EN: This Datasheet/Document is presented by the manufacturer.

Please visit our website for pricing and availability at www.hestore.hu.

PIC18F24/25Q10

28-Pin, Low-Power, High-Performance Microcontrollers

Description

PIC18F24/25Q10 microcontrollers feature Analog, Core Independent, and Communication Peripherals for a wide range of general purpose and low-power applications. These 28 -pin devices are equipped with a 10-bit ADC with Computation (ADC^2) 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
- Operating Speed:
  - DC – 64 MHz clock input over the full V_{DD} range
  - 62.5 ns minimum instruction cycle
- Programmable 2-Level Interrupt Priority
- 31-Level Deep Hardware Stack
- Three 8-Bit Timers (TMR2/4/6) with Hardware Limit Timer (HLT)
- Four 16-Bit Timers (TMR0/1/3/5)
- Low-Current Power-on Reset (POR)
- Power-up Timer (PWRT)
- Brown-out Reset (BOR)
- Low-Power BOR (LPBOR) Option
- Windowed Watchdog Timer (WWDT):
  - Watchdog Reset on too long or too short interval between watchdog clear events
  - Variable prescaler selection
  - Variable window size selection
  - All sources configurable in hardware or software

Memory

- Up to 32K Bytes Program Flash Memory
- Up to 2048 Bytes Data SRAM Memory
- 256 Bytes Data EEPROM
- Programmable Code Protection

© 2018 Microchip Technology Inc.

- Direct, Indirect and Relative Addressing modes

Operating Characteristics

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

Power-Saving 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 mode (XLP)
  - Sleep: 500 nA typical @ 1.8V
  - Sleep and Watchdog Timer: 900 nA typical @ 1.8V

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:
  - Two 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:
  - One Enhanced USART (EUSART) with Auto-Baud Detect, Auto-wake-up on Start. RS-232, RS-485, LIN compatible
  - SPI
  - I^2C, SMBus and PMBus™ compatible
- Up to 25 I/O Pins and One Input Pin:
  - Individually programmable pull-ups
  - Slew rate control
  - Interrupt-on-change on all pins
  - Input level selection control
- 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+HLT):
  - Hardware monitoring and Fault detection
- Peripheral Pin Select (PPS):
  - Enables pin mapping of digital I/O
- Data Signal Modulator (DSM)

Analog Peripherals

- 10-Bit Analog-to-Digital Converter with Computation (ADC^2):
  - 24 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
  - 8-bit hardware acquisition timer
- 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},[V_{Ref+} - V_{Ref-}], FVR)
  - Internal connections to comparators and ADC
- Two Comparators (CMP):
  - Four external inputs
  - External output via PPS
- Fixed Voltage Reference (FVR) Module:
  - 1.024V, 2.048V and 4.096V output levels
  - Two buffered outputs: One for DAC/CMP and one for ADC

Clocking Structure

- High-Precision Internal Oscillator Block (HFINTOSC):
  - Selectable frequencies up to 64 MHz
  - ±1% at calibration
- 32 kHz Low-Power Internal Oscillator (LFINTOSC)
- External 32 kHz Crystal Oscillator (SOSC)
- External High-frequency Oscillator Block:
  - Three crystal/resonator modes

© 2018 Microchip Technology Inc.

  - Digital Clock Input mode
  - 4x PLL with external sources
- Fail-Safe Clock Monitor:
  - Allows for safe shutdown if external clock stops
- Oscillator Start-up Timer (OST)

Programming/Debug Features

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

PIC18F24/25Q10 Family Types

Table 1. Devices included in this data sheet

Device	Program Memory Flash (bytes)	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	CLC	Low Voltage Detect (LVD)	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)
PIC18F24Q10	16k	1024	256	25	4	2	24	1	1	2/2	1	0	1	3	Y	Y	1	1	Y	Y	Y	I
PIC18F25Q10	32k	2048	256	25	4	2	24	1	1	2/2	1	0	1	3	Y	Y	1	1	Y	Y	Y	I

Table 2. Devices not included in this data sheet

Device	Program Memory Flash (bytes)	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	CLC	Low Voltage Detect (LVD)	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)
PIC18F26Q10	64k	3615	1024	25	4	2	24	1	1	2/2	1	8	1	3	Y	Y	2	2	Y	Y	Y	I
PIC18F27Q10	128k	3615	1024	25	4	2	24	1	1	2/2	1	8	1	3	Y	Y	2	2	Y	Y	Y	I
PIC18F45Q10	32k	2048	256	36	4	2	35	1	1	2/2	1	8	1	3	Y	Y	2	2	Y	Y	Y	I
PIC18F46Q10	64k	3615	1024	36	4	2	35	1	1	2/2	1	8	1	3	Y	Y	2	2	Y	Y	Y	I
PIC18F47Q10	128k	3615	1024	36	4	2	35	1	1	2/2	1	8	1	3	Y	Y	2	2	Y	Y	Y	I

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

Data Sheet Index:

1. DS40001996 Data Sheet, 28/40-Pin, 8-bit Flash Microcontrollers
2. DS(TBD) Data Sheet, 28/40-Pin, 8-bit Flash Microcontrollers

Packages

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

Packages	SPDIP (SP)	SOIC (SO)	SSOP (SS)	QFN (ML) (6x6x0.9)	VQFN (STX) (4x4x1)	TQFP (PT)	PDIP (P)	VQFN (NHX) (5x5x0.9)	QFN (ML) (8x8)
PIC18F24Q10	•	•	•	•	•				
PIC18F25Q10	•	•	•	•	•				

Important: Pin details are subject to change.

Pin Diagrams

Figure 1. 28-pin SPDIP, SSOP, SOIC

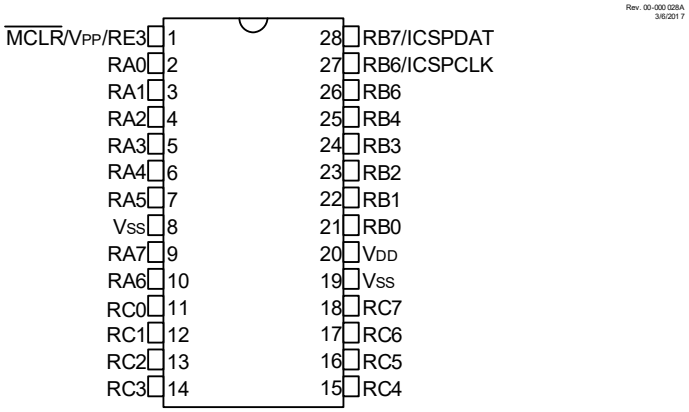
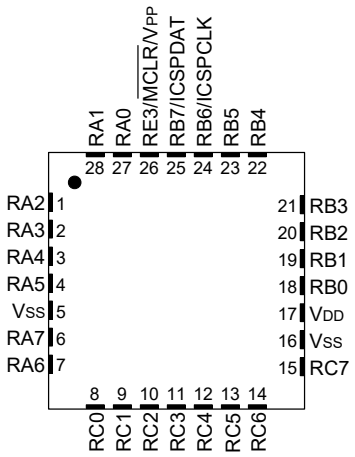


Figure 2. 28-pin QFN, VQFN



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

Pin Allocation Tables

Table 1. 28-Pin Allocation Table

I/O(2)	28-Pin SPDIP, SOIC, SSOP	28-Pin (V)QFN	A/D	Reference	Comparator	Timers	CCP	CWG	ZCD	Interrupt	EUSART	DSM	MSSP	Pull-up	Basic
RA0	2	27	ANA0	—	C1IN0- C2IN0-	—	—	—	—	IOCA0	—	—	—	Y	—
RA1	3	28	ANA1	—	C1IN1- C2IN1-	—	—	—	—	IOCA1	—	—	—	Y	—
RA2	4	1	ANA2	DAC1OUT1 Vref- (DAC) Vref- (ADC)	C1IN0+ C2IN0+	—	—	—	—	IOCA2	—	—	—	Y	—
RA3	5	2	ANA3	Vref+ (DAC) Vref+ (ADC)	C1IN1+	—	—	—	—	IOCA3	—	MDCARL(1)	—	Y	—
RA4	6	3	ANA4	—	—	T0CKI(1)	—	—	—	IOCA4	—	MDCARH(1)	—	Y	—
RA5	7	4	ANA5	—	—	—	—	—	—	IOCA5	—	MDSRC(1)	$\overline{SS1}$(1)	Y	—
RA6	10	7	ANA6	—	—	—	—	—	—	IOCA6	—	—	—	Y	CLKOUT OSC2
RA7	9	6	ANA7	—	—	—	—	—	—	IOCA7	—	—	—	Y	OSC1 CLKIN
RB0	21	18	ANB0	—	C2IN1+	—	—	CWG1(1)	ZCDIN	IOCB0 INT0(1)	—	—	—	Y	—
RB1	22	19	ANB1	—	C1IN3- C2IN3-	—	—	—	—	IOCB1 INT1(1)	—	—	—	Y	—
RB2	23	20	ANB2	—	—	—	—	—	—	IOCB2 INT2(1)	—	—	—	Y	—

I/O(2)	28-Pin SPDIP, SOIC, SSOP	28-Pin (V)QFN	A/D	Reference	Comparator	Timers	CCP	CWG	ZCD	Interrupt	EUSART	DSM	MSSP	Pull-up	Basic
RB3	24	21	ANB3	—	C1IN2- C2IN2-	—	—	—	—	IOCB3	—	—	—	Y	—
RB4	25	22	ANB4	—	—	T5G(1)	—	—	—	IOCB4	—	—	—	Y	—
RB5	26	23	ANB5	—	—	T1G(1)	—	—	—	IOCB5	—	—	—	Y	—
RB6	27	24	ANB6	—	—	—	—	—	—	IOCB6	—	—	—	Y	ICSPCLK
RB7	28	25	ANB7	DAC1OUT2	—	T6IN(1)	—	—	—	IOCB7	—	—	—	Y	ICSPDAT
RC0	11	8	ANC0	—	—	T1CKI(1) T3CKI(1) T3G(1)	—	—	—	IOCC0	—	—	—	Y	SOSCO
RC1	12	9	ANC1	—	—	—	CCP2(1)	—	—	IOCC1	—	—	—	Y	SOSCIN SOSCI
RC2	13	10	ANC2	—	—	T5CKI(1)	CCP1(1)	—	—	IOCC2	—	—	—	Y	—
RC3	14	11	ANC3	—	—	T2IN(1)	—	—	—	IOCC3	—	—	SCK1(1) SCL1(3,4)	Y	—
RC4	15	12	ANC4	—	—	—	—	—	—	IOCC4	—	—	SDI1(1) SDA1(3,4)	Y	—
RC5	16	13	ANC5	—	—	T4IN(1)	—	—	—	IOCC5	—	—	—	Y	—
RC6	17	14	ANC6	—	—	—	—	—	—	IOCC6	CK1(1,3)	—	—	Y	—
RC7	18	15	ANC7	—	—	—	—	—	—	IOCC7	RX1/ DT1(1,3)	—	—	Y	—
RE3	1	26	—	—	—	—	—	—	—	IOCE3	—	—	—	Y	Vpp/$\overline{MCLR}$
V_{SS}	19	16	—	—	—	—	—	—	—	—	—	—	—	—	V_{SS}
V_{DD}	20	17	—	—	—	—	—	—	—	—	—	—	—	—	V_{DD}
V_{SS}	8	5	—	—	—	—	—	—	—	—	—	—	—	—	V_{SS}
OUT(2)	—	—	ADGRDA ADGRDB	—	C1OUT C2OUT	TMR0	CCP1 CCP2 PWM3 PWM4	CWG1A CWG1B CWG1C CWG1D	—	—	TX1/ CK1(3) DT1(3)	DSM	SDO1 SCK1	—	—

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 the peripheral input selection table for details on which port pins may be used for this signal.
2. All output signals shown in this row are PPS remappable. These signals may be mapped to output onto one of several PORTx pin options as described in the peripheral output selection table.
3. This is a bidirectional signal. For normal module operation, the firmware should map this signal to the same pin in both the PPS input and PPS output registers.
4. These pins are configured for I^2C logic levels; The SCLx/SDAx signals may be assigned to any of these pins. PPS assignments to the other pins (e.g., RB1) will operate, but input logic levels will be standard TTL/ST as selected by the INLVL register, instead of the I^2C specific or SMBus input buffer thresholds.

Table of Contents

Description....1

Core Features.... 1

Memory....1

Operating Characteristics....2

Power-Saving Operation Modes....2

Digital Peripherals....2

Analog Peripherals....3

Clocking Structure.... 3

Programming/Debug Features....4

PIC18F24/25Q10 Family Types....4

Packages....5

Pin Diagrams....5

Pin Allocation Tables.... 6

1. Device Overview....11
2. Guidelines for Getting Started with PIC18F24/25Q10 Microcontrollers.... 19
3. Device Configuration....24
4. Oscillator Module (with Fail-Safe Clock Monitor)....40
5. Reference Clock Output Module.... 63
6. Power-Saving Operation Modes....69
7. (PMD) Peripheral Module Disable....79
8. Resets.... 88
9. (WWDT) Windowed Watchdog Timer....102
10. Memory Organization.... 114
11. (NVM) Nonvolatile Memory Control....149

12. 8x8 Hardware Multiplier.....179
13. Cyclic Redundancy Check (CRC) Module with Memory Scanner.....184
14. Interrupts.....204
15. I/O Ports.....235
16. Interrupt-on-Change.....269
17. (PPS) Peripheral Pin Select Module.....285
18. Timer0 Module.....295
19. Timer1 Module with Gate Control.....304
20. Timer2 Module.....323
21. Capture/Compare/PWM Module.....349
22. (PWM) Pulse-Width Modulation.....365
23. (ZCD) Zero-Cross Detection Module.....374
24. (CWG) Complementary Waveform Generator Module.....382
25. (DSM) Data Signal Modulator Module.....411
26. (MSSP) Master Synchronous Serial Port Module.....424
27. (EUSART) Enhanced Universal Synchronous Asynchronous Receiver Transmitter 489
28. (FVR) Fixed Voltage Reference.....524
29. Temperature Indicator Module.....529
30. (DAC) 5-Bit Digital-to-Analog Converter Module.....532
31. (ADC^2) Analog-to-Digital Converter with Computation Module.....538
32. (CMP) Comparator Module.....586
33. (HLVD) High/Low-Voltage Detect.....599
34. Register Summary.....607
35. In-Circuit Serial Programming™ (ICSP™) 615
36. Instruction Set Summary.....618
37. Development Support.....713

© 2018 Microchip Technology Inc.

38. Electrical Specifications....718
39. DC and AC Characteristics Graphs and Tables....749
40. Packaging Information....750
41. Revision History....764
The Microchip Web Site....765
Customer Change Notification Service....765
Customer Support....765
Product Identification System....766
Microchip Devices Code Protection Feature....766
Legal Notice....767
Trademarks....767
Quality Management System Certified by DNV....768
Worldwide Sales and Service....769

1. Device Overview

This document contains device specific information for the following devices:

- PIC18F24Q10
- PIC18F25Q10

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 PIC18F24/25Q10 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 Low-Power Technology

All of the devices in the PIC18F24/25Q10 family incorporate a range of features that can significantly reduce power consumption during operation. Key items include:

- Alternate Run Modes: By clocking the microcontroller 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 PIC18F24/25Q10family offer several different oscillator options. The PIC18F24/25Q10 family can be clocked from several different sources:

- HFINTOSC
  - 1-64 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 optimized for 32 kHz clock crystals
- A Phase Lock Loop (PLL) frequency multiplier (4x) is available to the External Oscillator modes enabling clock speeds of up to 64 MHz

- 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.

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 PIC18F24/25Q10 family includes 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 ADC^2 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 PIC18F24/25Q10 family are available in 28-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. Data Memory EEPROM
4. A/D channels
5. I/O ports
6. Enhanced USART

7. Input Voltage Range/Power Consumption

All other features for devices in this family are identical. These are summarized in the following Device Features table.

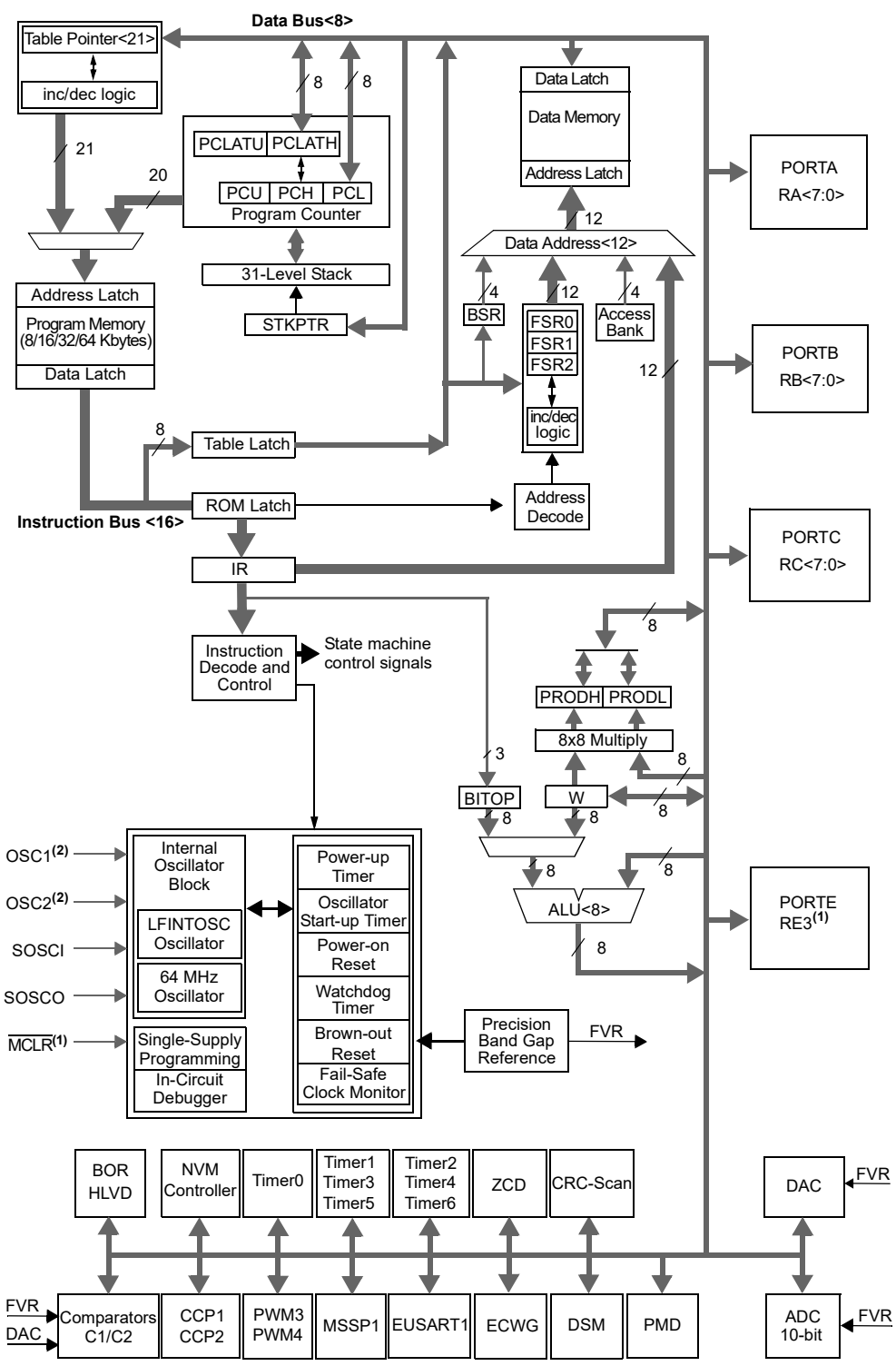
The pinouts for all devices are listed in the pin summary tables.

Table 1-1. Device Features

Features	PIC18F24Q10	PIC18F25Q10
Program Memory (Bytes)	16384	32768
Program Memory (Instructions)	8192	16384
Data Memory (Bytes)	1024	2048
Data EEPROM Memory (Bytes)	256	256
I/O Ports	A,B,C,E(1)	A,B,C,E(1)
Capture/Compare/PWM Modules (CCP)	2	2
10-Bit Pulse-Width Modulator (PWM)	2	2
10-Bit Analog-to-Digital Module (ADC^2) with Computation Accelerator	4 internal 24 external	4 internal 24 external
Packages	28-pin SPDIP 28-pin SOIC 28-pin SSOP 28-pin QFN 28-pin UQFN	28-pin SPDIP 28-pin SOIC 28-pin SSOP 28-pin QFN 28-pin UQFN
Interrupt Sources	36	
Timers (16-/8-bit)	4/3	
Serial Communications	1 MSSP, 1 EUSART	
Enhanced Complementary Waveform Generator (ECWG)	1	
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,	

Features	PIC18F24Q10	PIC18F25Q10
	RESET Instruction, Stack Overflow, Stack Underflow (PWRT, OST), $\overline{\text{MCLR}}$, WDT	
Instruction Set	75 Instructions; 83 with Extended Instruction Set enabled	
Operating Frequency	DC – 64 MHz	
Note 1: PORTE contains the single RE3 read-only bit.		

Figure 1-1. PIC18F24/25Q10 Family Block Diagram



Note 1: RE3 is only available when $\overline{\text{MCLR}}$ functionality is disabled.

2: OSC1/CLKIN and OSC2/CLKOUT are only available in select oscillator modes.

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 RegisterNamebits.ShortName. For example, the enable bit, EN, in the CM1CON0 register can be set in C programs with the instruction `CM1CON0bits.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

1.5 Register Legend

The table below describes the conventions for bit types and bit Reset values used in the current data sheet.

Table 1-2. Register Legend

Value	Description
RO	Read-only bit
W	Writable bit
U	Unimplemented bit, read as ‘0’
P	Programmable bit
‘1’	Bit is set
‘0’	Bit is cleared
x	Bit is unknown
u	Bit is unchanged
-n/n	Value at POR and BOR/Value at all other Resets
q	Reset Value is determined by hardware

Value	Description
f	Reset Value is determined by fuse setting
g	Reset Value at POR for PPS re-mappable signals

2. Guidelines for Getting Started with PIC18F24/25Q10 Microcontrollers

2.1 Basic Connection Requirements

Getting started with the PIC18F24/25Q10 family of 8-bit microcontrollers requires attention to a minimal set of device pin connections before proceeding with development.

The following pins must always be connected:

- All V_{DD} and V_{SS} pins (see 2.2 Power Supply Pins)
- $\overline{MCLR}$ pin (see 2.3 Master Clear (MCLR) Pin)

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

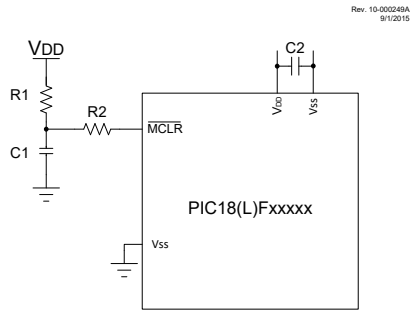
- ICSPCLK/ICSPDAT pins used for In-Circuit Serial Programming™ (ICSP™) and debugging purposes (see 2.4 In-Circuit Serial Programming™ ICSP™ Pins)
- OSCI and OSCO pins when an external oscillator source is used (see 2.5 External Oscillator Pins)

Additionally, the following pins may be required:

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

The minimum mandatory connections are shown in the figure below.

Figure 2-1. Recommended Minimum Connections



Key (all values are recommendations):
C1 and C2 : 0.1 µF, 20V ceramic
R1: 10 kΩ
R2: 100Ω to 470Ω

2.2 Power Supply Pins

2.2.1 Decoupling Capacitors

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

Consider the following criteria when using decoupling capacitors:

- Value and type of capacitor: A 0.1 µF (100 nF), 10-20V capacitor is recommended. The capacitor 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 µF to 0.001 µF. Place this second capacitor next to each primary decoupling capacitor. In high-speed circuit designs, consider implementing a decade pair of capacitances as close to the power and ground pins as possible (e.g., 0.1 µF in parallel with 0.001 µF).
- Maximizing performance: On the board layout from the power supply circuit, run the power and return traces to the decoupling capacitors first, and then to the device pins. This ensures that the decoupling capacitors are first in the power chain. Equally important is to keep the trace length between the capacitor and the power pins to a minimum, thereby reducing PCB trace inductance.

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 µF to 47 µF.

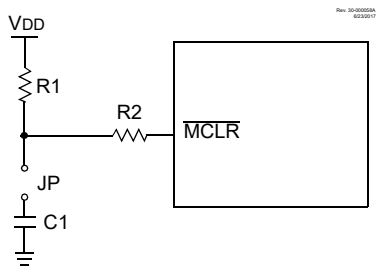
2.3 Master Clear ($\overline{MCLR}$) Pin

The $\overline{MCLR}$ pin provides two specific device functions: Device Reset, and Device Programming and Debugging. If programming and debugging are not required in the end application, a direct connection to V_{DD} may be all that is required. The addition of other components, to help increase the application's resistance to spurious Resets from voltage sags, may be beneficial. A typical configuration is shown in Figure 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 $\overline{MCLR}$ pin. Consequently, specific voltage levels (V_{IH} and V_{IL}) and fast signal transitions must not be adversely affected. Therefore, specific values of R1 and C1 will need to be adjusted based on the application and PCB requirements. For example, it is recommended that the capacitor, C1, be isolated from the $\overline{MCLR}$ pin during programming and debugging operations by using a jumper (Figure 2-2). The jumper is replaced for normal run-time operations.

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

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



Note:

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

2.4 In-Circuit Serial Programming™ ICSP™ Pins

The ICSPCLK and ICSPDAT 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 ICSPCLK and ICSPDAT pins are not recommended as they can interfere with the programmer/debugger communications to the device. If such discrete components are an application requirement, they 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., ICSPCLK/ICSPDAT 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 the *“Development Support”* section.

Related Links

37. Development Support

2.5 External Oscillator Pins

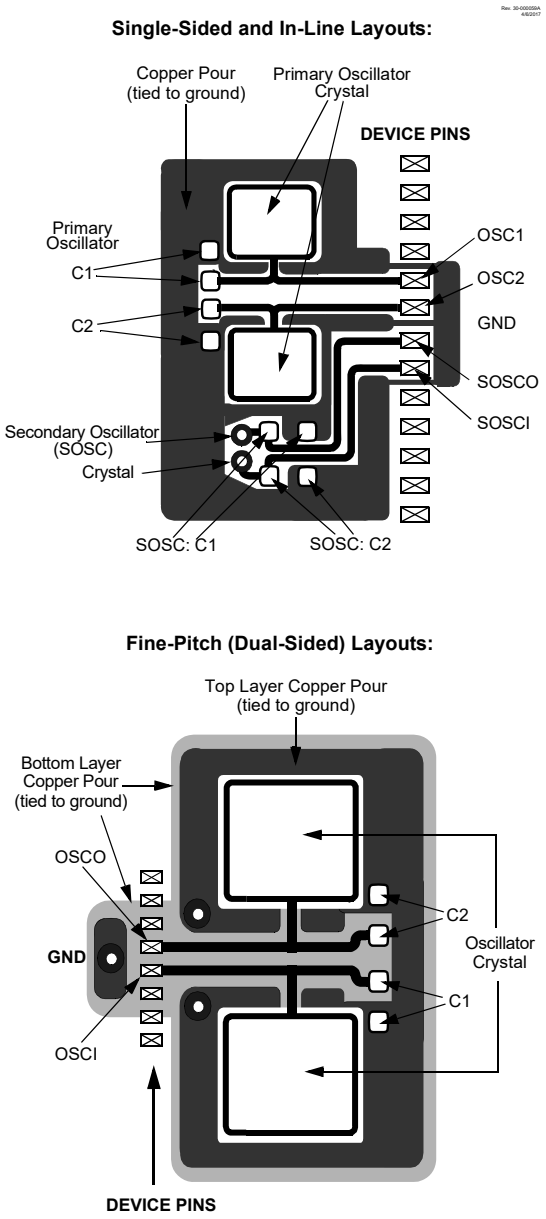
Many microcontrollers have options for at least two oscillators: a high-frequency primary oscillator and a low-frequency secondary oscillator.

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

Figure 2-3. Suggested Placement of the Oscillator Circuit



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

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

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

Related Links

4. Oscillator Module (with Fail-Safe Clock Monitor)

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 V_{SS} on unused pins to drive the output to logic low.

3. Device Configuration

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

3.1 Configuration Words

There are six Configuration Words that allow the user to select the device oscillator, reset, and memory protection options. These are implemented as Configuration Word 1 through Configuration Word 6 at 300000h through 30000Bh.

Important: The $\overline{\text{DEBUG}}$ 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'.

3.2 Code Protection

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

3.2.1 Program Memory Protection

The entire program memory space is protected from external reads and writes by the $\overline{\text{CP}}$ bit. When $\overline{\text{CP}}$ = 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.

3.2.2 Data Memory Protection

The entire Data EEPROM Memory space is protected from external reads and writes by the $\overline{\text{CPD}}$ bit. When $\overline{\text{CPD}}$ = 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.

3.3 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 bits define the size of the program memory block that is protected.

3.4 User ID

256 bytes in the memory space (200000h-20000FFh) are designated as ID locations where the user can store checksum or other code identification numbers. These locations are readable and writable during normal execution. See the *"User ID, Device ID and Configuration Word Access"* section for more information on accessing these memory locations. For more information on checksum calculation, see the *"PIC18F24/25Q10 Memory Programming Specification"*, (DS40001874).

3.5 Device ID and Revision ID

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

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

Related Links

11. (NVM) Nonvolatile Memory Control

3.6 Register Summary - Configuration Words

Address	Name	Bit Pos.								
0x300000	CONFIG1	7:0			RSTOSC[2:0]			FEXTOSC[2:0]		
		15:8			FCMEN		CSWEN			$\overline{\text{CLKOUTEN}}$
0x300002	CONFIG2	7:0	BOREN[1:0]		$\overline{\text{LPBOREN}}$				$\overline{\text{PWRTE}}$	MCLRE
		15:8	$\overline{\text{XINST}}$		$\overline{\text{DEBUG}}$	STVREN	PPS1WAY	$\overline{\text{ZCD}}$	BORV[1:0]	
0x300004	CONFIG3	7:0		WDTE[1:0]		WDTCPS[4:0]				
		15:8			WDTCCS[2:0]			WDTCWS[2:0]		
0x300006	CONFIG4	7:0					WRT3	WRT2	WRT1	WRT0
		15:8			LVP	SCANE		WRTD	WRTB	WRTC
0x300008	CONFIG5	7:0							$\overline{\text{CPD}}$	$\overline{\text{CP}}$
		15:8								
0x30000A	CONFIG6	7:0					EBTR3	EBTR2	EBTR1	EBTR0
		15:8							EBTRB	

3.7 Register Definitions: Configuration Words

3.7.1 CONFIG1

Name: CONFIG1
Address: 0x300000

Configuration word 1

Oscillators

Bit	15	14	13	12	11	10	9	8
			FCMEN		CSWEN			$\overline{\text{CLKOUTEN}}$
Access			R/W		R/W			R/W
Reset			1		1			1

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

Bit 13 – FCMEN Fail-Safe Clock Monitor Enable bit

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

Bit 11 – CSWEN Clock Switch Enable bit

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

Bit 8 – $\overline{\text{CLKOUTEN}}$ Clock Out Enable bit
If FEXTOSC = HS, XT, LP, then this bit is ignored.

Otherwise:

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

Bits 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.

Value	Description
111	EXTOSC operating per FEXTOSC bits (device manufacturing default)
110	HFINTOSC with HFFRQ = 4 MHz and CDIV = 4:1
101	LFINTOSC
100	SOSC
011	Reserved
010	EXTOSC with 4x PLL, with EXTOSC operating per FEXTOSC bits

Value	Description
001	Reserved
000	HFINTOSC with HFFRQ = 64 MHz and CDIV = 1:1. Resets COSC/NOSC to b'110'.

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

Value	Description
111	ECH (external clock) above 16 MHz
110	ECM (external clock) for 500 kHz to 16 MHz
101	ECL (external clock) below 500 kHz
100	Oscillator not enabled
011	Reserved (do not use)
010	HS (crystal oscillator) above 4 MHz
001	XT (crystal oscillator) above 500 kHz, below 4 MHz
000	LP (crystal oscillator) optimized for 32.768 kHz

Related Links

4.6.5 OSCFRQ
4.6.2 OSCCON2

3.7.2 CONFIG2

Name: CONFIG2
Address: 0x300002

Configuration Word 2

Supervisor

Bit	15	14	13	12	11	10	9	8
	$\overline{XINST}$		$\overline{DEBUG}$	STVREN	PPS1WAY	$\overline{ZCD}$	BORV[1:0]	
Access	R/W		R/W	R/W	R/W	R/W	R/W	R/W
Reset	1		1	1	1	1	1	1

Bit	7	6	5	4	3	2	1	0
	BOREN[1:0]		$\overline{LPBOREN}$				$\overline{PWRTE}$	MCLRE
Access	R/W	R/W	R/W				R/W	R/W
Reset	0	1	1				1	1

Bit 15 – $\overline{XINST}$ Extended Instruction Set Enable bit

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

Bit 13 – $\overline{DEBUG}$ Debugger Enable bit

Value	Description
1	Background debugger disabled
0	Background debugger enabled

Bit 12 – STVREN Stack Overflow/Underflow Reset Enable bit

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

Bit 11 – PPS1WAY PPSLOCKED bit One-Way Set Enable bit

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

Bit 10 – $\overline{ZCD}$ ZCD Disable bit

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

Bits 9:8 – BORV[1:0] Brown-out Reset Voltage Selection bit

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

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

Value	Description
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 – $\overline{\text{LPBOREN}}$ Low-Power BOR Enable bit

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

Bit 1 – $\overline{\text{PWRTE}}$ Power-up Timer Enable bit

Value	Description
1	PWRT disabled
0	PWRT enabled

Bit 0 – MCLRE Master Clear (MCLR) Enable bit

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

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

Related Links

7.4.3 PMD2

3.7.3 CONFIG3

Name: CONFIG3
Address: 0x300004

Configuration Word 3

Windowed Watchdog Timer

Bit	15	14	13	12	11	10	9	8
			WDTCCS[2:0]			WDTCWS[2:0]		
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			1	1	1	1	1	1

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

Bits 13:11 – WDTCCS[2:0] WDT Input Clock Selector bits

Value	Condition	Description
x	3.7.3.1 WDTE=00	These bits have no effect
111	3.7.3.1 WDTE≠00	Software Control
110 to 010	3.7.3.1 WDTE≠00	Reserved (Default to LFINTOSC)
001	3.7.3.1 WDTE≠00	WDT reference clock is the 31.25 kHz MFINTOSC
000	3.7.3.1 WDTE≠00	WDT reference clock is the 31.0 kHz LFINTOSC (default value)

Bits 10:8 – WDTCWS[2:0] WDT Window Select bits

WDTCWS	WDTCON1[WINDOW] at POR			Software control of WINDOW	Keyed access required?
	Value	Window delay Percent of time	Window opening Percent of time		
111	111	n/a	100	Yes	No
110	110	n/a	100	No	Yes
101	101	25	75		
100	100	37.5	62.5		
011	011	50	50		
010	010	62.5	37.5		
001	001	75	25		
000	000	87.5	12.5		

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

Value	Description
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

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

WDTCPS	WDTCON0[WDTPS] at POR				Software Control of WDTPS?
	Value	Divider Ratio		Typical Time Out (F_{IN} = 31 kHz)	
11111	01011	1:65536	2^{16}	2s	Yes
11110 ... 10011	11110 ... 10011	1:32	2^5	1 ms	No
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	

3.7.4 CONFIG4

Name: CONFIG4
Address: 0x300006

Configuration Word 4

Memory Write Protection

Bit	15	14	13	12	11	10	9	8
			LVP	SCANE		WRTD	WRTB	WRTC
Access			R/W	R/W		R/W	R/W	R/W
Reset			1	1		1	1	1

Bit	7	6	5	4	3	2	1	0
					WRT3	WRT2	WRT1	WRT0
Access					R/W	R/W	R/W	R/W
Reset					1	1	1	1

Bit 13 – LVP Low-Voltage Programming Enable bit
The LVP bit cannot be written (to zero) while operating from the LVP programming interface. The purpose of this rule is to prevent the user from dropping out of LVP mode while programming from LVP mode, or accidentally eliminating LVP mode from the Configuration state.

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

Bit 12 – SCANE Scanner Enable bit

Value	Description
1	Scanner module is available for use, PMD0[SCANMD] bit enables the module
0	Scanner module is NOT available for use, PMD0[SCANMD] bit is ignored

Bit 10 – WRTD Data EEPROM Write Protection bit

Value	Description
1	Data EEPROM NOT write-protected
0	Data EEPROM write-protected

Bit 9 – WRTB Boot Block Write Protection bit

Value	Description
1	Boot Block NOT write-protected
0	Boot Block write-protected

Bit 8 – WRTC Configuration Register Write Protection bit

Value	Description
1	Configuration Registers NOT write-protected
0	Configuration Registers write-protected

Bits 0, 1, 2, 3 – WRTn User NVM Self-Write Protection bits

Value	Description
1	Corresponding Memory Block NOT write-protected
0	Corresponding Memory Block write-protected

Related Links

10.1 Program Memory Organization

11.3.4 Operation During Code-Protect and Write-Protect

3.7.5 CONFIG5

Name: CONFIG5
Address: 0x300008

Configuration Word 5

Code Protection

Bit	15	14	13	12	11	10	9	8
Access								
Reset								

Bit	7	6	5	4	3	2	1	0
							$\overline{CPD}$	$\overline{CP}$
Access							RO	RO
Reset							1	1

Bit 1 – $\overline{CPD}$ Data NVM (DFM) Memory Code Protection bit

Value	Description
1	Data NVM code protection disabled
0	Data NVM code protection enabled

Bit 0 – $\overline{CP}$ User NVM Program Memory Code Protection bit

Value	Description
1	User NVM code protection disabled
0	User NVM code protection enabled

3.7.6 CONFIG6

Name: CONFIG6
Address: 0x30000A

Configuration Word 6

Memory Read Protection

Bit	15	14	13	12	11	10	9	8
							EBTRB	
Access							R/W	
Reset							1	

Bit	7	6	5	4	3	2	1	0
					EBTR3	EBTR2	EBTR1	EBTR0
Access					R/W	R/W	R/W	R/W
Reset					1	1	1	1

Bit 9 – EBTRB Table Read Protection bit

Value	Description
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

Bits 0, 1, 2, 3 – EBTRn Table Read Protection bits

Value	Description
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

Related Links

10.1 Program Memory Organization

3.8 Register Summary - Device and Revision

Address	Name	Bit Pos.								
0x3FFFFC	REVISION ID	7:0	MJRREV[1:0]		MNRREV[5:0]					
		15:8	1010[3:0]				MJRREV[5:2]			
0x3FFFFE	DEVICE ID	7:0	DEV[7:0]							
		15:8	DEV[15:8]							

3.9 Register Definitions: Device and Revision

3.9.1 DEVICE ID

Name: DEVICE ID
Address: 0x3FFFFE

Device ID Register

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

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

Bits 15:0 – DEV[15:0]
Device ID bits

Device	Device ID
PIC18F24Q10	71C0h
PIC18F25Q10	71A0h

3.9.2 REVISION ID

Name: REVISION ID
Address: 0x3FFFFC

Revision ID Register

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

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

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

Bits 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 = b'00 0000'

Bits 5:0 – MNRREV[5:0] Minor Revision ID bits
These bits are used to identify a minor revision.

4. 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 determine the type of oscillator that will be used when the device runs after Reset, including when it is first powered up.

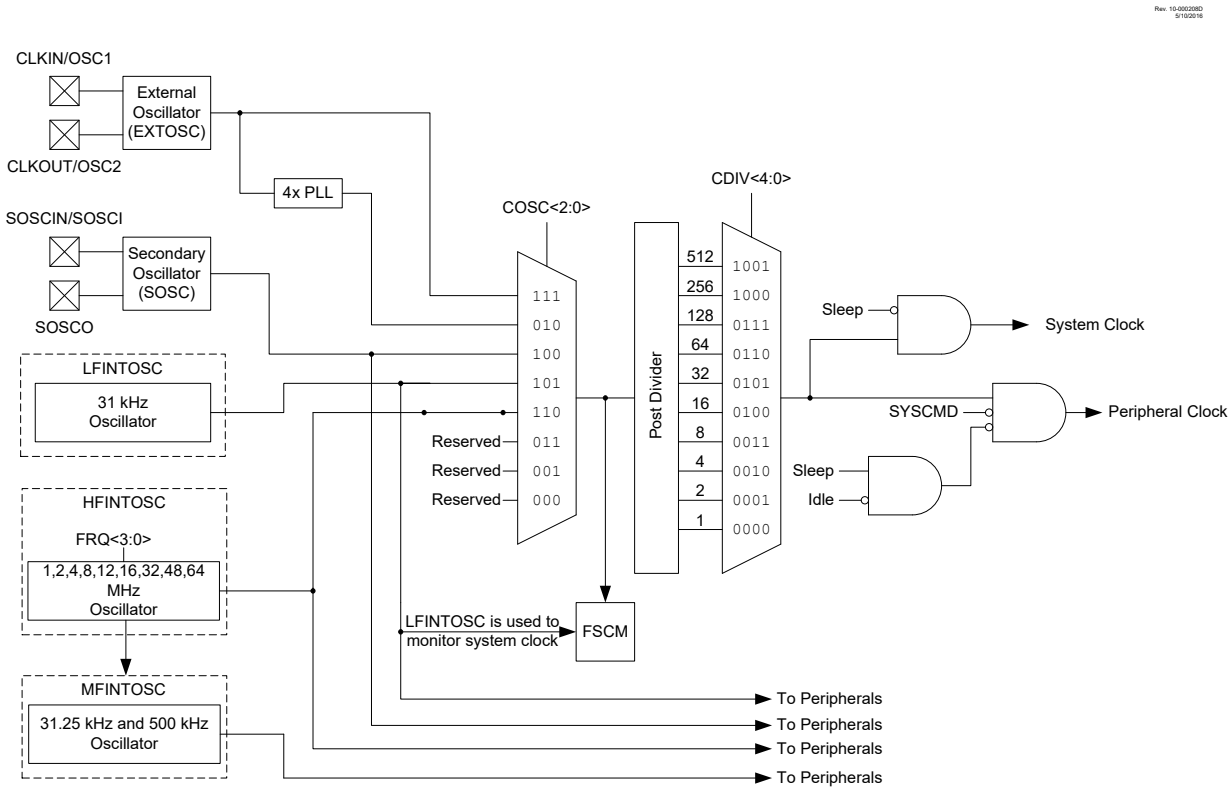
If an external clock source is selected, the FEXTOSC bits of Configuration Word 1 must be used in conjunction with the RSTOSC bits to select the External Clock mode.

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:

- ECL – External Clock Low-Power mode (below 100 kHz)
- ECM – External Clock Medium Power mode (100 kHz to 16 MHz)
- ECH – External Clock High-Power mode (above 16 MHz)
- LP – 32 kHz Low-Power Crystal mode
- XT – Medium Gain Crystal or Ceramic Resonator Oscillator mode (between 500 kHz and 4 MHz)
- 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. Multiple device clock frequencies may be derived from these clock sources.

Figure 4-1. Simplified PIC® MCU Clock Source Block Diagram



Related Links

3.7.1 CONFIG1

4.2 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. 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.

The system clock can be selected between external or internal clock sources via the NOSC bits. 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 $\overline{\text{CLKOUTEN}}$ bit in the CONFIG1H register. If enabled, the clock out signal is always at a frequency of $F_{OSC}/4$.

Related Links

4.6.5 OSCFRQ

4.2.1.4 4x PLL
4.3 Clock Switching

4.2.1 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 to switch the system clock source.

Related Links

4.3 Clock Switching

4.2.1.1 EC Mode

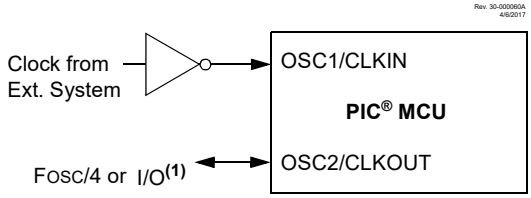
The External Clock (EC) mode allows an externally generated logic level signal to be the system clock source. When operating in this mode, an external clock source is connected to the OSC1 input. OSC2/CLKOUT is available for general purpose I/O or CLKOUT. The following figure shows the pin connections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

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

- ECH – High power, above 16 MHz
- ECM – Medium power, 100 kHz-16 MHz
- ECL – Low power, below 100 kHz

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



Note:

1. Output depends upon $\overline{\text{CLKOUTEN}}$ bit of the Configuration Words (CONFIG1H).

4.2.1.2 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.

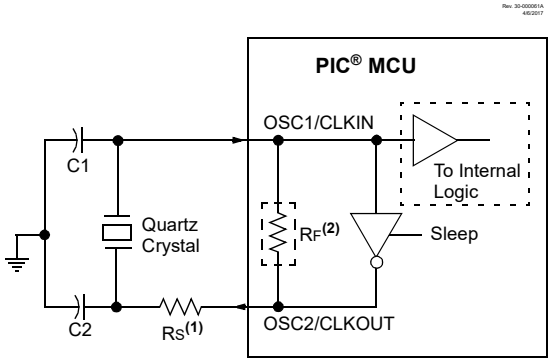
LP Oscillator mode selects the lowest gain setting of the internal inverter-amplifier. LP mode current consumption is the least of the three modes. This mode is designed to drive only 32.768 kHz tuning-fork type crystals (watch crystals).

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 500 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.

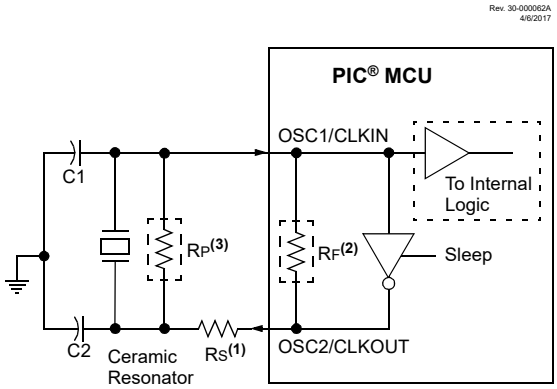
Figure 4-3. Quartz Crystal Operation (LP, XT or HS Mode)



Note:

1. A series resistor (R_S) may be required for quartz crystals with low drive level.
2. The value of R_F varies with the Oscillator mode selected (typically between 2 MΩ to 10 MΩ).

Figure 4-4. Ceramic Resonator Operation (XT or HS Mode)



Note:

1. A series resistor (R_S) may be required for ceramic resonators with low drive level.
2. The value of R_F varies with the Oscillator mode selected (typically between 2 MΩ to 10 MΩ).
3. An additional parallel feedback resistor (R_P) may be required for proper ceramic resonator operation.

4.2.1.3 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.

4.2.1.4 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.

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 to '010' (enable EXTOSC with 4x PLL).

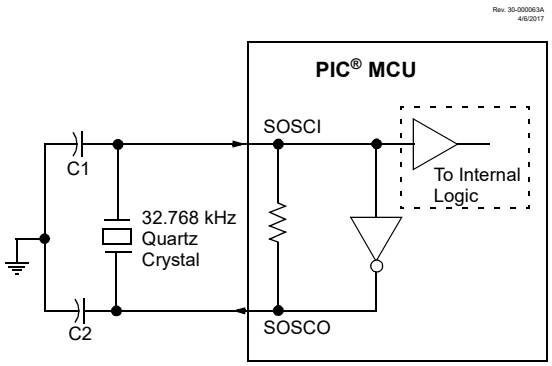
Related Links

38.4.3 PLL Specifications

4.2.1.5 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 SOSCI 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.

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 V_{DD} 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)

Related Links

4.3 Clock Switching

4.2.2 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 as the default system clock upon a device Reset.
- Write the NOSC<2:0> bits to switch the system clock source to the internal oscillator during run-time.

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 $\overline{\text{CLKOUTEN}}$ 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.

Related Links

4.3 Clock Switching

4.6.5 OSCFRQ

4.6.6 OSCTUNE

4.2.2.1 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' (F_{OSC} = 1 MHz) or '000' (F_{OSC} = 64 MHz) to set the oscillator upon device Power-up or Reset.
- Write to the NOSC<2:0> bits during run-time.

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

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

Related Links

4.3 Clock Switching

4.2.2.2 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.

4.2.2.3 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 during run-time.

Related Links

4.3 Clock Switching

4.2.2.4 ADCRC (also referred to as FRC)

The ADCRC is an oscillator dedicated to the ADC^2 module. The ADCRC oscillator can be manually enabled using the ADOEN bit. The ADCRC runs at a fixed frequency of 600 kHz. ADCRC is automatically enabled if it is selected as the clock source for the ADC^2 module.

4.2.3 Oscillator Status and Adjustments

4.2.3.1 Internal Oscillator Frequency Adjustment

The internal oscillator is factory-calibrated. This internal oscillator can be adjusted in software by writing to the OSCTUNE register.

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.

Related Links

4.6.6 OSCTUNE

4.2.3.2 Oscillator Status and Manual Enable

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

Related Links

4.6.4 OSCSTAT
4.6.7 OSCEN

4.2.3.3 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.

4.3 Clock Switching

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

- External oscillator
- Internal Oscillator Block (INTOSC)

Important: 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.

4.3.1 New Oscillator Source (NOSC) and New Divider Selection Request (NDIV) Bits

The New Oscillator Source (NOSC) and New Divider Selection Request (NDIV) bits 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 source will be ready immediately. The device may enter Sleep while waiting for the switch.

When the new oscillator is ready, the New Oscillator Ready (NOSCR) bit 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 can also be polled to determine when the oscillator is ready in lieu of an interrupt.

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

If the Clock Switch Hold (CSWHOLD) bit 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 F_{OSC} 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.

Related Links

4.3.3 Clock Switch and Sleep

4.3.2 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.

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

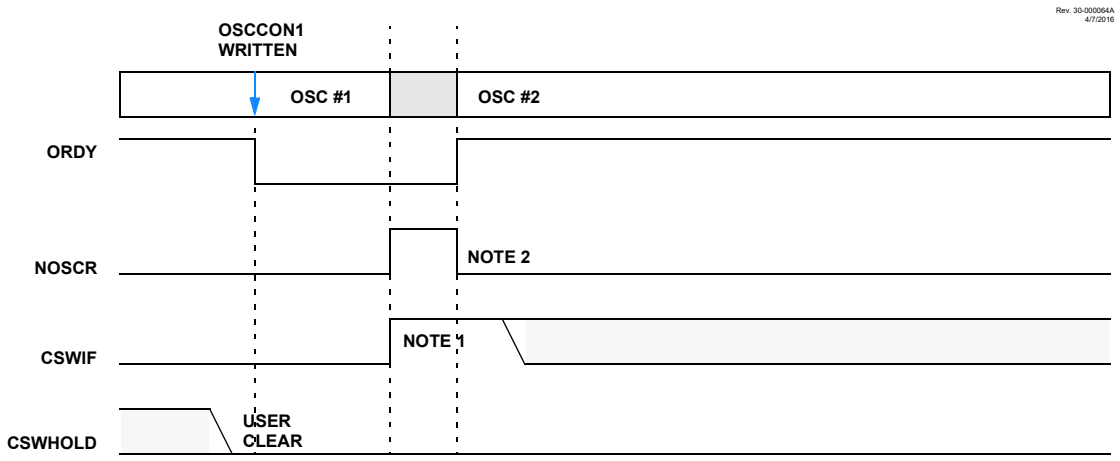
4.3.3 Clock Switch and Sleep

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.

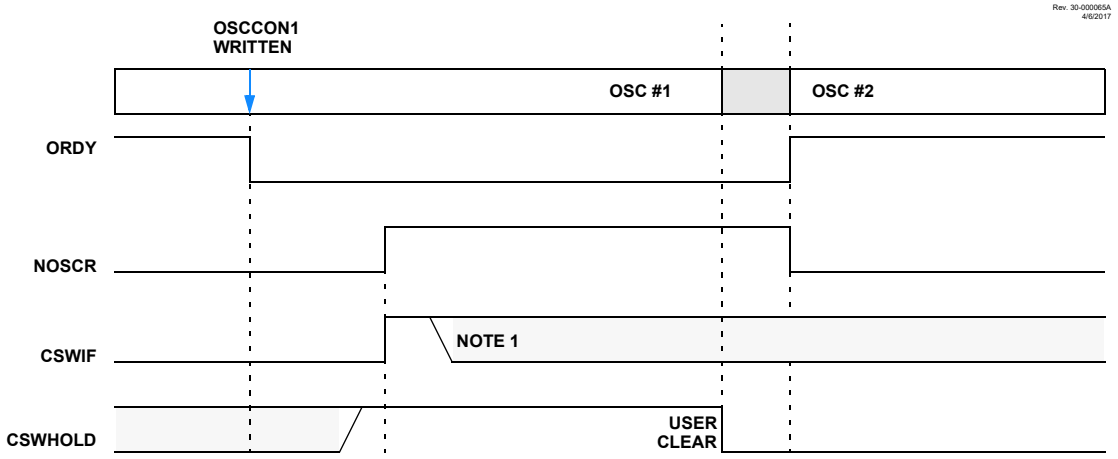
Figure 4-6. Clock Switch (CSWHOLD = 0)



Note:

1. CSWIF is asserted coincident with NOSCR; interrupt is serviced at OSC#2 speed.
2. The assertion of NOSCR is hidden from the user because it appears only for the duration of the switch.

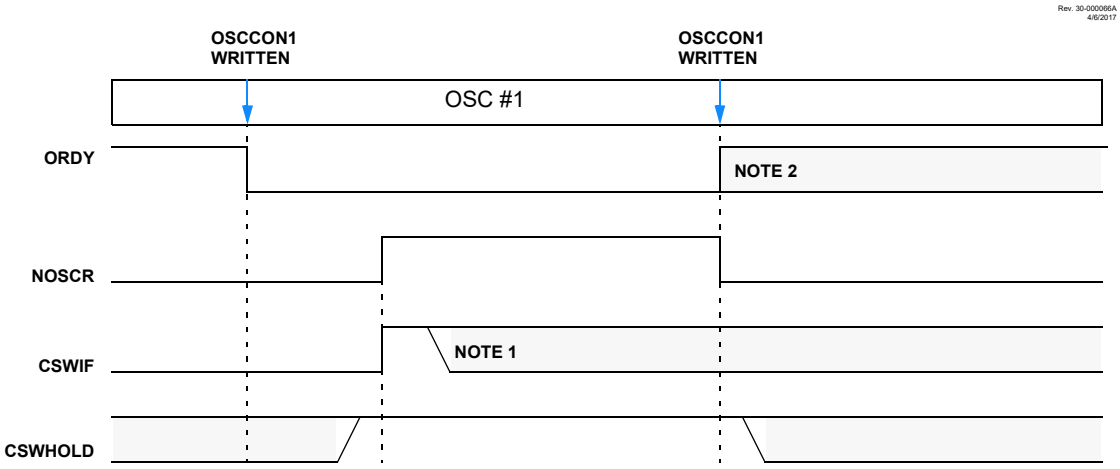
Figure 4-7. Clock Switch (CSWHOLD = 1)



Note:

1. CSWIF is asserted coincident with NOSCR, and may be cleared before or after clearing CSWHOLD = 0.

Figure 4-8. Clock Switch Abandoned



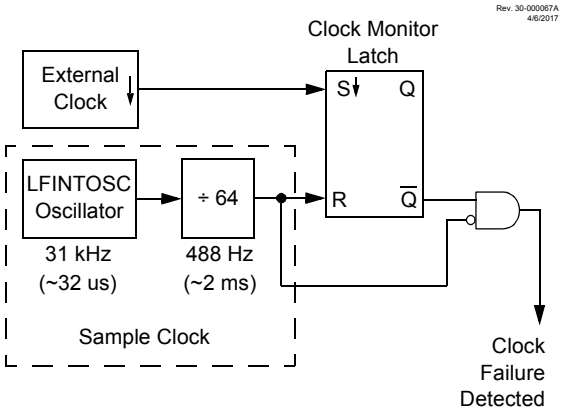
Note:

1. CSWIF may be cleared before or after rewriting OSCCON1; CSWIF is not automatically cleared.
2. ORDY = 0 if OSCCON1 does not match OSCCON2; a new switch will begin.

4.4 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



4.4.1 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.

4.4.2 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.

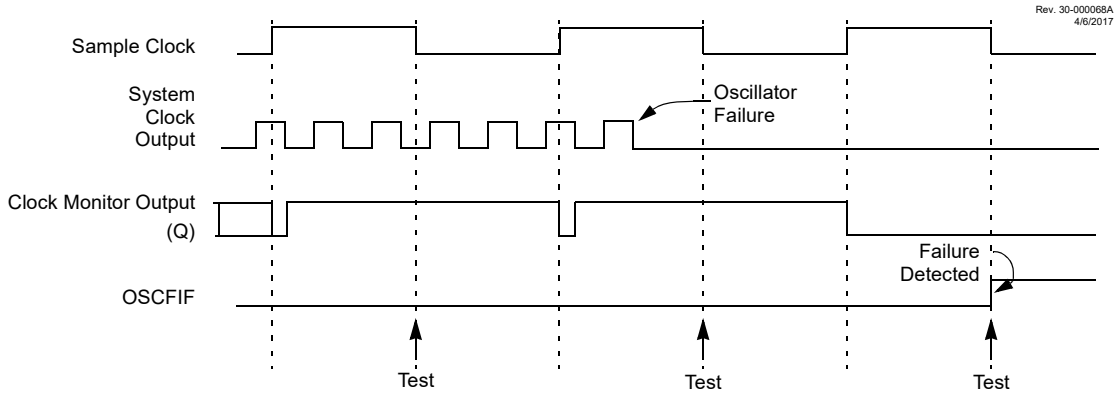
4.4.3 Fail-Safe Condition Clearing

The Fail-Safe condition is cleared after a Reset, executing a `SLEEP` instruction or changing the NOSC and NDIV bits. 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.

4.4.4 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



Note: The system clock is normally at a much higher frequency than the sample clock. The relative frequencies in this example have been chosen for clarity.

4.5 Register Summary - OSC

Address	Name	Bit Pos.								
0x0ED3	OSCCON1	7:0		NOSC[2:0]			NDIV[3:0]			
0x0ED4	OSCCON2	7:0		COSC[2:0]			CDIV[3:0]			
0x0ED5	OSCCON3	7:0	CSWHOLD	SOSCPWR		ORDY	NOSCR			
0x0ED6	OSCSTAT	7:0	EXTOR	HFOR	MFOR	LFOR	SOR	ADOR		PLLR
0x0ED7	OSCEN	7:0	EXTOEN	HFOEN	MFOEN	LFOEN	SOSCEN	ADOEN		
0x0ED8	OSCTUNE	7:0			HFTUN[5:0]					
0x0ED9	OSCFRQ	7:0					HFFRQ[3:0]			

4.6 Register Definitions: Oscillator Control

4.6.1 OSCCON1

Name: OSCCON1
Address: 0xED3

Oscillator Control Register1

Bit	7	6	5	4	3	2	1	0
		NOSC[2:0]			NDIV[3:0]			
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		f	f	f	f	f	f	f

Bits 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.

Table 4-1. Default Oscillator Settings

CONFIG1[RSTOSC]	SFR Reset Values (fff ffff)			Initial F_{OSC} Frequency
	NOSC/COSC	NDIV/CDIV	OSCFRQ	
111	111	0000	4 MHz	EXTOSC per FEXTOSC
110	110	0010		F_{OSC} = 1 MHz (4 MHz/4)
101	101	0000		LFINTOSC
100	100	0000		SOSC
011	Reserved			
010	010	0000	4 MHz	EXTOSC + 4xPLL(4)
001	Reserved			
000	110	0000	64 MHz	F_{OSC} = 64 MHz

Table 4-2. NOSC Bit Settings

NOSC<2:0>	Clock Source
111	EXTOSC(5)
110	HFINTOSC(6)
101	LFINTOSC
100	SOSC
011	Reserved
010	EXTOSC + 4x PLL(7)
001	Reserved
000	Reserved

Bits 3:0 – NDIV[3:0] New Divider Selection Request bits(2,3)
The setting determines the new postscaler division ratio per Table 4-3.

Table 4-3. NDIV Bit Settings

NDIV<3:0>	Clock Divider
1111-1010	Reserved
1001	512
1000	256
0111	128
0110	64
0101	32
0100	16
0011	8
0010	4
0001	2
0000	1

Note:

1. The default value (f/f) is determined by the CONFIG1[RSTOSC] Configuration bits. See Table 4-1.
2. If NOSC is written with a reserved value (Table 4-2), the operation is ignored and NOSC is not written.
3. When CONFIG1[CSWEN] = 0, this register is read-only and cannot be changed from the POR value.
4. EXTOSC must meet the PLL specifications.
5. EXTOSC configured by CONFIG1[FEXTOSC].
6. HFINTOSC frequency is set with the 4.6.5.1 HFFRQ bits.
7. EXTOSC must meet the PLL specifications.

Related Links

3.7.1 CONFIG1

38.4.3 PLL Specifications

4.6.2 OSCCON2

Name: OSCCON2
Address: 0xED4

Oscillator Control Register 2

Bit	7	6	5	4	3	2	1	0
		COSC[2:0]			CDIV[3:0]			
Access		R	R	R	R	R	R	R
Reset		q	q	q	q	q	q	q

Bits 6:4 – COSC[2:0] Current Oscillator Source Select bits (read-only)(1,2)
Indicates the current source oscillator and PLL combination as shown in the following table.

Table 4-4. COSC Bit Settings

COSC/NOSC	Clock Source
111	EXTOSC(3)
110	HFINTOSC(4)
101	LFINTOSC
100	SOSC
011	Reserved
010	EXTOSC + 4x PLL(5)
001	Reserved
000	Reserved

Bits 3:0 – CDIV[3:0] Current Divider Select bits (read-only)(1,2)
Indicates the current postscaler division ratio as shown in the follwing table.

Table 4-5. CDIV Bit Settings

CDIV/NDIV	Clock Divider
1111-1010	Reserved
1001	512
1000	256
0111	128
0110	64
0101	32
0100	16
0011	8
0010	4

CDIV/NDIV	Clock Divider
0001	2
0000	1

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 4.6.1.1 NOSC/4.6.1.2 NDIV bits.
3. EXTOSC configured by the CONFIG1[FEXTOSC] bits.
4. HFINTOSC frequency is set with the 4.6.5.1 HFFRQ bits.
5. EXTOSC must meet the PLL specifications.

Related Links

3.7.1 CONFIG1

38.4.3 PLL Specifications

4.6.3 OSCCON3

Name: OSCCON3
Address: 0xED5

Oscillator Control Register 3

Bit	7	6	5	4	3	2	1	0
	CSWHOLD	SOSCPWR		ORDY	NOSCR			
Access	R/W/HC	R/W		RO	RO			
Reset	0	0		0	0			

Bit 7 – CSWHOLD Clock Switch Hold bit

Value	Description
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; when NOSCR becomes '1', the switch will occur

Bit 6 – SOSCPWR Secondary Oscillator Power Mode Select bit

Value	Description
1	Secondary oscillator operating in High-Power mode
0	Secondary oscillator operating in Low-Power mode

Bit 4 – ORDY Oscillator Ready bit (read-only)

Value	Description
1	OSCCON1 = OSCCON2; the current system clock is the clock specified by NOSC
0	A clock switch is in progress

Bit 3 – NOSCR New Oscillator is Ready bit (read-only)(1)

Value	Description
1	A clock switch is in progress and the oscillator selected by NOSC indicates a ready condition
0	A clock switch is not in progress, or the NOSC-selected oscillator is not yet ready

Note:

1. If CSWHOLD = 0, the user may not see this bit set because the bit is set for less than one instruction cycle.

4.6.4 OSCSTAT

Name: OSCSTAT
Address: 0xED6

Oscillator Status Register 1

Bit	7	6	5	4	3	2	1	0
	EXTOR	HFOR	MFOR	LFOR	SOR	ADOR		PLLR
Access	RO	RO	RO	RO	RO	RO		RO
Reset	q	q	q	q	q	q		q

Bit 7 – EXTOR EXTOSC (external) Oscillator Ready bit

Value	Description
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

Value	Description
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 bit

Value	Description
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

Value	Description
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

Value	Description
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

Value	Description
1	The oscillator is ready to be used
0	The oscillator is not enabled, or is not yet ready to be used

Bit 0 – PLLR PLL Ready bit

Value	Description
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.

4.6.5 OSCFRQ

Name: OSCFRQ
Address: 0xED9

HFINTOSC Frequency Selection Register

Bit	7	6	5	4	3	2	1	0
					HFFRQ[3:0]			
Access					R/W	R/W	R/W	R/W
Reset					q	q	q	q

Bits 3:0 – HFFRQ[3:0] HFINTOSC Frequency Selection bits

HFFRQ	Nominal Freq (MHz)
1001	Reserved
1010	
1111	
1110	
1101	
1100	
1011	
1000(1)	64
0111	48
0110	32
0101	16
0100	12
0011	8
0010(1)	4
0001	2
0000	1

Note:

1. Refer to Table 4-1 for more information.

4.6.6 OSCTUNE

Name: OSCTUNE
Address: 0xED8

HFINTOSC Tuning Register

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

Bits 5:0 – HFTUN[5:0] HFINTOSC Frequency Tuning bits

Value	Description
01 1111	Maximum frequency
00 0000	Center frequency. Oscillator module is running at the calibrated frequency (default value).
10 0000	Minimum frequency

4.6.7 OSCEN

Name: OSCEN
Address: 0xED7

Oscillator Manual Enable Register

Bit	7	6	5	4	3	2	1	0
	EXTOEN	HFOEN	MFOEN	LFOEN	SOSCEN	ADOEN		
Access	R/W	R/W	R/W	R/W	R/W	R/W		
Reset	0	0	0	0	0	0		

Bit 7 – EXTOEN External Oscillator Manual Request Enable bit

Value	Description
1	EXTOSC is explicitly enabled, operating as specified by CONFIG1[FEXTOSC]
0	EXTOSC is only enabled if requested by a peripheral

Bit 6 – HFOEN HFINTOSC Oscillator Manual Request Enable bit

Value	Description
1	HFINTOSC is explicitly enabled, operating as specified by OSCFRQ
0	HFINTOSC is only enabled if requested by a peripheral

Bit 5 – MFOEN MFINTOSC (500 kHz/31.25 kHz) Oscillator Manual Request Enable bit (Derived from HFINTOSC)

Value	Description
1	MFINTOSC is explicitly enabled
0	MFINTOSC is only enabled if requested by a peripheral

Bit 4 – LFOEN LFINTOSC (31 kHz) Oscillator Manual Request Enable bit

Value	Description
1	LFINTOSC is explicitly enabled
0	LFINTOSC is only enabled if requested by a peripheral

Bit 3 – SOSCEN Secondary Oscillator Manual Request Enable bit

Value	Description
1	Secondary Oscillator is explicitly enabled, operating as specified by SOSCPWR
0	Secondary Oscillator is only enabled if requested by a peripheral

Bit 2 – ADOEN ADC Oscillator Manual Request Enable bit

Value	Description
1	ADC oscillator is explicitly enabled
0	ADC oscillator is only enabled if requested by a peripheral

5. 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 routed internally 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

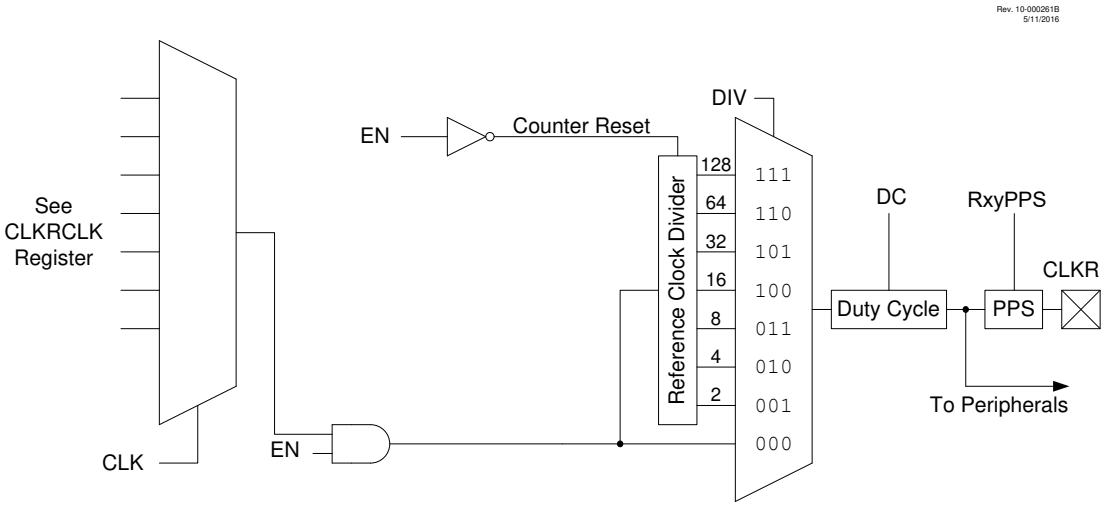
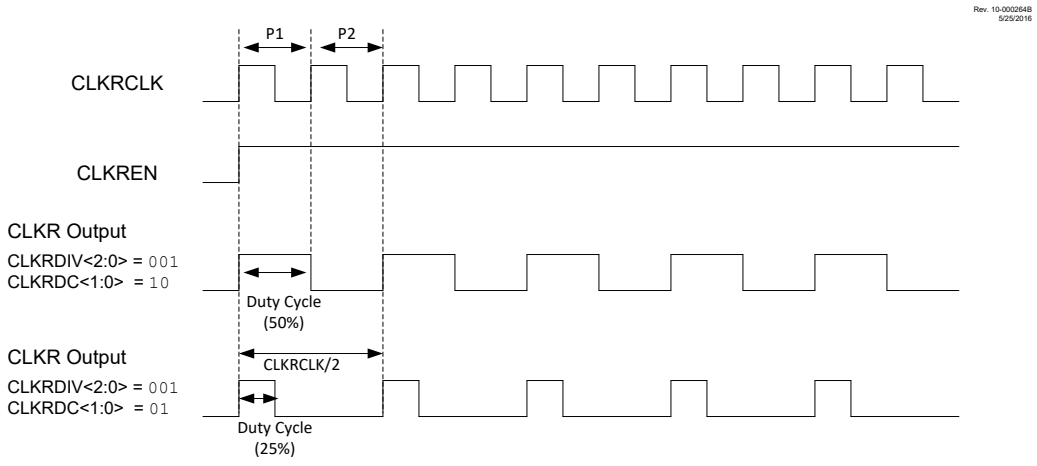


Figure 5-2. Clock Reference Timing



5.1 Clock Source

The clock source of the reference clock peripheral is selected with the CLK bits. The available clock sources are listed in the following table:

Table 5-1. CLKR Clock Sources

CLK	Clock Source
111-101	Unimplemented
100	SOSC
011	MFINTOSC (500 kHz)
010	LFINTOSC (31 kHz)
001	HFINTOSC
000	F_{OSC}

5.1.1 Clock Synchronization

The CLKR output signal is ensured to be glitch-free when the 5.6.1.1 EN bit is set to start the module and enable the CLKR output.

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

Clock dividers and clock duty cycles can be changed while the module is enabled but doing so may cause glitches to occur on the output. To avoid possible glitches, clock dividers and clock duty cycles should be changed only when the 5.6.1.1 EN bit is clear.

5.2 Programmable Clock Divider

The module takes the clock input and divides it based on the value of the 5.6.1.3 DIV bits.

The following configurations are available:

- Base Fosc value
- F_{OSC} divided by 2
- F_{OSC} divided by 4
- F_{OSC} divided by 8
- F_{OSC} divided by 16
- F_{OSC} divided by 32
- F_{OSC} divided by 64
- F_{OSC} 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 5.6.1.3 DIV bits should only be changed when the module is disabled (5.6.1.1 EN = 0).

5.3 Selectable Duty Cycle

The 5.6.1.2 DC bits are used to modify the duty cycle of the output clock. A duty cycle of 0%, 25%, 50%, or 75% can be selected for all clock rates when the 5.6.1.3 DIV value is not 0b000. When DIV=0b000

then the duty cycle defaults to 50% for all values of DC except 0b00 in which case the duty cycle is 0% (constant low output).

The duty cycle can be changed while the module is enabled. However, in order to prevent glitches on the output, the 5.6.1.2 DC bits should only be changed when the module is disabled (5.6.1.1 EN = 0).



Important: The 5.6.1.2 DC value at reset is 10. This makes the default duty cycle 50% and not 0%.

5.4 Operation in Sleep Mode

The reference clock module continues to operate and provide a signal output in Sleep for all clock source selections except F_{OSC} (5.6.2.1 CLK=0).

5.5 Register Summary: Reference CLK

Address	Name	Bit Pos.								
0x0F39	CLKRCON	7:0	EN			DC[1:0]		DIV[2:0]		
0x0F3A	CLKRCLK	7:0						CLK[2:0]		

5.6 Register Definitions: Reference Clock

Long bit name prefixes for the Reference Clock peripherals are shown in the following table. Refer to the "*Long Bit Names*" section for more information.

Table 5-2. TABLE 5-1:

Peripheral	Bit Name Prefix
CLKR	CLKR

Related Links

1.4.2.2 Long Bit Names

5.6.1 CLKRCON

Name: CLKRCON
Address: 0xF39

Reference Clock Control Register

Bit	7	6	5	4	3	2	1	0
	EN			DC[1:0]		DIV[2:0]		
Access	R/W			R/W	R/W	R/W	R/W	R/W
Reset	0			1	0	0	0	0

Bit 7 – EN
Reference Clock Module Enable bit

Value	Description
1	Reference clock module enabled
0	Reference clock module is disabled

Bits 4:3 – DC[1:0]
Reference Clock Duty Cycle bits(1)

Value	Description
11	Clock outputs duty cycle of 75%
10	Clock outputs duty cycle of 50%
01	Clock outputs duty cycle of 25%
00	Clock outputs duty cycle of 0%

Bits 2:0 – DIV[2:0]
Reference Clock Divider bits

Value	Description
111	Base clock value divided by 128
110	Base clock value divided by 64
101	Base clock value divided by 32
100	Base clock value divided by 16
011	Base clock value divided by 8
010	Base clock value divided by 4
001	Base clock value divided by 2
000	Base clock value

Note:

1. Bits are valid for reference clock divider values of two or larger, the base clock cannot be further divided.

5.6.2 CLKRCLK

Name: CLKRCLK
Address: 0xF3A

Clock Reference Clock Selection MUX

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

Bits 2:0 – CLK[2:0] CLKR Clock Selection bits
See the Clock Sources table.

6. 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 bit is set (6.6.2.2 DOZEN = 1), the CPU executes only one instruction cycle out of every N cycles as defined by the 6.6.2.5 DOZE bits. For example, if DOZE = 001, the instruction cycle ratio is 1:4. The CPU and memory execute for one instruction cycle and then lay idle for three instruction cycles. During the unused cycles, the peripherals continue to operate at the system clock speed.

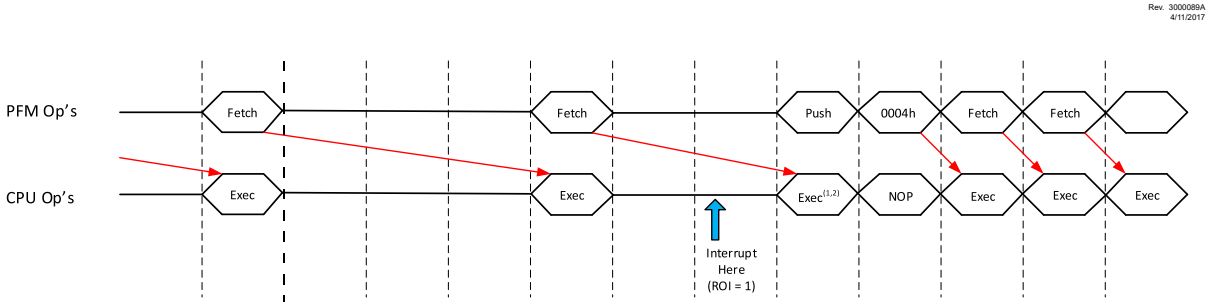
6.1.1 Doze Operation

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

- Doze enabled (6.6.2.2 DOZEN = 1)
- 6.6.2.5 DOZE = 001 (1:4) ratio
- Recover-on-Interrupt enabled (6.6.2.3 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)



Note 1: Multi-cycle instructions are executed to completion before fetching 0004h.
2: If the pre-fetched instruction clears GIE, the ISR will not occur, but DOZEN is still cleared and the CPU will resume execution at full speed.

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.

Figure 6-2. 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
3. The $\overline{TO}$ bit of the STATUS register is set
4. The CPU clock is disabled
5. LFINTOSC, SOSC, HFINTOSC and ADCRC are unaffected 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 V_{DD} or V_{SS} 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.

Related Links

30. (DAC) 5-Bit Digital-to-Analog Converter Module

28. (FVR) Fixed Voltage Reference

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 "*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)

Related Links

8.11 Determining the Cause of a Reset

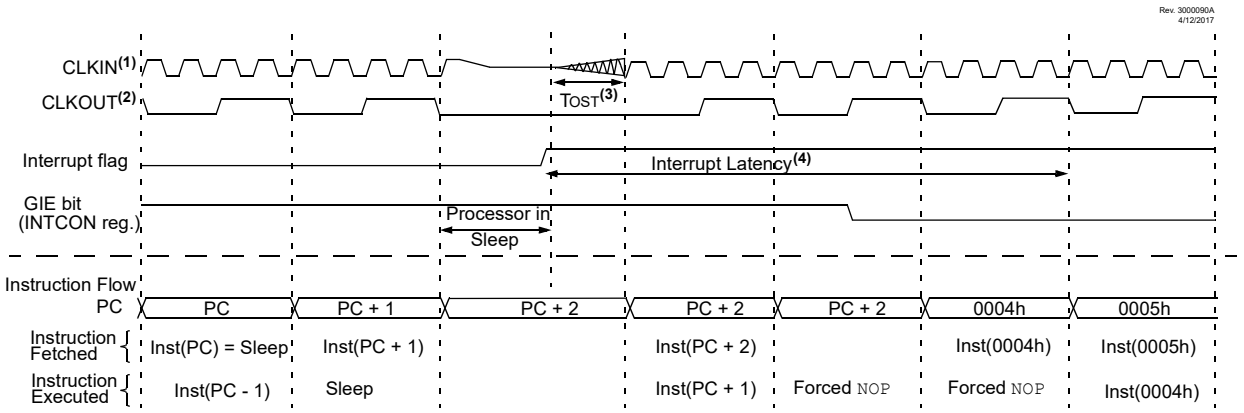
6.2.2 Wake-up Using Interrupts

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
  - $\overline{TO}$ bit of the STATUS register will not be set
  - $\overline{PD}$ 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
  - $\overline{TO}$ bit of the STATUS register will be set
  - $\overline{PD}$ 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 $\overline{PD}$ bit. If the $\overline{PD}$ bit is set, the SLEEP instruction was executed as a NOP.

Figure 6-3. WAKE-UP FROM SLEEP THROUGH INTERRUPT



Note:

1. External clock. High, Medium, Low mode assumed.
2. CLKOUT is shown here for timing reference.
3. T_{OST} = 1024 T_{OSC}. This delay does not apply to EC and INTOSC Oscillator modes.
4. GIE = 1 assumed. In this case after wake-up, the processor calls the ISR at 0004h. If GIE = 0, execution will continue in-line.

6.2.3 Low-Power Sleep Mode

The PIC18F24/25Q10 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 PIC18F24/25Q10devices 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.

6.3 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.



Important: If $\overline{\text{CLKOUTEN}}$ is enabled ($\overline{\text{CLKOUTEN}}$ = 0, Configuration Word 1H), the output will continue operating while in Idle.

6.3.1 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.

6.3.2 Idle and WWDT

When in Idle, the WWDT Reset is blocked and will instead wake the device. The WWDT wake-up is not an interrupt, therefore ROI does not apply.



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

6.4 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 F_{OSC} and $F_{OSC}/4$ clocks are unavailable. All the other clock sources are active, if enabled manually or through peripheral clock selection before the part enters Sleep.

6.5 Register Summary - Power Savings Control

Address	Name	Bit Pos.								
0x0ED2	CPUDOZE	7:0	IDLEN	DOZEN	ROI	DOE		DOZE[2:0]		
0x0ED3 ... 0x0ED9	Reserved									
0x0EDA	VREGCON	7:0		PMSYS[1:0]					VREGPM[1:0]	

6.6 Register Definitions: Power Savings Control

6.6.1 VREGCON

Name: VREGCON
Address: 0xEDA

Note:

1. System and peripheral inputs should not exceed 500 kHz in ULP mode.

Voltage Regulator Control Register

Bit	7	6	5	4	3	2	1	0
		PMSYS[1:0]					VREGPM[1:0]	
Access		RO	RO				R/W	R/W
Reset		g	g				1	0

Bits 6:5 – PMSYS[1:0] System Power Mode Status bits

Value	Description
11	Reserved
10	ULP regulator is active
01	Main regulator in LP mode is active
00	Main regulator in HP mode is active

Bits 1:0 – VREGPM[1:0] Voltage Regulator Power Mode Selection bit

Value	Description
11	Reserved. Do not use.
10	ULP regulator(1)
01	Main regulator in LP mode
00	Main regulator in HP mode

6.6.2 CPUDOZE

Name: CPUDOZE
Address: 0xED2

Doze and Idle Register

Bit	7	6	5	4	3	2	1	0
	IDLEN	DOZEN	ROI	DOE		DOZE[2:0]		
Access	R/W	R/W/HC/HS	R/W	R/W		R/W	R/W	R/W
Reset	0	0	0	0		0	0	0

Bit 7 – IDLEN Idle Enable bit
Reset States: POR/BOR = 0
All Other Resets = u

Value	Description
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)

Value	Description
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

Value	Description
1	Entering the Interrupt Service Routine (ISR) makes DOZEN = 0, bringing the CPU to full-speed operation
0	Interrupt entry does not change DOZEN

Bit 4 – DOE Doze-On-Exit bit

Value	Description
1	Executing RETFIE makes DOZEN = 1, bringing the CPU to reduced speed operation
0	RETFIE does not change DOZEN

Bits 2:0 – DOZE[2:0] Ratio of CPU Instruction Cycles to Peripheral Instruction Cycles

Value	Description
111	1:256
110	1:128
101	1:64
100	1:32
011	1:16
010	1:8

Value	Description
001	1:4
000	1:2

Note:

1. When ROI = 1 or DOE = 1, DOZEN is changed by hardware interrupt entry and/or exit.

7. (PMD) Peripheral Module Disable

This module provides the ability to selectively enable or disable a peripheral. Disabling a peripheral places it in its lowest possible power state. The user can disable unused modules to reduce the overall power consumption.

The PIC18F24/25Q10 devices address this requirement by allowing peripheral modules to be selectively enabled or disabled. Disabling a peripheral places it in the lowest possible power mode.

Important: All modules are ON by default following any system Reset.

7.1 Disabling a Module

A peripheral can be disabled by setting the corresponding peripheral disable bit in the PMDx register. Disabling a module has the following effects:

- The module is held in Reset and does not function.
- All the SFRs pertaining to that peripheral become “unimplemented”
  - Writing is disabled
  - Reading returns 0x00
- Module outputs are disabled

Related Links

15.1 I/O Priorities

7.2 Enabling a Module

Clearing the corresponding module disable bit in the PMDx register, re-enables the module and the SFRs will reflect the Power-on Reset values.

Important: There should be no reads/writes to the module SFRs for at least two instruction cycles after it has been re-enabled.

7.3 Register Summary - PMD

Address	Name	Bit Pos.								
0x0EDC	PMD0	7:0	SYSCMD	FVRMD	HLVDMD	CRCMD	SCANMD	NVMMD	CLKRMD	IOCMD
0x0EDD	PMD1	7:0		TMR6MD	TMR5MD	TMR4MD	TMR3MD	TMR2MD	TMR1MD	TMR0MD
0x0EDE	PMD2	7:0		DACMD	ADCMD			CMP2MD	CMP1MD	ZCDMD
0x0EDF	PMD3	7:0					PWM4MD	PWM3MD	CCP2MD	CCP1MD
0x0EE0	PMD4	7:0		UART1MD		MSSP1MD				CWG1MD
0x0EE1	PMD5	7:0								DSMMD

7.4 Register Definitions: Peripheral Module Disable

7.4.1 PMD0

Name: PMD0
Address: 0xEDC

PMD Control Register 0

Bit	7	6	5	4	3	2	1	0
	SYSCMD	FVRMD	HLVDMD	CRCMD	SCANMD	NVMMD	CLKRMD	IOCMD
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 7 – SYSCMD Disable Peripheral System Clock Network bit
Disables the System clock network(1)

Value	Description
1	System clock network disabled (F_{OSC})
0	System clock network enabled

Bit 6 – FVRMD Disable Fixed Voltage Reference bit

Value	Description
1	FVR module disabled
0	FVR module enabled

Bit 5 – HLVDMD Disable High-Low-Voltage Detect bit

Value	Description
1	HLVD module disabled
0	HLVD module enabled

Bit 4 – CRCMD Disable CRC Engine bit

Value	Description
1	CRC module disabled
0	CRC module enabled

Bit 3 – SCANMD Disable NVM Memory Scanner bit
Disables the Scanner module(2)

Value	Description
1	NVM Memory Scan module disabled
0	NVM Memory Scan module enabled

Bit 2 – NVMMD NVM Module Disable bit
Disables the NVM module(3)

Value	Description
1	All Memory reading and writing is disabled; NVMCON registers cannot be written
0	NVM module enabled

Bit 1 – CLKRMD Disable Clock Reference bit

Value	Description
1	CLKR module disabled
0	CLKR module enabled

Bit 0 – IOCMD Disable Interrupt-on-Change bit, All Ports

Value	Description
1	IOC module(s) disabled
0	IOC module(s) enabled

Note:

1. Clearing the SYSCMD bit disables the system clock (F_{OSC}) to peripherals, however peripherals clocked by $F_{OSC}/4$ are not affected.
2. Subject to SCANE bit in *Configuration Word 4*.
3. When enabling NVM, a delay of up to 1 µs is required before accessing data.

Related Links

3.7.4 CONFIG4

7.4.2 PMD1

Name: PMD1
Address: 0xEDD

PMD Control Register 1

Bit	7	6	5	4	3	2	1	0
		TMR6MD	TMR5MD	TMR4MD	TMR3MD	TMR2MD	TMR1MD	TMR0MD
Access		R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bits 0, 1, 2, 3, 4, 5, 6 – TMRnMD Disable Timer n bit

Value	Description
1	TMRn module disabled
0	TMRn module enabled

7.4.3 PMD2

Name: PMD2
Address: 0xEDE

PMD Control Register 2

Bit	7	6	5	4	3	2	1	0
		DACMD	ADCMD			CMP2MD	CMP1MD	ZCDMD
Access		R/W	R/W			R/W	R/W	R/W
Reset		0	0			0	0	0

Bit 6 – DACMD Disable DAC bit

Value	Description
1	DAC module disabled
0	DAC module enabled

Bit 5 – ADCMD Disable ADC bit

Value	Description
1	ADC module disabled
0	ADC module enabled

Bits 1, 2 – CMPnMD Disable Comparator CMPn bit

Value	Description
1	CMPn module disabled
0	CMPn module enabled

Bit 0 – ZCDMD Disable Zero-Cross Detect module bit(1)

Value	Description
1	ZCD module disabled
0	ZCD module enabled

Note:

1. Subject to $\overline{\text{ZCD}}$ bit in *Configuration Word 2*.

Related Links

3.7.2 CONFIG2

7.4.4 PMD3

Name: PMD3
Address: 0xEDF

PMD Control Register 3

Bit	7	6	5	4	3	2	1	0
					PWM4MD	PWM3MD	CCP2MD	CCP1MD
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

Bit 3 – PWM4MD Disable Pulse-Width Modulator PWM4 bit

Value	Description
1	PWM4 module disabled
0	PWM4 module enabled

Bit 2 – PWM3MD Disable Pulse-Width Modulator PWM3 bit

Value	Description
1	PWM3 module disabled
0	PWM3 module enabled

Bit 1 – CCP2MD Disable Pulse-Width Modulator CCP2 bit

Value	Description
1	CCP2 module disabled
0	CCP2 module enabled

Bit 0 – CCP1MD Disable Pulse-Width Modulator CCP1 bit

Value	Description
1	CCP1 module disabled
0	CCP1 module enabled

7.4.5 PMD4

Name: PMD4
Address: 0xEE0

PMD Control Register 4

Bit	7	6	5	4	3	2	1	0
		UART1MD		MSSP1MD				CWG1MD
Access		R/W		R/W				R/W
Reset		0		0				0

Bit 6 – UART1MD Disable EUSART1 bit

Value	Description
1	EUSART1 module disabled
0	EUSART1 module enabled

Bit 4 – MSSP1MD Disable MSSP1 bit

Value	Description
1	MSSP1 module disabled
0	MSSP1 module enabled

Bit 0 – CWG1MD Disable CWG1 Module bit

Value	Description
1	CWG1 module disabled
0	CWG1 module enabled

7.4.6 PMD5

Name: PMD5
Address: 0xEE1

PMD Control Register 5

Bit	7	6	5	4	3	2	1	0
								DSMMD
Access								R/W
Reset								0

Bit 0 – DSMMD Disable Data Signal Modulator bit

Value	Description
1	DSM module disabled
0	DSM module enabled

8. Resets

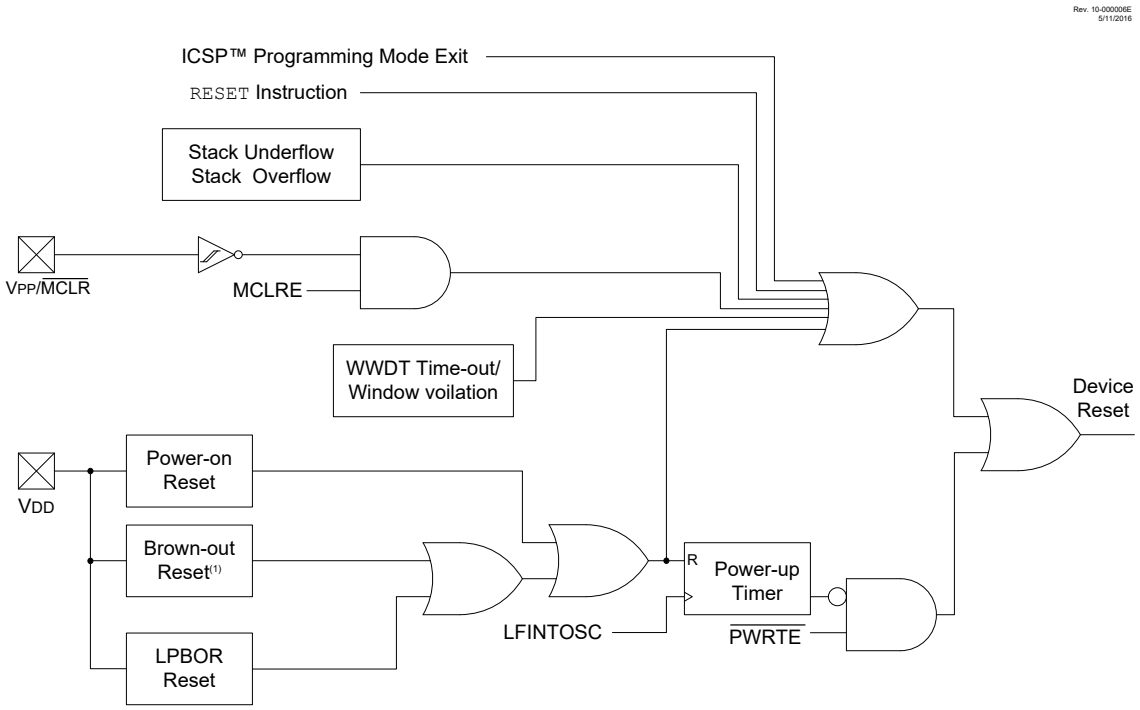
There are multiple ways to reset this device:

- Power-on Reset (POR)
- Brown-out Reset (BOR)
- Low-Power Brown-out Reset (LPBOR)
- $\overline{\text{MCLR}}$ Reset
- WDT Reset
- `RESET` instruction
- Stack Overflow
- Stack Underflow
- Programming mode exit

To allow V_{DD} 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 the block diagram below.

Figure 8-1. Simplified Block Diagram of On-Chip Reset Circuit



Note: See *"BOR Operating Conditions"* table for BOR active conditions.

8.1 Power-on Reset (POR)

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

8.2 Brown-out Reset (BOR)

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

The 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 *"BOR Operating Conditions"* table for more information.

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

A V_{DD} noise rejection filter prevents the BOR from triggering on small events. If V_{DD} falls below V_{BOR} for a duration greater than parameter T_{BORDC}, the device will reset.

Related Links

3.7.2 CONFIG2

38.4.5 Reset, WDT, Oscillator Start-up Timer, Power-up Timer, Brown-Out Reset and Low-Power Brown-Out Reset Specifications

8.2.1 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 V_{DD} is higher than the BOR threshold.

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

8.2.2 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 V_{DD} is higher than the BOR threshold.

BOR protection is not active during Sleep. The device wake-up will be delayed until the BOR is ready.

8.2.3 BOR Controlled by Software

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

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

BOR protection is unchanged by Sleep.

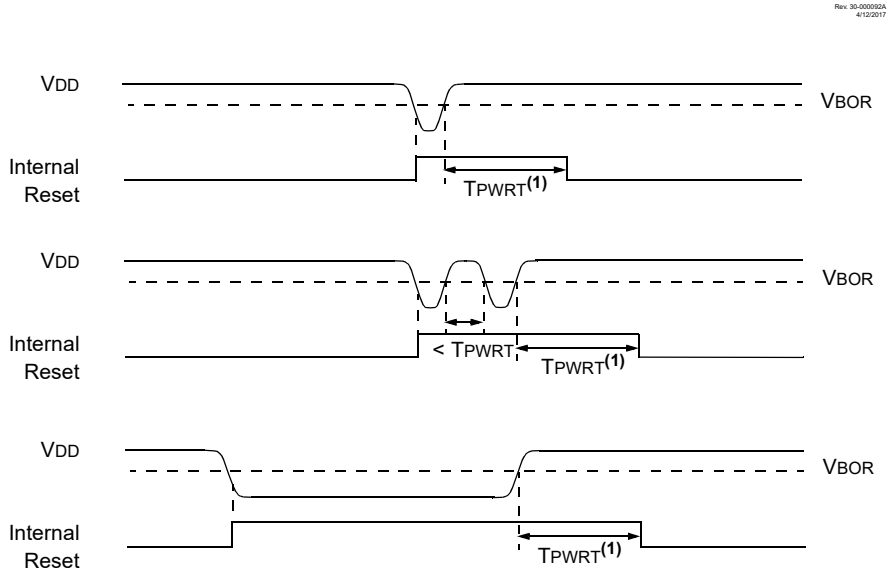
Table 8-1. BOR Operating Modes

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	

Note:

1. In this specific case, “Release of POR” and “Wake-up from Sleep”, there is no delay in start-up. The BOR ready flag, (BORRDY = 1), will be set before the CPU is ready to execute instructions because the BOR circuit is forced on by the BOREN<1:0> bits

Figure 8-2. Brown-out Situations



Note: T_{PWRT} delay only if $\overline{PWRTE}$ bit is programmed to ‘0’.

8.2.4 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 V_{DD}.

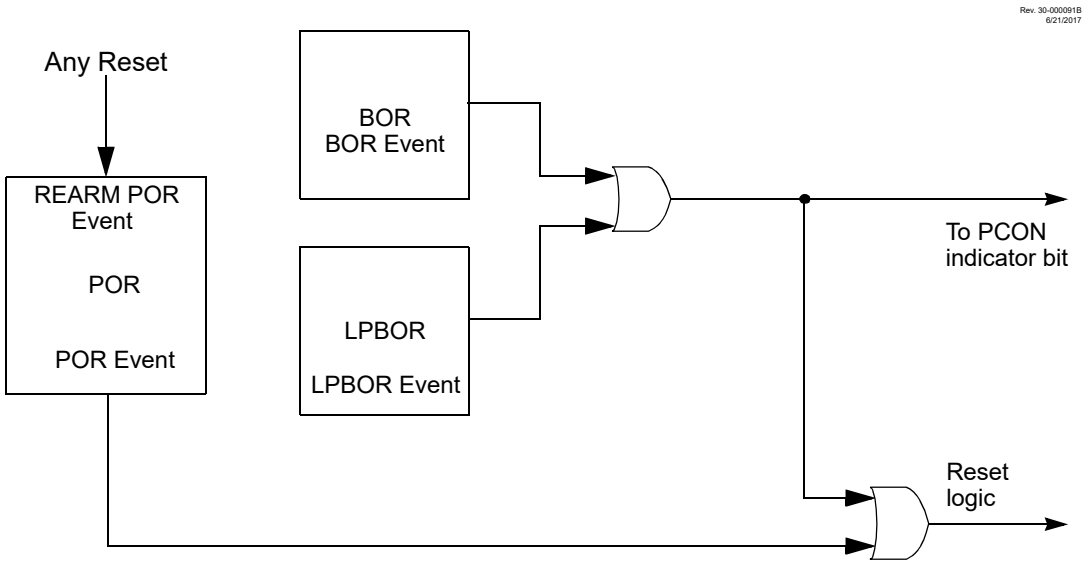
During Bulk Erase, the BOR is enabled at 1.9V, even if it is configured to some other value. If V_{DD} falls, the erase cycle will be aborted, but the device will not be reset.

8.3 Low-Power Brown-out Reset (LPBOR)

The Low-Power Brown-out Reset (LPBOR) provides an additional BOR circuit for low-power operation. Refer to the figure below to see how the BOR interacts with other modules.

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

Figure 8-3. LPBOR, BOR, POR Relationship



8.3.1 Enabling LPBOR

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

Related Links

3.7.2 CONFIG2

8.3.1.1 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 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.

8.4 $\overline{\text{MCLR}}$ Reset

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 (see table below). The $\overline{\text{RMCLR}}$ bit in the PCON0 register will be set to '0' if a $\overline{\text{MCLR}}$ has occurred.

Table 8-2. $\overline{MCLR}$ Configuration

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

8.4.1 $\overline{MCLR}$ Enabled

When $\overline{MCLR}$ is enabled and the pin is held low, the device is held in Reset. The $\overline{MCLR}$ pin is connected to V_{DD} through an internal weak pull-up.

The device has a noise filter in the $\overline{MCLR}$ Reset path. The filter will detect and ignore small pulses.

Important: An internal Reset event (RESET instruction, BOR, WWDT, POR, STKOVF, STKUNF) does not drive the $\overline{MCLR}$ pin low.

8.4.2 $\overline{MCLR}$ Disabled

When $\overline{MCLR}$ is disabled, the $\overline{MCLR}$ becomes input-only and pin functions such as internal weak pull-ups are under software control.

Related Links

15.1 I/O Priorities

8.5 Windowed Watchdog Timer (WWDT) Reset

The Windowed Watchdog Timer generates a Reset if the firmware does not issue a CLRWDT instruction within the time-out period or window set. The $\overline{TO}$ and $\overline{PD}$ bits in the STATUS register and the $\overline{RWDT}$ bit are changed to indicate a WDT Reset. The $\overline{WDTWV}$ bit indicates if the WDT Reset has occurred due to a timeout or a window violation.

Related Links

10.8.5 STATUS

9. (WWDT) Windowed Watchdog Timer

8.6 RESET Instruction

A RESET instruction will cause a device Reset. The $\overline{RI}$ bit will be set to '0'. See Table 8-3 for default conditions after a RESET instruction has occurred.

8.7 Stack Overflow/Underflow Reset

The device can reset when the Stack Overflows or Underflows. The STKOVF or STKUNF bits register indicate the Reset condition. These Resets are enabled by setting the STVREN bit in Configuration Words.

Related Links

3.7.2 CONFIG2

10.1.2.3 Stack Overflow and Underflow Resets

8.8 Programming Mode Exit

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

8.9 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 V_{DD} to rise to an acceptable level. The Power-up Timer is enabled by clearing the $\overline{\text{PWRTE}}$ 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).

8.10 Start-up Sequence

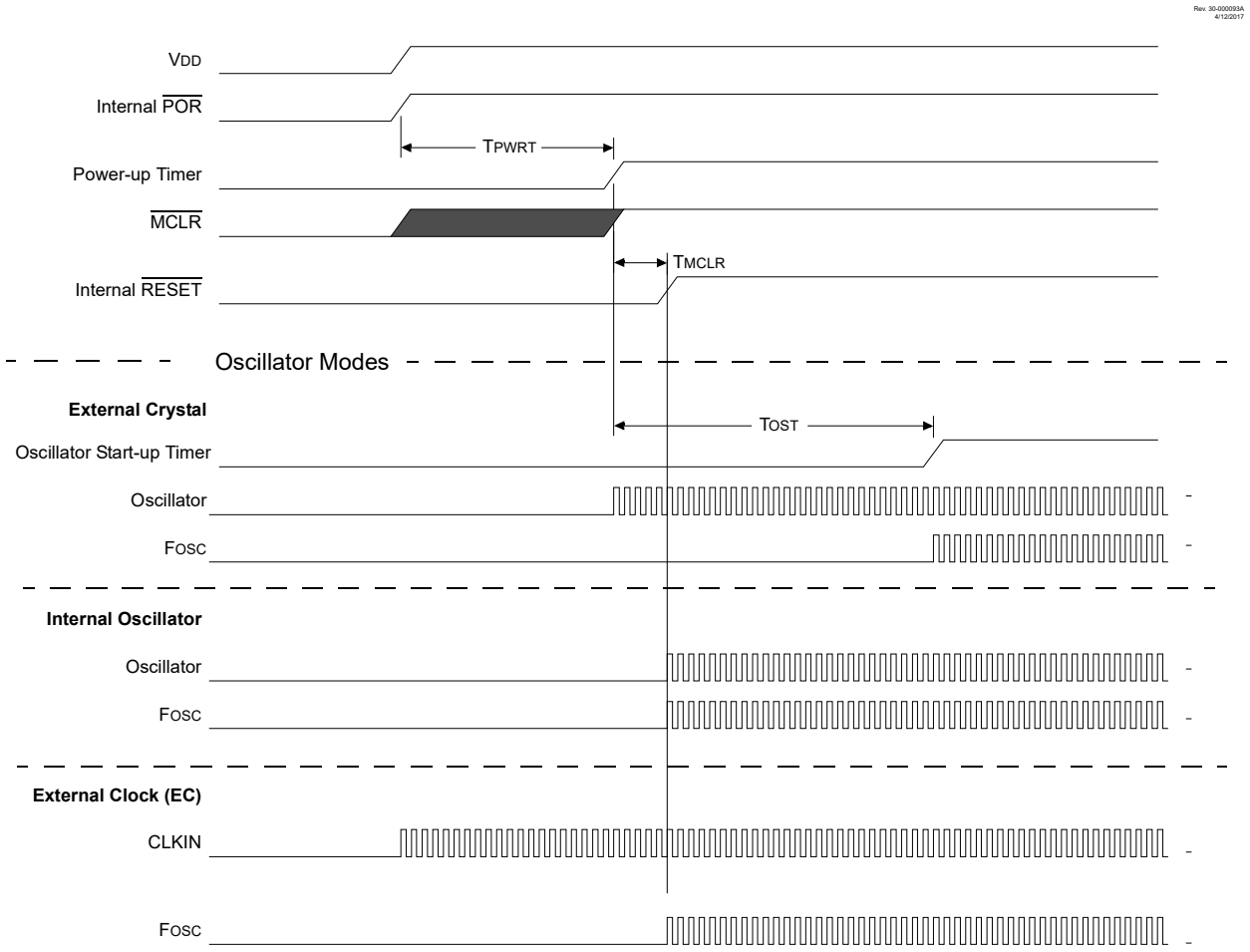
Upon the release of a POR or BOR, the following must occur before the device will begin executing:

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

The total time out will vary based on oscillator configuration and Power-up Timer configuration.

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 F_{OSC} cycles (see figure below). This is useful for testing purposes or to synchronize more than one device operating in parallel.

Figure 8-4. Reset Start-up Sequence



Related Links

4. Oscillator Module (with Fail-Safe Clock Monitor)

8.11 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. The following table shows the Reset conditions of these registers.

Table 8-3. Reset Condition for Special Registers

Condition	Program Counter	STATUS Register(2,3)	PCON0 Register	PCON1 Register
Power-on Reset	0	-110 0000	0011 110x	---- -1-1
Brown-out Reset	0	-110 0000	0011 11u0	---- -u-u
$\overline{MCLR}$ Reset during normal operation	0	-uuu uuuu	uuuu 0uuu	uuuu-u-u
$\overline{MCLR}$ Reset during Sleep	0	-10u uuuu	uuuu 0uuu	uuuu-u-u

Condition	Program Counter	STATUS Register(2,3)	PCON0 Register	PCON1 Register
WDT Time-out Reset	0	-0uu uuuu	uuu0 uuuu	uuuu-u-u
WDT Wake-up from Sleep	PC + 2	-00u uuuu	uuuu uuuu	uuuu-u-u
WWDT Window Violation Reset	0	-uuu uuuu	uu0u uuuu	uuuu-u-u
Interrupt Wake-up from Sleep	PC + 2(1)	-10u 0uuu	uuuu uuuu	uuuu-u-u
RESET Instruction Executed	0	-uuu uuuu	uuuu u0uu	uuuu-u-u
Stack Overflow Reset (STVREN = 1)	0	-uuu uuuu	1uuu uuuu	uuuu-u-u
Stack Underflow Reset (STVREN = 1)	0	-uuu uuuu	u1uu uuuu	uuuu-u-u
Data Protection (Fuse fault)	0	---u uuuu	uuuu uuuu	---- -u-0
VREG or ULP Ready fault	0	---1 1000	0011 001u	---- -0-1

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.

Related Links

10.8.5 STATUS

8.12 Power Control (PCON0) Register

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

- Brown-out Reset ($\overline{BOR}$)
- Power-on Reset ($\overline{POR}$)
- Reset Instruction Reset ($\overline{RI}$)
- $\overline{MCLR}$ Reset ($\overline{RMCLR}$)
- Watchdog Timer Reset ($\overline{RWDT}$)

- Watchdog Window Violation ($\overline{\text{WDTWV}}$)
- Stack Underflow Reset (STKUNF)
- Stack Overflow Reset (STKOVF)

The Power Control register bits are shown in 8.14.2 PCON0.

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.

8.13 Register Summary - BOR Control and Power Control

Address	Name	Bit Pos.								
0x0EDB	BORCON	7:0	SBOREN							BORRDY
0x0EDC ... 0x0FD5	Reserved									
0x0FD6	PCON1	7:0						RVREG		RCM
0x0FD7	PCON0	7:0	STKOVF	STKUNF	WDTWV	RWDT	RMCLR	RI	POR	BOR

8.14 Register Definitions: Power Control

8.14.1 BORCON

Name: BORCON
Address: 0xEDB

Brown-out Reset Control Register

Bit	7	6	5	4	3	2	1	0
	SBOREN							BORRDY
Access	R/W							R
Reset	1							q

Bit 7 – SBOREN Software Brown-out Reset Enable bit
Reset States: POR/BOR = 1
All Other Resets = u

Value	Condition	Description
–	If BOREN≠ 01	SBOREN is read/write, but has no effect on the BOR.
1	If BOREN= 01	BOR Enabled
0	If BOREN= 01	BOR Disabled

Bit 0 – BORRDY Brown-out Reset Circuit Ready Status bit
Reset States: POR/BOR = q
All Other Resets = u

Value	Description
1	The Brown-out Reset Circuit is active and armed
0	The Brown-out Reset Circuit is disabled or is warming up

Related Links
3.7.2 CONFIG2

8.14.2 PCON0

Name: PCON0
Address: 0xFD7

Power Control Register 0

Bit	7	6	5	4	3	2	1	0
	STKOVF	STKUNF	$\overline{\text{WDTWV}}$	$\overline{\text{RWDT}}$	$\overline{\text{RMCLR}}$	$\overline{\text{RI}}$	$\overline{\text{POR}}$	$\overline{\text{BOR}}$
Access	R/W/HS	R/W/HS	R/W/HC	R/W/HC	R/W/HC	R/W/HC	R/W/HC	R/W/HC
Reset	0	0	1	1	1	1	0	q

Bit 7 – STKOVF Stack Overflow Flag bit
Reset States: POR/BOR = 0
All Other Resets = q

Value	Description
1	A Stack Overflow occurred (more CALLs than fit on the stack)
0	A Stack Overflow has not occurred or set to '0' by firmware

Bit 6 – STKUNF Stack Underflow Flag bit
Reset States: POR/BOR = 0
All Other Resets = q

Value	Description
1	A Stack Underflow occurred (more RETURNs than CALLs)
0	A Stack Underflow has not occurred or set to '0' by firmware

Bit 5 – $\overline{\text{WDTWV}}$ Watchdog Window Violation Flag bit
Reset States: POR/BOR = 1
All Other Resets = q

Value	Description
1	A WDT window violation has not occurred or set to '1' by firmware
0	A CLRWDT instruction was issued when the WDT Reset window was closed (set to '0' in hardware when a WDT window violation Reset occurs)

Bit 4 – $\overline{\text{RWDT}}$ WDT Reset Flag bit
Reset States: POR/BOR = 1
All Other Resets = q

Value	Description
1	A WDT overflow/time-out Reset has not occurred or set to '1' by firmware
0	A WDT overflow/time-out Reset has occurred (set to '0' in hardware when a WDT Reset occurs)

Bit 3 – $\overline{\text{RMCLR}}$ $\overline{\text{MCLR}}$ Reset Flag bit
Reset States: POR/BOR = 1
All Other Resets = q

Value	Description
1	A $\overline{\text{MCLR}}$ Reset has not occurred or set to '1' by firmware
0	A $\overline{\text{MCLR}}$ Reset has occurred (set to '0' in hardware when a $\overline{\text{MCLR}}$ Reset occurs)

Bit 2 – $\overline{\text{RI}}$ RESET Instruction Flag bit
Reset States: POR/BOR = 1
All Other Resets = q

Value	Description
1	A RESET instruction has not been executed or set to '1' by firmware
0	A RESET instruction has been executed (set to '0' in hardware upon executing a RESET instruction)

Bit 1 – $\overline{\text{POR}}$ Power-on Reset Status bit
Reset States: POR/BOR = 0
All Other Resets = u

Value	Description
1	No Power-on Reset occurred or set to '1' by firmware
0	A Power-on Reset occurred (set to '0' in hardware when a Power-on Reset occurs)

Bit 0 – $\overline{\text{BOR}}$ Brown-out Reset Status bit
Reset States: POR/BOR = q
All Other Resets = u

Value	Description
1	No Brown-out Reset occurred or set to '1' by firmware
0	A Brown-out Reset occurred (set to '0' in hardware when a Brown-out Reset occurs)

8.14.3 PCON1

Name: PCON1
Address: 0xFD6

Power Control Register 1

Bit	7	6	5	4	3	2	1	0
						$\overline{\text{RVREG}}$		$\overline{\text{RCM}}$
Access						R/W/HC		R/W/HC
Reset						1		1

Bit 2 – $\overline{\text{RVREG}}$ Main LDO Voltage Regulator Reset Flag bit
Reset States: POR/BOR = 1
All Other Resets = q

Value	Description
1	No LDO or ULP “ready” Reset has occured; or set to ‘1’ by firmware
0	LDO or ULP “ready” Reset has occurred (VDDCORE reached its minimum spec)

Bit 0 – $\overline{\text{RCM}}$ Configuration Memory Reset Flag bit
Reset States: POR/BOR = 1
All Other Resets = q

Value	Description
1	A Reset occurred due to corruption of the configuration and/or calibration data latches
0	The configuration and calibration latches have not been corrupted

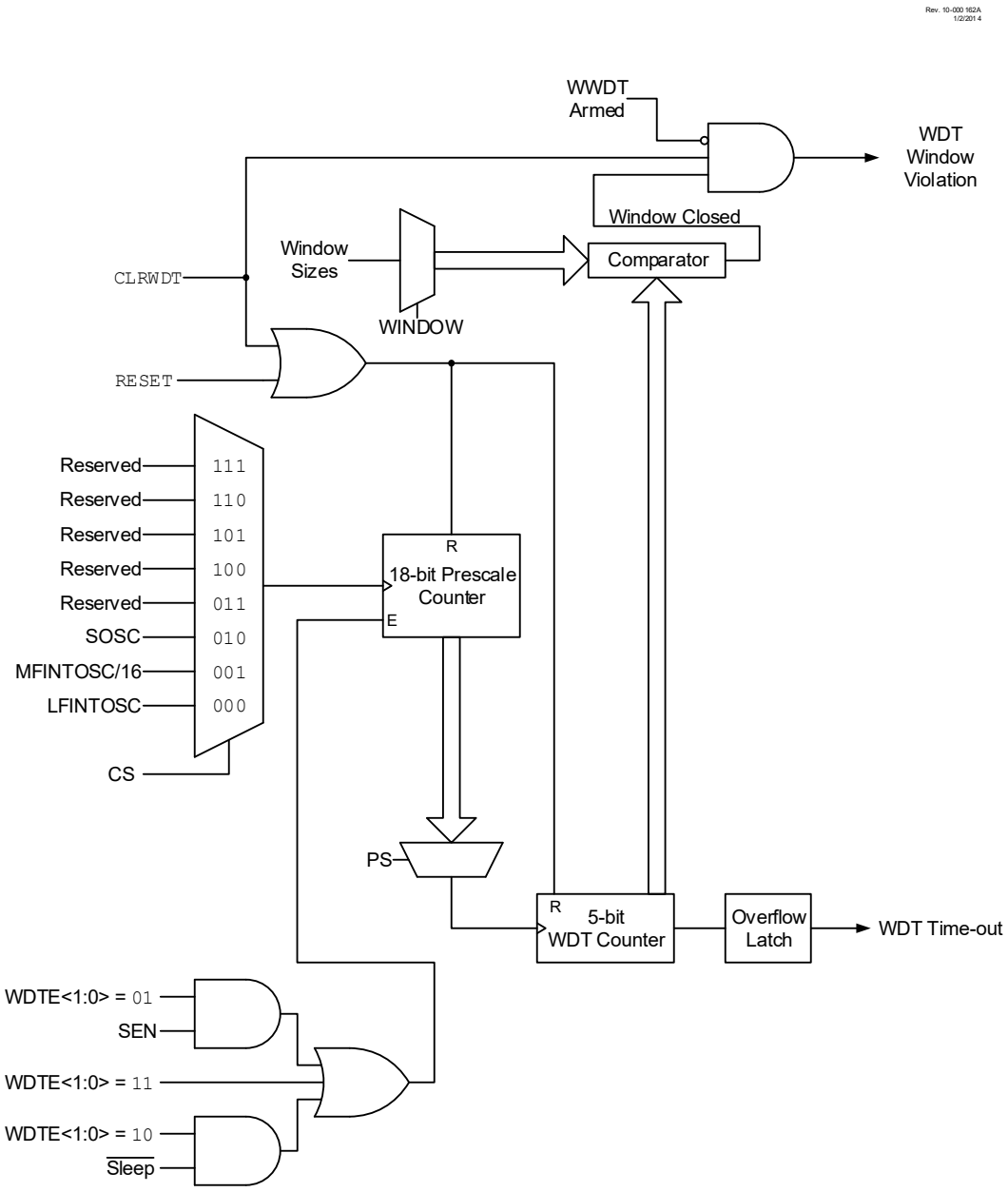
9. (WWDT) Windowed Watchdog Timer

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 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 Configuration bits.

If WDTE = 'b1x, then the clock source will be enabled depending on the WDTCCS Configuration bits.

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

Time intervals in this chapter are based on a minimum nominal interval of 1 ms. See *"Electrical Specifications"* for LFINTOSC and MFINTOSC tolerances.

Related Links

3.7.3 CONFIG3

38.4.2 Internal Oscillator Parameters(1)

9.2 WWDT Operating Modes

The Windowed Watchdog Timer module has four operating modes controlled by the WDTE bits in Configuration Words. See Table 9-1.

9.2.1 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.

9.2.2 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.

9.2.3 WWDT Controlled by Software

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

WWDT protection is unchanged by Sleep. See the following table 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

9.3 Time-out Period

If the WDTCPS Configuration bits default to 0'b11111, then the WDTPS bits set the time-out period from 1 ms to 256 seconds (nominal). If any value other than the default value is assigned to WDTCPS Configuration bits, then the timer period will be based on the WDTCPS bits in the CONFIG3 register. After a Reset, the default time-out period is 2s.

Related Links

3.7.3 CONFIG3

9.4 Watchdog Window

The Windowed Watchdog Timer has an optional Windowed mode that is controlled by the WDTCWS Configuration bits and WINDOW bits. 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 Configuration bits, or the WINDOW bits, if WDTCWS = 111.

The five Most Significant bits of the WDTTMR register are used to determine whether the window is open, as defined by the WINDOW bits.

In the event of a window violation, a Reset will be generated and the $\overline{\text{WDTWV}}$ bit of the PCON0 register will be cleared. This bit is set by a POR or can be set in firmware.

Related Links
8.14.2 PCON0

9.5 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 9.8.2 WDTCON1 registers

9.5.1 CLRWDT Considerations (Windowed Mode)

When in Windowed mode, the WWDT must be armed before a CLRWDT instruction will clear the timer. This is performed by reading the WDTCON0 register. Executing a CLRWDT instruction without performing such an arming action will trigger a window violation regardless of whether the window is open or not.

See Table 9-2 for more information.

9.6 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.

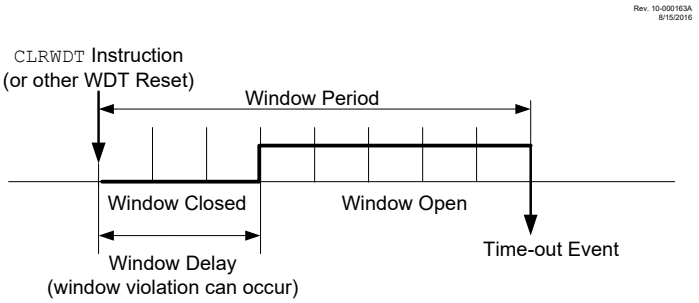
When a WWDT time-out occurs while the device is in Sleep, no Reset is generated. Instead, the device wakes up and resumes operation. The $\overline{\text{TO}}$ and $\overline{\text{PD}}$ bits in the STATUS register are changed to indicate the event. The $\overline{\text{RWDT}}$ bit in the PCON0 register can also be used.

Table 9-2. WWDT Clearing Conditions

Conditions	WWDT
WDTE = 00	Cleared
WDTE = 01 and SEN = 0	

Conditions	WWDT
WDTE = 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



Related Links

4.2.1.3 Oscillator Start-up Timer (OST)

10.8.5 STATUS

8.14.2 PCON0

10. Memory Organization

9.7 Register Summary - WDT Control

Address	Name	Bit Pos.								
0x0ECD	WDTCON0	7:0			WDTPS[4:0]					SEN
0x0ECE	WDTCON1	7:0		WDTCS[2:0]				WINDOW[2:0]		
0x0ECF	WDTPSL	7:0	PSCNTL[7:0]							
0x0ED0	WDTPSH	7:0	PSCNTH[7:0]							
0x0ED1	WDTTMR	7:0	WDTTMR[4:0]					STATE	PSCNT[1:0]	

9.8 Register Definitions: Windowed Watchdog Timer Control

9.8.1 WDTCON0

Name: WDTCON0
Address: 0xECD

Watchdog Timer Control Register 0

Bit	7	6	5	4	3	2	1	0
			WDTPS[4:0]					SEN
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			q	q	q	q	q	0

Bits 5:1 – WDTPS[4:0] Watchdog Timer Prescale Select bits(1)
Bit Value = Prescale Rate

Value	Description
11111 to 10011	Reserved. Results in minimum interval (1 ms)
10010	1:8388608 (2^{23}) (Interval 256s nominal)
10001	1:4194304 (2^{22}) (Interval 128s nominal)
10000	1:2097152 (2^{21}) (Interval 64s nominal)
01111	1:1048576 (2^{20}) (Interval 32s nominal)
01110	1:524288 (2^{19}) (Interval 16s nominal)
01101	1:262144 (2^{18}) (Interval 8s nominal)
01100	1:131072 (2^{17}) (Interval 4s nominal)
01011	1:65536 (Interval 2s nominal) (Reset value)
01010	1:32768 (Interval 1s nominal)
01001	1:16384 (Interval 512 ms nominal)
01000	1:8192 (Interval 256 ms nominal)
00111	1:4096 (Interval 128 ms nominal)
00110	1:2048 (Interval 64 ms nominal)
00101	1:1024 (Interval 32 ms nominal)
00100	1:512 (Interval 16 ms nominal)
00011	1:256 (Interval 8 ms nominal)
00010	1:128 (Interval 4 ms nominal)
00001	1:64 (Interval 2 ms nominal)
00000	1:32 (Interval 1 ms nominal)

Bit 0 – SEN Software Enable/Disable for Watchdog Timer bit

Value	Condition	Description
–	If WDTE = 1x	This bit is ignored
1	If WDTE = 01	WDT is turned on
0	If WDTE = 01	WDT is turned off
–	If WDTE = 00	This bit is ignored

Note:

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

2. When WDTCPS in CONFIG3 = 11111, the Reset value (q) of WDTPS is '01011'. Otherwise, the Reset value of WDTPS is equal to WDTCPS in CONFIG3.
3. When WDTCPS in CONFIG3L ≠ 11111, these bits are read-only.

9.8.2 WDTCON1

Name: WDTCON1
Address: 0xECE

Watchdog Timer Control Register 1

Bit	7	6	5	4	3	2	1	0
		WDTCS[2:0]				WINDOW[2:0]		
Access		R/W	R/W	R/W		R/W	R/W	R/W
Reset		q	q	q		q	q	q

Bits 6:4 – WDTCS[2:0] Watchdog Timer Clock Select bits

Value	Description
111 to 010	Reserved
001	MFINTOSC 31.25 kHz
000	LFINTOSC 31 kHz

Bits 2:0 – WINDOW[2:0] Watchdog Timer Window Select bits

WINDOW	Window delay Percent of time	Window opening Percent of time
111	N/A	100
110	12.5	87.5
101	25	75
100	37.5	62.5
011	50	50
010	62.5	37.5
001	75	25
000	87.5	12.5

Note:

1. If WDTCCS in CONFIG3 = 111, the Reset value of WDTCS is '000'.
2. The Reset value (q) of WINDOW is determined by the value of WDTCWS in the CONFIG3 register.
3. If WDTCCS in CONFIG3 ≠ 111, these bits are read-only.
4. If WDTCWS in CONFIG3 ≠ 111, these bits are read-only.

9.8.3 WDTPSH

Name: WDTPSH
Address: 0xED0

WWDT Prescale Select High Register (Read-Only)

Bit	7	6	5	4	3	2	1	0
	PSCNTH[7:0]							
Access	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – PSCNTH[7:0] 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.

9.8.4 WDTPSL

Name: WDTPSL
Address: 0xECF

WWDT Prescale Select Low Register (Read-Only)

Bit	7	6	5	4	3	2	1	0
	PSCNTL[7:0]							
Access	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – PSCNTL[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.

9.8.5 WDTTMR

Name: WDTTMR
Address: 0xED1

WDT Timer Register (Read-Only)

Bit	7	6	5	4	3	2	1	0
	WDTTMR[4:0]					STATE	PSCNT[1:0]	
Access	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0

Bits 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

Value	Description
1	WDT is armed
0	WDT is not armed

Bits 1:0 – PSCNT[1:0] 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.

10. Memory Organization

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

- Program Memory
- Data RAM
- Data EEPROM

In Harvard architecture devices, the data and program memories use separate buses which 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 the Nonvolatile Memory (NVM) Control Section.

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).

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

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

Figure 10-1. Program and Data Memory Map

Rev. 40-000101B
6/2/2017

Address	Device	
	PIC18F24Q10	PIC18F25Q10
Note 1	Stack (31 Levels)	
00 0000h	Reset Vector	
…	…	
00 0008h	Interrupt Vector High	
…	…	
00 0018h	Interrupt Vector Low	
…	…	
00 001Ah 00 3FFFh	Program Flash Memory (8 KW)	Program Flash Memory (16 KW)
00 4000h 00 7FFFh	Not Present(2)	
00 8000h 00 FFFFh		Not Present(2)
01 0000h 01 FFFFh		
02 0000h 1F FFFFh		
20 0000h 20 00FFh	User IDs (256 Words)(3)	
20 0100h 2F FFFFh	Reserved	
30 0000h 30 000Bh	Configuration Words (6 Words)(3)	
30 000Ch 30 02FFh	Reserved	
30 0300h 30 FFFBh	Unimplemented	
31 0000h 31 00FFh	Data EEPROM (256 Bytes)	
31 0100h 3F FFFBh	Unimplemented	
3F FFFCh 3F FFFDh	Revision ID (1 Word)(4)	
3F FFFEh 3F FFFFh	Device ID (1 Word)(4)	

Note 1: The stack is a separate SRAM panel, apart from all user memory panels.

2: The addresses do not roll over. The region is read as ‘0’.

3: Not code-protected.

4: Device/Revision IDs are hard-coded in silicon.

Figure 10-2. Memory Map and Code Protection Control

Rev. 40-000100B
4/20/2017

Region	Address	Device	
		PIC18F24Q10	PIC18F25Q10
PFM	00 0000h to 00 07FFh	Boot Block 1 KW CP, WRTB, EBTRB	Boot Block 1 KW CP, WRTB, EBTRB
	00 0800h to 00 1FFFh	Block 0 3 KW CP, WRT0, EBTR0	Block 0 3 KW CP, WRT0, EBTR0
	00 2000h to 00 3FFFh	Block 1 4 KW CP, WRT1, EBTR1	Block 1 4 KW CP, WRT1, EBTR1
	00 4000h to 00 5FFFh	Not Present	Block 2 4 KW CP, WRT2, EBTR2
	00 6000h to 00 7FFFh		Block 3 4 KW CP, WRT3, EBTR3
	00 8000h to 00 BFFFh		Not Present
	00 C000h to 00 FFFFh		
	01 0000h to 01 3FFFh		
	01 4000h to 01 7FFFh		
	01 8000h to 01 BFFFh		
	01 C000h to 01 FFFFh		
CONFIG	30 0000h to 30 000Bh	6 Words WRTC	
Data EEPROM	31 0000h to 31 00FFh	256 Words CPD, WRTD	
	31 0100h to 31 01FFh	Unimplemented	

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 10.1.3.1 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.

