W-0/0 | U-0 | R/W-0/0 | R/W-0/0 |
|--------|-------|-----|---------|---------|-----|---------|---------|
| ABDOVF | RCIDL | — | SCKP | BRG16 | — | WUE | ABDEN |
| bit 7 | | | | | | | bit 0 |

Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

bit 7 **ABDOVF:** Auto-Baud Detect Overflow bit

Asynchronous mode:

1 = Auto-baud timer overflowed

0 = Auto-baud timer did not overflow

Synchronous mode:

Don't care

bit 6 **RCIDL:** Receive Idle Flag bit

Asynchronous mode:

1 = Receiver is Idle

0 = Start bit has been received and the receiver is receiving

Synchronous mode:

Don't care

bit 5 **Unimplemented:** Read as '0'

bit 4 **SCKP:** Synchronous Clock Polarity Select bit

Asynchronous mode:

1 = Idle state for transmit (TX) is a low level (transmit data inverted)

0 = Idle state for transmit (TX) is a high level (transmit data is non-inverted)

Synchronous mode:

1 = Data is clocked on rising edge of the clock

0 = Data is clocked on falling edge of the clock

bit 3 **BRG16:** 16-bit Baud Rate Generator bit

1 = 16-bit Baud Rate Generator is used

0 = 8-bit Baud Rate Generator is used

bit 2 **Unimplemented:** Read as '0'

bit 1 **WUE:** Wake-up Enable bit

Asynchronous mode:

1 = Receiver is waiting for a falling edge. No character will be received, byte RCxIF will be set. WUE will automatically clear after RCxIF is set.

0 = Receiver is operating normally

Synchronous mode:

Don't care

bit 0 **ABDEN:** Auto-Baud Detect Enable bit

Asynchronous mode:

1 = Auto-Baud Detect mode is enabled (clears when auto-baud is complete)

0 = Auto-Baud Detect mode is disabled

Synchronous mode:

Don't care

28.2 EUSART Asynchronous Mode

The EUSART transmits and receives data using the standard non-return-to-zero (NRZ) format. NRZ is implemented with two levels: a V_{OH} Mark state which represents a '1' data bit, and a V_{OL} Space state which represents a '0' data bit. NRZ refers to the fact 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 eight or nine data bits 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. Each transmitted bit persists for a period of 1/(Baud Rate). An on-chip dedicated 8-bit/16-bit Baud Rate Generator is used to derive standard baud rate frequencies from the system oscillator. See [Table 28-5](#) for examples of baud rate configurations.

The EUSART transmits and receives the LSb first. The EUSART's transmitter and receiver are functionally independent, but share the same data format and baud rate. Parity is not supported by the hardware, but can be implemented in software and stored as the ninth data bit.

28.2.1 EUSART ASYNCHRONOUS TRANSMITTER

The EUSART transmitter block diagram is shown in [Figure 28-1](#). 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 TXxREG register.

28.2.1.1 Enabling the Transmitter

The EUSART transmitter is enabled for asynchronous operations by configuring the following three control bits:

- TXEN = 1
- SYNC = 0
- SPEN = 1

All other EUSART control bits are assumed to be in their default state.

Setting the TXEN bit of the TXxSTA register enables the transmitter circuitry of the EUSART. Clearing the SYNC bit of the TXxSTA register configures the EUSART for asynchronous operation. Setting the SPEN bit of the RCxSTA register enables the EUSART and automatically configures the TXx/CKx I/O pin as an output. If the TXx/CKx pin is shared with an analog peripheral, the analog I/O function must be disabled by clearing the corresponding ANSEL bit.

Note: The TXxIF Transmitter Interrupt flag is set when the TXEN enable bit is set.

28.2.1.2 Transmitting Data

A transmission is initiated by writing a character to the TXxREG register. If this is the first character, or the previous character has been completely flushed from the TSR, the data in the TXxREG 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 TXxREG until the Stop bit of the previous character has been transmitted. The pending character in the TXxREG is then transferred to the TSR in one T_{cy} immediately following the Stop bit transmission. The transmission of the Start bit, data bits and Stop bit sequence commences immediately following the transfer of the data to the TSR from the TXxREG.

28.2.1.3 Transmit Data Polarity

The polarity of the transmit data can be controlled with the SCKP bit of the BAUDxCON register. The default state of this bit is '0' which selects high true transmit idle and data bits. Setting the SCKP bit to '1' will invert the transmit data resulting in low true idle and data bits. The SCKP bit controls transmit data polarity in Asynchronous mode only. In Synchronous mode, the SCKP bit has a different function. See [Section 28.5.1.2 "Clock Polarity"](#).

28.2.1.4 Transmit Interrupt Flag

The TXxIF interrupt flag bit of the PIR3/4 registers is set whenever the EUSART transmitter is enabled and no character is being held for transmission in the TXxREG. In other words, the TXxIF bit is only clear when the TSR is busy with a character and a new character has been queued for transmission in the TXxREG. The TXxIF flag bit is not cleared immediately upon writing TXxREG. TXxIF becomes valid in the second instruction cycle following the write execution. Polling TXxIF immediately following the TXxREG write will return invalid results. The TXxIF bit is read-only, it cannot be set or cleared by software.

The TXxIF interrupt can be enabled by setting the TXxIE interrupt enable bit of the PIE3/4 registers. However, the TXxIF flag bit will be set whenever the TXxREG is empty, regardless of the state of TXxIE enable bit.

To use interrupts when transmitting data, set the TXxIE bit only when there is more data to send. Clear the TXxIE interrupt enable bit upon writing the last character of the transmission to the TXxREG.

28.2.1.5 TSR Status

The TRMT bit of the TXxSTA register indicates the status of the TSR register. This is a read-only bit. The TRMT bit is set when the TSR register is empty and is cleared when a character is transferred to the TSR register from the TXxREG. The TRMT bit remains clear until all bits have been shifted out of the TSR register. No interrupt logic is tied to this bit, so the user has to poll this bit to determine the TSR status.

Note: The TSR register is not mapped in data memory, so it is not available to the user.

28.2.1.6 Transmitting 9-Bit Characters

The EUSART supports 9-bit character transmissions. When the TX9 bit of the TXxSTA register is set, the EUSART will shift nine bits out for each character transmitted. The TX9D bit of the TXxSTA register is the ninth, and Most Significant data bit. When transmitting 9-bit data, the TX9D data bit must be written before writing the eight Least Significant bits into the TXxREG. All nine bits of data will be transferred to the TSR shift register immediately after the TXxREG is written.

A special 9-bit Address mode is available for use with multiple receivers. See [Section 28.2.2.7 “Address Detection”](#) for more information on the Address mode.

28.2.1.7 Asynchronous Transmission Setup:

1. Initialize the SPxBRGH, SPxBRGL register pair and the BRGH and BRG16 bits to achieve the desired baud rate (see [Section 28.4 “EUSART Baud Rate Generator \(BRG\)”](#)).
2. Enable the asynchronous serial port by clearing the SYNC bit and setting the SPEN bit.
3. If 9-bit transmission is desired, set the TX9 control bit. A set ninth data bit will indicate that the eight Least Significant data bits are an address when the receiver is set for address detection.
4. Set SCKP bit if inverted transmit is desired.
5. Enable the transmission by setting the TXEN control bit. This will cause the TXxIF interrupt bit to be set.
6. If interrupts are desired, set the TXxIE interrupt enable bit of the PIE3/4 registers. An interrupt will occur immediately provided that the GIE and PEIE bits of the INTCON register are also set.
7. If 9-bit transmission is selected, the ninth bit should be loaded into the TX9D data bit.
8. Load 8-bit data into the TXxREG register. This will start the transmission.

FIGURE 28-3: ASYNCHRONOUS TRANSMISSION

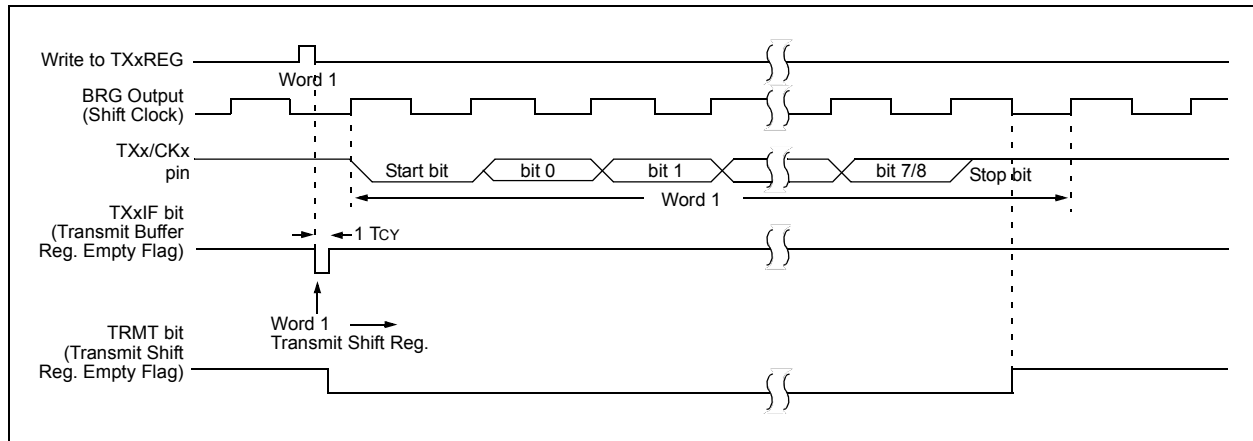


FIGURE 28-4: ASYNCHRONOUS TRANSMISSION (BACK-TO-BACK)

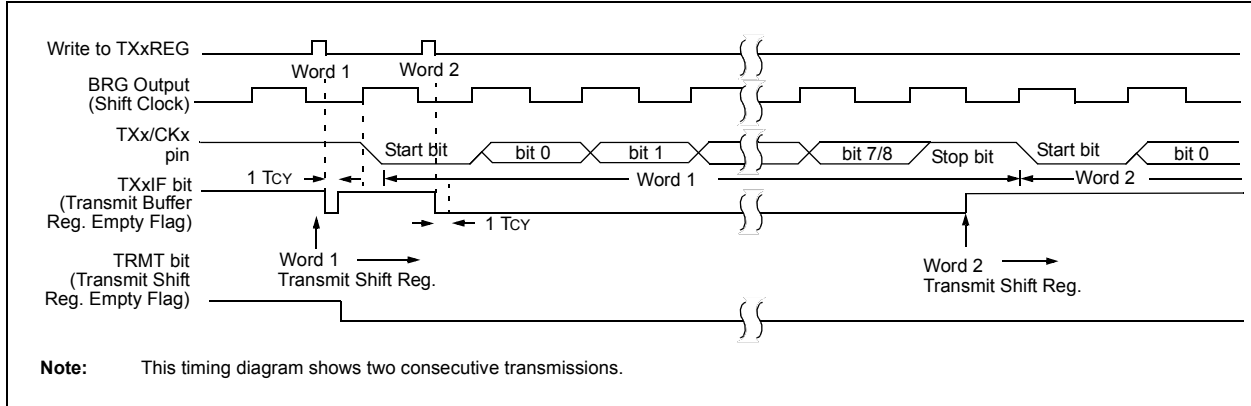


TABLE 28-1: SUMMARY OF REGISTERS ASSOCIATED WITH ASYNCHRONOUS TRANSMISSION

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on Page |
|----------|--|-----------|-------------|-------|---------|---------|---------|---------|------------------|
| BAUDxCON | ABDOVF | RCIDL | — | SCKP | BRG16 | — | WUE | ABDEN | 446 |
| INTCON | GIE/GIEH | PEIE/GIEL | IPEN | — | INT3EDG | INT2EDG | INT1EDG | INT0EDG | 172 |
| PIE3 | RC2IE | TX2IE | RC1IE | TX1IE | BCL2IE | SSP2IE | BCL1IE | SSP1IE | 176 |
| PIR3 | RC2IF | TX2IF | RC1IF | TX1IF | BCL2IF | SSP2IF | BCL1IF | SSP1IF | 176 |
| IPR3 | RC2IP | TX2IP | RC1IP | TX1IP | BCL2IP | SSP2IP | BCL1IP | SSP1IP | 197 |
| PIE4 | — | — | RC5IE | TX5IE | RC4IE | TX4IE | RC3IE | TX3IE | 188 |
| PIR4 | — | — | RC5IF | TX5IF | RC4IF | TX4IF | RC3IF | TX3IF | 177 |
| IPR4 | — | — | RC5IP | TX5IP | RC4IP | TX4IP | RC3IP | TX3IP | 198 |
| RCxSTA | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 445 |
| RxyPPS | — | — | RxyPPS<5:0> | | | | | | 227 |
| TXxPPS | — | — | TXPPS<5:0> | | | | | | 224 |
| SPxBRGH | EUSARTx Baud Rate Generator, High Byte | | | | | | | | 455* |
| SPxBRGL | EUSARTx Baud Rate Generator, Low Byte | | | | | | | | 455* |
| TXxREG | EUSARTx Transmit Register | | | | | | | | 447* |
| TXxSTA | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D | 444 |

Legend: — = unimplemented location, read as '0'. Shaded cells are not used for asynchronous transmission.

* Page provides register information.

28.2.2 EUSART ASYNCHRONOUS RECEIVER

The Asynchronous mode is typically used in RS-232 systems. The receiver block diagram is shown in [Figure 28-2](#). The data is received on the RXx/DTx pin and drives the data recovery block. The data recovery block is actually a high-speed shifter operating at 16 times the baud rate, whereas the serial Receive Shift Register (RSR) operates at the bit rate. When all eight or nine 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 start servicing the EUSART receiver. The FIFO and RSR registers are not directly accessible by software. Access to the received data is via the RCxREG register.

28.2.2.1 Enabling the Receiver

The EUSART receiver is enabled for asynchronous operation by configuring the following three control bits:

- CREN = 1
- SYNC = 0
- SPEN = 1

All other EUSART control bits are assumed to be in their default state.

Setting the CREN bit of the RCxSTA register enables the receiver circuitry of the EUSART. Clearing the SYNC bit of the TXxSTA register configures the EUSART for asynchronous operation. Setting the SPEN bit of the RCxSTA register enables the EUSART. The programmer must set the corresponding TRIS bit to configure the RXx/DTx I/O pin as an input.

Note: If the RX/DT function is on an analog pin, the corresponding ANSEL bit must be cleared for the receiver to function.

28.2.2.2 Receiving Data

The receiver data recovery circuit initiates character reception on the falling edge of the first bit. The first bit, also known as the Start bit, is always a zero. The data recovery circuit counts one-half bit time to the center of the Start bit and verifies that the bit is still a zero. If it is not a zero then the data recovery circuit aborts character reception, without generating an error, and resumes looking for the falling edge of the Start bit. If the Start bit zero verification succeeds then the data recovery circuit counts a full bit time to the center of the next bit. The bit is then sampled by a majority detect circuit and the resulting '0' or '1' is shifted into the RSR. This repeats until all data bits have been sampled and shifted into the RSR. One final bit time is measured and the level sampled. This is the Stop bit, which is always a '1'. If the data recovery circuit samples a '0' in the Stop bit position then a framing error is set for this character, otherwise the framing error is cleared for this character. See [Section 28.2.2.4 "Receive Framing Error"](#) for more information on framing errors.

Immediately after all data bits and the Stop bit have been received, the character in the RSR is transferred to the EUSART receive FIFO and the RCxIF interrupt flag bit of the PIR3/4 registers are set. The top character in the FIFO is transferred out of the FIFO by reading the RCxREG register.

Note: If the receive FIFO is overrun, no additional characters will be received until the overrun condition is cleared. See [Section 28.2.2.5 "Receive Overrun Error"](#) for more information on overrun errors.

28.2.2.3 Receive Interrupts

The RCxIF interrupt flag bit of the PIR3/4 registers are set whenever the EUSART receiver is enabled and there is an unread character in the receive FIFO. The RCxIF interrupt flag bit is read-only, it cannot be set or cleared by software.

RCxIF interrupts are enabled by setting all of the following bits:

- RCxIE, Interrupt Enable bit of the PIE3/4 registers
- PEIE, Peripheral Interrupt Enable bit of the INTCON register
- GIE, Global Interrupt Enable bit of the INTCON register

The RCxIF interrupt flag bit will be set when there is an unread character in the FIFO, regardless of the state of interrupt enable bits.

28.2.2.4 Receive Framing Error

Each character in the receive FIFO buffer has a corresponding framing error Status bit. A framing error indicates that a Stop bit was not seen at the expected time. The framing error status is accessed via the FERR bit of the RCxSTA register. The FERR bit represents the status of the top unread character in the receive FIFO. Therefore, the FERR bit must be read before reading the RCxREG.

The FERR bit is read-only and only applies to the top unread character in the receive FIFO. A framing error (FERR = 1) does not preclude reception of additional characters. It is not necessary to clear the FERR bit. Reading the next character from the FIFO buffer will advance the FIFO to the next character and the next corresponding framing error.

The FERR bit can be forced clear by clearing the SPEN bit of the RCxSTA register which resets the EUSART. Clearing the CREN bit of the RCxSTA register does not affect the FERR bit. A framing error by itself does not generate an interrupt.

| |
|---|
| Note: If all receive characters in the receive FIFO have framing errors, repeated reads of the RCxREG will not clear the FERR bit. |
|---|

28.2.2.5 Receive Overrun Error

The receive FIFO buffer can hold two characters. An overrun error will be generated if a third character, in its entirety, is received before the FIFO is accessed. When this happens the OERR bit of the RCxSTA register is set. The characters already in the FIFO buffer can be read but no additional characters will be received until the error is cleared. The error must be cleared by either clearing the CREN bit of the RCxSTA register or by resetting the EUSART by clearing the SPEN bit of the RCxSTA register.

28.2.2.6 Receiving 9-Bit Characters

The EUSART supports 9-bit character reception. When the RX9 bit of the RCxSTA register is set the EUSART will shift nine bits into the RSR for each character received. The RX9D bit of the RCxSTA register is the ninth and Most Significant data bit of the top unread character in the receive FIFO. When reading 9-bit data from the receive FIFO buffer, the RX9D data bit must be read before reading the eight Least Significant bits from the RCxREG.

28.2.2.7 Address Detection

A special Address Detection mode is available for use when multiple receivers share the same transmission line, such as in RS-485 systems. Address detection is enabled by setting the ADDEN bit of the RCxSTA register.

Address detection requires 9-bit character reception. When address detection is enabled, only characters with the ninth data bit set will be transferred to the receive FIFO buffer, thereby setting the RCxIF interrupt bit. All other characters will be ignored.

Upon receiving an address character, user software determines if the address matches its own. Upon address match, user software must disable address detection by clearing the ADDEN bit before the next Stop bit occurs. When user software detects the end of the message, determined by the message protocol used, software places the receiver back into the Address Detection mode by setting the ADDEN bit.

28.2.2.8 Asynchronous Reception Setup:

1. Initialize the SPxBRGH:SPxBRGL register pair and the BRGH and BRG16 bits to achieve the desired baud rate (see [Section 28.4 “EUSART Baud Rate Generator \(BRG\)”](#)).
2. Clear the ANSEL bit for the RXx pin (if applicable).
3. Enable the serial port by setting the SPEN bit. The SYNC bit must be clear for asynchronous operation.
4. If interrupts are desired, set the RCxIE bit of the PIE3/4 registers and the GIE and PEIE bits of the INTCON register.
5. If 9-bit reception is desired, set the RX9 bit.
6. Enable reception by setting the CREN bit.
7. The RCxIF 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 RCxIE interrupt enable bit was also set.
8. Read the RCxSTA register to get the error flags and, if 9-bit data reception is enabled, the ninth data bit.
9. Get the received eight Least Significant data bits from the receive buffer by reading the RCxREG register.
10. If an overrun occurred, clear the OERR flag by clearing the CREN receiver enable bit.

28.2.2.9 9-Bit Address Detection Mode Setup

This mode would typically be used in RS-485 systems. To set up an Asynchronous Reception with Address Detect Enable:

1. Initialize the SPxBRGH:SPxBRGL register pair and the BRGH and BRG16 bits to achieve the desired baud rate (see [Section 28.4 “EUSART Baud Rate Generator \(BRG\)”](#)).
2. Clear the ANSEL bit for the RXx pin (if applicable).
3. Enable the serial port by setting the SPEN bit. The SYNC bit must be clear for asynchronous operation.
4. If interrupts are desired, set the RCxIE bit of the PIE3/4 registers and the GIE and PEIE bits of the INTCON register.
5. Enable 9-bit reception by setting the RX9 bit.
6. Enable address detection by setting the ADDEN bit.
7. Enable reception by setting the CREN bit.
8. The RCxIF interrupt flag bit will be set when a character with the ninth bit set is transferred from the RSR to the receive buffer. An interrupt will be generated if the RCxIE interrupt enable bit was also set.
9. Read the RCxSTA register to get the error flags. The ninth data bit will always be set.
10. Get the received eight Least Significant data bits from the receive buffer by reading the RCxREG register. Software determines if this is the device's address.
11. If an overrun occurred, clear the OERR flag by clearing the CREN receiver enable bit.
12. If the device has been addressed, clear the ADDEN bit to allow all received data into the receive buffer and generate interrupts.

FIGURE 28-5: ASYNCHRONOUS RECEPTION

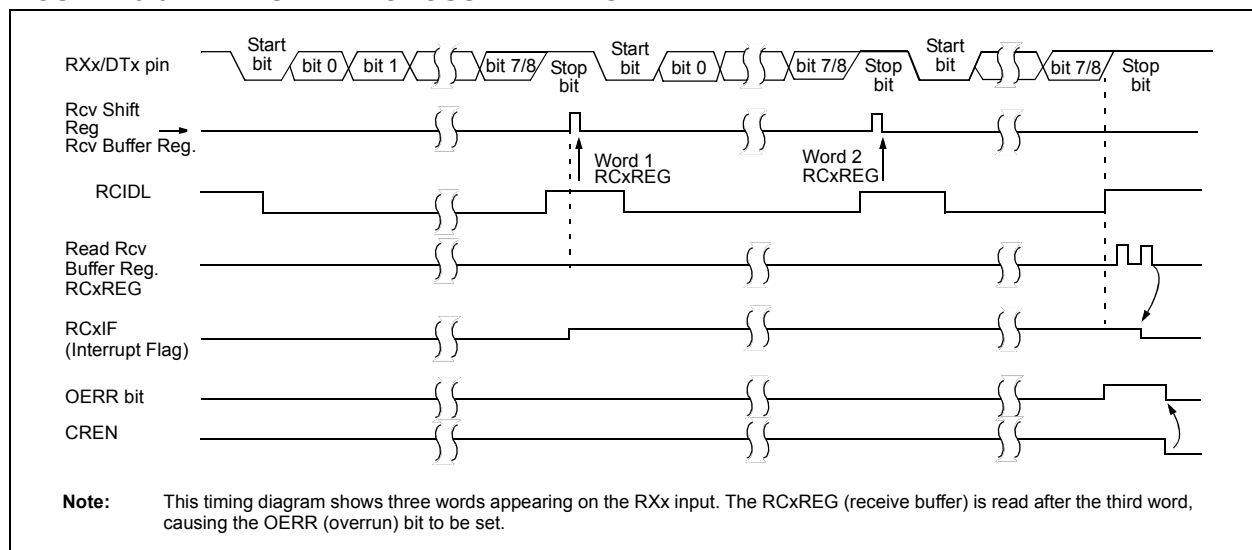


TABLE 28-2: SUMMARY OF REGISTERS ASSOCIATED WITH ASYNCHRONOUS RECEPTION

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on Page |
|----------|--|-----------|-------------|-------|---------|---------|---------|---------|------------------|
| BAUDxCON | ABDOVF | RCIDL | — | SCKP | BRG16 | — | WUE | ABDEN | 446 |
| INTCON | GIE/GIEH | PEIE/GIEL | IPEN | — | INT3EDG | INT2EDG | INT1EDG | INT0EDG | 172 |
| PIE3 | RC2IE | TX2IE | RC1IE | TX1IE | BCL2IE | SSP2IE | BCL1IE | SSP1IE | 187 |
| PIR3 | RC2IF | TX2IF | RC1IF | TX1IF | BCL2IF | SSP2IF | BCL1IF | SSP1IF | 176 |
| IPR3 | RC2IP | TX2IP | RC1IP | TX1IP | BCL2IP | SSP2IP | BCL1IP | SSP1IP | 197 |
| PIE4 | — | — | RC5IE | TX5IE | RC4IE | TX4IE | RC3IE | TX3IE | 188 |
| PIR4 | — | — | RC5IF | TX5IF | RC4IF | TX4IF | RC3IF | TX3IF | 177 |
| IPR4 | — | — | RC5IP | TX5IP | RC4IP | TX4IP | RC3IP | TX3IP | 198 |
| RCxREG | EUSARTx Receive Register | | | | | | | | 450* |
| RCxSTA | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 445 |
| RxyPPS | — | — | RxyPPS<5:0> | | | | | | 227 |
| RXxPPS | — | — | RXPPS<5:0> | | | | | | 224 |
| SPxBRGH | EUSARTx Baud Rate Generator, High Byte | | | | | | | | 455* |
| SPxBRGL | EUSARTx Baud Rate Generator, Low Byte | | | | | | | | 455* |
| TXxSTA | CSRC | TX9 | TXEN | SYNC | SENDDB | BRGH | TRMT | TX9D | 444 |

Legend: — = unimplemented location, read as '0'. Shaded cells are not used for asynchronous reception.

* Page provides register information.

28.3 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 in the OSCTUNE register allows for fine resolution changes to the system clock source. See [Section 4.3.2.3 “Internal Oscillator Frequency Adjustment”](#) for more information.

The other method adjusts the value in the Baud Rate Generator. This can be done automatically with the Auto-Baud Detect feature (see [Section 28.4.1 “Auto-Baud Detect”](#)). There may not be fine enough resolution when adjusting the Baud Rate Generator to compensate for a gradual change in the peripheral clock frequency.

28.4 EUSART Baud Rate Generator (BRG)

The Baud Rate Generator (BRG) is an 8-bit or 16-bit timer that is dedicated to the support of both the asynchronous and synchronous EUSART operation. By default, the BRG operates in 8-bit mode. Setting the BRG16 bit of the BAUDxCON register selects 16-bit mode.

The SPxBRGH, SPxBRGL register pair determines the period of the free running baud rate timer. In Asynchronous mode the multiplier of the baud rate period is determined by both the BRGH bit of the TXxSTA register and the BRG16 bit of the BAUDxCON register. In Synchronous mode, the BRGH bit is ignored.

Table 28-3 contains the formulas for determining the baud rate. Example 28-1 provides a sample calculation for determining the baud rate and baud rate error.

Typical baud rates and error values for various asynchronous modes have been computed for your convenience and are shown in Table 28-5. It may be advantageous to use the high baud rate (BRGH = 1), or the 16-bit BRG (BRG16 = 1) to reduce the baud rate error. The 16-bit BRG mode is used to achieve slow baud rates for fast oscillator frequencies.

Writing a new value to the SPxBRGH, SPxBRGL register pair 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 RCIDL bit to make sure that the receive operation is idle before changing the system clock.

EXAMPLE 28-1: CALCULATING BAUD RATE ERROR

For a device with FOSC of 16 MHz, desired baud rate of 9600, Asynchronous mode, 8-bit BRG:

$$\text{Desired Baud Rate} = \frac{F_{OSC}}{64(SP_xBRGH:SP_xBRGL + 1)}$$

Solving for SPxBRGH:SPxBRGL:

$$SPBRGH:SPBRGL = \frac{F_{OSC}}{\text{Desired Baud Rate} \cdot 64} - 1$$

$$= \frac{16000000}{9600 \cdot 64} - 1$$

$$= [25.042] = 25$$

$$\text{Calculated Baud Rate} = \frac{16000000}{64(25 + 1)}$$

$$= 9615$$

$$\text{Error} = \frac{\text{Calc. Baud Rate} - \text{Desired Baud Rate}}{\text{Desired Baud Rate}}$$

$$= \frac{(9615 - 9600)}{9600} = 0.16\%$$

TABLE 28-3: BAUD RATE FORMULAS

| Configuration Bits | | | BRG/EUSART Mode | Baud Rate Formula |
|--------------------|-------|------|---------------------|----------------------|
| SYNC | BRG16 | BRGH | | |
| 0 | 0 | 0 | 8-bit/Asynchronous | $F_{osc}/[64 (n+1)]$ |
| 0 | 0 | 1 | 8-bit/Asynchronous | $F_{osc}/[16 (n+1)]$ |
| 0 | 1 | 0 | 16-bit/Asynchronous | |
| 0 | 1 | 1 | 16-bit/Asynchronous | $F_{osc}/[4 (n+1)]$ |
| 1 | 0 | x | 8-bit/Synchronous | |
| 1 | 1 | x | 16-bit/Synchronous | |

Legend: x = Don't care, n = value of SPxBRGH:SPxBRGL register pair.

TABLE 28-4: SUMMARY OF REGISTERS ASSOCIATED WITH THE BAUD RATE GENERATOR

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on Page |
|----------|--|-------|-------|-------|-------|-------|-------|-------|------------------|
| BAUDxCON | ABDOVF | RCIDL | — | SCKP | BRG16 | — | WUE | ABDEN | 446 |
| RCxSTA | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 445 |
| SPxBRGH | EUSARTx Baud Rate Generator, High Byte | | | | | | | | 455* |
| SPxBRGL | EUSARTx Baud Rate Generator, Low Byte | | | | | | | | 455* |
| TXxSTA | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D | 444 |

Legend: — = unimplemented location, read as '0'. Shaded cells are not used for the Baud Rate Generator.

* Page provides register information.

TABLE 28-5: SAMPLE BAUD RATES FOR ASYNCHRONOUS MODES

| BAUD RATE | SYNC = 0, BRGH = 0, BRG16 = 0 | | | | | | | | | | | |
|-----------|-------------------------------|---------|-----------------------|-------------------|---------|-----------------------|-------------------|---------|-----------------------|--------------------|---------|-----------------------|
| | Fosc = 32.000 MHz | | | Fosc = 20.000 MHz | | | Fosc = 18.432 MHz | | | Fosc = 11.0592 MHz | | |
| | Actual Rate | % Error | SPBRG value (decimal) | Actual Rate | % Error | SPBRG value (decimal) | Actual Rate | % Error | SPBRG value (decimal) | Actual Rate | % Error | SPBRG value (decimal) |
| 300 | — | — | — | — | — | — | — | — | — | — | — | — |
| 1200 | — | — | — | 1221 | 1.73 | 255 | 1200 | 0.00 | 239 | 1200 | 0.00 | 143 |
| 2400 | 2404 | 0.16 | 207 | 2404 | 0.16 | 129 | 2400 | 0.00 | 119 | 2400 | 0.00 | 71 |
| 9600 | 9615 | 0.16 | 51 | 9470 | -1.36 | 32 | 9600 | 0.00 | 29 | 9600 | 0.00 | 17 |
| 10417 | 10417 | 0.00 | 47 | 10417 | 0.00 | 29 | 10286 | -1.26 | 27 | 10165 | -2.42 | 16 |
| 19.2k | 19.23k | 0.16 | 25 | 19.53k | 1.73 | 15 | 19.20k | 0.00 | 14 | 19.20k | 0.00 | 8 |
| 57.6k | 55.55k | -3.55 | 3 | — | — | — | 57.60k | 0.00 | 7 | 57.60k | 0.00 | 2 |
| 115.2k | — | — | — | — | — | — | — | — | — | — | — | — |

| BAUD RATE | SYNC = 0, BRGH = 0, BRG16 = 0 | | | | | | | | | | | |
|-----------|-------------------------------|---------|-----------------------|------------------|---------|-----------------------|-------------------|---------|-----------------------|------------------|---------|-----------------------|
| | Fosc = 8.000 MHz | | | Fosc = 4.000 MHz | | | Fosc = 3.6864 MHz | | | Fosc = 1.000 MHz | | |
| | Actual Rate | % Error | SPBRG value (decimal) | Actual Rate | % Error | SPBRG value (decimal) | Actual Rate | % Error | SPBRG value (decimal) | Actual Rate | % Error | SPBRG value (decimal) |
| 300 | — | — | — | 300 | 0.16 | 207 | 300 | 0.00 | 191 | 300 | 0.16 | 51 |
| 1200 | 1202 | 0.16 | 103 | 1202 | 0.16 | 51 | 1200 | 0.00 | 47 | 1202 | 0.16 | 12 |
| 2400 | 2404 | 0.16 | 51 | 2404 | 0.16 | 25 | 2400 | 0.00 | 23 | — | — | — |
| 9600 | 9615 | 0.16 | 12 | — | — | — | 9600 | 0.00 | 5 | — | — | — |
| 10417 | 10417 | 0.00 | 11 | 10417 | 0.00 | 5 | — | — | — | — | — | — |
| 19.2k | — | — | — | — | — | — | 19.20k | 0.00 | 2 | — | — | — |
| 57.6k | — | — | — | — | — | — | 57.60k | 0.00 | 0 | — | — | — |
| 115.2k | — | — | — | — | — | — | — | — | — | — | — | — |

| BAUD RATE | SYNC = 0, BRGH = 1, BRG16 = 0 | | | | | | | | | | | |
|-----------|-------------------------------|---------|-----------------------|-------------------|---------|-----------------------|-------------------|---------|-----------------------|--------------------|---------|-----------------------|
| | Fosc = 32.000 MHz | | | Fosc = 20.000 MHz | | | Fosc = 18.432 MHz | | | Fosc = 11.0592 MHz | | |
| | Actual Rate | % Error | SPBRG value (decimal) | Actual Rate | % Error | SPBRG value (decimal) | Actual Rate | % Error | SPBRG value (decimal) | Actual Rate | % Error | SPBRG value (decimal) |
| 300 | — | — | — | — | — | — | — | — | — | — | — | — |
| 1200 | — | — | — | — | — | — | — | — | — | — | — | — |
| 2400 | — | — | — | — | — | — | — | — | — | — | — | — |
| 9600 | 9615 | 0.16 | 207 | 9615 | 0.16 | 129 | 9600 | 0.00 | 119 | 9600 | 0.00 | 71 |
| 10417 | 10417 | 0.00 | 191 | 10417 | 0.00 | 119 | 10378 | -0.37 | 110 | 10473 | 0.53 | 65 |
| 19.2k | 19.23k | 0.16 | 103 | 19.23k | 0.16 | 64 | 19.20k | 0.00 | 59 | 19.20k | 0.00 | 35 |
| 57.6k | 57.14k | -0.79 | 34 | 56.82k | -1.36 | 21 | 57.60k | 0.00 | 19 | 57.60k | 0.00 | 11 |
| 115.2k | 117.64k | 2.12 | 16 | 113.64k | -1.36 | 10 | 115.2k | 0.00 | 9 | 115.2k | 0.00 | 5 |

PIC18(L)F67K40

TABLE 28-5: SAMPLE BAUD RATES FOR ASYNCHRONOUS MODES (CONTINUED)

| BAUD RATE | SYNC = 0, BRGH = 1, BRG16 = 0 | | | | | | | | | | | |
|-----------|-------------------------------|---------|-----------------------|------------------|---------|-----------------------|-------------------|---------|-----------------------|------------------|---------|-----------------------|
| | Fosc = 8.000 MHz | | | Fosc = 4.000 MHz | | | Fosc = 3.6864 MHz | | | Fosc = 1.000 MHz | | |
| | Actual Rate | % Error | SPBRG value (decimal) | Actual Rate | % Error | SPBRG value (decimal) | Actual Rate | % Error | SPBRG value (decimal) | Actual Rate | % Error | SPBRG value (decimal) |
| 300 | — | — | — | — | — | — | — | — | — | 300 | 0.16 | 207 |
| 1200 | — | — | — | 1202 | 0.16 | 207 | 1200 | 0.00 | 191 | 1202 | 0.16 | 51 |
| 2400 | 2404 | 0.16 | 207 | 2404 | 0.16 | 103 | 2400 | 0.00 | 95 | 2404 | 0.16 | 25 |
| 9600 | 9615 | 0.16 | 51 | 9615 | 0.16 | 25 | 9600 | 0.00 | 23 | — | — | — |
| 10417 | 10417 | 0.00 | 47 | 10417 | 0.00 | 23 | 10473 | 0.53 | 21 | 10417 | 0.00 | 5 |
| 19.2k | 19231 | 0.16 | 25 | 19.23k | 0.16 | 12 | 19.2k | 0.00 | 11 | — | — | — |
| 57.6k | 55556 | -3.55 | 8 | — | — | — | 57.60k | 0.00 | 3 | — | — | — |
| 115.2k | — | — | — | — | — | — | 115.2k | 0.00 | 1 | — | — | — |

| BAUD RATE | SYNC = 0, BRGH = 0, BRG16 = 1 | | | | | | | | | | | |
|-----------|-------------------------------|---------|-----------------------|-------------------|---------|-----------------------|-------------------|---------|-----------------------|--------------------|---------|-----------------------|
| | Fosc = 32.000 MHz | | | Fosc = 20.000 MHz | | | Fosc = 18.432 MHz | | | Fosc = 11.0592 MHz | | |
| | Actual Rate | % Error | SPBRG value (decimal) | Actual Rate | % Error | SPBRG value (decimal) | Actual Rate | % Error | SPBRG value (decimal) | Actual Rate | % Error | SPBRG value (decimal) |
| 300 | 300.0 | 0.00 | 6666 | 300.0 | -0.01 | 4166 | 300.0 | 0.00 | 3839 | 300.0 | 0.00 | 2303 |
| 1200 | 1200 | -0.02 | 3332 | 1200 | -0.03 | 1041 | 1200 | 0.00 | 959 | 1200 | 0.00 | 575 |
| 2400 | 2401 | -0.04 | 832 | 2399 | -0.03 | 520 | 2400 | 0.00 | 479 | 2400 | 0.00 | 287 |
| 9600 | 9615 | 0.16 | 207 | 9615 | 0.16 | 129 | 9600 | 0.00 | 119 | 9600 | 0.00 | 71 |
| 10417 | 10417 | 0.00 | 191 | 10417 | 0.00 | 119 | 10378 | -0.37 | 110 | 10473 | 0.53 | 65 |
| 19.2k | 19.23k | 0.16 | 103 | 19.23k | 0.16 | 64 | 19.20k | 0.00 | 59 | 19.20k | 0.00 | 35 |
| 57.6k | 57.14k | -0.79 | 34 | 56.818 | -1.36 | 21 | 57.60k | 0.00 | 19 | 57.60k | 0.00 | 11 |
| 115.2k | 117.6k | 2.12 | 16 | 113.636 | -1.36 | 10 | 115.2k | 0.00 | 9 | 115.2k | 0.00 | 5 |

| BAUD RATE | SYNC = 0, BRGH = 0, BRG16 = 1 | | | | | | | | | | | |
|-----------|-------------------------------|---------|-----------------------|------------------|---------|-----------------------|-------------------|---------|-----------------------|------------------|---------|-----------------------|
| | Fosc = 8.000 MHz | | | Fosc = 4.000 MHz | | | Fosc = 3.6864 MHz | | | Fosc = 1.000 MHz | | |
| | Actual Rate | % Error | SPBRG value (decimal) | Actual Rate | % Error | SPBRG value (decimal) | Actual Rate | % Error | SPBRG value (decimal) | Actual Rate | % Error | SPBRG value (decimal) |
| 300 | 299.9 | -0.02 | 1666 | 300.1 | 0.04 | 832 | 300.0 | 0.00 | 767 | 300.5 | 0.16 | 207 |
| 1200 | 1199 | -0.08 | 416 | 1202 | 0.16 | 207 | 1200 | 0.00 | 191 | 1202 | 0.16 | 51 |
| 2400 | 2404 | 0.16 | 207 | 2404 | 0.16 | 103 | 2400 | 0.00 | 95 | 2404 | 0.16 | 25 |
| 9600 | 9615 | 0.16 | 51 | 9615 | 0.16 | 25 | 9600 | 0.00 | 23 | — | — | — |
| 10417 | 10417 | 0.00 | 47 | 10417 | 0.00 | 23 | 10473 | 0.53 | 21 | 10417 | 0.00 | 5 |
| 19.2k | 19.23k | 0.16 | 25 | 19.23k | 0.16 | 12 | 19.20k | 0.00 | 11 | — | — | — |
| 57.6k | 55556 | -3.55 | 8 | — | — | — | 57.60k | 0.00 | 3 | — | — | — |
| 115.2k | — | — | — | — | — | — | 115.2k | 0.00 | 1 | — | — | — |

PIC18(L)F67K40

TABLE 28-5: SAMPLE BAUD RATES FOR ASYNCHRONOUS MODES (CONTINUED)

| BAUD RATE | SYNC = 0, BRGH = 1, BRG16 = 1 or SYNC = 1, BRG16 = 1 | | | | | | | | | | | |
|-----------|--|---------|-----------------------|-------------------|---------|-----------------------|-------------------|---------|-----------------------|--------------------|---------|-----------------------|
| | Fosc = 32.000 MHz | | | Fosc = 20.000 MHz | | | Fosc = 18.432 MHz | | | Fosc = 11.0592 MHz | | |
| | Actual Rate | % Error | SPBRG value (decimal) | Actual Rate | % Error | SPBRG value (decimal) | Actual Rate | % Error | SPBRG value (decimal) | Actual Rate | % Error | SPBRG value (decimal) |
| 300 | 300.0 | 0.00 | 26666 | 300.0 | 0.00 | 16665 | 300.0 | 0.00 | 15359 | 300.0 | 0.00 | 9215 |
| 1200 | 1200 | 0.00 | 6666 | 1200 | -0.01 | 4166 | 1200 | 0.00 | 3839 | 1200 | 0.00 | 2303 |
| 2400 | 2400 | 0.01 | 3332 | 2400 | 0.02 | 2082 | 2400 | 0.00 | 1919 | 2400 | 0.00 | 1151 |
| 9600 | 9604 | 0.04 | 832 | 9597 | -0.03 | 520 | 9600 | 0.00 | 479 | 9600 | 0.00 | 287 |
| 10417 | 10417 | 0.00 | 767 | 10417 | 0.00 | 479 | 10425 | 0.08 | 441 | 10433 | 0.16 | 264 |
| 19.2k | 19.18k | -0.08 | 416 | 19.23k | 0.16 | 259 | 19.20k | 0.00 | 239 | 19.20k | 0.00 | 143 |
| 57.6k | 57.55k | -0.08 | 138 | 57.47k | -0.22 | 86 | 57.60k | 0.00 | 79 | 57.60k | 0.00 | 47 |
| 115.2k | 115.9k | 0.64 | 68 | 116.3k | 0.94 | 42 | 115.2k | 0.00 | 39 | 115.2k | 0.00 | 23 |

| BAUD RATE | SYNC = 0, BRGH = 1, BRG16 = 1 or SYNC = 1, BRG16 = 1 | | | | | | | | | | | |
|-----------|--|---------|-----------------------|------------------|---------|-----------------------|-------------------|---------|-----------------------|------------------|---------|-----------------------|
| | Fosc = 8.000 MHz | | | Fosc = 4.000 MHz | | | Fosc = 3.6864 MHz | | | Fosc = 1.000 MHz | | |
| | Actual Rate | % Error | SPBRG value (decimal) | Actual Rate | % Error | SPBRG value (decimal) | Actual Rate | % Error | SPBRG value (decimal) | Actual Rate | % Error | SPBRG value (decimal) |
| 300 | 300.0 | 0.00 | 6666 | 300.0 | 0.01 | 3332 | 300.0 | 0.00 | 3071 | 300.1 | 0.04 | 832 |
| 1200 | 1200 | -0.02 | 1666 | 1200 | 0.04 | 832 | 1200 | 0.00 | 767 | 1202 | 0.16 | 207 |
| 2400 | 2401 | 0.04 | 832 | 2398 | 0.08 | 416 | 2400 | 0.00 | 383 | 2404 | 0.16 | 103 |
| 9600 | 9615 | 0.16 | 207 | 9615 | 0.16 | 103 | 9600 | 0.00 | 95 | 9615 | 0.16 | 25 |
| 10417 | 10417 | 0 | 191 | 10417 | 0.00 | 95 | 10473 | 0.53 | 87 | 10417 | 0.00 | 23 |
| 19.2k | 19.23k | 0.16 | 103 | 19.23k | 0.16 | 51 | 19.20k | 0.00 | 47 | 19.23k | 0.16 | 12 |
| 57.6k | 57.14k | -0.79 | 34 | 58.82k | 2.12 | 16 | 57.60k | 0.00 | 15 | — | — | — |
| 115.2k | 117.6k | 2.12 | 16 | 111.1k | -3.55 | 8 | 115.2k | 0.00 | 7 | — | — | — |

28.4.1 AUTO-BAUD DETECT

The EUSART module supports automatic detection and calibration of the baud rate.

In the Auto-Baud Detect (ABD) mode, 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 rising edges including the Stop bit edge.

Setting the ABDEN bit of the BAUDxCON register starts the auto-baud calibration sequence. While the ABD sequence takes place, the EUSART state machine is held in Idle. On the first rising edge of the receive line, after the Start bit, the SPxBRG begins counting up using the BRG counter clock as shown in Figure 28-6. The fifth rising edge will occur on the RX pin at the end of the eighth bit period. At that time, an accumulated value totaling the proper BRG period is left in the SPxBRGH, SPxBRGL register pair, the ABDEN bit is automatically cleared and the RCxIF interrupt flag is set. The value in the RCxREG needs to be read to clear the RCxIF interrupt. RCxREG content should be discarded. When calibrating for modes that do not use the SPxBRGH register the user can verify that the SPxBRGL register did not overflow by checking for 00h in the SPxBRGH register.

The BRG auto-baud clock is determined by the BRG16 and BRGH bits as shown in Table 28-6. During ABD, both the SPxBRGH and SPxBRGL registers are used as a 16-bit counter, independent of the BRG16 bit setting. While calibrating the baud rate period, the SPx-

BRGH and SPxBRGL registers are clocked at 1/8th the BRG base clock rate. The resulting byte measurement is the average bit time when clocked at full speed.

Note 1: If the WUE bit is set with the ABDEN bit, auto-baud detection will occur on the byte following the Break character (see Section 28.4.3 “Auto-Wake-up on Break”).

2: It is up to the user to determine that the incoming character baud rate is within the range of the selected BRG clock source. Some combinations of oscillator frequency and EUSART baud rates are not possible.

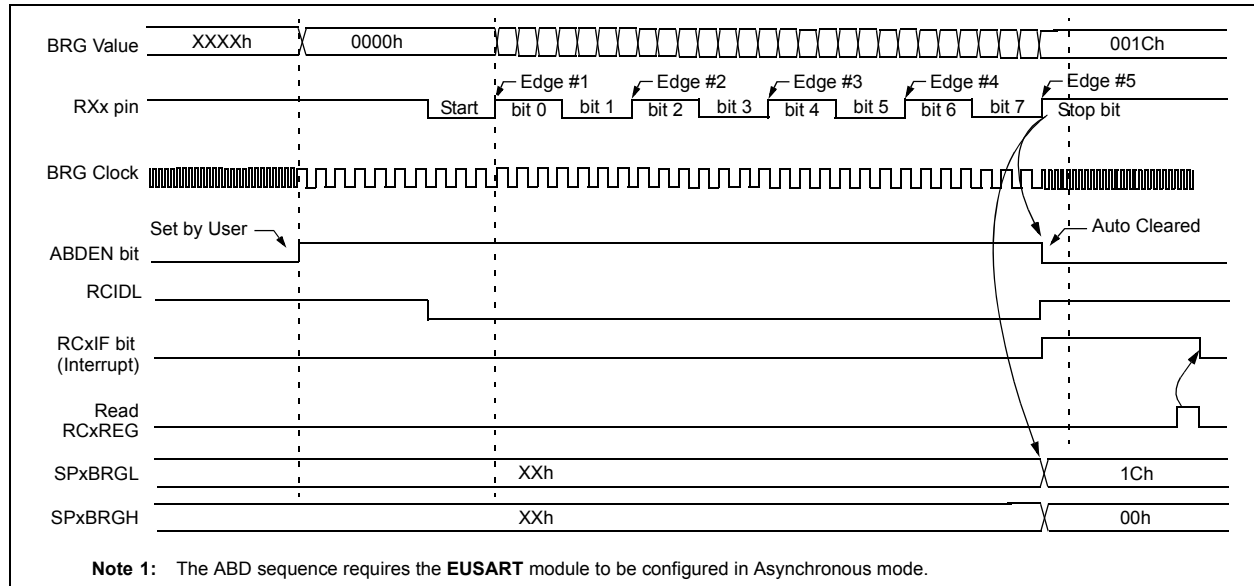
3: During the auto-baud process, the auto-baud counter starts counting at one. Upon completion of the auto-baud sequence, to achieve maximum accuracy, subtract 1 from the SPxBRGH:SPxBRGL register pair.

TABLE 28-6: BRG COUNTER CLOCK RATES

| BRG16 | BRGH | BRG Base Clock | BRG ABD Clock |
|-------|------|----------------|---------------|
| 1 | 1 | Fosc/4 | Fosc/32 |
| 1 | 0 | Fosc/16 | Fosc/128 |
| 0 | 1 | Fosc/16 | Fosc/128 |
| 0 | 0 | Fosc/64 | Fosc/512 |

Note: During the ABD sequence, SPxBRGL and SPxBRGH registers are both used as a 16-bit counter, independent of the BRG16 setting.

FIGURE 28-6: AUTOMATIC BAUD RATE CALIBRATION



28.4.2 AUTO-BAUD OVERFLOW

During the course of automatic baud detection, the ABDOVF bit of the BAUDxCON register will be set if the baud rate counter overflows before the fifth rising edge is detected on the RXx pin. The ABDOVF bit indicates that the counter has exceeded the maximum count that can fit in the 16 bits of the SPxBRGH:SPxBRGL register pair. After the ABDOVF bit has been set, the counter continues to count until the fifth rising edge is detected on the RXx pin. Upon detecting the fifth RX edge, the hardware will set the RCxIF interrupt flag and clear the ABDEN bit of the BAUDxCON register. The RCxIF flag can be subsequently cleared by reading the RCxREG register. The ABDOVF flag of the BAUDxCON register can be cleared by software directly.

To terminate the auto-baud process before the RCxIF flag is set, clear the ABDEN bit then clear the ABDOVF bit of the BAUDxCON register. The ABDOVF bit will remain set if the ABDEN bit is not cleared first.

28.4.3 AUTO-WAKE-UP ON BREAK

During Sleep mode, all clocks to the EUSART are suspended. Because of this, the Baud Rate Generator is inactive and a proper character reception cannot be performed. The Auto-Wake-up feature allows the controller to wake-up due to activity on the RX/DT line. This feature is available only in Asynchronous mode.

The Auto-Wake-up feature is enabled by setting the WUE bit of the BAUDxCON register. Once set, the normal receive sequence on RX/DT is disabled, and the EUSART remains in an Idle state, monitoring for a wake-up event independent of the CPU mode. A wake-up event consists of a high-to-low transition on the RX/DT line. (This coincides with the start of a Sync Break or a wake-up signal character for the LIN protocol.)

The EUSART module generates an RCxIF interrupt coincident with the wake-up event. The interrupt is generated synchronously to the Q clocks in normal CPU operating modes (Figure 28-7), and asynchronously if the device is in Sleep mode (Figure 28-8). The interrupt condition is cleared by reading the RCxREG register.

The WUE bit is automatically cleared by the low-to-high transition 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 EUSART module is in Idle mode waiting to receive the next character.

28.4.3.1 Special Considerations

Break Character

To avoid character errors or character fragments during a wake-up event, the wake-up character must be all zeros.

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 non-zero 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 in the character will be received as a fragmented character and subsequent characters can result in framing or overrun errors.

Therefore, the initial character in the transmission must be all '0's. This must be ten 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

Oscillator start-up time must be considered, especially in applications using oscillators with longer start-up intervals (i.e., LP, XT or HS/PLL mode). 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 EUSART.

WUE Bit

The wake-up event causes a receive interrupt by setting the RCxIF bit. The WUE bit is cleared in hardware by a rising edge on RX/DT. The interrupt condition is then cleared in software by reading the RCxREG register and discarding its contents.

To ensure that no actual data is lost, check the RCIDL 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.

FIGURE 28-7: AUTO-WAKE-UP BIT (WUE) TIMING DURING NORMAL OPERATION

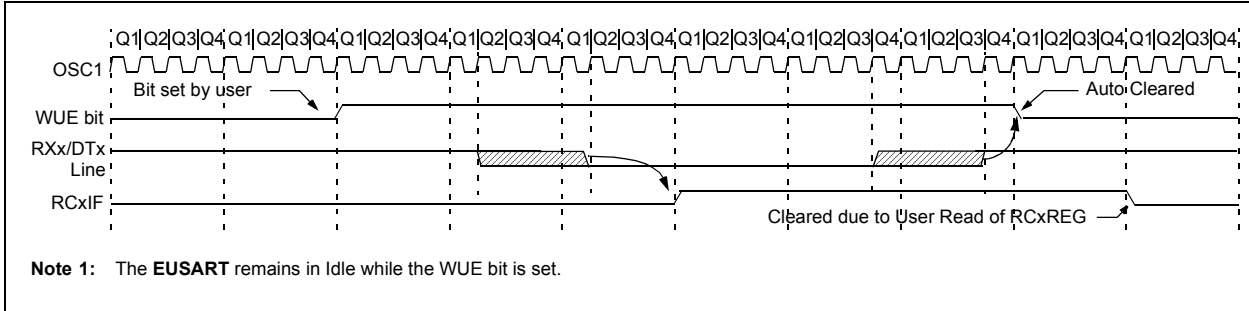
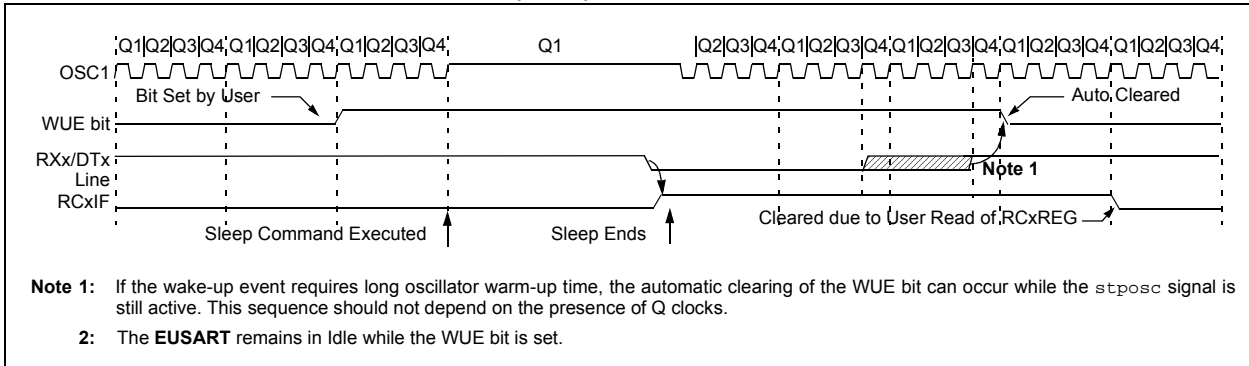


FIGURE 28-8: AUTO-WAKE-UP BIT (WUE) TIMINGS DURING SLEEP



28.4.4 BREAK CHARACTER SEQUENCE

The EUSART module has the capability of sending the special Break character sequences that are required by the LIN bus standard. A Break character consists of a Start bit, followed by 12 '0' bits and a Stop bit.

To send a Break character, set the SENDB and TXEN bits of the TXxSTA register. The Break character transmission is then initiated by a write to the TXxREG. The value of data written to TXxREG will be ignored and all '0's will be transmitted.

The SENDB bit is automatically reset by hardware after the corresponding Stop bit is sent. This allows the user to preload the transmit FIFO with the next transmit byte following the Break character (typically, the Sync character in the LIN specification).

The TRMT bit of the TXxSTA register indicates when the transmit operation is active or idle, just as it does during normal transmission. See [Figure 28-9](#) for the timing of the Break character sequence.

28.4.4.1 Break and Sync Transmit Sequence

The following sequence will start a message frame header made up of a Break, followed by an auto-baud Sync byte. This sequence is typical of a LIN bus master.

1. Configure the EUSART for the desired mode.
2. Set the TXEN and SENDB bits to enable the Break sequence.
3. Load the TXxREG with a dummy character to initiate transmission (the value is ignored).
4. Write '55h' to TXxREG to load the Sync character into the transmit FIFO buffer.
5. After the Break has been sent, the SENDB bit is reset by hardware and the Sync character is then transmitted.

When the TXxREG becomes empty, as indicated by the TXxIF, the next data byte can be written to TXxREG.

28.4.5 RECEIVING A BREAK CHARACTER

The Enhanced EUSART module can receive a Break character in two ways.

