PIC18F6585/8585/6680/8680 Data Sheet

64/68/80-Pin High-Performance, 64-Kbyte Enhanced Flash Microcontrollers with ECAN Module

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as “unbreakable.”

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip’s code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
ISO/TS 16949:2002

Microchip received ISO/TS-16949:2002 quality system certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona and Mountain View, California in October 2003. The Company’s quality system processes and procedures are for its PICmicro® 8-bit MCUs, 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.

MICROCHIP

PIC18F6585/8585/6680/8680

64/68/80-Pin High-Performance, 64-Kbyte Enhanced Flash Microcontrollers with ECAN Module

High-Performance RISC CPU:

- Source code compatible with the PIC16 and PIC17 instruction sets
- Linear program memory addressing to 2 Mbytes
- Linear data memory addressing to 4096 bytes
- 1 Kbyte of data EEPROM
- Up to 10 MIPs operation:
  - DC – 40 MHz osc./clock input
  - 4 MHz-10 MHz osc./clock input with PLL active
- 16-bit wide instructions, 8-bit wide data path
- Priority levels for interrupts
- 31-level, software accessible hardware stack
- 8 x 8 Single-Cycle Hardware Multiplier

External Memory Interface (PIC18F8X8X Devices Only):

- Address capability of up to 2 Mbytes
- 16-bit interface

Peripheral Features:

- High current sink/source 25 mA/25 mA
- Four external interrupt pins
- Timer0 module: 8-bit/16-bit timer/counter
- Timer1 module: 16-bit timer/counter
- Timer2 module: 8-bit timer/counter
- Timer3 module: 16-bit timer/counter
- Secondary oscillator clock option – Timer1/Timer3
- One Capture/Compare/PWM (CCP) module:
  - Capture is 16-bit, max. resolution 6.25 ns (T_{CY}/16)
  - Compare is 16-bit, max. resolution 100 ns (T_{CY})
  - PWM output: PWM resolution is 1 to 10-bit
- Enhanced Capture/Compare/PWM (ECCP) module:
  - Same Capture/Compare features as CCP
  - One, two or four PWM outputs
  - Selectable polarity
  - Programmable dead time
  - Auto-shutdown on external event
  - Auto-restart
- Master Synchronous Serial Port (MSSP) module with two modes of operation:
  - 3-wire SPI™ (supports all 4 SPI modes)
  - I^2C™ Master and Slave mode
- Enhanced Addressable USART module:
  - Supports RS-232, RS-485 and LIN 1.2
  - Programmable wake-up on Start bit
  - Auto-baud detect
- Parallel Slave Port (PSP) module

Analog Features:

- Up to 16-channel, 10-bit Analog-to-Digital Converter module (A/D) with:
  - Fast sampling rate
  - Programmable acquisition time
  - Conversion available during Sleep
- Programmable 16-level Low-Voltage Detection (LVD) module:
  - Supports interrupt on Low-Voltage Detection
- Programmable Brown-out Reset (BOR)
- Dual analog comparators:
  - Programmable input/output configuration

ECAN Module Features:

- Message bit rates up to 1 Mbps
- Conforms to CAN 2.0B ACTIVE Specification
- Fully backward compatible with PIC18XXX8 CAN modules
- Three modes of operation:
  - Legacy, Enhanced Legacy, FIFO
- Three dedicated transmit buffers with prioritization
- Two dedicated receive buffers
- Six programmable receive/transmit buffers
- Three full 29-bit acceptance masks
- 16 full 29-bit acceptance filters with dynamic association
- DeviceNet™ data byte filter support
- Automatic remote frame handling
- Advanced Error Management features

Special Microcontroller Features:

- 100,000 erase/write cycle Enhanced Flash program memory typical
- 1,000,000 erase/write cycle Data EEPROM memory typical
- 1-second programming time
- Flash/Data EEPROM Retention: > 40 years
- Self-reprogrammable under software control
- Power-on Reset (POR), Power-up Timer (PWRT) and Oscillator Start-up Timer (OST)
- Watchdog Timer (WDT) with its own On-Chip RC Oscillator
- Programmable code protection
- Power saving Sleep mode
- Selectable oscillator options including:
  - Software enabled 4x Phase Lock Loop (of primary oscillator)
  - Secondary Oscillator (32 kHz) clock input
- In-Circuit Serial Programming™ (ICSP™) via two pins
- MPLAB® In-Circuit Debug (ICD) via two pins

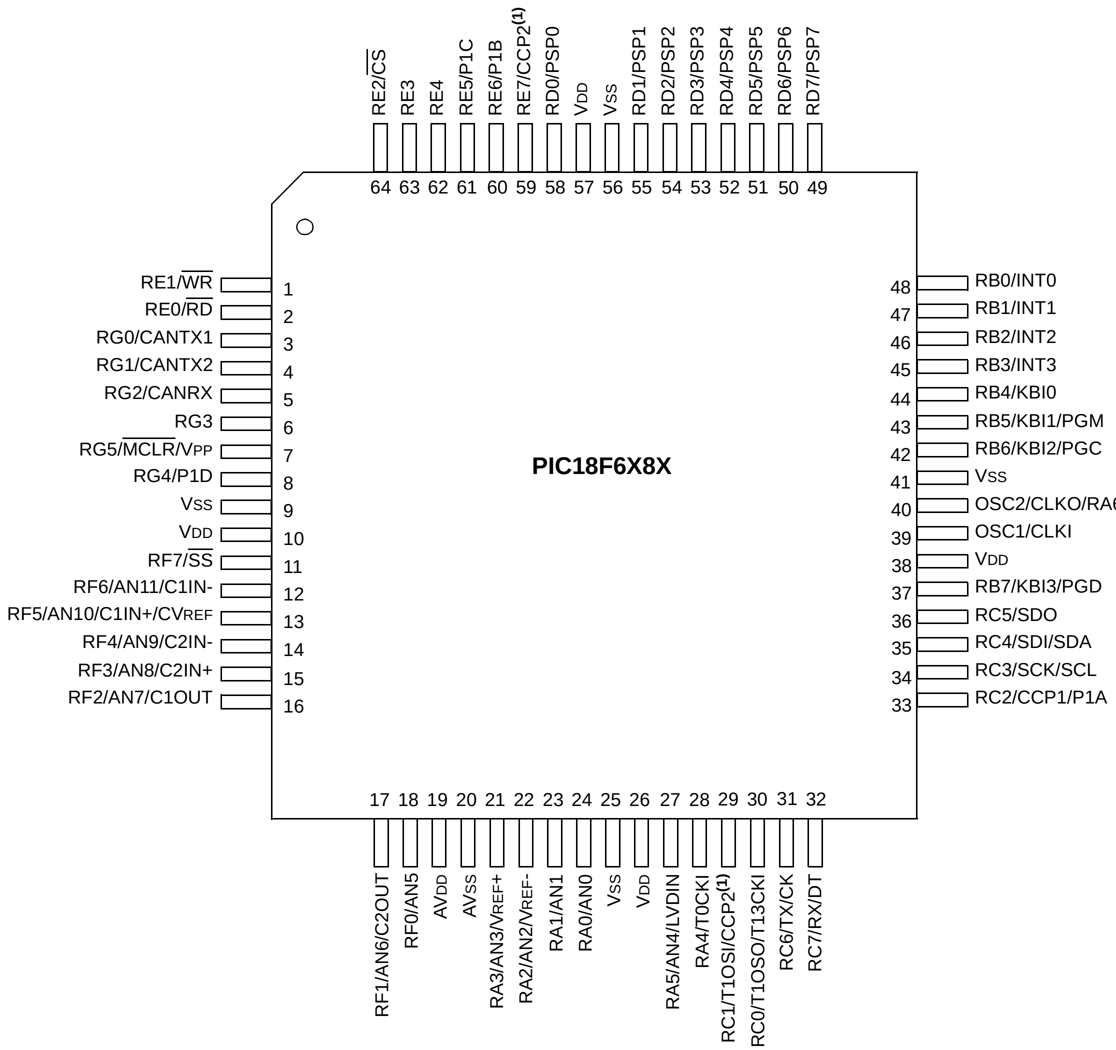
CMOS Technology:

- Low-power, high-speed Flash technology
- Fully static design
- Wide operating voltage range (2.0V to 5.5V)
- Industrial and Extended temperature ranges

Device	Program Memory		Data Memory		I/O	10-bit A/D (ch)	CCP/ ECCP (PWM)	MSSP		ECAN/ AUSART	Timers 8-bit/16-bit	EMA
	Bytes	# Single-Word Instructions	SRAM (bytes)	EEPROM (bytes)				SPI	Master I^2C			
PIC18F6585	48K	24576	3328	1024	53	12	1/1	Y	Y	Y/Y	2/3	N
PIC18F6680	64K	32768	3328	1024	53	12	1/1	Y	Y	Y/Y	2/3	N
PIC18F8585	48K	24576	3328	1024	69	16	1/1	Y	Y	Y/Y	2/3	Y
PIC18F8680	64K	32768	3328	1024	69	16	1/1	Y	Y	Y/Y	2/3	Y

Pin Diagrams

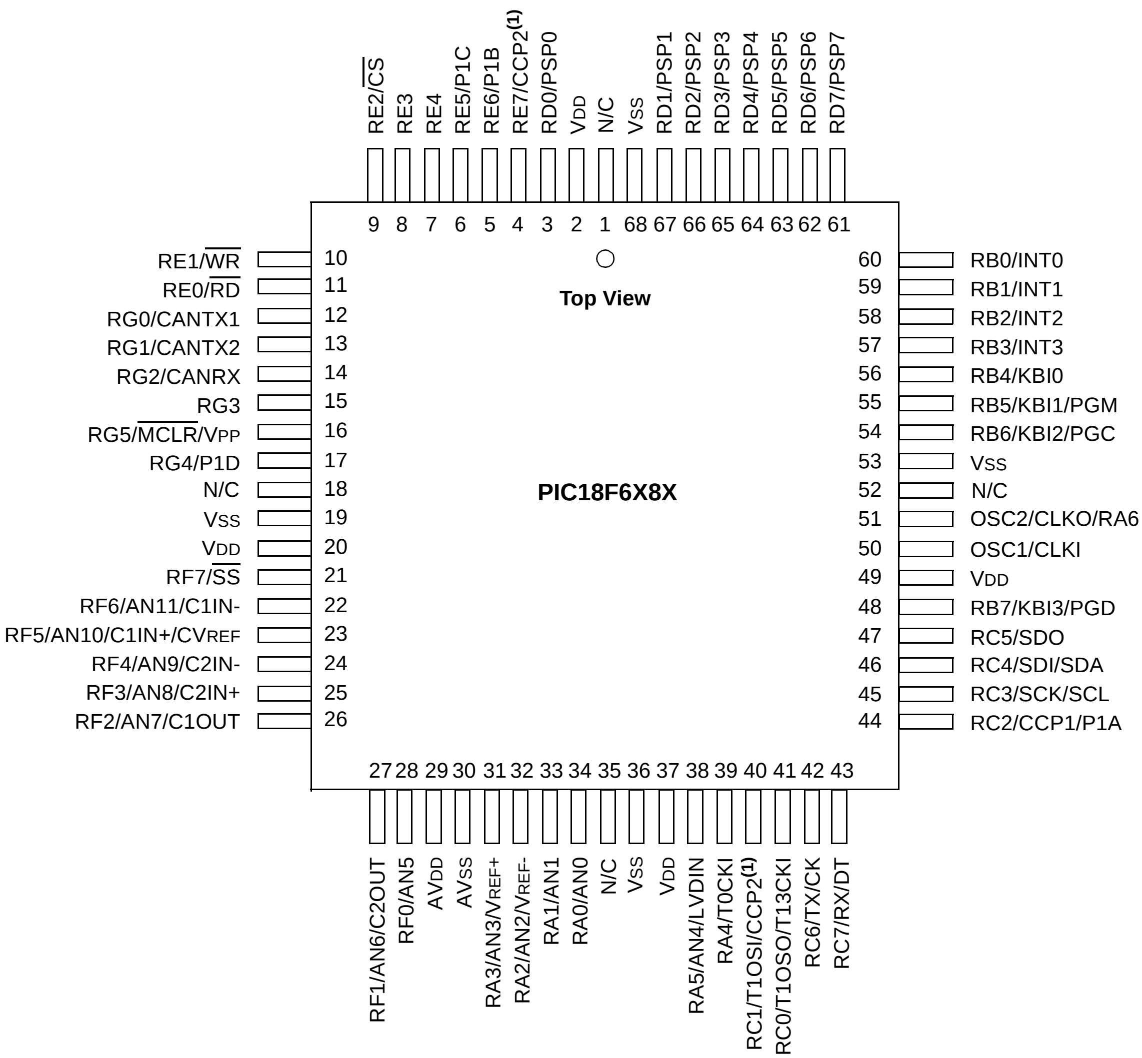
64-Pin TQFP



Note 1: CCP2 pin placement depends on CCP2MX setting.

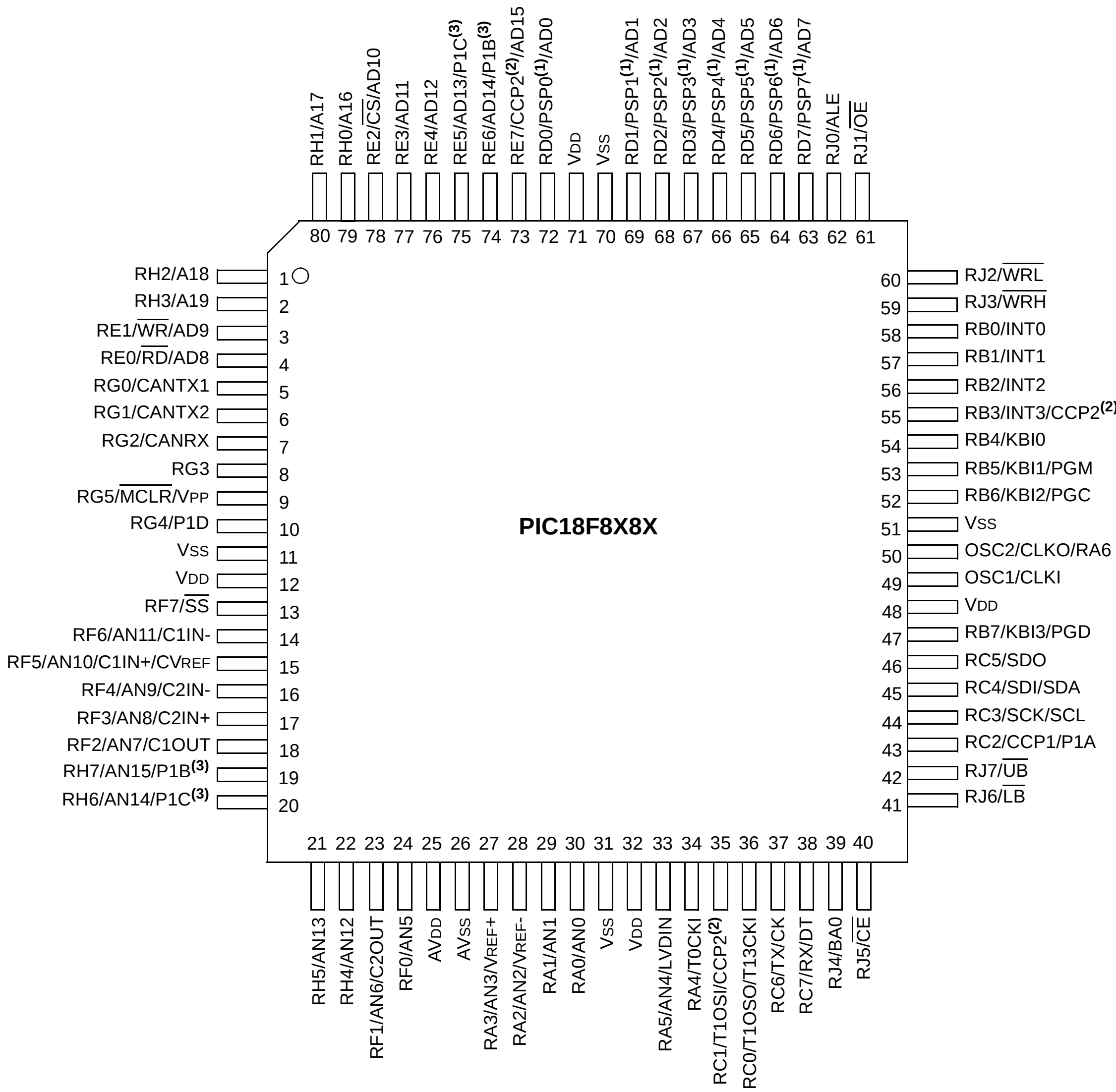
Pin Diagrams (Continued)

68-Pin PLCC



Note 1: CCP2 pin placement depends on CCP2MX setting.

Pin Diagrams (Continued)



Note 1: PSP is available only in Microcontroller mode.

2: CCP2 pin placement depends on CCP2MX and Processor mode settings.

3: P1B and P1C pin placement depends on ECCPMX setting.

Table of Contents

TO OUR VALUED CUSTOMERS

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

If you have any questions or comments regarding this publication, please contact the Marketing Communications Department via E-mail at **docerrors@mail.microchip.com** or fax the **Reader Response Form** in the back of this data sheet to (480) 792-4150. We welcome your feedback.

Most Current Data Sheet

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

http://www.microchip.com

You can determine the version of a data sheet by examining its literature number found on the bottom outside corner of any page. The last character of the literature number is the version number, (e.g., DS30000A is version A of document DS30000).

Errata

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

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

- Microchip's Worldwide Web site; http://www.microchip.com
- Your local Microchip sales office (see last page)
- The Microchip Corporate Literature Center; U.S. FAX: (480) 792-7277

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

Customer Notification System

Register on our Web site at **www.microchip.com/cn** to receive the most current information on all of our products.

NOTES:

1.0 DEVICE OVERVIEW

This document contains device specific information for the following devices:

- PIC18F6585
- PIC18F6680
- PIC18F8585
- PIC18F8680

PIC18F6X8X devices are available in 64-pin TQFP and 68-pin PLCC packages. PIC18F8X8X devices are available in the 80-pin TQFP package. They are differentiated from each other in four ways:

1. Flash program memory (48 Kbytes for PIC18FX585 devices, 64 Kbytes for PIC18FX680)
2. A/D channels (12 for PIC18F6X8X devices, 16 for PIC18F8X8X)
3. I/O ports (7 on PIC18F6X8X devices, 9 on PIC18F8X8X)
4. External program memory interface (present only on PIC18F8X8X devices)

All other features for devices in the PIC18F6585/8585/6680/8680 family are identical. These are summarized in Table 1-1.