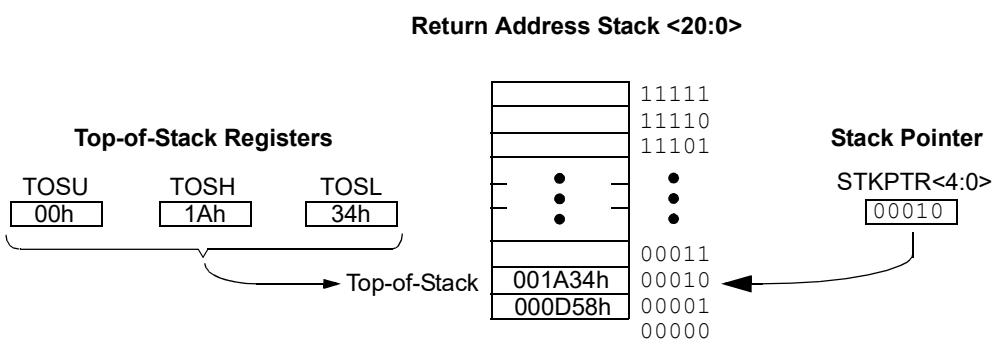
The Stack Pointer is initialized to '0b00000' after all Resets. There is no RAM associated with the location corresponding to a Stack Pointer value of '0b00000'; this is only a Reset value. Status bits in the PCON0 register indicate if the stack is full or has overflowed or has under-flowed.

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 (see Figure 10-3). This allows users to implement a software stack if necessary. After a CALL, RCALL or interrupt, the software can read the pushed value by reading the TOSU:TOSH:TOSL registers. These values can be placed on a user defined software stack. At return time, the software can return these values to TOSU:TOSH:TOSL and do a return.

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

Figure 10-3. Return Address Stack and Associated Registers



10.1.2.2 Return Stack Pointer

The 10.8.4 STKPTR register contains the Stack Pointer value. The STKOVF (Stack Overflow) Status bit and the STKUNF (Stack Underflow) Status bit can be accessed using the PCON0 register. The value of the Stack Pointer can be 0 through 31. On Reset, the Stack Pointer value will be zero. The user may read and write the Stack Pointer value. This feature can be used by a Real-Time Operating System (RTOS) for stack maintenance. After the PC is pushed onto the stack 32 times (without popping any values off the stack), the STKOVF bit is set. The STKOVF bit is cleared by software or by a POR. The action that takes place when the stack becomes full depends on the state of the STVREN (Stack Overflow Reset Enable) Configuration bit.

If STVREN is set (default), a Reset will be generated and a Stack Overflow will be indicated by the STKOVF bit when the 32nd push is initiated. This includes `CALL` and `CALLW` instructions, as well as stacking the return address during an interrupt response. The STKOVF bit will remain set and the Stack Pointer will be set to zero.

If STVREN is cleared, the STKOVF bit will be set on the 32nd push and the Stack Pointer will remain at 31 but no Reset will occur. Any additional pushes will overwrite the 31st push but the STKPTR will remain at 31.

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

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

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

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

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

Related Links

3.7.2 CONFIG2

10.1.2.3 Stack Overflow and Underflow Resets

Device Resets on Stack Overflow and Stack Underflow conditions are enabled by setting the STVREN Configuration bit in Configuration. When STVREN is set, a Full or Underflow condition will set the respective STKOVF or STKUNF bit and then cause a device Reset. When STVREN is cleared, a Full or Underflow condition will set the respective 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.

10.1.2.4 PUSH and POP Instructions

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

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

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

10.1.2.5 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 fast register stack . 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.



Important: The $\overline{TO}$ and $\overline{PD}$ 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.

The following example 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
```

10.1.3 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

10.1.3.1 Computed GOTO

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

A look-up table can be formed with an ADDWF PCL instruction and a group of RETLW nn instructions. The W register is loaded with an offset into the table before executing a call to that table. The first instruction of the called routine is the ADDWF PCL instruction. The next instruction executed will be one of the RETLW *nn* instructions that returns the value 'nn' to the calling function.

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

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

Example 10-2. Computed GOTO Using an Offset Value

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

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

10.1.3.2 Table Reads and Table Writes

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

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

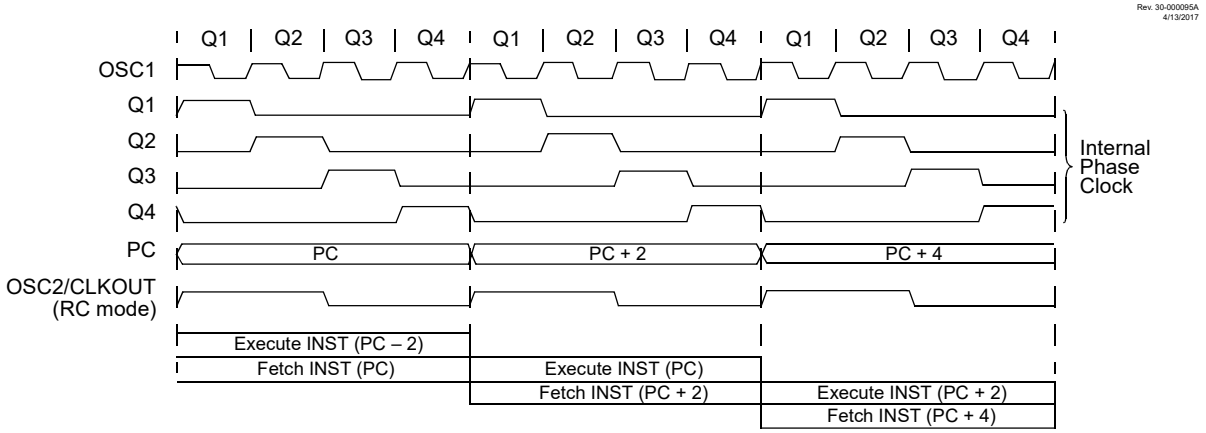
Table read and table write operations are discussed further in the Table Reads and Table Writes section.

10.2 PIC18 Instruction Cycle

10.2.1 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 the following figure.

Figure 10-4. CLOCK/INSTRUCTION CYCLE



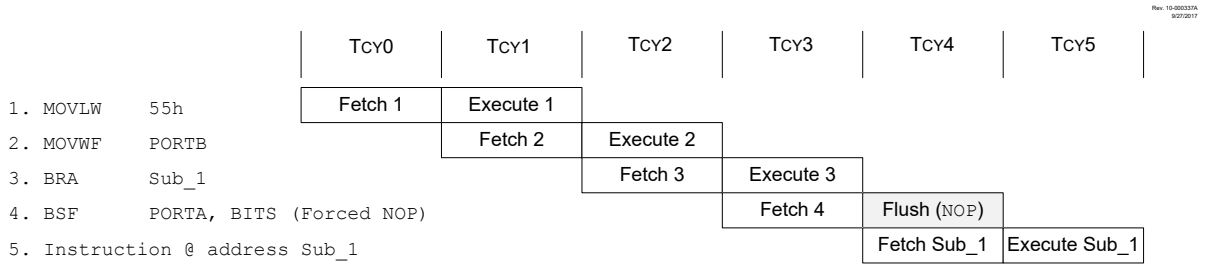
10.2.2 Instruction Flow/Pipelining

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 as shown in the figure below.

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 10-5. Instruction Pipeline Flow



All instructions are single cycle except for any program branches. These take two cycles since the fetch instruction is “flushed” from the pipeline while the new instruction is being fetched and then executed.

10.2.3 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 10.1.1 Program Counter).

The Instructions in Program Memory figure below shows how instruction words are stored in the program memory.

The CALL and GOTO instructions have the absolute program memory address embedded into the instruction. Since instructions are always stored on word boundaries, the data contained in the instruction is a word address. The word address is written to PC<20:1>, which accesses the desired byte address in program memory. Instruction #2 in the example shows how the instruction GOTO 0006h is encoded in the program memory. Program branch instructions, which encode a relative address offset, operate in the same manner. The offset value stored in a branch instruction represents the number of single-word instructions that the PC will be offset by. The Instruction Set Summary provides further details of the instruction set.

Figure 10-6. Instructions in Program Memory

Rev. 30-000112A4/19/2017

			LSB = 1	LSB = 0	Word Address ↓
	Program Memory Byte Locations →				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

Related Links

36. Instruction Set Summary

10.2.4 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. The Two-Word Instructions figure below shows how this works.

Important: See the 10.6 PIC18 Instruction Execution and the Extended Instruction Set section for information on two-word instructions in the extended instruction set.

Figure 10-7. Two-Word Instructions

Rev. 30-000113A
4/19/2017

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

10.3 Data Memory Organization

Important: The operation of some aspects of data memory are changed when the PIC18 extended instruction set is enabled. See 10.6 PIC18 Instruction Execution and the Extended Instruction Set for more information.

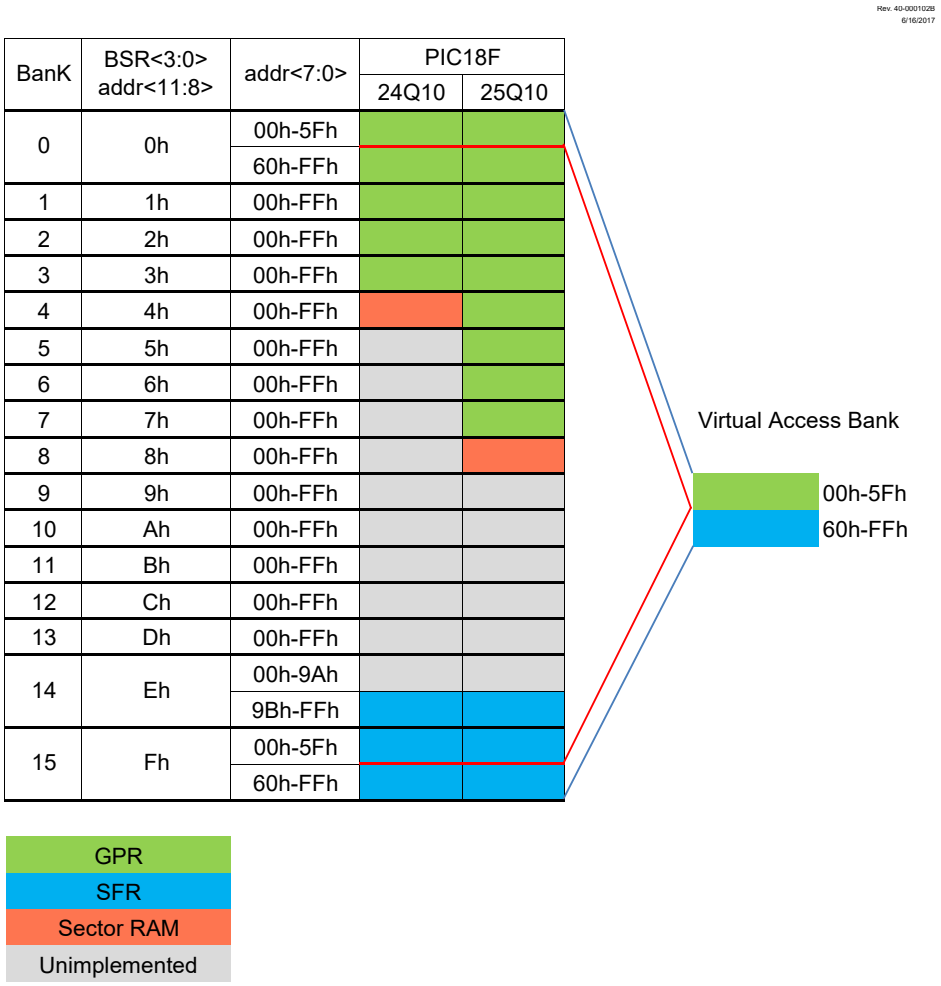
The data memory in PIC18 devices is implemented as static RAM. Each register in the data memory has a 12-bit address, allowing up to 4096 bytes of data memory. The memory space is divided into as many as 16 banks that contain 256 bytes each. The figure below shows the data memory organization for all devices in the device family.

The data memory contains Special Function Registers (SFRs) and General Purpose Registers (GPRs). The SFRs are used for control and status of the controller and peripheral functions, while GPRs are used for data storage and scratchpad operations in the user's application. Any read of an unimplemented location will read as '0'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). The 10.3.2 Access Bank section provides a detailed description of the Access RAM.

Figure 10-8. Data Memory Map



BanK	BSR<3:0> addr<11:8>	addr<7:0>	PIC18F	
			24Q10	25Q10
0	0h	00h-5Fh		
		60h-FFh		
1	1h	00h-FFh		
2	2h	00h-FFh		
3	3h	00h-FFh		
4	4h	00h-FFh		
5	5h	00h-FFh		
6	6h	00h-FFh		
7	7h	00h-FFh		
8	8h	00h-FFh		
9	9h	00h-FFh		
10	Ah	00h-FFh		
11	Bh	00h-FFh		
12	Ch	00h-FFh		
13	Dh	00h-FFh		
14	Eh	00h-9Ah		
		9Bh-FFh		
15	Fh	00h-5Fh		
		60h-FFh		

10.3.1 Bank Select Register

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 (10.8.13 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.

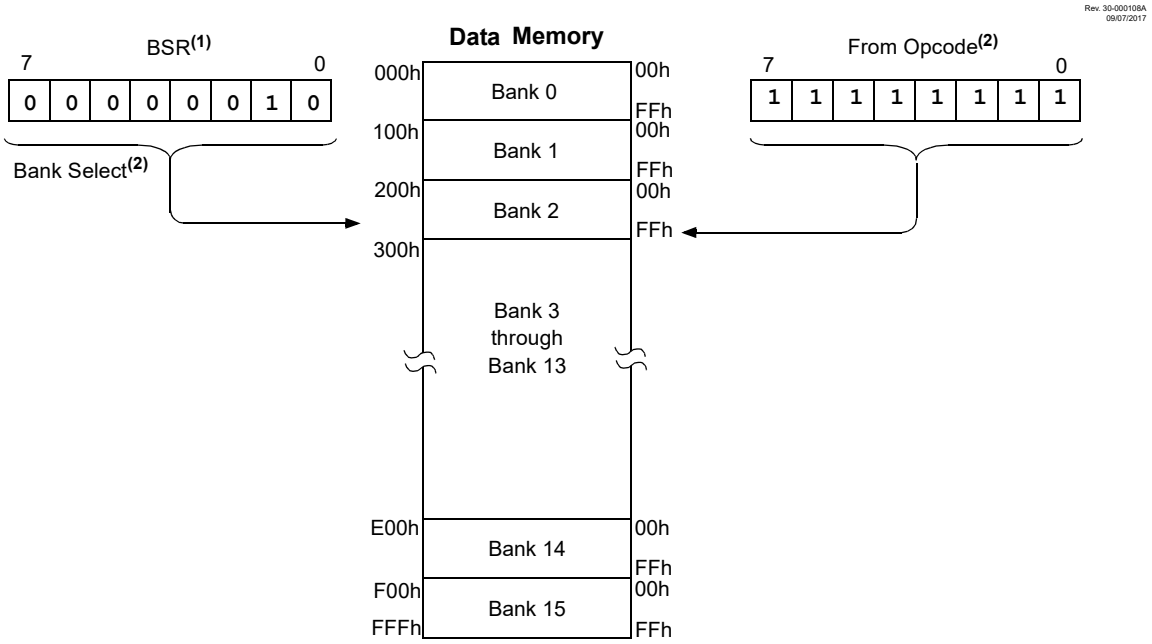
The value of the BSR indicates the bank in data memory; the eight bits in the instruction show the location in the bank and can be thought of as an offset from the bank's lower boundary. The relationship between the BSR's value and the bank division in data memory is shown in the figure below.

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 the following figure 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 10-9. Use of the Bank Select Register (Direct Addressing)



Note 1: The Access RAM bit of the instruction can be used to force an override of the selected bank (BSR<3:0>) to the registers of the Access Bank.

2: The MOVFF instruction embeds the entire 12-bit address in the instruction.

10.3.2 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 Block 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 (see Data Memory Map).

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 the 10.5.3 Mapping the Access Bank in Indexed Literal Offset Mode section.

10.3.3 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.

10.3.4 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. A list of these registers is given in the register summary table.

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.

Related Links

34. Register Summary

10.3.5 Status Register

The 10.8.5 STATUS register contains the arithmetic status of the ALU. As with any other SFR, it can be the operand for any instruction.

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

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

For other instructions that do not affect Status bits, see the instruction set summaries.

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

Related Links

36. Instruction Set Summary

10.4 Data Addressing Modes

Important: The execution of some instructions in the core PIC18 instruction set are changed when the PIC18 extended instruction set is enabled. See 10.5 Data Memory and the Extended Instruction Set for more information.

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

The addressing modes are:

- Inherent
- Literal
- Direct
- Indirect

An additional addressing mode, Indexed Literal Offset, is available when the extended instruction set is enabled (XINST Configuration bit = 1). Its operation is discussed in greater detail in 10.5.1 Indexed Addressing with Literal Offset.

10.4.1 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.

10.4.2 Direct Addressing

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

In the core PIC18 instruction set, bit-oriented and byte-oriented instructions use some version of direct addressing by default. All of these instructions include some 8-bit literal address as their Least Significant Byte. This address specifies either a register address in one of the banks of data RAM (see 10.3.3 General Purpose Register File) or a location in the Access Bank (see 10.3.2 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 (see 10.3.1 Bank Select Register) 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.

10.4.3 Indirect Addressing

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

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

Example 10-3. How to Clear RAM (Bank 1) Using Indirect Addressing

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

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

CONTINUE:                ; YES, continue
```

10.4.3.1 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, the FSR value can target any location in any bank regardless of the BSR value. However, the Access RAM bit must be cleared to 0 to ensure that the INDF register in Access space is the object of the operation instead of a register in one of the other banks. The assembler default value for the Access RAM bit is zero when targeting any of the indirect operands.

10.4.3.2 FSR Registers and POSTINC, POSTDEC, PREINC and PLUSW

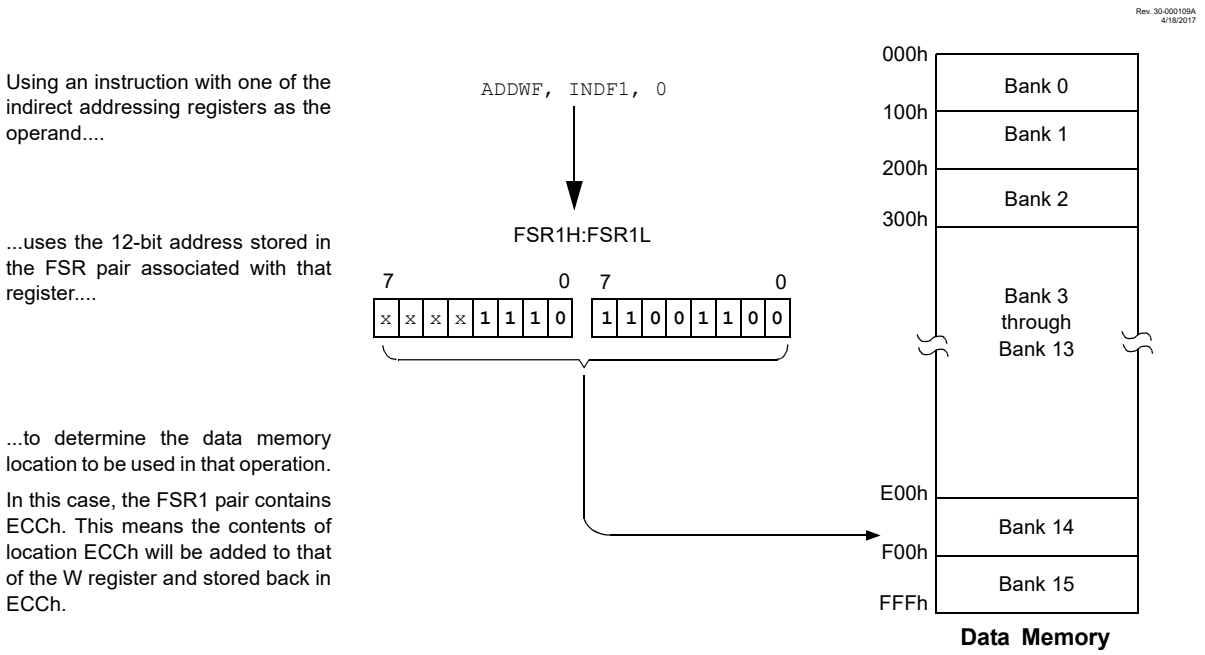
In addition to the INDF operand, each FSR register pair also has four additional indirect operands. Like INDF, these are “virtual” registers which cannot be directly read or written. Accessing these registers

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 10-10. Indirect Addressing



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

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

10.4.3.3 Operations by FSRs on FSRs

Indirect addressing operations that target other FSRs or virtual registers represent special cases. For example, using an FSR to point to one of the virtual registers will not result in successful operations. As a

specific case, assume that FSR0H:FSR0L contains the address of INDF1. Attempts to read the value of the INDF1 using INDF0 as an operand will return 00h. Attempts to write to INDF1 using INDF0 as the operand will result in a NOP.

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

10.5 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.

10.5.1 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.

10.5.2 Instructions Affected by Indexed Literal Offset Mode

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

Additionally, byte-oriented and bit-oriented instructions are not affected if they do not use the Access Bank (Access RAM bit is ‘1’), or include a file address of 60h or above. Instructions meeting these criteria

will continue to execute as before. A comparison of the different possible addressing modes when the extended instruction set is enabled is shown in the following figure.

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

Figure 10-11. Comparing Addressing Options for Bit-Oriented and Byte-Oriented Instructions (Extended Instruction Set Enabled)

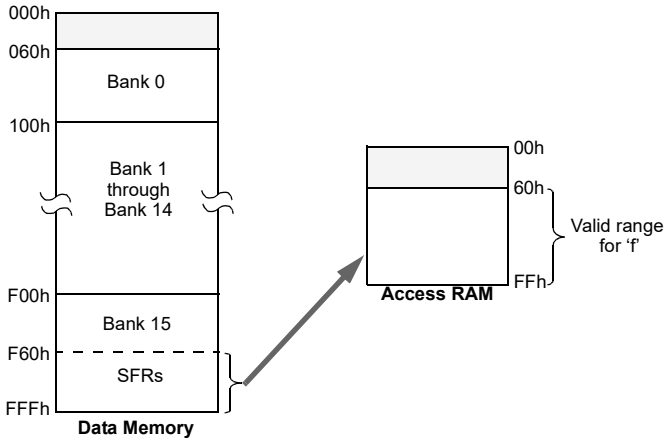
Rev. 30-000110A
4/18/2017

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

When 'a' = 0 and f ≥ 60h:

The instruction executes in Direct Forced mode. 'f' is interpreted as a location in the Access RAM between 060h and 0FFh. This is the same as locations F60h to FFFh (Bank 15) of data memory.

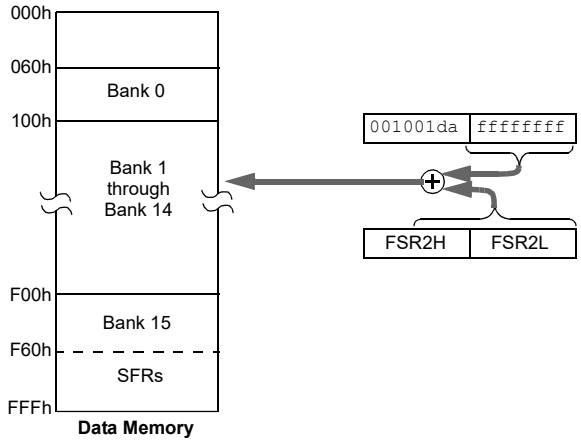
Locations below 60h are not available in this addressing mode.



When 'a' = 0 and f ≤ 5Fh:

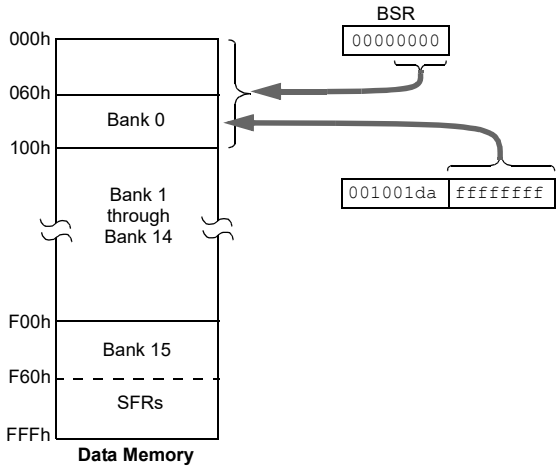
The instruction executes in Indexed Literal Offset mode. 'f' is interpreted as an offset to the address value in FSR2. The two are added together to obtain the address of the target register for the instruction. The address can be anywhere in the data memory space.

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.



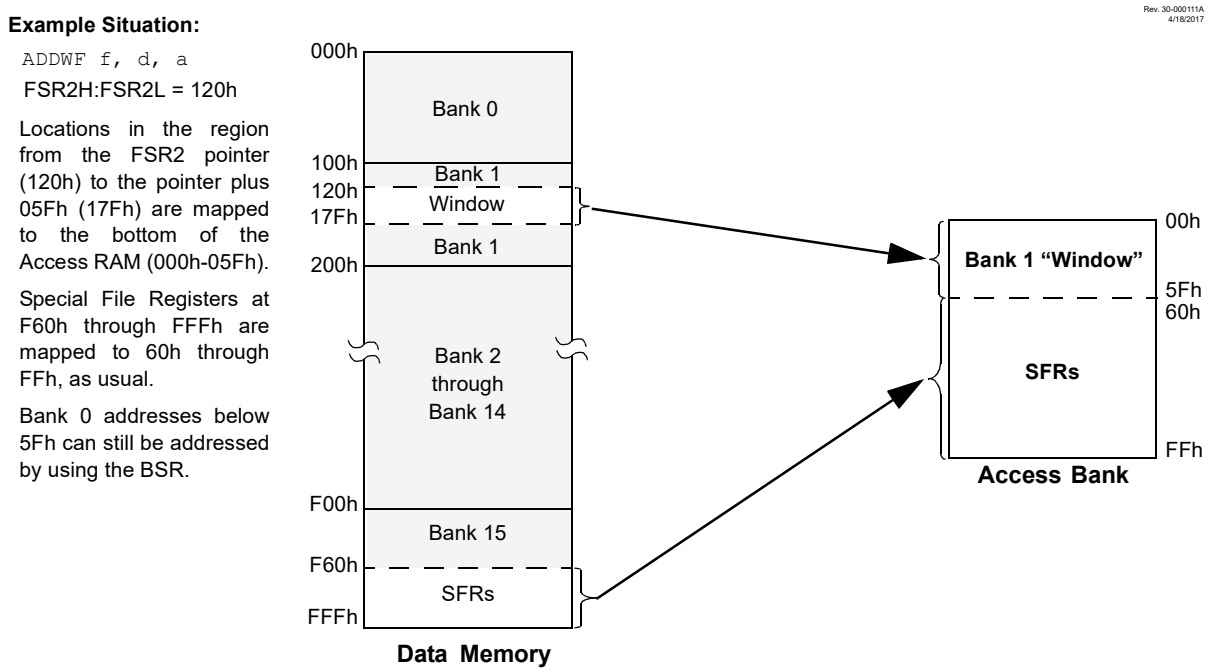
Related Links

36.2.1 Extended Instruction Syntax

10.5.3 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 10.3.2 Access Bank). An example of Access Bank remapping in this addressing mode is shown in the following figure.

Figure 10-12. Remapping the Access Bank with Indexed Literal Offset Addressing



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.

10.6 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 the Extended Instruction Set section.

Related Links

36.2.1 Extended Instruction Syntax

10.7 Register Summary: Memory and Status

Address	Name	Bit Pos.								
0x0FD8	STATUS	7:0		$\overline{TO}$	$\overline{PD}$	N	OV	Z	DC	C
0x0FD9	FSR2	7:0	FSRL[7:0]							
		15:8					FSRH[3:0]			
0x0FDB	PLUSW2	7:0	PLUSW[7:0]							
0x0FDC	PREINC2	7:0	PREINC[7:0]							
0x0FDD	POSTDEC2	7:0	POSTDEC[7:0]							
0x0FDE	POSTINC2	7:0	POSTINC[7:0]							
0x0FDF	INDF2	7:0	INDF[7:0]							
0x0FE0	BSR	7:0					BSR[3:0]			
0x0FE1	FSR1	7:0	FSRL[7:0]							
		15:8					FSRH[3:0]			
0x0FE3	PLUSW1	7:0	PLUSW[7:0]							
0x0FE4	PREINC1	7:0	PREINC[7:0]							
0x0FE5	POSTDEC1	7:0	POSTDEC[7:0]							
0x0FE6	POSTINC1	7:0	POSTINC[7:0]							
0x0FE7	INDF1	7:0	INDF[7:0]							
0x0FE8	WREG	7:0	WREG[7:0]							
0x0FE9	FSR0	7:0	FSRL[7:0]							
		15:8					FSRH[3:0]			
0x0FEB	PLUSW0	7:0	PLUSW[7:0]							
0x0FEC	PREINC0	7:0	PREINC[7:0]							
0x0FED	POSTDEC0	7:0	POSTDEC[7:0]							
0x0FEE	POSTINC0	7:0	POSTINC[7:0]							
0x0FEF	INDF0	7:0	INDF[7:0]							
0x0FF0 ... 0x0FF8	Reserved									
0x0FF9	PCL	7:0	PCL[7:0]							
0x0FFA	PCLAT	7:0	PCLATH[7:0]							
		15:8				PCLATU[4:0]				
0x0FFC	STKPTR	7:0				STKPTR[4:0]				
0x0FFD	TOS	7:0	TOSL[7:0]							
		15:8	TOSH[7:0]							
		23:16				TOSU[4:0]				

10.8 Register Definitions: Memory and Status

10.8.1 PCL

Name: PCL
Address: 0xFF9

Low byte of the Program Counter

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

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

10.8.2 PCLAT

Name: PCLAT
Address: 0xFFA

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

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

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

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

Bits 7:0 – PCLATH[7:0] High PC Latch register
Holding register for Program Counter bits <16:8>

10.8.3 TOS

Name: TOS
Address: 0xFFD

Top-Of-Stack Registers.
Contents of the stack pointed to by the 10.8.4 STKPTR register. This is the value that will be loaded into the Program Counter upon a `RETURN` or `RETFIE` instruction.

Bit	23	22	21	20	19	18	17	16
				TOSU[4:0]				
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	TOSH[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

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

Bits 20:16 – TOSU[4:0] Upper byte of TOS register
Bits <21:17> of the TOS

Bits 15:8 – TOSH[7:0] High Byte of the TOS Register
Bits <16:8> of the TOS

Bits 7:0 – TOSL[7:0] Low Byte TOS Register
Bits <7:0> of the TOS

10.8.4 STKPTR

Name: STKPTR
Address: 0xFFC

Stack Pointer Register

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

Bits 4:0 – STKPTR[4:0] Stack Pointer Location bits

10.8.5 STATUS

Name: STATUS
Address: 0xFD8

Status Register

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

Bit 6 – $\overline{TO}$ Time-Out bit
Reset States: POR/BOR = 1
All Other Resets = q

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

Bit 5 – $\overline{PD}$ Power-Down bit
Reset States: POR/BOR = 1
All Other Resets = q

Value	Description
1	Set at power-up or by execution of `CLRWDT` instruction
0	Cleared 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`).
Reset States: POR/BOR = 0
All Other Resets = u

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

Bit 3 – OV Overflow 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.
Reset States: POR/BOR = 0
All Other Resets = u

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

Bit 2 – Z Zero bit
Reset States: POR/BOR = 0

All Other Resets = u

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

Bit 1 – DC Digit Carry/Borrow bit
ADDWF, ADDLW, SUBLW, SUBWF instructions(1)
Reset States: POR/BOR = 0
All Other Resets = u

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

Bit 0 – C Carry/Borrow bit
ADDWF, ADDLW, SUBLW, SUBWF instructions(1,2)
Reset States: POR/BOR = 0
All Other Resets = u

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

Note:

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

10.8.6 WREG

Name: WREG
Address: 0xFE8

Shadow of Working Data Register

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

Bits 7:0 – WREG[7:0]

10.8.7 INDF

Name: INDFx
Address: 0xFEF,0xFE7,0xFDF

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

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

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

10.8.8 POSTDEC

Name: POSTDECx
Address: 0xFED,0xFE5,0xFDD

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

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

Bits 7:0 – POSTDEC[7:0]

10.8.9 POSTINC

Name: POSTINCx
Address: 0xFEE,0xFE6,0xFDE

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

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

Bits 7:0 – POSTINC[7:0]

10.8.10 PREINC

Name: PREINCx
Address: 0xFEC,0xFE4,0xFDC

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

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

Bits 7:0 – PREINC[7:0]

10.8.11 PLUSW

Name: PLUSWx
Address: 0xFEB,0xFE3,0xFDB

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

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

Bits 7:0 – PLUSW[7:0]

10.8.12 FSR

Name: FSRx
Address: 0xFE9,0xFE1,0xFD9

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

Bit	15	14	13	12	11	10	9	8
					FSRH[3:0]			
Access					R/W	R/W	R/W	R/W
Reset					0	0	0	0

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

Bits 11:8 – FSRH[3:0]
Most Significant address of INDF data

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

10.8.13 BSR

Name: BSR
Address: 0xFE0

Bank Select Register
The BSR indicates the data memory bank which is bits 11:8 of the GPR address.

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

Bits 3:0 – BSR[3:0]
Four Most Significant bits of the data memory address

11. (NVM) Nonvolatile Memory Control

Nonvolatile memory is separated into three categories: Program Flash Memory (PFM) including User IDs, Configuration words, and Data Flash Memory (DFM). DFM is also referred to as EEPROM because it is written one byte at a time and the erase before write is automatic. Although User IDs are above the PFM section they are included in the PFM category because the read and write access is identical for both.

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

PFM and DFM can be protected in two ways: code protection and write protection. Code protection (Configuration bits $\overline{CP}$ for PFM and $\overline{CPD}$ for DFM) disables read and write access through an external device programmer. Code protection does not affect the self-write and erase functionality whereas write protection does. Code protection write protection can only be reset by a bulk erase performed by an external device programmer. A PFM bulk erase clears the program space, Configuration bits, and User IDs. A bulk erase only clears DFM if the $\overline{CPD}$ = 0. When $\overline{CPD}$ = 1 then bulk erasing DFM requires setting the Program Counter to the DFM area before the Bulk Erase command is issued, and then only the DFM area is cleared. See the programming specification for more details. Write protection prevents writes to NVM areas tagged for protection by the WRTn Configuration bits. Attempts to write a protected location will set the NVMERR bit.

PFM and Configuration Words can be accessed by either the Table Pointer or NVM controls. DFM can be accessed only by the NVM controls. The PFM access is by single byte, single word, or full sector. A sector is 256 bytes (128 PFM words). The sector memory occupies one full bank of RAM space located in the RAM bank following the last occupied GPR bank. Sector memory is also referred to elsewhere in this document as the write block holding registers. The Table Pointer accesses memory by bytes. The NVM control accesses the DFM section by bytes and the other sections by words and sectors.

The NVM controls include five independent access functions with five corresponding control bits. The controls are as follows:

- RD – Single byte/word read
- WR – single byte/word write
- SECRD – Sector read
- SECWR – Sector write
- SECER – Sector erase

The NVMADR registers determine the address of the memory region being accessed by the NVM control. The TBLPTR registers determine the address of the memory being accessed by the Table Pointer functions. The following table indicates which controls operate in each region.

Table 11-1. NVM Organization Table

Region	Address Range	Table Pointer TBLRD	NVMCON1				
			RD	WR	SECRD	SECWR	SECER
PFM	00 0000h 01 FFFFh	•	•	•	•	•	•
User IDs	20 0000h 20 00FFh	•	•	•	•	•	•
CONFIG	30 0000h 30 000Bh	•	•		•	•	
DFM	31 0000h 31 00FFh		•	•			

11.1 Program Flash Memory

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

A read from program memory is executed one byte at a time. A program memory erase is executed on blocks of n bytes at a time also referred to as sectors. Refer to the following table for write and erase block sizes. A Bulk Erase operation cannot be issued from user code. A write to program memory can be executed by sectors or single words.

Table 11-2. Flash Memory Organization by Device

Device	Sector Erase Size (Words)	Holding Registers (Bytes)	TBLPTR LSbs (Holding Address)	Program Flash Memory (Words)	Data Flash Memory (Bytes)
PIC18F24Q10	128	256	8	8192	256
PIC18F25Q10				16384	

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 sectors. A sector is the minimum size that can be erased by user software.

After a sector has been erased, all or a portion of this sector can be programmed. Data can be written directly into PFM one 16-bit word at a time using the NVMADR and NVMCON1 controls or as a full sector from sector RAM which is also referred to as the holding registers. These 8-bit registers are located in the

RAM bank following the last GPR RAM bank. The holding registers are directly accessible as any other SFR/GPR register and also may be loaded via sequential writes using the TABLAT and 11.5.7 TBLPTR registers.

Important: To modify only a portion of a previously programmed sector the contents of the entire sector must be read and saved in RAM prior to the sector erase. The SECRD operation is the easiest way to do this. Then, the new data can be written into the holding registers to reprogram the sector of PFM. However, any unprogrammed locations can be written using the single word write operation without first erasing the sector.

11.1.1 Table Pointer Operations

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 SFR registers associated with these operations include:

- TABLAT register
- TBLPTR registers

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

11.1.1.1 Table Pointer Register

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

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

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+*`	TBLPTR is incremented before the read/write

TBLWT+*	

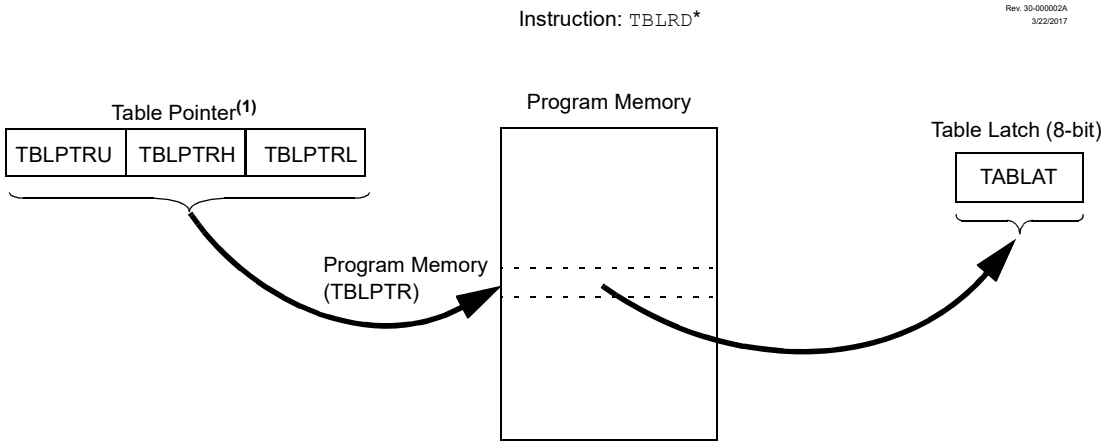
11.1.1.2 Table Latch Register

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

11.1.1.3 Table Read Operations

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

Figure 11-1. Table Read Operation

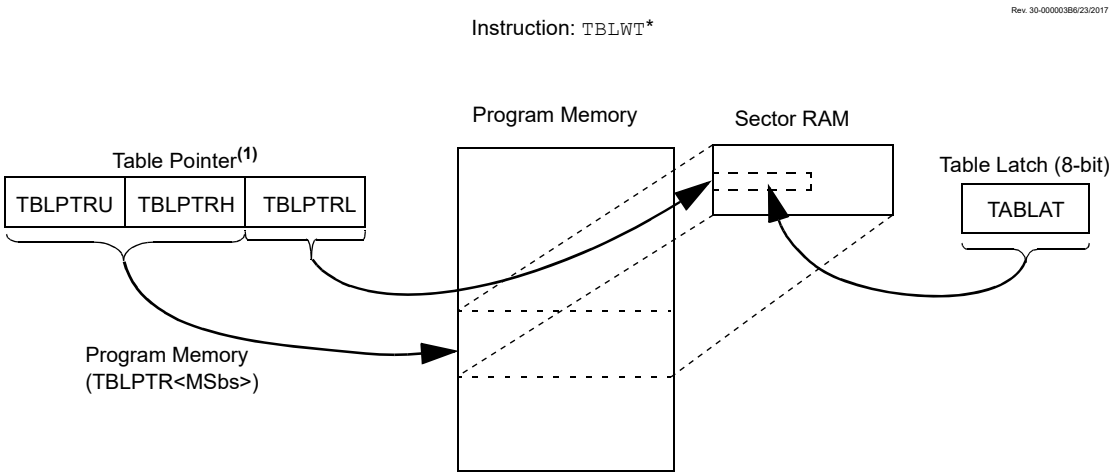


Note 1: Table Pointer register points to a byte in program memory.

11.1.1.4 Table Write Operations

The table write operation stores one byte of data from the TABLAT register into a sector RAM holding register. The following figure shows the operation of a table write from the TABLAT register to the holding register space. The procedure to write the contents of the holding registers into program memory is detailed in the *"Writing to Program Flash Memory"* section.

Figure 11-2. Table Write Operation



Note 1: During table writes the Table Pointer points to two areas. The LSbs of TBLPRTL point to an address within the sector RAM space. The MSbs of the Table Pointer determine where the sector will eventually be written.

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.

Related Links

11.1.4 Writing to Program Flash Memory

11.1.1.5 Table Pointer Boundaries

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

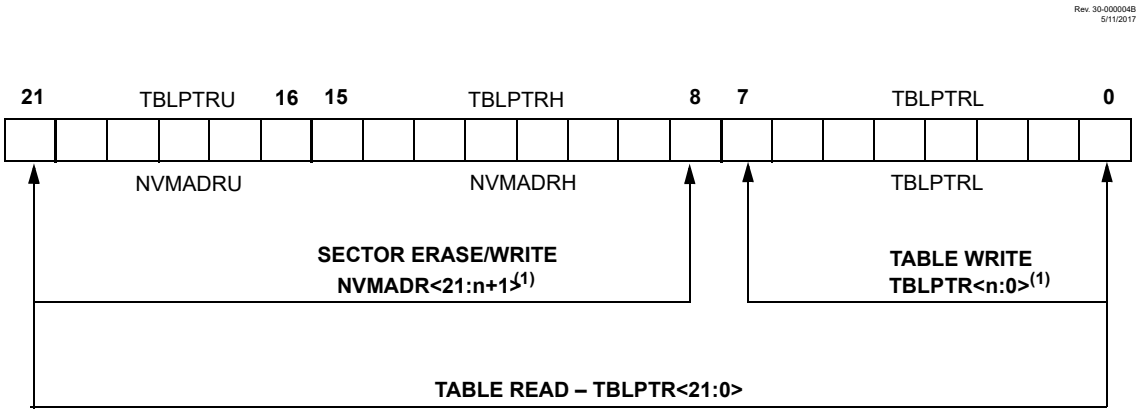
When a TBLRD is executed, all 22 bits of the TBLPTR determine which byte is read from program memory directly into the TABLAT register.

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 may vary depending on the device (see the *"Flash Memory Organization by Device"* table in the *"Program Flash Memory"* section). The LSbs of the TBLPTRL register determine which specific address within the holding register block is written to. The size of the write block determines the number of LSbs. The MSbs of the Table Pointer have no effect during TBLWT operations.

When a PFM sector write is executed the entire holding register block is written to the Flash memory sector at the address determined by the MSbs of the NVMADR. The LSbs are ignored during sector writes. For more detail, see the 11.1.4 Writing to Program Flash Memory section.

The following figure illustrates the relevant boundaries of TBLPTR and NVMADR based on NVM control operations.

Figure 11-3. Table Pointer Boundaries Based on Operation



Note:

1. See the *"Flash Memory Organization by Device"* table in the *"Program Flash Memory"* section for the write holding registers block size.

Related Links

11.1 Program Flash Memory

11.1.1.6 Reading the Program Flash Memory

The `TBLRD` instruction retrieves data from program memory at the TBLPTR location and places it into the TABLAT SFR register. Table reads from program memory are performed one byte at a time. 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

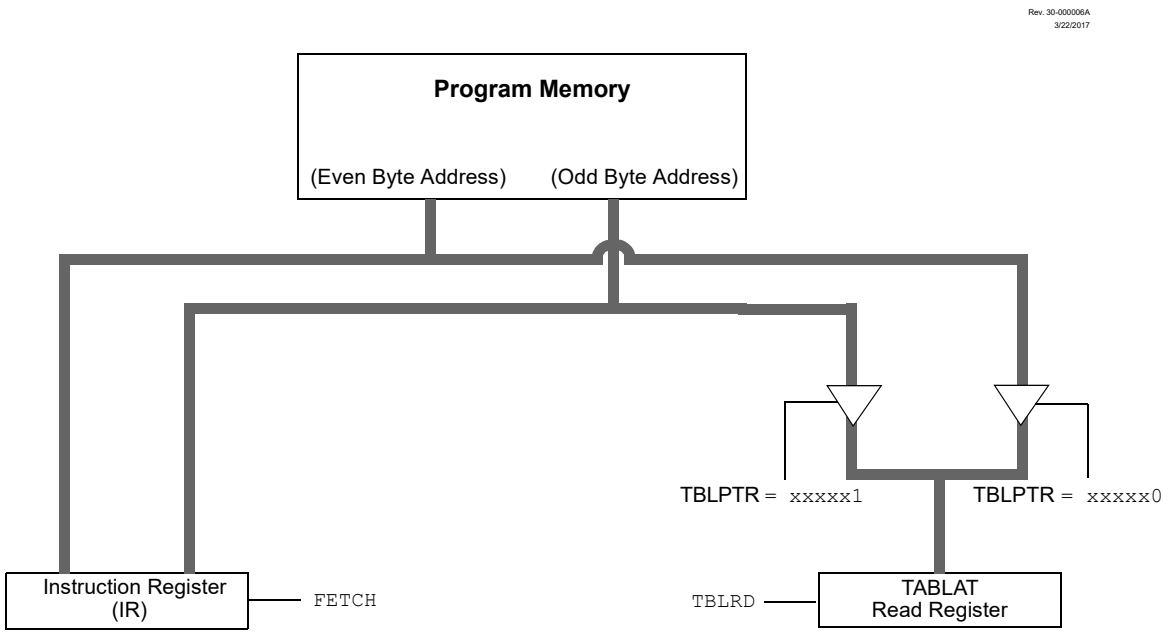
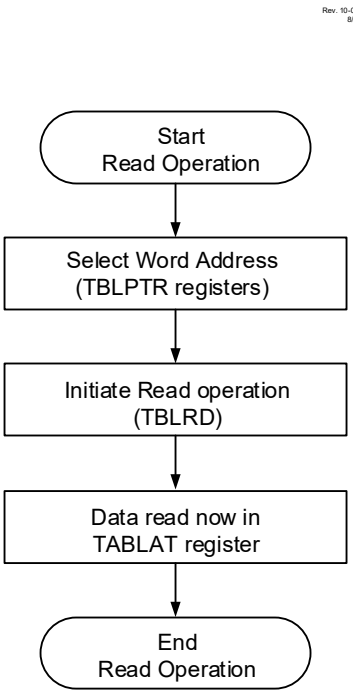


Figure 11-5. Program Flash Memory Read Flowchart



Example 11-1. Reading a Program Flash Memory Word

```
    MOVLW   CODE_ADDR_UPPER    ; Load TBLPTR with the base
    MOVWF   TBLPTRU            ; address of the word
    MOVLW   CODE_ADDR_HIGH
    MOVWF   TBLPTRH
    MOVLW   CODE_ADDR_LOW
    MOVWF   TBLPTRL
READ_WORD:
```

```
    TBLRD*+                     ; read into TABLAT and increment
    MOVF    TABLAT, W           ; get data
    MOVWF   WORD_EVEN
    TBLRD*+                     ; read into TABLAT and increment
    MOVFW   TABLAT, W           ; get data
    MOVF    WORD_ODD
```

11.1.2 NVM Unlock Sequence

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

- PFM sector erase
- PFM sector write from holding registers
- PFM sector read into write block holding registers
- PFM word write directly to NVM
- DFM byte write directly to NVM
- Write to Configuration Words

Each of these operations correspond to one of four unlock code sequences as shown in the following table:

Table 11-4. NVM Unlock Codes

Operation	First Unlock Byte	Second Unlock Byte	NVMCON1 Operation Bit
Word/Byte write	55h	AAh	WR
Sector write	DDh	22h	SECWR
Sector erase	CCh	33h	SECER
Sector read	BBh	44h	SECRD

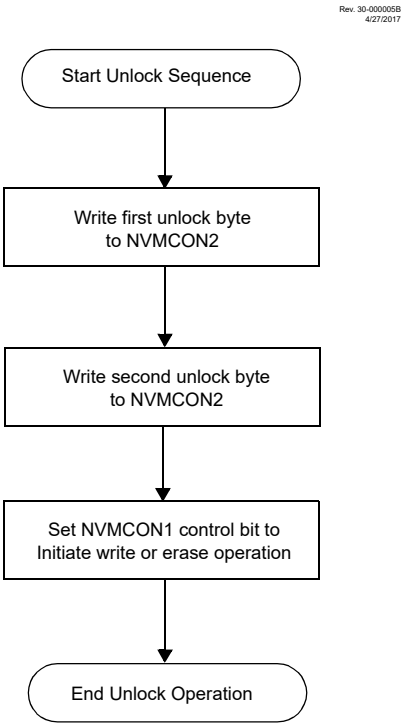
The general unlock sequence consists of the following steps and must be completed in order:

- Write first unlock byte to NVMCON2
- Write second unlock byte to NMVCON2
- Set the operation control bit of NVMCON1

For PFM and Configuration Word operations, once the control bit is set the processor will stall internal operations until the operation is complete and then resume with the next instruction. DFM operations do not stall the CPU.

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 for PFM word write

```
    BCF     INTCON,GIE      ; Recommended so sequence is not interrupted
    MOVF    UpperAddr,W     ; set the target address
    MOVWF   NVMADRU
    MOVF    HighAddr,W
    MOVWF   NVMADRH
    MOVF    LowAddr,W
    MOVWF   NVMADRL
    MOVF    HighByte,W      ; high byte of word to be written
    MOVWF   NVMDATH         ; store in high byte transfer register
    MOVF    LowByte,W       ; low byte of word to be written
    MOVWF   NVMDATL         ; store in low byte transfer register
    BSF     NVMCON0,NVMEN   ; Enable NVM operation
    MOVLW   55h             ; Load first unlock byte
; ------------------ Required Sequence -----------------------------
    MOVWF   NVMCON2         ; Step 1: Load first byte into NVMCON2
    MOVLW   AAh             ; Step 2: Load W with second unlock byte
    MOVWF   NVMCON2         ; Step 3: Load second byte into NVMCON2
    BSF     NVMCON1,WR      ; Step 4: Set WR bit to begin write
; ------------------------------------------------------------------
    BSF     INTCON,GIE      ; Re-enable interrupts
```

Important:

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.

11.1.3 Erasing Program Flash Memory (PFM)

The minimum erase block is always one sector. 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 sector erase size of 128 words, a block of 128 words (256 bytes) of program memory is erased. The NVMADR<21:8> bits point to the block being erased. The NVMADR<7:0> bits are ignored.

The NVMCON0 and NVMCON1 registers command the erase operation. The 11.5.1.1 NVMEN bit must be set to enable write operations. The 11.5.2.1 SECER bit is set to initiate the erase operation.

The NVM unlock sequence described in the 11.1.2 NVM Unlock Sequence section must be used which guards 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.

11.1.3.1 PFM Erase Sequence

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

1. Set NVMADR to an address within the intended sector.
2. Set the 11.5.1.1 NVMEN bit to enable NVM
3. Perform the unlock sequence as described the 11.1.2 NVM Unlock Sequence section
4. Set the 11.5.2.1 SECER bit

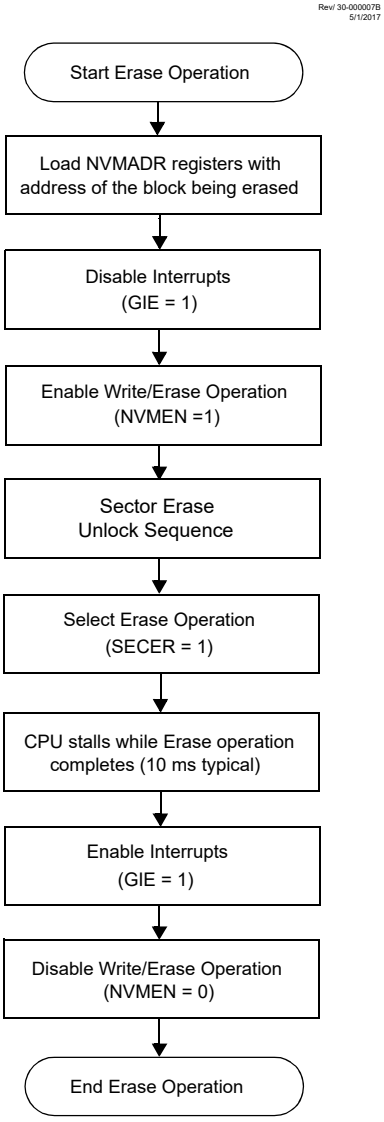
If the PFM address is write-protected, the 11.5.2.1 SECER bit will be cleared and the erase operation will not take place, 11.5.1.2 NVMERR is signaled in this scenario.

The operation erases the sector indicated by masking the LSbs of the current NVMADR.

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

The holding register data is not affected by erase operations and NVMEN will remain unchanged.

Figure 11-7. PFM Sector Erase Flowchart



Example 11-3. Erasing a Program Flash Memory block

```
; This sample sector erase routine assumes that the target address
; specified by CODE_ADDR_UPPER and CODE_ADDR_HIGH contain a value within
; the PFM address range of the device.

           MOVLW   CODE_ADDR_UPPER    ; load NVMADR with the base
           MOVWF   NVMADRU            ; address of the memory block
           MOVLW   CODE_ADDR_HIGH
           MOVWF   NVMADRH
ERASE_BLOCK:
           BSF     NVMCON0, NVMEN     ; enable Program Flash Memory
           BCF     INTCON, GIE        ; disable interrupts
Required   MOVLW   CCh                ; first unlock byte for erase
Sequence   MOVWF   NVMCON2            ; write CCh
           MOVLW   33h                ; second unlock byte for erase
           MOVWF   NVMCON2            ; write 33h
           BSF     NVMCON1, SECER     ; start erase (CPU stalls)
```

```
            BSF     INTCON, GIE         ; re-enable interrupts
            BCF     NVMCON0, NVMEN      ; disable writes to memory
```

Important:

1. If a write or erase operation is terminated by an unexpected event, NVMERR bit will be set which the user can check to decide whether a rewrite of the location(s) is needed.
2. NVMERR is set if SECER is written to '1' while NVMADR points to a write-protected address.
3. NVMERR is set if SECER is written to '1' while NVMADR points to an invalid address location (Refer to the device memory map and NVM Organization Table).

Related Links

11. (NVM) Nonvolatile Memory Control

11.1.4 Writing to Program Flash Memory

Program memory can be written either one word at a time or a sector at a time.

A single word is written by setting the NVMADR to the target address and loading NVMDAT with the desired word. The word is then transferred to Flash memory with the 11.5.2.3 WR unlock and write sequence.

A sector is written by first loading a block of holding registers and then executing a sector write sequence. The programming write block size is specified as the holding registers, also referred to as sector RAM, in the Flash Memory Organization by Device table. Table writes are used to write the holding register bytes that are then transferred to Flash memory with the 11.5.2.2 SECWR unlock and write sequence. There are only as many holding registers as there are bytes in a write block.

Since the table latch (11.5.6 TABLAT) is only a single byte, the TBLWT instruction needs to be executed multiple times for each sector programming operation. The write protection state is ignored for table writes. 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 sector of memory is started by setting NVMADR to an address within the target sector and executing a sector write unlock sequence followed immediately by setting the 11.5.2.2 SECWR bit.

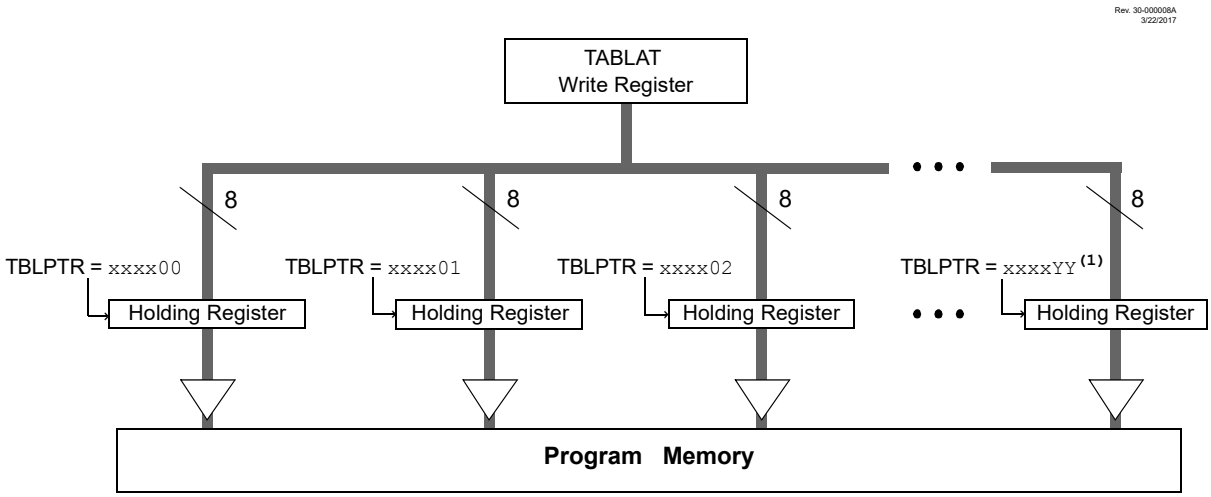
If the PFM address in the NVMADR is write-protected, or if NVMADR points to an invalid location, the SECWR bit is cleared without any effect and the 11.5.1.2 NVMERR is set.

CPU operation is suspended during a long write cycle and resumes when the operation is complete. The long write operation completes in one extended instruction cycle. When complete, the SECWR or WR bit is cleared by hardware and NVMIF is set. An interrupt will occur if NVMIE is also set. The holding registers are unchanged. NVMEN 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.

Important: The holding registers are undefined on device Resets and unchanged after write operations. Individual bytes of program memory may be modified, provided that the modification does not attempt to change any bit from a '0' to a '1'. When modifying individual bytes with a sector write operation, it is necessary to load all holding registers with either FFh or the existing contents of memory before executing a long write operation. The fastest way to do this is by performing a sector read operation.

Figure 11-8. Table Writes to Program Flash Memory



Note: Refer to Flash Memory Organization by Device table for number of holding registers (e.g. YY = FFh for 256 holding registers).

Related Links

11.1 Program Flash Memory

11.1.4.1 PFM Sector Write Sequence

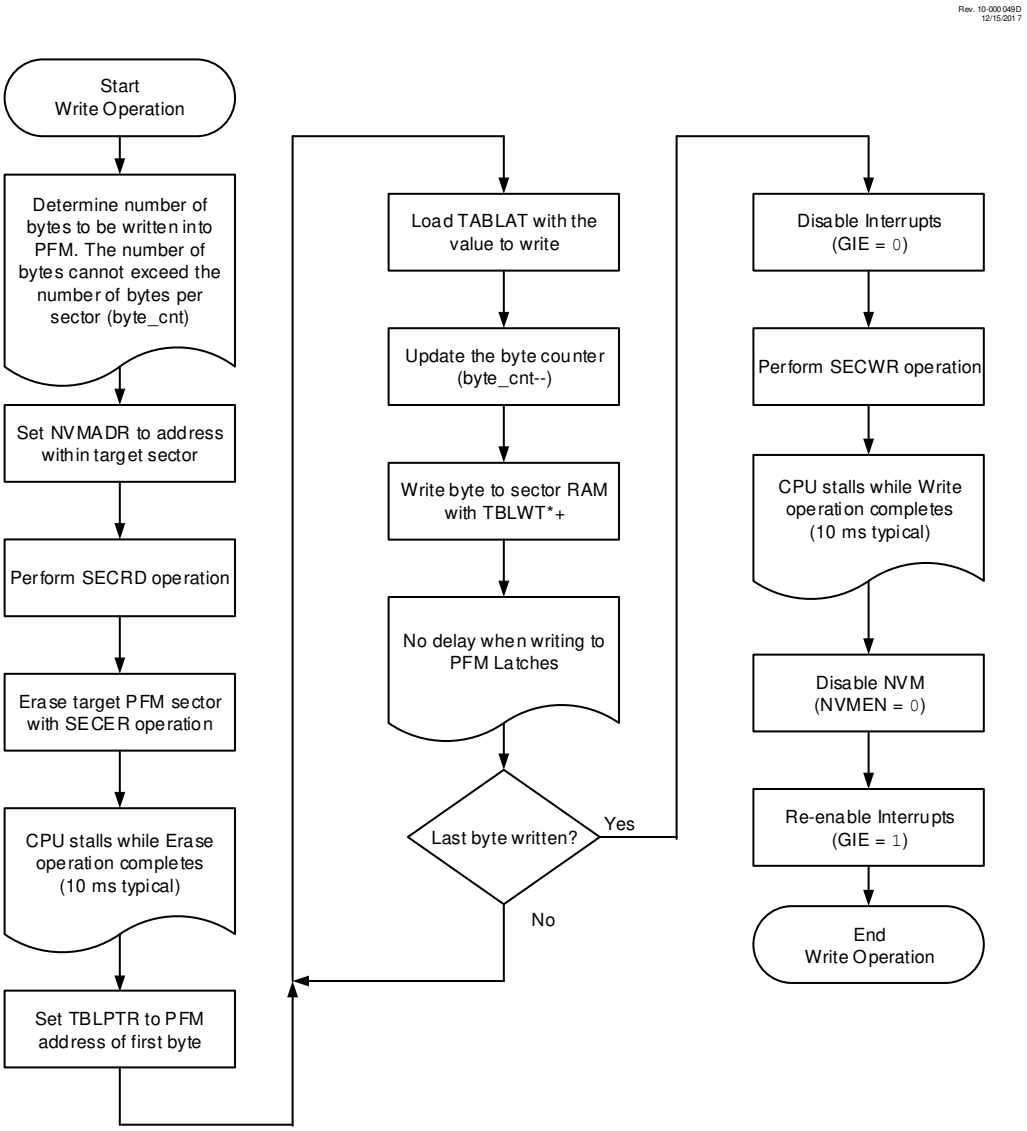
The sequence of events for programming a block of internal program memory location should be:

1. Set NVMADR with the target sector address.
2. Read the PFM sector into RAM with the SECRD operation.
3. Execute the sector erase procedure (see 11.1.3.1 PFM Erase Sequence).
4. SECER is set as the last step in the erase sequence.
5. The CPU will stall for the duration of the erase (about 10 ms using internal timer).
6. Load TBLPTR with address of first byte being updated.
7. Write the n-byte block into the holding registers with auto-increment. Refer to the Flash Memory Organization by Device table for the number of holding registers.
8. Disable interrupts.
9. Execute the sector write unlock sequence (see 11.1.2 NVM Unlock Sequence).
10. SECWR bit is set as last step in the unlock sequence.
11. The CPU will stall for the duration of the write (about 10 ms using internal timer).
12. Re-enable interrupts.
13. Verify the memory (table read).

This procedure will require about 20 ms to update each block of memory. See the "Memory Programming Specifications" for more detail. An example of the required code is given below.

Important: Before setting the SECWR bit, the NVMADR value needs to be within the intended address range of the target PFM sector.

Figure 11-9. Program Flash Memory (PFM) Write Flowchart



Example 11-4. Writing a Sector of Program Flash Memory

```
; Code sequence to modify one word in a programmed sector of PFM
; Calling routine should check WREG for the following errors:
;
;    00h = Successful modification
;    01h = Read error
;    02h = Erase error
;    03h = Write error
;
READ_BLOCK:
        MOVLW   CODE_ADDR_UPPER     ; load NVMADR with the base
        MOVWF   NVMADRU             ; address of the memory sector
        MOVLW   CODE_ADDR_HIGH
```

```
        MOVWF   NVMADRH
        MOVLW   CODE_ADDR_LOW
        MOVWF   NVMADRL
        BCF     INTCON, GIE         ; disable interrupts
        BSF     NVMCON0, NVMEN      ; enable NVM
; ----- Required Sequence -----
        MOVLW   0BBh
        MOVWF   NVMCON2             ; first unlock byte = 0BBh
        MOVLW   44h
        MOVWF   NVMCON2             ; second unlock byte = 44h
        BSF     NVMCON1, SECRD      ; start sector read (CPU stall)
; -----------------------------
        BTFSC   NVMCON0, NVMERR     ; Verify no error occurred during read
        BRA     NVM_RDERR           ; return read error code
ERASE_BLOCK:                        ; NVMADR is already pointing to target
block
; ----- Required Sequence -----
        MOVLW   0CCh
        MOVWF   NVMCON2             ; first unlock byte = 0CCh
        MOVLW   33h
        MOVWF   NVMCON2             ; second unlock byte = 33h
        BSF     NVMCON1, SECER      ; start sector erase (CPU stall)
; -----------------------------
        BTFSC   NVMCON0, NVMERR     ; Verify no error occurred during erase
        BRA     NVM_ERERR           ; return erase error code
MODIFY_WORD:
        MOVLW   TARGET_ADDR_UPPER   ; load TBLPTR with the target address
        MOVWF   TBLPTRU             ; of the LSByte
        MOVLW   TARGET_ADDR_HIGH
        MOVWF   TBLPTRH
        MOVLW   TARGET_ADDR_LOW
        MOVWF   TBLPTRL
        MOVLW   NEW_DATA_LOW        ; update holding register
        MOVWF   TABLAT
        TBLWT*+
        MOVLW   NEW_DATA_HIGH
        MOVWF   TABLAT
        TBLWT*+
PROGRAM_MEMORY:                     ; NVMADR is already pointing to target
block
; ----- Required Sequence -----
        MOVLW   0DDh
        MOVWF   NVMCON2             ; first unlock byte = 0DDh
        MOVLW   22h
        MOVWF   NVMCON2             ; second unlock byte = 22h
        BSF     NVMCON1, SECWR      ; start sector programming (CPU stall)
; -----------------------------
        BTFSC   NVMCON0, NVMERR     ; Verify no error occurred during write
        BRA     NVM_WRERR           ; return sector write error code
        CLRF    WREG,F              ; return with no error
        BRA     NVM_EXIT
NVM_RDERR:
        MOVLW   01h
        BRA     NVM_EXIT
NVM_ERERR:
        MOVLW   02h
        BRA     NVM_EXIT
NVM_WRERR:
        MOVLW   03h
NVM_EXIT:
        BCF    NVMCON0, NVMEN       ; disable NVM
        BSF    INTCON, GIE          ; re-enable interrupts
        RETURN
```

Related Links

38.3.5 Memory Programming Specifications

11.1.4.2 PFM Word Write Sequence

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

1. Set NVMADR with the target word address.

2. Load NVMDAT with desired word.
3. Disable interrupts.
4. Execute the word/byte write unlock sequence (see 11.1.2 NVM Unlock Sequence).
5. WR bit is set as last step in the unlock sequence.
6. The CPU will stall for the duration of the write (about 50 µs using internal timer).
7. Re-enable interrupts.
8. Verify the memory (word read).

Example 11-5. Writing a Word of Program Flash Memory

```
; Code sequence to program one erased word of PFM
; Target address is in WORD_ADDR_UPPER:WORD_ADDR_HIGH:WORD_ADDR_LOW
; Target data is in WORD_HIGH_BYTE:WORD_LOW_BYTE
; Calling routine should check WREG for the following errors:
;
;    00h = Successful modification
;    03h = Write error
;
WORD_WRITE:
        MOVF    WORD_ADDR_UPPER, W  ; load NVMADR with the target
        MOVWF   NVMADRU             ; address of the word
        MOVF    WORD_ADDR_HIGH, W
        MOVWF   NVMADRH
        MOVF    WORD_ADDR_LOW, W
        MOVWF   NVMADRL
        MOVF    WORD_HIGH_BYTE, W   ; load NVMDAT with desired
        MOVWF   NVMDATH             ; word
        MOVF    WORD_LOW_BYTE, W
        MOVWF   NVMDATL
        BCF     INTCON, GIE         ; disable interrupts
        BSF     NVMCON0, NVMEN      ; enable NVM
PROGRAM_WORD:
; ----- Required Sequence -----
        MOVLW   055h
        MOVWF   NVMCON2             ; first unlock byte = 055h
        MOVLW   0AAh
        MOVWF   NVMCON2             ; second unlock byte = 0AAh
        BSF     NVMCON1, WR         ; start word programming (CPU stall)
; -----------------------------
        BTFSC   NVMCON0, NVMERR     ; Verify no error occurred during write
        BRA     NVM_WRERR           ; return sector write error code
VERIFY_WORD:
        BSF     NVMCON0, RD         ; Retrieve word from PFM
        MOVF    WORD_HIGH_BYTE, W   ; Verify high byte
        CPFSEQ  NVMDATH
        BRA     NVM_RDERR           ; not equal - return error code
        MOVF    WORD_LOW_BYTE, W    ; Verify low byte
        CPFSEQ  NVMDATL
        BRA     NVM_RDERR           ; not equal - return error code

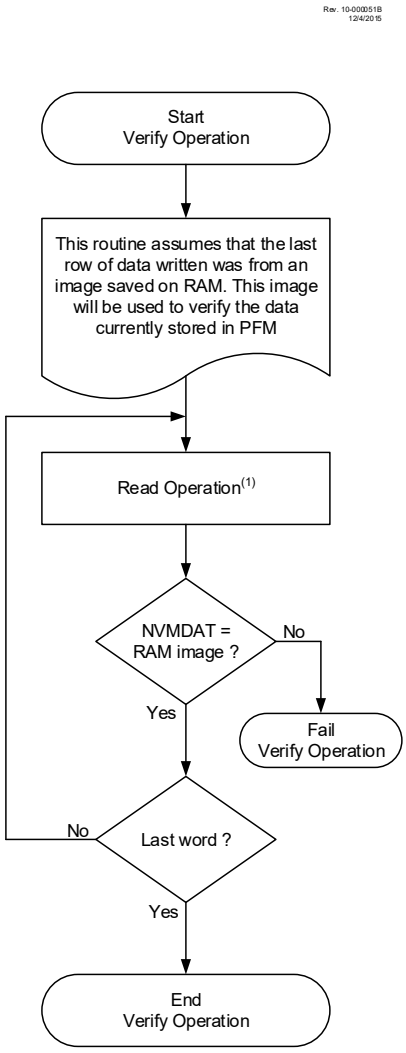
        CLRF    WREG,F              ; return with no error
        BRA     NVM_EXIT
NVM_RDERR:
        MOVLW   01h
        BRA     NVM_EXIT
NVM_WRERR:
        MOVLW   03h
NVM_EXIT:
        BCF    NVMCON0, NVMEN       ; disable NVM
        BSF    INTCON, GIE          ; re-enable interrupts
        RETURN
```

11.1.4.3 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 sector RAM after the last write is complete.

Figure 11-10. Program Flash Memory Verify Flowchart



11.1.4.4 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 $\overline{\text{MCLR}}$ Reset or a WDT Time-out Reset during normal operation, the NVMERR bit will be set which the user can check to decide whether a rewrite of the location(s) is needed.

11.1.4.5 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, RD, SECWR, SECRD, and SECER bits are gated through the NVMEN bit. It is suggested to have the NVMEN bit cleared at all times except during memory writes and reads. This prevents memory operations if any of the control bits are set accidentally.
2. The NVM unlock sequence must be performed each time before all but the RD operation.

11.2 User ID, Device ID and Configuration Word Access

The NVMADR value determines which NVM address space is accessed. The User IDs and Configuration Words areas allow read and write whereas Device and Revision IDs allow read-only (see NVM Organization table).

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

Writing to the Configuration space can only be done with the SECWR operation. A sector erase operation on Configuration space cannot be performed with the SECER operation. When a SECWR operation is performed on the Configuration space a sector erase is performed automatically before the sector write. Any code protection settings that are not enabled will remain not enabled after the sector write operation unless the new values enable them. However, any code protection settings that are enabled cannot be disabled by a self write of the Configuration space. The user can modify the Configuration space by the following steps:

1. Read Configuration space with the unlock and SECRD sequence.
2. Modify the desired configuration word holding registers using TBLPTR address, TABLAT data, and `TBLWT*` instruction.
3. Write the holding registers to Configuration space with the unlock and SECWR sequence.

Related Links

11. (NVM) Nonvolatile Memory Control

11.3 Data Flash Memory (DFM)

The Data Flash Memory is a nonvolatile memory array, also referred to as EEPROM. The DFM is separate from the Program Flash Memory, which is used for long-term storage of program data. The DFM is mapped above program memory space and is indirectly addressed through the NVM Special Function Registers (SFRs). The DFM is readable and writable during normal operation over the entire V_{DD} range.

Five SFRs are used to read and write to the data DFM. They are:

- NVMCON0
- NVMCON1
- NVMCON2
- NVMDAT
- NVMADR

The DFM can only be read and written one byte at a time. When interfacing to the data memory block, NVMDATL holds the 8-bit data for read/write and the 11.5.5 NVMADR register holds the address of the DFM location being accessed.

The DFM is rated for high erase/write cycle endurance. A byte write automatically erases the location and writes the new data (erase-before-write). The write time is controlled by an internal programming timer; it will vary with voltage and temperature as well as from chip-to-chip. Refer to the Data EEPROM Memory parameters in the electrical specifications section for the limits.

Related Links

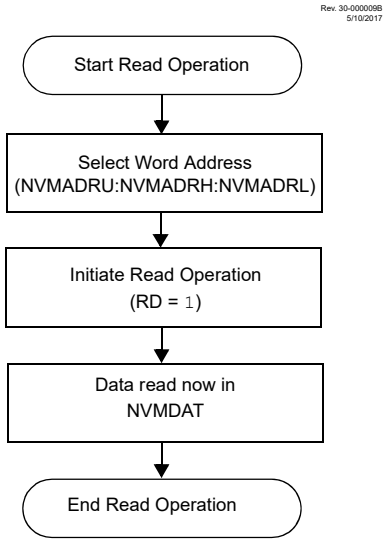
38.3.5 Memory Programming Specifications

11.3.1 Reading the DFM

To read a DFM location, the user must write the address to the NVMADR register, enable NVM control by setting the 11.5.1.1 NVMEN bit, and then set the 11.5.2.5 RD control bit. 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 the following flowchart.

Figure 11-11. DFM Read Flowchart



Example 11-6. DFM Read

```
; Data Flash Memory Address to read
        BSF     NVMCON0, NVMEN          ; Enable NVM
        MOVF    DFM_ADDRL, W            ;
        MOVWF   NVMADRL                 ; Setup Address low byte
        MOVF    DFM_ADDRH, W            ;
        MOVWF   NVMADRH                 ; Setup Address high byte
        MOVF    DFM_ADDRU, W            ;
        MOVWF   NVMADRU                 ; Setup Address upper byte
        BSF     NVMCON1, RD             ; Issue EE Read
        MOVF    NVMDAT, W               ; W = EE_DATA
        BCF     NVMCON0, NVMEN          ; Disable NVM
```

Only byte reads are supported for DFM. Reading a block of DFM with a SECRD operation is not supported.

11.3.2 Writing to DFM

To write a DFM location, the address must first be written to the NVMADR register and the data written to the NVMDATL register. The sequence shown in 11.1.2 NVM Unlock Sequence must be followed to initiate the write cycle. Block writes, also referred to as sector writes, are not supported for the DFM.

The write will not begin if NVM Unlock sequence is not exactly followed for each byte. It is strongly recommended that interrupts be disabled during this code segment.

Additionally, the 11.5.1.1 NVMEN bit must be set to enable NVM control. This mechanism prevents accidental writes to DFM due to unexpected code execution (i.e., runaway programs). The NVMEN bit

should be kept clear at all times, except when updating the DFM. The NVMEN bit is not cleared by hardware.

After a write sequence has been initiated, NVMCON1, NVMADR and NVMDAT cannot be modified. The WR bit will be inhibited from being set unless the NVMEN bit is set.

After a write sequence has been initiated, clearing the NVMEN bit will not affect this write cycle. A single DFM byte is written and the operation includes an implicit erase cycle for that word. 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.

Example 11-7. DFM Write

```
; Data Flash Memory Address to write
        MOVF    DFM_ADDRL, W        ;
        MOVWF   NVMADRL             ; Setup Address low byte
        MOVF    DFM_ADDRH, W        ;
        MOVWF   NVMADRH             ; Setup Address high byte
        MOVF    DFM_ADDRU, W        ;
        MOVWF   NVMADRU             ; Setup Address upper byte
; Data Memory Value to write
        MOVF    DMF_DATA, W         ;
        MOVWF   NVMDATL             ;
; Enable writes
        BSF     NVMCON0, NVMEN      ; Enable NVM
; Disable interrupts
        BCF     INTCON, GIE         ;
; -------- Required Sequence ---------
        MOVLW   55h                 ;
        MOVWF   NVMCON2             ;
        MOVLW   AAh                 ;
        MOVWF   NVMCON2             ;
        BSF     NVMCON1, WR         ; Set WR bit to begin write
; ------------------------------------
; Wait for write to complete
        BTFSC   NVMCON1, WR         ; DFM writes do not stall the CPU
        BRA     $-2
; Enable INT
        BSF     INTCON, GIE         ;
; Disable writes
        BCF     NVMCON0, NVMEN      ;
```

11.3.3 DFM 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.

11.3.4 Operation During Code-Protect and Write-Protect

DFM has its own code-protect bits in the Configuration Words. In-Circuit Serial Programming read and write operations are disabled when code protection is enabled. However, internal reads operate normally. Internal writes operate normally provided that write protection is not enabled.

If the DFM is write-protected or if NVMADR points at an invalid address location, the WR bit is cleared without any effect. NVMERR is signaled in this scenario.

11.3.5 Protection Against Spurious Write

There are conditions when the user may not want to write to the DFM. To protect against spurious DFM writes, various mechanisms have been implemented. On any Reset, the NVMEN bit is cleared. In addition, writes to the DFM are blocked during the Power-up Timer period (T_{PWRT}).

The unlock sequence and the NVMEN bit together help prevent an accidental write during brown-out, power glitch or software malfunction.

11.3.6 Erasing the DFM

DFM can be erased by writing 0xFF to all locations in the memory that need to be erased.

Example 11-8. DFM Erase Routine

```
        CLRF    NVMADRL             ; Clear address low byte
        CLRF    NVMADRH             ; Clear address high byte
        MOVLW   31h                 ; EEPROM upper address
        MOVWF   NVMADRU             ; Set address upper byte
        SETF    NVMDAT              ; Load 0xFF to data register
        BCF     INTCON, GIE         ; Disable interrupts
        BSF     NVMCON0, NVMEN      ; Enable NVM
Loop:                               ; Loop to refresh array
        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 EEPROM is 256 bytes or less
        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     NVMCON0, NVMEN      ; Disable NVM
        BSF     INTCON, GIE         ; Enable interrupts
```

11.4 Register Summary: NVM Control

Address	Name	Bit Pos.								
0x0F79	NVMADR	7:0	NVMADRL[7:0]							
		15:8	NVMADRH[7:0]							
		23:16			NVMADRU[5:0]					
0x0F7C	NVMDAT	7:0	NVMDATL[7:0]							
		15:8	NVMDATH[7:0]							
0x0F7E	Reserved									
0x0F7F	NVMCON0	7:0	NVMEN			NVMERR	Reserved			
0x0F80	NVMCON1	7:0		SECER	SECWR	WR			SECRD	RD
0x0F81	NVMCON2	7:0	NVMCON2[7:0]							
0x0F82 ... 0x0FF4	Reserved									
0x0FF5	TABLAT	7:0	TABLAT[7:0]							
0x0FF6	TBLPTR	7:0	TBLPTRL[7:0]							
		15:8	TBLPTRH[7:0]							
		23:16			TBLPTR21	TBLPTRU[4:0]				

11.5 Register Definitions: Nonvolatile Memory

11.5.1 NVMCON0

Name: NVMCON0
Address: 0xF7F

Nonvolatile Memory Control 0 Register

Bit	7	6	5	4	3	2	1	0
	NVMEN			NVMERR	Reserved			
Access	R/W			R/W/HS	R/W			
Reset	0			x	0			

Bit 7 – NVMEN NVM Enable bit

Value	Description
1	NVM is enabled for all operations
0	NVM is disabled for all operations except single byte or word read.

Bit 4 – NVMERR
NVM Error Flag bit

Value	Description
1	An attempted NVM write or sector read did not complete successfully. Must be cleared by software.
0	All NVM operations have completed successfully.

Bit 3 – Reserved
Reserved - Do not use. This bit must be maintained as '0'.

11.5.2 NVMCON1

Name: NVMCON1
Address: 0xF80

Nonvolatile Memory Control 1 Register

Bit	7	6	5	4	3	2	1	0
		SECER	SECWR	WR			SECRD	RD
Access		R/S/HC	R/S/HC	R/S/HC			R/S/HC	R/S/HC
Reset		0	0	0			0	0

Bit 6 – SECER
NVM Sector Erase Enable Control bit

Value	Condition	Description
1	NVMADR points to PFM	Immediately following the sector erase unlock sequence, perform a sector erase operation. Stays set until operation is complete. Cannot be cleared by software.
0	NVMADR points to PFM	NVM sector erase operation is complete and inactive.

Bit 5 – SECWR
NVM Sector Write Enable Control bit

Value	Condition	Description
1	NVMADR points to PFM or CONFIG	Immediately following the sector write unlock sequence, initiates the PFM sector write operation. Stays set until the write is complete. Cannot be cleared by software.
0	NVMADR points to PFM or CONFIG	NVM sector write operation is complete and inactive.

Bit 4 – WR
NVM Write Control bit

Value	Condition	Description
1	NVMADR points to DFM	Immediately following the DFM write unlock sequence, initiates a DFM byte erase/write sequence using data in the NVMDATL register. Stays set until the operation is complete. Cannot be cleared by software.
1	NVMADR points to PFM	Immediately following the PFM write unlock sequence, initiates an NVM word write sequence using data in the NVMDATH:L register pair. Stays set until the operation is complete. Cannot be cleared by software.
0	NVMADR points to PFM or DFM	NVM byte/word write operation is complete and inactive.

Bit 1 – SECRD
PFM Sector Read Enable Control bit

Value	Condition	Description
1	NVMADR points to PFM or CONFIG	Immediately following the sector read unlock sequence, initiates a read of one full sector from PFM into sector RAM. Stays set until the read is complete. Cannot be cleared by software.
0	NVMADR points to PFM or CONFIG	PFM sector read operation is complete and inactive.

Bit 0 – RD
NVM Read Enable Control bit

Value	Condition	Description
1	NVMADR points to DFM	Initiates a read of one byte from DFM into the NVMDATL register. Stays set until the read is complete. Cannot be cleared by software.
1	NVMADR points to PFM	Initiates a read of one word from PFM into the NVMDATH:L registers. Stays set until the read is complete. Cannot be cleared by software.
0	NVMADR points to PFM or DFM	NVM read operation is complete and inactive.

11.5.3 NVMCON2

Name: NVMCON2
Address: 0xF81

Nonvolatile Memory Control 2 Register
Note: This register always reads zeros, regardless of data written.
Refer to the NVM Unlock Sequence section

Bit	7	6	5	4	3	2	1	0
	NVMCON2[7:0]							
Access	WO	WO	WO	WO	WO	WO	WO	WO
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – NVMCON2[7:0]

11.5.4 NVMDAT

Name: NVMDAT
Address: 0xF7C

NVM Data Register

Bit	15	14	13	12	11	10	9	8
	NVMDATH[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

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

Bits 15:8 – NVMDATH[7:0]
NVMDAT Most Significant Byte of data written to and read from PFM.

Bits 7:0 – NVMDATL[7:0]
NVMDAT Least Significant Byte of data written to and read from PFM or DFM.

11.5.5 NVMADR

Name: NVMADR
Address: 0xF79

NVM Address Register

Bit	23	22	21	20	19	18	17	16
			NVMADRU[5:0]					
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	NVMADRH[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

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

Bits 21:16 – NVMADRU[5:0]
NVM address bits <21:16> .

Bits 15:8 – NVMADRH[7:0]
High byte of NVM address.

Bits 7:0 – NVMADRL[7:0]
Low byte of NVM address.

11.5.6 TABLAT

Name: TABLAT
Address: 0xFF5

Program, Configuration, Device ID, and User ID Memory Data

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

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

11.5.7 TBLPTR

Name: TBLPTR
Address: 0xFF6

Program, Configuration, Device ID and User ID Memory Address

Bit	23	22	21	20	19	18	17	16
			TBLPTR21	TBLPTRU[4:0]				
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	TBLPTRH[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

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

Bit 21 – TBLPTR21 NVM Most Significant Address bit

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

Bits 20:16 – TBLPTRU[4:0] NVM Upper Address bits

Bits 15:8 – TBLPTRH[7:0] High Byte of NVM Address bits

Bits 7:0 – TBLPTRL[7:0] Low Byte of NVM Address bits

12. 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

```
MOVF    ARG1, W   ;

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

Example 12-2. 8x8 Signed Multiply Routine

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

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

Routine	Multiply Method	Program Memory (Words)	Cycles (Max)	Time			
				@ 64 MHz	@ 40 MHz	@ 10 MHz	@ 4 MHz
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. The equation below shows the algorithm that is used. The 32-bit result is stored in four registers (RES<3:0>).

16 x 16 Unsigned Multiplication Algorithm

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

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

Example 12-4 shows the sequence to do a 16 x 16 signed multiply. The equation below 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.

16 x 16 Signed Multiplication Algorithm

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

Example 12-4. 16 x 16 Signed Multiply Routine

```
        MOVF    ARG1L, W
        MULW    ARG2L           ; ARG1L * ARG2L → PRODH:PRODL
        MOVF    PRODH, RES1     ;
        MOVFF   PRODL, RES0     ;
;
        MOVF    ARG1H, W
        MULWF   ARG2H           ; ARG1H * ARG2H → PRODH:PRODL
        MOVFF   PRODH, RES3     ;
        MOVFF   PRODL, RES2     ;
;
        MOVF    ARG1L, W
        MULWF   ARG2H           ; ARG1L * ARG2H → PRODH:PRODL
        MOVF    PRODL, W        ;
        ADDWF   RES1, F         ; Add cross products
        MOVF    PRODH, W        ;
        ADDWFC  RES2, F         ;
        CLRF    WREG            ;
        ADDWFC  RES3, F         ;
;
        MOVF    ARG1H, W        ;
        MULWF   ARG2L           ; ARG1H * ARG2L → PRODH:PRODL
        MOVF    PRODL, W        ;
        ADDWF   RES1, F         ; Add cross products
        MOVF    PRODH, W        ;
        ADDWFC  RES2, F         ;
        CLRF    WREG            ;
        ADDWFC  RES3, F         ;
;
        BTFSS   ARG2H, 7        ; ARG2H:ARG2L neg?
        BRA     SIGN_ARG1       ; no, check ARG1
        MOVF    ARG1L, W        ;
        SUBWF   RES2            ;
        MOVF    ARG1H, W        ;
        SUBWFB  RES3
;

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

12.3 Register Summary - 8x8 Hardware Multiplier

Address	Name	Bit Pos.								
0x0FF3	PROD	7:0	PRODL[7:0]							
		15:8	PRODH[7:0]							

12.4 Register Definitions: 8x8 Hardware Multiplier

12.4.1 PROD

Name: PROD
Address: 0xFF3

Product Register Pair

The PROD register stores the 16-bit result yielded by the unsigned operation performed by the 8x8 hardware multiplier.

Bit	15	14	13	12	11	10	9	8
	PRODH[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	x	x	x	x	x	x	x	x

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

Bits 15:8 – PRODH[7:0]
PROD Most Significant bits

Bits 7:0 – PRODL[7:0]
PROD Least Significant bits

13. 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

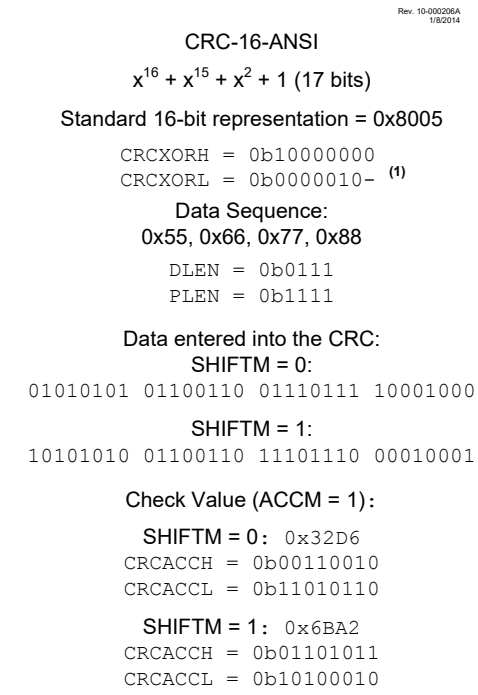
13.1 CRC Module Overview

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 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 13.12.4 CRCACC registers for user storage. The CRC module uses an XOR shift register implementation to perform the polynomial division required for the CRC calculation.

Figure 13-1. CRC Example



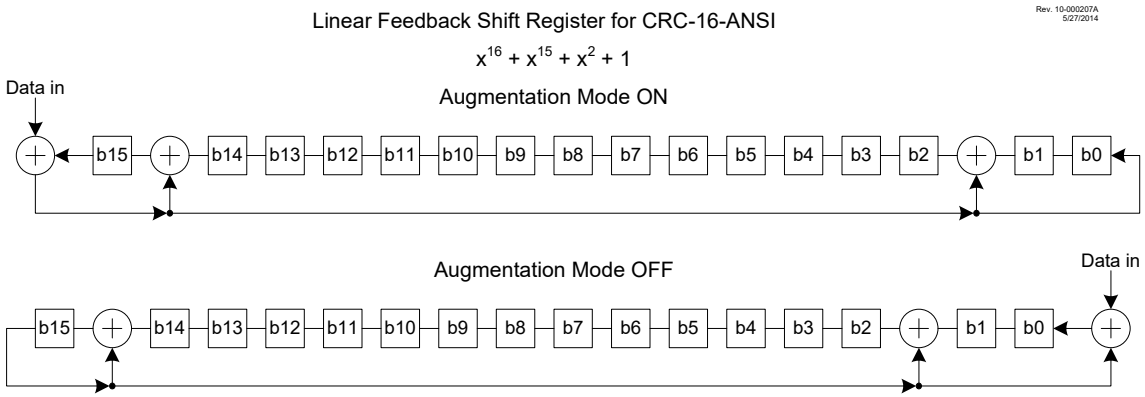
Note 1: Bit 0 is unimplemented. The LSb of any CRC polynomial is always '1' and will always be treated as a '1' by the CRC for calculating the CRC check value. This bit will be read in software as a '0'.

13.3 CRC Polynomial Implementation

Any polynomial can be used. The polynomial and accumulator sizes are determined by the 13.12.2.2 PLEN bits. For an n-bit accumulator, PLEN = n-1 and the corresponding polynomial is n+1 bits. Therefore, the accumulator can be any size up to 16 bits with a corresponding polynomial up to 17 bits. The MSb and LSb of the polynomial are always '1' which is forced by hardware. However, the LSb of the CRCXORL register is unimplemented and always reads as '0'.

All polynomial bits between the MSb and LSb are specified by the 13.12.6 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. Refer to Figure 13-1.

Figure 13-2. CRC LFSR Example



13.4 CRC Data Sources

Data can be input to the CRC module in two ways:

- User data using the 13.12.3 CRCDAT registers
- From Flash Memory using the Program Memory Scanner

Up to 16 bits of data per word are specified with the 13.12.2.1 DLEN bits. Only the number of data bits in the CRCDATA registers specified by DLEN will be used, other data bits in CRCDATA registers will be ignored.

Data is moved into the 13.12.5 CRCSHIFT as an intermediate to calculate the check value located in the 13.12.4 CRCACC registers.

The 13.12.1.5 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 registers to the appropriate value before beginning the CRC.

13.4.1 CRC from User Data

To use the CRC module on data input from the user, the user must write the data to the CRCDAT registers. The data from the CRCDAT registers will be latched into the shift registers on any write to the CRCDATL register.

13.4.2 CRC from Flash

To use the CRC module on data located in Flash memory, the user can initialize the Program Memory Scanner as defined in the 13.8 Program Memory Scan Configuration section.

13.5 CRC Check Value

The CRC check value will be located in the CRCACC registers after the CRC calculation has finished. The check value will depend on the 13.12.1.4 ACCM and 13.12.1.5 SHIFTM mode settings.

When the ACCM bit is set, the CRC module augments the data with a number of zeros equal to the length of the polynomial to align the final check value. When the ACCM bit is not set, the CRC will stop at the end of the data. A number of zeros equal to the length of the polynomial can then be entered into 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. When creating a check value to be appended to a data stream, then a bit reversal must be performed on the final value to achieve the correct checksum. CRC can be used to do this reversal by following the steps below:

1. Save CRCACC value in user RAM space
2. Clear the CRCACC registers
3. Clear the CRCXOR registers
4. Write the saved CRCACC value to the CRCDAT input

The properly oriented check value will be in the CRCACC registers as the result.

13.6 CRC Interrupt

The CRC will generate an interrupt when the 13.12.1.3 BUSY bit transitions from 1 to 0. The CRCIF Interrupt Flag bit of the PIRx 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 PIEx register.

13.7 Configuring the CRC

The following steps illustrate how to properly configure the CRC.

1. Determine if the automatic program memory scan will be used with the scanner or manual calculation through the SFR interface and perform the actions specified in 13.4 CRC Data Sources, depending on which decision was made.
2. If desired, seed a starting CRC value into the 13.12.4 CRCACC registers.
3. Program the 13.12.6 CRCXOR registers with the desired generator polynomial.
4. Program the 13.12.2.1 DLENbits with the length of the data word - 1 (refer to Figure 13-1). This determines how many times the shifter will shift into the accumulator for each data word.
5. Program the 13.12.2.2 PLEN bits with the length of the polynomial -2 (refer to Figure 13-1).
6. Determine whether shifting in trailing zeros is desired and set the 13.12.1.4 ACCM bit accordingly.
7. Likewise, determine whether the MSb or LSb should be shifted first and write the 13.12.1.5 SHIFTM bit accordingly.
8. Set the 13.12.1.2 GO bit to begin the shifting process.
9. If manual SFR entry is used, monitor the 13.12.1.6 FULL bit. When FULL = 0, another word of data can be written to the 13.12.3 CRCDAT registers, keeping in mind that Most Significant Byte, CRCDATH, should be written first if the data has more than eight bits, as the shifter will begin upon the CRCDATL register being written.
10. If the scanner is used, the scanner will automatically stuff words into the CRCDAT registers as needed, as long as the 13.12.7.2 SCANGO bit is set.

11. If using the Flash memory scanner, monitor the PIRx SCANIF bit (or the SCANGO bit) for the scanner to finish pushing information into the CRCDATA registers. After the scanner is completed, monitor the 13.12.7.3 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 CRCACC registers.
12. If manual entry is used, monitor the BUSY bit to determine when the CRCACC registers hold the valid check value.

13.8 Program Memory Scan Configuration

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 the following steps need to performed:

1. Set both the 13.12.1.1 EN and 13.12.7.1 SCANEN bits. If they get disabled, all internal states of the scanner and the CRC are reset. However, the CRC SFR registers are unaffected.
2. Choose which memory access mode is to be used (see 13.10 Scanning Modes) and set the 13.12.7.6 MODE bits accordingly.
3. Based on the memory access mode, set the 13.12.7.5 INTM bits to the appropriate Interrupt mode (see 13.10.5 Interrupt Interaction)
4. Set the 13.12.8 SCANLADR and 13.12.9 SCANHADR registers with the respective beginning and ending locations in memory that are to be scanned.
5. The 13.12.1.2 GO bit must be set before setting the 13.12.7.2 SCANGO bit. Setting the SCANGO bit starts the scan. Both the 13.12.1.1 EN and 13.12.1.2 GO bits must be enabled to use the scanner. When either of these bits are disabled, the scan aborts and the 13.12.7.4 INVALID bit 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 to terminate the scan early if desired.

13.9 Scanner Interrupt

The scanner will trigger an interrupt when the SCANGO bit transitions from '1' to '0'. The SCANIF interrupt flag of PIRx 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 PIEx register.

13.10 Scanning Modes

The memory scanner can scan in four modes: Burst, Peek, Concurrent, and Triggered. These modes are controlled by the 13.12.7.6 MODE bits. The four modes are summarized in Table 13-1.

13.10.1 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.

13.10.2 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.

13.10.3 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 source which is determined by the 13.12.10 SCANTRIG register.

13.10.4 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 13-1. 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

13.10.5 Interrupt Interaction

The 13.12.7.5 INTM bit controls the scanner's response to interrupts depending on which mode the NVM scanner is in, as described in the following table.

Table 13-2. Scan Interrupt Modes

INTM	MODE<1:0>		
	MODE == Burst	MODE == CONCURENT 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	This bit is ignored

INTM	MODE<1:0>		
	MODE == Burst	MODE == CONCURENT or TRIGGERED	MODE ==PEEK
		scan resumes when the interrupt is complete.	
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.

13.10.6 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.

13.10.7 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 the following table.

Table 13-3. 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.

ICD Halt	Scanner Operating Mode		
	Peek	Concurrent Triggered	Burst
PC Breakpoint		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. Note that the burst can be interrupted by an external halt.
Data Breakpoint		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.	
Single Step		If a scan cycle is ready after the debug instruction is executed, the scan will read PFM and then the ICD is re-entered.	
SWBP and ICDINST		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.

13.10.8 Peripheral Module Disable

Both the CRC and scanner module can be disabled individually by setting the CRCMD and SCANMD bits of the PMD0 register. 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.

13.11 Register Summary - CRC

Address	Name	Bit Pos.								
0x0F44	SCANLADR	7:0	SCANLADRL[7:0]							
		15:8	SCANLADRH[7:0]							
		23:16			SCANLADRU[5:0]					
0x0F47	SCANHADR	7:0	SCANHADRL[7:0]							
		15:8	SCANHADRH[7:0]							
		23:16			SCANHADRU[5:0]					
0x0F4A	SCANCON0	7:0	SCANEN	SCANGO	BUSY	INVALID	INTM		MODE[1:0]	
0x0F4B	SCANTRIG	7:0					TSEL[3:0]			
0x0F4C ... 0x0F6E	Reserved									
0x0F6F	CRCDAT	7:0	CRCDATL[7:0]							
		15:8	CRCDATH[7:0]							
0x0F71	CRCACC	7:0	CRCACCL[7:0]							
		15:8	CRCACCH[7:0]							
0x0F73	CRCSHIFT	7:0	CRCSHIFTL[7:0]							
		15:8	CRCSHIFTH[7:0]							
0x0F75	CRCXOR	7:0	CRCXORL[6:0]							CRCXORL0
		15:8	CRCXORH[7:0]							
0x0F77	CRCCON0	7:0	EN	GO	BUSY	ACCM			SHIFTM	FULL
0x0F78	CRCCON1	7:0	DLEN[3:0]				PLEN[3:0]			

13.12 Register Definitions: CRC and Scanner Control

Long bit name prefixes for the CRC are shown in the table below. Refer to the *"Long Bit Names Section"* for more information.

Table 13-4. CRC Long Bit Name Prefixes

Peripheral	Bit Name Prefix
CRC	CRC

Related Links

1.4.2.2 Long Bit Names

13.12.1 CRCCON0

Name: CRCCON0
Address: 0xF77
Reset: 0

CRC Control Register 0

Bit	7	6	5	4	3	2	1	0
	EN	GO	BUSY	ACCM			SHIFTM	FULL
Access	R/W	R/W	RO	R/W			R/W	RO
Reset	0	0	0	0			0	0

Bit 7 – EN CRC Enable bit

Value	Description
1	CRC module is released from Reset
0	CRC is disabled and consumes no operating current

Bit 6 – GO CRC Start bit

Value	Description
1	Start CRC serial shifter
0	CRC serial shifter turned off

Bit 5 – BUSY CRC Busy bit

Value	Description
1	Shifting in progress or pending
0	All valid bits in shifter have been shifted into accumulator and EMPTY = 1

Bit 4 – ACCM Accumulator Mode bit

Value	Description
1	Data is augmented with zeros
0	Data is not augmented with zeros

Bit 1 – SHIFTM Shift Mode bit

Value	Description
1	Shift right (LSb)
0	Shift left (MSb)

Bit 0 – FULL Data Path Full Indicator bit

Value	Description
1	CRCDATH/L registers are full
0	CRCDATH/L registers have shifted their data into the shifter

13.12.2 CRCCON1

Name: CRCCON1
Address: 0xF78
Reset: 0

CRC Control Register 1

Bit	7	6	5	4	3	2	1	0
	DLEN[3:0]				PLEN[3:0]			
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:4 – DLEN[3:0] Data Length bits
Denotes the length of the data word -1 (See Figure 13-1)

Bits 3:0 – PLEN[3:0] Polynomial Length bits
Denotes the length of the polynomial -1 (See Figure 13-1)

13.12.3 CRCDAT

Name: CRCDAT
Address: 0xF6F

CRC Data Register

Bit	15	14	13	12	11	10	9	8
	CRCDATH[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	x	x	x	x	x	x	x	x

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

Bits 15:8 – CRCDATH[7:0] CRC Input/Output Data Most Significant Byte

Bits 7:0 – CRCDATL[7:0] CRC Input/Output Data Least Significant Byte

13.12.4 CRCACC

Name: CRCACC
Address: 0xF71
Reset: 0

CRC Accumulator Register

Bit	15	14	13	12	11	10	9	8
	CRCACCH[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

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

Bits 15:8 – CRCACCH[7:0] CRC Accumulator Register most significant byte
Writing to this register writes the Most Significant Byte of the CRC accumulator register. Reading from this register reads the Most Significant Byte of the CRC accumulator.

Bits 7:0 – CRCACCL[7:0] CRC Accumulator Register least significant byte
Writing to this register writes the Least Significant Byte of the CRC accumulator register. Reading from this register reads the Least Significant Byte of the CRC accumulator.

13.12.5 CRCSHIFT

Name: CRCSHIFT
Address: 0xF73
Reset: 0

CRC Shift Register

Bit	15	14	13	12	11	10	9	8
	CRCSHIFTH[7:0]							
Access	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	CRCSHIFTL[7:0]							
Access	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0

Bits 15:8 – CRCSHIFTH[7:0] CRC Shifter Register Most Significant Byte
Reading from this register reads the Most Significant Byte of the CRC Shifter.

Bits 7:0 – CRCSHIFTL[7:0] CRC Shifter Register Least Significant Byte
Reading from this register reads the Least Significant Byte of the CRC Shifter.

13.12.6 CRCXOR

Name: CRCXOR
Address: 0xF75

CRC XOR Register

Bit	15	14	13	12	11	10	9	8
	CRCXORH[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	x	x	x	x	x	x	x	x

Bit	7	6	5	4	3	2	1	0
	CRCXORL[6:0]							CRCXORL0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	U
Reset	x	x	x	x	x	x	x	1

Bits 15:8 – CRCXORH[7:0] XOR of Polynomial Term XN Enable Most Significant Byte

Bits 7:1 – CRCXORL[6:0] XOR of Polynomial Term XN Enable Least Significant Byte

Bit 0 – CRCXORL0 LSbit is unimplemented. Read as 1

13.12.7 SCANCON0

Name: SCANCON0
Address: 0xF4A

Scanner Access Control Register 0

Bit	7	6	5	4	3	2	1	0
	SCANEN	SCANGO	BUSY	INVALID	INTM		MODE[1:0]	
Access	R/W	R/W/HC	R	R	R/W		R/W	R/W
Reset	0	0	0	1	0		0	0

Bit 7 – SCANEN Scanner Enable bit(1)

Value	Description
1	Scanner is enabled
0	Scanner is disabled, internal states are reset

Bit 6 – SCANGO Scanner GO bit(2, 3)

Value	Description
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)

Value	Description
1	Scanner cycle is in process
0	Scanner cycle is complete (or never started)

Bit 4 – INVALID Scanner Abort Signal bit

Value	Description
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

Value	Condition	Description
X	MODE = 10	This bit is ignored
1	MODE = 01	CPU is stalled until all data is transferred. SCANGO is overridden (to zero) during interrupt operation; scanner resumes after returning from interrupt
0	MODE = 01	CPU is stalled until all data is transferred. SCANGO is not affected by interrupts, the interrupt response will be affected
1	MODE = 00 OR 01	SCANGO is overridden (to zero) during interrupt operation; scan operations resume after returning from interrupt
0	MODE = 00 OR 01	Interrupts do not prevent NVM access

Bits 1:0 – MODE[1:0] Memory Access Mode bits(5)

Value	Description
11	Triggered mode
10	Peek mode
01	Burst mode
00	Concurrent mode

Note:

1. Setting SCANEN = 0 (SCANCON0 register) does not affect any other register content.
2. This bit is cleared when LADR > HADR (and a data cycle is not occurring).
3. If INTM = 1, this bit is overridden (to zero, but not cleared) during an interrupt response.
4. BUSY = 1 when the NVM is being accessed, or when the CRC sends a ready signal.
5. See Table 13-1 for more detailed information.
6. 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.
7. CRCEN and CRCGO bits must be set before setting SCANGO bit. Refer to 13.8 Program Memory Scan Configuration.

13.12.8 SCANLADR

Name: SCANLADR
Address: 0xF44
Reset: 0

Scan Low Address Register

Bit	23	22	21	20	19	18	17	16
			SCANLADRU[5:0]					
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			0	0	0	0	0	0

Bit	15	14	13	12	11	10	9	8
	SCANLADRH[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

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

Bits 21:16 – SCANLADRU[5:0] Scan Start/Current Address upper byte
Upper bits of the current address to be fetched from, value increments on each fetch of memory.

Bits 15:8 – SCANLADRH[7:0] Scan Start/Current Address high byte
High byte of the current address to be fetched from, value increments on each fetch of memory.

Bits 7:0 – SCANLADRL[7:0] Scan Start/Current Address low byte
Low byte of the current address to be fetched from, value increments on each fetch of memory.

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`.
2. While SCANGO = `1`, writing to this register is ignored.

13.12.9 SCANHADR

Name: SCANHADR
Address: 0xF47
Reset: 0

Scan High Address Register

Bit	23	22	21	20	19	18	17	16
			SCANHADRU[5:0]					
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			1	1	1	1	1	1

Bit	15	14	13	12	11	10	9	8
	SCANHADRH[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1

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

Bits 21:16 – SCANHADRU[5:0] Scan End Address bits
Upper bits of the address at the end of the designated scan

Bits 15:8 – SCANHADRH[7:0] Scan End Address bits
High byte of the address at the end of the designated scan

Bits 7:0 – SCANHADRL[7:0] Scan End Address bits
Low byte 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.
2. While SCANGO = 1, writing to this register is ignored.

13.12.10 SCANTRIG

Name: SCANTRIG
Address: 0xF4B
Reset: 0

SCAN Trigger Selection Register

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

Bits 3:0 – TSEL[3:0] Scanner Data Trigger Input Selection bits

Table 13-5. SCAN Trigger Sources

TSEL	Trigger Source
1111	Reserved
1110	Reserved
1101	Reserved
1100	Reserved
1011	Reserved
1010	Reserved
1001	Reserved
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

14. Interrupts

The PIC18F24/25Q10 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
- PIRx (Interrupt flags)
- PIEx (Interrupt enables)
- IPRx (High/Low interrupt priority)

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 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 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. 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 GIEH bit enables all interrupts which have their associated bit in the IPRx register set. When the GIEH 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.

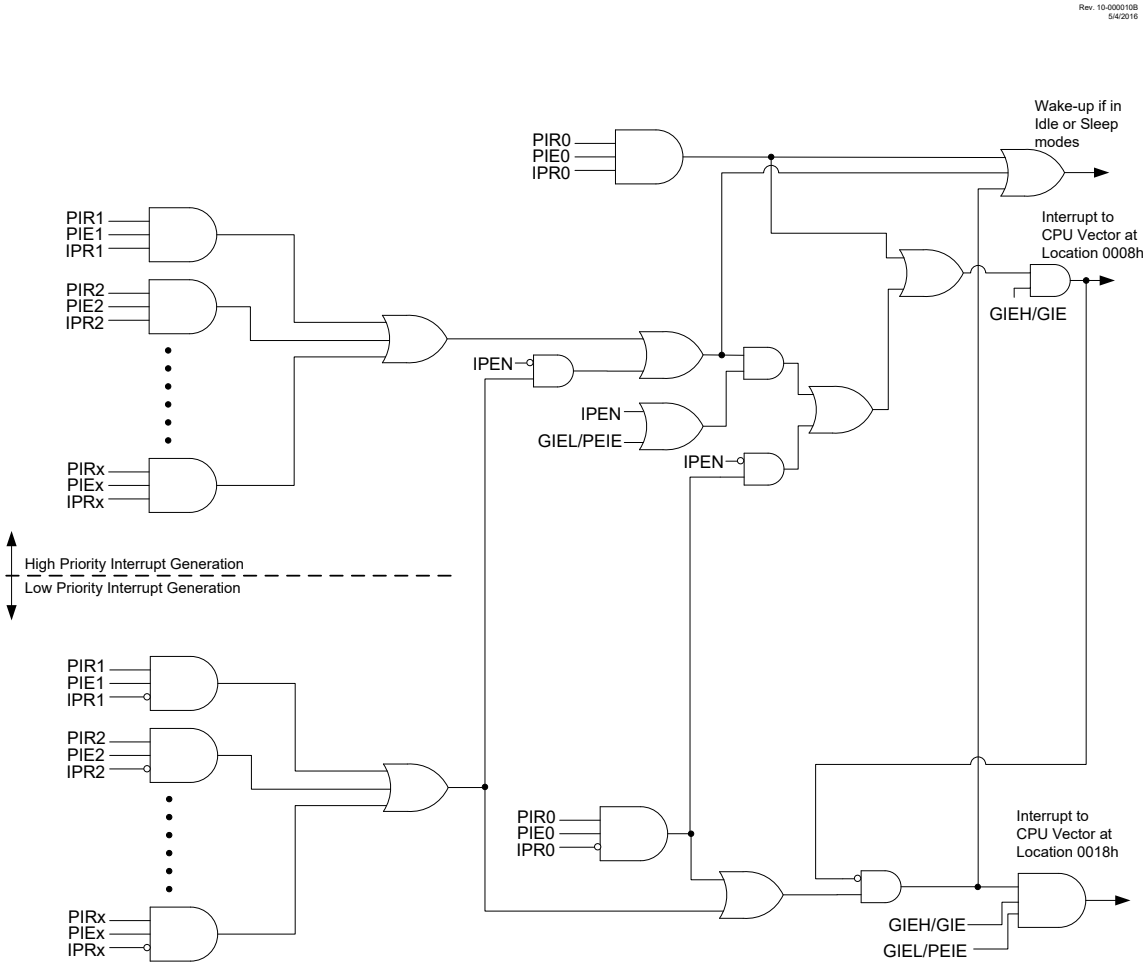
The “return from interrupt” instruction, `RETFIE`, exits the interrupt routine and sets the GIE/GIEH bit (GIEH or GIEL if priority levels are used), which re-enables interrupts.

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.



Important: 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 8 PIR registers.

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 8 Peripheral Interrupt Enable registers. 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 8 Peripheral Interrupt Priority registers. Using the priority bits requires that the Interrupt Priority Enable (IPEN) bit be set.

14.8 INTn Pin Interrupts

PIC18F24/25Q10 devices have 3 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 14.13.1 INTCON register is set (= 1), the interrupt is triggered by a rising edge. It the bit is clear, the trigger is on the falling edge.

All external interrupts (INT0, INT1, INT2) 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 corresponding interrupt priority bit (INT0P, INT1P, INT2P) of the 14.13.18 IPR0 register.

14.9 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. Interrupt priority for Timer0 is determined by the value contained in the interrupt priority bit, TMR0IP. See *"Timer0 Module"* for further details on the Timer0 module.

14.10 Interrupt-on-Change

An input change on any port pins that support IOC sets Flag bit, IOCIF. The interrupt can be enabled/disabled by setting/clearing the enable bit, IOCIE. 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 chapter *"Interrupt-on-Change"*.