The first method to detect a Break character uses the FERR bit of the RCxSTA register and the received data as indicated by RCxREG. The Baud Rate Generator is assumed to have been initialized to the expected baud rate.

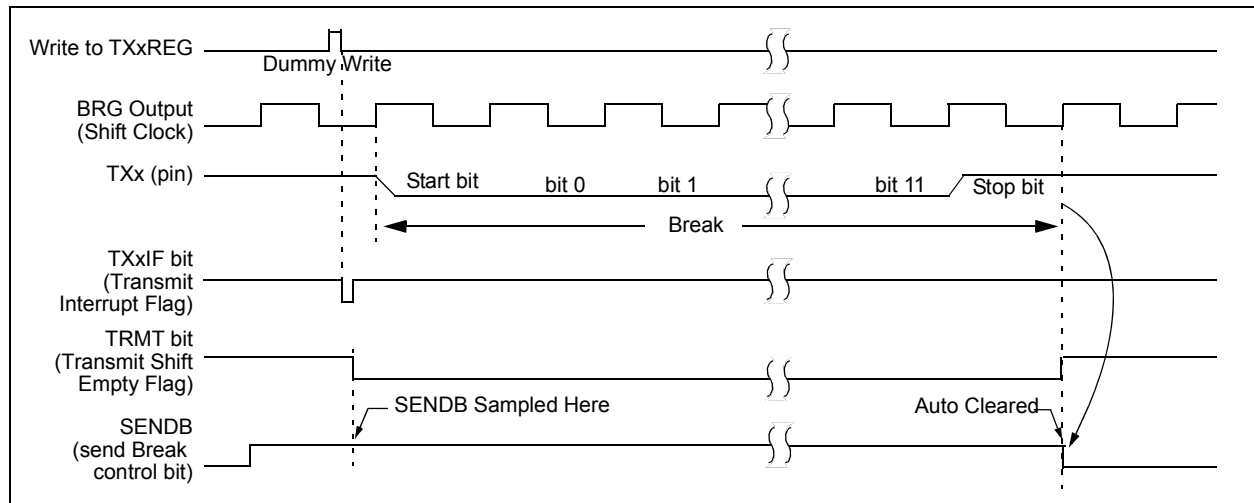
A Break character has been received when;

- RCxIF bit is set
- FERR bit is set
- RCxREG = 00h

The second method uses the Auto-Wake-up feature described in [Section 28.4.3 "Auto-Wake-up on Break"](#). By enabling this feature, the EUSART will sample the next two transitions on RX/DT, cause an RCxIF interrupt, and receive the next data byte followed by another interrupt.

Note that following a Break character, the user will typically want to enable the Auto-Baud Detect feature. For both methods, the user can set the ABDEN bit of the BAUDxCON register before placing the EUSART in Sleep mode.

FIGURE 28-9: SEND BREAK CHARACTER SEQUENCE



28.5 EUSART Synchronous Mode

Synchronous serial communications are typically used in systems with a single master and one or more slaves. The master device contains the necessary circuitry for baud rate generation and supplies the clock for all devices in the system. Slave devices can take advantage of the master clock by eliminating the internal clock generation circuitry.

There are two signal lines in Synchronous mode: a bidirectional data line and a clock line. Slaves use the external clock supplied by the master to shift the serial data into and out of their respective receive and transmit shift registers. Since the data line is bidirectional, synchronous operation is half-duplex only. Half-duplex refers to the fact that master and slave devices can receive and transmit data but not both simultaneously. The EUSART can operate as either a master or slave device.

Start and Stop bits are not used in synchronous transmissions.

28.5.1 SYNCHRONOUS MASTER MODE

The following bits are used to configure the EUSART for synchronous master operation:

- SYNC = 1
- CSRC = 1
- SREN = 0 (for transmit); SREN = 1 (for receive)
- CREN = 0 (for transmit); CREN = 1 (for receive)
- SPEN = 1

Setting the SYNC bit of the TXxSTA register configures the device for synchronous operation. Setting the CSRC bit of the TXxSTA register configures the device as a master. Clearing the SREN and CREN bits of the RCxSTA register ensures that the device is in the Transmit mode, otherwise the device will be configured to receive. Setting the SPEN bit of the RCxSTA register enables the EUSART.

28.5.1.1 Master Clock

Synchronous data transfers use a separate clock line, which is synchronous with the data. A device configured as a master transmits the clock on the TX/CK line. The TXx/CKx pin output driver is automatically enabled when the EUSART is configured for synchronous transmit or receive operation. Serial data bits change on the leading edge to ensure they are valid at the trailing edge of each clock. One clock cycle is generated for each data bit. Only as many clock cycles are generated as there are data bits.

28.5.1.2 Clock Polarity

A clock polarity option is provided for Microwire compatibility. Clock polarity is selected with the SCKP bit of the BAUDxCON register. Setting the SCKP bit sets the clock Idle state as high. When the SCKP bit is set, the data changes on the falling edge of each clock. Clearing the SCKP bit sets the Idle state as low. When the SCKP bit is cleared, the data changes on the rising edge of each clock.

28.5.1.3 Synchronous Master Transmission

Data is transferred out of the device on the RXx/DTx pin. The RXx/DTx and TXx/CKx pin output drivers are automatically enabled when the EUSART is configured for synchronous master transmit operation.

A transmission is initiated by writing a character to the TXxREG register. If the TSR still contains all or part of a previous character the new character data is held in the TXxREG until the last bit of the previous character has been transmitted. If this is the first character, or the previous character has been completely flushed from the TSR, the data in the TXxREG is immediately transferred to the TSR. The transmission of the character commences immediately following the transfer of the data to the TSR from the TXxREG.

Each data bit changes on the leading edge of the master clock and remains valid until the subsequent leading clock edge.

Note: The TSR register is not mapped in data memory, so it is not available to the user.

28.5.1.4 Synchronous Master Transmission Setup:

1. Initialize the SPxBRGH, SPxBRGL register pair and the BRGH and BRG16 bits to achieve the desired baud rate (see [Section 28.4 “EUSART Baud Rate Generator \(BRG\)”](#)).
2. Enable the synchronous master serial port by setting bits SYNC, SPEN and CSRC.
3. Disable Receive mode by clearing bits SREN and CREN.
4. Enable Transmit mode by setting the TXEN bit.
5. If 9-bit transmission is desired, set the TX9 bit.
6. If interrupts are desired, set the TXxIE bit of the PIE3/4 registers and the GIE and PEIE bits of the INTCON register.
7. If 9-bit transmission is selected, the ninth bit should be loaded in the TX9D bit.
8. Start transmission by loading data to the TXxREG register.

FIGURE 28-10: SYNCHRONOUS TRANSMISSION

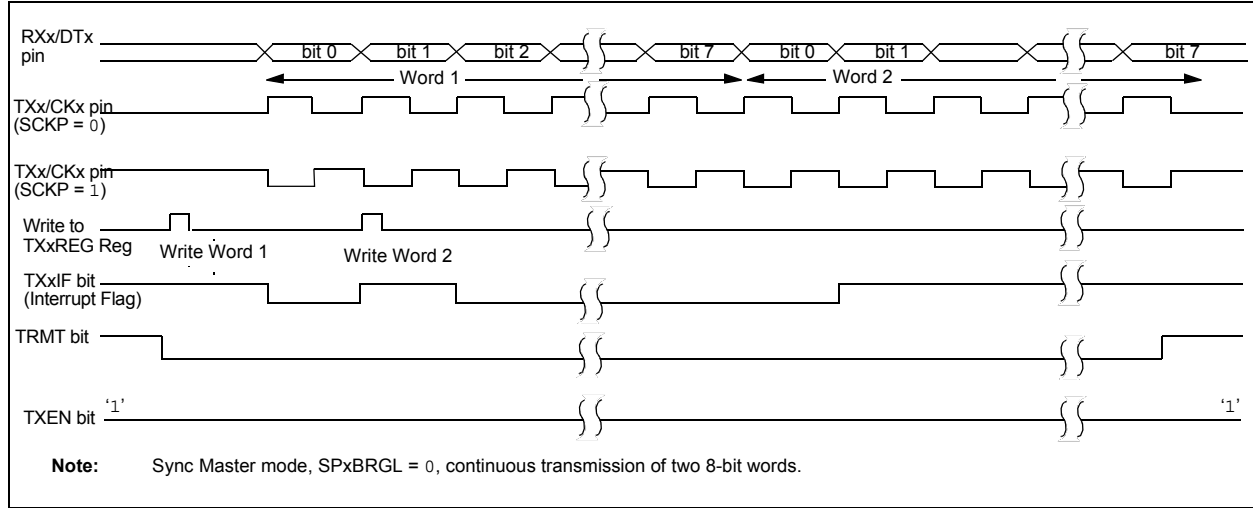


FIGURE 28-11: SYNCHRONOUS TRANSMISSION (THROUGH TXEN)

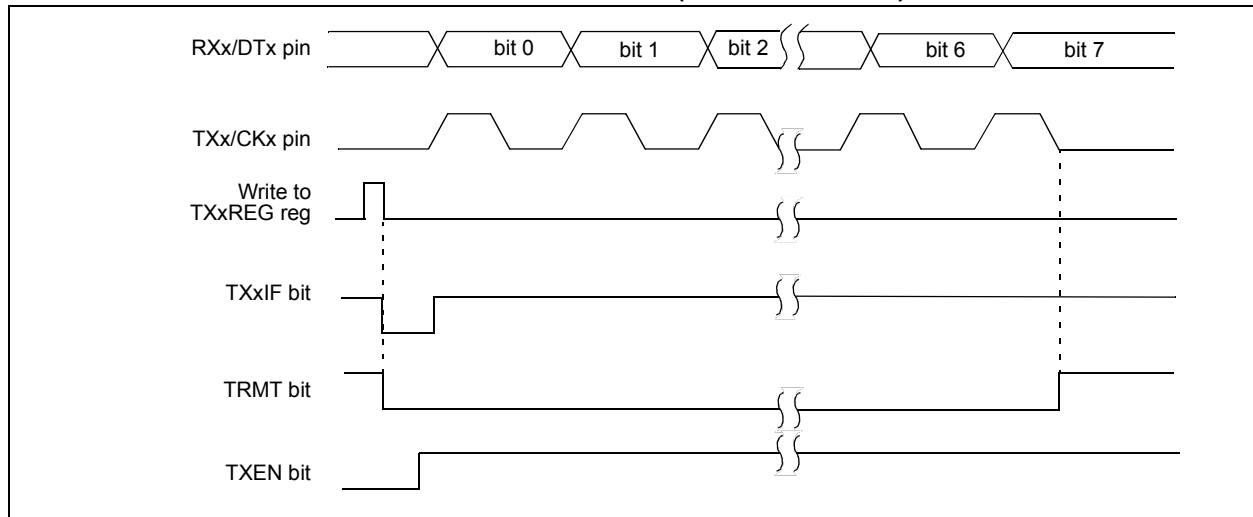


TABLE 28-7: SUMMARY OF REGISTERS ASSOCIATED WITH SYNCHRONOUS MASTER TRANSMISSION

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on Page |
|----------|--|-----------|-------------|-------|---------|---------|---------|---------|------------------|
| BAUDxCON | ABDOVF | RCIDL | — | SCKP | BRG16 | — | WUE | ABDEN | 446 |
| INTCON | GIE/GIEH | PEIE/GIEL | IPEN | — | INT3EDG | INT2EDG | INT1EDG | INT0EDG | 172 |
| PIE3 | RC2IE | TX2IE | RC1IE | TX1IE | BCL2IE | SSP2IE | BCL1IE | SSP1IE | 187 |
| PIR3 | RC2IF | TX2IF | RC1IF | TX1IF | BCL2IF | SSP2IF | BCL1IF | SSP1IF | 176 |
| IPR3 | RC2IP | TX2IP | RC1IP | TX1IP | BCL2IP | SSP2IP | BCL1IP | SSP1IP | 197 |
| PIE4 | — | — | RC5IE | TX5IE | RC4IE | TX4IE | RC3IE | TX3IE | 188 |
| PIR4 | — | — | RC5IF | TX5IF | RC4IF | TX4IF | RC3IF | TX3IF | 177 |
| IPR4 | — | — | RC5IP | TX5IP | RC4IP | TX4IP | RC3IP | TX3IP | 198 |
| RCxSTA | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 445 |
| RxyPPS | — | — | RxyPPS<5:0> | | | | | | 227 |
| TxxPPS | — | — | TXPPS<5:0> | | | | | | 224 |
| SPxBRGH | EUSARTx Baud Rate Generator, High Byte | | | | | | | | 455* |
| SPxBRGL | EUSARTx Baud Rate Generator, Low Byte | | | | | | | | 455* |
| TxxREG | EUSARTx Transmit Data Register | | | | | | | | 447* |
| TxxSTA | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D | 444 |

Legend: — = unimplemented location, read as '0'. Shaded cells are not used for synchronous master transmission.

* Page provides register information.

28.5.1.5 Synchronous Master Reception

Data is received at the RXx/DTx pin. The RXx/DTx pin output driver is automatically disabled when the EUSART is configured for synchronous master receive operation.

In Synchronous mode, reception is enabled by setting either the Single Receive Enable bit (SREN of the RCxSTA register) or the Continuous Receive Enable bit (CREN of the RCxSTA register).

When SREN is set and CREN is clear, only as many clock cycles are generated as there are data bits in a single character. The SREN bit is automatically cleared at the completion of one character. When CREN is set, clocks are continuously generated until CREN is cleared. If CREN is cleared in the middle of a character the CK clock stops immediately and the partial character is discarded. If SREN and CREN are both set, then SREN is cleared at the completion of the first character and CREN takes precedence.

To initiate reception, set either SREN or CREN. Data is sampled at the RXx/DTx pin on the trailing edge of the TX/CK clock pin and is shifted into the Receive Shift Register (RSR). When a complete character is received into the RSR, the RCxIF bit is set and the character is automatically transferred to the two character receive FIFO. The Least Significant eight bits of the top character in the receive FIFO are available in RCxREG. The RCxIF bit remains set as long as there are unread characters in the receive FIFO.

Note: If the RX/DT function is on an analog pin, the corresponding ANSEL bit must be cleared for the receiver to function.

28.5.1.6 Slave Clock

Synchronous data transfers use a separate clock line, which is synchronous with the data. A device configured as a slave receives the clock on the TX/CK line. The TXx/CKx pin output driver is automatically disabled when the device is configured for synchronous slave transmit or receive operation. Serial data bits change on the leading edge to ensure they are valid at the trailing edge of each clock. One data bit is transferred for each clock cycle. Only as many clock cycles should be received as there are data bits.

Note: If the device is configured as a slave and the TX/CK function is on an analog pin, the corresponding ANSEL bit must be cleared.

28.5.1.7 Receive Overrun Error

The receive FIFO buffer can hold two characters. An overrun error will be generated if a third character, in its entirety, is received before RCxREG is read to access the FIFO. When this happens the OERR bit of the RCxSTA register is set. Previous data in the FIFO will not be overwritten. The two characters in the FIFO buffer can be read, however, no additional characters will be received until the error is cleared. The OERR bit can only be cleared by clearing the overrun condition. If the overrun error occurred when the SREN bit is set and CREN is clear then the error is cleared by reading RCxREG. If the overrun occurred when the CREN bit is set then the error condition is cleared by either clearing the CREN bit of the RCxSTA register or by clearing the SPEN bit which resets the EUSART.

28.5.1.8 Receiving 9-Bit Characters

The EUSART supports 9-bit character reception. When the RX9 bit of the RCxSTA register is set the EUSART will shift nine bits into the RSR for each character received. The RX9D bit of the RCxSTA register is the ninth, and Most Significant, data bit of the top unread character in the receive FIFO. When reading 9-bit data from the receive FIFO buffer, the RX9D data bit must be read before reading the eight Least Significant bits from the RCxREG.

28.5.1.9 Synchronous Master Reception Setup:

1. Initialize the SPxBRGH:SPxBRGL register pair for the appropriate baud rate. Set or clear the BRGH and BRG16 bits, as required, to achieve the desired baud rate.
2. Clear the ANSEL bit for the RXx pin (if applicable).
3. Enable the synchronous master serial port by setting bits SYNC, SPEN and CSRC.
4. Ensure bits CREN and SREN are clear.
5. If interrupts are desired, set the RCxIE bit of the PIE3/4 registers and the GIE and PEIE bits of the INTCON register.
6. If 9-bit reception is desired, set bit RX9.
7. Start reception by setting the SREN bit or for continuous reception, set the CREN bit.
8. Interrupt flag bit RCxIF will be set when reception of a character is complete. An interrupt will be generated if the enable bit RCxIE was set.
9. Read the RCxSTA register to get the ninth bit (if enabled) and determine if any error occurred during reception.
10. Read the 8-bit received data by reading the RCxREG register.
11. If an overrun error occurs, clear the error by either clearing the CREN bit of the RCxSTA register or by clearing the SPEN bit which resets the EUSART.

FIGURE 28-12: SYNCHRONOUS RECEPTION (MASTER MODE, SREN)

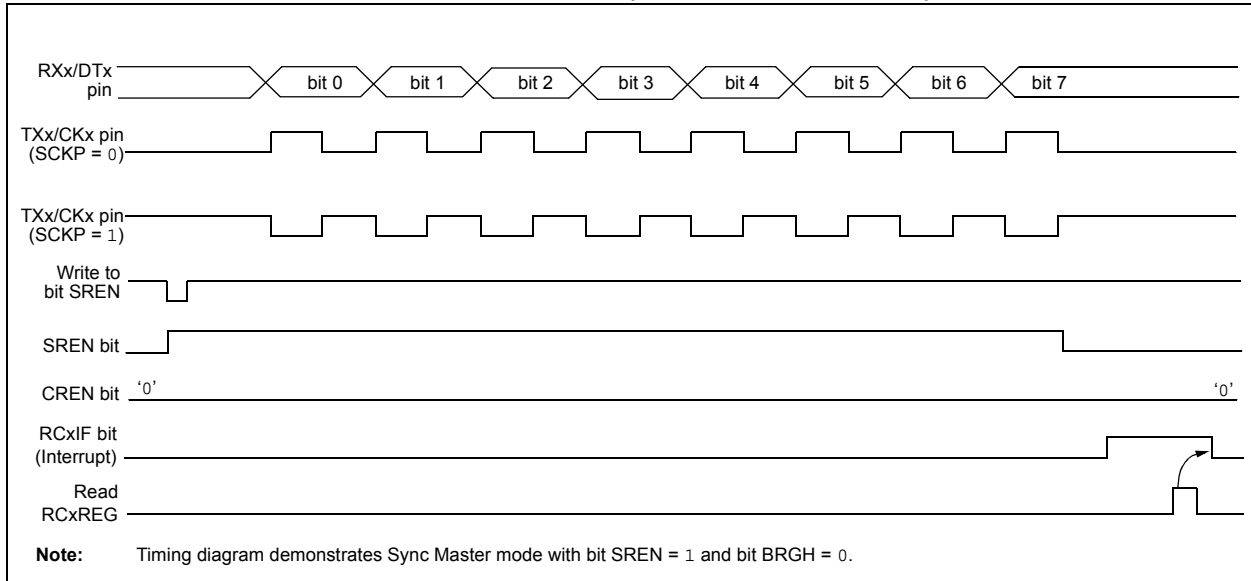


TABLE 28-8: SUMMARY OF REGISTERS ASSOCIATED WITH SYNCHRONOUS MASTER RECEPTION

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on Page |
|----------|--|-----------|-------------|---------|---------|---------|---------|---------|------------------|
| ANSELB | ANSELB7 | ANSELB6 | ANSELB5 | ANSELB4 | ANSELB3 | ANSELB2 | ANSELB1 | ANSELB0 | 212 |
| BAUDxCON | ABDOVF | RCIDL | — | SCKP | BRG16 | — | WUE | ABDEN | 446 |
| INTCON | GIE/GIEH | PEIE/GIEL | IPEN | — | INT3EDG | INT2EDG | INT1EDG | INT0EDG | 172 |
| PIE3 | RC2IE | TX2IE | RC1IE | TX1IE | BCL2IE | SSP2IE | BCL1IE | SSP1IE | 187 |
| PIR3 | RC2IF | TX2IF | RC1IF | TX1IF | BCL2IF | SSP2IF | BCL1IF | SSP1IF | 176 |
| IPR3 | RC2IP | TX2IP | RC1IP | TX1IP | BCL2IP | SSP2IP | BCL1IP | SSP1IP | 197 |
| PIE4 | — | — | RC5IE | TX5IE | RC4IE | TX4IE | RC3IE | TX3IE | 188 |
| PIR4 | — | — | RC5IF | TX5IF | RC4IF | TX4IF | RC3IF | TX3IF | 177 |
| IPR4 | — | — | RC5IP | TX5IP | RC4IP | TX4IP | RC3IP | TX3IP | 198 |
| RCxREG | EUSARTx Receive Data Register | | | | | | | | 450* |
| RCxSTA | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 445 |
| RxyPPS | — | — | RxyPPS<5:0> | | | | | | 227 |
| RXxPPS | — | — | RXPPS<5:0> | | | | | | 224 |
| SPxBRGH | EUSARTx Baud Rate Generator, High Byte | | | | | | | | 455* |
| SPxBRGL | EUSARTx Baud Rate Generator, Low Byte | | | | | | | | 455* |
| TXxSTA | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D | 444 |

Legend: — = unimplemented location, read as '0'. Shaded cells are not used for synchronous master reception.

* Page provides register information.

28.5.2 SYNCHRONOUS SLAVE MODE

The following bits are used to configure the EUSART for synchronous slave operation:

- SYNC = 1
- CSRC = 0
- SREN = 0 (for transmit); SREN = 1 (for receive)
- CREN = 0 (for transmit); CREN = 1 (for receive)
- SPEN = 1

Setting the SYNC bit of the TXxSTA register configures the device for synchronous operation. Clearing the CSRC bit of the TXxSTA register configures the device as a slave. Clearing the SREN and CREN bits of the RCxSTA register ensures that the device is in the Transmit mode, otherwise the device will be configured to receive. Setting the SPEN bit of the RCxSTA register enables the EUSART.

28.5.2.1 EUSART Synchronous Slave Transmit

The operation of the Synchronous Master and Slave modes are identical (see [Section 28.5.1.3 “Synchronous Master Transmission”](#)), except in the case of the Sleep mode.

If two words are written to the TXxREG and then the SLEEP instruction is executed, the following will occur:

1. The first character will immediately transfer to the TSR register and transmit.
2. The second word will remain in the TXxREG register.
3. The TXxIF bit will not be set.
4. After the first character has been shifted out of TSR, the TXxREG register will transfer the second character to the TSR and the TXxIF bit will now be set.
5. If the PEIE and TXxIE bits are set, the interrupt will wake the device from Sleep and execute the next instruction. If the GIE bit is also set, the program will call the Interrupt Service Routine.

28.5.2.2 Synchronous Slave Transmission Setup

1. Set the SYNC and SPEN bits and clear the CSRC bit.
2. Clear the ANSEL bit for the CKx pin (if applicable).
3. Clear the CREN and SREN bits.
4. If interrupts are desired, set the TXxIE bit of the PIE3/4 registers and the GIE and PEIE bits of the INTCON register.
5. If 9-bit transmission is desired, set the TX9 bit.
6. Enable transmission by setting the TXEN bit.
7. If 9-bit transmission is selected, insert the Most Significant bit into the TX9D bit.
8. Start transmission by writing the Least Significant eight bits to the TXxREG register.

TABLE 28-9: SUMMARY OF REGISTERS ASSOCIATED WITH SYNCHRONOUS SLAVE TRANSMISSION

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on Page |
|----------|--------------------------------|-----------|-------------|-------|---------|---------|---------|---------|------------------|
| BAUDxCON | ABDOVF | RCIDL | — | SCKP | BRG16 | — | WUE | ABDEN | 446 |
| INTCON | GIE/GIEH | PEIE/GIEL | IPEN | — | INT3EDG | INT2EDG | INT1EDG | INT0EDG | 172 |
| PIE3 | RC2IE | TX2IE | RC1IE | TX1IE | BCL2IE | SSP2IE | BCL1IE | SSP1IE | 187 |
| PIR3 | RC2IF | TX2IF | RC1IF | TX1IF | BCL2IF | SSP2IF | BCL1IF | SSP1IF | 176 |
| IPR3 | RC2IP | TX2IP | RC1IP | TX1IP | BCL2IP | SSP2IP | BCL1IP | SSP1IP | 197 |
| PIE4 | — | — | RC5IE | TX5IE | RC4IE | TX4IE | RC3IE | TX3IE | 188 |
| PIR4 | — | — | RC5IF | TX5IF | RC4IF | TX4IF | RC3IF | TX3IF | 177 |
| IPR4 | — | — | RC5IP | TX5IP | RC4IP | TX4IP | RC3IP | TX3IP | 198 |
| RCxSTA | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 445 |
| RxyPPS | — | — | RxyPPS<5:0> | | | | | | 227 |
| TXxPPS | — | — | TXPPS<5:0> | | | | | | 224 |
| TXxREG | EUSARTx Transmit Data Register | | | | | | | | 447* |
| TXxSTA | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D | 444 |

Legend: — = unimplemented location, read as '0'. Shaded cells are not used for synchronous slave transmission.

* Page provides register information.

28.5.2.3 EUSART Synchronous Slave Reception

The operation of the Synchronous Master and Slave modes is identical ([Section 28.5.1.5 “Synchronous Master Reception”](#)), with the following exceptions:

- Sleep
- CREN bit is always set, therefore the receiver is never idle
- SREN bit, which is a “don’t care” in Slave mode

A character may be received while in Sleep mode by setting the CREN bit prior to entering Sleep. Once the word is received, the RSR register will transfer the data to the RCxREG register. If the RCxIE enable bit is set, the interrupt generated will wake the device from Sleep and execute the next instruction. If the GIE bit is also set, the program will branch to the interrupt vector.

28.5.2.4 Synchronous Slave Reception Setup:

1. Set the SYNC and SPEN bits and clear the CSRC bit.
2. Clear the ANSEL bit for both the CKx and DTx pins (if applicable).
3. If interrupts are desired, set the RCxIE bit of the PIE3/4 registers and the GIE and PEIE bits of the INTCON register.
4. If 9-bit reception is desired, set the RX9 bit.
5. Set the CREN bit to enable reception.
6. The RCxIF bit will be set when reception is complete. An interrupt will be generated if the RCxIE bit was set.
7. If 9-bit mode is enabled, retrieve the Most Significant bit from the RX9D bit of the RCxSTA register.
8. Retrieve the eight Least Significant bits from the receive FIFO by reading the RCxREG register.
9. If an overrun error occurs, clear the error by either clearing the CREN bit of the RCxSTA register or by clearing the SPEN bit which resets the EUSART.

TABLE 28-10: SUMMARY OF REGISTERS ASSOCIATED WITH SYNCHRONOUS SLAVE RECEPTION

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on Page |
|----------|------------------------------|-----------|-------------|-------|---------|---------|---------|---------|------------------|
| BAUDxCON | ABDOVF | RCIDL | — | SCKP | BRG16 | — | WUE | ABDEN | 446 |
| INTCON | GIE/GIEH | PEIE/GIEL | IPEN | — | INT3EDG | INT2EDG | INT1EDG | INT0EDG | 172 |
| PIE3 | RC2IE | TX2IE | RC1IE | TX1IE | BCL2IE | SSP2IE | BCL1IE | SSP1IE | 187 |
| PIR3 | RC2IF | TX2IF | RC1IF | TX1IF | BCL2IF | SSP2IF | BCL1IF | SSP1IF | 176 |
| IPR3 | RC2IP | TX2IP | RC1IP | TX1IP | BCL2IP | SSP2IP | BCL1IP | SSP1IP | 197 |
| PIE4 | — | — | RC5IE | TX5IE | RC4IE | TX4IE | RC3IE | TX3IE | 188 |
| PIR4 | — | — | RC5IF | TX5IF | RC4IF | TX4IF | RC3IF | TX3IF | 176 |
| IPR4 | — | — | RC5IP | TX5IP | RC4IP | TX4IP | RC3IP | TX3IP | 198 |
| RCxREG | EUSART Receive Data Register | | | | | | | | 450* |
| RCxSTA | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 445 |
| RxyPPS | — | — | RxyPPS<5:0> | | | | | | 227 |
| RXxPPS | — | — | RXPPS<5:0> | | | | | | 224 |
| TXxSTA | CSRC | TX9 | TXEN | SYNC | SENDB | BRGH | TRMT | TX9D | 444 |

Legend: — = unimplemented location, read as ‘0’. Shaded cells are not used for synchronous slave reception.

* Page provides register information.

28.6 EUSART Operation During Sleep

The EUSART will remain active during Sleep only in the Synchronous Slave mode. All other modes require the system clock and therefore cannot generate the necessary signals to run the Transmit or Receive Shift registers during Sleep.

Synchronous Slave mode uses an externally generated clock to run the Transmit and Receive Shift registers.

28.6.1 SYNCHRONOUS RECEIVE DURING SLEEP

To receive during Sleep, all the following conditions must be met before entering Sleep mode:

- RCxSTA and TXxSTA Control registers must be configured for Synchronous Slave Reception (see [Section 28.5.2.4 “Synchronous Slave Reception Setup:”](#)).
- If interrupts are desired, set the RCxIE bit of the PIE3/4 registers and the GIE and PEIE bits of the INTCON register.
- The RCxIF interrupt flag must be cleared by reading RCxREG to unload any pending characters in the receive buffer.

Upon entering Sleep mode, the device will be ready to accept data and clocks on the RXx/DTx and TXx/CKx pins, respectively. When the data word has been completely clocked in by the external device, the RCxIF interrupt flag bit of the PIR3/4 registers will be set. Thereby, waking the processor from Sleep.

Upon waking from Sleep, the instruction following the SLEEP instruction will be executed. If the Global Interrupt Enable (GIE) bit of the INTCON register is also set, then the Interrupt Service Routine at address 004h will be called.

28.6.2 SYNCHRONOUS TRANSMIT DURING SLEEP

To transmit during Sleep, all the following conditions must be met before entering Sleep mode:

- The RCxSTA and TXxSTA Control registers must be configured for synchronous slave transmission (see [Section 28.5.2.2 “Synchronous Slave Transmission Setup”](#)).
- The TXxIF interrupt flag must be cleared by writing the output data to the TXxREG, thereby filling the TSR and transmit buffer.
- If interrupts are desired, set the TXxIE bit of the PIE3/4 registers and the PEIE bit of the INTCON register.
- Interrupt enable bits TXxIE of the PIE3 register and PEIE of the INTCON register must set.

Upon entering Sleep mode, the device will be ready to accept clocks on TXx/CKx pin and transmit data on the RXx/DTx pin. When the data word in the TSR has been completely clocked out by the external device, the pending byte in the TXxREG will transfer to the TSR and the TXxIF flag will be set. Thereby, waking the processor from Sleep. At this point, the TXxREG is available to accept another character for transmission, which will clear the TXxIF flag.

Upon waking from Sleep, the instruction following the SLEEP instruction will be executed. If the Global Interrupt Enable (GIE) bit is also set then the Interrupt Service Routine at address 0004h will be called.

29.0 FIXED VOLTAGE REFERENCE (FVR)

The Fixed Voltage Reference, or FVR, is a stable voltage reference, independent of VDD, 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 the following:

- ADC input channel
- ADC positive reference
- Comparator input
- Digital-to-Analog Converter (DAC)

The FVR can be enabled by setting the FVREN bit of the FVRCON register.

Note: Fixed Voltage Reference output cannot exceed VDD.

29.1 Independent Gain Amplifiers

The output of the FVR, which is connected to the ADC, Comparators, and DAC, 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.

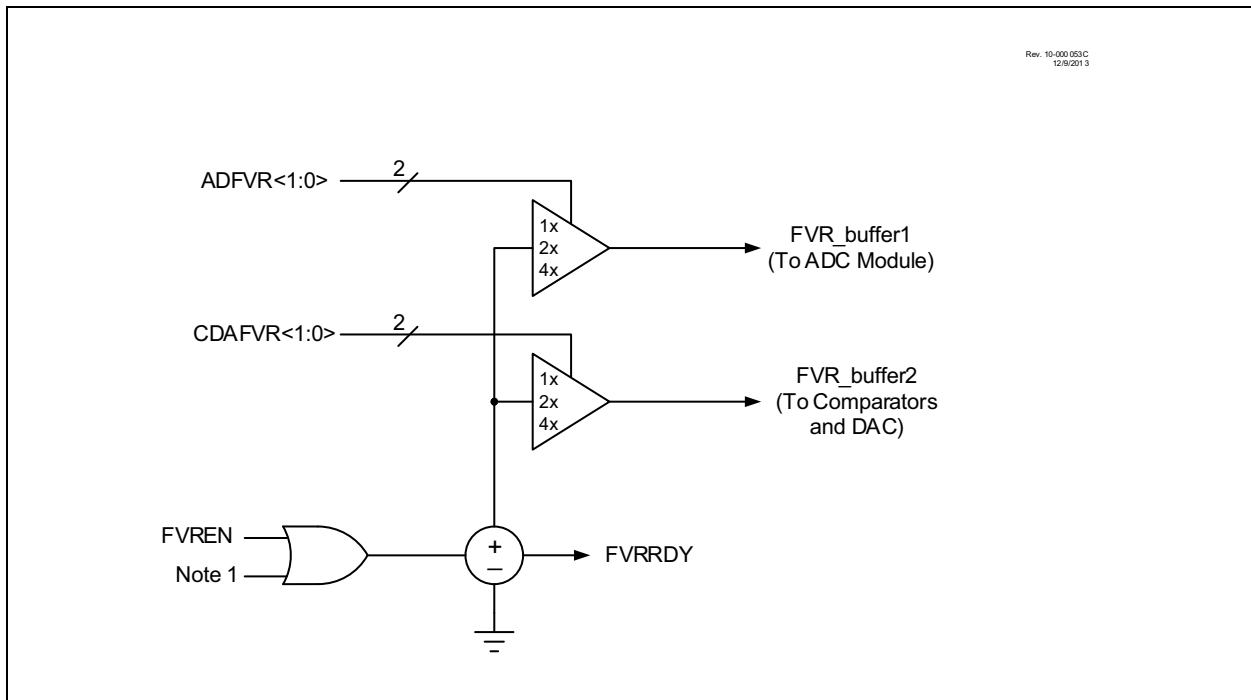
The ADFVR<1:0> bits of the FVRCON register are used to enable and configure the gain amplifier settings for the reference supplied to the ADC module. Reference **Section 32.0 “Analog-to-Digital Converter with Computation (ADC2) Module”** for additional information.

The CDAFVR<1:0> bits of the FVRCON register are used to enable and configure the gain amplifier settings for the reference supplied to the DAC and comparator module. Reference **Section 31.0 “5-Bit Digital-to-Analog Converter (DAC) Module”** and **Section 33.0 “Comparator Module (C1/2/3)”** for additional information.

29.2 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 FVRRDY bit of the FVRCON register will be set.

FIGURE 29-1: VOLTAGE REFERENCE BLOCK DIAGRAM



29.3 Register Definitions: FVR Control

REGISTER 29-1: FVRCON: FIXED VOLTAGE REFERENCE CONTROL REGISTER

| | | | | | | | |
|---------|-----------------------|---------------------|----------------------|-------------|---------|------------|---------|
| R/W-0/0 | R-q/q | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| FVREN | FVRRDY ⁽¹⁾ | TSEN ⁽³⁾ | TSRNG ⁽³⁾ | CDAFVR<1:0> | | ADFVR<1:0> | |
| bit 7 | | | | | | bit 0 | |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | q = Value depends on condition |

| | |
|---------|---|
| bit 7 | FVREN: Fixed Voltage Reference Enable bit 1 = Fixed Voltage Reference is enabled 0 = Fixed Voltage Reference is disabled |
| bit 6 | FVRRDY: Fixed Voltage Reference Ready Flag bit ⁽¹⁾ 1 = Fixed Voltage Reference output is ready for use 0 = Fixed Voltage Reference output is not ready or not enabled |
| bit 5 | TSEN: Temperature Indicator Enable bit ⁽³⁾ 1 = Temperature Indicator is enabled 0 = Temperature Indicator is disabled |
| bit 4 | TSRNG: Temperature Indicator Range Selection bit ⁽³⁾ 1 = VOUT = VDD - 4VT (High Range) 0 = VOUT = VDD - 2VT (Low Range) |
| bit 3-2 | CDAFVR<1:0>: Comparator FVR Buffer Gain Selection bits 11 = Comparator FVR Buffer Gain is 4x, (4.096V) ⁽²⁾ 10 = Comparator FVR Buffer Gain is 2x, (2.048V) ⁽²⁾ 01 = Comparator FVR Buffer Gain is 1x, (1.024V) 00 = Comparator FVR Buffer is off |
| bit 1-0 | ADFVR<1:0>: ADC FVR Buffer Gain Selection bit 11 = ADC FVR Buffer Gain is 4x, (4.096V) ⁽²⁾ 10 = ADC FVR Buffer Gain is 2x, (2.048V) ⁽²⁾ 01 = ADC FVR Buffer Gain is 1x, (1.024V) 00 = ADC FVR Buffer is off |

- Note 1:** FVRRDY is always '1'.
Note 2: Fixed Voltage Reference output cannot exceed VDD.
Note 3: See **Section 30.0 "Temperature Indicator Module"** for additional information.

TABLE 29-1: SUMMARY OF REGISTERS ASSOCIATED WITH FIXED VOLTAGE REFERENCE

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on page |
|----------|-------|--------|-------|------------|-------------|------------|------------|-------|------------------|
| FVRCON | FVREN | FVRRDY | TSEN | TSRNG | CDAFVR<1:0> | | ADFVR<1:0> | | 474 |
| ADCON0 | ADON | ADCONT | — | ADCS | — | ADFM | — | ADGO | 499 |
| CMxNCH | — | — | — | — | — | CxNCH<2:0> | | 520 | |
| CMxPCH | — | — | — | — | — | CxPCH<2:0> | | 521 | |
| DAC1CON1 | — | — | — | DAC1R<4:0> | | | | | 480 |

Legend: — = Unimplemented location, read as '0'. Shaded cells are not used with the Fixed Voltage Reference.

30.0 TEMPERATURE INDICATOR MODULE

This family of devices is equipped with a temperature circuit designed to measure the operating temperature of the silicon die. The circuit's range of operating temperature falls between -40°C and $+85^{\circ}\text{C}$. The output is a voltage that is proportional to the device temperature. The output of the temperature indicator is internally connected to the device ADC.

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. Reference Application Note AN1333, "Use and Calibration of the Internal Temperature Indicator" (DS00001333) for more details regarding the calibration process.

30.1 Circuit Operation

Figure 30-1 shows a simplified block diagram of the temperature circuit. The proportional voltage output is achieved by measuring the forward voltage drop across multiple silicon junctions.

Equation 30-1 describes the output characteristics of the temperature indicator.

EQUATION 30-1: V_{OUT} RANGES

$$\text{High Range: } V_{OUT} = V_{DD} - 4V_T$$

$$\text{Low Range: } V_{OUT} = V_{DD} - 2V_T$$

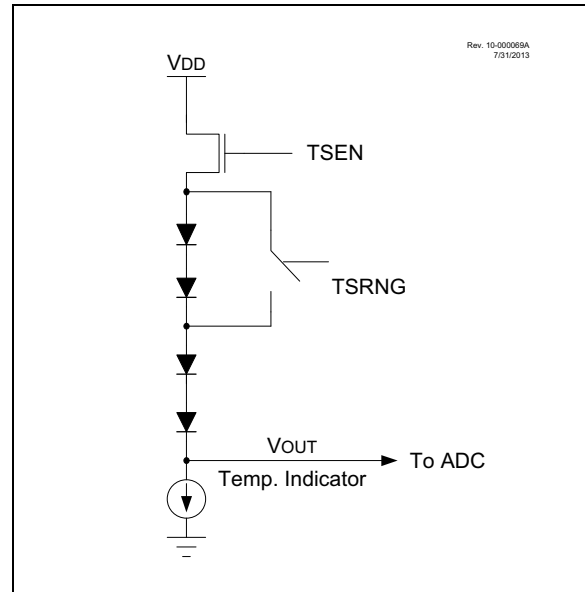
The temperature sense circuit is integrated with the Fixed Voltage Reference (FVR) module. See Section 29.0 "Fixed Voltage Reference (FVR)" for more information.

The circuit is enabled by setting the TSEN bit of the FVRCON register. When disabled, the circuit draws no current.

The circuit operates in either high or low range. The high range, selected by setting the TSRNG bit of the FVRCON register, provides a wider output voltage. This provides more resolution over the temperature range, but may be less consistent from part to part. This 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 of the FVRCON register. The low range generates a lower voltage drop and thus, a lower bias voltage is needed to operate the circuit. The low range is provided for low voltage operation.

FIGURE 30-1: TEMPERATURE CIRCUIT DIAGRAM



30.2 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 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.

Table 30-1 shows the recommended minimum V_{DD} vs. range setting.

TABLE 30-1: RECOMMENDED V_{DD} VS. RANGE

| Min. V_{DD} , TSRNG = 1 | Min. V_{DD} , TSRNG = 0 |
|---------------------------|---------------------------|
| 3.6V | 1.8V |

30.3 Temperature Output

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 Section 32.0 "Analog-to-Digital Converter with Computation (ADC2) Module" for detailed information.

30.4 ADC Acquisition Time

To ensure accurate temperature measurements, the user must wait at least 200 μs after the ADC input multiplexer is connected to the temperature indicator output before the conversion is performed. In addition, the user must wait 200 μs between consecutive conversions of the temperature indicator output.

TABLE 30-2: SUMMARY OF REGISTERS ASSOCIATED WITH THE TEMPERATURE INDICATOR

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on page |
|--------|-------|--------|-------|-------|------------|-------|------------|-------|---------------------|
| FVRCON | FVREN | FVRRDY | TSEN | TSRNG | CDFVR<1:0> | | ADFVR<1:0> | | 474 |

Legend: — = Unimplemented location, read as '0'. Shaded cells are unused by the temperature indicator module.

31.1 Output Voltage Selection

The DAC has 32 voltage level ranges. The 32 levels are set with the DAC1R<4:0> bits of the DAC1CON1 register.

The DAC output voltage can be determined by using Equation 31-1.

31.2 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 Table 37-16.

31.3 DAC Voltage Reference Output

The unbuffered DAC voltage can be output to the DACxOUTn pin(s) by setting the respective DACOEn bit(s) of the DACxCON0 register. Selecting the DAC reference voltage for output on either DACxOUTn pin automatically overrides the digital output buffer, the weak pull-up and digital input threshold detector functions of that pin.

Reading the DACxOUTn pin when it has been configured for DAC reference voltage output will always return a '0'.

Note: The unbuffered DAC output (DACxOUTn) is not intended to drive an external load.

31.4 Operation During Sleep

When the device wakes up from Sleep through an interrupt or a Windowed Watchdog Timer Time-out, the contents of the DACxCON0 register are not affected. To minimize current consumption in Sleep mode, the voltage reference should be disabled.

31.5 Effects of a Reset

A device Reset affects the following:

- DACx is disabled.
- DACx output voltage is removed from the DACxOUTn pin(s).
- The DAC1R<4:0> range select bits are cleared.

EQUATION 31-1: DAC OUTPUT VOLTAGE

IF DACEN = 1

$$DACx_output = \left((V_{REF+} - V_{REF-}) \times \frac{DACR[4:0]}{2^5} \right) + V_{REF-}$$

Note: See the DAC1CON0 register for the available VSOURCE+ and VSOURCE- selections.

31.6 Register Definitions: DAC Control

Long bit name prefixes for the DAC peripheral is shown in Table 31-1. Refer to Section 1.4.2.2 “Long Bit Names” for more information.

TABLE 31-1:

| Peripheral | Bit Name Prefix |
|------------|-----------------|
| DAC | DAC |

REGISTER 31-1: DAC1CON0: DAC CONTROL REGISTER

| R/W-0/0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | U-0 | R/W-0/0 |
|---------|-----|---------|---------|----------|---------|-----|---------|
| EN | — | OE1 | OE2 | PSS<1:0> | | — | NSS |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as ‘0’ |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| ‘1’ = Bit is set | ‘0’ = Bit is cleared | |

| | |
|---------|---|
| bit 7 | EN: DAC Enable bit 1 = DAC is enabled 0 = DAC is disabled |
| bit 6 | Unimplemented: Read as ‘0’ |
| bit 5 | OE1: DAC Voltage Output Enable bit 1 = DAC voltage level is output on the DAC1OUT1 pin 0 = DAC voltage level is disconnected from the DAC1OUT1 pin |
| bit 4 | OE2: DAC Voltage Output Enable bit 1 = DAC voltage level is output on the DAC1OUT2 pin 0 = DAC voltage level is disconnected from the DAC1OUT2 pin |
| bit 3-2 | PSS<1:0>: DAC Positive Source Select bit 11 = Reserved 10 = FVR buffer 01 = VREF+ 00 = AVDD |
| bit 1 | Unimplemented: Read as ‘0’ |
| bit 0 | NSS: DAC Negative Source Select bit 1 = VREF- 0 = AVSS |

REGISTER 31-2: DAC1CON1: DAC DATA REGISTER

| | | | | | | | |
|-------|-----|-----|------------|---------|---------|---------|---------|
| U-0 | U-0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| — | — | — | DAC1R<4:0> | | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-5 **Unimplemented:** Read as '0'
bit 4-0 **DAC1R<4:0>:** Data Input Register for DAC bits

TABLE 31-2: SUMMARY OF REGISTERS ASSOCIATED WITH THE DAC MODULE

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on page |
|----------|-------|--------|-------|------------|-------------|-------|------------|-------|---------------------|
| DAC1CON0 | EN | — | OE1 | OE2 | PSS<1:0> | | — | NSS | 479 |
| DAC1CON1 | — | — | — | DAC1R<4:0> | | | | | 480 |
| FVRCON | FVREN | FVRRDY | TSEN | TSRNG | CDAFVR<1:0> | | ADFVR<1:0> | | 474 |

Legend: — = Unimplemented location, read as '0'. Shaded cells are not used with the DAC module.

32.0 ANALOG-TO-DIGITAL CONVERTER WITH COMPUTATION (ADC²) MODULE

The Analog-to-Digital Converter with Computation (ADC²) allows conversion of an analog input signal to a 10-bit binary representation of that signal. This device uses analog inputs, which 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 10-bit binary result via successive approximation and stores the conversion result into the ADC result registers (ADRESH:ADRESL register pair).

Additionally, the following features are provided within the ADC module:

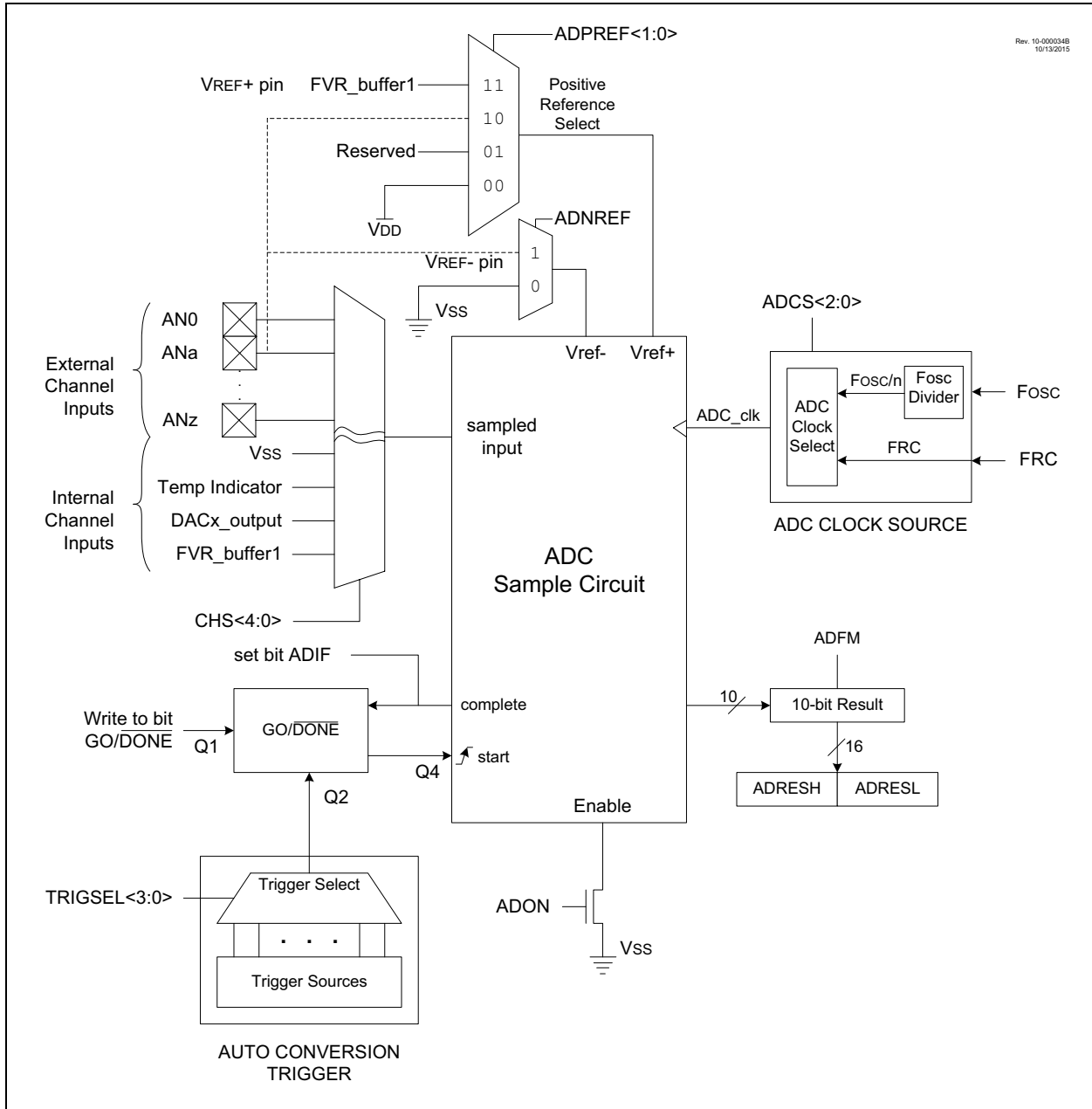
- 8-bit Acquisition Timer
- Hardware Capacitive Voltage Divider (CVD) support:
 - 8-bit precharge timer
 - Adjustable sample and hold capacitor array
 - Guard ring digital output drive
- Automatic repeat and sequencing:
 - Automated double sample conversion for CVD
 - Two sets of result registers (Result and Previous result)
 - Auto-conversion trigger
 - Internal retrigger
- Computation features:
 - Averaging and low-pass filter functions
 - Reference comparison
 - 2-level threshold comparison
 - Selectable interrupts

Figure 32-1 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.

FIGURE 32-1: ADC² BLOCK DIAGRAM



32.1 ADC Configuration

When configuring and using 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

32.1.1 PORT CONFIGURATION

The ADC can be used to convert both analog and digital signals. When converting analog signals, the I/O pin should be configured for analog by setting the associated TRIS and ANSEL bits. Refer to [Section 15.0 “I/O Ports”](#) for more information.

Note: Analog voltages on any pin that is defined as a digital input may cause the input buffer to conduct excess current.

32.1.2 CHANNEL SELECTION

There are several channel selections available:

- Eight PORTA pins (RA<7:0>)
- Eight PORTB pins (RB<7:0>)
- Eight PORTD pins (RD<7:0>)
- Eight PORTE pins (RE<7:0>)
- Eight PORTF pins ((RF<7:0>)
- Seven PORTG pins (RG<7:6>, RG<4:0>)
- Temperature Indicator
- DAC output
- Fixed Voltage Reference (FVR)
- AVss (ground)

The ADPCH register determines which channel is connected to the sample and hold circuit.

When changing channels, a delay is required before starting the next conversion. 0

Refer to [Section 32.2 “ADC Operation”](#) for more information.

32.1.3 ADC VOLTAGE REFERENCE

The ADPREF<1:0> bits of the ADREF register provide control of the positive voltage reference. The positive voltage reference can be:

- VREF+ pin
- VDD
- FVR 1.024V
- FVR 2.048V
- FVR 4.096V

The ADNREF bit of the ADREF register provides control of the negative voltage reference. The negative voltage reference can be:

- VREF- pin
- VSS

See [Section 29.0 “Fixed Voltage Reference \(FVR\)”](#) for more details on the Fixed Voltage Reference.

32.1.4 CONVERSION CLOCK

The source of the conversion clock is software selectable via the ADCLK register and the ADCS bits of the ADCON0 register. There are 66 possible clock options:

- Fosc/2
- Fosc/4
- Fosc/6
- Fosc/8
- Fosc/10
- .
- .
- .
- Fosc/128
- FRC (dedicated RC oscillator)

The time to complete one bit conversion is defined as TAD. One full 10-bit conversion requires 11.5 TAD periods as shown in [Figure 32-2](#).

For correct conversion, the appropriate TAD specification must be met. Refer to [Table 37-14](#) for more information. [Table 32-1](#) gives examples of appropriate ADC clock selections.

Note 1: Unless using the FRC, any changes in the system clock frequency will change the ADC clock frequency, which may adversely affect the ADC result.

2: The internal control logic of the ADC runs off of the clock selected by the ADCS bit of ADCON0. What this can mean is when the ADCS bit of ADCON0 is set to ‘1’ (ADC runs on FRC), there may be unexpected delays in operation when setting ADC control bits.

TABLE 32-1: ADC CLOCK PERIOD (TAD) Vs. DEVICE OPERATING FREQUENCIES^(1,4)

| ADC Clock Period (TAD) | | Device Frequency (Fosc) | | | | | | |
|------------------------|---------------------|-------------------------|-------------------------|-----------------------|-----------------------|------------------------|------------------------|-------------------------|
| ADC Clock Source | ADCS<5:0> | 64 MHz | 32 MHz | 20 MHz | 16 MHz | 8 MHz | 4 MHz | 1 MHz |
| FOSC/2 | 000000 | 31.25 ns ⁽²⁾ | 62.5 ns ⁽²⁾ | 100 ns ⁽²⁾ | 125 ns ⁽²⁾ | 250 ns ⁽²⁾ | 500 ns ⁽²⁾ | 2.0 μs |
| FOSC/4 | 000001 | 62.5 ns ⁽²⁾ | 125 ns ⁽²⁾ | 200 ns ⁽²⁾ | 250 ns ⁽²⁾ | 500 ns ⁽²⁾ | 1.0 μs | 4.0 μs |
| FOSC/6 | 000010 | 125 ns ⁽²⁾ | 187.5 ns ⁽²⁾ | 300 ns ⁽²⁾ | 375 ns ⁽²⁾ | 750 ns ⁽²⁾ | 1.5 μs | 6.0 μs |
| FOSC/8 | 000011 | 187.5 ns ⁽²⁾ | 250 ns ⁽²⁾ | 400 ns ⁽²⁾ | 500 ns ⁽²⁾ | 1.0 μs | 2.0 μs | 8.0 μs ⁽³⁾ |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| FOSC/16 | 000100 | 250 ns ⁽²⁾ | 500 ns ⁽²⁾ | 800 ns ⁽²⁾ | 1.0 μs | 2.0 μs | 4.0 μs | 16.0 μs ⁽³⁾ |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| FOSC/128 | 111111 | 2.0 μs | 4.0 μs | 6.4 μs | 8.0 μs | 16.0 μs ⁽³⁾ | 32.0 μs ⁽²⁾ | 128.0 μs ⁽²⁾ |
| FRC | ADCS(ADCON0<4>) = 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 |

Legend: Shaded cells are outside of recommended range.