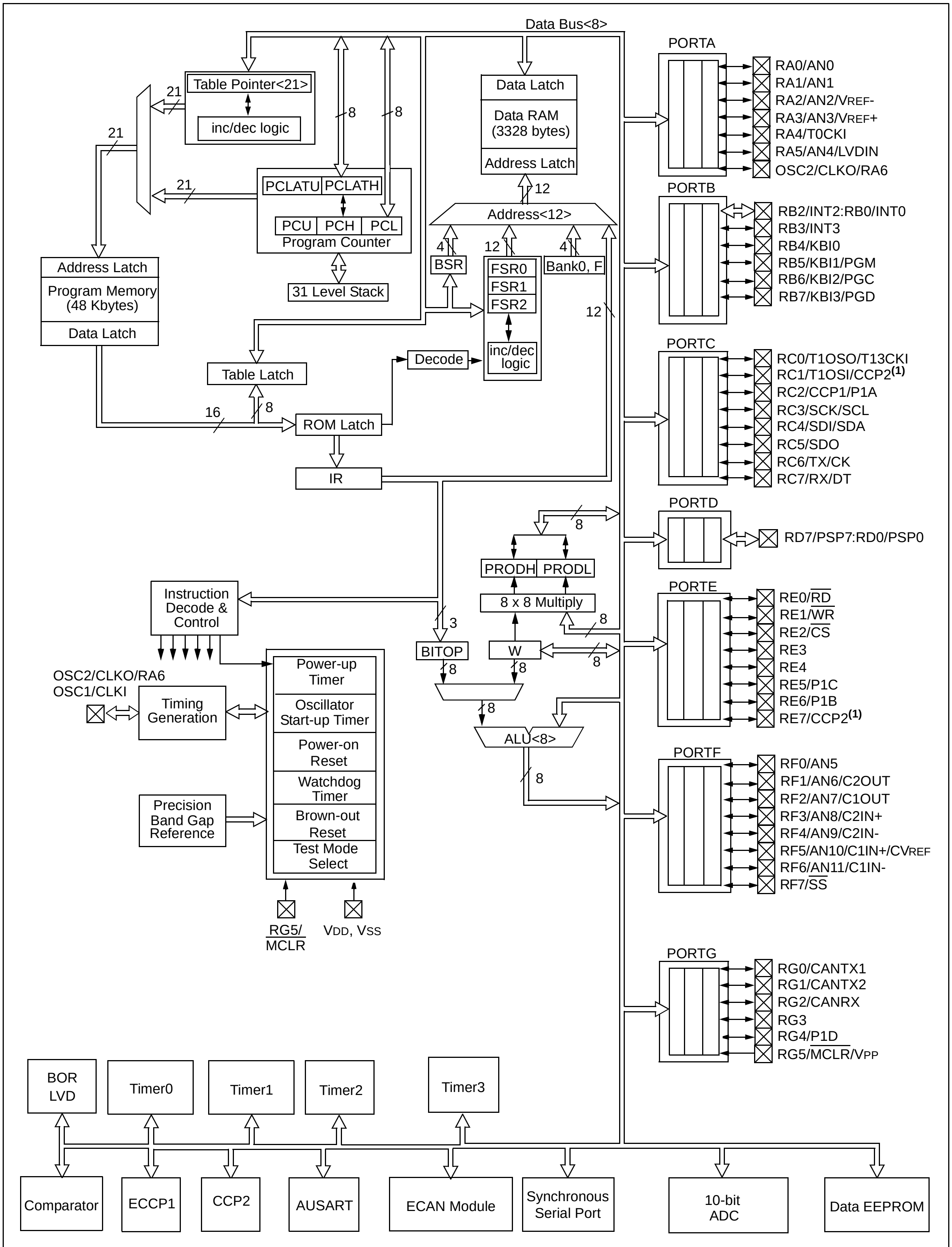
Block diagrams of the PIC18F6X8X and PIC18F8X8X devices are provided in Figure 1-1 and Figure 1-2, respectively. The pinouts for these device families are listed in Table 1-2.

TABLE 1-1: PIC18F6585/8585/6680/8680 DEVICE FEATURES

Features	PIC18F6585	PIC18F6680	PIC18F8585	PIC18F8680
Operating Frequency	DC – 40 MHz	DC – 40 MHz	DC – 40 MHz DC – 25 MHz w/EMA	DC – 40 MHz DC – 25 MHz w/EMA
Program Memory (Bytes)	48K	64K	48K (2 MB EMA)	64K (2 MB EMA)
Program Memory (Instructions)	24576	32768	24576	32768
Data Memory (Bytes)	3328	3328	3328	3328
Data EEPROM Memory (Bytes)	1024	1024	1024	1024
External Memory Interface	No	No	Yes	Yes
Interrupt Sources	29	29	29	29
I/O Ports	Ports A-G	Ports A-G	Ports A-H, J	Ports A-H, J
Timers	4	4	4	4
Capture/Compare/PWM Module	1	1	1	1
Enhanced Capture/Compare/PWM Module	1	1	1	1
Serial Communications	MSSP, Enhanced AUSART, ECAN	MSSP, Enhanced AUSART, ECAN	MSSP, Enhanced AUSART, ECAN	MSSP, Enhanced AUSART, ECAN
Parallel Communications	PSP	PSP	PSP(1)	PSP(1)
10-bit Analog-to-Digital Module	12 input channels	12 input channels	16 input channels	16 input channels
Resets (and Delays)	POR, BOR, `RESET` Instruction, Stack Full, Stack Underflow (PWRT, OST)	POR, BOR, `RESET` Instruction, Stack Full, Stack Underflow (PWRT, OST)	POR, BOR, `RESET` Instruction, Stack Full, Stack Underflow (PWRT, OST)	POR, BOR, `RESET` Instruction, Stack Full, Stack Underflow (PWRT, OST)
Programmable Low-Voltage Detect	Yes	Yes	Yes	Yes
Programmable Brown-out Reset	Yes	Yes	Yes	Yes
Instruction Set	75 Instructions	75 Instructions	75 Instructions	75 Instructions
Package	64-pin TQFP, 68-pin PLCC	64-pin TQFP, 68-pin PLCC	80-pin TQFP	80-pin TQFP

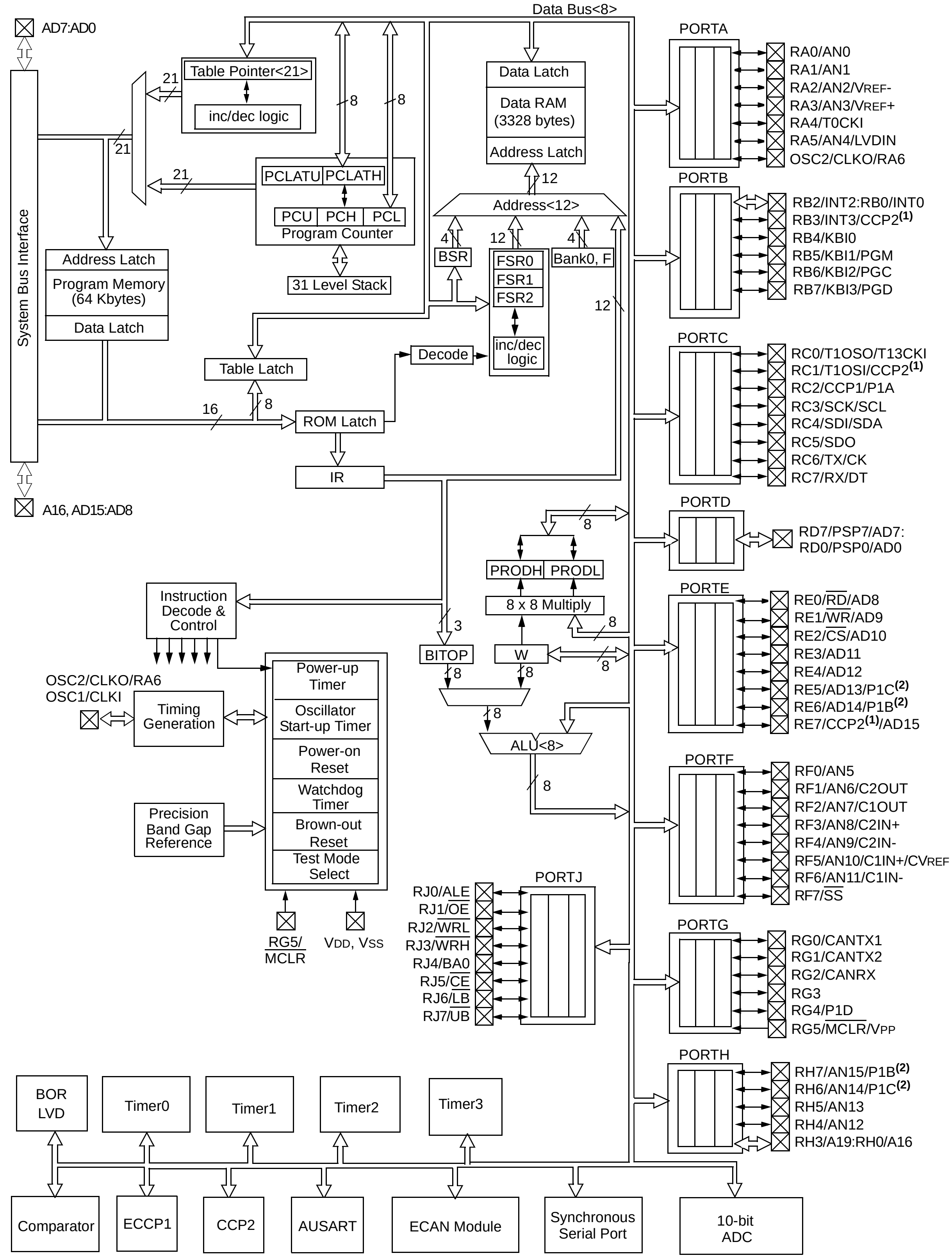
Note 1: PSP is only available in Microcontroller mode.

FIGURE 1-1: PIC18F6X8X BLOCK DIAGRAM



Note 1: The CCP2 pin placement depends on the CCP2MX and Processor mode settings.

FIGURE 1-2: PIC18F8X8X BLOCK DIAGRAM



Note 1: The CCP2 pin placement depends on the CCP2MX and Processor mode settings.

2: P1B and P1C pin placement depends on the ECCPMX setting.

TABLE 1-2: PIC18F6585/8585/6680/8680 PINOUT I/O DESCRIPTIONS

Pin Name	Pin Number			Pin Type	Buffer Type	Description
	PIC18F6X8X		PIC18F8X8X			
	TQFP	PLCC	TQFP			
RG5/$\overline{\text{MCLR}}$/VPP	7	16	9			Master Clear (input) or programming voltage (input).
RG5				I	ST	General purpose input pin.
$\overline{\text{MCLR}}$				I	ST	Master Clear (Reset) input. This pin is an active-low Reset to the device.
VPP				P		Programming voltage input.
OSC1/CLKI	39	50	49			Oscillator crystal or external clock input.
OSC1				I	CMOS/ST	Oscillator crystal input or external clock source input. ST buffer when configured in RC mode; otherwise CMOS.
CLKI				I	CMOS	External clock source input. Always associated with pin function OSC1 (see OSC1/CLKI, OSC2/CLKO pins).
OSC2/CLKO/RA6	40	51	50			Oscillator crystal or clock output.
OSC2				O	—	Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode.
CLKO				O	—	In RC mode, OSC2 pin outputs CLKO which has 1/4 the frequency of OSC1 and denotes the instruction cycle rate.
RA6				I/O	TTL	General purpose I/O pin.

Legend:
TTL = TTL compatible input
ST = Schmitt Trigger input with CMOS levels
I = Input
P = Power
CMOS = CMOS compatible input or output
Analog = Analog input
O = Output
OD = Open-Drain (no P diode to VDD)

Note 1: Alternate assignment for CCP2 in all operating modes except Microcontroller – applies to PIC18F8X8X only.
2: Default assignment when CCP2MX is set.
3: External memory interface functions are only available on PIC18F8X8X devices.
4: CCP2 is multiplexed with this pin by default when configured in Microcontroller mode; otherwise, it is multiplexed with either RB3 or RC1.
5: PORTH and PORTJ are only available on PIC18F8X8X (80-pin) devices.
6: PSP is available in Microcontroller mode only.
7: On PIC18F8X8X devices, these pins can be multiplexed with RH7/RH6 by changing the ECCPMX configuration bit.

TABLE 1-2: PIC18F6585/8585/6680/8680 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number			Pin Type	Buffer Type	Description
	PIC18F6X8X		PIC18F8X8X			
	TQFP	PLCC	TQFP			
						PORTA is a bidirectional I/O port.
RA0/AN0	24	34	30			
RA0				I/O	TTL	Digital I/O.
AN0				I	Analog	Analog input 0.
RA1/AN1	23	33	29			
RA1				I/O	TTL	Digital I/O.
AN1				I	Analog	Analog input 1.
RA2/AN2/V_{REF}-	22	32	28			
RA2				I/O	TTL	Digital I/O.
AN2				I	Analog	Analog input 2.
V_{REF}-				I	Analog	A/D reference voltage (Low) input.
RA3/AN3/V_{REF}+	21	31	27			
RA3				I/O	TTL	Digital I/O.
AN3				I	Analog	Analog input 3.
V_{REF}+				I	Analog	A/D reference voltage (High) input.
RA4/T0CKI	28	39	34			
RA4				I/O	ST/OD	Digital I/O – Open-drain when configured as output.
T0CKI				I	ST	Timer0 external clock input.
RA5/AN4/LVDIN	27	38	33			
RA5				I/O	TTL	Digital I/O.
AN4				I	Analog	Analog input 4.
LVDIN				I	Analog	Low-voltage detect input.
RA6						See the OSC2/CLKO/RA6 pin.

Legend: TTL = TTL compatible input; CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; Analog = Analog input; I = Input; O = Output; P = Power; OD = Open-Drain (no P diode to V_{DD})

Note 1: Alternate assignment for CCP2 in all operating modes except Microcontroller – applies to PIC18F8X8X only.

2: Default assignment when CCP2MX is set.

3: External memory interface functions are only available on PIC18F8X8X devices.

4: CCP2 is multiplexed with this pin by default when configured in Microcontroller mode; otherwise, it is multiplexed with either RB3 or RC1.

5: PORTH and PORTJ are only available on PIC18F8X8X (80-pin) devices.

6: PSP is available in Microcontroller mode only.

7: On PIC18F8X8X devices, these pins can be multiplexed with RH7/RH6 by changing the ECCPMX configuration bit.

TABLE 1-2: PIC18F6585/8585/6680/8680 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number			Pin Type	Buffer Type	Description
	PIC18F6X8X		PIC18F8X8X			
	TQFP	PLCC	TQFP			
						PORTB is a bidirectional I/O port. PORTB can be software programmed for internal weak pull-ups on all inputs.
RB0/INT0	48	60	58			
RB0				I/O	TTL	Digital I/O.
INT0				I	ST	External interrupt 0.
RB1/INT1	47	59	57			
RB1				I/O	TTL	Digital I/O.
INT1				I	ST	External interrupt 1.
RB2/INT2	46	58	56			
RB2				I/O	TTL	Digital I/O.
INT2				I	ST	External interrupt 2.
RB3/INT3/CCP2	45	57	55			
RB3				I/O	TTL	Digital I/O.
INT3				I/O	ST	External interrupt 3.
CCP2[(1)]				I/O	ST	Capture 2 input/Compare 2 output/ PWM 2 output.
RB4/KBI0	44	56	54			
RB4				I/O	TTL	Digital I/O.
KBI0				I	ST	Interrupt-on-change pin.
RB5/KBI1/PGM	43	55	53			
RB5				I/O	TTL	Digital I/O.
KBI1				I	ST	Interrupt-on-change pin.
PGM				I/O	ST	Low-Voltage ICSP Programming enable pin.
RB6/KBI2/PGC	42	54	52			
RB6				I/O	TTL	Digital I/O.
KBI2				I	ST	Interrupt-on-change pin.
PGC				I/O	ST	In-circuit debugger and ICSP programming clock.
RB7/KBI3/PGD	37	48	47			
RB7				I/O	TTL	Digital I/O.
KBI3				I/O	ST	Interrupt-on-change pin.
PGD						In-circuit debugger and ICSP programming data.

Legend:
TTL = TTL compatible input
ST = Schmitt Trigger input with CMOS levels
I = Input
P = Power
CMOS = CMOS compatible input or output
Analog = Analog input
O = Output
OD = Open-Drain (no P diode to V_{DD})

Note 1: Alternate assignment for CCP2 in all operating modes except Microcontroller – applies to PIC18F8X8X only.
2: Default assignment when CCP2MX is set.
3: External memory interface functions are only available on PIC18F8X8X devices.
4: CCP2 is multiplexed with this pin by default when configured in Microcontroller mode; otherwise, it is multiplexed with either RB3 or RC1.
5: PORTH and PORTJ are only available on PIC18F8X8X (80-pin) devices.
6: PSP is available in Microcontroller mode only.
7: On PIC18F8X8X devices, these pins can be multiplexed with RH7/RH6 by changing the ECCPMX configuration bit.

TABLE 1-2: PIC18F6585/8585/6680/8680 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number PIC18F6X8X TQFP	Pin Number PIC18F6X8X PLCC	Pin Number PIC18F8X8X TQFP	Pin Type	Buffer Type	Description
						PORTC is a bidirectional I/O port.
RC0/T1OSO/T13CKI	30	41	36			
RC0				I/O	ST	Digital I/O.
T1OSO				O	—	Timer1 oscillator output.
T13CKI				I	ST	Timer1/Timer3 external clock input.
RC1/T1OSI/CCP2	29	40	35			
RC1				I/O	ST	Digital I/O.
T1OSI				I	CMOS	Timer1 oscillator input.
CCP2(1, 4)				I/O	ST	CCP2 Capture input/Compare output/ PWM 2 output.
RC2/CCP1/P1A	33	44	43			
RC2				I/O	ST	Digital I/O.
CCP1				I/O	ST	CCP1 Capture input/Compare output.
P1A				I/O	ST	CCP1 PWM output A.
RC3/SCK/SCL	34	45	44			
RC3				I/O	ST	Digital I/O.
SCK				I/O	ST	Synchronous serial clock input/output for SPI mode.
SCL				I/O	ST	Synchronous serial clock input/output for I^2C mode.
RC4/SDI/SDA	35	46	45			
RC4				I/O	ST	Digital I/O.
SDI				I	ST	SPI data in.
SDA				I/O	ST	I^2C data I/O.
RC5/SDO	36	47	46			
RC5				I/O	ST	Digital I/O.
SDO				O	—	SPI data out.
RC6/TX/CK	31	42	37			
RC6				I/O	ST	Digital I/O.
TX				O	—	USART asynchronous transmit.
CK				I/O	ST	USART synchronous clock (see RX/DT).
RC7/RX/DT	32	43	38			
RC7				I/O	ST	Digital I/O.
RX				I	ST	USART 1 asynchronous receive.
DT				I/O	ST	USART 1 synchronous data (see TX/CK).