Related Links

16. Interrupt-on-Change

14.11 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 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. Saving Status, WREG and BSR Registers in RAM 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
```

Related Links

10.1.2.5 Fast Register Stack

14.12 Register Summary - Interrupt Control

Address	Name	Bit Pos.								
0x0EB5	IPR0	7:0			TMR0IP	IOCIP		INT2IP	INT1IP	INT0IP
0x0EB6	IPR1	7:0	OSCFIP	CSWIP					ADTIP	ADIP
0x0EB7	IPR2	7:0	HLVDIP	ZCDIP					C2IP	C1IP
0x0EB8	IPR3	7:0			RC1IP	TX1IP			BCL1IP	SSP1IP
0x0EB9	IPR4	7:0			TMR6IP	TMR5IP	TMR4IP	TMR3IP	TMR2IP	TMR1IP
0x0EBA	IPR5	7:0						TMR5GIP	TMR3GIP	TMR1GIP
0x0EBB	IPR6	7:0							CCP2IP	CCP1IP
0x0EBC	IPR7	7:0	SCANIP	CRCIP	NVMIP					CWG1IP
0x0EBD	PIE0	7:0			TMR0IE	IOCIE		INT2IE	INT1IE	INT0IE
0x0EBE	PIE1	7:0	OSCFIE	CSWIE					ADTIE	ADIE
0x0EBF	PIE2	7:0	HLVDIE	ZCDIE					C2IE	C1IE
0x0EC0	PIE3	7:0			RC1IE	TX1IE			BCL1IE	SSP1IE
0x0EC1	PIE4	7:0			TMR6IE	TMR5IE	TMR4IE	TMR3IE	TMR2IE	TMR1IE
0x0EC2	PIE5	7:0						TMR5GIE	TMR3GIE	TMR1GIE
0x0EC3	PIE6	7:0							CCP2IE	CCP1IE
0x0EC4	PIE7	7:0	SCANIE	CRCIE	NVMIE					CWG1IE
0x0EC5	PIR0	7:0			TMR0IF	IOCIF		INT2IF	INT1IF	INT0IF
0x0EC6	PIR1	7:0	OSCFIF	CSWIF					ADTIF	ADIF
0x0EC7	PIR2	7:0	HLVDIF	ZCDIF					C2IF	C1IF
0x0EC8	PIR3	7:0			RC1IF	TX1IF			BCL1IF	SSP1IF
0x0EC9	PIR4	7:0			TMR6IF	TMR5IF	TMR4IF	TMR3IF	TMR2IF	TMR1IF
0x0ECA	PIR5	7:0						TMR5GIF	TMR3GIF	TMR1GIF
0x0ECB	PIR6	7:0							CCP2IF	CCP1IF
0x0ECC	PIR7	7:0	SCANIF	CRCIF	NVMIF					CWG1IF
0x0ECD ... 0x0FF1	Reserved									
0x0FF2	INTCON	7:0	GIE/GIEH	PEIE/GIEL	IPEN			INT2EDG	INT1EDG	INT0EDG

14.13 Register Definitions: Interrupt Control

14.13.1 INTCON

Name: INTCON
Address: 0xFF2

Interrupt Control Register

Bit	7	6	5	4	3	2	1	0
	GIE/GIEH	PEIE/GIEL	IPEN			INT2EDG	INT1EDG	INT0EDG
Access	R/W	R/W	R/W			R/W	R/W	R/W
Reset	0	0	0			1	1	1

Bit 7 – GIE/GIEH Global Interrupt Enable bit

Value	Condition	Description
1	If IPEN = 1	Enables all unmasked interrupts and cleared by hardware for high-priority interrupts only
0	If IPEN = 1	Disables all interrupts
1	If IPEN = 0	Enables all unmasked interrupts and cleared by hardware for all interrupts
0	If IPEN = 0	Disables all interrupts

Bit 6 – PEIE/GIEL Peripheral Interrupt Enable bit

Value	Condition	Description
1	If IPEN = 1	Enables all low-priority interrupts and cleared by hardware for low-priority interrupts only
0	If IPEN = 1	Disables all low-priority interrupts
1	If IPEN = 0	Enables all unmasked peripheral interrupts
0	If IPEN = 0	Disables all peripheral interrupts

Bit 5 – IPEN Interrupt Priority Enable bit

Value	Description
1	Enable priority levels on interrupts
0	Disable priority levels on interrupts

Bits 0, 1, 2 – INTxEDG External Interrupt ‘x’ Edge Select bit

Value	Description
1	Interrupt on rising edge of INTx pin
0	Interrupt on falling edge of INTx pin

Important: 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.

14.13.2 PIR0

Name: PIR0
Address: 0xEC5

Peripheral Interrupt Request (Flag) Register 0

Bit	7	6	5	4	3	2	1	0
			TMR0IF	IOCIF		INT2IF	INT1IF	INT0IF
Access			R/W	R		R/W	R/W	R/W
Reset			0	0		0	0	0

Bit 5 – TMR0IF Timer0 Interrupt Flag bit(1)

Value	Description
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)

Value	Description
1	IOC event has occurred (must be cleared by software)
0	IOC event has not occurred

Bits 0, 1, 2 – INTxIF External Interrupt ‘x’ Flag bit(1,3)

Value	Description
1	External Interrupt ‘x’ has occurred
0	External Interrupt ‘x’ has not occurred

Note:

1. Interrupts are not disabled by the PEIE bit.
2. IOCIF is a read-only bit; to clear the interrupt condition, all bits in the IOCF register must be cleared.
3. The external interrupt GPIO pin is selected by the INTPPS register.

14.13.3 PIR1

Name: PIR1
Address: 0xEC6

Peripheral Interrupt Request (Flag) Register 1

Bit	7	6	5	4	3	2	1	0
	OSCFIF	CSWIF					ADTIF	ADIF
Access	R/W	R/W					R/W	R/W
Reset	0	0					0	0

Bit 7 – OSCFIF Oscillator Fail Interrupt Flag bit

Value	Description
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)

Value	Description
1	New oscillator is ready for switch (must be cleared by software)
0	New oscillator is not ready for switch or has not been started

Bit 1 – ADTIF ADC Threshold Interrupt Flag bit

Value	Description
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

Value	Description
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.

Related Links

4.3.3 Clock Switch and Sleep

14.13.4 PIR2

Name: PIR2
Address: 0xEC7

Peripheral Interrupt Request (Flag) Register 2

Bit	7	6	5	4	3	2	1	0
	HLVDIF	ZCDIF					C2IF	C1IF
Access	R/W	R/W					R/W	R/W
Reset	0	0					0	0

Bit 7 – HLVDIF HLVD Interrupt Flag bit

Value	Description
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

Value	Description
1	ZCD Output has changed (must be cleared in software)
0	ZCD Output has not changed

Bits 0, 1 – CxIF Comparator ‘x’ Interrupt Flag bit

Value	Description
1	Comparator Cx output has changed (must be cleared by software)
0	Comparator Cx output has not changed

14.13.5 PIR3

Name: PIR3
Address: 0xEC8

Peripheral Interrupt Request (Flag) Register 3

Bit	7	6	5	4	3	2	1	0
			RC1IF	TX1IF			BCL1IF	SSP1IF
Access			R	R			R/W	R/W
Reset			0	0			0	0

Bit 5 – RCxIF EUSARTx Receive Interrupt Flag bit

Value	Description
1	The EUSARTx receive buffer, RCxREG, is full (cleared by reading RCxREG)
0	The EUSARTx receive buffer is empty

Bit 4 – TXxIF EUSARTx Transmit Interrupt Flag bit

Value	Description
1	The EUSARTx transmit buffer, TXxREG, is empty (cleared by writing TXxREG)
0	The EUSARTx transmit buffer is full

Bit 1 – BCLxIF MSSPx Bus Collision Interrupt Flag bit

Value	Description
1	A bus collision has occurred while the MSSPx module configured in I^2C master was transmitting (must be cleared in software)
0	No bus collision occurred

Bit 0 – SSPxIF Synchronous Serial Port ‘x’ Interrupt Flag bit

Value	Description
1	The transmission/reception is complete (must be cleared in software)
0	Waiting to transmit/receive

14.13.6 PIR4

Name: PIR4
Address: 0xEC9

Peripheral Interrupt Request (Flag) Register 4

Bit	7	6	5	4	3	2	1	0
			TMR6IF	TMR5IF	TMR4IF	TMR3IF	TMR2IF	TMR1IF
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			0	0	0	0	0	0

Bit 5 – TMR6IF TMR6 to PR6 Match Interrupt Flag bit

Value	Description
1	TMR6 to PR6 match occurred (must be cleared in software)
0	No TMR6 to PR6 match occurred

Bit 4 – TMR5IF TMR5 Overflow Interrupt Flag bit

Value	Description
1	TMR5 register overflowed (must be cleared in software)
0	TMR5 register did not overflow

Bit 3 – TMR4IF TMR4 to PR4 Match Interrupt Flag bit

Value	Description
1	TMR4 to PR4 match occurred (must be cleared in software)
0	No TMR4 to PR4 match occurred

Bit 2 – TMR3IF TMR3 Overflow Interrupt Flag bit

Value	Description
1	TMR3 register overflowed (must be cleared in software)
0	TMR3 register did not overflow

Bit 1 – TMR2IF TMR2 to PR2 Match Interrupt Flag bit

Value	Description
1	TMR2 to PR2 match occurred (must be cleared in software)
0	No TMR2 to PR2 match occurred

Bit 0 – TMR1IF TMR1 Overflow Interrupt Flag bit

Value	Description
1	TMR1 register overflowed (must be cleared in software)
0	TMR1 register did not overflow

14.13.7 PIR5

Name: PIR5
Address: 0xECA

Peripheral Interrupt Request (Flag) Register 5

Bit	7	6	5	4	3	2	1	0
						TMR5GIF	TMR3GIF	TMR1GIF
Access						R/W	R/W	R/W
Reset						0	0	0

Bit 2 – TMR5GIF TMR5 Gate Interrupt Flag bit

Value	Description
1	TMR5 gate interrupt occurred (must be cleared in software)
0	No TMR5 gate occurred

Bit 1 – TMR3GIF TMR3 Gate Interrupt Flag bit

Value	Description
1	TMR3 gate interrupt occurred (must be cleared in software)
0	No TMR3 gate occurred

Bit 0 – TMR1GIF TMR1 Gate Interrupt Flag bit

Value	Description
1	TMR1 gate interrupt occurred (must be cleared in software)
0	No TMR1 gate occurred

14.13.8 PIR6

Name: PIR6
Address: 0xECB

PIR6 Peripheral Interrupt Request (Flag) Register 6

Bit	7	6	5	4	3	2	1	0
							CCP2IF	CCP1IF
Access							R/W	R/W
Reset							0	0

Bit 1 – CCP2IF ECCP2 Interrupt Flag bit

Value	Condition	Description
1	Capture mode	A TMR register capture occurred (must be cleared in software)
0	Capture mode	No TMR register capture occurred
1	Compare mode	A TMR register compare match occurred (must be cleared in software)
0	Compare mode	No TMR register compare match occurred
–	PWM mode	Unused in PWM mode.

Bit 0 – CCP1IF ECCP1 Interrupt Flag bit

Value	Condition	Description
1	Capture mode	A TMR register capture occurred (must be cleared in software)
0	Capture mode	No TMR register capture occurred
1	Compare mode	A TMR register compare match occurred (must be cleared in software)
0	Compare mode	No TMR register compare match occurred
–	PWM mode	Unused in PWM mode.

14.13.9 PIR7

Name: PIR7
Address: 0xECC

Peripheral Interrupt Request (Flag) Register 7

Bit	7	6	5	4	3	2	1	0
	SCANIF	CRCIF	NVMIF					CWG1IF
Access	R/W	R/W	R/W					R/W
Reset	0	0	0					0

Bit 7 – SCANIF SCAN Interrupt Flag bit

Value	Description
1	SCAN interrupt has occurred (must be cleared in software)
0	SCAN interrupt has not occurred or has not been started

Bit 6 – CRCIF CRC Interrupt Flag bit

Value	Description
1	CRC interrupt has occurred (must be cleared in software)
0	CRC interrupt has not occurred or has not been started

Bit 5 – NVMIF NVM Interrupt Flag bit

Value	Description
1	NVM interrupt has occurred (must be cleared in software)
0	NVM interrupt has not occurred or has not been started

Bit 0 – CWG1IF CWG Interrupt Flag bit

Value	Description
1	CWG interrupt has occurred (must be cleared in software)
0	CWG interrupt has not occurred or has not been started

14.13.10 PIE0

Name: PIE0
Address: 0xEBD

Peripheral Interrupt Enable Register 0

Bit	7	6	5	4	3	2	1	0
			TMR0IE	IOCIE		INT2IE	INT1IE	INT0IE
Access			R/W	R/W		R/W	R/W	R/W
Reset			0	0		0	0	0

Bit 5 – TMR0IE Timer0 Interrupt Enable bit(1)

Value	Description
1	Enabled
0	Disabled

Bit 4 – IOCIE Interrupt-on-Change Enable bit(1)

Value	Description
1	Enabled
0	Disabled

Bits 0, 1, 2 – INTxIE External Interrupt ‘x’ Enable bit(1)

Value	Description
1	Enabled
0	Disabled

Note:

1. PIR0 interrupts are not disabled by the PEIE bit in the INTCON register.

14.13.11 PIE1

Name: PIE1
Address: 0xEBE

Peripheral Interrupt Enable Register 1

Bit	7	6	5	4	3	2	1	0
	OSCFIE	CSWIE					ADTIE	ADIE
Access	R/W	R/W					R/W	R/W
Reset	0	0					0	0

Bit 7 – OSCFIE Oscillator Fail Interrupt Enable bit

Value	Description
1	Enabled
0	Disabled

Bit 6 – CSWIE Clock-Switch Interrupt Enable bit

Value	Description
1	Enabled
0	Disabled

Bit 1 – ADTIE ADC Threshold Interrupt Enable bit

Value	Description
1	Enabled
0	Disabled

Bit 0 – ADIE ADC Interrupt Enable bit

Value	Description
1	Enabled
0	Disabled

14.13.12 PIE2

Name: PIE2
Address: 0xEBF

Peripheral Interrupt Enable Register 2

Bit	7	6	5	4	3	2	1	0
	HLVDIE	ZCDIE					C2IE	C1IE
Access	R/W	R/W					R/W	R/W
Reset	0	0					0	0

Bit 7 – HLVDIE HLVD Interrupt Enable bit

Value	Description
1	Enabled
0	Disabled

Bit 6 – ZCDIE Zero-Cross Detect Interrupt Enable bit

Value	Description
1	Enabled
0	Disabled

Bits 0, 1 – CxIE Comparator ‘x’ Interrupt Enable bit

Value	Description
1	Enabled
0	Disabled

14.13.13 PIE3

Name: PIE3
Address: 0xEC0

Peripheral Interrupt Enable Register 3

Bit	7	6	5	4	3	2	1	0
			RC1IE	TX1IE			BCL1IE	SSP1IE
Access			R/W	R/W			R/W	R/W
Reset			0	0			0	0

Bit 5 – RCxIE EUSARTx Receive Interrupt Enable bit

Value	Description
1	Enabled
0	Disabled

Bit 4 – TXxIE EUSARTx Transmit Interrupt Enable bit

Value	Description
1	Enabled
0	Disabled

Bit 1 – BCLxIE MSSPx Bus Collision Interrupt Enable bit

Value	Description
1	Enabled
0	Disabled

Bit 0 – SSPxIE Synchronous Serial Port ‘x’ Interrupt Enable bit

Value	Description
1	Enabled
0	Disabled

14.13.14 PIE4

Name: PIE4
Address: 0xEC1

Peripheral Interrupt Enable Register 4

Bit	7	6	5	4	3	2	1	0
			TMR6IE	TMR5IE	TMR4IE	TMR3IE	TMR2IE	TMR1IE
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			0	0	0	0	0	0

Bit 5 – TMR6IE TMR6 to PR6 Match Interrupt Enable bit

Value	Description
1	Enabled
0	Disabled

Bit 4 – TMR5IE TMR5 Overflow Interrupt Enable bit

Value	Description
1	Enabled
0	Disabled

Bit 3 – TMR4IE TMR4 to PR4 Match Interrupt Enable bit

Value	Description
1	Enabled
0	Disabled

Bit 2 – TMR3IE TMR3 Overflow Interrupt Enable bit

Value	Description
1	Enabled
0	Disabled

Bit 1 – TMR2IE TMR2 to PR2 Match Interrupt Enable bit

Value	Description
1	Enabled
0	Disabled

Bit 0 – TMR1IE TMR1 Overflow Interrupt Enable bit

Value	Description
1	Enabled
0	Disabled

14.13.15 PIE5

Name: PIE5
Address: 0xEC2

Peripheral Interrupt Enable Register 5

Bit	7	6	5	4	3	2	1	0
						TMR5GIE	TMR3GIE	TMR1GIE
Access						R/W	R/W	R/W
Reset						0	0	0

Bit 2 – TMR5GIE TMR5 Gate Interrupt Enable bit

Value	Description
1	Enabled
0	Disabled

Bit 1 – TMR3GIE TMR3 Gate Interrupt Enable bit

Value	Description
1	Enabled
0	Disabled

Bit 0 – TMR1GIE TMR1 Gate Interrupt Enable bit

Value	Description
1	Enabled
0	Disabled

14.13.16 PIE6

Name: PIE6
Address: 0xEC3

Peripheral Interrupt Enable Register 6

Bit	7	6	5	4	3	2	1	0
							CCP2IE	CCP1IE
Access							R/W	R/W
Reset							0	0

Bit 1 – CCP2IE ECCP2 Interrupt Enable bit

Value	Description
1	Enabled
0	Disabled

Bit 0 – CCP1IE ECCP1 Interrupt Enable bit

Value	Description
1	Enabled
0	Disabled

14.13.17 PIE7

Name: PIE7
Address: 0xEC4

Peripheral Interrupt Enable Register 7

Bit	7	6	5	4	3	2	1	0
	SCANIE	CRCIE	NVMIE					CWG1IE
Access	R/W	R/W	R/W					R/W
Reset	0	0	0					0

Bit 7 – SCANIE SCAN Interrupt Enable bit

Value	Description
1	Enabled
0	Disabled

Bit 6 – CRCIE CRC Interrupt Enable bit

Value	Description
1	Enabled
0	Disabled

Bit 5 – NVMIE NVM Interrupt Enable bit

Value	Description
1	Enabled
0	Disabled

Bit 0 – CWG1IE CWG Interrupt Enable bit

Value	Description
1	Enabled
0	Disabled

14.13.18 IPR0

Name: IPR0
Address: 0xEB5

Peripheral Interrupt Priority Register 0

Bit	7	6	5	4	3	2	1	0
			TMR0IP	IOCIP		INT2IP	INT1IP	INT0IP
Access			R/W	R/W		R/W	R/W	R/W
Reset			1	1		0	0	1

Bit 5 – TMR0IP Timer0 Interrupt Priority bit

Value	Description
1	High priority
0	Low priority

Bit 4 – IOCIP Interrupt-on-Change Priority bit

Value	Description
1	High priority
0	Low priority

Bits 0, 1, 2 – INTxIP External Interrupt ‘x’ Priority bit

Value	Description
1	High priority
0	Low priority

14.13.19 IPR1

Name: IPR1
Address: 0xEB6

Peripheral Interrupt Priority Register 1

Bit	7	6	5	4	3	2	1	0
	OSCFIP	CSWIP					ADTIP	ADIP
Access	R/W	R/W					R/W	R/W
Reset	1	1					1	1

Bit 7 – OSCFIP Oscillator Fail Interrupt Priority bit

Value	Description
1	High priority
0	Low priority

Bit 6 – CSWIP Clock-Switch Interrupt Priority bit

Value	Description
1	High priority
0	Low priority

Bit 1 – ADTIP ADC Threshold Interrupt Priority bit

Value	Description
1	High priority
0	Low priority

Bit 0 – ADIP ADC Interrupt Priority bit

Value	Description
1	High priority
0	Low priority

14.13.20 IPR2

Name: IPR2
Address: 0xEB7

Peripheral Interrupt Priority Register 2

Bit	7	6	5	4	3	2	1	0
	HLVDIP	ZCDIP					C2IP	C1IP
Access	R/W	R/W					R/W	R/W
Reset	1	1					1	1

Bit 7 – HLVDIP HLVD Interrupt Priority bit

Value	Description
1	High priority
0	Low priority

Bit 6 – ZCDIP Zero-Cross Detect Interrupt Priority bit

Value	Description
1	High priority
0	Low priority

Bits 0, 1 – CxIP Comparator ‘x’ Interrupt Priority bit

Value	Description
1	High priority
0	Low priority

14.13.21 IPR3

Name: IPR3
Address: 0xEB8

Peripheral Interrupt Priority Register 3

Bit	7	6	5	4	3	2	1	0
			RC1IP	TX1IP			BCL1IP	SSP1IP
Access			R/W	R/W			R/W	R/W
Reset			1	1			1	1

Bit 5 – RCxIP EUSARTx Receive Interrupt Priority bit

Value	Description
1	High priority
0	Low priority

Bit 4 – TXxIP EUSARTx Transmit Interrupt Priority bit

Value	Description
1	High priority
0	Low priority

Bit 1 – BCLxIP MSSPx Bus Collision Interrupt Priority bit

Value	Description
1	High priority
0	Low priority

Bit 0 – SSPxIP Synchronous Serial Port 'x' Interrupt Priority bit

Value	Description
1	High priority
0	Low priority

14.13.22 IPR4

Name: IPR4
Address: 0xEB9

Peripheral Interrupt Priority Register 4

Bit	7	6	5	4	3	2	1	0
			TMR6IP	TMR5IP	TMR4IP	TMR3IP	TMR2IP	TMR1IP
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			1	1	1	1	1	1

Bit 5 – TMR6IP TMR6 to PR6 Match Interrupt Priority bit

Value	Description
1	High priority
0	Low priority

Bit 4 – TMR5IP TMR5 Overflow Interrupt Priority bit

Value	Description
1	High priority
0	Low priority

Bit 3 – TMR4IP TMR4 to PR4 Match Interrupt Priority bit

Value	Description
1	High priority
0	Low priority

Bit 2 – TMR3IP TMR3 Overflow Interrupt Priority bit

Value	Description
1	High priority
0	Low priority

Bit 1 – TMR2IP TMR2 to PR2 Match Interrupt Priority bit

Value	Description
1	High priority
0	Low priority

Bit 0 – TMR1IP TMR1 Overflow Interrupt Priority bit

Value	Description
1	High priority
0	Low priority

14.13.23 IPR5

Name: IPR5
Address: 0xEBA

Peripheral Interrupt Priority Register 5

Bit	7	6	5	4	3	2	1	0
						TMR5GIP	TMR3GIP	TMR1GIP
Access						R/W	R/W	R/W
Reset						1	1	1

Bit 2 – TMR5GIP TMR5 Gate Interrupt Priority bit

Value	Description
1	High priority
0	Low priority

Bit 1 – TMR3GIP TMR3 Gate Interrupt Priority bit

Value	Description
1	High priority
0	Low priority

Bit 0 – TMR1GIP TMR1 Gate Interrupt Priority bit

Value	Description
1	High priority
0	Low priority

14.13.24 IPR6

Name: IPR6
Address: 0xEBB

Peripheral Interrupt Priority Register

Bit	7	6	5	4	3	2	1	0
							CCP2IP	CCP1IP
Access							R/W	R/W
Reset							1	1

Bit 1 – CCP2IP ECCP2 Interrupt Priority bit

Value	Description
1	High priority
0	Low priority

Bit 0 – CCP1IP ECCP1 Interrupt Priority bit

Value	Description
1	High priority
0	Low priority

14.13.25 IPR7

Name: IPR7
Address: 0xEBC

Peripheral Interrupt Priority Register

Bit	7	6	5	4	3	2	1	0
	SCANIP	CRCIP	NVMIP					CWG1IP
Access	R/W	R/W	R/W					R/W
Reset	1	1	1					1

Bit 7 – SCANIP SCAN Interrupt Priority bit

Value	Description
1	High priority
0	Low priority

Bit 6 – CRCIP CRC Interrupt Priority bit

Value	Description
1	High priority
0	Low priority

Bit 5 – NVMIP NVM Interrupt Priority bit

Value	Description
1	High priority
0	Low priority

Bit 0 – CWG1IP CWG Interrupt Priority bit

Value	Description
1	High priority
0	Low priority

15. I/O Ports

Table 15-1. Port Availability per Device

Device	PORTA	PORTB	PORTC	PORTD	PORTE
PIC18F2xQ10	●	●	●		●

Each port has eight registers to control the operation. These registers are:

- PORTx registers (reads the levels on the pins of the device)
- LATx registers (output latch)
- TRISx registers (data direction)
- ANSELx registers (analog select)
- WPUx registers (weak pull-up)
- INLVLx (input level control)
- SLRCONx registers (slew rate control)
- ODCONx registers (open-drain control)

Most port pins share functions with device peripherals, both analog and digital. In general, when a peripheral is enabled on a port pin, that pin cannot be used as a general purpose output; however, the pin can still be read.

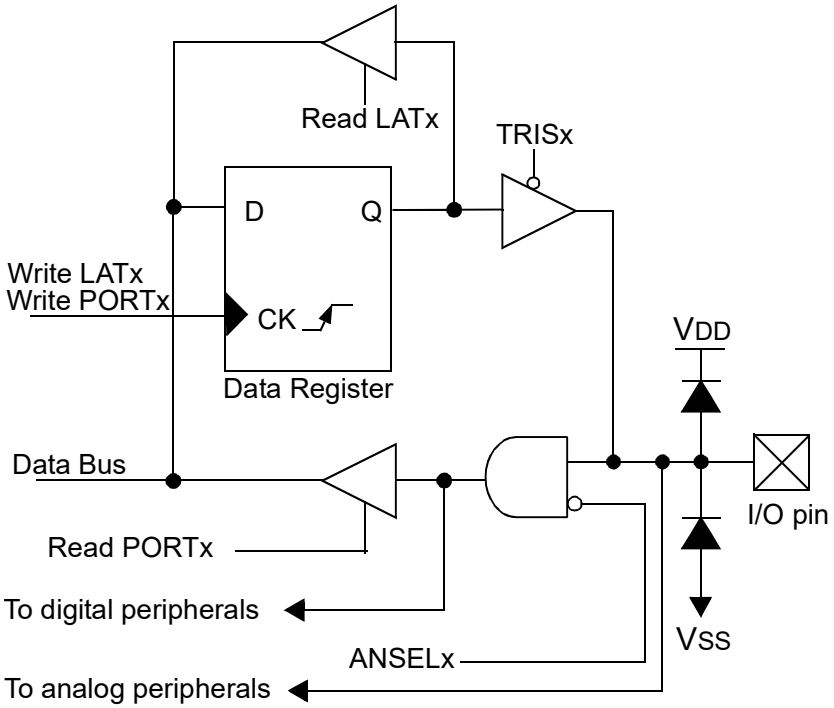
The Data Latch (LATx registers) is useful for read-modify-write operations on the value that the I/O pins are driving.

A write operation to the LATx register has the same effect as a write to the corresponding PORTx register. A read of the LATx register reads of the values held in the I/O PORT latches, while a read of the PORTx register reads the actual I/O pin value.

Ports that support analog inputs have an associated ANSELx register. When an ANSELx bit is set, the digital input buffer associated with that bit is disabled.

Disabling the input buffer prevents analog signal levels on the pin between a logic high and low from causing excessive current in the logic input circuitry. A simplified model of a generic I/O port, without the interfaces to other peripherals, is shown in the following figure:

Figure 15-1. Generic I/O Port Operation



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 *"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

Related Links

17. (PPS) Peripheral Pin Select Module

15.2 PORTx Registers

In this section the generic names such as PORTx, LATx, TRISx, etc. can be associated with all ports. For availability of PORTD refer to Table 15-1. The functionality of PORTE is different compared to other ports and is explained in a separate section.

15.2.1 Data Register

PORTx is an 8-bit wide, bidirectional port. The corresponding data direction register is TRISx. Setting a TRISx bit (‘1’) will make the corresponding PORTA 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 PORTA.

Reading the PORTx register reads the status of the pins, whereas writing to it will write to the PORT latch. All write operations are read-modify-write operations. Therefore, a write to a port implies that the port pins are read, this value is modified and then written to the PORT data latch (LATx).

The PORT data latch LATx holds the output port data and contains the latest value of a LATx or PORTx write.

Example 15-1. EXAMPLE-1: Initializing PORTA

```
; This code example illustrates initializing the PORTA register.
;  The other ports are initialized in the same manner.
        CLRF    LATA            ; Set all output bits to zero
        MOVLW   B'11111000'     ; Set RA<7:3> as inputs and RA<2:0> as outputs
        MOVWF   TRISA           ;
        BANKSEL ANSELA
        CLRF    ANSELA          ; All pins are digital I/O
```

15.2.2 Direction Control

The TRISx register 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’.

Related Links

15.5.5 TRISA
15.5.6 TRISB
15.5.7 TRISC

15.2.3 Analog Control

The ANSELx register 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.

Important: The ANSELx bits default to the Analog mode after Reset. To use any pins as digital general purpose or peripheral inputs, the corresponding ANSEL bits must be initialized to ‘0’ by user software.

Related Links

15.5.11 ANSELA
15.5.12 ANSELB

15.2.4 Open-Drain Control

The ODCONx register 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.

Important: It is not necessary to set open-drain control when using the pin for I^2C; the I^2C module controls the pin and makes the pin open-drain.

Related Links

15.5.18 ODCONA
15.5.19 ODCONB
15.5.20 ODCONC

15.2.5 Slew Rate Control

The SLRCONx register controls the slew rate option for each port pin. Slew rate for each port pin can be controlled independently. When 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.

Related Links

15.5.21 SLRCONA
15.5.22 SLRCONB
15.5.23 SLRCONC

15.2.6 Input Threshold Control

The INLVLx register controls the input voltage threshold for each of the available PORTx input pins. A selection between the Schmitt Trigger CMOS or the TTL compatible thresholds is available. The input threshold is important in determining the value of a read of the PORTx register and also the level at which an interrupt-on-change occurs, if that feature is enabled. See link below for more information on threshold levels.

Important: 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.

Related Links

15.5.24 INLVLA
15.5.25 INLVLB
15.5.26 INLVLC
15.5.27 INLVLE

15.2.7 Weak Pull-up Control

The WPUx register controls the individual weak pull-ups for each port pin.

Related Links

15.5.14 WPUA
15.5.15 WPUB
15.5.16 WPUC
15.5.17 WPUE

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. Individual pins can be independently configured to generate an interrupt. The interrupt-on-change module is present on all the pins . For further details about the IOC module, refer to "*Interrupt-on-Change*" chapter.

Related Links

16. Interrupt-on-Change

15.3 PORTE Registers

Example 15-2. EXAMPLE-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
```

15.3.1 PORTE on 28-Pin Devices

For 28-pin devices, PORTE is only available when Master Clear functionality is disabled (MCLRE = 0). In this case, PORTE is a single bit, input-only port comprised of RE3 only. The pin operates as previously described. RE3 in PORTE register is a read-only bit and will read '1' when MCLRE = 1 (i.e., Master Clear enabled).

15.3.2 RE3 Weak Pull-Up

The port RE3 pin has an individually controlled weak internal pull-up. When set, the WPUE3 bit enables the RE3 pin pull-up. When the RE3 port pin is configured as $\overline{\text{MCLR}}$, (CONFIG2L, MCLRE = 1 and CONFIG4H, LVP = 0), or configured for Low-Voltage Programming, (MCLRE = x and LVP = 1), the pull-up is always enabled and the WPUE3 bit has no effect.

15.3.3 PORTE Interrupt-on-Change

The interrupt-on-change feature is available only on the RE3 pin for all devices.

Related Links

16. Interrupt-on-Change

15.4 Register Summary - Input/Output

Address	Name	Bit Pos.								
0x0F08	INLVLA	7:0	INLVLA7	INLVLA6	INLVLA5	INLVLA4	INLVLA3	INLVLA2	INLVLA1	INLVLA0
0x0F09	SLRCONA	7:0	SLRA7	SLRA6	SLRA5	SLRA4	SLRA3	SLRA2	SLRA1	SLRA0
0x0F0A	ODCONA	7:0	ODCA7	ODCA6	ODCA5	ODCA4	ODCA3	ODCA2	ODCA1	ODCA0
0x0F0B	WPUA	7:0	WPUA7	WPUA6	WPUA5	WPUA4	WPUA3	WPUA2	WPUA1	WPUA0
0x0F0C	ANSELA	7:0	ANSELA7	ANSELA6	ANSELA5	ANSELA4	ANSELA3	ANSELA2	ANSELA1	ANSELA0
0x0F0D ... 0x0F0F	Reserved									
0x0F10	INLVLB	7:0	INLVLB7	INLVLB6	INLVLB5	INLVLB4	INLVLB3	INLVLB2	INLVLB1	INLVLB0
0x0F11	SLRCONB	7:0	SLRB7	SLRB6	SLRB5	SLRB4	SLRB3	SLRB2	SLRB1	SLRB0
0x0F12	ODCONB	7:0	ODCB7	ODCB6	ODCB5	ODCB4	ODCB3	ODCB2	ODCB1	ODCB0
0x0F13	WPUB	7:0	WPUB7	WPUB6	WPUB5	WPUB4	WPUB3	WPUB2	WPUB1	WPUB0
0x0F14	ANSELB	7:0	ANSELB7	ANSELB6	ANSELB5	ANSELB4	ANSELB3	ANSELB2	ANSELB1	ANSELB0
0x0F15 ... 0x0F17	Reserved									
0x0F18	INLVLC	7:0	INLVLC7	INLVLC6	INLVLC5	INLVLC4	INLVLC3	INLVLC2	INLVLC1	INLVLC0
0x0F19	SLRCONC	7:0	SLRC7	SLRC6	SLRC5	SLRC4	SLRC3	SLRC2	SLRC1	SLRC0
0x0F1A	ODCONC	7:0	ODCC7	ODCC6	ODCC5	ODCC4	ODCC3	ODCC2	ODCC1	ODCC0
0x0F1B	WPUC	7:0	WPUC7	WPUC6	WPUC5	WPUC4	WPUC3	WPUC2	WPUC1	WPUC0
0x0F1C	ANSELC	7:0	ANSELC7	ANSELC6	ANSELC5	ANSELC4	ANSELC3	ANSELC2	ANSELC1	ANSELC0
0x0F1D ... 0x0F24	Reserved									
0x0F25	INLVLE	7:0					INLVLE3			
0x0F26 ... 0x0F27	Reserved									
0x0F28	WPUE	7:0					WPUE3			
0x0F29 ... 0x0F81	Reserved									
0x0F82	LATA	7:0	LATA7	LATA6	LATA5	LATA4	LATA3	LATA2	LATA1	LATA0
0x0F83	LATB	7:0	LATB7	LATB6	LATB5	LATB4	LATB3	LATB2	LATB1	LATB0
0x0F84	LATC	7:0	LATC7	LATC6	LATC5	LATC4	LATC3	LATC2	LATC1	LATC0
0x0F85 ... 0x0F86	Reserved									
0x0F87	TRISA	7:0	TRISA7	TRISA6	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0
0x0F88	TRISB	7:0	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0
0x0F89	TRISC	7:0	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0
0x0F8A ... 0x0F8B	Reserved									

Address	Name	Bit Pos.								
0x0F8C	PORTA	7:0	RA7	RA6	RA5	RA4	RA3	RA2	RA1	RA0
0x0F8D	PORTB	7:0	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0
0x0F8E	PORTC	7:0	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0
0x0F8F	Reserved									
0x0F90	PORTE	7:0					RE3			

15.5 Register Definitions: Port Control

15.5.1 PORTA

Name: PORTA
Address: 0xF8C

PORTA Register
Note: Writes to PORTA are actually written to the corresponding LATA register.
Reads from PORTA register return actual I/O pin values.

Bit	7	6	5	4	3	2	1	0
	RA7	RA6	RA5	RA4	RA3	RA2	RA1	RA0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	x	x	x	x	x	x	x	x

Bits 0, 1, 2, 3, 4, 5, 6, 7 – RAn Port I/O Value bits
Reset States: POR/BOR = xxxxxxxx
All Other Resets = uuuuuuuu

Value	Description
1	Port pin is ≥ V_{IH}
0	Port pin is ≤ V_{IL}

15.5.2 PORTB

Name: PORTB
Address: 0xF8D

PORTB Register

Bit	7	6	5	4	3	2	1	0
	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	x	x	x	x	x	x	x	x

Bits 0, 1, 2, 3, 4, 5, 6, 7 – RBn Port I/O Value bits
Note: Bits RB6 and RB7 read '1' while in Debug mode.

Reset States: POR/BOR = xxxxxxxx
All Other Resets = uuuuuuuu

Value	Description
1	Port pin is ≥ V_{IH}
0	Port pin is ≤ V_{IL}

Note: Writes to PORTB are actually written to the corresponding LATB register.
Reads from PORTB register return actual I/O pin values.

15.5.3 PORTC

Name: PORTC
Address: 0xF8E

PORTC Register

Bit	7	6	5	4	3	2	1	0
	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	x	x	x	x	x	x	x	x

Bits 0, 1, 2, 3, 4, 5, 6, 7 – RCn Port I/O Value bits
Reset States: POR/BOR = xxxxxxxx
All Other Resets = uuuuuuuu

Value	Description
1	Port pin is ≥ V_{IH}
0	Port pin is ≤ V_{IL}

Note: Writes to PORTC are actually written to the corresponding LATC register. Reads from PORTC register return actual I/O pin values.

15.5.4 PORTE

Name: PORTE
Address: 0xF90

PORTE Register
Note: Writes to PORTE are actually written to the corresponding LATE register.
Reads from PORTE register return actual I/O pin values.

Bit	7	6	5	4	3	2	1	0
					RE3			
Access					R/W			
Reset					x			

Bit 3 – RE3 Port I/O Value bits
Note: Bit RE3 is read-only, and will read ‘1’ when MCLRE = 1 (Master Clear enabled).

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

Value	Description
1	Port pin is ≥ V_{IH}
0	Port pin is ≤ V_{IL}

15.5.5 TRISA

Name: TRISA
Address: 0xF87

Tri-State Control Register

Bit	7	6	5	4	3	2	1	0
	TRISA7	TRISA6	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1

Bits 0, 1, 2, 3, 4, 5, 6, 7 – TRISAn TRISA Port I/O Tri-state Control bits

Value	Description
1	Port output driver is disabled
0	Port output driver is enabled

15.5.6 TRISB

Name: TRISB
Address: 0xF88

Tri-State Control Register

Bit	7	6	5	4	3	2	1	0
	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1

Bits 0, 1, 2, 3, 4, 5, 6, 7 – TRISBn TRISB Port I/O Tri-state Control bits
Note: Bits TRISB6 and TRISB7 read ‘1’ while in Debug mode.

Value	Description
1	Port output driver is disabled
0	Port output driver is enabled

15.5.7 TRISC

Name: TRISC
Address: 0xF89

Tri-State Control Register

Bit	7	6	5	4	3	2	1	0
	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1

Bits 0, 1, 2, 3, 4, 5, 6, 7 – TRISCn TRISC Port I/O Tri-state Control bits

Value	Description
1	Port output driver is disabled
0	Port output driver is enabled

15.5.8 LATA

Name: LATA
Address: 0xF82

Output Latch Register

Bit	7	6	5	4	3	2	1	0
	LATA7	LATA6	LATA5	LATA4	LATA3	LATA2	LATA1	LATA0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	x	x	x	x	x	x	x	x

Bits 0, 1, 2, 3, 4, 5, 6, 7 – LATAn Output Latch A Value bits
Reset States: POR/BOR = xxxxxxxx
All Other Resets = uuuuuuuuu

Note: Writes to LATA are equivalent with writes to the corresponding PORTA register. Reads from LATA register return register values, not I/O pin values.

15.5.9 LATB

Name: LATB
Address: 0xF83

Output Latch Register

Bit	7	6	5	4	3	2	1	0
	LATB7	LATB6	LATB5	LATB4	LATB3	LATB2	LATB1	LATB0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	x	x	x	x	x	x	x	x

Bits 0, 1, 2, 3, 4, 5, 6, 7 – LATBn Output Latch B Value bits
Reset States: POR/BOR = xxxxxxxx
All Other Resets = uuuuuuuuu

Note: Writes to LATB are equivalent with writes to the corresponding PORTB register. Reads from LATB register return register values, not I/O pin values.

15.5.10 LATC

Name: LATC
Address: 0xF84

Output Latch Register

Bit	7	6	5	4	3	2	1	0
	LATC7	LATC6	LATC5	LATC4	LATC3	LATC2	LATC1	LATC0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	x	x	x	x	x	x	x	x

Bits 0, 1, 2, 3, 4, 5, 6, 7 – LATCn Output Latch C Value bits
Reset States: POR/BOR = xxxxxxxx
All Other Resets = uuuuuuuuu

Note: Writes to LATC are equivalent with writes to the corresponding PORTC register. Reads from LATC register return register values, not I/O pin values.

15.5.11 ANSELA

Name: ANSELA
Address: 0xF0C

Analog Select Register

Bit	7	6	5	4	3	2	1	0
	ANSELA7	ANSELA6	ANSELA5	ANSELA4	ANSELA3	ANSELA2	ANSELA1	ANSELA0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1

Bits 0, 1, 2, 3, 4, 5, 6, 7 – ANSELAn Analog Select on Pins RA<7:0>

Value	Description
1	Digital Input buffers are disabled
0	ST and TTL input buffers are enabled

15.5.12 ANSELB

Name: ANSELB
Address: 0xF14
Reset: 0x00

Analog Select Register

Bit	7	6	5	4	3	2	1	0
	ANSELB7	ANSELB6	ANSELB5	ANSELB4	ANSELB3	ANSELB2	ANSELB1	ANSELB0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1

Bits 0, 1, 2, 3, 4, 5, 6, 7 – ANSELBn Analog Select on Pins RB<7:0>

Value	Description
1	Digital Input buffers are disabled
0	ST and TTL input buffers are enabled

15.5.13 ANSELC

Name: ANSELC
Address: 0xF1C

Analog Select Register

Bit	7	6	5	4	3	2	1	0
	ANSELC7	ANSELC6	ANSELC5	ANSELC4	ANSELC3	ANSELC2	ANSELC1	ANSELC0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1

Bits 0, 1, 2, 3, 4, 5, 6, 7 – ANSELCn Analog Select on Pins RC<7:0>

Value	Description
1	Digital Input buffers are disabled
0	ST and TTL input buffers are enabled

15.5.14 WPUA

Name: WPUA
Address: 0xF0B

Weak Pull-up Register

Bit	7	6	5	4	3	2	1	0
	WPUA7	WPUA6	WPUA5	WPUA4	WPUA3	WPUA2	WPUA1	WPUA0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 0, 1, 2, 3, 4, 5, 6, 7 – WPUAn Weak Pull-up PORTA Control bits

Value	Description
1	Weak Pull-up enabled
0	Weak Pull-up disabled

15.5.15 WPUB

Name: WPUB
Address: 0xF13

Weak Pull-up Register

Bit	7	6	5	4	3	2	1	0
	WPUB7	WPUB6	WPUB5	WPUB4	WPUB3	WPUB2	WPUB1	WPUB0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 0, 1, 2, 3, 4, 5, 6, 7 – WPUBn Weak Pull-up PORTA Control bits

Value	Description
1	Weak Pull-up enabled
0	Weak Pull-up disabled

15.5.16 WPUC

Name: WPUC
Address: 0xF1B

Weak Pull-up Register

Bit	7	6	5	4	3	2	1	0
	WPUC7	WPUC6	WPUC5	WPUC4	WPUC3	WPUC2	WPUC1	WPUC0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 0, 1, 2, 3, 4, 5, 6, 7 – WPUCn Weak Pull-up PORTC Control bits

Value	Description
1	Weak Pull-up enabled
0	Weak Pull-up disabled

15.5.17 WPUE

Name: WPUE
Address: 0xF28

Weak Pull-up Register

Bit	7	6	5	4	3	2	1	0
					WPUE3			
Access					R/W			
Reset					0			

Bit 3 – WPUE3 Weak Pull-up PORTE Control bits
Note: If MCLRE = 1, the weak pull-up in RE3 is always enabled; bit WPUE3 is not affected.

Value	Description
1	Weak Pull-up enabled
0	Weak Pull-up disabled

15.5.18 ODCONA

Name: ODCONA
Address: 0xF0A

Open-Drain Control Register

Bit	7	6	5	4	3	2	1	0
	ODCA7	ODCA6	ODCA5	ODCA4	ODCA3	ODCA2	ODCA1	ODCA0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 0, 1, 2, 3, 4, 5, 6, 7 – ODCAn Open-Drain Configuration on Pins Rx<7:0>

Value	Description
1	Output drives only low-going signals (sink current only)
0	Output drives both high-going and low-going signals (source and sink current)

15.5.19 ODCONB

Name: ODCONB
Address: 0xF12

Open-Drain Control Register

Bit	7	6	5	4	3	2	1	0
	ODCB7	ODCB6	ODCB5	ODCB4	ODCB3	ODCB2	ODCB1	ODCB0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 0, 1, 2, 3, 4, 5, 6, 7 – ODCBn Open-Drain Configuration on Pins Rx<7:0>

Value	Description
1	Output drives only low-going signals (sink current only)
0	Output drives both high-going and low-going signals (source and sink current)

15.5.20 ODCONC

Name: ODCONC
Address: 0xF1A

Open-Drain Control Register

Bit	7	6	5	4	3	2	1	0
	ODCC7	ODCC6	ODCC5	ODCC4	ODCC3	ODCC2	ODCC1	ODCC0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 0, 1, 2, 3, 4, 5, 6, 7 – ODCCn Open-Drain Configuration on Pins Rx<7:0>

Value	Description
1	Output drives only low-going signals (sink current only)
0	Output drives both high-going and low-going signals (source and sink current)

15.5.21 SLRCONA

Name: SLRCONA
Address: 0xF09

Slew Rate Control Register

Bit	7	6	5	4	3	2	1	0
	SLRA7	SLRA6	SLRA5	SLRA4	SLRA3	SLRA2	SLRA1	SLRA0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1

Bits 0, 1, 2, 3, 4, 5, 6, 7 – SLRAn Slew Rate Control on Pins Rx<7:0>, respectively

Value	Description
1	Port pin slew rate is limited
0	Port pin slews at maximum rate

15.5.22 SLRCONB

Name: SLRCONB
Address: 0xF11

Slew Rate Control Register

Bit	7	6	5	4	3	2	1	0
	SLRB7	SLRB6	SLRB5	SLRB4	SLRB3	SLRB2	SLRB1	SLRB0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1

Bits 0, 1, 2, 3, 4, 5, 6, 7 – SLRBn Slew Rate Control on Pins Rx<7:0>, respectively

Value	Description
1	Port pin slew rate is limited
0	Port pin slews at maximum rate

15.5.23 SLRCONC

Name: SLRCONC
Address: 0xF19

Slew Rate Control Register

Bit	7	6	5	4	3	2	1	0
	SLRC7	SLRC6	SLRC5	SLRC4	SLRC3	SLRC2	SLRC1	SLRC0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1

Bits 0, 1, 2, 3, 4, 5, 6, 7 – SLRCn Slew Rate Control on Pins Rx<7:0>, respectively

Value	Description
1	Port pin slew rate is limited
0	Port pin slews at maximum rate

15.5.24 INLVLA

Name: INLVLA
Address: 0xF08

Input Level Control Register

Bit	7	6	5	4	3	2	1	0
	INLVLA7	INLVLA6	INLVLA5	INLVLA4	INLVLA3	INLVLA2	INLVLA1	INLVLA0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1

Bits 0, 1, 2, 3, 4, 5, 6, 7 – INLVLAn Input Level Select on Pins Rx<7:0>, respectively

Value	Description
1	ST input used for port reads and interrupt-on-change
0	TTL input used for port reads and interrupt-on-change

15.5.25 INLVLB

Name: INLVLB
Address: 0xF10

Input Level Control Register

Bit	7	6	5	4	3	2	1	0
	INLVLB7	INLVLB6	INLVLB5	INLVLB4	INLVLB3	INLVLB2	INLVLB1	INLVLB0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1

Bits 0, 1, 2, 3, 4, 5, 6, 7 – INLVLBn Input Level Select on Pins Rx<7:0>, respectively
Note: INLVLB2 / INLVLB1: Pins read the I^2C ST inputs when MSSP inputs select these pins, and I^2C mode is enabled.

Value	Description
1	ST input used for port reads and interrupt-on-change
0	TTL input used for port reads and interrupt-on-change

15.5.26 INLVLC

Name: INLVLC
Address: 0xF18

Input Level Control Register

Bit	7	6	5	4	3	2	1	0
	INLVLC7	INLVLC6	INLVLC5	INLVLC4	INLVLC3	INLVLC2	INLVLC1	INLVLC0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1

Bits 0, 1, 2, 3, 4, 5, 6, 7 – INLVLCn Input Level Select on Pins Rx<7:0>, respectively
Note: INLVLC4 / INLVLC3: Pins read the I^2C ST inputs when MSSP inputs select these pins, and I^2C mode is enabled.

Value	Description
1	ST input used for port reads and interrupt-on-change
0	TTL input used for port reads and interrupt-on-change

15.5.27 INLVLE

Name: INLVLE
Address: 0xF25

Input Level Control Register

Bit	7	6	5	4	3	2	1	0
					INLVLE3			
Access					R/W			
Reset					1			

Bit 3 – INLVLE3 Input Level Select on Pins Rx<7:0>, respectively

Value	Description
1	ST input used for port reads and interrupt-on-change
0	TTL input used for port reads and interrupt-on-change

16. Interrupt-on-Change

16.1 Features

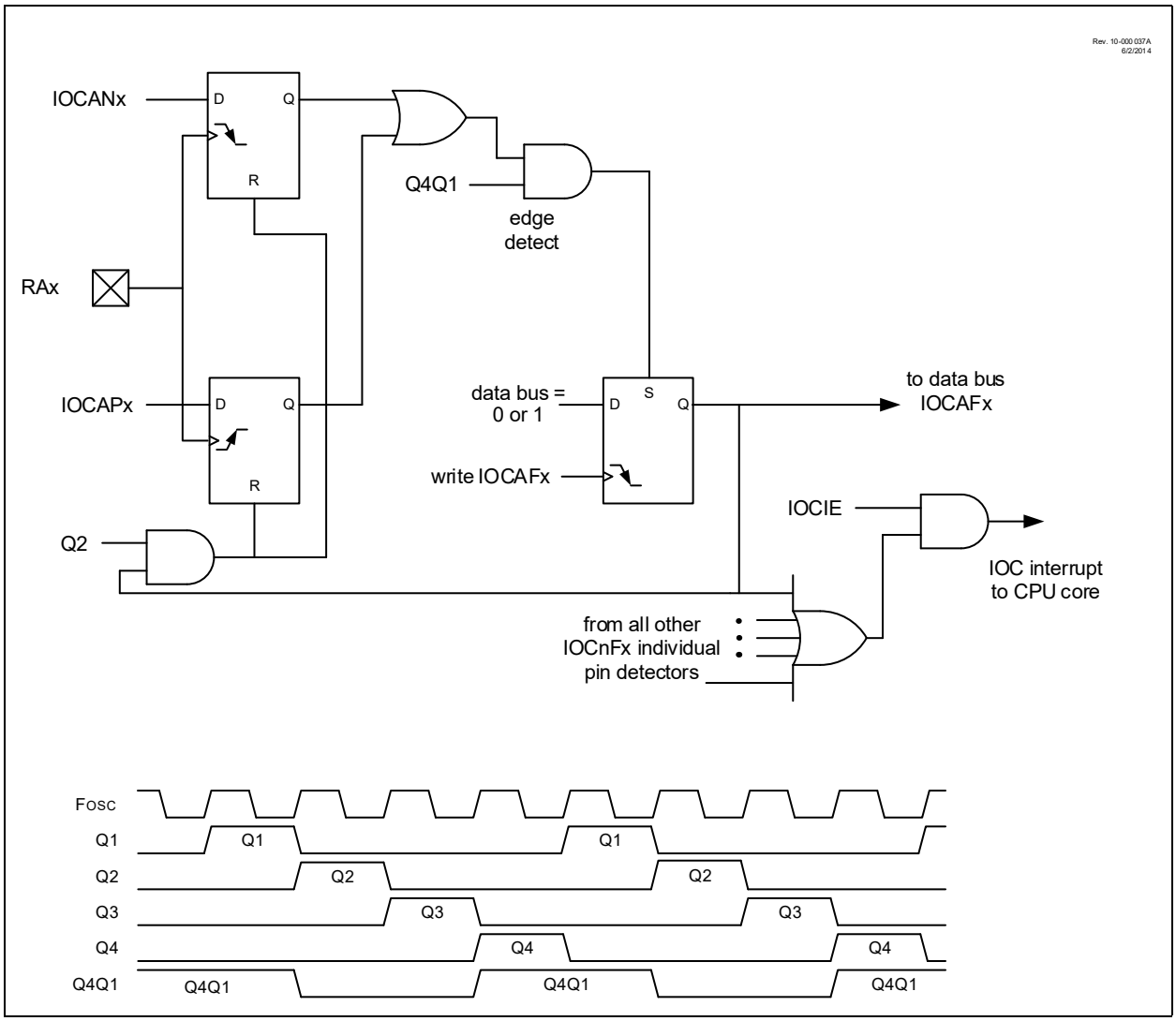
- Interrupt-on-Change enable (Master Switch)
- Individual pin configuration
- Rising and falling edge detection
- Individual pin interrupt flags

16.2 Overview

All the pins of PORTA, PORTB, PORTC, and pin RE3 of PORTE can be configured to operate as interrupt-on-change (IOC) pins on PIC18F24/25Q10 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.

16.3 Block Diagram

Figure 16-1. Interrupt-on-Change Block Diagram (PORTA Example)



16.4 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.

Related Links

14.13.10 PIE0

16.5 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.

16.6 Interrupt Flags

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 IOCAFx, IOCBFx, IOCCFx and IOCEF3 bits.

Related Links

14.13.2 PIR0

16.7 Clearing Interrupt Flags

The individual status flags, (IOCAFx, IOCBFx, IOCCFx and IOCEF3) 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 (PORTA Example)

```
MOVLW    0xff
XORWF    IOCAF, W
ANDWF    IOCAF, F
```

16.8 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.

16.9 Register Summary - Interrupt-on-Change

Address	Name	Bit Pos.								
0x0F05	IOCAF	7:0	IOCAF7	IOCAF6	IOCAF5	IOCAF4	IOCAF3	IOCAF2	IOCAF1	IOCAF0
0x0F06	IOCAN	7:0	IOCAN7	IOCAN6	IOCAN5	IOCAN4	IOCAN3	IOCAN2	IOCAN1	IOCAN0
0x0F07	IOCAP	7:0	IOCAP7	IOCAP6	IOCAP5	IOCAP4	IOCAP3	IOCAP2	IOCAP1	IOCAP0
0x0F08 ... 0x0F0C	Reserved									
0x0F0D	IOCBF	7:0	IOCBF7	IOCBF6	IOCBF5	IOCBF4	IOCBF3	IOCBF2	IOCBF1	IOCBF0
0x0F0E	IOCBN	7:0	IOCBN7	IOCBN6	IOCBN5	IOCBN4	IOCBN3	IOCBN2	IOCBN1	IOCBN0
0x0F0F	IOCBP	7:0	IOCBP7	IOCBP6	IOCBP5	IOCBP4	IOCBP3	IOCBP2	IOCBP1	IOCBP0
0x0F10 ... 0x0F14	Reserved									
0x0F15	IOCCF	7:0	IOCCF7	IOCCF6	IOCCF5	IOCCF4	IOCCF3	IOCCF2	IOCCF1	IOCCF0
0x0F16	IOCCN	7:0	IOCCN7	IOCCN6	IOCCN5	IOCCN4	IOCCN3	IOCCN2	IOCCN1	IOCCN0
0x0F17	IOCCP	7:0	IOCCP7	IOCCP6	IOCCP5	IOCCP4	IOCCP3	IOCCP2	IOCCP1	IOCCP0
0x0F18 ... 0x0F21	Reserved									
0x0F22	IOCEF	7:0					IOCEF3			
0x0F23	IOCEN	7:0					IOCEN3			
0x0F24	IOCEP	7:0					IOCEP3			

16.10 Register Definitions: Interrupt-on-Change Control

16.10.1 IOCAF

Name: IOCAF
Address: 0xF05

PORTA Interrupt-on-Change Flag Register Example

Bit	7	6	5	4	3	2	1	0
	IOCAF7	IOCAF6	IOCAF5	IOCAF4	IOCAF3	IOCAF2	IOCAF1	IOCAF0
Access	R/W/HS	R/W/HS	R/W/HS	R/W/HS	R/W/HS	R/W/HS	R/W/HS	R/W/HS
Reset	0	0	0	0	0	0	0	0

Bits 0, 1, 2, 3, 4, 5, 6, 7 – IOCAFn Interrupt-on-Change Flag bits

Value	Condition	Description
1	IOCAP[n]=1	A positive edge was detected on the RA[n] pin
1	IOCAN[n]=1	A negative edge was detected on the RA[n] pin
0	IOCAP[n]=x and IOCAN[n]=x	No change was detected, or the user cleared the detected change

16.10.2 IOCBF

Name: IOCBF
Address: 0xF0D

PORTB Interrupt-on-Change Flag Register Example

Bit	7	6	5	4	3	2	1	0
	IOCBF7	IOCBF6	IOCBF5	IOCBF4	IOCBF3	IOCBF2	IOCBF1	IOCBF0
Access	R/W/HS	R/W/HS	R/W/HS	R/W/HS	R/W/HS	R/W/HS	R/W/HS	R/W/HS
Reset	0	0	0	0	0	0	0	0

Bits 0, 1, 2, 3, 4, 5, 6, 7 – IOCBFn Interrupt-on-Change Flag bits

Value	Condition	Description
1	IOCBP[n]=1	A positive edge was detected on the RB[n] pin
1	IOCBN[n]=1	A negative edge was detected on the RB[n] pin
0	IOCBP[n]=x and IOCBN[n]=x	No change was detected, or the user cleared the detected change

16.10.3 IOCCF

Name: IOCCF
Address: 0xF15

PORTC Interrupt-on-Change Flag Register

Bit	7	6	5	4	3	2	1	0
	IOCCF7	IOCCF6	IOCCF5	IOCCF4	IOCCF3	IOCCF2	IOCCF1	IOCCF0
Access	R/W/HS	R/W/HS	R/W/HS	R/W/HS	R/W/HS	R/W/HS	R/W/HS	R/W/HS
Reset	0	0	0	0	0	0	0	0

Bits 0, 1, 2, 3, 4, 5, 6, 7 – IOCCFn Interrupt-on-Change Flag bits

Value	Condition	Description
1	IOCCP[n]=1	A positive edge was detected on the RC[n] pin
1	IOCCN[n]=1	A negative edge was detected on the RC[n] pin
0	IOCCP[n]=x and IOCCN[n]=x	No change was detected, or the user cleared the detected change

16.10.4 IOCEF

Name: IOCEF
Address: 0xF22

PORTE Interrupt-on-Change Flag Register

Bit	7	6	5	4	3	2	1	0
					IOCEF3			
Access					R/W/HS			
Reset					0			

Bit 3 – IOCEF3 PORTE Interrupt-on-Change Flag bits(1)

Value	Condition	Description
1	IOCEP[n]=1	A positive edge was detected on the RE[n] pin
1	IOCEN[n]=1	A negative edge was detected on the RE[n] pin
0	IOCEP[n]=x and IOCEN[n]=x	No change was detected, or the user cleared the detected change

Note:

1. If MCLRE = 1 or LVP = 1, RE3 port functionality is disabled and IOC on RE3 is not available.

16.10.5 IOCAN

Name: IOCAN
Address: 0xF06

Interrupt-on-Change Negative Edge Register Example

Bit	7	6	5	4	3	2	1	0
	IOCAN7	IOCAN6	IOCAN5	IOCAN4	IOCAN3	IOCAN2	IOCAN1	IOCAN0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 0, 1, 2, 3, 4, 5, 6, 7 – IOCANn Interrupt-on-Change Negative Edge Enable bits

Value	Description
1	Interrupt-on-Change enabled on the IOCA 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

16.10.6 IOCBN

Name: IOCBN
Address: 0xF0E

Interrupt-on-Change Negative Edge Register Example

Bit	7	6	5	4	3	2	1	0
	IOCBN7	IOCBN6	IOCBN5	IOCBN4	IOCBN3	IOCBN2	IOCBN1	IOCBN0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 0, 1, 2, 3, 4, 5, 6, 7 – IOCBNn Interrupt-on-Change Negative Edge Enable bits

Value	Description
1	Interrupt-on-Change enabled on the IOCA 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

16.10.7 IOCCN

Name: IOCCN
Address: 0xF16

Interrupt-on-Change Negative Edge Register Example

Bit	7	6	5	4	3	2	1	0
	IOCCN7	IOCCN6	IOCCN5	IOCCN4	IOCCN3	IOCCN2	IOCCN1	IOCCN0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 0, 1, 2, 3, 4, 5, 6, 7 – IOCCNn Interrupt-on-Change Negative Edge Enable bits

Value	Description
1	Interrupt-on-Change enabled on the IOCA 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

16.10.8 IOCEN

Name: IOCEN
Address: 0xF23

Interrupt-on-Change Negative Edge Register Example

Bit	7	6	5	4	3	2	1	0
					IOCEN3			
Access					R/W			
Reset					0			

Bit 3 – IOCEN3 Interrupt-on-Change Negative Edge Enable bits(1)

Value	Description
1	Interrupt-on-Change enabled on the IOCA 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

Note:

1. If MCLRE = 1 or LVP = 1, RE3 port functionality is disabled and IOC on RE3 is not available.

16.10.9 IOCAP

Name: IOCAP
Address: 0xF07

Interrupt-on-Change Positive Edge Register

Bit	7	6	5	4	3	2	1	0
	IOCAP7	IOCAP6	IOCAP5	IOCAP4	IOCAP3	IOCAP2	IOCAP1	IOCAP0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 0, 1, 2, 3, 4, 5, 6, 7 – IOCAPn Interrupt-on-Change Positive Edge Enable bits

Value	Description
1	Interrupt-on-Change enabled on the IOCA 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.

16.10.10 IOCBP

Name: IOCBP
Address: 0xF0F

Interrupt-on-Change Positive Edge Register

Bit	7	6	5	4	3	2	1	0
	IOCBP7	IOCBP6	IOCBP5	IOCBP4	IOCBP3	IOCBP2	IOCBP1	IOCBP0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 0, 1, 2, 3, 4, 5, 6, 7 – IOCBPn Interrupt-on-Change Positive Edge Enable bits

Value	Description
1	Interrupt-on-Change enabled on the IOCB 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.

16.10.11 IOCCP

Name: IOCCP
Address: 0xF17

Interrupt-on-Change Positive Edge Register

Bit	7	6	5	4	3	2	1	0
	IOCCP7	IOCCP6	IOCCP5	IOCCP4	IOCCP3	IOCCP2	IOCCP1	IOCCP0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 0, 1, 2, 3, 4, 5, 6, 7 – IOCCPn Interrupt-on-Change Positive Edge Enable bits

Value	Description
1	Interrupt-on-Change enabled on the IOCC 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.

16.10.12 IOCEP

Name: IOCEP
Address: 0xF24

Interrupt-on-Change Positive Edge Register

Bit	7	6	5	4	3	2	1	0
					IOCEP3			
Access					R/W			
Reset					0			

Bit 3 – IOCEP3 Interrupt-on-Change Positive Edge Enable bit(1)

Value	Description
1	Interrupt-on-Change enabled on the IOCE 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.

Note:

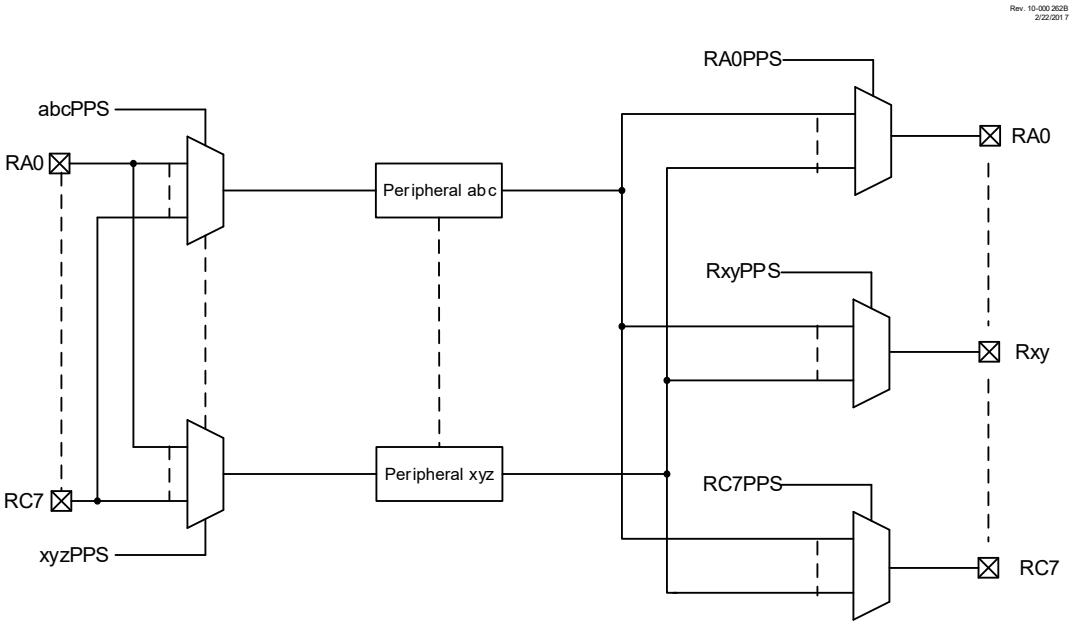
1. If MCLRE = 1 or LVP = 1, RE3 port functionality is disabled and IOC on RE3 is not available.

17. (PPS) Peripheral Pin Select 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 figure below.

The peripheral input is selected with the peripheral 17.9.1 xxxPPS register, and the peripheral output is selected with the PORT 17.9.2 RxyPPS register . For example, to select PORTC<7> as the EUSART RX input, set RXxPPS to 0x17 as shown in the input table, and to select PORTC<6> as the EUSART TX output set RC6PPS to 0x09 as shown in the output table.

Figure 17-1. Simplified PPS Block Diagram



17.1 PPS Inputs

Each peripheral has an xxxPPS register with which the input pin to the peripheral is selected. Not all ports are available for input as shown in the following table.

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.

Important: The notation “xxx” in the generic register name is a place holder for the peripheral identifier. For example, xxx = INT0 for the INT0PPS register.

Table 17-1. PPS Input Selection Register Details

Peripheral	PPS Input Register	Default Pin Selection at POR	Register Reset Value at POR	PORT From Which Input Is Available		
Interrupt 0	INT0PPS	RB0	0x08	A	B	—
Interrupt 1	INT1PPS	RB1	0x09	A	B	—
Interrupt 2	INT2PPS	RB2	0x0A	A	B	—
Timer0 Clock	T0CKIPPS	RA4	0x04	A	B	—
Timer1 Clock	T1CKIPPS	RC0	0x10	A	—	C
Timer1 Gate	T1GPPS	RB5	0x0D	—	B	C
Timer3 Clock	T3CKIPPS	RC0	0x10	—	B	C
Timer3 Gate	T3GPPS	RC0	0x10	A	—	C
Timer5 Clock	T5CKIPPS	RC2	0x12	A	—	C
Timer5 Gate	T5GPPS	RB4	0x0C	—	B	—
Timer2 Clock	T2INPPS	RC3	0x13	A	—	C
Timer4 Clock	T4INPPS	RC5	0x15	—	B	C
Timer6 Clock	T6INPPS	RB7	0x0F	—	B	—
ADC Conversion Trigger	ADACTPPS	RB4	0x0C	—	B	C
CCP1	CCP1PPS	RC2	0x12	—	B	C
CCP2	CCP2PPS	RC1	0x11	—	B	C
CWG	CWG1PPS	RB0	0x08	—	B	C
DSM Carrier Low	MDCARLPPS	RA3	0x03	A	—	C
DSM Carrier High	MDCARHPPS	RA4	0x04	A	—	C
DSM Source	MDSRCPPS	RA5	0x05	A	—	C
EUSART1 Receive	RX1PPS	RC7	0x17	—	B	C
EUSART1 Clock	CK1PPS	RC6	0x16	—	B	C
MSSP1 Clock	SSP1CLKPPS	RC3	0x13	—	B	C
MSSP1 Data	SSP1DATPPS	RC4	0x14	—	B	C
MSSP1 Slave Select	SSP1SSPPS	RA5	0x05	A	—	C

17.2 PPS Outputs

Each I/O pin has an RxyPPS register with which the pin output source is selected. With few exceptions, the port TRIS control associated with that pin retains control over the pin output driver. Peripherals that

control the pin output driver as part of the peripheral operation will override the TRIS control as needed. These peripherals include:

- EUSART (synchronous operation)
- MSSP (I^2C)

Although every pin has its own RxyPPS peripheral selection register, the selections are identical for every pin as shown in the following table.

Important: The notation “Rxy” is a place holder for the pin identifier. The 'x' holds the place of the PORT letter and the 'y' holds the place of the bit number. For example, Rxy = RA0 for the RA0PPS register.

Table 17-2. Peripheral PPS Output Selection Codes

RxyPPS	Pin Rxy Output Source	PORT To Which Output Can Be Directed		
0x13	ADGRDB	A	—	C
0x12	ADGRDA	A	—	C
0x11	DSM	A	—	C
0x10	CLKR	—	B	C
0x0F	TMR0	—	B	C
0x0E	MSSP1 (SDO/SDA)	—	B	C
0x0D	MSSP1 (SCK/SCL)	—	B	C
0x0C	CMP2	A	—	C
0x0B	CMP1	A	—	C
0x0A	EUSART1 (DT)	—	B	C
0x09	EUSART1 (TX/CK)	—	B	C
0x08	PWM4	A	—	C
0x07	PWM3	A	—	C
0x06	CCP2	—	B	C
0x05	CCP1	—	B	C
0x04	CWG1D	—	B	C
0x03	CWG1C	—	B	C
0x02	CWG1B	—	B	C
0x01	CWG1A	—	B	C
0x00	LATxy	A	B	C

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 (DT/RXxPPS and TX/CKxPPS pins for synchronous operation)
- MSSP (I^2C SDA/SSPxDATPPS and SCL/SSPxCLKPPS)



Important: The I^2C default inputs, and a limited number of other alternate pins, are I^2C and SMBus compatible. Clock and data signals can be routed to any pin, however pins without I^2C compatibility will operate at standard TTL/ST logic levels as selected by the INLVL register. See the INLVL register for each port to determine which pins are I^2C and SMBus compatible.

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 the following examples.

Example 17-1. PPS Lock Sequence

```
; Disable interrupts:
   BCF    INTCON,GIE
; Bank to PPSLOCK register
   BANKSEL 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,PPSLOCKED
; Enable Interrupts
   BSF     INTCON,GIE
```

Example 17-2. PPS Unlock Sequence

```
; Disable interrupts:
   BCF     INTCON,GIE
; Bank to PPSLOCK register
   BANKSEL 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,PPSLOCKED
; 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 input selection register table. The PPS one-way is also removed.

17.8 Register Summary - PPS

Address	Name	Bit Pos.								
0x0E9B	PPSLOCK	7:0								PPSLOCKED
0x0E9C	INT0PPS	7:0					PORT	PIN[2:0]		
0x0E9D	INT1PPS	7:0					PORT	PIN[2:0]		
0x0E9E	INT2PPS	7:0					PORT	PIN[2:0]		
0x0E9F	T0CKIPPS	7:0					PORT	PIN[2:0]		
0x0EA0	T1CKIPPS	7:0				PORT[1:0]		PIN[2:0]		
0x0EA1	T1GPPS	7:0				PORT[1:0]		PIN[2:0]		
0x0EA2	T3CKIPPS	7:0				PORT[1:0]		PIN[2:0]		
0x0EA3	T3GPPS	7:0				PORT[1:0]		PIN[2:0]		
0x0EA4	T5CKIPPS	7:0				PORT[1:0]		PIN[2:0]		
0x0EA5	T5GPPS	7:0				PORT[1:0]		PIN[2:0]		
0x0EA6	T2INPPS	7:0				PORT[1:0]		PIN[2:0]		
0x0EA7	T4INPPS	7:0				PORT[1:0]		PIN[2:0]		
0x0EA8	T6INPPS	7:0				PORT[1:0]		PIN[2:0]		
0x0EA9	ADACTPPS	7:0				PORT[1:0]		PIN[2:0]		
0x0EAA	CCP1PPS	7:0				PORT[1:0]		PIN[2:0]		
0x0EAB	CCP2PPS	7:0				PORT[1:0]		PIN[2:0]		
0x0EAC	CWG1PPS	7:0				PORT[1:0]		PIN[2:0]		
0x0EAD	MDCARLPPS	7:0				PORT[1:0]		PIN[2:0]		
0x0EAE	MDCARHPPS	7:0				PORT[1:0]		PIN[2:0]		
0x0EAF	MDSRCPPS	7:0				PORT[1:0]		PIN[2:0]		
0x0EB0	RX1PPS	7:0				PORT[1:0]		PIN[2:0]		
0x0EB1	CK1PPS	7:0				PORT[1:0]		PIN[2:0]		
0x0EB2	SSP1CLKPPS	7:0				PORT[1:0]		PIN[2:0]		
0x0EB3	SSP1DATPPS	7:0				PORT[1:0]		PIN[2:0]		
0x0EB4	SSP1SSPPS	7:0				PORT[1:0]		PIN[2:0]		
0x0EB5 ... 0x0EE1	Reserved									
0x0EE2	RA0PPS	7:0				PPS[4:0]				
0x0EE3	RA1PPS	7:0				PPS[4:0]				
0x0EE4	RA2PPS	7:0				PPS[4:0]				
0x0EE5	RA3PPS	7:0				PPS[4:0]				
0x0EE6	RA4PPS	7:0				PPS[4:0]				
0x0EE7	RA5PPS	7:0				PPS[4:0]				
0x0EE8	RA6PPS	7:0				PPS[4:0]				
0x0EE9	RA7PPS	7:0				PPS[4:0]				
0x0EEA	RB0PPS	7:0				PPS[4:0]				
0x0EEB	RB1PPS	7:0				PPS[4:0]				
0x0EEC	RB2PPS	7:0				PPS[4:0]				
0x0EED	RB3PPS	7:0				PPS[4:0]				
0x0EEE	RB4PPS	7:0				PPS[4:0]				
0x0EEF	RB5PPS	7:0				PPS[4:0]				

Address	Name	Bit Pos.								
0x0EF0	RB6PPS	7:0				PPS[4:0]				
0x0EF1	RB7PPS	7:0				PPS[4:0]				
0x0EF2	RC0PPS	7:0				PPS[4:0]				
0x0EF3	RC1PPS	7:0				PPS[4:0]				
0x0EF4	RC2PPS	7:0				PPS[4:0]				
0x0EF5	RC3PPS	7:0				PPS[4:0]				
0x0EF6	RC4PPS	7:0				PPS[4:0]				
0x0EF7	RC5PPS	7:0				PPS[4:0]				
0x0EF8	RC6PPS	7:0				PPS[4:0]				
0x0EF9	RC7PPS	7:0				PPS[4:0]				

17.9 Register Definitions: PPS Input and Output Selection

17.9.1 Peripheral xxx Input Selection

Name: xxxPPS

Important: The Reset value of this register is determined by the device default for each peripheral.
Refer to the input selection table for a list of available ports and default pin locations.

Bit	7	6	5	4	3	2	1	0
				PORT[1:0]		PIN[2:0]		
Access				R/W	R/W	R/W	R/W	R/W
Reset				g	g	g	g	g

Bits 4:3 – PORT[1:0] Peripheral xxx Input PORT Selection bits
See the input selection table for a list of available ports and default pin locations.

Value	Description
10	PORTC
01	PORTB
00	PORTA

Bits 2:0 – PIN[2:0] Peripheral xxx Input Pin Selection bits

Value	Description
111	Peripheral input is from PORTx Pin 7 (Rx7)
110	Peripheral input is from PORTx Pin 6 (Rx6)
101	Peripheral input is from PORTx Pin 5 (Rx5)
100	Peripheral input is from PORTx Pin 4 (Rx4)
011	Peripheral input is from PORTx Pin 3 (Rx3)
010	Peripheral input is from PORTx Pin 2 (Rx2)
001	Peripheral input is from PORTx Pin 1 (Rx1)
000	Peripheral input is from PORTx Pin 0 (Rx0)

17.9.2 Pin Rxy Output Source Selection Register

Name: RxyPPS

Important: See 17.8 Register Summary - PPS for the address offset of each individual register.

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

Bits 4:0 – RxyPPS[4:0] Pin Rxy Output Source Selection bits
See output source selection table for source codes.

17.9.3 PPS Lock Register

Name: PPSLOCK
Address: 0xE9B

Bit	7	6	5	4	3	2	1	0
								PPSLOCKED
Access								R/W
Reset								0

Bit 0 – PPSLOCKED PPS Locked bit

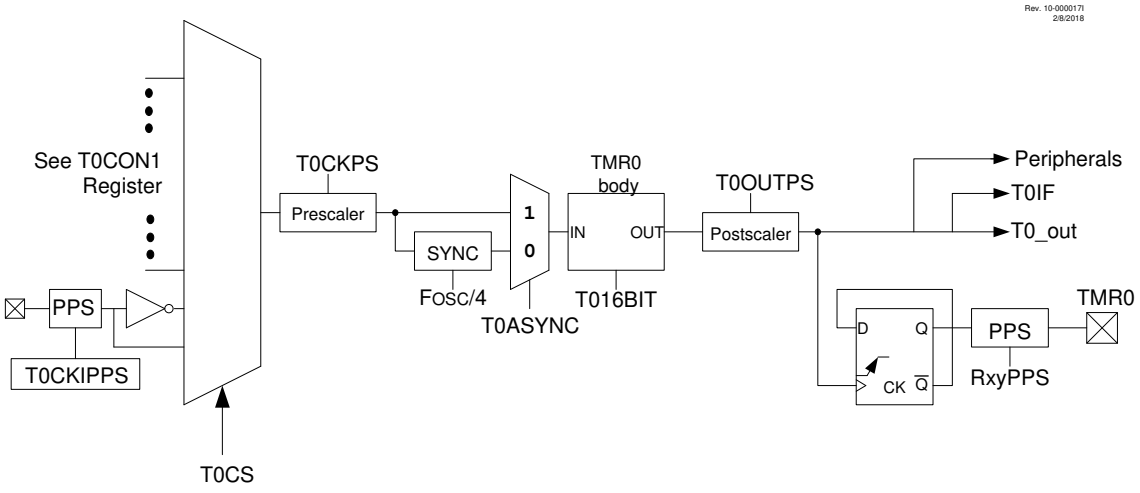
Value	Description
1	PPS is locked. PPS selections can not be changed.
0	PPS is not locked. PPS selections can be changed.

18. Timer0 Module

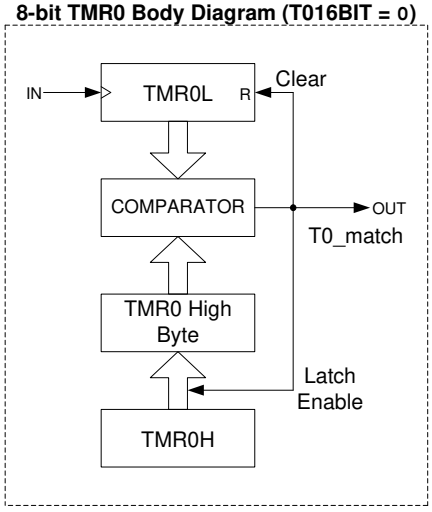
Timer0 module has the following features:

- 8-Bit B\Timer with Programmable Period
- 16-Bit Timer
- Selectable Clock Sources
- Synchronous and Asynchronous Operation
- Programmable Prescaler and Postscaler
- Interrupt on Match or Overflow
- Output on I/O Pin (via PPS) or to Other Peripherals
- Operation During Sleep

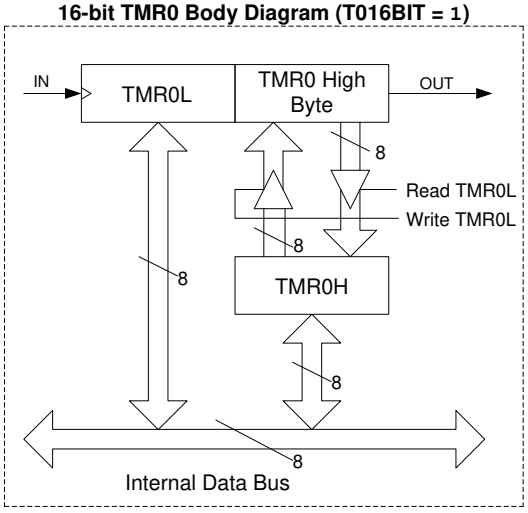
Figure 18-1. Timer0 Block Diagram



8-bit TMR0 Body Diagram (T016BIT = 0)



16-bit TMR0 Body Diagram (T016BIT = 1)



18.1 Timer0 Operation

Timer0 can operate as either an 8-bit or 16-bit timer. The mode is selected with the T016BIT bit.

18.1.1 8-bit Mode

In this mode Timer0 increments on the rising edge of the selected clock source. A prescaler on the clock input gives several prescale options (see prescaler control bits, T0CKPS).

In this mode as shown in Figure 18-1, a buffered version of TMR0H is maintained. This is compared with the value of TMR0L on each cycle of the selected clock source. When the two values match, the following events occur:

- TMR0L is reset
- The contents of TMR0H are copied to the TMR0H buffer for next comparison

18.1.2 16-Bit Mode

In this mode Timer0 increments on the rising edge of the selected clock source. A prescaler on the clock input gives several prescale options (see prescaler control bits, T0CKPS).

In this mode TMR0H:TMR0L form the 16-bit timer value. As shown in Figure 18-1, read and write of the TMR0H register are buffered. TMR0H register is updated with the contents of the high byte of Timer0 during a read of TMR0L register. Similarly, a write to the high byte of Timer0 takes place through the TMR0H buffer register. The high byte is updated with the contents of TMR0H register when a write occurs to TMR0L register. This allows all 16 bits of Timer0 to be read and written at the same time.

Timer0 rolls over to 0x0000 on incrementing past 0xFFFF. This makes the timer free running. TMR0L/H registers cannot be reloaded in this mode once started.

18.2 Clock Selection

Timer0 has several options for clock source selections, option to operate synchronously/asynchronously and a programmable prescaler.

18.2.1 Clock Source Selection

The T0CS bits are used to select the clock source for Timer0. The possible clock sources are listed in the table below.

Table 18-1. Timer 0 Clock Source Selections

T0CS	Clock Source
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)

18.2.2 Synchronous Mode

When the T0ASYNC bit is clear, Timer0 clock is synchronized to the system clock ($F_{OSC}/4$). When operating in Synchronous mode, Timer0 clock frequency cannot exceed $F_{OSC}/4$. During Sleep mode system clock is not available and Timer0 cannot operate.

18.2.3 Asynchronous Mode

When the T0ASYNC bit is set, Timer0 increments with each rising edge of the input source (or output of the prescaler, if used). Asynchronous mode allows Timer0 to continue operation during Sleep mode provided the selected clock source is available.

18.2.4 Programmable Prescaler

Timer0 has 16 programmable input prescaler options ranging from 1:1 to 1:32768. The prescaler values are selected using the T0CKPS bits.

The prescaler counter is not directly readable or writable. The prescaler counter is cleared on the following events:

- A write to the TMR0L register
- A write to either the T0CON0 or T0CON1 registers
- Any device Reset

Related Links

8. Resets

18.3 Timer0 Output and Interrupt

18.3.1 Programmable Postscaler

Timer0 has 16 programmable output postscaler options ranging from 1:1 to 1:16. The postscaler values are selected using the T0OUTPS bits. The postscaler divides the output of Timer0 by the selected ratio.

The postscaler counter is not directly readable or writable. The postscaler counter is cleared on the following events:

- A write to the TMR0L register
- A write to either the T0CON0 or T0CON1 registers
- Any device Reset

18.3.2 Timer0 Output

TMR0_out is the output of the postscaler. TMR0_out toggles on every match between TMR0L and TMR0H in 8-bit mode, or when TMR0H:TMR0L rolls over in 16-bit mode. If the output postscaler is used, the output is scaled by the ratio selected.

The Timer0 output can be routed to an I/O pin via the RxyPPS output selection register. The Timer0 output can be monitored through software via the T0OUT output bit.

Related Links

17.2 PPS Outputs

18.3.3 Timer0 Interrupt

The Timer0 Interrupt Flag bit (TMR0IF) is set when the TMR0_out toggles. If the Timer0 interrupt is enabled (TMR0IE), the CPU will be interrupted when the TMR0IF bit is set.

When the postscaler bits (T0OUTPS) are set to 1:1 operation (no division), the T0IF flag bit will be set with every TMR0 match or rollover. In general, the TMR0IF flag bit will be set every T0OUTPS +1 matches or rollovers.

18.3.4 Timer0 Example

Timer0 Configuration:

- Timer0 mode = 16-bit
- Clock Source = $F_{OSC}/4$ (250 kHz)
- Synchronous operation
- Prescaler = 1:1
- Postscaler = 1:2 (T0OUTPS = `1`)

In this case the TMR0_out toggles every two rollovers of TMR0H:TMR0L. i.e., (0xFFFF)*2*(1/250kHz) = 524.28 ms

18.4 Operation During Sleep

When operating synchronously, Timer0 will halt when the device enters Sleep mode.

When operating asynchronously and selected clock source is active, Timer0 will continue to increment and wake the device from Sleep mode if Timer0 interrupt is enabled.

18.5 Register Summary - Timer0

Address	Name	Bit Pos.								
0x0FD2	TMR0L	7:0	TMR0L[7:0]							
0x0FD3	TMR0H	7:0	TMR0H[7:0]							
0x0FD4	T0CON0	7:0	T0EN		T0OUT	T016BIT	T0OUTPS[3:0]			
0x0FD5	T0CON1	7:0	T0CS[2:0]			T0ASYNC	T0CKPS[3:0]			

18.6 Register Definitions: Timer0 Control

18.6.1 T0CON0

Name: T0CON0
Address: 0xFD4

Timer0 Control Register 0

Bit	7	6	5	4	3	2	1	0
	T0EN		T0OUT	T016BIT	T0OUTPS[3:0]			
Access	R/W		R	R/W	R/W	R/W	R/W	R/W
Reset	0		0	0	0	0	0	0

Bit 7 – T0EN TMR0 Enable bit

Value	Description
1	The module is enabled and operating
0	The module is disabled

Bit 5 – T0OUT TMR0 Output bit

Bit 4 – T016BIT TMR0 Operating as 16-Bit Timer Select bit

Value	Description
1	TMR0 is a 16-bit timer
0	TMR0 is an 8-bit timer

Bits 3:0 – T0OUTPS[3:0] TMR0 Output Postscaler (Divider) Select bits

Value	Description
1111	1:16 Postscaler
1110	1:15 Postscaler
1101	1:14 Postscaler
1100	1:13 Postscaler
1011	1:12 Postscaler
1010	1:11 Postscaler
1001	1:10 Postscaler
1000	1:9 Postscaler
0111	1:8 Postscaler
0110	1:7 Postscaler
0101	1:6 Postscaler
0100	1:5 Postscaler
0011	1:4 Postscaler
0010	1:3 Postscaler
0001	1:2 Postscaler
0000	1:1 Postscaler

18.6.2 T0CON1

Name: T0CON1
Address: 0xFD5

Timer0 Control Register 1

Bit	7	6	5	4	3	2	1	0
	T0CS[2:0]			T0ASYNC	T0CKPS[3:0]			
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:5 – T0CS[2:0] Timer0 Clock Source Select bits
Refer the clock source selection table

Bit 4 – T0ASYNC TMR0 Input Asynchronization Enable bit

Value	Description
1	The input to the TMR0 counter is not synchronized to system clocks
0	The input to the TMR0 counter is synchronized to Fosc/4

Bits 3:0 – T0CKPS[3:0] Prescaler Rate Select bit

Value	Description
1111	1:32768
1110	1:16384
1101	1:8192
1100	1:4096
1011	1:2048
1010	1:1024
1001	1:512
1000	1:256
0111	1:128
0110	1:64
0101	1:32
0100	1:16
0011	1:8
0010	1:4
0001	1:2
0000	1:1

18.6.3 TMR0H

Name: TMR0H
Address: 0xFD3

Timer0 Period/Count High Register

Bit	7	6	5	4	3	2	1	0
	TMR0H[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – TMR0H[7:0] TMR0 Most Significant Counter bits

Value	Condition	Description
0 to 255	T016BIT = 0	8-bit Timer0 Period Value. TMR0L continues counting from 0 when this value is reached.
0 to 255	T016BIT = 1	16-bit Timer0 Most Significant Byte

18.6.4 TMR0L

Name: TMR0L
Address: 0xFD2

Timer0 Period/Count Low Register

Bit	7	6	5	4	3	2	1	0
	TMR0L[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – TMR0L[7:0] TMR0 Least Significant Counter bits

Value	Condition	Description
0 to 255	T016BIT = 0	8-bit Timer0 Counter bits
0 to 255	T016BIT = 1	16-bit Timer0 Least Significant Byte

19. Timer1 Module with Gate Control

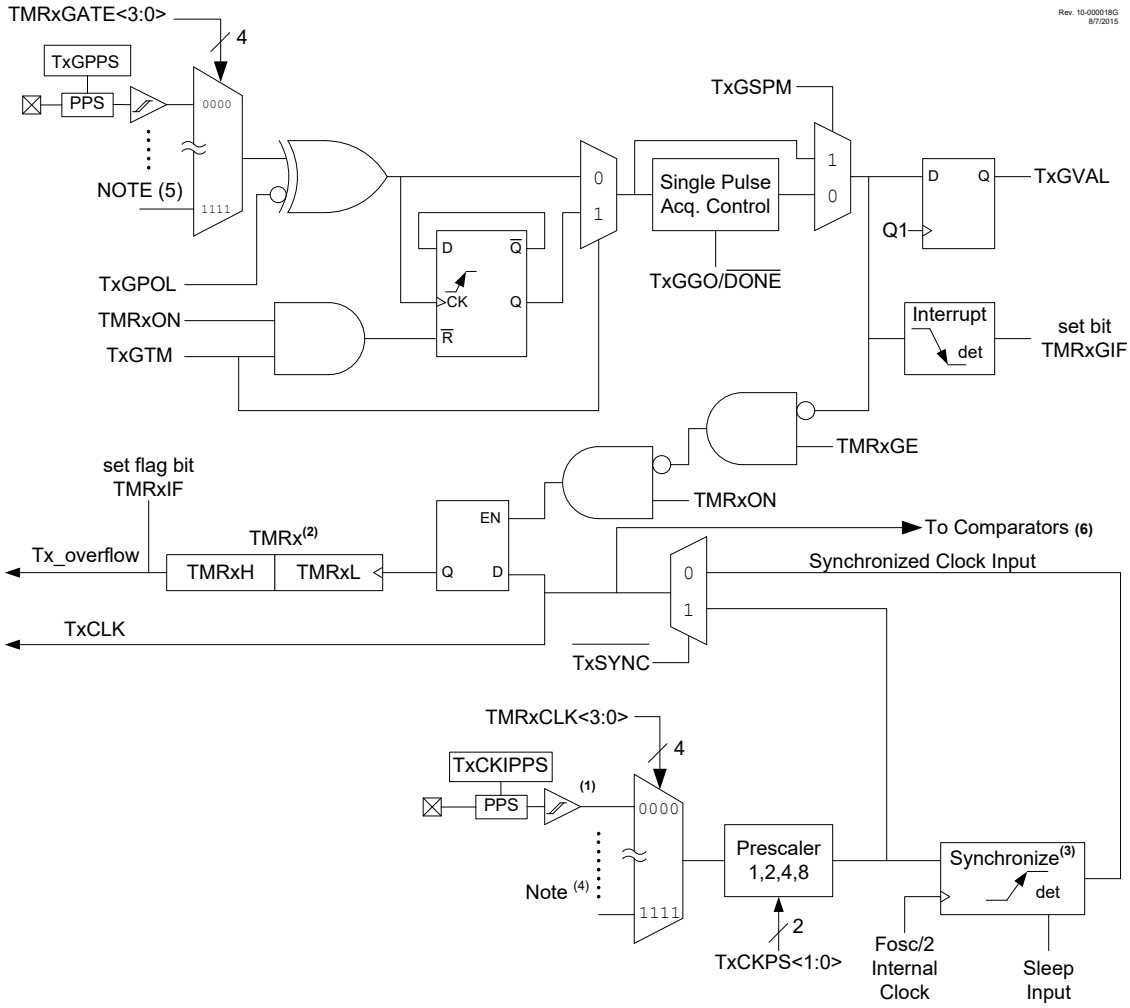
Timer1 module is a 16-bit timer/counter with the following features:

- 16-Bit Timer/Counter Register Pair (TMRxH:TMRxL)
- Programmable Internal or External Clock Source
- 2-Bit Prescaler
- Optionally Synchronized Comparator Out
- Multiple Timer1 Gate (count enable) Sources
- Interrupt-on-Overflow
- Wake-Up on Overflow (external clock, Asynchronous mode only)
- 16-Bit Read/Write Operation
- Time Base for the Capture/Compare Function with the CCP modules
- Special Event Trigger (with CCP)
- Selectable Gate Source Polarity
- Gate Toggle mode
- Gate Single-Pulse mode
- Gate Value Status
- Gate Event Interrupt



Important: References to module Timer1 apply to all the odd numbered timers on this device.

Figure 19-1. Timer1 Block Diagram



Note:

1. This signal comes from the pin seleted by TxCKIPPS.
2. TMRx register increments on rising edge.
3. Synchronize does not operate while in Sleep.
4. See TMRxCLK for clock source selections.
5. See TMRxGATE for gate source selection.
6. Synchronized comparator output should not be used in conjunction with synchronized input clock.

19.1 Timer1 Operation

The Timer1 module is a 16-bit incrementing counter that 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 is enabled by configuring the ON and GE bits in the TxCON and TxGCON registers, respectively. The table below displays the Timer1 enable selections.

Table 19-1. Timer1 Enable Selections

ON	GE	Timer1 Operation
1	1	Count Enabled
1	0	Always On
0	1	Off
0	0	Off

19.2 Clock Source Selection

The CS bits select the clock source for Timer1. These bits allow the selection of several possible synchronous and asynchronous clock sources. The table below lists the clock source selections.

Table 19-2. Timer Clock Sources

CS	Clock Source		
	Timer1	Timer3	Timer5
1111-1100	Reserved	Reserved	Reserved
1011	TMR5 overflow	TMR5 overflow	Reserved
1010	TMR3 overflow	Reserved	TMR3 overflow
1001	Reserved	TMR1 overflow	TMR1 overflow
1000	TMR0 overflow	TMR0 overflow	TMR0 overflow
0111	CLKREF	CLKREF	CLKREF
0110	SOSC	SOSC	SOSC
0101	MFINTOSC (500 kHz)	MFINTOSC (500 kHz)	MFINTOSC (500 kHz)
0100	LFINTOSC	LFINTOSC	LFINTOSC
0011	HFINTOSC	HFINTOSC	HFINTOSC
0010	Fosc	Fosc	Fosc
0001	Fosc/4	Fosc/4	Fosc/4
0000	T1CKIPPS	T3CKIPPS	T5CKIPPS

19.2.1 Internal Clock Source

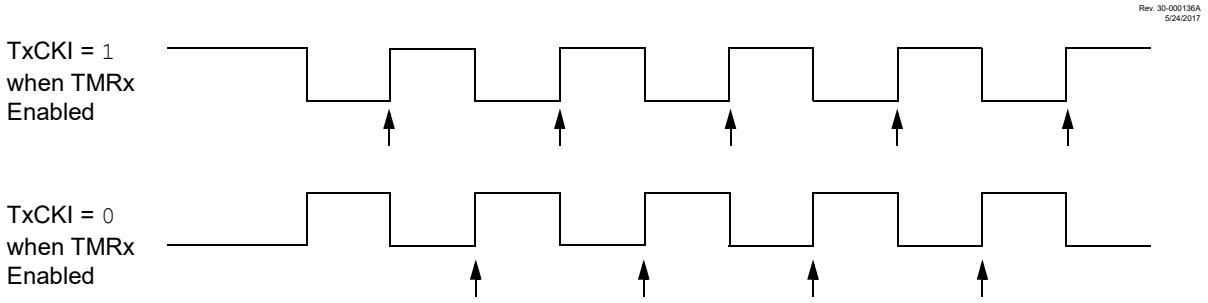
When the internal clock source is selected the TMRxH:TMRxL register pair will increment on multiples of F_{OSC} as determined by the Timer1 prescaler.

When the F_{OSC} internal clock source is selected, the Timer1 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 value. To utilize the full resolution of Timer1, an asynchronous input signal must be used to gate the Timer1 clock input.

Important: In Counter mode, a falling edge must be registered by the counter prior to the first incrementing rising edge after any one or more of the following conditions:

- Timer1 enabled after POR
- Write to TMRxH or TMRxL
- Timer1 is disabled
- Timer1 is disabled (TMRxON = 0) when TxCKI is high then Timer1 is enabled (TMRxON = 1) when TxCKI is low. Refer to the figure below.

Figure 19-2. Timer1 Incrementing Edge



Note:

1. Arrows indicate counter increments.
2. In Counter mode, a falling edge must be registered by the counter prior to the first incrementing rising edge of the clock.

19.2.2 External Clock Source

When the external clock source is selected, the Timer1 module may work as a timer or a counter.

When enabled to count, Timer1 is incremented on the rising edge of the external clock input of the TxCKIPPS pin. This external clock source can be synchronized to the system clock or it can run asynchronously.

19.3 Timer1 Prescaler

Timer1 has four prescaler options allowing 1, 2, 4 or 8 divisions of the clock input. The 19.14.1.1 CKPS bits control the prescale counter. The prescale counter is not directly readable or writable; however, the prescaler counter is cleared upon a write to TMRxH or TMRxL.

19.4 Secondary Oscillator

A secondary low-power 32.768 kHz oscillator circuit is built-in between pins SOSCI (input) and SOSCO (amplifier output). This internal circuit is to be used in conjunction with an external 32.768 kHz crystal. The secondary oscillator is not dedicated only to Timer1; it can also be used by other modules.

The oscillator circuit is enabled by setting the SOSCEN bit of the OSCEN register. This can be used as one of the Timer1 clock sources selected with the 19.14.3.1 CS bits.The oscillator will continue to run during Sleep.

Important: The oscillator requires a start-up and stabilization time before use. Thus, the SOSCEN bit of the OSCEN register should be set and a suitable delay observed prior to enabling Timer1. A software check can be performed to confirm if the secondary oscillator is enabled and ready to use. This is done by polling the SOR bit of the OSCSTAT.

Related Links

4.2.1.5 Secondary Oscillator

19.5 Timer1 Operation in Asynchronous Counter Mode

When the 19.14.1.2 SYNC control bit 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 19.5.1 Reading and Writing Timer1 in Asynchronous Counter Mode).

Important: When switching from synchronous to asynchronous operation, it is possible to skip an increment. When switching from asynchronous to synchronous operation, it is possible to produce an additional increment.

19.5.1 Reading and Writing Timer1 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.

19.6 Timer1 16-Bit Read/Write Mode

Timer1 can be configured to read and write all 16 bits of data, to and from, the 8-bit TMRxL and TMRxH registers, simultaneously. The 16-bit read and write operations are enabled by setting the 19.14.1.3 RD16 bit.

To accomplish this function, the TMRxH register value is mapped to a buffer register called the TMRxH buffer register. While in 16-Bit mode, the TMRxH register is not directly readable or writable and all read and write operations take place through the use of this TMRxH buffer register.

When a read from the TMRxL register is requested, the value of the TMRxH register is simultaneously loaded into the TMRxH buffer register. When a read from the TMRxH register is requested, the value is provided from the TMRxH buffer register instead. This provides the user with the ability to accurately read all 16 bits of the Timer1 value from a single instance in time. Refer the figure below for more details.

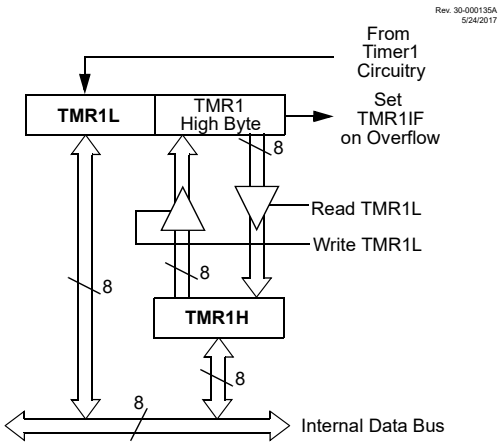
In contrast, when not in 16-Bit mode, the user must read each register separately and determine if the values have become invalid due to a rollover that may have occurred between the read operations.

When a write request of the TMRxL register is requested, the TMRxH buffer register is simultaneously updated with the contents of the TMRxH register. The value of TMRxH must be preloaded into the

TMRxH buffer register prior to the write request for the TMRxL register. This provides the user with the ability to write all 16 bits to the TMRxL:TMRxH register pair at the same time.

Any requests to write to the TMRxH directly does not clear the Timer1 prescaler value. The prescaler value is only cleared through write requests to the TMRxL register.

Figure 19-3. Timer1 16-Bit Read/Write Mode Block Diagram



19.7 Timer1 Gate

Timer1 can be configured to count freely or the count can be enabled and disabled using Timer1 gate circuitry. This is also referred to as Timer1 gate enable.

Timer1 gate can also be driven by multiple selectable sources.

19.7.1 Timer1 Gate Enable

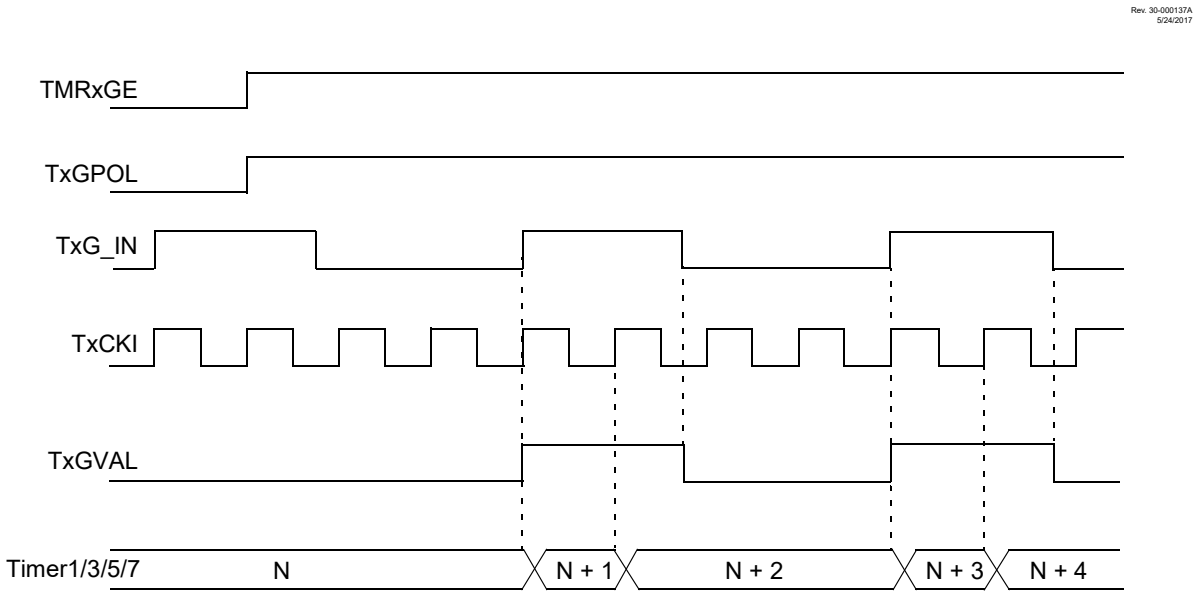
The Timer1 Gate Enable mode is enabled by setting the GE bit. The polarity of the Timer1 Gate Enable mode is configured using the GPOL bit.

When Timer1 Gate Enable mode is enabled, Timer1 will increment on the rising edge of the Timer1 clock source. When Timer1 Gate signal is inactive, the timer will not increment and hold the current count. Enable mode is disabled, no incrementing will occur and Timer1 will hold the current count. See figure below for timing details.

Table 19-3. Timer1 Gate Enable Selections

TMRxCLK	GPOL	TxG	Timer1 Operation
↑	1	1	Counts
↑	1	0	Holds Count
↑	0	1	Holds Count
↑	0	0	Counts

Figure 19-4. Timer1 Gate Enable Mode



19.7.2 Timer1 Gate Source Selection

The gate source for Timer1 is selected using the GSS bits. The polarity selection for the gate source is controlled by the GPOL bit. The table below lists the gate source selections.

Table 19-4. Timer Gate Sources

GSS	Gate Source		
	Timer1	Timer3	Timer5
1111	Reserved	Reserved	Reserved
1110	ZCDOUT	ZCDOUT	ZCDOUT
1101	CMP2OUT	CMP2OUT	CMP2OUT
1100	CMP1OUT	CMP1OUT	CMP1OUT
1011	PWM4OUT	PWM4OUT	PWM4OUT
1010	PWM3OUT	PWM3OUT	PWM3OUT
1001	CCP2OUT	CCP2OUT	CCP2OUT
1000	CCP1OUT	CCP1OUT	CCP1OUT
0111	TMR6OUT (post-scaled)	TMR6OUT (post-scaled)	TMR6OUT (post-scaled)
0110	TMR5 overflow	TMR5 overflow	Reserved
0101	TMR4OUT (post-scaled)	TMR4OUT (post-scaled)	TMR4OUT (post-scaled)
0100	TMR3 overflow	Reserved	TMR3 overflow
0011	TMR2OUT (post-scaled)	TMR2OUT (post-scaled)	TMR2OUT (post-scaled)

GSS	Gate Source		
	Timer1	Timer3	Timer5
0010	Reserved	TMR1 overflow	TMR1 overflow
0001	TMR0 overflow	TMR0 overflow	TMR0 overflow
0000	Pin selected by T1GPPS	Pin selected by T3GPPS	Pin selected by T5GPPS

Any of the above mentioned signals can be used to trigger the gate. The output of the CMPx can be synchronized to the Timer1 clock or left asynchronous. For more information refer to the Comparator Output Synchronization section.

Related Links

32.4.1 Comparator Output Synchronization

19.7.3 Timer1 Gate Toggle Mode

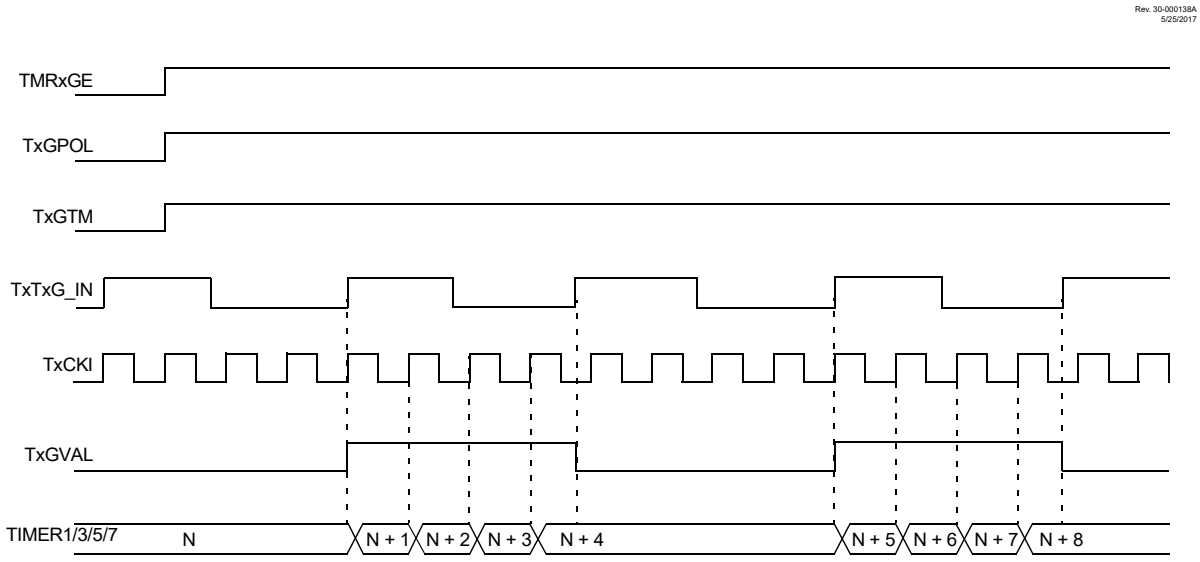
When Timer1 Gate Toggle mode is enabled, it is possible to measure the full-cycle length of a Timer1 gate signal, as opposed to the duration of a single level pulse.

The Timer1 gate source is routed through a flip-flop that changes state on every incrementing edge of the signal. See figure below for timing details.

Timer1 Gate Toggle mode is enabled by setting the 19.14.2.3 GTM bit. 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.

Important: Enabling Toggle mode at the same time as changing the gate polarity may result in indeterminate operation.

Figure 19-5. TIMER1 GATE TOGGLE MODE



19.7.4 Timer1 Gate Single-Pulse Mode

When Timer1 Gate Single-Pulse mode is enabled, it is possible to capture a single-pulse gate event. Timer1 Gate Single-Pulse mode is first enabled by setting the 19.14.2.4 GSPM bit. Next, the 19.14.2.5 GGO/DONE bit must be set. The Timer1 will be fully enabled on the next incrementing edge. On the next trailing edge of the pulse, the GGO/$\overline{\text{DONE}}$ bit will automatically be cleared. No other gate events will be allowed to increment Timer1 until the GGO/$\overline{\text{DONE}}$ bit is once again set in software.

Clearing the GSPM bit will also clear the GGO/$\overline{\text{DONE}}$ bit. See figure below for timing details.

Enabling the Toggle mode and the Single-Pulse mode simultaneously will permit both sections to work together. This allows the cycle times on the Timer1 gate source to be measured. See figure below for timing details.

Figure 19-6. TIMER1 GATE SINGLE-PULSE MODE

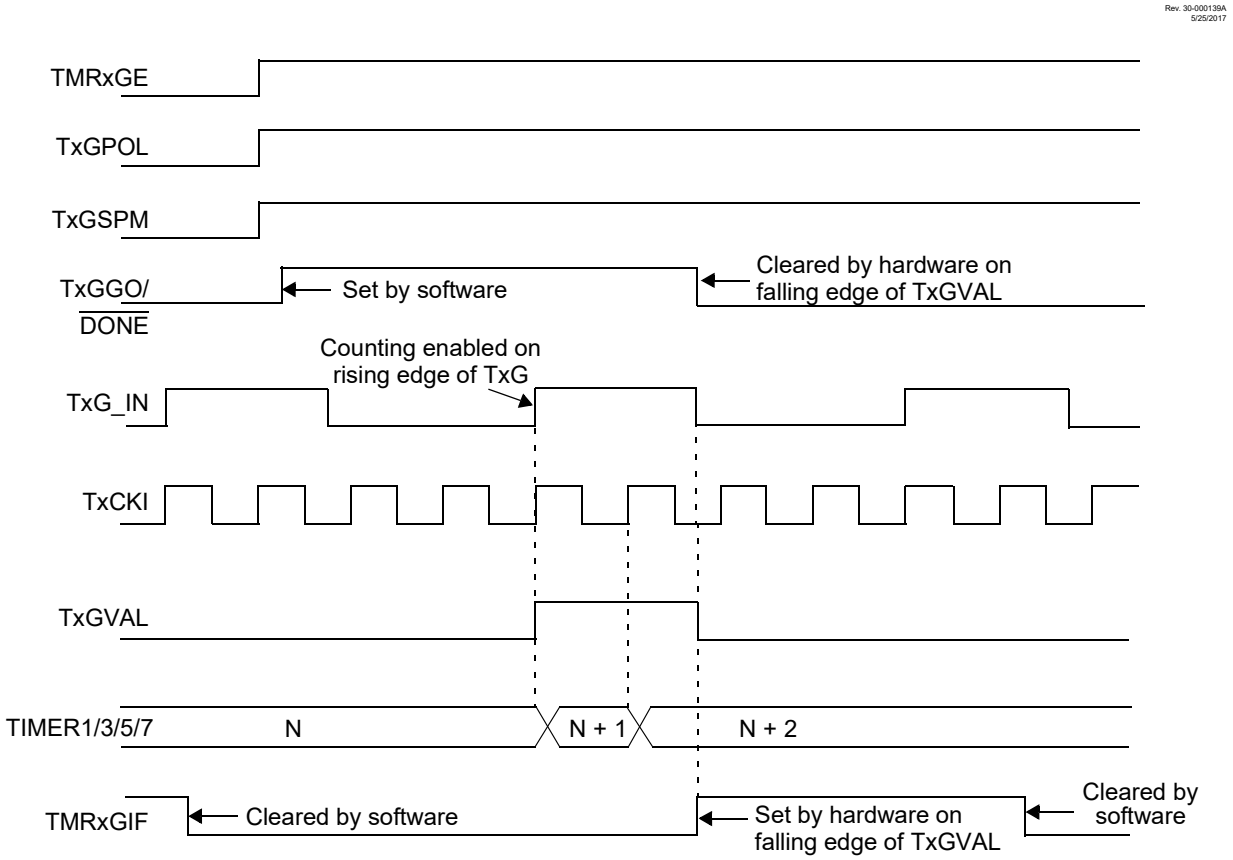
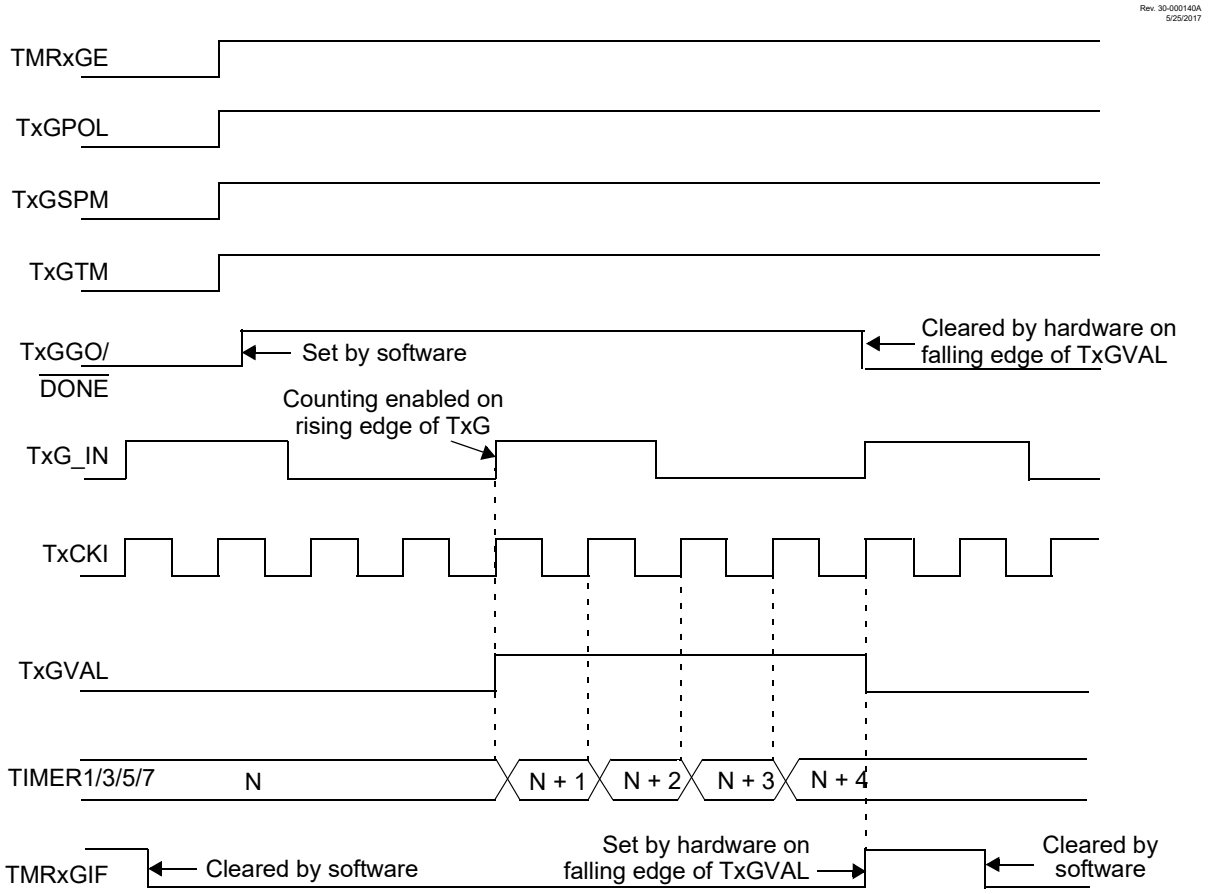


Figure 19-7. TIMER1 GATE SINGLE-PULSE AND TOGGLE COMBINED MODE



19.7.5 Timer1 Gate Value Status

When Timer1 Gate Value Status is utilized, it is possible to read the most current level of the gate control value. The value is stored in the GVAL bit in the TxGCON register. The GVAL bit is valid even when the Timer1 gate is not enabled (GE bit is cleared).

19.7.6 Timer1 Gate Event Interrupt

When Timer1 gate event interrupt is enabled, it is possible to generate an interrupt upon the completion of a gate event. When the falling edge of GVAL occurs, the TMRxGIF flag bit in 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 gate is not enabled (GE bit is cleared).

For more information on selecting high or low priority status for the Timer1 gate event interrupt see the Interrupts chapter.

Related Links

14.2 Interrupt Priority

19.8 Timer1 Interrupt

The Timer1 register pair (TMRxH:TMRxL) increments to FFFFh and rolls over to 0000h. When Timer1 rolls over, the Timer1 interrupt flag bit of the PIRx register is set. To enable the interrupt-on-rollover, the following bits must be set:

- TMRxON bit of the TxCON register
- TMRxIE bits of the PIEx 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 overflow interrupt, see the Interrupts chapter.

Important: The TMRxH:TMRxL register pair and the TMRxIF bit should be cleared before enabling interrupts.

Related Links

14.2 Interrupt Priority

19.9 Timer1 Operation During Sleep

Timer1 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 PIEx register must be set
- PEIE/GIEL bit of the INTCON register must be set
- $\overline{\text{TxSYNC}}$ bit of the TxCON register must be set
- Configure the TMRxCLK register for using secondary oscillator as the clock source
- Enable the SOSCEN bit of the OSCEN register

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 $\overline{\text{TxSYNC}}$ bit setting.

19.10 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.

© 2018 Microchip Technology Inc.

For more information, see Capture/Compare/PWM Module(CCP) chapter.

Related Links

21. Capture/Compare/PWM Module

19.11 CCP Special Event Trigger

When any of the CCPs are configured to trigger a special event, the trigger will clear the TMRxH:TMRxL register pair. This special event does not cause a Timer1 interrupt. The CCP module may still be configured to generate a CCP interrupt.

In this mode of operation, the CCPRxH:CCPRxL register pair becomes the period register for Timer1.

Timer1 should be synchronized and $F_{OSC}/4$ should be selected as the clock source in order to utilize the Special Event Trigger. Asynchronous operation of Timer1 can cause a Special Event Trigger to be missed.

In the event that a write to TMRxH or TMRxL coincides with a Special Event Trigger from the CCP, the write will take precedence.

19.12 Peripheral Module Disable

When a peripheral is not used or inactive, the module can be disabled by setting the Module Disable bit in the PMD registers. This will reduce power consumption to an absolute minimum. Setting the PMD bits holds the module in Reset and disconnects the module's clock source. The Module Disable bits for Timer1 (TMR1MD) are in the PMD1 register. See Peripheral Module Disable (PMD) chapter for more information.

Related Links

7.3 Register Summary - PMD

19.13 Register Summary - Timer1

Address	Name	Bit Pos.								
0x0FC0	TMR5	7:0	TMRxL[7:0]							
		15:8	TMRxH[7:0]							
0x0FC2	T5CON	7:0			CKPS[1:0]			SYNC	RD16	ON
0x0FC3	T5GCON	7:0	GE	GPOL	GTM	GSPM	GGO/DONE	GVAL		
0x0FC4	TMR5GATE	7:0					GSS[3:0]			
0x0FC5	TMR5CLK	7:0					CS[3:0]			
0x0FC6	TMR3	7:0	TMRxL[7:0]							
		15:8	TMRxH[7:0]							
0x0FC8	T3CON	7:0			CKPS[1:0]			SYNC	RD16	ON
0x0FC9	T3GCON	7:0	GE	GPOL	GTM	GSPM	GGO/DONE	GVAL		
0x0FCA	TMR3GATE	7:0					GSS[3:0]			
0x0FCB	TMR3CLK	7:0					CS[3:0]			
0x0FCC	TMR1	7:0	TMRxL[7:0]							
		15:8	TMRxH[7:0]							
0x0FCE	T1CON	7:0			CKPS[1:0]			SYNC	RD16	ON
0x0FCF	T1GCON	7:0	GE	GPOL	GTM	GSPM	GGO/DONE	GVAL		
0x0FD0	TMR1GATE	7:0					GSS[3:0]			
0x0FD1	TMR1CLK	7:0					CS[3:0]			

19.14 Register Definitions: Timer1

Long bit name prefixes for the odd numbered timers is shown in the following table. Refer to the *"Long Bit Names"* section for more information.

Table 19-5. Timer1 prefixes

Peripheral	Bit Name Prefix
Timer1	T1
Timer3	T3
Timer5	T5

Related Links

1.4.2.2 Long Bit Names

19.14.1 TxCON

Name: TxCON
Address: 0xFCE,0xFC8,0xFC2

Timer Control Register

Bit	7	6	5	4	3	2	1	0
			CKPS[1:0]			$\overline{\text{SYNC}}$	RD16	ON
Access			R/W	R/W		R/W	R/W	R/W
Reset			0	0		0	0	0

Bits 5:4 – CKPS[1:0] Timer Input Clock Prescale Select bits
Reset States: POR/BOR = 00
All Other Resets = uu

Value	Description
11	1:8 Prescale value
10	1:4 Prescale value
01	1:2 Prescale value
00	1:1 Prescale value

Bit 2 – $\overline{\text{SYNC}}$ Timer External Clock Input Synchronization Control bit
Reset States: POR/BOR = 0
All Other Resets = u

Value	Condition	Description
X	CS = $F_{OSC}/4$ or F_{OSC}	This bit is ignored. Timer uses the incoming clock as is.
1	Else	Do not synchronize external clock input
0	Else	Synchronize external clock input with system clock

Bit 1 – RD16 16-Bit Read/Write Mode Enable bit
Reset States: POR/BOR = 0
All Other Resets = u

Value	Description
1	Enables register read/write of Timer in one 16-bit operation
0	Enables register read/write of Timer in two 8-bit operations

Bit 0 – ON Timer On bit
Reset States: POR/BOR = 0
All Other Resets = u

Value	Description
1	Enables Timer
0	Disables Timer

19.14.2 TxGCON

Name: TxGCON
Address: 0xFCF,0xFC9,0xFC3

Timer Gate Control Register

Bit	7	6	5	4	3	2	1	0
	GE	GPOL	GTM	GSPM	GGO/DONE	GVAL		
Access	R/W	R/W	R/W	R/W	R/W	RO		
Reset	0	0	0	0	0	x		

Bit 7 – GE Timer Gate Enable bit
Reset States: POR/BOR = 0
All Other Resets = u

Value	Condition	Description
1	19.14.1.4 ON = 1	Timer counting is controlled by the Timer gate function
0	19.14.1.4 ON = 1	Timer is always counting
X	19.14.1.4 ON = 0	This bit is ignored

Bit 6 – GPOL Timer Gate Polarity bit
Reset States: POR/BOR = 0
All Other Resets = u

Value	Description
1	Timer gate is active-high (Timer counts when gate is high)
0	Timer gate is active-low (Timer counts when gate is low)

Bit 5 – GTM Timer Gate Toggle Mode bit
Timer Gate Flip-Flop Toggles on every rising edge

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

Value	Description
1	Timer Gate Toggle mode is enabled
0	Timer Gate Toggle mode is disabled and Toggle flip-flop is cleared

Bit 4 – GSPM Timer Gate Single Pulse Mode bit
Reset States: POR/BOR = 0
All Other Resets = u

Value	Description
1	Timer Gate Single Pulse mode is enabled and is controlling Timer gate)
0	Timer Gate Single Pulse mode is disabled

Bit 3 – GGO/$\overline{\text{DONE}}$ Timer Gate Single Pulse Acquisition Status bit
This bit is automatically cleared when TxGSPM is cleared.

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

Value	Description
1	Timer Gate Single Pulse Acquisition is ready, waiting for an edge
0	Timer Gate Single Pulse Acquisition has completed or has not been started.

Bit 2 – GVAL Timer Gate Current State bit
Indicates the current state of the Timer gate that could be provided to TMRxH:TMRxL

Unaffected by Timer Gate Enable (TMRxGE)

19.14.3 TMRxCLK

Name: TMRxCLK
Address: 0xFD1,0xFCB,0xFC5

Timer Clock Source Selection Register

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

Bits 3:0 – CS[3:0] Timer Clock Source Selection bits
Refer to the clock source selection table.

Reset States: POR/BOR = 0000
All Other Resets = uuuu

19.14.4 TMRxGATE

Name: TMRxGATE
Address: 0xFD0,0xFCA,0xFC4

Timer Gate Source Selection Register

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

Bits 3:0 – GSS[3:0] Timer Gate Source Selection bits
Refer to the gate source selection table.

Reset States: POR/BOR = 0000
All Other Resets = uuuu

19.14.5 TMRx

Name: TMRx
Address: 0xFCC,0xFC6,0xFC0

Timer Low Byte Register

Bit	15	14	13	12	11	10	9	8
	TMRxH[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

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

Bits 15:8 – TMRxH[7:0] Timer Most Significant Byte
Reset States: POR/BOR = 00000000
All Other Resets = uuuuuuuu

Bits 7:0 – TMRxL[7:0] Timer Least Significant Byte
Reset States: POR/BOR = 00000000
All Other Resets = uuuuuuuu

20. Timer2 Module

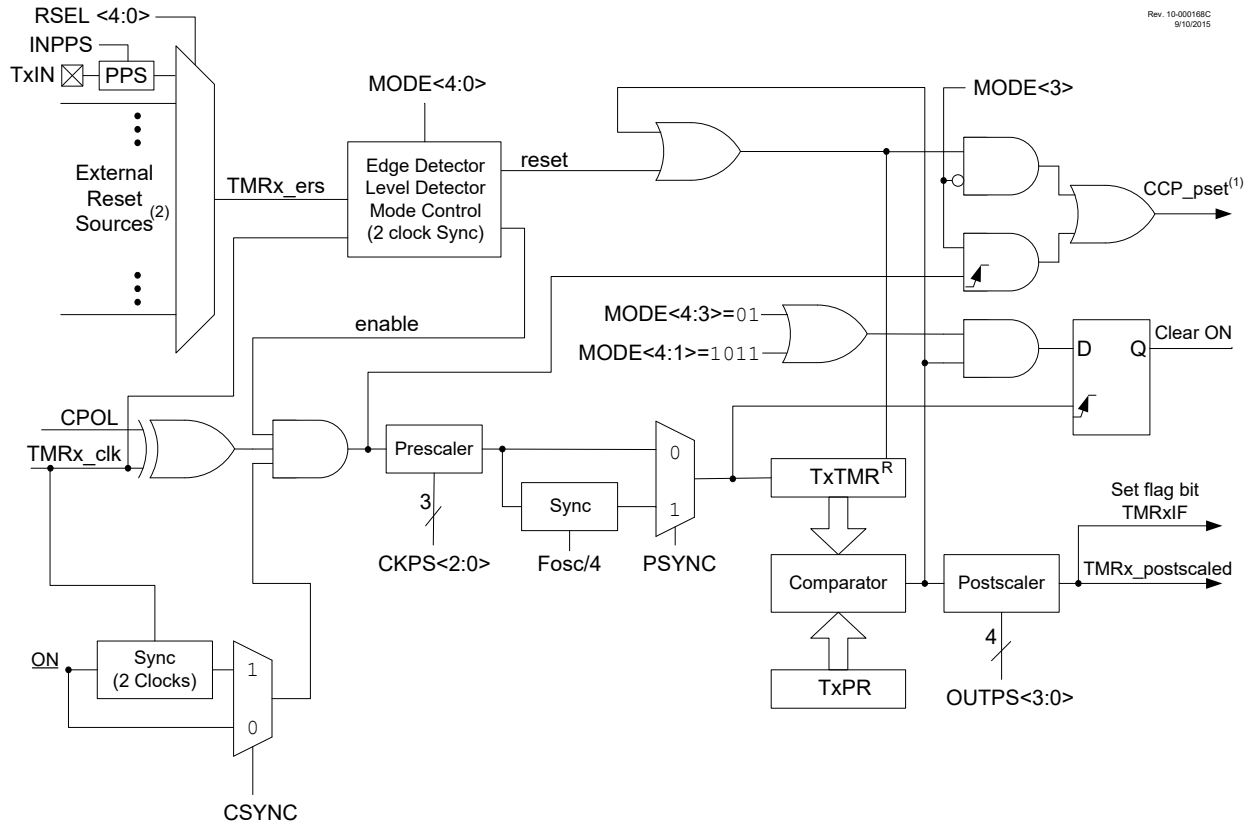
The Timer2 module is a 8-bit timer that incorporates the following features:

- 8-Bit Timer and Period Registers
- Readable and Writable
- Software Programmable Prescaler (1:1 to 1:128)
- Software Programmable Postscaler (1:1 to 1:16)
- Interrupt on T2TMR Match with T2PR
- One-Shot Operation
- Full Asynchronous Operation
- Includes Hardware Limit Timer (HLT)
- Alternate Clock Sources
- External Timer Reset Signal Sources
- Configurable Timer Reset Operation

See Figure 20-1 for a block diagram of Timer2. See table below for the clock source selections.

Important: References to module Timer2 apply to all the even numbered timers on this device. (Timer2, Timer4, etc.)

Figure 20-1. Timer2 with Hardware Limit Timer (HLT) Block Diagram



Note:

1. Signal to the CCP to trigger the PWM pulse.
2. See TxRST for external Reset sources.

Table 20-1. Clock Source Selection

CS<3:0>	Clock Source		
	Timer2	Timer4	Timer6
1111-1001	Reserved	Reserved	Reserved
1000	ZCD_OUT	ZCD_OUT	ZCD_OUT
0111	CLKREF_OUT	CLKREF_OUT	CLKREF_OUT
0110	SOSC	SOSC	SOSC
0101	MFINTOSC (31 kHz)	MFINTOSC (31 kHz)	MFINTOSC (31 kHz)
0100	LFINTOSC	LFINTOSC	LFINTOSC
0011	HFINTOSC	HFINTOSC	HFINTOSC
0010	Fosc	Fosc	Fosc

CS<3:0>	Clock Source		
	Timer2	Timer4	Timer6
0001	Fosc/4	Fosc/4	Fosc/4
0000	Pin selected by T2INPPS	Pin selected by T4INPPS	Pin selected by T6INPPS

20.1 Timer2 Operation

Timer2 operates in three major modes:

- Free Running Period
- One-shot
- Monostable

Within each mode there are several options for starting, stopping, and reset. Table 20-3 lists the options.

In all modes, the T2TMR count register is incremented on the rising edge of the clock signal from the programmable prescaler. When T2TMR equals T2PR, a high level is output to the postscaler counter. T2TMR is cleared on the next clock input.

An external signal from hardware can also be configured to gate the timer operation or force a T2TMR count Reset. In Gate modes the counter stops when the gate is disabled and resumes when the gate is enabled. In Reset modes the T2TMR count is reset on either the level or edge from the external source.

The T2TMR and T2PR registers are both directly readable and writable. The T2TMR register is cleared and the T2PR register initializes to FFh on any device Reset. Both the prescaler and postscaler counters are cleared on the following events:

- A write to the T2TMR register
- A write to the T2CON register
- Any device Reset
- External Reset Source event that resets the timer.

Important: T2TMR is not cleared when T2CON is written.

20.1.1 Free Running Period Mode

The value of T2TMR is compared to that of the Period register, T2PR, on each clock cycle. When the two values match, the comparator resets the value of T2TMR to 00h on the next cycle and increments the output postscaler counter. When the postscaler count equals the value in the OUTPS bits of the T2CON register then a one clock period wide pulse occurs on the TMR2_postscaled output, and the postscaler count is cleared.

20.1.2 One-Shot Mode

The One-Shot mode is identical to the Free Running Period mode except that the ON bit is cleared and the timer is stopped when T2TMR matches T2PR and will not restart until the ON bit is cycled off and on. Postscaler (OUTPS) values other than zero are ignored in this mode because the timer is stopped at the first period event and the postscaler is reset when the timer is restarted.

20.1.3 Monostable Mode

Monostable modes are similar to One-Shot modes except that the ON bit is not cleared and the timer can be restarted by an external Reset event.

20.2 Timer2 Output

The Timer2 module's primary output is TMR2_postscaled, which pulses for a single TMR2_clk period upon each match of the postscaler counter and the OUTPS bits of the T2CON register. The postscaler is incremented each time the T2TMR value matches the T2PR value. This signal can be selected as an input to several other input modules:

- The ADC module, as an auto-conversion trigger
- CWG, as an auto-shutdown source
- The CRC memory scanner, as a trigger for triggered mode
- Gate source for odd numbered timers (Timer1, Timer3, etc.)
- Alternate SPI clock
- Reset signals for other instances of even numbered timers (Timer2, Timer4, etc.)

In addition, the Timer2 is also used by the CCP module for pulse generation in PWM mode. See *"PWM Overview"* and *"Pulse-width Modulation"* sections for more details on setting up Timer2 for use with the CCP and PWM modules.

Related Links

21.4 PWM Overview

22. (PWM) Pulse-Width Modulation

20.3 External Reset Sources

In addition to the clock source, the Timer2 also takes in an external Reset source. This external Reset source is selected for each timer with the corresponding TxRST register. This source can control starting and stopping of the timer, as well as resetting the timer, depending on which mode the timer is in. Reset source selections are shown in the following table.

Table 20-2. External Reset Sources

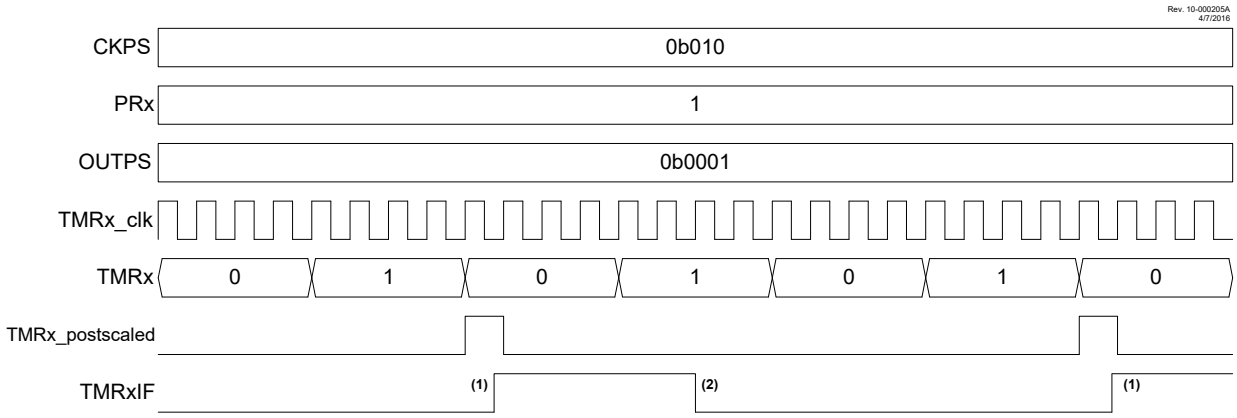
RSEL<3:0>	Reset Source		
	TMR2	TMR4	TMR6
1011-1111	Reserved	Reserved	Reserved
1010	ZCD_OUT	ZCD_OUT	ZCD_OUT
1001	CMP2OUT	CMP2OUT	CMP2OUT
1000	CMP1OUT	CMP1OUT	CMP1OUT
0111	PWM4OUT	PWM4OUT	PWM4OUT
0110	PWM3OUT	PWM3OUT	PWM3OUT
0101	CCP2OUT	CCP2OUT	CCP2OUT
0100	CCP1OUT	CCP1OUT	CCP1OUT

RSEL<3:0>	Reset Source		
	TMR2	TMR4	TMR6
0011	TMR6 post-scaled	TMR6 post-scaled	Reserved
0010	TMR4 post-scaled	Reserved	TMR4 post-scaled
0001	Reserved	TMR2 post-scaled	TMR2 post-scaled
0000	Pin selected by T2INPPS	Pin selected by T4INPPS	Pin selected by T6INPPS

20.4 Timer2 Interrupt

Timer2 can also generate a device interrupt. The interrupt is generated when the postscaler counter matches with the selected postscaler value (OUTPS bits of T2CON register). The interrupt is enabled by setting the TMR2IE interrupt enable bit. Interrupt timing is illustrated in the figure below.

Figure 20-2. Timer2 Prescaler, Postscaler, and Interrupt Timing Diagram



Note:

1. Setting the interrupt flag is synchronized with the instruction clock.
2. Cleared by software.

20.5 Operating Modes

The mode of the timer is controlled by the MODE bits of the T2HLT register. Edge-Triggered modes require six Timer clock periods between external triggers. Level-Triggered modes require the triggering level to be at least three Timer clock periods long. External triggers are ignored while in Debug mode.

Table 20-3. Operating Modes Table

Mode	MODE<4:0>		Output Operation	Operation	Timer Control		
	<4:3>	<2:0>			Start	Reset	Stop
Free Running Period	00	000	Period Pulse	Software gate (Figure 20-3)	ON = 1	—	ON = 0
		001		Hardware gate, active-high	ON = 1 and TMRx_ers = 1	—	ON = 0 or TMRx_ers = 0

Mode	MODE<4:0>		Output Operation	Operation	Timer Control		
	<4:3>	<2:0>			Start	Reset	Stop
				(Figure 20-4)			
		010		Hardware gate, active-low	ON = 1 and TMRx_ers = 0	—	ON = 0 or TMRx_ers = 1
		011	Period Pulse with Hardware Reset	Rising or falling edge Reset	ON = 1	TMRx_ers ↕	ON = 0
		100		Rising edge Reset (Figure 20-5)		TMRx_ers ↑	
		101		Falling edge Reset		TMRx_ers ↓	
		110		Low level Reset		TMRx_ers = 0	ON = 0 or TMRx_ers = 0
		111		High level Reset (Figure 20-6)		TMRx_ers = 1	ON = 0 or TMRx_ers = 1
One-shot	01	000	One-shot	Software start (Figure 20-7)	ON = 1	—	ON = 0 or Next clock after TMRx = PRx (Note 2)
		001	Edge Triggered Start (Note 1)	Rising edge start (Figure 20-8)	ON = 1 and TMRx_ers ↑	—	
		010		Falling edge start	ON = 1 and TMRx_ers ↓	—	
		011		Any edge start	ON = 1 and TMRx_ers ↕	—	
		100	Edge Triggered Start and Hardware Reset (Note 1)	Rising edge start and Rising edge Reset (Figure 20-9)	ON = 1 and TMRx_ers ↑	TMRx_ers ↑	
		101		Falling edge start and Falling edge Reset	ON = 1 and TMRx_ers ↓	TMRx_ers ↓	
		110		Rising edge start and	ON = 1 and	TMRx_ers = 0	

Mode	MODE<4:0>		Output Operation	Operation	Timer Control		
	<4:3>	<2:0>			Start	Reset	Stop
				Low level Reset (Figure 20-10)	TMRx_ers ↑		
		111		Falling edge start and High level Reset	ON = 1 and TMRx_ers ↓	TMRx_ers = 1	
Mono-stable	10	000	Reserved				
		001	Edge Triggered Start (Note 1)	Rising edge start (Figure 20-11)	ON = 1 and TMRx_ers ↑	—	ON = 0 or Next clock after TMRx = PRx (Note 3)
		010		Falling edge start	ON = 1 and TMRx_ers ↓	—	
		011		Any edge start	ON = 1 and TMRx_ers ↕	—	
Reserved		100	Reserved				
Reserved		101	Reserved				
One-shot		110	Level Triggered Start and Hardware Reset	High level start and Low level Reset (Figure 20-12)	ON = 1 and TMRx_ers = 1	TMRx_ers = 0	ON = 0 or Held in Reset (Note 2)
		111		Low level start & High level Reset	ON = 1 and TMRx_ers = 0	TMRx_ers = 1	
Reserved	11	xxx	Reserved				

Note:

1. If ON = 0 then an edge is required to restart the timer after ON = 1.
2. When T2TMR = T2PR then the next clock clears ON and stops T2TMR at 00h.
3. When T2TMR = T2PR then the next clock stops T2TMR at 00h but does not clear ON.

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 T2CON register are cleared).

- The diagrams illustrate any clock except $F_{OSC}/4$ and show clock-sync delays of at least two full cycles for both ON and Timer2_ers. When using $F_{OSC}/4$, 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 the "*PWM Overview*" section. The signals are not a part of the Timer2 module.

Related Links

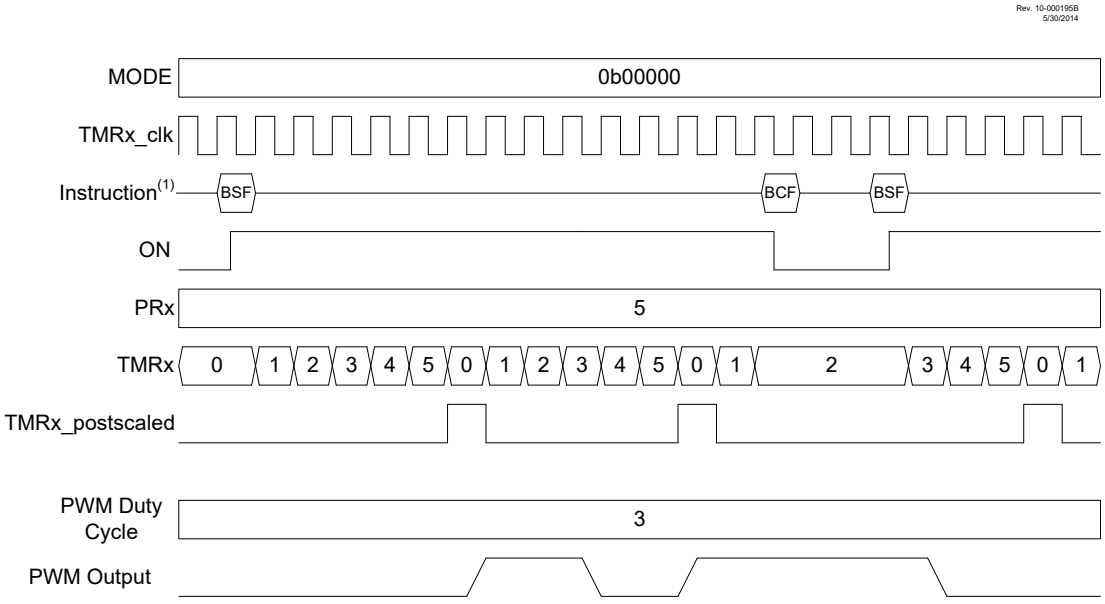
21.4 PWM Overview

22. (PWM) Pulse-Width Modulation

20.6.1 Software Gate Mode

This mode corresponds to legacy Timer2 operation. The timer increments with each clock input when ON = `1` and does not increment when ON = `0`. When the 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 20-3. With PRx = 5, the counter advances until TMRx = 5, and goes to zero with the next clock.

Figure 20-3. Software Gate Mode Timing Diagram (MODE = 00000)



Note:

1. BSF and BCF represent Bit-Set File and Bit-Clear File instructions executed by the CPU to set or clear the ON bit of TxCON. CPU execution is asynchronous to the timer clock input.

Related Links

21.4 PWM Overview

22. (PWM) Pulse-Width Modulation

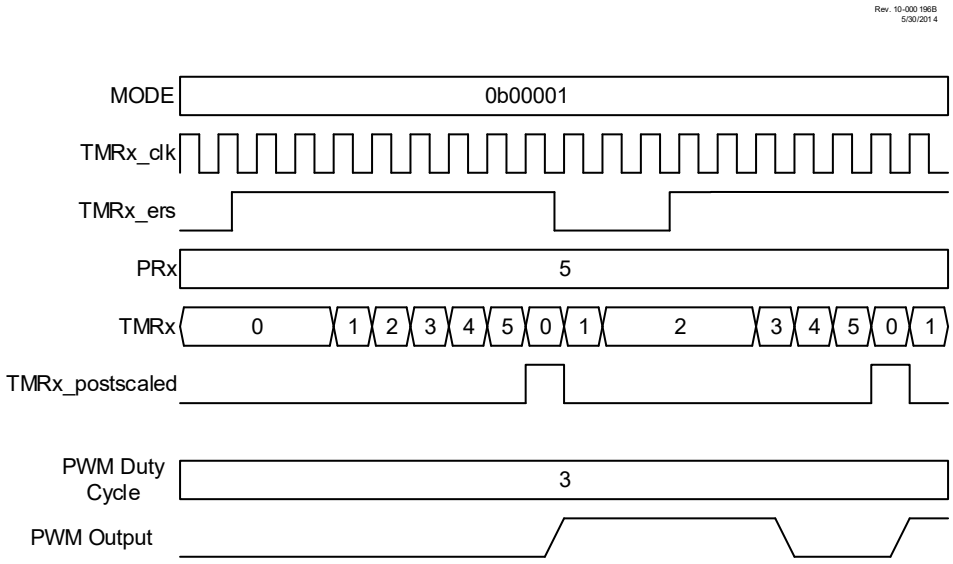
20.6.2 Hardware Gate Mode

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, then the timer is stopped when the external signal is low.

Figure 20-4 illustrates the Hardware Gating mode for MODE<4:0> = 00001 in which a high input level starts the counter.

Figure 20-4. Hardware Gate Mode Timing Diagram (MODE = 00001)



Related Links

21.4 PWM Overview

22. (PWM) Pulse-Width Modulation

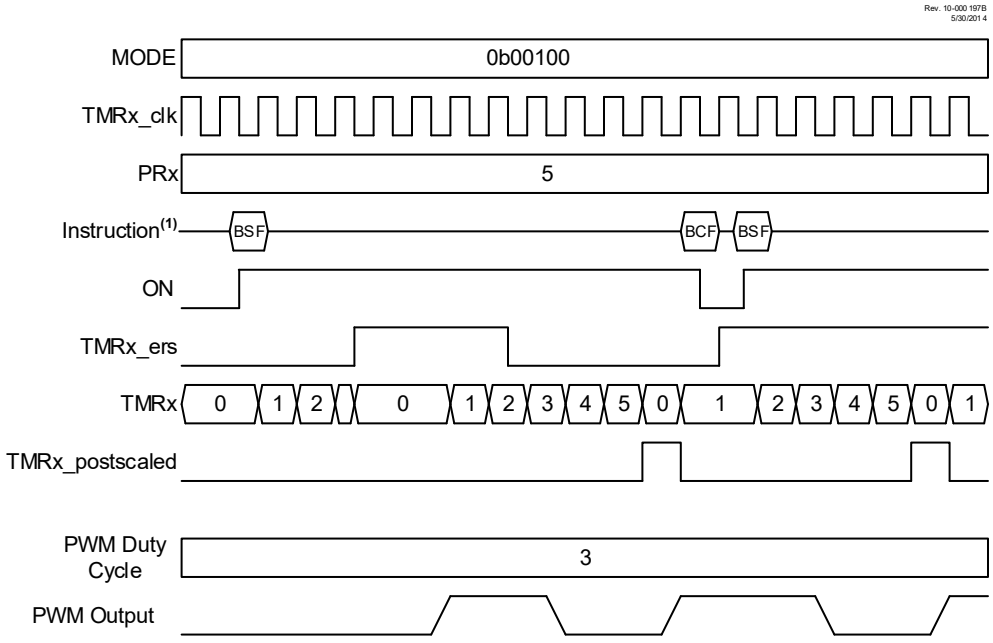
20.6.3 Edge-Triggered Hardware Limit Mode

In Hardware Limit mode, the timer can be reset by the TMRx_ers external signal before the timer reaches the period count. Three types of Resets are possible:

- Reset on rising or falling edge (MODE<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 20-5.

Figure 20-5. Edge-Triggered Hardware Limit Mode Timing Diagram (MODE = 00100)



Note:

1. BSF and BCF represent Bit-Set File and Bit-Clear File instructions executed by the CPU to set or clear the ON bit of TxCON. CPU execution is asynchronous to the timer clock input.

Related Links

21.4 PWM Overview

22. (PWM) Pulse-Width Modulation

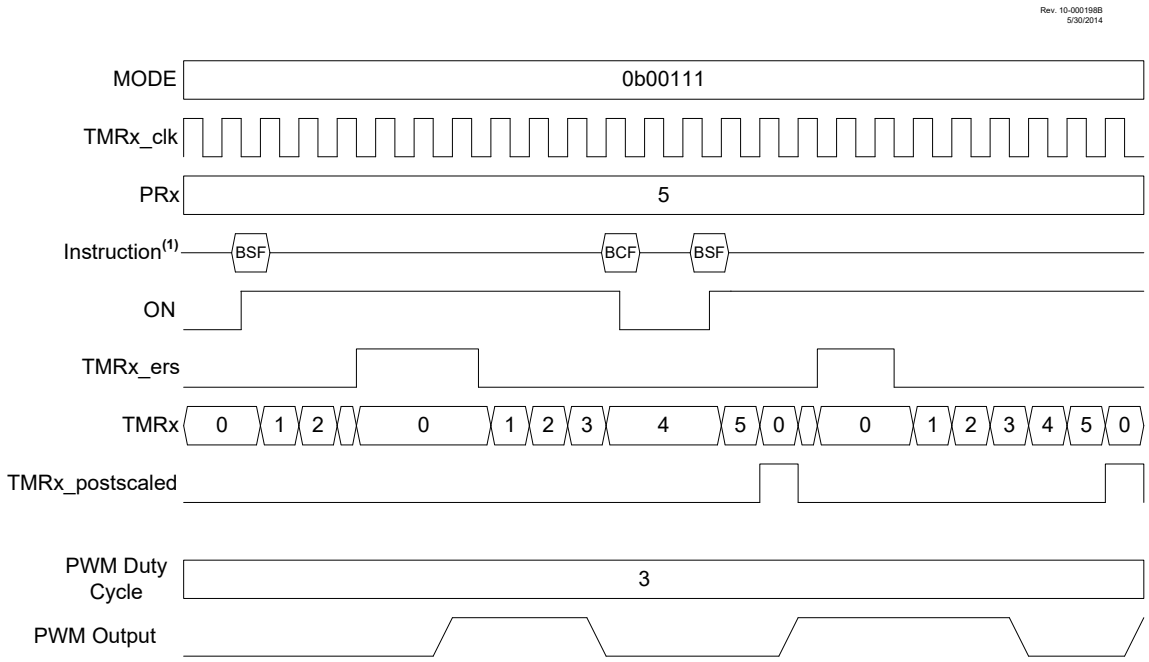
20.6.4 Level-Triggered Hardware Limit Mode

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 20-6. 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 the PWM output will be set high when the timer starts counting and then set low only when the timer count matches the CCPRx value. The timer is reset when either the timer count matches the PRx value or two clock periods after the external Reset signal goes true and stays true.

The timer starts counting, and the PWM output is set high, on either the clock following the PRx match or two clocks after the external Reset signal relinquishes the Reset. The PWM output will remain high until the timer counts up to match the CCPRx pulse width value. If the external Reset signal goes true while the PWM output is high then the PWM output will remain high until the Reset signal is released allowing the timer to count up to match the CCPRx value.

Figure 20-6. Level-Triggered Hardware Limit Mode Timing Diagram (MODE = 00111)



Note:

1. BSF and BCF represent Bit-Set File and Bit-Clear File instructions executed by the CPU to set or clear the ON bit of TxCON. CPU execution is asynchronous to the timer clock input.

Related Links

21.4 PWM Overview

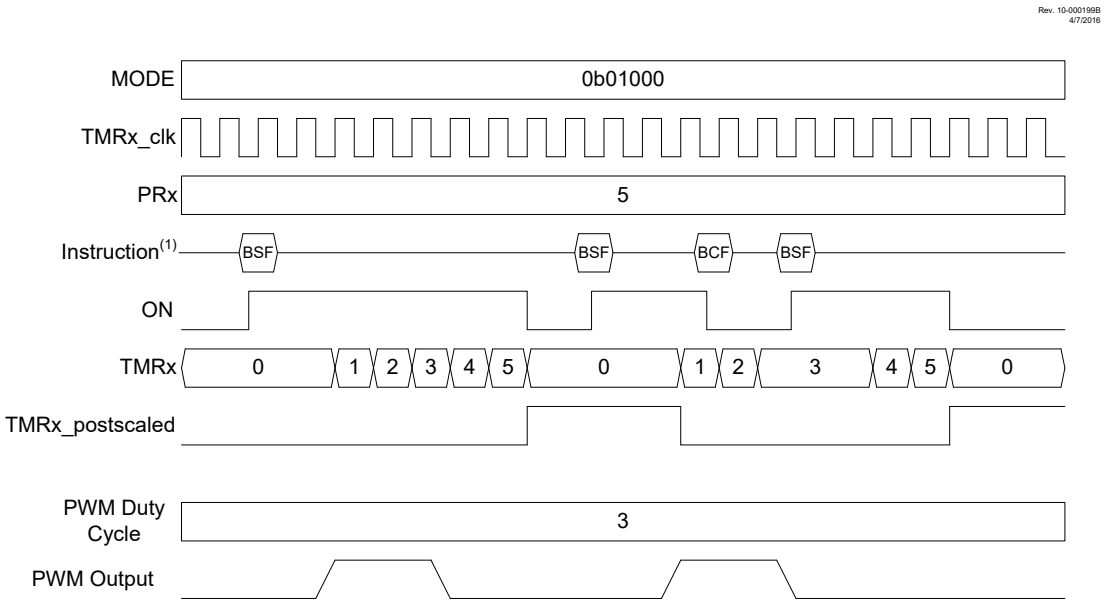
22. (PWM) Pulse-Width Modulation

20.6.5 Software Start One-Shot Mode

In One-Shot mode the timer resets and the ON bit is cleared when the timer value matches the 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 20-7. In the example, ON is controlled by BSF and BCF instructions. In the first case, a BSF instruction sets ON and the counter runs to completion and clears ON. In the second case, a BSF instruction starts the cycle, 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 20-7. Software Start One-shot Mode Timing Diagram (MODE = 01000)



Note:

1. BSF and BCF represent Bit-Set File and Bit-Clear File instructions executed by the CPU to set or clear the ON bit of TxCON. CPU execution is asynchronous to the timer clock input.

Related Links

21.4 PWM Overview

22. (PWM) Pulse-Width Modulation

20.6.6 Edge-Triggered One-Shot Mode

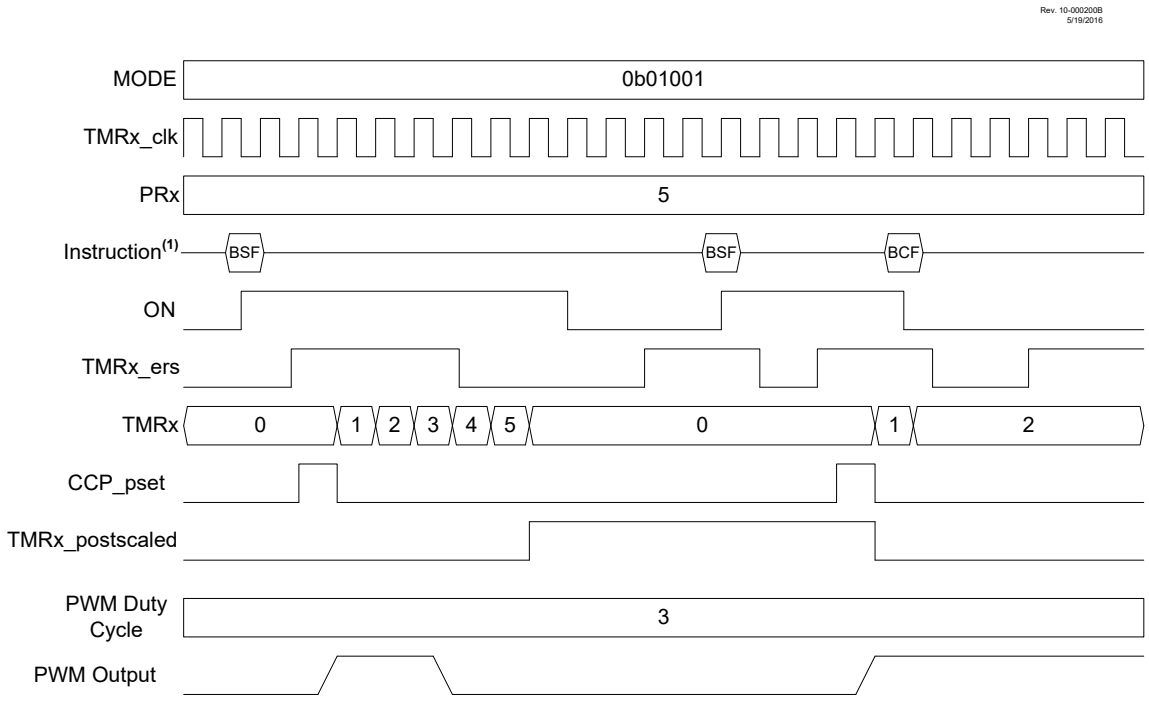
The Edge-Triggered One-Shot modes start the timer on an edge from the external signal input, after the ON bit is set, and clear the ON bit when the timer matches the PRx period value. The following edges will start the timer:

- 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 20-8 illustrates operation in the rising edge One-Shot mode.

When Edge-Triggered One-Shot mode is used in conjunction with the CCP then the edge-trigger will activate the PWM drive and the PWM drive will deactivate when the timer matches the CCPRx pulse width value and stay deactivated when the timer halts at the PRx period count match.

Figure 20-8. Edge-Triggered One-Shot Mode Timing Diagram (MODE = 01001)



Note:

1. BSF and BCF represent Bit-Set File and Bit-Clear File instructions executed by the CPU to set or clear the ON bit of TxCON. CPU execution is asynchronous to the timer clock input.

Related Links

21.4 PWM Overview

22. (PWM) Pulse-Width Modulation

20.6.7 Edge-Triggered Hardware Limit One-Shot Mode

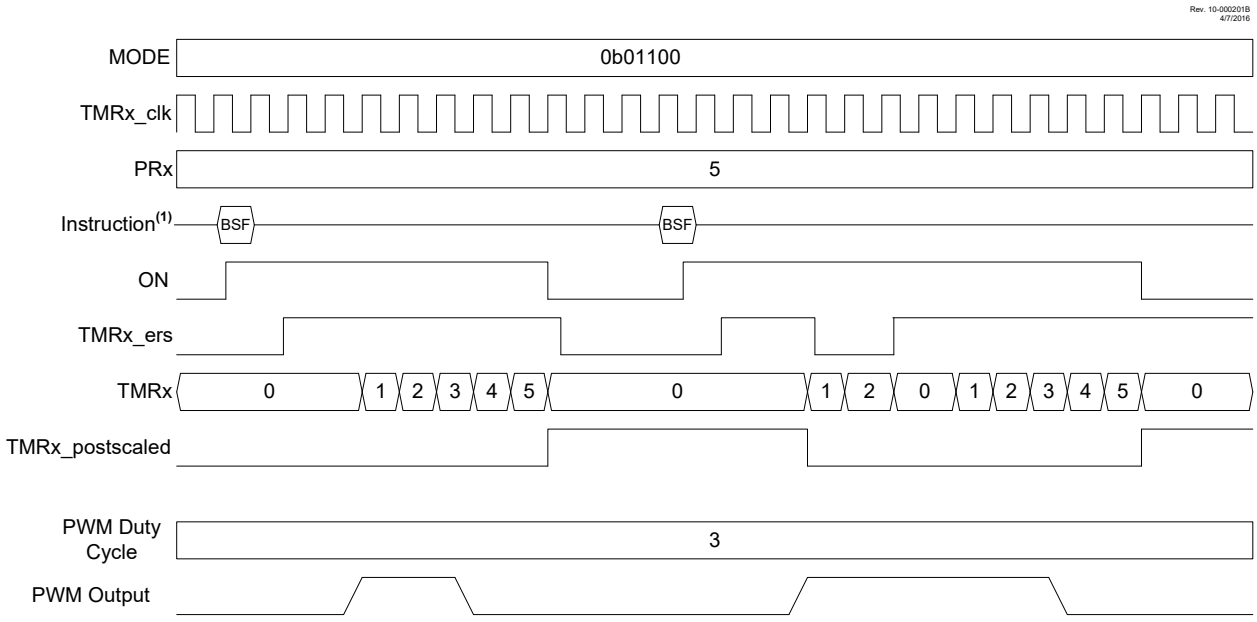
In Edge-Triggered Hardware Limit One-Shot modes the timer starts on the first external signal edge after the ON bit is set and resets on all subsequent edges. Only the first edge after the ON bit is set is needed to start the timer. The counter will resume counting automatically two clocks after all 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 20-9 illustrates the rising edge hardware limit one-shot operation.

When this mode is used in conjunction with the CCP then the first starting edge trigger, and all subsequent Reset edges, will activate the PWM drive. The PWM drive will deactivate when the timer matches the CCPRx pulse-width value and stay deactivated until the timer halts at the PRx period match unless an external signal edge resets the timer before the match occurs.

Figure 20-9. Edge-Triggered Hardware Limit One-Shot Mode Timing Diagram (MODE = 01100)



Note:

1. BSF and BCF represent Bit-Set File and Bit-Clear File instructions executed by the CPU to set or clear the ON bit of TxCON. CPU execution is asynchronous to the timer clock input.

Related Links

21.4 PWM Overview

22. (PWM) Pulse-Width Modulation

20.6.8 Level Reset, Edge-Triggered Hardware Limit One-Shot Modes

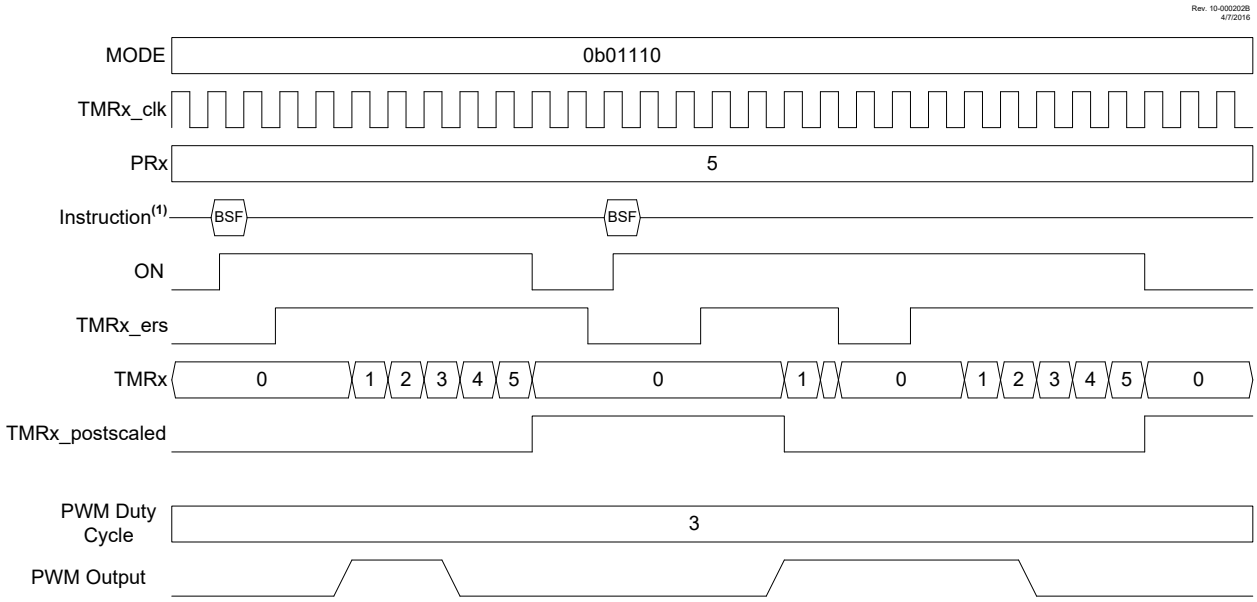
In Level -Triggered One-Shot mode the timer count is reset on the external signal level and starts counting on the rising/falling edge of the transition from Reset level to the active level while the ON bit is set. Reset levels are selected as follows:

- Low Reset level (MODE<4:0> = 01110)
- High Reset level (MODE<4:0> = 01111)

When the timer count matches the PRx period count, 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.

Figure 20-10. Low Level Reset, Edge-Triggered hardware Limit one-Shot Mode Timing Diagram (MODE = 01110)



Note:

1. BSF and BCF represent Bit-Set File and Bit-Clear File instructions executed by the CPU to set or clear the ON bit of TxCON. CPU execution is asynchronous to the timer clock input.

Related Links

21.4 PWM Overview

22. (PWM) Pulse-Width Modulation

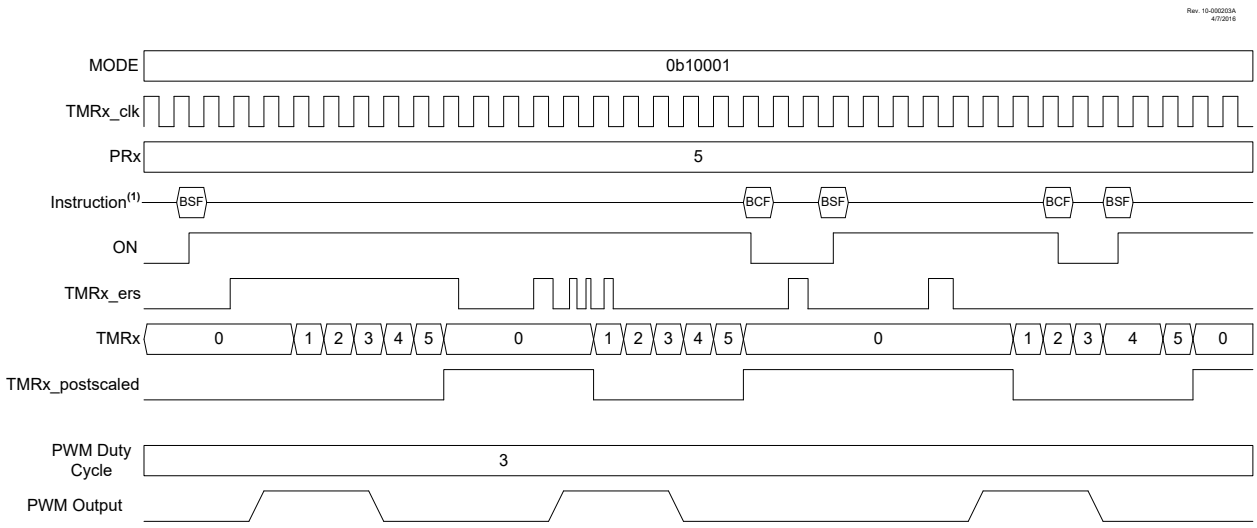
20.6.9 Edge-Triggered Monostable Modes

The Edge-Triggered Monostable modes start the timer on an edge from the external Reset signal input, after the ON bit is set, and stop incrementing the timer when the timer matches the PRx period value. The following edges will start the timer:

- Rising edge (MODE<4:0> = 10001)
- Falling edge (MODE<4:0> = 10010)
- Rising or Falling edge (MODE<4:0> = 10011)

When an Edge-Triggered Monostable mode is used in conjunction with the CCP PWM operation, the PWM drive goes active with the external Reset signal edge that starts the timer, but will not go active when the timer matches the PRx value. While the timer is incrementing, additional edges on the external Reset signal will not affect the CCP PWM.

Figure 20-11. Rising Edge-Triggered Monostable Mode Timing Diagram (MODE = 10001)



Note:

1. BSF and BCF represent Bit-Set File and Bit-Clear File instructions executed by the CPU to set or clear the ON bit of TxCON. CPU execution is asynchronous to the timer clock input.

Related Links

21.4 PWM Overview

22. (PWM) Pulse-Width Modulation

20.6.10 Level-Triggered Hardware Limit One-Shot Modes

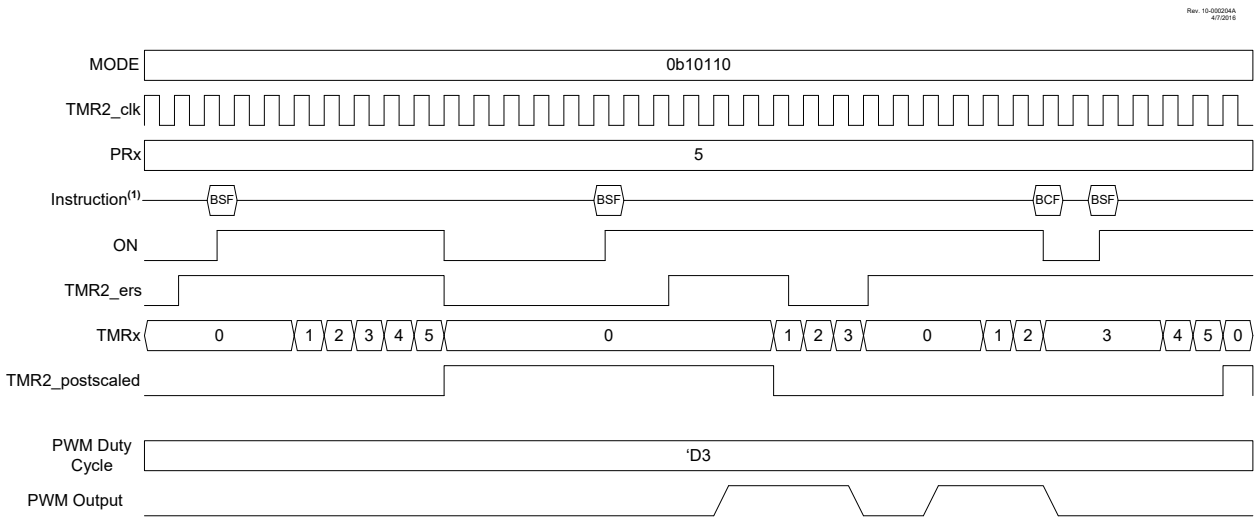
The Level-Triggered Hardware Limit One-Shot modes hold the timer in Reset on an external Reset level and start counting when both the ON bit is set and the external signal is not at the Reset level. If one of either the external signal is not in Reset or the ON bit is set, then the other signal being set/made active will start the timer. Reset levels are selected as follows:

- Low Reset level (MODE<4:0> = 10110)
- High Reset level (MODE<4:0> = 10111)

When the timer count matches the PRx period count, 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, the timer will stay in Reset until both the ON bit is set and the external signal is not at the Reset level.

When Level-Triggered Hardware Limit One-Shot modes are used in conjunction with the CCP PWM operation, the PWM drive goes active with either the external signal edge or the setting of the ON bit, whichever of the two starts the timer.

Figure 20-12. Level-Triggered hardware Limit one-Shot Mode Timing Diagram (MODE = 10110)



Note:

1. BSF and BCF represent Bit-Set File and Bit-Clear File instructions executed by the CPU to set or clear the ON bit of TxCON. CPU execution is asynchronous to the timer clock input.

Related Links

21.4 PWM Overview

22. (PWM) Pulse-Width Modulation

20.7 Timer2 Operation During Sleep

When PSYNC = 1, Timer2 cannot be operated while the processor is in Sleep mode. The contents of the T2TMR and T2PR registers will remain unchanged while processor is in Sleep mode.

When PSYNC = 0, Timer2 will operate in Sleep as long as the clock source selected is also still running. If any internal oscillator is selected as the clock source, it will stay active during Sleep mode.

20.8 Register Summary - Timer2

Address	Name	Bit Pos.								
0x0FAE	T6TMR	7:0	TxTMR[7:0]							
0x0FAF	T6PR	7:0	TxPR[7:0]							
0x0FB0	T6CON	7:0	ON		CKPS[2:0]			OUTPS[3:0]		
0x0FB1	T6HLT	7:0	PSYNC	CPOL	CSYNC		MODE[4:0]			
0x0FB2	T6CLKCON	7:0						CS[3:0]		
0x0FB3	T6RST	7:0						RSEL[3:0]		
0x0FB4	T4TMR	7:0	TxTMR[7:0]							
0x0FB5	T4PR	7:0	TxPR[7:0]							
0x0FB6	T4CON	7:0	ON		CKPS[2:0]			OUTPS[3:0]		
0x0FB7	T4HLT	7:0	PSYNC	CPOL	CSYNC		MODE[4:0]			
0x0FB8	T4CLKCON	7:0						CS[3:0]		
0x0FB9	T4RST	7:0						RSEL[3:0]		
0x0FBA	T2TMR	7:0	TxTMR[7:0]							
0x0FBB	T2PR	7:0	TxPR[7:0]							
0x0FBC	T2CON	7:0	ON		CKPS[2:0]			OUTPS[3:0]		
0x0FBD	T2HLT	7:0	PSYNC	CPOL	CSYNC		MODE[4:0]			
0x0FBE	T2CLKCON	7:0						CS[3:0]		
0x0FBF	T2RST	7:0						RSEL[3:0]		

20.9 Register Definitions: Timer2 Control

Long bit name prefixes for the Timer2 peripherals are shown in table below. Refer to Section "*Long Bit Names*" for more information.

Table 20-4. Timer2 long bit name prefixes

Peripheral	Bit Name Prefix
Timer2	T2
Timer4	T4
Timer6	T6



Notice: References to module Timer2 apply to all the even numbered timers on this device. (Timer2, Timer4, etc.)

Related Links

1.4.2.2 Long Bit Names

20.9.1 TxTMR

Name: TxTMR
Address: 0xFBA,0xFB4,0xFAE

Timer Counter Register

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

Bits 7:0 – TxTMR[7:0] Timerx Counter bits

20.9.2 TxPR

Name: TxPR
Address: 0xFBB,0xFB5,0xFAF

Timer Period Register

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

Bits 7:0 – TxPR[7:0] Timer Period Register bits

Value	Description
0 – 255	The timer restarts at '0' when TxTMR reaches TxPR value

20.9.3 TxCON

Name: TxCON
Address: 0xFBC,0xFB6,0xFB0

Timerx Control Register

Bit	7	6	5	4	3	2	1	0
	ON	CKPS[2:0]			OUTPS[3:0]			
Access	R/W/HC	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 7 – ON
Timer On bit(1)

Value	Description
1	Timer is on
0	Timer is off: all counters and state machines are reset

Bits 6:4 – CKPS[2:0] Timer Clock Prescale Select bits

Value	Description
111	1:128 Prescaler
110	1:64 Prescaler
101	1:32 Prescaler
100	1:16 Prescaler
011	1:8 Prescaler
010	1:4 Prescaler
001	1:2 Prescaler
000	1:1 Prescaler

Bits 3:0 – OUTPS[3:0] Timer Output Postscaler Select bits

Value	Description
1111	1:16 Postscaler
1110	1:15 Postscaler
1101	1:14 Postscaler
1100	1:13 Postscaler
1011	1:12 Postscaler
1010	1:11 Postscaler
1001	1:10 Postscaler
1000	1:9 Postscaler
0111	1:8 Postscaler
0110	1:7 Postscaler
0101	1:6 Postscaler
0100	1:5 Postscaler
0011	1:4 Postscaler
0010	1:3 Postscaler

Value	Description
0001	1:2 Postscaler
0000	1:1 Postscaler

Note:

1. In certain modes, the ON bit will be auto-cleared by hardware. See Table 20-3.

20.9.4 TxHLT

Name: TxHLT
Address: 0xFBD,0xFB7,0xFB1

Timer Hardware Limit Control Register

Bit	7	6	5	4	3	2	1	0
	PSYNC	CPOL	CSYNC	MODE[4:0]				
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 7 – PSYNC
Timer Prescaler Synchronization Enable bit(1, 2)

Value	Description
1	Timer Prescaler Output is synchronized to $F_{OSC}/4$
0	Timer Prescaler Output is not synchronized to $F_{OSC}/4$

Bit 6 – CPOL
Timer Clock Polarity Selection bit(3)

Value	Description
1	Falling edge of input clock clocks timer/prescaler
0	Rising edge of input clock clocks timer/prescaler

Bit 5 – CSYNC
Timer Clock Synchronization Enable bit(4, 5)

Value	Description
1	ON bit is synchronized to timer clock input
0	ON bit is not synchronized to timer clock input

Bits 4:0 – MODE[4:0]
Timer Control Mode Selection bits(6, 7)

Value	Description
00000 to 11111	See Table 20-3

Note:

1. Setting this bit ensures that reading TxTMR will return a valid data value.
2. When this bit is '1', Timer 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 input clocks after the ON bit is set.
6. Unless otherwise indicated, all modes start upon ON = 1 and stop upon ON = 0 (stops occur without affecting the value of TxTMR).

7. When TxTMR = TxPR, the next clock clears TxTMR, regardless of the operating mode.

20.9.5 TxCLKCON

Name: TxCLKCON
Address: 0xFBE,0xFB8,0xFB2

Timer Clock Source Selection Register

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

Bits 3:0 – CS[3:0] Timer Clock Source Selection bits

Value	Description
n	See Clock Source Selection table

20.9.6 TxRST

Name: TxRST
Address: 0xFBF,0xFB9,0xFB3

Timer External Reset Signal Selection Register

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

Bits 3:0 – RSEL[3:0]
External Reset Source Selection Bits

Value	Description
n	See External Reset Sources table

21. 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 two standard Capture/Compare/PWM modules (CCP1 and CCP2). 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 CCPTMRS register. The default timer selection is TMR1 when using Capture/Compare mode and T2TMR when using PWM mode in the CCPx module.

It should be noted that the Capture/Compare mode operation is described with respect to TMR1 and the PWM mode operation is described with respect to T2TMR in the following sections.

The Capture and Compare functions are identical for all CCP modules.

Important:

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 6 that vary with the selected mode. Various timers are available to the CCP modules in Capture, Compare or PWM modes, as shown in the table below.

Table 21-1. CCP Mode - Timer Resources

CCP Mode	Timer Resource
Capture	Timer1, Timer3 or Timer5
Compare	
PWM	Timer2, Timer4 or Timer6

The assignment of a particular timer to a module is determined by the timer to CCP enable bits in the CCPTMRS register. 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 Capture Mode

Capture mode makes use of the 16-bit odd numbered timer resources (Timer1, Timer3, etc.). When an event occurs on the capture source, the 16-bit CCPRx register captures and stores the 16-bit value of the TMRx register. An event is defined as one of the following and is configured by the MODE bits:

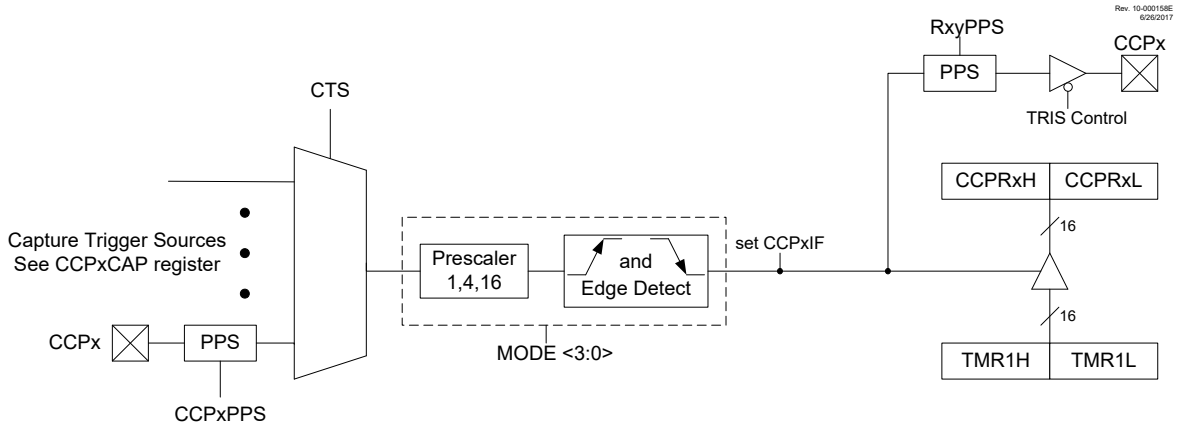
- Every falling edge of CCPx input
- Every rising edge of CCPx input
- Every 4th rising edge of CCPx input
- Every 16th rising edge of CCPx input
- Every edge of CCPx input (rising or falling)

When a capture is made, the Interrupt Request Flag bit CCPxIF of the PIRx register is set. The interrupt flag must be cleared in software. If another capture occurs before the value in the CCPRx register is read, the old captured value is overwritten by the new captured value.

Important: If an event occurs during a 2-byte read, the high and low-byte data will be from different events. It is recommended while reading the CCPRxH:CCPRxL register pair to either disable the module or read the register pair twice for data integrity.

The following figure shows a simplified diagram of the capture operation.

Figure 21-1. Capture Mode Operation Block Diagram



21.2.1 Capture Sources

In Capture mode, the CCPx pin should be configured as an input by setting the associated TRIS control bit.

Important: If the CCPx pin is configured as an output, a write to the port can cause a capture condition.

The capture source is selected by configuring the CTS bits as shown in the following table:

Table 21-2. Capture Trigger Sources

CTS	Source
11	IOC Interrupt
10	CMP2_output
01	CMP1_output
00	Pin selected by CCPxPPS

21.2.2 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 *"Timer1 Module with Gate Control"* for more information on configuring Timer1.

Related Links

19. Timer1 Module with Gate Control

21.2.3 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 PIEx register clear to avoid false interrupts. Additionally, the user should clear the CCPxIF interrupt flag bit of the PIRx register following any change in Operating mode.

Important: Clocking Timer1 from the system clock (F_{OSC}) 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 ($F_{OSC}/4$) or from an external clock source.

21.2.4 CCP Prescaler

There are four prescaler settings specified by the MODE bits. Whenever the CCP module is turned off, or the CCP module is not in Capture mode, the prescaler counter is cleared. Any Reset will clear the prescaler counter.

Switching from one capture prescaler to another does not clear the prescaler and may generate a false interrupt. To avoid this unexpected operation, turn the module off by clearing the CCPxCON register before changing the prescaler. The example below demonstrates the code to perform this function.

Example 21-1. Changing Between Capture Prescalers