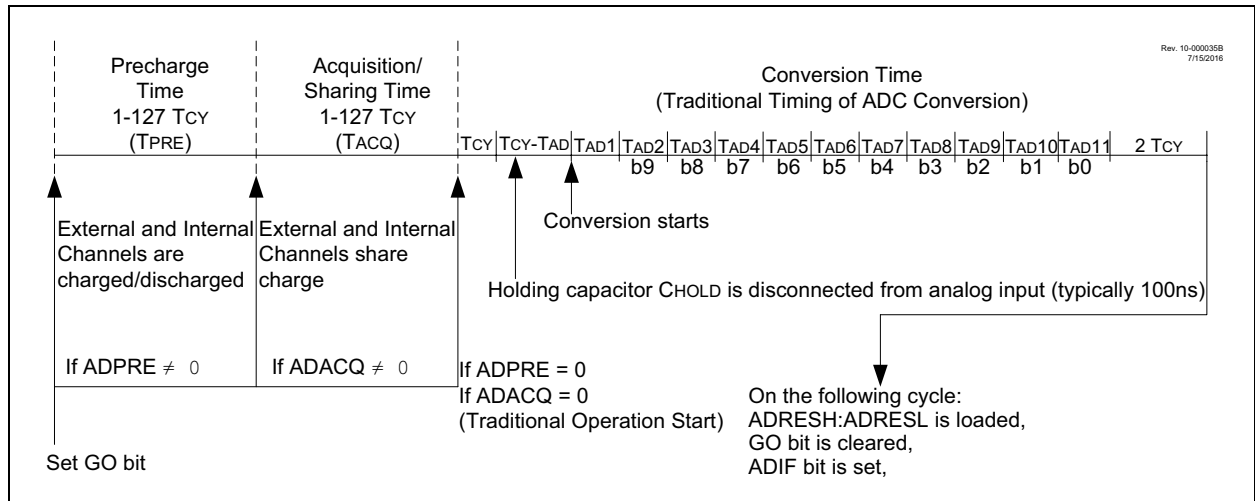
Note 1: See TAD parameter for FRC source typical TAD value.

2: These values violate the required TAD time.

3: Outside the recommended TAD time.

4: The ADC clock period (TAD) and total ADC conversion time can be minimized when the ADC clock is derived from the system clock FOSC. However, the FRC oscillator source must be used when conversions are to be performed with the device in Sleep mode.

FIGURE 32-2: ANALOG-TO-DIGITAL CONVERSION TAD CYCLES



32.1.5 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 PIR1 register. The ADC Interrupt Enable is the ADIE bit in the PIE1 register. The ADIF bit must be cleared in software.

Note 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 FRC oscillator is selected.

This interrupt can be generated while the device is operating or while in Sleep. If the device is in Sleep, the interrupt will wake-up the device. Upon waking from Sleep, the next instruction following the SLEEP instruction is always executed. If the user is attempting to wake-up from Sleep and resume in-line code execution, the ADIE bit of the PIE1 register and the PEIE bit of the INTCON register must both be set and the GIE bit of the INTCON register must be cleared. If all three of these bits are set, the execution will switch to the Interrupt Service Routine.

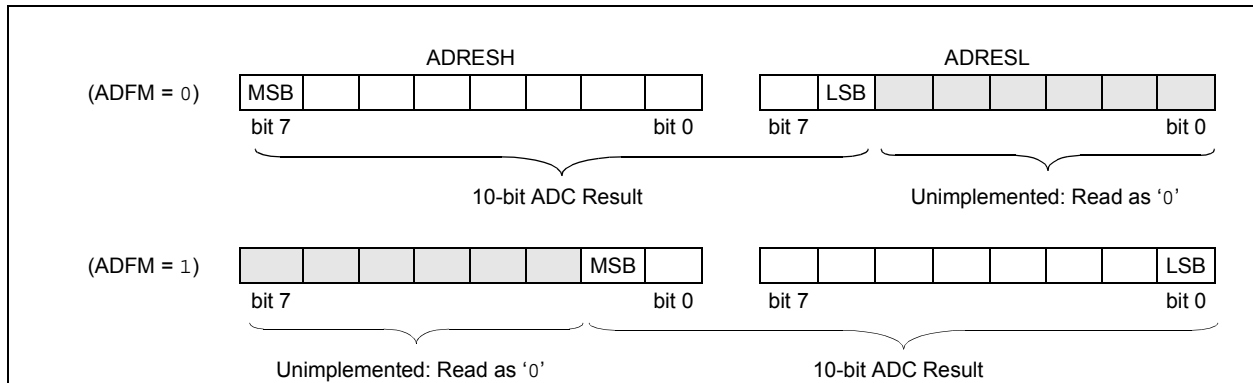
32.1.6 RESULT FORMATTING

The 10-bit ADC conversion result can be supplied in two formats, left justified or right justified. The ADFM bits of the ADCON0 register controls the output format.

Figure 32-3 shows the two output formats.

Writes to the ADRES register pair are always right justified regardless of the selected format mode. Therefore, data read after writing to ADRES when ADFRM0 = 0 will be shifted left six places. For example, writing 0xFF to ADRESL will be read as 0xC0 in ADRESL and 0x3F logical OR'd with whatever was in the two MS bits in ADRESH.

FIGURE 32-3: 10-BIT ADC CONVERSION RESULT FORMAT



32.2 ADC Operation

32.2.1 STARTING A CONVERSION

To enable the ADC module, the ADON bit of the ADCON0 register must be set to a '1'. A conversion may be started by any of the following:

- Software setting the ADGO bit of ADCON0 to '1'
- An external trigger (selected by [Register 32-3](#))
- A continuous-mode retrigger (see section [Section 32.5.8 "Continuous Sampling mode"](#))

Note: The ADGO bit should not be set in the same instruction that turns on the ADC. Refer to [Section 32.2.7 "ADC Conversion Procedure \(Basic Mode\)"](#).

32.2.2 COMPLETION OF A CONVERSION

When any individual conversion is complete, the value already in ADRES is written into ADPREV (if ADPSIS = 1) and the new conversion results appear in ADRES. When the conversion completes, the ADC module will:

- Clear the ADGO bit (unless the ADCONT bit of ADCON0 is set)
- Set the ADIF Interrupt Flag bit
- Set the ADMATH bit
- Update ADACC

When ADDSEN = 0 then after every conversion, or when ADDSEN = 1 then after every other conversion, the following events occur:

- ADERR is calculated
- ADTIF is set if ADERR calculation meets threshold comparison

Importantly, filter and threshold computations occur after the conversion itself is complete. As such, interrupt handlers responding to ADIF should check ADTIF before reading filter and threshold results.

32.2.3 TERMINATING A CONVERSION

If a conversion must be terminated before completion, the ADGO bit can be cleared in software. The ADRESH and ADRESL registers will be updated with the partially complete Analog-to-Digital conversion sample. Incomplete bits will match the last bit converted. In this case, filter and/or threshold occur.

Note: A device Reset forces all registers to their Reset state. Thus, the ADC module is turned off and any pending conversion is terminated.

32.2.4 ADC OPERATION DURING SLEEP

The ADC module can operate during Sleep. This requires the ADC clock source to be set to the FRC option. When the FRC 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 ADC module is turned off after the conversion completes, although the ADON bit remains set.

32.2.5 EXTERNAL TRIGGER DURING SLEEP

If the external trigger is received during sleep while ADC clock source is set to the FRC, 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 FRC, the trigger will be recorded, but the conversion will not begin until the device exits Sleep.

32.2.6 AUTO-CONVERSION TRIGGER

The Auto-conversion Trigger allows periodic ADC measurements without software intervention. When a rising edge of the selected source occurs, the ADGO bit is set by hardware.

The Auto-conversion Trigger source is selected with the ADOACT<4:0> bits of the ADOACT register.

Using the Auto-conversion Trigger does not assure proper ADC timing. It is the user's responsibility to ensure that the ADC timing requirements are met. See [Table 32-2](#) for auto-conversion sources.

TABLE 32-2: ADC AUTO-CONVERSION TABLE

| Source Peripheral | Description |
|-------------------|---|
| ADCACTPPS | Pin selected by ADCACTPPS |
| TMR0 | Timer0 overflow condition |
| TMR1/3/5/7 | Timer1/3/5/7 overflow condition |
| TMR2/4/6/8 | Match between Timer2/4/6/8 postscaled value and PR2/4/6/8 |
| CCP1/2/3/4/5 | CCP1/2/3/4/5 output |
| PWM/6/7 | PWM/6/7 output |
| C1/2/3 | Comparator C1/2/3 output |
| IOC | Interrupt-on-change interrupt trigger |
| ADERR | Read of ADERRH register |
| ADRESH | Read of ADRESH register |
| ADPCH | Write of ADPCH register |
| SMT1/2 | Signal Measurement Timer 1/2 Out |

32.2.7 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 (precharge+acquisition)
 - Turn on ADC module
3. Configure ADC interrupt (optional):
 - Clear ADC interrupt flag
 - Enable ADC interrupt
 - Enable peripheral interrupt (PEIE bit)
 - Enable global interrupt (GIE bit)⁽¹⁾
4. If ADACQ = 0, software must wait the required acquisition time⁽²⁾.
5. Start conversion by setting the ADGO bit.
6. Wait for ADC conversion to complete by one of the following:
 - Polling the ADGO bit
 - Waiting for the ADC interrupt (interrupts enabled)
7. Read ADC Result.
8. Clear the ADC interrupt flag (required if interrupt is enabled).

EXAMPLE 32-1: ADC CONVERSION

```

;This code block configures the ADC
;for polling, VDD and VSS references, FRC
;oscillator and AN0 input.
;
;Conversion start & polling for completion
;are included.
;
BANKSEL    ADCON1    ;
MOVLW     B'11110000' ;Right justify,
                                ;FRC oscillator
MOVWF     ADCON1    ;Vdd and Vss Vref
BANKSEL    TRISA     ;
BSF       TRISA,0   ;Set RA0 to input
BANKSEL    ANSEL     ;
BSF       ANSEL,0   ;Set RA0 to analog
BANKSEL    ADCON0    ;
MOVLW     B'00000001' ;Select channel AN0
MOVWF     ADCON0    ;Turn ADC On
CALL      SampleTime ;Acquisition delay
BSF       ADCON0,ADGO ;Start conversion
BTFSC    ADCON0,ADGO ;Is conversion done?
GOTO     $-1        ;No, test again
BANKSEL    ADRESH    ;
MOVF     ADRESH,W   ;Read upper 2 bits
MOVWF    RESULTHI   ;store in GPR space
BANKSEL    ADRESL    ;
MOVF     ADRESL,W   ;Read lower 8 bits
MOVWF    RESULTLO   ;Store in GPR space
    
```

Note 1: The global interrupt can be disabled if the user is attempting to wake-up from Sleep and resume in-line code execution.

2: Refer to [Section 32.3 “ADC Acquisition Requirements”](#).

32.3 ADC Acquisition Requirements

For the ADC to meet its specified accuracy, the charge holding capacitor (CHOLD) must be allowed to fully charge to the input channel voltage level. The Analog Input model is shown in Figure 32-4. The source impedance (RS) and the internal sampling switch (RSS) impedance directly affect the time required to charge the capacitor CHOLD. The sampling switch (RSS) impedance varies over the device voltage (VDD), refer to Figure 32-4. **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 must be completed before the conversion can be started. To calculate the minimum acquisition time, Equation 32-1 may be used. This equation assumes that 1/2 LSB error is used (1,024 steps for the ADC). The 1/2 LSB error is the maximum error allowed for the ADC to meet its specified resolution.

EQUATION 32-1: ACQUISITION TIME EXAMPLE

Assumptions: Temperature = 50°C and external impedance of 10kΩ 5.0V VDD

$$\begin{aligned} T_{ACQ} &= \text{Amplifier Settling Time} + \text{Hold Capacitor Charging Time} + \text{Temperature Coefficient} \\ &= T_{AMP} + T_C + T_{COFF} \\ &= 2\mu s + T_C + [(Temperature - 25^\circ C)(0.05\mu s/^\circ C)] \end{aligned}$$

The value for TC can be approximated with the following equations:

$$V_{APPLIED} \left(1 - \frac{1}{(2^{n+1}) - 1} \right) = V_{CHOLD} \quad ;[1] \text{ } V_{CHOLD} \text{ charged to within } 1/2 \text{ lsb}$$

$$V_{APPLIED} \left(1 - e^{-\frac{T_C}{RC}} \right) = V_{CHOLD} \quad ;[2] \text{ } V_{CHOLD} \text{ charge response to } V_{APPLIED}$$

$$V_{APPLIED} \left(1 - e^{-\frac{T_C}{RC}} \right) = V_{APPLIED} \left(1 - \frac{1}{(2^{n+1}) - 1} \right) \quad ;\text{combining [1] and [2]}$$

Note: Where n = number of bits of the ADC.

Solving for TC:

$$\begin{aligned} T_C &= -CHOLD(RIC + RSS + RS) \ln(1/2047) \\ &= -10pF(1k\Omega + 7k\Omega + 10k\Omega) \ln(0.0004885) \\ &= 1.37\mu s \end{aligned}$$

Therefore:

$$\begin{aligned} T_{ACQ} &= 2\mu s + 892ns + [(50^\circ C - 25^\circ C)(0.05\mu s/^\circ C)] \\ &= 4.62\mu s \end{aligned}$$

Note 1: The reference voltage (VREF) has no effect on the equation, since it cancels itself out.

2: The charge holding capacitor (CHOLD) is not discharged after each conversion.

3: The maximum recommended impedance for analog sources is 10 kΩ. This is required to meet the pin leakage specification.

FIGURE 32-4: ANALOG INPUT MODEL

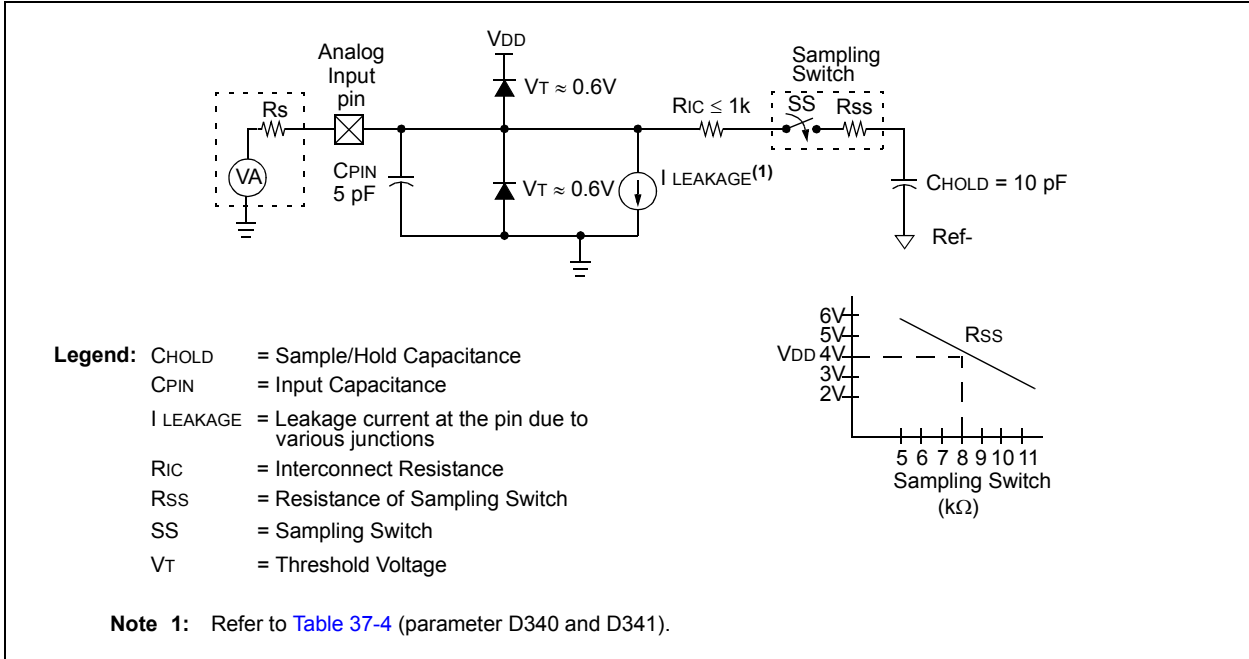
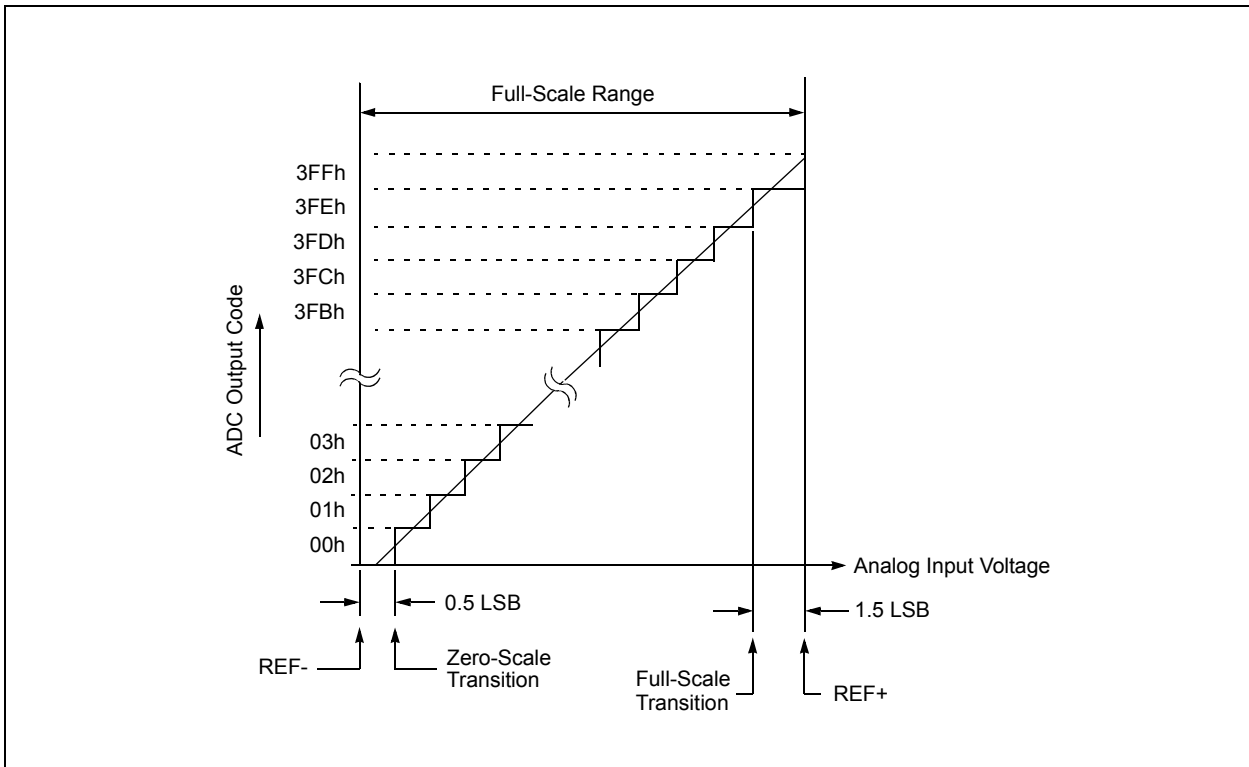


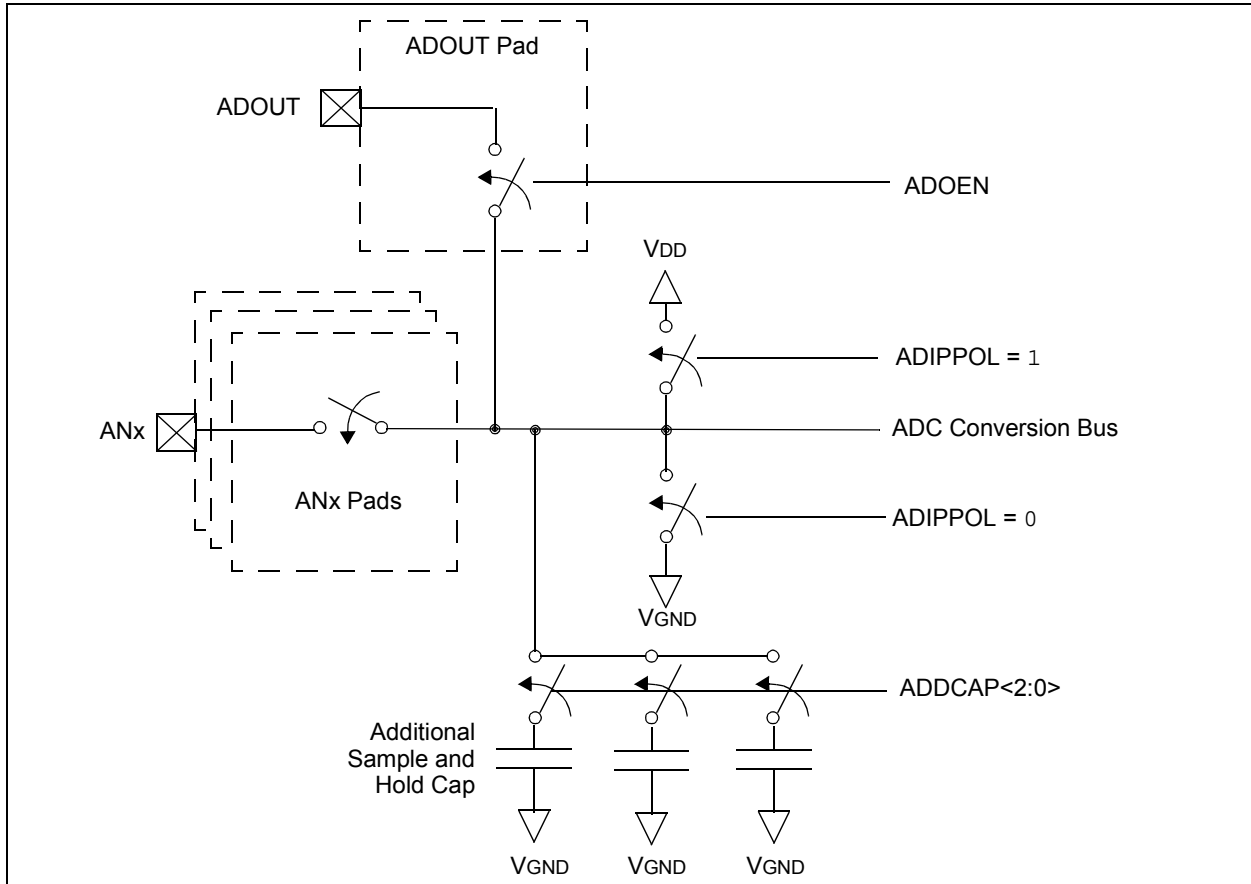
FIGURE 32-5: ADC TRANSFER FUNCTION



32.4 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. Figure 32-6 shows the basic block diagram of the CVD portion of the ADC module.

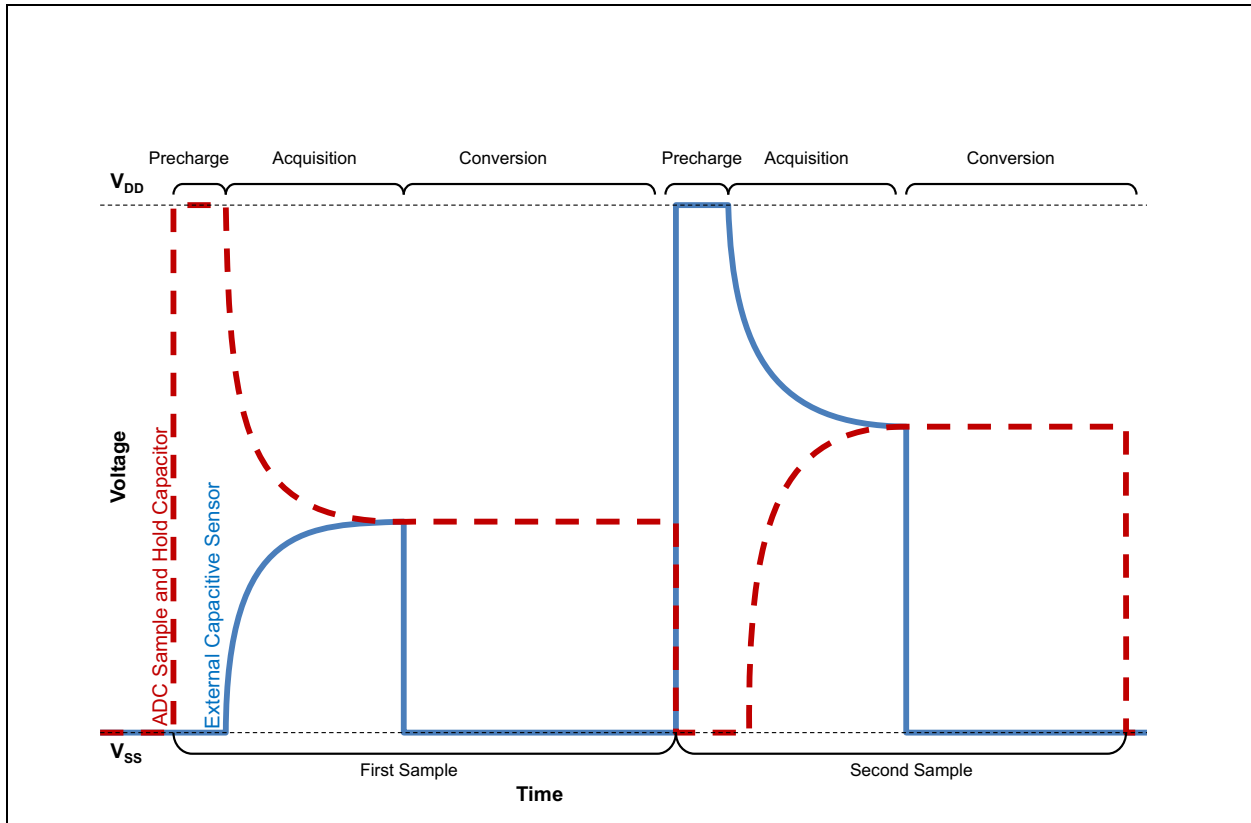
FIGURE 32-6: HARDWARE CAPACITIVE VOLTAGE DIVIDER BLOCK DIAGRAM



32.4.1 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 V_{SS} , while the path to the sensor node is also discharged to V_{DD} or V_{SS} . Typically, this node is discharged to the level opposite that of C_{HOLD} . When the precharge phase is complete, the V_{DD}/V_{SS} bias paths for the two nodes are shut off and C_{HOLD} and the path to the external sensor node are re-connected, 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 for both the C_{HOLD} and the inverted sensor nodes. [Figure 32-7](#) shows the waveform for two inverted CVD measurements, which is known as differential CVD measurement.

FIGURE 32-7: DIFFERENTIAL CVD MEASUREMENT WAVEFORM



32.4.2 PRECHARGE CONTROL

The Precharge stage is an optional period of time that brings the external channel and internal sample and hold capacitor to known voltage levels. Precharge is enabled by writing a non-zero value to the ADPRE register. This stage is initiated when an ADC conversion begins, either from setting the ADGO 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, CHOLD is disconnected from the outer portion of the sample path that leads to the external capacitive sensor and is connected to either VDD or VSS, depending on the value of the ADPPOL bit of ADCON1. At the same time, the port pin logic of the selected analog channel is overridden to drive a digital high or low out, in order to precharge the outer portion of the ADC's sample path, which includes the external sensor. The output polarity of this override is also determined by the ADPPOL bit of ADCON1. The amount of time that this charging needs is controlled by the ADPRE register.

Note: The external charging overrides the TRIS setting of the respective I/O pin. If there is a device attached to this pin, Precharge should not be used.

32.4.3 ACQUISITION CONTROL

The Acquisition stage is an optional 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. If ADPRE = 0, acquisition starts at the beginning of conversion. When ADPRE = 1, the acquisition stage begins when precharge 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 CHOLD. This allows charge averaging to proceed between the precharged channel and the CHOLD capacitor.

Note: When ADPRE! = 0, acquisition time cannot be '0'. In this case, setting ADACQ to '0' will set a maximum acquisition time (256 ADC clock cycles). When precharge is disabled, setting ADACQ to '0' will disable hardware acquisition time control.

32.4.4 GUARD RING OUTPUTS

Figure 32-8 shows a typical guard ring circuit. CGUARD represents the capacitance of the guard ring trace placed on the PCB board. The user selects values for RA and RB that will create a voltage profile on CGUARD, 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, see Application Note AN1478, "mTouch™ Sensing Solution Acquisition Methods Capacitive Voltage Divider" (DS01478).

The ADC has two guard ring drive outputs, ADGRDA and ADGRDB. These outputs can be routed through PPS controls to I/O pins (see Section 17.0 "Peripheral Pin Select (PPS) Module" for details) and the polarity of these outputs are controlled by the ADGPOL and ADIPEN bits of ADCON1.

At the start of the first precharge stage, both outputs are set to match the ADGPOL bit of ADCON1. Once the acquisition stage begins, ADGRDA changes polarity, while ADGRDB remains unchanged. When performing a double sample conversion, setting the ADIPEN bit of ADCON1 causes both guard ring outputs to transition to the opposite polarity of ADGPOL 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 32-8 and Figure 32-9.

FIGURE 32-8: GUARD RING CIRCUIT

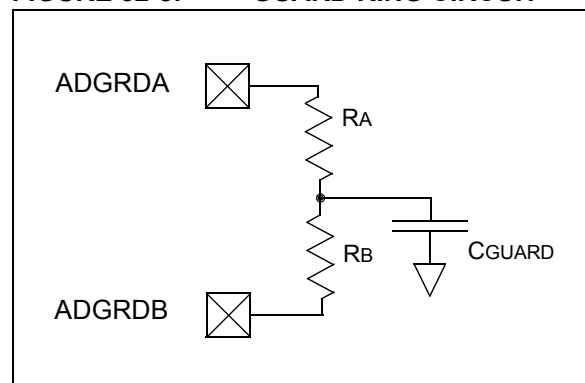


FIGURE 32-9: DIFFERENTIAL CVD WITH GUARD RING OUTPUT WAVEFORM

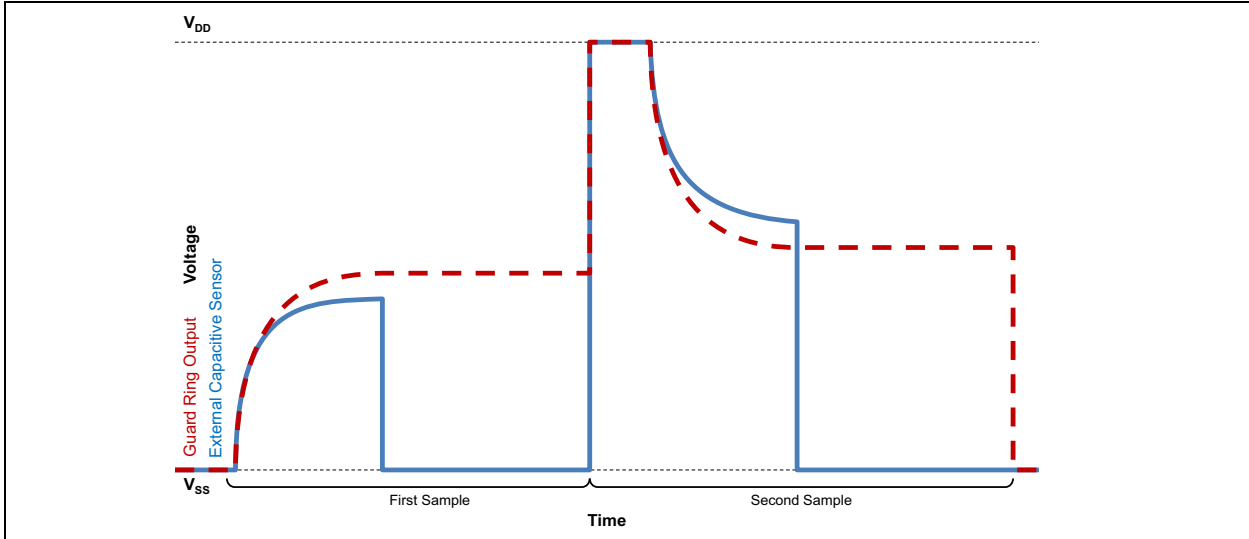
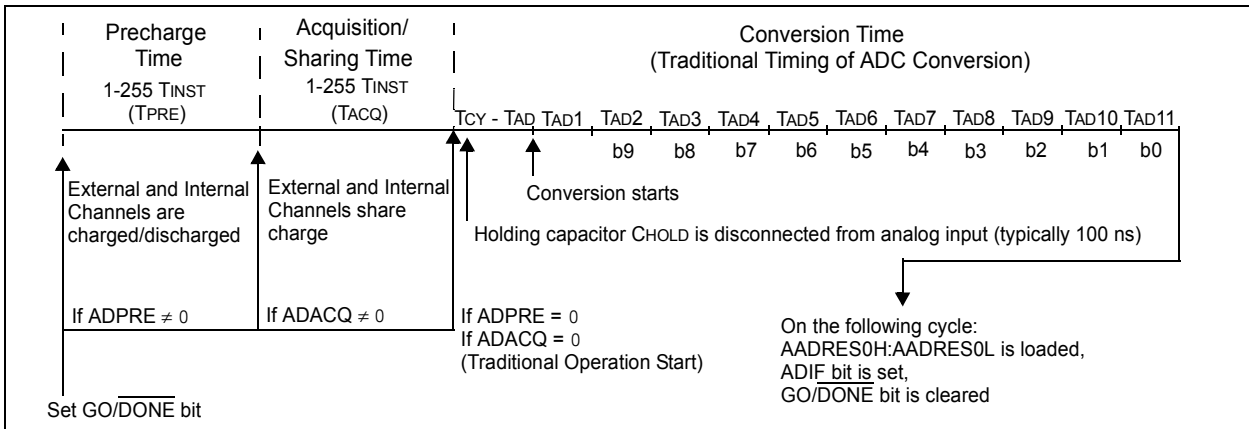


FIGURE 32-10: HARDWARE CVD SEQUENCE TIMING DIAGRAM



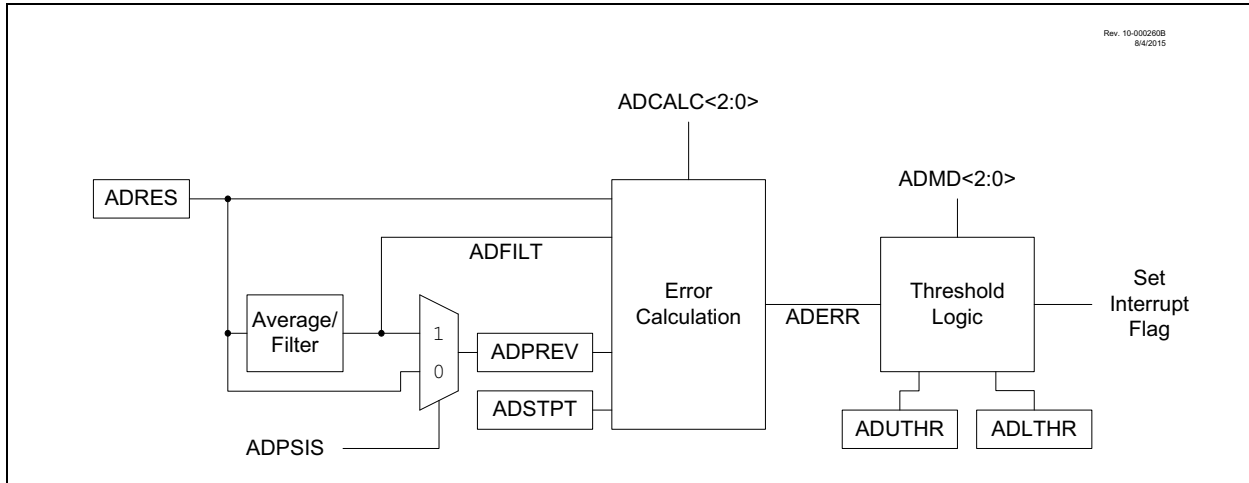
32.4.5 ADDITIONAL SAMPLE AND HOLD CAPACITANCE

Additional capacitance can be added in parallel with the internal sample and hold capacitor (CHOLD) by using the ADCAP register. This register selects a digitally programmable capacitance which 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. See [Figure 32-11](#).

32.5 Computation Operation

The ADC module hardware is equipped with post conversion computation features. These features provide data post-processing functions that can be operated on the ADC conversion result, including digital filtering/averaging and threshold comparison functions.

FIGURE 32-11: COMPUTATIONAL FEATURES SIMPLIFIED BLOCK DIAGRAM



The operation of the ADC computational features is controlled by **ADMD <2:0>** bits in the **ADCON2** register.

The module can be operated in one of five modes:

- **Basic:** This is a legacy mode. In this mode, ADC conversion occurs on single (**ADDSEN = 0**) or double (**ADDSEN = 1**) samples. **ADIF** is set after all the conversion are complete.
- **Accumulate:** With each trigger, the ADC conversion result is added to accumulator and **ADCNT** 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 **ADRPT** 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 [Table 32-3](#) below.

TABLE 32-3: COMPUTATION MODES

| Mode | ADMD | Bit Clear Conditions | Value after Trigger completion | | Threshold Operations | | | Value at ADTIF interrupt | | |
|--------------------|------|---|--|---|-------------------------------|-------------------------|-------------------|--------------------------|--------------------------|-------|
| | | ADACC and ADCNT | ADACC | ADCNT | Retrigger | Threshold Test | Interrupt | ADAOV | ADFLTR | ADCNT |
| Basic | 0 | ADACL _R = 1 | Unchanged | Unchanged | No | Every Sample | If threshold=true | N/A | N/A | count |
| Accumulate | 1 | ADACL _R = 1 | S + ADACC or (S2-S1) + ADACC | If (ADCNT=FF): ADCNT, otherwise: ADCNT+1 | No | Every Sample | If threshold=true | ADACC Overflow | ADACC/2 ^{ADCRS} | count |
| Average | 2 | ADACL _R = 1 or ADCNT >= ADRPT at ADGO or retrigger | S + ADACC or (S2-S1) + ADACC | If (ADCNT=FF): ADCNT, otherwise: ADCNT+1 | No | If ADCNT >= ADRPT | If threshold=true | ADACC Overflow | ADACC/2 ^{ADCRS} | count |
| Burst Average | 3 | ADACL _R = 1 or ADGO 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 ^{ADCRS} | ADRPT |
| Low-pass Filter | 4 | ADACL _R = 1 | S+ADACC-ADACC/ 2 ^{ADCRS} or (S2-S1)+ADACC-ADACC/ 2 ^{ADCRS} | If (ADCNT=FF): ADCNT, otherwise: ADCNT+1 | No | If ADCNT >= ADRPT | If threshold=true | ADACC Overflow | Filtered Value | count |

Note: S1 and S2 are abbreviations for Sample 1 and Sample 2, respectively. When ADDSEN = 0, S1 = ADRES; When ADDSEN = 1, S1 = ADPREV and S2 = ADRES.

32.5.1 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 is a 16-bit wide register which can be accessed through the ADACCH:ADACCL register pair.

Upon each trigger event (the ADGO bit set or external event trigger), the ADC conversion result is added to the accumulator. If the accumulated value exceeds $2^{(\text{accumulator_width}-1)} = 2^{16} = 65535$, the overflow bit ADAOV in the ADSTAT register is set.

The number of samples to be accumulated is determined by the ADRPT (A/D Repeat Setting) register. Each time a sample is added to the accumulator, the ADCNT register is incremented. Once ADRPT samples are accumulated (ADCNT = ADRPT), an accumulator clear command can be issued by the software by setting the ADACLR bit in the ADCON2 register. Setting the ADACLR bit will also clear the ADAOV (Accumulator overflow) bit in the ADSTAT

register, as well as the ADCNT register. The ADACLR bit is cleared by the hardware when accumulator clearing action is complete.

Note: When ADC is operating from FRC, five FRC clock cycles are required to execute the ADACC clearing operation.

The ADCRS <2:0> bits in the ADCON2 register control the data shift on the accumulator result, which effectively divides the value in accumulator (ADACCH:ADACCL) register pair. For the Accumulate mode of the digital filter, the shift provides a simple scaling operation. For the Average/Burst Average mode, the shift bits are used to determine number of samples for averaging. For the Low-pass Filter mode, the shift is an integral part of the filter, and determines the cut-off frequency of the filter. Table 32-4 shows the -3 dB cut-off frequency in ωT (radians) and the highest signal attenuation obtained by this filter at nyquist frequency ($\omega T = \pi$).

TABLE 32-4: LOW-PASS FILTER -3 dB CUT-OFF FREQUENCY

| ADCRS | ωT (radians) @ -3 dB Frequency | dB @ $F_{\text{nyquist}}=1/(2T)$ |
|-------|--|----------------------------------|
| 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 |
| 7 | 0.0078 | -48.1 |

32.5.2 BASIC MODE

Basic mode (ADMD = 000) 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 features involving the digital filter/average features are used.

32.5.3 ACCUMULATE MODE

In Accumulate mode (ADMD = 001), after every conversion, the ADC result is added to the ADACC register. The ADACC register is right-shifted by the value of the ADCRS bits in the ADCON2 register. This right-shifted value is copied in to the ADFLT register. The Formatting mode does not affect the right-justification of the ADACC value. Upon each sample, ADCNT is also incremented, incrementing the number of samples accumulated. After each sample and accumulation, the ADACC value has a threshold comparison performed on it (see Section 32.5.7 “Threshold Comparison”) and the ADTIF interrupt may trigger.

32.5.4 AVERAGE MODE

In Average Mode (ADMD = 010), the ADACC registers accumulate with each ADC sample, much as in Accumulate mode, and the ADCNT register increments with each sample. The ADFLT register is also updated with the right-shifted value of the ADACC register. The value of the ADCRS 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 $\text{ADRPT} = 2^{\text{ADCNT}}$, then the final accumulated value will be divided by number of samples, allowing for a threshold comparison operation on the average of all gathered samples.

32.5.5 BURST AVERAGE MODE

The Burst Average mode (ADMD = 011) acts the same as the Average mode in most respects. The one way it differs is that it continuously retriggers ADC sampling until the ADCNT value is greater than or equal to ADRPT, even if Continuous Sampling mode (see [Section 32.5.8 “Continuous Sampling mode”](#)) is not enabled. This allows for a threshold comparison on the average of a short burst of ADC samples.

32.5.6 LOW-PASS FILTER MODE

The Low-pass Filter mode (ADMD = 100) acts similarly to the Average mode in how it handles samples (accumulates samples until ADCNT value greater than or equal to ADRPT, then triggers 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 average, then performs a threshold comparison on the results. (see [Table 32-3](#) for a more detailed description of the mathematical operation). In this mode, the ADCRS bits determine the cut-off frequency of the low-pass filter (as demonstrated by [Table 32-4](#)).

32.5.7 THRESHOLD COMPARISON

At the end of each computation:

- The conversion results are latched and held stable at the end-of-conversion.
- The error is calculated based on a difference calculation which is selected by the ADCALC<2:0> bits in the ADCON3 register. The value can be one of the following calculations (see [Register 32-4](#) for more details):
 - The first derivative of single measurements
 - The CVD result in CVD mode
 - The current result vs. a setpoint
 - The current result vs. the filtered/average result
 - The first derivative of the filtered/average value
 - Filtered/average value vs. a setpoint
- The result of the calculation (ADERR) is compared to the upper and lower thresholds, ADUTH<ADUTHH:ADUTHL> and ADLTH<ADLTHH:ADLTHL> registers, to set the ADUTHR and ADLTHR flag bits. The threshold logic is selected by ADTMD<2:0> bits in the ADCON3 register. 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.

Note 1: The threshold tests are signed operations.

2: If ADAOV is set, a threshold interrupt is signaled.

32.5.8 CONTINUOUS SAMPLING MODE

Setting the ADCONT bit in the ADCON0 register automatically retriggers a new conversion cycle after updating the ADACC register. That means the ADGO bit is set to generate automatic retriggering, until the device Reset occurs or the A/D Stop-on-interrupt bit (ADSOI in the ADCON3 register) is set (correct logic).

32.5.9 DOUBLE SAMPLE CONVERSION

Double sampling is enabled by setting the ADDSEN bit of the ADCON1 register. When this bit is set, two conversions are required before the module will calculate threshold error (each conversion must still be triggered separately). The first conversion will set the ADMATH bit of the ADSTAT register 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 ADPSIS) 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 ADCALC).

32.6 Register Definitions: ADC Control

REGISTER 32-1: ADCON0: ADC CONTROL REGISTER 0

| R/W-0/0 | R/W-0/0 | U-0 | R/W-0/0 | U-0 | R/W-0/0 | U-0 | R/W/HC-0 |
|---------|---------|-----|---------|-----|---------|-----|----------|
| ADON | ADCONT | — | ADCS | — | ADFM | — | ADGO |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | HC = Bit is cleared by hardware |

| | |
|-------|--|
| bit 7 | ADON: ADC Enable bit 1 = ADC is enabled 0 = ADC is disabled |
| bit 6 | ADCONT: ADC Continuous Operation Enable bit 1 = ADGO is retriggered upon completion of each conversion trigger until ADTIF is set (if ADSOI is set) or until ADGO is cleared (regardless of the value of ADSOI) 0 = ADC is cleared upon completion of each conversion trigger |
| bit 5 | Unimplemented: Read as '0' |
| bit 4 | ADCS: ADC Clock Selection bit 1 = Clock supplied from FRC dedicated oscillator 0 = Clock supplied by FOSC, divided according to ADCLK register |
| bit 3 | Unimplemented: Read as '0' |
| bit 2 | ADFM: ADC results Format/alignment Selection 1 = ADRES and ADPREV data are right-justified 0 = ADRES and ADPREV data are left-justified, zero-filled |
| bit 1 | Unimplemented: Read as '0' |
| bit 0 | ADGO: ADC Conversion Status bit 1 = ADC conversion cycle in progress. Setting this bit starts an ADC conversion cycle. The bit is cleared by hardware as determined by the ADCONT bit 0 = ADC conversion completed/not in progress |

REGISTER 32-2: ADCON1: ADC CONTROL REGISTER 1

| | | | | | | | |
|---------|---------|---------|-----|-----|-----|-----|---------|
| R/W-0/0 | R/W-0/0 | R/W-0/0 | U-0 | U-0 | U-0 | U-0 | R/W-0/0 |
| ADPPOL | ADIPEN | ADGPOL | – | – | – | – | ADDSEN |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7 **ADPPOL:** Precharge Polarity bit
If $ADPRE > 0x00$:

| ADPPOL | Action During 1st Precharge Stage | |
|--------|------------------------------------|----------------------------------|
| | External (selected analog I/O pin) | Internal (AD sampling capacitor) |
| 1 | Shorted to AVDD | C_{HOLD} shorted to VSS |
| 0 | Shorted to VSS | C_{HOLD} shorted to AVDD |

Otherwise:

The bit is ignored

bit 6 **ADIPEN:** A/D Inverted Precharge Enable bit

If $ADDSEN = 1$

1 = The precharge and guard signals in the second conversion cycle are the opposite polarity of the first cycle

0 = Both Conversion cycles use the precharge and guards specified by ADPPOL and ADGPOL

Otherwise:

The bit is ignored

bit 5 **ADGPOL:** Guard Ring Polarity Selection bit

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 4-1 **Unimplemented:** Read as '0'

bit 0 **ADDSEN:** Double-sample enable bit

1 = Two conversions are performed on each trigger. Data from the first conversion appears in ADPREV

0 = One conversion is performed for each trigger

REGISTER 32-3: ADCON2: ADC CONTROL REGISTER 2

| | | | | | | | |
|---------|------------|---------|---------|----------|-----------|---------|---------|
| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W/HC-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| ADPSIS | ADCRS<2:0> | | | ADACL | ADMD<2:0> | | |
| bit 7 | | | | bit 0 | | | |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | HC = Bit is cleared by hardware |

- bit 7 **ADPSIS:** ADC Previous Sample Input Select bits
 1 = ADFLTR is transferred to ADPREV at start-of-conversion
 0 = ADRES is transferred to ADPREV at start-of-conversion
- bit 6-4 **ADCRS<2:0>:** ADC Accumulated Calculation Right Shift Select bits
 If ADMD = 100:
 Low-pass filter time constant is 2^{ADCRS} , filter gain is 1:1
 If ADMD = 001, 010 or 011:
 The accumulated value is right-shifted by ADCRS (divided by 2^{ADCRS})^(1,2)
Otherwise:
 Bits are ignored
- bit 3 **ADACL:** A/D Accumulator Clear Command bit⁽³⁾
 0 = Clearing action is complete (or not started)
 1 = ADACC, ADAOV and ADCNT registers are cleared
- bit 2-0 **ADMD<2:0>:** ADC Operating Mode Selection bits⁽⁴⁾
 111-101 = Reserved
 100 = Low-pass Filter mode
 011 = Burst Average mode
 010 = Average mode
 001 = Accumulate mode
 000 = Basic (Legacy) mode

- Note 1:** To correctly calculate an average, the number of samples (set in ADRPT) must be 2^{ADCRS} .
- 2:** ADCRS = 3'b111 is a reserved option.
- 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 [Table 32-2](#) for Full mode descriptions.

REGISTER 32-4: ADCON3: ADC CONTROL REGISTER 3

| | | | | | | | |
|-------|-------------|---------|---------|----------|------------|---------|---------|
| U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W/HC-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| — | ADCALC<2:0> | | | ADSOI | ADTMD<2:0> | | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | HC = Bit is cleared by hardware |

bit 7 **Unimplemented:** Read as '0'

bit 6-4 **ADCALC<2:0>:** ADC Error Calculation Mode Select bits

| ADCALC | Action During 1st Precharge Stage | | Application |
|--------|-----------------------------------|--|--|
| | ADDSSEN = 0 Single-Sample Mode | ADDSSEN = 1 CVD Double-Sample Mode ⁽¹⁾ | |
| 111 | Reserved | Reserved | Reserved |
| 110 | Reserved | Reserved | Reserved |
| 101 | ADLFTR-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 in CVD mode ⁽²⁾ |

bit 3 **ADSOI:** ADC Stop-on-Interrupt bit

If ADCONT = 1 :

- 1 = ADGO is cleared when the threshold conditions are met, otherwise the conversion is retrIGGERED
- 0 = ADGO is not cleared by hardware, must be cleared by software to stop retriggers

bit 2-0 **ADTMD<2:0>:** Threshold Interrupt Mode Select bits

- 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

Note 1: When ADPSIS = 0, the value of ADRES-ADPREV) is the value of (S2-S1) from [Table 32-3](#).

2: When ADPSIS = 0

3: When ADPSIS = 1.

REGISTER 32-5: ADSTAT: ADC STATUS REGISTER

| | | | | | | | |
|-------|--------|--------|-------------|-----|-------------|-------|-------|
| R-0/0 | R-0/0 | R-0/0 | R/HS/HC-0/0 | U-0 | R-0/0 | R-0/0 | R-0/0 |
| ADAOV | ADUTHR | ADLTHR | ADMATH | — | ADSTAT<2:0> | | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | HS/HC = Bit is set/cleared by hardware |

- bit 7 **ADAOV:** ADC Accumulator Overflow bit
1 = ADC accumulator or ADERR calculation have overflowed
0 = ADC accumulator and ADERR calculation have not overflowed
- bit 6 **ADUTHR:** ADC Module Greater-than Upper Threshold Flag bit
1 = ADERR > ADUTH
0 = ADERR ≤ ADUTH
- bit 5 **ADLTHR:** ADC Module Less-than Lower Threshold Flag bit
1 = ADERR < ADLTH
0 = ADERR ≥ ADLTH
- bit 4 **ADMATH:** ADC Module Computation Status bit
1 = Registers ADACC, ADFLTR, ADUTH, ADLTH and the ADAOV bit are updating or have already updated
0 = Associated registers/bits have not changed since this bit was last cleared
- bit 3 **Unimplemented:** Read as '0'
- bit 2-0 **ADSTAT<2:0>:** ADC Module Cycle Multistage Status bits⁽¹⁾
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 = Not used
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

Note 1: If ADCS = 1, and FOSC < FRC, these bits may be invalid.

REGISTER 32-6: ADCLK: ADC CLOCK SELECTION REGISTER

| | | | | | | | |
|-------|-----|-----------|---------|---------|---------|---------|---------|
| U-0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| — | — | ADCS<5:0> | | | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-6 **Unimplemented:** Read as '0'

bit 5-0 **ADCS<5:0>:** ADC Conversion Clock Select bits
 111111 = Fosc/128
 111110 = Fosc/126
 111101 = Fosc/124
 •
 •
 •
 000000 = Fosc/2

REGISTER 32-7: ADREF: ADC REFERENCE SELECTION REGISTER

| | | | | | | | |
|-------|-----|-----|---------|-----|-----|-------------|---------|
| U-0 | U-0 | U-0 | R/W-0/0 | U-0 | U-0 | R/W-0/0 | R/W-0/0 |
| — | — | — | ADNREF | — | — | ADPREF<1:0> | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-5 **Unimplemented:** Read as '0'

bit 4 **ADNREF:** ADC Negative Voltage Reference Selection bit
 1 = VREF- is connected to external VREF-
 0 = VREF- is connected to AVSS

bit 3-2 **Unimplemented:** Read as '0'

bit 1-0 **ADPREF:** ADC Positive Voltage Reference Selection bits
 11 = VREF+ is connected to internal Fixed Voltage Reference (FVR) module
 10 = VREF+ is connected to external VREF+
 01 = Reserved
 00 = VREF+ is connected to VDD

REGISTER 32-8: ADPCH: ADC POSITIVE CHANNEL SELECTION REGISTER

| | | | | | | | |
|-------|-----|------------|---------|---------|---------|---------|---------|
| U-0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| — | — | ADPCH<5:0> | | | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-6 **Unimplemented:** Read as '0'

bit 5-0 **ADPCH<5:0>:** ADC Positive Input Channel Selection bits

| | |
|---|--|
| 111111 = Fixed Voltage Reference (FVR) ⁽²⁾ | 011111 = AND7 |
| 111110 = DAC1 output ⁽¹⁾ | 011110 = AND6 |
| 111101 = Temperature Indicator ⁽³⁾ | 011101 = AND5 |
| 111100 = AVss (Analog Ground) | 011100 = AND4 |
| 111011 = Reserved. No channel connected. | 011011 = AND3 |
| • | 011010 = AND2 |
| • | 011001 = AND1 |
| • | 011000 = AND0 |
| 111000 = Reserved. No channel connected. | 010111 = Reserved. No channel connected. |
| 110111 = ANG7 | • |
| 110110 = ANG6 | • |
| 110101 = Reserved | • |
| 110100 = ANG4 | 010000 = Reserved. No channel connected. |
| 110011 = ANG3 | 001111 = ANB7 |
| 110010 = ANG2 | 001110 = ANB6 |
| 110001 = ANG1 | 001101 = ANB5 |
| 110000 = ANG0 | 001100 = ANB4 |
| 101111 = ANF7 | 001011 = ANB3 |
| 101110 = ANF6 | 001010 = ANB2 |
| 101101 = ANF5 | 001001 = ANB1 |
| 101100 = ANF4 | 001000 = ANB0 |
| 101011 = ANF3 | 000111 = ANA7 |
| 101010 = ANF2 | 000110 = ANA6 |
| 101001 = ANF1 | 000101 = ANA5 |
| 101000 = ANF0 | 000100 = ANA4 |
| 100111 = ANE7 | 000011 = ANA3 |
| 100110 = ANE6 | 000010 = ANA2 |
| 100101 = ANE5 | 000001 = ANA1 |
| 100100 = ANE4 | 000000 = ANA0 |
| 100011 = ANE3 | |
| 100010 = ANE2 | |
| 100001 = ANE1 | |
| 100000 = ANE0 | |

- Note** 1: See [Section 31.0 “5-Bit Digital-to-Analog Converter \(DAC\) Module”](#) for more information.
 2: See [Section 29.0 “Fixed Voltage Reference \(FVR\)”](#) for more information.
 3: See [Section 30.0 “Temperature Indicator Module”](#) for more information.

REGISTER 32-9: ADPRE: ADC PRECHARGE TIME CONTROL REGISTER

| | | | | | | | |
|------------|---------|---------|---------|---------|---------|---------|---------|
| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| ADPRE<7:0> | | | | | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-0 **ADPRE<7:0>**: Precharge Time Select bits
 11111111 = Precharge time is 255 clocks of the selected ADC clock
 11111110 = Precharge time is 254 clocks of the selected ADC clock
 •
 •
 •
 00000001 = Precharge time is 1 clock of the selected ADC clock
 00000000 = Precharge time is not included in the data conversion cycle

REGISTER 32-10: ADACQ: ADC ACQUISITION TIME CONTROL REGISTER

| | | | | | | | |
|------------|---------|---------|---------|---------|---------|---------|---------|
| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| ADACQ<7:0> | | | | | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-0 **ADACQ<7:0>**: Acquisition (charge share time) Select bits
 11111111 = Acquisition time is 255 clocks of the selected ADC clock
 11111110 = Acquisition time is 254 clocks of the selected ADC clock
 •
 •
 •
 00000001 = Acquisition time is 1 clock of the selected ADC clock
 00000000 = Acquisition time is not included in the data conversion cycle

Note: If ADPRE is not equal to '0', then ADACQ = b'00000000 means Acquisition time is 256 clocks of the selected ADC clock.