Legend: TTL = TTL compatible input; CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; Analog = Analog input; I = Input; O = Output; P = Power; OD = Open-Drain (no P diode to V_{DD})

Note 1: Alternate assignment for CCP2 in all operating modes except Microcontroller – applies to PIC18F8X8X only.
2: Default assignment when CCP2MX is set.
3: External memory interface functions are only available on PIC18F8X8X devices.
4: CCP2 is multiplexed with this pin by default when configured in Microcontroller mode; otherwise, it is multiplexed with either RB3 or RC1.
5: PORTH and PORTJ are only available on PIC18F8X8X (80-pin) devices.
6: PSP is available in Microcontroller mode only.
7: On PIC18F8X8X devices, these pins can be multiplexed with RH7/RH6 by changing the ECCPMX configuration bit.

TABLE 1-2: PIC18F6585/8585/6680/8680 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	PIC18F6X8X TQFP	PIC18F6X8X PLCC	PIC18F8X8X TQFP	Pin Type	Buffer Type	Description
						PORTD is a bidirectional I/O port. These pins have TTL input buffers when external memory is enabled.
RD0/PSP0/AD0	58	3	72			
RD0				I/O	ST	Digital I/O.
PSP0(6)				I/O	TTL	Parallel Slave Port data.
AD0(3)				I/O	TTL	External memory address/data 0.
RD1/PSP1/AD1	55	67	69			
RD1				I/O	ST	Digital I/O.
PSP1(6)				I/O	TTL	Parallel Slave Port data.
AD1(3)				I/O	TTL	External memory address/data 1.
RD2/PSP2/AD2	54	66	68			
RD2				I/O	ST	Digital I/O.
PSP2(6)				I/O	TTL	Parallel Slave Port data.
AD2(3)				I/O	TTL	External memory address/data 2.
RD3/PSP3/AD3	53	65	67			
RD3				I/O	ST	Digital I/O.
PSP3(6)				I/O	TTL	Parallel Slave Port data.
AD3(3)				I/O	TTL	External memory address/data 3.
RD4/PSP4/AD4	52	64	66			
RD4				I/O	ST	Digital I/O.
PSP4(6)				I/O	TTL	Parallel Slave Port data.
AD4(3)				I/O	TTL	External memory address/data 4.
RD5/PSP5/AD5	51	63	65			
RD5				I/O	ST	Digital I/O.
PSP5(6)				I/O	TTL	Parallel Slave Port data.
AD5(3)				I/O	TTL	External memory address/data 5.
RD6/PSP6/AD6	50	62	64			
RD6				I/O	ST	Digital I/O.
PSP6(6)				I/O	TTL	Parallel Slave Port data.
AD6(3)				I/O	TTL	External memory address/data 6.
RD7/PSP7/AD7	49	61	63			
RD7				I/O	ST	Digital I/O.
PSP7(6)				I/O	TTL	Parallel Slave Port data.
AD7(3)				I/O	TTL	External memory address/data 7.

Legend:
TTL = TTL compatible input; CMOS = CMOS compatible input or output
ST = Schmitt Trigger input with CMOS levels; Analog = Analog input
I = Input; O = Output
P = Power; OD = Open-Drain (no P diode to V_{DD})

Note 1: Alternate assignment for CCP2 in all operating modes except Microcontroller – applies to PIC18F8X8X only.
2: Default assignment when CCP2MX is set.
3: External memory interface functions are only available on PIC18F8X8X devices.
4: CCP2 is multiplexed with this pin by default when configured in Microcontroller mode; otherwise, it is multiplexed with either RB3 or RC1.
5: PORTH and PORTJ are only available on PIC18F8X8X (80-pin) devices.
6: PSP is available in Microcontroller mode only.
7: On PIC18F8X8X devices, these pins can be multiplexed with RH7/RH6 by changing the ECCPMX configuration bit.

TABLE 1-2: PIC18F6585/8585/6680/8680 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number			Pin Type	Buffer Type	Description
	PIC18F6X8X		PIC18F8X8X			
	TQFP	PLCC	TQFP			
						PORTE is a bidirectional I/O port.
RE0/$\overline{RD}$/AD8	2	11	4			
RE0				I/O	ST	Digital I/O.
$\overline{RD}$(6)				I	TTL	Read control for Parallel Slave Port (see $\overline{WR}$ and $\overline{CS}$ pins).
AD8(3)				I/O	TTL	External memory address/data 8.
RE1/$\overline{WR}$/AD9	1	10	3			
RE1				I/O	ST	Digital I/O.
$\overline{WR}$(6)				I	TTL	Write control for Parallel Slave Port (see $\overline{CS}$ and $\overline{RD}$ pins).
AD9(3)				I/O	TTL	External memory address/data 9.
RE2/$\overline{CS}$/AD10	64	9	78			
RE2				I/O	ST	Digital I/O.
$\overline{CS}$(6)				I	TTL	Chip select control for Parallel Slave Port (see $\overline{RD}$ and $\overline{WR}$).
AD10(3)				I/O	TTL	External memory address/data 10.
RE3/AD11	63	8	77			
RE3				I/O	ST	Digital I/O.
AD11(3)				I/O	TTL	External memory address/data 11.
RE4/AD12	62	7	76			
RE4				I/O	ST	Digital I/O.
AD12(3)				I/O	TTL	External memory address/data 12.
RE5/AD13/P1C	61	6	75			
RE5				I/O	ST	Digital I/O.
AD13(3)				I/O	TTL	External memory address/data 13.
P1C(7)				I/O	ST	ECCP1 PWM output C.
RE6/AD14/P1B	60	5	74			
RE6				I/O	ST	Digital I/O.
AD14(3)				I/O	TTL	External memory address/data 14.
P1B(7)				I/O	ST	ECCP1 PWM output B.
RE7/CCP2/AD15	59	4	73			
RE7				I/O	ST	Digital I/O.
CCP2(1,4)				I/O	ST	Capture 2 input/Compare 2 output/ PWM 2 output.
AD15(3)				I/O	TTL	External memory address/data 15.

Legend: TTL = TTL compatible input; CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; Analog = Analog input; I = Input; O = Output; P = Power; OD = Open-Drain (no P diode to V_{DD})

Note 1: Alternate assignment for CCP2 in all operating modes except Microcontroller – applies to PIC18F8X8X only.
2: Default assignment when CCP2MX is set.
3: External memory interface functions are only available on PIC18F8X8X devices.
4: CCP2 is multiplexed with this pin by default when configured in Microcontroller mode; otherwise, it is multiplexed with either RB3 or RC1.
5: PORTH and PORTJ are only available on PIC18F8X8X (80-pin) devices.
6: PSP is available in Microcontroller mode only.
7: On PIC18F8X8X devices, these pins can be multiplexed with RH7/RH6 by changing the ECCPMX configuration bit.

TABLE 1-2: PIC18F6585/8585/6680/8680 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number			Pin Type	Buffer Type	Description
	PIC18F6X8X		PIC18F8X8X			
	TQFP	PLCC	TQFP			
						PORTF is a bidirectional I/O port.
RF0/AN5	18	28	24			
RF0				I/O	ST	Digital I/O.
AN5				I	Analog	Analog input 5.
RF1/AN6/C2OUT	17	27	23			
RF1				I/O	ST	Digital I/O.
AN6				I	Analog	Analog input 6.
C2OUT				O	ST	Comparator 2 output.
RF2/AN7/C1OUT	16	26	18			
RF2				I/O	ST	Digital I/O.
AN7				I	Analog	Analog input 7.
C1OUT				O	ST	Comparator 1 output.
RF3/AN8/C2IN+	15	25	17			
RF1				I/O	ST	Digital I/O.
AN8				I	Analog	Analog input 8.
C2IN+				I	Analog	Comparator 2 input (+).
RF4/AN9/C2IN-	14	24	16			
RF1				I/O	ST	Digital I/O.
AN9				I	Analog	Analog input 9.
C2IN-				I	Analog	Comparator 2 input (-).
RF5/AN10/C1IN+/CVREF	13	23	15			
RF1						
AN10				I/O	ST	Digital I/O.
C1IN+				I	Analog	Analog input 10.
CVREF				I	Analog	Comparator 1 input (+).
				O	Analog	Comparator VREF output.
RF6/AN11/C1IN-	12	22	14			
RF6				I/O	ST	Digital I/O.
AN11				I	Analog	Analog input 11.
C1IN-				I	Analog	Comparator 1 input (-)
RF7/$\overline{SS}$	11	21	13			
RF7				I/O	ST	Digital I/O.
$\overline{SS}$				I	TTL	SPI slave select input.

Legend:
- TTL = TTL compatible input
- ST = Schmitt Trigger input with CMOS levels
- I = Input
- P = Power
- CMOS = CMOS compatible input or output
- Analog = Analog input
- O = Output
- OD = Open-Drain (no P diode to VDD)

Note 1: Alternate assignment for CCP2 in all operating modes except Microcontroller – applies to PIC18F8X8X only.
2: Default assignment when CCP2MX is set.
3: External memory interface functions are only available on PIC18F8X8X devices.
4: CCP2 is multiplexed with this pin by default when configured in Microcontroller mode; otherwise, it is multiplexed with either RB3 or RC1.
5: PORTH and PORTJ are only available on PIC18F8X8X (80-pin) devices.
6: PSP is available in Microcontroller mode only.
7: On PIC18F8X8X devices, these pins can be multiplexed with RH7/RH6 by changing the ECCPMX configuration bit.

TABLE 1-2: PIC18F6585/8585/6680/8680 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number			Pin Type	Buffer Type	Description
	PIC18F6X8X		PIC18F8X8X			
	TQFP	PLCC	TQFP			
						PORTG is a bidirectional I/O port.
RG0/CANTX1	3	12	5			
RG0				I/O	ST	Digital I/O.
CANTX1				O	TTL	CAN bus transmit 1.
RG1/CANTX2	4	13	6			
RG1				I/O	ST	Digital I/O.
CANTX2				O	TTL	CAN bus transmit 2.
RG2/CANRX	5	14	7			
RG2				I/O	ST	Digital I/O.
CANRX				I	TTL	CAN bus receive.
RG3	6	15	8			
RG3				I/O	ST	Digital I/O.
RG4/P1D	8	17	10			
RG4				I/O	ST	Digital I/O.
P1D				O	TTL	ECCP1 PWM output D.
RG5	7	16	9	I	ST	General purpose input pin.

Legend: TTL = TTL compatible input; CMOS = CMOS compatible input or output
ST = Schmitt Trigger input with CMOS levels; Analog = Analog input
I = Input; O = Output
P = Power; OD = Open-Drain (no P diode to V_{DD})

Note 1: Alternate assignment for CCP2 in all operating modes except Microcontroller – applies to PIC18F8X8X only.
2: Default assignment when CCP2MX is set.
3: External memory interface functions are only available on PIC18F8X8X devices.
4: CCP2 is multiplexed with this pin by default when configured in Microcontroller mode; otherwise, it is multiplexed with either RB3 or RC1.
5: PORTH and PORTJ are only available on PIC18F8X8X (80-pin) devices.
6: PSP is available in Microcontroller mode only.
7: On PIC18F8X8X devices, these pins can be multiplexed with RH7/RH6 by changing the ECCPMX configuration bit.

TABLE 1-2: PIC18F6585/8585/6680/8680 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number PIC18F6X8X TQFP	Pin Number PIC18F6X8X PLCC	Pin Number PIC18F8X8X TQFP	Pin Type	Buffer Type	Description
						PORTH is a bidirectional I/O port(5).
RH0/A16	—	—	79			
RH0				I/O	ST	Digital I/O.
A16				O	TTL	External memory address 16.
RH1/A17	—	—	80			
RH1				I/O	ST	Digital I/O.
A17				O	TTL	External memory address 17.
RH2/A18	—	—	1			
RH2				I/O	ST	Digital I/O.
A18				O	TTL	External memory address 18.
RH3/A19	—	—	2			
RH3				I/O	ST	Digital I/O.
A19				O	TTL	External memory address 19.
RH4/AN12	—	—	22			
RH4				I/O	ST	Digital I/O.
AN12				I	Analog	Analog input 12.
RH5/AN13	—	—	21			
RH5				I/O	ST	Digital I/O.
AN13				I	Analog	Analog input 13.
RH6/AN14/P1C	—	—	20			
RH6				I/O	ST	Digital I/O.
AN14				I	Analog	Analog input 14.
P1C(7)				I/O	ST	Alternate CCP1 PWM output C.
RH7/AN15/P1B	—	—	19			
RH7				I/O	ST	Digital I/O.
AN15				I	Analog	Analog input 15.
P1B(7)						Alternate CCP1 PWM output B.

Legend: TTL = TTL compatible input; CMOS = CMOS compatible input or output
ST = Schmitt Trigger input with CMOS levels; Analog = Analog input
I = Input; O = Output
P = Power; OD = Open-Drain (no P diode to V_{DD})

Note 1: Alternate assignment for CCP2 in all operating modes except Microcontroller – applies to PIC18F8X8X only.
2: Default assignment when CCP2MX is set.
3: External memory interface functions are only available on PIC18F8X8X devices.
4: CCP2 is multiplexed with this pin by default when configured in Microcontroller mode; otherwise, it is multiplexed with either RB3 or RC1.
5: PORTH and PORTJ are only available on PIC18F8X8X (80-pin) devices.
6: PSP is available in Microcontroller mode only.
7: On PIC18F8X8X devices, these pins can be multiplexed with RH7/RH6 by changing the ECCPMX configuration bit.

TABLE 1-2: PIC18F6585/8585/6680/8680 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number			Pin Type	Buffer Type	Description
	PIC18F6X8X		PIC18F8X8X			
	TQFP	PLCC	TQFP			
						PORTJ is a bidirectional I/O port(5).
RJ0/ALE	—	—	62			
RJ0				I/O	ST	Digital I/O.
ALE				O	TTL	External memory address latch enable.
RJ1/$\overline{OE}$	—	—	61			
RJ1				I/O	ST	Digital I/O.
$\overline{OE}$				O	TTL	External memory output enable.
RJ2/$\overline{WRL}$	—	—	60			
RJ2				I/O	ST	Digital I/O.
$\overline{WRL}$				O	TTL	External memory write low control.
RJ3/$\overline{WRH}$	—	—	59			
RJ3				I/O	ST	Digital I/O.
$\overline{WRH}$				O	TTL	External memory write high control.
RJ4/BA0	—	—	39			
RJ4				I/O	ST	Digital I/O.
BA0				O	TTL	System bus byte address 0 control.
RJ5/$\overline{CE}$	—	—	40	I/O	ST	Digital I/O
$\overline{CE}$				O	TTL	External memory chip enable.
RJ6/$\overline{LB}$	—	—	42			
RJ6				I/O	ST	Digital I/O.
$\overline{LB}$				O	TTL	External memory low byte select.
RJ7/$\overline{UB}$	—	—	41			
RJ7				I/O	ST	Digital I/O.
$\overline{UB}$				O	TTL	External memory high byte select.
VSS	9, 25, 41, 56	19, 36, 53, 68	11, 31, 51, 70	P	—	Ground reference for logic and I/O pins.
VDD	10, 26, 38, 57	2, 20, 37, 49	12, 32, 48, 71	P	—	Positive supply for logic and I/O pins.
AVSS	20	30	26	P	—	Ground reference for analog modules.
AVDD	19	29	25	P	—	Positive supply for analog modules.
NC	—	1, 18, 35, 52	—	—	—	No connect.

Legend:
TTL = TTL compatible input
ST = Schmitt Trigger input with CMOS levels
I = Input
P = Power
CMOS = CMOS compatible input or output
Analog = Analog input
O = Output
OD = Open-Drain (no P diode to VDD)

Note 1: Alternate assignment for CCP2 in all operating modes except Microcontroller – applies to PIC18F8X8X only.
2: Default assignment when CCP2MX is set.
3: External memory interface functions are only available on PIC18F8X8X devices.
4: CCP2 is multiplexed with this pin by default when configured in Microcontroller mode; otherwise, it is multiplexed with either RB3 or RC1.
5: PORTH and PORTJ are only available on PIC18F8X8X (80-pin) devices.
6: PSP is available in Microcontroller mode only.
7: On PIC18F8X8X devices, these pins can be multiplexed with RH7/RH6 by changing the ECCPMX configuration bit.

NOTES:

2.0 OSCILLATOR CONFIGURATIONS

2.1 Oscillator Types

The PIC18F6585/8585/6680/8680 devices can be operated in eleven different oscillator modes. The user can program four configuration bits (FOSC3, FOSC2, FOSC1 and FOSC0) to select one of these eleven modes:

1. LP Low-Power Crystal
2. XT Crystal/Resonator
3. HS High-Speed Crystal/Resonator
4. RC External Resistor/Capacitor
5. EC External Clock
6. ECIO External Clock with I/O pin enabled
7. HS+PLL High-Speed Crystal/Resonator with PLL enabled
8. RCIO External Resistor/Capacitor with I/O pin enabled
9. ECIO+SPLL External Clock with software controlled PLL
10. ECIO+PLL External Clock with PLL and I/O pin enabled
11. HS+SPLL High-Speed Crystal/Resonator with software control

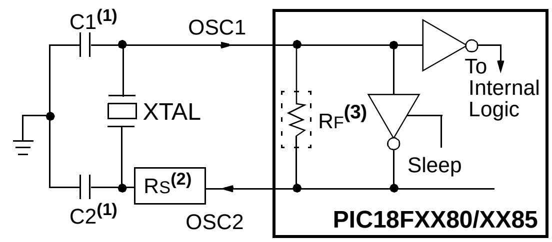
2.2 Crystal Oscillator/Ceramic Resonators

In XT, LP, HS, HS+PLL or HS+SPLL Oscillator modes, a crystal or ceramic resonator is connected to the OSC1 and OSC2 pins to establish oscillation. Figure 2-1 shows the pin connections.