```
BANKSEL CCP1CON        ;(only needed when CCP1CON is not in ACCESS space)
CLRF    CCP1CON        ;Turn CCP module off
```

```
MOVLW   NEW_CAPT_PS    ;CCP ON and Prescaler select → W
MOVWF   CCP1CON        ;Load CCP1CON with this value
```

21.2.5 Capture During Sleep

Capture mode depends upon the Timer1 module for proper operation. There are two options for driving the Timer1 module in Capture mode. It can be driven by the instruction clock ($F_{OSC}/4$), or by an external clock source.

When Timer1 is clocked by $F_{OSC}/4$, Timer1 will not increment during Sleep. When the device wakes from Sleep, Timer1 will continue from its previous state.

Capture mode will operate during Sleep when Timer1 is clocked by an external clock source.

21.3 Compare Mode

The Compare mode function described in this section is available and identical for all CCP modules.

Compare mode makes use of the 16-bit odd numbered Timer resources (Timer1, Timer3, etc.). The 16-bit value of the CCPRx register is constantly compared against the 16-bit value of the TMRx register. When a match occurs, one of the following events can occur:

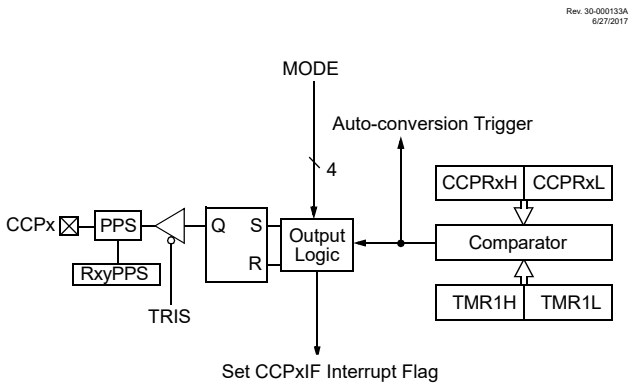
- Toggle the CCPx output and clear TMRx
- Toggle the CCPx output without clearing TMRx
- Set the CCPx output
- Clear the CCPx output
- Pulse output
- Pulse output and clear TMRx

The action on the pin is based on the value of the MODE control bits. 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 = `'0001'` or `'1011'`, the CCP resets the TMRx register.

The following figure shows a simplified diagram of the compare operation.

Figure 21-2. Compare Mode Operation Block Diagram



21.3.1 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 *"Peripheral Pin Select (PPS) Module"* for more details.

The CCP output can also be used as an input for other peripherals.

Important: Clearing the CCPxCON register will force the CCPx compare output latch to the default low level. This is not the PORT I/O data latch.

Related Links

17. (PPS) Peripheral Pin Select Module

21.3.2 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 "Timer1 Module with Gate Control" for more information on configuring Timer1.

Important: Clocking Timer1 from the system clock (F_{OSC}) 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 ($F_{OSC}/4$) or from an external clock source.

21.3.3 Auto-Conversion Trigger

All CCPx modes set the CCP Interrupt Flag (CCPxIF). 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 "Auto-Conversion Trigger" for more information.

Important: 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.

Related Links

31.2.6 Auto-Conversion Trigger

21.3.4 Compare During Sleep

Since F_{OSC} is shut down during Sleep mode, the Compare mode will not function properly during Sleep, unless the timer is running. The device will wake on interrupt (if enabled).

21.4 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

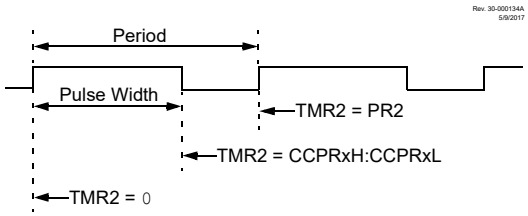
steps applied, which shortens the pulse width, supplies less power. The PWM period is defined as the duration of one complete cycle or the total amount of ON and OFF time combined.

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.

The shows a typical waveform of the PWM signal.

Figure 21-3. CCP PWM Output Signal



21.4.1 Standard PWM Operation

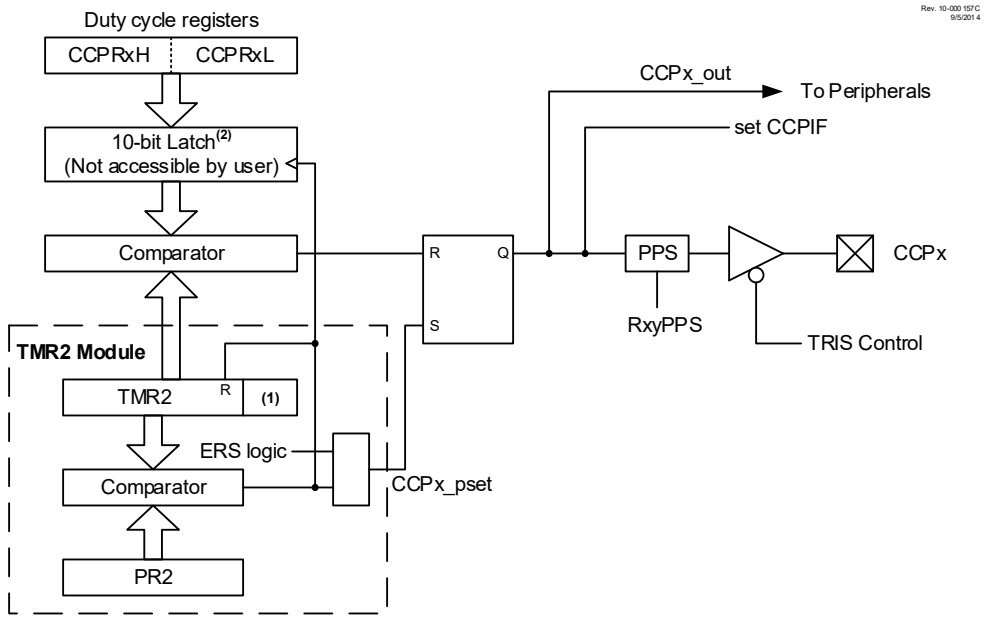
The standard PWM function described in this section is available and identical for all CCP modules.

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:

- Even numbered TxPR registers (T2PR, T4PR, etc)
- Even numbered TxCON registers (T2CON, T4CON, etc)
- 16-bit CCPRx registers
- CCPxCON registers

It is required to have $F_{OSC}/4$ as the clock input to TxTMR for correct PWM operation. The following figure shows a simplified block diagram of PWM operation.

Figure 21-4. Simplified PWM Block Diagram



Note:

1. 8-bit timer is concatenated with two bits generated by F_{OSC} or two bits of the internal prescaler to create 10-bit time base.
2. The alignment of the 10 bits from the CCPRx register is determined by the CCPxFMT bit.

Important: The corresponding TRIS bit must be cleared to enable the PWM output on the CCPx pin.

21.4.2 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 T2PR register with the PWM period value.
3. Configure the CCP module for the PWM mode by loading the CCPxCON register with the appropriate values.
4. Load the CCPRx register with the PWM duty cycle value and configure the FMT bit to set the proper register alignment.
5. Configure and start Timer2:
   - Clear the TMR2IF interrupt flag bit of the PIRx 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 T2ON bit.

6. Enable PWM output pin:
   – Wait until the timer overflows and the TMR2IF bit of the PIRx register is set. See Note below.
   – Enable the CCPx pin output driver by clearing the associated TRIS bit.

Important: 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.

Related Links
20.9.3 TxCON

21.4.3 Timer2 Timer Resource

The PWM standard mode makes use of the 8-bit Timer2 timer resources to specify the PWM period.

21.4.4 PWM Period

The PWM period is specified by the T2PR register of Timer2. The PWM period can be calculated using the formula in the equation below.

Equation 21-1. PWM Period

$$PWMPeriod = [(T2PR + 1)] \bullet 4 \bullet T_{OSC} \bullet (TMR2\mathrm{Pr}escaleValue)$$

where $T_{OSC} = 1/F_{OSC}$

When T2TMR is equal to T2PR, the following three events occur on the next increment cycle:

- T2TMR is cleared
- The CCPx pin is set. (Exception: If the PWM duty cycle = 0%, the pin will not be set.)
- The PWM duty cycle is transferred from the CCPRx register into a 10-bit buffer.

Important: The Timer postscaler (see *"Timer2 Interrupt"*) is not used in the determination of the PWM frequency.

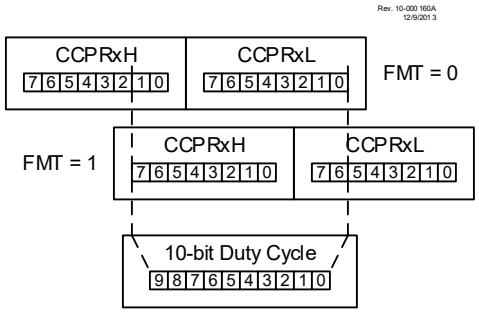
Related Links
20.4 Timer2 Interrupt

21.4.5 PWM Duty Cycle

The PWM duty cycle is specified by writing a 10-bit value to the CCPRx register. The alignment of the 10-bit value is determined by the FMT bit (see Figure 21-5). The CCPRx register 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 T2PR and T2TMR.

The equations below are used to calculate the PWM pulse width and the PWM duty cycle ratio.

Figure 21-5. PWM 10-Bit Alignment



Equation 21-2. Pulse Width

$$Pulse\ Width = (CCPRxH{:}CCPRxL\ register\ value) \bullet T_{OSC} \bullet (TMR2\ \mathrm{Prescale}\ Value)$$

Equation 21-3. Duty Cycle

$$DutyCycleRatio = \frac{(CCPRxH{:}CCPRxL\ register\ value)}{4(T2PR + 1)}$$

The CCPRx register is used to double buffer the PWM duty cycle. This double buffering is essential for glitchless PWM operation.

The 8-bit timer T2TMR register is concatenated with either the 2-bit internal system clock (F_{OSC}), or two bits of the prescaler, to create the 10-bit time base. The system clock is used if the Timer2 prescaler is set to 1:1.

When the 10-bit time base matches the CCPRx register, then the CCPx pin is cleared (see Figure 21-4).

21.4.6 PWM Resolution

The resolution determines the number of available duty cycles for a given period. For example, a 10-bit resolution will result in 1024 discrete duty cycles, whereas an 8-bit resolution will result in 256 discrete duty cycles.

The maximum PWM resolution is ten bits when T2PR is 255. The resolution is a function of the T2PR register value as shown below.

Equation 21-4. PWM Resolution

$$\mathrm{Resolution} = \frac{\log[4(T2PR + 1)]}{\log(2)} bits$$

Important: If the pulse-width value is greater than the period, the assigned PWM pin(s) will remain unchanged.

Table 21-3. Example PWM Frequencies and Resolutions (F_{OSC} = 20 MHz)

PWM Frequency	1.22 kHz	4.88 kHz	19.53 kHz	78.12 kHz	156.3 kHz	208.3 kHz
Timer Prescale	16	4	1	1	1	1
T2PR Value	0xFF	0xFF	0xFF	0x3F	0x1F	0x17
Maximum Resolution (bits)	10	10	10	8	7	6.6

Table 21-4. Example PWM Frequencies and Resolutions (F_{OSC} = 8 MHz)

PWM Frequency	1.22 kHz	4.90 kHz	19.61 kHz	76.92 kHz	153.85 kHz	200.0 kHz
Timer Prescale	16	4	1	1	1	1
T2PR Value	0x65	0x65	0x65	0x19	0x0C	0x09
Maximum Resolution (bits)	8	8	8	6	5	5

21.4.7 Operation in Sleep Mode

In Sleep mode, the T2TMR register will not increment and the state of the module will not change. If the CCPx pin is driving a value, it will continue to drive that value. When the device wakes up, T2TMR will continue from the previous state.

21.4.8 Changes in System Clock Frequency

The PWM frequency is derived from the system clock frequency. Any changes in the system clock frequency will result in changes to the PWM frequency. See the *"Oscillator Module (with Fail-Safe Clock Monitor)"* section for additional details.

Related Links

4. Oscillator Module (with Fail-Safe Clock Monitor)

21.4.9 Effects of Reset

Any Reset will force all ports to Input mode and the CCP registers to their Reset states.

21.5 Register Summary - CCP Control

Address	Name	Bit Pos.								
0x0FA5	CCPR2	7:0	CCPRL[7:0]							
		15:8	CCPRH[7:0]							
0x0FA7	CCP2CON	7:0	EN		OUT	FMT	MODE[3:0]			
0x0FA8	CCP2CAP	7:0							CTS[1:0]	
0x0FA9	CCPR1	7:0	CCPRL[7:0]							
		15:8	CCPRH[7:0]							
0x0FAB	CCP1CON	7:0	EN		OUT	FMT	MODE[3:0]			
0x0FAC	CCP1CAP	7:0							CTS[1:0]	
0x0FAD	CCPTMRS	7:0	P4TSEL[1:0]		P3TSEL[1:0]		C2TSEL[1:0]		C1TSEL[1:0]	

21.6 Register Definitions: CCP Control

Long bit name prefixes for the CCP peripherals are shown in the following table. Refer to the *"Long Bit Names"* section for more information.

Table 21-5. CCP Long bit name prefixes

Peripheral	Bit Name Prefix
CCP1	CCP1
CCP2	CCP2

Related Links

1.4.2.2 Long Bit Names

21.6.1 CCPxCON

Name: CCPxCON
Address: 0xFAB,0xFA7

CCP Control Register

Bit	7	6	5	4	3	2	1	0
	EN		OUT	FMT	MODE[3:0]			
Access	R/W		RO	R/W	R/W	R/W	R/W	R/W
Reset	0		x	0	0	0	0	0

Bit 7 – EN CCP Module Enable bit

Value	Description
1	CCP is enabled
0	CCP is disabled

Bit 5 – OUT CCP Output Data bit (read-only)

Bit 4 – FMT CCPW (pulse-width) Value Alignment bit

Value	Condition	Description
x	Capture mode	Not used
x	Compare mode	Not used
1	PWM mode	Left-aligned format
0	PWM mode	Right-aligned format

Bits 3:0 – MODE[3:0] CCP Mode Select bits

Table 21-6. 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

MODE	Operating Mode	Operation	Set CCPxIF
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 = '0000' or EN = 0.
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.

Note:

1. The set and clear operations of the Compare mode are reset by setting MODE = '0000' or EN = 0.
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.

21.6.2 CCPxCAP

Name: CCPxCAP
Address: 0xFAC,0xFA8

Capture Trigger Input Selection Register

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

Bits 1:0 – CTS[1:0] Capture Trigger Input Selection bits

Table 21-7. Capture Trigger Sources

CTS	Source
11	IOC Interrupt
10	CMP2_output
01	CMP1_output
00	Pin selected by CCPxPPS

21.6.3 CCPRx

Name: CCPRx
Address: 0xFA9,0xFA5

Capture/Compare/Pulse Width Register

Bit	15	14	13	12	11	10	9	8
	CCPRH[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	x	x	x	x	x	x	x	x

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

Bits 15:8 – CCPRH[7:0]
Capture/Compare/Pulse Width High byte

Value	Name	Description
0 to 255	21.6.1.4 MODE = Capture	High byte of 16-bit captured value
0 to 255	21.6.1.4 MODE = Compare	High byte of 16-bit compare value
0,1,2,3	21.6.1.4 MODE = PWM & 21.6.1.3 FMT=0	CCPRH<1:0>=Bits<9:8> of 10-bit Pulse width value CCPRH<7:2> not used
0 to 255	21.6.1.4 MODE = PWM & 21.6.1.3 FMT=1	Bits<9:2> of 10-bit Pulse width value

Bits 7:0 – CCPRL[7:0]
Capture/Compare/Pulse Width Low byte

Value	Name	Description
0 to 255	21.6.1.4 MODE = Capture	Low byte of 16-bit captured value
0 to 255	21.6.1.4 MODE = Compare	Low byte of 16-bit compare value
0 to 255	21.6.1.4 MODE = PWM & 21.6.1.3 FMT=0	Bits<7:0> of 10-bit Pulse width value
0,64,128,192	21.6.1.4 MODE = PWM & 21.6.1.3 FMT=1	CCPRL<7:6>=Bits<1:0> of 10-bit Pulse width value CCPRL<5:0> not used

21.6.4 CCPTMRS

Name: CCPTMRS
Address: 0xFAD

CCP Timers Control Register

Bit	7	6	5	4	3	2	1	0
	P4TSEL[1:0]		P3TSEL[1:0]		C2TSEL[1:0]		C1TSEL[1:0]	
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	1	0	1	0	1	0	1

Bits 4:5, 6:7 – PnTSEL PWMn Timer Selection bits

Value	Description
11	PWMn based on Timer6
10	PWMn based on Timer4
01	PWMn based on Timer2
00	Reserved

Bits 0:1, 2:3 – CnTSEL CCPn Timer Selection bits

Value	Description
11	CCPn is based off Timer5 in Capture/Compare mode and Timer6 in PWM mode
10	CCPn is based off Timer3 in Capture/Compare mode and Timer4 in PWM mode
01	CCPn is based off Timer1 in Capture/Compare mode and Timer2 in PWM mode
00	Reserved

22. (PWM) Pulse-Width Modulation

The PWM module generates a Pulse-Width Modulated signal determined by the duty cycle, period, and resolution that are configured by the following registers:

- TxPR
- TxCON
- PWMxDC
- PWMxCON

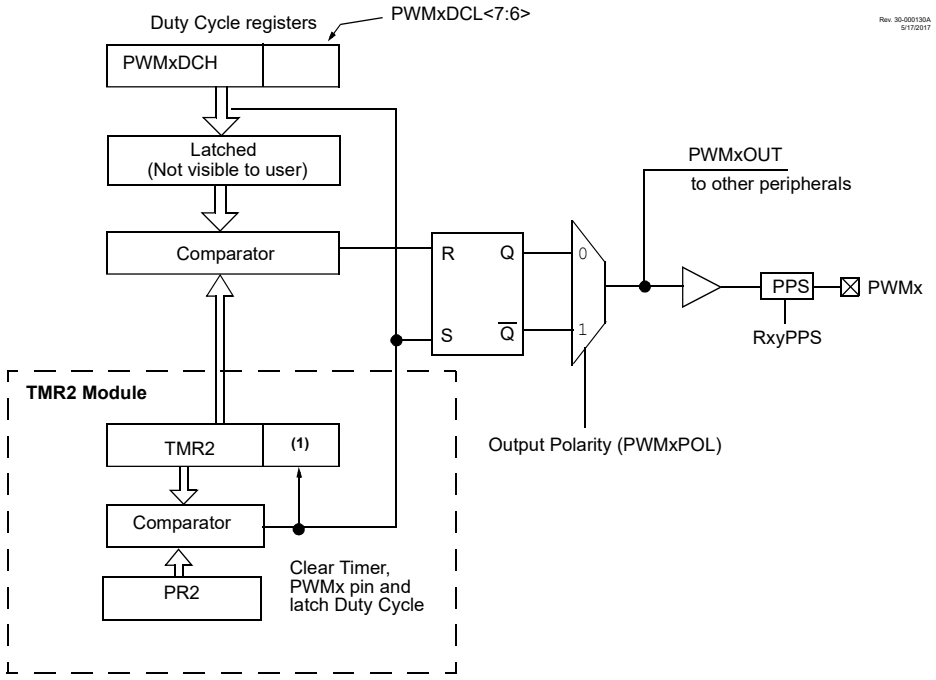
Important: The corresponding TRIS bit must be cleared to enable the PWM output on the PWMx pin.

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 CCPTMRS register. 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.

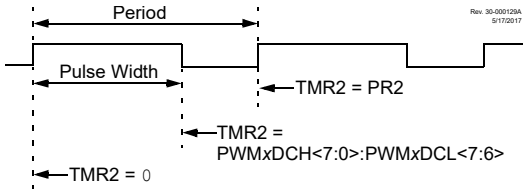
Figure 22-2 shows a typical waveform of the PWM signal.

Figure 22-1. Simplified PWM Block Diagram



Note 1: 8-bit timer is concatenated with the two Least Significant bits of 1/FOSC adjusted by the Timer2 prescaler to create a 10-bit time base.

Figure 22-2. PWM Output



For a step-by-step procedure on how to set up this module for PWM operation, refer to 22.9 Setup for PWM Operation using PWMx Output Pins.

22.1 Fundamental Operation

The PWM module produces a 10-bit resolution output. The PWM timer can be selected using the PxTSEL bits in the CCPTMRS register. The default selection for PWMx is TMR2. Note that the PWM module operation in the following sections is described with respect to TMR2. Timer2 and T2PR 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.

Important: 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 T2TMR is cleared. Each PWMx is cleared when TxTMR 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 T2PR, the PWM output is never cleared (100% duty cycle).

Important: The PWMxDCH and PWMxDCL registers are double buffered. The buffers are updated when T2TMR matches T2PR. Care should be taken to update both registers before the timer match occurs.

22.2 PWM Output Polarity

The output polarity is inverted by setting the POL bit.

22.3 PWM Period

The PWM period is specified by the TxPR register The PWM period can be calculated using the formula of 22.3 PWM Period. It is required to have $F_{OSC}/4$ as the selected clock input to the timer for correct PWM operation.

Equation 22-1. PWM Period

$$PWMPeriod = [(T2PR) + 1] \bullet 4 \bullet Tosc \bullet (TMR2\ \mathrm{Prescale}Value)$$

Note: $T_{OSC} = 1/F_{OSC}$

When T2TMR is equal to T2PR, the following three events occur on the next increment cycle:

- T2TMR 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.

Important: The Timer2 postscaler has no effect on the PWM operation.

22.4 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.

The formulas below are used to calculate the PWM pulse width and the PWM duty cycle ratio.

Equation 22-2. Pulse Width

$$PulseWidth = (PWMxDCH{:}PWMxDCL<7{:}6>) \bullet Tosc \bullet (TMR2\mathrm{P}rescaleValue)$$

Note: $T_{OSC} = 1/F_{OSC}$

Equation 22-3. Duty Cycle Ratio

$$DutyCycleRatio = \frac{(PWMxDCH{:}PWMxDCL<7{:}6>)}{4(T2PR+1)}$$

The 8-bit timer T2TMR register is concatenated with the two Least Significant bits of $1/F_{OSC}$, 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.

22.5 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 T2PR is 255. The resolution is a function of the T2PR register value as shown below.

Equation 22-4. PWM Resolution

$$\mathrm{R}esolution = \frac{\log[4(T2PR+1)]}{\log(2)} bits$$

Important: If the pulse-width value is greater than the period, the assigned PWM pin(s) will remain unchanged.

Table 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
T2PR 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
T2PR Value	0x65	0x65	0x65	0x19	0x0C	0x09
Maximum Resolution (bits)	8	8	8	6	5	5

22.6 Operation in Sleep Mode

In Sleep mode, the T2TMR 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, T2TMR will continue from its previous state.

22.7 Changes in System Clock Frequency

The PWM frequency is derived from the system clock frequency (F_{OSC}). Any changes in the system clock frequency will result in changes to the PWM frequency.

Related Links

4. Oscillator Module (with Fail-Safe Clock Monitor)

22.8 Effects of Reset

Any Reset will force all ports to Input mode and the PWM registers to their Reset states.

22.9 Setup for PWM Operation using PWMx Output 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 T2PR 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 PIRx register.(1)
   - 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 PIRx register is set.(2)
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.

22.9.1 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.

22.10 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 T2PR 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 PIRx register.(1)
   - 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. Wait until Timer2 overflows, TMR2IF bit of the PIRx register is set.(1)
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.

22.11 Register Summary - Registers Associated with PWM

Address	Name	Bit Pos.								
0x0F9F	PWM4DC	7:0	DCL[1:0]							
		15:8	DCH[7:0]							
0x0FA1	PWM4CON	7:0	EN		OUT	POL				
0x0FA2	PWM3DC	7:0	DCL[1:0]							
		15:8	DCH[7:0]							
0x0FA4	PWM3CON	7:0	EN		OUT	POL				
0x0FA5 ... 0x0FAC	Reserved									
0x0FAD	CCPTMRS	7:0	P4TSEL[1:0]		P3TSEL[1:0]		C2TSEL[1:0]		C1TSEL[1:0]	

22.12 Register Definitions: PWM Control

Long bit name prefixes for the PWM peripherals are shown in the table below. Refer to the *"Long Bit Names Section"* for more information.

Table 22-3. PWM Bit Name Prefixes

Peripheral	Bit Name Prefix
PWM3	PWM3
PWM4	PWM4

Related Links

1.4.2.2 Long Bit Names

22.12.1 PWMxCON

Name: PWMxCON
Address: 0xFA4,0xFA1

PWM Control Register

Bit	7	6	5	4	3	2	1	0
	EN		OUT	POL				
Access	R/W		RO	R/W				
Reset	0		0	0				

Bit 7 – EN PWM Module Enable bit

Value	Description
1	PWM module is enabled
0	PWM module is disabled

Bit 5 – OUT PWM Module Output Level When Bit is Read

Bit 4 – POL PWM Output Polarity Select bit

Value	Description
1	PWM output is inverted
0	PWM output is normal

22.12.2 CCPTMRS

Name: CCPTMRS
Address: 0xFAD

CCP Timers Control Register

Bit	7	6	5	4	3	2	1	0
	P4TSEL[1:0]		P3TSEL[1:0]		C2TSEL[1:0]		C1TSEL[1:0]	
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	1	0	1	0	1	0	1

Bits 7:6 – P4TSEL[1:0] PWM4 Timer Selection bits

Value	Description
11	PWM4 based on TMR6
10	PWM4 based on TMR4
01	PWM4 based on TMR2
00	Reserved

Bits 5:4 – P3TSEL[1:0] PWM3 Timer Selection bits

Value	Description
11	PWM3 based on TMR6
10	PWM3 based on TMR4
01	PWM3 based on TMR2
00	Reserved

Bits 3:2 – C2TSEL[1:0] CCP2 Timer Selection bits

Value	Description
11	CCP2 is based off Timer5 in Capture/Compare mode and Timer6 in PWM mode
10	CCP2 is based off Timer3 in Capture/Compare mode and Timer4 in PWM mode
01	CCP2 is based off Timer1 in Capture/Compare mode and Timer2 in PWM mode
00	Reserved

Bits 1:0 – C1TSEL[1:0] CCP1 Timer Selection bits

Value	Description
11	CCP1 is based off Timer5 in Capture/Compare mode and Timer6 in PWM mode
10	CCP1 is based off Timer3 in Capture/Compare mode and Timer4 in PWM mode
01	CCP1 is based off Timer1 in Capture/Compare mode and Timer2 in PWM mode
00	Reserved

© 2018 Microchip Technology Inc.

22.12.3 PWMxDC

Name: PWMxDC
Address: 0xFA2,0xF9F

PWM Duty Cycle Register

Bit	15	14	13	12	11	10	9	8
	DCH[7:0]							
Access								
Reset	x	x	x	x	x	x	x	x

Bit	7	6	5	4	3	2	1	0
	DCL[1:0]							
Access								
Reset	x	x						

Bits 15:8 – DCH[7:0] PWM Duty Cycle Most Significant bits
These bits are the MSbs of the PWM duty cycle.

Reset States: POR/BOR = xxxxxxxx
All Other Resets = uuuuuuuu

Bits 7:6 – DCL[1:0] PWM Duty Cycle Least Significant bits
These bits are the LSbs of the PWM duty cycle.

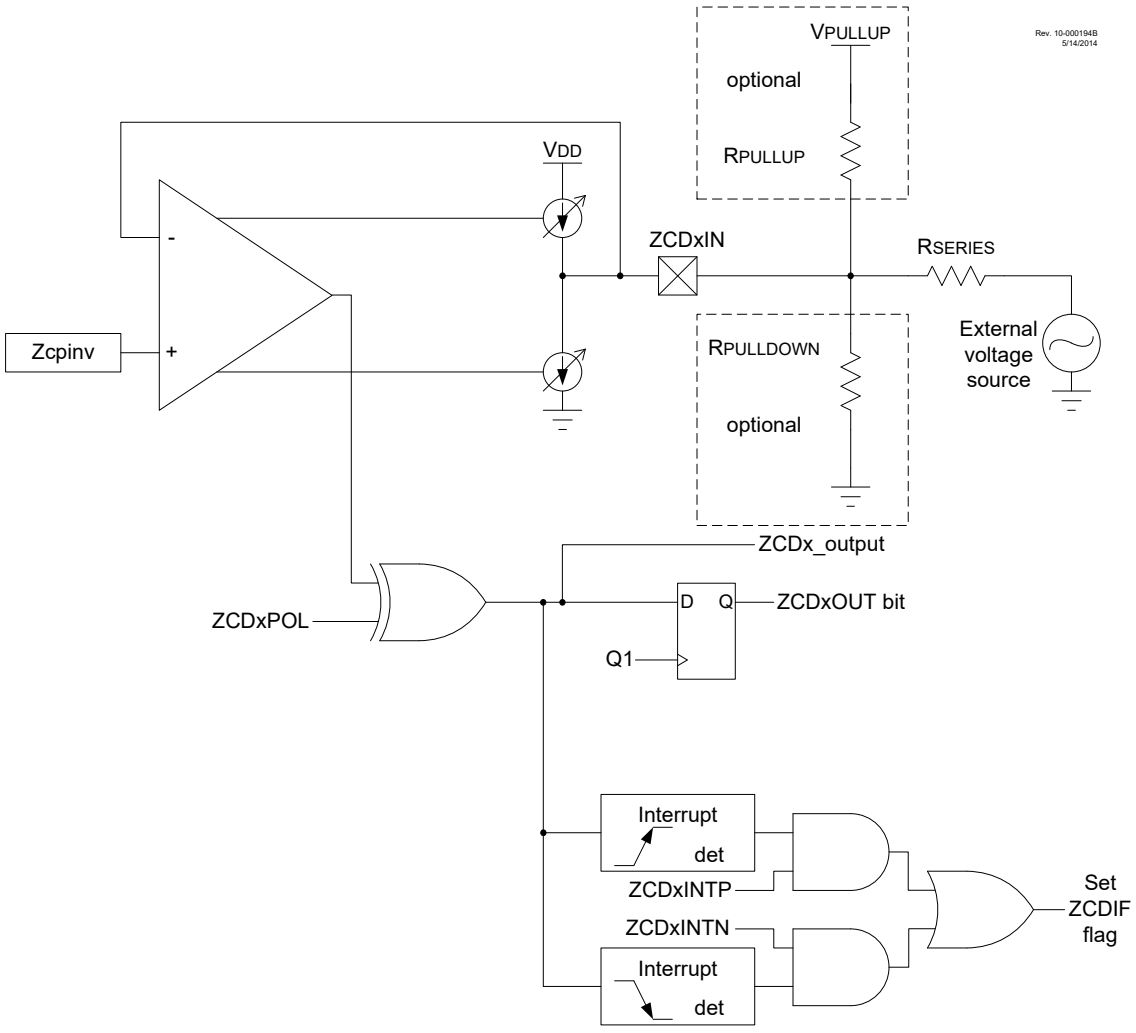
Reset States: POR/BOR = xx
All Other Resets = uu

23. (ZCD) Zero-Cross Detection Module

The ZCD module detects when an A/C signal crosses through the ground potential. The actual zero crossing threshold is the zero crossing reference voltage, Z_{CPINV}, which is typically 0.75V above ground.

The connection to the signal to be detected is through a series current-limiting resistor. The module applies a current source or sink to the ZCD pin to maintain a constant voltage on the pin, thereby preventing the pin voltage from forward biasing the ESD protection diodes. When the applied voltage is greater than the reference voltage, the module sinks current. When the applied voltage is less than the reference voltage, the module sources current. The current source and sink action keeps the pin voltage constant over the full range of the applied voltage. The ZCD module is shown in the following simplified block diagram.

Figure 23-1. Simplified ZCD Block Diagram



The ZCD module is useful when monitoring an A/C waveform for, but not limited to, the following purposes:

- A/C period measurement
- Accurate long term time measurement

- Dimmer phase delayed drive
- Low EMI cycle switching

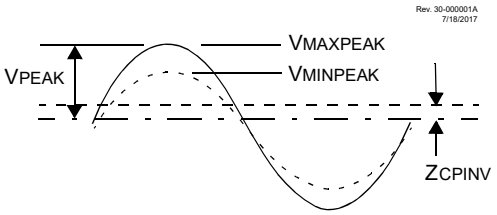
23.1 External Resistor Selection

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 µA. 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 23-2. External Voltage Source



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 OUT bit is set when the current sink is active, and cleared when the current source is active. The OUT bit is affected by the polarity bit.

The OUT signal can also be used as input to other modules. This is controlled by the registers of the corresponding module. OUT can be used as follows:

- Gate source for TMR1/3/5
- Clock source for TMR2/4/6
- Reset source for TMR2/4/6

23.3 ZCD Logic Polarity

The POL bit inverts the OUT bit relative to the current source and sink output. When the POL bit is set, a OUT high indicates that the current source is active, and a low output indicates that the current sink is active.

The POL bit affects the ZCD interrupts.

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 PIRx register will be set when either edge detector is triggered and its associated enable bit is set. The INTP enables rising edge interrupts and the INTN bit enables falling edge interrupts. 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 IPRx register.

To fully enable the interrupt, the following bits must be set:

- ZCDIE bit of the PIEx register
- INTP bit for rising edge detection
- INTN bit for falling edge detection
- PEIE and GIE bits of the INTCON register

Changing the POL bit will cause an interrupt, regardless of the level of the SEN bit.

The ZCDIF bit of the PIRx register must be cleared in software as part of the interrupt service. If another edge is detected while this flag is being cleared, the flag will still be set at the end of the sequence.

23.5 Correction for Z_{CPINV} 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.

23.5.1 Correction by AC Coupling

When the external voltage source is sinusoidal, the effects of the Z_{CPINV} offset can be eliminated by isolating the external voltage source from the ZCD pin with a capacitor, in addition to the voltage reducing resistor. The capacitor will cause a phase shift resulting in the ZCD output switch in advance of the actual zero crossing event. The phase shift will be the same for both rising and falling zero crossings, which can be compensated for by either delaying the CPU response to the ZCD switch by a timer or other means, or selecting a capacitor value large enough that the phase shift is negligible.

To determine the series resistor and capacitor values for this configuration, start by computing the impedance, Z, to obtain a peak current of 300 µA. Next, arbitrarily select a suitably large non-polar capacitor and compute its reactance, Xc, at the external voltage source frequency. Finally, compute the series resistor, capacitor peak voltage, and phase shift by the formulas shown below.

When this technique is used and the input signal is not present, the ZCD will tend to oscillate. To avoid this oscillation, connect the ZCD pin to V_{DD} or GND with a high-impedance resistor such as 200K.

Equation 23-2. R-C Equations

V_{PEAK} = external voltage source peak voltage

f = external voltage source frequency

C = series capacitor

R = series resistor

V_C = Peak capacitor voltage

Φ = Capacitor induced zero crossing phase advance in radians

T_Φ = Time ZC event occurs before actual zero crossing

$$Z = \frac{V_{PEAK}}{3 \times 10^{-4}}$$

$$X_C = \frac{1}{2\pi f C}$$

$$R = \sqrt{Z^2 - {X_C}^2}$$

$$V_C = X_C\left(3 \times 10^{-4}\right)$$

$$\Phi = \tan^{-1}\theta\left(\frac{X_C}{R}\right)$$

$$T_\Phi = \frac{\Phi}{2\pi f}$$

Equation 23-3. R-C Calcuation Example

$$V_{rms} = 120$$

$$V_{PEAK} = V_{rms} \times \sqrt{2} = 169.7$$

$$f = 60\ Hz$$

$$C = 0.1\ \mu F$$

$$Z = \frac{V_{PEAK}}{3 \times 10^{-4}} = \frac{169.7}{3 \times 10^{-4}} = 565.7\ k\Omega$$

$$X_C = \frac{1}{2\pi f C} = \frac{1}{2\pi \times 60 \times 10^{-7}} = 26.53\ k\Omega$$

$$R = \sqrt{Z^2 - {X_C}^2} = 565.1\ k\Omega\ (computed)$$

$$R_a = 560\ k\Omega\ (used)$$

$$Z_R = \sqrt{{R_a}^2 + {X_C}^2} = 560.6\ k\Omega$$

$$I_{PEAK} = \frac{V_{PEAK}}{Z_R} = 302.7 \times 10^{-6} A$$

$$V_C = X_C \times I_{PEAK} = 8.0\ V$$

$$\Phi = \tan^{-1}\theta\left(\frac{X_C}{R}\right) = 0.047 radians$$

$$T_\Phi = \frac{\Phi}{2\pi f} = 125.6 \mu s$$

23.5.2 Correction By Offset Current

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 below.

Equation 23-4. ZCD Event Offset

When External Voltage source is relative to V_{SS}

$$T_{offset} = \frac{\sin^{-1}\left(\frac{Z_{CPINV}}{V_{PEAK}}\right)}{2\pi f}$$

When External Voltage source is relative to V_{DD}

$$T_{offset} = \frac{\sin^{-1}\left(\frac{V_{DD} - Z_{CPINV}}{V_{PEAK}}\right)}{2\pi f}$$

This offset time can be compensated for by adding a pull-up or pull-down biasing resistor to the ZCD pin. A pull-up resistor is used when the external voltage source is varying relative to V_{SS}. A pull-down resistor is used when the voltage is varying relative to V_{DD}. The resistor adds a bias to the ZCD pin so that the target external voltage source must go to zero to pull the pin voltage to the Z_{CPINV} switching voltage. The pull-up or pull-down value can be determined with the equations shown below.

Equation 23-5. ZCD Pull-up/Pull-down Resistor

When External Voltage source is relative to V_{SS}

$$R_{pullup} = \frac{R_{SERIES}\left(V_{pullup} - Z_{CPINV}\right)}{Z_{CPINV}}$$

When External Voltage source is relative to V_{DD}

$$R_{pulldown} = \frac{R_{SERIES}(Z_{CPINV})}{(V_{DD} - Z_{CPINV})}$$

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 ± 600 µA 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 ± 600 µA and the minimum is at least ± 100 µA, compute the series resistance as shown in Equation 23-6. The compensating pull-up for this series resistance can be determined with the equations shown in Equation 23-5 because the pull-up value is independent from the peak voltage.

Equation 23-6. Series R for V range

$$R_{SERIES} = \frac{V_{MAX_PEAK} + V_{MIN_PEAK}}{7 \times 10^{-4}}$$

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 SEN bit must be set to enable the ZCD module.

23.9 Disabling the ZCD Module

The ZCD module can be disabled in two ways:

1. The $\overline{ZCD}$ Configuration bit disables the ZCD module when set. When this is the case then the ZCD module will be enabled by setting the 23.11.1.1 SEN bit. When the $\overline{ZCD}$ bit is clear, the ZCD is always enabled and the SEN bit has no effect.
2. The ZCD can also be disabled using the ZCDMD bit of the PMDx register. This is subject to the status of the $\overline{ZCD}$ bit.

23.10 Register Summary: ZCD Control

Address	Name	Bit Pos.								
0x0F2D	ZCDCON	7:0	SEN		OUT	POL			INTP	INTN

23.11 Register Definitions: ZCD Control

Long bit name prefixes for the ZCD peripherals are shown in the table below. Refer to the *"Long Bit Names Section"* for more information.

Table 23-1. ZCD Long Bit Name Prefixes

Peripheral	Bit Name Prefix
ZCD	ZCD

Related Links

1.4.2.2 Long Bit Names

1.4.2.2 Long Bit Names

23.11.1 ZCDCON

Name: ZCDCON
Address: 0xF2D

Zero-Cross Detect Control Register

Bit	7	6	5	4	3	2	1	0
	SEN		OUT	POL			INTP	INTN
Access	R/W		RO	R/W			R/W	R/W
Reset	0		x	0			0	0

Bit 7 – SEN Zero-Cross Detect Software Enable bit
This bit is ignored when ZCD fuse is cleared.

Value	Condition	Description
X	ZCD Config fuse = 0	Zero-cross detect is always enabled. ZCD. This bit is ignored. source and sink current.
1	ZCD Config fuse = 1	Zero-cross detect is enabled. ZCD pin is forced to output to source and sink current.
0	ZCD Config fuse = 1	Zero-cross detect is disabled. ZCD pin operates according to PPS and TRIS controls.

Bit 5 – OUT Zero-Cross Detect Data Output bit

Value	Condition	Description
1	POL = 0	ZCD pin is sinking current
0	POL = 0	ZCD pin is sourcing current
1	POL = 1	ZCD pin is sourcing current
0	POL = 1	ZCD pin is sinking current

Bit 4 – POL Zero-Cross Detect Polarity bit

Value	Description
1	ZCD logic output is inverted
0	ZCD logic output is not inverted

Bit 1 – INTP Zero-Cross Detect Positive-Going Edge Interrupt Enable bit

Value	Description
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 – INTN Zero-Cross Detect Negative-Going Edge Interrupt Enable bit

Value	Description
1	ZCDIF bit is set on high-to-low ZCD_output transition
0	ZCDIF bit is unaffected by high-to-low ZCD_output transition

24. (CWG) Complementary Waveform Generator Module

The Complementary Waveform Generator (CWG) produces half-bridge, full-bridge, and steering of PWM waveforms. It is backwards compatible with previous CCP functions. The PIC18F24/25Q10 family has 1 instance(s) 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 24.7 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 24.11 Auto-Shutdown.

24.2 Operating Modes

The CWG module can operate in six different modes, as specified by the MODE bits:

- Half-Bridge mode
- Push-Pull mode
- Asynchronous Steering mode
- Synchronous Steering mode
- Full-Bridge mode, Forward
- Full-Bridge mode, Reverse

All modes accept a single pulse data input, and provide up to four outputs as described in the following sections.

All modes include auto-shutdown control as described in 24.11 Auto-Shutdown

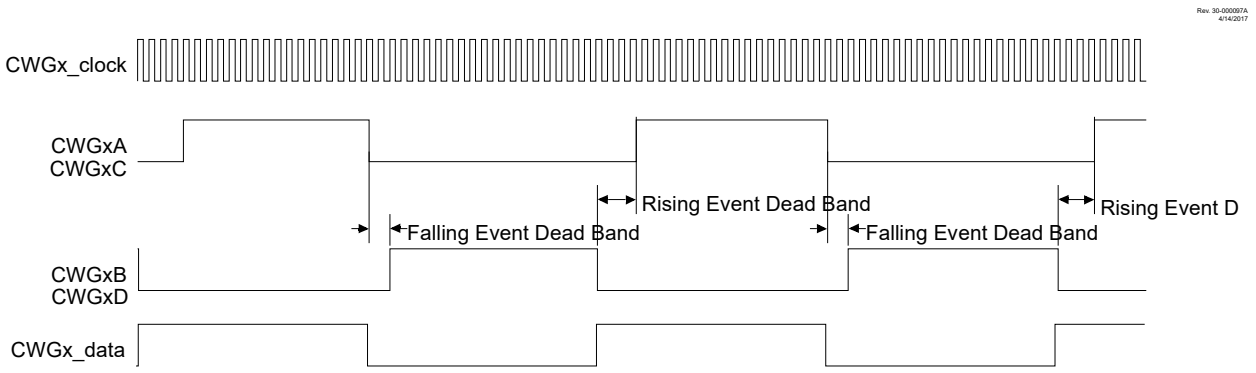
Important: Except as noted for Full-Bridge mode (24.2.3 Full-Bridge Modes), mode changes should only be performed while 24.15.1.1 EN = 0.

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 24-1. 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 24.7 Dead-Band Control. The output steering feature cannot be used in this mode. A basic block diagram of this mode is shown in Figure 24-2.

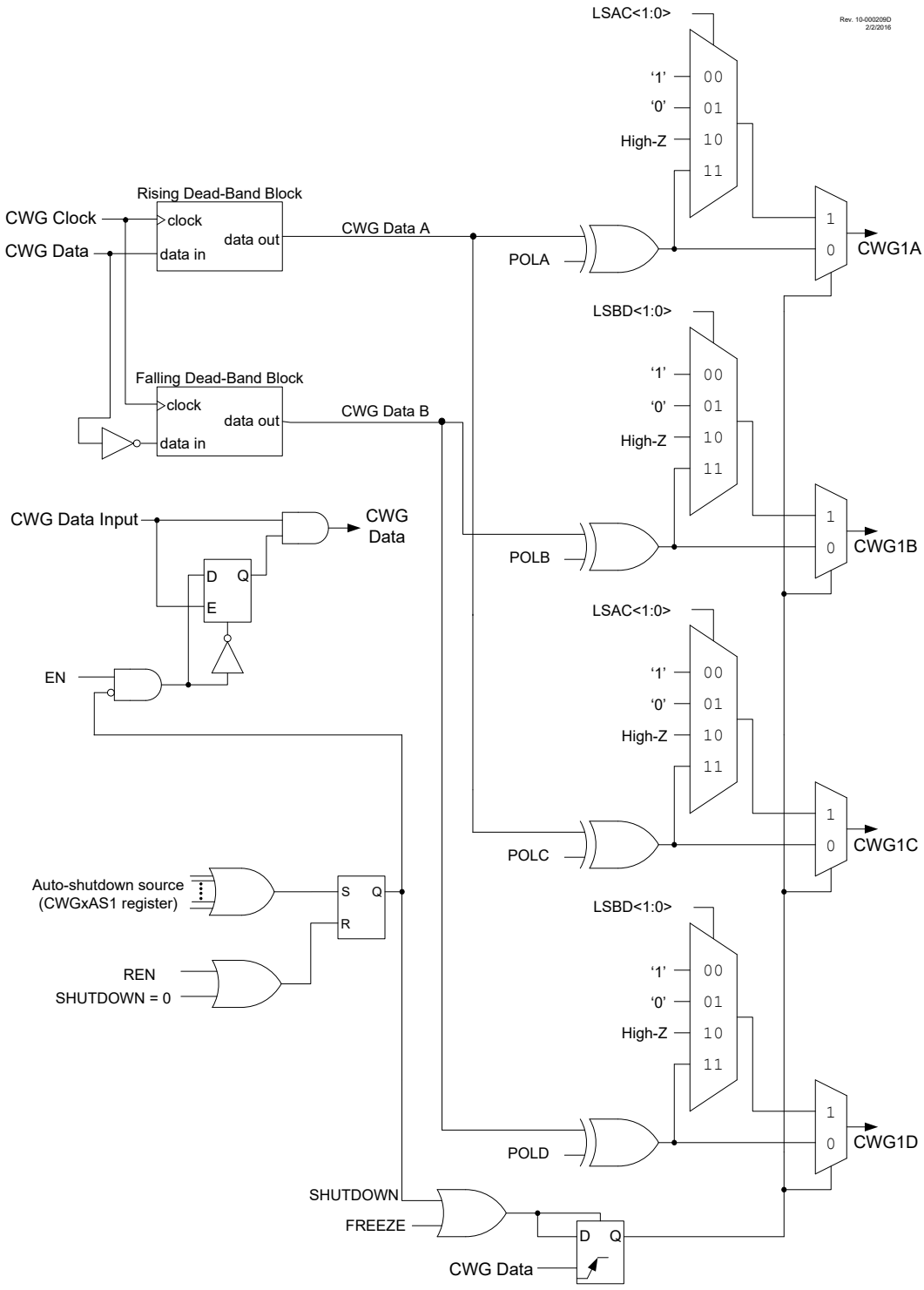
The unused outputs CWGxC and CWGxD drive similar signals, with polarity independently controlled by the POLC and POLD bits, respectively.

Figure 24-1. CWG Half-Bridge Mode Operation



Note: CWGx_rising_src = CCP1_out, CWGx_falling_src = ~CCP1_out

Figure 24-2. Simplified CWG Block Diagram (Half-Bridge Mode, MODE<2:0> = 100)



24.2.2 Push-Pull Mode

In Push-Pull mode, two output signals are generated, alternating copies of the input as illustrated in Figure 24-3. This alternation creates the push-pull effect required for driving some transformer-based

power supply designs. Steering modes are not used in Push-Pull mode. A basic block diagram for the Push-Pull mode is shown in Figure 24-4.

The push-pull sequencer is reset whenever EN = 0 or if an auto-shutdown event occurs. The sequencer is clocked by the first input pulse, and the first output appears on CWG1A.

The unused outputs CWGxC and CWGxD drive copies of CWGxA and CWGxB, respectively, but with polarity controlled by the POLC and POLD bits of the CWGxCON1 register, respectively.

Figure 24-3. CWG Push-Pull Mode Operation

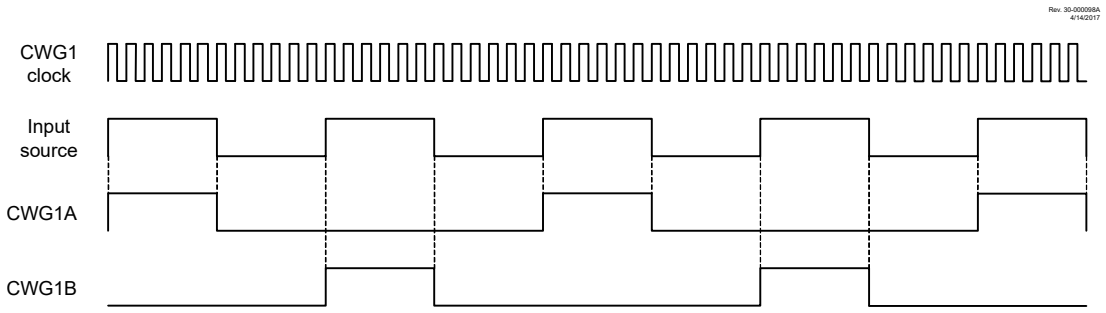
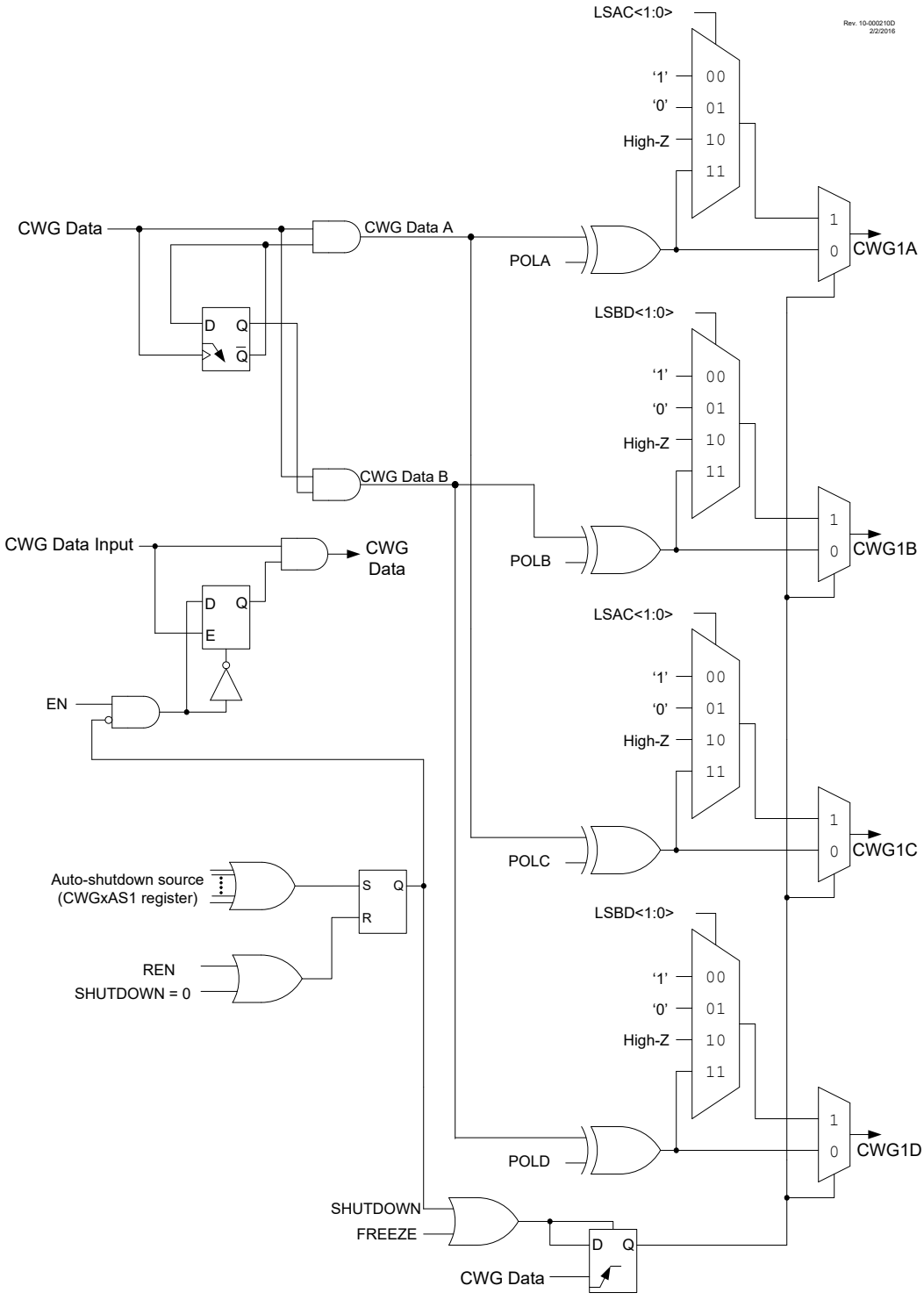


Figure 24-4. Simplified CWG Block Diagram (Push-Pull Mode, MODE<2:0> = 101)



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 CWGxCON0 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

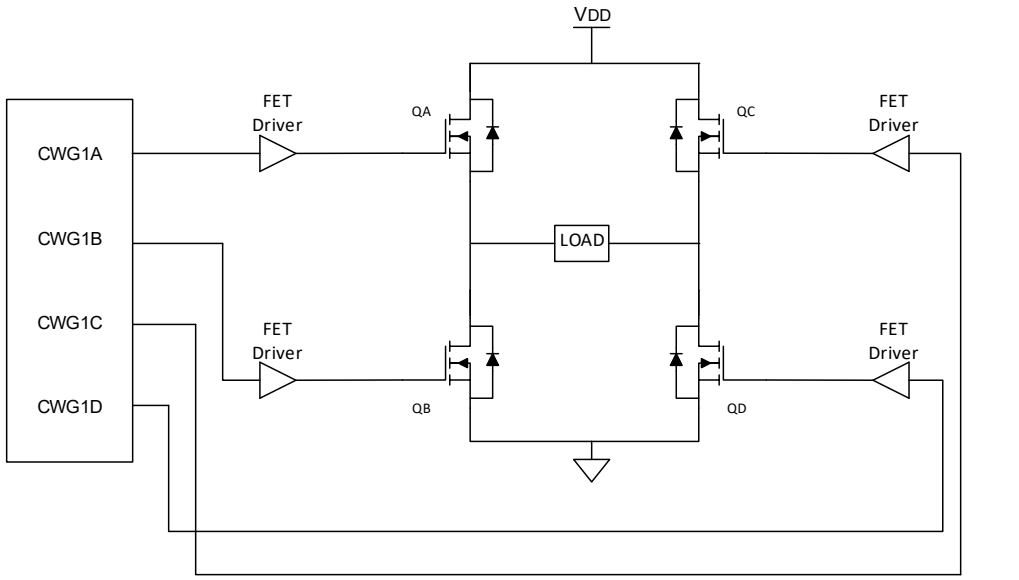
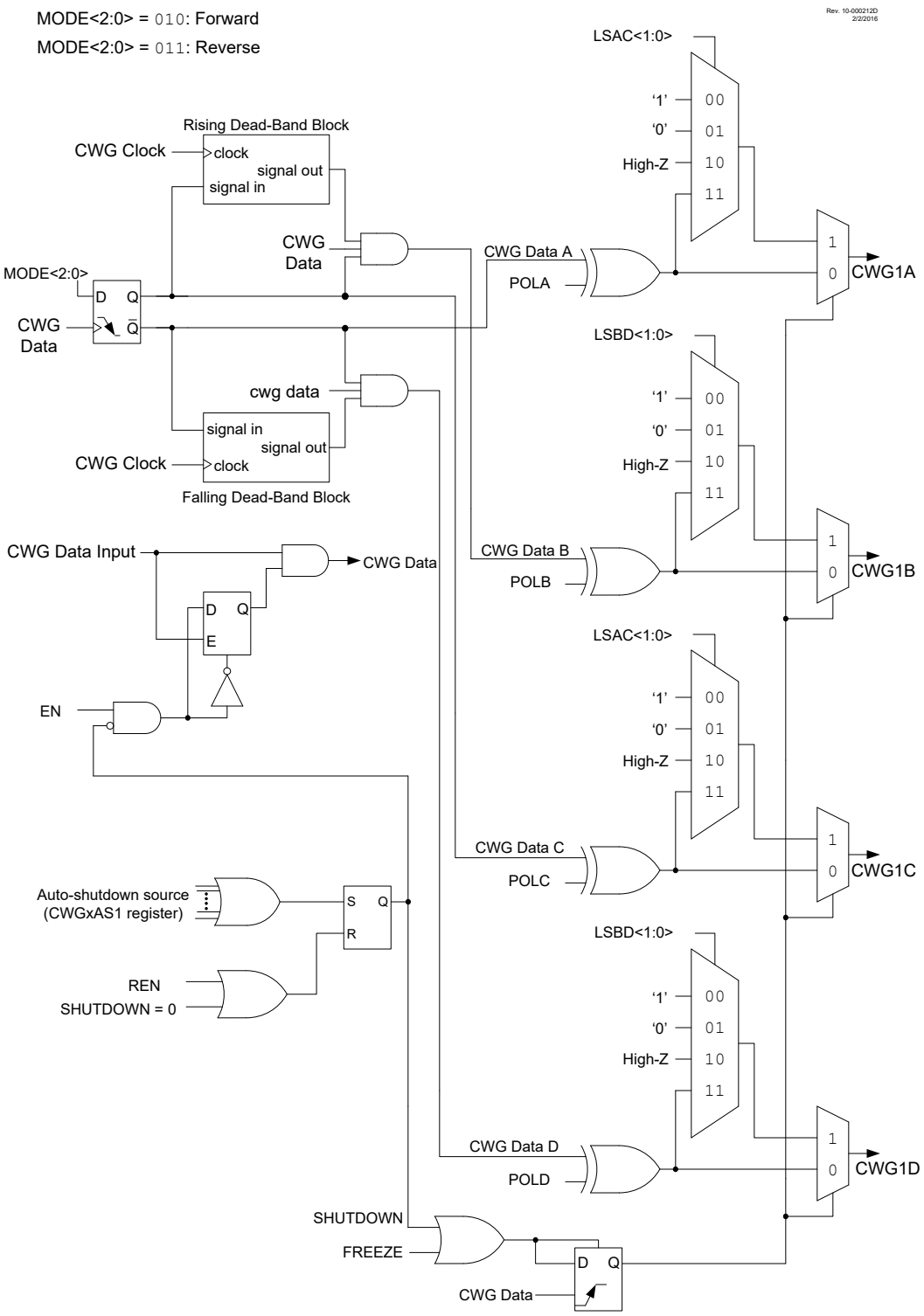


Figure 24-6. Simplified CWG Block Diagram (Forward and Reverse Full-Bridge Modes)

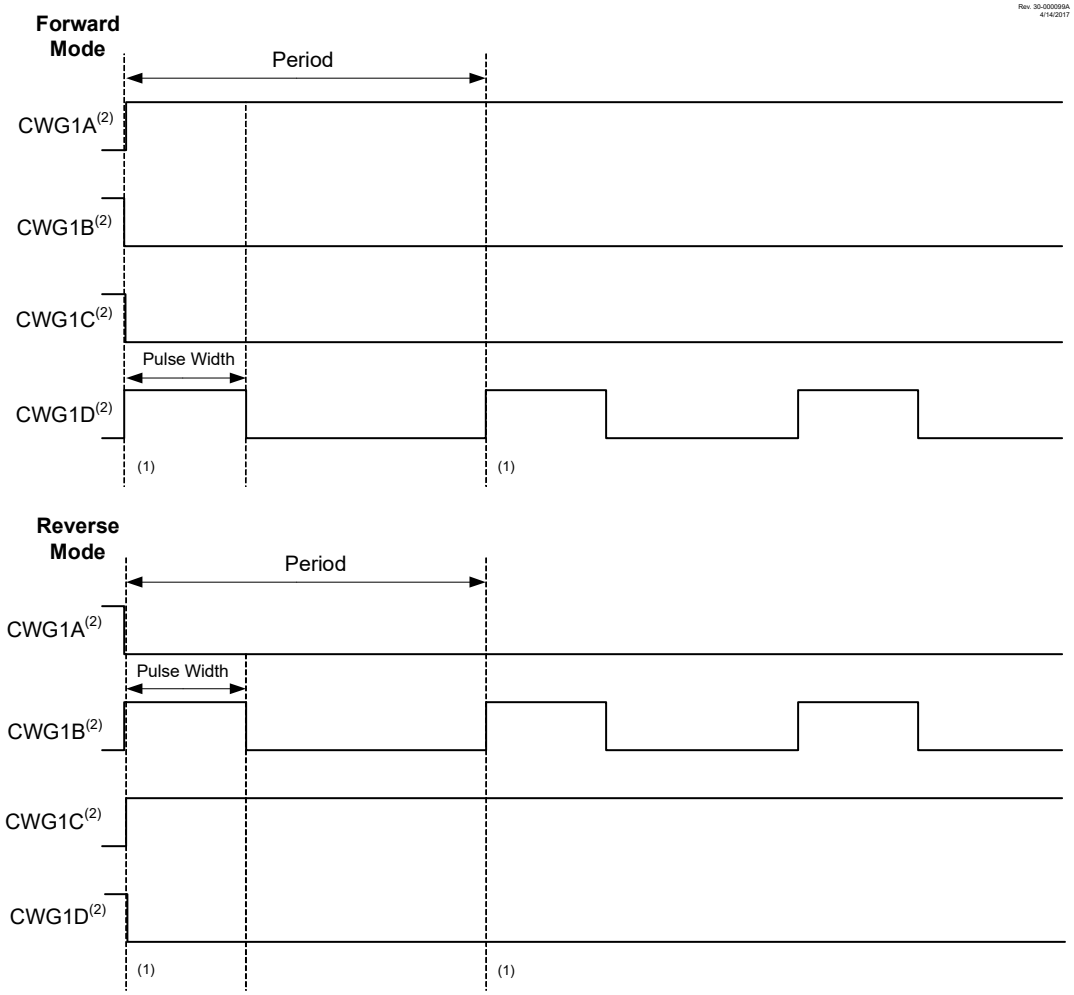


In Forward Full-Bridge mode (MODE = 010), CWGxA is driven to its active state, CWGxB and CWGxC are driven to their inactive state, and CWGxD is modulated by the input signal, as shown in Figure 24-7.

In Reverse Full-Bridge mode (MODE = 011), CWGxC is driven to its active state, CWGxA and CWGxD 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 24.7 Dead-Band Control, with additional details in 24.8 Rising Edge and Reverse Dead Band and 24.9 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 CWGxCON0 while keeping MODE<2:1> static, without disabling the CWG module.

Figure 24-7. Example of Full-Bridge Output



Note:

1. A rising CWG data input creates a rising event on the modulated output.
2. Output signals shown as active-high; all POLy bits are clear.

24.2.3.1 Direction Change in Full-Bridge Mode

In Full-Bridge mode, changing MODE controls the forward/reverse direction. Direction changes occur on the next rising edge of the modulated input.

A direction change is initiated in software by changing the MODE bits. The sequence is illustrated in Figure 24-8.

- The associated active output CWGxA and the inactive output CWGxC are switched to drive in the opposite direction.

- The previously modulated output CWGxD is switched to the inactive state, and the previously inactive output CWGxB 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.

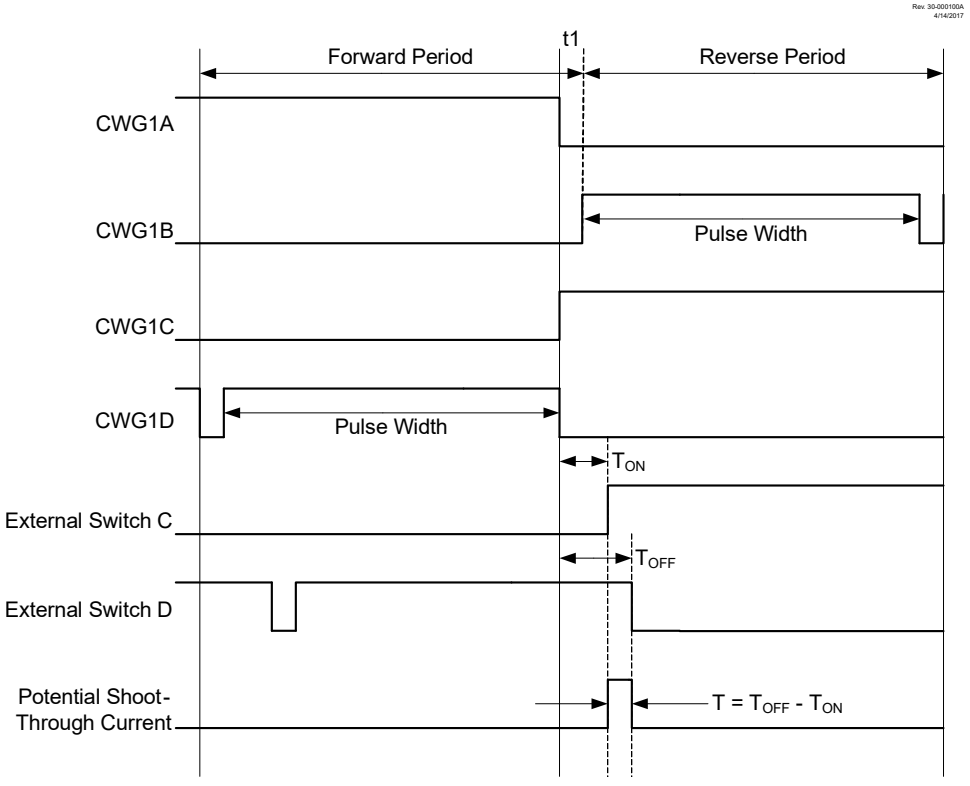
The dead-band delay is inserted only when changing directions, and only the modulated output is affected. The statically-configured outputs (CWGxA and CWGxC) are not afforded dead band, and switch essentially simultaneously.

The following figure shows an example of the CWG outputs changing directions from forward to reverse, at near 100% duty cycle. In this example, at time t1, the output of CWGxA and CWGxD become inactive, while output CWGxC becomes active. Since the turn-off time of the power devices is longer than the turn-on time, a shoot-through current will flow through 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.

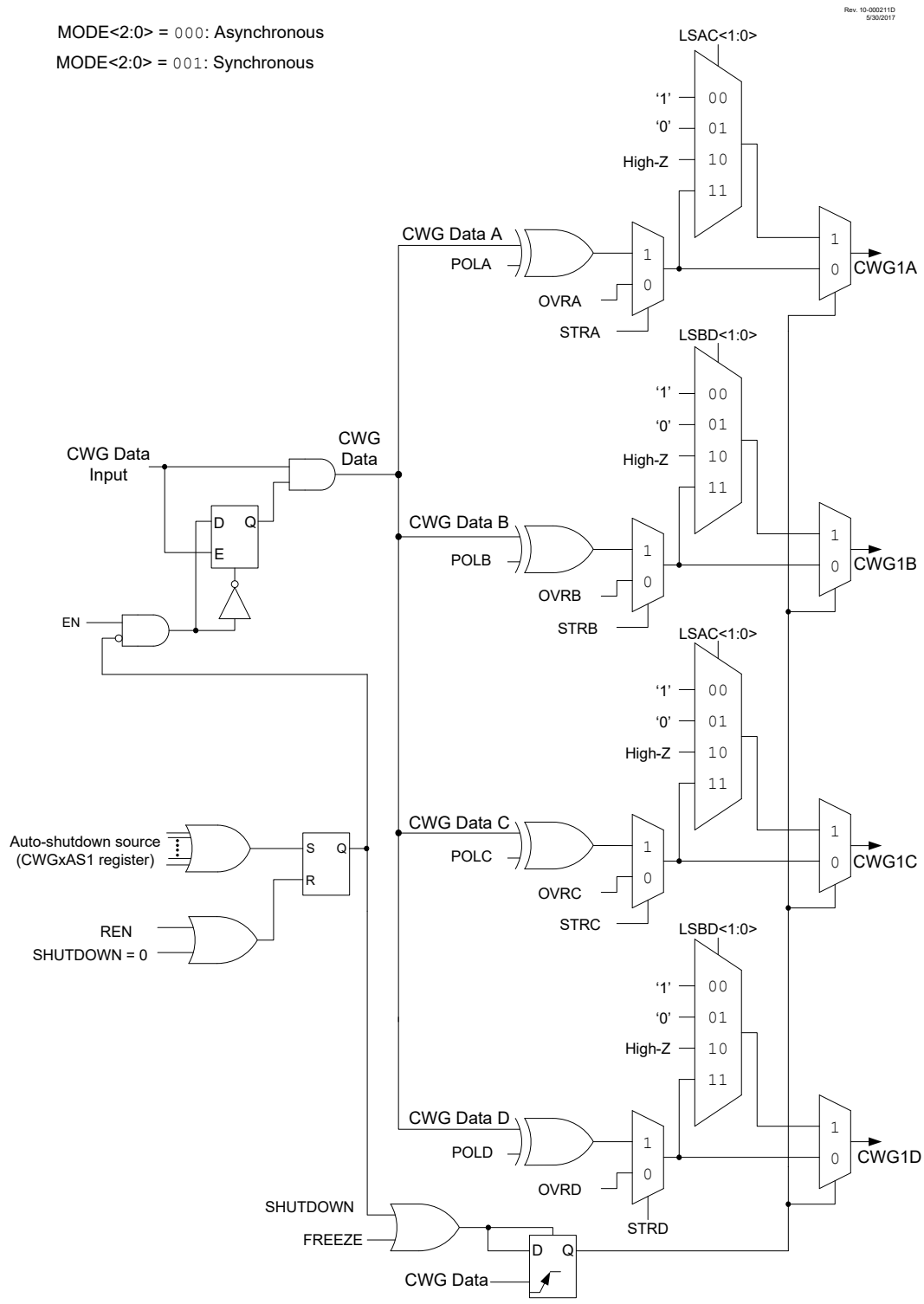
Figure 24-8. Example of PWM Direction Change at Near 100% Duty Cycle



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. 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.

Figure 24-9. Simplified CWG Block Diagram (Output Steering Modes)



For example, when STRA = 0 then the corresponding pin is held at the level defined by OVRA. When STRA = 1, then the pin is driven by the modulated input signal.

The POLy bits control the signal polarity only when STRy = 1.

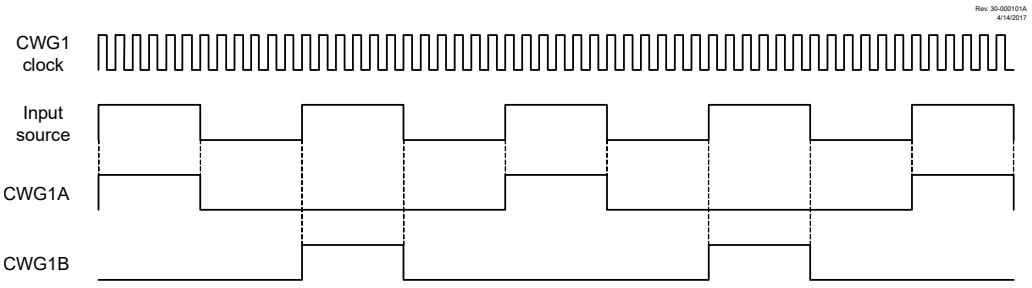
The CWG auto-shutdown operation also applies in Steering modes as described in 24.11 Auto-Shutdown". An auto-shutdown event will only affect pins that have STRy = 1.

24.2.4.1 Synchronous Steering Mode

In Synchronous Steering mode (24.15.1.3 MODE = 001), changes to steering selection registers take effect on the next rising edge of the modulated data input (see figure below). In Synchronous Steering mode, the output will always produce a complete waveform.

Important: Only the STRx bits are synchronized; the OVRx bits are not synchronized.

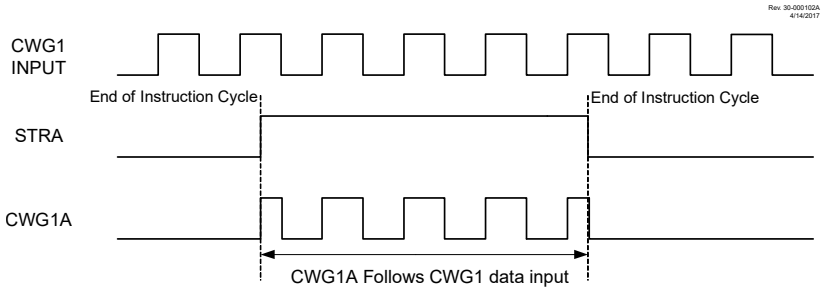
Figure 24-10. Example of Synchronous Steering (MODE = 001)



24.2.4.2 Asynchronous Steering Mode

In Asynchronous mode (24.15.1.3 MODE = 000), steering takes effect at the end of the instruction cycle that writes to STRx. In Asynchronous Steering mode, the output signal may be an incomplete waveform (see figure below). This operation may be useful when the user firmware needs to immediately remove a signal from the output pin.

Figure 24-11. Example of Asynchronous Steering (MODE = 000)



24.3 Start-up Considerations

The application hardware must use the proper external pull-up and/or pull-down resistors on the CWG output pins. This is required because all I/O pins are forced to high-impedance at Reset.

The polarity control bits (24.15.2.2 POLy) allow the user to choose whether the output signals are active-high or active-low.

24.4 Clock Source

The clock source is used to drive the dead-band timing circuits. The CWG module allows the following clock sources to be selected:

- F_{OSC} (system clock)
- HFINTOSC

When the HFINTOSC is selected, the HFINTOSC will be kept running during Sleep. Therefore, CWG modes requiring dead band can operate in Sleep, provided that the CWG data input is also active during Sleep. The clock sources are selected using the CS bit. The system clock F_{OSC}, is disabled in Sleep and thus dead-band control cannot be used.

24.5 Selectable Input Sources

The CWG generates the output waveforms from the input sources which are selected with the ISM bits as shown below.

Table 24-1. CWG Data Input Sources

ISM	Data Source
111	DSM_out
110	CMP2_out
101	CMP1_out
100	PWM4_out
011	PWM3_out
010	CCP2_out
001	CCP1_out
000	Pin selected by CWGxPPS

24.6 Output Control

24.6.1 CWG Outputs

Each CWG output can be routed to a Peripheral Pin Select (PPS) output via the RxyPPS register.

Related Links

17. (PPS) Peripheral Pin Select Module

24.6.2 Polarity Control

The polarity of each CWG output can be selected independently. When the output polarity bit is set, the corresponding output is active-high. Clearing the output polarity bit configures the corresponding output as active-low. However, polarity does not affect the override levels. Output polarity is selected with the POLy bits. Auto-shutdown and steering options are unaffected by polarity.

24.7 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.

© 2018 Microchip Technology Inc.

Dead band is timed by counting CWG clock periods from zero up to the value in the rising or falling dead-band counter registers.

24.7.1 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 24-1.

24.7.2 Dead-Band Functionality in Full-Bridge mode

In Full-Bridge mode, the dead-band counters are used when undergoing a direction change. The MODE<0> bit can be set or cleared while the CWG is running, allowing for changes from Forward to Reverse mode. The CWGxA and CWGxC signals will change immediately upon the first rising input edge following a direction change, but the modulated signals (CWGxB or CWGxD, depending on the direction of the change) will experience a delay dictated by the dead-band counters.

24.8 Rising Edge and Reverse Dead Band

In Half-Bridge mode, the rising edge dead band delays the turn-on of the CWGxA output after the rising edge of the CWG data input. In Full-Bridge mode, the reverse dead-band delay is only inserted when changing directions from Forward mode to Reverse mode, and only the modulated output CWGxB is affected.

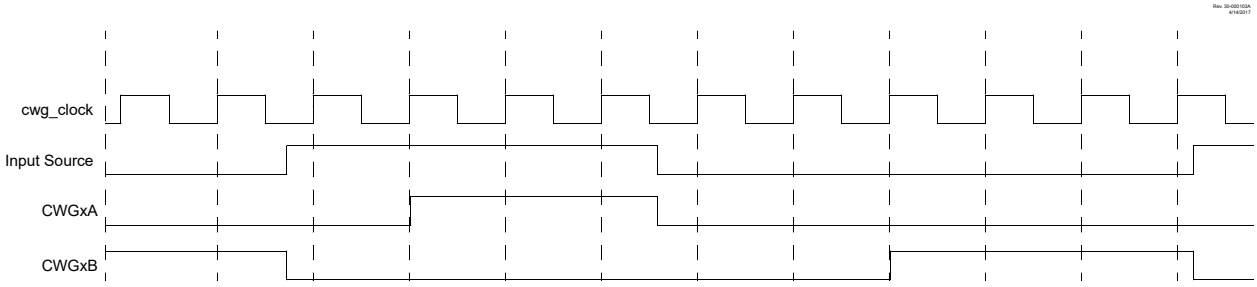
The 24.15.8 CWGxDBR register determines the duration of the dead-band interval on the rising edge of the input source signal. This duration is from 0 to 64 periods of the CWG clock.

Dead band is always initiated on the edge of the input source signal. A count of zero indicates that no dead band is present.

If the input source signal reverses polarity before the dead-band count is completed, then no signal will be seen on the respective output.

The CWGxDBR register value is double-buffered. When EN = 0, the buffer is loaded when CWG1DBR is written. When 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 is set. Refer to the following figure for an example.

Figure 24-12. Dead-Band Operation, CWGxDBR = 0x01, CWGxDBF = 0x02



24.9 Falling Edge and Forward Dead Band

In Half-Bridge mode, the falling edge dead band delays the turn-on of the CWGxB output at the falling edge of the CWG data input. In Full-Bridge mode, the forward dead-band delay is only inserted when changing directions from Reverse mode to Forward mode, and only the modulated output CWGxD is affected.

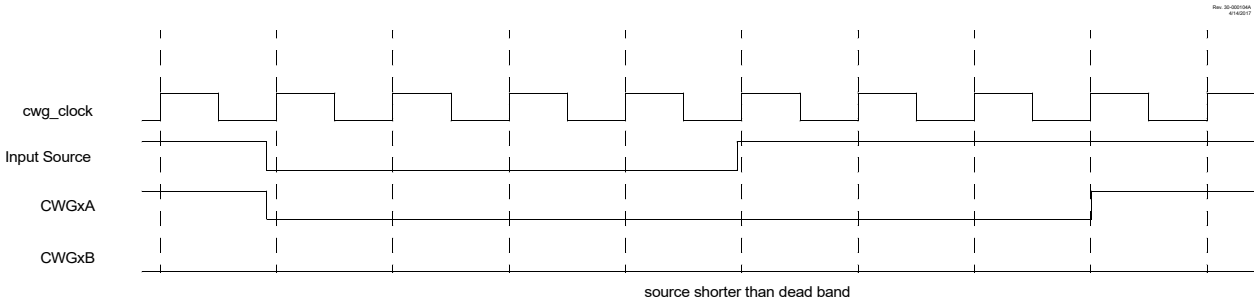
The 24.15.9 CWGxDBF register determines the duration of the dead-band interval on the falling edge of the input source signal. This duration is from zero to 64 periods of CWG clock.

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 CWGxDBF register value is double-buffered. When EN = 0, the buffer is loaded when CWGxDBF is written. When EN = 1, then the buffer will be loaded at the rising edge following the first falling edge of the data input after the LD is set. Refer to the following figure for an example.

Figure 24-13. Dead-Band Operation, CWGxDBR = 0x03, CWGxDBF = 0x06, Source Shorter Than Dead Band



24.10 Dead-Band Jitter

When the rising and falling edges of the input source are asynchronous to the CWG clock, it creates jitter in the dead-band time delay. The maximum jitter is equal to one CWG clock period. Refer to the equations below for more details.

Equation 24-1. Dead-Band Delay Time Calculation

$$T_{DEAD-BAND_MIN} = \frac{1}{F_{CWG_CLOCK}} \bullet DBx<5:0>$$

$$T_{DEAD-BAND_MAX} = \frac{1}{F_{CWG_CLOCK}} \bullet DBx<5:0> + 1$$

$$T_{JITTER} = T_{DEAD-BAND_MAX} - T_{DEAD-BAND_MIN}$$

$$T_{JITTER} = \frac{1}{F_{CWG_CLOCK}}$$

$$T_{DEAD-BAND_MAX} = T_{DEAD-BAND_MIN} + T_{JITTER}$$

Equation 24-2. Dead-Band Delay Example Calculation

$$DBx<5:0> = 0x0A = 10$$

$$F_{CWG_CLOCK} = 8\,MHz$$

$$T_{JITTER} = \frac{1}{8\,MHz} = 125ns$$

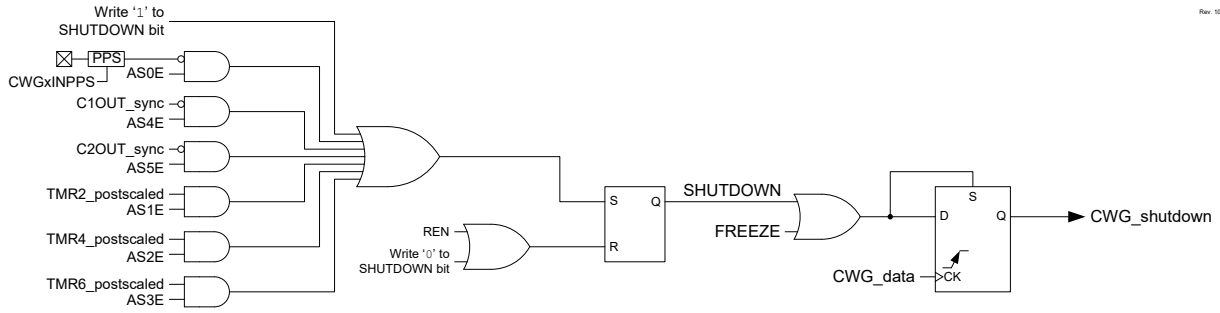
$$T_{DEAD-BAND_MIN} = 125ns \bullet 10 = 125\mu s$$

$$T_{DEAD-BAND_MAX} = 1.25\mu s + 0.125\,\mu s = 1.37\mu s$$

24.11 Auto-Shutdown

Auto-shutdown is a method to immediately override the CWG output levels with specific overrides that allow for safe shutdown of the circuit. The shutdown state can be either cleared automatically or held until cleared by software. The auto-shutdown circuit is illustrated in the following figure.

Figure 24-14. CWG Shutdown Block Diagram



24.11.1 Shutdown

The shutdown state can be entered by either of the following two methods:

- Software generated
- External Input

24.11.1.1 Software Generated Shutdown

Setting the SHUTDOWN bit will force the CWG into the shutdown state.

When the auto-restart is disabled, the shutdown state will persist as long as the SHUTDOWN bit is set.

When auto-restart is enabled, the SHUTDOWN bit will clear automatically and resume operation on the next rising edge event. The SHUTDOWN bit indicates when a shutdown condition exists. The bit may be set or cleared in software or by hardware.

24.11.1.2 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 24.15.6.3 LSBD and 24.15.6.4 LSAC bits. Several input sources can be selected to cause a shutdown condition. All input sources are active-low. The shutdown input sources are individually enabled by the 24.15.7.1 ASyE bits as shown in the following table:

Table 24-2. Shutdown Sources

ASyE	Source
AS5E	CMP2_out (low causes shutdown)
AS4E	CMP1_out (low causes shutdown)
AS3E	TMR6_postscaled (high causes shutdown)
AS2E	TMR4_postscaled (high causes shutdown)
AS1E	TMR2_postscaled (high causes shutdown)
AS0E	Pin selected by CWGxPPS (low causes shutdown)

Important: Shutdown inputs are level sensitive, not edge sensitive. The shutdown state cannot be cleared, except by disabling auto-shutdown, as long as the shutdown input level persists.

24.11.1.3 Pin Override Levels

The levels driven to the CWG outputs during an auto-shutdown event are controlled by the 24.15.6.3 LSBD and 24.15.6.4 LSAC bits. The LSBD bits control CWG1B/D output levels, while the LSAC bits control the CWG1A/C output levels.

24.11.1.4 Auto-Shutdown Interrupts

When an auto-shutdown event occurs, either by software or hardware setting SHUTDOWN, the CWG1IF flag bit of the PIRx register is set.

Related Links

14.13.9 PIR7

24.11.2 Auto-Shutdown Restart

After an auto-shutdown event has occurred, there are two ways to resume operation:

- Software controlled
- Auto-restart

In either case, the shutdown source must be cleared before the restart can take place. That is, either the shutdown condition must be removed, or the corresponding 24.15.7.1 ASyE bit must be cleared.

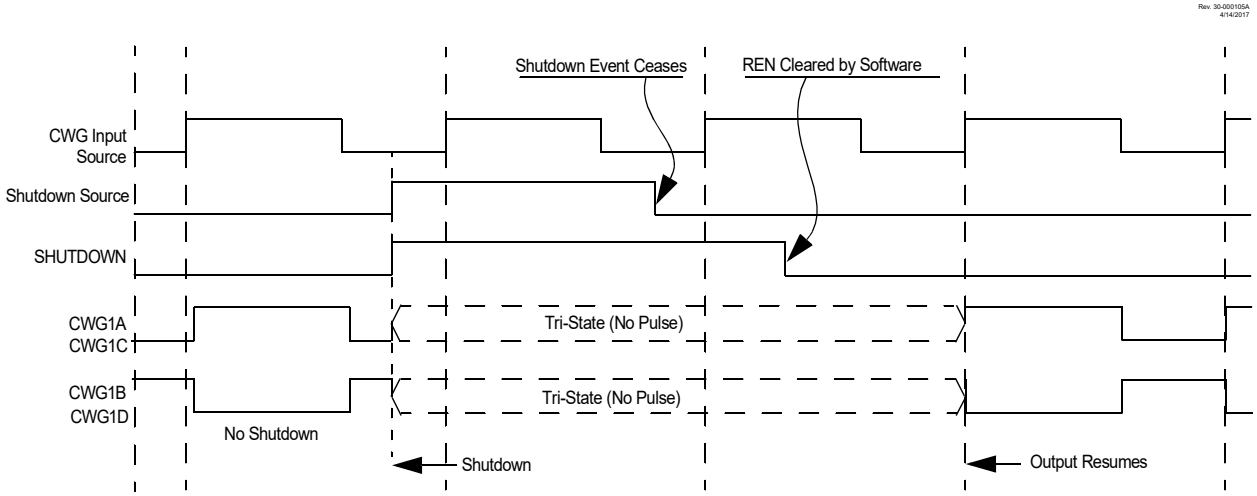
24.11.2.1 Software-Controlled Restart

When the REN bit is clear (REN = 0), the CWG module must be restarted after an auto-shutdown event through software.

Once all auto-shutdown sources are removed, the software must clear SHUTDOWN. Once SHUTDOWN is cleared, the CWG module will resume operation upon the first rising edge of the CWG data input.

Important: The SHUTDOWN bit cannot be cleared in software if the auto-shutdown condition is still present.

Figure 24-15. SHUTDOWN FUNCTIONALITY, AUTO-RESTART DISABLED (REN = 0, LSAC = 01, LSBD = 01)



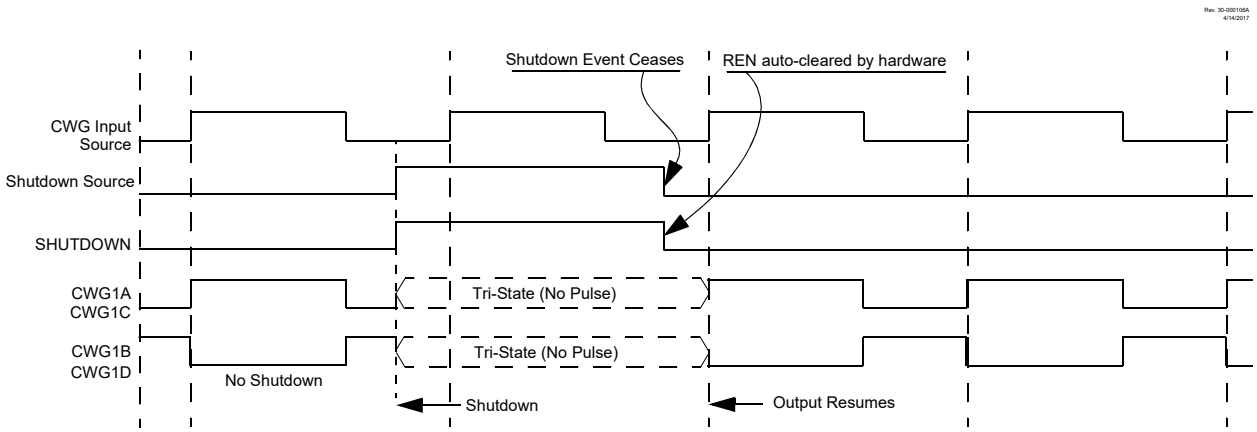
24.11.2.2 Auto-Restart

When the REN bit is set (REN = 1), the CWG module will restart from the shutdown state automatically.

Once all auto-shutdown conditions are removed, the hardware will automatically clear SHUTDOWN. Once SHUTDOWN is cleared, the CWG module will resume operation upon the first rising edge of the CWG data input.

Important: The SHUTDOWN bit cannot be cleared in software if the auto-shutdown condition is still present.

Figure 24-16. SHUTDOWN FUNCTIONALITY, AUTO-RESTART ENABLED (REN = 1, LSAC = 01, LSBD = 01)



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 bits to set the output operating mode.
4. Configure the POLy bits to set the output polarities.
5. Configure the ISM bits to select the data input source.
6. If a steering mode is selected, configure the STRy bits to select the desired output on the CWG outputs.
7. Configure the LSBD and LSAC bits to select the auto-shutdown output override states (this is necessary even if not using auto-shutdown because start-up will be from a shutdown state).
8. If auto-restart is desired, set the REN bit.
9. If auto-shutdown is desired, configure the ASyE bits to select the shutdown source.
10. Set the desired rising and falling dead-band times with the CWGxDBR and CWGxDBF registers.
11. Select the clock source with the CS bits.
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.

24.14 Register Summary - CWG Control

Address	Name	Bit Pos.								
0x0F3B	CWG1CLK	7:0								CS
0x0F3C	CWG1ISM	7:0						ISM[2:0]		
0x0F3D	CWG1DBR	7:0			DBR[5:0]					
0x0F3E	CWG1DBF	7:0			DBF[5:0]					
0x0F3F	CWG1CON0	7:0	EN	LD				MODE[2:0]		
0x0F40	CWG1CON1	7:0			IN		POLD	POLC	POLB	POLA
0x0F41	CWG1AS0	7:0	SHUTDOWN	REN	LSBD[1:0]		LSAC[1:0]			
0x0F42	CWG1AS1	7:0			AS5E	AS4E	AS3E	AS2E	AS1E	AS0E
0x0F43	CWG1STR	7:0	OVRD	OVRC	OVRB	OVRA	STRD	STRC	STRB	STRA

24.15 Register Definitions: CWG Control

Long bit name prefixes for the CWG peripherals are shown in the table below. Refer to the *"Long Bit Names Section"* for more information.

Table 24-3. CWG Bit Name Prefixes

Peripheral	Bit Name Prefix
CWG1	CWG1

Related Links

1.4.2.2 Long Bit Names

24.15.1 CWGxCON0

Name: CWGxCON0
Address: 0x0F3F

CWG Control Register 0

Bit	7	6	5	4	3	2	1	0
	EN	LD				MODE[2:0]		
Access	R/W	R/W/HC				R/W	R/W	R/W
Reset	0	0				0	0	0

Bit 7 – EN CWG1 Enable bit

Value	Description
1	Module is enabled
0	Module is disabled

Bit 6 – LD CWG1 Load Buffers bit(1)

Value	Description
1	Dead-band count buffers to be loaded on CWG data rising edge, following first falling edge after this bit is set
0	Buffers remain unchanged

Bits 2:0 – MODE[2:0] CWG1 Mode bits

Value	Description
111	Reserved
110	Reserved
101	CWG outputs operate in Push-Pull mode
100	CWG outputs operate in Half-Bridge mode
011	CWG outputs operate in Reverse Full-Bridge mode
010	CWG outputs operate in Forward Full-Bridge mode
001	CWG outputs operate in Synchronous Steering mode
000	CWG outputs operate in Asynchronous Steering mode

Note:

1. This bit can only be set after EN = 1; it cannot be set in the same cycle when EN is set.

24.15.2 CWGxCON1

Name: CWGxCON1
Address: 0x0F40

CWG Control Register 1

Bit	7	6	5	4	3	2	1	0
			IN		POLD	POLC	POLB	POLA
Access			RO		R/W	R/W	R/W	R/W
Reset			x		0	0	0	0

Bit 5 – IN CWG Input Value bit (read-only)

Value	Description
1	CWG input is a logic 1
0	CWG input is a logic 0

Bits 0, 1, 2, 3 – POLy CWG Output 'y' Polarity bit

Value	Description
1	Signal output is inverted polarity
0	Signal output is normal polarity

24.15.3 CWGxCLK

Name: CWGxCLK
Address: 0x0F3B

CWGx Clock Input Selection Register

Bit	7	6	5	4	3	2	1	0
								CS
Access								R/W
Reset								0

Bit 0 – CS Clock Source
CWG Clock Source Selection Select bits

Value	Description
1	HFINTOSC (remains operating during Sleep)
0	F_{OSC}

24.15.4 CWGxISM

Name: CWGxISM
Address: 0x0F3C

CWGx Input Selection Register

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

Bits 2:0 – ISM[2:0] CWG Data Input Source Select bits

Table 24-4. CWG Data Input Sources

ISM	Data Source
111	DSM_out
110	CMP2_out
101	CMP1_out
100	PWM4_out
011	PWM3_out
010	CCP2_out
001	CCP1_out
000	Pin selected by CWGxPPS

24.15.5 CWGxSTR

Name: CWGxSTR
Address: 0x0F43

CWG Steering Control Register (1)

Bit	7	6	5	4	3	2	1	0
	OVRD	OVRC	OVRB	OVRA	STRD	STRC	STRB	STRA
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bits 4, 5, 6, 7 – OVRy Steering Data OVR'y' bit

Value	Condition	Description
x	STRy = 1	CWGx'y' output has the CWG data input waveform with polarity control from POLy bit
1	STRy = 0 and POLy = x	CWGx'y' output is high
0	STRy = 0 and POLy = x	CWGx'y' output is low

Bits 0, 1, 2, 3 – STRy STR'y' Steering Enable bit(2)

Value	Description
1	CWGx'y' output has the CWG data input waveform with polarity control from POLy bit
0	CWGx'y' output is assigned to value of OVRy bit

Note:

1. The bits in this register apply only when MODE = '00x' (24.15.1 CWGxCON0, Steering modes).
2. This bit is double-buffered when MODE = '001'.

24.15.6 CWGxAS0

Name: CWGxAS0
Address: 0x0F41

CWG Auto-Shutdown Control Register 0

Bit	7	6	5	4	3	2	1	0
	SHUTDOWN	REN	LSBD[1:0]		LSAC[1:0]			
Access	R/W/HS/HC	R/W	R/W	R/W	R/W	R/W		
Reset	0	0	0	1	0	1		

Bit 7 – SHUTDOWN Auto-Shutdown Event Status bit(1,2)

Value	Description
1	An auto-shutdown state is in effect
0	No auto-shutdown event has occurred

Bit 6 – REN Auto-Restart Enable bit

Value	Description
1	Auto-restart is enabled
0	Auto-restart is disabled

Bits 5:4 – LSBD[1:0] CWGxB and CWGxD Auto-Shutdown State Control bits

Value	Description
11	A logic '1' is placed on CWGxB/D when an auto-shutdown event occurs.
10	A logic '0' is placed on CWGxB/D when an auto-shutdown event occurs.
01	Pin is tri-stated on CWGxB/D when an auto-shutdown event occurs.
00	The inactive state of the pin, including polarity, is placed on CWGxB/D after the required dead-band interval when an auto-shutdown event occurs.

Bits 3:2 – LSAC[1:0] CWGxA and CWGxC Auto-Shutdown State Control bits

Value	Description
11	A logic '1' is placed on CWGxA/C when an auto-shutdown event occurs.
10	A logic '0' is placed on CWGxA/C when an auto-shutdown event occurs.
01	Pin is tri-stated on CWGxA/C when an auto-shutdown event occurs.
00	The inactive state of the pin, including polarity, is placed on CWGxA/C after the required dead-band interval when an auto-shutdown event occurs.

Note:

1. This bit may be written while EN = 0 (24.15.1 CWGxCON0), 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.

24.15.7 CWGxAS1

Name: CWGxAS1
Address: 0x0F42

CWG Auto-Shutdown Control Register 1

Bit	7	6	5	4	3	2	1	0
			AS5E	AS4E	AS3E	AS2E	AS1E	AS0E
Access			R/W	R/W	R/W	R/W	R/W	R/W
Reset			0	0	0	0	0	0

Bits 0, 1, 2, 3, 4, 5 – ASyE CWG Auto-shutdown Source ASyE Enable bit(1)

Table 24-5. Shutdown Sources

ASyE	Source
AS5E	CMP2_out (low causes shutdown)
AS4E	CMP1_out (low causes shutdown)
AS3E	TMR6_postscaled (high causes shutdown)
AS2E	TMR4_postscaled (high causes shutdown)
AS1E	TMR2_postscaled (high causes shutdown)
AS0E	Pin selected by CWGxPPS (low causes shutdown)

Value	Description
1	Auto-shutdown for source ASyE is enabled
0	Auto-shutdown for source ASyE is disabled

Note: This bit may be written while EN = 0 (24.15.1 CWGxCON0), to place the outputs into the shutdown configuration.
The outputs will remain in auto-shutdown state until the next rising edge of the CWG data input after this bit is cleared.

24.15.8 CWGxDBR

Name: CWGxDBR
Address: 0x0F3D

CWG Rising Dead-Band Count Register

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

Bits 5:0 – DBR[5:0] CWG Rising Edge Triggered Dead-Band Count bits
Reset States: POR/BOR = xxxxxx
All Other Resets = uuuuuu

Value	Description
n	Dead band is active no less than n, and no more than n+1, CWG clock periods after the rising edge
0	0 CWG clock periods. Dead-band generation is bypassed

24.15.9 CWGxDBF

Name: CWGxDBF
Address: 0x0F3E

CWG Falling Dead-Band Count Register

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

Bits 5:0 – DBF[5:0] CWG Falling Edge Triggered Dead-Band Count bits
Reset States: POR/BOR = xxxxxx
All Other Resets = uuuuuu

Value	Description
n	Dead band is active no less than n, and no more than n+1, CWG clock periods after the falling edge
0	0 CWG clock periods. Dead-band generation is bypassed

25. (DSM) Data Signal Modulator 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:

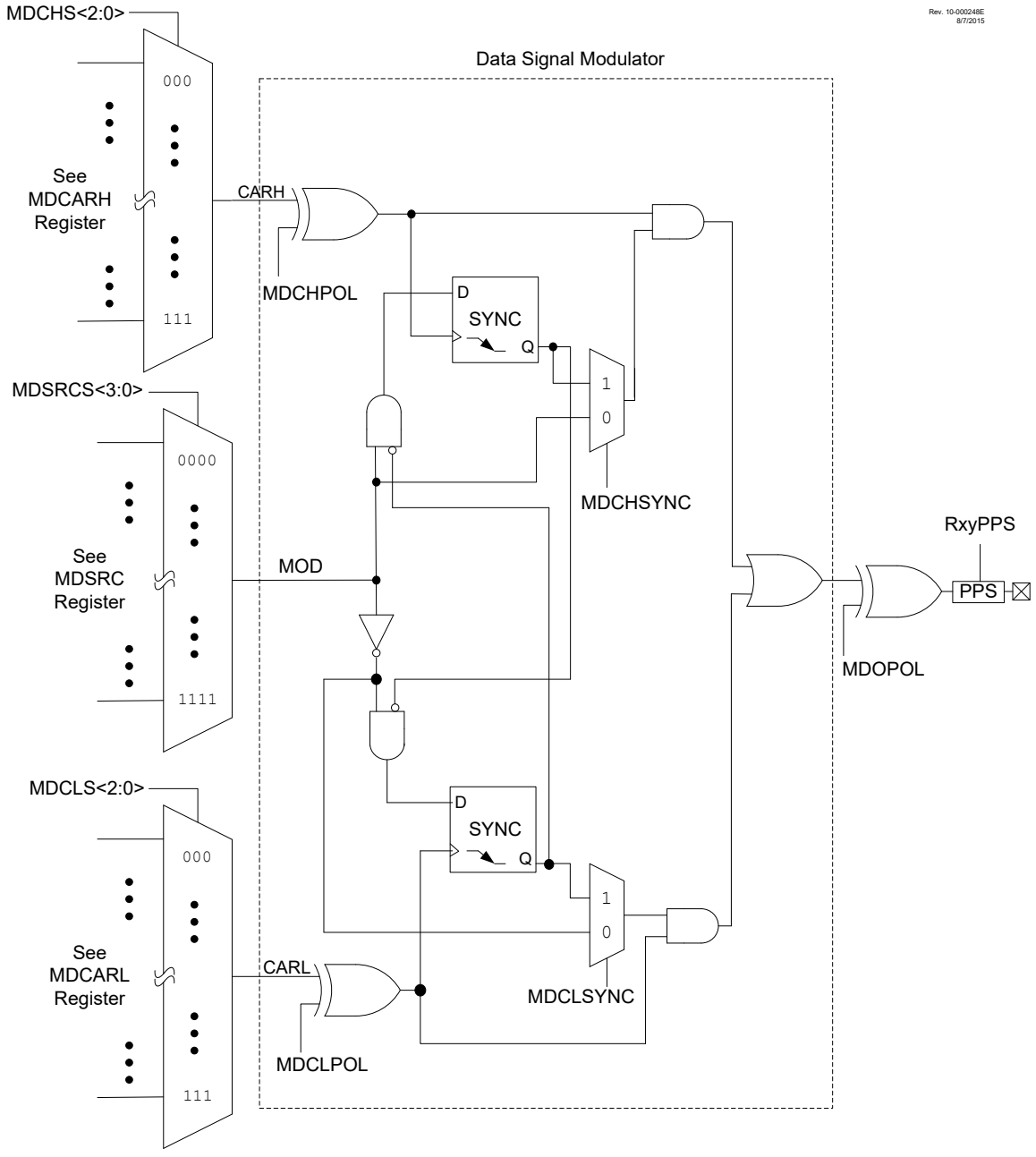
- Frequency-Shift Keying (FSK)
- Phase-Shift Keying (PSK)
- On-Off Keying (OOK)

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

The figure below shows a Simplified Block Diagram of the Data Signal Modulator peripheral.

Figure 25-1. Simplified Block Diagram of the Data Signal Modulator



25.1 DSM Operation

The DSM module can be enabled by setting the EN bit in the MDCON0 register. Clearing the EN bit, disables the output of the module but retain the carrier and source signal selections. The module will resume operation when the EN bit is set again. The output of the DSM module can be rerouted to several pins using the RxyPPS register. When the EN bit is cleared the output pin is held low.

25.2 Modulator Signal Sources

The modulator signal can be supplied from the following sources selected with the SRCS bits:

Table 25-1. MDSRC Selection MUX Connections

SRCS<3:0>	Connection
1011-1111	Reserved
1010	MSSP1 - SDO
1001	EUSART1 TX (TX/CK output)
1000	EUSART1 RX (DT output)
0111	CMP2 OUT
0110	CMP1 OUT
0101	PWM4 OUT
0100	PWM3 OUT
0011	CCP2 OUT
0010	CCP1 OUT
0001	MDBIT
0000	Pin selected by MDSRCPPS

25.3 Carrier Signal Sources

The carrier high signal and carrier low signal can be supplied from the following sources.

The carrier high signal is selected by configuring the CHS bits.

Table 25-2. MDCARH Source Selections

MDCARH	
CHS<2:0>	Connection
111	PWM4 OUT
110	PWM3 OUT
101	CCP2 OUT
100	CCP1 OUT
011	CLKREF output

MDCARH	
CHS<2:0>	Connection
010	HFINTOSC
001	FOSC (system clock)
000	Pin selected by MDCARHPPS

The carrier low signal is selected by configuring the CLS bits.

Table 25-3. MDCARL Source Selections

MDCARL	
CLS<2:0>	Connection
111	PWM4 OUT
110	PWM3 OUT
101	CCP2 OUT
100	CCP1 OUT
011	CLKREF output
010	HFINTOSC
001	FOSC (system clock)
000	Pin selected by MDCARLPPS

25.4 Carrier Synchronization

During the time when the DSM switches between carrier high and carrier low signal sources, the carrier data in the modulated output signal can become truncated. To prevent this, the carrier signal can be synchronized to the modulator signal. When synchronization is enabled, the carrier pulse that is being mixed at the time of the transition is allowed to transition low before the DSM switches over to the next carrier source.

Synchronization is enabled separately for the carrier high and carrier low signal sources. Synchronization for the carrier high signal is enabled by setting the CHSYNC bit. Synchronization for the carrier low signal is enabled by setting the CLSYNC bit.

The figures below show the timing diagrams of using various synchronization methods.

Figure 25-2. On Off Keying (OOK) Synchronization

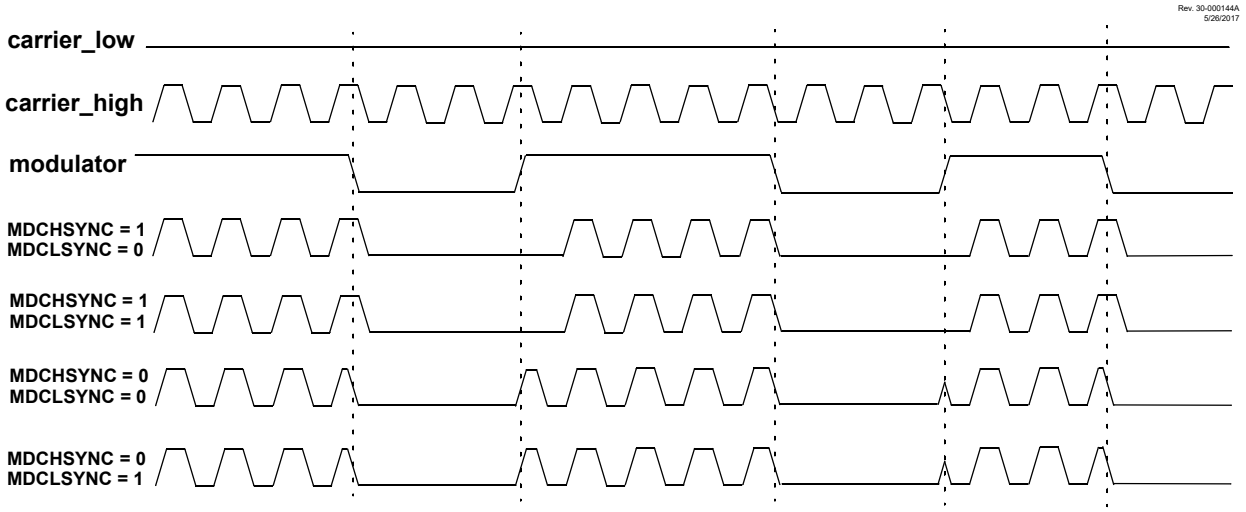


Figure 25-3. No Synchronization (MDCHSYNC = 0, MDCLSYNC = 0)

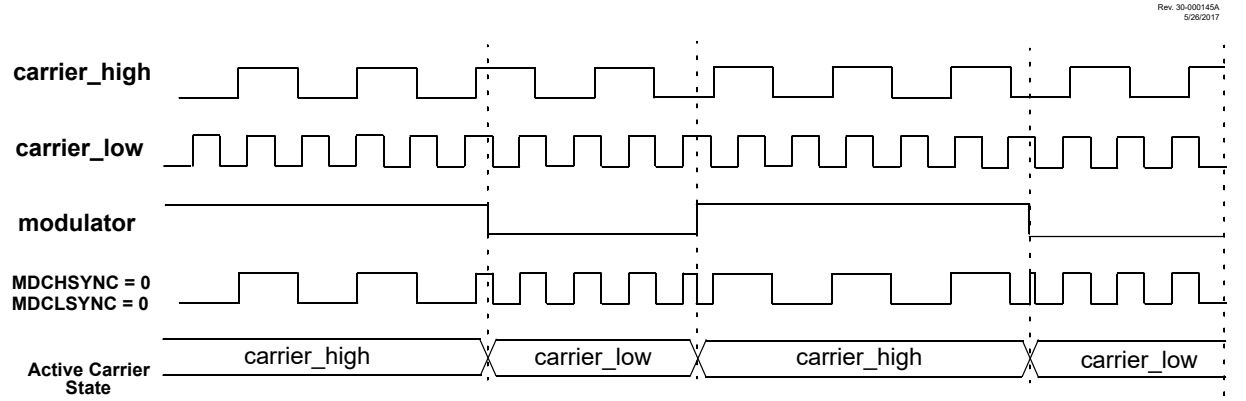


Figure 25-4. Carrier High Synchronization (MDCHSYNC = 1, MDCLSYNC = 0)

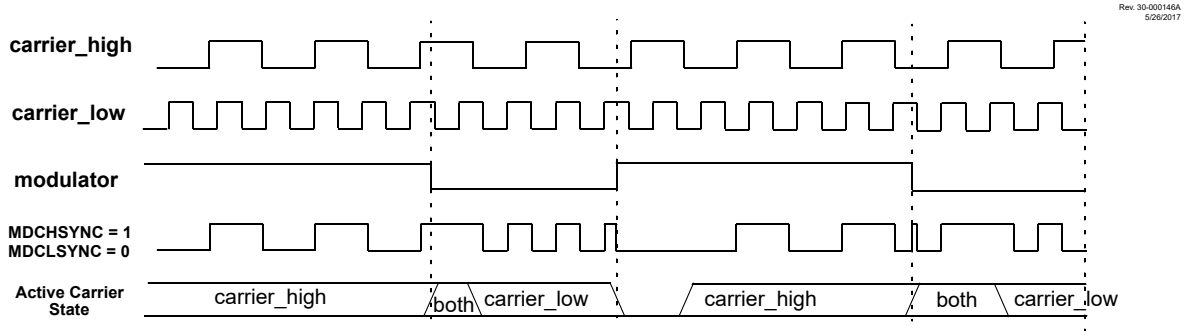


Figure 25-5. Carrier Low Synchronization (MDCHSYNC = 0, MDCLSYNC = 1)

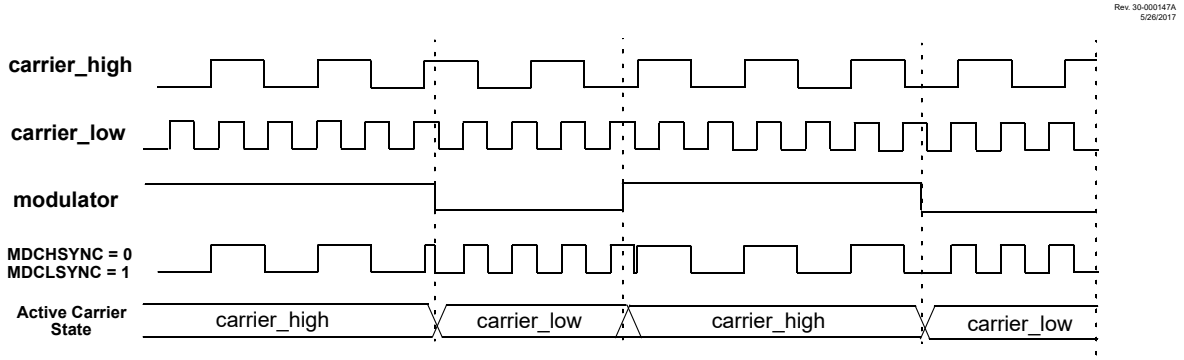
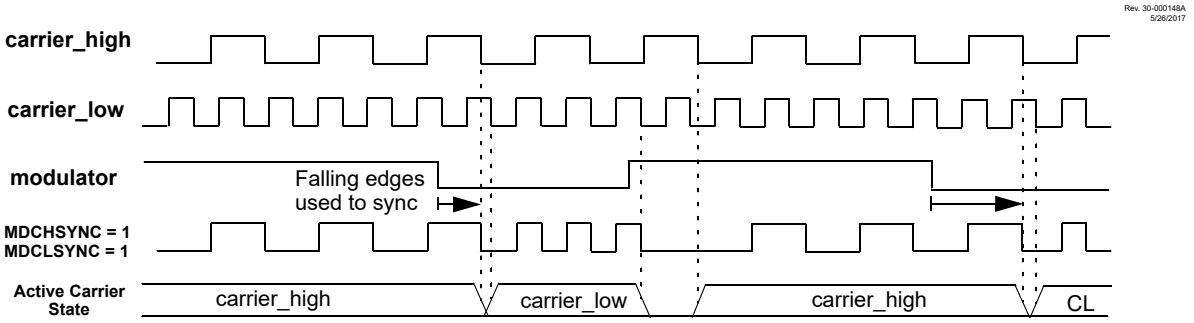


Figure 25-6. Full Synchronization (MDCHSYNC = 1, MDCLSYNC = 1)



25.5 Carrier Source Polarity Select

The signal provided from any selected input source for the carrier high and carrier low signals can be inverted. Inverting the signal for the carrier high and low source is enabled by setting the CHPOL bit and the CLPOL bit, respectively.

25.6 Programmable Modulator Data

The BIT bit can be selected as the modulation source. This gives the user the ability to provide software driven modulation.

25.7 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 OPOL bit.

25.8 Operation in Sleep Mode

The DSM can still operate during Sleep, if the Carrier and Modulator input sources are also still operable during Sleep. Refer to *"Power-Saving Operation Modes"* for more details.

Related Links

6.4 Peripheral Operation in Power-Saving Modes

25.9 Effects of a Reset

Upon any device Reset, the DSM module is disabled. The user's firmware is responsible for initializing the module before enabling the output. All the registers are reset to their default values.

25.10 Peripheral Module Disable

The DSM module can be completely disabled using the PMD module to achieve maximum power saving. When the DSMMD bit of PMDx register is set, the DSM module is completely disabled. This puts the module in its lowest power consumption state. When enabled again all the registers of the DSM module default to POR status.

Related Links

7.4 Register Definitions: Peripheral Module Disable

25.11 Register Summary - DSM

Address	Name	Bit Pos.								
0x0F4C	MDCON0	7:0	EN		OUT	OPOL				BIT
0x0F4D	MDCON1	7:0			CHPOL	CHSYNC			CLPOL	CLSYNC
0x0F4E	MDSRC	7:0					SRCS[3:0]			
0x0F4F	MDCARL	7:0						CLS[2:0]		
0x0F50	MDCARH	7:0						CHS[2:0]		

25.12 Register Definitions: Modulation Control

Long bit name prefixes for the Modulation Control peripherals are shown in the table below. Refer to the *"Long Bit Names Section"* for more information.

Table 25-4. Modulation Control Long Bit Name Prefixes

Peripheral	Bit Name Prefix
MD	MD

Related Links

1.4.2.2 Long Bit Names

25.12.1 MDCON0

Name: MDCON0
Address: 0xF4C

Modulation Control Register 0

Bit	7	6	5	4	3	2	1	0
	EN		OUT	OPOL				BIT
Access	R/W		R/W	R/W				R/W
Reset	0		0	0				0

Bit 7 – EN Modulator Module Enable bit

Value	Description
1	Modulator module is enabled and mixing input signals
0	Modulator module is disabled and has no output

Bit 5 – OUT Modulator Output bit
Displays the current output value of the modulator module.

Bit 4 – OPOL Modulator Output Polarity Select bit

Value	Description
1	Modulator output signal is inverted; idle high output
0	Modulator output signal is not inverted; idle low output

Bit 0 – BIT Modulation Source Select Input bit
Allows software to manually set modulation source input to module

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.

25.12.2 MDCON1

Name: MDCON1
Address: 0xF4D

Modulation Control Register 1

Bit	7	6	5	4	3	2	1	0
			CHPOL	CHSYNC			CLPOL	CLSYNC
Access			R/W	R/W			R/W	R/W
Reset			0	0			0	0

Bit 5 – CHPOL Modulator High Carrier Polarity Select bit

Value	Description
1	Selected high carrier signal is inverted
0	Selected high carrier signal is not inverted

Bit 4 – CHSYNC Modulator High Carrier Synchronization Enable bit

Value	Description
1	Modulator waits for a falling edge on the high time carrier signal before allowing a switch to the low time carrier
0	Modulator output is not synchronized to the high time carrier signal

Bit 1 – CLPOL Modulator Low Carrier Polarity Select bit

Value	Description
1	Selected low carrier signal is inverted
0	Selected low carrier signal is not inverted

Bit 0 – CLSYNC Modulator Low Carrier Synchronization Enable bit

Value	Description
1	Modulator waits for a falling edge on the low time carrier signal before allowing a switch to the high time carrier
0	Modulator output is not synchronized to the low time carrier signal

Note:

1. Narrowed carrier pulse widths or spurs may occur in the signal stream if the carrier is not synchronized.

25.12.3 MDCARH

Name: MDCARH
Address: 0xF50

Modulation High Carrier Control Register

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

Bits 2:0 – CHS[2:0] Modulator Carrier High Selection bits

Table 25-5. MDCARH Source Selections

MDCARH	
CHS<2:0>	**Connection**
111	PWM4 OUT
110	PWM3 OUT
101	CCP2 OUT
100	CCP1 OUT
011	CLKREF output
010	HFINTOSC
001	FOSC (system clock)
000	Pin selected by MDCARHPPS

25.12.4 MDCARL

Name: MDCARL
Address: 0xF4F

Modulation Low Carrier Control Register

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

Bits 2:0 – CLS[2:0] Modulator Carrier Low Input Selection bits

Table 25-6. MDCARL Source Selections

MDCARL	
CLS<2:0>	**Connection**
111	PWM4 OUT
110	PWM3 OUT
101	CCP2 OUT
100	CCP1 OUT
011	CLKREF output
010	HFINTOSC
001	FOSC (system clock)
000	Pin selected by MDCARLPPS

25.12.5 MDSRC

Name: MDSRC
Address: 0xF4E

Modulation Source Control Register

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

Bits 3:0 – SRCS[3:0] Modulator Source Selection bits

Table 25-7. MDSRC Selection MUX Connections

SRCS<3:0>	Connection
1011-1111	Reserved
1010	MSSP1 - SDO
1001	EUSART1 TX (TX/CK output)
1000	EUSART1 RX (DT output)
0111	CMP2 OUT
0110	CMP1 OUT
0101	PWM4 OUT
0100	PWM3 OUT
0011	CCP2 OUT
0010	CCP1 OUT
0001	MDBIT
0000	Pin selected by MDSRCPPS

26. (MSSP) Master Synchronous Serial Port Module

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 MSSP module can operate in one of two modes:

- Serial Peripheral Interface (SPI)
- Inter-Integrated Circuit (I^2C)

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 I^2C 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

26.1 SPI Mode Overview

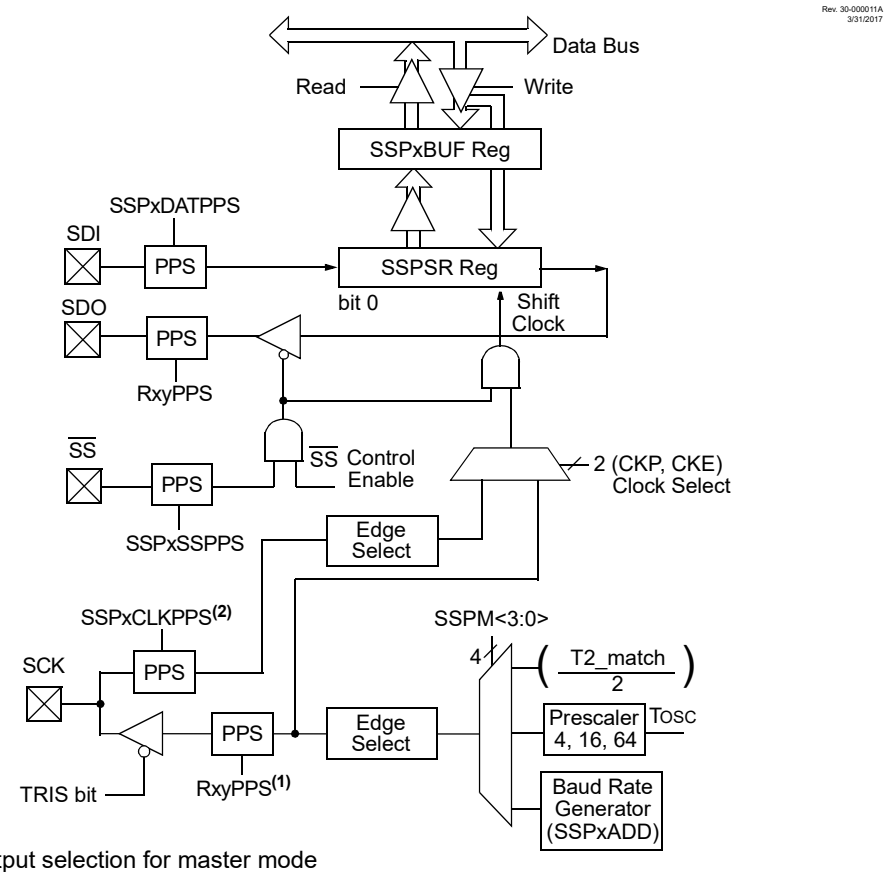
The Serial Peripheral Interface (SPI) bus is a synchronous serial data communication bus that operates in Full-Duplex mode. Devices communicate in a master/slave environment where the master device initiates the communication. A slave device is controlled through a Chip Select known as Slave Select.

The SPI bus specifies four signal connections:

- Serial Clock (SCK)
- Serial Data Out (SDO)
- Serial Data In (SDI)
- Slave Select ($\overline{SS}$)

The following figure shows the block diagram of the MSSP module when operating in SPI mode.

Figure 26-1. MSSP Block Diagram (SPI mode)



Note 1: Output selection for master mode

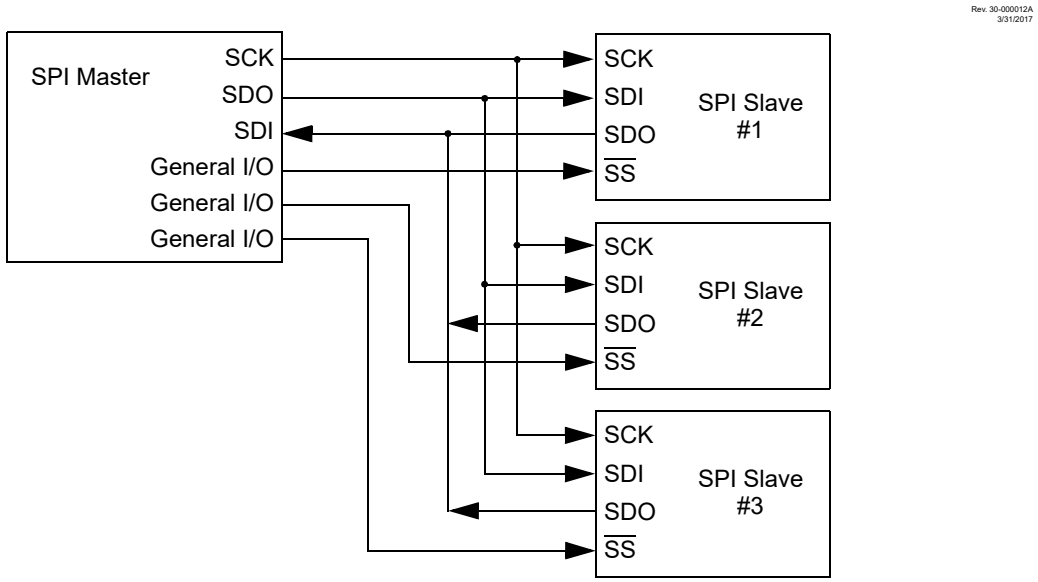
2: Input selection for slave and master mode

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.

The figure below 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 26-2. SPI Master and Multiple Slave Connection



26.1.1 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 for 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.

One of the five SPI master modes uses the SSPxADD value to determine the Baud Rate Generator clock frequency. More information on the Baud Rate Generator is available in 26.7 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.

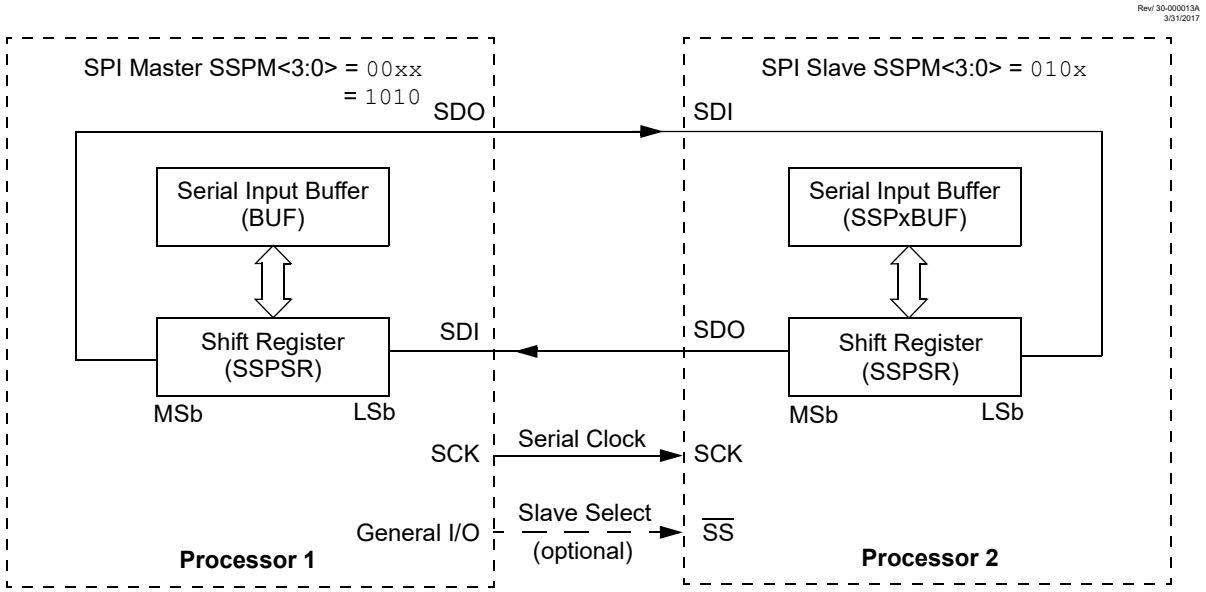
26.2 SPI Mode Operation

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.

The following figure shows a typical connection between two processors configured as master and slave devices.

Figure 26-3. SPI Master/Slave Connection



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, 26.9.2.3 SSPEN, must be set. To reset or reconfigure SPI mode, clear the SSPEN bit, re-initialize the SSPxCONx registers and then set the SSPEN bit. The SDI, SDO, SCK and $\overline{SS}$ serial port pins are selected with the PPS controls. 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
- The RxyPPS and SSPxCLKPPS controls must select the same pin
- $\overline{SS}$ 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, 26.9.1.8 BF, 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, 26.9.2.1 WCOL, 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, 26.9.1.8 BF, 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.

26.2.1 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 26-3) 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 26.9.2.4 CKP bit and the 26.9.1.2 CKE bit. This then, would give waveforms for SPI communication as shown in Figure 26-4, Figure 26-6, Figure 26-7 and Figure 26-8, where the MSB is transmitted first. In Master mode, the SPI clock rate (bit rate) is user programmable to be one of the following:

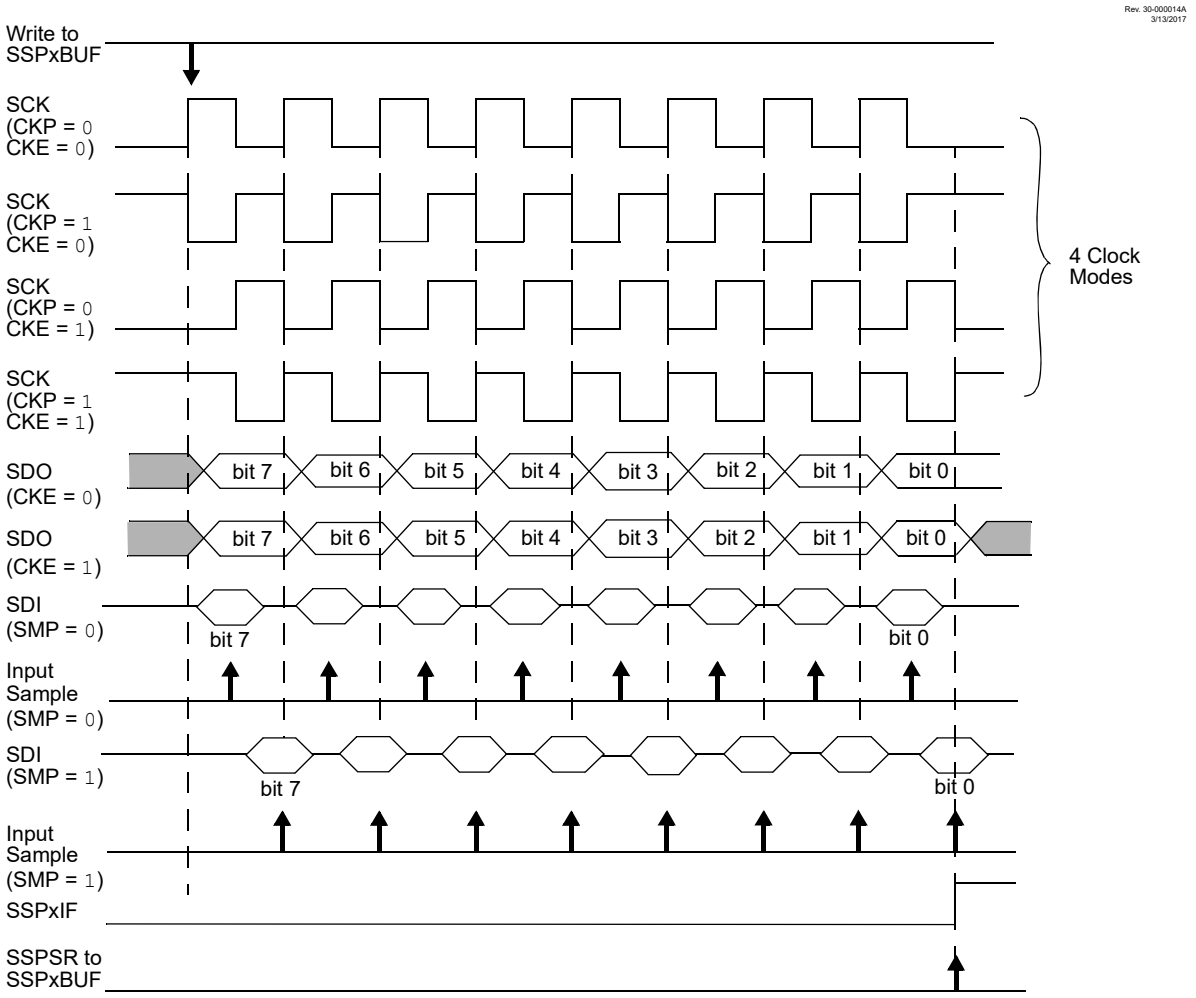
- $F_{OSC}/4$ (or T_{CY})
- $F_{OSC}/16$ (or 4 * T_{CY})
- $F_{OSC}/64$ (or 16 * T_{CY})
- Timer2 output/2
- $F_{OSC}/(4 * (SSPxADD + 1))$

Figure 26-4 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.

Important: 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. The pin that is selected using the SSPxCLKPPS register should also be made a digital I/O. This is done by clearing the corresponding ANSEL bit.

Figure 26-4. SPI Mode Waveform (Master Mode)



26.2.2 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 26.9.2.4 CKP bit.

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.

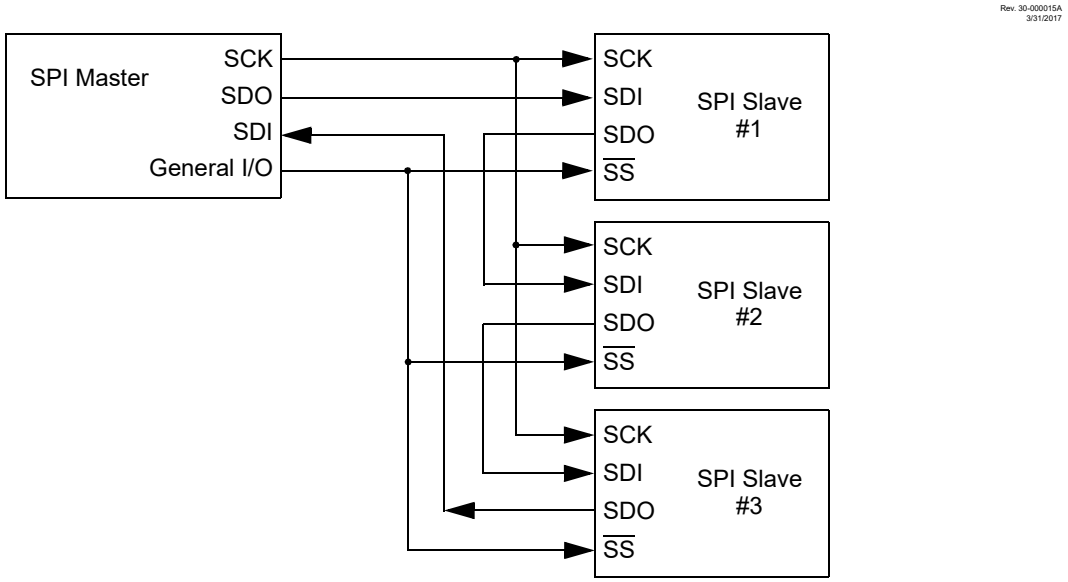
26.2.3 Daisy-Chain Configuration

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.

The following figure shows the block diagram of a typical daisy-chain connection when operating in SPI mode.

Figure 26-5. SPI Daisy-Chain Connection



In a daisy-chain configuration, only the most recent byte on the bus is required by the slave. Setting the 26.9.4.4 BOEN bit 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.

26.2.4 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 (26.9.2.5 SSPM = 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 (26.9.2.5 SSPM = 0100), the SPI module will reset if the $\overline{SS}$ pin is set to V_{DD}.

2. When the SPI is used in Slave mode with 26.9.1.2 CKE set; the user must enable $\overline{SS}$ pin control.
3. While operated in SPI Slave mode the 26.9.1.1 SMP bit 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 26-6. Slave Select Synchronous Waveform

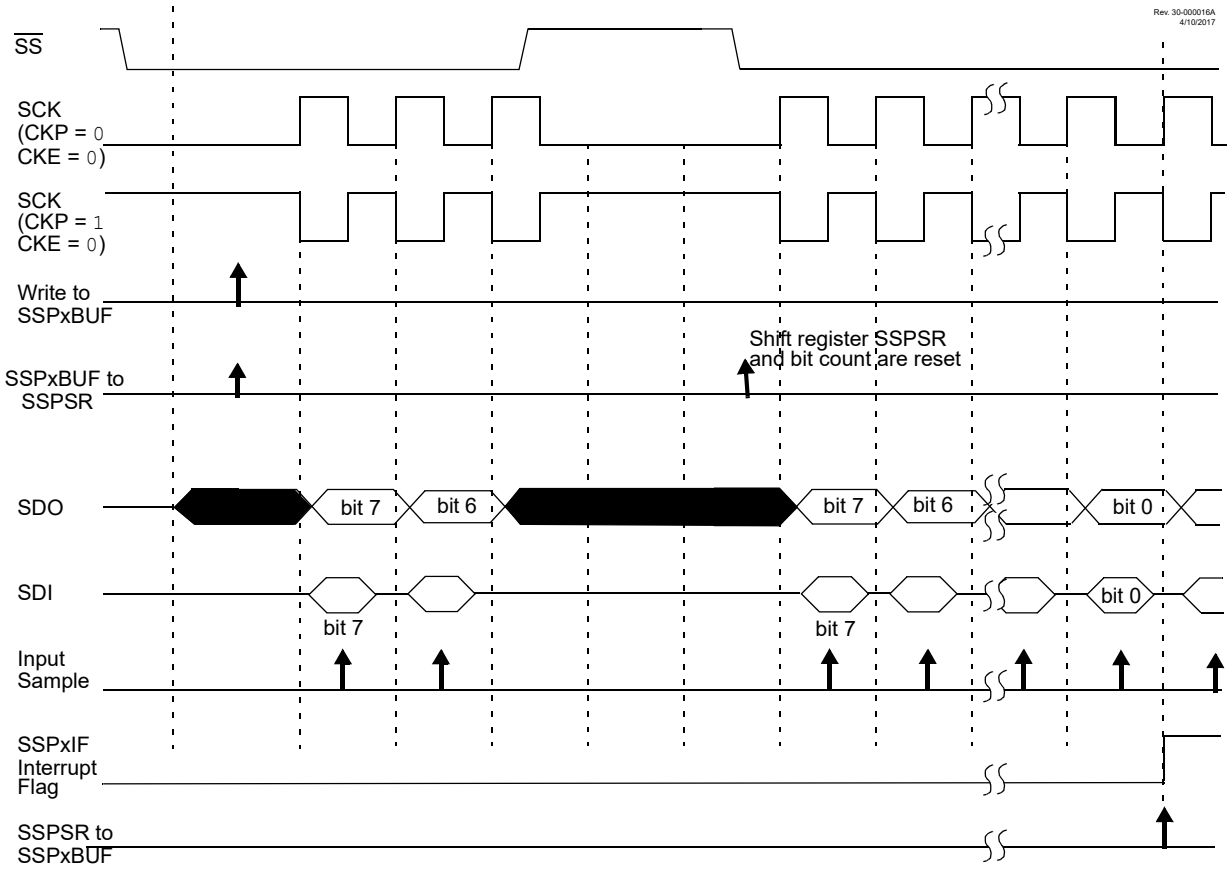


Figure 26-7. SPI Mode Waveform (Slave Mode with CKE = 0)

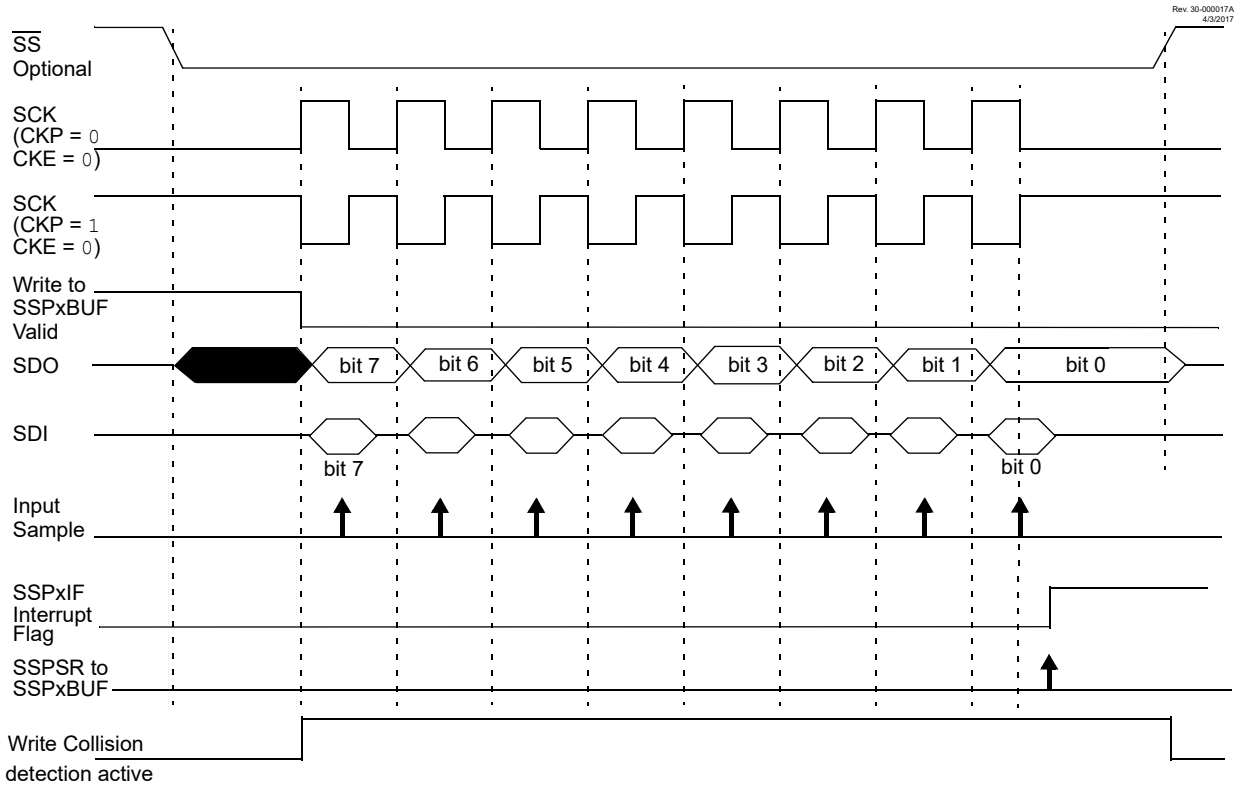
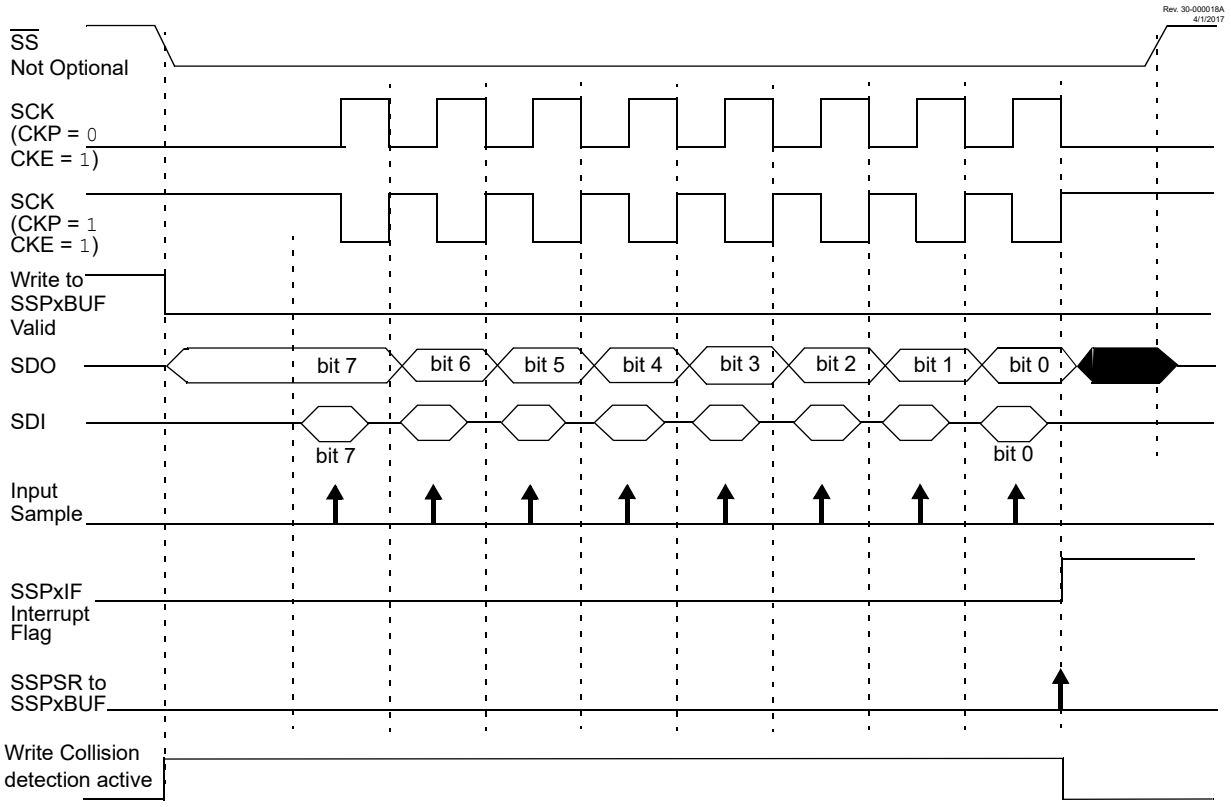


Figure 26-8. SPI Mode Waveform (Slave Mode with CKE = 1)



26.2.5 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.

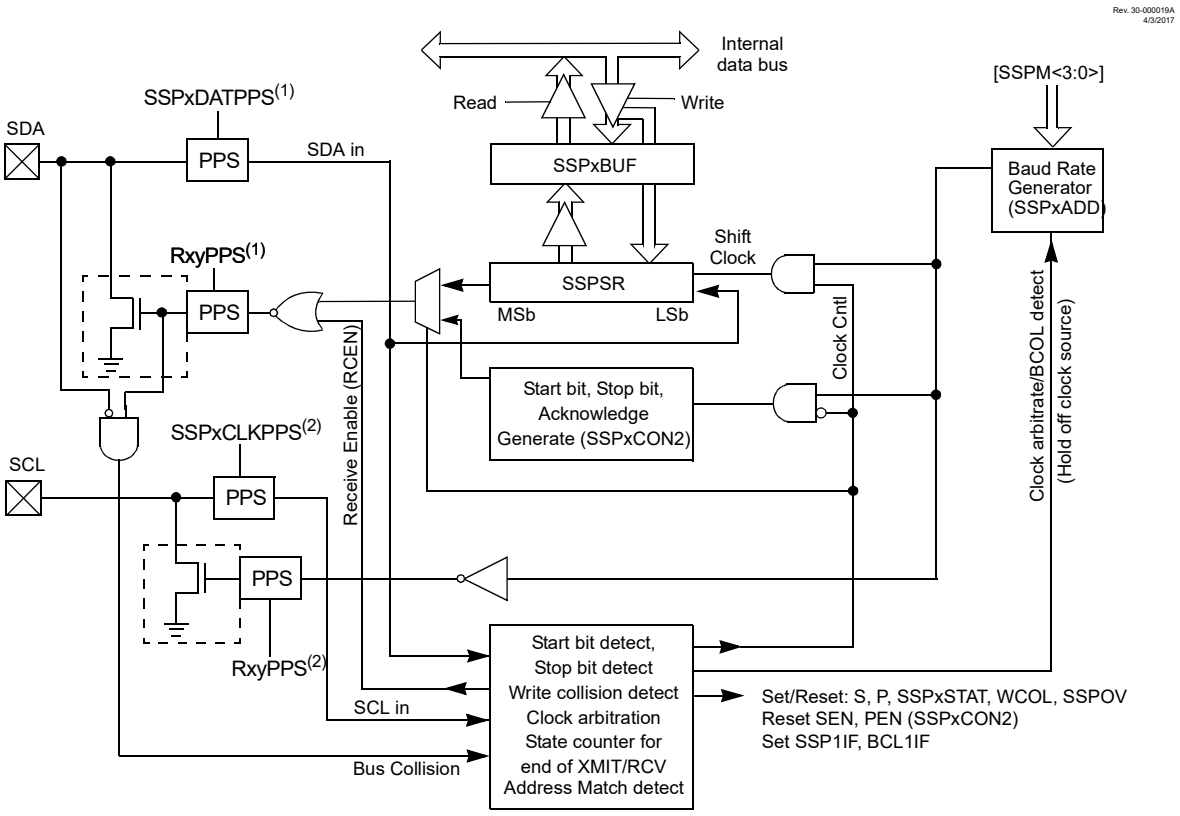
In SPI Slave mode, the SPI Transmit/Receive Shift register operates asynchronously to the device. This allows the device to be placed in Sleep mode and data to be shifted into the SPI Transmit/Receive Shift register. When all eight bits have been received, the MSSP interrupt flag bit will be set and if enabled, will wake the device.

26.3 I^2C Mode Overview

The Inter-Integrated Circuit (I^2C) bus is a multi-master serial data communication bus. Devices communicate in a master/slave environment where the master devices initiate the communication. A

slave device is controlled through addressing. The following two diagrams show block diagrams of the I²C Master and Slave modes, respectively.

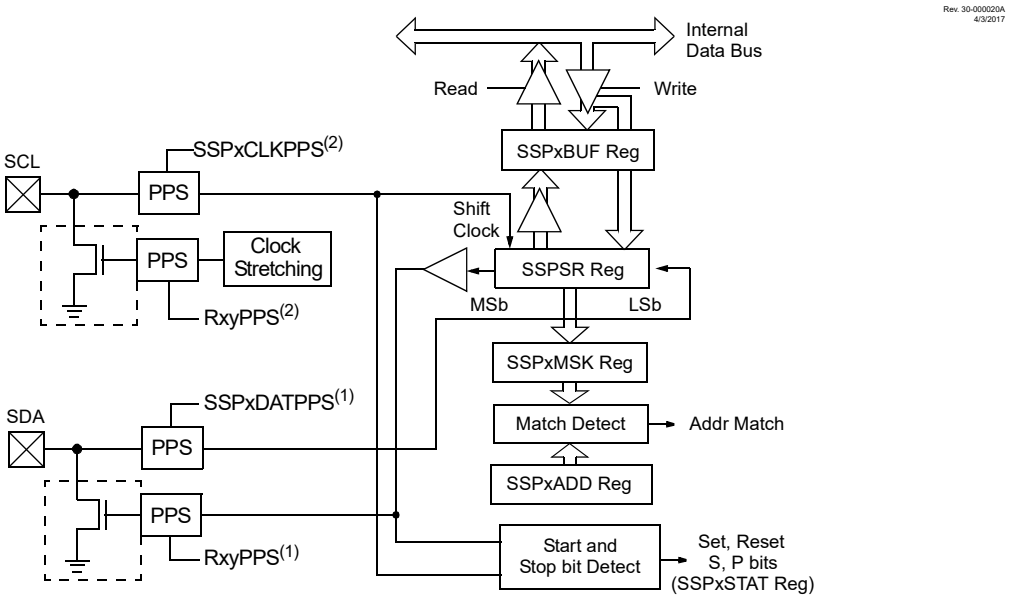
Figure 26-9. MSSP Block Diagram (I²C Master mode)



Note 1: SDA pin selections must be the same for input and output

2: SCL pin selections must be the same for input and output

Figure 26-10. MSSP Block Diagram (I²C Slave mode)



Note 1: SDA pin selections must be the same for input and output

2: SCL pin selections must be the same for input and output

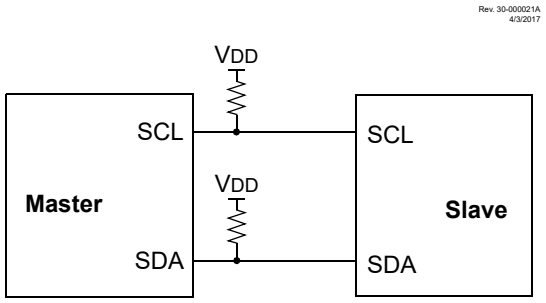
The I²C bus specifies two signal connections:

- 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.

The following diagram shows a typical connection between two processors configured as master and slave devices.

Figure 26-11. I²C Master/ Slave Connection



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 $\overline{ACK}$. 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.

The Acknowledge bit ($\overline{ACK}$) 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 $\overline{ACK}$ 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 $\overline{ACK}$ 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 $\overline{ACK}$ 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 $\overline{ACK}$ bit when it is in receive mode.

The I^2C 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.

26.3.1 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.

26.4 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.

26.4.1 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.

26.4.2 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.

26.4.3 Byte Format

All communication in I^2C 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.

26.4.4 Definition of I^2C Terminology

There is language and terminology in the description of I^2C communication that have definitions specific to I^2C. 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 I^2C specification.

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.

TERM	Description
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/$\overline{W}$ bit clear, and is ready to clock in data.
Read Request	Master sends an address byte with the R/$\overline{W}$ 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.

26.4.5 SDA and SCL Pins

Selection of any I^2C mode with the 26.9.2.3 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 I^2C 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.

26.4.6 SDA Hold Time

The hold time of the SDA pin is selected by the 26.9.4.5 SDAHT bit. 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.

I^2C Bus Terms

26.4.7 Start Condition

The I^2C 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 26-12 shows wave forms for Start and Stop conditions.

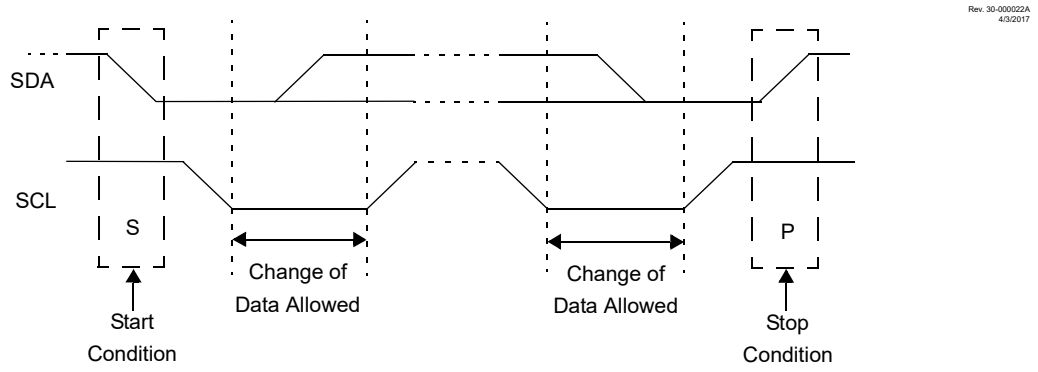
A bus collision can occur on a Start condition if the module samples the SDA line low before asserting it low. This does not conform to the I^2C Specification that states no bus collision can occur on a Start.

26.4.8 Stop Condition

A Stop condition is a transition of the SDA line from low-to-high state while the SCL line is high.

Important: 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.

Figure 26-12. I²C Start and Stop Conditions



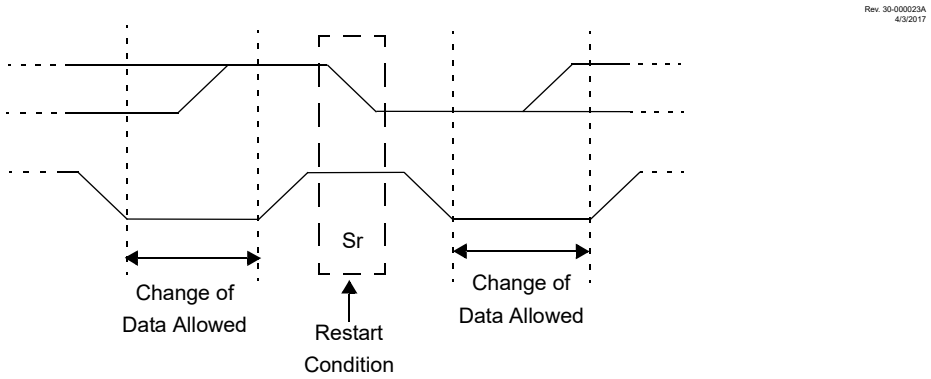
26.4.9 Restart Condition

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 26-13 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/$\overline{W}$ bit set. The slave logic will then hold the clock and prepare to clock out data.

After a full match with R/$\overline{W}$ clear in 10-bit mode, a prior match flag is set and maintained until a Stop condition, a high address with R/$\overline{W}$ clear, or high address match fails.

Figure 26-13. I²C Restart Condition



26.4.10 Start/Stop Condition Interrupt Masking

The 26.9.4.3 SCIE and 26.9.4.2 PCIE bits can enable the generation of an interrupt in Slave modes that do not typically support this function. These bits will have no effect in Slave modes where interrupt on Start and Stop detect are already enabled.

26.4.11 Acknowledge Sequence

The ninth SCL pulse for any transferred byte in I^2C 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 ($\overline{ACK}$) 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 $\overline{ACK}$ is placed in the 26.9.3.2 ACKSTAT bit.

Slave software, when the 26.9.4.7 AHEN and 26.9.4.8 DHEN bits are set, allow the user to set the $\overline{ACK}$ value sent back to the transmitter. The 26.9.3.3 ACKDT bit is set/cleared to determine the response.

Slave hardware will generate an $\overline{ACK}$ response if both the AHEN and DHEN bits are clear. However, tf the 26.9.1.8 BF bit or the 26.9.2.2 SSPOV bit are set when a byte is received then the $\overline{ACK}$ will not be sent by the slave.

When the module is addressed, after the eighth falling edge of SCL on the bus, the 26.9.4.1 ACKTIM bit is set. The ACKTIM bit indicates the acknowledge time of the active bus. The ACKTIM Status bit is only active when either the AHEN bit or DHEN bit is enabled.

26.5 I^2C Slave Mode Operation

The MSSP Slave mode operates in one of four modes selected by the 26.9.2.5 SSPM bits. 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.

26.5.1 Slave Mode Addresses

The SSPxADD register 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 SSPxMSK register affects the address matching process. See 26.5.9 SSP Mask Register for more information.

26.5.1.1 I^2C 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.

26.5.1.2 I^2C 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 26.9.1.7 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/$\overline{W}$ 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.

26.5.2 Slave Reception

When the R/$\overline{W}$ bit of a matching received address byte is clear, the 26.9.1.6 R/W bit 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 26.9.1.8 BF is set, or bit 26.9.2.2 SSPOV is set. The 26.9.4.4 BOEN bit modifies this operation. For more information see SSPxCON3.

An MSSP interrupt is generated for each transferred data byte. Flag bit, SSPxIF, must be cleared by software.

When the 26.9.3.8 SEN bit is set, SCL will be held low (clock stretch) following each received byte. The clock must be released by setting the 26.9.2.4 CKP bit, except sometimes in 10-bit mode. See 26.5.6.2 10-bit Addressing Mode for more detail.

26.5.2.1 7-bit Addressing Reception

This section describes a standard sequence of events for the MSSP module configured as an I^2C slave in 7-bit Addressing mode. Figure 26-14 and Figure 26-15 is used as a visual reference for this description.

This is a step by step process of what typically must be done to accomplish I^2C communication.

1. Start bit detected.
2. 26.9.1.5 S bit is set; SSPxIF is set if interrupt on Start detect is enabled.
3. Matching address with 26.9.1.6 R/W bit clear is received.
4. The slave pulls SDA low sending an $\overline{ACK}$ 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 26.9.3.8 SEN = `1`; Slave software sets 26.9.2.4 CKP bit to release the SCL line.
8. The master clocks out a data byte.
9. Slave drives SDA low sending an $\overline{ACK}$ 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 26.9.1.4 P bit, and the bus goes idle.

26.5.2.2 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 $\overline{ACK}$ 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 I^2C communication. Figure 26-16 displays a module using both address and data holding. Figure 26-17 includes the operation with the SEN bit of the SSPxCON2 register set.

1. 26.9.1.5 S bit is set; SSPxIF is set if interrupt on Start detect is enabled.
2. Matching address with 26.9.1.6 R/W bit clear is clocked in. SSPxIF is set and 26.9.2.4 CKP cleared after the eighth falling edge of SCL.
3. Slave clears the SSPxIF.
4. Slave can look at the 26.9.4.1 ACKTIM bit to determine if the SSPxIF was after or before the $\overline{ACK}$.
5. Slave reads the address value from SSPxBUF, clearing the BF flag.
6. Slave sets $\overline{ACK}$ value clocked out to the master by setting 26.9.3.3 ACKDT.
7. Slave releases the clock by setting 26.9.2.4 CKP.
8. SSPxIF is set after an $\overline{ACK}$, not after a NACK.
9. If 26.9.3.8 SEN = 1, the slave hardware will stretch the clock after the $\overline{ACK}$.
10. Slave clears SSPxIF.

Important: 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 26.9.2.4 CKP cleared after eighth falling edge of SCL for a received data byte.
12. Slave looks at 26.9.4.1 ACKTIM bit 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 $\overline{ACK}$ = 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 26.9.1.4 P bit.

Figure 26-14. I²C Slave, 7-bit Address, Reception (SEN = 0, AHEN = 0, DHEN = 0)

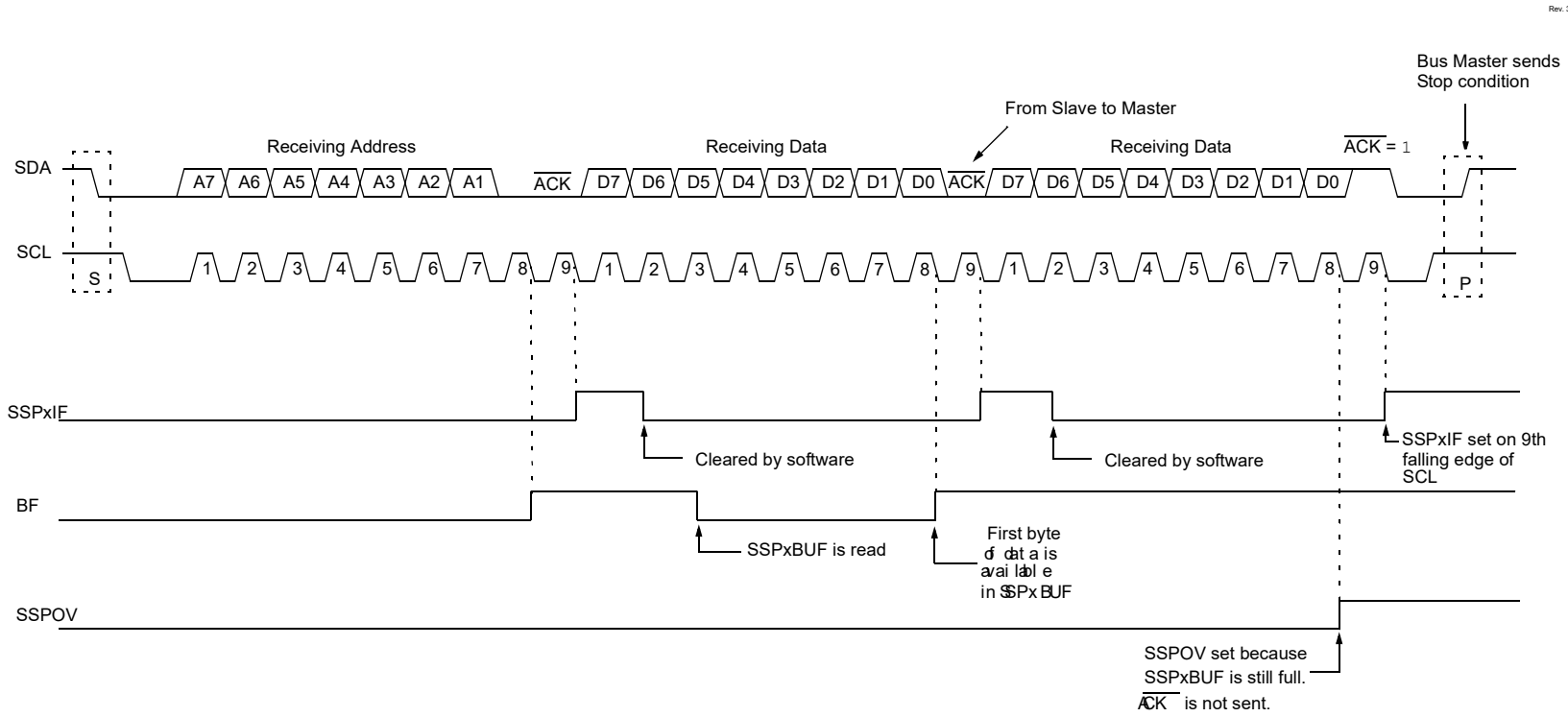
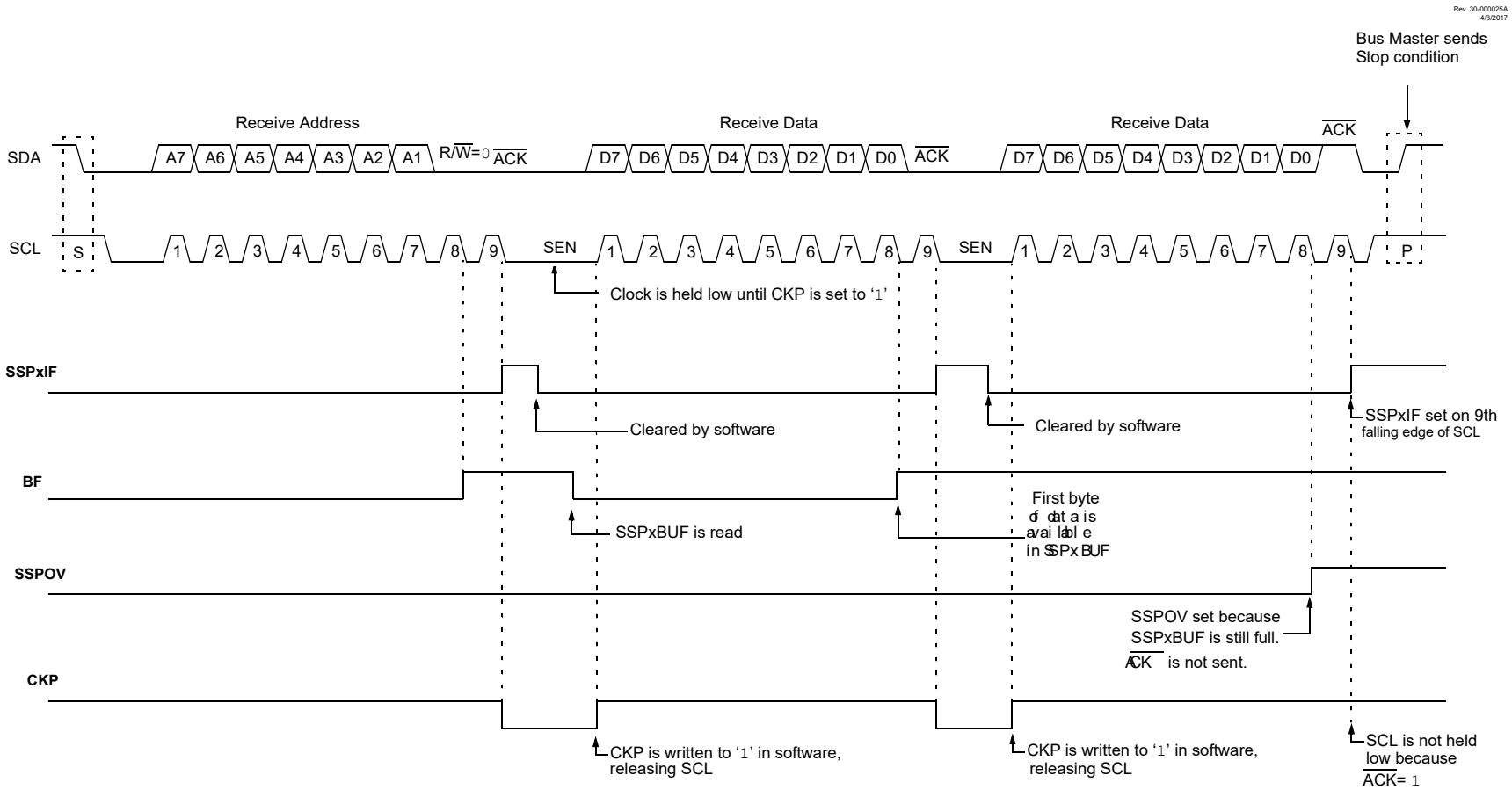
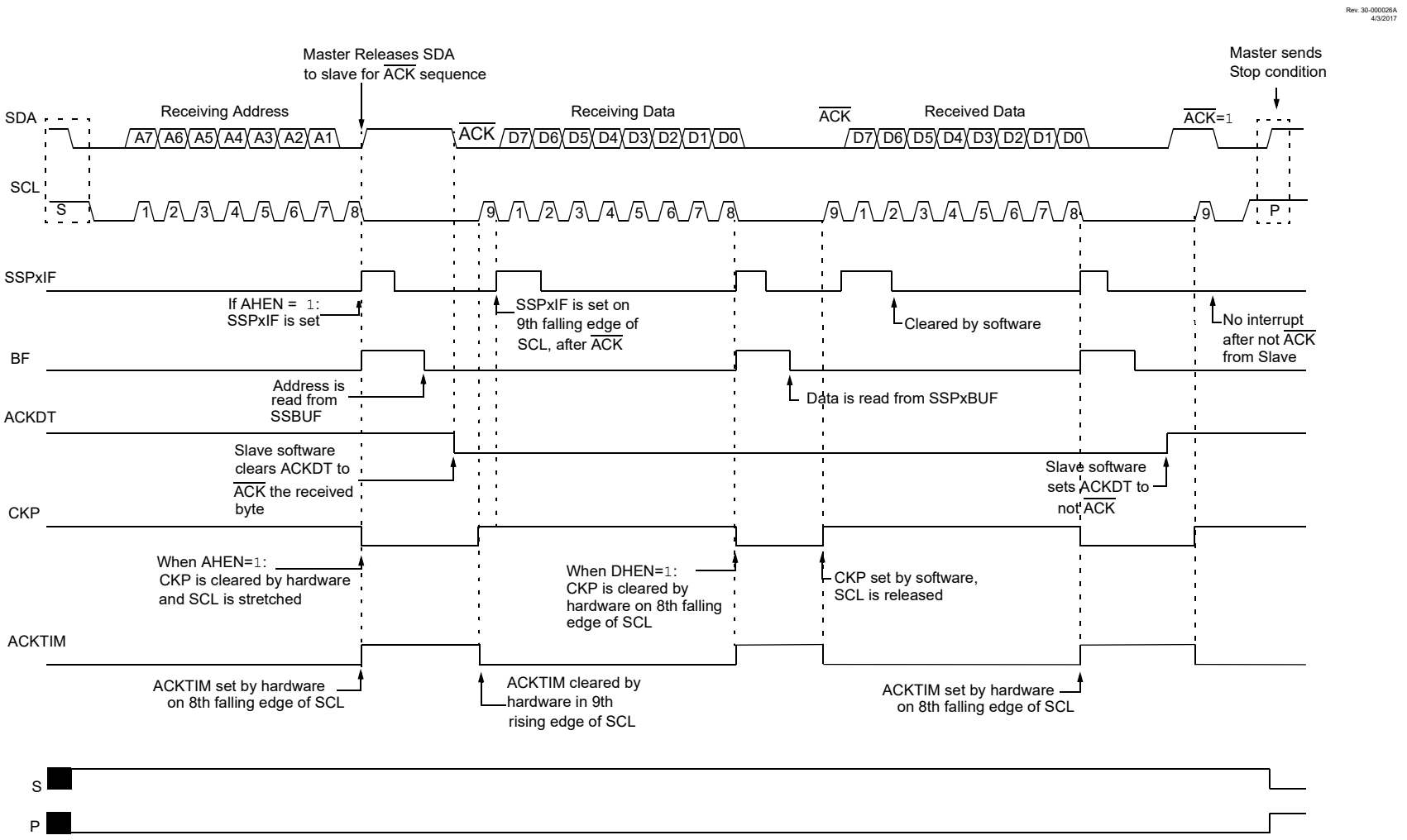


Figure 26-15. I²C Slave, 7-bit Address, Reception (SEN = 1, AHEN = 0, DHEN = 0)



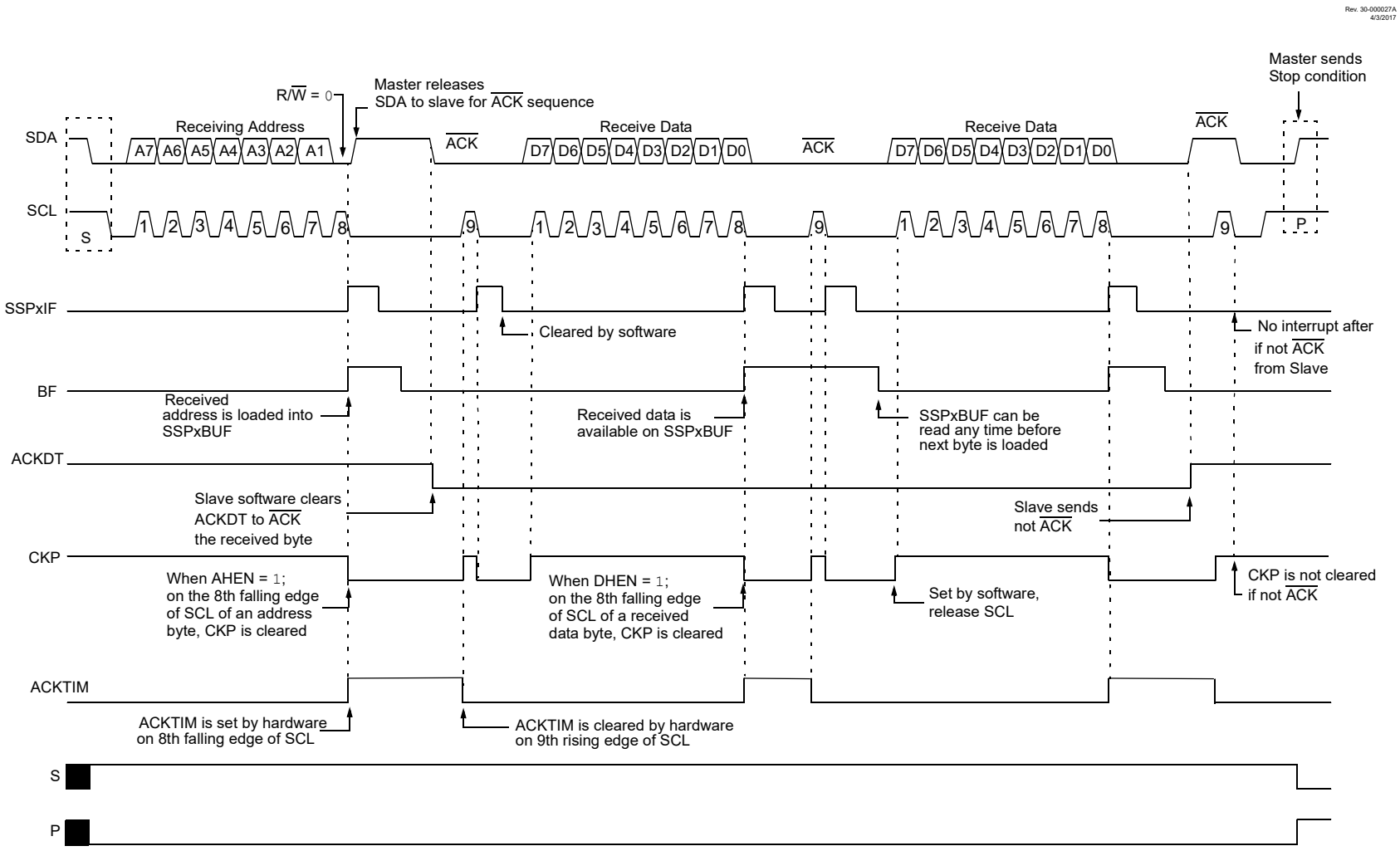
© 2018 Microchip Technology Inc.

Figure 26-16. I²C Slave, 7-bit Address, Reception (SEN = 0, AHEN = 1, DHEN = 1)



© 2018 Microchip Technology Inc.

Figure 26-17. I²C Slave, 7-bit Address, Reception (SEN = 1, AHEN = 1, DHEN = 1)



26.5.3 Slave Transmission

When the R/$\overline{W}$ bit of the incoming address byte is set and an address match occurs, the 26.9.1.6 R/W bit 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 26.9.2.4 CKP bit and the SCL pin is held low (see 26.5.6 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 26.9.2.4 CKP bit. 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 26.9.3.2 ACKSTAT bit. 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.

26.5.3.1 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 26.9.4.6 SBCDE bit is set, the BCLxIF bit of the PIRx 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.