REGISTER 32-11: ADCAP: ADC ADDITIONAL SAMPLE CAPACITOR SELECTION REGISTER

| | | | | | | | | |
|-------|-----|-----|------------|---------|---------|---------|---------|-------|
| U-0 | U-0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | |
| — | — | — | ADCAP<4:0> | | | | | |
| bit 7 | | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-5 **Unimplemented:** Read as '0'

bit 4-0 **ADCAP<4:0>:** ADC Additional Sample Capacitor Selection bits

11111 = 31 pF

11110 = 30 pF

11101 = 29 pF

•

•

•

00011 = 3 pF

00010 = 2 pF

00001 = 1 pF

00000 = No additional capacitance

REGISTER 32-12: ADRPT: ADC REPEAT SETTING REGISTER

| | | | | | | | | |
|------------|---------|---------|---------|---------|---------|---------|---------|-------|
| R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | |
| ADRPT<7:0> | | | | | | | | |
| bit 7 | | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-0 **Unimplemented:** Read as '0'

bit 7-0 **ADRPT<7:0>:** ADC Repeat Threshold bits

Counts the number of times that the ADC has been triggered and is used along with ADCNT to determine when the error threshold is checked when the computation is Low-pass Filter, Burst Average, or Average modes. See [Table 32-3](#) for more details.

REGISTER 32-13: ADCNT: ADC REPEAT COUNTER REGISTER

| | | | | | | | |
|------------|---------|---------|---------|---------|---------|---------|---------|
| R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u |
| ADCNT<7:0> | | | | | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-0 **ADCNT<7:0>**: ADC Repeat Count bits
 Determines the number of times that the ADC is triggered before the threshold is checked when the computation is Low-pass Filter, Burst Average, or Average modes. See [Table 32-2](#) for more details.

REGISTER 32-14: ADFLTRH: ADC FILTER HIGH BYTE REGISTER

| | | | | | | | |
|--------------|-----|-----|-----|-----|-----|-----|-------|
| R-x | R-x | R-x | R-x | R-x | R-x | R-x | R-x |
| ADFLTR<15:8> | | | | | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-0 **ADFLTR<15:8>**: ADC Filter Output Most Significant bits
 In Accumulate, Average, and Burst Average mode, this is equal to ADACC right shifted by the ADCRS bits of ADCON2. In LPF mode, this is the output of the Low-pass Filter.

REGISTER 32-15: ADFLTRL: ADC FILTER LOW BYTE REGISTER

| | | | | | | | |
|-------------|-----|-----|-----|-----|-----|-----|-------|
| R-x | R-x | R-x | R-x | R-x | R-x | R-x | R-x |
| ADFLTR<7:0> | | | | | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-0 **ADFLTR<7:0>**: ADC Filter Output Least Significant bits
 In Accumulate, Average, and Burst Average mode, this is equal to ADACC right shifted by the ADCRS bits of ADCON2. In LPF mode, this is the output of the Low-pass Filter.

REGISTER 32-16: ADRESH: ADC RESULT REGISTER HIGH, ADFM = 0

| | | | | | | | |
|------------|---------|---------|---------|---------|---------|---------|---------|
| R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u |
| ADRES<9:2> | | | | | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-0 **ADRES<9:2>**: ADC Result Register bits
Upper eight bits of 10-bit conversion result.

REGISTER 32-17: ADRESL: ADC RESULT REGISTER LOW, ADFM = 0

| | | | | | | | |
|------------|---------|-----|-----|-----|-----|-----|-------|
| R/W-x/u | R/W-x/u | U-0 | U-0 | U-0 | U-0 | U-0 | U-0 |
| ADRES<1:0> | | — | — | — | — | — | — |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-6 **ADRES<1:0>**: ADC Result Register bits. Lower two bits of 10-bit conversion result.
bit 5-0 **Reserved**: Do not use.

REGISTER 32-18: ADRESH: ADC RESULT REGISTER HIGH, ADFM = 1

| | | | | | | | |
|-------|-----|-----|-----|-----|-----|------------|---------|
| U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | R/W-x/u | R/W-x/u |
| — | — | — | — | — | — | ADRES<9:8> | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-2 **Reserved**: Do not use.
bit 1-0 **ADRES<9:8>**: ADC Sample Result bits. Upper two bits of 10-bit conversion result.

REGISTER 32-19: ADRESL: ADC RESULT REGISTER LOW, ADFM = 1

| | | | | | | | |
|------------|---------|---------|---------|---------|---------|---------|---------|
| R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u |
| ADRES<7:0> | | | | | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-0 **ADRES<7:0>**: ADC Result Register bits. Lower eight bits of 10-bit conversion result.

REGISTER 32-20: ADPREVH: ADC PREVIOUS RESULT REGISTER

| | | | | | | | |
|--------------|-----|-----|-----|-----|-----|-----|-------|
| R-x | R-x | R-x | R-x | R-x | R-x | R-x | R-x |
| ADPREV<15:8> | | | | | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-0 **ADPREV<15:8>**: Previous ADC Results bits
If ADPSIS = 1:
 Upper byte of ADFLTR at the start of current ADC conversion
If ADPSIS = 0:
 Upper bits of ADRES at the start of current ADC conversion⁽¹⁾

Note 1: If ADPSIS = 0, ADPREVH and ADPREVL are formatted the same way as ADRES is, depending on the ADFM bit.

REGISTER 32-21: ADPREVL: ADC PREVIOUS RESULT REGISTER

| | | | | | | | |
|-------------|-----|-----|-----|-----|-----|-----|-------|
| R-x | R-x | R-x | R-x | R-x | R-x | R-x | R-x |
| ADPREV<7:0> | | | | | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-0 **ADPREV<7:0>**: Previous ADC Results bits
 If ADPSIS = 1:
 Lower byte of ADFLTR at the start of current ADC conversion
 If ADPSIS = 0:
 Lower bits of ADRES at the start of current ADC conversion⁽¹⁾

Note 1: If ADPSIS = 0, ADPREVH and ADPREVL are formatted the same way as ADRES is, depending on the ADFM bit.

REGISTER 32-22: ADACCH: ADC ACCUMULATOR REGISTER HIGH

| | | | | | | | |
|-------------|---------|---------|---------|---------|---------|---------|---------|
| R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u |
| ADACC<15:8> | | | | | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-0 **ADACC<15:8>**: ADC Accumulator MSB. Upper eight bits of accumulator value. See [Table 32-2](#) for more details.

REGISTER 32-23: ADACCL: ADC ACCUMULATOR REGISTER LOW

| | | | | | | | |
|------------|---------|---------|---------|---------|---------|---------|---------|
| R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u |
| ADACC<7:0> | | | | | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-0 **ADACC<7:0>**: ADC Accumulator LSB. Lower eight bits of accumulator value. See [Table 32-2](#) for more details.

REGISTER 32-24: ADSTPTH: ADC THRESHOLD SETPOINT REGISTER HIGH

| | | | | | | | |
|--------------|---------|---------|---------|---------|---------|---------|---------|
| R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u | R/W-x/u |
| ADSTPT<15:8> | | | | | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-0 **ADSTPT<15:8>**: ADC Threshold Setpoint MSB. Upper byte of ADC threshold setpoint, depending on ADCALC, may be used to determine ADERR, see [Register 23-1](#) for more details.

REGISTER 32-25: ADSTPTL: ADC THRESHOLD SETPOINT REGISTER LOW

| | | | | | | | |
|-------------|---------|---------|---------|---------|---------|---------|---------|
| R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x |
| ADSTPT<7:0> | | | | | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-0 **ADSTPT<7:0>**: ADC Threshold Setpoint LSB. Lower byte of ADC threshold setpoint, depending on ADCALC, may be used to determine ADERR, see [Register 23-1](#) for more details.

REGISTER 32-26: ADERRH: ADC SETPOINT ERROR REGISTER HIGH

| | | | | | | | |
|------------|-----|-----|-----|-----|-----|-----|-------|
| R-x | R-x | R-x | R-x | R-x | R-x | R-x | R-x |
| ADERR<7:0> | | | | | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-0 **ADERR<7:0>**: ADC Setpoint Error MSB. Upper byte of ADC Setpoint Error. Setpoint Error calculation is determined by ADCALC bits of ADCON3, see [Register 23-1](#) for more details.

REGISTER 32-27: ADERRL: ADC SETPOINT ERROR LOW BYTE REGISTER

| | | | | | | | |
|------------|-----|-----|-----|-----|-----|-----|-------|
| R-x | R-x | R-x | R-x | R-x | R-x | R-x | R-x |
| ADERR<7:0> | | | | | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-0 **ADERR<7:0>**: ADC Setpoint Error LSB. Lower byte of ADC Setpoint Error calculation is determined by ADCALC bits of ADCON3, see [Register 23-1](#) for more details.

REGISTER 32-28: ADLTHH: ADC LOWER THRESHOLD HIGH BYTE REGISTER

| | | | | | | | |
|-------------|---------|---------|---------|---------|---------|---------|---------|
| R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x |
| ADLTH<15:8> | | | | | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-0 **ADLTH<15:8>**: ADC Lower Threshold MSB. ADLTH and ADUTH are compared with ADERR to set the ADUTHR and ADLTHR bits of ADSTAT. Depending on the setting of ADTMD, an interrupt may be triggered by the results of this comparison.

REGISTER 32-29: ADLTHL: ADC LOWER THRESHOLD LOW BYTE REGISTER

| | | | | | | | |
|------------|---------|---------|---------|---------|---------|---------|---------|
| R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x |
| ADLTH<7:0> | | | | | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-0 **ADLTH<7:0>**: ADC Lower Threshold LSB. ADLTH and ADUTH are compared with ADERR to set the ADUTHR and ADLTHR bits of ADSTAT. Depending on the setting of ADTMD, an interrupt may be triggered by the results of this comparison.

REGISTER 32-30: ADUTHH: ADC UPPER THRESHOLD HIGH BYTE REGISTER

| | | | | | | | |
|-------------|---------|---------|---------|---------|---------|---------|---------|
| R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x |
| ADUTH<15:8> | | | | | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-0 **ADUTH<15:8>**: ADC Upper Threshold MSB. ADLTH and ADUTH are compared with ADERR to set the ADUTHR and ADLTHR bits of ADSTAT. Depending on the setting of ADTMD, an interrupt may be triggered by the results of this comparison.

REGISTER 32-31: ADUTHL: ADC UPPER THRESHOLD LOW BYTE REGISTER

| | | | | | | | |
|------------|---------|---------|---------|---------|---------|---------|---------|
| R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x | R/W-x/x |
| ADUTH<7:0> | | | | | | | |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|----------------------|----------------------|---|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| u = Bit is unchanged | x = Bit is unknown | -n/n = Value at POR and BOR/Value at all other Resets |
| '1' = Bit is set | '0' = Bit is cleared | |

bit 7-0 **ADUTH<7:0>**: ADC Upper Threshold LSB. ADLTH and ADUTH are compared with ADERR to set the ADUTHR and ADLTHR bits of ADSTAT. Depending on the setting of ADTMD, an interrupt may be triggered by the results of this comparison.

REGISTER 32-32: ADOACT: ADC AUTO CONVERSION TRIGGER CONTROL REGISTER

| | | | | | | | | |
|-------|-----|-----|------------|---------|---------|---------|---------|-------|
| U-0 | U-0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | R/W-0/0 | |
| — | — | — | ADACT<4:0> | | | | | |
| bit 7 | | | | | | | | bit 0 |

Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

-n/n = Value at POR and BOR/Value at all other Resets

'1' = Bit is set

'0' = Bit is cleared

bit 7-5

Unimplemented: Read as '0'

bit 4-0

ADACT<4:0>: Auto-Conversion Trigger Select Bits

11111 = Software write to ADPCH

11110 = Reserved, do not use

11101 = Software read of ADRESH

11100 = Software read of ADERRH

11011 = Reserved, do not use

•

•

•

11000 = Reserved, do not use

10111 = Interrupt-on-change Interrupt Flag

10110 = C3_out

10101 = C2_out

10100 = C1_out

10011 = PWM7_out

10010 = PWM6_out

10001 = CCP5_out

10000 = CCP4_out

01111 = CCP3_out

01110 = CCP2_out

01101 = CCP1_out

01100 = SMT2_overflow

01011 = SMT1_overflow

01010 = TMR8_postscaled

01001 = TMR7_overflow

01000 = TMR6_postscaled

00111 = TMR5_overflow

00110 = TMR4_postscaled

00101 = TMR3_overflow

00100 = TMR2_postscaled

00011 = TMR1_overflow

00010 = TMR0_overflow

00001 = Pin selected by ADACTPPS

00000 = External Trigger Disabled

PIC18(L)F67K40

TABLE 32-5: SUMMARY OF REGISTERS ASSOCIATED WITH ADC

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on Page |
|----------|--------------|-------------|------------|------------|-------------|------------|-------------|---------|------------------|
| INTCON | GIE/GIEH | PEIE/GIEL | IPEN | — | INT3EDG | INT2EDG | INT1EDG | INT0EDG | 172 |
| PIE1 | OSCFIE | CSWIE | — | — | — | — | ADTIE | ADIE | 185 |
| PIR1 | OSCFIF | CSWIF | — | — | — | — | ADTIF | ADIF | 174 |
| ADCON0 | ADON | ADCON | — | ADCS | — | ADFM | — | ADGO | 499 |
| ADCON1 | ADPPOL | ADIPEN | ADGPOL | — | — | — | — | ADDSSEN | 500 |
| ADCON2 | ADPSIS | ADCRS<2:0> | | | ADACL | ADMD<2:0> | | | 501 |
| ADCON3 | — | ADCALC<2:0> | | | ADSOI | ADTMD<2:0> | | | 502 |
| ADACT | — | — | — | — | ADACT<4:0> | | | | 501 |
| ADRESH | ADRESH<7:0> | | | | | | | | 509, 509 |
| ADRESL | ADRESL<7:0> | | | | | | | | 509, 510 |
| ADPREVH | ADPREV<15:8> | | | | | | | | 510 |
| ADPREVL | ADPREV<7:0> | | | | | | | | 511 |
| ADACCH | ADACC<15:8> | | | | | | | | 511 |
| ADACCL | ADACC<7:0> | | | | | | | | 511 |
| ADSTPTH | ADSTPT<15:8> | | | | | | | | 512 |
| ADSTPT | ADSTPT<7:0> | | | | | | | | 512 |
| ADERRL | ADERR<7:0> | | | | | | | | 513 |
| ADLTHH | ADLTH<15:8> | | | | | | | | 513 |
| ADLTHL | ADLTH<7:0> | | | | | | | | 513 |
| ADUTHH | ADUTH<15:8> | | | | | | | | 514 |
| ADUTHL | ADUTH<7:0> | | | | | | | | 514 |
| ADSTAT | ADAOV | ADUTHR | ADLTHR | ADMATH | ADSTAT<3:0> | | | | 503 |
| ADCLK | — | — | ADCS<5:0> | | | | | | 504 |
| ADREF | — | — | — | ADNREF | — | — | ADPREF<1:0> | | 504 |
| ADPCH | — | — | ADPCH<5:0> | | | | | | 505 |
| ADPRE | ADPRE<7:0> | | | | | | | | 506 |
| ADACQ | ADACQ<7:0> | | | | | | | | 506 |
| ADCAP | — | — | — | ADCAP<4:0> | | | | | 507 |
| ADRPT | ADRPT<7:0> | | | | | | | | 507 |
| ADCNT | ADCNT<7:0> | | | | | | | | 508 |
| ADFLTRH | ADFLTR<15:8> | | | | | | | | 508 |
| ADFLTRL | ADFLTR<7:0> | | | | | | | | 508 |
| FVRCON | FVREN | FVRRDY | TSEN | TSRNG | CDAFVR<1:0> | | ADFVR<1:0> | | 474 |
| DAC1CON1 | — | — | — | DAC1R<4:0> | | | | | 480 |
| OSCSTAT | EXTOR | HFOR | MFOR | LFOR | SOR | ADOR | — | PLL | 37 |

Legend: — = unimplemented read as '0'. Shaded cells are not used for the ADC module.

33.0 COMPARATOR MODULE (C1/2/3)

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 PIC18(L)F6xK40 family has three comparators (C1/2/3).

The analog comparator module includes the following features:

- Programmable input selection
- Programmable output polarity
- Rising/falling output edge interrupts
- Wake-up from Sleep
- CWG Auto-shutdown source
- Selectable voltage reference
- ADC Auto-trigger
- TMR1/3/5/7 Gate
- TMR2/4/6/8 Reset
- CCP Capture Mode Input
- DSM Modulator Source
- Input and Window signal to Signal Measurement Timer

33.1 Comparator Overview

A single comparator is shown in [Figure 33-1](#) 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 33-1: SINGLE COMPARATOR

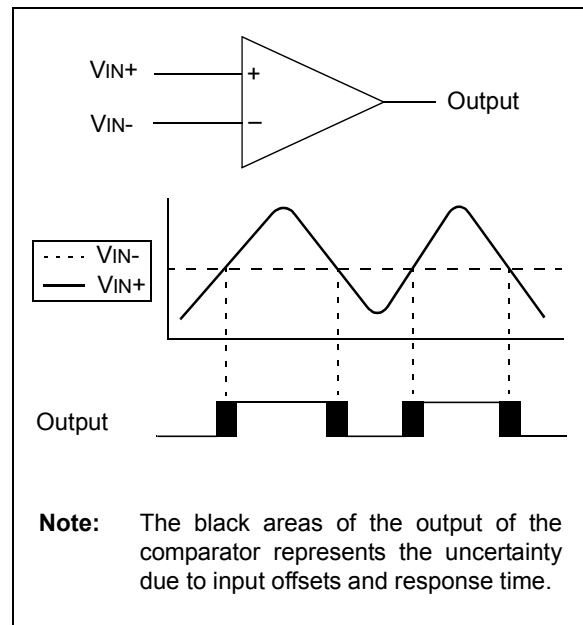
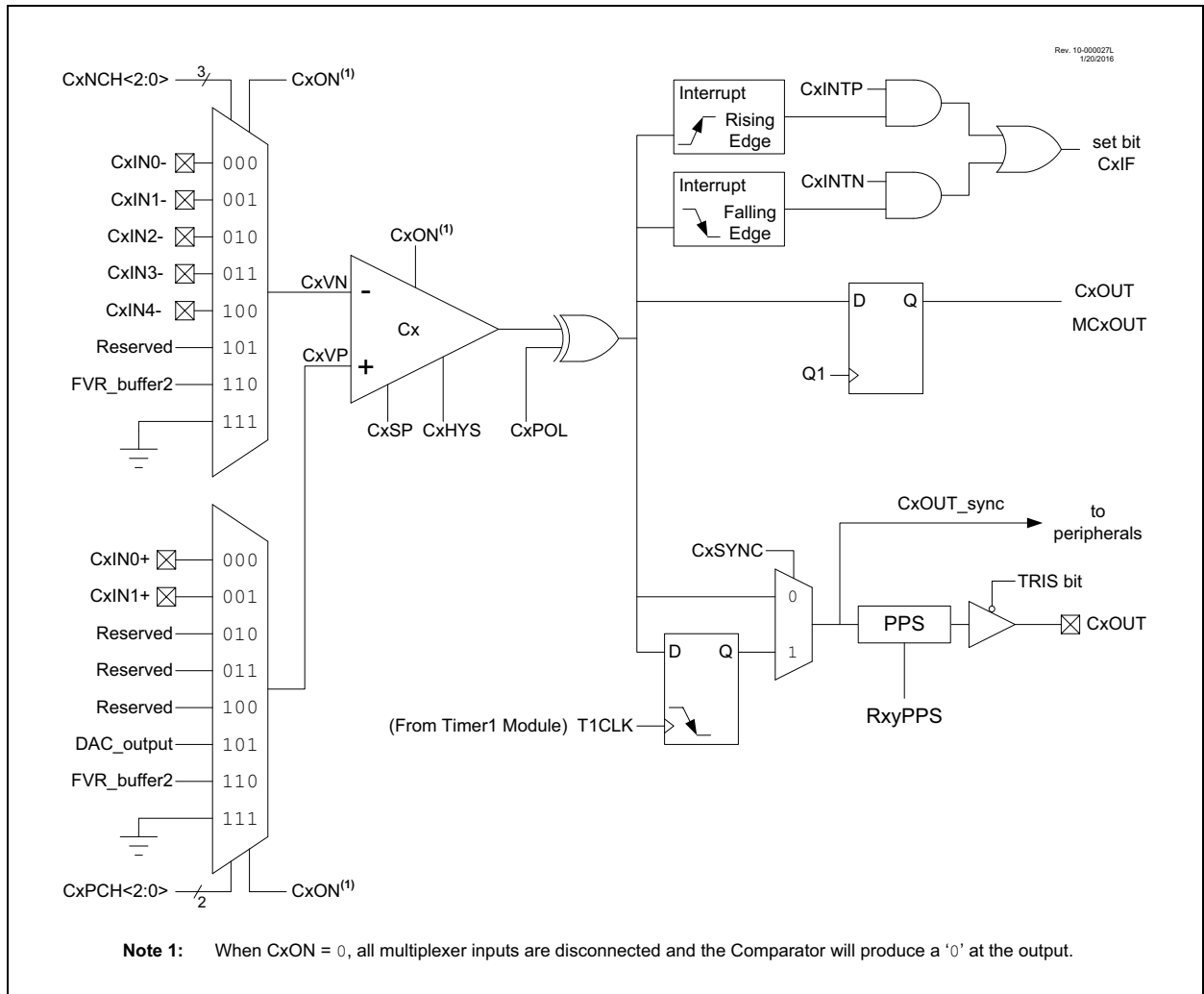


FIGURE 33-2: COMPARATOR MODULE SIMPLIFIED BLOCK DIAGRAM



33.2 Register Definitions: Comparator Control

Long bit name prefixes for the Comparators are shown in Table 33-1. Refer to Section 1.4.2.2 “Long Bit Names” for more information.

TABLE 33-1:

| Peripheral | Bit Name Prefix |
|------------|-----------------|
| C1 | C1 |
| C2 | C2 |
| C3 | C3 |

REGISTER 33-1: CMxCON0: COMPARATOR x CONTROL REGISTER 0

| | | | | | | | |
|---------|-------|-----|---------|-----|-----|---------|---------|
| R/W-0/0 | R-0/0 | U-0 | R/W-0/0 | U-0 | U-1 | R/W-0/0 | R/W-0/0 |
| EN | OUT | — | POL | — | — | HYS | SYNC |
| bit 7 | | | | | | | bit 0 |

Legend:

| | | |
|-------------------|------------------|------------------------------------|
| R = Readable bit | W = Writable bit | U = Unimplemented bit, read as '0' |
| -n = Value at POR | '1' = Bit is set | '0' = Bit is cleared |
| | | x = Bit is unknown |

- bit 7 **EN:** Comparator Enable bit
 1 = Comparator is enabled
 0 = Comparator is disabled and consumes no active power

- bit 6 **OUT:** Comparator Output bit
 If POL = 0 (non-inverted polarity):
 1 = CxVP > CxVN
 0 = CxVP < CxVN
 If POL = 1 (inverted polarity):
 1 = CxVP < CxVN
 0 = CxVP > CxVN

- bit 5 **Unimplemented:** Read as '0'

- bit 4 **POL:** Comparator Output Polarity Select bit
 1 = Comparator output is inverted
 0 = Comparator output is not inverted

- bit 3 **Unimplemented:** Read as '0'

- bit 2 **Unimplemented:** Read as '1'

- bit 1 **HYS:** Comparator Hysteresis Enable bit
 1 = Comparator hysteresis enabled
 0 = Comparator hysteresis disabled

- bit 0 **SYNC:** Comparator Output Synchronous Mode bit
 1 = Comparator output to Timer1/3/5/7 and I/O pin is synchronous to changes on Timer1 clock source.
 0 = Comparator output to Timer1/3/5/7 and I/O pin is asynchronous
 Output updated on the falling edge of Timer1/3/5/7 clock source.

REGISTER 33-2: CMxCON1: COMPARATOR x CONTROL REGISTER 1

| | | | | | | | |
|-------|-----|-----|-----|-----|-----|---------|---------|
| U-0 | U-0 | U-0 | U-0 | U-0 | U-0 | R/W-0/0 | R/W-0/0 |
| — | — | — | — | — | — | INTP | INTN |
| bit 7 | | | | | | bit 0 | |

Legend:

R = Readable bit W = Writable bit U = Unimplemented bit, read as '0'
 -n = Value at POR '1' = Bit is set '0' = Bit is cleared x = Bit is unknown

bit 7-2 Unimplemented: Read as '0'

bit 1 **INTP**: Comparator Interrupt on Positive-Going Edge Enable bit

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 bit

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

REGISTER 33-3: CMxNCH: COMPARATOR x INVERTING CHANNEL SELECT REGISTER

| | | | | | | | |
|-------|-----|-----|-----|-----|----------|---------|---------|
| U-0 | U-0 | U-0 | U-0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| — | — | — | — | — | NCH<2:0> | | |
| bit 7 | | | | | | bit 0 | |

Legend:

R = Readable bit W = Writable bit U = Unimplemented bit, read as '0'
 -n = Value at POR '1' = Bit is set '0' = Bit is cleared x = Bit is unknown

bit 7-3 Unimplemented: Read as '0'

bit 2-0 **NCH<2:0>**: Comparator Inverting Input Channel Select bits

| NCH | C1 Selection | C2 Selection | C3 Selection |
|-----|--------------|--------------|--------------|
| 111 | Vss | Vss | Vss |
| 110 | FVR buffer | FVR buffer | FVR buffer |
| 101 | N/C | N/C | N/C |
| 100 | C1IN4- | C2IN4- | C3IN4- |
| 011 | C1IN3- | C2IN3- | C3IN3- |
| 010 | C1IN2- | C2IN2- | C3IN2- |
| 001 | C1IN1- | C2IN1- | C3IN1- |
| 000 | C1IN0- | C2IN0- | C3IN0- |

REGISTER 33-4: CMxPCH: COMPARATOR x NON-INVERTING CHANNEL SELECT REGISTER

| | | | | | | | |
|-------|-----|-----|-----|-----|----------|---------|---------|
| U-0 | U-0 | U-0 | U-0 | U-0 | R/W-0/0 | R/W-0/0 | R/W-0/0 |
| — | — | — | — | — | PCH<2:0> | | |
| bit 7 | | | | | bit 0 | | |

Legend:

R = Readable bit W = Writable bit U = Unimplemented bit, read as '0'
-n = Value at POR '1' = Bit is set '0' = Bit is cleared x = Bit is unknown

bit 7-3 **Unimplemented:** Read as '0'
bit 2-0 **PCH<2:0>:** Comparator Non-Inverting Input Channel Select bits

| PCH | C1 Selection | C2 Selection | C3 Selection |
|-----|--------------|--------------|--------------|
| 111 | AVss | AVss | AVss |
| 110 | FVR2 | FVR2 | FVR2 |
| 101 | DAC1 | DAC1 | DAC1 |
| 100 | N/C | N/C | N/C |
| 011 | N/C | N/C | N/C |
| 010 | N/C | N/C | N/C |
| 001 | C1IN1+ | C2IN1+ | C3IN1+ |
| 000 | C1IN0+ | C2IN0+ | C3IN0+ |

REGISTER 33-5: CMOUT: COMPARATOR OUTPUT REGISTER

| | | | | | | | |
|-------|-----|-----|-----|-----|--------|--------|--------|
| U-0 | U-0 | U-0 | U-0 | U-0 | R-0/0 | R-0/0 | R-0/0 |
| — | — | — | — | — | MC3OUT | MC2OUT | MC1OUT |
| bit 7 | | | | | bit 0 | | |

Legend:

R = Readable bit W = Writable bit U = Unimplemented bit, read as '0'
-n = Value at POR '1' = Bit is set '0' = Bit is cleared x = Bit is unknown

bit 7-3 **Unimplemented:** Read as '0'
bit 2 **MC3OUT:** Mirror copy of C3OUT bit
bit 1 **MC2OUT:** Mirror copy of C2OUT bit
bit 0 **MC1OUT:** Mirror copy of C1OUT bit

33.3 Comparator Control

Each comparator has two control registers: CMxCON0 and CMxCON1.

The CMxCON0 register (see [Register 33-1](#)) contains Control and Status bits for the following:

- Enable
- Output
- Output polarity
- Hysteresis enable
- Timer1 output synchronization

The CMxCON1 register (see [Register 33-2](#)) contains Control bits for the following:

- Interrupt on positive/negative edge enables
- Positive input channel selection
- Negative input channel selection

33.3.1 COMPARATOR ENABLE

Setting the EN bit of the CMxCON0 register enables the comparator for operation. Clearing the CxEN bit disables the comparator resulting in minimum current consumption.

33.3.2 COMPARATOR OUTPUT

The output of the comparator can be monitored by reading either the CxOUT bit of the CMxCON0 register or the MCxOUT bit of the CMOUT register.

The comparator output can also be routed to an external pin through the RxyPPS register ([Register 17-2](#)). The corresponding TRIS bit must be clear to enable the pin as an output.

Note 1: The internal output of the comparator is latched with each instruction cycle. Unless otherwise specified, external outputs are not latched.

33.3.3 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 CxPOL bit of the CMxCON0 register. Clearing the CxPOL bit results in a non-inverted output.

[Table 33-2](#) shows the output state versus input conditions, including polarity control.

TABLE 33-2: COMPARATOR OUTPUT STATE VS. INPUT CONDITIONS

| Input Condition | CxPOL | CxOUT |
|-----------------|-------|-------|
| $CxVN > CxVP$ | 0 | 0 |
| $CxVN < CxVP$ | 0 | 1 |
| $CxVN > CxVP$ | 1 | 1 |
| $CxVN < CxVP$ | 1 | 0 |

33.4 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 CxHYS bit of the CMxCON0 register.

See Comparator Specifications in [Table 37-15](#) for more information.

33.5 Timer1/3/5/7 Gate Operation

The output resulting from a comparator operation can be used as a source for gate control of Timer1/3/5/7. See [Section 19.8 “Timer1/3/5/7 Gate”](#) for more information. This feature is useful for timing the duration or interval of an analog event.

It is recommended that the comparator output be synchronized to Timer1. This ensures that Timer1 does not increment while a change in the comparator is occurring.

33.5.1 COMPARATOR OUTPUT SYNCHRONIZATION

The output from a comparator can be synchronized with Timer1 by setting the SYNC bit of the CMxCON0 register.

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. See the Comparator Block Diagram ([Figure 33-2](#)) and the Timer1 Block Diagram ([Figure 19-1](#)) for more information.

33.6 Comparator Interrupt

An interrupt can be generated upon a change in the output value of the comparator for each comparator, a rising edge detector and a falling edge detector are present.

When either edge detector is triggered and its associated enable bit is set (CxINTP and/or CxINTN bits of the CMxCON1 register), the Corresponding Interrupt Flag bit (CxIF bit of the PIR2 register) will be set.

To enable the interrupt, you must set the following bits:

- EN and POL bits of the CMxCON0 register
- CxIE bit of the PIE2 register
- INTP bit of the CMxCON1 register (for a rising edge detection)
- INTN bit of the CMxCON1 register (for a falling edge detection)
- PEIE and GIE bits of the INTCON register

The associated interrupt flag bit, CxIF bit of the PIR2 register, must be cleared in software. If another edge is detected while this flag is being cleared, the flag will still be set at the end of the sequence.

Note: Although a comparator is disabled, an interrupt can be generated by changing the output polarity with the CxPOL bit of the CMxCON0 register, or by switching the comparator on or off with the CxEN bit of the CMxCON0 register.

33.7 Comparator Positive Input Selection

Configuring the PCH<2:0> bits of the CMxPCH register directs an internal voltage reference or an analog pin to the non-inverting input of the comparator:

- CxIN0+, CxIN1+ analog pin
- DAC output
- FVR (Fixed Voltage Reference)
- AVss (Ground)

See [Section 29.0 “Fixed Voltage Reference \(FVR\)”](#) for more information on the Fixed Voltage Reference module.

See [Section 31.0 “5-Bit Digital-to-Analog Converter \(DAC\) Module”](#) for more information on the DAC input signal.

Any time the comparator is disabled (CxEN = 0), all comparator inputs are disabled.

33.8 Comparator Negative Input Selection

The NCH<2:0> bits of the CMxNCH register direct an analog input pin and internal reference voltage or analog ground to the inverting input of the comparator:

- CxIN0-, CxIN1-, CxIN2-, CxIN3-, CxIN4- analog pin
- FVR (Fixed Voltage Reference)
- Analog Ground

Note: 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.

33.9 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 [Table 37-15](#) and [Table 37-17](#) for more details.

33.10 Analog Input Connection Considerations

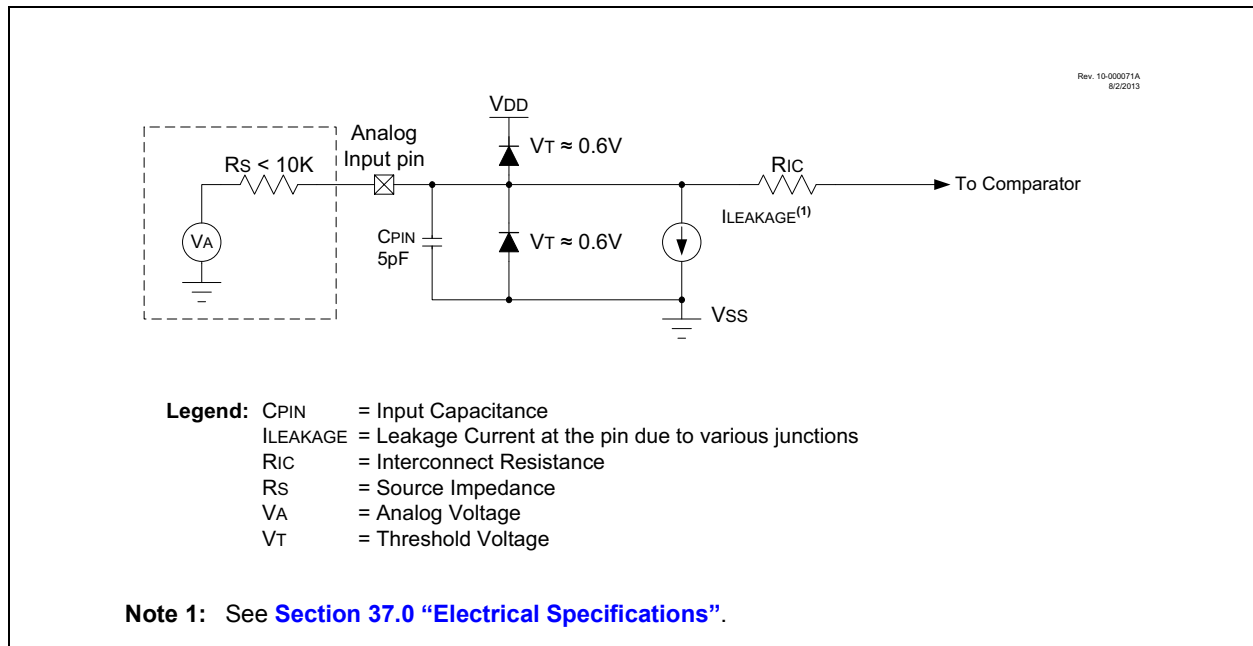
A simplified circuit for an analog input is shown in [Figure 33-3](#). Since the analog input pins share their connection with a digital input, they have reverse biased ESD protection diodes to V_{DD} and V_{SS} . The analog input, therefore, must be between V_{SS} and V_{DD} . If the input voltage deviates from this range by more than 0.6V in either direction, one of the diodes is forward biased and a latch-up may occur.

A maximum source impedance of 10 k Ω is recommended for the analog sources. Also, any external component connected to an analog input pin, such as a capacitor or a Zener diode, should have very little leakage current to minimize inaccuracies introduced.

Note 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 is specified.

FIGURE 33-3: ANALOG INPUT MODEL



33.11 CWG1 Auto-Shutdown Source

The output of the comparator module can be used as an auto-shutdown source for the CWG1 module. When the output of the comparator is active and the corresponding WGASxE is enabled, the CWG operation will be suspended immediately (see [Section 24.10.1.2 “External Input Source”](#)).

33.12 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.

33.13 TMR2/4/6/8 Reset

The output of the comparator module can be used to reset Timer2. When the TxERS register is appropriately set, the timer will reset when the Comparator output goes high.

33.14 Operation in Sleep Mode

The comparator module can operate during Sleep. The comparator clock source is based on the Timer1 clock source. If the Timer1 clock source is either the system clock (FOSC) or the instruction clock (FOSC/4), Timer1 will not operate during Sleep, and synchronized comparator outputs will not operate.

A comparator interrupt will wake the device from Sleep. The CxIE bits of the PIE2 register must be set to enable comparator interrupts.

TABLE 33-3: SUMMARY OF REGISTERS ASSOCIATED WITH COMPARATOR MODULE

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Reset Values on page |
|---------|----------|-----------|-------------|-------|-------------|----------|------------|---------|----------------------|
| CMxCON0 | EN | OUT | — | POL | — | — | HYS | SYNC | 519 |
| CMxCON1 | — | — | — | — | — | — | CxINTP | CxINTN | 520 |
| CMxNCH | — | — | — | — | — | NCH<2:0> | | | 520 |
| CMxPCH | — | — | — | — | — | PCH<2:0> | | | 521 |
| CMOUT | — | — | — | — | — | MC3OUT | MC2OUT | MC1OUT | 521 |
| FVRCON | FVREN | FVRRDY | TSEN | TSRNG | CDAFVR<1:0> | | ADFVR<1:0> | | 474 |
| INTCON | GIE/GIEH | PEIE/GIEL | IPEN | — | INT3EDG | INT2EDG | INT1EDG | INT0EDG | 172 |
| PIR2 | HLVDIF | ZCDIF | — | — | — | C3IF | C2IF | C1IF | 175 |
| PIE2 | HLVDIE | ZCDIE | — | — | — | C3IE | C2IE | C1IE | 186 |
| IPR2 | HLVDIP | ZCDIP | — | — | — | C3IP | C2IP | C1IP | 196 |
| PMD3 | — | DACMD | ADCMD | — | CMP3MD | CMP2MD | CMP1MD | ZCDMD | 68 |
| RxyPPS | — | — | RxyPPS<5:0> | | | | | | 227 |

Legend: — = unimplemented, read as '0'. Shaded cells are unused by the comparator module.

34.0 HIGH/LOW-VOLTAGE DETECT (HLVD)

The PIC18(L)F6xK40 of devices has a High/Low-Voltage Detect module (HLVD). This is a programmable circuit that sets both a device voltage trip point and the direction of change from that point (positive going, negative going or both). If the device experiences an excursion past the trip point in that direction, an interrupt flag is set. If the interrupt is enabled, the program execution branches to the interrupt vector address and the software responds to the interrupt.

Complete control of the HLVD module is provided through the HLVDCON0 and HLVDCON1 register. This allows the circuitry to be “turned off” by the user under software control, which minimizes the current consumption for the device.

The module’s block diagram is shown in [Figure 34-1](#).

Since the HLVD can be software enabled through the HLVDEN bit, setting and clearing the enable bit does not produce a false HLVD event glitch. Each time the HLVD module is enabled, the circuitry requires some time to stabilize. The RDY bit (HLVDCON0<4>) is a read-only bit used to indicate when the band gap reference voltages are stable.

The module can only generate an interrupt after the module is turned ON and the band gap reference voltages are ready.

The HLVDINTH and HLVDINTL bits determine the overall operation of the module. When HLVDINTH is set, the module monitors for rises in VDD above the trip point set by the HLVDCON1 register. When HLVDINTL is set, the module monitors for drops in VDD below the trip point set by the HLVDCON1 register. When both the HLVDINTH and HLVDINTL bits are set, any changes above or below the trip point set by the HLVDCON1 register can be monitored.

The OUT bit can be read to determine if the voltage is greater than or less than the voltage level selected by the HLVDCON1 register.

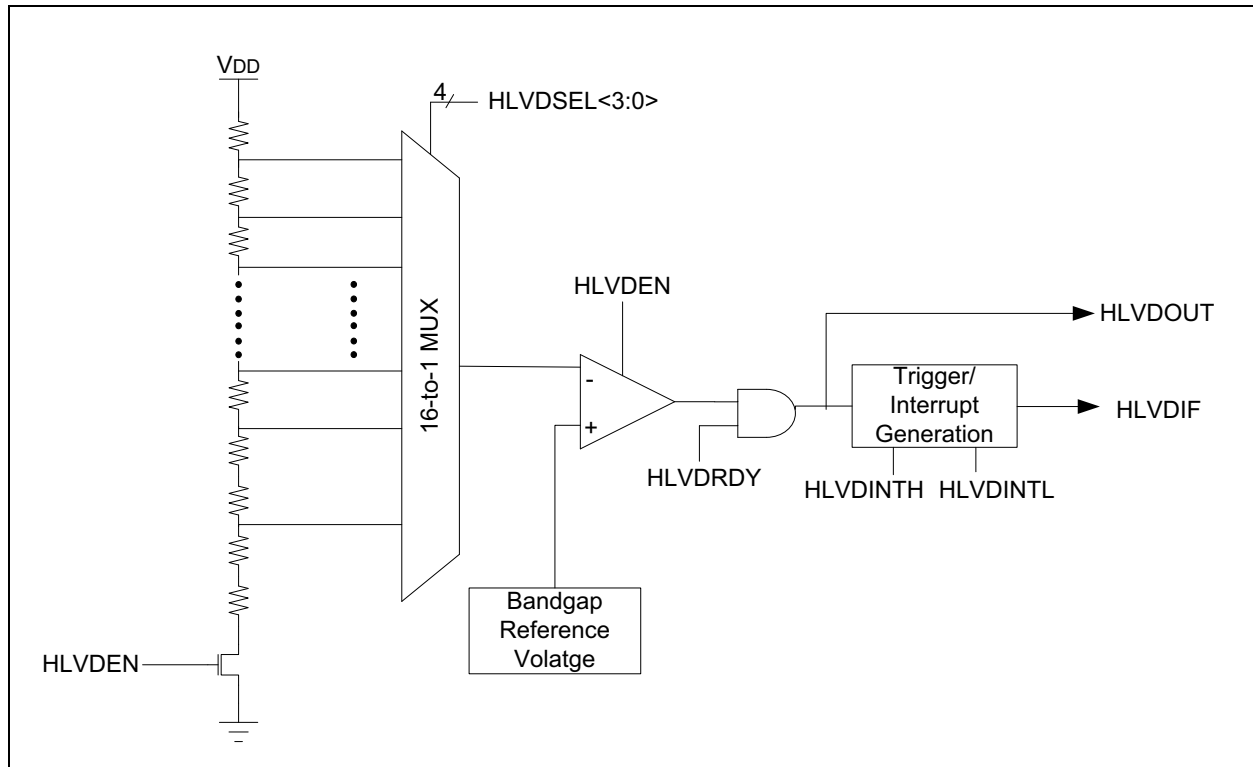
34.1 Operation

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 to any of 16 values. The trip point is selected by programming the HLVDSEL<3:0> bits (HLVDCON1<3:0>).

FIGURE 34-1: HLVD MODULE BLOCK DIAGRAM



34.2 HLVD Setup

To set up the HLVD module:

1. Select the desired HLVD trip point by writing the value to the HLVDSEL<3:0> bits of the HLVDCON1 register.
2. Depending on the application to detect high-voltage peaks or low-voltage drops or both, set the HLVDINTH or HLVDINTL bit appropriately.
3. Enable the HLVD module by setting the HLVDEN bit.
4. Clear the HLVD interrupt flag (PIR2 register), which may have been set from a previous interrupt.
5. If interrupts are desired, enable the HLVD interrupt by setting the HLVDIE in the PIE2 register and GIE bits.

An interrupt will not be generated until the HLVDRDY bit is set.

Note: Before changing any module settings (HLVDINTH, HLVDINTL, HLVDSEL<3:0>), first disable the module (HLVDEN = 0), make the changes and re-enable the module. This prevents the generation of false HLVD events.

34.3 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 electrical specification Parameter **D206** (Table 37-3Table 38-3).

Depending on the application, the HLVD module does not need to operate constantly. To reduce current requirements, the HLVD circuitry may only need to be enabled for short periods where the voltage is checked. After such a check, the module could be disabled.

34.4 HLVD Start-up Time

The internal reference voltage of the HLVD module, specified in electrical specification (Table 37-17), may be used by other internal circuitry, such as the programmable Brown-out Reset. If the HLVD or other circuits using the 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_{FVRS} , is an interval that is independent of device clock speed. It is specified in electrical specification (Table 37-17).

The HLVD interrupt flag is not enabled until T_{FVRS} 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 Figure 34-2 or Figure 34-3).

FIGURE 34-2: LOW-VOLTAGE DETECT OPERATION (HLVDINTL = 1)

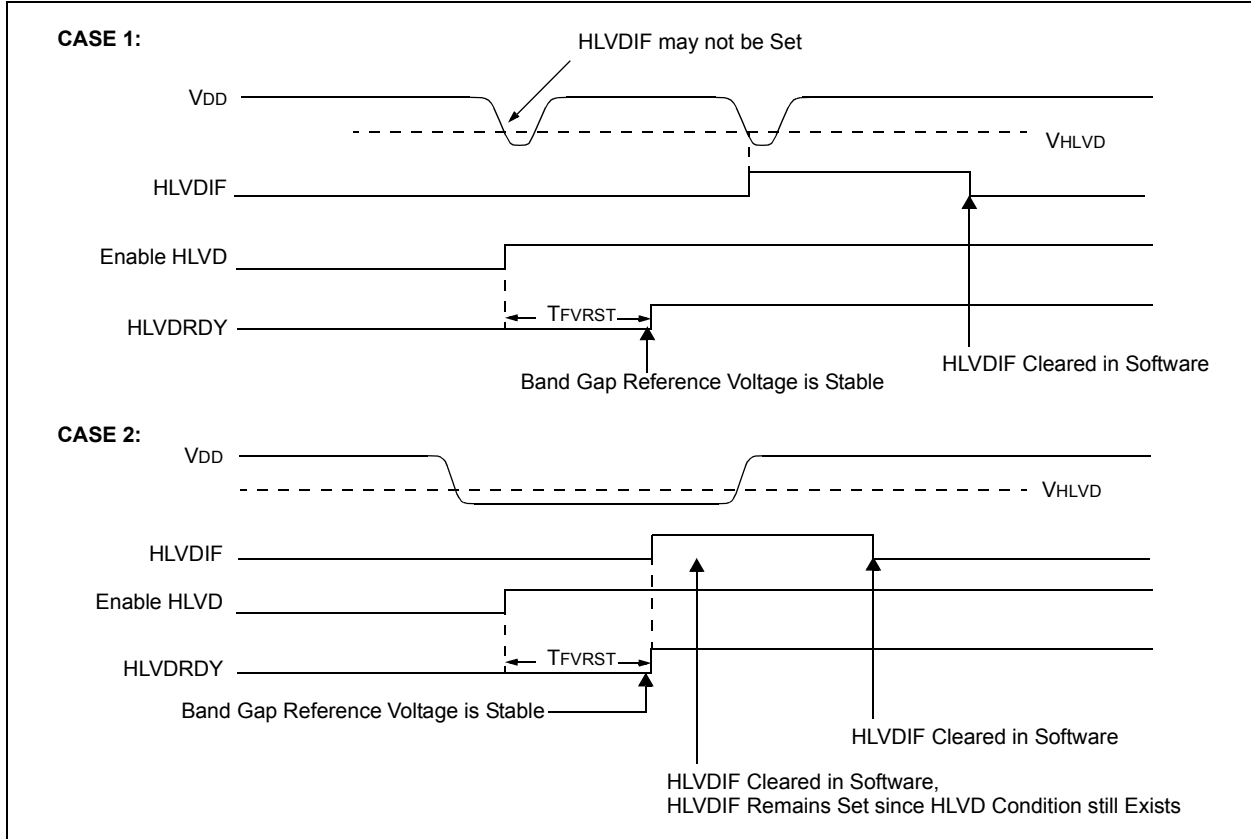
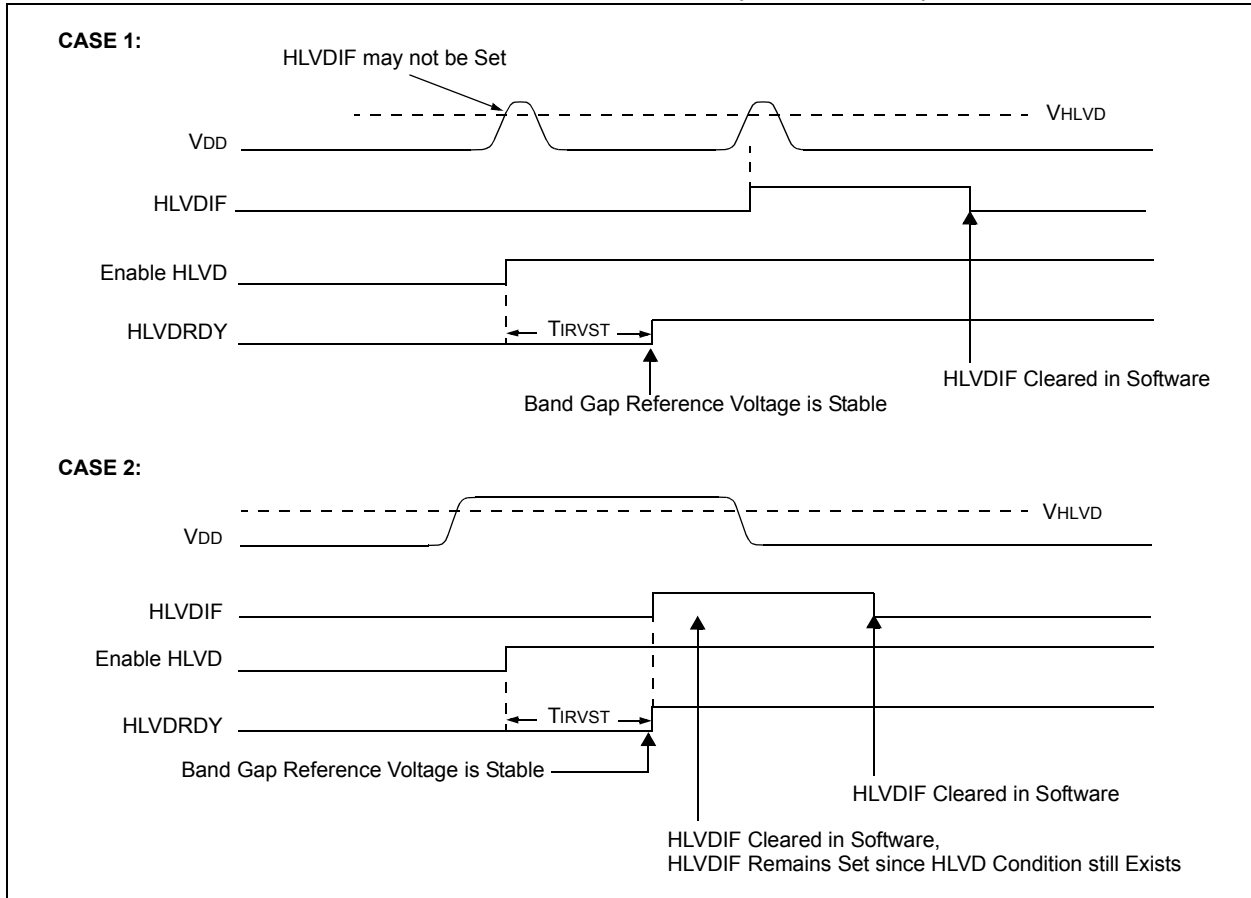


FIGURE 34-3: HIGH-VOLTAGE DETECT OPERATION (HLVDINTH = 1)

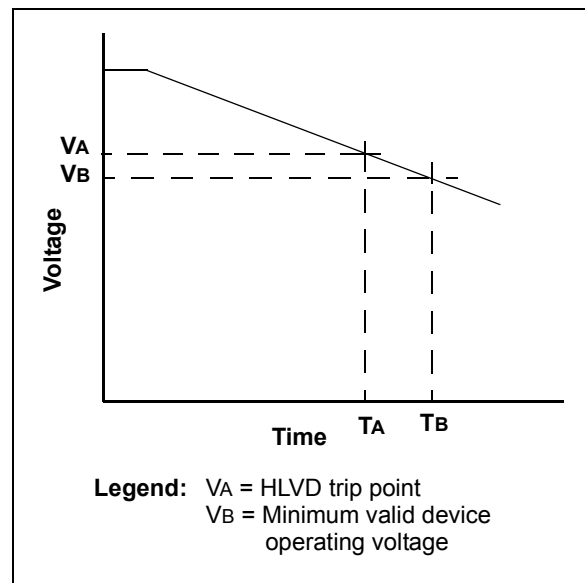


34.5 Applications

In many applications, it is desirable to detect a drop below, or rise above, a particular voltage threshold. For example, the HLVD module could 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 would indicate a High-Voltage Detect from, for example, 3.3V to 5V (the voltage on USB) and vice versa for a detach. This feature could save a design a few extra components and an attach signal (input pin).

For general battery applications, Figure 34-4 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 could cause the execution of an Interrupt Service Routine (ISR), which would allow the application to perform "housekeeping tasks" and a controlled shutdown before the device voltage exits the valid operating range at T_B. This would give the application a time window, represented by the difference between T_A and T_B, to safely exit.

FIGURE 34-4: TYPICAL LOW-VOLTAGE DETECT APPLICATION



34.6 Operation During Sleep

When enabled, the HLVD circuitry continues to operate during Sleep. If the device voltage crosses the trip point, the HLVDIF bit will be set and the device will wake up from Sleep. Device execution will continue from the interrupt vector address if interrupts have been globally enabled.

34.7 Operation During Idle and Doze Modes

In both Idle and Doze modes, the module is active and events are generated if peripheral is enabled.

34.8 Operation During Freeze

When in Freeze mode, no new event or interrupt can be generated. The state of the LVDRDY bit is frozen.

Register reads and writes through the CPU interface are allowed.

34.9 Effects of a Reset

A device Reset forces all registers to their Reset state. This forces the HLVD module to be turned off.

34.10 Register Definitions: HLVD Control

Long bit name prefixes for the HLVD peripheral is shown in [Table 34-1](#). Refer to [Section 1.4.2.2 “Long Bit Names”](#) for more information.

TABLE 34-1:

| Peripheral | Bit Name Prefix |
|------------|-----------------|
| HLVD | HLVD |

REGISTER 34-1: HLVDCON1: LOW-VOLTAGE DETECT CONTROL REGISTER 1

| U-0 | U-0 | U-0 | U-0 | R/W-0/u | R/W-0/u | R/W-0/u | R/W-0/u |
|-------|-----|-----|-----|----------|---------|---------|---------|
| — | — | — | — | SEL<3:0> | | | |
| bit 7 | | | | bit 0 | | | |

Legend:

R = Readable bit
-n = Value at POR

W = Writable bit
'1' = Bit is set

U = Unimplemented bit, read as '0'
'0' = Bit is cleared

u = Bit is unchanged

bit 7-4

Unimplemented: Read as '0'

bit 3-0

SEL<3:0>: High/Low Voltage Detection Limit Selection bits

| SEL<3:0> | Typical Voltage |
|----------|-----------------|
| 1111 | Reserved |
| 1110 | 4.63V |
| 1101 | 4.32V |
| 1100 | 4.12V |
| 1011 | 3.91V |
| 1010 | 3.71V |
| 1001 | 3.60V |
| 1000 | 3.4V |
| 0111 | 3.09V |
| 0110 | 2.88V |
| 0101 | 2.78V |
| 0100 | 2.57V |
| 0011 | 2.47V |
| 0010 | 2.26V |
| 0001 | 2.06V |
| 0000 | 1.85V |

REGISTER 34-2: HLVDCON0: HIGH/LOW-VOLTAGE DETECT CONTROL REGISTER 0

| | | | | | | | |
|---------|-----|-----|-----|-----|-----|---------|---------|
| R/W-0/0 | U-0 | R-x | R-x | U-0 | U-0 | R/W-0/0 | R/W-0/0 |
| EN | — | OUT | RDY | — | — | INTH | INTL |
| bit 7 | | | | | | | bit 0 |

Legend:

R = Readable bit W = Writable bit U = Unimplemented bit, read as '0'
 -n = Value at POR '1' = Bit is set '0' = Bit is cleared x = Bit is unknown

- bit 7 **EN:** High/Low-voltage Detect Power Enable bit
 1 = Enables HLVD, powers up HLVD circuit and supporting reference circuitry
 0 = Disables HLVD, powers down HLVD and supporting circuitry
- bit 6 **Unimplemented:** Read as '0'
- bit 5 **OUT:** HLVD Comparator Output bit
 1 = Voltage ≤ selected detection limit (HLVDL<3:0>)
 0 = Voltage ≥ selected detection limit (HLVDL<3:0>)
- bit 4 **RDY:** Band Gap Reference Voltages Stable Status Flag bit
 1 = Indicates HLVD Module is ready and output is stable
 0 = Indicates HLVD Module is not ready
- bit 3-2 **Unimplemented:** Read as '0'
- bit 1 **INTH:** HLVD Positive going (High Voltage) Interrupt Enable
 1 = HLVDIF will be set when voltage ≥ selected detection limit (HLVDSEL<3:0>)
 0 = HLVDIF will not be set
- bit 0 **INTL:** HLVD Negative going (Low Voltage) Interrupt Enable
 1 = HLVDIF will be set when voltage ≤ selected detection limit (HLVDSEL<3:0>)
 0 = HLVDIF will not be set

TABLE 34-2: REGISTERS ASSOCIATED WITH HIGH/LOW-VOLTAGE DETECT MODULE

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Register on Page |
|----------|----------|-----------|-------|-------|----------|---------|---------|---------|------------------|
| HLVDCON0 | EN | — | OUT | RDY | — | — | INTH | INTL | 533 |
| HLVDCON1 | — | — | — | — | SEL<3:0> | | | | 532 |
| INTCON | GIE/GIEH | PEIE/GIEL | IPEN | — | INT3EDG | INT2EDG | INT1EDG | INT0EDG | 172 |
| PIR2 | HLVDIF | ZCDIF | — | — | — | C3IF | C2IF | C1IF | 175 |
| PIE2 | HLVDIE | ZCDIE | — | — | — | C3IE | C2IE | C1IE | 186 |
| IPR2 | HLVDIP | ZCDIP | — | — | — | C3IP | C2IP | C1IP | 196 |
| PMD0 | SYSCMD | FVRMD | HLVDM | CRCMD | SCANMD | NVMMD | CLKRMD | IOCMD | 65 |

Note 1: — = unimplemented, read as '0'. Shaded cells are unused by the HLVD module.