The PIC18F6585/8585/6680/8680 oscillator design requires the use of a parallel cut crystal.

Note:	Use of a series cut crystal may give a frequency out of the crystal manufacturers specifications.

FIGURE 2-1: CRYSTAL/CERAMIC RESONATOR OPERATION (HS, XT OR LP CONFIGURATION)



Note 1: See Table 2-1 and Table 2-2 for recommended values of C1 and C2.

2: A series resistor (RS) may be required for AT strip cut crystals.

3: RF varies with the oscillator mode chosen.

TABLE 2-1: CAPACITOR SELECTION FOR CERAMIC RESONATORS

Ranges Tested:			
Mode	**Freq**	**C1**	**C2**
XT	455 kHz 2.0 MHz 4.0 MHz	68-100 pF 15-68 pF 15-68 pF	68-100 pF 15-68 pF 15-68 pF
HS	8.0 MHz 16.0 MHz	10-68 pF 10-22 pF	10-68 pF 10-22 pF
These values are for design guidance only. See notes following this table.			
Resonators Used:			
2.0 MHz	Murata Erie CSA2.00MG	± 0.5%	
4.0 MHz	Murata Erie CSA4.00MG	± 0.5%	
8.0 MHz	Murata Erie CSA8.00MT	± 0.5%	
16.0 MHz	Murata Erie CSA16.00MX	± 0.5%	
All resonators used did not have built-in capacitors.			

Note 1: Higher capacitance increases the stability of the oscillator, but also increases the start-up time.

2: When operating below 3V VDD, or when using certain ceramic resonators at any voltage, it may be necessary to use high gain HS mode, try a lower frequency resonator, or switch to a crystal oscillator.

3: Since each resonator/crystal has its own characteristics, the user should consult the resonator/crystal manufacturer for appropriate values of external components, or verify oscillator performance.

TABLE 2-2: CAPACITOR SELECTION FOR CRYSTAL OSCILLATOR

Ranges Tested:			
Mode	**Freq**	**C1**	**C2**
LP	32.0 kHz	33 pF	33 pF
	200 kHz	15 pF	15 pF
XT	200 kHz	47-68 pF	47-68 pF
	1.0 MHz	15 pF	15 pF
	4.0 MHz	15 pF	15 pF
HS	4.0 MHz	15 pF	15 pF
	8.0 MHz	15-33 pF	15-33 pF
	20.0 MHz	15-33 pF	15-33 pF
	25.0 MHz	TBD	TBD
These values are for design guidance only. See notes following this table.			
Crystals Used			
32.0 kHz	Epson C-001R32.768K-A		± 20 PPM
200 kHz	STD XTL 200.000KHz		± 20 PPM
1.0 MHz	ECS ECS-10-13-1		± 50 PPM
4.0 MHz	ECS ECS-40-20-1		± 50 PPM
8.0 MHz	Epson CA-301 8.000M-C		± 30 PPM
20.0 MHz	Epson CA-301 20.000M-C		± 30 PPM

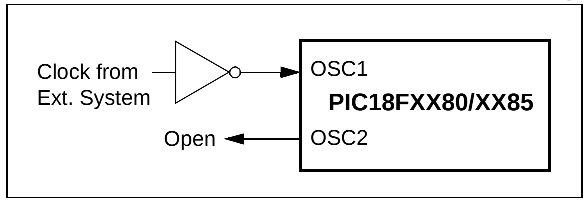
Note 1: Higher capacitance increases the stability of the oscillator, but also increases the start-up time.

2: Rs (see Figure 2-1) may be required in HS mode, as well as XT mode, to avoid overdriving crystals with low drive level specifications.

3: Since each resonator/crystal has its own characteristics, the user should consult the resonator/crystal manufacturer for appropriate values of external components, or verify oscillator performance.

An external clock source may also be connected to the OSC1 pin in the HS, XT and LP modes, as shown in Figure 2-2.

FIGURE 2-2: EXTERNAL CLOCK INPUT OPERATION (HS, XT OR LP OSC CONFIGURATION)

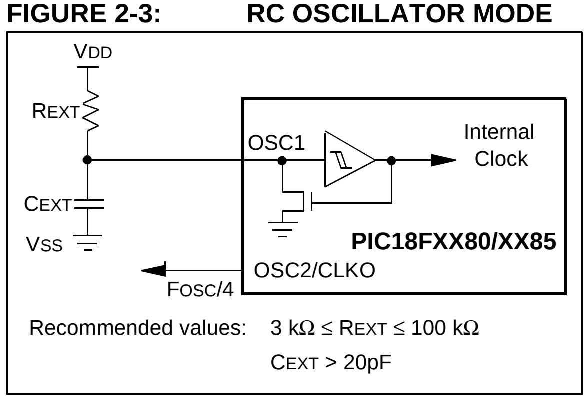


2.3 RC Oscillator

For timing insensitive applications, the “RC” and “RCIO” device options offer additional cost savings. The RC oscillator frequency is a function of the supply voltage, the resistor (R_{EXT}) and capacitor (C_{EXT}) values and the operating temperature. In addition to this, the oscillator frequency will vary from unit to unit, due to normal process parameter variation. Furthermore, the difference in lead frame capacitance between package types will also affect the oscillation frequency, especially for low C_{EXT} values. The user also needs to take into account variation due to tolerance of external R and C components used. Figure 2-3 shows how the R/C combination is connected.

In the RC Oscillator mode, the oscillator frequency divided by 4 is available on the OSC2 pin. This signal may be used for test purposes or to synchronize other logic.

FIGURE 2-3: RC OSCILLATOR MODE



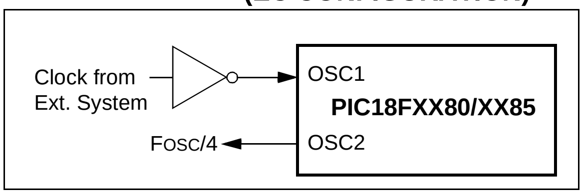
The RCIO Oscillator mode functions like the RC mode except that the OSC2 pin becomes an additional general purpose I/O pin. The I/O pin becomes bit 6 of PORTA (RA6).

2.4 External Clock Input

The EC, ECIO, EC+PLL and EC+SPLL Oscillator modes require an external clock source to be connected to the OSC1 pin. The feedback device between OSC1 and OSC2 is turned off in these modes to save current. There is a maximum 1.5 µs start-up required after a Power-on Reset, or wake-up from Sleep mode.

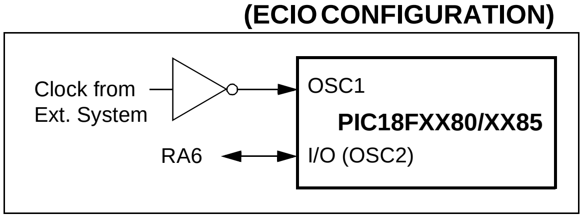
In the EC Oscillator mode, the oscillator frequency divided by 4 is available on the OSC2 pin. This signal may be used for test purposes or to synchronize other logic. Figure 2-4 shows the pin connections for the EC Oscillator mode.

FIGURE 2-4: EXTERNAL CLOCK INPUT OPERATION (EC CONFIGURATION)



The ECIO Oscillator mode functions like the EC mode, except that the OSC2 pin becomes an additional general purpose I/O pin. The I/O pin becomes bit 6 of PORTA (RA6). Figure 2-5 shows the pin connections for the ECIO Oscillator mode.

FIGURE 2-5: EXTERNAL CLOCK INPUT OPERATION (ECIO CONFIGURATION)



2.5 Phase Locked Loop (PLL)

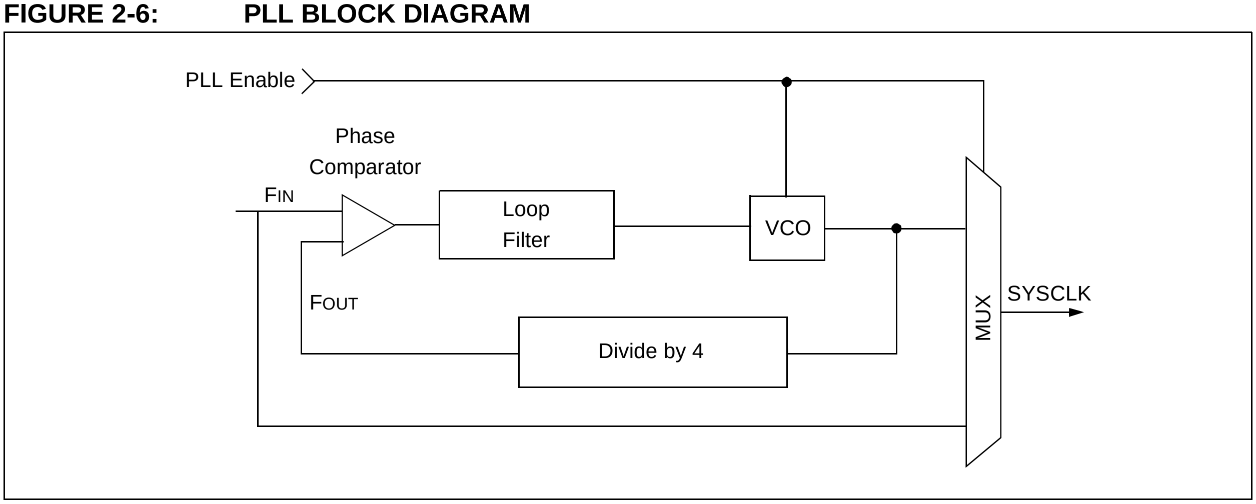
A Phase Locked Loop circuit is provided as a programmable option for users that want to multiply the frequency of the incoming oscillator signal by 4. For an input clock frequency of 10 MHz, the internal clock frequency will be multiplied to 40 MHz. This is useful for customers who are concerned with EMI due to high-frequency crystals.

The PLL can only be enabled when the oscillator configuration bits are programmed for High-Speed Oscillator or External Clock mode. If they are programmed for any other mode, the PLL is not enabled and the system clock will come directly from OSC1. There are two types of PLL modes: Software Controlled PLL and Configuration bits Controlled PLL. In Software Controlled PLL mode, PIC18F6585/8585/6680/8680 executes at regular clock frequency after all Reset conditions. During execution, application can enable PLL and switch to 4x clock frequency operation by setting the PLLEN bit in the OSCCON register. In Configuration bits Controlled PLL mode, PIC18F6585/8585/6680/8680 always executes with 4x clock frequency.

The type of PLL is selected by programming the FOSC<3:0> configuration bits in the CONFIG1H Configuration register. The oscillator mode is specified during device programming.

A PLL lock timer is used to ensure that the PLL has locked before device execution starts. The PLL lock timer has a time-out that is called TPLL.

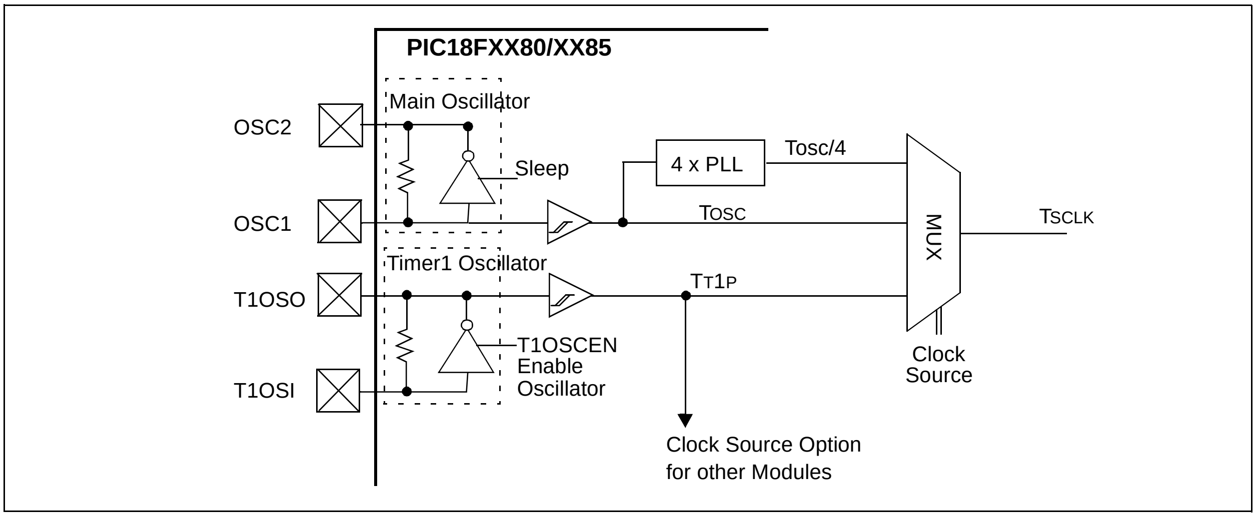
FIGURE 2-6: PLL BLOCK DIAGRAM



2.6 Oscillator Switching Feature

The PIC18F6585/8585/6680/8680 devices include a feature that allows the system clock source to be switched from the main oscillator to an alternate low-frequency clock source. For the PIC18F6585/8585/6680/8680 devices, this alternate clock source is the Timer1 oscillator. If a low-frequency crystal (32 kHz, for example) has been attached to the Timer1 oscillator pins and the Timer1 oscillator has been enabled, the device can switch to a low-power execution mode. Figure 2-7 shows a block diagram of the system clock sources. The clock switching feature is enabled by programming the Oscillator Switching Enable (OSCSEN) bit in configuration register, CONFIG1H, to a '0'. Clock switching is disabled in an erased device. See **Section 12.0 "Timer1 Module"** for further details of the Timer1 oscillator. See **Section 24.0 "Special Features of the CPU"** for configuration register details.

FIGURE 2-7: DEVICE CLOCK SOURCES



2.6.1 SYSTEM CLOCK SWITCH BIT

The system clock source switching is performed under software control. The System Clock Switch bits, SCS1:SCS0 (OSCCON<1:0>), control the clock switching. When the SCS0 bit is '0', the system clock source comes from the main oscillator that is selected by the FOSC configuration bits in configuration register, CONFIG1H. When the SCS0 bit is set, the system clock source will come from the Timer1 oscillator. The SCS0 bit is cleared on all forms of Reset.

When FOSC bits are programmed for software PLL mode, the SCS1 bit can be used to select between primary oscillator/clock and PLL output. The SCS1 bit will only have an effect on the system clock if the PLL is enabled (PLLEN = 1) and locked (LOCK = 1), else it will be forced clear. When programmed with Configuration Controlled PLL mode, the SCS1 bit will be forced clear.

Note:	The Timer1 oscillator must be enabled and operating to switch the system clock source. The Timer1 oscillator is enabled by setting the T1OSCEN bit in the Timer1 Control register (T1CON). If the Timer1 oscillator is not enabled, then any write to the SCS0 bit will be ignored (SCS0 bit forced cleared) and the main oscillator will continue to be the system clock source.

REGISTER 2-1: OSCCON REGISTER

U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	—	LOCK	PLLEN	SCS1	SCS0
bit 7							bit 0

bit 7-4 **Unimplemented:** Read as '0'

bit 3 **LOCK:** Phase Lock Loop Lock Status bit
1 = Phase Lock Loop output is stable as system clock
0 = Phase Lock Loop output is not stable and output cannot be used as system clock

bit 2 **PLLEN(1):** Phase Lock Loop Enable bit
1 = Enable Phase Lock Loop output as system clock
0 = Disable Phase Lock Loop

bit 1 **SCS1:** System Clock Switch bit 1
When PLLEN and LOCK bits are set:
1 = Use PLL output
0 = Use primary oscillator/clock input pin
When PLLEN or LOCK bit is cleared:
Bit is forced clear.

bit 0 **SCS0(2):** System Clock Switch bit 0
When $\overline{\text{OSCSEN}}$ configuration bit = 0 and T1OSCEN bit = 1:
1 = Switch to Timer1 oscillator/clock pin
0 = Use primary oscillator/clock input pin
When $\overline{\text{OSCSEN}}$ and T1OSCEN are in other states:
Bit is forced clear.

Note 1: PLLEN bit is ignored when configured for ECIO+PLL and HS+PLL. This bit is used in ECIO+SPLL and HS+SPLL modes only.

2: The setting of SCS0 = 1 supersedes SCS1 = 1.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

2.6.2 OSCILLATOR TRANSITIONS

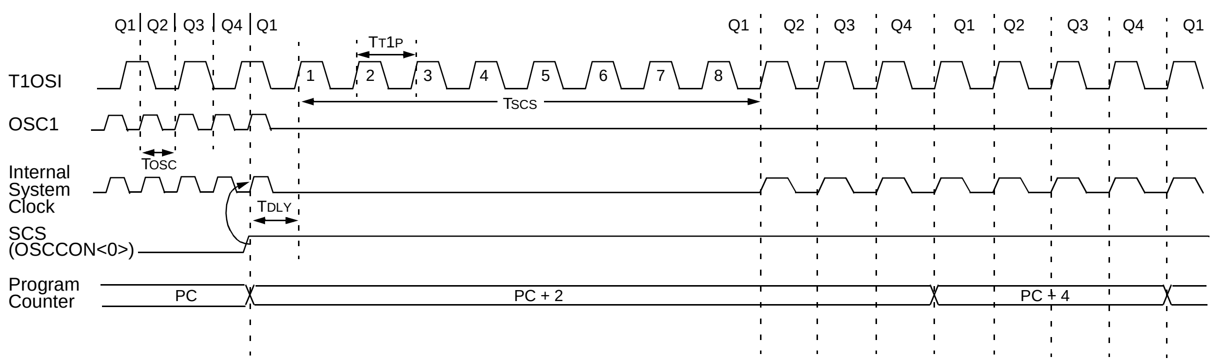
PIC18F6585/8585/6680/8680 devices contain circuitry to prevent "glitches" when switching between oscillator sources. Essentially, the circuitry waits for eight rising edges of the clock source that the processor is switching to. This ensures that the new clock source is stable and that its pulse width will not be less than the shortest pulse width of the two clock sources.

A timing diagram, indicating the transition from the main oscillator to the Timer1 oscillator, is shown in Figure 2-8. The Timer1 oscillator is assumed to be running all the time. After the SCS0 bit is set, the processor is frozen at the next occurring Q1 cycle. After eight synchronization cycles are counted from the Timer1 oscillator, operation resumes. No additional delays are required after the synchronization cycles.