26.5.3.2 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 26-18 can be used as a reference to this list.

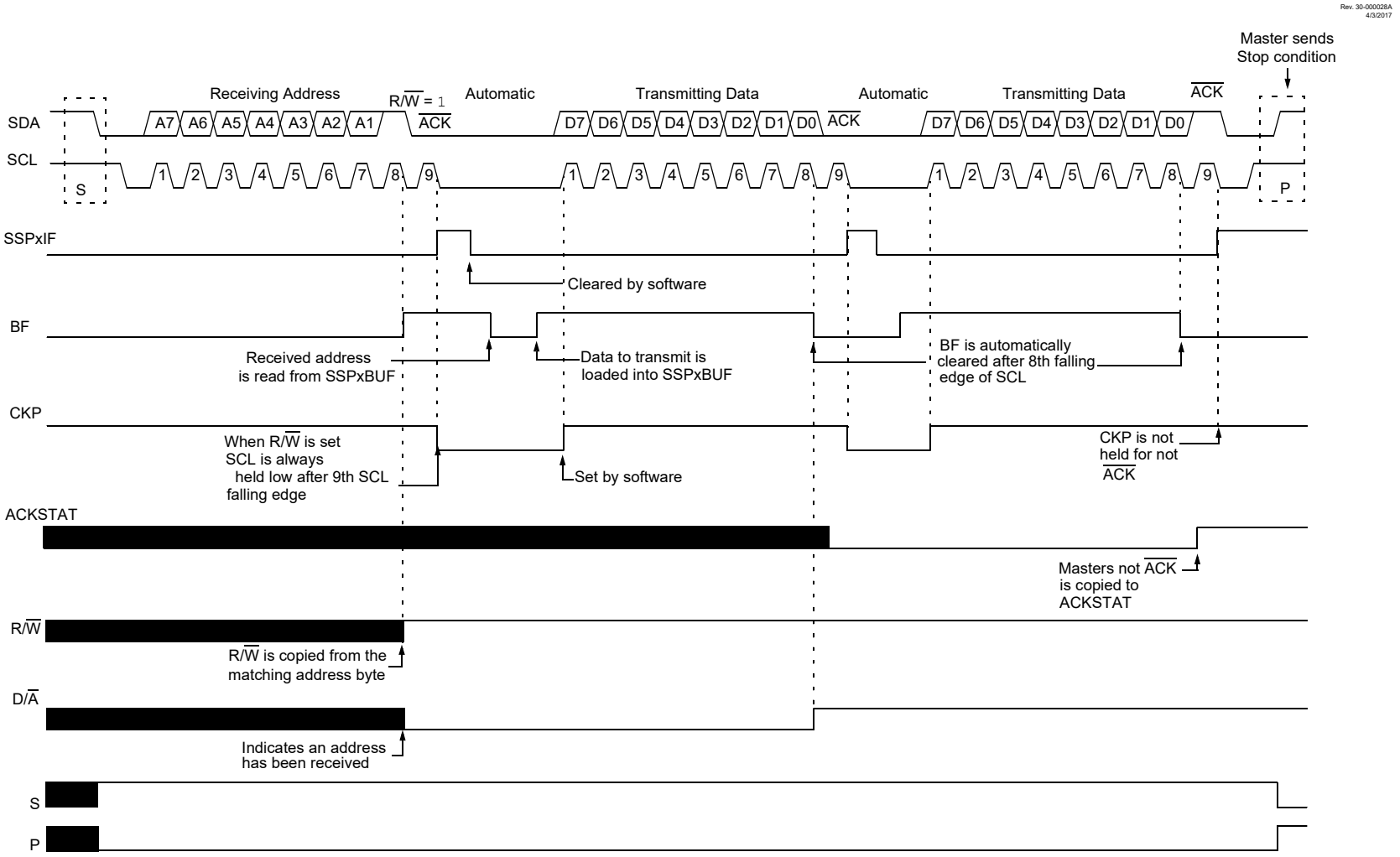
1. Master sends a Start condition on SDA and SCL.
2. 26.9.1.5 S bit is set; SSPxIF is set if interrupt on Start detect is enabled.
3. Matching address with R/$\overline{W}$ bit set is received by the Slave setting SSPxIF bit.
4. Slave hardware generates an $\overline{ACK}$ and sets SSPxIF.
5. SSPxIF bit is cleared by user.
6. Software reads the received address from SSPxBUF, clearing BF.
7. R/$\overline{W}$ is set so 26.9.2.4 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 26.9.3.2 ACKSTAT bit to see if the master wants to clock out more data.

Important:

1. If the master $\overline{\text{ACK}}$s then the clock will be stretched.
2. ACKSTAT is the only bit updated on the rising edge of the ninth SCL clock instead of the falling edge.

13. Steps 9-13 are repeated for each transmitted byte.
14. If the master sends a not $\overline{\text{ACK}}$; 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 26-18. I²C Slave, 7-bit Address, Transmission (AHEN = 0)



26.5.3.3 7-bit Transmission with Address Hold Enabled

Setting the 26.9.4.7 AHEN bit 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 26-19 displays a standard waveform of a 7-bit address slave transmission with AHEN enabled.

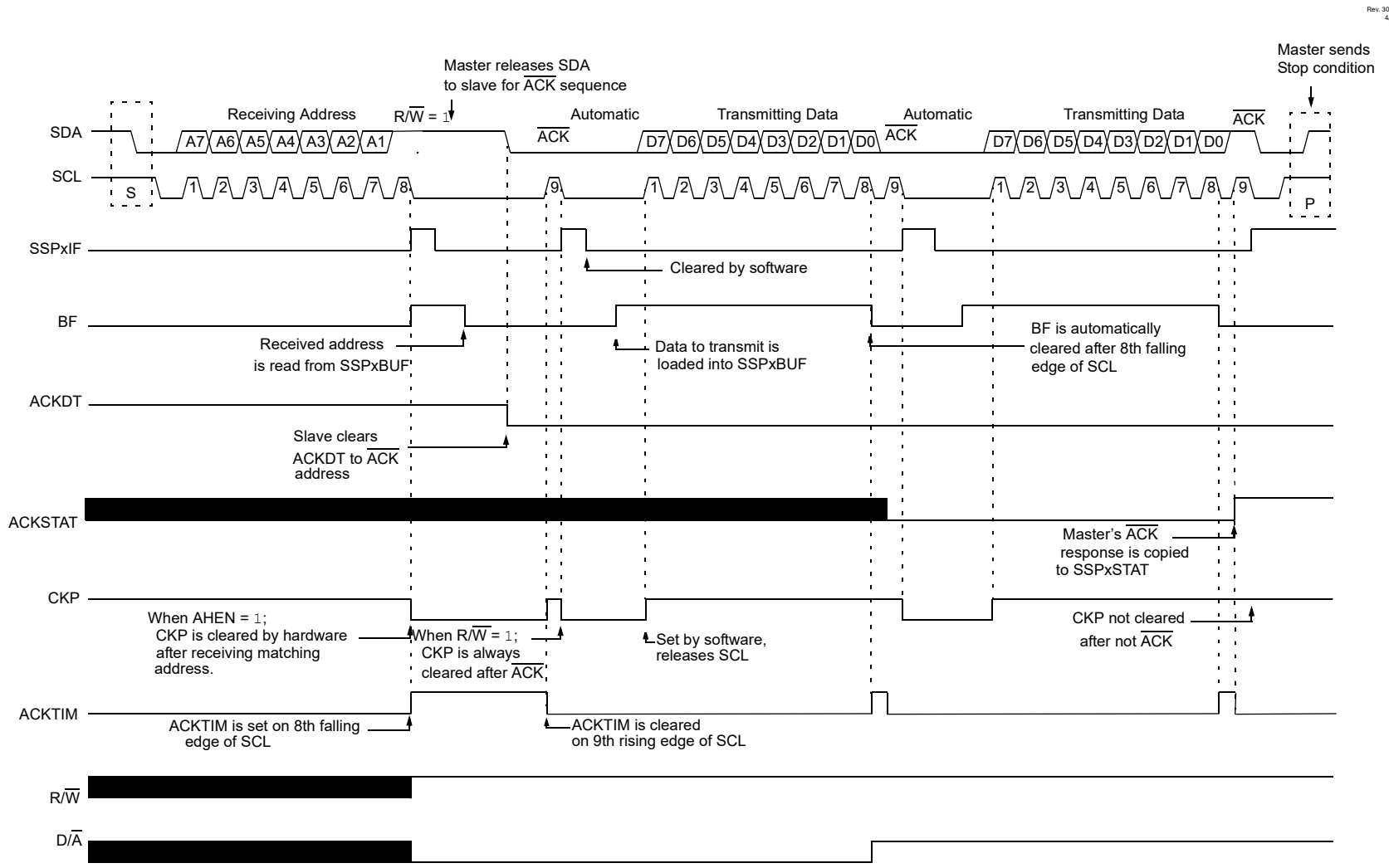
1. Bus starts Idle.
2. Master sends Start condition; the 26.9.1.5 S bit is set; SSPxIF is set if interrupt on Start detect is enabled.
3. Master sends matching address with 26.9.1.6 R/W bit set. After the eighth falling edge of the SCL line the 26.9.2.4 CKP bit is cleared and SSPxIF interrupt is generated.
4. Slave software clears SSPxIF.
5. Slave software reads the 26.9.4.1 ACKTIM, 26.9.1.6 R/W and 26.9.1.3 D/A bits 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 26.9.3.3 ACKDT bit accordingly.
8. Slave sets the 26.9.2.4 CKP bit releasing SCL.
9. Master clocks in the $\overline{ACK}$ value from the slave.
10. Slave hardware automatically clears the 26.9.2.4 CKP bit and sets SSPxIF after the $\overline{ACK}$ if the 26.9.1.6 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.

Important: SSPxBUF cannot be loaded until after the $\overline{ACK}$.

13. Slave sets the 26.9.2.4 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 26.9.3.2 ACKSTAT bit.
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.

Important: 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 26-19. I²C Slave, 7-bit Address, Transmission (AHEN = 1)



26.5.4 Slave Mode 10-bit Address Reception

This section describes a standard sequence of events for the MSSP module configured as an I^2C slave in 10-bit Addressing mode.

Figure 26-20 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 I^2C communication.

1. Bus starts Idle.
2. Master sends Start condition; 26.9.1.5 S bit is set; SSPxIF is set if interrupt on Start detect is enabled.
3. Master sends matching high address with 26.9.1.6 R/W bit clear; 26.9.1.7 UA bit 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.



Important: Updates to the SSPxADD register are not allowed until after the $\overline{\text{ACK}}$ sequence.

9. Slave sends $\overline{\text{ACK}}$ and SSPxIF is set.



Important: 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. 26.9.2.4 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 26.9.3.8 SEN bit is set, 26.9.2.4 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 26.9.3.8 SEN is set the slave sets 26.9.2.4 CKP to release the SCL.
18. Steps 13-17 repeat for each received byte.
19. Master sends Stop to end the transmission.

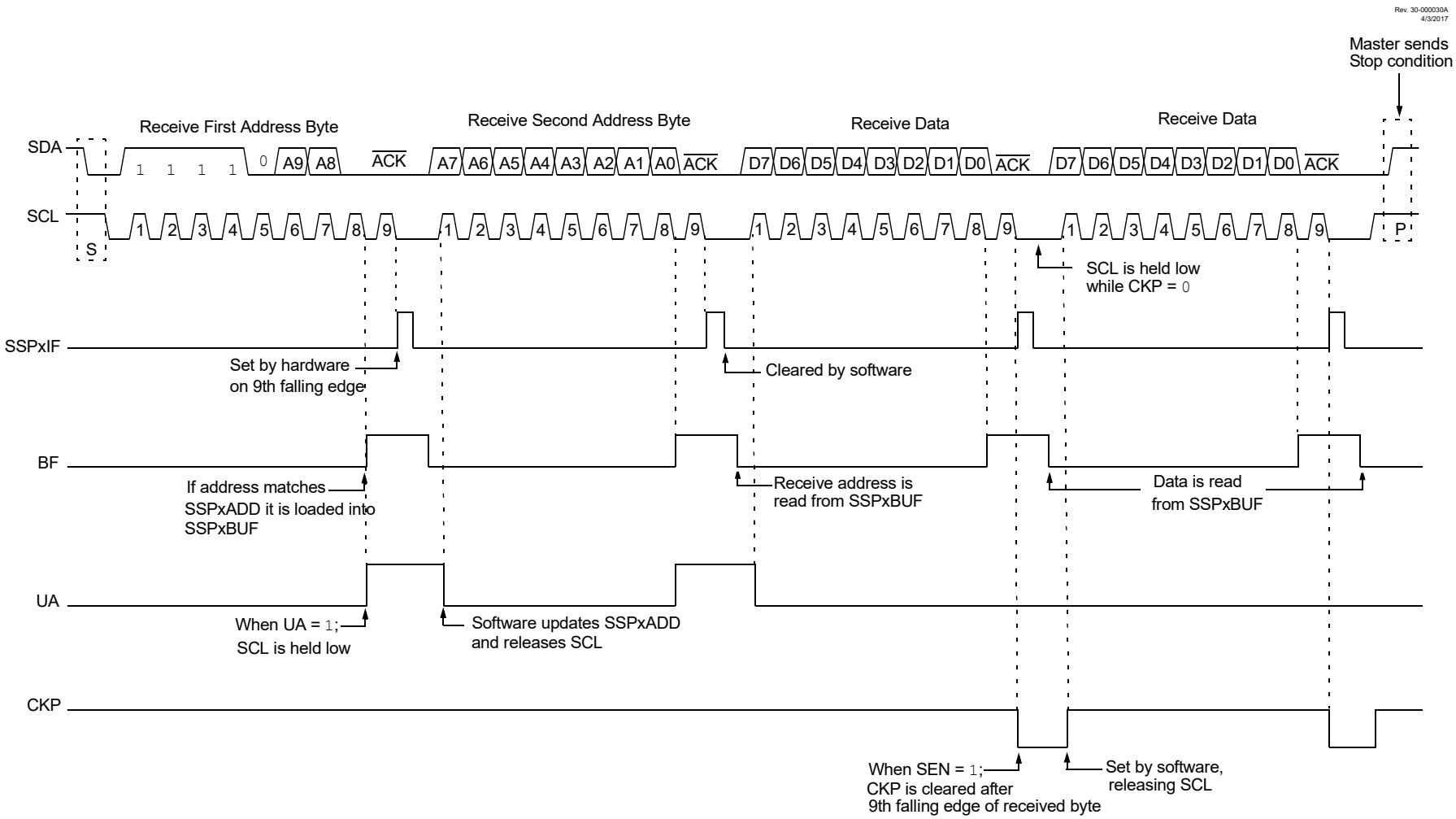
26.5.5 10-bit Addressing with Address or Data Hold

Reception using 10-bit addressing with 26.9.4.7 AHEN or 26.9.4.8 DHEN set is the same as with 7-bit modes. The only difference is the need to update the SSPxADD register using the 26.9.1.7 UA bit. All

functionality, specifically when the 26.9.2.4 CKP bit is cleared and SCL line is held low are the same. Figure 26-21 can be used as a reference of a slave in 10-bit addressing with 26.9.4.7 AHEN set.

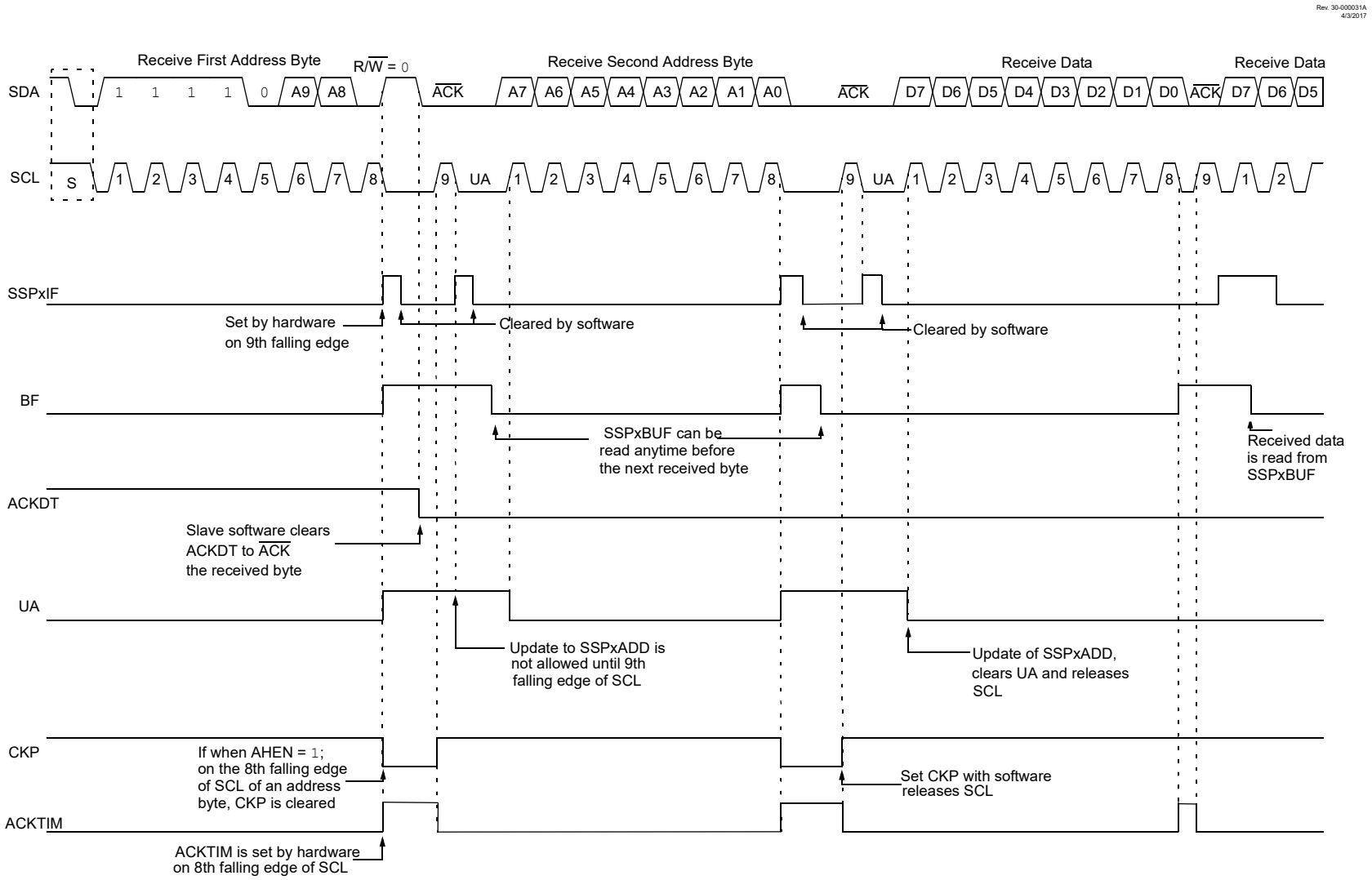
Figure 26-22 shows a standard waveform for a slave transmitter in 10-bit Addressing mode.

Figure 26-20. I²C Slave, 10-bit Address, Reception (SEN = 1, AHEN = 0, DHEN = 0)



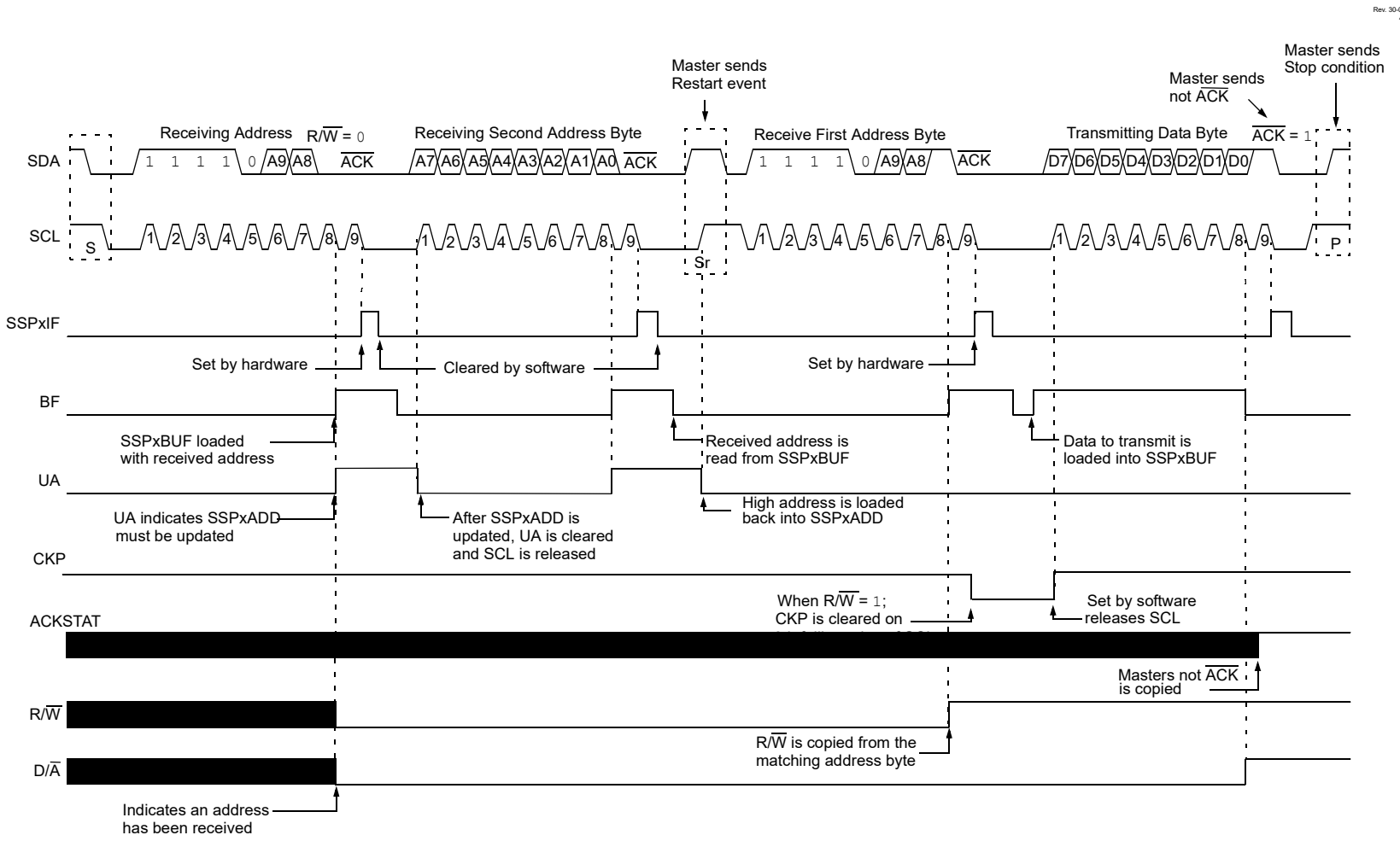
© 2018 Microchip Technology Inc.

Figure 26-21. I²C Slave, 10-bit Address, Reception (SEN = 0, AHEN = 1, DHEN = 0)



© 2018 Microchip Technology Inc.

Figure 26-22. I²C Slave, 10-bit Address, Transmission (SEN = 0, AHEN = 0, DHEN = 0)



© 2018 Microchip Technology Inc.

26.5.6 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 26.9.2.4 CKP bit 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.

26.5.6.1 Normal Clock Stretching

Following an $\overline{ACK}$ if the 26.9.1.6 R/W bit 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 26.9.3.8 SEN bit is set, the slave hardware will always stretch the clock after the $\overline{ACK}$ sequence. Once the slave is ready; 26.9.2.4 CKP is set by software and communication resumes.

Important:

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.

26.5.6.2 10-bit Addressing Mode

In 10-bit Addressing mode, when the 26.9.1.7 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.

Important: Previous versions of the module did not stretch the clock if the second address byte did not match.

26.5.6.3 Byte NACKing

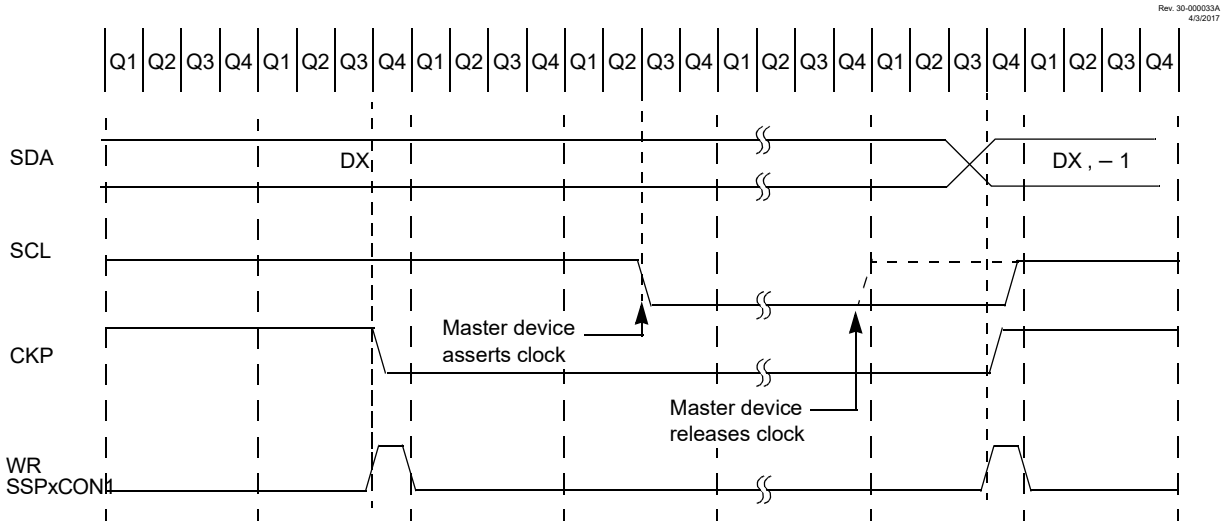
When the 26.9.4.7 AHEN bit is set; CKP is cleared by hardware after the eighth falling edge of SCL for a received matching address byte. When the 26.9.4.8 DHEN bit is set; CKP is cleared after the eighth falling edge of SCL for received data.

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.

26.5.7 Clock Synchronization and the CKP bit

Any time the 26.9.2.4 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 I^2C 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 I^2C bus have released SCL. This ensures that a write to the CKP bit will not violate the minimum high time requirement for SCL (see the following figure).

Figure 26-23. Clock Synchronization Timing

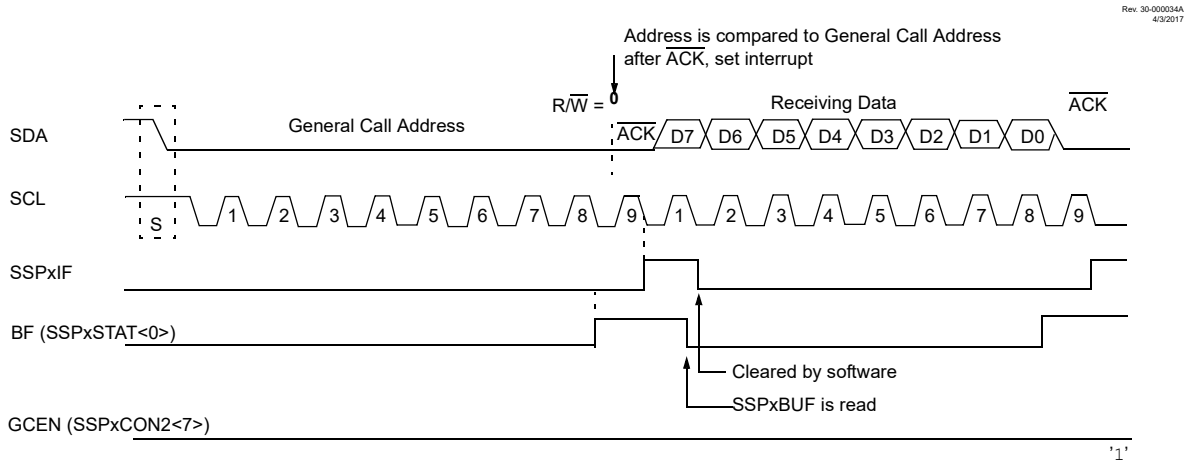


26.5.8 General Call Address Support

The addressing procedure for the I^{2}C 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 I^{2}C protocol, defined as address 0x00. When the 26.9.3.1 GCEN bit is set, the slave module will automatically $\overline{ACK}$ 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/$\overline{W}$ bit clear, an interrupt is generated and slave software can read SSPxBUF and respond. The following figure shows a general call reception sequence.

Figure 26-24. Slave Mode General Call Address 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 26.9.4.7 AHEN bit 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 26.9.3.4 ACKEN value and release the clock with communication progressing as it would normally.

26.5.9 SSP Mask Register

An SSP Mask register (SSPxMSK) 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.

26.6 I²C Master Mode

Master mode is enabled by setting and clearing the appropriate 26.9.2.5 SSPM bits and setting the 26.9.2.3 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 (26.9.1.4 P) and Start (26.9.1.5 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

Important:

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 26.9.2.1 WCOL bit will be set, indicating that a write to the SSPxBUF did not occur.
2. Master mode suspends Start/Stop detection when sending the Start/Stop condition by means of the SEN/PEN control bits. The SSPxIF bit is set at the end of the Start/Stop generation when hardware clears the control bit.

26.6.1 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 26.9.1.6 R/W bit. In this case, the 26.9.1.6 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.

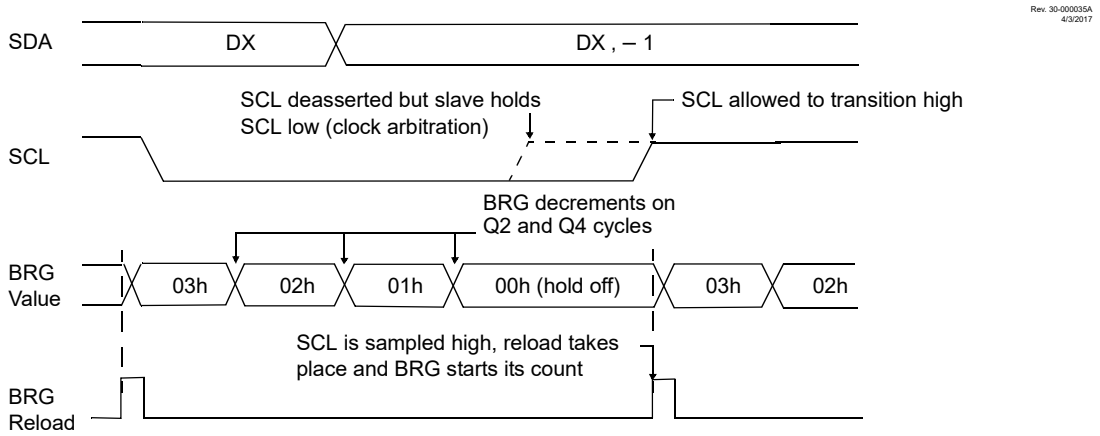
In Master Receive mode, the first byte transmitted contains the slave address of the transmitting device (7 bits) and the R/$\overline{W}$ bit. In this case, the R/$\overline{W}$ bit will be logic '1'. Thus, the first byte transmitted is a 7-bit slave address followed by a '1' to indicate the receive 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 26.7 Baud Rate Generator for more detail.

26.6.2 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 26.9.6 SSPxADD 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 as shown in the following figure.

Figure 26-25. Baud Rate Generator Timing with Clock Arbitration



26.6.3 WCOL Status Flag

If the user writes the SSPxBUF when a Start, Restart, Stop, Receive or Transmit sequence is in progress, the 26.9.2.1 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.

Important: Because queuing of events is not allowed, writing to the lower five bits of SSPxCON2 is disabled until the Start condition is complete.

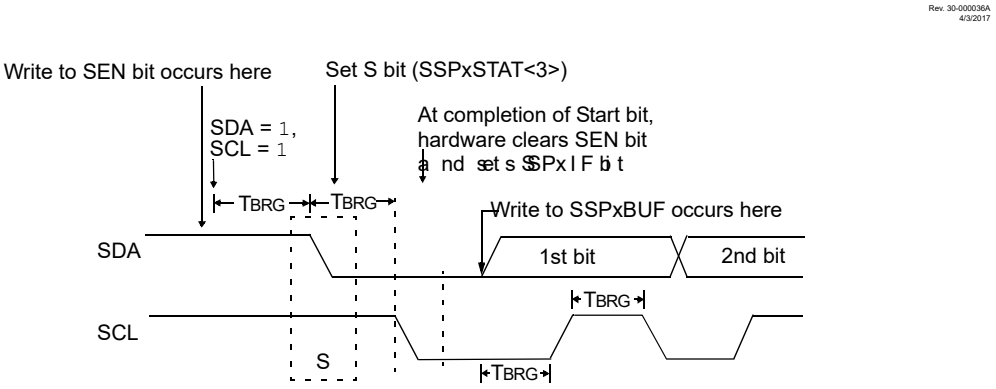
26.6.4 I²C Master Mode Start Condition Timing

To initiate a Start condition (Figure 26-26), the user sets the 26.9.3.8 SEN Start Enable bit. If the SDA and SCL pins are sampled high, the Baud Rate Generator is reloaded with the contents of SSPxADD and starts its count. If SCL and SDA are both sampled high when the Baud Rate Generator times out (T_{BRG}), the SDA pin is driven low. The action of the SDA being driven low while SCL is high is the Start condition and causes the 26.9.1.5 S bit to be set. Following this, the Baud Rate Generator is reloaded with the contents of SSPxADD and resumes its count. When the Baud Rate Generator times out (T_{BRG}), the SEN bit will be automatically cleared by hardware; the Baud Rate Generator is suspended, leaving the SDA line held low and the Start condition is complete.

Important:

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 26-26. First Start Bit Timing



26.6.5 I²C Master Mode Repeated Start Condition Timing

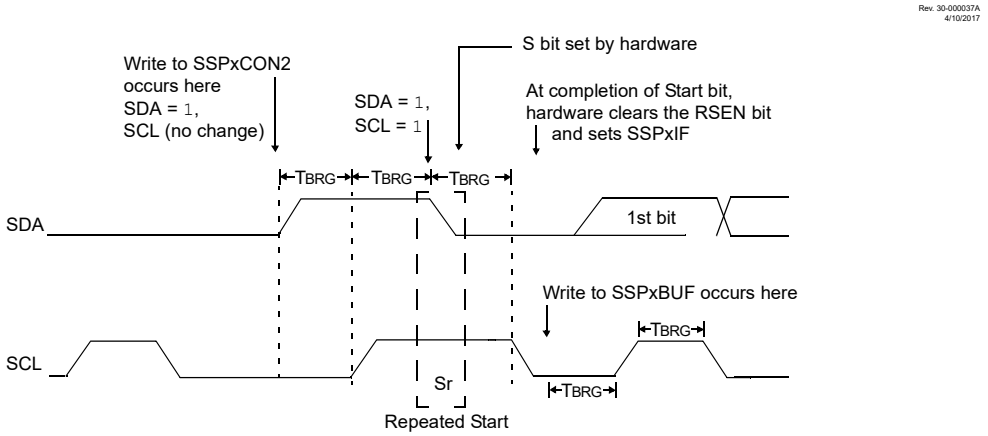
A Repeated Start condition (Figure 26-27) occurs when the 26.9.3.7 RSEN bit is programmed high and the master state machine is no longer active. When the RSEN bit is set, the SCL pin is asserted low. When the SCL pin is sampled low, the Baud Rate Generator is loaded and begins counting. The SDA pin is released (brought high) for one Baud Rate Generator count (T_{BRG}). 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 T_{BRG}. This action is then followed by assertion of the SDA pin (SDA = 0) for one T_{BRG} while SCL is high. SCL is asserted low. Following this, the RSEN bit 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 26.9.1.5 S bit will be set. The SSPxIF bit will not be set until the Baud Rate Generator has timed out.

Important:

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'.

Figure 26-27. Repeated Start Condition Waveform



26.6.6 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 (T_{BRG}). Data should be valid before SCL is released high. When the SCL pin is released high, it is held that way for T_{BRG}. 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 $\overline{ACK}$ bit during the ninth bit time if an address match occurred, or if data was received properly. The status of $\overline{ACK}$ is written into the 26.9.3.2 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 26-28).

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/$\overline{W}$ 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 $\overline{ACK}$ 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.

26.6.6.1 BF Status Flag

In Transmit mode, the 26.9.1.8 BF bit is set when the CPU writes to SSPxBUF and is cleared when all eight bits are shifted out.

26.6.6.2 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 26.9.2.1 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.

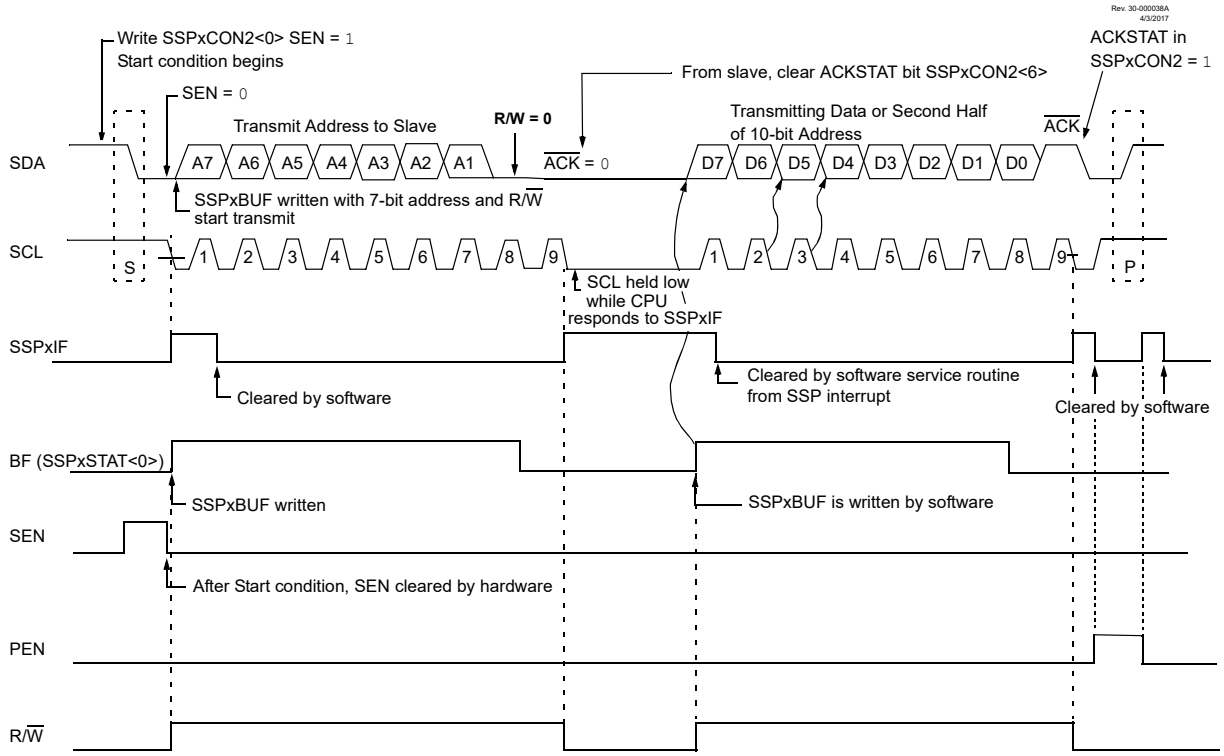
26.6.6.3 ACKSTAT Status Flag

In Transmit mode, the 26.9.3.2 ACKSTAT bit is cleared when the slave has sent an Acknowledge ($\overline{ACK}$ = 0) and is set when the slave does not Acknowledge ($\overline{ACK}$ = 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.

26.6.6.4 Typical transmit sequence:

1. The user generates a Start condition by setting the 26.9.3.8 SEN bit.
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 $\overline{ACK}$ bit from the slave device and writes its value into the 26.9.3.2 ACKSTAT bit.
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 $\overline{ACK}$ bit from the slave device and writes its value into the 26.9.3.2 ACKSTAT bit.
12. Steps 8-11 are repeated for all transmitted data bytes.
13. The user generates a Stop or Restart condition by setting the 26.9.3.6 PEN or 26.9.3.7 RSEN bits. Interrupt is generated once the Stop/Restart condition is complete.

Figure 26-28. I²C Master Mode Waveform (Transmission, 7 or 10-bit Address)



26.6.7 I²C Master Mode Reception

Master mode reception (Figure 26-29) is enabled by programming the 26.9.3.5 RCEN Receive Enable bit.

Important: 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 all the following events occur:

- 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
- The Baud Rate Generator is suspended from counting
- The SCL pin is held 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, 26.9.3.4 ACKEN bit.

26.6.7.1 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.

26.6.7.2 SSPOV Status Flag

In receive operation, the 26.9.2.2 SSPOV bit is set when eight bits are received into the SSPSR while the BF flag bit is already set from a previous reception.

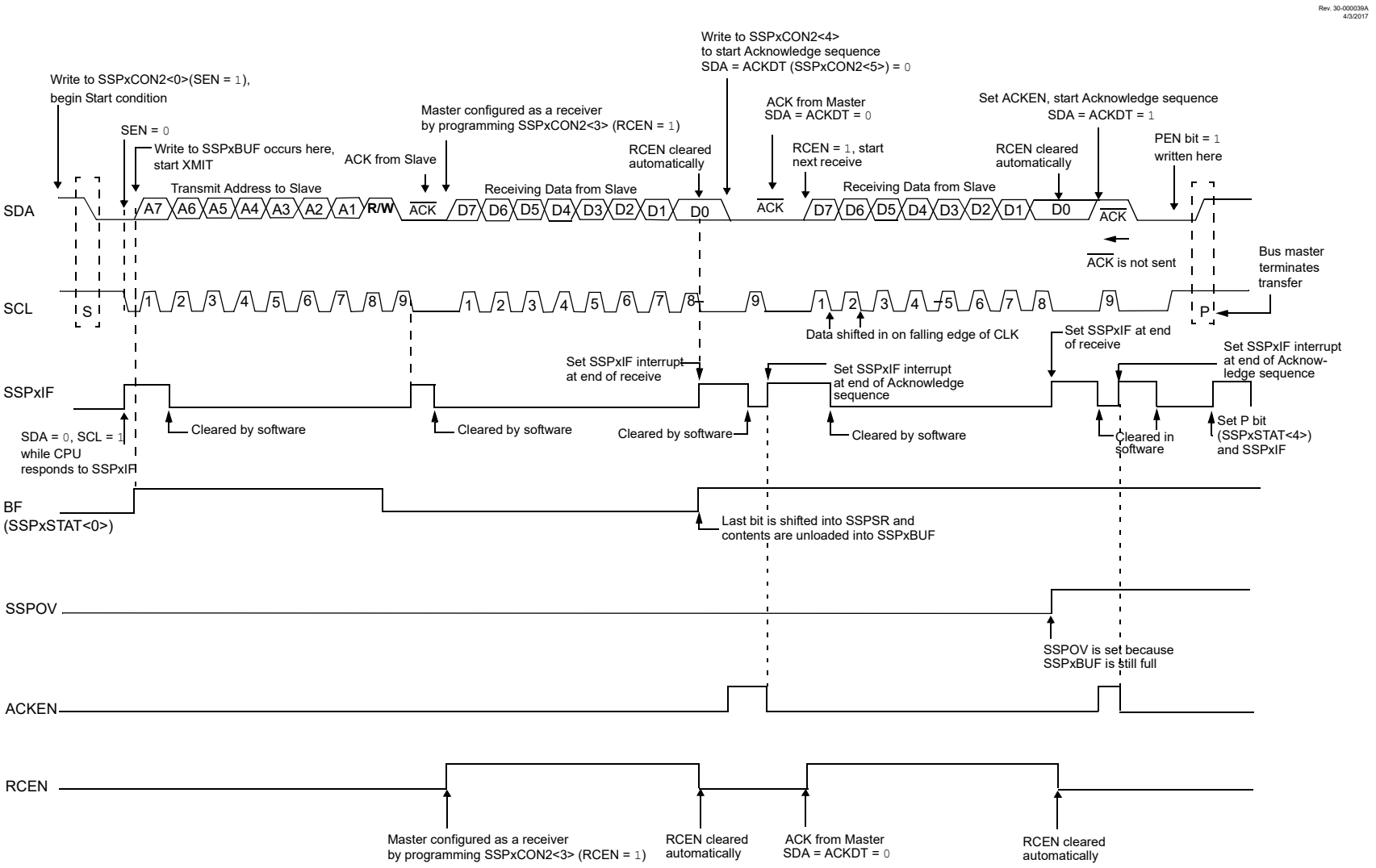
26.6.7.3 WCOL Status Flag

If the user writes the SSPxBUF when a receive is already in progress (i.e., SSPSR is still shifting in a data byte), the 26.9.2.1 WCOL bit is set and the contents of the buffer are unchanged (the write does not occur).

26.6.7.4 Typical Receive Sequence:

1. The user generates a Start condition by setting the 26.9.3.8 SEN bit.
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 26.9.1.6 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 26.9.3.2 ACKSTAT bit.
7. The MSSP module generates an interrupt at the end of the ninth clock cycle by setting the SSPxIF bit.
8. User sets the 26.9.3.5 RCEN bit 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 SSPUF which clears BF.
11. Master sets the $\overline{\text{ACK}}$ value to be sent to slave in the 26.9.3.3 ACKDT bit and initiates the $\overline{\text{ACK}}$ by setting the 26.9.3.4 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.

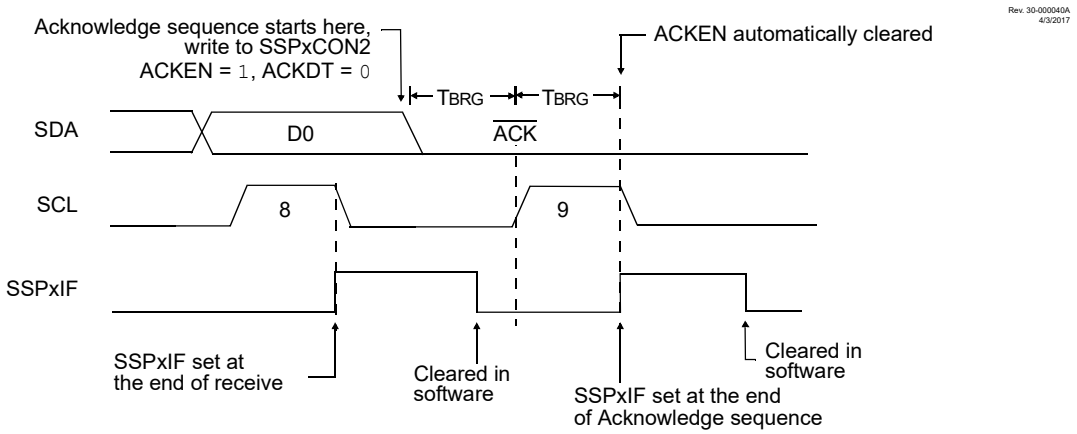
Figure 26-29. I²C Master Mode Waveform (Reception, 7-bit Address)



26.6.8 Acknowledge Sequence Timing

An Acknowledge sequence is enabled by setting the Acknowledge Sequence Enable 26.9.3.4 ACKEN bit. 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 (T_{BRG}) and the SCL pin is deasserted (pulled high). When the SCL pin is sampled high (clock arbitration), the Baud Rate Generator counts for T_{BRG}. 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 26-30. Acknowledge Sequence Waveform



Note: TBRG = one Baud Rate Generator period.

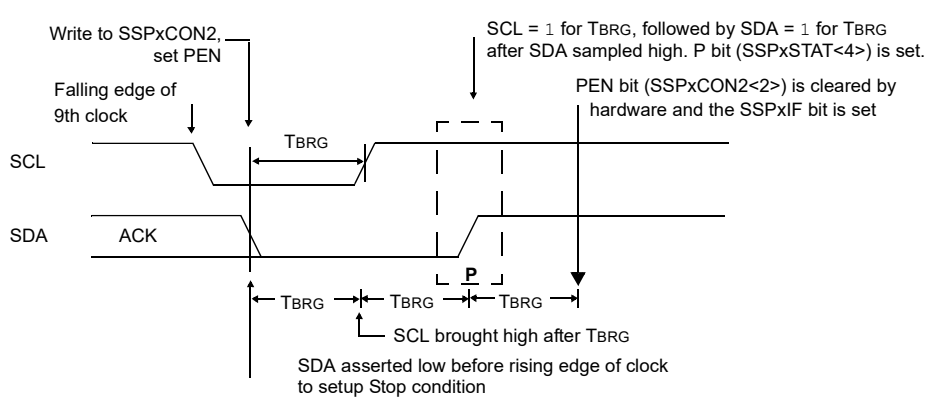
26.6.8.1 Acknowledge Write Collision

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).

26.6.9 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 26.9.3.6 PEN bit. 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 T_{BRG} (Baud Rate Generator rollover count) later, the SDA pin will be deasserted. When the SDA pin is sampled high while SCL is high, the 26.9.1.4 PP bit is set. One T_{BRG} later, the PEN bit is cleared and the SSPxIF bit is set.

Figure 26-31. Stop Condition in Receive or Transmit Mode



Note: TBRG = one Baud Rate Generator period.

26.6.9.1 Write Collision on Stop

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).

26.6.10 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).

26.6.11 Effects of a Reset

A Reset disables the MSSP module and terminates the current transfer.

26.6.12 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 (26.9.1.4 P) and Start (26.9.1.5 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, 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

26.6.13 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 26-32).

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 I^2C bus is free, the user can resume communication by asserting a Start condition.

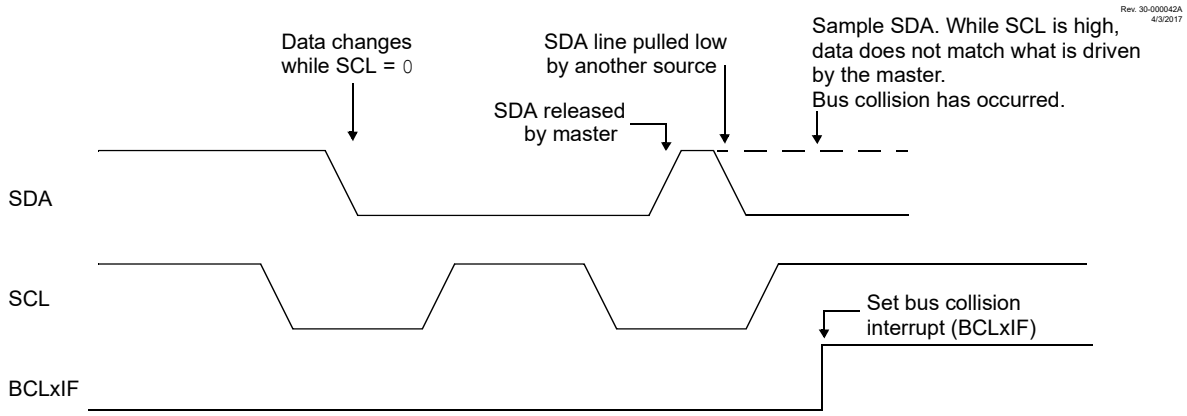
If a Start, Repeated Start, Stop or Acknowledge condition was in progress when the bus collision occurred, the condition is aborted, the SDA and SCL lines are deasserted and the respective control bits in the SSPxCON2 register are cleared. When the user services the bus collision Interrupt Service Routine and if the I^2C 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 I^2C bus can be taken when the 26.9.1.4 P bit is set, or the bus is Idle and the 26.9.1.5 S and P bits are cleared.

Figure 26-32. Bus Collision Timing for Transmit and Acknowledge



26.6.13.1 Bus Collision During a Start Condition

During a Start condition, a bus collision occurs if:

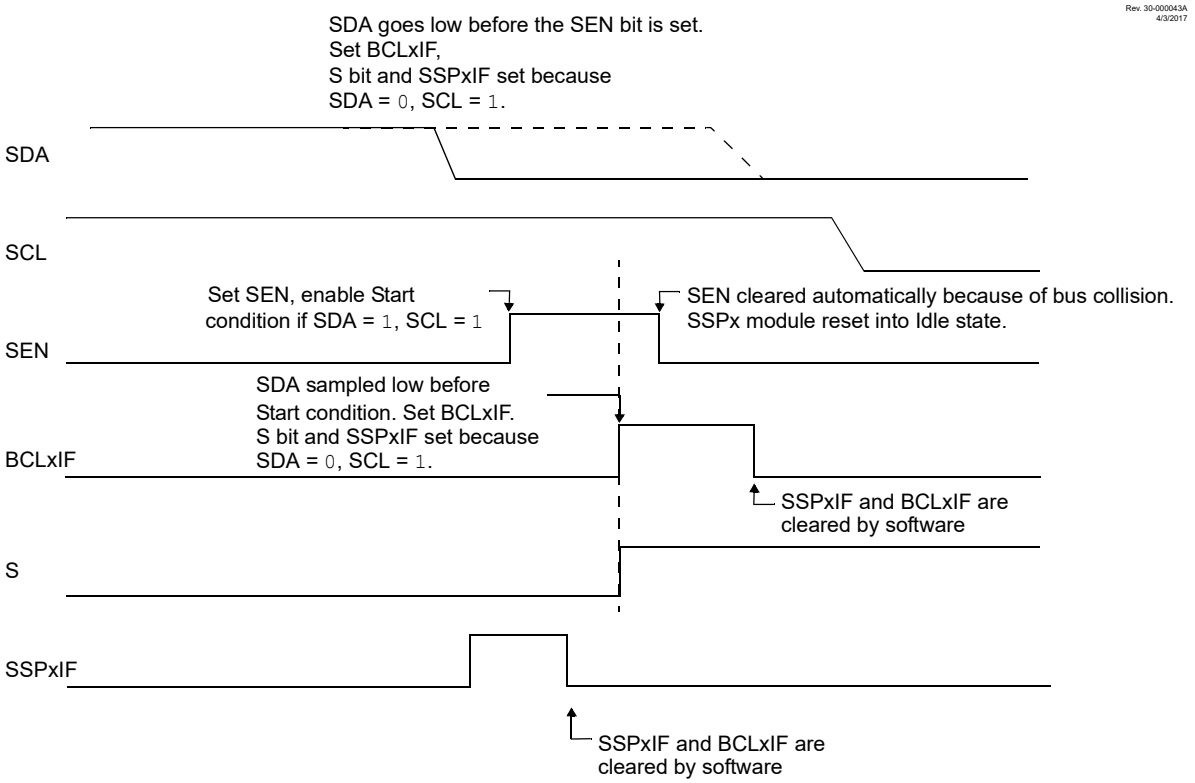
1. SDA or SCL are sampled low at the beginning of the Start condition (Figure 26-33).
2. SCL is sampled low before SDA is asserted low (Figure 26-34).

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:

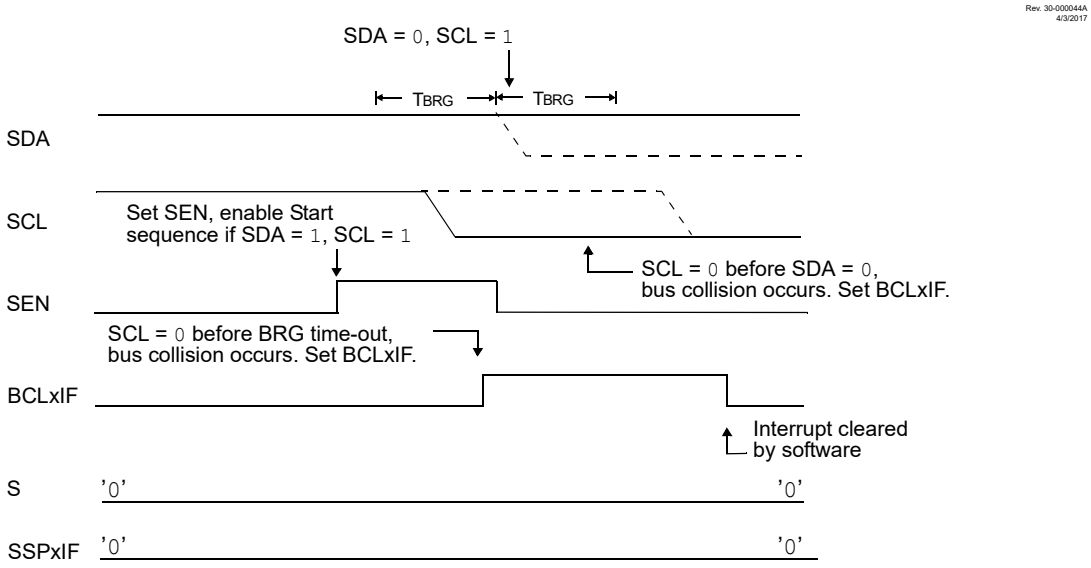
- the Start condition is aborted,
- the BCLxIF flag is set and
- the MSSP module is reset to its Idle state (Figure 26-33).

Figure 26-33. Bus Collision During Start Condition (SDA Only)



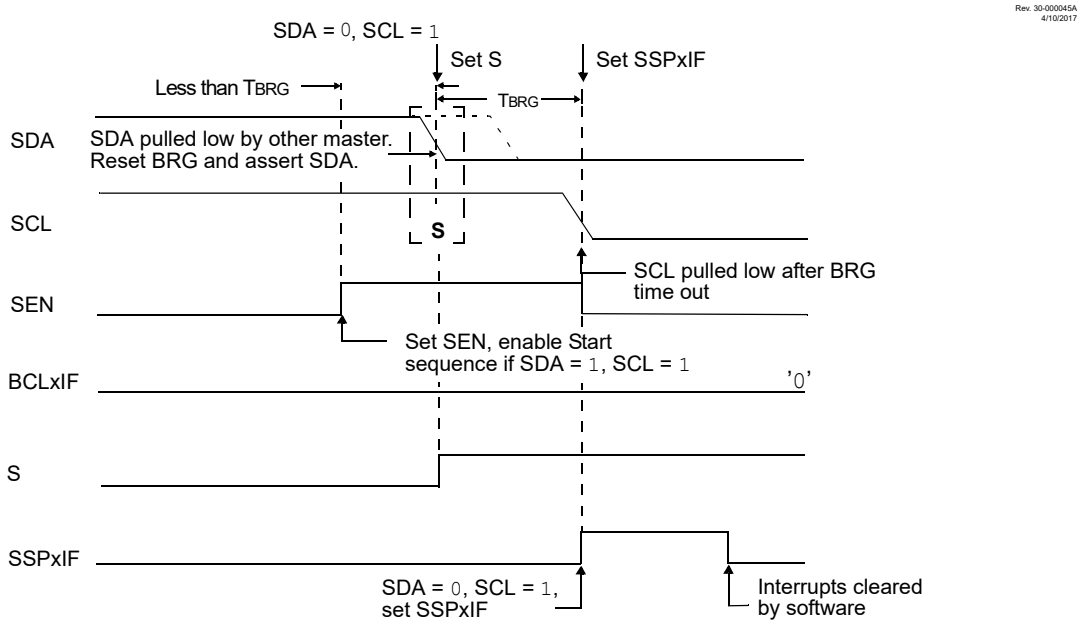
The Start condition begins with the SDA and SCL pins deasserted. When the SDA pin is sampled high, the Baud Rate Generator is loaded and counts down. If the SCL pin is sampled low while SDA is high, a bus collision occurs because it is assumed that another master is attempting to drive a data '1' during the Start condition.

Figure 26-34. Bus Collision During Start Condition (SCL = 0)



If the SDA pin is sampled low during this count, the BRG is reset and the SDA line is asserted early (Figure 26-35). 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.

Figure 26-35. BRG Reset Due to SDA Arbitration During Start Condition



Important: 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.

26.6.13.2 Bus Collision During a Repeated Start Condition

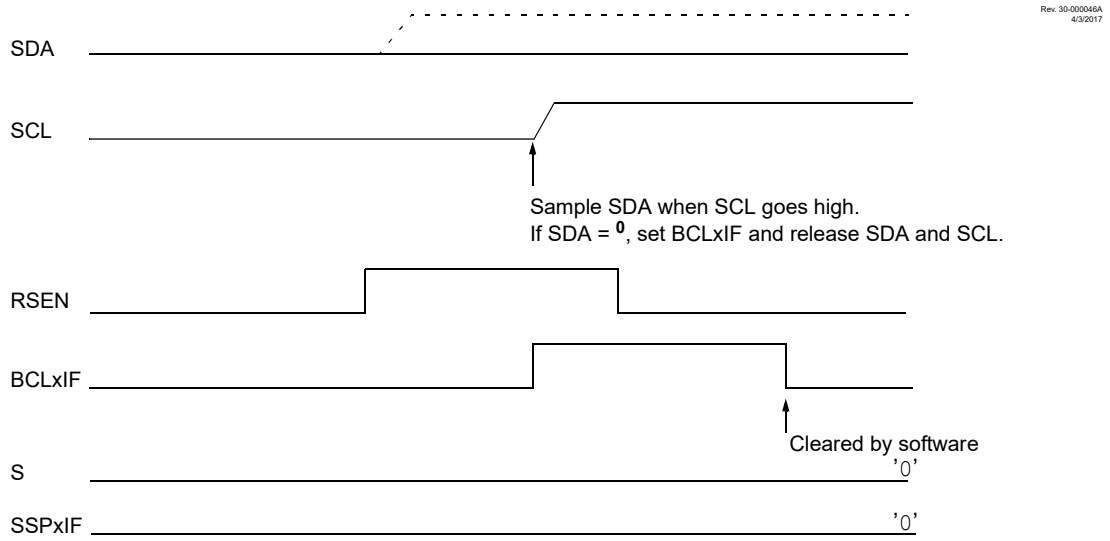
During a Repeated Start condition, a bus collision occurs if:

1. A low level is sampled on SDA when SCL goes from low level to high level (Case 1).
2. 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 26-36). 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.

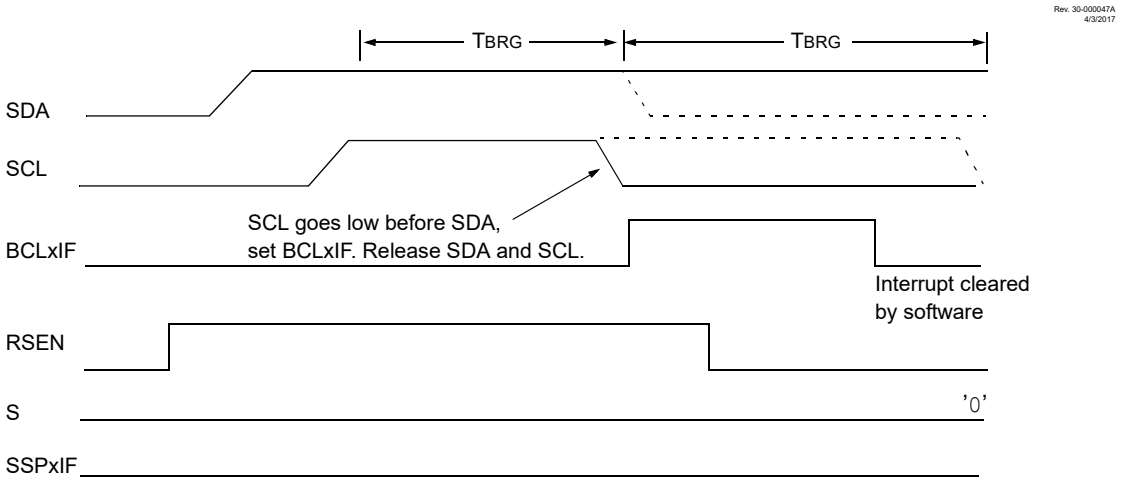
Figure 26-36. Bus Collision During a Repeated Start Condition (Case 1)



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 26-37.

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 26-37. Bus Collision During Repeated Start Condition (Case 2)



26.6.13.3 Bus Collision During a Stop Condition

Bus collision occurs during a Stop condition if:

1. After the SDA pin has been deasserted and allowed to float high, SDA is sampled low after the BRG has timed out (Case 1).
2. 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 26-38). 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 26-39).

Figure 26-38. Bus Collision During a Stop Condition (Case 1)

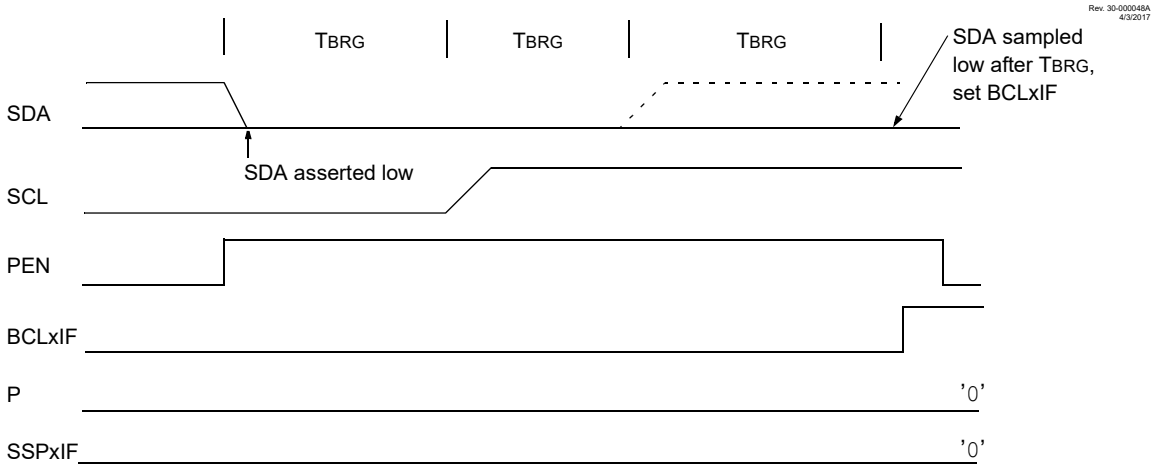
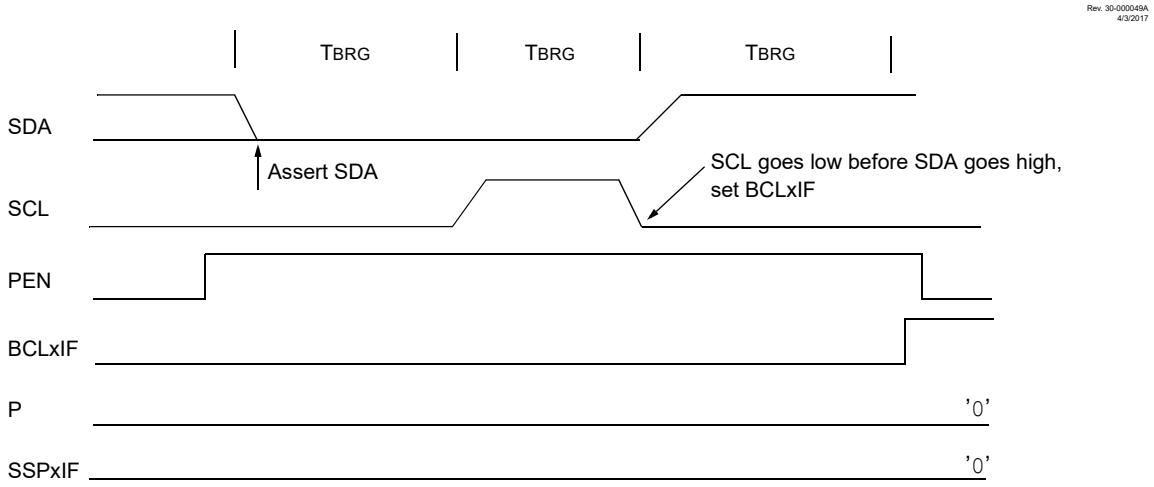


Figure 26-39. Bus Collision During a Stop Condition (Case 2)



26.7 Baud Rate Generator

The MSSP module has a Baud Rate Generator available for clock generation in both I^2C and SPI Master modes. The Baud Rate Generator (BRG) reload value is placed in the SSPxADD register. 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.

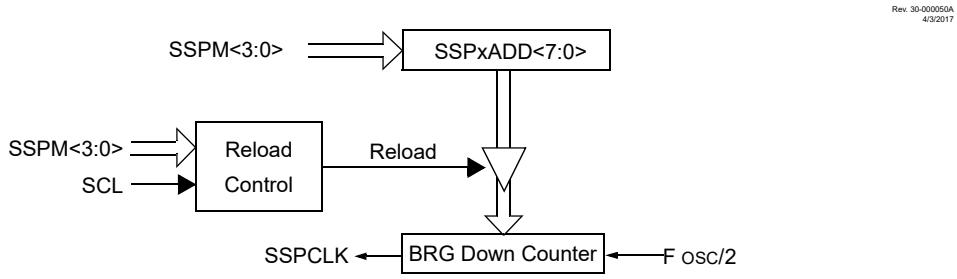
An internal signal "Reload" shown in Figure 26-40 triggers the value from SSPxADD to be loaded into the BRG counter. This occurs twice for each oscillation of the module clock line. The logic dictating when the reload signal is asserted depends on the mode in which the MSSP is being operated.

Table 26-1 illustrates clock rates based on instruction cycles and the BRG value loaded into SSPxADD.

Example 26-1. MSSP Baud Rate Generator Frequency Equation

$$F_{CLOCK} = \frac{F_{OSC}}{4 \times (SSPxADD + 1`)}$$

Figure 26-40. Baud Rate Generator Block Diagram



Important: Values of 0x00, 0x01 and 0x02 are not valid for SSPxADD when used as a Baud Rate Generator for I^2C. This is an implementation limitation.

Table 26-1. MSSP Clock Rate w/BRG

F_{OSC}	F_{CY}	BRG Value	Fclock (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 the "*Electrical Specifications*" section, Internal Oscillator Parameters, to ensure the system is designed to support Iol requirements.

26.8 Register Summary: MSSP Control

Address	Name	Bit Pos.								
0x0F91	SSP1BUF	7:0	BUF[7:0]							
0x0F92	SSP1ADD	7:0	ADD[7:0]							
0x0F93	SSP1MSK	7:0	MSK[6:0]							MSK0
0x0F94	SSP1STAT	7:0	SMP	CKE	$\overline{D/A}$	P	S	$\overline{R/W}$	UA	BF
0x0F95	SSP1CON1	7:0	WCOL	SSPOV	SSPEN	CKP	SSPM[3:0]			
0x0F96	SSP1CON2	7:0	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN
0x0F97	SSP1CON3	7:0	ACKTIM	PCIE	SCIE	BOEN	SDAHT	SBCDE	AHEN	DHEN

26.9 Register Definitions: MSSP Control

26.9.1 SSPxSTAT

Name: SSPxSTAT
Address: 0x0F94

MSSP Status Register

Bit	7	6	5	4	3	2	1	0
	SMP	CKE	$\overline{D/A}$	P	S	$\overline{R/W}$	UA	BF
Access	R/W	R/W	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0

Bit 7 – SMP Slew Rate Control bit

Value	Mode	Description
1	SPI Master	Input data is sampled at the end of data output time
0	SPI Master	Input data is sampled at the middle of data output time
0	SPI Slave	Keep this bit cleared in SPI Slave mode
1	I^2C	Slew rate control is disabled for Standard Speed mode (100 kHz and 1 MHz)
0	I^2C	Slew rate control is enabled for High-Speed mode (400 kHz)

Bit 6 – CKE

SPI: Clock select bit(4) I^2C: SMBus Select bit

Value	Mode	Description
1	SPI	Transmit occurs on the transition from active to Idle clock state
0	SPI	Transmit occurs on the transition from Idle to active clock state
1	I^2C	Enables SMBus-specific inputs
0	I^2C	Disables SMBus-specific inputs

Bit 5 – D/$\overline{A}$
Data/$\overline{Address}$ bit

Value	Mode	Description
x	SPI or I^2C Master	Reserved
1	I^2C Slave	Indicates that the last byte received or transmitted was data
0	I^2C Slave	Indicates that the last byte received or transmitted was address

Bit 4 – P
Stop bit(1)

Value	Mode	Description
x	SPI	Reserved
1	I^2C	Stop bit was detected last
0	I^2C	Stop bit was not detected last

Bit 3 – S
Start bit(1)

Value	Mode	Description
x	SPI	Reserved
1	I²C	Start bit was detected last
0	I²C	Start bit was not detected last

Bit 2 – R/$\overline{W}$
Read/$\overline{Write}$ Information bit(2,3)

Value	Mode	Description
x	SPI	Reserved
1	I²C Slave	Read
0	I²C Slave	Write
1	I²C Master	Transmit is in progress
0	I²C Master	Transmit is not in progress

Bit 1 – UA Update Address bit (10-Bit Slave mode only)

Value	Mode	Description
x	All other modes	Reserved
1	I²C 10-bit Slave	Indicates that the user needs to update the address in the SSPxADD register
0	I²C 10-bit Slave	Address does not need to be updated

Bit 0 – BF
Buffer Full Status bit(5)

Value	Mode	Description
1	I²C Transmit	Character written to SSPxBUF has not been sent
0	I²C Transmit	SSPxBUF is ready for next character
1	SPI and I²C Receive	Received character in SSPxBUF has not been read
0	SPI and I²C Receive	Received character in SSPxBUF has been read

Note:

1. This bit is cleared on Reset and when SSPEN is cleared.
2. In I²C Slave mode this bit holds the R/$\overline{W}$ 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 $\overline{ACK}$ bit.
3. ORing this bit with SEN, RSEN, PEN, RCEN or ACKEN will indicate if the MSSP is in Active mode.
4. Polarity of clock state is set by the CKP bit.
5. I²C receive status does not include $\overline{ACK}$ and Stop bits.

26.9.2 SSPxCON1

Name: SSPxCON1
Address: 0x0F95

MSSP Control Register 1

Bit	7	6	5	4	3	2	1	0
	WCOL	SSPOV	SSPEN	CKP	SSPM[3:0]			
Access	R/W/HS	R/W/HS	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 7 – WCOL
Write Collision Detect bit

Value	Mode	Description
1	SPI	A write to the SSPxBUF register was attempted while the previous byte was still transmitting (must be cleared by software)
1	I^2C Master transmit	A write to the SSPxBUF register was attempted while the I^2C conditions were not valid for a transmission to be started (must be cleared by software)
1	I^2C Slave transmit	The SSPxBUF register is written while it is still transmitting the previous word (must be cleared in software)
0	SPI or I^2C Master or Slave transmit	No collision
x	Master or Slave receive	Don't care

Bit 6 – SSPOV
Receive Overflow Indicator bit(1)

Value	Mode	Description
1	SPI Slave	A byte is received while the SSPxBUF register is still holding the previous byte. The user must read SSPxBUF, even if only transmitting data, to avoid setting overflow. (must be cleared in software)
1	I^2C Receive	A byte is received while the SSPxBUF register is still holding the previous byte (must be cleared in software)
0	SPI Slave or I^2C Receive	No overflow
x	SPI Master or I^2C Master transmit	Don't care

Bit 5 – SSPEN
Master Synchronous Serial Port Enable bit.(2)

Value	Mode	Description
1	SPI	Enables the serial port. The SCKx, SDOx, SDIx, and $\overline{SSx}$ pin selections must be made with the PPS controls. Each signal must be configured with the corresponding TRIS control to the direction appropriate for the mode selected.
1	I^2C	Enables the serial port. The SDAx and SCLx pin selections must be made with the PPS controls. Since both signals are bidirectional the PPS input pin and PPS output pin selections must be made that specify the same pin. Both pins must be configured as inputs with the corresponding TRIS controls.
0	All	Disables serial port and configures these pins as I/O port pins

Bit 4 – CKP
SCK Release Control bit

Value	Mode	Description
1	SPI	Idle state for the clock is a high level
0	SPI	Idle state for the clock is a low level
1	I^2C Slave	Releases clock
0	I^2C Slave	Holds clock low (clock stretch), used to ensure data setup time
x	I^2C Master	Unused in this mode

Bits 3:0 – SSPM[3:0]
Master Synchronous Serial Port Mode Select bits(4)

Value	Description
1111	I^2C Slave mode: 10-bit address with Start and Stop bit interrupts enabled
1110	I^2C Slave mode: 7-bit address with Start and Stop bit interrupts enabled
1101	Reserved - do not use
1100	Reserved - do not use
1011	I^2C Firmware Controlled Master mode (slave Idle)
1010	SPI Master mode: Clock = $F_{OSC}/(4*(SSPxADD+1))$. SSPxADD must be greater than 0.(3)
1001	Reserved - do not use
1000	I^2C Master mode: Clock = $F_{OSC}/(4 * (SSPxADD + 1))$
0111	I^2C Slave mode: 10-bit address
0110	I^2C Slave mode: 7-bit address
0101	SPI Slave mode: Clock = SCKx pin. $\overline{SSx}$ pin control is disabled
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 = Fosc/64
0001	SPI Master mode: Clock = Fosc/16
0000	SPI Master mode: Clock = Fosc/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 I^2C mode only.

26.9.3 SSPxCON2

Name: SSPxCON2
Address: 0x0F96

Control Register for I²C Operation Only

MSSP Control Register 2

Bit	7	6	5	4	3	2	1	0
	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN
Access	R/W	R/W/HC	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 7 – GCEN
General Call Enable bit (Slave mode only)

Value	Mode	Description
x	Master mode	Don't care
1	Slave mode	General call is enabled
0	Slave mode	General call is not enabled

Bit 6 – ACKSTAT Acknowledge Status bit (Master Transmit mode only)

Value	Description
1	Acknowledge was not received from slave
0	Acknowledge was received from slave

Bit 5 – ACKDT
Acknowledge Data bit (Master Receive mode only)**(1)**

Value	Description
1	Not Acknowledge
0	Acknowledge

Bit 4 – ACKEN
Acknowledge Sequence Enable bit**(2)**

Value	Description
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)**

Value	Description
1	Enables Receive mode for I²C
0	Receive is Idle

Bit 2 – PEN
Stop Condition Enable bit (Master mode only)**(2)**

Value	Description
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 (Master mode only)(2)

Value	Description
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 (Master mode only)(2)

Value	Description
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 I^2C 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).

26.9.4 SSPxCON3

Name: SSPxCON3
Address: 0x0F97

MSSP Control Register 3

Bit	7	6	5	4	3	2	1	0
	ACKTIM	PCIE	SCIE	BOEN	SDAHT	SBCDE	AHEN	DHEN
Access	R/HS/HC	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 7 – ACKTIM Acknowledge Time Status bit
Unused in Master mode.

Value	Mode	Description
x	SPI or I²C Master	This bit is not used
1	I²C Slave and AHEN = 1 or DHEN = 1	Eighth falling edge of SCL has occurred and the $\overline{ACK}$/NACK state is active
0	I²C Slave	$\overline{ACK}$/NACK state is not active. Transitions low on ninth rising edge of SCL.

Bit 6 – PCIE
Stop Condition Interrupt Enable bit(1)

Value	Mode	Description
x	SPI or 26.9.2.5 SSPM = 1111 or 0111	Don't care
1	26.9.2.5 SSPM ≠ 1111 and 26.9.2.5 SSPM ≠ 0111	Enable interrupt on detection of Stop condition
0	26.9.2.5 SSPM ≠ 1111 and 26.9.2.5 SSPM ≠ 0111	Stop detection interrupts are disabled

Bit 5 – SCIE Start Condition Interrupt Enable bit

Value	Mode	Description
x	SPI or 26.9.2.5 SSPM = 1111 or 0111	Don't care
1	26.9.2.5 SSPM ≠ 1111 and 26.9.2.5 SSPM ≠ 0111	Enable interrupt on detection of Start condition
0	26.9.2.5 SSPM ≠ 1111 and 26.9.2.5 SSPM ≠ 0111	Start detection interrupts are disabled

Bit 4 – BOEN
Buffer Overwrite Enable bit(2)

Value	Mode	Description
1	SPI	SSPxBUF is updated every time a new data byte is available, ignoring the BF bit
0	SPI	If a new byte is receive with BF set then SSPOV is set and SSPxBUF is not updated
1	I²C	SSPxBUF is updated every time a new data byte is available, ignoring the SSPOV effect on updating the buffer
0	I²C	SSPxBUF is only updated when SSPOV is clear

Bit 3 – SDAHT SDA Hold Time Selection bit

Value	Mode	Description
x	SPI	Not used in SPI mode
1	I^2C	Minimum of 300ns hold time on SDA after the falling edge of SCL
0	I^2C	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.

Value	Mode	Description
x	SPI or I^2C Master	Don't care
1	I^2C Slave	Collision detection is enabled
0	I^2C Slave	Collision detection is not enabled

Bit 1 – AHEN Address Hold Enable bit

Value	Mode	Description
x	SPI or I^2C Master	Don't care
1	I^2C Slave	Address hold is enabled. As a result CKP is cleared after the eighth falling SCL edge of an address byte reception. Software must set the CKP bit to resume operation.
0	I^2C Slave	Address hold is not enabled

Bit 0 – DHEN Data Hold Enable bit

Value	Mode	Description
x	SPI or I^2C Master	Don't care
1	I^2C Slave	Data hold is enabled. As a result CKP is cleared after the eighth falling SCL edge of a data byte reception. Software must set the CKP bit to resume operation.
0	I^2C Slave	Data hold is not enabled

Note:

1. This bit has no effect in Slave modes that Start and Stop condition detection is explicitly listed as enabled.
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.

26.9.5 SSPxBUF

Name: SSPxBUF
Address: 0x0F91

MSSP Data Buffer Register

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

Bits 7:0 – BUF[7:0] MSSP Input and Output Data Buffer bits

26.9.6 SSPxADD

Name: SSPxADD
Address: 0x0F92

MSSP Baud Rate Divider and Address Register

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

Bits 7:0 – ADD[7:0]

- SPI and I^2C Master: Baud rate divider
- I^2C Slave: Address bits

Value	Mode	Description
3 to 255	SPI and I^2C Master	Baud rate divider. SCK/SCL pin clock period = ((n + 1) *4)/F_{OSC}. Values less than 3 are not valid.
2,4,6,8	I^2C 10-bit Slave MS Address	Bits 7-3 and Bit 0 are not used and are don't care. Bits 2:1 are bits 9:8 of the 10-bit Slave Most Significant Address
n	I^2C 10-bit Slave LS Address	Bits 7:0 of 10-Bit Slave Least Significant Address
2*(1 to 127)	I^2C 7-bit Slave	Bit 0 is not used and is don't care. Bits 7:1 are the 7-bit Slave Address

26.9.7 SSPxMSK

Name: SSPxMSK
Address: 0x0F93

MSSP Address Mask Register

Bit	7	6	5	4	3	2	1	0
	MSK[6:0]							MSK0
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	1	1	1	1	1	1	1	1

Bits 7:1 – MSK[6:0] Mask bits

Value	Mode	Description
1	I^2C Slave	The received address bit n is compared to SSPxADD bit n to detect I^2C address match
0	I^2C Slave	The received address bit n is not used to detect I^2C address match

Bit 0 – MSK0
Mask bit for I^2C 10-bit Slave mode

Value	Mode	Description
1	I^2C 10-bit Slave	The received address bit 0 is compared to SSPxADD bit 0 to detect I^2C address match
0	I^2C 10-bit Slave	The received address bit 0 is not used to detect I^2C address match
x	SPI or I^2C 7-bit	Don't care

27. (EUSART) Enhanced Universal Synchronous Asynchronous Receiver Transmitter

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

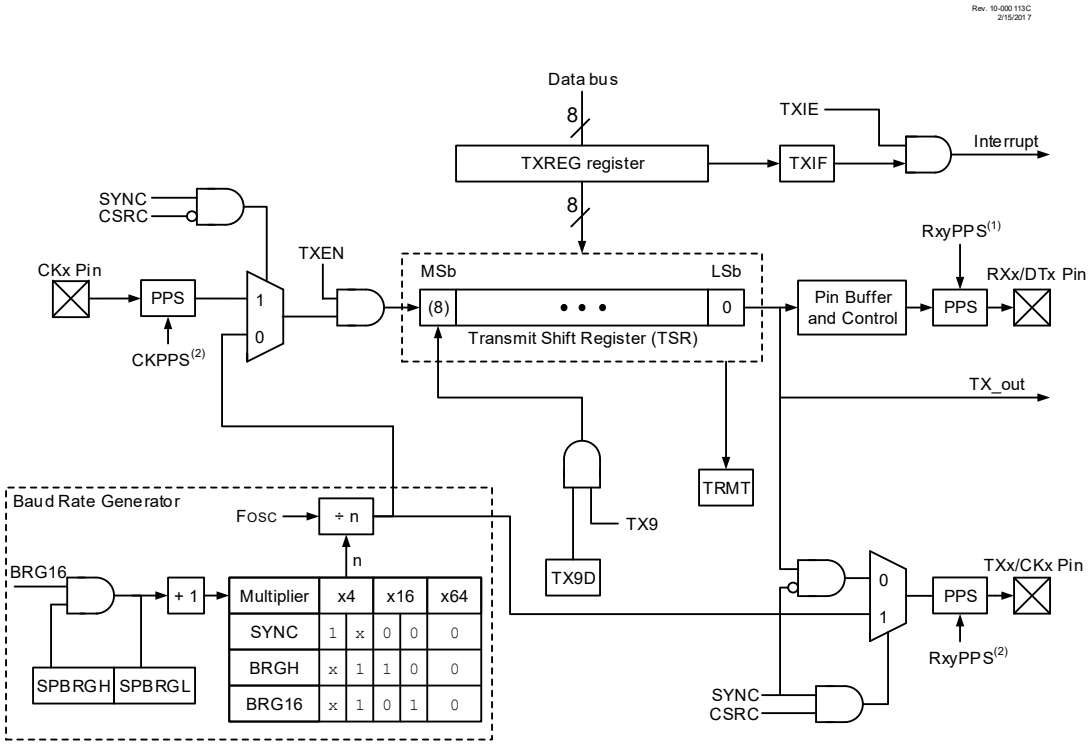
Block diagrams of the EUSART transmitter and receiver are shown in Figure 27-1 and Figure 27-2.

The operation of the EUSART module consists of six registers:

- Transmit Status and Control (27.6.2 TXxSTA)
- Receive Status and Control (27.6.1 RCxSTA)
- Baud Rate Control (27.6.3 BAUDxCON)
- Baud Rate Value (27.6.4 SPxBRG)
- Receive Data Register (27.6.5 RCxREG)
- Transmit Data Register (27.6.6 TXxREG)

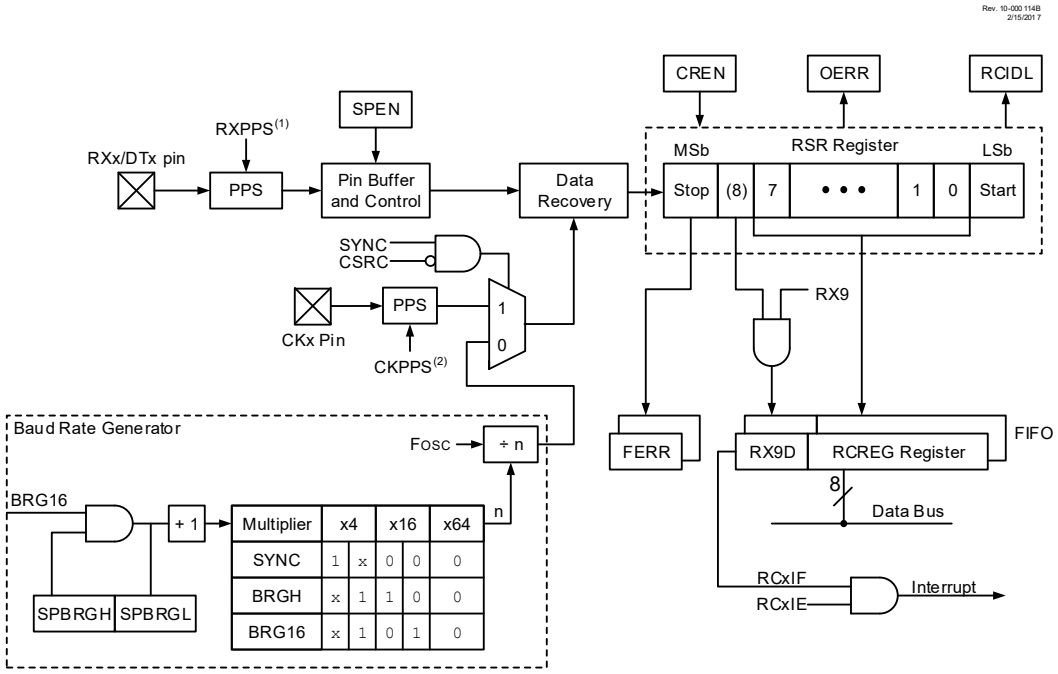
The RXx/DTx and TXx/CKx input pins are selected with the RXxPPS and TXxPPS registers, respectively. TXx, CKx, and DTx output pins are selected with each pin's RxyPPS register. Since the RX input is coupled with the DT output in Synchronous mode, it is the user's responsibility to select the same pin for both of these functions when operating in Synchronous mode. The EUSART control logic will control the data direction drivers automatically.

Figure 27-1. EUSART Transmit Block Diagram



Note 1: In Synchronous mode, the DT output and RX input PPS selections should enable the same pin.
2: In Master Synchronous mode the TX output and CK input PPS selections should enable the same pin.

Figure 27-2. EUSART Receive Block Diagram



Note 1: In Synchronous mode, the DT output and RX input PPS selections should enable the same pin.
2: In Master Synchronous mode the TX output and CK input PPS selections should enable the same pin.

27.1 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 27-2 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.

27.1.1 EUSART Asynchronous Transmitter

The Figure 27-1 is a simplified representation of the transmitter. 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.

27.1.1.1 Enabling the Transmitter

The EUSART transmitter is enabled for asynchronous operations by configuring the following three control bits:

- TXEN = 1 (enables the transmitter circuitry of the EUSART)
- SYNC = 0 (configures the EUSART for asynchronous operation)
- SPEN = 1 (enables the EUSART and automatically enables the output drivers for the RxyPPS selected as the TXx/CKx output)

All other EUSART control bits are assumed to be in their default state.

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.

Important: The TXxIF Transmitter Interrupt flag is set when the TXEN enable bit is set and the TSR is idle.

27.1.1.2 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.

27.1.1.3 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 the 27.3.1.2 Clock Polarity section for more detail.

27.1.1.4 Transmit Interrupt Flag

The TXxIF interrupt flag bit of the PIRx register 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 PIEx register. 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.

27.1.1.5 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 needs to poll this bit to determine the TSR status.

Important: The TSR register is not mapped in data memory, so it is not available to the user.

27.1.1.6 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 the 27.1.2.7 Address Detection section for more information on the Address mode.

27.1.1.7 Asynchronous Transmission Setup

1. Initialize the SPxBRGH, SPxBRGL register pair and the BRGH and BRG16 bits to achieve the desired baud rate (see 27.2 EUSART Baud Rate Generator (BRG)).
2. Select the transmit output pin by writing the appropriate value to the RxyPPS register.
3. Enable the asynchronous serial port by clearing the SYNC bit and setting the SPEN bit.
4. 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.
5. Set SCKP bit if inverted transmit is desired.
6. Enable the transmission by setting the TXEN control bit. This will cause the TXxIF interrupt bit to be set.
7. If interrupts are desired, set the TXxIE interrupt enable bit of the PIEx register
8. An interrupt will occur immediately provided that the GIE and PEIE bits of the INTCON register are also set.
9. If 9-bit transmission is selected, the ninth bit should be loaded into the TX9D data bit.
10. Load 8-bit data into the TXxREG register. This will start the transmission.

Figure 27-3. Asynchronous Transmission

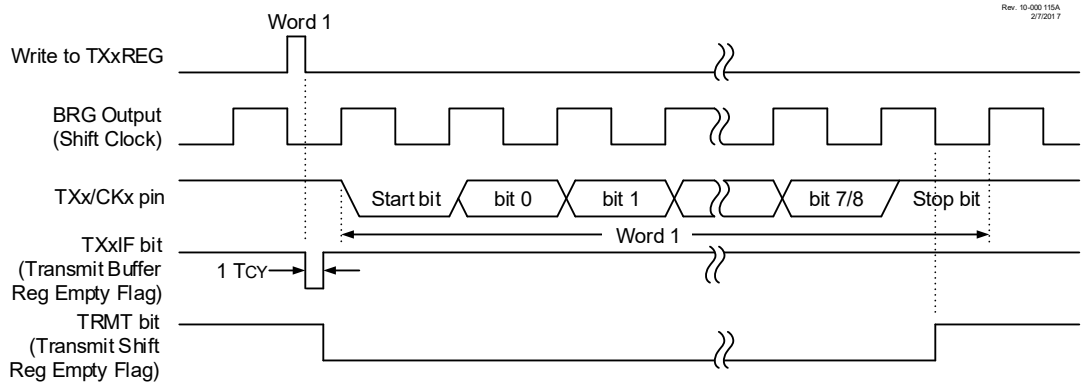
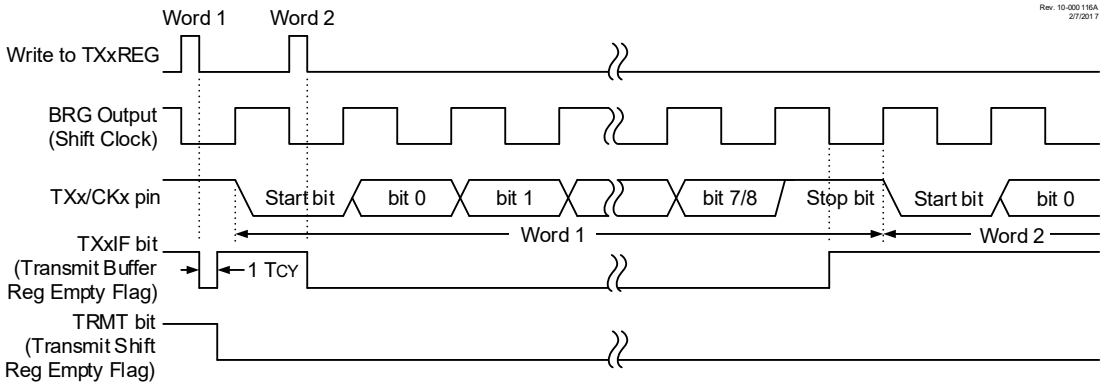


Figure 27-4. Asynchronous Transmission (Back-to-Back)



27.1.2 EUSART Asynchronous Receiver

The Asynchronous mode is typically used in RS-232 systems. A simplified representation of the receiver is shown in the Figure 27-2. 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.

27.1.2.1 Enabling the Receiver

The EUSART receiver is enabled for asynchronous operation by configuring the following three control bits:

- CREN = 1 (enables the receiver circuitry of the EUSART)
- SYNC = 0 (configures the EUSART for asynchronous operation)
- SPEN = 1 (enables the EUSART)

All other EUSART control bits are assumed to be in their default state.

The user must set the RXxPPS register to select the RXx/DTx I/O pin and set the corresponding TRIS bit to configure the pin as an input.

Important: If the RX/DT function is on an analog pin, the corresponding ANSEL bit must be cleared for the receiver to function.

27.1.2.2 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 the 27.1.2.4 Receive Framing Error section 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 PIRx register is set. The top character in the FIFO is transferred out of the FIFO by reading the RCxREG register.

Important: If the receive FIFO is overrun, no additional characters will be received until the overrun condition is cleared. See the 27.1.2.5 Receive Overrun Error section for more information.

27.1.2.3 Receive Interrupts

The RCxIF interrupt flag bit of the PIRx register is 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 PIEx register
- 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.

27.1.2.4 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.

Important: If all receive characters in the receive FIFO have framing errors, repeated reads of the RCxREG will not clear the FERR bit.

27.1.2.5 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.

27.1.2.6 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.

27.1.2.7 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.

27.1.2.8 Asynchronous Reception Setup

1. Initialize the SPxBRGH:SPxBRGL register pair and the BRGH and BRG16 bits to achieve the desired baud rate (see the 27.2 EUSART Baud Rate Generator (BRG) section).
2. Set the RXxPPS register to select the RXx/DTx input pin.
3. Clear the ANSEL bit for the RXx pin (if applicable).
4. Enable the serial port by setting the SPEN bit. The SYNC bit must be clear for asynchronous operation.
5. If interrupts are desired, set the RCxIE bit of the PIEx register and the GIE and PEIE bits of the INTCON register.
6. If 9-bit reception is desired, set the RX9 bit.
7. Enable reception by setting the CREN bit.
8. 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.
9. Read the RCxSTA register to get the error flags and, if 9-bit data reception is enabled, the ninth data bit.
10. Get the received eight Least Significant data bits from the receive buffer by reading the RCxREG register.
11. If an overrun occurred, clear the OERR flag by clearing the CREN receiver enable bit.

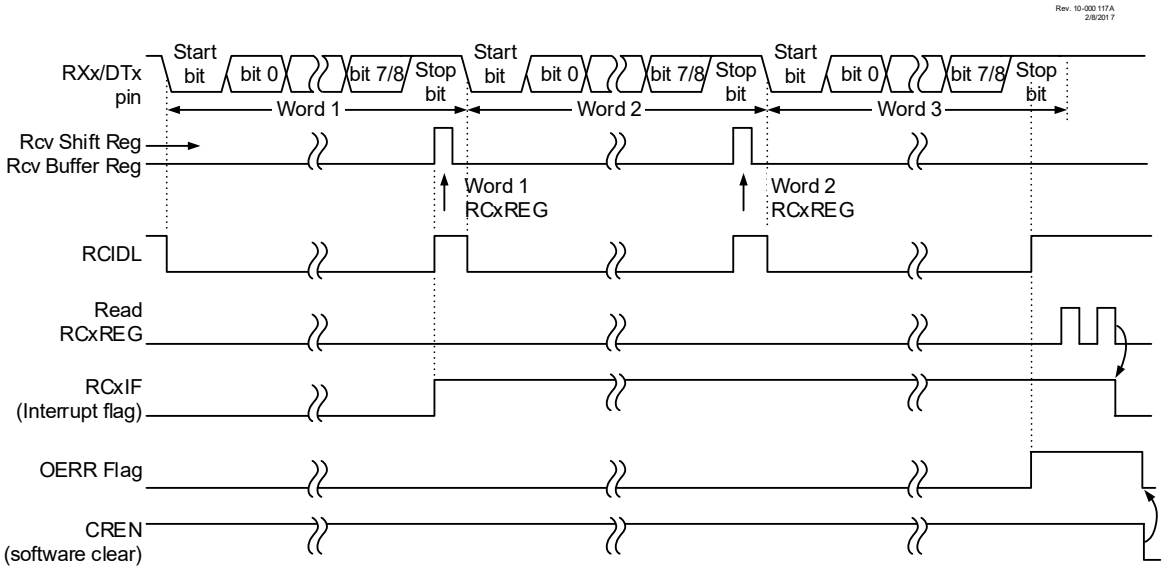
27.1.2.9 9-Bit Address Detection Mode Setup

This mode would typically be used in RS-485 systems. To set up an Asynchronous Reception with Address Detect Enable follow these steps:

1. Initialize the SPxBRGH:SPxBRGL register pair and the BRGH and BRG16 bits to achieve the desired baud rate (see the 27.2 EUSART Baud Rate Generator (BRG) section).
2. Set the RXxPPS register to select the RXx input pin.
3. Clear the ANSEL bit for the RXx pin (if applicable).
4. Enable the serial port by setting the SPEN bit. The SYNC bit must be clear for asynchronous operation.

5. If interrupts are desired, set the RCxIE bit of the PIEx register and the GIE and PEIE bits of the INTCON register.
6. Enable 9-bit reception by setting the RX9 bit.
7. Enable address detection by setting the ADDEN bit.
8. Enable reception by setting the CREN bit.
9. 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 is also set.
10. Read the RCxSTA register to get the error flags. The ninth data bit will always be set.
11. 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.
12. If an overrun occurred, clear the OERR flag by clearing the CREN receiver enable bit.
13. If the device has been addressed, clear the ADDEN bit to allow all received data into the receive buffer and generate interrupts.

Figure 27-5. Asynchronous Reception



Note: This timing diagram shows three bytes appearing on the RXx input. The OERR flag is set because the RCxREG is not read before the third word is received.

27.1.3 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.

The other method adjusts the value in the Baud Rate Generator. This can be done automatically with the Auto-Baud Detect feature (see 27.2.1 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.

27.2 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 27-1 contains the formulas for determining the baud rate. Equation 27-1 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 and are shown in Table 27-2. 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. The BRGH bit is used to achieve very high baud rates.

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.

Equation 27-1. Calculating Baud Rate Error

For a device with Fosc of 16 MHz, desired baud rate of 9600, Asynchronous mode, 8-bit BRG:

$$DesiredBaudrate = \frac{F_{OSC}}{(64 \times SPxBRG) + 1}$$

Solving for SPxBRG:

$$SPxBRG = \frac{F_{OSC}}{64 \times DesiredBaudrate} - 1$$

$$SPxBRG = \frac{16000000}{64 \times 9600} - 1$$

$$SPxBRG = 25.042 \simeq 25$$

$$CalculatedBaudrate = \frac{16000000}{64 \times (25 + 1)}$$

$$CalculatedBaudrate = 9615$$

$$Error = \frac{CalculatedBaudrate - DesiredBaudrate}{DesiredBaudrate}$$

$$Error = \frac{9615 - 9600}{9600}$$

$$Error = 0.16\,\%$$

Table 27-1. 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	

Note: x = Don't care, n = value of SPxBRGH:SPxBRGL register pair.

Table 27-2. Sample Baud Rates for Asynchronous Modes

BAUD RATE	SYNC = 0, BRGH = 0, BRG16 = 0											
	F_{OSC} = 32.000 MHz			F_{OSC} = 20.000 MHz			F_{OSC} = 18.432 MHz			F_{OSC} = 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

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	—	—	—

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	—	—	—

27.2.1 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 27-6. The fifth rising edge will occur on the RXx 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 27-3. 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 SPxBRGH 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:

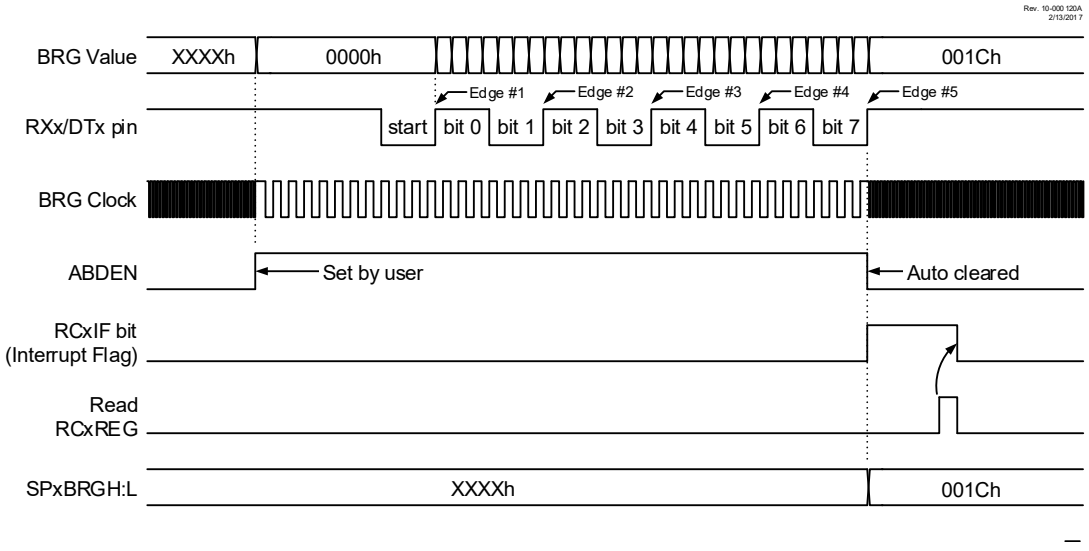
1. If the WUE bit is set with the ABDEN bit, auto-baud detection will occur on the byte following the Break character (see 27.2.3 Auto-Wake-up on Break).
2. It is up to the user to determine that the incoming character baud rate is within the range of the selected BRG clock source. Some combinations 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 27-3. BRG Counter Clock Rates

BRG16	BRGH	BRG Base Clock	BRG ABD Clock
1	1	$F_{OSC}/4$	$F_{OSC}/32$
1	0	$F_{OSC}/16$	$F_{OSC}/128$
0	1	$F_{OSC}/16$	$F_{OSC}/128$
0	0	$F_{OSC}/64$	$F_{OSC}/512$

Note: During the ABD sequence, SPxBRGL and SPxBRGH registers are both used as a 16-bit counter, independent of the BRG16 setting.

Figure 27-6. Automatic Baud Rate Calibration



27.2.2 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.

27.2.3 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 as shown in Figure 27-7, and asynchronously if the device is in Sleep mode as shown in Figure 27-8. 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.

27.2.3.1 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

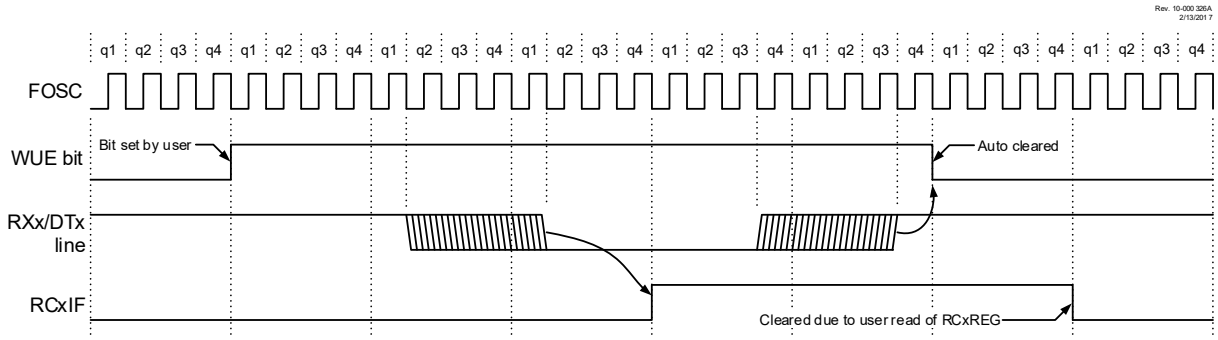
Oscillator start-up time must be considered, especially in applications using oscillators with longer start-up intervals (i.e., LP, XT or HS/PLL mode). The Sync Break (or wake-up signal) character must be of sufficient length, and be followed by a sufficient interval, to allow enough time for the selected 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.

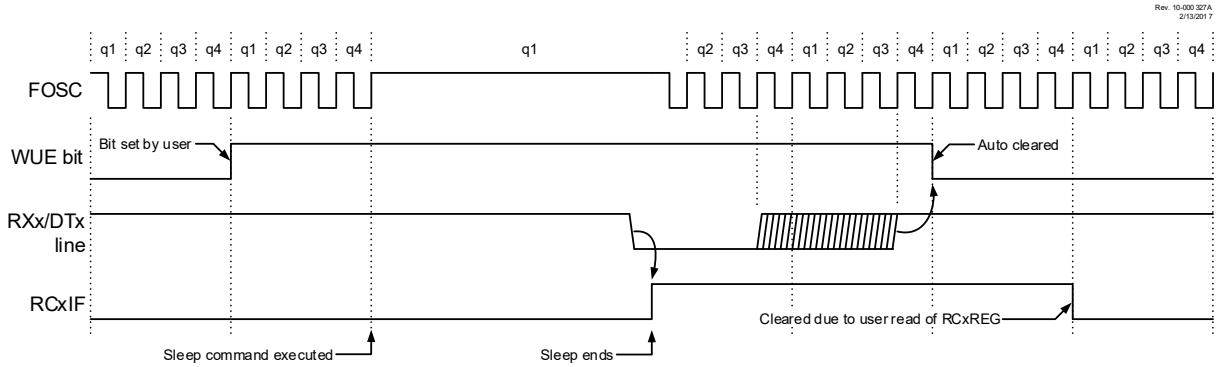
To ensure that no actual data is lost, check the RCIDL bit to verify that a receive operation is not in process before setting the WUE bit. If a receive operation is not occurring, the WUE bit may then be set just prior to entering the Sleep mode.

Figure 27-7. Auto-Wake-up Bit (WUE) Timing During Normal Operation



Note 1: The EUSART remains in idle while the WUE bit is set.

Figure 27-8. Auto-Wake-up Bit (WUE) Timings During Sleep



Note 1: The EUSART remains in idle while the WUE bit is set.

27.2.4 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 27-9 for more detail.

27.2.4.1 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.

27.2.5 Receiving a Break Character

The 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.

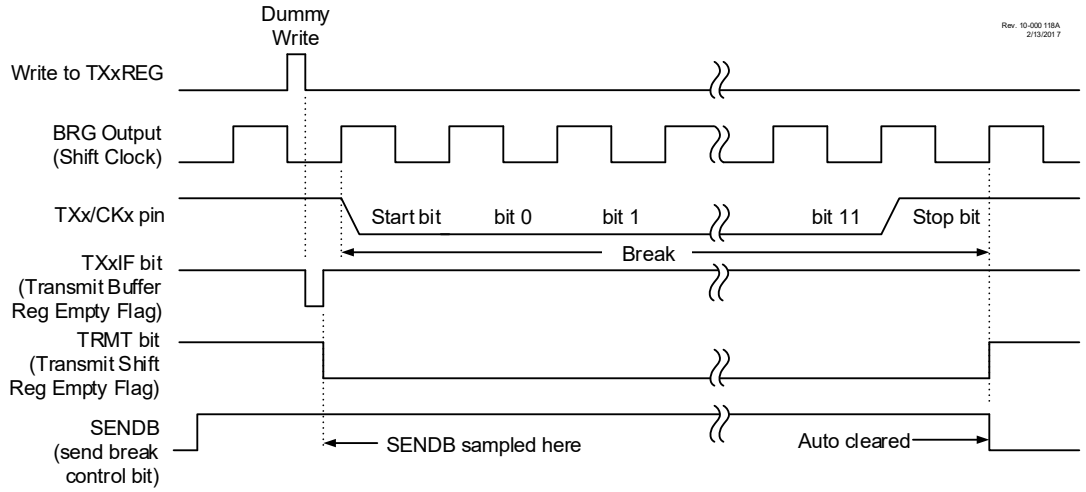
A Break character has been received when all three of the following conditions are true:

- RCxIF bit is set
- FERR bit is set
- RCxREG = 00h

The second method uses the Auto-Wake-up feature described in 27.2.3 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 27-9. Send Break Character Sequence



27.3 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.

27.3.1 Synchronous Master Mode

The following bits are used to configure the EUSART for synchronous master operation:

- SYNC = 1 (configures the EUSART for synchronous operation)
- CSRC = 1 (configures the EUSART as the master)
- SREN = 0 (for transmit); SREN = 1 (recommended setting to receive 1 byte)
- CREN = 0 (for transmit); CREN = 1 (to receive continuously)
- SPEN = 1 (enables the EUSART)



Important: 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.

27.3.1.1 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.

27.3.1.2 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.

27.3.1.3 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.

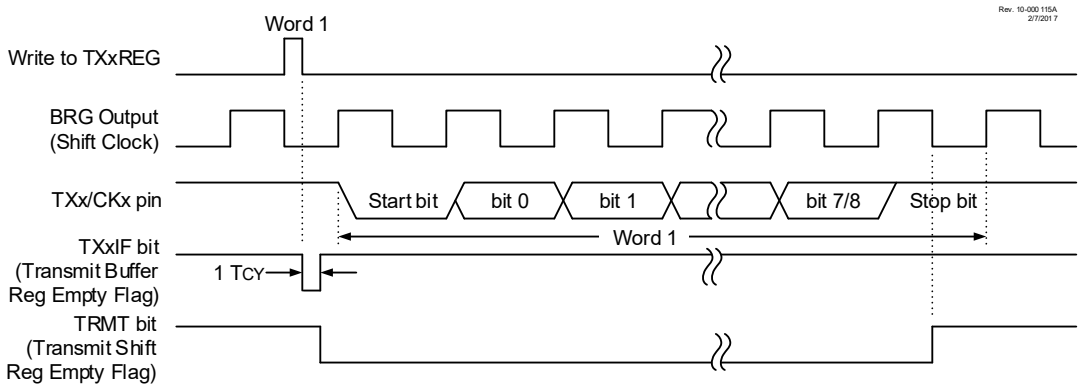
Each data bit changes on the leading edge of the master clock and remains valid until the subsequent leading clock edge.

Note: The TSR register is not mapped in data memory, so it is not available to the user.

27.3.1.4 Synchronous Master Transmission Setup

1. Initialize the SPxBRGH, SPxBRGL register pair and the BRG16 bit to achieve the desired baud rate (see 27.2 EUSART Baud Rate Generator (BRG)).
2. Select the transmit output pin by writing the appropriate values to the RxyPPS register and RXxPPS register. Both selections should enable the same pin.
3. Select the clock output pin by writing the appropriate values to the RxyPPS register and CKxPPS register. Both selections should enable the same pin.
4. Enable the synchronous master serial port by setting bits SYNC, SPEN and CSRC.
5. Disable Receive mode by clearing bits SREN and CREN.
6. Enable Transmit mode by setting the TXEN bit.
7. If 9-bit transmission is desired, set the TX9 bit.
8. If interrupts are desired, set the TXxIE bit of the PIEx register and the GIE and PEIE bits of the INTCON register.
9. If 9-bit transmission is selected, the ninth bit should be loaded in the TX9D bit.
10. Start transmission by loading data to the TXxREG register.

Figure 27-10. Synchronous Transmission



27.3.1.5 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.

27.3.1.6 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.

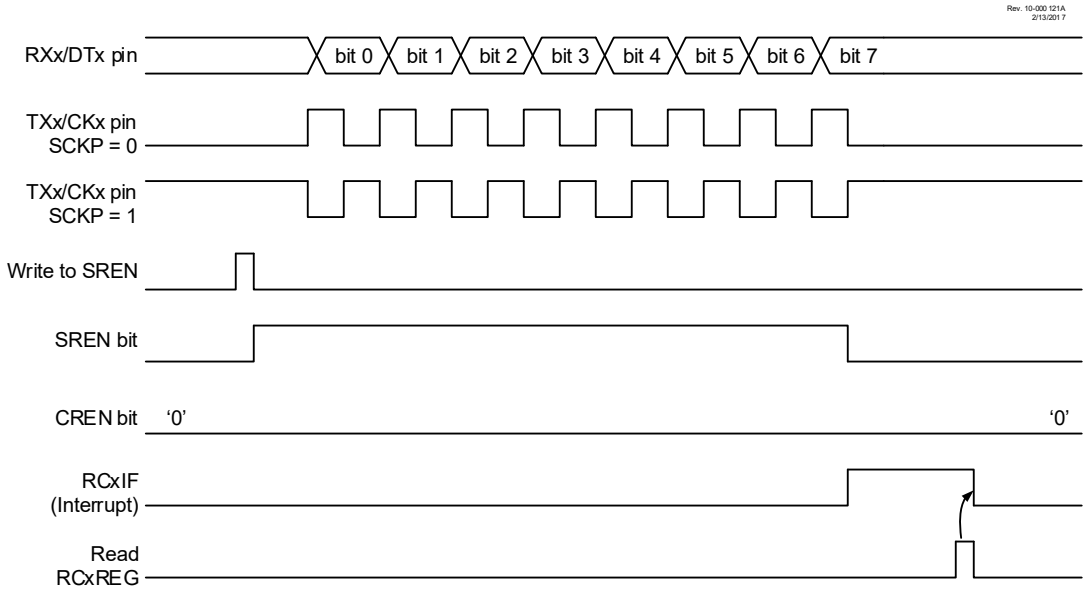
27.3.1.7 Receiving 9-Bit Characters

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.

27.3.1.8 Synchronous Master Reception Setup

1. Initialize the SPxBRGH:SPxBRGL register pair and set or clear the BRG16 bit, as required, to achieve the desired baud rate.
2. Select the receive input pin by writing the appropriate values to the RxyPPS register and RXxPPS register. Both selections should enable the same pin.
3. Select the clock output pin by writing the appropriate values to the RxyPPS register and CKxPPS register. Both selections should enable the same pin.
4. Clear the ANSEL bit for the RXx pin (if applicable).
5. Enable the synchronous master serial port by setting bits SYNC, SPEN and CSRC.
6. Ensure bits CREN and SREN are clear.
7. If interrupts are desired, set the RCxIE bit of the PIEx register and the GIE and PEIE bits of the INTCON register.
8. If 9-bit reception is desired, set bit RX9.
9. Start reception by setting the SREN bit or for continuous reception, set the CREN bit.
10. 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.
11. Read the RCxSTA register to get the ninth bit (if enabled) and determine if any error occurred during reception.
12. Read the 8-bit received data by reading the RCxREG register.
13. 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 27-11. Synchronous Reception (Master Mode, SREN)



27.3.2 Synchronous Slave Mode

The following bits are used to configure the EUSART for synchronous slave operation:

- SYNC = 1 (configures the EUSART for synchronous operation.)
- CSRC = 0 (configures the EUSART as a slave)
- SREN = 0 (for transmit); SREN = 1 (for single byte receive)
- CREN = 0 (for transmit); CREN = 1 (recommended setting for continuous receive)
- SPEN = 1 (enables the EUSART)

Important: 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.

27.3.2.1 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.

Important: 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.

27.3.2.2 EUSART Synchronous Slave Transmit

The operation of the Synchronous Master and Slave modes are identical (see 27.3.1.3 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.

27.3.2.3 Synchronous Slave Transmission Setup

1. Set the SYNC and SPEN bits and clear the CSRC bit.
2. Select the transmit output pin by writing the appropriate values to the RxyPPS register and RXxPPS register. Both selections should enable the same pin.
3. Select the clock input pin by writing the appropriate value to the CKxPPS register.
4. Clear the ANSEL bit for the CKx pin (if applicable).
5. Clear the CREN and SREN bits.
6. If interrupts are desired, set the TXxIE bit of the PIEx register and the GIE and PEIE bits of the INTCON register.
7. If 9-bit transmission is desired, set the TX9 bit.
8. Enable transmission by setting the TXEN bit.
9. If 9-bit transmission is selected, insert the Most Significant bit into the TX9D bit.
10. Prepare for transmission by writing the Least Significant eight bits to the TXxREG register. The word will be transmitted in response to the Master clocks at the CKx pin.

27.3.2.4 EUSART Synchronous Slave Reception

The operation of the Synchronous Master and Slave modes is identical (see 27.3.1.5 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.

27.3.2.5 Synchronous Slave Reception Setup:

1. Set the SYNC and SPEN bits and clear the CSRC bit.
2. Select the receive input pin by writing the appropriate value to the RXxPPS register.
3. Select the clock input pin by writing the appropriate values to the CKxPPS register.
4. Clear the ANSEL bit for both the TXx/CKx and RXx/DTx pins (if applicable).
5. If interrupts are desired, set the RCxIE bit of the PIEx register and the GIE and PEIE bits of the INTCON register.

6. If 9-bit reception is desired, set the RX9 bit.
7. Set the CREN bit to enable reception.
8. The RCxIF bit will be set when reception is complete. An interrupt will be generated if the RCxIE bit was set.
9. If 9-bit mode is enabled, retrieve the Most Significant bit from the RX9D bit of the RCxSTA register.
10. Retrieve the eight Least Significant bits from the receive FIFO 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.

27.4 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.

27.4.1 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 27.3.2.5 Synchronous Slave Reception Setup:).
- If interrupts are desired, set the RCxIE bit of the PIEx register 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 PIRx register 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.

27.4.2 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 27.3.2.3 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.
- Interrupt enable bits TXxIE of the PIEx register and PEIE of the INTCON register must set.
- If interrupts are desired, set the GIEx bit of the INTCON register.

Upon entering Sleep mode, the device will be ready to accept clocks on the 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. Writing TXxREG 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.

27.5 Register Summary - EUSART

Address	Name	Bit Pos.								
0x0F98	RC1REG	7:0	RCREG[7:0]							
0x0F99	TX1REG	7:0	TXREG[7:0]							
0x0F9A	SP1BRG	7:0	SPBRGL[7:0]							
		15:8	SPBRGH[7:0]							
0x0F9C	RC1STA	7:0	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
0x0F9D	TX1STA	7:0	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D
0x0F9E	BAUD1CON	7:0	ABDOVF	RCIDL		SCKP	BRG16		WUE	ABDEN

27.6 Register Definitions: EUSART Control

27.6.1 RCxSTA

Name: RCxSTA
Address: 0x0F9C

Receive Status and Control Register

Bit	7	6	5	4	3	2	1	0
	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
Access	R/W	R/W	R/W	R/W	R/W	RO	R/HC	R/HC
Reset	0	0	0	0	0	0	0	0

Bit 7 – SPEN Serial Port Enable bit

Value	Description
1	Serial port enabled
0	Serial port disabled (held in Reset)

Bit 6 – RX9 9-Bit Receive Enable bit

Value	Description
1	Selects 9-bit reception
0	Selects 8-bit reception

Bit 5 – SREN Single Receive Enable bit
Controls reception. This bit is cleared by hardware when reception is complete

Value	Condition	Description
1	SYNC = 1 AND CSRC = 1	Start single receive
0	SYNC = 1 AND CSRC = 1	Single receive is complete
X	SYNC = 0 OR CSRC = 0	Don't care

Bit 4 – CREN Continuous Receive Enable bit

Value	Condition	Description
1	SYNC = 1	Enables continuous receive until enable bit CREN is cleared (CREN overrides SREN)
0	SYNC = 1	Disables continuous receive
1	SYNC = 0	Enables receiver
0	SYNC = 0	Disables receiver

Bit 3 – ADDEN Address Detect Enable bit

Value	Condition	Description
1	SYNC = 0 AND RX9 = 1	The receive buffer is loaded and the interrupt occurs only when the ninth received bit is set
0	SYNC = 0 AND RX9 = 1	All bytes are received and interrupt always occurs. Ninth bit can be used as parity bit
X	RX9 = 0 OR SYNC = 1	Don't care

Bit 2 – FERR Framing Error bit

Value	Description
1	Unread byte in 27.6.5 RCxREG has a framing error
0	Unread byte in 27.6.5 RCxREG does not have a framing error

Bit 1 – OERR Overrun Error bit

Value	Description
1	Overrun error (can be cleared by clearing either SPEN or CREN bit)
0	No overrun error

Bit 0 – RX9D Ninth bit of Received Data
This can be address/data bit or a parity bit which is determined by user firmware.

27.6.2 TXxSTA

Name: TXxSTA
Address: 0x0F9D

Transmit Status and Control Register

Bit	7	6	5	4	3	2	1	0
	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D
Access	R/W	R/W	R/W	R/W	R/W	R/W	RO	R/W
Reset	0	0	0	0	0	0	1	0

Bit 7 – CSRC Clock Source Select bit

Value	Condition	Description
1	27.6.2.4 SYNC= 1	Master mode (clock generated internally from BRG)
0	27.6.2.4 SYNC= 1	Slave mode (clock from external source)
X	27.6.2.4 SYNC= 0	Don't care

Bit 6 – TX9 9-bit Transmit Enable bit

Value	Description
1	Selects 9-bit transmission
0	Selects 8-bit transmission

Bit 5 – TXEN Transmit Enable bit
Enables transmitter(1)

Value	Description
1	Transmit enabled
0	Transmit disabled

Bit 4 – SYNC EUSART Mode Select bit

Value	Description
1	Synchronous mode
0	Asynchronous mode

Bit 3 – SENDB Send Break Character bit

Value	Condition	Description
1	27.6.2.4 SYNC= 0	Send Sync Break on next transmission (cleared by hardware upon completion)
0	27.6.2.4 SYNC= 0	Sync Break transmission disabled or completed
X	27.6.2.4 SYNC= 1	Don't care

Bit 2 – BRGH High Baud Rate Select bit

Value	Condition	Description
1	27.6.2.4 SYNC= 0	High speed, if BRG16 = 1, baud rate is baudclk/4; else baudclk/16
0	27.6.2.4 SYNC= 0	Low speed
X	27.6.2.4 SYNC= 1	Don't care

Bit 1 – TRMT Transmit Shift Register (TSR) Status bit

Value	Description
1	TSR is empty
0	TSR is not empty

Bit 0 – TX9D Ninth bit of Transmit Data
Can be address/data bit or a parity bit.

Note:

1. 27.6.1.3 SREN and 27.6.1.4 CREN bits override TXEN in Sync mode.

27.6.3 BAUDxCON

Name: BAUDxCON
Address: 0x0F9E

Baud Rate Control Register

Bit	7	6	5	4	3	2	1	0
	ABDOVF	RCIDL		SCKP	BRG16		WUE	ABDEN
Access	RO	RO		RW	RW		RW	RW
Reset	0	0		0	0		0	0

Bit 7 – ABDOVF Auto-Baud Detect Overflow bit

Value	Condition	Description
1	27.6.2.4 SYNC= 0	Auto-baud timer overflowed
0	27.6.2.4 SYNC= 0	Auto-baud timer did not overflow
X	27.6.2.4 SYNC= 1	Don't care

Bit 6 – RCIDL Receive Idle Flag bit

Value	Condition	Description
1	27.6.2.4 SYNC= 0	Receiver is Idle
0	27.6.2.4 SYNC= 0	Start bit has been received and the receiver is receiving
X	27.6.2.4 SYNC= 1	Don't care

Bit 4 – SCKP Synchronous Clock Polarity Select bit

Value	Condition	Description
1	27.6.2.4 SYNC= 0	Idle state for transmit (TX) is a low level (transmit data inverted)
0	27.6.2.4 SYNC= 0	Idle state for transmit (TX) is a high level (transmit data is non-inverted)
1	27.6.2.4 SYNC= 1	Data is clocked on rising edge of the clock
0	27.6.2.4 SYNC= 1	Data is clocked on falling edge of the clock

Bit 3 – BRG16 16-bit Baud Rate Generator Select bit

Value	Description
1	16-bit Baud Rate Generator is used
0	8-bit Baud Rate Generator is used

Bit 1 – WUE Wake-up Enable bit

Value	Condition	Description
1	27.6.2.4 SYNC= 0	Receiver is waiting for a falling edge. Upon falling edge no character will be received and flag RCxIF will be set. WUE will automatically clear after RCxIF is set.
0	27.6.2.4 SYNC= 0	Receiver is operating normally
X	27.6.2.4 SYNC= 1	Don't care

Bit 0 – ABDEN Auto-Baud Detect Enable bit

Value	Condition	Description
1	27.6.2.4 SYNC= 0	Auto-Baud Detect mode is enabled (clears when auto-baud is complete)
0	27.6.2.4 SYNC= 0	Auto-Baud Detect is complete or mode is disabled
X	27.6.2.4 SYNC= 1	Don't care

27.6.4 SPxBRG

Name: SPxBRG
Address: 0x0F9A

Baud Rate Determination Register

Bit	15	14	13	12	11	10	9	8
	SPBRGH[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

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

Bits 15:8 – SPBRGH[7:0] Baud Rate High Byte Register

Bits 7:0 – SPBRGL[7:0] Baud Rate Low Byte Register

27.6.5 RCxREG

Name: RCxREG
Address: 0x0F98

Receive Data Register

Bit	7	6	5	4	3	2	1	0
	RCREG[7:0]							
Access	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0

Bits 7:0 – RCREG[7:0] Receive data

27.6.6 TXxREG

Name: TXxREG
Address: 0x0F99

Transmit Data Register

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

Bits 7:0 – TXREG[7:0] Transmit Data

28. (FVR) Fixed Voltage Reference

The Fixed Voltage Reference, or FVR, is a stable voltage reference, independent of V_{DD}, with the following selectable output levels:

- 1.024V
- 2.048V
- 4.096V

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.

Important: Fixed Voltage Reference output cannot exceed V_{DD}.

28.1 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 the ADC chapter 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.

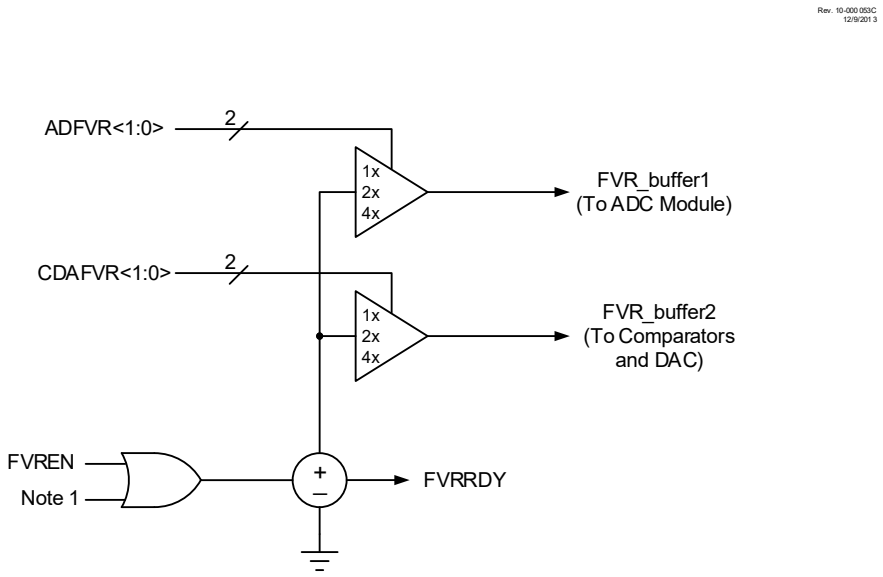
Related Links

31. (ADC2) Analog-to-Digital Converter with Computation Module
32. (CMP) Comparator Module
30. (DAC) 5-Bit Digital-to-Analog Converter Module

28.2 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 28-1. Voltage Reference Block Diagram



28.3 Register Summary - FVR

Address	Name	Bit Pos.								
0x0F2C	FVRCON	7:0	FVREN	FVRRDY	TSEN	TSRNG	CDAFVR[1:0]		ADFVR[1:0]	

28.4 Register Definitions: FVR Control

28.4.1 FVRCON

Name: FVRCON
Address: 0xF2C

Fixed Voltage Reference Control Register

Bit	7	6	5	4	3	2	1	0
	FVREN	FVRRDY	TSEN	TSRNG	CDAFVR[1:0]		ADFVR[1:0]	
Access	R/W	R	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	q	0	0	0	0	0	0

Bit 7 – FVREN Fixed Voltage Reference Enable bit

Value	Description
1	Fixed Voltage Reference is enabled
0	Fixed Voltage Reference is disabled

Bit 6 – FVRRDY Fixed Voltage Reference Ready Flag bit

Value	Description
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(2)

Value	Description
1	Temperature Indicator is enabled
0	Temperature Indicator is disabled

Bit 4 – TSRNG
Temperature Indicator Range Selection bit(2)

Value	Description
1	$V_{OUT} = V_{DD}$ - 4Vt (High Range)
0	$V_{OUT} = V_{DD}$ - 2Vt (Low Range)

Bits 3:2 – CDAFVR[1:0] Comparator FVR Buffer Gain Selection bits

Value	Description
11	Comparator FVR Buffer Gain is 4x, (4.096V)(1)
10	Comparator FVR Buffer Gain is 2x, (2.048V)(1)
01	Comparator FVR Buffer Gain is 1x, (1.024V)
00	Comparator FVR Buffer is off

Bits 1:0 – ADFVR[1:0] ADC FVR Buffer Gain Selection bit

Value	Description
11	ADC FVR Buffer Gain is 4x, (4.096V)(1)
10	ADC FVR Buffer Gain is 2x, (2.048V)(1)

Value	Description
01	ADC FVR Buffer Gain is 1x, (1.024V)
00	ADC FVR Buffer is off

Note:

1. Fixed Voltage Reference output cannot exceed V_{DD}.
2. See *Temperature Indicator Module* section for additional information.

Related Links

29. Temperature Indicator Module

29. 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°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. Refer to Application Note AN1333, *"Use and Calibration of the Internal Temperature Indicator"* (DS00001333) for more details regarding the calibration process.

29.1 Circuit Operation

Figure 29-1 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.

The following equation describes the output characteristics of the temperature indicator.

Equation 29-1. V_{OUT} Ranges

$$High\ Range:\ V_{OUT} = V_{DD} - 4V_T$$

$$Low\ Range:\ V_{OUT} = V_{DD} - 2V_T$$

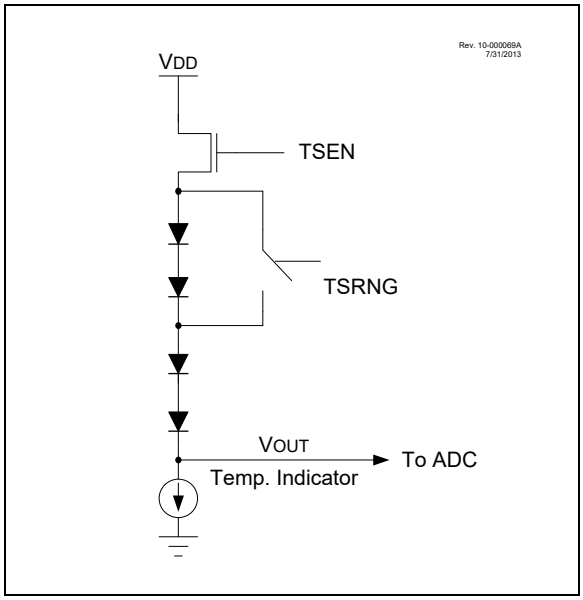
The temperature sense circuit is integrated with the Fixed Voltage Reference (FVR) module. See *"Fixed Voltage Reference (FVR)"* chapter for more information.

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 29-1. Temperature Circuit Diagram



Related Links

28. (FVR) Fixed Voltage Reference

29.2 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 29-1 shows the recommended minimum V_{DD} vs. range setting.

Table 29-1. Recommended V_{DD} vs. Range

Min. V_{DD}, TSRNG = 1	Min. V_{DD}, TSRNG = 0
3.6V	1.8V

29.3 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 *"Analog-to-Digital Converter with Computation (ADC^2) Module"* chapter for detailed information.

Related Links

31. (ADC2) Analog-to-Digital Converter with Computation Module

29.4 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.

30. (DAC) 5-Bit Digital-to-Analog Converter Module

The Digital-to-Analog Converter supplies a variable voltage reference, ratiometric with the input source, with 32 selectable output levels.

The positive input source ($V_{SOURCE}+$) of the DAC can be connected to:

- FVR Buffer
- External $V_{REF}+$ pin
- V_{DD} supply voltage

The negative input source ($V_{SOURCE}-$) of the DAC can be connected to:

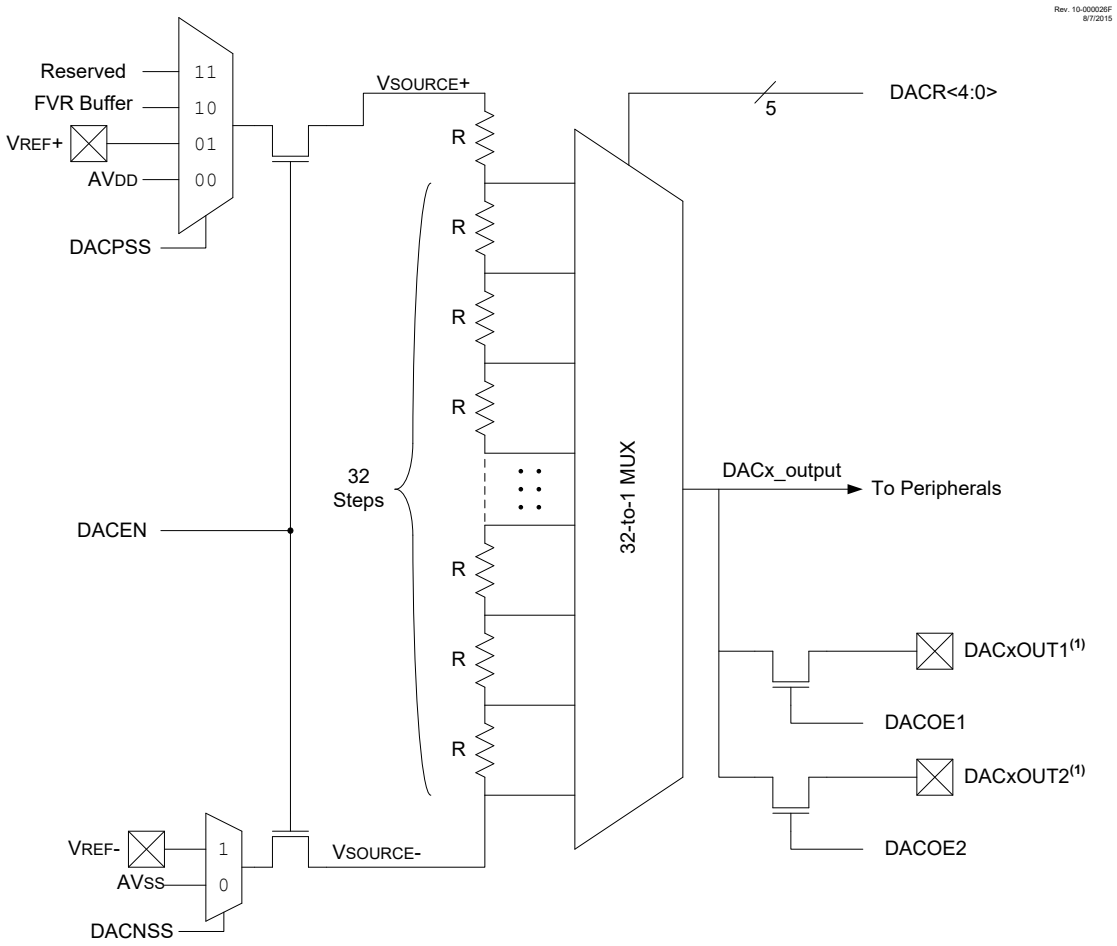
- External $V_{REF}-$ pin
- V_{SS}

The output of the DAC (DACx_output) can be selected as a reference voltage to the following:

- Comparator positive input
- ADC input channel
- DACxOUT1 pin
- DACxOUT2 pin

The Digital-to-Analog Converter (DAC) can be enabled by setting the EN bit.

Figure 30-1. Digital-to-Analog Converter Block Diagram



Note:

1. The unbuffered DACx_output is provided on the DACxOUT pin(s).

30.1 Output Voltage Selection

The DAC has 32 voltage level ranges. The 32 levels are set with the DAC1R bits.

The DAC output voltage can be determined by using the following equation.

Equation 30-1. DAC Output Voltage

When EN = 1:

$$DACx_output = \left(\left(V_{REF+} - V_{REF-}\right) x \frac{DACR[4:0]}{2^5}\right) + V_{REF-}$$

Note: See the DAC1CON0 register for the available $V_{SOURCE}+$ and $V_{SOURCE}-$ selections.

30.2 Ratiometric Output Level

The DAC output value is derived using a resistor ladder with each end of the ladder tied to a positive and negative voltage reference input source. If the voltage of either input source fluctuates, a similar fluctuation will result in the DAC output value.

The value of the individual resistors within the ladder can be found in the *"5-Bit DAC Specifications"* table from the *"Electrical Specifications"* chapter.

Related Links

38.4.10 5-Bit DAC Specifications

30.3 DAC Voltage Reference Output

The unbuffered DAC voltage can be output to the DACxOUTn pin(s) by setting the respective OEn bit(s). 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'.

Important: The unbuffered DAC output (DACxOUTn) is not intended to drive an external load.

30.4 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.

30.5 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 range select bits are cleared.

30.6 Register Summary - DAC Control

Address	Name	Bit Pos.								
0x0F2E	DAC1CON0	7:0	EN		OE1	OE2	PSS[1:0]			NSS
0x0F2F	DAC1CON1	7:0				DAC1R[4:0]				

30.7 Register Definitions: DAC Control

Long bit name prefixes for the DAC are shown in the table below. Refer to the *"Long Bit Names Section"* for more information.

Table 30-1. DAC Long Bit Name Prefixes

Peripheral	Bit Name Prefix
DAC	DAC

Related Links

1.4.2.2 Long Bit Names

30.7.1 DAC1CON0

Name: DAC1CON0
Address: 0xF2E

DAC Control Register

Bit	7	6	5	4	3	2	1	0
	EN		OE1	OE2	PSS[1:0]			NSS
Access	R/W		R/W	R/W	R/W	R/W		R/W
Reset	0		0	0	0	0		0

Bit 7 – EN DAC Enable bit

Value	Description
1	DAC is enabled
0	DAC is disabled

Bit 5 – OE1 DAC Voltage Output Enable bit

Value	Description
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

Value	Description
1	DAC voltage level is output on the DAC1OUT2 pin
0	DAC voltage level is disconnected from the DAC1OUT2 pin

Bits 3:2 – PSS[1:0] DAC Positive Source Select bit

Value	Description
11	Reserved
10	FVR buffer
01	$V_{REF}+$
00	AV_{DD}

Bit 0 – NSS DAC Negative Source Select bit

Value	Description
1	$V_{REF}-$
0	AV_{SS}

30.7.2 DAC1CON1

Name: DAC1CON1
Address: 0xF2F

DAC Data Register

Bit	7	6	5	4	3	2	1	0
				DAC1R[4:0]				
Access				R/W	R/W	R/W	R/W	R/W
Reset				0	0	0	0	0

Bits 4:0 – DAC1R[4:0] Data Input Register for DAC bits

31. (ADC^2) Analog-to-Digital Converter with Computation Module

The Analog-to-Digital Converter with Computation (ADC^2) 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 (31.7.15 ADRES).

Additionally, the following features are provided within the ADC module:

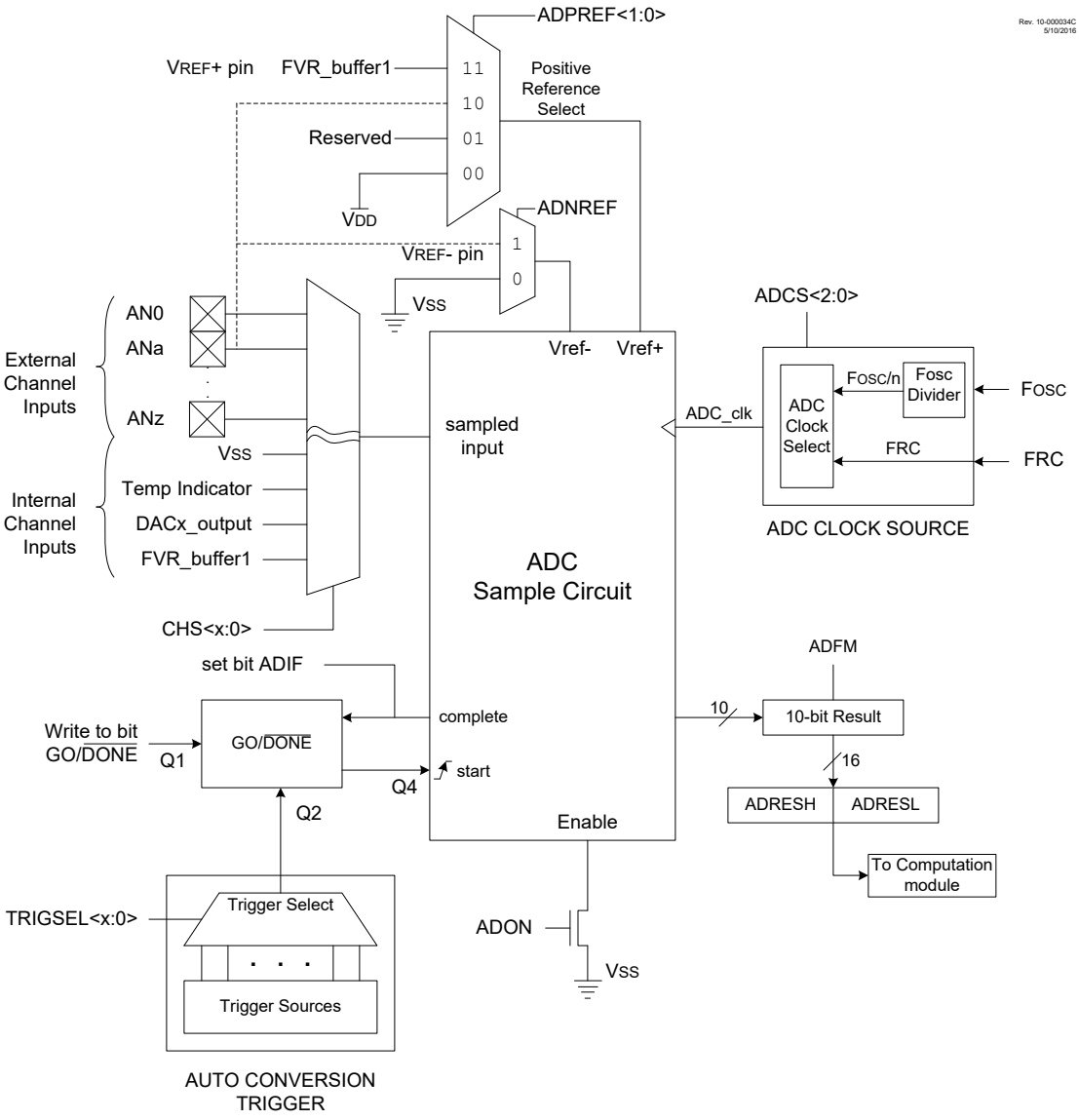
- 8-bit Acquisition Timer
- Hardware Capacitive Voltage Divider (CVD) support:
  - 8-bit precharge timer
  - Adjustable sample and hold capacitor array
  - Guard ring digital output drive
- Automatic repeat and sequencing:
  - Automated double sample conversion for CVD
  - Two sets of result registers (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 31-1 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 31-1. ADC^2 Block Diagram



31.1 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

31.1.1 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 the *"I/O Ports"* section for more information.

Important: Analog voltages on any pin that is defined as a digital input may cause the input buffer to conduct excess current.

Related Links

15. I/O Ports

31.1.2 Channel Selection

The 31.7.8 ADPCH register determines which channel is connected to the sample and hold circuit.

There are several channel selections available as shown in the following selection table:

Table 31-1. ADC Positive Input Channel Selections

ADPCH	ADC Positive Channel Input
111111	Fixed Voltage Reference (FVR)
111110	DAC1 output
111101	Temperature Indicator
111100	AVSS (Analog Ground)
011000-111011	Reserved. No channel connected.
010111	RC7/ANC7
010110	RC6/ANC6
010101	RC5/ ANC5
010100	RC4/ ANC4
010011	RC3/ANC3
010010	RC2/ANC2
010001	RC1/ ANC1
010000	RC0/ANC0
001111	RB7/ANB7
001110	RB6/ANB6
001101	RB5/ANB5

ADPCH	ADC Positive Channel Input
001100	RB4/ ANB4
001011	RB3/ANB3
001010	RB2/ ANB2
001001	RB1/ ANB1
001000	RB0/ANB0
000111	RA7/ANA7
000110	RA6/ANA6
000101	RA5/ANA5
000100	RA4/ ANA4
000011	RA3/ ANA3
000010	RA2/ ANA2
000001	RA1/ ANA1
000000	RA0/ANA0

When changing channels, a delay is required before starting the next conversion.

Refer to Section “ADC Operation” for more information.

Important: It is recommended that when switching from an ADC channel of a higher voltage to a channel of a lower voltage, the software selects the Vss channel before switching. If the ADC does not have a dedicated Vss input channel, the Vss selection (DAC1R<4:0> = b'00000') through the DAC output channel can be used. If the DAC is in use, a free input channel can be connected to Vss, and can be used in place of the DAC.

31.1.3 ADC Voltage Reference

The 31.7.7.2 ADPREF bits provide control of the positive voltage reference. The positive voltage reference can be:

- V_{REF}+ pin
- V_{DD}
- FVR 1.024V
- FVR 2.048V
- FVR 4.096V

The 31.7.7.1 ADNREF bit provides control of the negative voltage reference. The negative voltage reference can be:

- V_{REF}- pin
- V_{SS}

31.1.4 Conversion Clock

The conversion clock source is software selected with the ADCS bit. When ADCS = 1 the ADC clock source is an internal fixed-frequency clock referred to as FRC. When ADCS = 0 the ADC clock frequencies are derived from F_{OSC}. The 31.7.6 ADCLK register selects one of 64 possible clock options from $F_{OSC}/2$ to $F_{OSC}/128$:

- $F_{OSC}/2$
- $F_{OSC}/4$
- $F_{OSC}/6$
- $F_{OSC}/8$
- $F_{OSC}/10$
- ...
- $F_{OSC}/128$

The time to complete one bit conversion is defined as the T_{AD}. One full 10-bit conversion requires 11.5 T_{AD} periods as shown in Figure 31-2.

For correct conversion, the appropriate T_{AD} specification must be met. Refer to the *"ADC Timing Specifications"* for more information. The *"ADC Clock Period"* table below gives examples of appropriate ADC clock selections.

Important:

1. Except for the FRC clock source, 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 ADCS. When the ADCS is set to '1' (ADC runs on FRC), there may be unexpected delays in operation when setting ADC control bits.

Table 31-2. ADC Clock Period (T_{AD}) Vs. Device Operating Frequencies(1,4)

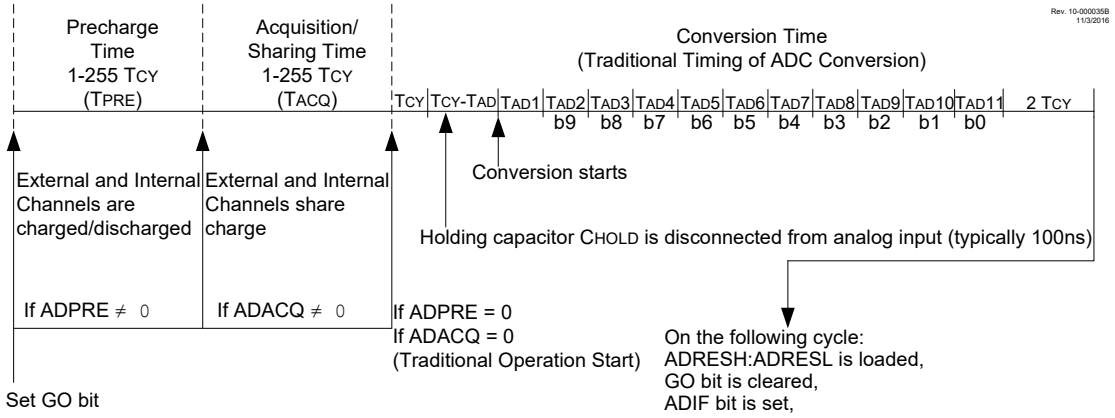
ADC Clock Period (T_{AD})		Device Frequency (F_{OSC})						
ADC Clock Source	ADCLK	64 MHz	32 MHz	20 MHz	16 MHz	8 MHz	4 MHz	1 MHz
$F_{OSC}/2$	000000	31.25 ns(2)	62.5 ns(2)	100 ns(2)	125 ns(2)	250 ns(2)	500 ns(2)	2.0 µs
$F_{OSC}/4$	000001	62.5 ns(2)	125 ns(2)	200 ns(2)	250 ns(2)	500 ns(2)	1.0 µs	4.0 µs
$F_{OSC}/6$	000010	125 ns(2)	187.5 ns(2)	300 ns(2)	375 ns(2)	750 ns(2)	1.5 µs	6.0 µs
$F_{OSC}/8$	000011	187.5 ns(2)	250 ns(2)	400 ns(2)	500 ns(2)	1.0 µs	2.0 µs	8.0 µs(3)
...	...	...	...	...	...	...	...	...
$F_{OSC}/16$	000100	250 ns(2)	500 ns(2)	800 ns(2)	1.0 µs	2.0 µs	4.0 µs	16.0 µs(3)
...	...	...	...	...	...	...	...	...

ADC Clock Period (T_{AD})		Device Frequency (F_{OSC})						
ADC Clock Source	ADCLK	64 MHz	32 MHz	20 MHz	16 MHz	8 MHz	4 MHz	1 MHz
F_{OSC}/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=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

Note:

1. See T_{AD} parameter in the "Electrical Specifications" section for FRC source typical T_{AD} value.
2. These values violate the required T_{AD} time.
3. Outside the recommended T_{AD} time.
4. The ADC clock period (T_{AD}) and total ADC conversion time can be minimized when the ADC clock is derived from the system clock F_{OSC}. However, the FRC oscillator source must be used when conversions are to be performed with the device in Sleep mode.

Figure 31-2. Analog-to-Digital Conversion T_{AD} Cycles



Related Links

38.4.8 Analog-to-Digital Converter (ADC) Conversion Timing Specifications

31.1.5 Interrupts

The ADC module allows for the ability to generate an interrupt upon completion of an Analog-to-Digital Conversion. The ADC Interrupt Flag is the ADIF bit in the PIRx register. The ADC Interrupt Enable is the ADIE bit in the PIEx register. The ADIF bit must be cleared in software.

Important:

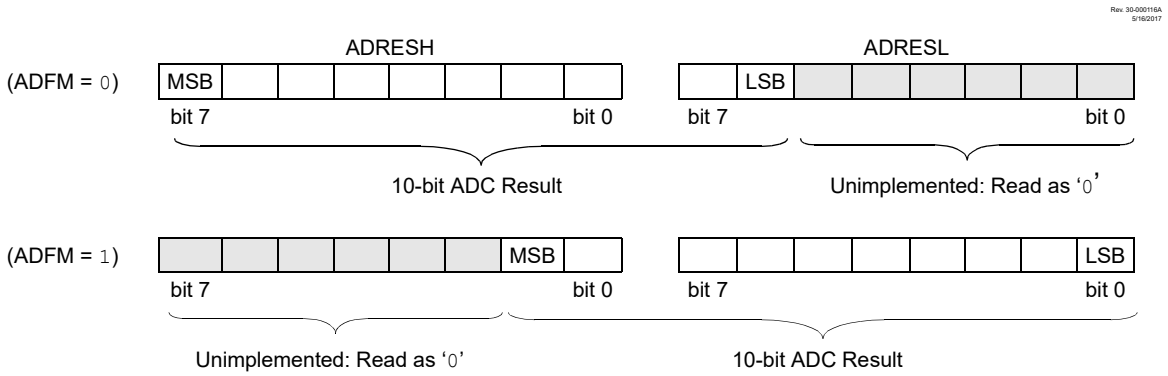
1. The ADIF bit is set at the completion of every conversion, regardless of whether or not the ADC interrupt is enabled.
2. The ADC operates during Sleep only when the FRC oscillator is selected.

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 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.

31.1.6 Result Formatting

The 10-bit ADC conversion result can be supplied in two formats, left justified or right justified. The 31.7.1.4 ADFM bit controls the output format as shown in the following figure.

Figure 31-3. 10-Bit ADC Conversion Result Format



31.2 ADC Operation

31.2.1 Starting a Conversion

To enable the ADC module, the 31.7.1.1 ADON must be set to a '1'. A conversion may be started by any of the following:

- Software setting the 31.7.1.5 ADGO bit to '1'
- An external trigger (source selected by 31.7.22 ADACT)
- A continuous-mode retrigger (see section 31.5.8 Continuous Sampling Mode)

.

Important: The ADGO bit should not be set in the same instruction that turns on the ADC. Refer to 31.2.7 ADC Conversion Procedure (Basic Mode).

31.2.2 Completion of a Conversion

When any individual conversion is complete, the value already in 31.7.15 ADRES is written into 31.7.16 ADPREV (if 31.7.3.1 ADPSIS = 0) and the new conversion results appear in ADRES. When the conversion completes, the ADC module will:

- Clear the ADGO bit (unless the 31.7.1.2 ADCONT bit is set)
- Set the ADIF Interrupt Flag bit

- Set the 31.7.5.4 ADMATH bit
- Update 31.7.17 ADACC

After every conversion when 31.7.2.4 ADDSEN = 0, or after every other conversion when ADDSEN = 1, the following events occur:

- 31.7.19 ADERR is calculated
- ADTIF interrupt is set if ADERR calculation meets threshold comparison

Important: 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.

31.2.3 Terminating a Conversion

If a conversion must be terminated before completion, the ADGO bit can be cleared in software. The partial conversion results will be discarded and the ADRES registers will retain the value from the previous conversion.

Important: A device Reset forces all registers to their Reset state. Thus, the ADC module is turned off and any pending conversion is terminated.

31.2.4 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.

31.2.5 External Trigger During Sleep

If the external trigger is received during sleep while the ADC clock source is set to the FRC, the ADC module will perform the conversion and set the ADIF bit upon completion.

If an external trigger is received when the ADC clock source is something other than FRC, the trigger will be recorded, but the conversion will not begin until the device exits Sleep.

31.2.6 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 31.7.22.1 ADACT bits.

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 the following table for auto-conversion sources.

Table 31-3. ADC Auto-Conversion Trigger Sources

ADACT	Auto-Conversion Trigger Source
11111	Software write to ADPCH
11110	Reserved, do not use
11101	Software read of ADRESH
11100	Software read of ADERRH
10000 to 11011	Reserved, do not use
01111	Interrupt-on-change Interrupt Flag
01110	C2_out
01101	C1_out
01100	PWM4_out
01011	PWM3_out
01010	CCP2_trigger
01001	CCP1_trigger
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

31.2.7 ADC Conversion Procedure (Basic Mode)

This is an example procedure for using the ADC to perform an Analog-to-Digital Conversion:

1. Configure Port:
   1.1. Disable pin output driver (Refer to the TRISx register)
   1.2. Configure pin as analog (Refer to the ANSELx register)
2. Configure the ADC module:
   2.1. Select ADC conversion clock
   2.2. Configure voltage reference
   2.3. Select ADC input channel (precharge+acquisition)
   2.4. Turn on ADC module
3. Configure ADC interrupt (optional):
   3.1. Clear ADC interrupt flag

3.2. Enable ADC interrupt
3.3. Enable peripheral interrupt (PEIE bit)
3.4. 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:
   6.1. Polling the ADGO bit
   6.2. Waiting for the ADC interrupt (interrupts enabled)
7. Read ADC Result.
8. Clear the ADC interrupt flag (required if interrupt is enabled).

Important:

1. With global interrupts disabled, the device will wake from Sleep but will not enter an Interrupt Service Routine.
2. Refer to 31.3 ADC Acquisition Requirements.

Example 31-1. ADC Conversion (assembly)