35.0 IN-CIRCUIT SERIAL PROGRAMMING™ (ICSP™)

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
- MCLR/VPP
- VDD
- VSS

In Program/Verify mode the program memory, User IDs and the Configuration Words 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 “PIC18(L)F67K40 Memory Programming Specification” (DS40001822).

35.1 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 MCLR/VPP to VIH.

35.2 Low-Voltage Programming Entry Mode

The Low-Voltage Programming Entry mode allows the PIC® Flash MCUs to be programmed using VDD only, without high voltage. When the LVP bit of Configuration Words 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. MCLR is brought to VIL.
2. A 32-bit key sequence is presented on ICSPDAT, while clocking ICSPCLK.

Once the key sequence is complete, MCLR must be held at VIL for as long as Program/Verify mode is to be maintained.

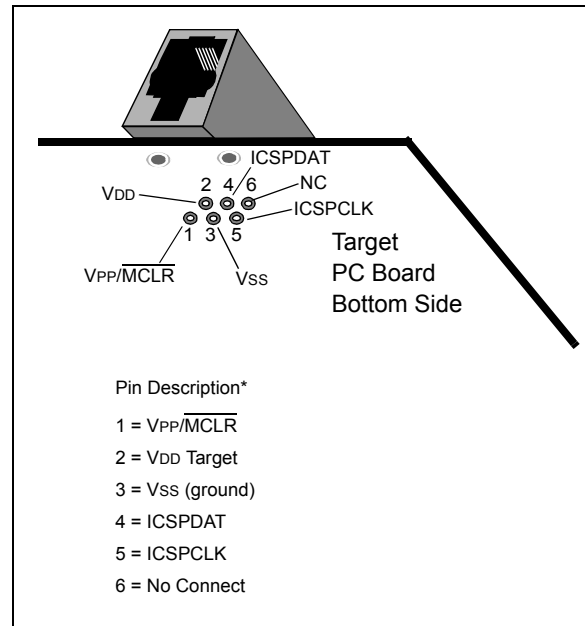
If low-voltage programming is enabled (LVP = 1), the MCLR Reset function is automatically enabled and cannot be disabled. See Section 8.6 “MCLR” for more information.

The LVP bit can only be reprogrammed to ‘0’ by using the High-Voltage Programming mode.

35.3 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 35-1.

FIGURE 35-1: ICD RJ-11 STYLE CONNECTOR INTERFACE



Another connector often found in use with the PICkit™ programmers is a standard 6-pin header with 0.1 inch spacing. Refer to Figure 35-2.

For additional interface recommendations, refer to your 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 35-3 for more information.

FIGURE 35-2: PICKIT™ PROGRAMMER STYLE CONNECTOR INTERFACE

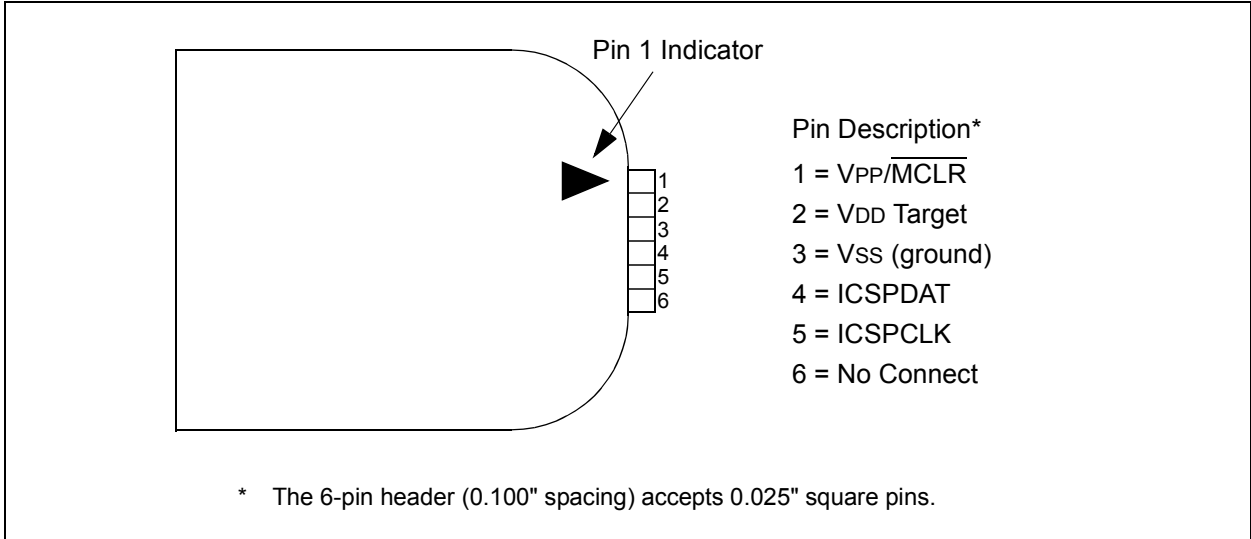
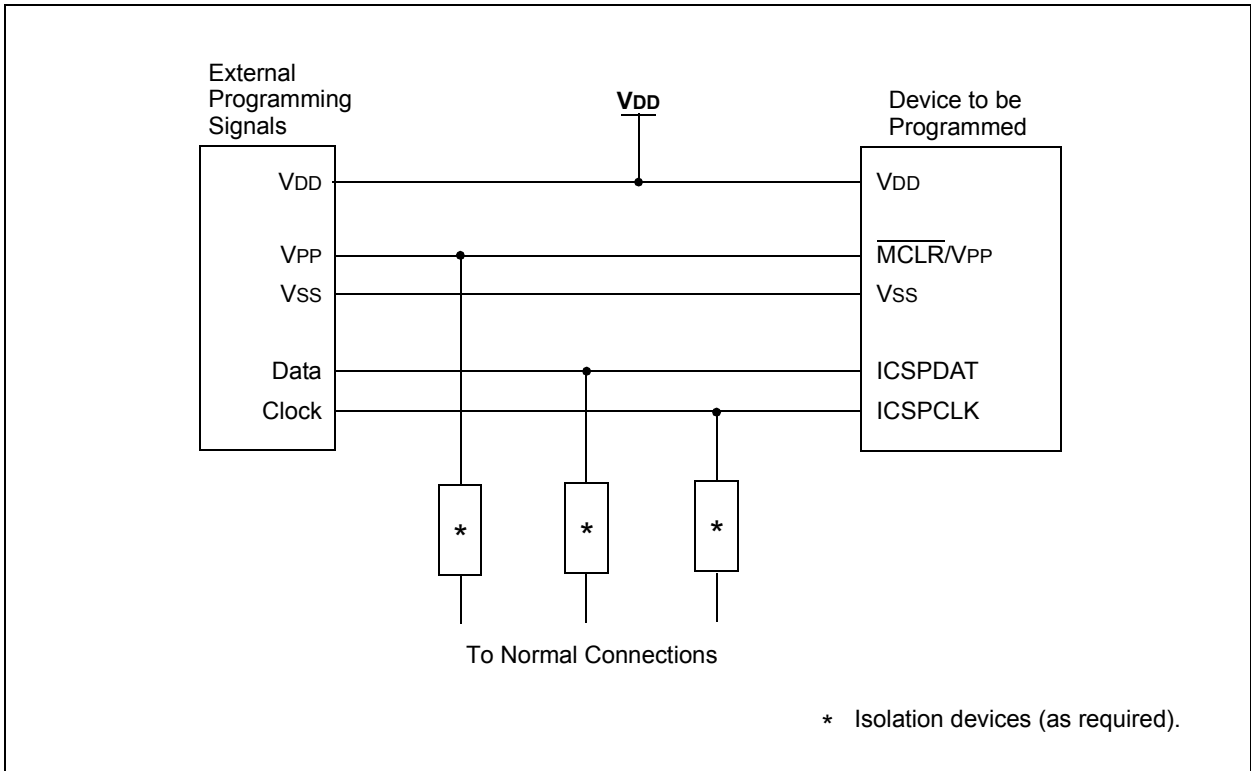


FIGURE 35-3: TYPICAL CONNECTION FOR ICSP™ PROGRAMMING



36.0 INSTRUCTION SET SUMMARY

PIC18(L)F67K40 devices incorporate the standard set of 75 PIC18 core instructions, as well as an extended set of eight new instructions, for the optimization of code that is recursive or that utilizes a software stack. The extended set is discussed later in this section.

36.1 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 four instructions that require two program memory locations.

Each single-word instruction is a 16-bit word divided into an opcode, which specifies the instruction type and one or more operands, which further specify 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 36-2](#) lists **byte-oriented**, **bit-oriented**, **literal** and **control** operations. [Table 36-1](#) shows the opcode field descriptions.

Most **byte-oriented** instructions have three operands:

1. The file register (specified by 'f')
2. The destination of the result (specified by 'd')
3. 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:

1. The file register (specified by 'f')
2. The bit in the file register (specified by 'b')
3. 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 four double-word instructions. These instructions were made double-word to contain the required information in 32 bits. In the second word, the four MSBs are '1's. If this second 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 double-word instructions execute in two 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 μ s. If a conditional test is true, or the program counter is changed as a result of an instruction, the instruction execution time is 2 μ s. Two-word branch instructions (if true) would take 3 μ s.

[Figure 36-1](#) shows 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 36-2](#), lists the standard instructions recognized by the Microchip Assembler (MPASM[™]).

[Section 36.1.1 "Standard Instruction Set"](#) provides a description of each instruction.

TABLE 36-1: OPCODE FIELD DESCRIPTIONS

| Field | Description |
|-----------------|---|
| a | RAM access bit a = 0: RAM location in Access RAM (BSR register is ignored) a = 1: RAM bank is specified by BSR register |
| bbb | Bit address within an 8-bit file register (0 to 7). |
| BSR | Bank Select Register. Used to select the current RAM bank. |
| C, DC, Z, OV, N | ALU Status bits: Carry, Digit Carry, Zero, Overflow, Negative. |
| d | Destination select bit d = 0: store result in WREG d = 1: store result in file register f |
| dest | Destination: either the WREG register or the specified register file location. |
| f | 8-bit Register file address (00h to FFh) or 2-bit FSR designator (0h to 3h). |
| f _s | 12-bit Register file address (000h to FFFh). This is the source address. |
| f _d | 12-bit Register file address (000h to FFFh). This is the destination address. |
| GIE | Global Interrupt Enable bit. |
| k | Literal field, constant data or label (may be either an 8-bit, 12-bit or a 20-bit value). |
| label | Label name. |
| mm | The mode of the TBLPTR register for the table read and table write instructions. Only used with table read and table write instructions: |
| * | No change to register (such as TBLPTR with table reads and writes) |
| ++ | Post-Increment register (such as TBLPTR with table reads and writes) |
| *- | Post-Decrement register (such as TBLPTR with table reads and writes) |
| ++* | Pre-Increment register (such as TBLPTR with table reads and writes) |
| n | The relative address (2's complement number) for relative branch instructions or the direct address for CALL/BRANCH and RETURN instructions. |
| 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. |
| PD | Power-down bit. |
| PRODH | Product of Multiply High Byte. |
| PRODL | Product of Multiply Low Byte. |
| s | Fast Call/Return mode select bit s = 0: do not update into/from shadow registers s = 1: certain registers loaded into/from shadow registers (Fast mode) |
| TBLPTR | 21-bit Table Pointer (points to a Program Memory location). |
| TABLAT | 8-bit Table Latch. |
| T _O | Time-out bit. |
| TOS | Top-of-Stack. |
| u | Unused or unchanged. |
| WDT | Watchdog Timer. |
| 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. |
| z _s | 7-bit offset value for indirect addressing of register files (source). |
| z _d | 7-bit offset value for indirect addressing of register files (destination). |
| { } | Optional argument. |
| [text] | Indicates an indexed address. |
| (text) | The contents of text. |
| [expr]<n> | Specifies bit n of the register indicated by the pointer expr. |
| → | Assigned to. |
| < > | Register bit field. |
| ∈ | In the set of. |
| <i>italics</i> | User defined term (font is Courier). |

FIGURE 36-1: General Format for Instructions

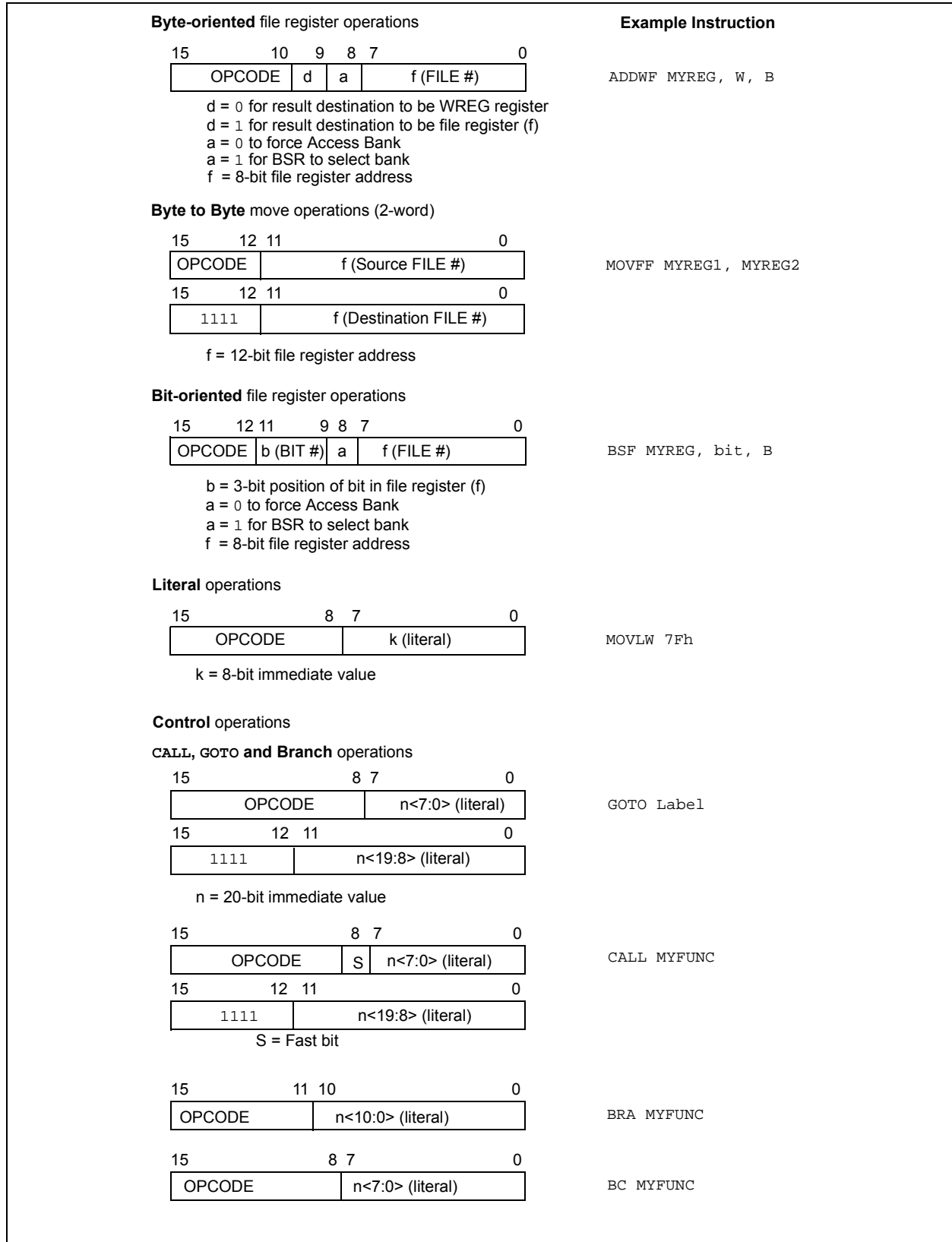


TABLE 36-2: INSTRUCTION SET

| Mnemonic, Operands | Description | Cycles | 16-Bit Instruction Word | | | | Status Affected | Notes | |
|---------------------------------|---------------------------------|---|-------------------------|------|------|------|--------------------|-----------------|------------|
| | | | MSb | LSb | | | | | |
| BYTE-ORIENTED OPERATIONS | | | | | | | | | |
| ADDWF | f, d, a | Add WREG and f | 1 | 0010 | 01da | ffff | ffff | C, DC, Z, OV, N | 1, 2 |
| ADDWFC | f, d, a | Add WREG and CARRY bit to f | 1 | 0010 | 00da | ffff | ffff | C, DC, Z, OV, N | 1, 2 |
| ANDWF | f, d, a | AND WREG with f | 1 | 0001 | 01da | ffff | ffff | Z, N | 1, 2 |
| CLRF | f, a | Clear f | 1 | 0110 | 101a | ffff | ffff | Z | 2 |
| COMF | f, d, a | Complement f | 1 | 0001 | 11da | ffff | ffff | Z, N | 1, 2 |
| CPFSEQ | f, a | Compare f with WREG, skip = | 1 (2 or 3) | 0110 | 001a | ffff | ffff | None | 4 |
| CPFSGT | f, a | Compare f with WREG, skip > | 1 (2 or 3) | 0110 | 010a | ffff | ffff | None | 4 |
| CPFSLT | f, a | Compare f with WREG, skip < | 1 (2 or 3) | 0110 | 000a | ffff | ffff | None | 1, 2 |
| DECf | f, d, a | Decrement f | 1 | 0000 | 01da | ffff | ffff | C, DC, Z, OV, N | 1, 2, 3, 4 |
| DECFSZ | f, d, a | Decrement f, Skip if 0 | 1 (2 or 3) | 0010 | 11da | ffff | ffff | None | 1, 2, 3, 4 |
| DCFSNZ | f, d, a | Decrement f, Skip if Not 0 | 1 (2 or 3) | 0100 | 11da | ffff | ffff | None | 1, 2 |
| INCF | f, d, a | Increment f | 1 | 0010 | 10da | ffff | ffff | C, DC, Z, OV, N | 1, 2, 3, 4 |
| INCFSZ | f, d, a | Increment f, Skip if 0 | 1 (2 or 3) | 0011 | 11da | ffff | ffff | None | 4 |
| INFSNZ | f, d, a | Increment f, Skip if Not 0 | 1 (2 or 3) | 0100 | 10da | ffff | ffff | None | 1, 2 |
| IORWF | f, d, a | Inclusive OR WREG with f | 1 | 0001 | 00da | ffff | ffff | Z, N | 1, 2 |
| MOVF | f, d, a | Move f | 1 | 0101 | 00da | ffff | ffff | Z, N | 1 |
| MOVFF | f _s , f _d | Move f _s (source) to 1st word f _d (destination) 2nd word | 2 | 1100 | ffff | ffff | ffff | None | |
| 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, 2 |
| NEGF | f, a | Negate f | 1 | 0110 | 110a | ffff | ffff | C, DC, Z, OV, N | |
| RLCF | f, d, a | Rotate Left f through Carry | 1 | 0011 | 01da | ffff | ffff | C, Z, N | 1, 2 |
| RLNCF | f, d, a | Rotate Left f (No Carry) | 1 | 0100 | 01da | ffff | ffff | Z, N | |
| RRCF | f, d, a | Rotate Right f through Carry | 1 | 0011 | 00da | ffff | ffff | C, Z, N | |
| RRNCF | f, d, a | Rotate Right f (No Carry) | 1 | 0100 | 00da | ffff | ffff | Z, N | |
| SETF | f, a | Set f | 1 | 0110 | 100a | ffff | ffff | None | 1, 2 |
| SUBFWB | f, d, a | Subtract f from WREG with borrow | 1 | 0101 | 01da | ffff | ffff | C, DC, Z, OV, N | |
| SUBWF | f, d, a | Subtract WREG from f | 1 | 0101 | 11da | ffff | ffff | C, DC, Z, OV, N | 1, 2 |
| SUBWFB | f, d, a | Subtract WREG from f with borrow | 1 | 0101 | 10da | ffff | ffff | C, DC, Z, OV, N | |
| SWAPF | f, d, a | Swap nibbles in f | 1 | 0011 | 10da | ffff | ffff | None | 4 |
| TSTFSZ | f, a | Test f, skip if 0 | 1 (2 or 3) | 0110 | 011a | ffff | ffff | None | 1, 2 |
| XORWF | f, d, a | Exclusive OR WREG with f | 1 | 0001 | 10da | ffff | ffff | Z, N | |

- Note 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 this instruction is executed on the TMR0 register (and where applicable, 'd' = 1), the prescaler will be cleared if assigned.
- 3:** If Program Counter (PC) is modified or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a `NOF`.
- 4:** Some instructions are two-word instructions. The second word of these instructions will be executed as a `NOF` 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.

TABLE 36-2: INSTRUCTION SET (CONTINUED)

| Mnemonic, Operands | Description | Cycles | 16-Bit Instruction Word | | | | Status Affected | Notes | |
|--------------------------------|-------------|--------------------------------|-------------------------|------|------|------|--------------------|-----------------------------------|------|
| | | | MSb | | | LSb | | | |
| BIT-ORIENTED OPERATIONS | | | | | | | | | |
| BCF | f, b, a | Bit Clear f | 1 | 1001 | bbba | ffff | ffff | None | 1, 2 |
| BSF | f, b, a | Bit Set f | 1 | 1000 | bbba | ffff | ffff | None | 1, 2 |
| BTFSC | f, b, a | Bit Test f, Skip if Clear | 1 (2 or 3) | 1011 | bbba | ffff | ffff | None | 3, 4 |
| BTFSS | f, b, a | Bit Test f, Skip if Set | 1 (2 or 3) | 1010 | bbba | ffff | ffff | None | 3, 4 |
| BTG | f, b, a | Bit Toggle f | 1 | 0111 | bbba | ffff | ffff | None | 1, 2 |
| CONTROL OPERATIONS | | | | | | | | | |
| BC | n | Branch if Carry | 1 (2) | 1110 | 0010 | nnnn | nnnn | None | 4 |
| BN | n | Branch if Negative | 1 (2) | 1110 | 0110 | nnnn | nnnn | None | |
| BNC | n | Branch if Not Carry | 1 (2) | 1110 | 0011 | nnnn | nnnn | None | |
| BNN | n | Branch if Not Negative | 1 (2) | 1110 | 0111 | nnnn | nnnn | None | |
| BNOV | n | Branch if Not Overflow | 1 (2) | 1110 | 0101 | nnnn | nnnn | None | |
| BNZ | n | Branch if Not Zero | 1 (2) | 1110 | 0001 | nnnn | nnnn | None | |
| BOV | n | Branch if Overflow | 1 (2) | 1110 | 0100 | nnnn | nnnn | None | |
| BRA | n | Branch Unconditionally | 2 | 1101 | 0nnn | nnnn | nnnn | None | |
| BZ | n | Branch if Zero | 1 (2) | 1110 | 0000 | nnnn | nnnn | None | |
| CALL | k, s | Call subroutine 1st word | 2 | 1110 | 110s | kkkk | kkkk | None | |
| | | 2nd word | | 1111 | kkkk | kkkk | kkkk | | |
| CLRWDT | — | Clear Watchdog Timer | 1 | 0000 | 0000 | 0000 | 0100 | \overline{TO} , \overline{PD} | |
| DAW | — | Decimal Adjust WREG | 1 | 0000 | 0000 | 0000 | 0111 | C | |
| GOTO | k | Go to address 1st word | 2 | 1110 | 1111 | kkkk | kkkk | None | |
| | | 2nd word | | 1111 | kkkk | kkkk | kkkk | | |
| NOP | — | No Operation | 1 | 0000 | 0000 | 0000 | 0000 | None | |
| NOP | — | No Operation | 1 | 1111 | xxxx | xxxx | xxxx | None | |
| 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 | |
| RCALL | n | Relative Call | 2 | 1101 | 1nnn | nnnn | nnnn | None | |
| RESET | | Software device Reset | 1 | 0000 | 0000 | 1111 | 1111 | All | |
| RETFIE | s | Return from interrupt enable | 2 | 0000 | 0000 | 0001 | 000s | GIE/GIEH, PEIE/GIEL | |
| RETLW | k | Return with literal in WREG | 2 | 0000 | 1100 | kkkk | kkkk | None | |
| RETURN | s | Return from Subroutine | 2 | 0000 | 0000 | 0001 | 001s | None | |
| SLEEP | — | Go into Standby mode | 1 | 0000 | 0000 | 0000 | 0011 | \overline{TO} , \overline{PD} | |

- Note 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 this instruction is executed on the TMR0 register (and where applicable, 'd' = 1), the prescaler will be cleared if assigned.
- 3:** 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`.
- 4:** Some instructions are two-word instructions. The second word 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.

TABLE 36-2: INSTRUCTION SET (CONTINUED)

| Mnemonic, Operands | Description | Cycles | 16-Bit Instruction Word | | | | Status Affected | Notes | |
|--|-------------|--|-------------------------|------|------|------|--------------------|-----------------|--|
| | | | MSb | | | LSb | | | |
| LITERAL OPERATIONS | | | | | | | | | |
| 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, k | Move literal (12-bit) 2nd word to FSR(f) 1st word | 2 | 1110 | 1110 | 00ff | kkkk | None | |
| MOVLB | k | Move literal to BSR<3:0> | 1 | 0000 | 0001 | 0000 | 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 | |
| SUBLW | k | Subtract WREG from literal | 1 | 0000 | 1000 | kkkk | kkkk | C, DC, Z, OV, N | |
| XORLW | k | Exclusive OR literal with WREG | 1 | 0000 | 1010 | kkkk | kkkk | Z, N | |
| DATA MEMORY ↔ PROGRAM MEMORY OPERATIONS | | | | | | | | | |
| TBLRD* | | Table Read | 2 | 0000 | 0000 | 0000 | 1000 | None | |
| TBLRD*+ | | Table Read with post-increment | | 0000 | 0000 | 0000 | 1001 | None | |
| TBLRD*- | | Table Read with post-decrement | | 0000 | 0000 | 0000 | 1010 | None | |
| TBLRD*+ | | Table Read with pre-increment | | 0000 | 0000 | 0000 | 1011 | None | |
| TBLWT* | | Table Write | 2 | 0000 | 0000 | 0000 | 1100 | None | |
| TBLWT*+ | | Table Write with post-increment | | 0000 | 0000 | 0000 | 1101 | None | |
| TBLWT*- | | Table Write with post-decrement | | 0000 | 0000 | 0000 | 1110 | None | |
| TBLWT*+ | | Table Write with pre-increment | | 0000 | 0000 | 0000 | 1111 | None | |

- Note 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 this instruction is executed on the TMR0 register (and where applicable, 'd' = 1), the prescaler will be cleared if assigned.
- 3:** 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`.
- 4:** Some instructions are two-word instructions. The second word 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.

36.1.1 STANDARD INSTRUCTION SET

ADDLW ADD literal to W

| | | | | |
|-------------------|---|------------------|--------------|------------|
| Syntax: | ADDLW k | | | |
| Operands: | $0 \leq k \leq 255$ | | | |
| 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: | $0 \leq f \leq 255$ $d \in [0,1]$ $a \in [0,1]$ | | | |
| Operation: | $(W) + (f) \rightarrow \text{dest}$ | | | |
| 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). If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See Section 36.2.3 "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

Note: 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).

ADDWFC **ADD W and CARRY bit to f**

Syntax: ADDWFC f {,d {,a}}

Operands: $0 \leq f \leq 255$
 $d \in [0,1]$
 $a \in [0,1]$

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 placed in W. If 'd' is '1', the result is placed in data memory location 'f'. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See [Section 36.2.3 "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: $0 \leq k \leq 255$

Operation: (W) .AND. k → W

Status Affected: N, Z

Encoding:

| | | | |
|------|------|------|------|
| 0000 | 1011 | kkkk | kkkk |
|------|------|------|------|

Description: The contents of W are AND'ed 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: $0 \leq f \leq 255$
 $d \in [0,1]$
 $a \in [0,1]$

Operation: (W) .AND. (f) → dest

Status Affected: N, Z

Encoding:

| | | | |
|------|------|------|------|
| 0001 | 01da | ffff | ffff |
|------|------|------|------|

Description: The contents of W are AND'ed 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).
 If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.
 If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See [Section 36.2.3 "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: $-128 \leq n \leq 127$

Operation: if CARRY bit is '1'
 $(PC) + 2 + 2n \rightarrow PC$

Status Affected: None

Encoding:

| | | | |
|------|------|------|------|
| 1110 | 0010 | nnnn | nnnn |
|------|------|------|------|

Description: If the CARRY bit is '1', then the program will branch.
 The 2'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 2-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)

PIC18(L)F67K40

| BCF | Bit Clear f | | | | | | | | |
|-------------------|---|--------------|--------------------|------|------|--------|-------------------|--------------|--------------------|
| Syntax: | BCF f, b {,a} | | | | | | | | |
| Operands: | $0 \leq f \leq 255$ $0 \leq b \leq 7$ $a \in [0,1]$ | | | | | | | | |
| Operation: | $0 \rightarrow f \langle b \rangle$ | | | | | | | | |
| Status Affected: | None | | | | | | | | |
| Encoding: | <table border="1"> <tr> <td>1001</td> <td>bbba</td> <td>ffff</td> <td>ffff</td> </tr> </table> | 1001 | bbba | ffff | ffff | | | | |
| 1001 | bbba | ffff | ffff | | | | | | |
| Description: | Bit 'b' in register 'f' is cleared. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See Section 36.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode" for details. | | | | | | | | |
| Words: | 1 | | | | | | | | |
| Cycles: | 1 | | | | | | | | |
| Q Cycle Activity: | <table border="1"> <thead> <tr> <th>Q1</th> <th>Q2</th> <th>Q3</th> <th>Q4</th> </tr> </thead> <tbody> <tr> <td>Decode</td> <td>Read register 'f'</td> <td>Process Data</td> <td>Write register 'f'</td> </tr> </tbody> </table> | Q1 | Q2 | Q3 | Q4 | Decode | Read register 'f' | Process Data | Write register 'f' |
| 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: | $-128 \leq n \leq 127$ | | | | | | | | | | | | |
| Operation: | if NEGATIVE bit is '1' $(PC) + 2 + 2n \rightarrow PC$ | | | | | | | | | | | | |
| Status Affected: | None | | | | | | | | | | | | |
| Encoding: | <table border="1"> <tr> <td>1110</td> <td>0110</td> <td>nnnn</td> <td>nnnn</td> </tr> </table> | 1110 | 0110 | nnnn | nnnn | | | | | | | | |
| 1110 | 0110 | nnnn | nnnn | | | | | | | | | | |
| Description: | If the NEGATIVE bit is '1', then the program will branch. The 2'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 2-cycle instruction. | | | | | | | | | | | | |
| Words: | 1 | | | | | | | | | | | | |
| Cycles: | 1(2) | | | | | | | | | | | | |
| Q Cycle Activity: | <table border="1"> <thead> <tr> <th>Q1</th> <th>Q2</th> <th>Q3</th> <th>Q4</th> </tr> </thead> <tbody> <tr> <td>Decode</td> <td>Read literal 'n'</td> <td>Process Data</td> <td>Write to PC</td> </tr> <tr> <td>No operation</td> <td>No operation</td> <td>No operation</td> <td>No operation</td> </tr> </tbody> </table> | Q1 | Q2 | Q3 | Q4 | Decode | Read literal 'n' | Process Data | Write to PC | No operation | No operation | No operation | No operation |
| Q1 | Q2 | Q3 | Q4 | | | | | | | | | | |
| Decode | Read literal 'n' | Process Data | Write to PC | | | | | | | | | | |
| No operation | No operation | No operation | No operation | | | | | | | | | | |
| If No Jump: | <table border="1"> <thead> <tr> <th>Q1</th> <th>Q2</th> <th>Q3</th> <th>Q4</th> </tr> </thead> <tbody> <tr> <td>Decode</td> <td>Read literal 'n'</td> <td>Process Data</td> <td>No operation</td> </tr> </tbody> </table> | Q1 | Q2 | Q3 | Q4 | Decode | Read literal 'n' | Process Data | No operation | | | | |
| 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)

PIC18(L)F67K40

BNC **Branch if Not Carry**

Syntax: BNC n

Operands: $-128 \leq n \leq 127$

Operation: if CARRY bit is '0'
(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 2'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 2-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: $-128 \leq n \leq 127$

Operation: if NEGATIVE bit is '0'
(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 2'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 2-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)

PIC18(L)F67K40

BNOV **Branch if Not Overflow**

Syntax: BNOV n

Operands: -128 ≤ n ≤ 127

Operation: if OVERFLOW bit is '0'
(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 2'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 2-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 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'
(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 2'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 2-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)

BRA Unconditional Branch

Syntax: BRA n

Operands: $-1024 \leq n \leq 1023$

Operation: $(PC) + 2 + 2n \rightarrow PC$

Status Affected: None

Encoding:

| | | | |
|------|------|------|------|
| 1101 | 0nnn | nnnn | nnnn |
|------|------|------|------|

Description: Add the 2'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 2-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: $0 \leq f \leq 255$
 $0 \leq b \leq 7$
 $a \in [0,1]$

Operation: $1 \rightarrow f$

Status Affected: None

Encoding:

| | | | |
|------|------|------|------|
| 1000 | bbba | ffff | ffff |
|------|------|------|------|

Description: Bit 'b' in register 'f' is set.
If 'a' is '0', the Access Bank is selected.
If 'a' is '1', the BSR is used to select the GPR bank.
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See [Section 36.2.3 "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: BSF FLAG_REG, 7, 1

Before Instruction
FLAG_REG = 0Ah

After Instruction
FLAG_REG = 8Ah

PIC18(L)F67K40

BTFSK Bit Test File, Skip if Clear

Syntax: BTFSK f, b {,a}

Operands: $0 \leq f \leq 255$
 $0 \leq b \leq 7$
 $a \in [0,1]$

Operation: skip if (f) = 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 2-cycle instruction.
 If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.
 If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh).
 See [Section 36.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"](#) for details.

Words: 1

Cycles: 1(2)
Note: 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|--------|-------------------|--------------|--------------|
| Decode | Read register 'f' | Process Data | No operation |

If skip:

| Q1 | Q2 | Q3 | Q4 |
|--------------|--------------|--------------|--------------|
| No operation | No operation | No operation | No operation |

If skip and followed by 2-word instruction:

| Q1 | Q2 | Q3 | Q4 |
|--------------|--------------|--------------|--------------|
| No operation | No operation | No operation | No operation |
| No operation | No operation | No operation | No operation |

Example:

```
HERE BTFSK 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)

BTFSK Bit Test File, Skip if Set

Syntax: BTFSK f, b {,a}

Operands: $0 \leq f \leq 255$
 $0 \leq b < 7$
 $a \in [0,1]$

Operation: skip if (f) = 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 2-cycle instruction.
 If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.
 If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh).
 See [Section 36.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"](#) for details.

Words: 1

Cycles: 1(2)
Note: 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|--------|-------------------|--------------|--------------|
| Decode | Read register 'f' | Process Data | No operation |

If skip:

| Q1 | Q2 | Q3 | Q4 |
|--------------|--------------|--------------|--------------|
| No operation | No operation | No operation | No operation |

If skip and followed by 2-word instruction:

| Q1 | Q2 | Q3 | Q4 |
|--------------|--------------|--------------|--------------|
| No operation | No operation | No operation | No operation |
| No operation | No operation | No operation | No operation |

Example:

```
HERE BTFSK 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: $0 \leq f \leq 255$
 $0 \leq b < 7$
 $a \in [0,1]$

Operation: $(\overline{f} < b >) \rightarrow f < b >$

Status Affected: None

Encoding:

| | | | |
|------|------|------|------|
| 0111 | bbba | ffff | ffff |
|------|------|------|------|

Description: Bit 'b' in data memory location 'f' is inverted.
 If 'a' is '0', the Access Bank is selected.
 If 'a' is '1', the BSR is used to select the GPR bank.
 If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See [Section 36.2.3 "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]

BOV **Branch if Overflow**

Syntax: BOV n

Operands: $-128 \leq n \leq 127$

Operation: if OVERFLOW bit is '1'
 $(PC) + 2 + 2n \rightarrow PC$

Status Affected: None

Encoding:

| | | | |
|------|------|------|------|
| 1110 | 0100 | nnnn | nnnn |
|------|------|------|------|

Description: If the OVERFLOW bit is '1', then the program will branch.
 The 2'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 2-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)

PIC18(L)F67K40

BZ **Branch if Zero**

Syntax: BZ n

Operands: -128 ≤ n ≤ 127

Operation: if ZERO bit is '1'
(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 2'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 2-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 |
| No operation | No operation | No operation | No operation |

Example: HERE BZ 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
s ∈ [0,1]

Operation: (PC) + 4 → TOS,
k → PC<20:1>,
if s = 1
(W) → WS,
(Status) → STATUS, (BSR) → BSR

Status Affected: None

Encoding:

| | | | |
|------|---------------------|--------------------|-------------------|
| 1110 | 110s | k ₇ kkk | kkkk ₀ |
| 1111 | k ₁₉ kkk | kkkk | kkkk ₈ |

1st word (k<7:0>)
2nd word(k<19:8>)

Description: Subroutine call of entire 2-Mbyte memory range. First, return address (PC + 4) is pushed onto the return stack. If 's' = 1, the W, Status and BSR registers are also pushed into their respective shadow registers, WS, STATUS and BSR. If 's' = 0, no update occurs (default). Then, the 20-bit value 'k' is loaded into PC<20:1>. CALL is a 2-cycle instruction.

Words: 2

Cycles: 2

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|--------------|--|------------------|--|
| Decode | Read literal 'k'<7:0>, PUSH PC to stack | PUSH PC to stack | Read literal 'k'<19:8>, 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)
WS = W
BSRS = BSR
STATUS = Status

| CLRF | Clear f | | | | | | | | |
|-------------------|--|--------------|--------------------|------|------|--------|-------------------|--------------|--------------------|
| Syntax: | CLRF f{,a} | | | | | | | | |
| Operands: | $0 \leq f \leq 255$ $a \in [0,1]$ | | | | | | | | |
| Operation: | $000h \rightarrow f$ $1 \rightarrow Z$ | | | | | | | | |
| Status Affected: | Z | | | | | | | | |
| Encoding: | <table border="1"> <tr> <td>0110</td> <td>101a</td> <td>ffff</td> <td>ffff</td> </tr> </table> | 0110 | 101a | ffff | ffff | | | | |
| 0110 | 101a | ffff | ffff | | | | | | |
| Description: | <p>Clears the contents of the specified register.</p> <p>If 'a' is '0', the Access Bank is selected.</p> <p>If 'a' is '1', the BSR is used to select the GPR bank.</p> <p>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See Section 36.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode" for details.</p> | | | | | | | | |
| Words: | 1 | | | | | | | | |
| Cycles: | 1 | | | | | | | | |
| Q Cycle Activity: | <table border="1"> <thead> <tr> <th>Q1</th> <th>Q2</th> <th>Q3</th> <th>Q4</th> </tr> </thead> <tbody> <tr> <td>Decode</td> <td>Read register 'f'</td> <td>Process Data</td> <td>Write register 'f'</td> </tr> </tbody> </table> | Q1 | Q2 | Q3 | Q4 | Decode | Read register 'f' | Process Data | Write register 'f' |
| 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: | $000h \rightarrow$ WDT, $000h \rightarrow$ WDT postscaler, $1 \rightarrow \overline{TO}$, $1 \rightarrow \overline{PD}$ | | | | | | | | |
| Status Affected: | \overline{TO} , \overline{PD} | | | | | | | | |
| Encoding: | <table border="1"> <tr> <td>0000</td> <td>0000</td> <td>0000</td> <td>0100</td> </tr> </table> | 0000 | 0000 | 0000 | 0100 | | | | |
| 0000 | 0000 | 0000 | 0100 | | | | | | |
| Description: | CLRWDT instruction resets the Watchdog Timer. It also resets the postscaler of the WDT. Status bits, \overline{TO} and \overline{PD} , are set. | | | | | | | | |
| Words: | 1 | | | | | | | | |
| Cycles: | 1 | | | | | | | | |
| Q Cycle Activity: | <table border="1"> <thead> <tr> <th>Q1</th> <th>Q2</th> <th>Q3</th> <th>Q4</th> </tr> </thead> <tbody> <tr> <td>Decode</td> <td>No operation</td> <td>Process Data</td> <td>No operation</td> </tr> </tbody> </table> | Q1 | Q2 | Q3 | Q4 | Decode | No operation | Process Data | No operation |
| Q1 | Q2 | Q3 | Q4 | | | | | | |
| Decode | No operation | Process Data | No operation | | | | | | |

Example: CLRWDT

Before Instruction
 WDT Counter = ?

After Instruction
 WDT Counter = 00h
 WDT Postscaler = 0
 \overline{TO} = 1
 \overline{PD} = 1

COMF **Complement f**

Syntax: COMF f {,d {,a}}

Operands: $0 \leq f \leq 255$
 $d \in [0,1]$
 $a \in [0,1]$

Operation: $(\bar{f}) \rightarrow \text{dest}$

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 register 'f' (default). If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See [Section 36.2.3 "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 REG, 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: $0 \leq f \leq 255$
 $a \in [0,1]$

Operation: $(f) - (W)$,
skip if $(f) = (W)$
(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 $f = W$, then the fetched instruction is discarded and a NOP is executed instead, making this a 2-cycle instruction. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See [Section 36.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"](#) for details.

Words: 1

Cycles: 1(2)

Note: 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|--------|-------------------|--------------|--------------|
| Decode | Read register 'f' | Process Data | No operation |

If skip:

| Q1 | Q2 | Q3 | Q4 |
|--------------|--------------|--------------|--------------|
| No operation | No operation | No operation | No operation |

If skip and followed by 2-word instruction:

| Q1 | Q2 | Q3 | Q4 |
|--------------|--------------|--------------|--------------|
| 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 \neq W;
PC = Address (NEQUAL)

PIC18(L)F67K40

CPFSGT Compare f with W, skip if f > W

Syntax: CPFSGT f{,a}
 Operands: $0 \leq f \leq 255$
 $a \in [0,1]$
 Operation: $(f) - (W)$,
 skip if $(f) > (W)$
 (unsigned comparison)
 Status Affected: None
 Encoding:

| | | | |
|------|------|------|------|
| 0110 | 010a | ffff | ffff |
|------|------|------|------|

Description: Compares the contents of data memory location 'f' to the contents of the 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 2-cycle instruction. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See [Section 36.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"](#) for details.

Words: 1
 Cycles: 1(2)
Note: 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|--------|-------------------|--------------|--------------|
| Decode | Read register 'f' | Process Data | No operation |

If skip:

| Q1 | Q2 | Q3 | Q4 |
|--------------|--------------|--------------|--------------|
| No operation | No operation | No operation | No operation |

If skip and followed by 2-word instruction:

| Q1 | Q2 | Q3 | Q4 |
|--------------|--------------|--------------|--------------|
| 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 = ?
 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: $0 \leq f \leq 255$
 $a \in [0,1]$
 Operation: $(f) - (W)$,
 skip if $(f) < (W)$
 (unsigned comparison)
 Status Affected: None
 Encoding:

| | | | |
|------|------|------|------|
| 0110 | 000a | 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 less than the contents of W, then the fetched instruction is discarded and a NOP is executed instead, making this a 2-cycle instruction. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.

Words: 1
 Cycles: 1(2)
Note: 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|--------|-------------------|--------------|--------------|
| Decode | Read register 'f' | Process Data | No operation |

If skip:

| Q1 | Q2 | Q3 | Q4 |
|--------------|--------------|--------------|--------------|
| No operation | No operation | No operation | No operation |

If skip and followed by 2-word instruction:

| Q1 | Q2 | Q3 | Q4 |
|--------------|--------------|--------------|--------------|
| No operation | No operation | No operation | No operation |
| No operation | No operation | No operation | No operation |

Example: HERE CPFSLT REG, 1
 NLESS :
 LESS :

Before Instruction
 PC = Address (HERE)
 W = ?
 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 (W<3:0>) + 6 → W<3:0>; else (W<3:0>) → W<3:0>; If [W<7:4> + DC > 9] or [C = 1] then (W<7:4>) + 6 + DC → W<7:4>; else (W<7:4>) + DC → W<7:4> | | | | | | | | |
| Status Affected: | C | | | | | | | | |
| Encoding: | <table border="1"><tr><td>0000</td><td>0000</td><td>0000</td><td>0111</td></tr></table> | 0000 | 0000 | 0000 | 0111 | | | | |
| 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: | <table border="1"><thead><tr><th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr></thead><tbody><tr><td>Decode</td><td>Read register W</td><td>Process Data</td><td>Write W</td></tr></tbody></table> | Q1 | Q2 | Q3 | Q4 | Decode | Read register W | Process Data | Write W |
| Q1 | Q2 | Q3 | Q4 | | | | | | |
| Decode | Read register W | Process Data | Write W | | | | | | |

Example 1:

| DAW | |
|--------------------|-------|
| Before Instruction | |
| W | = A5h |
| C | = 0 |
| DC | = 0 |
| After Instruction | |
| W | = 05h |
| C | = 1 |
| DC | = 0 |

Example 2:

| | |
|--------------------|-------|
| 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 d ∈ [0,1] a ∈ [0,1] | | | | | | | | |
| Operation: | (f) – 1 → dest | | | | | | | | |
| Status Affected: | C, DC, N, OV, Z | | | | | | | | |
| Encoding: | <table border="1"><tr><td>0000</td><td>01da</td><td>ffff</td><td>ffff</td></tr></table> | 0000 | 01da | ffff | ffff | | | | |
| 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 register 'f' (default). If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. 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 Section 36.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode" for details. | | | | | | | | |
| Words: | 1 | | | | | | | | |
| Cycles: | 1 | | | | | | | | |
| Q Cycle Activity: | <table border="1"><thead><tr><th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr></thead><tbody><tr><td>Decode</td><td>Read register 'f'</td><td>Process Data</td><td>Write to destination</td></tr></tbody></table> | Q1 | Q2 | Q3 | Q4 | Decode | Read register 'f' | Process Data | Write to destination |
| 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: $0 \leq f \leq 255$
 $d \in [0,1]$
 $a \in [0,1]$

Operation: $(f) - 1 \rightarrow \text{dest}$,
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). If the result is '0', the next instruction, which is already fetched, is discarded and a NOP is executed instead, making it a 2-cycle instruction. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See [Section 36.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"](#) for details.

Words: 1

Cycles: 1(2)
Note: 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|--------|-------------------|--------------|----------------------|
| Decode | Read register 'f' | Process Data | Write to destination |

If skip:

| Q1 | Q2 | Q3 | Q4 |
|--------------|--------------|--------------|--------------|
| No operation | No operation | No operation | No operation |

If skip and followed by 2-word instruction:

| Q1 | Q2 | Q3 | Q4 |
|--------------|--------------|--------------|--------------|
| No operation | No operation | No operation | No operation |
| No operation | No operation | No operation | No operation |

Example:

```

HERE      DECFSZ   CNT, 1, 1
            GOTO    LOOP
            CONTINUE
    
```

Before Instruction
PC = Address (HERE)

After Instruction
CNT = CNT - 1
If CNT = 0;
PC = Address (CONTINUE)
If CNT \neq 0;
PC = Address (HERE + 2)

DCFSNZ **Decrement f, skip if not 0**

Syntax: DCFSNZ f {,d {,a}}

Operands: $0 \leq f \leq 255$
 $d \in [0,1]$
 $a \in [0,1]$

Operation: $(f) - 1 \rightarrow \text{dest}$,
skip if result \neq 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). If the result is not '0', the next instruction, which is already fetched, is discarded and a NOP is executed instead, making it a 2-cycle instruction. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See [Section 36.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"](#) for details.

Words: 1

Cycles: 1(2)
Note: 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|--------|-------------------|--------------|----------------------|
| Decode | Read register 'f' | Process Data | Write to destination |

If skip:

| Q1 | Q2 | Q3 | Q4 |
|--------------|--------------|--------------|--------------|
| No operation | No operation | No operation | No operation |

If skip and followed by 2-word instruction:

| Q1 | Q2 | Q3 | Q4 |
|--------------|--------------|--------------|--------------|
| 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 = ?

After Instruction
TEMP = TEMP - 1,
If TEMP = 0;
PC = Address (ZERO)
If TEMP \neq 0;
PC = Address (NZERO)

PIC18(L)F67K40

GOTO Unconditional Branch

Syntax: GOTO k

Operands: $0 \leq k \leq 1048575$

Operation: $k \rightarrow PC<20:1>$

Status Affected: None

Encoding:

| | | | |
|------|---------------------|--------------------|-------------------|
| 1110 | 1111 | k ₇ kkk | kkkk ₀ |
| 1111 | k ₁₉ kkk | kkkk | kkkk ₈ |

1st word (k<7:0>)
2nd word(k<19:8>)

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 2-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: GOTO THERE

After Instruction
PC = Address (THERE)

INCF Increment f

Syntax: INCF f{,d{,a}}

Operands: $0 \leq f \leq 255$
 $d \in [0,1]$
 $a \in [0,1]$

Operation: $(f) + 1 \rightarrow \text{dest}$

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 placed in W. If 'd' is '1', the result is placed back in register 'f' (default). If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See [Section 36.2.3 "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: $0 \leq f \leq 255$
 $d \in [0,1]$
 $a \in [0,1]$

Operation: $(f) + 1 \rightarrow \text{dest}$,
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). If the result is '0', the next instruction, which is already fetched, is discarded and a NOP is executed instead, making it a 2-cycle instruction. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See [Section 36.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"](#) for details.

Words: 1

Cycles: 1(2)
Note: 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|--------|-------------------|--------------|----------------------|
| Decode | Read register 'f' | Process Data | Write to destination |

If skip:

| Q1 | Q2 | Q3 | Q4 |
|--------------|--------------|--------------|--------------|
| No operation | No operation | No operation | No operation |

If skip and followed by 2-word instruction:

| Q1 | Q2 | Q3 | Q4 |
|--------------|--------------|--------------|--------------|
| 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
PC = Address (HERE)

After Instruction
CNT = CNT + 1
If CNT = 0;
PC = Address (ZERO)
If CNT \neq 0;
PC = Address (NZERO)

INFSNZ **Increment f, skip if not 0**

Syntax: INFSNZ f {,d {,a}}

Operands: $0 \leq f \leq 255$
 $d \in [0,1]$
 $a \in [0,1]$

Operation: $(f) + 1 \rightarrow \text{dest}$,
skip if result \neq 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). If the result is not '0', the next instruction, which is already fetched, is discarded and a NOP is executed instead, making it a 2-cycle instruction. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See [Section 36.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"](#) for details.

Words: 1

Cycles: 1(2)
Note: 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|--------|-------------------|--------------|----------------------|
| Decode | Read register 'f' | Process Data | Write to destination |

If skip:

| Q1 | Q2 | Q3 | Q4 |
|--------------|--------------|--------------|--------------|
| No operation | No operation | No operation | No operation |

If skip and followed by 2-word instruction:

| Q1 | Q2 | Q3 | Q4 |
|--------------|--------------|--------------|--------------|
| 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
PC = Address (HERE)

After Instruction
REG = REG + 1
If REG \neq 0;
PC = Address (NZERO)
If REG = 0;
PC = Address (ZERO)

PIC18(L)F67K40

IORLW **Inclusive OR literal with W**

Syntax: IORLW k

Operands: $0 \leq k \leq 255$

Operation: (W) .OR. k \rightarrow 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: $0 \leq f \leq 255$
 $d \in [0,1]$
 $a \in [0,1]$

Operation: (W) .OR. (f) \rightarrow dest

Status Affected: N, Z

Encoding:

| | | | |
|------|------|------|------|
| 0001 | 00da | ffff | ffff |
|------|------|------|------|

Description: Inclusive OR W with register 'f'. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default).
 If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.
 If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See [Section 36.2.3 "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

PIC18(L)F67K40

| LFSR | Load FSR | | | | | | | | | | | | | | | |
|-------------------|--|----------------------|---------------------|--------------------------------|---------------------|------|------|--------------------|----------------------|--------------|--------------------------------|--|--------|----------------------|--------------|----------------------------|
| Syntax: | LFSR f, k | | | | | | | | | | | | | | | |
| Operands: | 0 ≤ f ≤ 2 0 ≤ k ≤ 4095 | | | | | | | | | | | | | | | |
| Operation: | k → FSRf | | | | | | | | | | | | | | | |
| Status Affected: | None | | | | | | | | | | | | | | | |
| Encoding: | <table border="1"> <tr> <td>1110</td> <td>1110</td> <td>00ff</td> <td>k₁₁kkk</td> </tr> <tr> <td>1111</td> <td>0000</td> <td>k₇kkk</td> <td>kkkk</td> </tr> </table> | 1110 | 1110 | 00ff | k ₁₁ kkk | 1111 | 0000 | k ₇ kkk | kkkk | | | | | | | |
| 1110 | 1110 | 00ff | k ₁₁ kkk | | | | | | | | | | | | | |
| 1111 | 0000 | k ₇ kkk | kkkk | | | | | | | | | | | | | |
| Description: | The 12-bit literal 'k' is loaded into the File Select Register pointed to by 'f'. | | | | | | | | | | | | | | | |
| Words: | 2 | | | | | | | | | | | | | | | |
| Cycles: | 2 | | | | | | | | | | | | | | | |
| Q Cycle Activity: | <table border="1"> <thead> <tr> <th></th> <th>Q1</th> <th>Q2</th> <th>Q3</th> <th>Q4</th> </tr> </thead> <tbody> <tr> <td></td> <td>Decode</td> <td>Read literal 'k' MSB</td> <td>Process Data</td> <td>Write literal 'k' MSB to FSRfH</td> </tr> <tr> <td></td> <td>Decode</td> <td>Read literal 'k' LSB</td> <td>Process Data</td> <td>Write literal 'k' to FSRfL</td> </tr> </tbody> </table> | | Q1 | Q2 | Q3 | Q4 | | Decode | Read literal 'k' MSB | Process Data | Write literal 'k' MSB to FSRfH | | Decode | Read literal 'k' LSB | Process Data | Write literal 'k' to FSRfL |
| | Q1 | Q2 | Q3 | Q4 | | | | | | | | | | | | |
| | Decode | Read literal 'k' MSB | Process Data | Write literal 'k' MSB to FSRfH | | | | | | | | | | | | |
| | Decode | Read literal 'k' LSB | Process Data | Write literal 'k' to FSRfL | | | | | | | | | | | | |

Example: LFSR 2, 3ABh

After Instruction
 FSR2H = 03h
 FSR2L = ABh

| MOVf | Move f | | | | | | | | | | |
|-------------------|--|-------------------|--------------|---------|------|----|--|--------|-------------------|--------------|---------|
| Syntax: | MOVf f {,d {,a}} | | | | | | | | | | |
| Operands: | 0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1] | | | | | | | | | | |
| Operation: | f → dest | | | | | | | | | | |
| Status Affected: | N, Z | | | | | | | | | | |
| Encoding: | <table border="1"> <tr> <td>0101</td> <td>00da</td> <td>ffff</td> <td>ffff</td> </tr> </table> | 0101 | 00da | ffff | ffff | | | | | | |
| 0101 | 00da | ffff | ffff | | | | | | | | |
| Description: | The contents of register 'f' are moved to a destination dependent upon the status of 'd'. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default). Location 'f' can be anywhere in the 256-byte bank. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. 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 Section 36.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode" for details. | | | | | | | | | | |
| Words: | 1 | | | | | | | | | | |
| Cycles: | 1 | | | | | | | | | | |
| Q Cycle Activity: | <table border="1"> <thead> <tr> <th></th> <th>Q1</th> <th>Q2</th> <th>Q3</th> <th>Q4</th> </tr> </thead> <tbody> <tr> <td></td> <td>Decode</td> <td>Read register 'f'</td> <td>Process Data</td> <td>Write W</td> </tr> </tbody> </table> | | Q1 | Q2 | Q3 | Q4 | | Decode | Read register 'f' | Process Data | Write W |
| | Q1 | Q2 | Q3 | Q4 | | | | | | | |
| | Decode | Read register 'f' | Process Data | Write W | | | | | | | |

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: $0 \leq f_s \leq 4095$
 $0 \leq f_d \leq 4095$

Operation: $(f_s) \rightarrow f_d$

Status Affected: None

Encoding:

| | | | |
|------|------|------|------------|
| 1100 | ffff | ffff | ffff f_s |
| 1111 | ffff | ffff | ffff f_d |

1st word (source)
 2nd word (destin.)