The sequence of events that takes place when switching from the Timer1 oscillator to the main oscillator will depend on the mode of the main oscillator. In addition to eight clock cycles of the main oscillator, additional delays may take place.

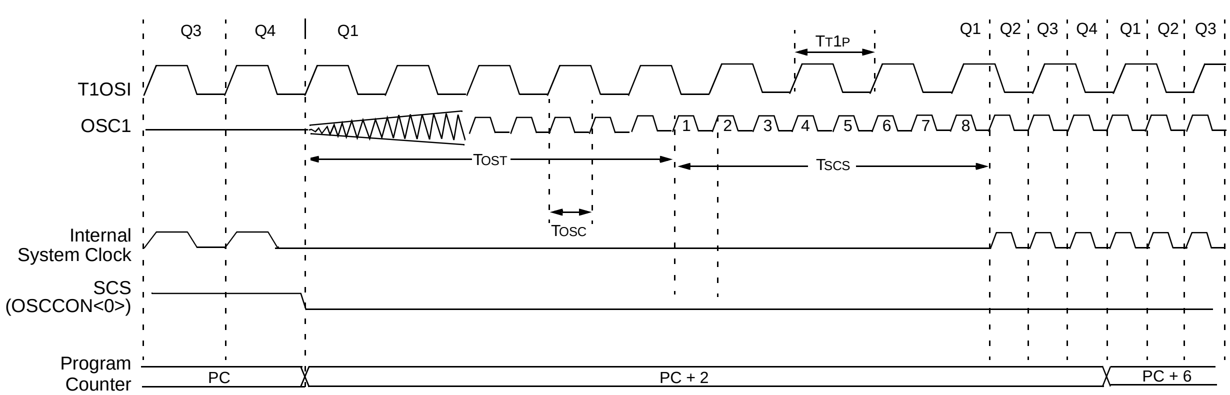
If the main oscillator is configured for an external crystal (HS, XT, LP), then the transition will take place after an oscillator start-up time (TOST) has occurred. A timing diagram, indicating the transition from the Timer1 oscillator to the main oscillator for HS, XT and LP modes, is shown in Figure 2-9.

FIGURE 2-8: TIMING DIAGRAM FOR TRANSITION FROM OSC1 TO TIMER1 OSCILLATOR



Note: TDLY is the delay from SCS high to first count of transition circuit.

FIGURE 2-9: TIMING FOR TRANSITION BETWEEN TIMER1 AND OSC1 (HS, XT, LP)

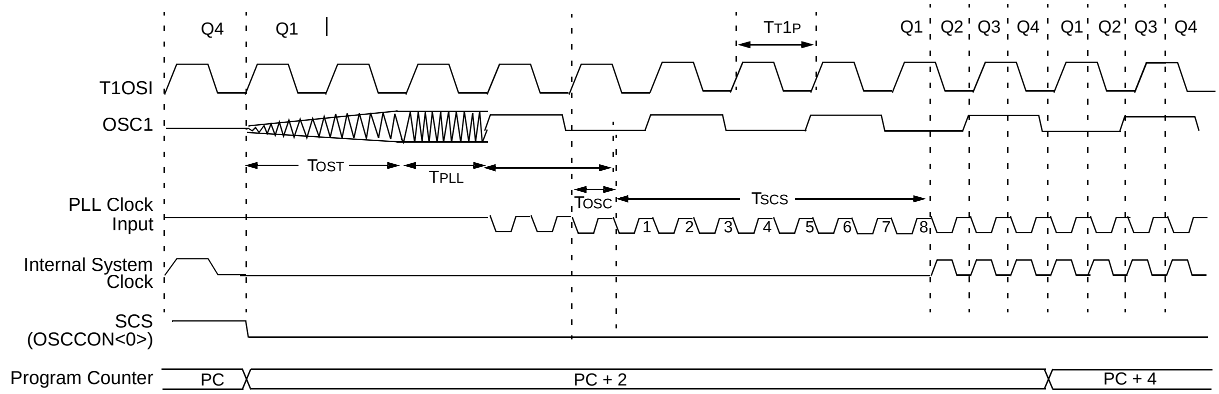


Note: TOST = 1024 TOSC (drawing not to scale).

If the main oscillator is configured for HS mode with PLL active, an oscillator start-up time (TOST) plus an additional PLL time-out (TPLL) will occur. The PLL time-out is typically 2 ms and allows the PLL to lock to the main oscillator frequency. A timing diagram, indicating the transition from the Timer1 oscillator to the main oscillator for HS-PLL mode, is shown in Figure 2-10.

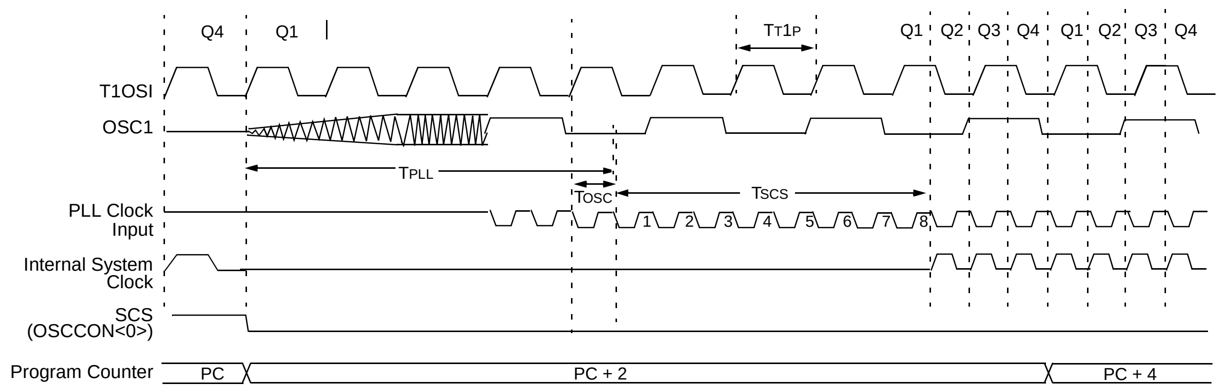
If the main oscillator is configured for EC mode with PLL active, only the PLL time-out (TPLL) will occur. The PLL time-out is typically 2 ms and allows the PLL to lock to the main oscillator frequency. A timing diagram, indicating the transition from the Timer1 oscillator to the main oscillator for EC with PLL active, is shown in Figure 2-11.

FIGURE 2-10: TIMING FOR TRANSITION BETWEEN TIMER1 AND OSC1 (HS WITH PLL ACTIVE, SCS1 = 1)



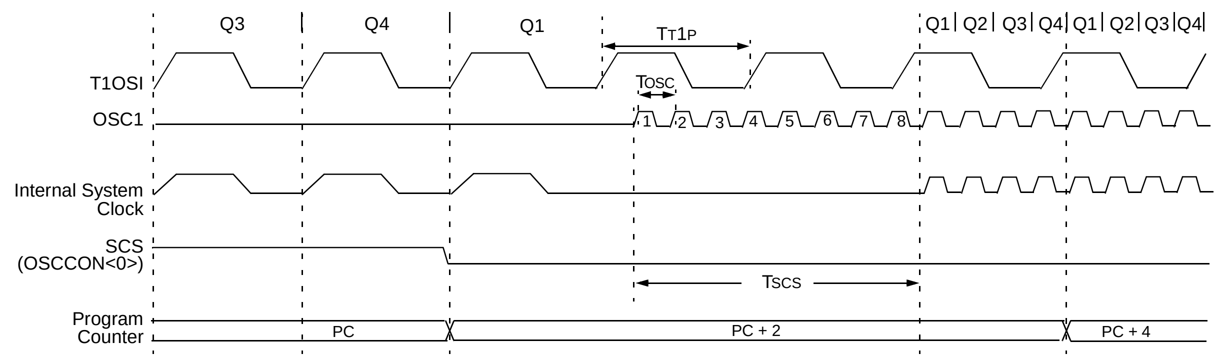
Note: TOST = 1024 TOSC (drawing not to scale).

FIGURE 2-11: TIMING FOR TRANSITION BETWEEN TIMER1 AND OSC1 (EC WITH PLL ACTIVE, SCS1 = 1)



If the main oscillator is configured in the RC, RCIO, EC or ECIO modes, there is no oscillator start-up time-out. Operation will resume after eight cycles of the main oscillator have been counted. A timing diagram, indicating the transition from the Timer1 oscillator to the main oscillator for RC, RCIO, EC and ECIO modes, is shown in Figure 2-12.

FIGURE 2-12: TIMING FOR TRANSITION BETWEEN TIMER1 AND OSC1 (RC, EC)



Note: RC Oscillator mode assumed.

2.7 Effects of Sleep Mode on the On-Chip Oscillator

When the device executes a `SLEEP` instruction, the on-chip clocks and oscillator are turned off and the device is held at the beginning of an instruction cycle (Q1 state). With the oscillator off, the OSC1 and OSC2 signals will stop oscillating. Since all the transistor switching currents have been removed, Sleep mode achieves the lowest current consumption of the device (only leakage currents). Enabling any on-chip feature that will operate during Sleep will increase the current consumed during Sleep. The user can wake from Sleep through external Reset, Watchdog Timer Reset, or through an interrupt.

TABLE 2-3: OSC1 AND OSC2 PIN STATES IN SLEEP MODE

OSC Mode	OSC1 Pin	OSC2 Pin
RC	Floating, external resistor should pull high	At logic low
RCIO	Floating, external resistor should pull high	Configured as PORTA, bit 6
ECIO	Floating	Configured as PORTA, bit 6
EC	Floating	At logic low
LP, XT, and HS	Feedback inverter disabled at quiescent voltage level	Feedback inverter disabled at quiescent voltage level

Note: See Table 3-1 in **Section 3.0 "Reset"**, for time-outs due to Sleep and $\overline{MCLR}$ Reset.

2.8 Power-up Delays

Power-up delays are controlled by two timers so that no external Reset circuitry is required for most applications. The delays ensure that the device is kept in Reset until the device power supply and clock are stable. For additional information on Reset operation, see **Section 3.0 "Reset"**.

The first timer is the Power-up Timer (PWRT) which optionally provides a fixed delay of 72 ms (nominal) on power-up only (POR and BOR). The second timer is the Oscillator Start-up Timer (OST), intended to keep the chip in Reset until the crystal oscillator is stable.

With the PLL enabled (HS+PLL and EC+PLL Oscillator mode), the time-out sequence following a Power-on Reset is different from other oscillator modes. The time-out sequence is as follows: First, the PWRT time-out is invoked after a POR time delay has expired. Then, the Oscillator Start-up Timer (OST) is invoked. However, this is still not a sufficient amount of time to allow the PLL to lock at high frequencies. The PWRT timer is used to provide an additional fixed 2 ms (nominal) time-out to allow the PLL ample time to lock to the incoming clock frequency.

NOTES:

3.0 RESET

The PIC18F6585/8585/6680/8680 devices differentiate between various kinds of Reset:

a) Power-on Reset (POR)
b) $\overline{\text{MCLR}}$ Reset during normal operation
c) $\overline{\text{MCLR}}$ Reset during Sleep
d) Watchdog Timer (WDT) Reset (during normal operation)
e) Programmable Brown-out Reset (BOR)
f) `RESET` Instruction
g) Stack Full Reset
h) Stack Underflow Reset

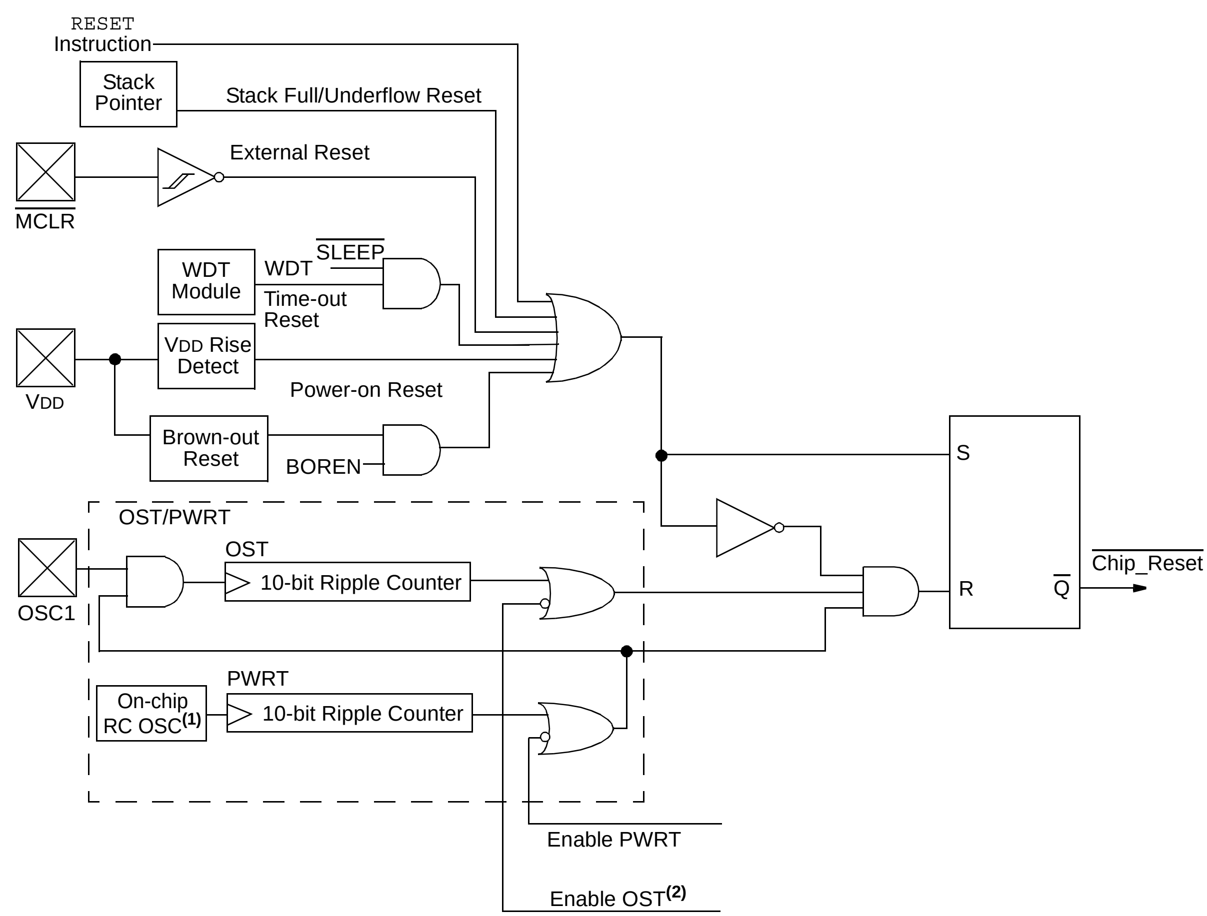
Most registers are unaffected by a Reset. Their status is unknown on POR and unchanged by all other Resets. The other registers are forced to a "Reset state" on Power-on Reset, $\overline{\text{MCLR}}$, WDT Reset, Brown-out Reset, $\overline{\text{MCLR}}$ Reset during Sleep and by the `RESET` instruction.

Most registers are not affected by a WDT wake-up since this is viewed as the resumption of normal operation. Status bits from the RCON register, $\overline{\text{RI}}$, $\overline{\text{TO}}$, $\overline{\text{PD}}$, $\overline{\text{POR}}$ and $\overline{\text{BOR}}$, are set or cleared differently in different Reset situations, as indicated in Table 3-2. These bits are used in software to determine the nature of the Reset. See Table 3-3 for a full description of the Reset states of all registers.

A simplified block diagram of the On-Chip Reset Circuit is shown in Figure 3-1.

The Enhanced MCU devices have a $\overline{\text{MCLR}}$ noise filter in the $\overline{\text{MCLR}}$ Reset path. The filter will detect and ignore small pulses. The $\overline{\text{MCLR}}$ pin is not driven low by any internal Resets, including the WDT.

FIGURE 3-1: SIMPLIFIED BLOCK DIAGRAM OF ON-CHIP RESET CIRCUIT



Note 1: This is a separate oscillator from the RC oscillator of the CLKI pin.

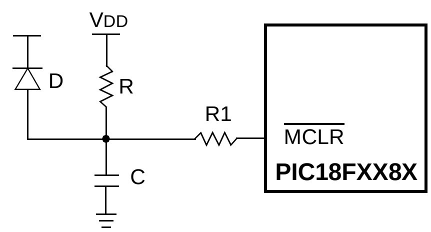
2: See Table 3-1 for time-out situations.

3.1 Power-on Reset (POR)

A Power-on Reset pulse is generated on-chip when VDD rise is detected. To take advantage of the POR circuitry, tie the MCLR pin through a 1 kΩ to 10 kΩ resistor to VDD. This will eliminate external RC components usually needed to create a Power-on Reset delay. A minimum rise rate for VDD is specified (parameter D004). For a slow rise time, see Figure 3-2.

When the device starts normal operation (i.e., exits the Reset condition), device operating parameters (voltage, frequency, temperature, etc.) must be met to ensure operation. If these conditions are not met, the device must be held in Reset until the operating conditions are met.

FIGURE 3-2: EXTERNAL POWER-ON RESET CIRCUIT (FOR SLOW VDD POWER-UP)



Note 1: External Power-on Reset circuit is required only if the VDD power-up slope is too slow. The diode D helps discharge the capacitor quickly when VDD powers down.

2: R < 40 kΩ is recommended to make sure that the voltage drop across R does not violate the device's electrical specification.

3: R1 = 1 kΩ to 10 kΩ will limit any current flowing into MCLR from external capacitor C, in the event of MCLR/VPP pin breakdown due to Electrostatic Discharge (ESD) or Electrical Overstress (EOS).

3.2 Power-up Timer (PWRT)

The Power-up Timer provides a fixed nominal time-out (parameter #33) only on power-up from the POR. The Power-up Timer operates on an internal RC oscillator. The chip is kept in Reset as long as the PWRT is active. The PWRT's time delay allows VDD to rise to an acceptable level. A configuration bit is provided to enable/disable the PWRT.

The power-up time delay will vary from chip-to-chip due to VDD, temperature and process variation. See DC parameter #33 for details.

3.3 Oscillator Start-up Timer (OST)

The Oscillator Start-up Timer (OST) provides 1024 oscillator cycles (from OSC1 input) delay after the PWRT delay is over (parameter #32). This ensures that the crystal oscillator or resonator has started and stabilized.