```
; 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
    clrf    ADCON1      ;
    clrf    ADCON2      ; Legacy mode, no filtering, ADRES->ADPREV
    clrf    ADCON3      ; no math functions
    clrf    ADREF       ; Vref = Vdd & Vss
    clrf    ADPCH       ; select RA0/AN0
    clrf    ADACQ       ; software controlled acquisition time
    clrf    ADCAP       ; default S&H capacitance
    clrf    ADRPT       ; no repeat measurements
    clrf    ADACT       ; auto-conversion disabled
    movlw   B'10010100' ; ADC On, right-justified, FRC clock
    movwf   ADCON0
    BANKSEL TRISA       ;
    bsf     TRISA,0     ; Set RA0 to input
    BANKSEL ANSEL       ;
    bsf     ANSEL,0     ; Set RA0 to analog
    call    SampleTime  ; Acquisiton delay
    BANKSEL ADCON0
    bsf     ADCON0,ADGO ; Start conversion
    btfsc   ADCON0,ADGO ; Is conversion done?
    goto    $-2         ; No, test again
    BANKSEL ADRESH      ;
    movf    ADRESH,W    ; Read upper 2 bits
    movwf   RESULTHI    ; store in GPR space
    movf    ADRESL,W    ; Read lower 8 bits
    movwf   RESULTLO    ; Store in GPR space
```

Example 31-2. ADC Conversion (C)

```
/*This code block configures the ADC
for polling, VDD and VSS references, ADCRC
oscillator and AN0 input.
Conversion start & polling for completion
are included.
  */
    void main() {
```

```
        //System Initialize
        initializeSystem();