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. Either source or destination can be W (a useful special situation). MOVFF is particularly useful for transferring a data memory location to a peripheral register (such as the transmit buffer or an I/O port). The MOVFF instruction cannot use the PCL, TOSU, TOSH or TOSL as the destination register.

Words: 2

Cycles: 2 (3)

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|--------|-------------------------------|--------------|---------------------------|
| Decode | Read register 'f' (src) | Process Data | No operation |
| Decode | No operation No dummy read | No operation | Write register 'f' (dest) |

Example: MOVFF REG1, REG2

Before Instruction

REG1 = 33h
 REG2 = 11h

After Instruction

REG1 = 33h
 REG2 = 33h

MOVLB **Move literal to low nibble in BSR**

Syntax: MOVLW k

Operands: $0 \leq k \leq 255$

Operation: $k \rightarrow \text{BSR}$

Status Affected: None

Encoding:

| | | | |
|------|------|------|------|
| 0000 | 0001 | kkkk | kkkk |
|------|------|------|------|

Description:

The 8-bit literal 'k' is loaded into the Bank Select Register (BSR). The value of BSR<7:4> always remains '0', regardless of the value of k_{7:k4}.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|--------|------------------|--------------|--------------------------|
| Decode | Read literal 'k' | Process Data | Write literal 'k' to BSR |

Example: MOVLB 5

Before Instruction

BSR Register = 02h

After Instruction

BSR Register = 05h

PIC18(L)F67K40

MOVLW Move literal to W

Syntax: MOVLW k

Operands: $0 \leq k \leq 255$

Operation: $k \rightarrow W$

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

After Instruction

W = 5Ah

MOVWF Move W to f

Syntax: MOVWF f{,a}

Operands: $0 \leq f \leq 255$
 $a \in [0,1]$

Operation: $(W) \rightarrow f$

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.

If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See [Section 36.2.3 "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: MOVWF REG, 0

Before Instruction

W = 4Fh
REG = FFh

After Instruction

W = 4Fh
REG = 4Fh

MULLW **Multiply literal with W**

Syntax: MULLW k

Operands: $0 \leq k \leq 255$

Operation: $(W) \times k \rightarrow \text{PRODH:PRODL}$

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.
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: $0 \leq f \leq 255$
 $a \in [0,1]$

Operation: $(W) \times (f) \rightarrow \text{PRODH:PRODL}$

Status Affected: None

Encoding:

| | | | |
|------|------|------|------|
| 0000 | 001a | ffff | ffff |
|------|------|------|------|

Description: An unsigned multiplication is carried out between the contents of W and the register file location 'f'. The 16-bit result is stored in the PRODH:PRODL register pair. PRODH contains the high byte. Both W and 'f' are unchanged.
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. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See [Section 36.2.3 "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: | $0 \leq f \leq 255$ $a \in [0,1]$ | | | | | | | | |
| Operation: | $(\bar{f}) + 1 \rightarrow f$ | | | | | | | | |
| Status Affected: | N, OV, C, DC, Z | | | | | | | | |
| Encoding: | <table border="1" style="display: inline-table; border-collapse: collapse;"> <tr> <td style="padding: 2px;">0110</td> <td style="padding: 2px;">110a</td> <td style="padding: 2px;">ffff</td> <td style="padding: 2px;">ffff</td> </tr> </table> | 0110 | 110a | ffff | ffff | | | | |
| 0110 | 110a | ffff | ffff | | | | | | |
| Description: | <p>Location 'f' is negated using two's complement. The result is placed in the data memory location 'f'. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See Section 36.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode" for details.</p> | | | | | | | | |
| Words: | 1 | | | | | | | | |
| Cycles: | 1 | | | | | | | | |
| Q Cycle Activity: | <table border="1" style="display: inline-table; border-collapse: collapse; text-align: center;"> <thead> <tr> <th>Q1</th> <th>Q2</th> <th>Q3</th> <th>Q4</th> </tr> </thead> <tbody> <tr> <td>Decode</td> <td>Read register 'f'</td> <td>Process Data</td> <td>Write register 'f'</td> </tr> </tbody> </table> | Q1 | Q2 | Q3 | Q4 | Decode | Read register 'f' | Process Data | Write register 'f' |
| 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: | <table border="1" style="display: inline-table; border-collapse: collapse;"> <tr> <td style="padding: 2px;">0000</td> <td style="padding: 2px;">0000</td> <td style="padding: 2px;">0000</td> <td style="padding: 2px;">0000</td> </tr> <tr> <td style="padding: 2px;">1111</td> <td style="padding: 2px;">xxxx</td> <td style="padding: 2px;">xxxx</td> <td style="padding: 2px;">xxxx</td> </tr> </table> | 0000 | 0000 | 0000 | 0000 | 1111 | xxxx | xxxx | xxxx |
| 0000 | 0000 | 0000 | 0000 | | | | | | |
| 1111 | xxxx | xxxx | xxxx | | | | | | |
| Description: | No operation. | | | | | | | | |
| Words: | 1 | | | | | | | | |
| Cycles: | 1 | | | | | | | | |
| Q Cycle Activity: | <table border="1" style="display: inline-table; border-collapse: collapse; text-align: center;"> <thead> <tr> <th>Q1</th> <th>Q2</th> <th>Q3</th> <th>Q4</th> </tr> </thead> <tbody> <tr> <td>Decode</td> <td>No operation</td> <td>No operation</td> <td>No operation</td> </tr> </tbody> </table> | Q1 | Q2 | Q3 | Q4 | Decode | No operation | No operation | No operation |
| 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: | <table border="1"> <tr> <td>0000</td> <td>0000</td> <td>0000</td> <td>0110</td> </tr> </table> | 0000 | 0000 | 0000 | 0110 | | | | |
| 0000 | 0000 | 0000 | 0110 | | | | | | |
| Description: | <p>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.</p> | | | | | | | | |
| Words: | 1 | | | | | | | | |
| Cycles: | 1 | | | | | | | | |
| Q Cycle Activity: | <table border="1"> <thead> <tr> <th>Q1</th> <th>Q2</th> <th>Q3</th> <th>Q4</th> </tr> </thead> <tbody> <tr> <td>Decode</td> <td>No operation</td> <td>POP TOS value</td> <td>No operation</td> </tr> </tbody> </table> | Q1 | Q2 | Q3 | Q4 | Decode | No operation | POP TOS value | No operation |
| 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 | = | NEW |

| PUSH | Push Top of Return Stack | | | | | | | | |
|-------------------|--|--------------|--------------|------|------|--------|-------------------------------|--------------|--------------|
| Syntax: | PUSH | | | | | | | | |
| Operands: | None | | | | | | | | |
| Operation: | (PC + 2) → TOS | | | | | | | | |
| Status Affected: | None | | | | | | | | |
| Encoding: | <table border="1"> <tr> <td>0000</td> <td>0000</td> <td>0000</td> <td>0101</td> </tr> </table> | 0000 | 0000 | 0000 | 0101 | | | | |
| 0000 | 0000 | 0000 | 0101 | | | | | | |
| Description: | <p>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.</p> | | | | | | | | |
| Words: | 1 | | | | | | | | |
| Cycles: | 1 | | | | | | | | |
| Q Cycle Activity: | <table border="1"> <thead> <tr> <th>Q1</th> <th>Q2</th> <th>Q3</th> <th>Q4</th> </tr> </thead> <tbody> <tr> <td>Decode</td> <td>PUSH PC + 2 onto return stack</td> <td>No operation</td> <td>No operation</td> </tr> </tbody> </table> | Q1 | Q2 | Q3 | Q4 | Decode | PUSH PC + 2 onto return stack | No operation | No operation |
| Q1 | Q2 | Q3 | Q4 | | | | | | |
| Decode | PUSH PC + 2 onto return stack | No operation | No operation | | | | | | |

Example:

| | | |
|----------------------|------|-------|
| | PUSH | |
| Before Instruction | | |
| TOS | = | 345Ah |
| PC | = | 0124h |
| After Instruction | | |
| PC | = | 0126h |
| TOS | = | 0126h |
| Stack (1 level down) | = | 345Ah |

PIC18(L)F67K40

RCALL **Relative Call**

Syntax: RCALL n

Operands: $-1024 \leq n \leq 1023$

Operation: (PC) + 2 → TOS,
(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 2'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 2-cycle instruction.

Words: 1

Cycles: 2

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|--------------|--------------------------------------|--------------|--------------|
| Decode | Read literal 'n' 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

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

After Instruction

Registers = Reset Value

Flags* = Reset Value

RETFIE **Return from Interrupt**

Syntax: RETFIE {s}

Operands: s ∈ [0,1]

Operation: (TOS) → PC,
 1 → GIE/GIEH or PEIE/GIEL,
 if s = 1
 (WS) → W,
 (STATUS) → Status,
 (BSRS) → BSR,
 PCLATU, PCLATH are unchanged.

Status Affected: GIE/GIEH, PEIE/GIEL.

Encoding:

| | | | |
|------|------|------|------|
| 0000 | 0000 | 0001 | 000s |
|------|------|------|------|

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. If 's' = 1, the contents of the shadow registers, WS, STATUS and BSRS, are loaded into their corresponding registers, W, Status and BSR. If 's' = 0, no update of these registers occurs (default).

Words: 1

Cycles: 2

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|--------------|--------------|--------------|---------------------------------------|
| Decode | No operation | No operation | POP PC from stack Set GIEH or GIEL |
| No operation | No operation | No operation | No operation |

Example: RETFIE 1

After Interrupt

| | | |
|---------------------|---|--------|
| PC | = | TOS |
| W | = | WS |
| BSR | = | BSRS |
| Status | = | STATUS |
| GIE/GIEH, PEIE/GIEL | = | 1 |

RETLW **Return literal to W**

Syntax: RETLW k

Operands: 0 ≤ k ≤ 255

Operation: k → W,
 (TOS) → PC,
 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 high address latch (PCLATH) remains unchanged.

Words: 1

Cycles: 2

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|--------------|------------------|--------------|----------------------------------|
| Decode | Read literal 'k' | Process Data | POP PC from stack, Write to W |
| No operation | No operation | No operation | No operation |