The OST time-out is invoked only for XT, LP and HS modes and only on Power-on Reset, or wake-up from Sleep.

3.4 PLL Lock Time-out

With the PLL enabled, the time-out sequence following a Power-on Reset is different from other oscillator modes. A portion of the Power-up Timer is used to provide a fixed time-out that is sufficient for the PLL to lock to the main oscillator frequency. This PLL lock time-out (TPLL) is typically 2 ms and follows the oscillator start-up time-out (OST).

3.5 Brown-out Reset (BOR)

A configuration bit, BOREN, can disable (if clear/programmed), or enable (if set) the Brown-out Reset circuitry. If VDD falls below parameter D005 for greater than parameter #35, the brown-out situation will reset the chip. A Reset may not occur if VDD falls below parameter D005 for less than parameter #35. The chip will remain in Brown-out Reset until VDD rises above BVDD. If the Power-up Timer is enabled, it will be invoked after VDD rises above BVDD; it then will keep the chip in Reset for an additional time delay (parameter #33). If VDD drops below BVDD while the Power-up Timer is running, the chip will go back into a Brown-out Reset and the Power-up Timer will be initialized. Once VDD rises above BVDD, the Power-up Timer will execute the additional time delay.

3.6 Time-out Sequence

On power-up, the time-out sequence is as follows: First, PWRT time-out is invoked after the POR time delay has expired. Then, OST is activated. The total time-out will vary based on oscillator configuration and the status of the PWRT. For example, in RC mode with the PWRT disabled, there will be no time-out at all. Figure 3-3, Figure 3-4, Figure 3-5, Figure 3-6 and Figure 3-7 depict time-out sequences on power-up.

Since the time-outs occur from the POR pulse, the time-outs will expire if MCLR is kept low long enough. Bringing MCLR high will begin execution immediately (Figure 3-5). This is useful for testing purposes or to synchronize more than one PIC18FXX8X device operating in parallel.

Table 3-2 shows the Reset conditions for some Special Function Registers while Table 3-3 shows the Reset conditions for all of the registers.

TABLE 3-1: TIME-OUT IN VARIOUS SITUATIONS

Oscillator Configuration	Power-up[2]		Brown-out	Wake-up from Sleep or Oscillator Switch
	$\overline{\text{PWRTE}}$ = 0	$\overline{\text{PWRTE}}$ = 1		
HS with PLL enabled[1]	72 ms + 1024 TOSC + 2ms	1024 TOSC + 2 ms	1024 TOSC + 2 ms	1024 TOSC + 2 ms
EC with PLL enabled[1]	72 ms + 2ms	1.5 μs + 2 ms	2 ms	1.5 μs + 2 ms
HS, XT, LP	72 ms + 1024 TOSC	1024 TOSC	1024 TOSC	1024 TOSC
EC	72 ms	1.5 μs	1.5 μs	1.5 μs[3]
External RC	72 ms	1.5 μs	1.5 μs	1.5 μs

Note 1: 2 ms is the nominal time required for the 4x PLL to lock.
2: 72 ms is the nominal power-up timer delay if implemented.
3: 1.5 μs is the recovery time from Sleep. There is no recovery time from oscillator switch.

REGISTER 3-1: RCON REGISTER BITS AND POSITIONS

R/W-0	U-0	U-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-0
IPEN	—	—	$\overline{\text{RI}}$	$\overline{\text{TO}}$	$\overline{\text{PD}}$	$\overline{\text{POR}}$	$\overline{\text{BOR}}$
bit 7							bit 0

Note: Refer to **Section 4.14 "RCON Register"** for bit definitions.

TABLE 3-2: STATUS BITS, THEIR SIGNIFICANCE AND THE INITIALIZATION CONDITION FOR RCON REGISTER

Condition	Program Counter	RCON Register	$\overline{\text{RI}}$	$\overline{\text{TO}}$	$\overline{\text{PD}}$	$\overline{\text{POR}}$	$\overline{\text{BOR}}$	STKFUL	STKUNF
Power-on Reset	0000h	`0--1 1100`	1	1	1	0	0	u	u
$\overline{\text{MCLR}}$ Reset during normal operation	0000h	`0--u uuuu`	u	u	u	u	u	u	u
Software Reset during normal operation	0000h	`0--0 uuuu`	0	u	u	u	u	u	u
Stack Full Reset during normal operation	0000h	`0--u uu11`	u	u	u	u	u	u	1
Stack Underflow Reset during normal operation	0000h	`0--u uu11`	u	u	u	u	u	1	u
$\overline{\text{MCLR}}$ Reset during Sleep	0000h	`0--u 10uu`	u	1	0	u	u	u	u
WDT Reset	0000h	`0--u 01uu`	1	0	1	u	u	u	u
WDT Wake-up	PC + 2	`u--u 00uu`	u	0	0	u	u	u	u
Brown-out Reset	0000h	`0--1 11u0`	1	1	1	1	0	u	u
Interrupt wake-up from Sleep	PC + 2[1]	`u--u 00uu`	u	1	0	u	u	u	u

Legend: u = unchanged, x = unknown, – = unimplemented bit, read as '0'
Note 1: When the wake-up is due to an interrupt and the GIEH or GIEL bits are set, the PC is loaded with the interrupt vector (000008h or 000018h).

TABLE 3-3: INITIALIZATION CONDITIONS FOR ALL REGISTERS

Register	Applicable Devices		Power-on Reset, Brown-out Reset	MCLR Resets WDT Reset RESET Instruction Stack Resets	Wake-up via WDT or Interrupt
TOSU	PIC18F6X8X	PIC18F8X8X	`---0 0000`	`---0 0000`	`---0 uuuu`(3)
TOSH	PIC18F6X8X	PIC18F8X8X	`0000 0000`	`0000 0000`	`uuuu uuuu`(3)
TOSL	PIC18F6X8X	PIC18F8X8X	`0000 0000`	`0000 0000`	`uuuu uuuu`(3)
STKPTR	PIC18F6X8X	PIC18F8X8X	`00-0 0000`	`uu-0 0000`	`uu-u uuuu`(3)
PCLATU	PIC18F6X8X	PIC18F8X8X	`---0 0000`	`---0 0000`	`---u uuuu`
PCLATH	PIC18F6X8X	PIC18F8X8X	`0000 0000`	`0000 0000`	`uuuu uuuu`
PCL	PIC18F6X8X	PIC18F8X8X	`0000 0000`	`0000 0000`	PC + 2(2)
TBLPTRU	PIC18F6X8X	PIC18F8X8X	`--00 0000`	`--00 0000`	`--uu uuuu`
TBLPTRH	PIC18F6X8X	PIC18F8X8X	`0000 0000`	`0000 0000`	`uuuu uuuu`
TBLPTRL	PIC18F6X8X	PIC18F8X8X	`0000 0000`	`0000 0000`	`uuuu uuuu`
TABLAT	PIC18F6X8X	PIC18F8X8X	`0000 0000`	`0000 0000`	`uuuu uuuu`
PRODH	PIC18F6X8X	PIC18F8X8X	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
PRODL	PIC18F6X8X	PIC18F8X8X	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
INTCON	PIC18F6X8X	PIC18F8X8X	`0000 000x`	`0000 000x`	`uuuu uuuu`(1)
INTCON2	PIC18F6X8X	PIC18F8X8X	`1111 1111`	`1111 1111`	`uuuu uuuu`(1)
INTCON3	PIC18F6X8X	PIC18F8X8X	`1100 0000`	`1100 0000`	`uuuu uuuu`(1)
INDF0	PIC18F6X8X	PIC18F8X8X	N/A	N/A	N/A
POSTINC0	PIC18F6X8X	PIC18F8X8X	N/A	N/A	N/A
POSTDEC0	PIC18F6X8X	PIC18F8X8X	N/A	N/A	N/A
PREINC0	PIC18F6X8X	PIC18F8X8X	N/A	N/A	N/A
PLUSW0	PIC18F6X8X	PIC18F8X8X	N/A	N/A	N/A
FSR0H	PIC18F6X8X	PIC18F8X8X	`---- xxxx`	`---- uuuu`	`---- uuuu`
FSR0L	PIC18F6X8X	PIC18F8X8X	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
WREG	PIC18F6X8X	PIC18F8X8X	`xxxx xxxx`	`uuuu uuuu`	`uuuu uuuu`
INDF1	PIC18F6X8X	PIC18F8X8X	N/A	N/A	N/A
POSTINC1	PIC18F6X8X	PIC18F8X8X	N/A	N/A	N/A
POSTDEC1	PIC18F6X8X	PIC18F8X8X	N/A	N/A	N/A
PREINC1	PIC18F6X8X	PIC18F8X8X	N/A	N/A	N/A
PLUSW1	PIC18F6X8X	PIC18F8X8X	N/A	N/A	N/A

Legend: `u` = unchanged, `x` = unknown, - = unimplemented bit, read as '`0`', `q` = value depends on condition. Shaded cells indicate conditions do not apply for the designated device.

Note 1: One or more bits in the INTCONx or PIRx registers will be affected (to cause wake-up).

2: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the PC is loaded with the interrupt vector (0008h or 0018h).

3: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the TOSU, TOSH and TOSL are updated with the current value of the PC. The STKPTR is modified to point to the next location in the hardware stack.

4: See Table 3-2 for Reset value for specific condition.

5: Bit 6 of PORTA, LATA, and TRISA are enabled in ECIO and RCIO Oscillator modes only. In all other oscillator modes, they are disabled and read '`0`'.

6: Bit 6 of PORTA, LATA and TRISA are not available on all devices. When unimplemented, they read '`0`'.

7: This register reads all '`0`'s until ECAN is set up in Mode 1 or Mode 2.

TABLE 3-3: INITIALIZATION CONDITIONS FOR ALL REGISTERS (CONTINUED)

Register	Applicable Devices		Power-on Reset, Brown-out Reset	MCLR Resets WDT Reset RESET Instruction Stack Resets	Wake-up via WDT or Interrupt
FSR1H	PIC18F6X8X	PIC18F8X8X	---- xxxx	---- uuuu	---- uuuu
FSR1L	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
BSR	PIC18F6X8X	PIC18F8X8X	---- 0000	---- 0000	---- uuuu
INDF2	PIC18F6X8X	PIC18F8X8X	N/A	N/A	N/A
POSTINC2	PIC18F6X8X	PIC18F8X8X	N/A	N/A	N/A
POSTDEC2	PIC18F6X8X	PIC18F8X8X	N/A	N/A	N/A
PREINC2	PIC18F6X8X	PIC18F8X8X	N/A	N/A	N/A
PLUSW2	PIC18F6X8X	PIC18F8X8X	N/A	N/A	N/A
FSR2H	PIC18F6X8X	PIC18F8X8X	---- xxxx	---- uuuu	---- uuuu
FSR2L	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
STATUS	PIC18F6X8X	PIC18F8X8X	---x xxxx	---u uuuu	---u uuuu
TMR0H	PIC18F6X8X	PIC18F8X8X	0000 0000	uuuu uuuu	uuuu uuuu
TMR0L	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
T0CON	PIC18F6X8X	PIC18F8X8X	1111 1111	1111 1111	uuuu uuuu
OSCCON	PIC18F6X8X	PIC18F8X8X	---- 0000	---- 0000	---- uuuu
LVDCON	PIC18F6X8X	PIC18F8X8X	--00 0101	--00 0101	--uu uuuu
WDTCON	PIC18F6X8X	PIC18F8X8X	---- ---0	---- ---0	---- ---u
RCON(4)	PIC18F6X8X	PIC18F8X8X	0--q 11qq	0--q qquu	u--u qquu
TMR1H	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
TMR1L	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
T1CON	PIC18F6X8X	PIC18F8X8X	0-00 0000	u-uu uuuu	u-uu uuuu
TMR2	PIC18F6X8X	PIC18F8X8X	0000 0000	0000 0000	uuuu uuuu
PR2	PIC18F6X8X	PIC18F8X8X	1111 1111	1111 1111	1111 1111
T2CON	PIC18F6X8X	PIC18F8X8X	-000 0000	-000 0000	-uuu uuuu
SSPBUF	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
SSPADD	PIC18F6X8X	PIC18F8X8X	0000 0000	0000 0000	uuuu uuuu
SSPSTAT	PIC18F6X8X	PIC18F8X8X	0000 0000	0000 0000	uuuu uuuu
SSPCON1	PIC18F6X8X	PIC18F8X8X	0000 0000	0000 0000	uuuu uuuu
SSPCON2	PIC18F6X8X	PIC18F8X8X	0000 0000	0000 0000	uuuu uuuu

Legend: u = unchanged, x = unknown, - = unimplemented bit, read as '0', q = value depends on condition. Shaded cells indicate conditions do not apply for the designated device.

Note 1: One or more bits in the INTCONx or PIRx registers will be affected (to cause wake-up).

2: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the PC is loaded with the interrupt vector (0008h or 0018h).

3: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the TOSU, TOSH and TOSL are updated with the current value of the PC. The STKPTR is modified to point to the next location in the hardware stack.

4: See Table 3-2 for Reset value for specific condition.

5: Bit 6 of PORTA, LATA, and TRISA are enabled in ECIO and RCIO Oscillator modes only. In all other oscillator modes, they are disabled and read '0'.

6: Bit 6 of PORTA, LATA and TRISA are not available on all devices. When unimplemented, they read '0'.

7: This register reads all '0's until ECAN is set up in Mode 1 or Mode 2.

TABLE 3-3: INITIALIZATION CONDITIONS FOR ALL REGISTERS (CONTINUED)

Register	Applicable Devices		Power-on Reset, Brown-out Reset	$\overline{MCLR}$ Resets WDT Reset RESET Instruction Stack Resets	Wake-up via WDT or Interrupt
ADRESH	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
ADRESL	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
ADCON0	PIC18F6X8X	PIC18F8X8X	--00 0000	--00 0000	--uu uuuu
ADCON1	PIC18F6X8X	PIC18F8X8X	--00 0000	--00 0000	--uu uuuu
ADCON2	PIC18F6X8X	PIC18F8X8X	0-00 0000	0-00 0000	u-uu uuuu
CCPR1H	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCPR1L	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCP1CON	PIC18F6X8X	PIC18F8X8X	0000 0000	0000 0000	uuuu uuuu
CCPR2H	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCPR2L	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
CCP2CON	PIC18F6X8X	PIC18F8X8X	--00 0000	--00 0000	--uu uuuu
CCPAS1	PIC18F6X8X	PIC18F8X8X	0000 0000	0000 0000	uuuu uuuu
CVRCON	PIC18F6X8X	PIC18F8X8X	0000 0000	0000 0000	uuuu uuuu
CMCON	PIC18F6X8X	PIC18F8X8X	0000 0000	0000 0000	uuuu uuuu
TMR3H	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
TMR3L	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
T3CON	PIC18F6X8X	PIC18F8X8X	0000 0000	uuuu uuuu	uuuu uuuu
PSPCON	PIC18F6X8X	PIC18F8X8X	0000 ----	0000 ----	uuuu ----
SPBRG	PIC18F6X8X	PIC18F8X8X	0000 0000	0000 0000	uuuu uuuu
RCREG	PIC18F6X8X	PIC18F8X8X	0000 0000	0000 0000	uuuu uuuu
TXREG	PIC18F6X8X	PIC18F8X8X	0000 0000	0000 0000	uuuu uuuu
TXSTA	PIC18F6X8X	PIC18F8X8X	0000 0010	0000 0010	uuuu uuuu
RCSTA	PIC18F6X8X	PIC18F8X8X	0000 000x	0000 000x	uuuu uuuu
EEADRH	PIC18F6X8X	PIC18F8X8X	---- --00	---- --00	---- --uu
EEADR	PIC18F6X8X	PIC18F8X8X	0000 0000	0000 0000	uuuu uuuu
EEDATA	PIC18F6X8X	PIC18F8X8X	0000 0000	0000 0000	uuuu uuuu
EECON2	PIC18F6X8X	PIC18F8X8X	xx-0 x000	uu-0 u000	uu-0 u000
EECON1	PIC18F6X8X	PIC18F8X8X	00-0 x000	00-0 u000	uu-u uuuu

Legend: u = unchanged, x = unknown, - = unimplemented bit, read as '0', q = value depends on condition. Shaded cells indicate conditions do not apply for the designated device.

Note 1: One or more bits in the INTCONx or PIRx registers will be affected (to cause wake-up).

2: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the PC is loaded with the interrupt vector (0008h or 0018h).

3: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the TOSU, TOSH and TOSL are updated with the current value of the PC. The STKPTR is modified to point to the next location in the hardware stack.

4: See Table 3-2 for Reset value for specific condition.

5: Bit 6 of PORTA, LATA, and TRISA are enabled in ECIO and RCIO Oscillator modes only. In all other oscillator modes, they are disabled and read '0'.

6: Bit 6 of PORTA, LATA and TRISA are not available on all devices. When unimplemented, they read '0'.

7: This register reads all '0's until ECAN is set up in Mode 1 or Mode 2.