        //Setup ADC
        ADCON0bits.FM = 1;       //right justify
        ADCON0bits.CS = 1;       //FRC Clock
        ADPCH = 0x00;            //RA0 is Analog channel
        TRISAbits.TRISA0 = 1;    //Set RA0 to input
        ANSELAbits.ANSELA0 = 1; //Set RA0 to analog
        ADCON0bits.ON = 1;       //Turn ADC On

    while (1) {
        ADCON0bits.GO = 1;      //Start conversion
        while (ADCON0bits.GO); //Wait for conversion done
        resultHigh = ADRESH;   //Read result
        resultLow = ADRESL;    //Read result
    }
}
```

31.3 ADC Acquisition Requirements

For the ADC to meet its specified accuracy, the charge holding capacitor (C_{HOLD}) must be allowed to fully charge to the input channel voltage level. The Analog Input model is shown in Figure 31-4. The source impedance (R_S) and the internal sampling switch (R_{SS}) impedance directly affect the time required to charge the capacitor C_{HOLD}. The sampling switch (R_{SS}) impedance varies over the device voltage (V_{DD}), refer to Figure 31-4. 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 31-1 may be used. This equation assumes that 1/2 LSb error is used (1024 steps for the ADC). The 1/2 LSb error is the maximum error allowed for the ADC to meet its specified resolution.

Equation 31-1. Acquisition Time Example

Assumptions: Temperature = 50°C; External impedance = 10kΩ; V_{DD} = 5.0V

T_{ACQ} = Amplifier Settling Time + Hold Capacitor Charging Time + Temperature Coefficient

$$T_{ACQ} = T_{AMP} + T_C + T_{COFF}$$

$$T_{ACQ} = 2\mu s + T_C + \left[\left(Temperature - 25°C\right)\left(0.05\mu s/°C\right)\right]$$

The value for T_C can be approximated with the following equations:

$$V_{APPLIED}\left(1 - \frac{1}{\left(2^{n+1}\right) - 1}\right) = V_{CHOLD}$$; [1] V_{CHOLD} charged to within ½ lsb

$$V_{APPLIED}\left(1 - e^{\frac{-T_C}{RC}}\right) = V_{CHOLD}$$; [2] V_{CHOLD} charge response to $V_{APPLIED}$

$$V_{APPLIED}\left(1 - e^{\frac{-T_C}{RC}}\right) = V_{APPLIED}\left(1 - \frac{1}{\left(2^{n+1}\right) - 1}\right)$$; Combining [1] and [2]

Note: Where n = ADC resolution in bits

Solving for T_C:

$$T_C = -C_{HOLD}\left(R_{IC} + R_{SS} + R_S\right) ln\left(1/2047\right)$$

$$T_C = -10pF\left(1k\Omega + 7k\Omega + 10k\Omega\right)\ ln\ \left(0.0004885\right)$$

$$T_C = 1.37\mu s$$

Therefore:

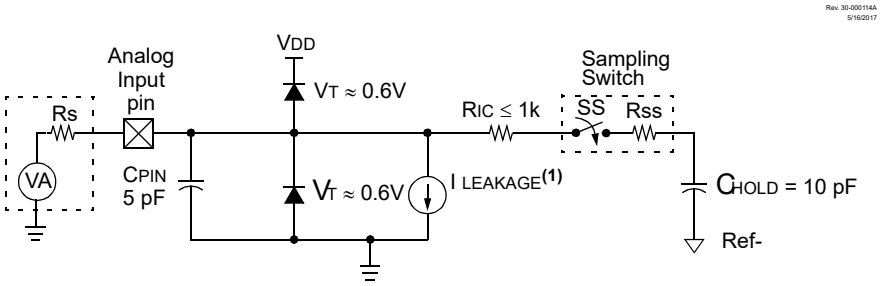
$$T_{ACQ} = 2\mu s + 892ns + \left[\left(50°C - 25°C\right)\ \left(0.05\mu s/°C\right)\right]$$

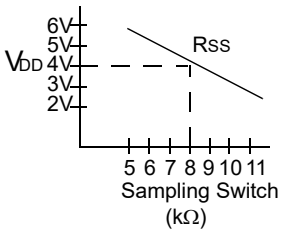
$$T_{ACQ} = 4.62\ \mu s$$

Note:

1. The reference voltage (V_{REF}) has no effect on the equation, since it cancels itself out.
2. The charge holding capacitor (C_{HOLD}) 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 31-4. Analog Input Model

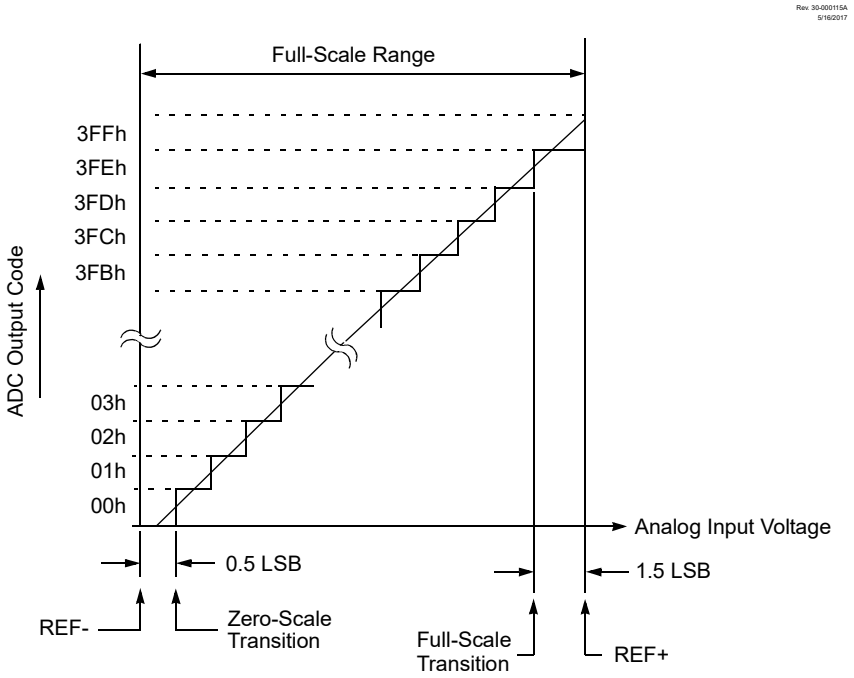




Legend:

C_{HOLD}	= Sample/Hold Capacitance
C_{PIN}	= Input Capacitance
I LEAKAGE	= Leakage current at the pin due to various junctions
R_{IC}	= Interconnect Resistance
R_{SS}	= Resistance of Sampling Switch
SS	= Sampling Switch
V_T	= Threshold Voltage

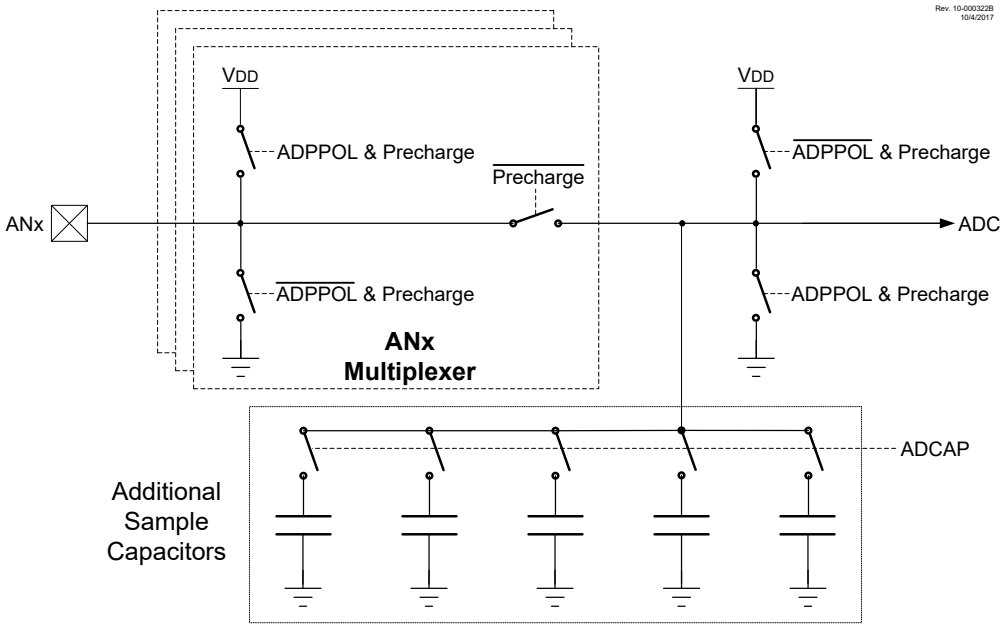
Figure 31-5. ADC Transfer Function



31.4 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. The following figure shows the basic block diagram of the CVD portion of the ADC module.

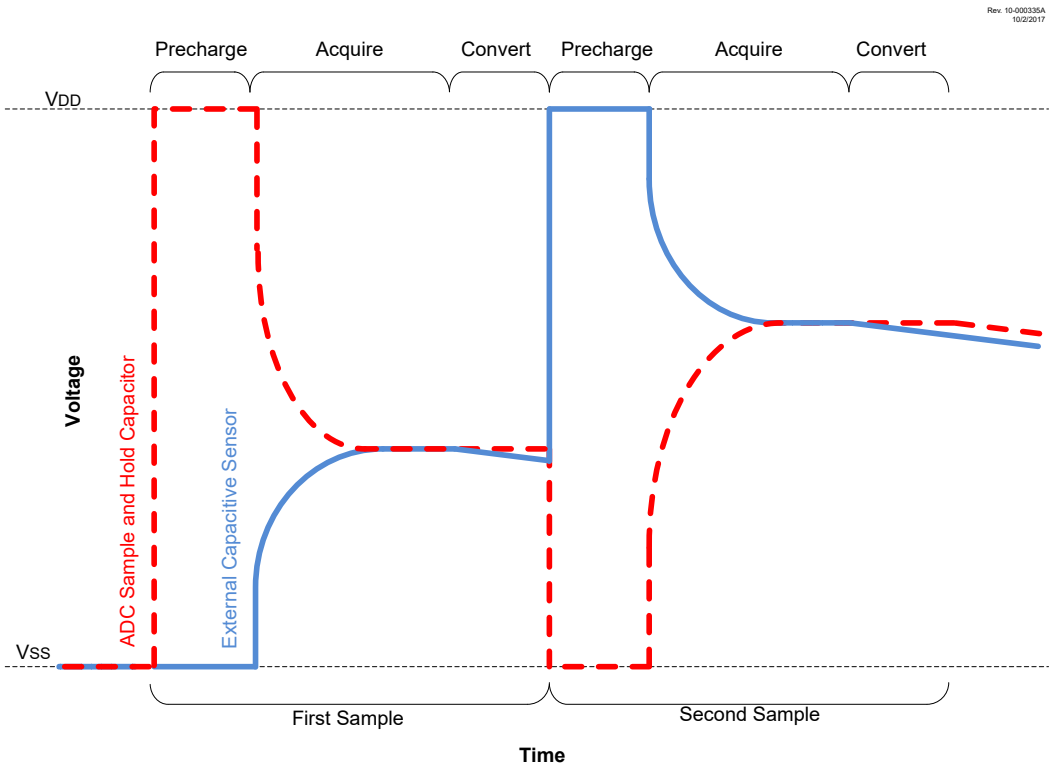
Figure 31-6. Hardware Capacitive Voltage Divider Block Diagram



31.4.1 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} the sensor node is also charged to V_{SS} or V_{DD}, respectively 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 the paths between C_{HOLD} and the external sensor node is reconnected, at which time the acquisition phase of the CVD operation begins. During acquisition, a capacitive voltage divider is formed between the precharged C_{HOLD} and sensor nodes, which results in a final voltage level setting on C_{HOLD} which is determined by the capacitances and precharge levels of the two nodes. After acquisition, the ADC converts the voltage level on C_{HOLD}. This process is then repeated with the selected precharge levels inverted for both the C_{HOLD} and the sensor nodes. The waveform for two CVD measurements, which is known as differential CVD measurement, is shown in the following figure.

Figure 31-7. Differential CVD Measurement Waveform



31.4.2 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, C_{HOLD} is disconnected from the outer portion of the sample path that leads to the external capacitive sensor and is connected to either V_{DD} or V_{SS}, depending on the value of the 31.7.2.1 ADPPOL bit. 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 such that the external sensor cap is charged opposite that of the internal C_{HOLD} cap. The amount of time that this charging needs is controlled by the ADPRE register.

Important: 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.

31.4.3 Acquisition Control for CVD (ADPRE > 0)

The Acquisition stage allows time for the voltage on the internal sample and hold capacitor to charge or discharge from the selected analog channel. This acquisition time is controlled by the ADACQ register.

When ADPRE = 0, acquisition starts at the beginning of conversion. When ADPRE > 0, 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 C_{HOLD}. This allows charge averaging to proceed between the precharged channel and the C_{HOLD} capacitor.

Important: When ADPRE > 0 setting ADACQ to '0' will set a maximum acquisition time (256 ADC clock cycles). When precharge is disabled, setting ADACQ to '0' will disable hardware acquisition time control.

31.4.4 Guard Ring Outputs

Figure 31-8 shows a typical guard ring circuit. C_{GUARD} represents the capacitance of the guard ring trace placed on the PCB board. The user selects values for R_A and R_B that will create a voltage profile on C_{GUARD}, which will match the selected acquisition channel.

The purpose of the guard ring is to generate a signal in phase with the CVD sensing signal to minimize the effects of the parasitic capacitance on sensing electrodes. It also can be used as a mutual drive for mutual capacitive sensing. For more information about active guard and mutual drive, see Application Note AN1478, "mTouch™ Sensing Solution Acquisition Methods Capacitive Voltage Divider".

The ADC has two guard ring drive outputs, ADGRDA and ADGRDB. These outputs can be routed through PPS controls to I/O pins (see **"Peripheral Pin Select (PPS) Module"** for details). The polarity of these outputs are controlled by the 31.7.2.3 ADGPOL and 31.7.2.2 ADIPEN bits.

At the start of the first precharge stage, both outputs are set to match the ADGPOL bit. Once the acquisition stage begins, ADGRDA changes polarity, while ADGRDB remains unchanged. When performing a double sample conversion, setting the ADIPEN bit 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 31-8 and Figure 31-9.

Figure 31-8. Guard Ring Circuit

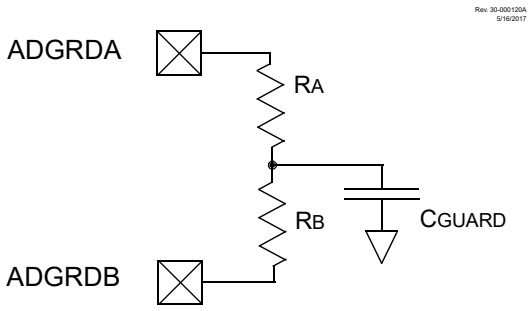


Figure 31-9. Differential CVD with Guard Ring Output Waveform

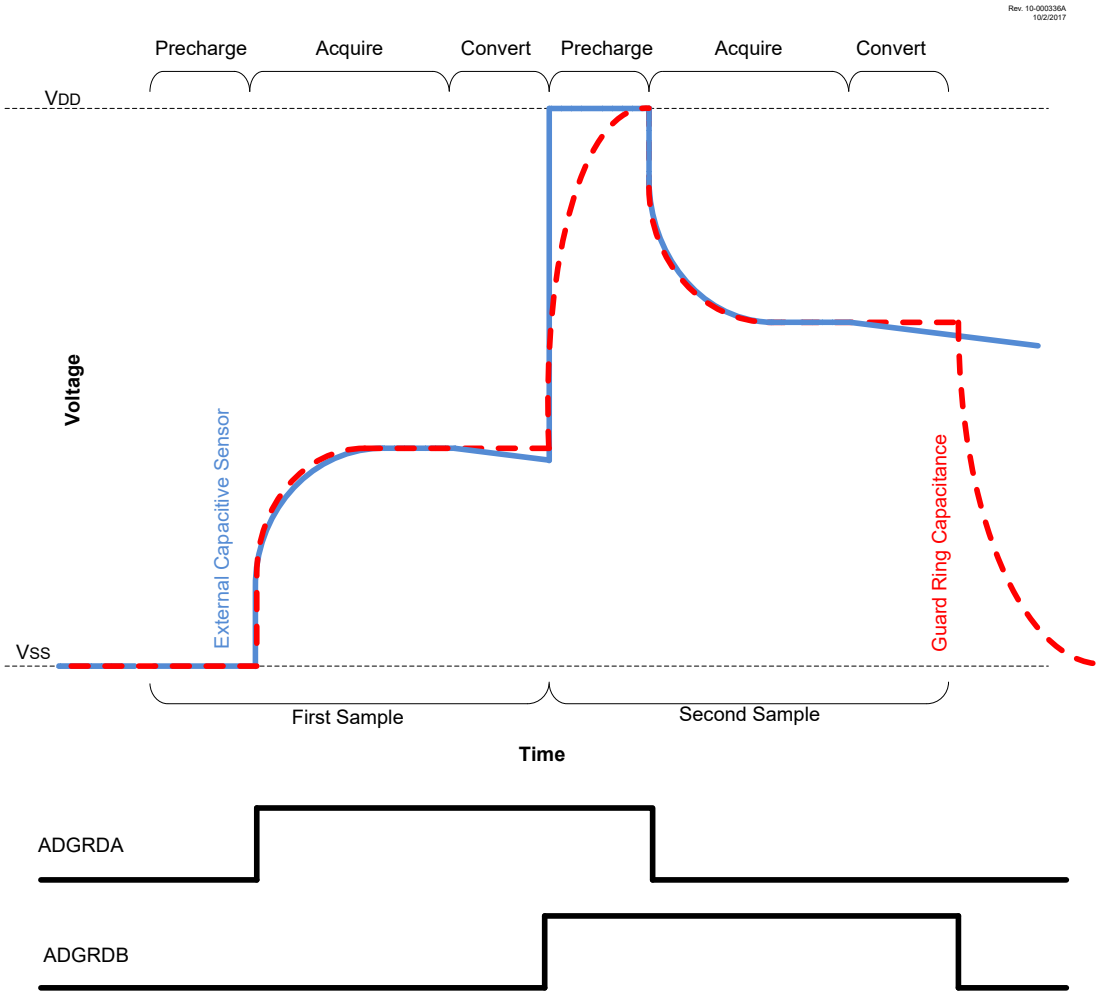
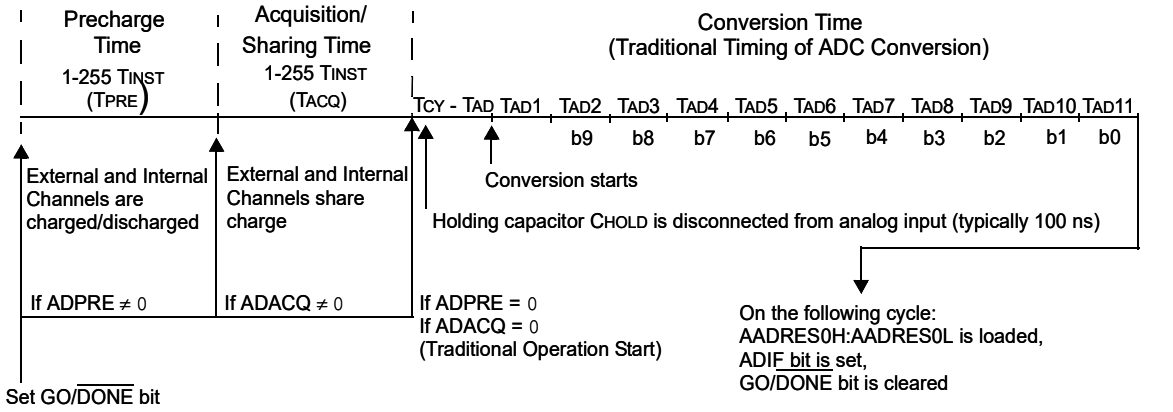


Figure 31-10. Hardware CVD Sequence Timing Diagram



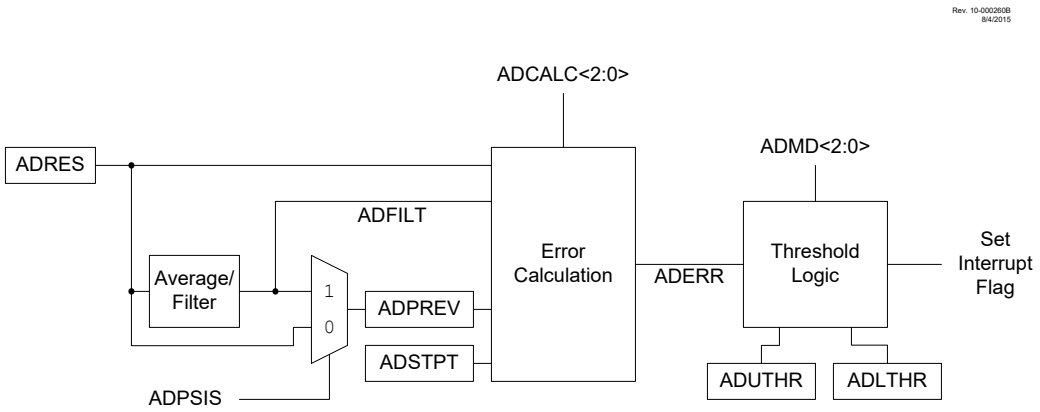
31.4.5 Additional Sample and Hold Capacitance

Additional capacitance can be added in parallel with the internal sample and hold capacitor (C_{HOLD}) by using the ADCAP register. This register selects a digitally programmable capacitance 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 31-11.

31.5 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 31-11. Computational Features Simplified Block Diagram



The operation of the ADC computational features is controlled by the 31.7.3.4 ADMD bits.

The module can be operated in one of five modes:

- Basic: This is a legacy mode. In this mode, ADC conversion occurs on single (ADDSEN = 0) or double (ADDSEN = 1) samples. ADIF is set after each conversion is complete.
- Accumulate: With each trigger, the ADC conversion result is added to the 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 the following table.

Table 31-4. Computation Modes

		Register Clear Event	Value after Cycle[2] Completion		Threshold Operations			Value at ADTIF Interrupt		
Mode	**ADMD**	**ADACC and ADCNT**	**ADACC[1]**	**ADCNT**	**Retrigger**	**Threshold Test**	**Interrupt**	**ADAOV**	**ADFLTR**	**ADCNT**
Basic	0	ADACLR = 1	Unchanged	Unchanged	No	Every Sample	If threshold=true	N/A	N/A	count
Accumulate	1	ADACLR = 1	S1 + 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	ADACLR = 1 or ADCNT>=ADRPT at ADGO or retrigger	S1 + 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	ADACLR = 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	ADACLR = 1	S1+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:										

© 2018 Microchip Technology Inc.

		Register Clear Event	Value after Cycle[2] Completion		Threshold Operations			Value at ADTIF Interrupt		
Mode	ADMD	ADACC and ADCNT	ADACC[1]	ADCNT	Retrigger	Threshold Test	Interrupt	ADAOV	ADFLTR	ADCNT

1. 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.
2. When ADDSEN = 0 then Cycle means one conversion. When ADDSEN = 1 the Cycle means two conversions.

© 2018 Microchip Technology Inc.

31.5.1 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 31.7.17 ADACC registers.

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 $2^{(accumulator_width)}$-1 = 2^{16} = 65535, the 31.7.5.1 ADAOV overflow bit 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 31.7.3.3 ADACLR bit. Setting the ADACLR bit will also clear the 31.7.5.1 ADAOV (Accumulator overflow) bit, as well as the ADCNT register. The ADACLR bit is cleared by the hardware when accumulator clearing action is complete.



Important: When ADC is operating from FRC, five FRC clock cycles are required to execute the ADACC clearing operation.

The 31.7.3.2 ADCRS bits control the data shift on the accumulator result, which effectively divides the value in accumulator (31.7.17 ADACC) registers. 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 31-5 shows the -3 dB cut-off frequency in ωT (radians) and the highest signal attenuation obtained by this filter at nyquist frequency (ωT = π).

Table 31-5. Low-pass Filter -3 dB Cut-off Frequency

ADCRS	ωT (radians) @ -3 dB Frequency	dB @ $F_{nyquist}$=1/(2T)
1	0.72	-9.5
2	0.284	-16.9
3	0.134	-23.5
4	0.065	-29.8
5	0.032	-36.0
6	0.016	-42.0
7	0.0078	-48.1

31.5.2 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.

31.5.3 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 31.7.3.2 ADCRS bits. 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 31.5.7 Threshold Comparison) and the ADTIF interrupt may trigger.

31.5.4 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 ADRPT = 2^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.

31.5.5 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 31.5.8 Continuous Sampling Mode) is not enabled. This allows for a threshold comparison on the average of a short burst of ADC samples.

31.5.6 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 31.5 Computation Operation 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 31.5.1 Digital Filter/Average).

31.5.7 Threshold Comparison

At the end of each computation:

- The conversion results are latched and held stable at the end-of-conversion.
- The error (31.7.19 ADERR) is calculated based on a difference calculation which is selected by the 31.7.4.1 ADCALC bits. The value can be one of the following calculations (see Table 31-6 for more details):
  - The first derivative of single measurements
  - The CVD result when double-sampling is enabled
  - 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, 31.7.21 ADUTH and 31.7.20 ADLTH registers, to set the 31.7.5.2 ADUTHR and 31.7.5.3 ADLTHR flag bits. The threshold logic is selected by 31.7.4.3 ADTMD bits. The threshold trigger option can be one of the following:
  - Never interrupt
  - Error is less than lower threshold
  - Error is greater than or equal to lower threshold

- Error is between thresholds (inclusive)
- Error is outside of thresholds
- Error is less than or equal to upper threshold
- Error is greater than upper threshold
- Always interrupt regardless of threshold test results
- If the threshold condition is met, the threshold interrupt flag ADTIF is set.

Note:

1. The threshold tests are signed operations.
2. If ADAOV is set, a threshold interrupt is signaled. It is good practice for threshold interrupt handlers to verify the validity of the threshold by checking ADAOV.

Table 31-6. ADC Error Calculation Mode

ADCALC	ADERR		Application
	ADDSEN = 0 Single-Sample Mode	ADDSEN = 1 CVD Double-Sample Mode(1)	
111	ADFLTR	ADFLTR	Filtered results above or below the threshold.
110	ADRES	ADRES	Measurement above or below the threshold
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(1,2)

Note:

1. When ADDSEN=1, ADERR is computed only after every second sample.
2. When ADPSIS = 0.
3. When ADPSIS = 1.

31.5.8 Continuous Sampling Mode

Setting the 31.7.1.2 ADCONT bit 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 31.7.4.2 ADSOI A/D Stop-on-interrupt bit is set (correct logic).

31.5.9 Double Sample Conversion

Double sampling is enabled by setting the 31.7.2.4 ADDSEN bit. When this bit is set, two conversions are required before the module will calculate threshold error. Each conversion must still be triggered separately when ADCONT = 0. The first conversion will set the 31.7.5.4 ADMATH bit and update ADACC, but will not calculate ADERR or trigger ADTIF. When the second conversion completes, the first value is transferred to ADPREV (depending on the setting of 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).

31.6 Register Summary - ADC Control

Address	Name	Bit Pos.								
0x0F51	ADACT	7:0				ADACT[4:0]				
0x0F52	ADCLK	7:0			ADCS[5:0]					
0x0F53	ADREF	7:0				ADNREF			ADPREF[1:0]	
0x0F54	ADCON1	7:0	ADPPOL	ADIPEN	ADGPOL					ADDSEN
0x0F55	ADCON2	7:0	ADPSIS	ADCRS[2:0]			ADACLR	ADMD[2:0]		
0x0F56	ADCON3	7:0		ADCALC[2:0]			ADSOI	ADTMD[2:0]		
0x0F57	ADACQ	7:0	ADACQ[7:0]							
0x0F58	ADCAP	7:0				ADCAP[4:0]				
0x0F59	ADPRE	7:0	ADPRE[7:0]							
0x0F5A	ADPCH	7:0			ADPCH[5:0]					
0x0F5B	ADCON0	7:0	ADON	ADCONT		ADCS		ADFM		ADGO
0x0F5C	ADPREV	7:0	ADPREVL[7:0]							
		15:8	ADPREVH[7:0]							
0x0F5E	ADRES	7:0	ADRESL[7:0]							
		15:8	ADRESH[7:0]							
0x0F60	ADSTAT	7:0	ADAOV	ADUTHR	ADLTHR	ADMATH		ADSTAT[2:0]		
0x0F61	ADRPT	7:0	ADRPT[7:0]							
0x0F62	ADCNT	7:0	ADCNT[7:0]							
0x0F63	ADSTPT	7:0	ADSTPTL[7:0]							
		15:8	ADSTPTH[7:0]							
0x0F65	ADLTH	7:0	ADLTHL[7:0]							
		15:8	ADLTHH[7:0]							
0x0F67	ADUTH	7:0	ADUTHL[7:0]							
		15:8	ADUTHH[7:0]							
0x0F69	ADERR	7:0	ADERRL[7:0]							
		15:8	ADERRH[7:0]							
0x0F6B	ADACC	7:0	ADACCL[7:0]							
		15:8	ADACCH[7:0]							
0x0F6D	ADFLTR	7:0	ADFLTRL[7:0]							
		15:8	ADFLTRH[7:0]							

31.7 Register Definitions: ADC Control

31.7.1 ADCON0

Name: ADCON0
Address: 0xF5B

ADC Control Register 0

Bit	7	6	5	4	3	2	1	0
	ADON	ADCONT		ADCS		ADFM		ADGO
Access	R/W	R/W		R/W		R/W		R/W/HC
Reset	0	0		0		0		0

Bit 7 – ADON ADC Enable bit

Value	Description
1	ADC is enabled
0	ADC is disabled

Bit 6 – ADCONT ADC Continuous Operation Enable bit

Value	Description
1	ADGO is retriggered upon completion of each conversion trigger until ADTIF is set (if 31.7.4.2 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 4 – ADCS ADC Clock Selection bit

Value	Description
1	Clock supplied from FRC dedicated oscillator
0	Clock supplied by Fosc, divided according to ADCLK register

Bit 2 – ADFM ADC results Format/alignment Selection

Value	Description
1	ADRES and ADPREV data are right-justified
0	ADRES and ADPREV data are left-justified, zero-filled

Bit 0 – ADGO ADC Conversion Status bit

Value	Description
1	ADC conversion cycle in progress. Setting this bit starts an ADC conversion cycle. The bit is cleared by hardware as determined by the ADCONT bit
0	ADC conversion completed/not in progress

31.7.2 ADCON1

Name: ADCON1
Address: 0xF54

ADC Control Register 1

Bit	7	6	5	4	3	2	1	0
	ADPPOL	ADIPEN	ADGPOL					ADDSEN
Access	R/W	R/W	R/W					R/W
Reset	0	0	0					0

Bit 7 – ADPPOL Precharge Polarity bit
Action During 1st Precharge Stage

Value	Condition	Description
x	ADPRE=0	Bit has no effect
1	ADPRE>0 & ADC input is I/O pin	Pin shorted to AV_{DD}
0	ADPRE>0 & ADC input is I/O pin	Pin shorted to V_{SS}
1	ADPRE>0 & ADC input is internal	C_{HOLD} Shorted to AV_{DD}
0	ADPRE>0 & ADC input is internal	C_{HOLD} Shorted to V_{SS}

Bit 6 – ADIPEN A/D Inverted Precharge Enable bit

Value	Condition	Description
x	ADDSEN = 0	Bit has no effect
1	ADDSEN = 1	The precharge and guard signals in the second conversion cycle are the opposite polarity of the first cycle
0	ADDSEN = 1	Both Conversion cycles use the precharge and guards specified by ADPPOL and ADGPOL

Bit 5 – ADGPOL Guard Ring Polarity Selection bit

Value	Description
1	ADC guard Ring outputs start as digital high during Precharge stage
0	ADC guard Ring outputs start as digital low during Precharge stage

Bit 0 – ADDSEN Double-Sample Enable bit

Value	Description
1	Two conversions are processed as a pair. The selected computation is performed after every second conversion.
0	Selected computation is performed after every conversion

31.7.3 ADCON2

Name: ADCON2
Address: 0xF55

ADC Control Register 2

Bit	7	6	5	4	3	2	1	0
	ADPSIS	ADCRS[2:0]			ADACLR	ADMD[2:0]		
Access	R/W	R/W	R/W	R/W	R/W/HC	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

Bit 7 – ADPSIS ADC Previous Sample Input Select bits

Value	Description
1	ADFLTR is transferred to ADPREV at start-of-conversion
0	ADRES is transferred to ADPREV at start-of-conversion

Bits 6:4 – ADCRS[2:0] ADC Accumulated Calculation Right Shift Select bits

Value	Condition	Description
0 to 7	ADMD = 'b100	Low-pass filter time constant is 2^{ADCRS}, filter gain is 1:1
0 to 7	ADMD =' b011 to 'b001	The accumulated value is right-shifted by ADCRS (divided by 2^{ADCRS})**(1,2)**
x	ADMD ='b000 to 'b001	These bits are ignored

Bit 3 – ADACLR A/D Accumulator Clear Command bit**(3)**

Value	Description
1	ADACC, ADAOV and ADCNT registers are cleared
0	Clearing action is complete (or not started)

Bits 2:0 – ADMD[2:0] ADC Operating Mode Selection bits**(4)**

Value	Description
111-101	Reserved
100	Low-pass Filter mode
011	Burst Average mode
010	Average mode
001	Accumulate mode
000	Basic (Legacy) mode

Note:

1. To correctly calculate an average, the number of samples (set in ADRPT) must be 2ADCRS.
2. ADCRS = '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 31-4 for Full mode descriptions.

31.7.4 ADCON3

Name: ADCON3
Address: 0xF56

ADC Control Register 3

Bit	7	6	5	4	3	2	1	0
		ADCALC[2:0]			ADSOI	ADTMD[2:0]		
Access		R/W	R/W	R/W	R/W/HC	R/W	R/W	R/W
Reset		0	0	0	0	0	0	0

Bits 6:4 – ADCALC[2:0] ADC Error Calculation Mode Select bits
See Table 31-6 table for selection details.

Bit 3 – ADSOI ADC Stop-on-Interrupt bit

Value	Condition	Description
1	ADCONT = 1	ADGO is cleared when the threshold conditions are met, otherwise the conversion is retriggered
0	ADCONT = 1	ADGO is not cleared by hardware, must be cleared by software to stop retriggers
x	ADCONT = 0	This bit is not used

Bits 2:0 – ADTMD[2:0] Threshold Interrupt Mode Select bits

Value	Description
111	Interrupt regardless of threshold test results
110	Interrupt if ADERR>ADUTH
101	Interrupt if ADERR≤ADUTH
100	Interrupt if ADERR<ADLTH or ADERR>ADUTH
011	Interrupt if ADERR>ADLTH and ADERR<ADUTH
010	Interrupt if ADERR≥ADLTH
001	Interrupt if ADERR<ADLTH
000	Never interrupt

31.7.5 ADSTAT

Name: ADSTAT
Address: 0xF60

ADC Status Register

Bit	7	6	5	4	3	2	1	0
	ADAOV	ADUTHR	ADLTHR	ADMATH		ADSTAT[2:0]		
Access	R/C/HS/HC	RO	RO	R/C/HS/HC		RO	RO	RO
Reset	0	0	0	0		0	0	0

Bit 7 – ADAOV ADC Accumulator Overflow bit

Value	Description
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

Value	Description
1	ADERR >ADUTH
0	ADERR≤ADUTH

Bit 5 – ADLTHR ADC Module Less-than Lower Threshold Flag bit

Value	Description
1	ADERR<ADLTH
0	ADERR≥ADLTH

Bit 4 – ADMATH ADC Module Computation Status bit

Value	Description
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

Bits 2:0 – ADSTAT[2:0]
ADC Module Cycle Multi-Stage Status bits(1)

Value	Description
111	ADC module is in 2nd conversion stage
110	ADC module is in 2nd acquisition stage
101	ADC module is in 2nd precharge stage
100	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 $F_{OSC}<F_{RC}$, the indicated status may not be valid.

31.7.6 ADCLK

Name: ADCLK
Address: 0xF52

ADC Clock Selection Register

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

Bits 5:0 – ADCS[5:0] ADC Conversion Clock Select bits

Value	Description
n	ADC Clock frequency = $F_{OSC}/(2*(n+1))$

31.7.7 ADREF

Name: ADREF
Address: 0xF53

ADC Reference Selection Register

Bit	7	6	5	4	3	2	1	0
				ADNREF			ADPREF[1:0]	
Access				R/W			R/W	R/W
Reset				0			0	0

Bit 4 – ADNREF ADC Negative Voltage Reference Selection bit

Value	Description
1	V_{REF}- is connected to external V_{REF}-
0	V_{REF}- is connected to AV_{SS}

Bits 1:0 – ADPREF[1:0] ADC Positive Voltage Reference Selection bits

Value	Description
11	V_{REF}+ is connected to internal Fixed Voltage Reference (FVR) module
10	V_{REF}+ is connected to external V_{REF}+
01	Reserved
00	V_{REF}+ is connected to V_{DD}

31.7.8 ADPCH

Name: ADPCH
Address: 0xF5A

ADC Positive Channel Selection Register

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

Bits 5:0 – ADPCH[5:0] ADC Positive Input Channel Selection bits
See Channel Selection Table for input selection details.

31.7.9 ADPRE

Name: ADPRE
Address: 0xF59

ADC Precharge Time Control Register

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

Bits 7:0 – ADPRE[7:0] Precharge Time Select bits

Value	Description
1 to 255	Number of ADC clocks in the precharge time.
0	Precharge time is not included in the data conversion cycle

31.7.10 ADACQ

Name: ADACQ
Address: 0xF57

ADC Acquisition Time Control Register

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

Bits 7:0 – ADACQ[7:0] Acquisition (charge share time) Select bits

Value	Description
0x01 to 0xFF	Number of ADC clock periods in the acquisition time
0x00	Acquisition time is not included in the data conversion cycle

31.7.11 ADCAP

Name: ADCAP
Address: 0xF58

ADC Additional Sample Capacitor Selection Register

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

Bits 4:0 – ADCAP[4:0] ADC Additional Sample Capacitor Selection bits

Value	Description
1 to 31	Number of pF in the additional capacitance
0	No additional capacitance

31.7.12 ADRPT

Name: ADRPT
Address: 0xF61

ADC Repeat Setting Register

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

Bits 7:0 – ADRPT[7:0] ADC Repeat Threshold bits
Determines the number of times that the ADC is triggered for a threshold check. When ADCNT reaches this value the error threshold is checked. Used when the computation mode is Low-pass Filter, Burst Average, or Average. See Table 31-4 for more details.

31.7.13 ADCNT

Name: ADCNT
Address: 0xF62

ADC Repeat Counter Register

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

Bits 7:0 – ADCNT[7:0] ADC Repeat Count bits
Counts the number of times that the ADC is triggered before the threshold is checked. When this value reaches ADPRT then the threshold is checked. Used when the computation mode is Low-pass Filter, Burst Average, or Average. See Table 31-4 for more details.

31.7.14 ADFLTR

Name: ADFLTR
Address: 0xF6D

ADC Filter Register. 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.

Bit	15	14	13	12	11	10	9	8
	ADFLTRH[7:0]							
Access	RO	RO	RO	RO	RO	RO	RO	RO
Reset	x	x	x	x	x	x	x	x

Bit	7	6	5	4	3	2	1	0
	ADFLTRL[7:0]							
Access	RO	RO	RO	RO	RO	RO	RO	RO
Reset	x	x	x	x	x	x	x	x

Bits 15:8 – ADFLTRH[7:0] ADC Filter Output Most Significant bits

Bits 7:0 – ADFLTRL[7:0] ADC Filter Output Least Significant bits

31.7.15 ADRES

Name: ADRES
Address: 0xF5E

ADC Result Register

Bit	15	14	13	12	11	10	9	8
	ADRESH[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

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

Bits 15:8 – ADRESH[7:0] ADC Result Register bits. High bits

Value	Condition	Description
0x00,0x01,0x02,0x03	31.7.1.4 ADFM=1	Upper 2 bits of result
0 to 0xFF	31.7.1.4 ADFM=0	Upper 8 bits of result

Bits 7:0 – ADRESL[7:0] ADC Result Register bits. Lower bits

Value	Condition	Description
0 to 0xFF	31.7.1.4 ADFM=1	Lower 8 bits of result
0x00,0x40,0x80,0xC0	31.7.1.4 ADFM=0	Lower 2 bits of result

31.7.16 ADPREV

Name: ADPREV
Address: 0xF5C

ADC Previous Result Register

Bit	15	14	13	12	11	10	9	8
	ADPREVH[7:0]							
Access	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	ADPREVL[7:0]							
Access	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0

Bits 15:8 – ADPREVH[7:0] Previous ADC Result Most Significant bits

Value	Condition	Description
0 to 0xFF	31.7.3.1 ADPSIS=1	Upper byte of ADFLTR at the start of current ADC conversion
varies	31.7.3.1 ADPSIS=0	Upper bits of ADRES at the start of current ADC conversion(1)

Bits 7:0 – ADPREVL[7:0] Previous ADC Result Least Significant bits

Value	Condition	Description
0 to 0xFF	31.7.3.1 ADPSIS=1	Lower byte of ADFLTR at the start of current ADC conversion
varies	31.7.3.1 ADPSIS=0	Lower bits of ADRES at the start of current ADC conversion(1)

Note: If ADPSIS = 0, ADPREVH and ADPREVL are formatted the same way as ADRES is, depending on the ADFM bit.

31.7.17 ADACC

Name: ADACC
Address: 0xF6B

ADC Accumulator Register
See Table 31-4 for more details.

Bit	15	14	13	12	11	10	9	8
	ADACCH[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	x	x	x	x	x	x	x	x

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

Bits 15:8 – ADACCH[7:0]
ADC Accumulator Most Significant Byte.
Reset States: POR/BOR = xxxxxxxx
All Other Resets = uuuuuuuu

Bits 7:0 – ADACCL[7:0]
ADC Accumulator Least Significant Byte.
Reset States: POR/BOR = xxxxxxxx
All Other Resets = uuuuuuuu

31.7.18 ADSTPT

Name: ADSTPT
Address: 0xF63

ADC Threshold Setpoint Register
Depending on 31.7.4.1 ADCALC, may be used to determine ADERR. See Table 31-6 for more details.

Bit	15	14	13	12	11	10	9	8
	ADSTPTH[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

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

Bits 15:8 – ADSTPTH[7:0]
ADC Threshold Setpoint Most Significant Byte.

Bits 7:0 – ADSTPTL[7:0]
ADC Threshold Setpoint Least Significant Byte.

31.7.19 ADERR

Name: ADERR
Address: 0xF69

ADC Setpoint Error Register. ADC Setpoint Error calculation is determined by the 31.7.4.1 ADCALC bits.

Bit	15	14	13	12	11	10	9	8
	ADERRH[7:0]							
Access	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0

Bit	7	6	5	4	3	2	1	0
	ADERRL[7:0]							
Access	RO	RO	RO	RO	RO	RO	RO	RO
Reset	0	0	0	0	0	0	0	0

Bits 15:8 – ADERRH[7:0]
ADC Setpoint Error MSB

Bits 7:0 – ADERRL[7:0]
ADC Setpoint Error LSB

31.7.20 ADLTH

Name: ADLTH
Address: 0xF65

ADC Lower Threshold Register

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.

Bit	15	14	13	12	11	10	9	8
	ADLTHH[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

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

Bits 15:8 – ADLTHH[7:0] ADC Lower Threshold MSB

Bits 7:0 – ADLTHL[7:0] ADC Lower Threshold LSB

31.7.21 ADUTH

Name: ADUTH
Address: 0xF67

ADC Upper Threshold Register

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.

Bit	15	14	13	12	11	10	9	8
	ADUTHH[7:0]							
Access	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Reset	0	0	0	0	0	0	0	0

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

Bits 15:8 – ADUTHH[7:0] ADC Upper Threshold MSB

Bits 7:0 – ADUTHL[7:0] ADC Upper Threshold LSB

31.7.22 ADACT

Name: ADACT
Address: 0xF51

ADC AUTO Conversion Trigger Source Selection Register

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

Bits 4:0 – ADACT[4:0] Auto-Conversion Trigger Select Bits

Value	Description
00000 to 11111	See Auto-Conversion Trigger Sources table.

32. (CMP) Comparator Module

Comparators are used to interface analog circuits to a digital circuit by comparing two analog voltages and providing a digital indication of their relative magnitudes. Comparators are very useful mixed-signal building blocks because they provide analog functionality independent of program execution. The PIC18F24/25Q10 devices have two comparators (C1/C2).

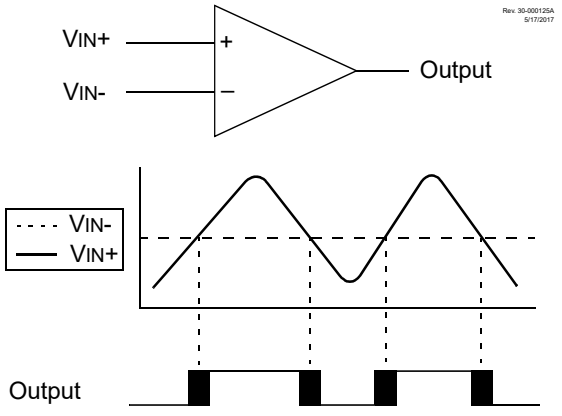
The analog comparator module includes the following features:

- 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
- Odd numbered timers (Timer1, Timer3, etc.) Gate
- Even numbered timers (Timer2, Timer4, etc.) Reset
- CCP Capture Mode Input
- DSM Modulator Source
- Input and Window Signal-to-Signal Measurement Timer

32.1 Comparator Overview

A single comparator is shown in Figure 32-1 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.

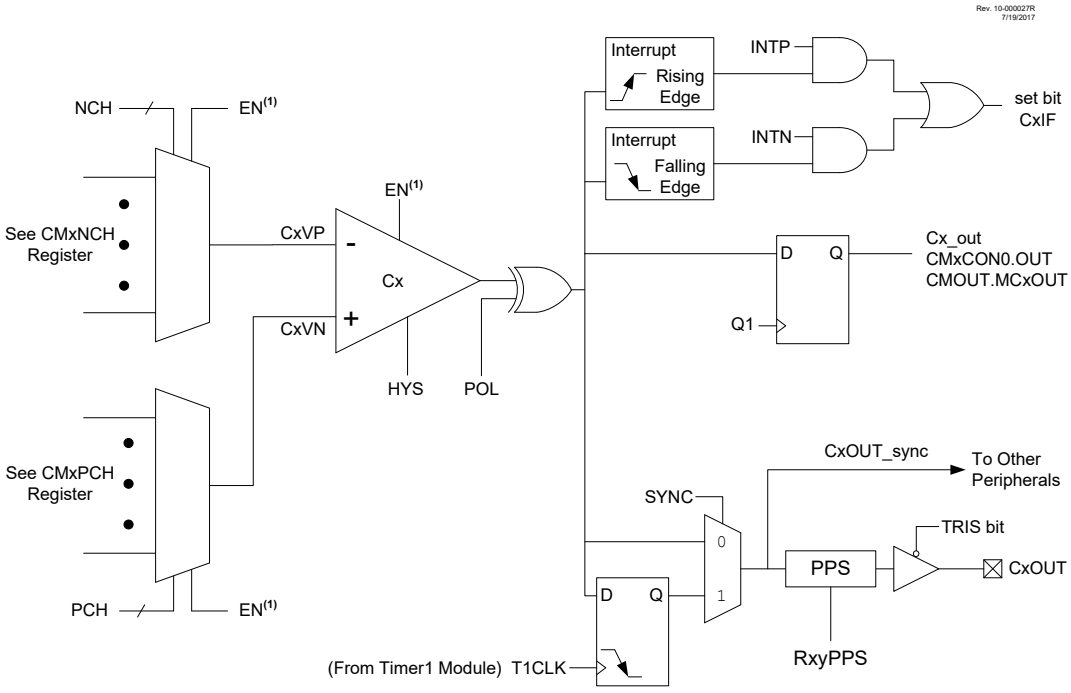
Figure 32-1. Single Comparator



Note:

1. The black areas of the output of the comparator represent the uncertainty due to input offsets and response time.

Figure 32-2. Comparator Module Simplified Block Diagram



Note 1: When EN = 0, all multiplexer inputs are disconnected and the Comparator will produce a '0' at the output.

Related Links

32.15.3 CMxNCH

32.15.4 CMxPCH

32.2 Comparator Control

Each comparator has two control registers: CMxCON0 and CMxCON1.

The CMxCON0 register contains Control and Status bits for the following:

- Enable
- Output
- Output polarity
- Hysteresis enable
- Timer1 output synchronization

The CMxCON1 register contains Control bits for the following:

- Interrupt on positive/negative edge enables
- Positive input channel selection
- Negative input channel selection

32.2.1 Comparator Enable

Setting the EN bit enables the comparator for operation. Clearing the CxEN bit disables the comparator, resulting in minimum current consumption.

32.2.2 Comparator Output

The output of the comparator can be monitored by reading either the CxOUT bit or the 32.15.5.1 MCxOUT bit.

The comparator output can also be routed to an external pin through the RxyPPS register. 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.

Related Links

17.9.2 RxyPPS

32.2.3 Comparator Output Polarity

Inverting the output of the comparator is functionally equivalent to swapping the comparator inputs. The polarity of the comparator output can be inverted by setting the CxPOL bit. Clearing the CxPOL bit results in a non-inverted output.

Table 32-1 shows the output state versus input conditions, including polarity control.

Table 32-1. 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

32.3 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.

See Comparator Specifications for more information.

Related Links

38.4.9 Comparator Specifications

32.4 Operation With Timer1 Gate

The output resulting from a comparator operation can be used as a source for gate control of the odd numbered timers (Timer1, Timer3, etc.). See the timer gate section 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 the timer by setting the 32.15.1.5 SYNC bit. This ensures that the timer does not increment while a change in the comparator is occurring. However, synchronization is only possible with the Timer1 clock source. Synchronization with the other odd numbered timers is only possible when they use the same clock source as Timer1.

Related Links

19.7 Timer1 Gate

32.4.1 Comparator Output Synchronization

The output from a comparator can be synchronized with Timer1 by setting the SYNC bit.

Once enabled, the comparator output is latched on the falling edge of the Timer1 source clock. If a prescaler is used with Timer1, the comparator output is latched after the prescaling function. To prevent a race condition, the comparator output is latched on the falling edge of the Timer1 clock source and Timer1 increments on the rising edge of its clock source. See the Figure 32-2 Comparator Block Diagram and the Timer1 Block Diagram for more information.

Related Links

19. Timer1 Module with Gate Control

32.5 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), the Corresponding Interrupt Flag bit (CxIF bit of the PIR2 register) will be set.

To enable the interrupt, the following bits must be set:

- EN and POL bits
- CxIE bit of the PIE2 register
- INTP bit (for a rising edge detection)
- INTN bit (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.



Important: Although a comparator is disabled, an interrupt can be generated by changing the output polarity with the CxPOL bit, or by switching the comparator on or off with the CxEN bit.

32.6 Comparator Positive Input Selection

Configuring the 32.15.4.1 PCH bits direct an internal voltage reference or an analog pin to the non-inverting input of the comparator:

PCH	Positive Input Source
111	AVSS
110	FVR_Buffer2
101	DAC_Output
100	CxPCH not connected
011	CxPCH not connected
010	CxPCH not connected

PCH	Positive Input Source
001	CxIN1+
000	CxIN0+

Important: To use 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.

See Fixed Voltage Reference (FVR) for more information on the Fixed Voltage Reference module.

See 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.

Related Links

28. (FVR) Fixed Voltage Reference
30. (DAC) 5-Bit Digital-to-Analog Converter Module

32.7 Comparator Negative Input Selection

The 32.15.3.1 NCH bits direct an analog input pin and internal reference voltage or analog ground to the inverting input of the comparator:

NCH	Negative Input Sources
111	AVSS
110	FVR_Buffer2
101	CxNCH not connected
100	CxNCH not connected
011	CxIN3-
010	CxIN2-
001	CxIN1-
000	CxIN0-

Important: To use 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.

32.8 Comparator Response Time

The comparator output is indeterminate for a period of time after the change of an input source or the selection of a new reference voltage. This period is referred to as the response time. The response time of the comparator differs from the settling time of the voltage reference. Therefore, both of these times must be considered when determining the total response time to a comparator input change. See the

Comparator and Voltage Reference Specifications in Comparator Specifications and Fixed Voltage Reference (FVR) Specifications for more details.

Related Links

38.4.9 Comparator Specifications

38.4.11 Fixed Voltage Reference (FVR) Specifications

32.9 Analog Input Connection Considerations

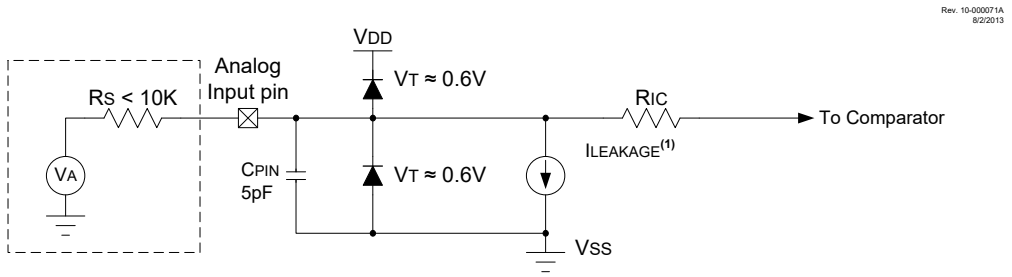
A simplified circuit for an analog input is shown in Figure 32-3. Since the analog input pins share their connection with a digital input, they have reverse biased ESD protection diodes to V_{DD} and V_{SS}. The analog input, therefore, must be between V_{SS} and V_{DD}. If the input voltage deviates 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 kΩ is recommended for the analog sources. Also, any external component connected to an analog input pin, such as a capacitor or a Zener diode, 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 32-3. Analog Input Model



Legend:

CPIN	= Input Capacitance
ILEAKAGE	= Leakage Current at the pin due to various junctions
RIC	= Interconnect Resistance
RS	= Source Impedance
VA	= Analog Voltage
VT	= Threshold Voltage

Note: See *Electrical Specifications* chapter.

Related Links

38. Electrical Specifications

32.10 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.

Related Links

24.11.1.2 External Input Source

32.11 ADC Auto-Trigger Source

The output of the comparator module can be used to trigger an ADC conversion. When the ADACT register is set to trigger on a comparator output, an ADC conversion will trigger when the comparator output goes high.

32.12 Even Numbered Timers Reset

The output of the comparator module can be used to reset the even numbered timers (Timer2, Timer4, etc.). When the TxERS register is appropriately set, the timer will reset when the comparator output goes high.

32.13 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 (F_{OSC}) or the instruction clock ($F_{OSC}/4$), 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 PIEx register must be set to enable comparator interrupts.

32.14 Register Summary - Comparator

Address	Name	Bit Pos.								
0x0F30	CM2CON0	7:0	EN	OUT		POL			HYS	SYNC
0x0F31	CM2CON1	7:0							INTP	INTN
0x0F32	CM2NCH	7:0						NCH[2:0]		
0x0F33	CM2PCH	7:0						PCH[2:0]		
0x0F34	CM1CON0	7:0	EN	OUT		POL			HYS	SYNC
0x0F35	CM1CON1	7:0							INTP	INTN
0x0F36	CM1NCH	7:0						NCH[2:0]		
0x0F37	CM1PCH	7:0						PCH[2:0]		
0x0F38	CMOUT	7:0							MC2OUT	MC1OUT

32.15 Register Definitions: Comparator Control

Long bit name prefixes for the comparator peripherals are shown in the table below. Refer to the *"Long Bit Names Section"* for more information.

Table 32-2. Comparator Bit Name Prefixes

Peripheral	Bit Name Prefix
C1	C1
C2	C2

Related Links

1.4.2.2 Long Bit Names

32.15.1 CMxCON0

Name: CMxCON0
Address: 0xF34,0xF30

Comparator x Control Register 0

Bit	7	6	5	4	3	2	1	0
	EN	OUT		POL			HYS	SYNC
Access	R/W	RO		R/W			R/W	R/W
Reset	0	0		0			0	0

Bit 7 – EN Comparator Enable bit

Value	Description
1	Comparator is enabled
0	Comparator is disabled and consumes no active power

Bit 6 – OUT Comparator Output bit

Value	Condition	Description
1	If POL = 0 (non-inverted polarity):	CxVP > CxVN
0	If POL = 0 (non-inverted polarity):	CxVP < CxVN
1	If POL = 1 (inverted polarity):	CxVP < CxVN
0	If POL = 1 (inverted polarity):	CxVP > CxVN

Bit 4 – POL Comparator Output Polarity Select bit

Value	Description
1	Comparator output is inverted
0	Comparator output is not inverted

Bit 1 – HYS Comparator Hysteresis Enable bit

Value	Description
1	Comparator hysteresis enabled
0	Comparator hysteresis disabled

Bit 0 – SYNC Comparator Output Synchronous Mode bit
Output updated on the falling edge of prescaled Timer1 clock.

Value	Description
1	Comparator output to Timer1 and I/O pin is synchronous to changes on the prescaled Timer1 clock.
0	Comparator output to Timer1 and I/O pin is asynchronous

32.15.2 CMxCON1

Name: CMxCON1
Address: 0xF35,0xF31

Comparator x Control Register 1

Bit	7	6	5	4	3	2	1	0
							INTP	INTN
Access							R/W	R/W
Reset							0	0

Bit 1 – INTP Comparator Interrupt on Positive-Going Edge Enable bit

Value	Description
1	The CxIF interrupt flag will be set upon a positive-going edge of the CxOUT bit
0	No interrupt flag will be set on a positive-going edge of the CxOUT bit

Bit 0 – INTN Comparator Interrupt on Negative-Going Edge Enable bit

Value	Description
1	The CxIF interrupt flag will be set upon a negative-going edge of the CxOUT bit
0	No interrupt flag will be set on a negative-going edge of the CxOUT bit

32.15.3 CMxNCH

Name: CMxNCH
Address: 0xF36,0xF32

Comparator x Inverting Channel Select Register

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

Bits 2:0 – NCH[2:0] Comparator Inverting Input Channel Select bits

NCH	Negative Input Sources
111	AVSS
110	FVR_Buffer2
101	CxNCH not connected
100	CxNCH not connected
011	CxIN3-
010	CxIN2-
001	CxIN1-
000	CxIN0-

32.15.4 CMxPCH

Name: CMxPCH
Address: 0xF37,0xF33

Comparator x Non-Inverting Channel Select Register

PCH	Positive Input Source
111	AVSS
110	FVR_Buffer2
101	DAC_Output
100	CxPCH not connected
011	CxPCH not connected
010	CxPCH not connected
001	CxIN1+
000	CxIN0+

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

Bits 2:0 – PCH[2:0] Comparator Non-Inverting Input Channel Select bits

32.15.5 CMOUT

Name: CMOUT
Address: 0xF38

Comparator Output Register

Bit	7	6	5	4	3	2	1	0
							MC2OUT	MC1OUT
Access							RO	RO
Reset							0	0

Bits 0, 1 – MCxOUT Mirror copy of CxOUT bit

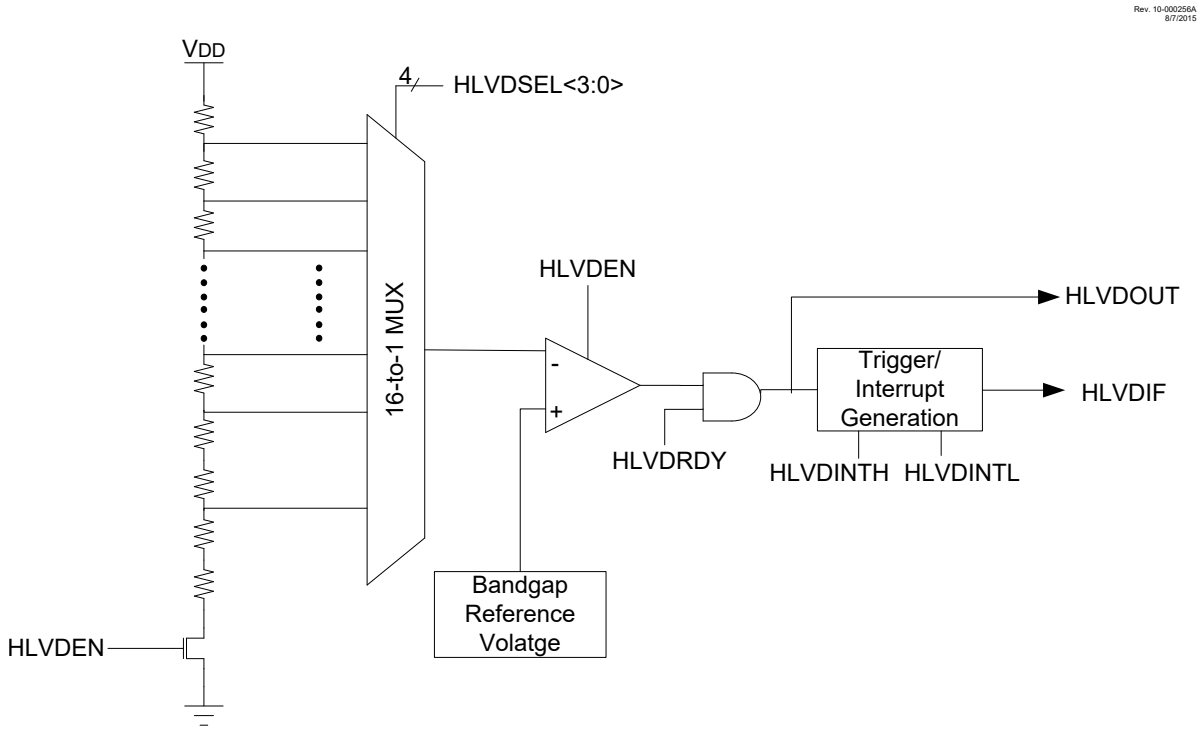
33. (HLVD) High/Low-Voltage Detect

The HLVD module can be configured to monitor the device voltage. This is useful in battery monitoring applications.

Complete control of the HLVD module is provided through the HLVDCON0 and HLVDCON1 registers.

The module's block diagram is shown in the figure below.

Figure 33-1. HLVD Module Block Diagram



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 RDY bit can be used to detect when the module is stable and ready to use.

The INTH and INTL bits determine the overall operation of the module. When INTH is set, the module monitors for rises in V_{DD} above the trip point set by the bits. When INTL is set, the module monitors for drops in V_{DD} below the trip point set by the 33.10.2.1 SEL bits. When both the INTH and INTL bits are set, any changes above or below the trip point set by the 33.10.2.1 SEL bits can be monitored.

The OUT bit can be read to determine if the voltage is greater than or less than the selected trip point.

33.1 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 33.10.2.1 SEL bits.

33.2 Setup

To set up the HLVD module:

1. Select the desired HLVD trip point by writing the value to the SEL bits of the HLVDCON1 register.
2. Depending on the application to detect high-voltage peaks or low-voltage drops or both, set the INTH or INTL bit appropriately.
3. Enable the HLVD module by setting the EN bit.
4. Clear the HLVD interrupt flag (HLVDIF), which may have been set from a previous interrupt.
5. If interrupts are desired, enable the HLVD interrupt by setting the HLVDIE and GIE bits.
   An interrupt will not be generated until the RDY bit is set.



Important: Before changing any module settings (interrupts and tripping point), first disable the module (EN = 0), make the changes and re-enable the module. This prevents the generation of false HLVD events.

Related Links

14.13.4 PIR2

14.13.13 PIE3

33.3 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.

Depending on the application, the HLVD module does not need to operate constantly. To reduce current consumption, the module can be enabled for short periods where the voltage is checked. After such a check, the module could be disabled.

Related Links

38.3.3 Power-Down Current (IPD)(1,2)

33.4 HLVD Start-up Time

If the HLVD or other circuits using the internal voltage reference are disabled to lower the device's current consumption, the reference voltage circuit will require time to become stable before a low or high-voltage condition can be reliably detected. This start-up time, T_{FVRST}, is an interval that is independent of device clock speed. It is specified in electrical specification.

The HLVD interrupt flag is not enabled until T_{FVRST} has expired and a stable reference voltage is reached. For this reason, brief excursions beyond the set point may not be detected during this interval (see the figures below).

Figure 33-2. Low-Voltage Detect Operation (INTL = 1)

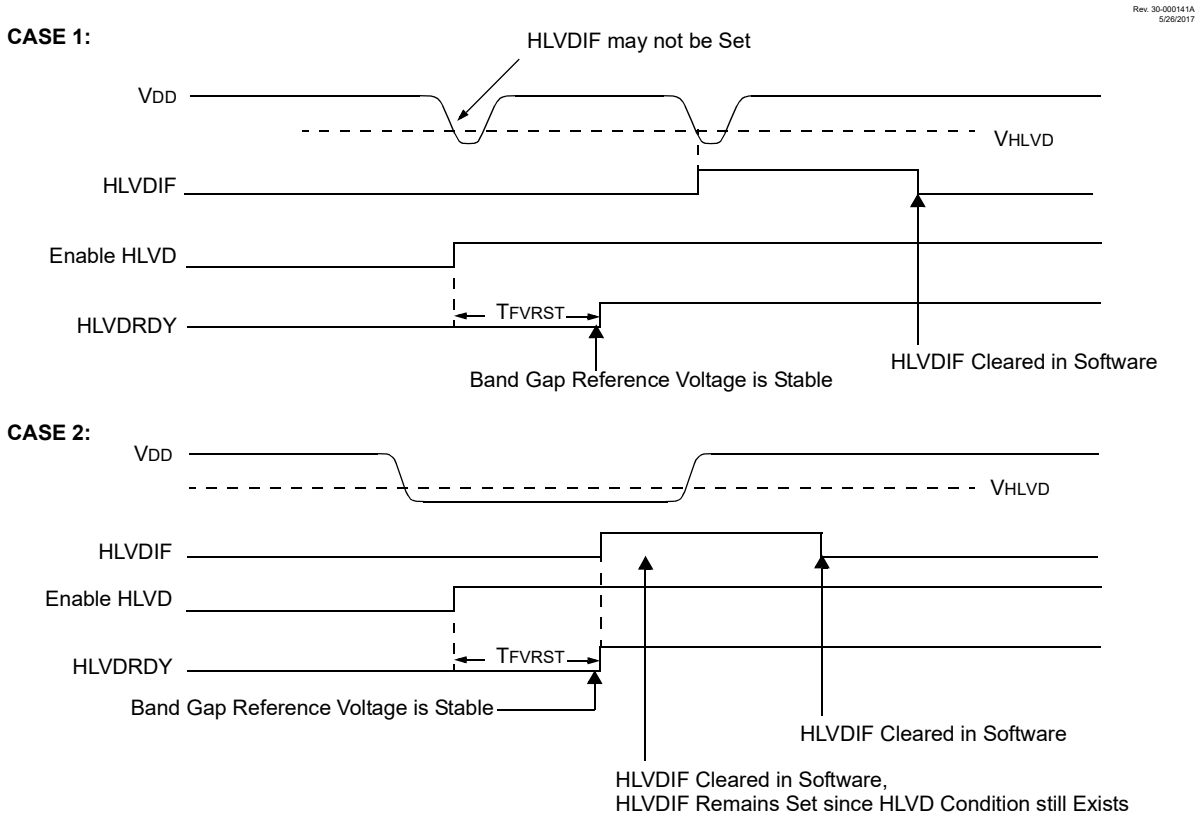
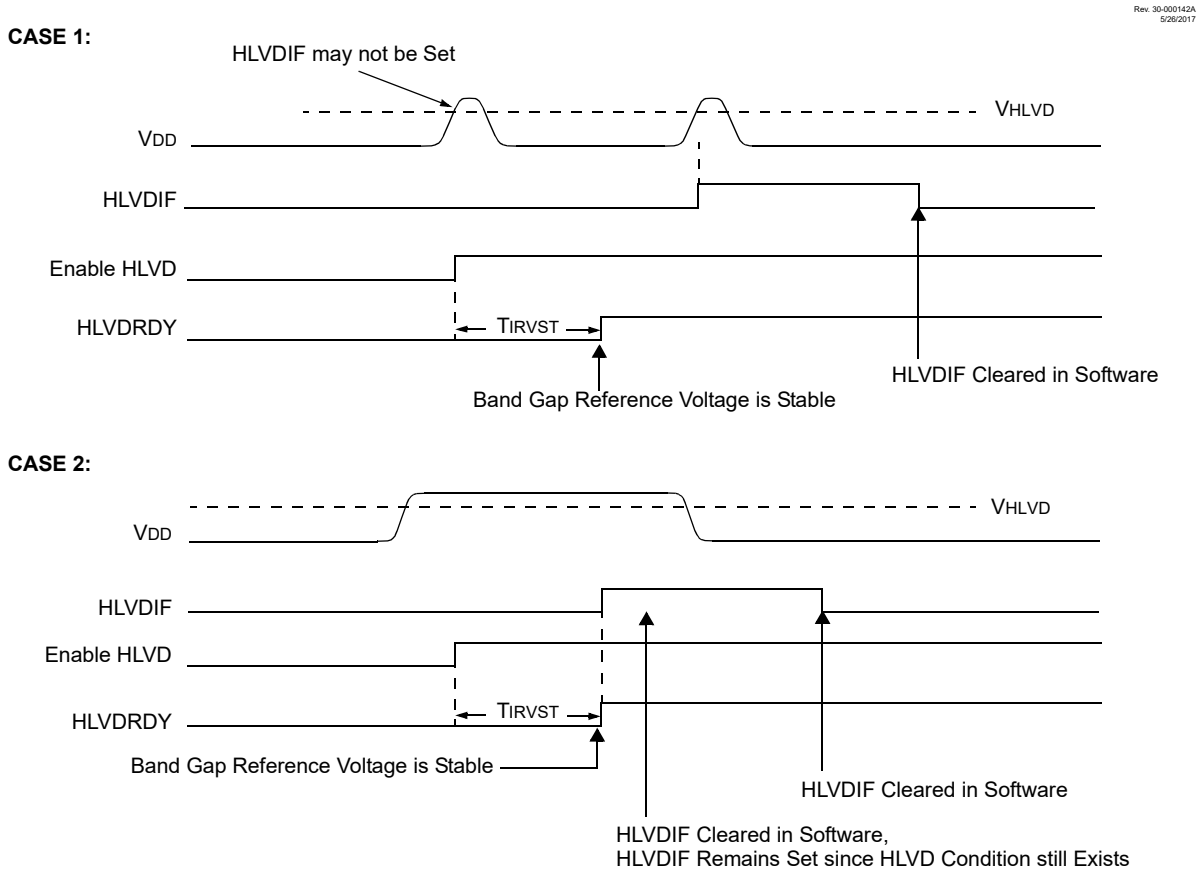


Figure 33-3. High-Voltage Detect Operation (INTH = 1)

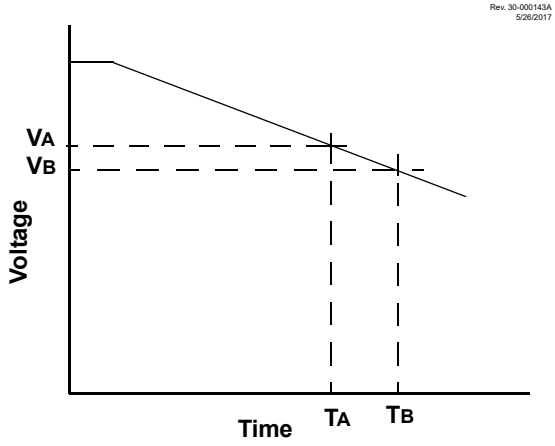


33.5 Applications

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, the figure below shows a possible voltage curve. Over time, the device voltage decreases. When the device voltage reaches voltage, Va, the HLVD logic generates an interrupt at time, Ta. 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 33-4. Typical Low-Voltage Detect Application



Legend: VA = HLVD trip point
VB = Minimum valid device
o per at ing vol tage

33.6 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.

33.7 Operation During Idle and Doze Modes

In both Idle and Doze modes, the module is active and events are generated if peripheral is enabled.

33.8 Effects of a Reset

A device Reset forces all registers to their Reset state. This forces the HLVD module to be turned off.

33.9 Register Summary - HLVD

Address	Name	Bit Pos.								
0x0F2A	HLVDCON0	7:0	EN		OUT	RDY			INTH	INTL
0x0F2B	HLVDCON1	7:0					SEL[3:0]			

33.10 Register Definitions: HLVD Control

Long bit name prefixes for the HLVD peripheral is shown in the following table. Refer to the *"Long Bit Names"* section for more information.

Table 33-1. HLVD Long Bit Name Prefixes

Peripheral	Bit Name Prefix
HLVD	HLVD

Related Links

1.4.2.2 Long Bit Names

33.10.1 HLVDCON0

Name: HLVDCON0
Address: 0xF2A

High/Low-Voltage Detect Control Register 0

Bit	7	6	5	4	3	2	1	0
	EN		OUT	RDY			INTH	INTL
Access	R/W		RO	RO			R/W	R/W
Reset	0		x	x			0	0

Bit 7 – EN High/Low-voltage Detect Power Enable bit

Value	Description
1	Enables the HLVD module
0	Disables the HLVD module

Bit 5 – OUT HLVD Comparator Output bit

Value	Description
1	Voltage ≤ selected detection limit (33.10.2.1 SEL)
0	Voltage ≥ selected detection limit (33.10.2.1 SEL)

Bit 4 – RDY Band Gap Reference Voltages Stable Status Flag bit

Value	Description
1	Indicates HLVD Module is ready and output is stable
0	Indicates HLVD Module is not ready

Bit 1 – INTH HLVD Positive going (High Voltage) Interrupt Enable

Value	Description
1	HLVDIF will be set when voltage ≥ selected detection limit (33.10.2.1 SEL)
0	HLVDIF will not be set

Bit 0 – INTL HLVD Negative going (Low Voltage) Interrupt Enable

Value	Description
1	HLVDIF will be set when voltage ≤ selected detection limit (33.10.2.1 SEL)
0	HLVDIF will not be set

33.10.2 HLVDCON1

Name: HLVDCON1
Address: 0xF2B

Low-Voltage Detect Control Register 1

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

Bits 3:0 – SEL[3:0] High/Low Voltage Detection Limit Selection bits

Table 33-2. HLVD Detection Limits

SEL	Detection Limit
1111	Reserved
1110 to 0000	See High/Low voltage detect characteristics

Reset States: POR/BOR = 0000
All other resets = uuuu

Related Links

38.4.6 High/Low-Voltage Detect Characteristics

34. Register Summary

Address	Name	Bit Pos.								
0x0E9B	PPSLOCK	7:0								PPSLOCKED
0x0E9C	INT0PPS	7:0					PORT	PIN[2:0]		
0x0E9D	INT1PPS	7:0					PORT	PIN[2:0]		
0x0E9E	INT2PPS	7:0					PORT	PIN[2:0]		
0x0E9F	T0CKIPPS	7:0					PORT	PIN[2:0]		
0x0EA0	T1CKIPPS	7:0				PORT[1:0]		PIN[2:0]		
0x0EA1	T1GPPS	7:0				PORT[1:0]		PIN[2:0]		
0x0EA2	T3CKIPPS	7:0				PORT[1:0]		PIN[2:0]		
0x0EA3	T3GPPS	7:0				PORT[1:0]		PIN[2:0]		
0x0EA4	T5CKIPPS	7:0				PORT[1:0]		PIN[2:0]		
0x0EA5	T5GPPS	7:0				PORT[1:0]		PIN[2:0]		
0x0EA6	T2INPPS	7:0				PORT[1:0]		PIN[2:0]		
0x0EA7	T4INPPS	7:0				PORT[1:0]		PIN[2:0]		
0x0EA8	T6INPPS	7:0				PORT[1:0]		PIN[2:0]		
0x0EA9	ADACTPPS	7:0				PORT[1:0]		PIN[2:0]		
0x0EAA	CCP1PPS	7:0				PORT[1:0]		PIN[2:0]		
0x0EAB	CCP2PPS	7:0				PORT[1:0]		PIN[2:0]		
0x0EAC	CWG1PPS	7:0				PORT[1:0]		PIN[2:0]		
0x0EAD	MDCARLPPS	7:0				PORT[1:0]		PIN[2:0]		
0x0EAE	MDCARHPPS	7:0				PORT[1:0]		PIN[2:0]		
0x0EAF	MDSRCPPS	7:0				PORT[1:0]		PIN[2:0]		
0x0EB0	RX1PPS	7:0				PORT[1:0]		PIN[2:0]		
0x0EB1	CK1PPS	7:0				PORT[1:0]		PIN[2:0]		
0x0EB2	SSP1CLKPPS	7:0				PORT[1:0]		PIN[2:0]		
0x0EB3	SSP1DATPPS	7:0				PORT[1:0]		PIN[2:0]		
0x0EB4	SSP1SSPPS	7:0				PORT[1:0]		PIN[2:0]		
0x0EB5	IPR0	7:0			TMR0IP	IOCIP		INT2IP	INT1IP	INT0IP
0x0EB6	IPR1	7:0	OSCFIP	CSWIP					ADTIP	ADIP
0x0EB7	IPR2	7:0	HLVDIP	ZCDIP					C2IP	C1IP
0x0EB8	IPR3	7:0			RC1IP	TX1IP			BCL1IP	SSP1IP
0x0EB9	IPR4	7:0			TMR6IP	TMR5IP	TMR4IP	TMR3IP	TMR2IP	TMR1IP
0x0EBA	IPR5	7:0						TMR5GIP	TMR3GIP	TMR1GIP
0x0EBB	IPR6	7:0							CCP2IP	CCP1IP
0x0EBC	IPR7	7:0	SCANIP	CRCIP	NVMIP					CWG1IP
0x0EBD	PIE0	7:0			TMR0IE	IOCIE		INT2IE	INT1IE	INT0IE
0x0EBE	PIE1	7:0	OSCFIE	CSWIE					ADTIE	ADIE
0x0EBF	PIE2	7:0	HLVDIE	ZCDIE					C2IE	C1IE
0x0EC0	PIE3	7:0			RC1IE	TX1IE			BCL1IE	SSP1IE
0x0EC1	PIE4	7:0			TMR6IE	TMR5IE	TMR4IE	TMR3IE	TMR2IE	TMR1IE
0x0EC2	PIE5	7:0						TMR5GIE	TMR3GIE	TMR1GIE
0x0EC3	PIE6	7:0							CCP2IE	CCP1IE
0x0EC4	PIE7	7:0	SCANIE	CRCIE	NVMIE					CWG1IE
0x0EC5	PIR0	7:0			TMR0IF	IOCIF		INT2IF	INT1IF	INT0IF

Address	Name	Bit Pos.								
0x0EC6	PIR1	7:0	OSCFIF	CSWIF					ADTIF	ADIF
0x0EC7	PIR2	7:0	HLVDIF	ZCDIF					C2IF	C1IF
0x0EC8	PIR3	7:0			RC1IF	TX1IF			BCL1IF	SSP1IF
0x0EC9	PIR4	7:0			TMR6IF	TMR5IF	TMR4IF	TMR3IF	TMR2IF	TMR1IF
0x0ECA	PIR5	7:0						TMR5GIF	TMR3GIF	TMR1GIF
0x0ECB	PIR6	7:0							CCP2IF	CCP1IF
0x0ECC	PIR7	7:0	SCANIF	CRCIF	NVMIF					CWG1IF
0x0ECD	WDTCON0	7:0			WDTPS[4:0]					SEN
0x0ECE	WDTCON1	7:0		WDTCS[2:0]				WINDOW[2:0]		
0x0ECF	WDTPSL	7:0	PSCNTL[7:0]							
0x0ED0	WDTPSH	7:0	PSCNTH[7:0]							
0x0ED1	WDTTMR	7:0	WDTTMR[4:0]					STATE	PSCNT[1:0]	
0x0ED2	CPUDOZE	7:0	IDLEN	DOZEN	ROI	DOE		DOZE[2:0]		
0x0ED3	OSCCON1	7:0		NOSC[2:0]			NDIV[3:0]			
0x0ED4	OSCCON2	7:0		COSC[2:0]			CDIV[3:0]			
0x0ED5	OSCCON3	7:0	CSWHOLD	SOSCPWR		ORDY	NOSCR			
0x0ED6	OSCSTAT	7:0	EXTOR	HFOR	MFOR	LFOR	SOR	ADOR		PLLR
0x0ED7	OSCEN	7:0	EXTOEN	HFOEN	MFOEN	LFOEN	SOSCEN	ADOEN		
0x0ED8	OSCTUNE	7:0			HFTUN[5:0]					
0x0ED9	OSCFRQ	7:0					HFFRQ[3:0]			
0x0EDA	VREGCON	7:0		PMSYS[1:0]					VREGPM[1:0]	
0x0EDB	BORCON	7:0	SBOREN							BORRDY
0x0EDC	PMD0	7:0	SYSCMD	FVRMD	HLVDMD	CRCMD	SCANMD	NVMMD	CLKRMD	IOCMD
0x0EDD	PMD1	7:0		TMR6MD	TMR5MD	TMR4MD	TMR3MD	TMR2MD	TMR1MD	TMR0MD
0x0EDE	PMD2	7:0		DACMD	ADCMD			CMP2MD	CMP1MD	ZCDMD
0x0EDF	PMD3	7:0					PWM4MD	PWM3MD	CCP2MD	CCP1MD
0x0EE0	PMD4	7:0		UART1MD		MSSP1MD				CWG1MD
0x0EE1	PMD5	7:0								DSMMD
0x0EE2	RA0PPS	7:0				PPS[4:0]				
0x0EE3	RA1PPS	7:0				PPS[4:0]				
0x0EE4	RA2PPS	7:0				PPS[4:0]				
0x0EE5	RA3PPS	7:0				PPS[4:0]				
0x0EE6	RA4PPS	7:0				PPS[4:0]				
0x0EE7	RA5PPS	7:0				PPS[4:0]				
0x0EE8	RA6PPS	7:0				PPS[4:0]				
0x0EE9	RA7PPS	7:0				PPS[4:0]				
0x0EEA	RB0PPS	7:0				PPS[4:0]				
0x0EEB	RB1PPS	7:0				PPS[4:0]				
0x0EEC	RB2PPS	7:0				PPS[4:0]				
0x0EED	RB3PPS	7:0				PPS[4:0]				
0x0EEE	RB4PPS	7:0				PPS[4:0]				
0x0EEF	RB5PPS	7:0				PPS[4:0]				
0x0EF0	RB6PPS	7:0				PPS[4:0]				
0x0EF1	RB7PPS	7:0				PPS[4:0]				
0x0EF2	RC0PPS	7:0				PPS[4:0]				
0x0EF3	RC1PPS	7:0				PPS[4:0]				

Address	Name	Bit Pos.								
0x0EF4	RC2PPS	7:0				PPS[4:0]				
0x0EF5	RC3PPS	7:0				PPS[4:0]				
0x0EF6	RC4PPS	7:0				PPS[4:0]				
0x0EF7	RC5PPS	7:0				PPS[4:0]				
0x0EF8	RC6PPS	7:0				PPS[4:0]				
0x0EF9	RC7PPS	7:0				PPS[4:0]				
0x0EFA ... 0x0F04	Reserved									
0x0F05	IOCAF	7:0	IOCAF7	IOCAF6	IOCAF5	IOCAF4	IOCAF3	IOCAF2	IOCAF1	IOCAF0
0x0F06	IOCAN	7:0	IOCAN7	IOCAN6	IOCAN5	IOCAN4	IOCAN3	IOCAN2	IOCAN1	IOCAN0
0x0F07	IOCAP	7:0	IOCAP7	IOCAP6	IOCAP5	IOCAP4	IOCAP3	IOCAP2	IOCAP1	IOCAP0
0x0F08	INLVLA	7:0	INLVLA7	INLVLA6	INLVLA5	INLVLA4	INLVLA3	INLVLA2	INLVLA1	INLVLA0
0x0F09	SLRCONA	7:0	SLRA7	SLRA6	SLRA5	SLRA4	SLRA3	SLRA2	SLRA1	SLRA0
0x0F0A	ODCONA	7:0	ODCA7	ODCA6	ODCA5	ODCA4	ODCA3	ODCA2	ODCA1	ODCA0
0x0F0B	WPUA	7:0	WPUA7	WPUA6	WPUA5	WPUA4	WPUA3	WPUA2	WPUA1	WPUA0
0x0F0C	ANSELA	7:0	ANSELA7	ANSELA6	ANSELA5	ANSELA4	ANSELA3	ANSELA2	ANSELA1	ANSELA0
0x0F0D	IOCBF	7:0	IOCBF7	IOCBF6	IOCBF5	IOCBF4	IOCBF3	IOCBF2	IOCBF1	IOCBF0
0x0F0E	IOCBN	7:0	IOCBN7	IOCBN6	IOCBN5	IOCBN4	IOCBN3	IOCBN2	IOCBN1	IOCBN0
0x0F0F	IOCBP	7:0	IOCBP7	IOCBP6	IOCBP5	IOCBP4	IOCBP3	IOCBP2	IOCBP1	IOCBP0
0x0F10	INLVLB	7:0	INLVLB7	INLVLB6	INLVLB5	INLVLB4	INLVLB3	INLVLB2	INLVLB1	INLVLB0
0x0F11	SLRCONB	7:0	SLRB7	SLRB6	SLRB5	SLRB4	SLRB3	SLRB2	SLRB1	SLRB0
0x0F12	ODCONB	7:0	ODCB7	ODCB6	ODCB5	ODCB4	ODCB3	ODCB2	ODCB1	ODCB0
0x0F13	WPUB	7:0	WPUB7	WPUB6	WPUB5	WPUB4	WPUB3	WPUB2	WPUB1	WPUB0
0x0F14	ANSELB	7:0	ANSELB7	ANSELB6	ANSELB5	ANSELB4	ANSELB3	ANSELB2	ANSELB1	ANSELB0
0x0F15	IOCCF	7:0	IOCCF7	IOCCF6	IOCCF5	IOCCF4	IOCCF3	IOCCF2	IOCCF1	IOCCF0
0x0F16	IOCCN	7:0	IOCCN7	IOCCN6	IOCCN5	IOCCN4	IOCCN3	IOCCN2	IOCCN1	IOCCN0
0x0F17	IOCCP	7:0	IOCCP7	IOCCP6	IOCCP5	IOCCP4	IOCCP3	IOCCP2	IOCCP1	IOCCP0
0x0F18	INLVLC	7:0	INLVLC7	INLVLC6	INLVLC5	INLVLC4	INLVLC3	INLVLC2	INLVLC1	INLVLC0
0x0F19	SLRCONC	7:0	SLRC7	SLRC6	SLRC5	SLRC4	SLRC3	SLRC2	SLRC1	SLRC0
0x0F1A	ODCONC	7:0	ODCC7	ODCC6	ODCC5	ODCC4	ODCC3	ODCC2	ODCC1	ODCC0
0x0F1B	WPUC	7:0	WPUC7	WPUC6	WPUC5	WPUC4	WPUC3	WPUC2	WPUC1	WPUC0
0x0F1C	ANSELC	7:0	ANSELC7	ANSELC6	ANSELC5	ANSELC4	ANSELC3	ANSELC2	ANSELC1	ANSELC0
0x0F1D ... 0x0F21	Reserved									
0x0F22	IOCEF	7:0					IOCEF3			
0x0F23	IOCEN	7:0					IOCEN3			
0x0F24	IOCEP	7:0					IOCEP3			
0x0F25	INLVLE	7:0					INLVLE3			
0x0F26 ... 0x0F27	Reserved									
0x0F28	WPUE	7:0					WPUE3			
0x0F29	Reserved									
0x0F2A	HLVDCON0	7:0	EN		OUT	RDY			INTH	INTL

Address	Name	Bit Pos.								
0x0F2B	HLVDCON1	7:0					SEL[3:0]			
0x0F2C	FVRCON	7:0	FVREN	FVRRDY	TSEN	TSRNG	CDAFVR[1:0]		ADFVR[1:0]	
0x0F2D	ZCDCON	7:0	SEN		OUT	POL			INTP	INTN
0x0F2E	DAC1CON0	7:0	EN		OE1	OE2	PSS[1:0]			NSS
0x0F2F	DAC1CON1	7:0				DAC1R[4:0]				
0x0F30	CM2CON0	7:0	EN	OUT		POL			HYS	SYNC
0x0F31	CM2CON1	7:0							INTP	INTN
0x0F32	CM2NCH	7:0						NCH[2:0]		
0x0F33	CM2PCH	7:0						PCH[2:0]		
0x0F34	CM1CON0	7:0	EN	OUT		POL			HYS	SYNC
0x0F35	CM1CON1	7:0							INTP	INTN
0x0F36	CM1NCH	7:0						NCH[2:0]		
0x0F37	CM1PCH	7:0						PCH[2:0]		
0x0F38	CMOUT	7:0							MC2OUT	MC1OUT
0x0F39	CLKRCON	7:0	EN			DC[1:0]		DIV[2:0]		
0x0F3A	CLKRCLK	7:0						CLK[2:0]		
0x0F3B	CWG1CLK	7:0								CS
0x0F3C	CWG1ISM	7:0						ISM[2:0]		
0x0F3D	CWG1DBR	7:0			DBR[5:0]					
0x0F3E	CWG1DBF	7:0			DBF[5:0]					
0x0F3F	CWG1CON0	7:0	EN	LD				MODE[2:0]		
0x0F40	CWG1CON1	7:0			IN		POLD	POLC	POLB	POLA
0x0F41	CWG1AS0	7:0	SHUTDOWN	REN	LSBD[1:0]		LSAC[1:0]			
0x0F42	CWG1AS1	7:0			AS5E	AS4E	AS3E	AS2E	AS1E	AS0E
0x0F43	CWG1STR	7:0	OVRD	OVRC	OVRB	OVRA	STRD	STRC	STRB	STRA
0x0F44	SCANLADR	7:0	SCANLADRL[7:0]							
		15:8	SCANLADRH[7:0]							
		23:16			SCANLADRU[5:0]					
0x0F47	SCANHADR	7:0	SCANHADRL[7:0]							
		15:8	SCANHADRH[7:0]							
		23:16			SCANHADRU[5:0]					
0x0F4A	SCANCON0	7:0	SCANEN	SCANGO	BUSY	INVALID	INTM		MODE[1:0]	
0x0F4B	SCANTRIG	7:0					TSEL[3:0]			
0x0F4C	MDCON0	7:0	EN		OUT	OPOL				BIT
0x0F4D	MDCON1	7:0			CHPOL	CHSYNC			CLPOL	CLSYNC
0x0F4E	MDSRC	7:0					SRCS[3:0]			
0x0F4F	MDCARL	7:0						CLS[2:0]		
0x0F50	MDCARH	7:0						CHS[2:0]		
0x0F51	ADACT	7:0				ADACT[4:0]				
0x0F52	ADCLK	7:0			ADCS[5:0]					
0x0F53	ADREF	7:0				ADNREF			ADPREF[1:0]	
0x0F54	ADCON1	7:0	ADPPOL	ADIPEN	ADGPOL					ADDSEN
0x0F55	ADCON2	7:0	ADPSIS	ADCRS[2:0]			ADACLR	ADMD[2:0]		
0x0F56	ADCON3	7:0		ADCALC[2:0]			ADSOI	ADTMD[2:0]		
0x0F57	ADACQ	7:0	ADACQ[7:0]							
0x0F58	ADCAP	7:0				ADCAP[4:0]				

Address	Name	Bit Pos.								
0x0F59	ADPRE	7:0	ADPRE[7:0]							
0x0F5A	ADPCH	7:0			ADPCH[5:0]					
0x0F5B	ADCON0	7:0	ADON	ADCONT		ADCS		ADFM		ADGO
0x0F5C	ADPREV	7:0	ADPREVL[7:0]							
		15:8	ADPREVH[7:0]							
0x0F5E	ADRES	7:0	ADRESL[7:0]							
		15:8	ADRESH[7:0]							
0x0F60	ADSTAT	7:0	ADAOV	ADUTHR	ADLTHR	ADMATH		ADSTAT[2:0]		
0x0F61	ADRPT	7:0	ADRPT[7:0]							
0x0F62	ADCNT	7:0	ADCNT[7:0]							
0x0F63	ADSTPT	7:0	ADSTPTL[7:0]							
		15:8	ADSTPTH[7:0]							
0x0F65	ADLTH	7:0	ADLTHL[7:0]							
		15:8	ADLTHH[7:0]							
0x0F67	ADUTH	7:0	ADUTHL[7:0]							
		15:8	ADUTHH[7:0]							
0x0F69	ADERR	7:0	ADERRL[7:0]							
		15:8	ADERRH[7:0]							
0x0F6B	ADACC	7:0	ADACCL[7:0]							
		15:8	ADACCH[7:0]							
0x0F6D	ADFLTR	7:0	ADFLTRL[7:0]							
		15:8	ADFLTRH[7:0]							
0x0F6F	CRCDAT	7:0	CRCDATL[7:0]							
		15:8	CRCDATH[7:0]							
0x0F71	CRCACC	7:0	CRCACCL[7:0]							
		15:8	CRCACCH[7:0]							
0x0F73	CRCSHIFT	7:0	CRCSHIFTL[7:0]							
		15:8	CRCSHIFTH[7:0]							
0x0F75	CRCXOR	7:0	CRCXORL[6:0]							CRCXORL0
		15:8	CRCXORH[7:0]							
0x0F77	CRCCON0	7:0	EN	GO	BUSY	ACCM			SHIFTM	FULL
0x0F78	CRCCON1	7:0	DLEN[3:0]				PLEN[3:0]			
0x0F79	NVMADR	7:0	NVMADRL[7:0]							
		15:8	NVMADRH[7:0]							
		23:16			NVMADRU[5:0]					
0x0F7C	NVMDAT	7:0	NVMDATL[7:0]							
		15:8	NVMDATH[7:0]							
0x0F7E	Reserved									
0x0F7F	NVMCON0	7:0	NVMEN			NVMERR	Reserved			
0x0F80	NVMCON1	7:0		SECER	SECWR	WR			SECRD	RD
0x0F81	NVMCON2	7:0	NVMCON2[7:0]							
0x0F82	LATA	7:0	LATA7	LATA6	LATA5	LATA4	LATA3	LATA2	LATA1	LATA0
0x0F83	LATB	7:0	LATB7	LATB6	LATB5	LATB4	LATB3	LATB2	LATB1	LATB0
0x0F84	LATC	7:0	LATC7	LATC6	LATC5	LATC4	LATC3	LATC2	LATC1	LATC0
0x0F85 ...	Reserved									

Address	Name	Bit Pos.								
0x0F86										
0x0F87	TRISA	7:0	TRISA7	TRISA6	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0
0x0F88	TRISB	7:0	TRISB7	TRISB6	TRISB5	TRISB4	TRISB3	TRISB2	TRISB1	TRISB0
0x0F89	TRISC	7:0	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0
0x0F8A ... 0x0F8B	Reserved									
0x0F8C	PORTA	7:0	RA7	RA6	RA5	RA4	RA3	RA2	RA1	RA0
0x0F8D	PORTB	7:0	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0
0x0F8E	PORTC	7:0	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0
0x0F8F	Reserved									
0x0F90	PORTE	7:0					RE3			
0x0F91	SSP1BUF	7:0	BUF[7:0]							
0x0F92	SSP1ADD	7:0	ADD[7:0]							
0x0F93	SSP1MSK	7:0	MSK[6:0]							MSK0
0x0F94	SSP1STAT	7:0	SMP	CKE	$\overline{D/A}$	P	S	$\overline{R/W}$	UA	BF
0x0F95	SSP1CON1	7:0	WCOL	SSPOV	SSPEN	CKP	SSPM[3:0]			
0x0F96	SSP1CON2	7:0	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN
0x0F97	SSP1CON3	7:0	ACKTIM	PCIE	SCIE	BOEN	SDAHT	SBCDE	AHEN	DHEN
0x0F98	RC1REG	7:0	RCREG[7:0]							
0x0F99	TX1REG	7:0	TXREG[7:0]							
0x0F9A	SP1BRG	7:0	SPBRGL[7:0]							
		15:8	SPBRGH[7:0]							
0x0F9C	RC1STA	7:0	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
0x0F9D	TX1STA	7:0	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D
0x0F9E	BAUD1CON	7:0	ABDOVF	RCIDL		SCKP	BRG16		WUE	ABDEN
0x0F9F	PWM4DC	7:0	DCL[1:0]							
		15:8	DCH[7:0]							
0x0FA1	PWM4CON	7:0	EN		OUT	POL				
0x0FA2	PWM3DC	7:0	DCL[1:0]							
		15:8	DCH[7:0]							
0x0FA4	PWM3CON	7:0	EN		OUT	POL				
0x0FA5	CCPR2	7:0	CCPRL[7:0]							
		15:8	CCPRH[7:0]							
0x0FA7	CCP2CON	7:0	EN		OUT	FMT	MODE[3:0]			
0x0FA8	CCP2CAP	7:0							CTS[1:0]	
0x0FA9	CCPR1	7:0	CCPRL[7:0]							
		15:8	CCPRH[7:0]							
0x0FAB	CCP1CON	7:0	EN		OUT	FMT	MODE[3:0]			
0x0FAC	CCP1CAP	7:0							CTS[1:0]	
0x0FAD	CCPTMRS	7:0	P4TSEL[1:0]		P3TSEL[1:0]		C2TSEL[1:0]		C1TSEL[1:0]	
0x0FAD	CCPTMRS	7:0	P4TSEL[1:0]		P3TSEL[1:0]		C2TSEL[1:0]		C1TSEL[1:0]	
0x0FAE	T6TMR	7:0	TxTMR[7:0]							
0x0FAF	T6PR	7:0	TxPR[7:0]							
0x0FB0	T6CON	7:0	ON		CKPS[2:0]		OUTPS[3:0]			
0x0FB1	T6HLT	7:0	PSYNC	CPOL	CSYNC	MODE[4:0]				

Address	Name	Bit Pos.								
0x0FB2	T6CLKCON	7:0					CS[3:0]			
0x0FB3	T6RST	7:0					RSEL[3:0]			
0x0FB4	T4TMR	7:0	TxTMR[7:0]							
0x0FB5	T4PR	7:0	TxPR[7:0]							
0x0FB6	T4CON	7:0	ON		CKPS[2:0]		OUTPS[3:0]			
0x0FB7	T4HLT	7:0	PSYNC	CPOL	CSYNC	MODE[4:0]				
0x0FB8	T4CLKCON	7:0					CS[3:0]			
0x0FB9	T4RST	7:0					RSEL[3:0]			
0x0FBA	T2TMR	7:0	TxTMR[7:0]							
0x0FBB	T2PR	7:0	TxPR[7:0]							
0x0FBC	T2CON	7:0	ON		CKPS[2:0]		OUTPS[3:0]			
0x0FBD	T2HLT	7:0	PSYNC	CPOL	CSYNC	MODE[4:0]				
0x0FBE	T2CLKCON	7:0					CS[3:0]			
0x0FBF	T2RST	7:0					RSEL[3:0]			
0x0FC0	TMR5	7:0	TMRxL[7:0]							
		15:8	TMRxH[7:0]							
0x0FC2	T5CON	7:0			CKPS[1:0]			$\overline{\text{SYNC}}$	RD16	ON
0x0FC3	T5GCON	7:0	GE	GPOL	GTM	GSPM	GGO/DONE	GVAL		
0x0FC4	TMR5GATE	7:0					GSS[3:0]			
0x0FC5	TMR5CLK	7:0					CS[3:0]			
0x0FC6	TMR3	7:0	TMRxL[7:0]							
		15:8	TMRxH[7:0]							
0x0FC8	T3CON	7:0			CKPS[1:0]			$\overline{\text{SYNC}}$	RD16	ON
0x0FC9	T3GCON	7:0	GE	GPOL	GTM	GSPM	GGO/DONE	GVAL		
0x0FCA	TMR3GATE	7:0					GSS[3:0]			
0x0FCB	TMR3CLK	7:0					CS[3:0]			
0x0FCC	TMR1	7:0	TMRxL[7:0]							
		15:8	TMRxH[7:0]							
0x0FCE	T1CON	7:0			CKPS[1:0]			$\overline{\text{SYNC}}$	RD16	ON
0x0FCF	T1GCON	7:0	GE	GPOL	GTM	GSPM	GGO/DONE	GVAL		
0x0FD0	TMR1GATE	7:0					GSS[3:0]			
0x0FD1	TMR1CLK	7:0					CS[3:0]			
0x0FD2	TMR0L	7:0	TMR0L[7:0]							
0x0FD3	TMR0H	7:0	TMR0H[7:0]							
0x0FD4	T0CON0	7:0	T0EN		T0OUT	T016BIT	T0OUTPS[3:0]			
0x0FD5	T0CON1	7:0	T0CS[2:0]			T0ASYNC	T0CKPS[3:0]			
0x0FD6	PCON1	7:0						$\overline{\text{RVREG}}$		$\overline{\text{RCM}}$
0x0FD7	PCON0	7:0	STKOVF	STKUNF	$\overline{\text{WDTWV}}$	$\overline{\text{RWDT}}$	$\overline{\text{RMCLR}}$	$\overline{\text{RI}}$	$\overline{\text{POR}}$	$\overline{\text{BOR}}$
0x0FD8	STATUS	7:0		$\overline{\text{TO}}$	$\overline{\text{PD}}$	N	OV	Z	DC	C
0x0FD9	FSR2	7:0	FSRL[7:0]							
		15:8					FSRH[3:0]			
0x0FDB	PLUSW2	7:0	PLUSW[7:0]							
0x0FDC	PREINC2	7:0	PREINC[7:0]							
0x0FDD	POSTDEC2	7:0	POSTDEC[7:0]							
0x0FDE	POSTINC2	7:0	POSTINC[7:0]							
0x0FDF	INDF2	7:0	INDF[7:0]							

Address	Name	Bit Pos.								
0x0FE0	BSR	7:0					BSR[3:0]			
0x0FE1	FSR1	7:0	FSRL[7:0]							
		15:8					FSRH[3:0]			
0x0FE3	PLUSW1	7:0	PLUSW[7:0]							
0x0FE4	PREINC1	7:0	PREINC[7:0]							
0x0FE5	POSTDEC1	7:0	POSTDEC[7:0]							
0x0FE6	POSTINC1	7:0	POSTINC[7:0]							
0x0FE7	INDF1	7:0	INDF[7:0]							
0x0FE8	WREG	7:0	WREG[7:0]							
0x0FE9	FSR0	7:0	FSRL[7:0]							
		15:8					FSRH[3:0]			
0x0FEB	PLUSW0	7:0	PLUSW[7:0]							
0x0FEC	PREINC0	7:0	PREINC[7:0]							
0x0FED	POSTDEC0	7:0	POSTDEC[7:0]							
0x0FEE	POSTINC0	7:0	POSTINC[7:0]							
0x0FEF	INDF0	7:0	INDF[7:0]							
0x0FF0 ... 0x0FF1	Reserved									
0x0FF2	INTCON	7:0	GIE/GIEH	PEIE/GIEL	IPEN			INT2EDG	INT1EDG	INT0EDG
0x0FF3	PROD	7:0	PRODL[7:0]							
		15:8	PRODH[7:0]							
0x0FF5	TABLAT	7:0	TABLAT[7:0]							
0x0FF6	TBLPTR	7:0	TBLPTRL[7:0]							
		15:8	TBLPTRH[7:0]							
		23:16			TBLPTR21	TBLPTRU[4:0]				
0x0FF9	PCL	7:0	PCL[7:0]							
0x0FFA	PCLAT	7:0	PCLATH[7:0]							
		15:8				PCLATU[4:0]				
0x0FFC	STKPTR	7:0				STKPTR[4:0]				
0x0FFD	TOS	7:0	TOSL[7:0]							
		15:8	TOSH[7:0]							
		23:16				TOSU[4:0]				

35. 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
- $\overline{MCLR}/V_{PP}$
- V_{DD}
- V_{SS}

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 " Memory Programming Specification" (DS40001874).

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 $\overline{MCLR}/V_{PP}$ to V_{IHH}.

35.2 Low-Voltage Programming Entry Mode

The Low-Voltage Programming Entry mode allows the PIC® Flash MCUs to be programmed using V_{DD} only, without high voltage. When the LVP 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. $\overline{MCLR}$ is brought to Vil.
2. A 32-bit key sequence is presented on ICSPDAT, while clocking ICSPCLK.

Once the key sequence is complete, $\overline{MCLR}$ must be held at V_{IL} for as long as Program/Verify mode is to be maintained.

If low-voltage programming is enabled (LVP = 1), the $\overline{MCLR}$ Reset function is automatically enabled and cannot be disabled. See the $\overline{MCLR}$ Section for more information.

The LVP bit can only be reprogrammed to '0' by using the High-Voltage Programming mode.

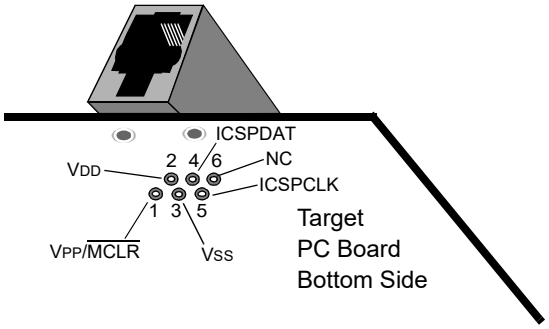
Related Links

8.4 MCLR Reset

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



Pin Description*

1 = $V_{PP}/\overline{MCLR}$

2 = V_{DD} Target

3 = V_{SS} (ground)

4 = ICSPDAT

5 = ICSPCLK

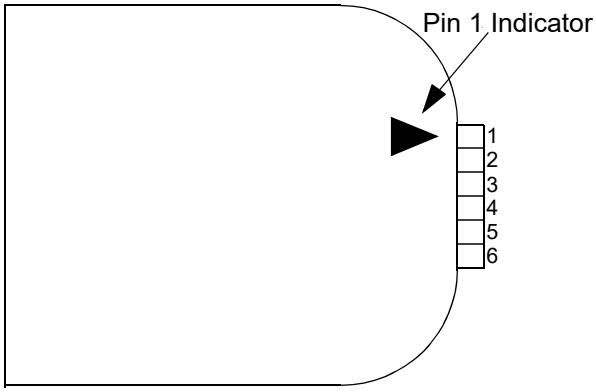
6 = No Connect

Another connector often found in use with the PICkit™ programmers is a standard 6-pin header with 0.1 inch spacing. Refer to Figure 35-2.

For additional interface recommendations, refer to the specific device programmer manual prior to PCB design.

It is recommended that isolation devices be used to separate the programming pins from other circuitry. The type of isolation is highly dependent on the specific application and may include devices such as resistors, diodes, or even jumpers. See Figure 35-3 for more information.

Figure 35-2. PICkit™ Programmer Style Connector Interface



Pin Description[1]

1 = $V_{PP}/\overline{MCLR}$

2 = V_{DD} Target

3 = V_{SS} (ground)

4 = ICSPDAT

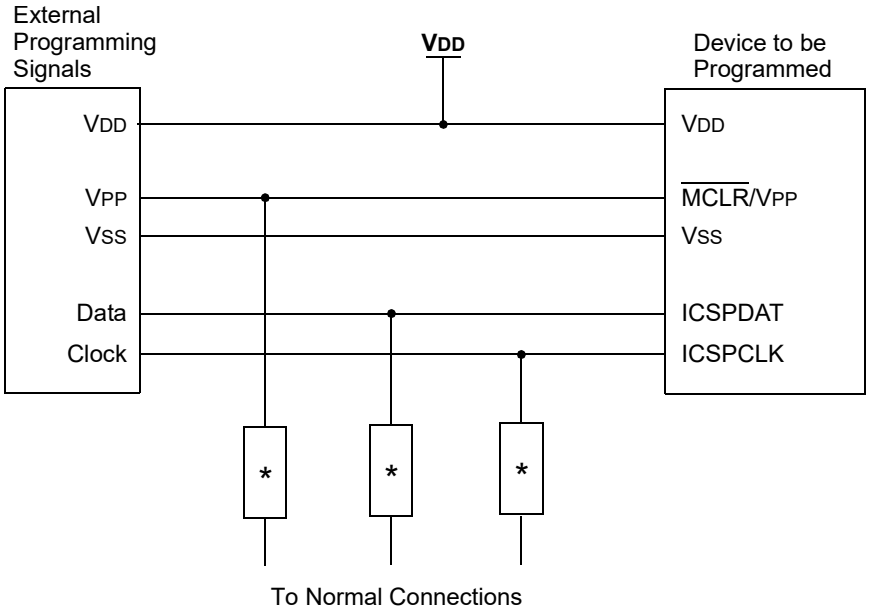
5 = ICSPCLK

6 = No Connect

Note:

1. **Note:** The 6-pin header (0.100" spacing) accepts 0.025" square pins.

Figure 35-3. Typical Connection for ICSP™ Programming



* Isolation devices (as required).

36. Instruction Set Summary

PIC18F24/25Q10 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 µs. If a conditional test is true, or the Program Counter is changed as a result of an instruction, the instruction execution time is 2 µs. Two-word branch instructions (if true) would take 3 µs.

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™).

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.

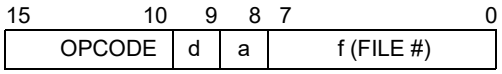
Field	Description
f_d	12-bit Register file address (000h to FFFh). This is the destination address.
GIE	Global Interrupt Enable bit.
k	Literal field, constant data or label (may be either an 8-bit, 12-bit or a 20-bit value).
label	Label name.
mm	The mode of the TBLPTR register for the table read and 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.

Field	Description
TO	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.
italics	User defined term (font is Courier).

Figure 36-1. General Format for Instructions

Byte-oriented file register operations

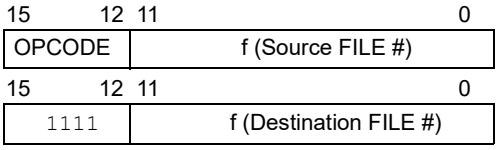
Example Instruction




```
ADDWF MYREG, W, B
```

d = 0 for result destination to be WREG register
d = 1 for result destination to be file register (f)
a = 0 to force Access Bank
a = 1 for BSR to select bank
f = 8-bit file register address

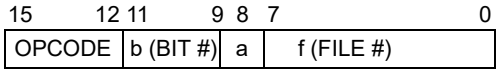
Byte to Byte move operations (2-word)




```
MOVFF MYREG1, MYREG2
```

f = 12-bit file register address

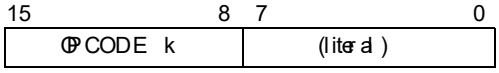
Bit-oriented file register operations




```
BSF MYREG, bit, B
```

b = 3-bit position of bit in file register (f)
a = 0 to force Access Bank
a = 1 for BSR to select bank
f = 8-bit file register address

Literal operations

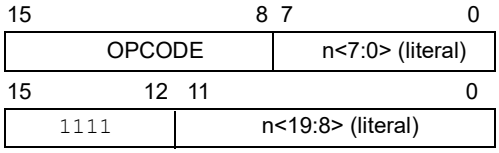



```
MOVLW 7Fh
```

k = 8-bit immediate value

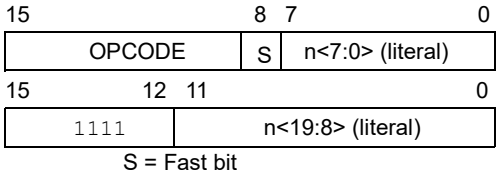
Control operations

CALL, GOTO and Branch operations



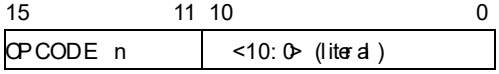

```
GOTO Label
```

n = 20-bit immediate value

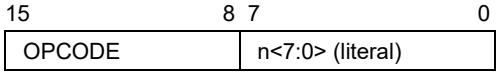