Example:

```

CALL TABLE ; W contains table
              ; offset value
              ; W now has
              ; table value
:
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,
if s = 1
(WS) → W,
(STATUS) → Status,
(BSRS) → BSR,
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, WS, STATUS and BSRS, are loaded into their corresponding registers, W, Status and BSR. If 's' = 0, no update of these registers occurs (default).

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: RETURN

After Instruction:
PC = TOS

RLCF Rotate Left f through Carry

Syntax: RLCF f {,d {,a}}

Operands: 0 ≤ f ≤ 255
d ∈ [0,1]
a ∈ [0,1]

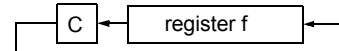
Operation: (f<n>) → dest<n + 1>,
(f<7>) → C,
(C) → dest<0>

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 placed in W. If 'd' is '1', the result is stored back in register 'f' (default).
If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.
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 [Section 36.2.3 "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: RLCF REG, 0, 0

Before Instruction

REG = 1110 0110
C = 0

After Instruction

REG = 1110 0110
W = 1100 1100
C = 1

RLNCF Rotate Left f (No Carry)

Syntax: RLNCF f {,d {,a}}

Operands: $0 \leq f \leq 255$
 $d \in [0,1]$
 $a \in [0,1]$

Operation: $(f\langle n \rangle) \rightarrow \text{dest}\langle n + 1 \rangle$,
 $(f\langle 7 \rangle) \rightarrow \text{dest}\langle 0 \rangle$

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 placed in W. If 'd' is '1', the result is stored back in register 'f' (default). If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See [Section 36.2.3 "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: RLNCF REG, 1, 0

Before Instruction
 REG = 1010 1011

After Instruction
 REG = 0101 0111

RRCF Rotate Right f through Carry

Syntax: RRCF f {,d {,a}}

Operands: $0 \leq f \leq 255$
 $d \in [0,1]$
 $a \in [0,1]$

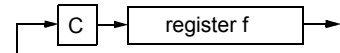
Operation: $(f\langle n \rangle) \rightarrow \text{dest}\langle n - 1 \rangle$,
 $(f\langle 0 \rangle) \rightarrow C$,
 $(C) \rightarrow \text{dest}\langle 7 \rangle$

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 placed in W. If 'd' is '1', the result is placed back in register 'f' (default). If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See [Section 36.2.3 "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: RRCF REG, 0, 0

Before Instruction
 REG = 1110 0110
 C = 0

After Instruction
 REG = 1110 0110
 W = 0111 0011
 C = 0

RRNCF **Rotate Right f (No Carry)**

Syntax: RRNCF f {,d {,a}}

Operands: $0 \leq f \leq 255$
 $d \in [0,1]$
 $a \in [0,1]$

Operation: $(f\langle n \rangle) \rightarrow \text{dest}\langle n - 1 \rangle$,
 $(f\langle 0 \rangle) \rightarrow \text{dest}\langle 7 \rangle$

Status Affected: N, Z

Encoding:

| | | | |
|------|------|------|------|
| 0100 | 00da | ffff | ffff |
|------|------|------|------|

Description: The contents of register 'f' are rotated one bit to the right. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default). If 'a' is '0', the Access Bank will be selected (default), overriding the BSR value. If 'a' is '1', then the bank will be selected as per the BSR value. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See [Section 36.2.3 "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: RRNCF REG, 1, 0

Before Instruction
REG = 1101 0111

After Instruction
REG = 1110 1011

Example 2: RRNCF REG, 0, 0

Before Instruction
W = ?
REG = 1101 0111

After Instruction
W = 1110 1011
REG = 1101 0111

SETF **Set f**

Syntax: SETF f {,a}

Operands: $0 \leq f \leq 255$
 $a \in [0,1]$

Operation: $FFh \rightarrow f$

Status Affected: None

Encoding:

| | | | |
|------|------|------|------|
| 0110 | 100a | ffff | ffff |
|------|------|------|------|

Description: The contents of the specified register are set to FFh. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See [Section 36.2.3 "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, 0 → WDT postscaler, 1 → \overline{TO} , 0 → \overline{PD} | | | | | | | | |
| Status Affected: | \overline{TO} , \overline{PD} | | | | | | | | |
| Encoding: | <table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>0000</td> <td>0000</td> <td>0000</td> <td>0011</td> </tr> </table> | 0000 | 0000 | 0000 | 0011 | | | | |
| 0000 | 0000 | 0000 | 0011 | | | | | | |
| Description: | The Power-down Status bit (\overline{PD}) is cleared. The Time-out Status bit (\overline{TO}) is set. Watchdog Timer and its postscaler are cleared. The processor is put into Sleep mode with the oscillator stopped. | | | | | | | | |
| Words: | 1 | | | | | | | | |
| Cycles: | 1 | | | | | | | | |
| Q Cycle Activity: | <table border="1" style="display: inline-table; vertical-align: middle;"> <thead> <tr> <th>Q1</th> <th>Q2</th> <th>Q3</th> <th>Q4</th> </tr> </thead> <tbody> <tr> <td>Decode</td> <td>No operation</td> <td>Process Data</td> <td>Go to Sleep</td> </tr> </tbody> </table> | Q1 | Q2 | Q3 | Q4 | Decode | No operation | Process Data | Go to Sleep |
| 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.

| SUBFWB | Subtract f from W with borrow | | | | | | | | |
|-------------------|---|--------------|----------------------|------|------|--------|-------------------|--------------|----------------------|
| Syntax: | SUBFWB f {,d {,a}} | | | | | | | | |
| Operands: | $0 \leq f \leq 255$ $d \in [0,1]$ $a \in [0,1]$ | | | | | | | | |
| Operation: | $(W) - (f) - (\overline{C}) \rightarrow \text{dest}$ | | | | | | | | |
| Status Affected: | N, OV, C, DC, Z | | | | | | | | |
| Encoding: | <table border="1" style="display: inline-table; vertical-align: middle;"> <tr> <td>0101</td> <td>01da</td> <td>ffff</td> <td>ffff</td> </tr> </table> | 0101 | 01da | ffff | ffff | | | | |
| 0101 | 01da | ffff | ffff | | | | | | |
| Description: | Subtract register 'f' and CARRY flag (borrow) from W (2's complement method). If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored in register 'f' (default). If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See Section 36.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode" for details. | | | | | | | | |
| Words: | 1 | | | | | | | | |
| Cycles: | 1 | | | | | | | | |
| Q Cycle Activity: | <table border="1" style="display: inline-table; vertical-align: middle;"> <thead> <tr> <th>Q1</th> <th>Q2</th> <th>Q3</th> <th>Q4</th> </tr> </thead> <tbody> <tr> <td>Decode</td> <td>Read register 'f'</td> <td>Process Data</td> <td>Write to destination</td> </tr> </tbody> </table> | Q1 | Q2 | Q3 | Q4 | Decode | Read register 'f' | Process Data | Write to destination |
| Q1 | Q2 | Q3 | Q4 | | | | | | |
| Decode | Read register 'f' | Process Data | Write to destination | | | | | | |

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 = 3

W = 2

C = 1

After Instruction

REG = FF

W = 2

C = 0

Z = 0

N = 1 ; result is negative

Example 2: SUBFWB REG, 0, 0

Before Instruction

REG = 2

W = 5

C = 1

After Instruction

REG = 2

W = 3

C = 1

Z = 0

N = 0 ; result is positive

Example 3: SUBFWB REG, 1, 0

Before Instruction

REG = 1

W = 2

C = 0

After Instruction

REG = 0

W = 2

C = 1

Z = 1 ; result is zero

SUBLW **Subtract W from literal**

Syntax: SUBLW k

Operands: $0 \leq k \leq 255$

Operation: $k - (W) \rightarrow W$

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 ; result is zero
Z = 1
N = 0

Example 3: SUBLW 02h

Before Instruction
W = 03h
C = ?

After Instruction
W = FFh ; (2's complement)
C = 0 ; result is negative
Z = 0
N = 1

SUBWF **Subtract W from f**

Syntax: SUBWF f {,d {,a}}

Operands: $0 \leq f \leq 255$
 $d \in [0,1]$
 $a \in [0,1]$

Operation: $(f) - (W) \rightarrow \text{dest}$

Status Affected: N, OV, C, DC, Z

Encoding:

| | | | |
|------|------|------|------|
| 0101 | 11da | ffff | ffff |
|------|------|------|------|

Description: Subtract W from register 'f' (2's complement method). If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f' (default).
If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See [Section 36.2.3 "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 = 3
W = 2
C = ?

After Instruction
REG = 1
W = 2
C = 1 ; result is positive
Z = 0
N = 0

Example 2: SUBWF REG, 0, 0

Before Instruction
REG = 2
W = 2
C = ?

After Instruction
REG = 2
W = 0
C = 1 ; result is zero
Z = 1
N = 0

Example 3: SUBWF REG, 1, 0

Before Instruction
REG = 1
W = 2
C = ?

After Instruction
REG = FFh ;(2's complement)
W = 2
C = 0 ; result is negative
Z = 0
N = 1

SUBWFB Subtract W from f with Borrow

Syntax: SUBWFB f {,d {,a}}

Operands: $0 \leq f \leq 255$
 $d \in [0,1]$
 $a \in [0,1]$

Operation: $(f) - (W) - (\overline{C}) \rightarrow \text{dest}$

Status Affected: N, OV, C, DC, Z

Encoding:

| | | | |
|------|------|------|------|
| 0101 | 10da | ffff | ffff |
|------|------|------|------|

Description: Subtract W and the CARRY flag (borrow) from register 'f' (2's complement method). If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f' (default). If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See [Section 36.2.3 "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 | |
| Z | = | 0 | |
| N | = | 0 | ; result is positive |

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) ; [2's comp] |
| W | = | 0Eh | (0000 1110) |
| C | = | 0 | |
| Z | = | 0 | |
| N | = | 1 | ; result is negative |

SWAPF Swap f

Syntax: SWAPF f {,d {,a}}

Operands: $0 \leq f \leq 255$
 $d \in [0,1]$
 $a \in [0,1]$

Operation: $(f<3:0>) \rightarrow \text{dest}<7:4>$,
 $(f<7:4>) \rightarrow \text{dest}<3:0>$

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 placed in W. If 'd' is '1', the result is placed in register 'f' (default). If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See [Section 36.2.3 "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: SWAPF REG, 1, 0

Before Instruction

| | | |
|-----|---|-----|
| REG | = | 53h |
|-----|---|-----|

After Instruction

| | | |
|-----|---|-----|
| REG | = | 35h |
|-----|---|-----|

PIC18(L)F67K40

TBLRD Table Read

Syntax: TBLRD (*; *+; *-; +*)

Operands: None

Operation: if TBLRD *,
(Prog Mem (TBLPTR)) → TABLAT;
TBLPTR – No Change;
if TBLRD *+,
(Prog Mem (TBLPTR)) → TABLAT;
(TBLPTR) + 1 → TBLPTR;
if TBLRD *-,
(Prog Mem (TBLPTR)) → TABLAT;
(TBLPTR) – 1 → TBLPTR;
if TBLRD +*,
(TBLPTR) + 1 → TBLPTR;
(Prog Mem (TBLPTR)) → TABLAT;

Status Affected: None

| | | | | |
|-----------|------|------|------|--------|
| Encoding: | 0000 | 0000 | 0000 | 10nn |
| | | | | nn=0 * |
| | | | | =1 *+ |
| | | | | =2 *- |
| | | | | =3 +* |

Description: This instruction is used to read the contents of Program Memory (P.M.). To address the program memory, a pointer called Table Pointer (TBLPTR) is used. The TBLPTR (a 21-bit pointer) points to each byte in the program memory. TBLPTR has a 2-Mbyte address range.

TBLPTR[0] = 0: Least Significant Byte of Program Memory Word
TBLPTR[0] = 1: Most Significant Byte of Program Memory Word

The TBLRD instruction can modify the value of TBLPTR as follows:

- no change
- post-increment
- post-decrement
- pre-increment

Words: 1

Cycles: 2

Q Cycle Activity:

| | Q1 | Q2 | Q3 | Q4 |
|--------------|------------------------------------|--------------|--------------|-----------------------------|
| Decode | No operation | No operation | No operation | No operation |
| No operation | No operation (Read Program Memory) | No operation | No operation | No operation (Write TABLAT) |

TBLRD Table Read (Continued)

Example1: TBLRD *+ ;

Before Instruction
TABLAT = 55h
TBLPTR = 00A356h
MEMORY (00A356h) = 34h

After Instruction
TABLAT = 34h
TBLPTR = 00A357h

Example2: TBLRD +* ;

Before Instruction
TABLAT = AAh
TBLPTR = 01A357h
MEMORY (01A357h) = 12h
MEMORY (01A358h) = 34h

After Instruction
TABLAT = 34h
TBLPTR = 01A358h

TBLWT Table Write

Syntax: TBLWT (*, *+, *-; +*)

Operands: None

Operation: if TBLWT*,
 (TABLELAT) → Holding Register;
 TBLPTR – No Change;
 if TBLWT*+,
 (TABLELAT) → Holding Register;
 (TBLPTR) + 1 → TBLPTR;
 if TBLWT*-,
 (TABLELAT) → Holding Register;
 (TBLPTR) – 1 → TBLPTR;
 if TBLWT*+,
 (TBLPTR) + 1 → TBLPTR;
 (TABLELAT) → Holding Register;

Status Affected: None

Encoding:

| | | | |
|------|------|------|---|
| 0000 | 0000 | 0000 | 11nn nn=0 * =1 *+ =2 *- =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 (P.M.). (Refer to [Section 11.1 “Program Flash Memory”](#) for additional details on programming Flash memory.)

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.

TBLPTR[0] = 0: Least Significant Byte of Program Memory Word

TBLPTR[0] = 1: Most Significant Byte of Program Memory Word

The TBLWT instruction can modify the value of TBLPTR as follows:

- no change
- post-increment
- post-decrement
- pre-increment

Words: 1

Cycles: 2

Q Cycle Activity:

| | Q1 | Q2 | Q3 | Q4 |
|--------------|----------------------------|--------------|--------------|---|
| Decode | No operation | No operation | No operation | No operation |
| No operation | No operation (Read TABLAT) | No operation | No operation | No operation (Write to Holding Register) |

TBLWT Table Write (Continued)

Example 1: TBLWT *+;

Before Instruction

| | | |
|----------------------------|---|---------|
| TABLAT | = | 55h |
| TBLPTR | = | 00A356h |
| HOLDING REGISTER (00A356h) | = | FFh |

After Instructions (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: $0 \leq f \leq 255$
 $a \in [0,1]$

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 2-cycle instruction. If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See [Section 36.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"](#) for details.

Words: 1

Cycles: 1(2)
Note: 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|--------|-------------------|--------------|--------------|
| Decode | Read register 'f' | Process Data | No operation |

If skip:

| Q1 | Q2 | Q3 | Q4 |
|--------------|--------------|--------------|--------------|
| No operation | No operation | No operation | No operation |

If skip and followed by 2-word instruction:

| Q1 | Q2 | Q3 | Q4 |
|--------------|--------------|--------------|--------------|
| 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 = 00h,
PC = Address (ZERO)
If CNT \neq 00h,
PC = Address (NZERO)

XORLW **Exclusive OR literal with W**

Syntax: XORLW k

Operands: $0 \leq k \leq 255$

Operation: (W) .XOR. k \rightarrow 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: $0 \leq f \leq 255$
 $d \in [0,1]$
 $a \in [0,1]$

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).
 If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.
 If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See [Section 36.2.3 "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

36.2 Extended Instruction Set

In addition to the standard 75 instructions of the PIC18 instruction set, PIC18(L)F67K40 devices also provide an optional extension to the core CPU functionality. The added features include eight 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 instructions, ADDFSR and SUBFSR, each have an additional special instantiation for using FSR2. 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 [Table 36-3](#). Detailed descriptions are provided in [Section 36.2.2 “Extended Instruction Set”](#). The opcode field descriptions in [Table 36-1](#) apply to both the standard and extended PIC18 instruction sets.

Note: 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.

36.2.1 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 [Section 36.2.3.1 “Extended Instruction Syntax with Standard PIC18 Commands”](#).

Note: In the past, square brackets have been used to denote optional arguments in the PIC18 and earlier instruction sets. In this text and going forward, optional arguments are denoted by braces (“{ }”).

TABLE 36-3: EXTENSIONS TO THE PIC18 INSTRUCTION SET

| Mnemonic, Operands | Description | Cycles | 16-Bit Instruction Word | | | | Status Affected |
|---------------------------------------|---|--------|-------------------------|------|------|------|--------------------|
| | | | MSb | | LSb | | |
| ADDFSR f, k | Add literal to FSR | 1 | 1110 | 1000 | ffkk | kkkk | None |
| ADDULNK k | Add literal to FSR2 and return | 2 | 1110 | 1000 | 11kk | kkkk | None |
| CALLW | Call subroutine using WREG | 2 | 0000 | 0000 | 0001 | 0100 | None |
| MOVSF z _s , f _d | Move z _s (source) to 1st word f _d (destination) 2nd word | 2 | 1110 | 1011 | 0zzz | zzzz | None |
| MOVSS z _s , z _d | Move z _s (source) to 1st word z _d (destination) 2nd word | 2 | 1110 | 1011 | 1zzz | zzzz | None |
| PUSHL k | Store literal at FSR2, decrement FSR2 | 1 | 1110 | 1010 | kkkk | kkkk | None |
| SUBFSR f, k | Subtract literal from FSR | 1 | 1110 | 1001 | ffkk | kkkk | None |
| SUBULNK k | Subtract literal from FSR2 and return | 2 | 1110 | 1001 | 11kk | kkkk | None |

36.2.2 EXTENDED INSTRUCTION SET

| ADDFSR | Add Literal to FSR | | | | | | | | |
|-------------------|---|--------------|--------------|------|------|--------|------------------|--------------|--------------|
| Syntax: | ADDFSR f, k | | | | | | | | |
| Operands: | 0 ≤ k ≤ 63 f ∈ [0, 1, 2] | | | | | | | | |
| Operation: | FSR(f) + k → FSR(f) | | | | | | | | |
| Status Affected: | None | | | | | | | | |
| Encoding: | <table border="1" style="display: inline-table;"><tr><td>1110</td><td>1000</td><td>ffkk</td><td>kkkk</td></tr></table> | 1110 | 1000 | ffkk | kkkk | | | | |
| 1110 | 1000 | ffkk | kkkk | | | | | | |
| Description: | The 6-bit literal 'k' is added to the contents of the FSR specified by 'f'. | | | | | | | | |
| Words: | 1 | | | | | | | | |
| Cycles: | 1 | | | | | | | | |
| Q Cycle Activity: | <table border="1" style="display: inline-table;"><thead><tr><th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr></thead><tbody><tr><td>Decode</td><td>Read literal 'k'</td><td>Process Data</td><td>Write to FSR</td></tr></tbody></table> | Q1 | Q2 | Q3 | Q4 | Decode | Read literal 'k' | Process Data | Write to FSR |
| 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

| ADDLNK | Add Literal to FSR2 and Return | | | | |
|------------------|---|------|------|------|------|
| Syntax: | ADDLNK k | | | | |
| Operands: | 0 ≤ k ≤ 63 | | | | |
| Operation: | FSR2 + k → FSR2, (TOS) → PC | | | | |
| Status Affected: | None | | | | |
| Encoding: | <table border="1" style="display: inline-table;"><tr><td>1110</td><td>1000</td><td>11kk</td><td>kkkk</td></tr></table> | 1110 | 1000 | 11kk | kkkk |
| 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 = 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 FSR |
| No Operation | No Operation | No Operation | No Operation |

Example: ADDLNK 23h

Before Instruction
FSR2 = 03FFh
PC = 0100h
After Instruction
FSR2 = 0422h
PC = (TOS)

Note: 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 syntax then becomes: {label} instruction argument(s).

PIC18(L)F67K40

CALLW Subroutine Call Using WREG

Syntax: CALLW

Operands: None

Operation: (PC + 2) → TOS,
(W) → PCL,
(PCLATH) → PCH,
(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 into 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 = 001006h

TOS = address (HERE + 2)

PCLATH = 10h

PCLATU = 00h

W = 06h

MOVSF Move Indexed to f

Syntax: MOVSF [z_s], f_d

Operands: 0 ≤ z_s ≤ 127
0 ≤ f_d ≤ 4095

Operation: ((FSR2) + z_s) → f_d

Status Affected: None

Encoding:

| | | | |
|------|------|------|-------------------|
| 1110 | 1011 | 0zzz | zzzz _s |
| 1111 | ffff | ffff | ffff _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). The MOVSF 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: 2

Cycles: 2

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|--------|-------------------------------|-----------------------|---------------------------|
| Decode | Determine source addr | Determine source addr | Read source reg |
| Decode | No operation No dummy read | No operation | Write register 'f' (dest) |

Example: MOVSF [05h], REG2

Before Instruction

FSR2 = 80h

Contents of 85h = 33h

REG2 = 11h

After Instruction

FSR2 = 80h

Contents of 85h = 33h

REG2 = 33h

MOVSS Move Indexed to Indexed

Syntax: MOVSS [z_s], [z_d]

Operands: 0 ≤ z_s ≤ 127
0 ≤ z_d ≤ 127

Operation: ((FSR2) + z_s) → ((FSR2) + z_d)

Status Affected: None

Encoding:

| | | | |
|------|------|------|-------------------|
| 1110 | 1011 | 1zzz | zzzz _s |
| 1111 | xxxx | xzzz | zzzz _d |

1st word (source)
2nd word (dest.)

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 4096-byte data memory space (000h to FFFh).

The MOVSS 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. 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 addr | Determine source addr | Read source reg |
| Decode | Determine dest addr | Determine dest addr | Write to dest reg |

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),
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 'k' | Process data | Write to destination |

Example: PUSHL 08h

Before Instruction

FSR2H:FSR2L = 01ECh

Memory (01ECh) = 00h

After Instruction

FSR2H:FSR2L = 01EBh

Memory (01ECh) = 08h

PIC18(L)F67K40

SUBFSR Subtract Literal from FSR

Syntax: SUBFSR f, k
 Operands: $0 \leq k \leq 63$
 $f \in [0, 1, 2]$
 Operation: $FSR(f) - k \rightarrow FSRf$
 Status Affected: None
 Encoding:

| | | | |
|------|------|------|------|
| 1110 | 1001 | ffkk | kkkk |
|------|------|------|------|

 Description: The 6-bit literal 'k' is subtracted from the contents of the FSR specified by 'f'.
 Words: 1
 Cycles: 1
 Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|--------|-------------------|--------------|----------------------|
| Decode | Read register 'f' | Process Data | Write to destination |

Example: SUBFSR 2, 23h

Before Instruction
 FSR2 = 03FFh
 After Instruction
 FSR2 = 03DCh

SUBULNK Subtract Literal from FSR2 and Return

Syntax: SUBULNK k
 Operands: $0 \leq k \leq 63$
 Operation: $FSR2 - k \rightarrow FSR2$
 (TOS) $\rightarrow PC$
 Status Affected: None
 Encoding:

| | | | |
|------|------|------|------|
| 1110 | 1001 | 11kk | kkkk |
|------|------|------|------|

 Description: The 6-bit literal 'k' is subtracted from the contents of the 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 $f = 3$ (binary '11'); it operates only on FSR2.
 Words: 1
 Cycles: 2
 Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|--------------|-------------------|--------------|----------------------|
| Decode | Read register 'f' | 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)

36.2.3 BYTE-ORIENTED AND BIT-ORIENTED INSTRUCTIONS IN INDEXED LITERAL OFFSET MODE

Note: Enabling the PIC18 instruction set extension may cause legacy applications to behave erratically or fail entirely.

In addition to eight new commands in the extended set, enabling the extended instruction set also enables Indexed Literal Offset Addressing mode ([Section 10.7.1 “Indexed Addressing with Literal Offset”](#)). This has a significant impact on the way that 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 that use 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 in order to preserve the Stack Pointer. Users must also keep in mind the syntax requirements of the extended instruction set (see [Section 36.2.3.1 “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 on the following page to show how execution is affected. The operand conditions shown in the examples are applicable to all instructions of these types.

36.2.3.1 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.

36.2.4 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.

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 the PIC18(L)F6xK40, it is very important to consider the type of code. A large, re-entrant application that is written in 'C' and would benefit 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: $0 \leq k \leq 95$
 $d \in [0,1]$

Operation: $(W) + ((FSR2) + k) \rightarrow \text{dest}$

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 register 'f' (default).

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|--------|----------|--------------|----------------------|
| Decode | Read '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: $0 \leq f \leq 95$
 $0 \leq b \leq 7$

Operation: $1 \rightarrow ((FSR2) + k) \langle b \rangle$

Status Affected: None

Encoding:

| | | | |
|------|------|------|------|
| 1000 | bbb0 | kkkk | kkkk |
|------|------|------|------|

Description: Bit 'b' of the register indicated by FSR2, offset by the value 'k', is set.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|--------|-------------------|--------------|----------------------|
| Decode | Read register 'f' | 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: $0 \leq k \leq 95$

Operation: $FFh \rightarrow ((FSR2) + k)$

Status Affected: None

Encoding:

| | | | |
|------|------|------|------|
| 0110 | 1000 | kkkk | kkkk |
|------|------|------|------|

Description: The contents of the register indicated by FSR2, offset by 'k', are set to FFh.

Words: 1

Cycles: 1

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|--------|----------|--------------|----------------|
| Decode | Read 'k' | Process Data | Write register |

Example: SETF [OFST]

Before Instruction

| | | |
|-------------------|---|-------|
| OFST | = | 2Ch |
| FSR2 | = | 0A00h |
| Contents of 0A2Ch | = | 00h |

After Instruction

| | | |
|-------------------|---|-----|
| Contents of 0A2Ch | = | FFh |
|-------------------|---|-----|

36.2.5 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 of the PIC18(L)F6xK40 family of devices. This includes the MPLAB C18 C compiler, MPASM assembly language and MPLAB Integrated Development Environment (IDE).

When selecting a target device for software development, MPLAB 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.

37.0 DEVELOPMENT SUPPORT

The PIC[®] microcontrollers (MCU) and dsPIC[®] digital signal controllers (DSC) are supported with a full range of software and hardware development tools:

- Integrated Development Environment
 - MPLAB[®] X IDE Software
- Compilers/Assemblers/Linkers
 - MPLAB XC Compiler
 - MPASM[™] Assembler
 - MPLINK[™] Object Linker/
MPLIB[™] Object Librarian
 - MPLAB Assembler/Linker/Librarian for
Various Device Families
- Simulators
 - MPLAB X SIM Software Simulator
- Emulators
 - MPLAB REAL ICE[™] In-Circuit Emulator
- In-Circuit Debuggers/Programmers
 - MPLAB ICD 3
 - PICKit[™] 3
- Device Programmers
 - MPLAB PM3 Device Programmer
- Low-Cost Demonstration/Development Boards,
Evaluation Kits and Starter Kits
- Third-party development tools

37.1 MPLAB X Integrated Development Environment Software

The MPLAB X IDE is a single, unified graphical user interface for Microchip and third-party software, and hardware development tool that runs on Windows[®], Linux and Mac OS[®] X. Based on the NetBeans IDE, MPLAB X IDE is an entirely new IDE with a host of free software components and plug-ins for high-performance application development and debugging. Moving between tools and upgrading from software simulators to hardware debugging and programming tools is simple with the seamless user interface.

With complete project management, visual call graphs, a configurable watch window and a feature-rich editor that includes code completion and context menus, MPLAB X IDE is flexible and friendly enough for new users. With the ability to support multiple tools on multiple projects with simultaneous debugging, MPLAB X IDE is also suitable for the needs of experienced users.

Feature-Rich Editor:

- Color syntax highlighting
- Smart code completion makes suggestions and provides hints as you type
- Automatic code formatting based on user-defined rules
- Live parsing

User-Friendly, Customizable Interface:

- Fully customizable interface: toolbars, toolbar buttons, windows, window placement, etc.
- Call graph window

Project-Based Workspaces:

- Multiple projects
- Multiple tools
- Multiple configurations
- Simultaneous debugging sessions

File History and Bug Tracking:

- Local file history feature
- Built-in support for Bugzilla issue tracker

37.2 MPLAB XC Compilers

The MPLAB XC Compilers are complete ANSI C compilers for all of Microchip's 8, 16, and 32-bit MCU and DSC devices. These compilers provide powerful integration capabilities, superior code optimization and ease of use. MPLAB XC Compilers run on Windows, Linux or MAC OS X.

For easy source level debugging, the compilers provide debug information that is optimized to the MPLAB X IDE.

The free MPLAB XC Compiler editions support all devices and commands, with no time or memory restrictions, and offer sufficient code optimization for most applications.

MPLAB XC Compilers include an assembler, linker and utilities. The assembler generates relocatable object files that can then be archived or linked with other relocatable object files and archives to create an executable file. MPLAB XC Compiler uses the assembler to produce its object file. Notable features of the assembler include:

- Support for the entire device instruction set
- Support for fixed-point and floating-point data
- Command-line interface
- Rich directive set
- Flexible macro language
- MPLAB X IDE compatibility

37.3 MPASM Assembler

The MPASM Assembler is a full-featured, universal macro assembler for PIC10/12/16/18 MCUs.

The MPASM Assembler generates relocatable object files for the MPLINK Object Linker, Intel® standard HEX files, MAP files to detail memory usage and symbol reference, absolute LST files that contain source lines and generated machine code, and COFF files for debugging.

The MPASM Assembler features include:

- Integration into MPLAB X IDE projects
- User-defined macros to streamline assembly code
- Conditional assembly for multipurpose source files
- Directives that allow complete control over the assembly process

37.4 MPLINK Object Linker/ MPLIB Object Librarian

The MPLINK Object Linker combines relocatable objects created by the MPASM Assembler. It can link relocatable objects from precompiled libraries, using directives from a linker script.

The MPLIB Object Librarian manages the creation and modification of library files of precompiled code. When a routine from a library is called from a source file, only the modules that contain that routine will be linked in with the application. This allows large libraries to be used efficiently in many different applications.

The object linker/librarian features include:

- Efficient linking of single libraries instead of many smaller files
- Enhanced code maintainability by grouping related modules together
- Flexible creation of libraries with easy module listing, replacement, deletion and extraction

37.5 MPLAB Assembler, Linker and Librarian for Various Device Families

MPLAB Assembler produces relocatable machine code from symbolic assembly language for PIC24, PIC32 and dsPIC DSC devices. MPLAB XC Compiler uses the assembler to produce its object file. The assembler generates relocatable object files that can then be archived or linked with other relocatable object files and archives to create an executable file. Notable features of the assembler include:

- Support for the entire device instruction set
- Support for fixed-point and floating-point data
- Command-line interface
- Rich directive set
- Flexible macro language
- MPLAB X IDE compatibility

37.6 MPLAB X SIM Software Simulator

The MPLAB X SIM Software Simulator allows code development in a PC-hosted environment by simulating the PIC MCUs and dsPIC DSCs on an instruction level. On any given instruction, the data areas can be examined or modified and stimuli can be applied from a comprehensive stimulus controller. Registers can be logged to files for further run-time analysis. The trace buffer and logic analyzer display extend the power of the simulator to record and track program execution, actions on I/O, most peripherals and internal registers.

The MPLAB X SIM Software Simulator fully supports symbolic debugging using the MPLAB XC Compilers, and the MPASM and MPLAB Assemblers. The software simulator offers the flexibility to develop and debug code outside of the hardware laboratory environment, making it an excellent, economical software development tool.

37.7 MPLAB REAL ICE In-Circuit Emulator System

The MPLAB REAL ICE In-Circuit Emulator System is Microchip's next generation high-speed emulator for Microchip Flash DSC and MCU devices. It debugs and programs all 8, 16 and 32-bit MCU, and DSC devices with the easy-to-use, powerful graphical user interface of the MPLAB X IDE.

The emulator is connected to the design engineer's PC using a high-speed USB 2.0 interface and is connected to the target with either a connector compatible with in-circuit debugger systems (RJ-11) or with the new high-speed, noise tolerant, Low-Voltage Differential Signal (LVDS) interconnection (CAT5).

The emulator is field upgradable through future firmware downloads in MPLAB X IDE. MPLAB REAL ICE offers significant advantages over competitive emulators including full-speed emulation, run-time variable watches, trace analysis, complex breakpoints, logic probes, a ruggedized probe interface and long (up to three meters) interconnection cables.

37.8 MPLAB ICD 3 In-Circuit Debugger System

The MPLAB ICD 3 In-Circuit Debugger System is Microchip's most cost-effective, high-speed hardware debugger/programmer for Microchip Flash DSC and MCU devices. It debugs and programs PIC Flash microcontrollers and dsPIC DSCs with the powerful, yet easy-to-use graphical user interface of the MPLAB IDE.

The MPLAB ICD 3 In-Circuit Debugger probe is connected to the design engineer's PC using a high-speed USB 2.0 interface and is connected to the target with a connector compatible with the MPLAB ICD 2 or MPLAB REAL ICE systems (RJ-11). MPLAB ICD 3 supports all MPLAB ICD 2 headers.

37.9 PICkit 3 In-Circuit Debugger/Programmer

The MPLAB PICkit 3 allows debugging and programming of PIC and dsPIC Flash microcontrollers at a most affordable price point using the powerful graphical user interface of the MPLAB IDE. The MPLAB PICkit 3 is connected to the design engineer's PC using a full-speed USB interface and can be connected to the target via a Microchip debug (RJ-11) connector (compatible with MPLAB ICD 3 and MPLAB REAL ICE). The connector uses two device I/O pins and the Reset line to implement in-circuit debugging and In-Circuit Serial Programming™ (ICSP™).

37.10 MPLAB PM3 Device Programmer

The MPLAB PM3 Device Programmer is a universal, CE compliant device programmer with programmable voltage verification at VDDMIN and VDDMAX for maximum reliability. It features a large LCD display (128 x 64) for menus and error messages, and a modular, detachable socket assembly to support various package types. The ICSP cable assembly is included as a standard item. In Stand-Alone mode, the MPLAB PM3 Device Programmer can read, verify and program PIC devices without a PC connection. It can also set code protection in this mode. The MPLAB PM3 connects to the host PC via an RS-232 or USB cable. The MPLAB PM3 has high-speed communications and optimized algorithms for quick programming of large memory devices, and incorporates an MMC card for file storage and data applications.

37.11 Demonstration/Development Boards, Evaluation Kits, and Starter Kits

A wide variety of demonstration, development and evaluation boards for various PIC MCUs and dsPIC DSCs allows quick application development on fully functional systems. Most boards include prototyping areas for adding custom circuitry and provide application firmware and source code for examination and modification.

The boards support a variety of features, including LEDs, temperature sensors, switches, speakers, RS-232 interfaces, LCD displays, potentiometers and additional EEPROM memory.

The demonstration and development boards can be used in teaching environments, for prototyping custom circuits and for learning about various microcontroller applications.

In addition to the PICDEM™ and dsPICDEM™ demonstration/development board series of circuits, Microchip has a line of evaluation kits and demonstration software for analog filter design, KEELOQ® security ICs, CAN, IrDA®, PowerSmart battery management, SEEVAL® evaluation system, Sigma-Delta ADC, flow rate sensing, plus many more.

Also available are starter kits that contain everything needed to experience the specified device. This usually includes a single application and debug capability, all on one board.

Check the Microchip web page (www.microchip.com) for the complete list of demonstration, development and evaluation kits.

37.12 Third-Party Development Tools

Microchip also offers a great collection of tools from third-party vendors. These tools are carefully selected to offer good value and unique functionality.

- Device Programmers and Gang Programmers from companies, such as SoftLog and CCS
- Software Tools from companies, such as Gimpel and Trace Systems
- Protocol Analyzers from companies, such as Saleae and Total Phase
- Demonstration Boards from companies, such as MikroElektronika, Digilent® and Olimex
- Embedded Ethernet Solutions from companies, such as EZ Web Lynx, WIZnet and IPLogika®

38.0 ELECTRICAL SPECIFICATIONS

38.1 Absolute Maximum Ratings^(†)

| | |
|---|-----------------------------------|
| 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 | |
| PIC18F67K40 | -0.3V to +6.5V |
| PIC18LF67K40 | -0.3V to +4.0V |
| on $\overline{\text{MCLR}}$ pin | -0.3V to +9.0V |
| on all other pins | -0.3V to (V _{DD} + 0.3V) |
| Maximum current | |
| on V _{SS} pin ⁽¹⁾ | |
| -40°C ≤ T _A ≤ +85°C | 350 mA |
| 85°C < T _A ≤ +125°C | 120 mA |
| on V _{DD} pin ⁽¹⁾ | |
| -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 |

Note 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 [Table 38-6](#) to calculate device specifications.

2: Power dissipation is calculated as follows:

$$P_{DIS} = V_{DD} \times \{I_{DD} - \sum I_{OH}\} + \sum \{(V_{DD} - V_{OH}) \times I_{OH}\} + \sum (V_{OI} \times I_{OL})$$

† 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.

38.2 Standard Operating Conditions

The standard operating conditions for any device are defined as:

Operating Voltage: $V_{DDMIN} \leq V_{DD} \leq V_{DDMAX}$

Operating Temperature: $T_{A_MIN} \leq T_A \leq T_{A_MAX}$

V_{DD} — Operating Supply Voltage⁽¹⁾

PIC18LF67K40

| | |
|--|-------|
| V _{DDMIN} (F _{osc} ≤ 16 MHz) | +1.8V |
| V _{DDMIN} (F _{osc} ≤ 32 MHz) | +2.5V |
| V _{DDMIN} (F _{osc} ≤ 64 MHz) | +3.0V |
| V _{DDMAX} | +3.6V |

PIC18F67K40

| | |
|--|-------|
| V _{DDMIN} (F _{osc} ≤ 16 MHz) | +2.3V |
| V _{DDMIN} (F _{osc} ≤ 32 MHz) | +2.5V |
| V _{DDMIN} (F _{osc} ≤ 64 MHz) | +3.0V |
| 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 Parameter [Supply Voltage](#), DS Characteristics: Supply Voltage.

FIGURE 38-1: VOLTAGE FREQUENCY GRAPH, $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$, PIC18F67K40 ONLY

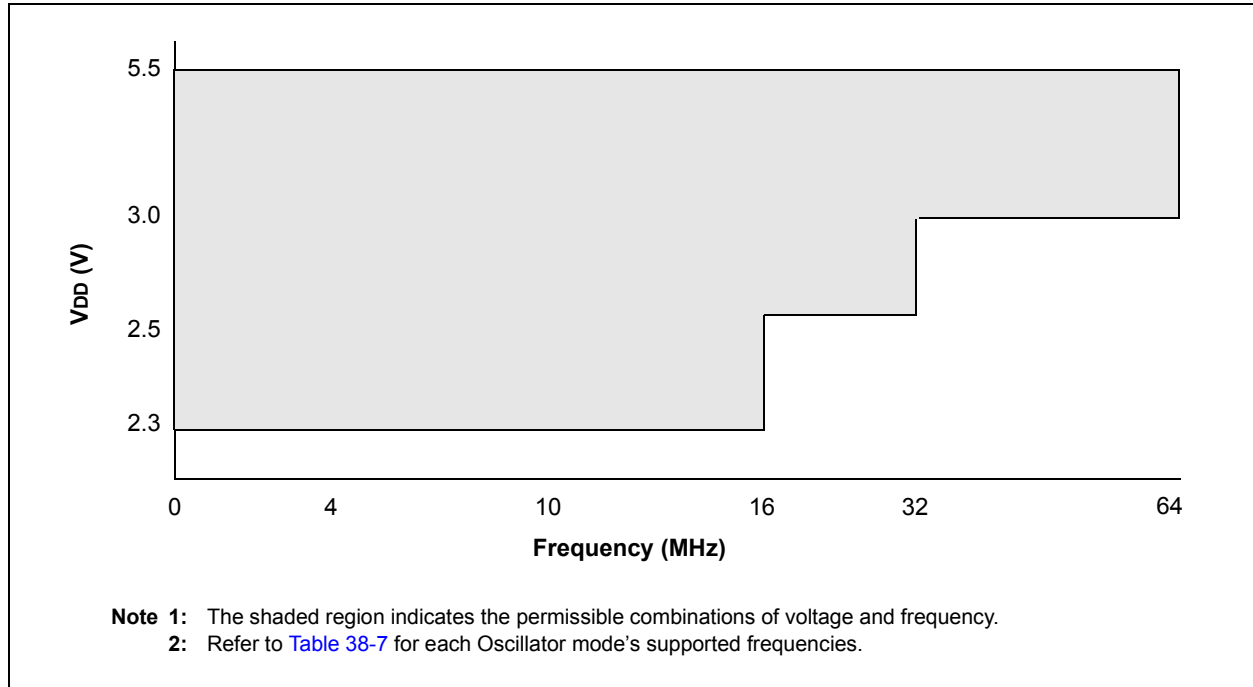
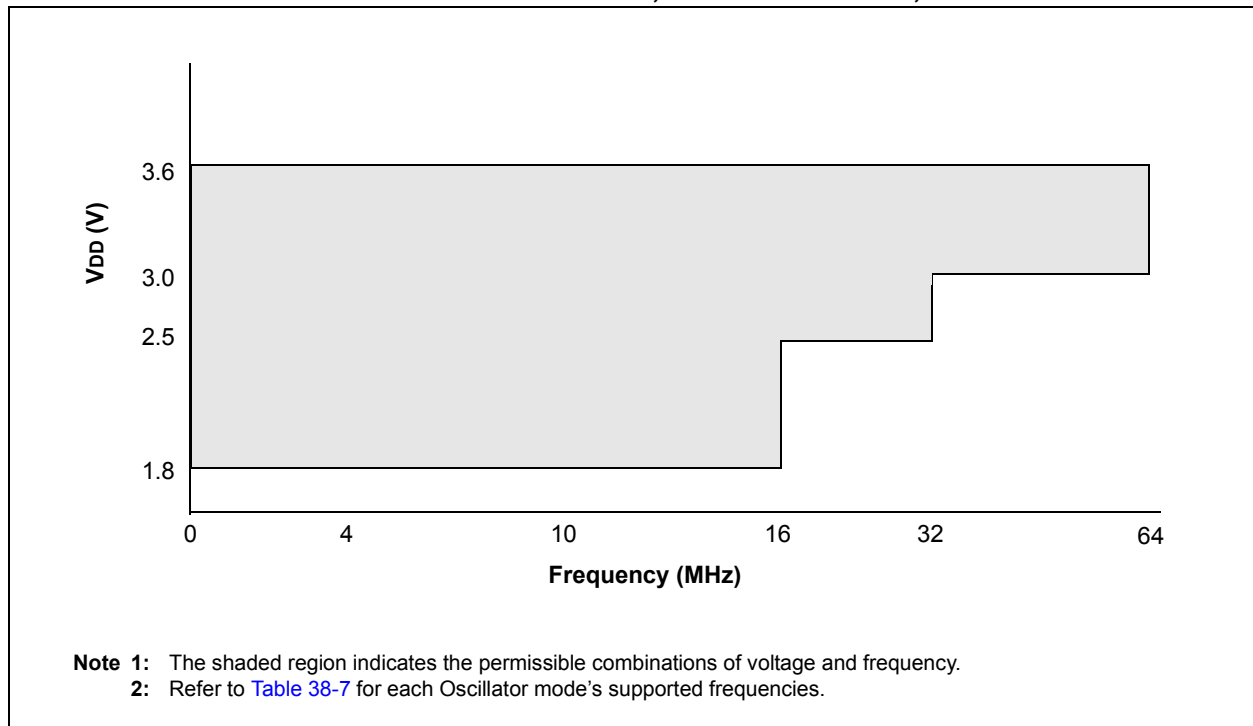


FIGURE 38-2: VOLTAGE FREQUENCY GRAPH, $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$, PIC18LF67K40 ONLY



38.3 DC Characteristics

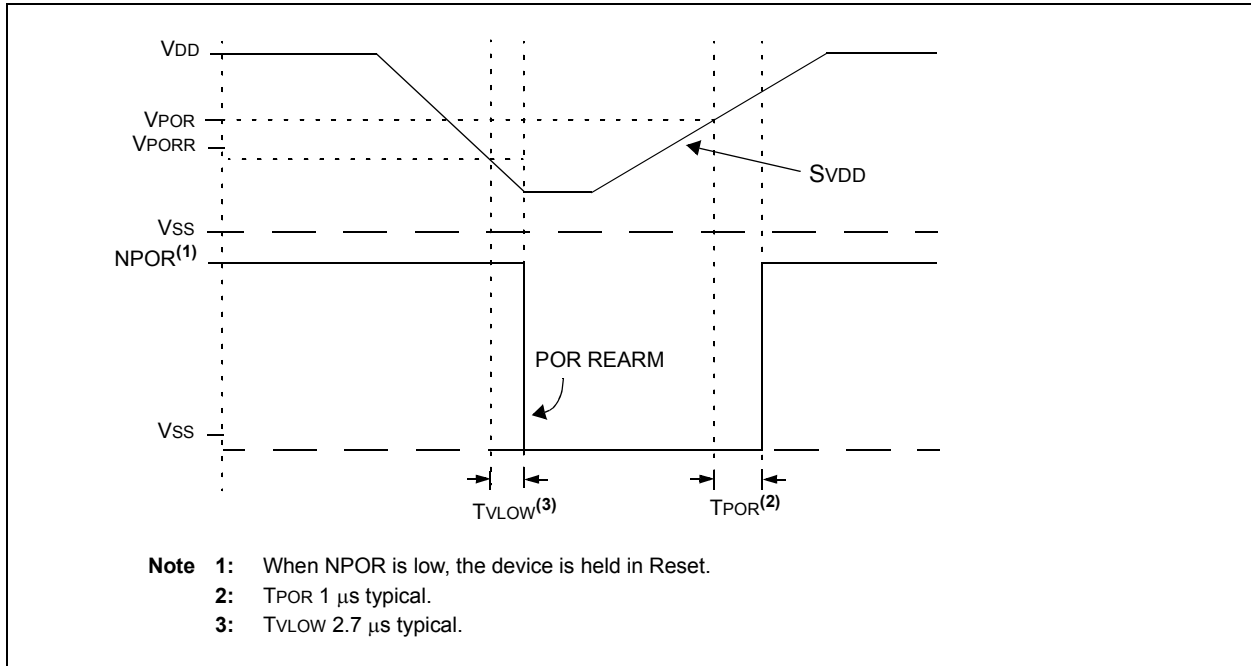
TABLE 38-1: SUPPLY VOLTAGE

| PIC18LF67K40 | | Standard Operating Conditions (unless otherwise stated) | | | | | |
|---|-------|---|------|-------|------|-------|--------------------------------------|
| PIC18F67K40 | | | | | | | |
| Param. No. | Sym. | Characteristic | Min. | Typ.† | Max. | Units | Conditions |
| Supply Voltage | | | | | | | |
| D002 | VDD | | 1.8 | — | 3.6 | V | Fosc ≤ 16 MHz |
| | | | 2.5 | — | 3.6 | V | Fosc > 16 MHz |
| | | | 3.0 | — | 3.6 | V | Fosc > 32 MHz |
| D002 | VDD | | 2.3 | — | 5.5 | V | Fosc ≤ 16 MHz |
| | | | 2.5 | — | 5.5 | V | Fosc > 16 MHz |
| | | | 3.0 | — | 5.5 | V | Fosc > 32 MHz |
| RAM Data Retention⁽¹⁾ | | | | | | | |
| D003 | VDR | | 1.5 | — | — | V | Device in Sleep mode |
| D003 | VDR | | 1.7 | — | — | V | Device in Sleep mode |
| Power-on Reset Release Voltage⁽²⁾ | | | | | | | |
| D004 | VPOR | | — | 1.6 | — | V | BOR or LPBOR disabled ⁽³⁾ |
| D004 | VPOR | | — | 1.6 | — | V | BOR or LPBOR disabled ⁽³⁾ |
| Power-on Reset Rearm Voltage⁽²⁾ | | | | | | | |
| D005 | VPORR | | — | 0.8 | — | V | BOR or LPBOR disabled ⁽³⁾ |
| D005 | VPORR | | — | 1.5 | — | V | BOR or LPBOR disabled ⁽³⁾ |
| VDD Rise Rate to ensure internal Power-on Reset signal⁽²⁾ | | | | | | | |
| D006 | SVDD | | 0.05 | — | — | V/ms | BOR or LPBOR disabled ⁽³⁾ |
| D006 | SVDD | | 0.05 | — | — | V/ms | BOR or LPBOR disabled ⁽³⁾ |

† 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:** This is the limit to which VDD can be lowered in Sleep mode without losing RAM data.
Note 2: See [Figure 38-3](#), POR and POR REARM with Slow Rising VDD.
Note 3: Please see [Table 38-11](#) for BOR and LPBOR trip point information.

FIGURE 38-3: POR AND POR REARM WITH SLOW RISING V_{DD}



PIC18(L)F67K40

TABLE 38-2: SUPPLY CURRENT (IDD)^(1,2,4)

| PIC18LF67K40 | | | Standard Operating Conditions (unless otherwise stated) | | | | | |
|--------------|------------------------------------|---|---|-------|------|-------|------------|---------------|
| PIC18F67K40 | | | | | | | | |
| Param. No. | Symbol | Device Characteristics | Min. | Typ.† | Max. | Units | Conditions | |
| | | | | | | | VDD | Note |
| D100 | IDD _{XT4} | XT = 4 MHz | — | 600 | — | μA | 3.0V | |
| D100 | IDD _{XT4} | XT = 4 MHz | — | 620 | — | μA | 3.0V | |
| D100A | IDD _{XT4} | XT = 4 MHz | — | 430 | — | μA | 3.0V | PMD's all 1's |
| D100A | IDD _{XT4} | XT = 4 MHz | — | 450 | — | μA | 3.0V | PMD's all 1's |
| D101 | IDD _{HFO16} | HFINTOSC = 16 MHz | — | 2.0 | — | mA | 3.0V | |
| D101 | IDD _{HFO16} | HFINTOSC = 16 MHz | — | 2.1 | — | mA | 3.0V | |
| D101A | IDD _{HFO16} | HFINTOSC = 16 MHz | — | 1.3 | — | mA | 3.0V | PMD's all 1's |
| D101A | IDD _{HFO16} | HFINTOSC = 16 MHz | — | 1.4 | — | mA | 3.0V | PMD's all 1's |
| D102 | IDD _{HFOPLL} | HFINTOSC = 64 MHz | — | 7.3 | — | mA | 3.0V | |
| D102 | IDD _{HFOPLL} | HFINTOSC = 64 MHz | — | 7.4 | — | mA | 3.0V | |
| D102A | IDD _{HFOPLL} | HFINTOSC = 64 MHz | — | 4.7 | — | mA | 3.0V | PMD's all 1's |
| D102A | IDD _{HFOPLL} | HFINTOSC = 64 MHz | — | 4.8 | — | mA | 3.0V | PMD's all 1's |
| D103 | IDD _{HSPLL32} | HS+PLL = 64 MHz | — | 7.3 | — | mA | 3.0V | |
| D103 | IDD _{HSPLL32} | HS+PLL = 64 MHz | — | 7.4 | — | mA | 3.0V | |
| D103A | IDD _{HSPLL32} | HS+PLL = 64 MHz | — | 4.7 | — | mA | 3.0V | PMD's all 1's |
| D103A | IDD _{HSPLL32} | HS+PLL = 64 MHz | — | 4.8 | — | mA | 3.0V | PMD's all 1's |
| D104 | IDD _{IDLE} | IDLE mode, HFINTOSC = 16 MHz | — | 1.4 | — | mA | 3.0V | |
| D104 | IDD _{IDLE} | IDLE mode, HFINTOSC = 16 MHz | — | 1.5 | — | mA | 3.0V | |
| D105 | IDD _{DOZE} ⁽³⁾ | DOZE mode, HFINTOSC = 16 MHz, Doze Ratio = 16 | — | 1.5 | — | mA | 3.0V | |
| D105 | IDD _{DOZE} ⁽³⁾ | DOZE mode, HFINTOSC = 16 MHz, Doze Ratio = 16 | — | 1.5 | — | mA | 3.0V | |

† 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 test conditions for all IDD measurements in active operation mode are: OSC1 = external square wave, from rail-to-rail; all I/O pins are outputs driven low; MCLR = VDD; 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: $IDD_{DOZE} = [IDD_{IDLE} * (N-1)/N] + IDD_{HFO16}/N$ where N = DOZE Ratio (Register 6-2).
 - 4: PMD bits are all in the default state, no modules are disabled.

PIC18(L)F67K40

TABLE 38-3: POWER-DOWN CURRENT (IPD)^(1,2)

| PIC18LF67K40 | | | | Standard Operating Conditions (unless otherwise stated) | | | | | |
|--------------|-----------|---------------------------------------|------|---|------------|-------------|-------|------------|----------------------------------|
| PIC18F67K40 | | | | Standard Operating Conditions (unless otherwise stated) VREGPM = 1 | | | | | |