TABLE 3-3: INITIALIZATION CONDITIONS FOR ALL REGISTERS (CONTINUED)

Register	Applicable Devices		Power-on Reset, Brown-out Reset	$\overline{MCLR}$ Resets WDT Reset RESET Instruction Stack Resets	Wake-up via WDT or Interrupt
IPR3	PIC18F6X8X	PIC18F8X8X	1111 1111	1111 1111	uuuu uuuu
PIR3	PIC18F6X8X	PIC18F8X8X	0000 0000	0000 0000	uuuu uuuu
PIE3	PIC18F6X8X	PIC18F8X8X	0000 0000	0000 0000	uuuu uuuu
IPR2	PIC18F6X8X	PIC18F8X8X	-1-1 1111	-1-1 1111	-u-u uuuu
PIR2	PIC18F6X8X	PIC18F8X8X	-0-0 0000	-0-0 0000	-u-u uuuu(1)
PIE2	PIC18F6X8X	PIC18F8X8X	-0-0 0000	-0-0 0000	-u-u uuuu
IPR1	PIC18F6X8X	PIC18F8X8X	1111 1111	1111 1111	uuuu uuuu
PIR1	PIC18F6X8X	PIC18F8X8X	0000 0000	0000 0000	uuuu uuuu(1)
PIE1	PIC18F6X8X	PIC18F8X8X	0000 0000	0000 0000	uuuu uuuu
MEMCON	PIC18F6X8X	PIC18F8X8X	0-00 --00	0-00 --00	u-uu --uu
TRISJ	PIC18F6X8X	PIC18F8X8X	1111 1111	1111 1111	uuuu uuuu
TRISH	PIC18F6X8X	PIC18F8X8X	1111 1111	1111 1111	uuuu uuuu
TRISG	PIC18F6X8X	PIC18F8X8X	---1 1111	---1 1111	---u uuuu
TRISF	PIC18F6X8X	PIC18F8X8X	1111 1111	1111 1111	uuuu uuuu
TRISE	PIC18F6X8X	PIC18F8X8X	0000 -111	0000 -111	uuuu -uuu
TRISD	PIC18F6X8X	PIC18F8X8X	1111 1111	1111 1111	uuuu uuuu
TRISC	PIC18F6X8X	PIC18F8X8X	1111 1111	1111 1111	uuuu uuuu
TRISB	PIC18F6X8X	PIC18F8X8X	1111 1111	1111 1111	uuuu uuuu
TRISA(5,6)	PIC18F6X8X	PIC18F8X8X	-111 1111(5)	-111 1111(5)	-uuu uuuu(5)
LATJ	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
LATH	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
LATG	PIC18F6X8X	PIC18F8X8X	---x xxxx	---u uuuu	---u uuuu
LATF	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
LATE	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
LATD	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
LATC	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
LATB	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
LATA(5,6)	PIC18F6X8X	PIC18F8X8X	-xxx xxxx(5)	-uuu uuuu(5)	-uuu uuuu(5)

Legend: u = unchanged, x = unknown, - = unimplemented bit, read as '0', q = value depends on condition. Shaded cells indicate conditions do not apply for the designated device.

Note 1: One or more bits in the INTCONx or PIRx registers will be affected (to cause wake-up).

2: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the PC is loaded with the interrupt vector (0008h or 0018h).

3: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the TOSU, TOSH and TOSL are updated with the current value of the PC. The STKPTR is modified to point to the next location in the hardware stack.

4: See Table 3-2 for Reset value for specific condition.

5: Bit 6 of PORTA, LATA, and TRISA are enabled in ECIO and RCIO Oscillator modes only. In all other oscillator modes, they are disabled and read '0'.

6: Bit 6 of PORTA, LATA and TRISA are not available on all devices. When unimplemented, they read '0'.

7: This register reads all '0's until ECAN is set up in Mode 1 or Mode 2.

TABLE 3-3: INITIALIZATION CONDITIONS FOR ALL REGISTERS (CONTINUED)

Register	Applicable Devices		Power-on Reset, Brown-out Reset	MCLR Resets WDT Reset RESET Instruction Stack Resets	Wake-up via WDT or Interrupt
PORTJ	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
PORTH	PIC18F6X8X	PIC18F8X8X	0000 xxxx	0000 uuuu	uuuu uuuu
PORTG	PIC18F6X8X	PIC18F8X8X	--xx xxxx	--uu uuuu	--uu uuuu
PORTF	PIC18F6X8X	PIC18F8X8X	x000 0000	u000 0000	u000 0000
PORTE	PIC18F6X8X	PIC18F8X8X	---- -000	---- -000	---- -uuu
PORTD	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
PORTC	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
PORTB	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
PORTA(5,6)	PIC18F6X8X	PIC18F8X8X	-x0x 0000(5)	-u0u 0000(5)	-uuu uuuu(5)
SPBRGH	PIC18F6X8X	PIC18F8X8X	0000 0000	0000 0000	uuuu uuuu
BAUDCON	PIC18F6X8X	PIC18F8X8X	-1-0 0-00	-1-0 0-00	-u-u u-uu
ECCP1DEL	PIC18F6X8X	PIC18F8X8X	0000 0000	0000 0000	uuuu uuuu
ECANCON	PIC18F6X8X	PIC18F8X8X	0001 0000	0001 0000	uuuu uuuu
TXERRCNT	PIC18F6X8X	PIC18F8X8X	0000 0000	0000 0000	uuuu uuuu
RXERRCNT	PIC18F6X8X	PIC18F8X8X	0000 0000	0000 0000	uuuu uuuu
COMSTAT	PIC18F6X8X	PIC18F8X8X	0000 0000	0000 0000	uuuu uuuu
CIOCON	PIC18F6X8X	PIC18F8X8X	0000 ----	0000 ----	uuuu ----
BRGCON3	PIC18F6X8X	PIC18F8X8X	00-- -000	00-- -000	uu-- -uuu
BRGCON2	PIC18F6X8X	PIC18F8X8X	0000 0000	0000 0000	uuuu uuuu
BRGCON1	PIC18F6X8X	PIC18F8X8X	0000 0000	0000 0000	uuuu uuuu
CANCON	PIC18F6X8X	PIC18F8X8X	1000 000-	1000 000-	uuuu uuu-
CANSTAT	PIC18F6X8X	PIC18F8X8X	100- 000-	100- 000-	uuu- uuu-
RXB0D7	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXB0D6	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXB0D5	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXB0D4	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXB0D3	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXB0D2	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXB0D1	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXB0D0	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXB0DLC	PIC18F6X8X	PIC18F8X8X	-xxx xxxx	-uuu uuuu	-uuu uuuu

Legend: u = unchanged, x = unknown, - = unimplemented bit, read as '0', q = value depends on condition. Shaded cells indicate conditions do not apply for the designated device.

Note 1: One or more bits in the INTCONx or PIRx registers will be affected (to cause wake-up).

2: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the PC is loaded with the interrupt vector (0008h or 0018h).

3: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the TOSU, TOSH and TOSL are updated with the current value of the PC. The STKPTR is modified to point to the next location in the hardware stack.

4: See Table 3-2 for Reset value for specific condition.

5: Bit 6 of PORTA, LATA, and TRISA are enabled in ECIO and RCIO Oscillator modes only. In all other oscillator modes, they are disabled and read '0'.

6: Bit 6 of PORTA, LATA and TRISA are not available on all devices. When unimplemented, they read '0'.

7: This register reads all '0's until ECAN is set up in Mode 1 or Mode 2.

TABLE 3-3: INITIALIZATION CONDITIONS FOR ALL REGISTERS (CONTINUED)

Register	Applicable Devices		Power-on Reset, Brown-out Reset	$\overline{\text{MCLR}}$ Resets WDT Reset RESET Instruction Stack Resets	Wake-up via WDT or Interrupt
RXB0EIDL	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXB0EIDH	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXB0SIDL	PIC18F6X8X	PIC18F8X8X	xxxx x-xx	uuuu u-uu	uuuu u-uu
RXB0SIDH	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXB0CON	PIC18F6X8X	PIC18F8X8X	000- 0000	000- 0000	uuu- uuuu
RXB1D7	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXB1D6	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXB1D5	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXB1D4	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXB1D3	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXB1D2	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXB1D1	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXB1D0	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXB1DLC	PIC18F6X8X	PIC18F8X8X	-xxx xxxx	-uuu uuuu	-uuu uuuu
RXB1EIDL	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXB1EIDH	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXB1SIDL	PIC18F6X8X	PIC18F8X8X	xxxx x-xx	uuuu u-uu	uuuu u-uu
RXB1SIDH	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXB1CON	PIC18F6X8X	PIC18F8X8X	000- 0000	000- 0000	uuu- uuuu
TXB0D7	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
TXB0D6	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
TXB0D5	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
TXB0D4	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
TXB0D3	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
TXB0D2	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
TXB0D1	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
TXB0D0	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
TXB0DLC	PIC18F6X8X	PIC18F8X8X	-x-- xxxx	-u-- uuuu	-u-- uuuu
TXB0EIDL	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
TXB0EIDH	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	-uuu uuuu
TXB0SIDL	PIC18F6X8X	PIC18F8X8X	xxx- x-xx	uuu- u-uu	uuu- u-uu

Legend: u = unchanged, x = unknown, - = unimplemented bit, read as '0', q = value depends on condition. Shaded cells indicate conditions do not apply for the designated device.

Note 1: One or more bits in the INTCONx or PIRx registers will be affected (to cause wake-up).

2: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the PC is loaded with the interrupt vector (0008h or 0018h).

3: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the TOSU, TOSH and TOSL are updated with the current value of the PC. The STKPTR is modified to point to the next location in the hardware stack.

4: See Table 3-2 for Reset value for specific condition.

5: Bit 6 of PORTA, LATA, and TRISA are enabled in ECIO and RCIO Oscillator modes only. In all other oscillator modes, they are disabled and read '0'.

6: Bit 6 of PORTA, LATA and TRISA are not available on all devices. When unimplemented, they read '0'.

7: This register reads all '0's until ECAN is set up in Mode 1 or Mode 2.

TABLE 3-3: INITIALIZATION CONDITIONS FOR ALL REGISTERS (CONTINUED)

Register	Applicable Devices		Power-on Reset, Brown-out Reset	$\overline{\text{MCLR}}$ Resets WDT Reset RESET Instruction Stack Resets	Wake-up via WDT or Interrupt
TXB0SIDH	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
TXB0CON	PIC18F6X8X	PIC18F8X8X	0000 0-00	0000 0-00	uuuu u-uu
TXB1D7	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
TXB1D6	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
TXB1D5	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
TXB1D4	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
TXB1D3	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
TXB1D2	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
TXB1D1	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
TXB1D0	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
TXB1DLC	PIC18F6X8X	PIC18F8X8X	-x-- xxxx	-u-- uuuu	-u-- uuuu
TXB1EIDL	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
TXB1EIDH	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
TXB1SIDL	PIC18F6X8X	PIC18F8X8X	xxx- x-xx	uuu- u-uu	uuu- uu-u
TXB1SIDH	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	-uuu uuuu
TXB1CON	PIC18F6X8X	PIC18F8X8X	0000 0-00	0000 0-00	uuuu u-uu
TXB2D7	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	0uuu uuuu
TXB2D6	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	0uuu uuuu
TXB2D5	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	0uuu uuuu
TXB2D4	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	0uuu uuuu
TXB2D3	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	0uuu uuuu
TXB2D2	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	0uuu uuuu
TXB2D1	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	0uuu uuuu
TXB2D0	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	0uuu uuuu
TXB2DLC	PIC18F6X8X	PIC18F8X8X	-x-- xxxx	-u-- uuuu	-u-- uuuu
TXB2EIDL	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
TXB2EIDH	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
TXB2SIDL	PIC18F6X8X	PIC18F8X8X	xxx- x-xx	uuu- u-uu	uuu- u-uu
TXB2SIDH	PIC18F6X8X	PIC18F8X8X	xxx- x-xx	uuu- u-uu	uuu- u-uu
TXB2CON	PIC18F6X8X	PIC18F8X8X	0000 0-00	0000 0-00	uuuu u-uu
RXM1EIDL	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu

Legend: u = unchanged, x = unknown, - = unimplemented bit, read as '0', q = value depends on condition. Shaded cells indicate conditions do not apply for the designated device.

Note 1: One or more bits in the INTCONx or PIRx registers will be affected (to cause wake-up).

2: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the PC is loaded with the interrupt vector (0008h or 0018h).

3: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the TOSU, TOSH and TOSL are updated with the current value of the PC. The STKPTR is modified to point to the next location in the hardware stack.

4: See Table 3-2 for Reset value for specific condition.

5: Bit 6 of PORTA, LATA, and TRISA are enabled in ECIO and RCIO Oscillator modes only. In all other oscillator modes, they are disabled and read '0'.

6: Bit 6 of PORTA, LATA and TRISA are not available on all devices. When unimplemented, they read '0'.

7: This register reads all '0's until ECAN is set up in Mode 1 or Mode 2.

TABLE 3-3: INITIALIZATION CONDITIONS FOR ALL REGISTERS (CONTINUED)

Register	Applicable Devices		Power-on Reset, Brown-out Reset	$\overline{MCLR}$ Resets WDT Reset RESET Instruction Stack Resets	Wake-up via WDT or Interrupt
RXM1EIDH	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXM1SIDL	PIC18F6X8X	PIC18F8X8X	xxx- x-xx	uuu- u-uu	uuu- u-uu
RXM1SIDH	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXM0EIDL	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXM0EIDH	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXM0SIDL	PIC18F6X8X	PIC18F8X8X	xxx- x-xx	uuu- u-uu	uuu- u-uu
RXM0SIDH	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF5EIDL	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF5EIDH	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF5SIDL	PIC18F6X8X	PIC18F8X8X	xxx- x-xx	uuu- u-uu	uuu- u-uu
RXF5SIDH	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF4EIDL	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF4EIDH	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF4SIDL	PIC18F6X8X	PIC18F8X8X	xxx- x-xx	uuu- u-uu	uuu- u-uu
RXF4SIDH	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF3EIDL	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF3EIDH	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF3SIDL	PIC18F6X8X	PIC18F8X8X	xxx- x-xx	uuu- u-uu	uuu- u-uu
RXF3SIDH	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF2EIDL	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF2EIDH	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF2SIDL	PIC18F6X8X	PIC18F8X8X	xxx- x-xx	uuu- u-uu	uuu- u-uu
RXF2SIDH	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF1EIDL	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF1EIDH	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF1SIDL	PIC18F6X8X	PIC18F8X8X	xxx- x-xx	uuu- u-uu	uuu- u-uu
RXF1SIDH	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF0EIDL	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF0EIDH	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF0SIDL	PIC18F6X8X	PIC18F8X8X	xxx- x-xx	uuu- u-uu	uuu- u-uu
RXF0SIDH	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu

Legend: u = unchanged, x = unknown, - = unimplemented bit, read as '0', q = value depends on condition. Shaded cells indicate conditions do not apply for the designated device.

Note 1: One or more bits in the INTCONx or PIRx registers will be affected (to cause wake-up).

2: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the PC is loaded with the interrupt vector (0008h or 0018h).

3: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the TOSU, TOSH and TOSL are updated with the current value of the PC. The STKPTR is modified to point to the next location in the hardware stack.

4: See Table 3-2 for Reset value for specific condition.

5: Bit 6 of PORTA, LATA, and TRISA are enabled in ECIO and RCIO Oscillator modes only. In all other oscillator modes, they are disabled and read '0'.

6: Bit 6 of PORTA, LATA and TRISA are not available on all devices. When unimplemented, they read '0'.

7: This register reads all '0's until ECAN is set up in Mode 1 or Mode 2.

TABLE 3-3: INITIALIZATION CONDITIONS FOR ALL REGISTERS (CONTINUED)

Register	Applicable Devices		Power-on Reset, Brown-out Reset	$\overline{\text{MCLR}}$ Resets WDT Reset RESET Instruction Stack Resets	Wake-up via WDT or Interrupt
B5D7(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
B5D6(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
B5D5(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
B5D4(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
B5D3(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
B5D2(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
B5D1(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
B5D0(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
B5DLC(7)	PIC18F6X8X	PIC18F8X8X	-xxx xxxx	-uuu uuuu	-uuu uuuu
B5EIDL(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
B5EIDH(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
B5SIDL(7)	PIC18F6X8X	PIC18F8X8X	xxxx x-xx	uuuu u-uu	uuuu u-uu
B5SIDH(7)	PIC18F6X8X	PIC18F8X8X	xxxx x-xx	uuuu u-uu	uuuu u-uu
B5CON(7)	PIC18F6X8X	PIC18F8X8X	0000 0000	0000 0000	uuuu uuuu
B4D7(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
B4D6(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
B4D5(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
B4D4(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
B4D3(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
B4D2(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
B4D1(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
B4D0(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
B4DLC(7)	PIC18F6X8X	PIC18F8X8X	-xxx xxxx	-uuu uuuu	-uuu uuuu
B4EIDL(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
B4EIDH(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
B4SIDL(7)	PIC18F6X8X	PIC18F8X8X	xxxx x-xx	uuuu u-uu	uuuu u-uu
B4SIDH(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
B4CON(7)	PIC18F6X8X	PIC18F8X8X	0000 0000	0000 0000	uuuu uuuu
B3D7(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
B3D6(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
B3D5(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu

Legend: u = unchanged, x = unknown, - = unimplemented bit, read as '0', q = value depends on condition. Shaded cells indicate conditions do not apply for the designated device.

Note 1: One or more bits in the INTCONx or PIRx registers will be affected (to cause wake-up).

2: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the PC is loaded with the interrupt vector (0008h or 0018h).

3: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the TOSU, TOSH and TOSL are updated with the current value of the PC. The STKPTR is modified to point to the next location in the hardware stack.

4: See Table 3-2 for Reset value for specific condition.

5: Bit 6 of PORTA, LATA, and TRISA are enabled in ECIO and RCIO Oscillator modes only. In all other oscillator modes, they are disabled and read '0'.

6: Bit 6 of PORTA, LATA and TRISA are not available on all devices. When unimplemented, they read '0'.

7: This register reads all '0's until ECAN is set up in Mode 1 or Mode 2.

TABLE 3-3: INITIALIZATION CONDITIONS FOR ALL REGISTERS (CONTINUED)

Register	Applicable Devices		Power-on Reset, Brown-out Reset	$\overline{MCLR}$ Resets WDT Reset RESET Instruction Stack Resets	Wake-up via WDT or Interrupt
B3D4(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
B3D3(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
B3D2(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
B3D1(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
B3D0(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
B3DLC(7)	PIC18F6X8X	PIC18F8X8X	-xxx xxxx	-uuu uuuu	-uuu uuuu
B3EIDL(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
B3EIDH(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
B3SIDL(7)	PIC18F6X8X	PIC18F8X8X	xxxx x-xx	uuuu u-uu	uuuu u-uu
B3SIDH(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
B3CON(7)	PIC18F6X8X	PIC18F8X8X	0000 0000	0000 0000	uuuu uuuu
B2D7(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
B2D6(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
B2D5(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
B2D4(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
B2D3(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
B2D2(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
B2D1(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
B2D0(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
B2DLC(7)	PIC18F6X8X	PIC18F8X8X	-xxx xxxx	-uuu uuuu	-uuu uuuu
B2EIDL(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
B2EIDH(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
B2SIDL(7)	PIC18F6X8X	PIC18F8X8X	xxxx x-xx	uuuu u-uu	uuuu u-uu
B2SIDH(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
B2CON(7)	PIC18F6X8X	PIC18F8X8X	0000 0000	0000 0000	uuuu uuuu
B1D7(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
B1D6(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
B1D5(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
B1D4(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
B1D3(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
B1D2(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu

Legend: u = unchanged, x = unknown, - = unimplemented bit, read as '0', q = value depends on condition. Shaded cells indicate conditions do not apply for the designated device.