```
CALL MYFUNC
```

S = Fast bit




```
BRA MYFUNC
```




```
BC MYFUNC
```

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	11da	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	2	1100	ffff	ffff	ffff	None	

Mnemonic, Operands		Description	Cycles	16-Bit Instruction Word				Status Affected	Notes
				MSb			LSb		
		f_d (destination) 2nd word		1111	ffff	ffff	ffff		
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	00da	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	
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

Mnemonic, Operands		Description	Cycles	16-Bit Instruction Word				Status Affected	Notes
				MSb			LSb		
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		

Mnemonic, Operands		Description	Cycles	16-Bit Instruction Word				Status Affected	Notes
				MSb			LSb		
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	$\overline{TO}$, $\overline{PD}$	
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	2	1110	1110	00ff	kkkk	None	
		to FSR(f) 1st word		1111	0000	kkkk	kkkk		
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	

Mnemonic, Operands		Description	Cycles	16-Bit Instruction Word				Status Affected	Notes
				MSb			LSb		
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 ≤ k ≤ 255			
Operation:	(W) + k → W			
Status Affected:	N, OV, C, DC, Z			
Encoding:	0000	1111	kkkk	kkkk
Description:	The contents of W are added to the 8-bit literal ‘k’ and the result is placed in W.			
Words:	1			
Cycles:	1			

Q Cycle Activity:			
Q1	Q2	Q3	Q4
Decode	Read literal ‘k’	Process Data	Write to W

Example:	ADDLW	15h
Before Instruction W = 10h After Instruction W = 25h		

ADDWF	ADD W to f			
Syntax:	ADDWF f {,d {,a}}			
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]			
Operation:	(W) + (f) → dest			
Status Affected:	N, OV, C, DC, Z			
Encoding:	0010	01da	ffff	ffff
Description:	Add W to register ‘f’. If ‘d’ is ‘0’, the result is stored in W. If ‘d’ is ‘1’, the result is stored back in register ‘f’ (default). If ‘a’ is ‘0’, the Access Bank is selected. If ‘a’ is ‘1’, the BSR is used to select the GPR bank.			

ADDWF	ADD W to f			
	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 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			

Important: All PIC18 instructions may take an optional label argument preceding the instruction mnemonic for use in symbolic addressing. If a label is used, the instruction format then becomes: {label} instruction argument(s).

ADDWFC	ADD W and CARRY bit to f			
Syntax:	ADDWFC f {,d {,a}}			
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [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’.			

ADDWFC	ADD W and CARRY bit to 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 ≤ 95 (5Fh).See 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 ≤ k ≤ 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 ≤ f ≤ 255 d ∈ [0,1] a ∈ [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 ≤ 95 (5Fh). See 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 ≤ n ≤ 127			
Operation:	if CARRY bit is ‘1’ (PC) + 2 + 2n → PC			
Status Affected:	None			
Encoding:	1110	0010	nnnn	nnnn
Description:	If the CARRY bit is ‘1’, then the program will branch. The 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)			

BCF	Bit Clear f			
Syntax:	BCF f, b {,a}			
Operands:	0 ≤ f ≤ 255 0 ≤ b ≤ 7 a ∈ [0,1]			
Operation:	0 → f<b>			
Status Affected:	None			
Encoding:	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 ≤ 95 (5Fh). See 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:	BCF	FLAG_REG, 7, 0
Before Instruction FLAG_REG = C7h After Instruction		

FLAG_REG = 47h

BN	Branch if Negative			
Syntax:	BN n			
Operands:	-128 ≤ n ≤ 127			
Operation:	if NEGATIVE bit is ‘1’ (PC) + 2 + 2n → PC			
Status Affected:	None			
Encoding:	1110	0110	nnnn	nnnn
Description:	If the NEGATIVE bit is ‘1’, then the program will branch. The 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	BN	Jump
Before Instruction PC = address (HERE) After Instruction If NEGATIVE = 1; PC = address (Jump)			

If NEGATIVE = 0; PC = address (HERE + 2)

BNC	Branch if Not Carry			
Syntax:	BNC n			
Operands:	-128 ≤ n ≤ 127			
Operation:	if 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 ≤ n ≤ 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)

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 ≤ n ≤ 1023			
Operation:	(PC) + 2 + 2n → 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 ≤ f ≤ 255 0 ≤ b ≤ 7 a ∈ [0,1]			
Operation:	1 → f<b>			
Status Affected:	None			
Encoding:	1000	bbba	ffff	ffff

BSF	Bit Set f			
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 ≤ 95 (5Fh). See 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		

BTFSC	Bit Test File, Skip if Clear			
Syntax:	BTFSC f, b {,a}			
Operands:	0 ≤ f ≤ 255 0 ≤ b ≤ 7 a ∈ [0,1]			
Operation:	skip if (f<b>) = 0			
Status Affected:	None			
Encoding:	1011	bbba	ffff	ffff
Description:	If bit ‘b’ in register ‘f’ is ‘0’, then the next instruction is skipped. If bit ‘b’ is ‘0’, then the next instruction fetched during the current instruction execution is discarded and a NOP is executed instead, making this a 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			

BTFSC	Bit Test File, Skip if Clear			
	mode whenever f ≤ 95 (5Fh). See 36.2.3 Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode for details.			
Words:	1			
Cycles:	1(2) Note: Three 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 FALSE TRUE	BTFSC : :	FLAG, 1, 0
Before Instruction PC = address (HERE) After Instruction If FLAG<1> = 0; PC = address (TRUE) If FLAG<1> = 1; PC = address (FALSE)			

BTFSS	Bit Test File, Skip if Set			
Syntax:	BTFSS f, b {,a}			
Operands:	0 ≤ f ≤ 255 0 ≤ b < 7 a ∈ [0,1]			
Operation:	skip if (f<b>) = 1			
Status Affected:	None			
Encoding:	1010	bbba	ffff	ffff
Description:	If bit 'b' in register 'f' is '1', then the next instruction is skipped. If bit 'b' is '1', then the next instruction fetched during the current instruction execution is discarded and a NOP is executed instead, making this a 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 ≤ 95 (5Fh). See 36.2.3 Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode for details.			
Words:	1			
Cycles:	1(2) Note: Three 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 FALSE TRUE	BTFSS : :	FLAG, 1, 0
Before Instruction PC = address (HERE) After Instruction If FLAG<1> = 0; PC = address (FALSE) If FLAG<1> = 1; PC = address (TRUE)			

BTG	Bit Toggle f			
Syntax:	BTG f, b {,a}			
Operands:	0 ≤ f ≤ 255 0 ≤ b < 7 a ∈ [0,1]			
Operation:	($\overline{f<b>}$) → 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 ≤ 95 (5Fh). See 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 ≤ n ≤ 127			
Operation:	if OVERFLOW bit is ‘1’ (PC) + 2 + 2n → PC			
Status Affected:	None			
Encoding:	1110	0100	nnnn	nnnn
Description:	If the OVERFLOW bit is ‘1’, then the program will branch. The 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)

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

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) → STATUSS, (BSR) → BSRS			
Status Affected:	None			
Encoding: 1st word (k<7:0>) 2nd word(k<19:8>)	1110 1111	110s k_{19}kkk	$k_7$kkk kkkk	kkkk$_0$ kkkk$_8$
Description:	Subroutine call of entire 2-Mbyte memory range. First, return address (PC + 4) is pushed onto the return stack. If ‘s’ = 1, the W, Status and BSR registers are also pushed into their respective shadow registers, WS, STATUSS and BSRS. 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	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

STATUSS = Status

CLRF	Clear f			
Syntax:	CLRF f {,a}			
Operands:	0 ≤ f ≤ 255 a ∈ [0,1]			
Operation:	000h → f 1 → Z			
Status Affected:	Z			
Encoding:	0110	101a	ffff	ffff
Description:	Clears the contents of the specified register. 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 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:	CLRF	FLAG_REG, 1
Before Instruction FLAG_REG = 5Ah After Instruction FLAG_REG = 00h		

CLRWDT	Clear Watchdog Timer			
Syntax:	CLRWDT			
Operands:	None			
Operation:	000h → WDT, 000h → WDT postscaler, 1 → $\overline{TO}$, 1 → $\overline{PD}$			
Status Affected:	$\overline{TO}$, $\overline{PD}$			
Encoding:	0000	0000	0000	0100
Description:	CLRWDT instruction resets the Watchdog Timer. It also resets the postscaler of the WDT. Status bits, $\overline{TO}$ and $\overline{PD}$, are set.			
Words:	1			
Cycles:	1			

Q Cycle Activity:			
Q1	Q2	Q3	Q4
Decode	No operation	Process Data	No operation

Example:	CLRWDT
Before Instruction WDT Counter = ? After Instruction WDT Counter = 00h WDT Postscaler = 0 $\overline{TO}$ = 1 $\overline{PD}$ = 1	

COMF	Complement f
Syntax:	COMF f {,d {,a}}
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]
Operation:	($\bar{f}$) → dest

COMF	Complement f			
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 ≤ 95 (5Fh). See 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 ≤ f ≤ 255 a ∈ [0,1]			
Operation:	(f) – (W), skip if (f) = (W) (unsigned comparison)			
Status Affected:	None			
Encoding:	0110	001a	ffff	ffff

CPFSEQ	Compare f with W, skip if f = W
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 ≤ 95 (5Fh). See 36.2.3 Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode for details.
Words:	1
Cycles:	1(2) Note: Three 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 ≠ ; PC = Address (NEQUAL)

CPFSGT	Compare f with W, skip if f > W			
Syntax:	CPFSGT f {,a}			
Operands:	0 ≤ f ≤ 255 a ∈ [0,1]			
Operation:	(f) – (), 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 ≤ 95 (5Fh). See 36.2.3 Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode for details.			
Words:	1			
Cycles:	1(2) Note: Three 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 ≤ f ≤ 255 a ∈ [0,1]			
Operation:	(f) – (), 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.			

CPFSLT	Compare f with W, skip if f < W			
Words:	1			
Cycles:	1(2) Note: Three 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

DAW	Decimal Adjust W Register			

	(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:	0000	0000	0000	0111
Description:	DAW adjusts the 8-bit value in W, resulting from the earlier addition of two variables (each in packed BCD format) and produces a correct packed BCD result.			
Words:	1			
Cycles:	1			

Q Cycle Activity:			
Q1	Q2	Q3	Q4
Decode	Read register W	Process Data	Write W

Example1:	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:	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 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:	DECF CNT,	1, 0
Before Instruction CNT = 01h Z = 0 After Instruction CNT = 00h Z = 1		

DECFSZ	Decrement f, skip if 0			
Syntax:	DECFSZ f {,d {,a}}			
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]			
Operation:	(f) – 1 → dest, skip if result = 0			
Status Affected:	None			
Encoding:	0010	11da	ffff	ffff
Description:	The contents of register ‘f’ are decremented. If ‘d’ is ‘0’, the result is placed in W. If ‘d’ is ‘1’, the result is placed back in register ‘f’ (default). 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 ≤ 95 (5Fh). See 36.2.3 Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode for details.			
Words:	1			
Cycles:	1(2) Note: Three 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 ≠ 0; PC = Address (HERE + 2)	

DCFSNZ	Decrement f, skip if not 0			
Syntax:	DCFSNZ f {,d {,a}}			
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]			
Operation:	(f) – 1 → dest, skip if result ≠ 0			
Status Affected:	None			
Encoding:	0100	11da	ffff	ffff
Description:	The contents of register ‘f’ are decremented. If ‘d’ is ‘0’, the result is placed in W. If ‘d’ is ‘1’, the result is placed back in register ‘f’ (default). 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 ≤ 95 (5Fh). See 36.2.3 Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode for details.			

DCFSNZ	Decrement f, skip if not 0			
Words:	1			
Cycles:	1(2) Note: Three 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 ≠ 0; PC = Address (NZERO)	

GOTO	Unconditional Branch
Syntax:	GOTO k
Operands:	0 ≤ k ≤ 1048575
Operation:	k → PC<20:1>

GOTO	Unconditional Branch			
Status Affected:	None			
Encoding: 1st word (k<7:0>) 2nd word(k<19:8>)	1110 1111	1111 k_{19}kkk	$k_7$kkk kkkk	kkkk$_0$ kkkk$_8$
Description:	GOTO allows an unconditional branch anywhere within entire 2-Mbyte memory range. The 20-bit value ‘k’ is loaded into PC<20:1>. GOTO is always a 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 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]			
Operation:	(f) + 1 → dest			
Status Affected:	C, DC, N, OV, Z			
Encoding:	0010	10da	ffff	ffff
Description:	The contents of register ‘f’ are incremented. If ‘d’ is ‘0’, the result is 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.			

INCF	Increment f			
	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 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 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]			
Operation:	(f) + 1 → dest, skip if result = 0			
Status Affected:	None			
Encoding:	0011	11da	ffff	ffff

INCFSZ	Increment f, skip if 0			
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 ≤ 95 (5Fh). See 36.2.3 Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode for details.			
Words:	1			
Cycles:	1(2) Note: Three 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 ≠ 0;

PC = Address (NZERO)

INFSNZ	Increment f, skip if not 0			
Syntax:	INFSNZ f {,d {,a}}			
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]			
Operation:	(f) + 1 → dest, skip if result ≠ 0			
Status Affected:	None			
Encoding:	0100	10da	ffff	ffff
Description:	The contents of register ‘f’ are incremented. If ‘d’ is ‘0’, the result is placed in W. If ‘d’ is ‘1’, the result is placed back in register ‘f’ (default). 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 ≤ 95 (5Fh). See 36.2.3 Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode for details.			
Words:	1			
Cycles:	1(2) Note: Three 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 ≠ 0; PC = Address (NZERO) If REG = 0; PC = Address (ZERO)	

IORLW	Inclusive OR literal with W			
Syntax:	IORLW k			
Operands:	0 ≤ k ≤ 255			
Operation:	(W) .OR. k → W			
Status Affected:	N, Z			
Encoding:	0000	1001	kkkk	kkkk
Description:	The contents of W are ORed with the 8-bit literal ‘k’. The result is placed in W.			
Words:	1			
Cycles:	1			

Q Cycle Activity:			
Q1	Q2	Q3	Q4
Decode	Read literal ‘k’	Process Data	Write to W

Example:	IORLW	35h
Before Instruction W = 9Ah After Instruction W = BFh		

IORWF	Inclusive OR W with f			
Syntax:	IORWF f {,d {,a}}			
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]			
Operation:	(W) .OR. (f) → dest			
Status Affected:	N, Z			
Encoding:	0001	00da	ffff	ffff
Description:	Inclusive OR W with register ‘f’. If ‘d’ is ‘0’, the result is 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 ≤ 95 (5Fh). See 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

LFSR	Load FSR			
Syntax:	LFSR f, k			
Operands:	0 ≤ f ≤ 2 0 ≤ k ≤ 4095			
Operation:	k → FSRf			
Status Affected:	None			
Encoding:	1110 1111	1110 0000	00ff $k_7$kkk	k_{11}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:			
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:	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			

MOVF	Move f			
	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 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 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:	$0 \le f_s \le 4095$ $0 \le f_d \le 4095$			
Operation:	$(f_s) \rightarrow f_d$			
Status Affected:	None			
Encoding: 1st word (source) 2nd word (destin.)	1100 1111	ffff ffff	ffff ffff	$ffff_s$ $ffff_d$

MOVFF	Move f to f			
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 ≤ k ≤ 255			
Operation:	k → 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:k_4.			

MOVLB	Move literal to low nibble in BSR			
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		

MOVLW	Move literal to W			
Syntax:	MOVLW k			
Operands:	0 ≤ k ≤ 255			
Operation:	k → W			
Status Affected:	None			
Encoding:	0000	1110	kkkk	kkkk
Description:	The 8-bit literal ‘k’ is loaded into W.			
Words:	1			
Cycles:	1			

Q Cycle Activity:			
Q1	Q2	Q3	Q4
Decode	Read literal ‘k’	Process Data	Write to W

Example:	MOVLW	5Ah
After Instruction W = 5Ah		

MOVWF	Move W to f			
Syntax:	MOVWF f {,a}			
Operands:	0 ≤ f ≤ 255 a ∈ [0,1]			
Operation:	(W) → f			
Status Affected:	None			
Encoding:	0110	111a	ffff	ffff
Description:	Move data from W to register ‘f’. Location ‘f’ can be anywhere in the 256-byte bank. 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 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 ≤ k ≤ 255
Operation:	(W) x k → PRODH:PRODL

MULLW	Multiply literal with W			
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 ≤ f ≤ 255 a ∈ [0,1]
Operation:	(W) x (f) → PRODH:PRODL
Status Affected:	None

MULWF	Multiply W with f			
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 ≤ 95 (5Fh). See Section 35.2.3 “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 ≤ f ≤ 255 a ∈ [0,1]			
Operation:	$(\overline{f}) + 1 \rightarrow f$			
Status Affected:	N, OV, C, DC, Z			
Encoding:	0110	110a	ffff	ffff
Description:	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 ≤ 95 (5Fh). See 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:	`NEGF REG, 1`
Before Instruction REG = 0011 1010 [3Ah] After Instruction REG = 1100 0110 [C6h]	

NOP	No Operation			
Syntax:	NOP			
Operands:	None			
Operation:	No operation			
Status Affected:	None			
Encoding:	0000 1111	0000 xxxx	0000 xxxx	0000 xxxx

NOP	No Operation			
Description:	No operation.			
Words:	1			
Cycles:	1			

Q Cycle Activity:			
Q1	Q2	Q3	Q4
Decode	No operation	No operation	No operation

Example:
None.

POP	Pop Top of Return Stack			
Syntax:	POP			
Operands:	None			
Operation:	(TOS) → bit bucket			
Status Affected:	None			
Encoding:	0000	0000	0000	0110
Description:	The TOS value is pulled off the return stack and is discarded. The TOS value then becomes the previous value that was pushed onto the return stack. This instruction is provided to enable the user to properly manage the return stack to incorporate a software stack.			
Words:	1			
Cycles:	1			

Q Cycle Activity:			
Q1	Q2	Q3	Q4
Decode	No operation	POP TOS value	No operation

Example:	POP GOTO	NEW		
Before Instruction TOS = 0031A2h Stack (1 level down) = 014332h After Instruction				

TOS = 014332h

PC = NEW

PUSH	Push Top of Return Stack			
Syntax:	PUSH			
Operands:	None			
Operation:	(PC + 2) → TOS			
Status Affected:	None			
Encoding:	0000	0000	0000	0101
Description:	The PC + 2 is pushed onto the top of the return stack. The previous TOS value is pushed down on the stack. This instruction allows implementing a software stack by modifying TOS and then pushing it onto the return stack.			
Words:	1			
Cycles:	1			

Q Cycle Activity:			
Q1	Q2	Q3	Q4
Decode	PUSH PC + 2 onto return stack	No operation	No operation

Example:	PUSH		

Before Instruction
TOS = 345Ah

PC = 0124h

After Instruction

PC = 0126h

TOS = 0126h

Stack (1 level down) = 345Ah

RCALL	Relative Call
Syntax:	RCALL n
Operands:	-1024 ≤ n ≤ 1023
Operation:	(PC) + 2 → TOS, (PC) + 2 + 2n → PC

RCALL	Relative Call			
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, (STATUSS) → 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, STATUSS 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 = STATUSS 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            ;  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

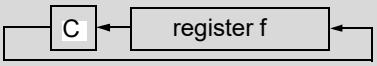
After Instruction

W = value of kn

RETURN	Return from Subroutine			
Syntax:	RETURN {s}			
Operands:	s ∈ [0,1]			
Operation:	(TOS) → PC, if s = 1 (WS) → W, (STATUSS) → 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, STATUSS 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

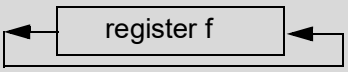
Example:	RETURN
After Instruction: PC = TOS	

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 35.2.3 “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

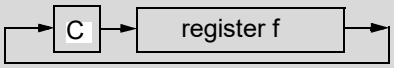
Example:	RLCF	REG, 0, 0
Before Instruction REG = 1110 0110 C = 0		

After Instruction REG = 1110 0110 W = 1100 1100 C = 1

RLNCF	Rotate Left f (No Carry)			
Syntax:	RLNCF f {,d {,a}}			
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]			
Operation:	(f<n>) → dest<n + 1>, (f<7>) → dest<0>			
Status Affected:	N, Z			
Encoding:	0100	01da	ffff	ffff
Description:	The contents of register ‘f’ are rotated one bit to the left. If ‘d’ is ‘0’, the result is 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 36.2.3 Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode for details. 			
Words:	1			
Cycles:	1			

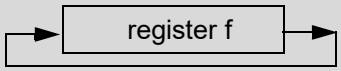
Q Cycle Activity:			
Q1	Q2	Q3	Q4
Decode	Read register ‘f’	Process Data	Write to destination

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 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]			
Operation:	(f<n>) → dest<n – 1>, (f<0>) → C, (C) → dest<7>			
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 ≤ 95 (5Fh). See 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 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]			
Operation:	(f<n>) → dest<n – 1>, (f<0>) → dest<7>			
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 ≤ 95 (5Fh). See 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 ≤ f ≤ 255 a ∈ [0,1]			
Operation:	FFh → 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 ≤ 95 (5Fh). See 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,

SLEEP	Enter Sleep mode			
	0 → WDT postscaler, 1 → $\overline{TO}$, 0 → $\overline{PD}$			
Status Affected:	$\overline{TO}$, $\overline{PD}$			
Encoding:	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:			
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 (Continued)			
Syntax:	SUBFWB f {,d {,a}}			
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]			
Operation:	(W) – (f) – ($\overline{C}$) → dest			
Status Affected:	N, OV, C, DC, Z			
Encoding:	0101	01da	ffff	ffff

SUBFWB	Subtract f from W with borrow (Continued)
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 ≤ 95 (5Fh). See Section 35.2.3 “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:	`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 N = 0	

SUBLW	**Subtract W from literal**			
Syntax:	SUBLW k			
Operands:	0 ≤ k ≤ 255			
Operation:	k – (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 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]			
Operation:	(f) – (W) → dest			
Status Affected:	N, OV, C, DC, Z			
Encoding:	0101	11da	ffff	ffff

SUBWF	Subtract W from f
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 ≤ 95 (5Fh). See Section 35.2.3 “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 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]			
Operation:	(f) – (W) – ($\overline{C}$) → dest			
Status Affected:	N, OV, C, DC, Z			
Encoding:	0101	10da	ffff	ffff
Description:	Subtract W and the CARRY flag (borrow) from register 'f' (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 ≤ 95 (5Fh). See 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 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]			
Operation:	(f<3:0>) → dest<7:4>, (f<7:4>) → dest<3:0>			
Status Affected:	None			
Encoding:	0011	10da	ffff	ffff
Description:	The upper and lower nibbles of register 'f' are exchanged. If 'd' is '0', the result is 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 ≤ 95 (5Fh). See 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

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			

TBLRD	Table Read			
Words:	1			
Cycles:	2			

Q Cycle Activity:			
Q1	Q2	Q3	Q4
Decode	No operation	No operation	No operation
No operation	No operation (Read Program Memory)	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 (Continued)	Table Write
Syntax:	TBLWT (*; *+; *-; +*)
Operands:	None
Operation:	if TBLWT*,

TBLWT (Continued)	Table Write			
	(TABLAT) → Holding Register; TBLPTR – No Change; if TBLWT*+, (TABLAT) → Holding Register; (TBLPTR) + 1 → TBLPTR; if TBLWT*-, (TABLAT) → Holding Register; (TBLPTR) – 1 → TBLPTR; if TBLWT+*, (TBLPTR) + 1 → TBLPTR; (TABLAT) → Holding Register;			
Status Affected:	None			
Encoding:	0000	0000	0000	11nn nn=0 * =1 *+ =2 *- =3 +*
Description:	This instruction uses the LSBs of TBLPTR to determine which of the holding registers the TABLAT is written to. The holding registers are used to program the contents of Program Memory (P.M.). (Refer to the *"Program Flash Memory"* section 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			

TBLWT (Continued)	Table Write			
Q Cycle Activity:				
	Q1	Q2	Q3	Q4
	Decode	No operation	No operation	No operation
	No operation	No operation (Read TABLAT)	No operation	No operation (Write to Holding Register)

TBLWT	Table Write (Continued)
Example1:	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 ≤ f ≤ 255 a ∈ [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 ≤ 95 (5Fh). See 36.2.3 Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode for details.			
Words:	1			
Cycles:	1(2) Note: Three 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 ≠ 00h, PC = Address (NZERO)	

XORLW	Exclusive OR literal with W			
Syntax:	XORLW k			
Operands:	0 ≤ k ≤ 255			
Operation:	(W) .XOR. k →			
Status Affected:	N, Z			
Encoding:	0000	1010	kkkk	kkkk
Description:	The contents of W are XORed with the 8-bit literal ‘k’. The result is placed in W.			
Words:	1			
Cycles:	1			

Q Cycle Activity:			
Q1	Q2	Q3	Q4
Decode	Read literal ‘k’	Process Data	Write to W

Example:	XORLW	0AFh
Before Instruction W = B5h After Instruction W = 1Ah		

XORWF	Exclusive OR W with f			
Syntax:	XORWF f {,d {,a}}			
Operands:	0 ≤ f ≤ 255 d ∈ [0,1] a ∈ [0,1]			
Operation:	(W) .XOR. (f) → dest			
Status Affected:	N, Z			
Encoding:	0001	10da	ffff	ffff
Description:	Exclusive OR the contents of W with register 'f'. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in the register 'f' (default). 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 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, PIC18F24/25Q10 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 36.2.1 Extended Instruction Syntax. Detailed descriptions are provided in 36.2.2 Extended Instruction Set. The opcode field descriptions in 36.1 Standard Instruction Set apply to both the standard and extended PIC18 instruction sets.

Important: The instruction set extension and the Indexed Literal Offset Addressing mode were designed for optimizing applications written in C; the user may likely never use these instructions directly in assembler. The syntax for these commands is provided as a reference for users who may be reviewing code that has been generated by a compiler.

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 36.2.3.1 Extended Instruction Syntax with Standard PIC18 Commands.

Important: 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	2	1110	1011	0zzz	zzzz	None
		f_d (destination) 2nd word		1111	ffff	ffff	ffff	
MOVSS	z_s, z_d	Move z_s (source) to 1st word	2	1110	1011	1zzz	zzzz	None
		z_d (destination) 2nd word		1111	xxxx	xzzz	zzzz	
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:	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:			
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	

ADDULNK	**Add Literal to FSR2 and Return**			
Syntax:	ADDULNK k			
Operands:	0 ≤ k ≤ 63			
Operation:	FSR2 + k → FSR2,			
	(TOS) → PC			
Status Affected:	None			
Encoding:	1110	1000	11kk	kkkk
Description:	The 6-bit literal ‘k’ is added to the contents of FSR2. A RETURN is then executed by loading the PC with the TOS. The instruction takes two cycles to execute; a `NOP` is performed during the second cycle. This may be thought of as a special case of the ADDFSR instruction, where f = 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:	ADDULNK 23h
Before Instruction FSR2 = 03FFh	

PC = 0100h After Instruction FSR2 = 0422h PC = (TOS)



Important: All PIC18 instructions may take an optional label argument preceding the instruction mnemonic for use in symbolic addressing. If a label is used, the instruction syntax then becomes: {label} instruction argument(s).

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:	$0 \le z_s \le 127$ $0 \le f_d \le 4095$			
Operation:	((FSR2) + z_s) → f_d			
Status Affected:	None			
Encoding: 1st word (source) 2nd word (destin.)	1110 1111	1011 ffff	0zzz ffff	$zzzz_s$ $ffff_d$
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:	$0 \le z_s \le 127$ $0 \le z_d \le 127$			
Operation:	((FSR2) + z_s) → ((FSR2) + z_d)			
Status Affected:	None			
Encoding: 1st word (source) 2nd word (dest.)	1110 1111	1011 xxxx	1zzz xzzz	$zzzz_s$ $zzzz_d$
Description	The contents of the source register are moved to the destination register. The addresses of the source and destination registers are determined by adding the 7-bit literal offsets ‘z_s’ or ‘z_d’, respectively, to the value of FSR2. Both registers can be located anywhere in the 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			

MOVSS	Move Indexed to Indexed			
	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

PUSHL	Store Literal at FSR2, Decrement FSR2			
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	

SUBFSR	Subtract Literal from FSR			
Syntax:	SUBFSR f, k			
Operands:	0 ≤ k ≤ 63			
	f ∈ [0, 1, 2]			
Operation:	FSR(f) – k → 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 ≤ k ≤ 63			
Operation:	FSR2 – k → FSR2			
	(TOS) → 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

Important: 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 “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 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.

Related Links

10.5 Data Memory and the Extended Instruction Set

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.

Related Links

10.5 Data Memory and the Extended Instruction Set

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 PIC18F24/25Q10 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 ≤ k ≤ 95 d ∈ [0,1]			
Operation:	(W) + ((FSR2) + k) → dest			
Status Affected:	N, OV, C, DC, Z			
Encoding:	0010	01d0	kkkk	kkkk
Description:	The contents of W are added to the contents of the register indicated by FSR2, offset by the value ‘k’. If ‘d’ is ‘0’, the result is stored in W. If ‘d’ is ‘1’, the result is stored back in 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 ≤ f ≤ 95 0 ≤ b ≤ 7			
Operation:	1 → ((FSR2) + k)<b>			
Status Affected:	None			
Encoding:	1000	bbb0	kkkk	kkkk
Description:	Bit ‘b’ of the register indicated by FSR2, offset by the value ‘k’, is set.			
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 ≤ k ≤ 95			
Operation:	FFh → ((FSR2) + k)			
Status Affected:	None			
Encoding:	0110	1000	kkkk	kkkk
Description:	The contents of the register indicated by FSR2, offset by ‘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 PIC18F24/25Q10 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. 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/library 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. Electrical Specifications

38.1 Absolute Maximum Ratings(†)

Parameter		Rating
Ambient temperature under bias		-40°C to +125°C
Storage temperature		-65°C to +150°C
Voltage on pins with respect to V_{SS}		
• **on V_{DD} pin:**		-0.3V to +6.5V
• **on $\overline{MCLR}$ pin:**		-0.3V to +9.0V
• **on all other pins:**		-0.3V to (V_{DD} + 0.3V)
Maximum current		
• **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	250 mA
	85°C < T_A ≤ +125°C	85 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

Important:

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 Thermal Characteristics to calculate device specifications.
2. Power dissipation is calculated as follows:
   $P_{DIS} = V_{DD} \times \{I_{DD} - \Sigma I_{OH}\} + \Sigma \{(V_{DD} - V_{OH}) \times I_{OH}\} + \Sigma (V_{OI} \times I_{OL})$

NOTICE: Stresses above those listed under *"Absolute Maximum Ratings"* may cause permanent damage to the device. This is a stress rating only and functional operation of the device at those or any other conditions above those indicated in the operation listings of this specification is not implied. Exposure above maximum rating conditions for extended periods may affect device reliability.

38.2 Standard Operating Conditions

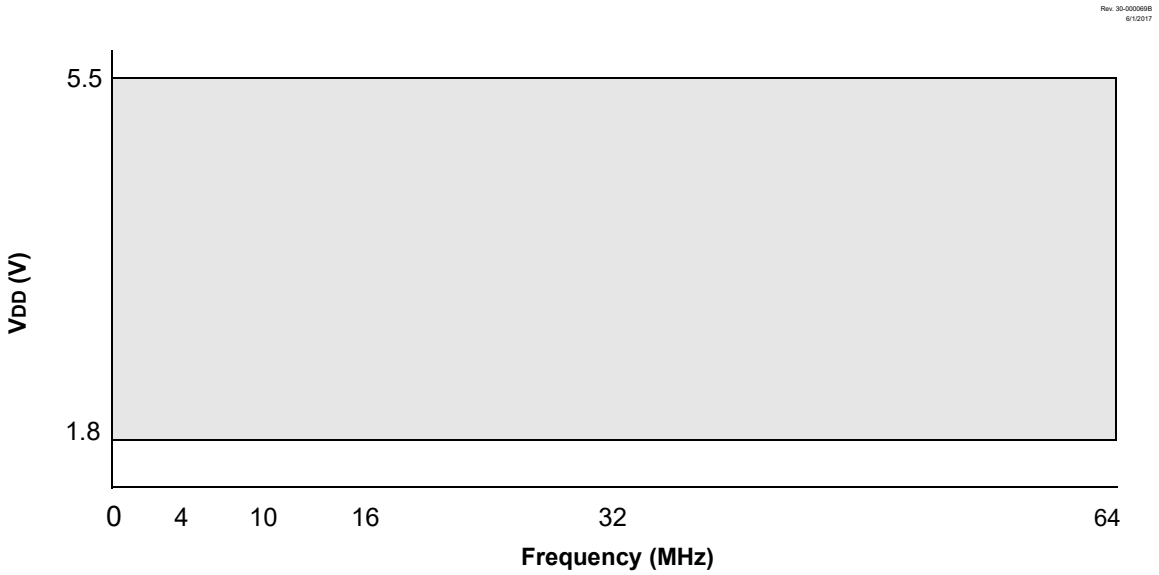
The standard operating conditions for any device are defined as:

Operating Voltage: $V_{DDMIN} \le V_{DD} \le V_{DDMAX}$

Operating Temperature: $T_{A_MIN} \le T_A \le T_{A_MAX}$

Parameter		Ratings
V_{DD} — Operating Supply Voltage(1)	V_{DDMIN}	+1.8V
	V_{DDMAX}	+5.5V
T_A — Operating Ambient Temperature Range		
Industrial Temperature	T_{A_MIN}	-40°C
	T_{A_MAX}	+85°C
Extended Temperature	T_{A_MIN}	-40°C
	T_{A_MAX}	+125°C

Figure 38-1. Voltage Frequency Graph, -40°C ≤ T_A≤ +125°C



Note:

1. The shaded region indicates the permissible combinations of voltage and frequency.
2. Refer to 38.4.1 External Clock/Oscillator Timing Requirements for each Oscillator mode's supported frequencies.

38.3 DC Characteristics

38.3.1 Supply Voltage

Table 38-1.

Standard Operating Conditions (unless otherwise stated)							
Param. No.	**Sym.**	**Characteristic**	**Min.**	**Typ.†**	**Max.**	**Units**	**Conditions**
Supply Voltage							
D002	V_{DD}		1.8	—	5.5	V	
RAM Data Retention(1)							

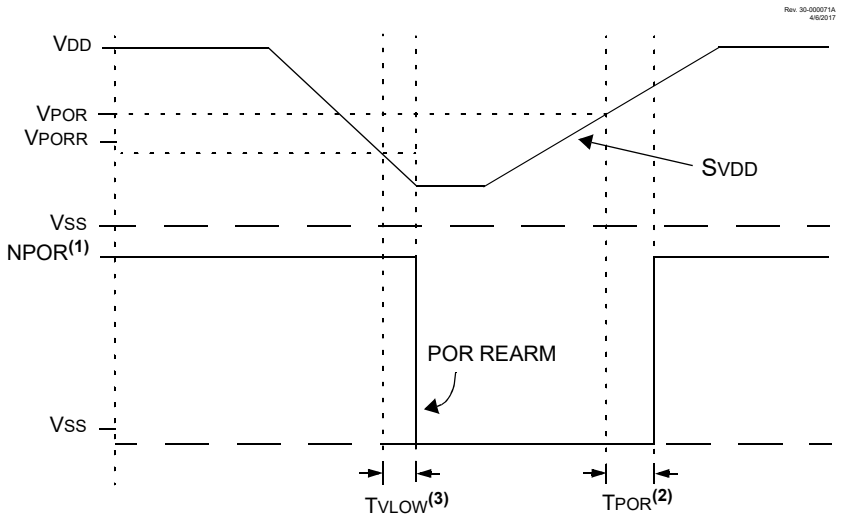
Standard Operating Conditions (unless otherwise stated)							
Param. No.	Sym.	Characteristic	Min.	Typ.†	Max.	Units	Conditions
D003	V_{DR}		1.7	—	—	V	Device in SLEEP mode
Power-on Reset Release Voltage(2)							
D004	V_{POR}		—	1.6	—	V	BOR or LPBOR disabled(3)
Power-on Reset Rearm Voltage(2)							
D005	V_{PORR}		—	1	—	V	BOR or LPBOR disabled(3)
V_{DD} Rise Rate to ensure internal Power-on Reset signal(2)							
D006	S_{VDD}		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 V_{DD} can be lowered in Sleep mode without losing RAM data.
2. See the following figure, POR and POR REARM with Slow Rising V_{DD}.
3. Please see 38.4.5 Reset, WDT, Oscillator Start-up Timer, Power-up Timer, Brown-Out Reset and Low-Power Brown-Out Reset Specifications for BOR and LPBOR trip point information.

Figure 38-2. POR and POR Rearm with Slow Rising V_{DD}



Note:

1. When NPOR is low, the device is held in Reset.
2. T_{POR} 1 µs typical.
3. T_{VLOW} 2.7 µs typical.

38.3.2 Supply Current (I_{DD})(1,2,4)

Table 38-2.

Standard Operating Conditions (unless otherwise stated)								
Param. No.	Sym.	Device Characteristics	Min.	Typ.†	Max.	Units	Conditions	
							V_{DD}	Note
D100	$I_{DD_{XT4}}$	XT = 4 MHz	—	440	—	µA	3.0V	
D100A	$I_{DD_{XT4}}$	XT = 4 MHz	—	370	—	µA	3.0V	PMD's all 1's
D101	$I_{DD_{HFO16}}$	HFINTOSC = 16 MHz	—	1.2	—	mA	3.0V	
D101A	$I_{DD_{HFO16}}$	HFINTOSC = 16 MHz	—	1.0	—	mA	3.0V	PMD's all 1's
D102	$I_{DD_{HFOPLL}}$	HFINTOSC = 64 MHz	—	4.0	—	mA	3.0V	
D102A	$I_{DD_{HFOPLL}}$	HFINTOSC = 64 MHz	—	3.1	—	mA	3.0V	PMD's all 1's
D103	$I_{DD_{HSPLL32}}$	HS+PLL = 64 MHz	—	3.9	—	mA	3.0V	
D103A	$I_{DD_{HSPLL32}}$	HS+PLL = 64 MHz	—	3.0	—	mA	3.0V	PMD's all 1's
D104	$I_{DD_{IDLE}}$	IDLE mode, HFINTOSC = 16 MHz	—	0.5	—	mA	3.0V	
D105	$I_{DD_{DOZE}}$(3)	DOZE mode, HFINTOSC = 16 MHz, Doze Ratio = 16	—	0.6	—	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 I_{DD} measurements in active operation mode are: OSC1 = external square wave, from rail-to-rail; all I/O pins are outputs driven low; $\overline{MCLR}$ = V_{DD}; WDT disabled.
2. The supply current is mainly a function of the operating voltage and frequency. Other factors, such as I/O pin loading and switching rate, oscillator type, internal code execution pattern and temperature, also have an impact on the current consumption.
3. $I_{DD_{DOZE}}$ = [$I_{DD_{IDLE}}$*(N-1)/N] + $I_{DD_{HFO}}$16/N where N = DOZE Ratio (see *CPUDOZE* register).
4. PMD bits are all in the default state, no modules are disabled.

Related Links

6.6.2 CPUDOZE

38.3.3 Power-Down Current (I_{PD})(1,2)

Table 38-3.

Standard Operating Conditions (unless otherwise stated)										
Param. No.	Sym.	Device Characteristics	Min.	Typ.†	Max. +85°C	Max. +125°C	Units	Conditions		
								V_{DD}	VREGPM	Note
D200	I_{PD}	I_{PD} Base	—	—	—	—	µA	3.0V	b'11'	Reserved
D200A	I_{PD}	I_{PD} Base	—	0.6	—	—	µA	3.0V	b'10'	
D200B			—	45	—	—	µA	3.0V	b'01'	
D200C			—	170	—	—	µA	3.0V	b'00'	
D201	I_{PD_WDT}	Low-Frequency Internal Oscillator/WDT	—	0.9	—	—	µA	3.0V	b'10'	
D202	I_{PD_SOSC}	Secondary Oscillator (S_{OSC})	—	1.1	—	—	µA	3.0V	b'10'	
D203	I_{PD_LPBOR}	Low-Power Brown-out Reset (LPBOR)	—	0.9	—	—	µA	3.0V	b'10'	
D204	I_{PD_FVR}	FVR	—	63	—	—	µA	3.0V	b'10'	FVRCON = 0x81 or 0x84
D205	I_{PD_BOR}	Brown-out Reset (BOR)	—	30	—	—	µA	3.0V	b'10'	
D206	I_{PD_HLVD}	High/Low Voltage Detect (HLVD)	—	22	—	—	µA	3.0V	b'10'	
D207	I_{PD_ADCA}	ADC - Active	—	330	—	—	µA	3.0V	b'10'	ADC is converting (4)
D208	I_{PD_CMP}	Comparator	—	48	—	—	µA	3.0V	b'10'	Only one comparator active

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 I_{DD} and the additional current consumed when this peripheral is enabled. The peripheral Δ current can be determined by subtracting the base I_{DD} or I_{PD}current from this limit. Max. values should be used when calculating total current consumption.

Standard Operating Conditions (unless otherwise stated)										
Param. No.	Sym.	Device Characteristics	Min.	Typ.†	Max. +85°C	Max. +125°C	Units	Conditions		
								V_{DD}	VREGPM	Note

2. The power-down current in Sleep mode does not depend on the oscillator type. Power-down current is measured with the part in Sleep mode with all I/O pins in high-impedance state and tied to V_{SS}.
3. All peripheral currents listed are on a per-peripheral basis if more than one instance of a peripheral is available.
4. ADC clock source is FRC.

38.3.4 I/O Ports

Table 38-4.

Standard Operating Conditions (unless otherwise stated)							
Param. No.	Sym.	Device Characteristics	Min.	Typ.†	Max.	Units	Conditions
Input Low Voltage							
	V_{IL}	I/O PORT:					
D300		• with TTL buffer	—	—	0.8	V	$4.5V \le V_{DD} \le 5.5V$
D301			—	—	0.15 V_{DD}	V	$1.8V \le V_{DD} \le 4.5V$
D302		• with Schmitt Trigger buffer	—	—	0.2 V_{DD}	V	$2.0V \le V_{DD} \le 5.5V$
D303		• with I^2C levels	—	—	0.3 V_{DD}	V	
D304		• with SMBus levels	—	—	0.8	V	$2.7V \le V_{DD} \le 5.5V$
D305		$\overline{MCLR}$	—	—	0.2 V_{DD}	V	
High Low Voltage							
	V_{IH}	I/O PORT:					
D320		• with TTL buffer	2.0	—	—	V	$4.5V \le V_{DD} \le 5.5V$
D321			0.25 V_{DD} +0.8	—	—	V	$1.8V \le V_{DD} \le 4.5V$
D322		• with Schmitt Trigger buffer	0.8V_{DD}	—	—	V	$2.0V \le V_{DD} \le 5.5V$
D323		• with I^2C levels	0.7 V_{DD}	—	—	V	
D324		• with SMBus levels	2.1	—	—	V	$2.7V \le V_{DD} \le 5.5V$
D325		$\overline{MCLR}$	0.7 V_{DD}	—	—	V	
Input Leakage Current(1)							
D340	I_{IL}	I/O PORTS	—	±5	±125	nA	$V_{SS} \le V_{PIN} \le V_{DD}$,

Standard Operating Conditions (unless otherwise stated)							
Param. No.	Sym.	Device Characteristics	Min.	Typ.†	Max.	Units	Conditions
							Pin at high-impedance, 85°C
D341			—	±5	±1000	nA	$V_{SS} \le V_{PIN} \le V_{DD}$, Pin at high-impedance, 125°C
D342		$\overline{MCLR}$(2)	—	±50	±200	nA	$V_{SS} \le V_{PIN} \le V_{DD}$, Pin at high-impedance, 85°C
Weak Pull-up Current							
D350	I_{PUR}		80	140	200	µA	V_{DD}=3.0V, $V_{PIN}=V_{SS}$
Output Low Voltage							
D360	V_{OL}	I/O PORTS	—	—	0.6	V	I_{OL}=10.0 mA, V_{DD}=3.0V
Output High Voltage							
D370	V_{OH}	I/O PORTS	V_{DD}-0.7	—	—	V	I_{OH}=6.0 mA, V_{DD}=3.0V
All I/O Pins							
D380	C_{IO}		—	5	50	pF	

Data in “Typ” column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

Note:

1. Negative current is defined as current sourced by the pin.
2. The leakage current on the $\overline{MCLR}$ pin is strongly dependent on the applied voltage level. The specified levels represent normal operating conditions. Higher leakage current may be measured at different input voltages.

38.3.5 Memory Programming Specifications

Table 38-5.

Standard Operating Conditions (unless otherwise stated)							
Param No.	Sym.	Device Characteristics	Min.	Typ†	Max.	Units	Conditions
Data EEPROM Memory Specifications							
MEM20	E_D	DataEE Byte Endurance	100k	—	—	E/W	-40°C≤T_A≤+85°C
MEM21	T_{D_RET}	Characteristic Retention	—	40	—	Year	Provided no other specifications are violated
MEM22	N_{D_REF}	Total Erase/Write Cycles before Refresh	1M	10M	—	E/W	-40°C≤ T_A≤+85°C

Standard Operating Conditions (unless otherwise stated)							
Param No.	Sym.	Device Characteristics	Min.	Typ†	Max.	Units	Conditions
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	—	10	TBD	ms	
Program Flash Memory Specifications							
MEM30	E_P	Flash Memory Cell Endurance	10k	—	—	E/W	-40°C≤Ta≤+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 Sector Write	—	6	TBD	ms	
MEM36	T_{SE}	Self-Timed Sector Erase	—	10	TBD	ms	
MEM37	T_{P_WRD}	Self-Timed Word Write	—	50	TBD	µs	

Data in “Typ” column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

Note:

1. Flash Memory Cell Endurance for the Flash memory is defined as: One Row Erase operation and one Self-Timed Write.

38.3.6 Thermal Characteristics

Table 38-6.

Standard Operating Conditions (unless otherwise stated)					
Param No.	Sym.	Characteristic	Typ.	Units	Conditions
TH01	θ_{JA}	Thermal Resistance Junction to Ambient	60	°C/W	28-pin SPDIP package
			80	°C/W	28-pin SOIC package
			90	°C/W	28-pin SSOP package
			27.5	°C/W	28-pin UQFN 4x4 mm package
			27.5	°C/W	28-pin QFN 6x6mm package
TH02	θ_{JC}	Thermal Resistance Junction to Case	31.4	°C/W	28-pin SPDIP package
			24	°C/W	28-pin SOIC package

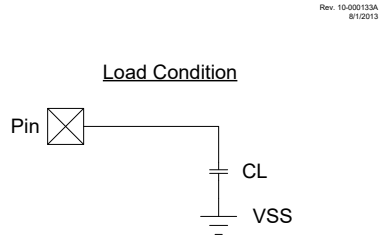
Standard Operating Conditions (unless otherwise stated)					
Param No.	**Sym.**	**Characteristic**	**Typ.**	**Units**	**Conditions**
			24	°C/W	28-pin SSOP package
			24	°C/W	28-pin UQFN 4x4mm package
			24	°C/W	28-pin QFN 6x6mm package
TH03	T_{JMAX}	Maximum Junction Temperature	150	°C	
TH04	PD	Power Dissipation	—	W	$PD=P_{INTERNAL}+P_{I/O}$ (3)
TH05	$P_{INTERNAL}$	Internal Power Dissipation	—	W	$P_{INTERNAL}=I_{DD}xV_{DD}$ (1)
TH06	$P_{I/O}$	I/O Power Dissipation	—	W	$P_{I/O}=\Sigma(I_{OL}*V_{OL})+\Sigma(I_{OH}*(V_{DD}-V_{OH}))$
TH07	P_{DER}	Derated Power	—	W	$P_{DER}=PD_{MAX}\ (T_J-T_A)/\theta_{JA}$ (2)

Note:

1. I_{DD} is current to run the chip alone without driving any load on the output pins.
2. T_A = Ambient Temperature, T_J = Junction Temperature.
3. See *"Absolute Maximum Ratings"* for total power dissipation.

38.4 AC Characteristics

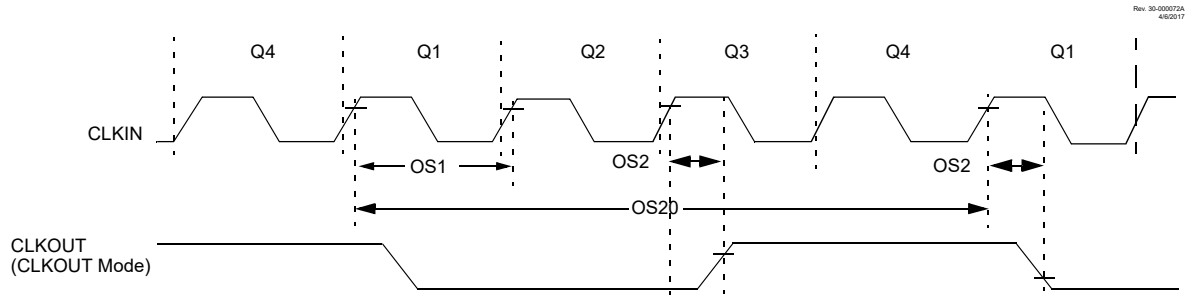
Figure 38-3. Load Conditions



Legend: CL=50 pF for all pins

38.4.1 External Clock/Oscillator Timing Requirements

Figure 38-4. Clock Timing



Note: See table below.

Table 38-7.

Standard Operating Conditions (unless otherwise stated)							
Param No.	**Sym.**	**Characteristic**	**Min.**	**Typ. †**	**Max.**	**Units**	**Conditions**
ECL Oscillator							
OS1	F_{ECL}	Clock Frequency	—	—	1	MHz	
OS2	T_{ECL_DC}	Clock Duty Cycle	40	—	60	%	
ECM Oscillator							
OS3	F_{ECM}	Clock Frequency	—	—	16	MHz	
OS4	T_{ECM_DC}	Clock Duty Cycle	40	—	60	%	
ECH Oscillator							
OS5	F_{ECH}	Clock Frequency	—	—	64	MHz	
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	**Note 4**
System Oscillator							
OS20	F_{OSC}	System Clock Frequency	—	—	64	MHz	(**Note 2, Note 3)**
OS21	F_{CY}	Instruction Frequency	—	$F_{OSC}/4$	—	MHz	
OS22	T_{CY}	Instruction Period	62.5	$1/F_{CY}$	—	ns	

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 (FOSC) is selected by the “main clock switch controls” as described in the *“Power Saving Operation Modes”* section.

Standard Operating Conditions (unless otherwise stated)							
Param No.	Sym.	Characteristic	Min.	Typ. †	Max.	Units	Conditions

3. The system clock frequency (FOSC) must meet the voltage requirements defined in the *"Standard Operating Conditions"* section.
4. LP, XT and HS oscillator modes require an appropriate crystal or resonator to be connected to the device. For clocking the device with the external square wave, one of the EC mode selections must be used.

Related Links

38.2 Standard Operating Conditions
6. Power-Saving Operation Modes

38.4.2 Internal Oscillator Parameters(1)

Table 38-8.

Standard Operating Conditions (unless otherwise stated)							
Param No.	Sym.	Characteristic	Min.	Typ. †	Max.	Units	Conditions
OS50	F_{HFOSC}	Precision Calibrated HFINTOSC Frequency	—	4 8 12 16 32 48 64	—	MHz	(**Note 2**)
OS51	$F_{HFOSCLP}$	Low-Power Optimized HFINTOSC Frequency	— —	1 2	— —	MHz MHz	Fundamental Freq.1 MHz Fundamental Freq. 2 MHz
OS52	F_{MFOSC}	Internal Calibrated MFINTOSC Frequency	—	500	—	kHz	
OS53*	F_{LFOSC}	Internal LFINTOSC Frequency	—	31	—	kHz	
OS54*	$T_{HFOSCST}$	HFINTOSC Wake-up from Sleep Start-up Time	— —	11 100	20 —	µs µs	VREGPM=0x VREGPM=1x

Standard Operating Conditions (unless otherwise stated)							
Param No.	Sym.	Characteristic	Min.	Typ. †	Max.	Units	Conditions
OS56	$T_{LFOSCST}$	LFINTOSC Wake-up from Sleep Start-up Time	—	0.2	—	ms	

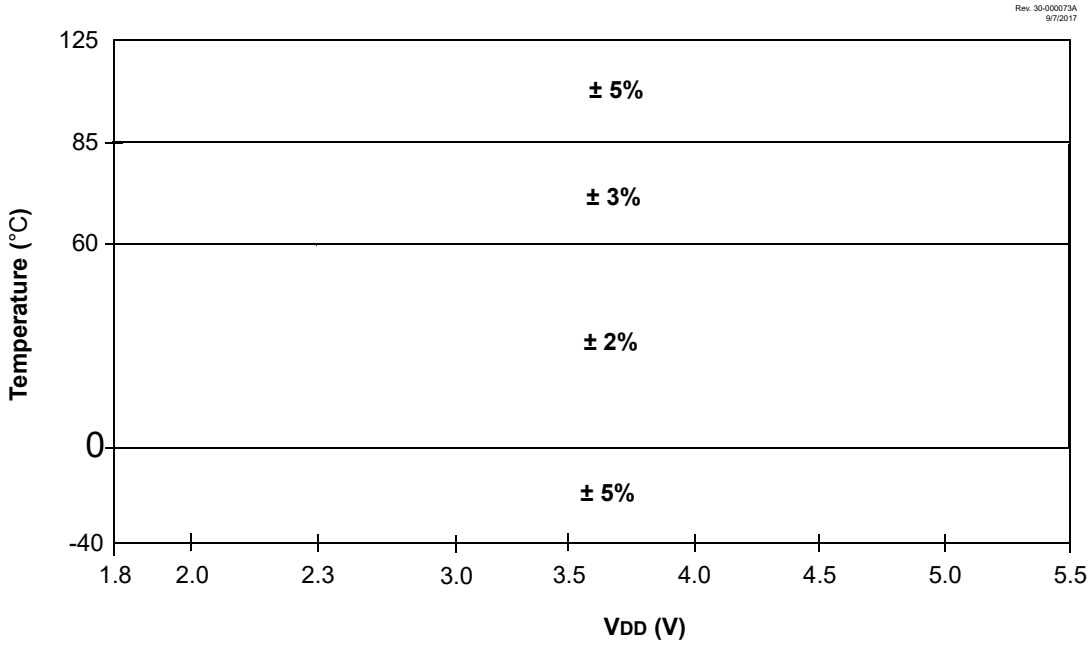
* - These parameters are characterized but not tested.

† 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, V_{DD} and V_{SS} must be capacitively decoupled as close to the device as possible. 0.1 µF and 0.01 µF values in parallel are recommended.
2. See the figure below.

Figure 38-5. Precision Calibrated HFINTOSC Frequency Accuracy Over Device V_{DD} and Temperature



38.4.3 PLL Specifications

Table 38-9.

Standard Operating Conditions (unless otherwise stated)							
Param No.	**Sym.**	**Characteristic**	**Min.**	**Typ. †**	**Max.**	**Units**	**Conditions**
PLL01	F_{PLLIN}	PLL Input Frequency Range	4	—	16	MHz	
PLL02	F_{PLLOUT}	PLL Output Frequency Range	16	—	64	MHz	(**Note 1**)
PLL03	F_{PLLST}	PLL Lock Time from Start-up	—	10	—	µs	
PLL04	F_{PLLJIT}	PLL Output Frequency Stability (Jitter)	-0.25	—	0.25	%	

* - These parameters are characterized but not tested.

† Data in “Typ” column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

Note:

1. The output frequency of the PLL must meet the F_{OSC} requirements listed in Parameter D002.

38.4.4 I/O and CLKOUT Timing Specifications

Figure 38-6. CLKOUT and I/O Timing

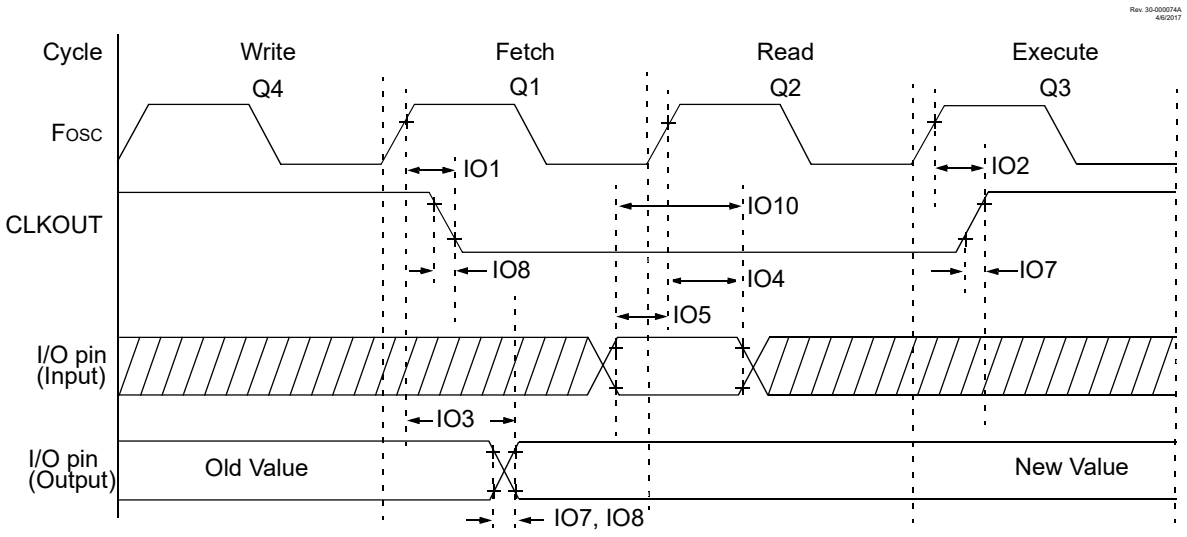
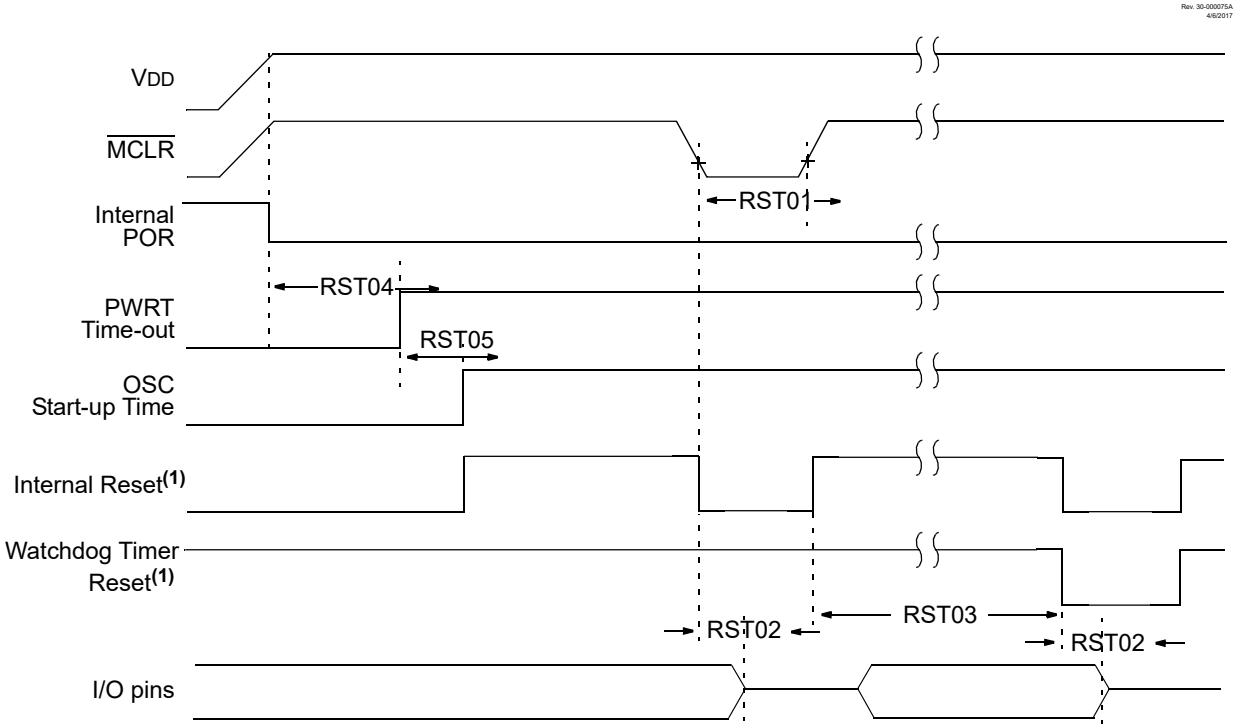


Table 38-10. I/O and CLKOUT Timing Specifications

Standard Operating Conditions (unless otherwise stated)							
Param No.	**Sym.**	**Characteristic**	**Min.**	**Typ. †**	**Max.**	**Units**	**Conditions**
IO1*	$T_{CLKOUTH}$	CLKOUT rising edge delay (rising edge F_{OSC} (Q1 cycle) to falling edge CLKOUT	—	—	70	ns	
IO2*	$T_{CLKOUTL}$	CLKOUT falling edge delay (rising edge F_{OSC} (Q3 cycle) to rising edge CLKOUT	—	—	72	ns	
IO3*	T_{IO_VALID}	Port output valid time (rising edge F_{OSC} (Q1 cycle) to port valid)	—	50	70	ns	
IO4*	T_{IO_SETUP}	Port input setup time (Setup time before rising edge F_{OSC} – Q2 cycle)	20	—	—	ns	
IO5*	T_{IO_HOLD}	Port input hold time (Hold time after rising edge F_{OSC} – Q2 cycle)	50	—	—	ns	
IO6*	T_{IOR_SLREN}	Port I/O rise time, slew rate enabled	—	25	—	ns	V_{DD}=3.0V
IO7*	T_{IOR_SLRDIS}	Port I/O rise time, slew rate disabled	—	5	—	ns	V_{DD}=3.0V
IO8*	T_{IOF_SLREN}	Port I/O fall time, slew rate enabled	—	25	—	ns	V_{DD}=3.0V
IO9*	T_{IOF_SLRDIS}	Port I/O fall time, slew rate disabled	—	5	—	ns	V_{DD}=3.0V
IO10*	T_{INT}	INT pin high or low time to trigger an interrupt	25	—	—	ns	
IO11*	T_{IOC}	Interrupt-on-Change minimum high or low time to trigger interrupt	25	—	—	ns	
* - These parameters are characterized but not tested.							

38.4.5 Reset, WDT, Oscillator Start-up Timer, Power-up Timer, Brown-Out Reset and Low-Power Brown-Out Reset Specifications

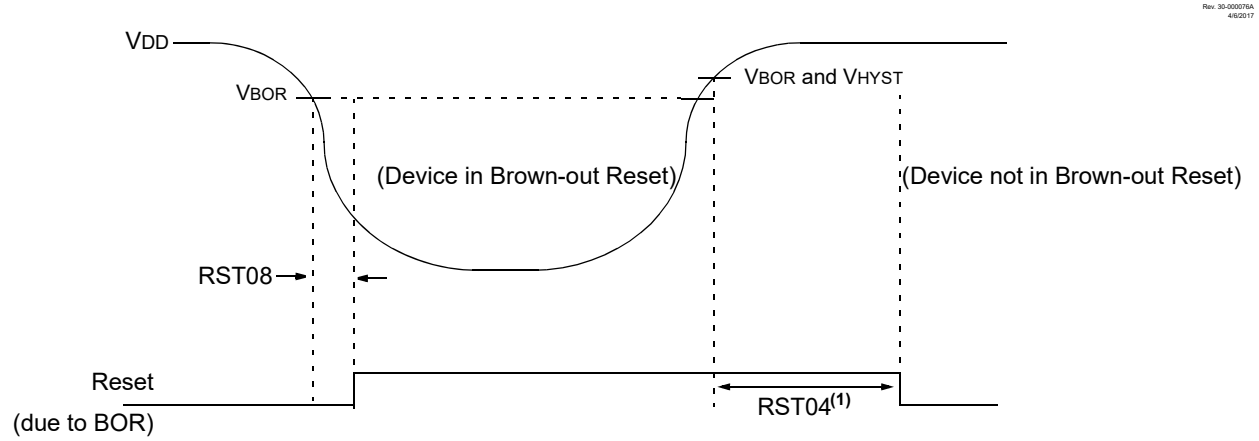
Figure 38-7. Reset, Watchdog Timer, Oscillator Start-up Timer and Power-up Timer Timing



Note:

1. Asserted low.

Figure 38-8. Brown-out Reset Timing and Characteristics



Note:

1. Only if $\overline{\text{PWRTE}}$ bit in the Configuration Word register is programmed to '1'; 2 ms delay if $\overline{\text{PWRTE}}$ = 0.

Table 38-11.

Standard Operating Conditions (unless otherwise stated)							
Param No.	**Sym.**	**Characteristic**	**Min.**	**Typ. †**	**Max.**	**Units**	**Conditions**
RST01*	T_{MCLR}	$\overline{MCLR}$ Pulse Width Low to ensure Reset	2	—	—	µs	
RST02*	T_{IOZ}	I/O high-impedance from Reset detection	—	—	2	µs	
RST03	T_{WDT}	Watchdog Timer Time-out Period	—	16	—	ms	1:512 Prescaler
RST04*	T_{PWRT}	Power-up Timer Period	—	65	—	ms	
RST05	T_{OST}	Oscillator Start-up Timer Period(1,2)	—	1024	—	T_{OSC}	
RST06	V_{BOR}	Brown-out Reset Voltage	2.7	2.85	3.0	V	BORV=00
			2.55	2.7	2.85	V	BORV=01
			2.3	2.45	2.6	V	BORV=10
			1.8	1.9	2.05	V	BORV=11
RST07	V_{BORHYS}	Brown-out Reset Hysteresis	—	40	—	mV	
RST08	T_{BORDC}	Brown-out Reset Response Time	—	3	—	µs	
RST09	V_{LPBOR}	Low-Power Brown-out Reset Voltage	—	1.9	—	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.
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.

38.4.6 High/Low-Voltage Detect Characteristics

Table 38-12.

Standard Operating Conditions (unless otherwise stated)							
Param No.	**Sym.**	**Characteristic**	**Min.**	**Typ.**	**Max.**	**Units**	**Conditions**
HLVD01	V_{DET}	Voltage Detect	—	1.90	—	V	HLVDSEL=b'0000'
			—	2.10	—	V	HLVDSEL=b'0001'
			—	2.25	—	V	HLVDSEL=b'0010'
			—	2.50	—	V	HLVDSEL=b'0011'
			—	2.60	—	V	HLVDSEL=b'0100'

Standard Operating Conditions (unless otherwise stated)							
Param No.	Sym.	Characteristic	Min.	Typ.	Max.	Units	Conditions
			—	2.75	—	V	HLVDSEL=b'0101'
			—	2.90	—	V	HLVDSEL=b'0110'
			—	3.15	—	V	HLVDSEL=b'0111'
			—	3.35	—	V	HLVDSEL=b'1000'
			—	3.60	—	V	HLVDSEL=b'1001'
			—	3.75	—	V	HLVDSEL=b'1010'
			—	4.00	—	V	HLVDSEL=b'1011'
			—	4.20	—	V	HLVDSEL=b'1100'
			—	4.35	—	V	HLVDSEL=b'1101'
			—	4.65	—	V	HLVDSEL=b'1110'

38.4.7 Analog-To-Digital Converter (ADC) Accuracy Specifications(1,2)

Table 38-13.

Standard 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	N_R	Resolution	—	—	10	bit	
AD02	E_{IL}	Integral Error	—	±0.1	±1.0	LSb	ADC_{REF}+=3.0V, ADC_{REF}- = 0V
AD03	E_{DL}	Differential Error	—	±0.1	±1.0	LSb	ADC_{REF}+=3.0V, ADC_{REF}- = 0V
AD04	E_{OFF}	Offset Error	—	0.5	±2.0	LSb	ADC_{REF}+=3.0V, ADC_{REF}- = 0V
AD05	E_{GN}	Gain Error	—	±0.2	±1.0	LSb	ADC_{REF}+=3.0V, ADC_{REF}- = 0V
AD06	V_{ADREF}	ADC Reference Voltage (AD_{REF}+ - AD_{REF}-)	1.8	—	V_{DD}	V	
AD07	V_{AIN}	Full-Scale Range	AD_{REF}-	—	AD_{REF}+	V	
AD08	Z_{AIN}	Recommended Impedance of Analog Voltage Source	—	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.

Standard 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**

Note:

1. Total Absolute Error is the sum of the offset, gain and integral non-linearity (INL) errors.
2. The ADC conversion result never decreases with an increase in the input and has no missing codes.
3. This is the impedance seen by the V_{REF} pads when the external reference pads are selected.

38.4.8 Analog-to-Digital Converter (ADC) Conversion Timing Specifications

Table 38-14.

Standard Operating Conditions (unless otherwise stated)							
Param No.	**Sym.**	**Characteristic**	**Min.**	**Typ. †**	**Max.**	**Units**	**Conditions**
AD20	T_{AD}	ADC Clock Period	1	—	9	µs	Using F_{OSC} as the ADC clock source ADOCS = 0
AD21			1	2	6	µs	Using F_{RC} as the ADC clock source ADOCS = 1
AD22	T_{CNV}	Conversion Time(1)	—	11+3T_{CY}	—	T_{AD}	Set of GO/$\overline{\text{DONE}}$ bit to Clear of GO/ $\overline{\text{DONE}}$ bit
AD23	T_{ACQ}	Acquisition Time	—	2	—	µs	
AD24	T_{HCD}	Sample and Hold Capacitor Disconnect Time	—	—	—	µs	F_{OSC}-based clock source F_{RC}-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 38-9. ADC Conversion Timing (ADC Clock F_{OSC}-Based)

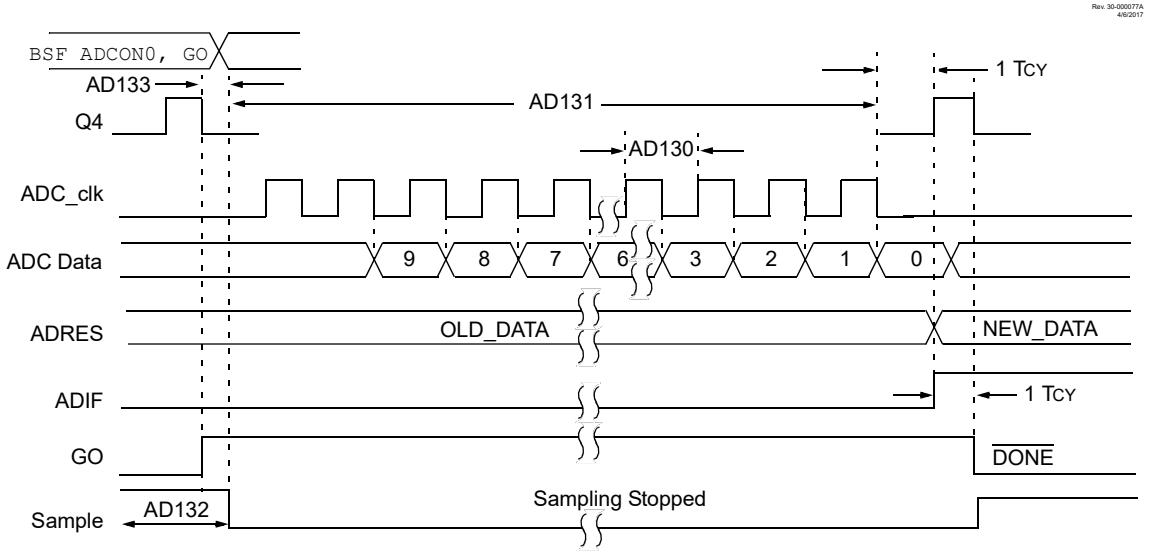
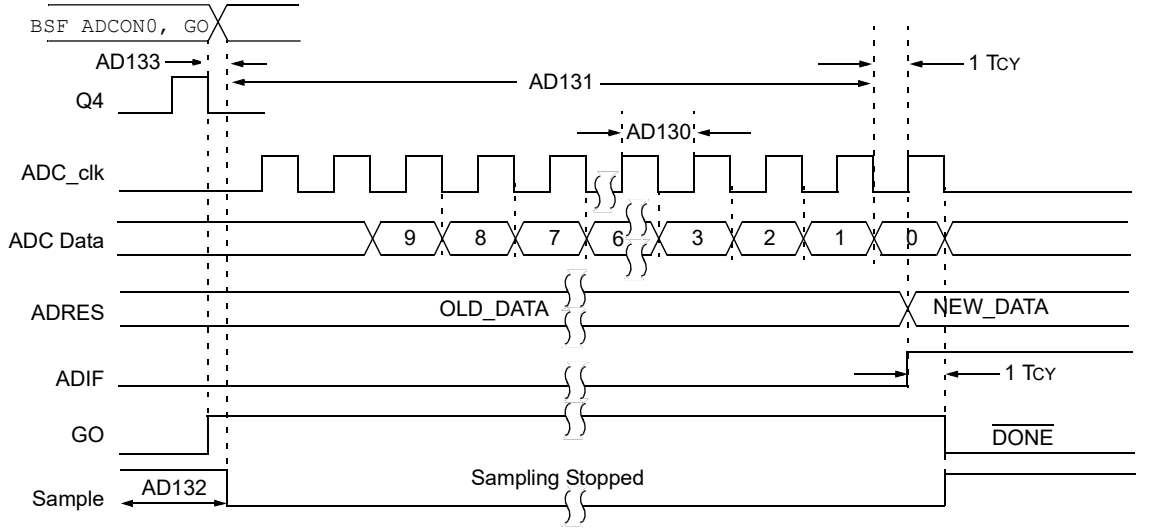


Figure 38-10. ADC Conversion Timing (ADC Clock from ADCRC)



Note:

1. If the ADC clock source is selected as ADCRC, a time of T_{CY} is added before the ADC clock starts. This allows the `SLEEP` instruction to be executed.

38.4.9 Comparator Specifications

Table 38-15.

Standard Operating Conditions (unless otherwise stated) V_{DD} = 3.0V, T_A = 25°C							
Param No.	**Sym.**	**Characteristic**	**Min.**	**Typ. †**	**Max.**	**Units**	**Conditions**
CM01	V_{IOFF}	Input Offset Voltage	—	—	±60	mV	$V_{ICM}=V_{DD}/2$
CM02	V_{ICM}	Input Common Mode Range	GND	—	V_{DD}	V	
CM03	CMRR	Common Mode Input Rejection Ratio	—	45	—	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.

† Data in “Typ” column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

Note:

1. Response time measured with one comparator input at $V_{DD}/2$, while the other input transitions from V_{SS} to V_{DD}.

38.4.10 5-Bit DAC Specifications

Table 38-16.

Standard Operating Conditions (unless otherwise stated) V_{DD} = 3.0V, T_A = 25°C							
Param No.	**Sym.**	**Characteristic**	**Min.**	**Typ. †**	**Max.**	**Units**	**Conditions**
DSB01	V_{LSB}	Step Size	—	$(V_{DACREF+}-V_{DACREF-})/32$	—	V	
DSB02	V_{ACC}	Absolute Accuracy	—	—	±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.

Standard Operating Conditions (unless otherwise stated) V_{DD} = 3.0V, T_A = 25°C							
Param No.	**Sym.**	**Characteristic**	**Min.**	**Typ. †**	**Max.**	**Units**	**Conditions**
† 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'.							

38.4.11 Fixed Voltage Reference (FVR) Specifications

Table 38-17.

Standard Operating Conditions (unless otherwise stated)							
Param No.	**Sym.**	**Characteristic**	**Min.**	**Typ. †**	**Max.**	**Units**	**Conditions**
FVR01	$V_{FVR}1$	1x Gain (1.024V)	-4	—	+4	%	VDD≥2.5V, -40°C to 85°C
FVR02	$V_{FVR}2$	2x Gain (2.048V)	-4	—	+4	%	VDD≥2.5V, -40°C to 85°C
FVR03	$V_{FVR}4$	4x Gain (4.096V)	-4	—	+4	%	VDD≥4.75V, -40°C to 85°C
FVR04	T_{FVRST}	FVR Start-up Time	—	25	—	µs	

38.4.12 Zero-Cross Detect (ZCD) Specifications

Table 38-18.

Standard Operating Conditions (unless otherwise stated) V_{DD} = 3.0V, T_A = 25°C							
Param No.	**Sym.**	**Characteristic**	**Min.**	**Typ. †**	**Max.**	**Units**	**Conditions**
ZC01	V_{PINZC}	Voltage on Zero Cross Pin	—	0.75	—	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.							

38.4.13 Timer0 and Timer1 External Clock Requirements

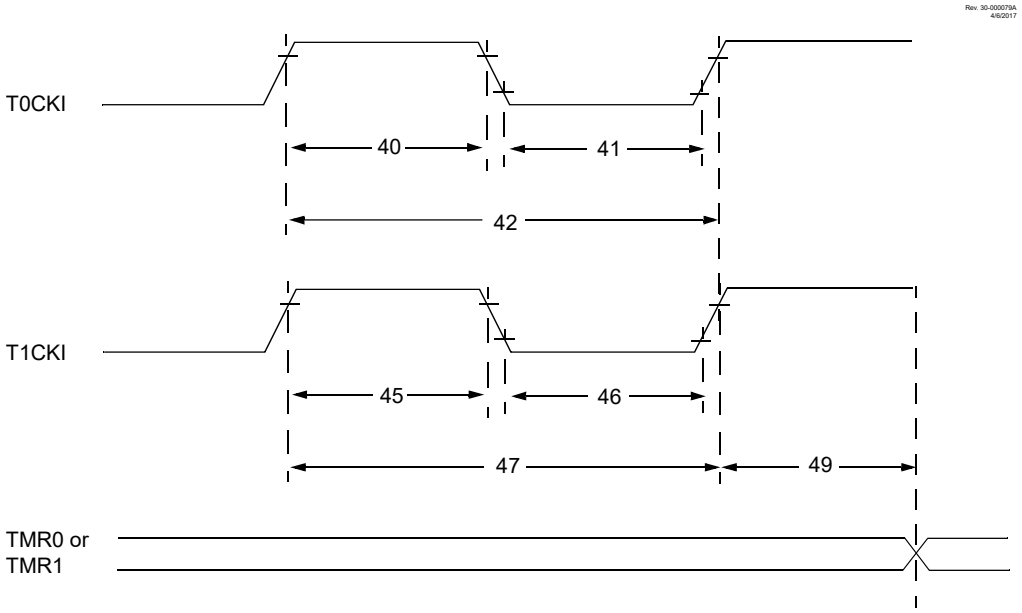
Table 38-19.

Standard Operating Conditions (unless otherwise stated) Operating Temperature: -40°C≤T_A≤+125°C								
Param No.	**Sym.**	**Characteristic**		**Min.**	**Typ. †**	**Max.**	**Units**	**Conditions**
40*	T_T0H	T0CKI High Pulse Width	No Prescaler	$0.5T_{CY}$ +20	—	—	ns	
			With Prescaler	10	—	—	ns	
41*	T_T0L	T0CKI Low Pulse Width	No Prescaler	$0.5T_{CY}$ +20	—	—	ns	
			With Prescaler	10	—	—	ns	
42*	T_T0P	T0CKI Period		Greater of: 20 or (T_{CY} +40)/N	—	—	ns	N = Prescale value
45*	T_T1H	T1CKI High Time	Synchronous, No Prescaler	$0.5T_{CY}$ +20	—	—	ns	
			Synchronous, with Prescaler	15	—	—	ns	
			Asynchronous	30	—	—	ns	
46*	T_T1L	T1CKI Low Time	Synchronous, No Prescaler	$0.5T_{CY}$ +20	—	—	ns	
			Synchronous, with Prescaler	15	—	—	ns	
			Asynchronous	30	—	—	ns	
47*	T_T1P	T1CKI Input Period	Synchronous	Greater of: 30 or (T_{CY} +40)/N	—	—	ns	N = Prescale value
			Asynchronous	60	—	—	ns	
49*	$TCKEZ_{TMR}1$	Delay from External Clock Edge to Timer Increment		2 T_{OSC}	—	7 T_{OSC}	—	Timers in Sync mode

* - These parameters are characterized but not tested.

† Data in “Typ” column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

Figure 38-11. Timer0 and Timing1 External Clock Timings



38.4.14 Capture/Compare/PWM Requirements (CCP)

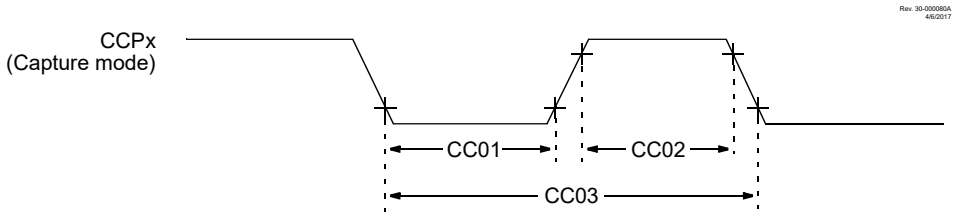
Table 38-20.

Standard Operating Conditions (unless otherwise stated) Operating Temperature: -40°C≤T_A≤+125°C								
Param No.	**Sym.**	**Characteristic**		**Min.**	**Typ. †**	**Max.**	**Units**	**Conditions**
CC01*	$T_{CC}L$	CCPx Input Low Time	No Prescaler	$0.5T_{CY}$+20	—	—	ns	
			With Prescaler	20	—	—	ns	
CC02*	$T_{CC}H$	CCPx Input High Time	No Prescaler	$0.5T_{CY}$+20	—	—	ns	
			With Prescaler	20	—	—	ns	
CC03*	$T_{CC}P$	CCPx Input Period		($3T_{CY}$ +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 38-12. Capture/Compare/PWM Timings (CCP)



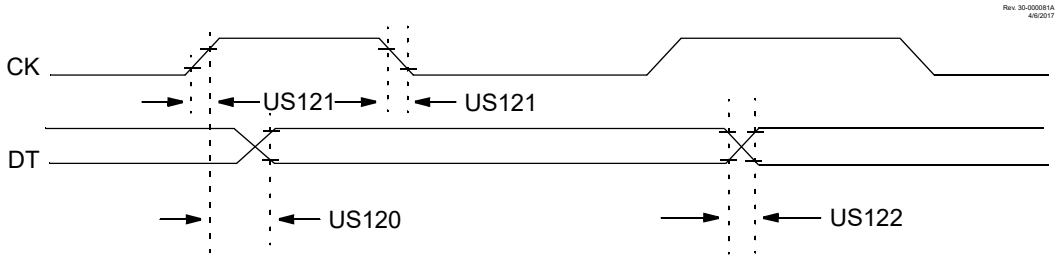
Note: Refer to Figure 38-3 for load conditions.

38.4.15 EUSART Synchronous Transmission Requirements

Table 38-21.

Standard Operating Conditions (unless otherwise stated)						
Param No.	**Sym.**	**Characteristic**	**Min.**	**Max.**	**Units**	**Conditions**
US120	$T_{CK}H2_{DT}V$	SYNC XMIT (Master and Slave)	—	80	ns	$3.0V \le V_{DD} \le 5.5V$
		Clock high to data-out valid	—	100	ns	$1.8V \le V_{DD} \le 5.5V$
US121	T_{CKRF}	Clock out rise time and fall time	—	45	ns	$3.0V \le V_{DD} \le 5.5V$
		(Master mode)	—	50	ns	$1.8V \le V_{DD} \le 5.5V$
US122	T_{DTRF}	Data-out rise time and fall time	—	45	ns	$3.0V \le V_{DD} \le 5.5V$
			—	50	ns	$1.8V \le V_{DD} \le 5.5V$

Figure 38-13. EUSART Synchronous Transmission (Master/Slave) Timing



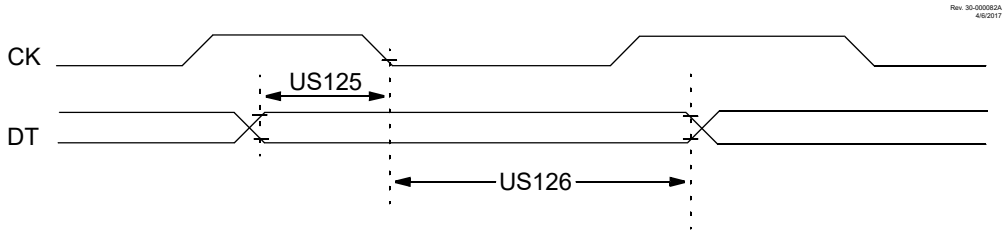
Note: Refer to Figure 38-3 for load conditions.

38.4.16 EUSART Synchronous Receive Requirements

Table 38-22.

Standard Operating Conditions (unless otherwise stated)						
Param No.	**Sym.**	**Characteristic**	**Min.**	**Max.**	**Units**	**Conditions**
US125	$T_{DT}V2_{CKL}$	SYNC RCV (Master and Slave)	10	—	ns	
		Data-setup before CK ↓ (DT hold time)				
US126	$T_{CK}L2_{DTL}$	Data-hold after CK ↓ (DT hold time)	15	—	ns	

Figure 38-14. EUSART Synchronous Receive (Master/Slave) Timing



Note: Refer to Figure 38-3 for load conditions.

38.4.17 SPI Mode Requirements

Table 38-23.

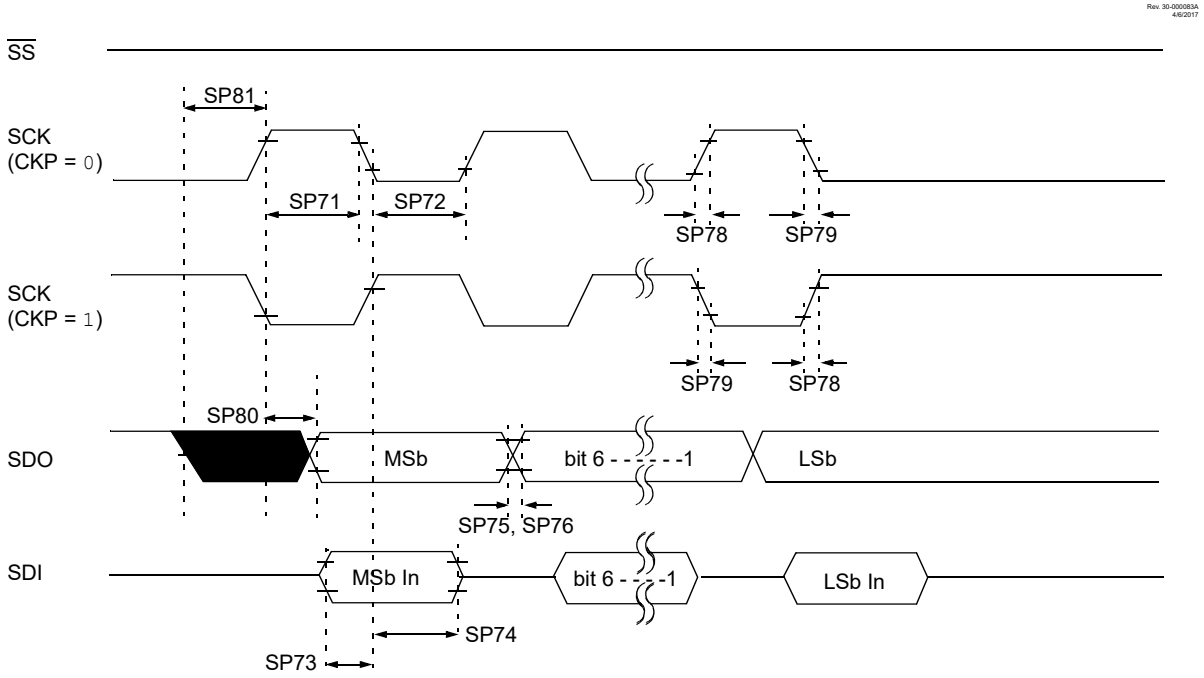
Standard Operating Conditions (unless otherwise stated)							
Param No.	**Sym.**	**Characteristic**	**Min.**	**Typ. †**	**Max.**	**Units**	**Conditions**
SP70*	$T_{SS}L2_{SC}H$, $T_{SS}L2_{SC}L$	$\overline{SS}$↓ to SCK↓ or SCK↑ input	2.25*T_{CY}	—	—	ns	
SP71*	$T_{SC}H$	SCK input high time (Slave mode)	T_{CY} + 20	—	—	ns	
SP72*	$T_{SC}L$	SCK input low time (Slave mode)	T_{CY} + 20	—	—	ns	
SP73*	$T_{DI}V2_{SC}H$, $T_{DI}V2_{SC}L$	Setup time of SDI data input to SCK edge	100	—	—	ns	
SP74*	$T_{SC}H2_{DI}L$, $T_{SC}L2_{DI}L$	Hold time of SDI data input to SCK edge	100	—	—	ns	
SP75*	$T_{DO}R$	SDO data output rise time	—	10	25	ns	3.0V≤V_{DD}≤5.5V
			—	25	50	ns	1.8V≤V_{DD}≤5.5V
SP76*	$T_{DO}F$	SDO data output fall time	—	10	25	ns	
SP77*	$T_{SS}H2_{DO}Z$	$\overline{SS}$↑ to SDO output high-impedance	10	—	50	ns	
SP78*	$T_{SC}R$	SCK output rise time (Master mode)	—	10	25	ns	3.0V≤V_{DD}≤5.5V
			—	25	50	ns	1.8V≤V_{DD}≤5.5V
SP79*	$T_{SC}F$	SCK output fall time (Master mode)	—	10	25	ns	
SP80*	$T_{SC}H2_{DO}V$, $T_{SC}L2_{DO}V$	SDO data output valid after SCK edge	—	—	50	ns	3.0V≤V_{DD}≤5.5V
			—	—	145	ns	1.8V≤V_{DD}≤5.5V

Standard Operating Conditions (unless otherwise stated)							
Param No.	Sym.	Characteristic	Min.	Typ. †	Max.	Units	Conditions
SP81*	$T_{DO}V2_{SC}H$, $T_{DO}V2_{SC}L$	SDO data output setup to SCK edge	1 T_{CY}	—	—	ns	
SP82*	$T_{SS}L2_{DO}V$	SDO data output valid after $\overline{SS}\downarrow$ edge	—	—	50	ns	
SP83*	$T_{SC}H2_{SS}H$, $T_{SC}L2_{SS}H$	$\overline{SS}\uparrow$ after SCK edge	1.5 T_{CY} + 40	—	—	ns	

* - These parameters are characterized but not tested.

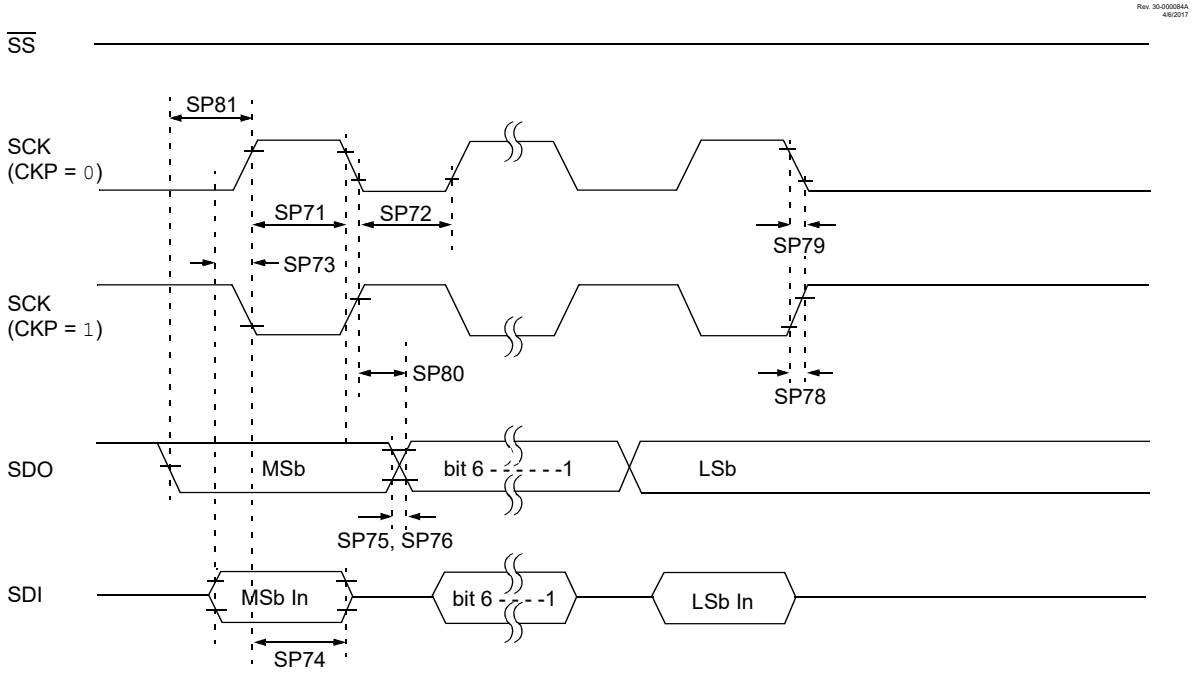
† Data in “Typ” column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

Figure 38-15. SPI Master Mode Timing (CKE = 0, SMP = 0)



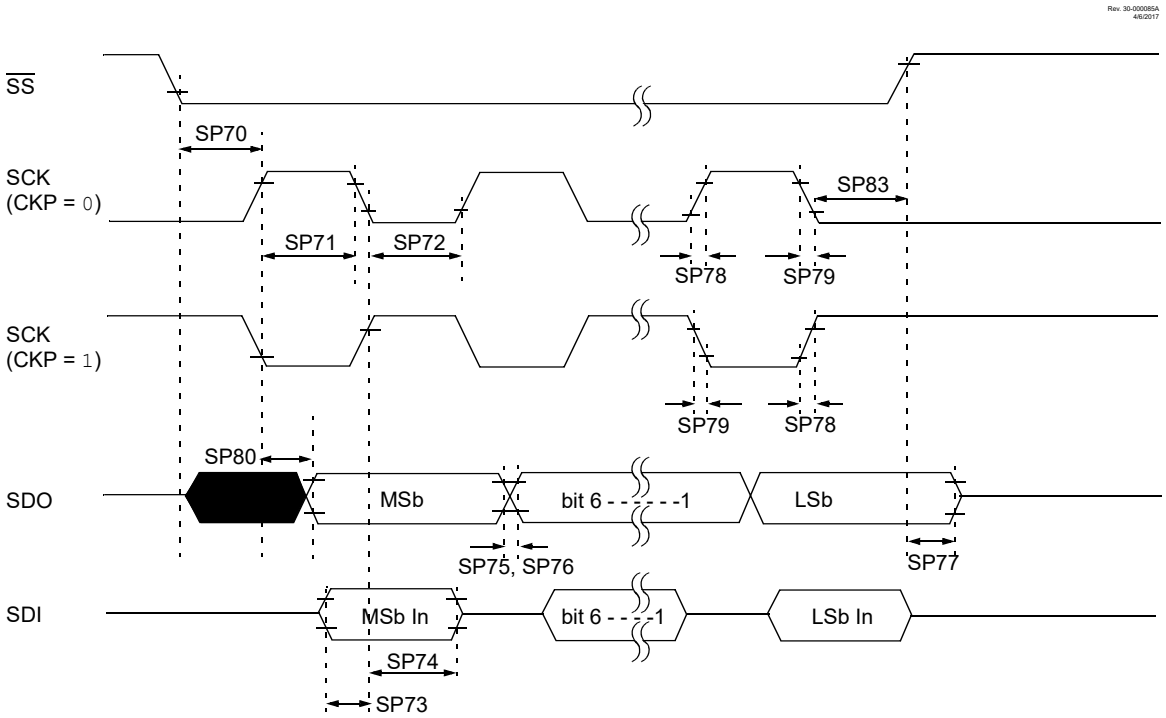
Note: Refer to Figure 38-3 for load conditions.

Figure 38-16. SPI Master Mode Timing (CKE = 1, SMP = 1)



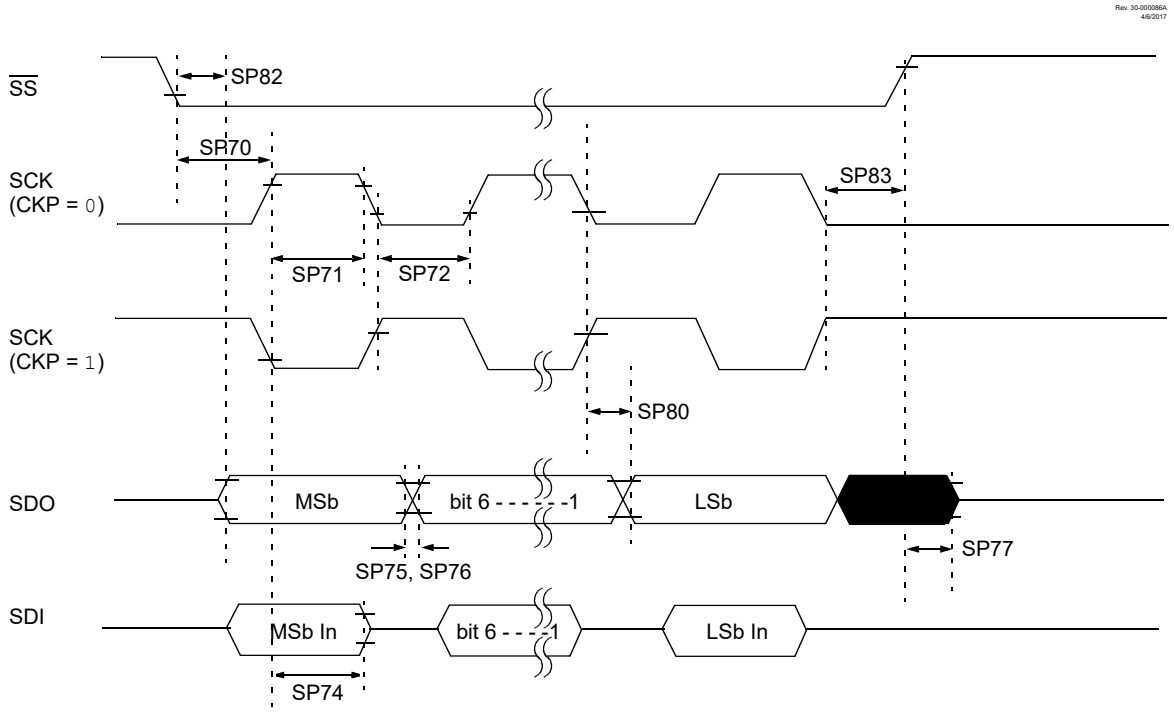
Note: Refer to Figure 38-3 for load conditions.

Figure 38-17. SPI Slave Mode Timing (CKE = 0)



Note: Refer to Figure 38-3 for load conditions.

Figure 38-18. SPI Slave Mode Timing (CKE = 1)



Note: Refer to Figure 38-3 for load conditions.

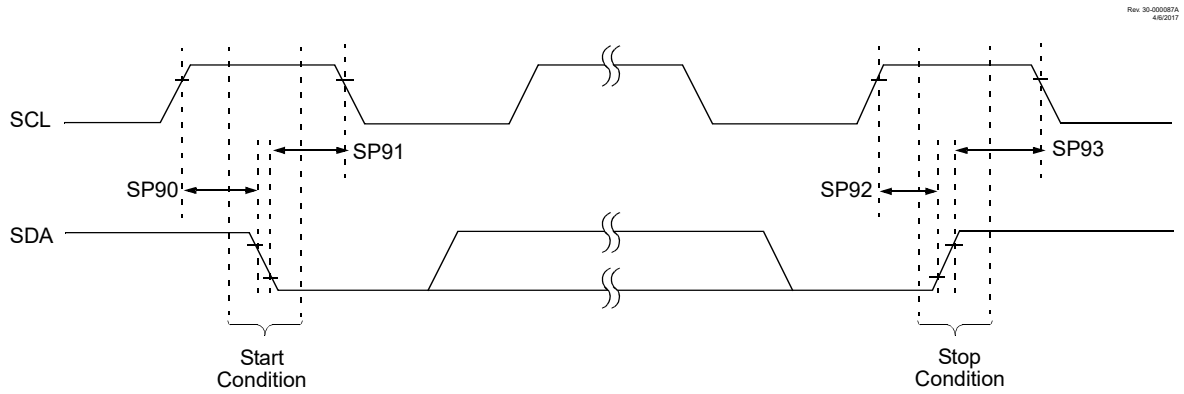
38.4.18 I²C Bus Start/Stop Bits Requirements

Table 38-24.

Standard Operating Conditions (unless otherwise stated)								
Param. No.	**Sym.**	**Characteristic**		**Min.**	**Typ. †**	**Max.**	**Units**	**Conditions**
SP90*	$T_{SU:STA}$	Start condition Setup time	100 kHz mode	4700	—	—	ns	Only relevant for Repeated Start Setup time 400 kHz mode 600 condition
			400 kHz mode	600	—	—		
SP91*	$T_{HD:STA}$	Start condition Hold time	100 kHz mode	4000	—	—	ns	After this period, the first clock Hold time 400 kHz mode 600 — — pulse is generated
			400 kHz mode	600	—	—		
SP92*	$T_{SU:STO}$	Stop condition Setup time	100 kHz mode	4700	—	—	ns	
			400 kHz mode	600	—	—		
SP93*	$T_{HD:STO}$	Stop condition Hold time	100 kHz mode	4000	—	—	ns	

Standard Operating Conditions (unless otherwise stated)								
Param. No.	Sym.	Characteristic		Min.	Typ. †	Max.	Units	Conditions
			400 kHz mode	600	—	—		
* - These parameters are characterized but not tested.								

Figure 38-19. I²C Bus Start/Stop Bits Timing



Note: Refer to Figure 38-3 for load conditions.

38.4.19 I²C Bus Data Requirements

Table 38-25.

Standard Operating Conditions (unless otherwise stated)							
Param. No.	Sym.	Characteristic		Min.	Max.	Units	Conditions
SP100*	T_{HIGH}	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*	T_{LOW}	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

Standard Operating Conditions (unless otherwise stated)							
Param. No.	**Sym.**	**Characteristic**		**Min.**	**Max.**	**Units**	**Conditions**
			SSP module	$1.5T_{CY}$	—		
SP102*	T_R	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*	T_F	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*	$T_{HD:DAT}$	Data input hold time	100 kHz mode	0	—	ns	
			400 kHz mode	0	0.9	µs	
SP107*	$T_{SU:DAT}$	Data input setup time	100 kHz mode	250	—	ns	(**Note 2**)
			400 kHz mode	100	—	ns	
SP109*	T_{AA}	Output valid from clock	100 kHz mode	—	3500	ns	(**Note 1**)
			400 kHz mode	—	—	ns	
SP110*	T_{BUF}	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		—		pF	

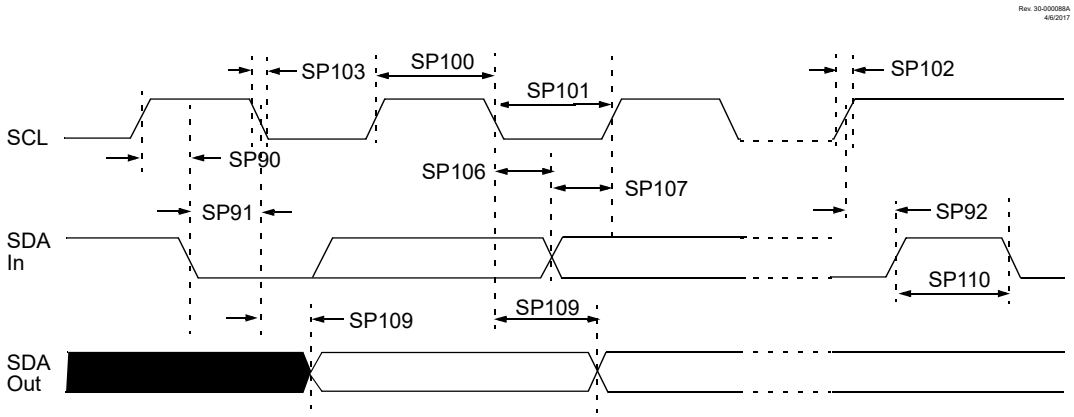
* - These parameters are characterized but not tested.

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) I^2C bus device can be used in a Standard mode (100 kHz) I^2C bus system, but the requirement $T_{SU:DAT} \geq 250$ ns must then be met. This will automatically be the case

Standard Operating Conditions (unless otherwise stated)						
Param. No.	Sym.	Characteristic	Min.	Max.	Units	Conditions
		if the device does not stretch the low period of the SCL signal. If such a device does stretch the low period of the SCL signal, it must output the next data bit to the SDA line TR max. + $T_{SU:DAT}$ = 1000 + 250 = 1250 ns (according to the Standard mode I^2C bus specification), before the SCL line is released.				

Figure 38-20. I^2C Bus Data Timing



Note: Refer to Figure 38-3 for load conditions.

39. DC and AC Characteristics Graphs and Tables

Graphs and Tables Graphs and tables are not available at this time.

40. Packaging Information

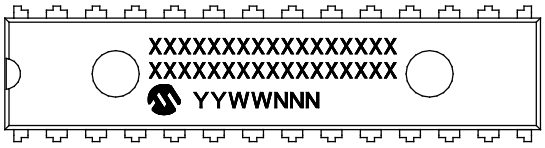
Package Marking Information

Rev. 30-009000A
5/17/2017

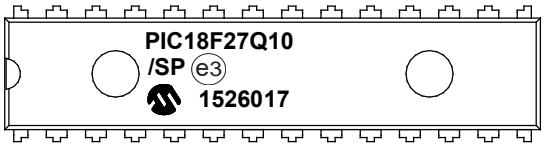
Legend:	XX...X	Customer-specific information or Microchip part number
	Y	Year code (last digit of calendar year)
	YY	Year code (last 2 digits of calendar year)
	WW	Week code (week of January 1 is week ‘01’)
	NNN	Alphanumeric traceability code
	(e3)	b-free JEDEC ® designator for Matte Tin (Sn)
	*	This 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.	

Rev. 30-009028A
5/17/2017

28-Lead SPDIP (.300”)

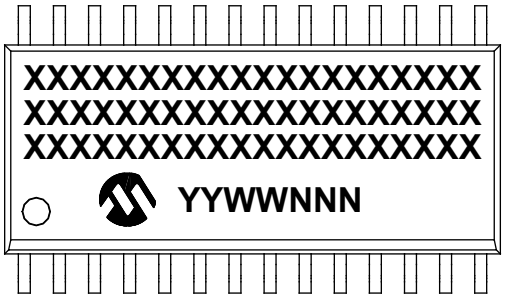


Example

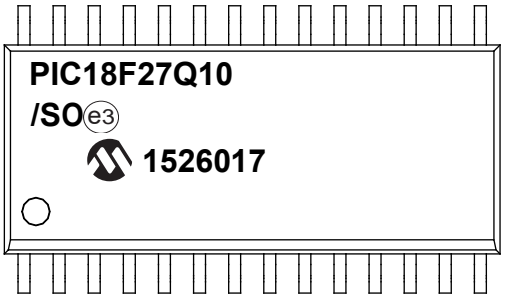


Rev. 30-009028B
5/17/2017

28-Lead SOIC (7.50 mm)

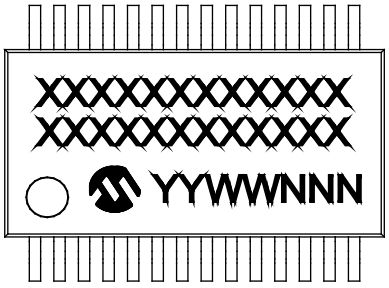


Example

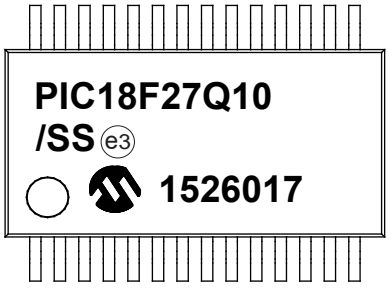


Rev. 30-009028C5/17/2017

28-Lead SSOP (5.30 mm)

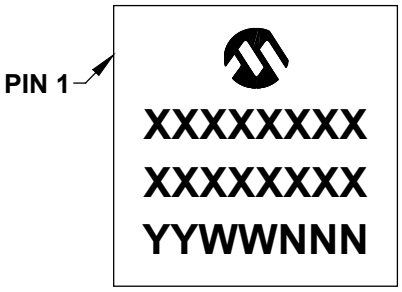


Example

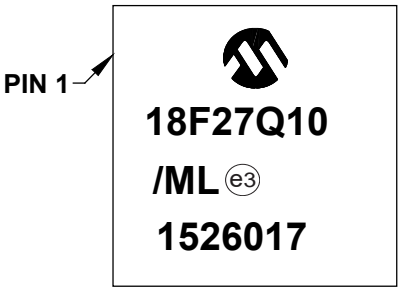


Rev. 30-009028D
5/17/2017

28-Lead QFN (6x6 mm)

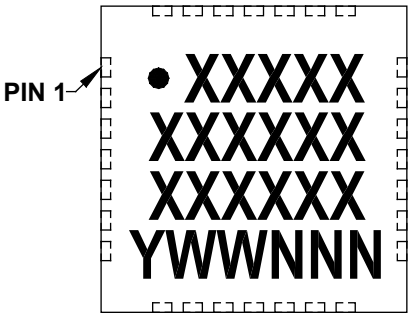


Example

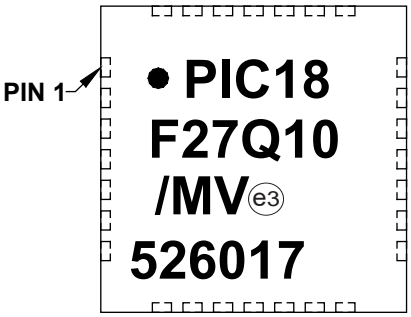


Rev. 30-009028F4/2/2018

28-Lead VQFN (4x4x1 mm)



Example

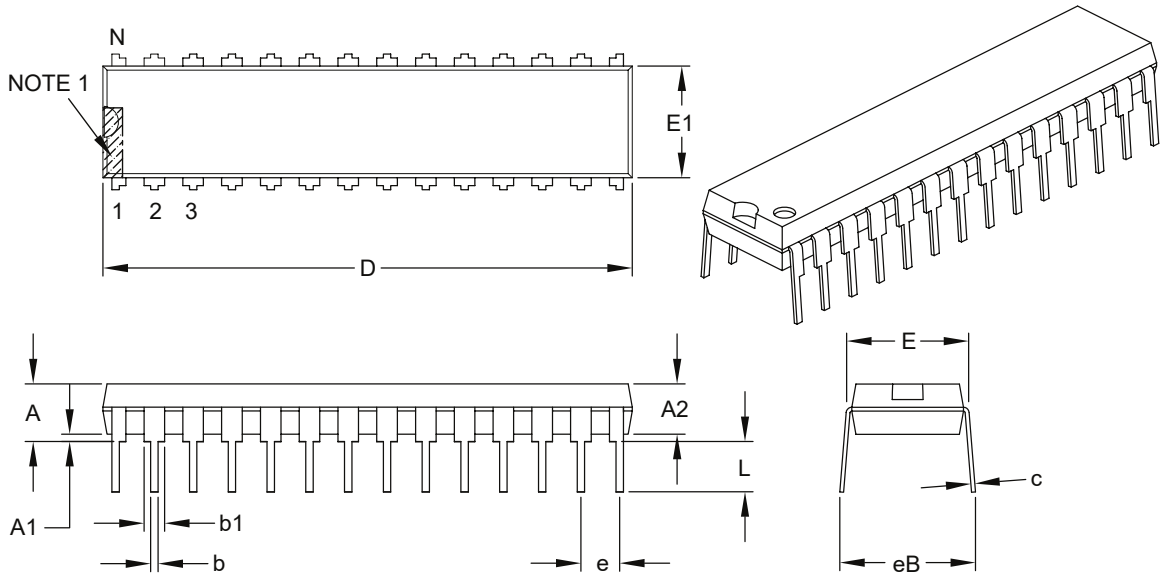


40.1 Package Details

The following sections give the technical details of the packages.

28-Lead Skinny Plastic Dual In-Line (SP) – 300 mil Body [SPDIP]

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



Units		INCHES		
Dimension Limits		MIN	NOM	MAX
Number of Pins	N		28	
Pitch	e		.100 BSC	
Top to Seating Plane	A	–	–	.200
Molded Package Thickness	A2	.120	.135	.150
Base to Seating Plane	A1	.015	–	–
Shoulder to Shoulder Width	E	.290	.310	.335
Molded Package Width	E1	.240	.285	.295
Overall Length	D	1.345	1.365	1.400
Tip to Seating Plane	L	.110	.130	.150
Lead Thickness	c	.008	.010	.015
Upper Lead Width	b1	.040	.050	.070
Lower Lead Width	b	.014	.018	.022
Overall Row Spacing §	eB	–	–	.430

Notes:

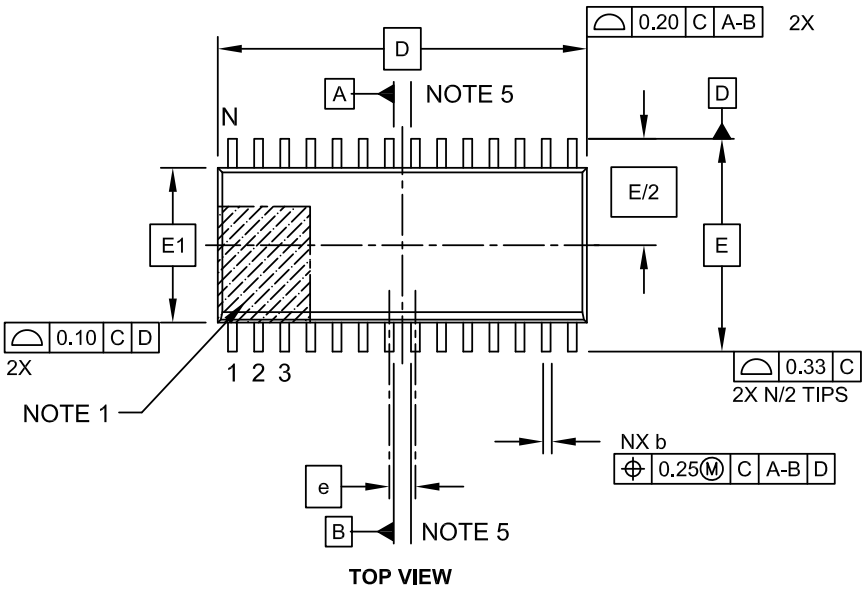
1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. § Significant Characteristic.
3. Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010" per side.
4. Dimensioning and tolerancing per ASME Y14.5M.

   BSC: Basic Dimension. Theoretically exact value shown without tolerances.

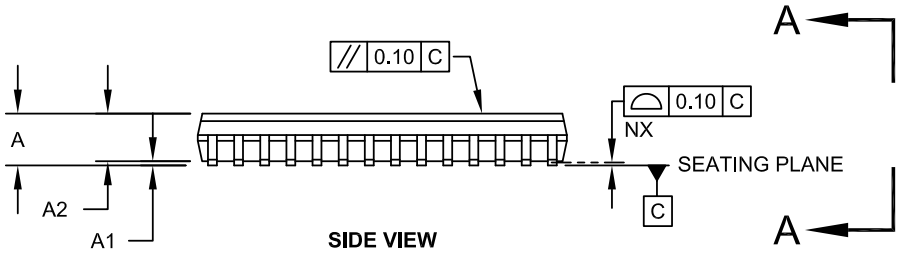
Microchip Technology Drawing C04-070B

28-Lead Plastic Small Outline (SO) - Wide, 7.50 mm Body [SOIC]

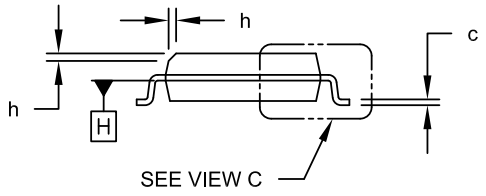
Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



TOP VIEW



SIDE VIEW

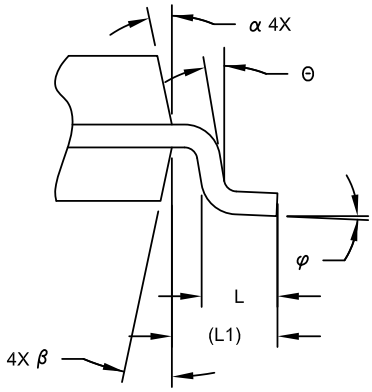


VIEW A-A

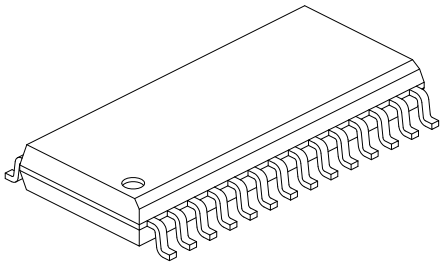
Microchip Technology Drawing C04-052C Sheet 1 of 2

28-Lead Plastic Small Outline (SO) - Wide, 7.50 mm Body [SOIC]

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



VIEW C



Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Number of Pins	N		28	
Pitch	e		1.27 BSC	
Overall Height	A	-	-	2.65
Molded Package Thickness	A2	2.05	-	-
Standoff §	A1	0.10	-	0.30
Overall Width	E		10.30 BSC	
Molded Package Width	E1		7.50 BSC	
Overall Length	D		17.90 BSC	
Chamfer (Optional)	h	0.25	-	0.75
Foot Length	L	0.40	-	1.27
Footprint	L1		1.40 REF	
Lead Angle	Θ	0°	-	-
Foot Angle	φ	0°	-	8°
Lead Thickness	c	0.18	-	0.33
Lead Width	b	0.31	-	0.51
Mold Draft Angle Top	α	5°	-	15°
Mold Draft Angle Bottom	β	5°	-	15°

Notes:

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. § Significant Characteristic
3. Dimension D does not include mold flash, protrusions or gate burrs, which shall not exceed 0.15 mm per end. Dimension E1 does not include interlead flash or protrusion, which shall not exceed 0.25 mm per side.
4. Dimensioning and tolerancing per ASME Y14.5M

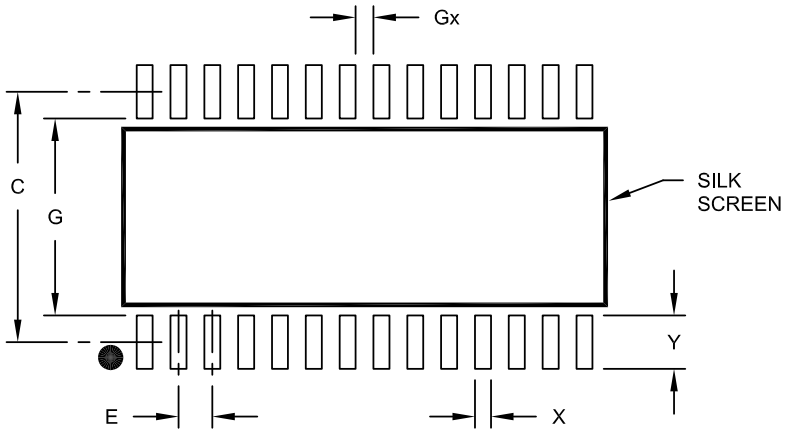
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.

   REF: Reference Dimension, usually without tolerance, for information purposes only.
5. Datums A & B to be determined at Datum H.

Microchip Technology Drawing C04-052C Sheet 2 of 2

28-Lead Plastic Small Outline (SO) - Wide, 7.50 mm Body [SOIC]

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



RECOMMENDED LAND PATTERN

Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Contact Pitch	E		1.27 BSC	
Contact Pad Spacing	C		9.40	
Contact Pad Width (X28)	X			0.60
Contact Pad Length (X28)	Y			2.00
Distance Between Pads	Gx	0.67		
Distance Between Pads	G	7.40		

Notes:

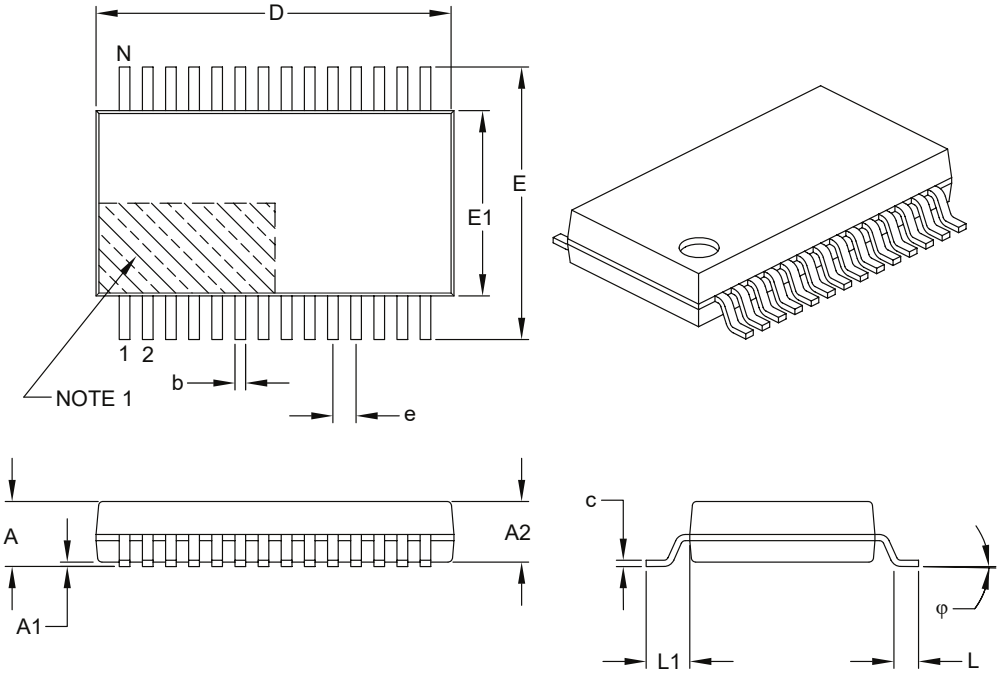
1. Dimensioning and tolerancing per ASME Y14.5M

BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2052A

28-Lead Plastic Shrink Small Outline (SS) – 5.30 mm Body [SSOP]

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Number of Pins	N		28	
Pitch	e		0.65 BSC	
Overall Height	A	–	–	2.00
Molded Package Thickness	A2	1.65	1.75	1.85
Standoff	A1	0.05	–	–
Overall Width	E	7.40	7.80	8.20
Molded Package Width	E1	5.00	5.30	5.60
Overall Length	D	9.90	10.20	10.50
Foot Length	L	0.55	0.75	0.95
Footprint	L1		1.25 REF	
Lead Thickness	c	0.09	–	0.25
Foot Angle	φ	0°	4°	8°
Lead Width	b	0.22	–	0.38

Notes:

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed 0.20 mm per side.
3. Dimensioning and tolerancing per ASME Y14.5M.

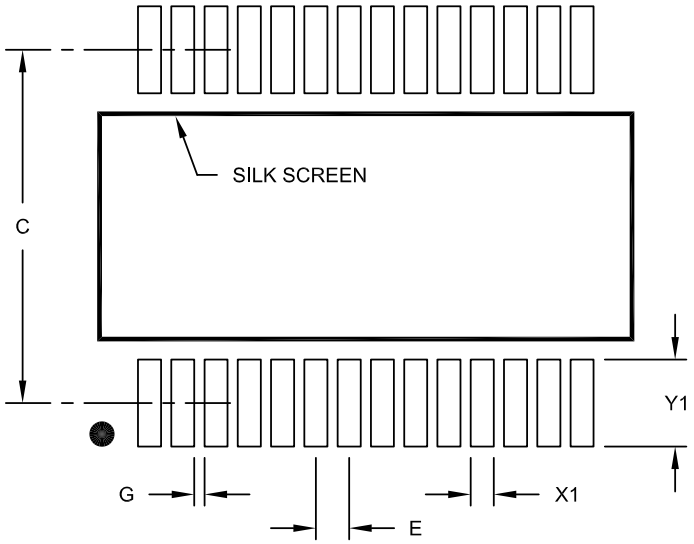
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.

   REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-073B

28-Lead Plastic Shrink Small Outline (SS) - 5.30 mm Body [SSOP]

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



RECOMMENDED LAND PATTERN

Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Contact Pitch	E	0.65 BSC		
Contact Pad Spacing	C		7.20	
Contact Pad Width (X28)	X1			0.45
Contact Pad Length (X28)	Y1			1.75
Distance Between Pads	G	0.20		

Notes:

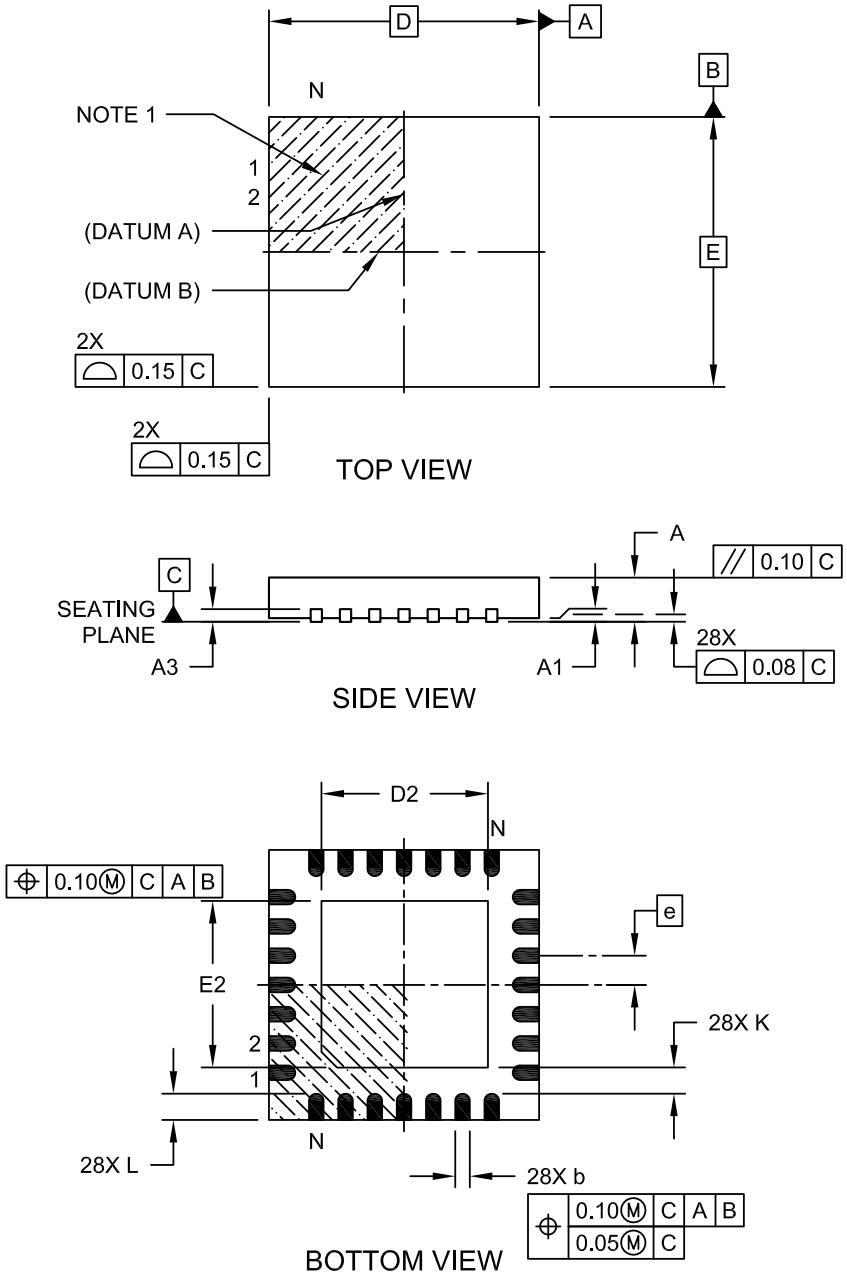
1. Dimensioning and tolerancing per ASME Y14.5M

BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2073A

28-Lead Plastic Quad Flat, No Lead Package (ML) - 6x6 mm Body [QFN]
With 0.55 mm Terminal Length

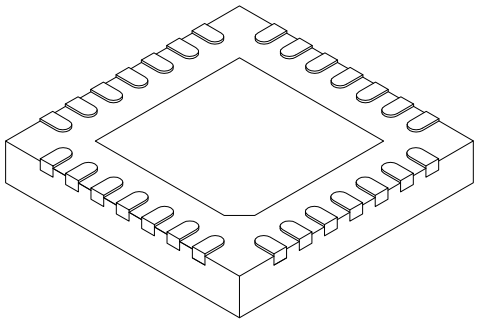
Note:	For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



Microchip Technology Drawing C04-105C Sheet 1 of 2

28-Lead Plastic Quad Flat, No Lead Package (ML) - 6x6 mm Body [QFN]
With 0.55 mm Terminal Length

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



	Units	MILLIMETERS		
Dimension	Limits	MIN	NOM	MAX
Number of Pins	N		28	
Pitch	e		0.65 BSC	
Overall Height	A	0.80	0.90	1.00
Standoff	A1	0.00	0.02	0.05
Terminal Thickness	A3		0.20 REF	
Overall Width	E		6.00 BSC	
Exposed Pad Width	E2	3.65	3.70	4.20
Overall Length	D		6.00 BSC	
Exposed Pad Length	D2	3.65	3.70	4.20
Terminal Width	b	0.23	0.30	0.35
Terminal Length	L	0.50	0.55	0.70
Terminal-to-Exposed Pad	K	0.20	-	-

Notes:

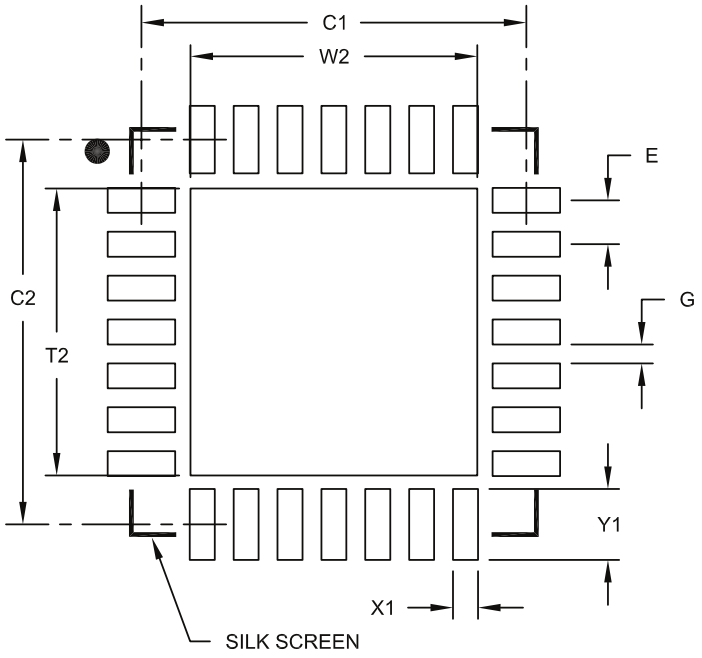
1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Package is saw singulated
3. Dimensioning and tolerancing per ASME Y14.5M.

BSC: Basic Dimension. Theoretically exact value shown without tolerances.
REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-105C Sheet 2 of 2

28-Lead Plastic Quad Flat, No Lead Package (ML) – 6x6 mm Body [QFN] with 0.55 mm Contact Length

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



RECOMMENDED LAND PATTERN

Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Contact Pitch	E		0.65 BSC	
Optional Center Pad Width	W2			4.25
Optional Center Pad Length	T2			4.25
Contact Pad Spacing	C1		5.70	
Contact Pad Spacing	C2		5.70	
Contact Pad Width (X28)	X1			0.37
Contact Pad Length (X28)	Y1			1.00
Distance Between Pads	G	0.20		

Notes:

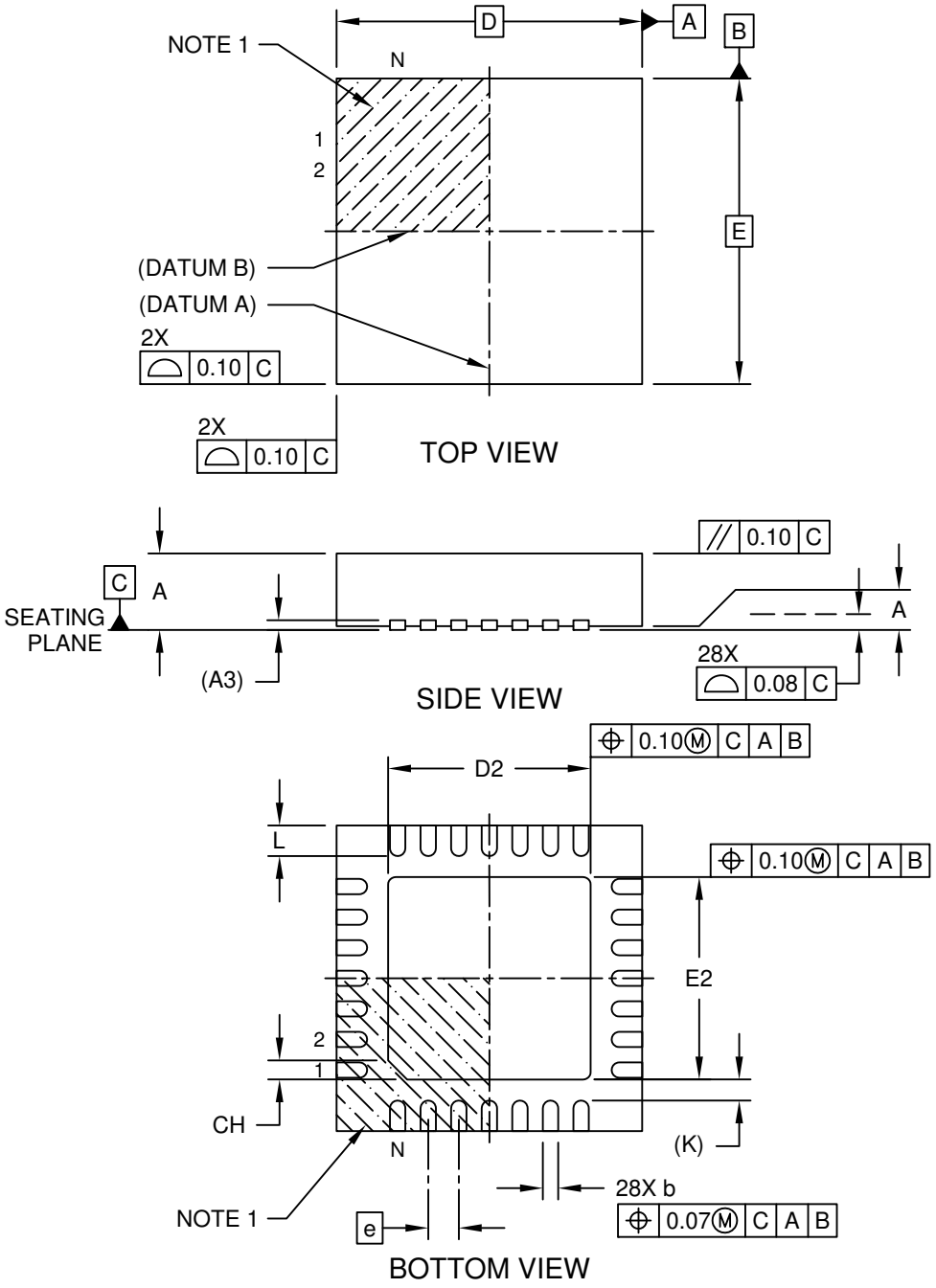
1. Dimensioning and tolerancing per ASME Y14.5M

BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing No. C04-2105A

28-Lead Very Thin Plastic Quad Flat, No Lead (STX) - 4x4 mm Body [VQFN] With 2.65x2.65 mm Exposed Pad

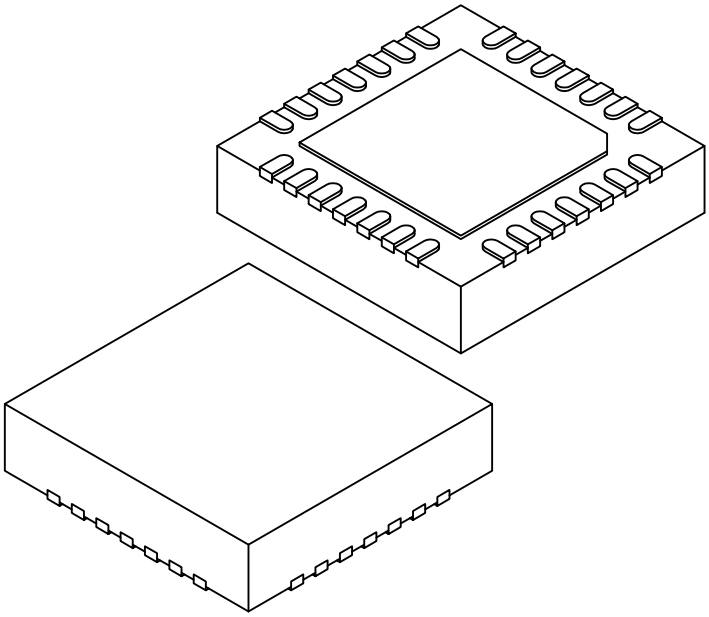
Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



Microchip Technology Drawing C04-456 Rev A Sheet 1 of 2

28-Lead Very Thin Plastic Quad Flat, No Lead (STX) - 4x4 mm Body [VQFN] With 2.65x2.65 mm Exposed Pad

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Number of Terminals	N		28	
Pitch	e		0.40 BSC	
Overall Height	A	0.80	0.90	1.00
Standoff	A1	0.00	0.02	0.05
Terminal Thickness	A3		0.127 REF	
Overall Length	D		4.00 BSC	
Exposed Pad Length	D2	2.55	2.65	2.75
Overall Width	E		4.00 BSC	
Exposed Pad Width	E2	2.55	2.65	2.75
Exposed Pad Corner Chamfer	CH	-	0.25	-
Terminal Width	b	0.15	0.20	0.25
Terminal Length	L	0.30	0.40	0.50
Terminal-to-Exposed-Pad	K		0.275 REF	

Notes:

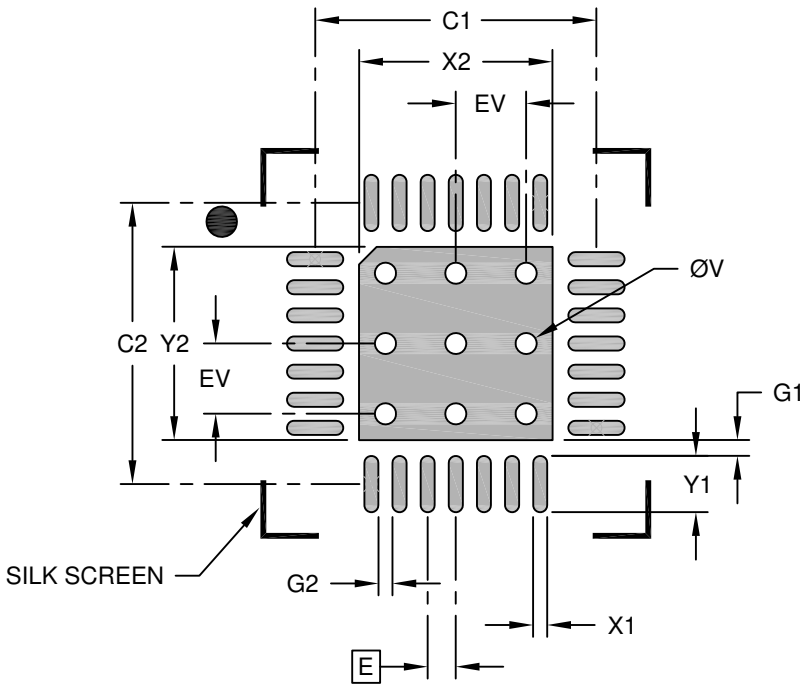
1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Package is saw singulated
3. Dimensioning and tolerancing per ASME Y14.5M

BSC: Basic Dimension. Theoretically exact value shown without tolerances.
REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-456 Rev A Sheet 2 of 2

28-Lead Very Thin Plastic Quad Flat, No Lead (STX) - 4x4 mm Body [VQFN] With 2.65x2.65 mm Exposed Pad

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



RECOMMENDED LAND PATTERN

Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Contact Pitch	E		0.40 BSC	
Optional Center Pad Width	X2			2.75
Optional Center Pad Length	Y2			2.75
Contact Pad Spacing	C1		4.00	
Contact Pad Spacing	C2		4.00	
Contact Pad Width (X28)	X1			0.20
Contact Pad Length (X28)	Y1			0.80
Contact Pad to Center Pad (X28)	G1	0.23		
Contact Pad to Contact Pad (X24)	G2	0.20		
Thermal Via Diameter	V		0.30	
Thermal Via Pitch	EV		1.00	

Notes:

1. Dimensioning and tolerancing per ASME Y14.5M
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.
2. For best soldering results, thermal vias, if used, should be filled or tented to avoid solder loss during reflow process

Microchip Technology Drawing C04-2456 Rev A

41. Revision History

Doc Rev.	Date	Comments
A	01/2018	Initial document release.
B	04/2018	Paragraph clarifications: 3.1, 3.2, 3.4, 10.4, 10.4.3 (example), 10.4.3.1, 10.4.3.3, 11.1, 11.1.4 (note), 11.1.4.1, 15.2.6 (related links), 17.3 (note), 33.10.2; Figure updates: 10-9, 10-10, 11-9; Table updates: 38-2, 38-3, 38-7, 38-9, 39-17; Replaced UQFN package with VQFN.

The Microchip Web Site

Microchip provides online support via our web site at http://www.microchip.com/. This web site is used as a means to make files and information easily available to customers. Accessible by using your favorite Internet browser, the web site 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 web site at http://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

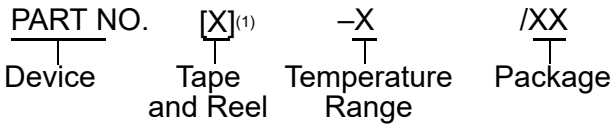
Customers should contact their distributor, representative or Field Application Engineer (FAE) for support. Local sales offices are also available to help customers. A listing of sales offices and locations is included in the back of this document.

Technical support is available through the web site at: http://www.microchip.com/support

© 2018 Microchip Technology Inc.

Product Identification System

To order or obtain information, e.g., on pricing or delivery, refer to the factory or the listed sales office.



Device:	PIC18F24Q10, PIC18F25Q10	
Tape & Reel Option:	Blank	= Tube
	T	= Tape & Reel
Temperature Range:	I	= -40°C to +85°C (Industrial)
	E	= -40°C to +125°C (Extended)
Package:	ML	= 28-lead QFN 6x6mm
	STX	= 28-lead VQFN 4x4x1mm
	SO	= 8-lead SOIC
	SP	= 28-lead Skinny Plastic DIP
	SS	= 28-lead SSOP

Examples:

- PIC18F24Q10-E/P 301: Extended temp., PDIP package, QTP pattern #301.
- PIC18F25Q10-E/SO = Extended temp., SOIC package.
- PIC18F24Q10T-I/ML = Tape and reel, Industrial temp., QFN package.

Note:

1. Tape and Reel identifier only appears in the catalog part number description. This identifier is used for ordering purposes and is not printed on the device package. Check with your Microchip Sales Office for package availability with the Tape and Reel option.
2. Small form-factor packaging options may be available. Please check http://www.microchip.com/packaging for small-form factor package availability, or contact your local Sales Office.

Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip 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.

© 2018 Microchip Technology Inc.

- 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.

Legal Notice

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.

Trademarks

The Microchip name and logo, the Microchip logo, AnyRate, AVR, AVR logo, AVR Freaks, BeaconThings, BitCloud, CryptoMemory, CryptoRF, dsPIC, FlashFlex, flexPWR, Heldo, JukeBlox, KeeLoq, KeeLoq logo, Kleer, LANCheck, LINK MD, maXStylus, maXTouch, MediaLB, megaAVR, MOST, MOST logo, MPLAB, OptoLyzer, PIC, picoPower, PICSTART, PIC32 logo, Prochip Designer, QTouch, RightTouch, SAM-BA, SpyNIC, SST, SST Logo, SuperFlash, tinyAVR, UNI/O, and XMEGA 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.

Adjacent Key Suppression, AKS, Analog-for-the-Digital Age, Any Capacitor, AnyIn, AnyOut, BodyCom, chipKIT, chipKIT logo, CodeGuard, CryptoAuthentication, CryptoCompanion, CryptoController, dsPICDEM, dsPICDEM.net, Dynamic Average Matching, DAM, ECAN, EtherGREEN, In-Circuit Serial Programming, ICSP, Inter-Chip Connectivity, JitterBlocker, KleerNet, KleerNet logo, Mindi, MiWi, motorBench, MPASM, MPF, MPLAB Certified logo, MPLIB, MPLINK, MultiTRAK, NetDetach, Omniscient Code Generation, PICDEM, PICDEM.net, PICkit, PICtail, PureSilicon, QMatrix, RightTouch logo, REAL ICE, Ripple Blocker, SAM-ICE, Serial Quad I/O, SMART-I.S., SQI, 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.

© 2018, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

ISBN: 978-1-5224-2881-7

Quality Management System Certified by DNV

ISO/TS 16949

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, KEELOQ® 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.

Worldwide Sales and Service

AMERICAS	ASIA/PACIFIC	ASIA/PACIFIC	EUROPE
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 **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 Tel: 317-536-2380 **Los Angeles** Mission Viejo, CA Tel: 949-462-9523 Fax: 949-462-9608 Tel: 951-273-7800 **Raleigh, NC** Tel: 919-844-7510 **New York, NY** Tel: 631-435-6000 **San Jose, CA** Tel: 408-735-9110 Tel: 408-436-4270 **Canada - Toronto** Tel: 905-695-1980 Fax: 905-695-2078	**Australia - Sydney** Tel: 61-2-9868-6733 **China - Beijing** Tel: 86-10-8569-7000 **China - Chengdu** Tel: 86-28-8665-5511 **China - Chongqing** Tel: 86-23-8980-9588 **China - Dongguan** Tel: 86-769-8702-9880 **China - Guangzhou** Tel: 86-20-8755-8029 **China - Hangzhou** Tel: 86-571-8792-8115 **China - Hong Kong SAR** Tel: 852-2943-5100 **China - Nanjing** Tel: 86-25-8473-2460 **China - Qingdao** Tel: 86-532-8502-7355 **China - Shanghai** Tel: 86-21-3326-8000 **China - Shenyang** Tel: 86-24-2334-2829 **China - Shenzhen** Tel: 86-755-8864-2200 **China - Suzhou** Tel: 86-186-6233-1526 **China - Wuhan** Tel: 86-27-5980-5300 **China - Xian** Tel: 86-29-8833-7252 **China - Xiamen** Tel: 86-592-2388138 **China - Zhuhai** Tel: 86-756-3210040	**India - Bangalore** Tel: 91-80-3090-4444 **India - New Delhi** Tel: 91-11-4160-8631 **India - Pune** Tel: 91-20-4121-0141 **Japan - Osaka** Tel: 81-6-6152-7160 **Japan - Tokyo** Tel: 81-3-6880- 3770 **Korea - Daegu** Tel: 82-53-744-4301 **Korea - Seoul** Tel: 82-2-554-7200 **Malaysia - Kuala Lumpur** Tel: 60-3-7651-7906 **Malaysia - Penang** Tel: 60-4-227-8870 **Philippines - Manila** Tel: 63-2-634-9065 **Singapore** Tel: 65-6334-8870 **Taiwan - Hsin Chu** Tel: 886-3-577-8366 **Taiwan - Kaohsiung** Tel: 886-7-213-7830 **Taiwan - Taipei** Tel: 886-2-2508-8600 **Thailand - Bangkok** Tel: 66-2-694-1351 **Vietnam - Ho Chi Minh** Tel: 84-28-5448-2100	**Austria - Wels** Tel: 43-7242-2244-39 Fax: 43-7242-2244-393 **Denmark - Copenhagen** Tel: 45-4450-2828 Fax: 45-4485-2829 **Finland - Espoo** Tel: 358-9-4520-820 **France - Paris** Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79 **Germany - Garching** Tel: 49-8931-9700 **Germany - Haan** Tel: 49-2129-3766400 **Germany - Heilbronn** Tel: 49-7131-67-3636 **Germany - Karlsruhe** Tel: 49-721-625370 **Germany - Munich** Tel: 49-89-627-144-0 Fax: 49-89-627-144-44 **Germany - Rosenheim** Tel: 49-8031-354-560 **Israel - Ra'anana** Tel: 972-9-744-7705 **Italy - Milan** Tel: 39-0331-742611 Fax: 39-0331-466781 **Italy - Padova** Tel: 39-049-7625286 **Netherlands - Drunen** Tel: 31-416-690399 Fax: 31-416-690340 **Norway - Trondheim** Tel: 47-7289-7561 **Poland - Warsaw** Tel: 48-22-3325737 **Romania - Bucharest** Tel: 40-21-407-87-50 **Spain - Madrid** Tel: 34-91-708-08-90 Fax: 34-91-708-08-91 **Sweden - Gothenberg** Tel: 46-31-704-60-40 **Sweden - Stockholm** Tel: 46-8-5090-4654 **UK - Wokingham** Tel: 44-118-921-5800 Fax: 44-118-921-5820

© 2018 Microchip Technology Inc.