| Param. No. | Symbol | Device Characteristics | Min. | Typ.† | Max. +85°C | Max. +125°C | Units | Conditions | |
| | | | | | | | | VDD | Note |
| D200 | IPD | IPD Base | — | 0.05 | 2 | 9 | μA | 3.0V | |
| D200 | IPD | IPD Base | — | 0.4 | 4 | 12 | μA | 3.0V | |
| D200A | | | — | 20 | — | — | μA | 3.0V | VREGPM = 0 |
| D201 | IPD_WDT | Low-Frequency Internal Oscillator/WDT | — | 0.4 | 3 | 10 | μA | 3.0V | |
| D201 | IPD_WDT | Low-Frequency Internal Oscillator/WDT | — | 0.6 | 5 | 13 | μA | 3.0V | |
| D202 | IPD_SOSC | Secondary Oscillator (SOSC) | — | 0.6 | 5 | 13 | μA | 3.0V | |
| D202 | IPD_SOSC | Secondary Oscillator (SOSC) | — | 0.8 | 8.5 | 15 | μA | 3.0V | |
| D203 | IPD_FVR | FVR | — | 31 | — | — | μA | 3.0V | FVRCON = 0X81 or 0x84 |
| D203 | IPD_FVR | FVR | — | 31 | — | — | μA | 3.0V | FVRCON = 0X81 or 0x84 |
| D204 | IPD_BOR | Brown-out Reset (BOR) | — | 9 | 14 | 18 | μA | 3.0V | |
| D204 | IPD_BOR | Brown-out Reset (BOR) | — | 14 | 19 | 21 | μA | 3.0V | |
| D205 | IPD_LPBOR | Low-Power Brown-out Reset (LPBOR) | — | 0.5 | — | — | μA | 3.0V | |
| D205 | IPD_LPBOR | Low-Power Brown-out Reset (LPBOR) | — | 0.7 | — | — | μA | 3.0V | |
| D206 | IPD_HLVD | High/Low Voltage Detect (HLVD) | — | 31 | — | — | μA | 3.0V | |
| D206 | IPD_HLVD | High/Low Voltage Detect (HLVD) | — | 32 | — | — | μA | 3.0V | |
| D207 | IPD_ADCA | ADC - Active | — | 250 | — | — | μA | 3.0V | ADC is converting ⁽⁴⁾ |
| D207 | IPD_ADCA | ADC - Active | — | 280 | — | — | μA | 3.0V | ADC is converting ⁽⁴⁾ |
| D208 | IPD_CMP | Comparator | — | 25 | 38 | 40 | μA | 3.0V | |
| D208 | IPD_CMP | Comparator | — | 28 | 40 | 50 | μA | 3.0V | |

† 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 peripheral current is the sum of the base IDD and the additional current consumed when this peripheral is enabled. The peripheral Δ current can be determined by subtracting the base IDD or IPD current from this limit. Max. values should 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 VSS.
 - 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 FRC.

TABLE 38-4: I/O PORTS

| Standard Operating Conditions (unless otherwise stated) | | | | | | | | |
|---|------------------|--|----------------------------|------|----------------------|-------|--|--|
| Param No. | Sym. | Characteristic | Min. | Typ† | Max. | Units | Conditions | |
| D300 D301 D302 D303 D304 D305 | V _{IL} | Input Low Voltage | | | | | | |
| | | I/O PORT: | | | | | | |
| | | with TTL buffer | — | — | 0.8 | V | 4.5V ≤ V _{DD} ≤ 5.5V | |
| | | | — | — | 0.15 V _{DD} | V | 1.8V ≤ V _{DD} ≤ 4.5V | |
| | | with Schmitt Trigger buffer | — | — | 0.2 V _{DD} | V | 2.0V ≤ V _{DD} ≤ 5.5V | |
| | | with I ² C levels | — | — | 0.3 V _{DD} | V | | |
| | | with SMBus levels | — | — | 0.8 | V | 2.7V ≤ V _{DD} ≤ 5.5V | |
| | | MCLR | — | — | 0.2 V _{DD} | V | | |
| D320 D321 D322 D323 D324 D325 | V _{IH} | Input High Voltage | | | | | | |
| | | I/O PORT: | | | | | | |
| | | with TTL buffer | 2.0 | — | — | V | 4.5V ≤ V _{DD} ≤ 5.5V | |
| | | | 0.25 V _{DD} + 0.8 | — | — | V | 1.8V ≤ V _{DD} ≤ 4.5V | |
| | | with Schmitt Trigger buffer | 0.8 V _{DD} | — | — | V | 2.0V ≤ V _{DD} ≤ 5.5V | |
| | | with I ² C levels | 0.7 V _{DD} | — | — | V | | |
| | | with SMBus levels | 2.1 | — | — | V | 2.7V ≤ V _{DD} ≤ 5.5V | |
| | | MCLR | 0.7 V _{DD} | — | — | V | | |
| D340 D341 D342 | I _{IL} | Input Leakage Current⁽¹⁾ | | | | | | |
| | | I/O Ports | — | ± 5 | ± 125 | nA | V _{SS} ≤ V _{PIN} ≤ V _{DD} , Pin at high-impedance, 85°C | |
| | | | — | ± 5 | ± 1000 | nA | V _{SS} ≤ V _{PIN} ≤ V _{DD} , Pin at high-impedance, 125°C | |
| | | MCLR ⁽²⁾ | — | ± 50 | ± 200 | nA | V _{SS} ≤ V _{PIN} ≤ V _{DD} , Pin at high-impedance, 85°C | |
| D350 | I _{PUR} | Weak Pull-up Current | | 25 | 120 | 200 | μA | V _{DD} = 3.0V, V _{PIN} = V _{SS} |
| D360 | V _{OL} | Output Low Voltage | | | | | | |
| | | I/O ports | — | — | 0.6 | V | I _{OL} = 10.0mA, V _{DD} = 3.0V | |
| D370 | V _{OH} | Output High Voltage | | | | | | |
| | | I/O ports | V _{DD} - 0.7 | — | — | V | I _{OH} = 6.0 mA, V _{DD} = 3.0V | |
| D380 | C _{IO} | All I/O pins | | — | 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.

Note 1: Negative current is defined as current sourced by the pin.

Note 2: The leakage current on the 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.

TABLE 38-5: MEMORY PROGRAMMING SPECIFICATIONS

| Standard Operating Conditions (unless otherwise stated) | | | | | | | |
|---|--------------------|---|--------------------|----------|--------------------|-------|--|
| Param No. | Sym. | Characteristic | 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 500k | 10M — | — — | E/W | -40°C ≤ T _A ≤ +60°C -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 | — | 4.0 | 5.0 | ms | |
| Program Flash Memory Specifications | | | | | | | |
| MEM30 | E _P | Flash Memory Cell Endurance | 10k | — | — | 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 Row Erase or Self-Timed Write | — | 2.0 | 2.5 | ms | |

† 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.

TABLE 38-6: THERMAL CHARACTERISTICS

Standard Operating Conditions (unless otherwise stated)

| Param No. | Sym. | Characteristic | Typ. | Units | Conditions |
|-----------|------------------|--|----------|--|--|
| TH01 | θ_{JA} | Thermal Resistance Junction to Ambient | 59 28 | $^{\circ}\text{C}/\text{W}$ $^{\circ}\text{C}/\text{W}$ | 64-pin TQFP package 64-pin QFN package |
| TH02 | θ_{JA} | Thermal Resistance Junction to Case | 18 6 | $^{\circ}\text{C}/\text{W}$ $^{\circ}\text{C}/\text{W}$ | 64-pin TQFP package 64-pin QFN package |
| TH03 | T_{JMAX} | Maximum Junction Temperature | 150 | $^{\circ}\text{C}$ | |
| TH04 | PD | Power Dissipation | — | W | $PD = P_{INTERNAL} + P_{I/O}$ ⁽³⁾ |
| TH05 | $P_{INTERNAL}$ | Internal Power Dissipation | — | W | $P_{INTERNAL} = I_{DD} \times V_{DD}$ ⁽¹⁾ |
| TH06 | P _{I/O} | I/O Power Dissipation | — | W | $P_{I/O} = \sum (I_{OL} * V_{OL}) + \sum (I_{OH} * (V_{DD} - V_{OH}))$ |
| TH07 | P _{DER} | Derated Power | — | W | $P_{DER} = P_{DMAX} (T_J - T_A) / \theta_{JA}$ ⁽²⁾ |

Note 1: I_{DD} is current to run the chip alone without driving any load on the output pins.

Note 2: T_A = Ambient Temperature, T_J = Junction Temperature

Note 3: See absolute maximum ratings for total power dissipation.

38.4 AC Characteristics

FIGURE 38-4: LOAD CONDITIONS



FIGURE 38-5: CLOCK TIMING

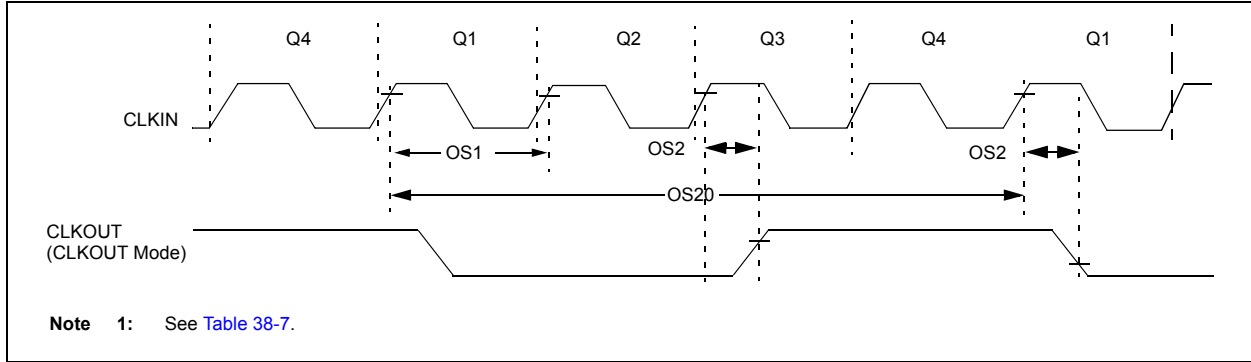


TABLE 38-7: EXTERNAL CLOCK/OSCILLATOR TIMING REQUIREMENTS

| Standard Operating Conditions (unless otherwise stated) | | | | | | | |
|---|---------------|------------------------|------|--------|------|-------|-------------------------|
| Param No. | Sym. | Characteristic | Min. | Typ† | Max. | Units | Conditions |
| ECL Oscillator | | | | | | | |
| OS1 | F_{ECL} | Clock Frequency | — | — | 500 | kHz | |
| OS2 | T_{ECL_DC} | Clock Duty Cycle | 40 | — | 60 | % | |
| ECM Oscillator | | | | | | | |
| OS3 | F_{ECM} | Clock Frequency | — | — | 4 | MHz | |
| OS4 | T_{ECM_DC} | Clock Duty Cycle | 40 | — | 60 | % | |
| ECH Oscillator | | | | | | | |
| OS5 | F_{ECH} | Clock Frequency | — | — | 32 | MHz | |
| 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 | | | | | | | |
| OS9 | F_{HS} | Clock Frequency | — | — | 20 | MHz | Note 4 |
| Secondary Oscillator | | | | | | | |
| OS10 | F_{SEC} | Clock Frequency | 32.4 | 32.768 | 33.1 | kHz | |
| System Oscillator | | | | | | | |
| OS20 | F_{OSC} | System Clock Frequency | — | — | 64 | MHz | (Note 2, 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.

- Note 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 [Section 6.0 "Power-Saving Operation Modes"](#).
- 3:** The system clock frequency (F_{OSC}) must meet the voltage requirements defined in the [Section 38.2 "Standard Operating Conditions"](#).
- 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.

TABLE 38-7: EXTERNAL CLOCK/OSCILLATOR TIMING REQUIREMENTS (CONTINUED)

Standard Operating Conditions (unless otherwise stated)

| Param No. | Sym. | Characteristic | Min. | Typ† | Max. | Units | Conditions |
|-----------|-----------------|-----------------------|------|---------------------|------|-------|------------|
| OS21 | F _{CY} | Instruction Frequency | — | F _{OSC} /4 | — | MHz | |
| OS22 | T _{CY} | Instruction Period | 62.5 | 1/F _{CY} | — | 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:** Instruction cycle period (T_{CY}) 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 [Section 6.0 “Power-Saving Operation Modes”](#).
- 3:** The system clock frequency (F_{OSC}) must meet the voltage requirements defined in the [Section 38.2 “Standard Operating Conditions”](#).
- 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.

TABLE 38-8: INTERNAL OSCILLATOR PARAMETERS⁽¹⁾

| Standard Operating Conditions (unless otherwise stated) | | | | | | | |
|---|----------|---|--------|--------------------------------|---------|------------|--------------------------|
| Param No. | Sym. | Characteristic | Min. | Typ† | Max. | Units | Conditions |
| OS50 | FHFOSC | Precision Calibrated HFINTOSC Frequency | — | 4 8 12 16 32 64 | — | MHz | (Note 2) |
| OS51 | FHFOSCLP | Low-Power Optimized HFINTOSC Frequency | — — | 1 2 | — | MHz MHz | |
| OS52 | FMFOSC | Internal Calibrated MFINTOSC Frequency | — | 500 | — | kHz | |
| OS53* | FLFOSC | Internal LFINTOSC Frequency | — | 31 | — | kHz | |
| OS54* | THFOSCST | HFINTOSC Wake-up from Sleep Start-up Time | — — | 11 50 | 20 — | μs μs | VREGPM = 0 VREGPM = 1 |
| OS56 | TLFOSCST | LFINTOSC Wake-up from Sleep Start-up Time | — | 0.2 | — | ms | |

* 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: To ensure these oscillator frequency tolerances, VDD and VSS 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 [Figure 38-6: Precision Calibrated HFINTOSC Frequency Accuracy Over Device VDD and Temperature.](#)

FIGURE 38-6: PRECISION CALIBRATED HFINTOSC FREQUENCY ACCURACY OVER DEVICE VDD AND TEMPERATURE

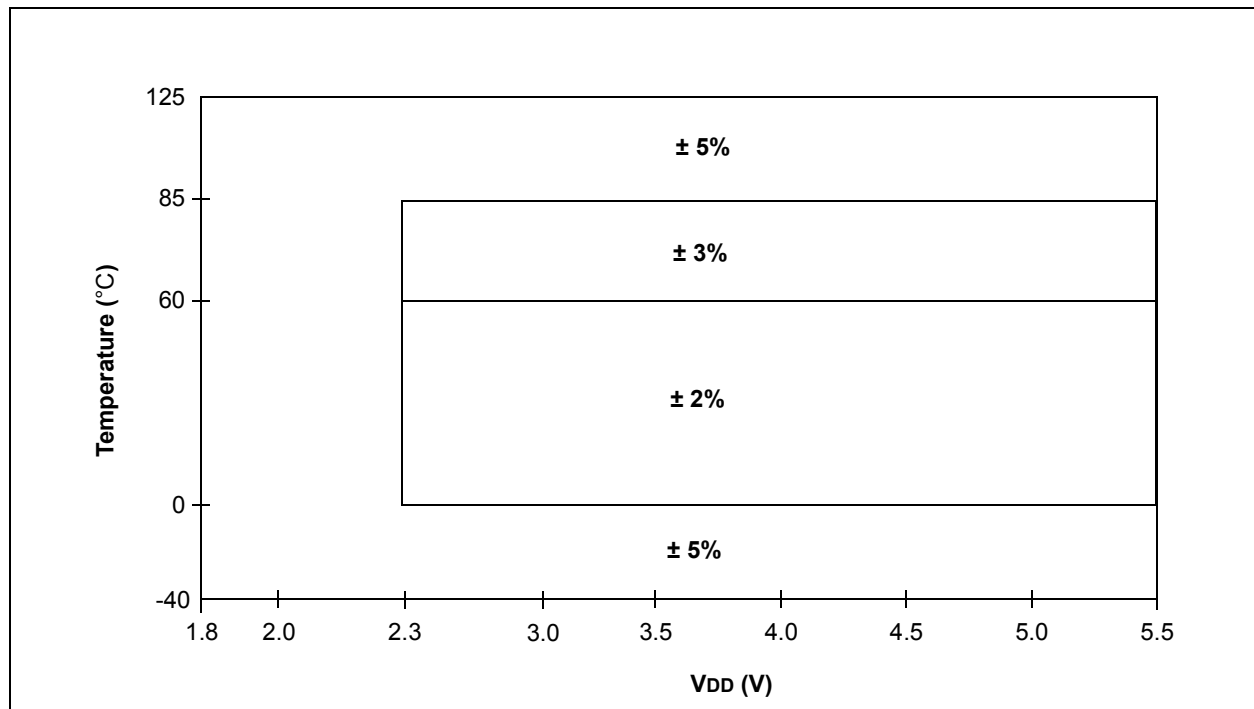


TABLE 38-9: PLL SPECIFICATIONS

| Standard Operating Conditions (unless otherwise stated) $V_{DD} \geq 2.5V$ | | | | | | | |
|--|---------|---|-------|------|------|---------|---------------|
| Param No. | Sym. | Characteristic | Min. | Typ† | Max. | Units | Conditions |
| PLL01 | FPLLIN | PLL Input Frequency Range | 4 | — | 16 | MHz | |
| PLL02 | FPLLOUT | PLL Output Frequency Range | 16 | — | 64 | MHz | Note 1 |
| PLL03 | TPLLST | PLL Lock Time from Start-up | — | 200 | — | μs | |
| PLL04 | FPLLJIT | PLL Output Frequency Stability (Jitter) | -0.25 | — | 0.25 | % | |

* These parameters are characterized but not tested.

† Data in “Typ” column is at 5V, 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 FOSC requirements listed in Parameter D002.

FIGURE 38-7: CLKOUT AND I/O TIMING

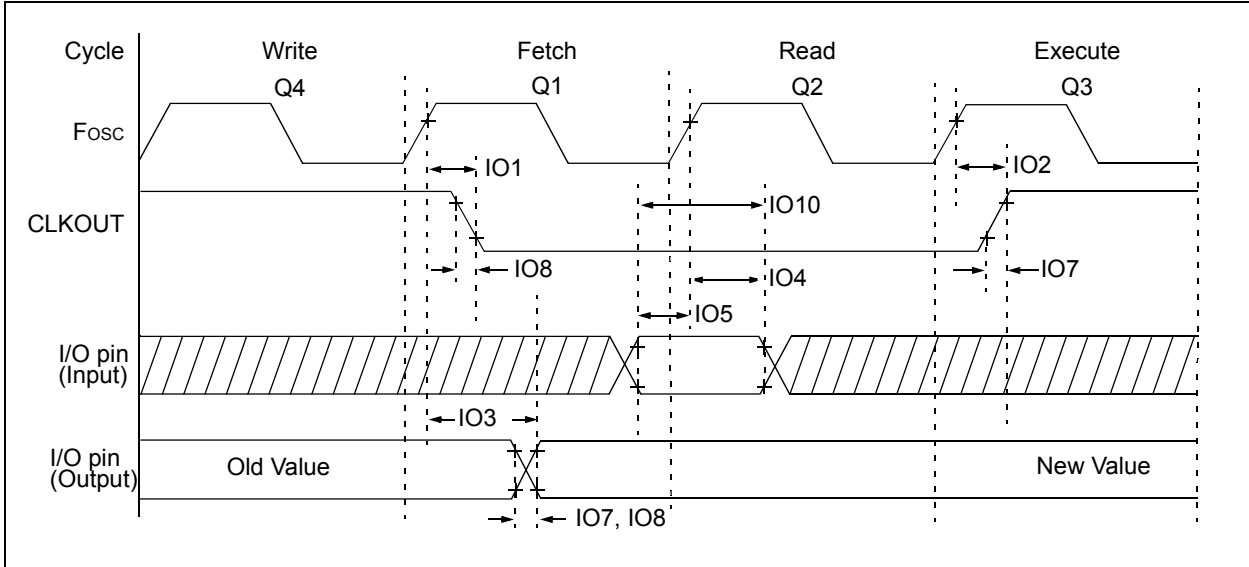


TABLE 38-10: I/O AND CLKOUT TIMING SPECIFICATIONS

| Standard Operating Conditions (unless otherwise stated) | | | | | | | |
|---|-------------------|---|------|------|------|-------|-----------------|
| Param No. | Sym. | Characteristic | Min. | Typ† | Max. | Units | Conditions |
| IO1* | $T_{CLKOUTH}$ | CLKOUT rising edge delay (rising edge Fosc (Q1 cycle) to falling edge CLKOUT) | — | — | 70 | ns | |
| IO2* | $T_{CLKOUTL}$ | CLKOUT falling edge delay (rising edge Fosc (Q3 cycle) to rising edge CLKOUT) | — | — | 72 | ns | |
| IO3* | T_{IO_VALID} | Port output valid time (rising edge Fosc (Q1 cycle) to port valid) | — | 50 | 70 | ns | |
| IO4* | T_{IO_SETUP} | Port input setup time (Setup time before rising edge Fosc – Q2 cycle) | 20 | — | — | ns | |
| IO5* | T_{IO_HOLD} | Port input hold time (Hold time after rising edge Fosc – 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.

FIGURE 38-8: RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER AND POWER-UP TIMER TIMING

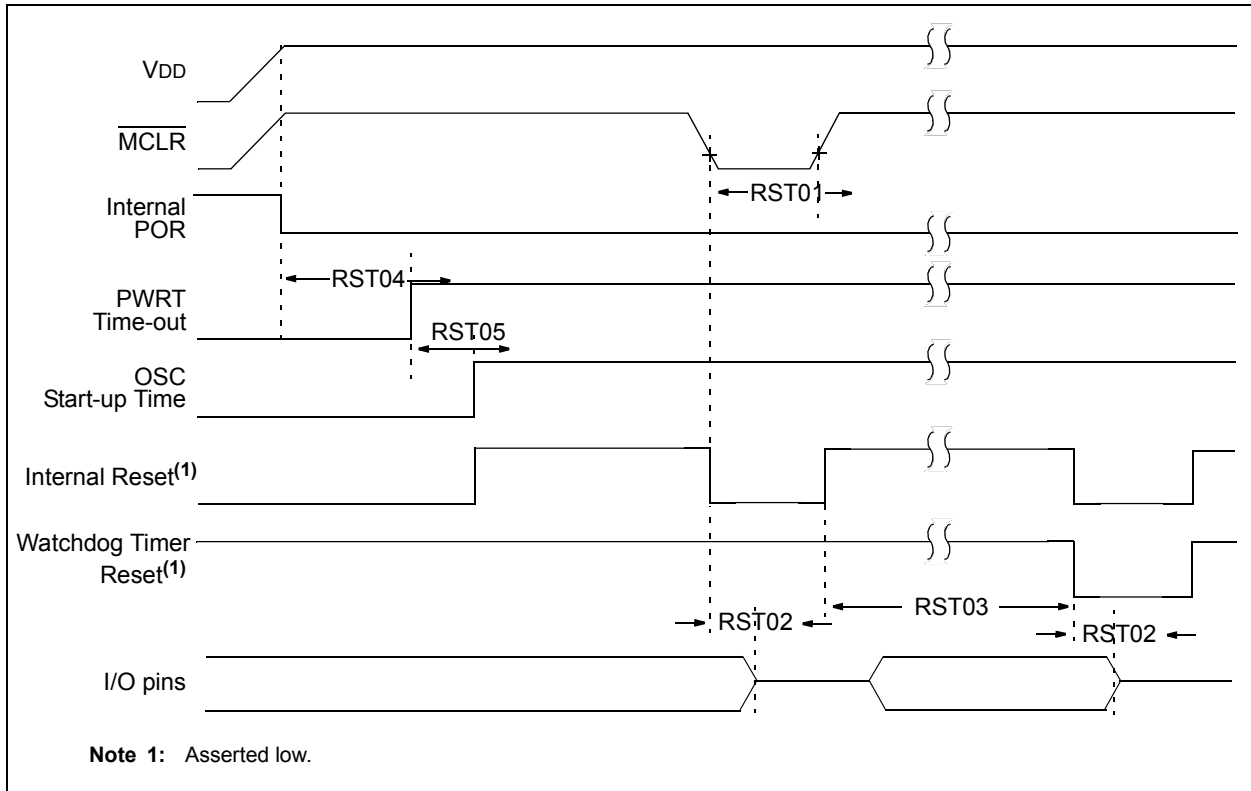
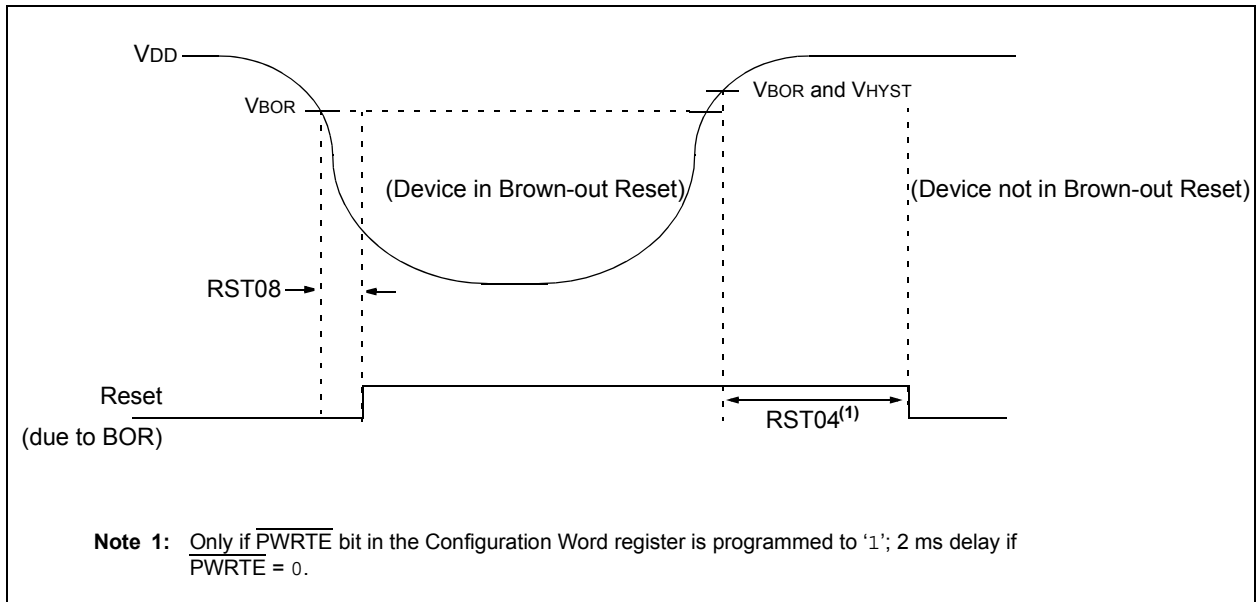


FIGURE 38-9: BROWN-OUT RESET TIMING AND CHARACTERISTICS



PIC18(L)F67K40

TABLE 38-11: RESET, WDT, OSCILLATOR START-UP TIMER, POWER-UP TIMER, BROWN-OUT RESET AND LOW-POWER BROWN-OUT RESET SPECIFICATIONS

| Standard Operating Conditions (unless otherwise stated) | | | | | | | |
|---|---------|---|----------------------------------|------------------------------------|-----------------------------------|------------------|--|
| Param No. | Sym. | Characteristic | Min. | Typ† | Max. | Units | Conditions |
| RST01* | TMCLR | MCLR Pulse Width Low to ensure Reset | 2 | — | — | μs | |
| RST02* | TIOZ | I/O high-impedance from Reset detection | — | — | 2 | μs | |
| RST03 | TWDT | Watchdog Timer Time-out Period | — | 16 | — | ms | 1:512 Prescaler |
| RST04* | TPWRT | Power-up Timer Period | — | 65 | — | ms | |
| RST05 | TOST | Oscillator Start-up Timer Period ^(1,2) | — | 1024 | — | T _{osc} | |
| RST06 | VBOR | Brown-out Reset Voltage ⁽⁴⁾ | 2.7 2.55 2.3 2.3 1.8 | 2.85 2.7 2.45 2.45 1.9 | 3.0 2.85 2.6 2.6 2.05 | V | BORV = 00 BORV = 01 BORV = 10 BORV = 11 (PIC18Fxxx) BORV = 11 (PIC18LFxxx) |
| RST07 | VBORHYS | Brown-out Reset Hysteresis | — | 40 | — | mV | |
| RST08 | TBORDC | Brown-out Reset Response Time | — | 3 | — | μs | |
| RST09 | VLPBOR | 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.

Note 1: By design, the Oscillator Start-up Timer (OST) counts the first 1024 cycles, independent of frequency.

Note 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.

TABLE 38-12: HIGH/LOW-VOLTAGE DETECT CHARACTERISTICS

| Standard Operating Conditions (unless otherwise stated) | | | | | | | |
|---|------------------|-------------------|------|------|------|-------|-------------------|
| Param. No. | Symbol | Characteristic | Min. | Typ† | Max. | Units | Conditions |
| HLVD01 | V _{DET} | Voltage Detection | — | 1.90 | — | V | HLVDSEL<3:0>=0000 |
| | | | — | 2.10 | — | V | HLVDSEL<3:0>=0001 |
| | | | — | 2.25 | — | V | HLVDSEL<3:0>=0010 |
| | | | — | 2.50 | — | V | HLVDSEL<3:0>=0011 |
| | | | — | 2.60 | — | V | HLVDSEL<3:0>=0100 |
| | | | — | 2.75 | — | V | HLVDSEL<3:0>=0101 |
| | | | — | 2.90 | — | V | HLVDSEL<3:0>=0110 |
| | | | — | 3.15 | — | V | HLVDSEL<3:0>=0111 |
| | | | — | 3.35 | — | V | HLVDSEL<3:0>=1000 |
| | | | — | 3.60 | — | V | HLVDSEL<3:0>=1001 |
| | | | — | 3.75 | — | V | HLVDSEL<3:0>=1010 |
| | | | — | 4.00 | — | V | HLVDSEL<3:0>=1011 |
| | | | — | 4.20 | — | V | HLVDSEL<3:0>=1100 |
| | | | — | 4.35 | — | V | HLVDSEL<3:0>=1101 |
| | | | — | 4.65 | — | V | HLVDSEL<3:0>=1110 |

TABLE 38-13: ANALOG-TO-DIGITAL CONVERTER (ADC) ACCURACY SPECIFICATIONS^(1,2):

| Operating Conditions (unless otherwise stated) | | | | | | | |
|--|--------------------|--|--------|------|-----------------|-------|--|
| V _{DD} = 3.0V, T _A = 25°C, T _{AD} = 1μs | | | | | | | |
| Param No. | Sym. | Characteristic | Min. | Typ† | Max. | Units | Conditions |
| AD01 | NR | Resolution | — | — | 10 | bit | |
| AD02 | EIL | Integral Error | — | ±0.1 | ±1.0 | LSb | ADCRE _{F+} = 3.0V, ADCRE _{F-} = 0V |
| AD03 | EDL | Differential Error | — | ±0.1 | ±1.0 | LSb | ADCRE _{F+} = 3.0V, ADCRE _{F-} = 0V |
| AD04 | E _{OFF} | Offset Error | — | 0.5 | ±2.0 | LSb | ADCRE _{F+} = 3.0V, ADCRE _{F-} = 0V |
| AD05 | E _{GN} | Gain Error | — | ±0.2 | ±1.0 | LSb | ADCRE _{F+} = 3.0V, ADCRE _{F-} = 0V |
| AD06 | V _{ADREF} | ADC Reference Voltage (ADREF+ - ADREF-) | 1.8 | — | V _{DD} | V | |
| AD07 | V _{AIN} | Full-Scale Range | ADREF- | — | ADREF+ | V | |
| AD08 | Z _{AIN} | Recommended Impedance of Analog Voltage Source | — | 10 | — | 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.

Note 1: Total Absolute Error is the sum of the offset, gain and integral non-linearity (INL) errors.

Note 2: The ADC conversion result never decreases with an increase in the input and has no missing codes.

Note 3: This is the impedance seen by the V_{REF} pads when the external reference pads are selected.

TABLE 38-14: ANALOG-TO-DIGITAL CONVERTER (ADC) CONVERSION TIMING SPECIFICATIONS

| Standard Operating Conditions (unless otherwise stated) | | | | | | | |
|---|------|---|------|-----------|------|-------|---|
| Param No. | Sym. | Characteristic | Min. | Typ† | Max. | Units | Conditions |
| AD20 | TAD | ADC Clock Period | 1 | — | 9 | μs | Using Fosc as the ADC clock source ADOCS = 0 |
| AD21 | | | — | 2 | — | μs | Using FRC as the ADC clock source ADOCS = 1 |
| AD22 | TCNV | Conversion Time ⁽¹⁾ | — | 11 + 3TCY | — | TAD | Set of GO/DONE bit to Clear of GO/DONE bit |
| AD23 | TACQ | Acquisition Time | — | 2 | — | μs | |
| AD24 | THCD | Sample and Hold Capacitor Disconnect Time | — | — | — | μs | FOSC-based clock source FRC-based clock source |

* 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: Does not apply for the ADCRC oscillator.

FIGURE 38-10: ADC CONVERSION TIMING (ADC CLOCK Fosc-BASED)

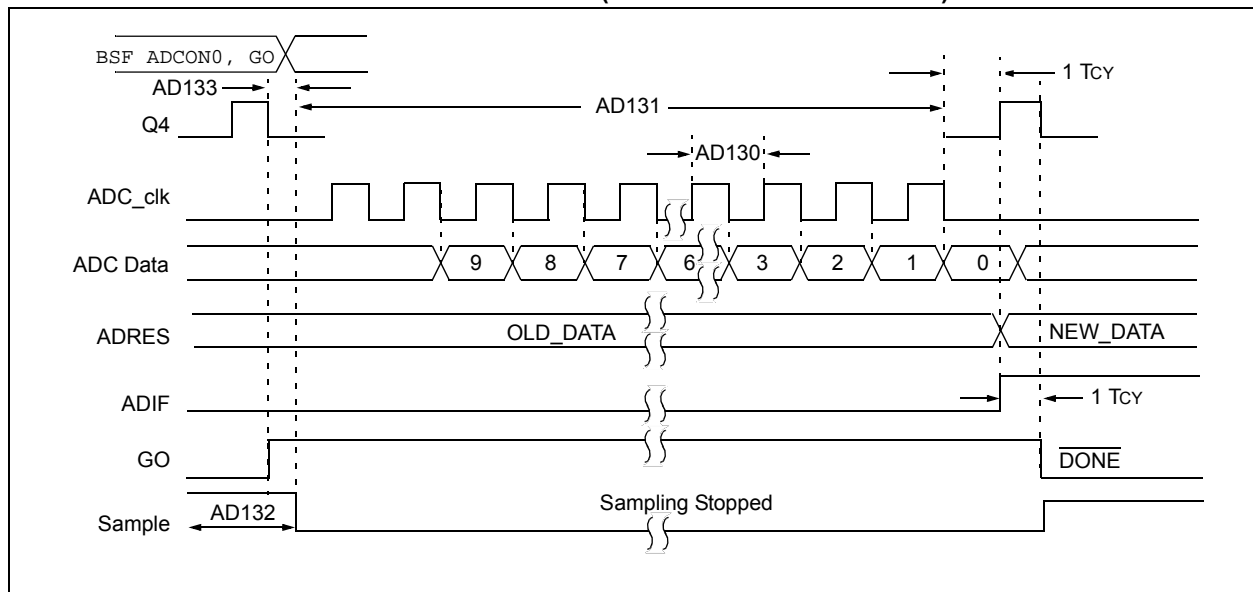


FIGURE 38-11: ADC CONVERSION TIMING (ADC CLOCK FROM ADCRC)

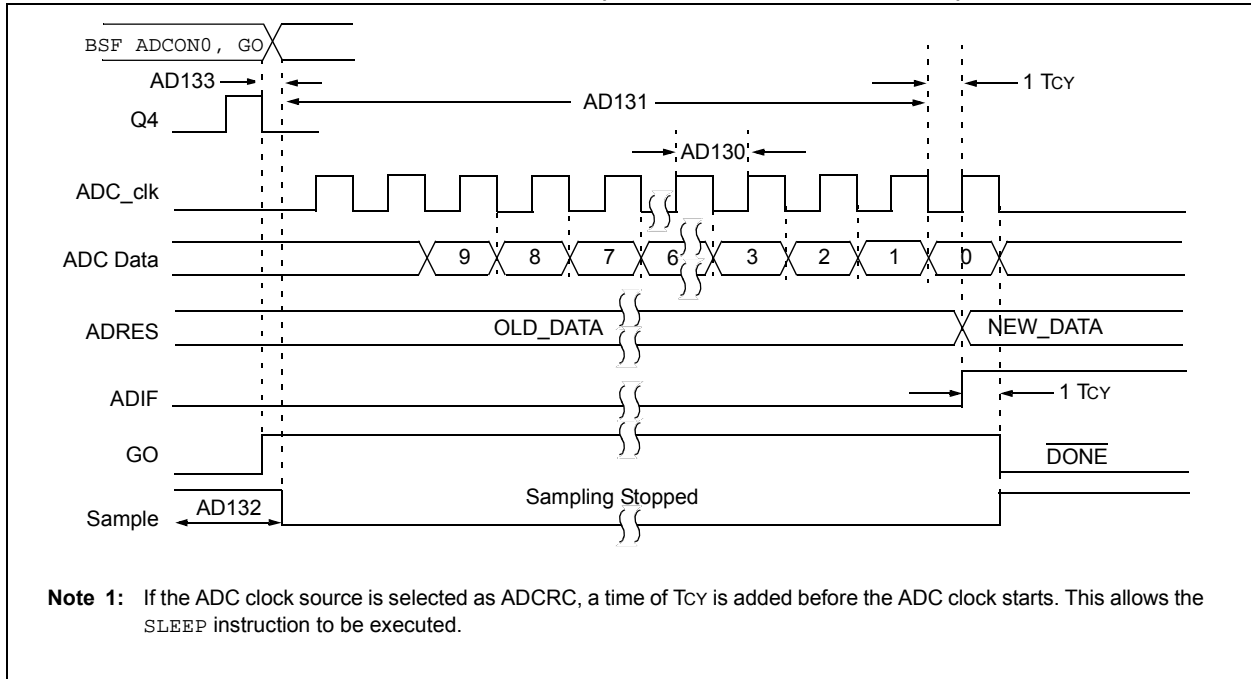


TABLE 38-15: COMPARATOR SPECIFICATIONS

| Operating Conditions (unless otherwise stated) V _{DD} = 3.0V, T _A = 25°C | | | | | | | |
|---|------------------------------------|-----------------------------------|------|------|-----------------|-------|---------------------------------------|
| Param No. | Sym. | Characteristics | Min. | Typ. | Max. | Units | Comments |
| CM01 | V _{IOFF} | Input Offset Voltage | — | — | ±30 | 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 | 15 | 25 | 35 | 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.

Note 1: Response time measured with one comparator input at V_{DD}/2, while the other input transitions from V_{SS} to V_{DD}.

2: A mode change includes changing any of the control register values, including module enable.

TABLE 38-16: 5-BIT DAC SPECIFICATIONS

| Standard Operating Conditions (unless otherwise stated) V _{DD} = 3.0V, T _A = 25°C | | | | | | | |
|--|-------------------|-------------------------------|------|---|-------|-------|----------|
| Param No. | Sym. | Characteristics | Min. | Typ. | Max. | Units | Comments |
| DSB01 | V _{LSB} | Step Size | — | (V _{DACREF+} - V _{DACREF-}) / 32 | — | V | |
| DSB01 | V _{ACC} | Absolute Accuracy | — | — | ± 0.5 | LSb | |
| DSB03* | R _{UNIT} | Unit Resistor Value | — | 5000 | — | Ω | |
| DSB04* | T _{ST} | Settling Time(¹) | — | — | 10 | μs | |

* 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: Settling time measured while DACR<4:0> transitions from '00000' to '01111'.

TABLE 38-17: FIXED VOLTAGE REFERENCE (FVR) SPECIFICATIONS

| Standard Operating Conditions (unless otherwise stated) | | | | | | | |
|---|--------------------|-------------------|------|------|------|-------|--|
| Param. No. | Symbol | Characteristic | Min. | Typ. | Max. | Units | Conditions |
| FVR01 | V _{FVR1} | 1x Gain (1.024V) | -4 | — | +4 | % | V _{DD} ≥ 2.5V, -40°C to 85°C |
| FVR02 | V _{FVR2} | 2x Gain (2.048V) | -4 | — | +4 | % | V _{DD} ≥ 2.5V, -40°C to 85°C |
| FVR03 | V _{FVR4} | 4x Gain (4.096V) | -5 | — | +5 | % | V _{DD} ≥ 4.75V, -40°C to 85°C |
| FVR04 | T _{FVRST} | FVR Start-up Time | — | 25 | — | us | |

TABLE 38-18: ZERO CROSS DETECT (ZCD) SPECIFICATIONS

| Standard Operating Conditions (unless otherwise stated) V _{DD} = 3.0V, T _A = 25°C | | | | | | | |
|--|----------------------|--------------------------------|-----|------|-----|-------|----------|
| Param. No. | Sym. | Characteristics | Min | Typ† | Max | Units | Comments |
| ZC01 | V _{PINZC} | Voltage on Zero Cross Pin | — | 0.75 | — | 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.

FIGURE 38-12: TIMER0 AND TIMER1 EXTERNAL CLOCK TIMINGS

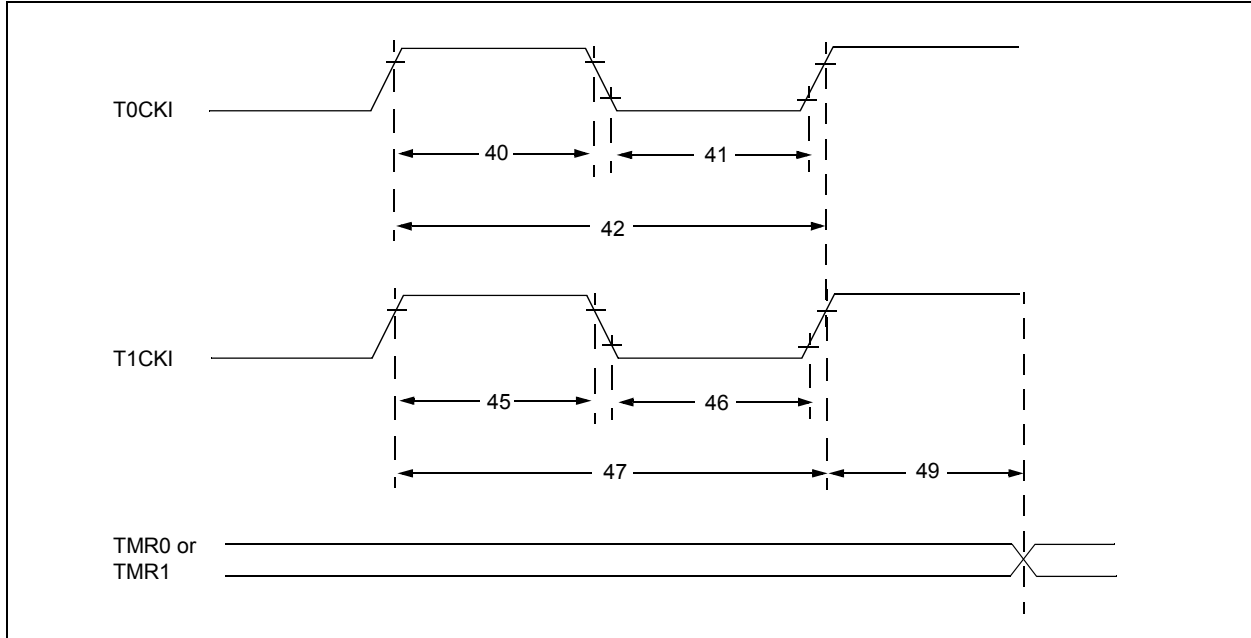


TABLE 38-19: TIMER0 AND TIMER1 EXTERNAL CLOCK REQUIREMENTS

| Standard Operating Conditions (unless otherwise stated) | | | | | | | | |
|--|-----------|---|-----------------------------|--|------|-------------|-------|---------------------|
| Operating Temperature $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ | | | | | | | | |
| Param No. | Sym. | Characteristic | | Min. | Typ† | Max. | Units | Conditions |
| 40* | Tt0H | T0CKI High Pulse Width | No Prescaler | $0.5 T_{CY} + 20$ | — | — | ns | |
| | | | With Prescaler | 10 | — | — | ns | |
| 41* | Tt0L | T0CKI Low Pulse Width | No Prescaler | $0.5 T_{CY} + 20$ | — | — | ns | |
| | | | With Prescaler | 10 | — | — | ns | |
| 42* | Tt0P | T0CKI Period | | Greater of: 20 or $\frac{T_{CY} + 40}{N}$ | — | — | ns | N = prescale value |
| 45* | Tt1H | T1CKI High Time | Synchronous, No Prescaler | $0.5 T_{CY} + 20$ | — | — | ns | |
| | | | Synchronous, with Prescaler | 15 | — | — | ns | |
| | | | Asynchronous | 30 | — | — | ns | |
| 46* | Tt1L | T1CKI Low Time | Synchronous, No Prescaler | $0.5 T_{CY} + 20$ | — | — | ns | |
| | | | Synchronous, with Prescaler | 15 | — | — | ns | |
| | | | Asynchronous | 30 | — | — | ns | |
| 47* | Tt1P | T1CKI Input Period | Synchronous | Greater of: 30 or $\frac{T_{CY} + 40}{N}$ | — | — | ns | N = prescale value |
| | | | Asynchronous | 60 | — | — | ns | |
| 49* | TCKEZTMR1 | 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 38-13: CAPTURE/COMPARE/PWM TIMINGS (CCP)

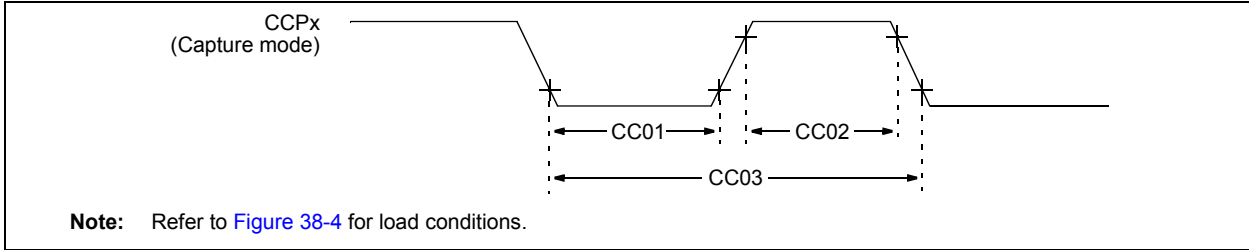


TABLE 38-20: CAPTURE/COMPARE/PWM REQUIREMENTS (CCP)

| Standard Operating Conditions (unless otherwise stated) | | | | | | | | |
|--|------|----------------------|----------------|--------------------------|------|------|-------|--------------------|
| Operating Temperature $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ | | | | | | | | |
| Param No. | Sym. | Characteristic | | Min. | Typ† | Max. | Units | Conditions |
| CC01* | TccL | CCPx Input Low Time | No Prescaler | $0.5T_{CY} + 20$ | — | — | ns | |
| | | | With Prescaler | 20 | — | — | ns | |
| CC02* | TccH | CCPx Input High Time | No Prescaler | $0.5T_{CY} + 20$ | — | — | ns | |
| | | | With Prescaler | 20 | — | — | ns | |
| CC03* | TccP | CCPx Input Period | | $\frac{3T_{CY} + 40}{N}$ | — | — | 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 38-14: EUSART SYNCHRONOUS TRANSMISSION (MASTER/SLAVE) TIMING

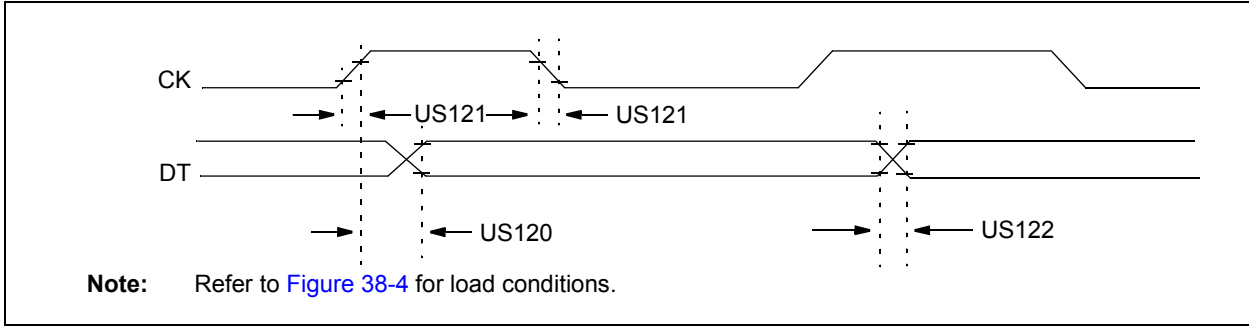


TABLE 38-21: EUSART SYNCHRONOUS TRANSMISSION REQUIREMENTS

| Standard Operating Conditions (unless otherwise stated) | | | | | | |
|---|----------|--|------|------|-------|------------------------------|
| Param. No. | Symbol | Characteristic | Min. | Max. | Units | Conditions |
| US120 | TckH2DTV | SYNC XMIT (Master and Slave) Clock high to data-out valid | — | 80 | ns | $3.0V \leq V_{DD} \leq 5.5V$ |
| | | | — | 100 | ns | $1.8V \leq V_{DD} \leq 5.5V$ |
| US121 | TckRF | Clock out rise time and fall time (Master mode) | — | 45 | ns | $3.0V \leq V_{DD} \leq 5.5V$ |
| | | | — | 50 | ns | $1.8V \leq V_{DD} \leq 5.5V$ |
| US122 | TDTRF | Data-out rise time and fall time | — | 45 | ns | $3.0V \leq V_{DD} \leq 5.5V$ |
| | | | — | 50 | ns | $1.8V \leq V_{DD} \leq 5.5V$ |

FIGURE 38-15: EUSART SYNCHRONOUS RECEIVE (MASTER/SLAVE) TIMING

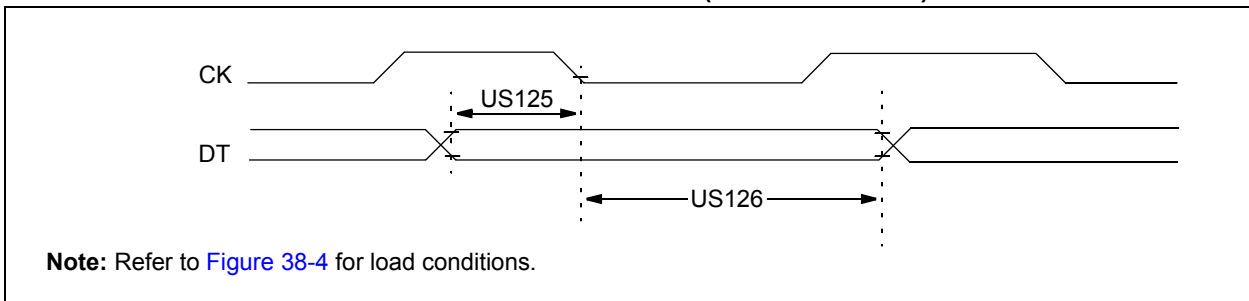


TABLE 38-22: EUSART SYNCHRONOUS RECEIVE REQUIREMENTS

| Standard Operating Conditions (unless otherwise stated) | | | | | | |
|---|----------|--|------|------|-------|------------|
| Param. No. | Symbol | Characteristic | Min. | Max. | Units | Conditions |
| US125 | TDTV2CKL | SYNC RCV (Master and Slave) Data-setup before CK ↓ (DT hold time) | 10 | — | ns | |
| US126 | TckL2DTL | Data-hold after CK ↓ (DT hold time) | 15 | — | ns | |

FIGURE 38-16: SPI MASTER MODE TIMING (CKE = 0, SMP = 0)

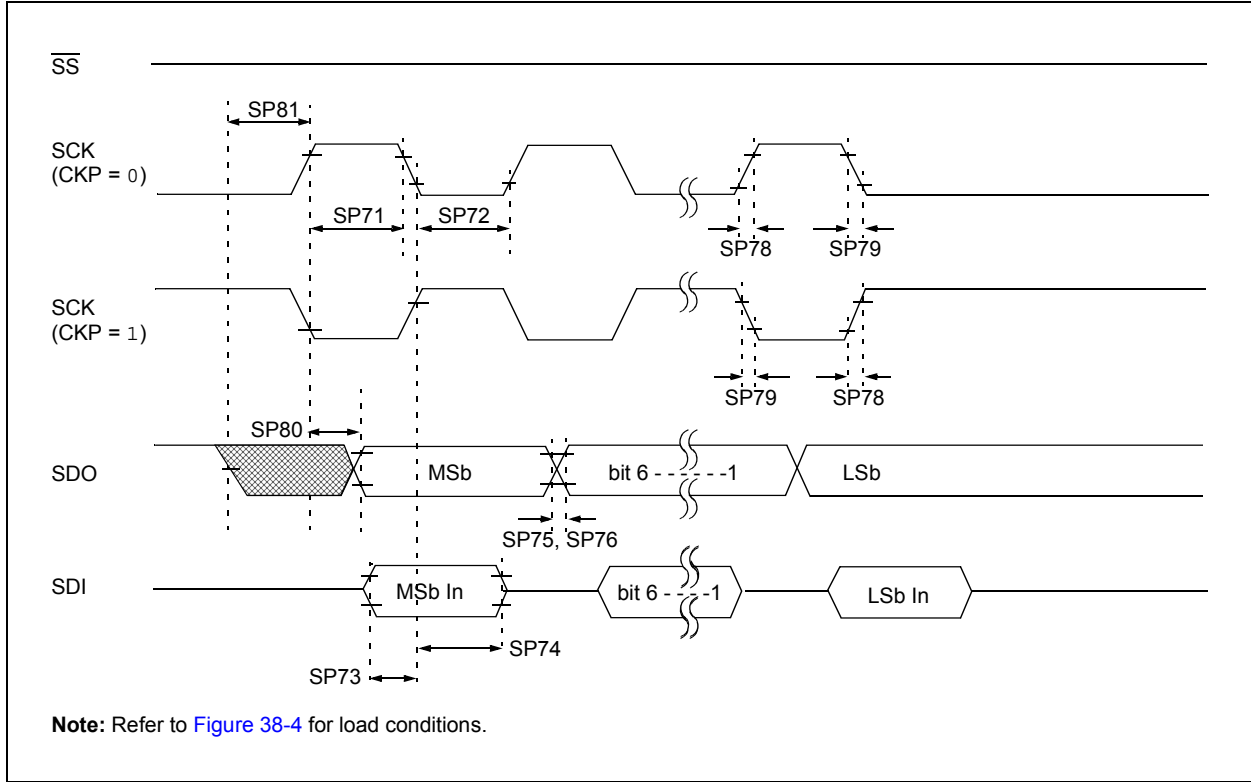


FIGURE 38-17: SPI MASTER MODE TIMING (CKE = 1, SMP = 1)

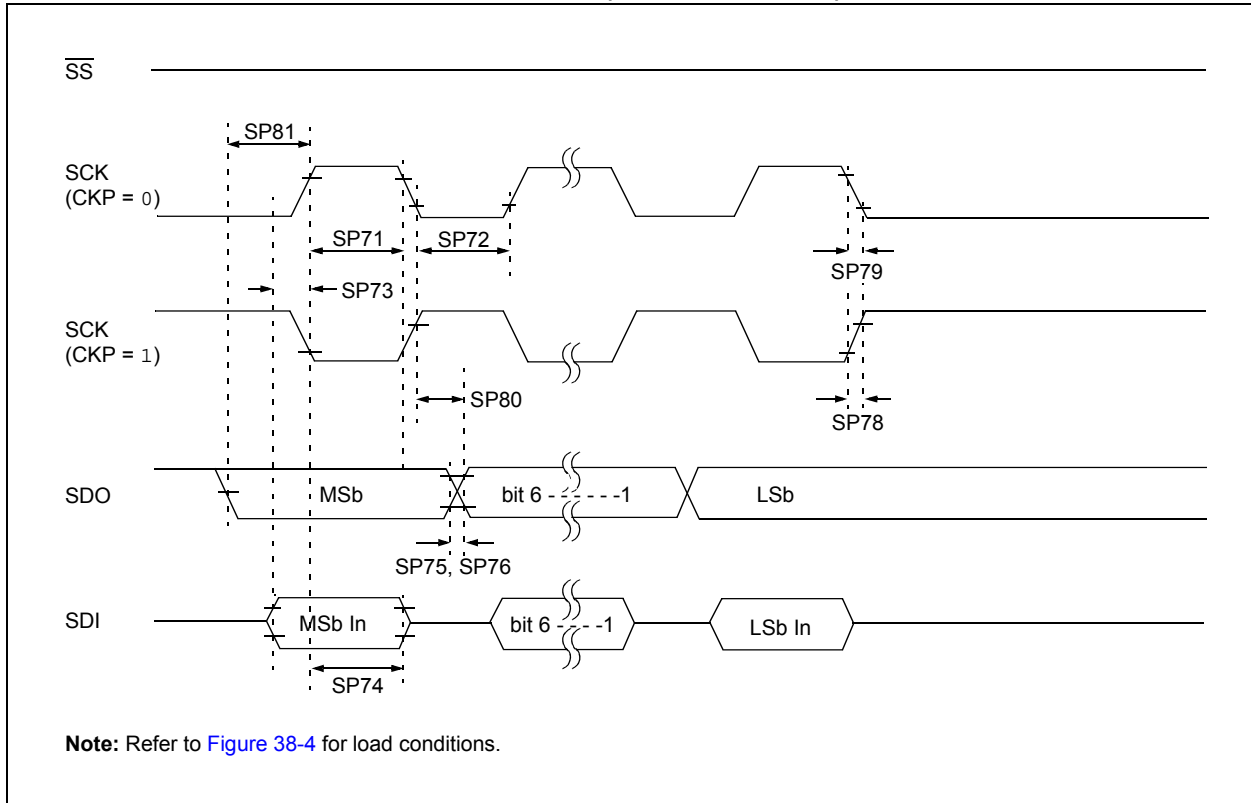


FIGURE 38-18: SPI SLAVE MODE TIMING (CKE = 0)

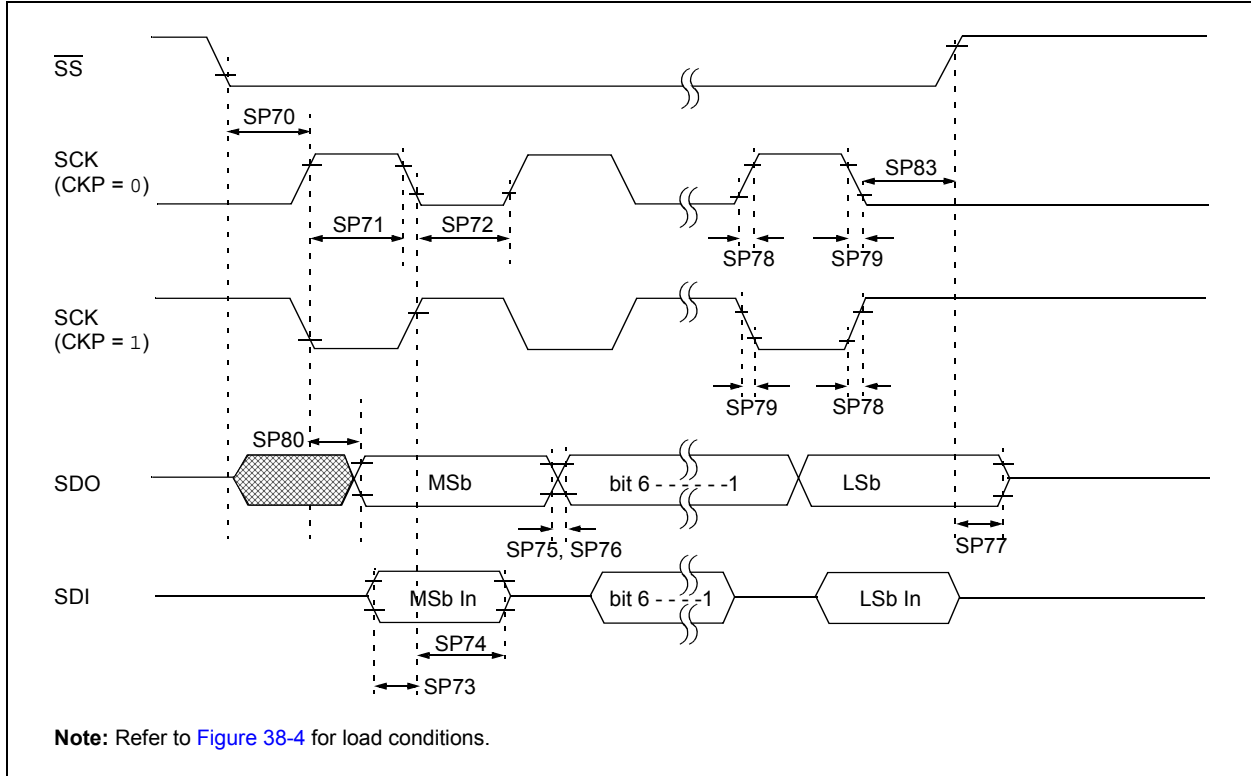


FIGURE 38-19: SPI SLAVE MODE TIMING (CKE = 1)

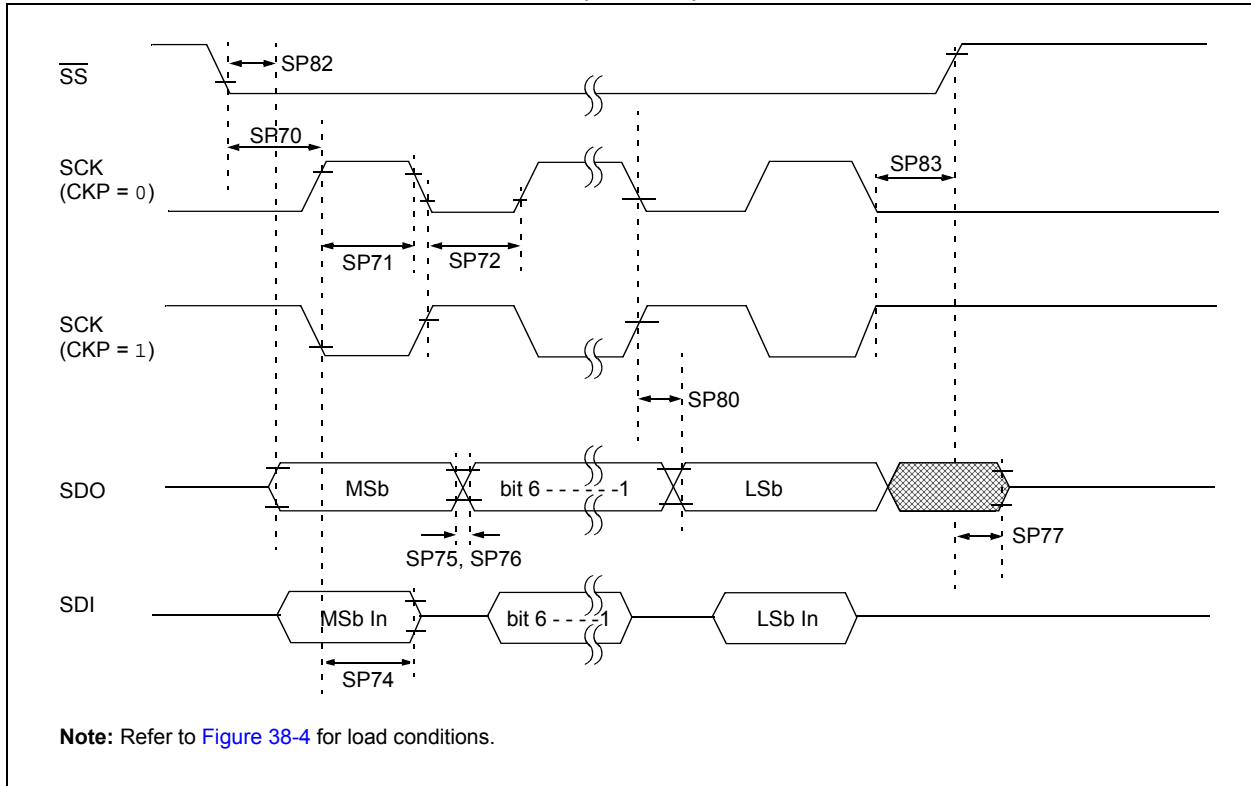


TABLE 38-23: SPI MODE REQUIREMENTS

| Standard Operating Conditions (unless otherwise stated) | | | | | | | |
|---|-----------------------|---|---------------------|------|------|-------|------------------------------|
| Param No. | Symbol | Characteristic | Min. | Typ† | Max. | Units | Conditions |
| SP70* | TssL2scH, TssL2scL | $\overline{SS}\downarrow$ to SCK \downarrow or SCK \uparrow input | $2.25 \cdot T_{CY}$ | — | — | ns | |
| SP71* | TscH | SCK input high time (Slave mode) | $T_{CY} + 20$ | — | — | ns | |
| SP72* | TscL | SCK input low time (Slave mode) | $T_{CY} + 20$ | — | — | ns | |
| SP73* | TdIV2scH, TdIV2scL | Setup time of SDI data input to SCK edge | 100 | — | — | ns | |
| SP74* | Tsch2dIL, TscL2dIL | Hold time of SDI data input to SCK edge | 100 | — | — | ns | |
| SP75* | TdoR | SDO data output rise time | — | 10 | 25 | ns | $3.0V \leq V_{DD} \leq 5.5V$ |
| | | | — | 25 | 50 | ns | $1.8V \leq V_{DD} \leq 5.5V$ |
| SP76* | TdoF | SDO data output fall time | — | 10 | 25 | ns | |
| SP77* | TssH2doZ | $\overline{SS}\uparrow$ to SDO output high-impedance | 10 | — | 50 | ns | |
| SP78* | TscR | SCK output rise time (Master mode) | — | 10 | 25 | ns | $3.0V \leq V_{DD} \leq 5.5V$ |
| | | | — | 25 | 50 | ns | $1.8V \leq V_{DD} \leq 5.5V$ |
| SP79* | TscF | SCK output fall time (Master mode) | — | 10 | 25 | ns | |
| SP80* | Tsch2doV, TscL2doV | SDO data output valid after SCK edge | — | — | 50 | ns | $3.0V \leq V_{DD} \leq 5.5V$ |
| | | | — | — | 145 | ns | $1.8V \leq V_{DD} \leq 5.5V$ |
| SP81* | TdoV2scH, TdoV2scL | SDO data output setup to SCK edge | $1 T_{CY}$ | — | — | ns | |
| SP82* | TssL2doV | SDO data output valid after $\overline{SS}\downarrow$ edge | — | — | 50 | ns | |
| SP83* | Tsch2ssH, TscL2ssH | $\overline{SS}\uparrow$ after SCK edge | $1.5 T_{CY} + 40$ | — | — | 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.

FIGURE 38-20: I²C BUS START/STOP BITS TIMING

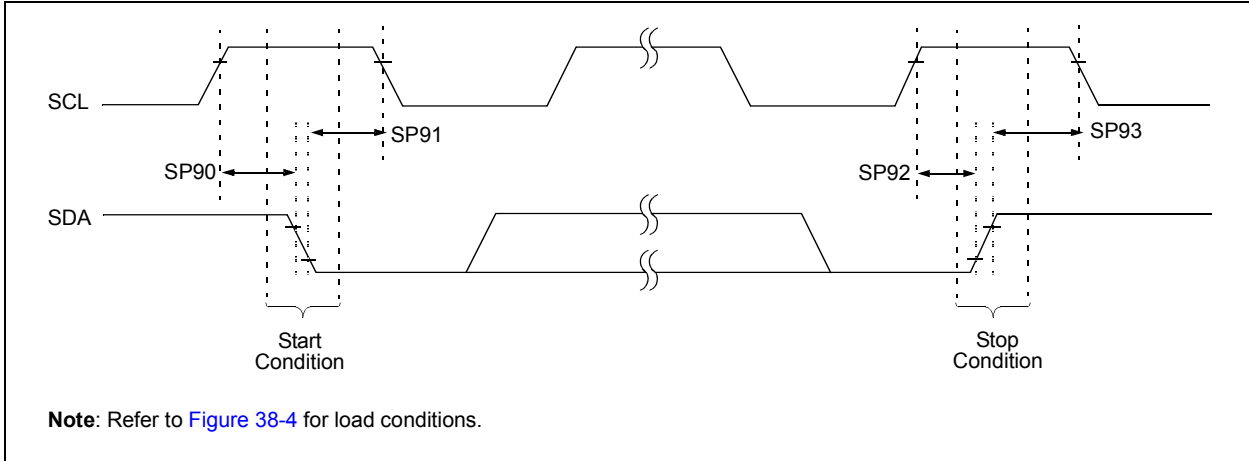


TABLE 38-24: I²C BUS START/STOP BITS REQUIREMENTS

| Standard Operating Conditions (unless otherwise stated) | | | | | | | | |
|---|---------|-----------------|--------------|------|-----|------|-------|---|
| Param No. | Symbol | Characteristic | | Min. | Typ | Max. | Units | Conditions |
| SP90* | TSU:STA | Start condition | 100 kHz mode | 4700 | — | — | ns | Only relevant for Repeated Start condition |
| | | Setup time | 400 kHz mode | 600 | — | — | | |
| SP91* | THD:STA | Start condition | 100 kHz mode | 4000 | — | — | ns | After this period, the first clock pulse is generated |
| | | Hold time | 400 kHz mode | 600 | — | — | | |
| SP92* | TSU:STO | Stop condition | 100 kHz mode | 4700 | — | — | ns | |
| | | Setup time | 400 kHz mode | 600 | — | — | | |
| SP93 | THD:STO | Stop condition | 100 kHz mode | 4000 | — | — | ns | |
| | | Hold time | 400 kHz mode | 600 | — | — | | |

* These parameters are characterized but not tested.

FIGURE 38-21: I²C BUS DATA TIMING

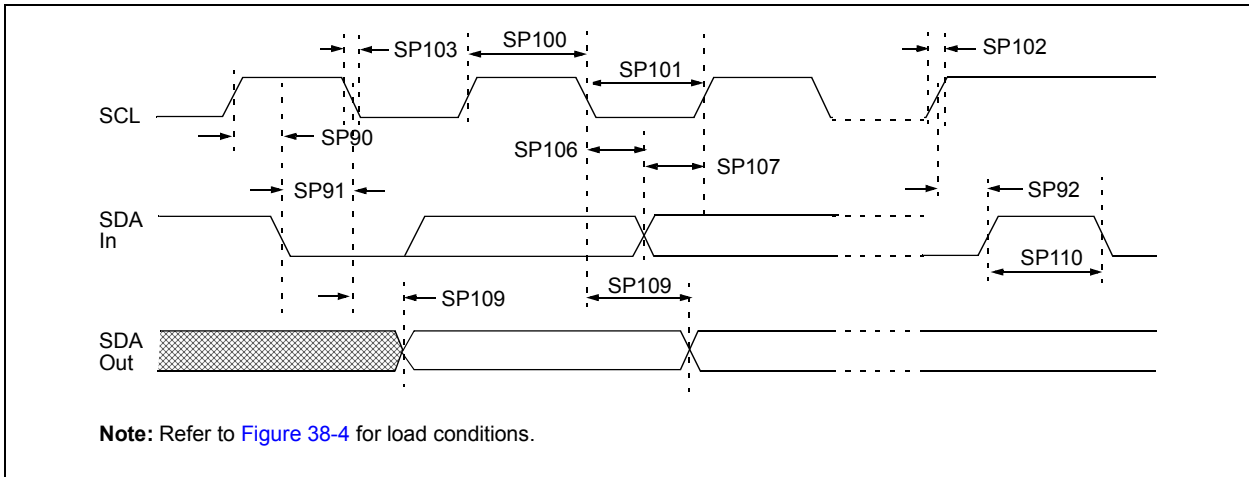


TABLE 38-25: I²C BUS DATA REQUIREMENTS

| Standard Operating Conditions (unless otherwise stated) | | | | | | | |
|---|----------------|-------------------------|--------------|------------------------|-------|------------|---|
| Param. No. | Symbol | Characteristic | Min. | Max. | Units | Conditions | |
| SP100* | THIGH | Clock high time | 100 kHz mode | 4.0 | — | μs | Device must operate at a minimum of 1.5 MHz |
| | | | 400 kHz mode | 0.6 | — | μs | Device must operate at a minimum of 10 MHz |
| | | | SSP module | 1.5T _{cy} | — | | |
| SP101* | TLOW | Clock low time | 100 kHz mode | 4.7 | — | μs | Device must operate at a minimum of 1.5 MHz |
| | | | 400 kHz mode | 1.3 | — | μs | Device must operate at a minimum of 10 MHz |
| | | | SSP module | 1.5T _{cy} | — | | |
| SP102* | TR | SDA and SCL rise time | 100 kHz mode | — | 1000 | ns | |
| | | | 400 kHz mode | 20 + 0.1C _B | 300 | ns | C _B is specified to be from 10-400 pF |
| SP103* | TF | SDA and SCL fall time | 100 kHz mode | — | 250 | ns | |
| | | | 400 kHz mode | 20 + 0.1C _B | 250 | ns | C _B is specified to be from 10-400 pF |
| SP106* | THD:DAT | Data input hold time | 100 kHz mode | 0 | — | ns | |
| | | | 400 kHz mode | 0 | 0.9 | μs | |
| SP107* | TSU:DAT | Data input setup time | 100 kHz mode | 250 | — | ns | (Note 2) |
| | | | 400 kHz mode | 100 | — | ns | |
| SP109* | TAA | Output valid from clock | 100 kHz mode | — | 3500 | ns | (Note 1) |
| | | | 400 kHz mode | — | — | ns | |
| SP110* | TBUF | Bus free time | 100 kHz mode | 4.7 | — | μs | Time the bus must be free before a new transmission can start |
| | | | 400 kHz mode | 1.3 | — | μs | |
| SP111 | C _B | Bus capacitive loading | — | 400 | pF | | |

* These parameters are characterized but not tested.

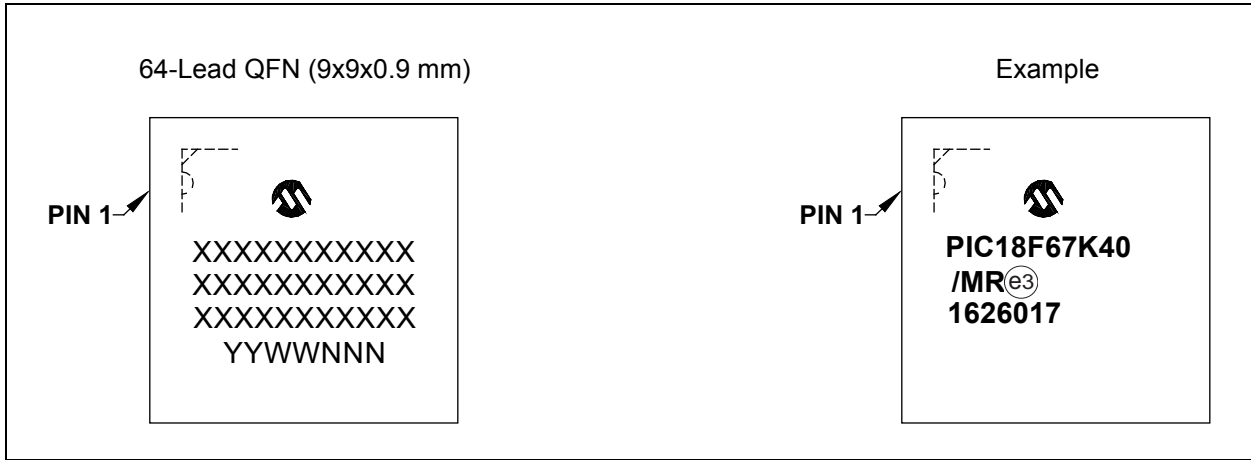
- Note 1:** As a transmitter, the device must provide this internal minimum delay time to bridge the undefined region (min. 300 ns) of the falling edge of SCL to avoid unintended generation of Start or Stop conditions.
- 2:** A Fast mode (400 kHz) I²C bus device can be used in a Standard mode (100 kHz) I²C bus system, but the requirement TSU:DAT ≥ 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. + TSU:DAT = 1000 + 250 = 1250 ns (according to the Standard mode I²C bus specification), before the SCL line is released.

39.0 DC AND AC CHARACTERISTICS GRAPHS AND TABLES

Graphs and tables are not available at this time.

40.0 PACKAGING INFORMATION

Package Marking Information



64-Lead TQFP (10x10x1 mm)

Example



| | | |
|----------------|--------|--|
| 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) |
| | * | This package is Pb-free. The Pb-free JEDEC designator ((e3)) can be found on the outer packaging for this package. |

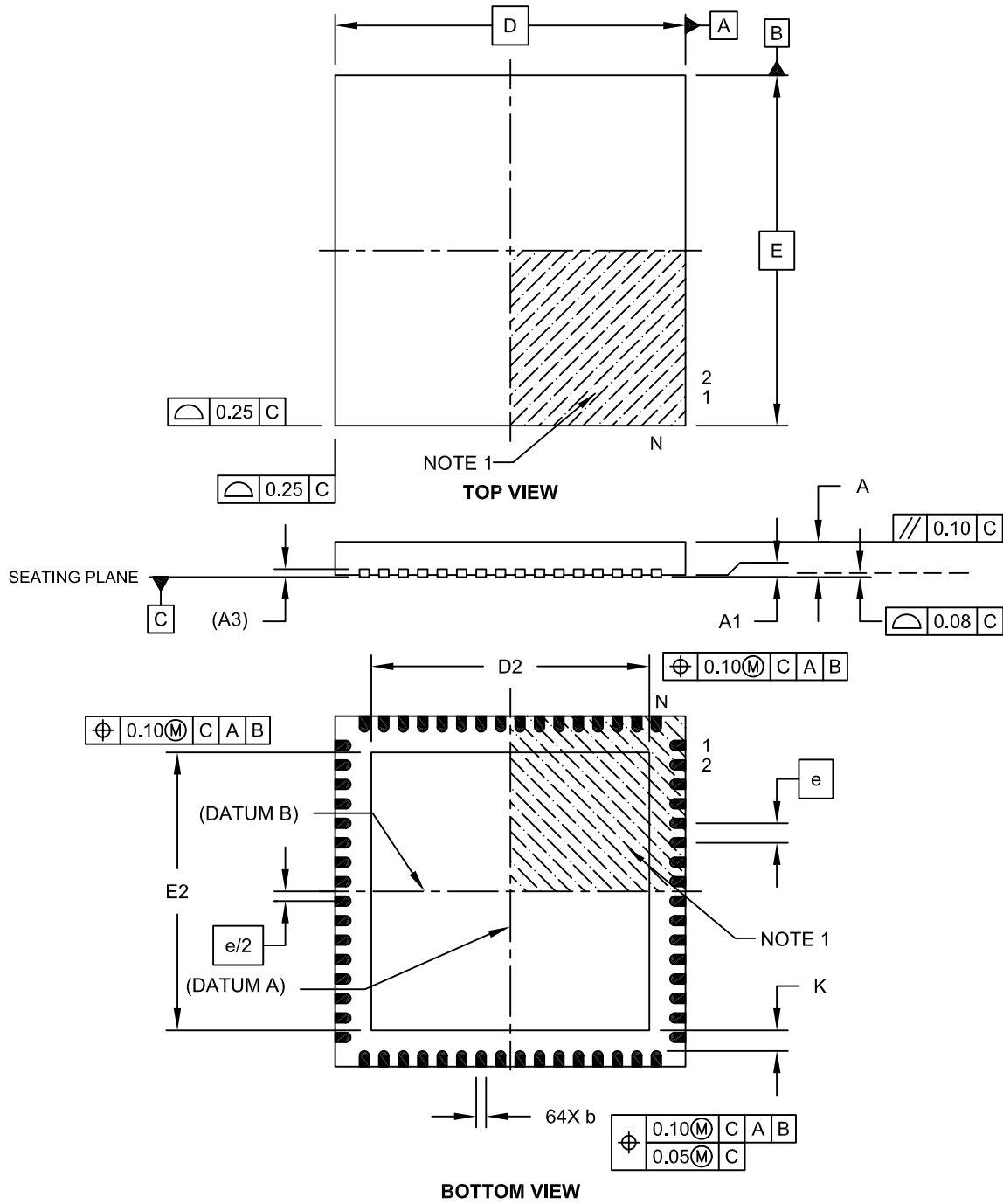
Note: In the event the full Microchip part number cannot be marked on one line, it will be carried over to the next line, thus limiting the number of available characters for customer-specific information.

40.1 Package Details

The following sections give the technical details of the packages.

64-Lead Plastic Quad Flat, No Lead Package (MR) – 9x9x0.9 mm Body [QFN] With 7.15 x 7.15 Exposed Pad [QFN]

Note: For the most current package drawings, please see the Microchip Packaging Specification located at <http://www.microchip.com/packaging>

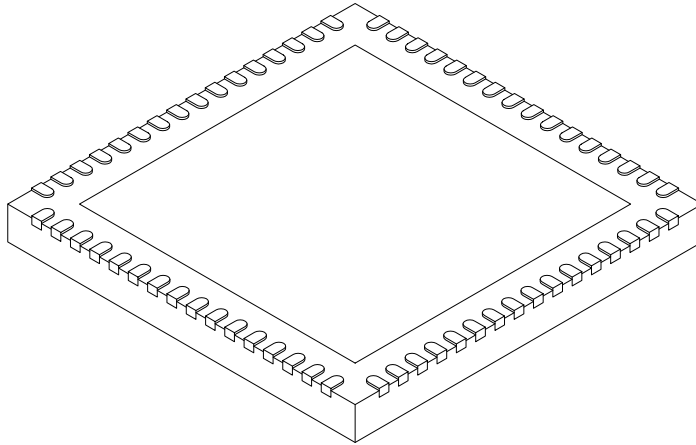


Microchip Technology Drawing C04-149C Sheet 1 of 2

PIC18(L)F67K40

64-Lead Plastic Quad Flat, No Lead Package (MR) – 9x9x0.9 mm Body [QFN] With 7.15 x 7.15 Exposed Pad [QFN]

Note: For the most current package drawings, please see the Microchip Packaging Specification located at <http://www.microchip.com/packaging>



| Dimension Limits | Units | MILLIMETERS | | |
|------------------------|-------|-------------|------|------|
| | | MIN | NOM | MAX |
| Number of Pins | N | 64 | | |
| Pitch | e | 0.50 BSC | | |
| Overall Height | A | 0.80 | 0.90 | 1.00 |
| Standoff | A1 | 0.00 | 0.02 | 0.05 |
| Contact Thickness | A3 | 0.20 REF | | |
| Overall Width | E | 9.00 BSC | | |
| Exposed Pad Width | E2 | 7.05 | 7.15 | 7.50 |
| Overall Length | D | 9.00 BSC | | |
| Exposed Pad Length | D2 | 7.05 | 7.15 | 7.50 |
| Contact Width | b | 0.18 | 0.25 | 0.30 |
| Contact Length | L | 0.30 | 0.40 | 0.50 |
| Contact-to-Exposed Pad | K | 0.20 | - | - |

Notes:

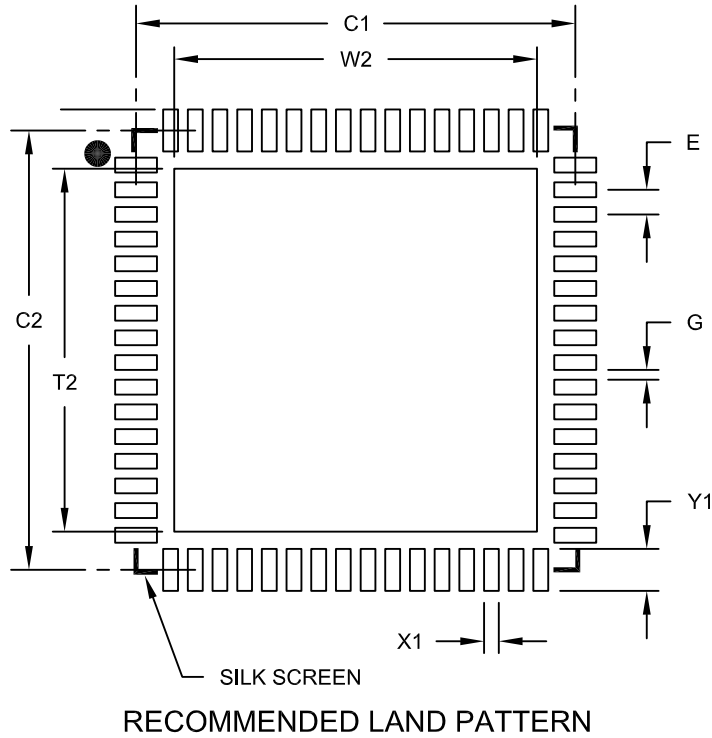
- Pin 1 visual index feature may vary, but must be located within the hatched area.
- Package is saw singulated.
- Dimensioning and tolerancing per ASME Y14.5M.
BSC: Basic Dimension. Theoretically exact value shown without tolerances.
REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-149C Sheet 2 of 2

PIC18(L)F67K40

64-Lead Plastic Quad Flat, No Lead Package (MR) – 9x9x0.9 mm Body [QFN]
With 0.40 mm Contact Length

Note: For the most current package drawings, please see the Microchip Packaging Specification located at <http://www.microchip.com/packaging>



| Dimension Limits | Units | MILLIMETERS | | |
|----------------------------|-------|-------------|------|------|
| | | MIN | NOM | MAX |
| Contact Pitch | E | 0.50 BSC | | |
| Optional Center Pad Width | W2 | | | 7.35 |
| Optional Center Pad Length | T2 | | | 7.35 |
| Contact Pad Spacing | C1 | | 8.90 | |
| Contact Pad Spacing | C2 | | 8.90 | |
| Contact Pad Width (X64) | X1 | | | 0.30 |
| Contact Pad Length (X64) | Y1 | | | 0.85 |
| Distance Between Pads | G | 0.20 | | |

Notes:

1. Dimensioning and tolerancing per ASME Y14.5M

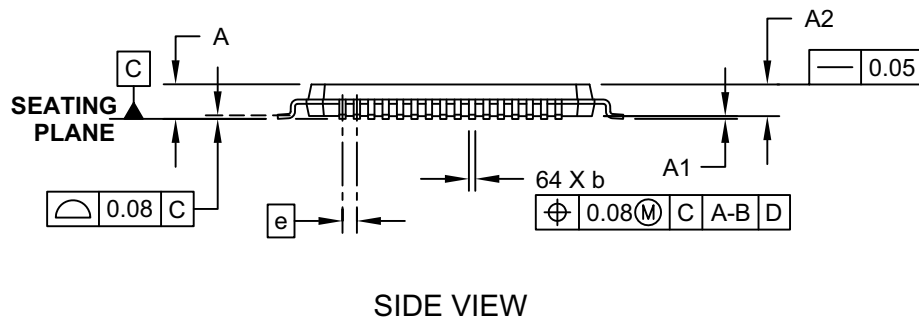
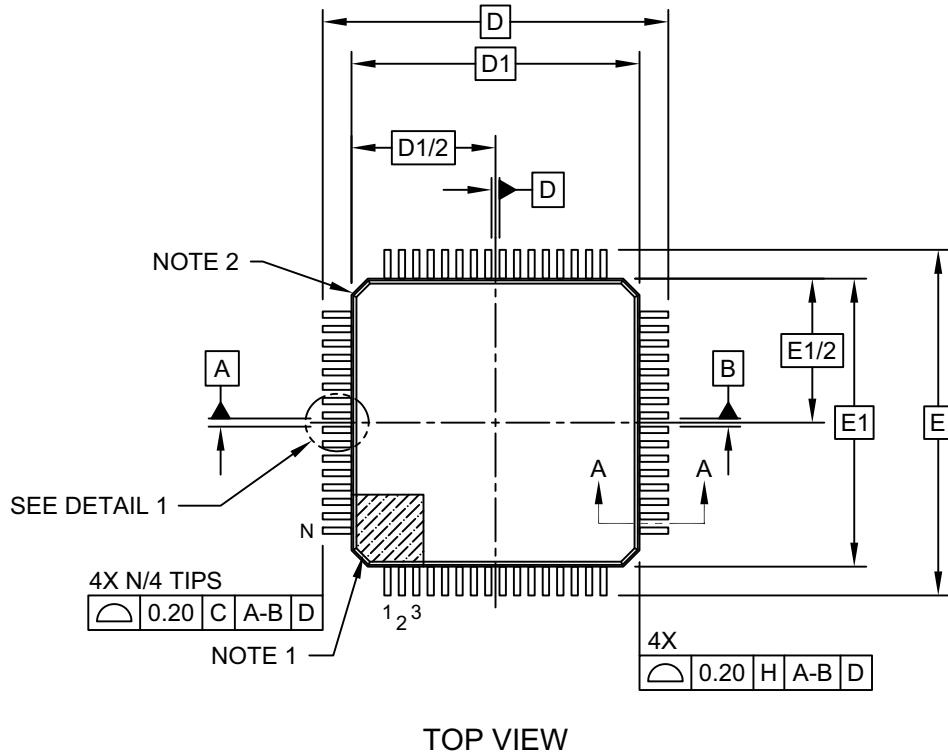
BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2149A

PIC18(L)F67K40

64-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>

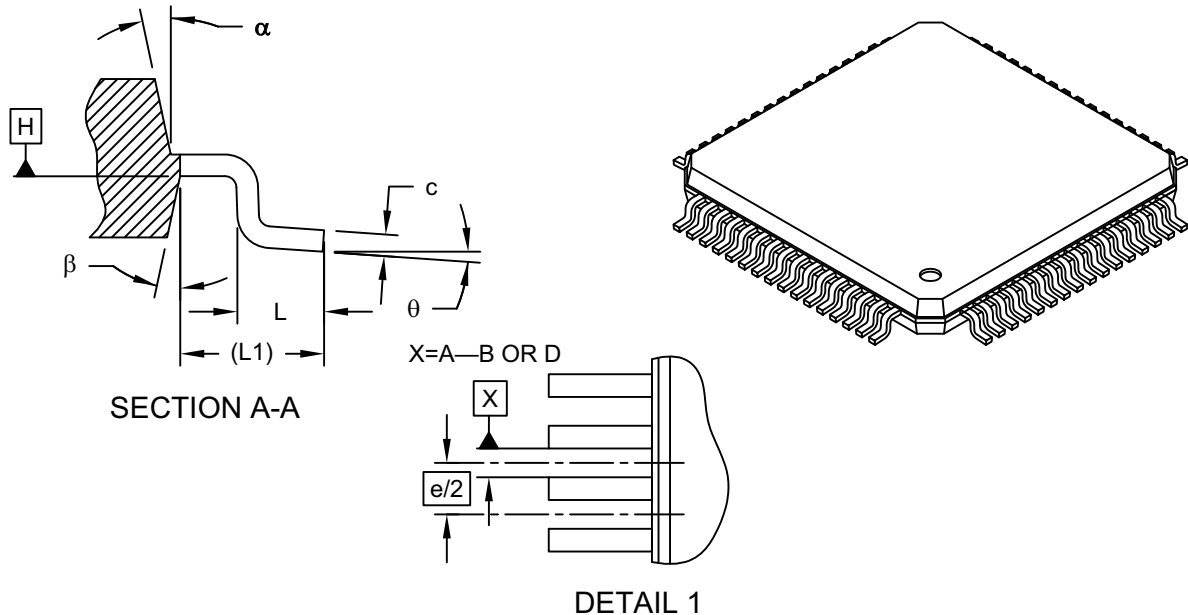


Microchip Technology Drawing C04-085C Sheet 1 of 2

PIC18(L)F67K40

64-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>



| Dimension Limits | Units | MILLIMETERS | | |
|--------------------------|----------|-------------|------|------|
| | | MIN | NOM | MAX |
| Number of Leads | N | 64 | | |
| Lead Pitch | e | 0.50 BSC | | |
| Overall Height | A | - | - | 1.20 |
| Molded Package Thickness | A2 | 0.95 | 1.00 | 1.05 |
| Standoff | A1 | 0.05 | - | 0.15 |
| Foot Length | L | 0.45 | 0.60 | 0.75 |
| Footprint | L1 | 1.00 REF | | |
| Foot Angle | ϕ | 0° | 3.5° | 7° |
| Overall Width | E | 12.00 BSC | | |
| Overall Length | D | 12.00 BSC | | |
| Molded Package Width | E1 | 10.00 BSC | | |
| Molded Package Length | D1 | 10.00 BSC | | |
| Lead Thickness | c | 0.09 | - | 0.20 |
| Lead Width | b | 0.17 | 0.22 | 0.27 |
| Mold Draft Angle Top | α | 11° | 12° | 13° |
| Mold Draft Angle Bottom | β | 11° | 12° | 13° |

Notes:

- Pin 1 visual index feature may vary, but must be located within the hatched area.
- Chamfers at corners are optional; size may vary.
- Dimensions D1 and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed 0.25mm per side.
- Dimensioning and tolerancing per ASME Y14.5M

BSC: Basic Dimension. Theoretically exact value shown without tolerances.

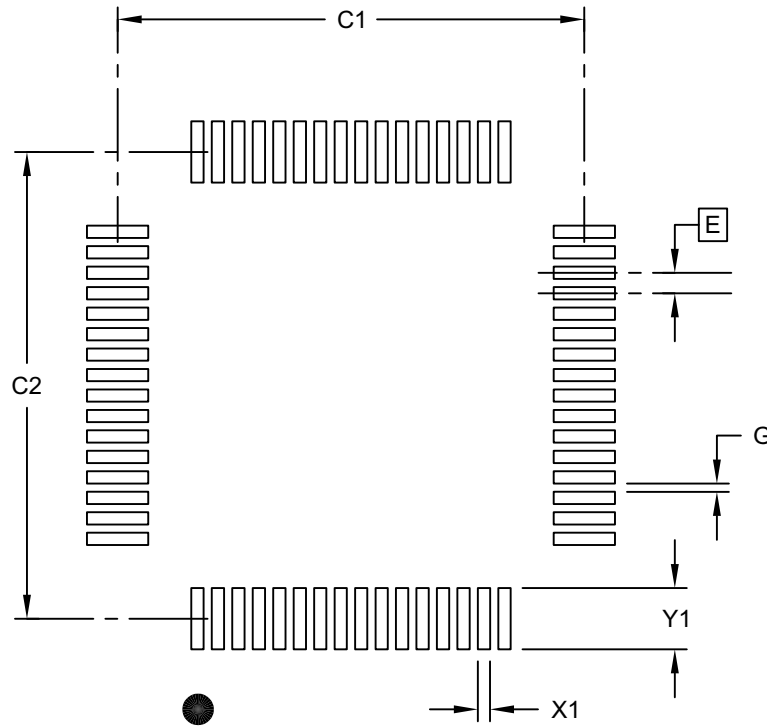
REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-085C Sheet 2 of 2

PIC18(L)F67K40

64-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

| Dimension Limits | Units | MILLIMETERS | | |
|--------------------------|-------|-------------|-------|------|
| | | MIN | NOM | MAX |
| Contact Pitch | E | 0.50 BSC | | |
| Contact Pad Spacing | C1 | | 11.40 | |
| Contact Pad Spacing | C2 | | 11.40 | |
| Contact Pad Width (X28) | X1 | | | 0.30 |
| Contact Pad Length (X28) | 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.

Microchip Technology Drawing C04-2085B Sheet 1 of 1

APPENDIX A: REVISION HISTORY

Revision A (6/2016)

Initial release of this document.

Revision B (9/2016)

Updated Peripheral Module, Memory and Core features descriptions on cover page. Updated the PIC18(L)F2x/4xK40 Family Types Table. Updated Examples 11-1, 11-3, 11-5 and 11-6; Figure 32-2; Registers 4-2, 4-5, 13-18 and 32-6; Sections 1.2, 4.4.1, 4.5, 4.5.4, 17.3, 17.5, 17.7, 18.1, 18.1.1, 18.1.1.1, 18.1.2, 18.1.6, 18.3, 18.4, 18.7, 19.0, 19.8.1, 20.0, 21.3 and 26.3; Tables 4-2, 10-4, 38-5 and 38-14.

APPENDIX B: DEVICE DIFFERENCES

The differences between the devices listed in this data sheet are shown in [Table B-1](#).

TABLE B-1: DEVICE DIFFERENCES

| Features ⁽¹⁾ | PIC18(L)F65K40 | PIC18(L)F66K40 | PIC18(L)F67K40 |
|-------------------------|----------------|----------------|----------------|
| Program Memory (Bytes) | 32768 | 65536 | 131072 |
| SRAM (Bytes) | 2048 | 3562 | 3562 |

Note 1: PIC18F6xK40: operating voltage, 2.3V-5.5V. PIC18LF6xK40: operating voltage, 1.8V-3.6V.

THE MICROCHIP WEBSITE

Microchip provides online support via our website at www.microchip.com. This website is used as a means to make files and information easily available to customers. Accessible by using your favorite Internet browser, the website contains the following information:

- **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 (FAQ), technical support requests, online discussion groups, Microchip consultant 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

CUSTOMER CHANGE NOTIFICATION SERVICE

Microchip's customer notification service helps keep customers current on Microchip products. Subscribers will receive e-mail notification whenever there are changes, updates, revisions or errata related to a specified product family or development tool of interest.

To register, access the Microchip website at www.microchip.com. Under "Support", click on "Customer Change Notification" and follow the registration instructions.

CUSTOMER SUPPORT

Users of Microchip products can receive assistance through several channels:

- Distributor or Representative
- Local Sales Office
- Field Application Engineer (FAE)
- Technical Support

Customers should contact their distributor, representative or Field Application Engineer (FAE) for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in the back of this document.

Technical support is available through the website at: <http://www.microchip.com/support>

PRODUCT IDENTIFICATION SYSTEM

To order or obtain information, e.g., on pricing or delivery, refer to the factory or the listed sales office.

| <u>PART NO.</u> | <u>[X]⁽²⁾</u> | - | <u>X</u> | <u>/XX</u> | <u>XXX</u> |
|---|--------------------------|---|-------------------|------------|------------|
| Device | Tape and Reel Option | | Temperature Range | Package | Pattern |
| <p>Device: PIC18F67K40, PIC18LF67K40</p> <p>Tape and Reel Option: Blank = standard packaging (tube or tray) T = Tape and Reel^{(1), (2)}</p> <p>Temperature Range: E = -40°C to +125°C (Extended) I = -40°C to +85°C (Industrial)</p> <p>Package: PT = 64-lead TQFP 10x10x1mm MR = 64-lead QFN 9x9x0.9mm</p> <p>Pattern: QTP, SQTP, Code or Special Requirements (blank otherwise)</p> | | | | | |
| <p>Examples:</p> <p>a) PIC18F67K40T-I/MR = Tape and reel, Industrial temp., QFN package.</p> | | | | | |
| <p>Note 1: Tape and Reel option is available for ML, MV, PT, SO and SS packages with industrial Temperature Range only.</p> <p>2: Tape and Reel identifier only appears in catalog part number description. This identifier is used for ordering purposes and is not printed on the device package.</p> | | | | | |

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as “unbreakable.”

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Information contained in this publication regarding device applications and the like 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. 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 ITS CONDITION, QUALITY, PERFORMANCE, MERCHANTABILITY OR FITNESS FOR PURPOSE. Microchip disclaims all liability arising from this information and its use. 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.

Microchip received ISO/TS-16949:2009 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEELoC® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.

**QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
= ISO/TS 16949 =**

Trademarks

The Microchip name and logo, the Microchip logo, AnyRate, dsPIC, FlashFlex, flexPWR, Heldo, JukeBlox, KeeLoq, KeeLoq logo, Klear, LANCheck, LINK MD, MediaLB, MOST, MOST logo, MPLAB, OptoLyzer, PIC, PICSTART, PIC32 logo, RightTouch, SpyNIC, SST, SST Logo, SuperFlash and UNI/O are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

ClockWorks, The Embedded Control Solutions Company, ETHERSYNCH, Hyper Speed Control, HyperLight Load, IntelliMOS, mTouch, Precision Edge, and QUIET-WIRE are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Analog-for-the-Digital Age, Any Capacitor, AnyIn, AnyOut, BodyCom, chipKIT, chipKIT logo, CodeGuard, dsPICDEM, dsPICDEM.net, Dynamic Average Matching, DAM, ECAN, EtherGREEN, In-Circuit Serial Programming, ICSP, Inter-Chip Connectivity, JitterBlocker, KlearNet, KlearNet logo, MiWi, motorBench, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, MultiTRAK, NetDetach, Omniscient Code Generation, PICDEM, PICDEM.net, PICkit, PICtail, PureSilicon, RightTouch logo, REAL ICE, Ripple Blocker, Serial Quad I/O, SQL, SuperSwitcher, SuperSwitcher II, Total Endurance, TSHARC, USBCheck, VariSense, ViewSpan, WiperLock, Wireless DNA, 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.

Silicon Storage Technology is a registered trademark 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.

© 2016, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

ISBN: 978-1-5224-0949-6



MICROCHIP

Worldwide Sales and Service

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
Technical Support:
<http://www.microchip.com/support>
Web Address:
www.microchip.com

Atlanta
Duluth, GA
Tel: 678-957-9614
Fax: 678-957-1455

Austin, TX
Tel: 512-257-3370

Boston
Westborough, MA
Tel: 774-760-0087
Fax: 774-760-0088

Chicago
Itasca, IL
Tel: 630-285-0071
Fax: 630-285-0075

Cleveland
Independence, OH
Tel: 216-447-0464
Fax: 216-447-0643

Dallas
Addison, TX
Tel: 972-818-7423
Fax: 972-818-2924

Detroit
Novi, MI
Tel: 248-848-4000

Houston, TX
Tel: 281-894-5983

Indianapolis
Noblesville, IN
Tel: 317-773-8323
Fax: 317-773-5453

Los Angeles
Mission Viejo, CA
Tel: 949-462-9523
Fax: 949-462-9608

New York, NY
Tel: 631-435-6000

San Jose, CA
Tel: 408-735-9110

Canada - Toronto
Tel: 905-695-1980
Fax: 905-695-2078

ASIA/PACIFIC

Asia Pacific Office
Suites 3707-14, 37th Floor
Tower 6, The Gateway
Harbour City, Kowloon

Hong Kong
Tel: 852-2943-5100
Fax: 852-2401-3431

Australia - Sydney
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

China - Beijing
Tel: 86-10-8569-7000
Fax: 86-10-8528-2104

China - Chengdu
Tel: 86-28-8665-5511
Fax: 86-28-8665-7889

China - Chongqing
Tel: 86-23-8980-9588
Fax: 86-23-8980-9500

China - Dongguan
Tel: 86-769-8702-9880

China - Guangzhou
Tel: 86-20-8755-8029

China - Hangzhou
Tel: 86-571-8792-8115
Fax: 86-571-8792-8116

China - Hong Kong SAR
Tel: 852-2943-5100
Fax: 852-2401-3431

China - Nanjing
Tel: 86-25-8473-2460
Fax: 86-25-8473-2470

China - Qingdao
Tel: 86-532-8502-7355
Fax: 86-532-8502-7205

China - Shanghai
Tel: 86-21-5407-5533
Fax: 86-21-5407-5066

China - Shenyang
Tel: 86-24-2334-2829
Fax: 86-24-2334-2393

China - Shenzhen
Tel: 86-755-8864-2200
Fax: 86-755-8203-1760

China - Wuhan
Tel: 86-27-5980-5300
Fax: 86-27-5980-5118

China - Xian
Tel: 86-29-8833-7252
Fax: 86-29-8833-7256

ASIA/PACIFIC

China - Xiamen
Tel: 86-592-2388138
Fax: 86-592-2388130

China - Zhuhai
Tel: 86-756-3210040
Fax: 86-756-3210049

India - Bangalore
Tel: 91-80-3090-4444
Fax: 91-80-3090-4123

India - New Delhi
Tel: 91-11-4160-8631
Fax: 91-11-4160-8632

India - Pune
Tel: 91-20-3019-1500

Japan - Osaka
Tel: 81-6-6152-7160
Fax: 81-6-6152-9310

Japan - Tokyo
Tel: 81-3-6880-3770
Fax: 81-3-6880-3771

Korea - Daegu
Tel: 82-53-744-4301
Fax: 82-53-744-4302

Korea - Seoul
Tel: 82-2-554-7200
Fax: 82-2-558-5932 or
82-2-558-5934

Malaysia - Kuala Lumpur
Tel: 60-3-6201-9857
Fax: 60-3-6201-9859

Malaysia - Penang
Tel: 60-4-227-8870
Fax: 60-4-227-4068

Philippines - Manila
Tel: 63-2-634-9065
Fax: 63-2-634-9069

Singapore
Tel: 65-6334-8870
Fax: 65-6334-8850

Taiwan - Hsin Chu
Tel: 886-3-5778-366
Fax: 886-3-5770-955

Taiwan - Kaohsiung
Tel: 886-7-213-7828

Taiwan - Taipei
Tel: 886-2-2508-8600
Fax: 886-2-2508-0102

Thailand - Bangkok
Tel: 66-2-694-1351
Fax: 66-2-694-1350

EUROPE

Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

Denmark - Copenhagen
Tel: 45-4450-2828
Fax: 45-4485-2829

France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

Germany - Dusseldorf
Tel: 49-2129-3766400

Germany - Karlsruhe
Tel: 49-721-625370

Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

Italy - Venice
Tel: 39-049-7625286

Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

Poland - Warsaw
Tel: 48-22-3325737

Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

Sweden - Stockholm
Tel: 46-8-5090-4654

UK - Wokingham
Tel: 44-118-921-5800
Fax: 44-118-921-5820

06/23/16