Note 1: One or more bits in the INTCONx or PIRx registers will be affected (to cause wake-up).

2: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the PC is loaded with the interrupt vector (0008h or 0018h).

3: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the TOSU, TOSH and TOSL are updated with the current value of the PC. The STKPTR is modified to point to the next location in the hardware stack.

4: See Table 3-2 for Reset value for specific condition.

5: Bit 6 of PORTA, LATA, and TRISA are enabled in ECIO and RCIO Oscillator modes only. In all other oscillator modes, they are disabled and read '0'.

6: Bit 6 of PORTA, LATA and TRISA are not available on all devices. When unimplemented, they read '0'.

7: This register reads all '0's until ECAN is set up in Mode 1 or Mode 2.

TABLE 3-3: INITIALIZATION CONDITIONS FOR ALL REGISTERS (CONTINUED)

Register	Applicable Devices		Power-on Reset, Brown-out Reset	MCLR Resets WDT Reset RESET Instruction Stack Resets	Wake-up via WDT or Interrupt
B1D1(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
B1D0(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
B1DLC(7)	PIC18F6X8X	PIC18F8X8X	-xxx xxxx	-uuu uuuu	-uuu uuuu
B1EIDL(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
B1EIDH(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
B1SIDL(7)	PIC18F6X8X	PIC18F8X8X	xxxx x-xx	uuuu u-uu	uuuu u-uu
B1SIDH(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
B1CON(7)	PIC18F6X8X	PIC18F8X8X	0000 0000	0000 0000	uuuu uuuu
B0D7(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
B0D6(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
B0D5(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
B0D4(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
B0D3(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
B0D2(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
B0D1(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
B0D0(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
B0DLC(7)	PIC18F6X8X	PIC18F8X8X	-xxx xxxx	-uuu uuuu	-uuu uuuu
B0EIDL(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
B0EIDH(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
B0SIDL(7)	PIC18F6X8X	PIC18F8X8X	xxxx x-xx	uuuu u-uu	uuuu u-uu
B0SIDH(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
B0CON(7)	PIC18F6X8X	PIC18F8X8X	0000 0000	0000 0000	uuuu uuuu
TXBIE(7)	PIC18F6X8X	PIC18F8X8X	---0 00--	---u uu--	---u uu--
BIE0(7)	PIC18F6X8X	PIC18F8X8X	0000 0000	0000 0000	uuuu uuuu
BSEL0(7)	PIC18F6X8X	PIC18F8X8X	0000 00--	0000 00--	uuuu uu--
MSEL3(7)	PIC18F6X8X	PIC18F8X8X	0000 0000	0000 0000	uuuu uuuu
MSEL2(7)	PIC18F6X8X	PIC18F8X8X	0000 0000	0000 0000	uuuu uuuu
MSEL1(7)	PIC18F6X8X	PIC18F8X8X	0000 0101	0000 0101	uuuu uuuu
MSEL0(7)	PIC18F6X8X	PIC18F8X8X	0101 0000	0101 0000	uuuu uuuu
SDFLC(7)	PIC18F6X8X	PIC18F8X8X	---0 0000	---0 0000	-u-- uuuu
RXFCON1(7)	PIC18F6X8X	PIC18F8X8X	0000 0000	0000 0000	uuuu uuuu

Legend: u = unchanged, x = unknown, - = unimplemented bit, read as '0', q = value depends on condition. Shaded cells indicate conditions do not apply for the designated device.

Note 1: One or more bits in the INTCONx or PIRx registers will be affected (to cause wake-up).

2: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the PC is loaded with the interrupt vector (0008h or 0018h).

3: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the TOSU, TOSH and TOSL are updated with the current value of the PC. The STKPTR is modified to point to the next location in the hardware stack.

4: See Table 3-2 for Reset value for specific condition.

5: Bit 6 of PORTA, LATA, and TRISA are enabled in ECIO and RCIO Oscillator modes only. In all other oscillator modes, they are disabled and read '0'.

6: Bit 6 of PORTA, LATA and TRISA are not available on all devices. When unimplemented, they read '0'.

7: This register reads all '0's until ECAN is set up in Mode 1 or Mode 2.

TABLE 3-3: INITIALIZATION CONDITIONS FOR ALL REGISTERS (CONTINUED)

Register	Applicable Devices		Power-on Reset, Brown-out Reset	$\overline{\text{MCLR}}$ Resets WDT Reset `RESET` Instruction Stack Resets	Wake-up via WDT or Interrupt
RXFCON0(7)	PIC18F6X8X	PIC18F8X8X	0000 0000	0000 0000	uuuu uuuu
RXFBCON7(7)	PIC18F6X8X	PIC18F8X8X	0000 0000	0000 0000	uuuu uuuu
RXFBCON6(7)	PIC18F6X8X	PIC18F8X8X	0000 0000	0000 0000	uuuu uuuu
RXFBCON5(7)	PIC18F6X8X	PIC18F8X8X	0000 0000	0000 0000	uuuu uuuu
RXFBCON4(7)	PIC18F6X8X	PIC18F8X8X	0000 0000	0000 0000	uuuu uuuu
RXFBCON3(7)	PIC18F6X8X	PIC18F8X8X	0000 0000	0000 0000	uuuu uuuu
RXFBCON2(7)	PIC18F6X8X	PIC18F8X8X	0001 0001	0001 0001	uuuu uuuu
RXFBCON1(7)	PIC18F6X8X	PIC18F8X8X	0001 0001	0001 0001	uuuu uuuu
RXFBCON0(7)	PIC18F6X8X	PIC18F8X8X	0000 0000	0000 0000	uuuu uuuu
RXF15EIDL(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF15EIDH(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF15SIDL(7)	PIC18F6X8X	PIC18F8X8X	xxx- x-xx	uuu- u-uu	uuu- u-uu
RXF15SIDH(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF14EIDL(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF14EIDH(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF14SIDL(7)	PIC18F6X8X	PIC18F8X8X	xxx- x-xx	uuu- u-uu	uuu- u-uu
RXF14SIDH(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF13EIDL(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF13EIDH(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF13SIDL(7)	PIC18F6X8X	PIC18F8X8X	xxx- x-xx	uuu- u-uu	uuu- u-uu
RXF13SIDH(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF12EIDL(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF12EIDH(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF12SIDL(7)	PIC18F6X8X	PIC18F8X8X	xxx- x-xx	uuu- u-uu	uuu- u-uu
RXF12SIDH(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF11EIDL(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF11EIDH(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF11SIDL(7)	PIC18F6X8X	PIC18F8X8X	xxx- x-xx	uuu- u-uu	uuu- u-uu
RXF11SIDH(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF10EIDL(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	-uuu uuuu
RXF10EIDH(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	-uuu uuuu

Legend: u = unchanged, x = unknown, - = unimplemented bit, read as '0', q = value depends on condition. Shaded cells indicate conditions do not apply for the designated device.

Note 1: One or more bits in the INTCONx or PIRx registers will be affected (to cause wake-up).

2: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the PC is loaded with the interrupt vector (0008h or 0018h).

3: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the TOSU, TOSH and TOSL are updated with the current value of the PC. The STKPTR is modified to point to the next location in the hardware stack.

4: See Table 3-2 for Reset value for specific condition.

5: Bit 6 of PORTA, LATA, and TRISA are enabled in ECIO and RCIO Oscillator modes only. In all other oscillator modes, they are disabled and read '0'.

6: Bit 6 of PORTA, LATA and TRISA are not available on all devices. When unimplemented, they read '0'.

7: This register reads all '0's until ECAN is set up in Mode 1 or Mode 2.

TABLE 3-3: INITIALIZATION CONDITIONS FOR ALL REGISTERS (CONTINUED)

Register	Applicable Devices		Power-on Reset, Brown-out Reset	$\overline{MCLR}$ Resets WDT Reset RESET Instruction Stack Resets	Wake-up via WDT or Interrupt
RXF10SIDL(7)	PIC18F6X8X	PIC18F8X8X	xxx- x-xx	uuu- u-uu	-uuu uuuu
RXF10SIDH(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	-uuu uuuu
RXF9EIDL(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	-uuu uuuu
RXF9EIDH(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	-uuu uuuu
RXF9SIDL(7)	PIC18F6X8X	PIC18F8X8X	xxx- x-xx	uuu- u-uu	-uuu uuuu
RXF9SIDH(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	-uuu uuuu
RXF8EIDL(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	-uuu uuuu
RXF8EIDH(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	-uuu uuuu
RXF8SIDL(7)	PIC18F6X8X	PIC18F8X8X	xxx- x-xx	uuu- u-uu	-uuu uuuu
RXF8SIDH(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	-uuu uuuu
RXF7EIDL(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	-uuu uuuu
RXF7EIDH(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	-uuu uuuu
RXF7SIDL(7)	PIC18F6X8X	PIC18F8X8X	xxx- x-xx	uuu- u-uu	-uuu uuuu
RXF7SIDH(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	-uuu uuuu
RXF6EIDL(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	-uuu uuuu
RXF6EIDH(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	-uuu uuuu
RXF6SIDL(7)	PIC18F6X8X	PIC18F8X8X	xxx- x-xx	uuu- u-uu	-uuu uuuu
RXF6SIDH(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	-uuu uuuu

Legend: u = unchanged, x = unknown, - = unimplemented bit, read as '0', q = value depends on condition. Shaded cells indicate conditions do not apply for the designated device.

Note 1: One or more bits in the INTCONx or PIRx registers will be affected (to cause wake-up).

2: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the PC is loaded with the interrupt vector (0008h or 0018h).

3: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the TOSU, TOSH and TOSL are updated with the current value of the PC. The STKPTR is modified to point to the next location in the hardware stack.

4: See Table 3-2 for Reset value for specific condition.

5: Bit 6 of PORTA, LATA, and TRISA are enabled in ECIO and RCIO Oscillator modes only. In all other oscillator modes, they are disabled and read '0'.

6: Bit 6 of PORTA, LATA and TRISA are not available on all devices. When unimplemented, they read '0'.

7: This register reads all '0's until ECAN is set up in Mode 1 or Mode 2.

FIGURE 3-3: TIME-OUT SEQUENCE ON POWER-UP (MCLR TIED TO VDD VIA 1 kΩ RESISTOR)

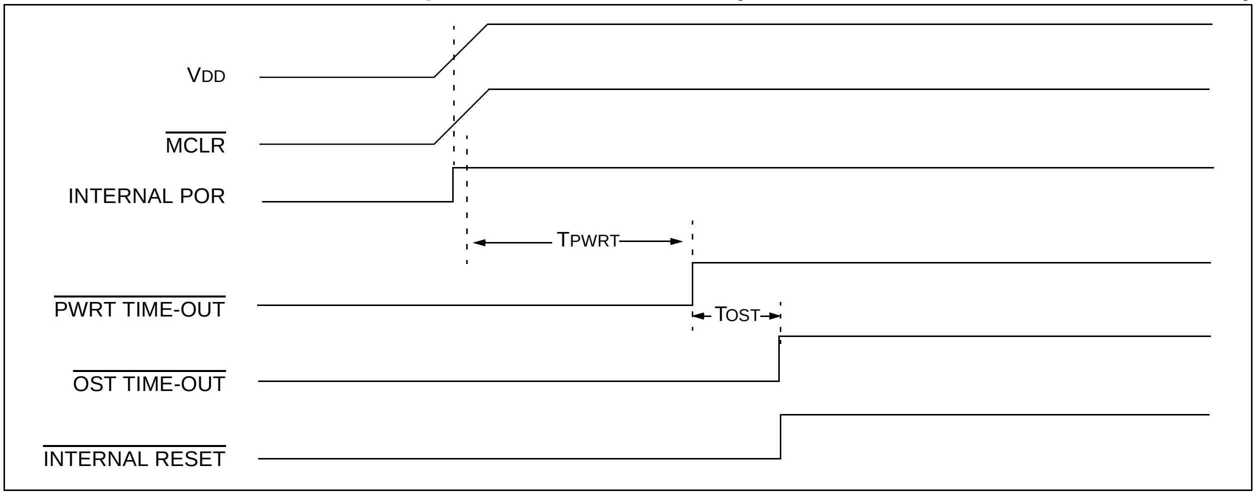


FIGURE 3-4: TIME-OUT SEQUENCE ON POWER-UP (MCLR NOT TIED TO VDD): CASE 1

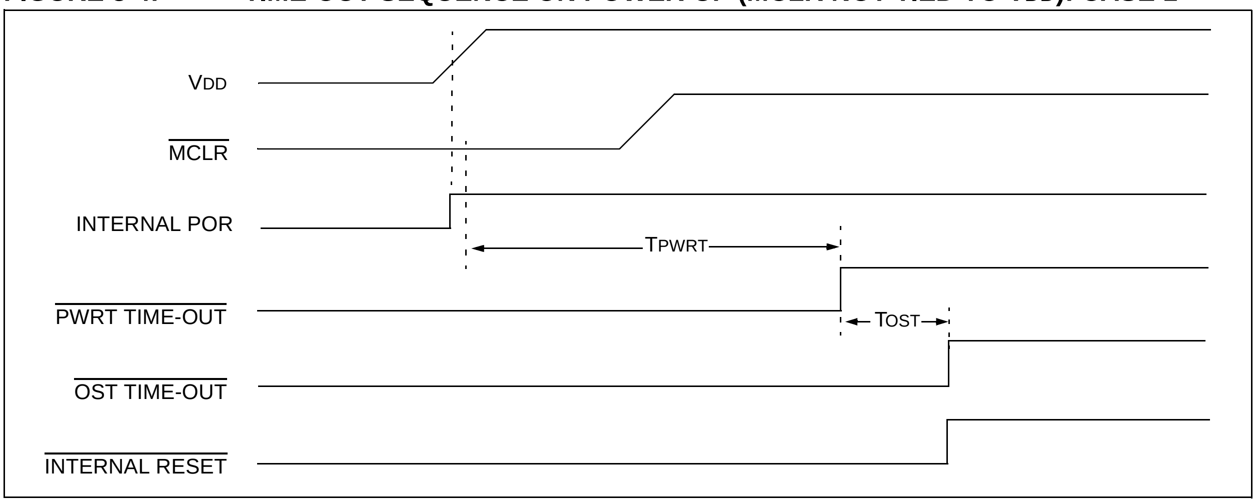


FIGURE 3-5: TIME-OUT SEQUENCE ON POWER-UP (MCLR NOT TIED TO VDD): CASE 2

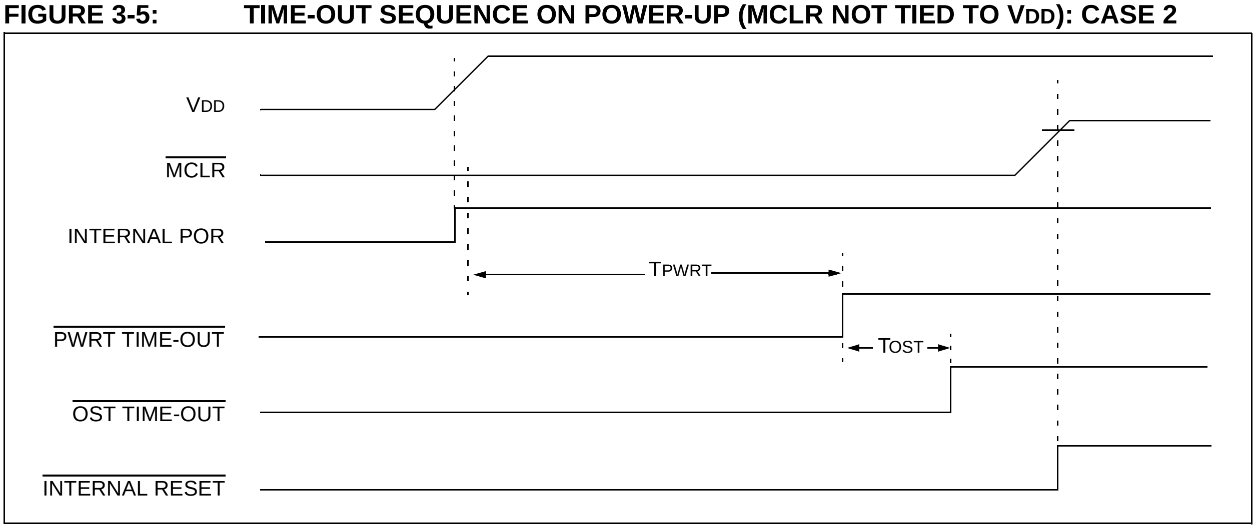


FIGURE 3-6: SLOW RISE TIME ($\overline{MCLR}$ TIED TO VDD VIA 1 kΩ RESISTOR)

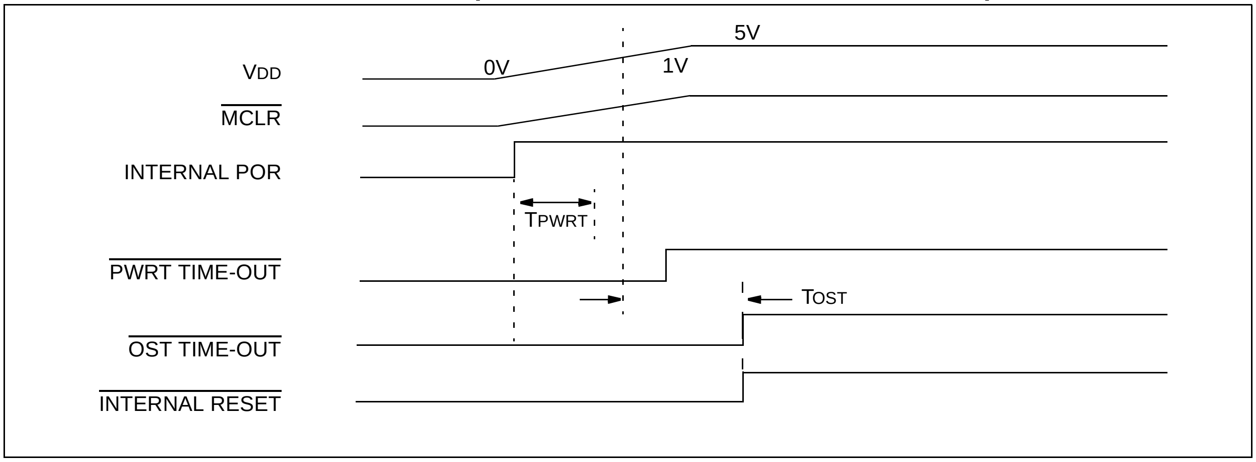
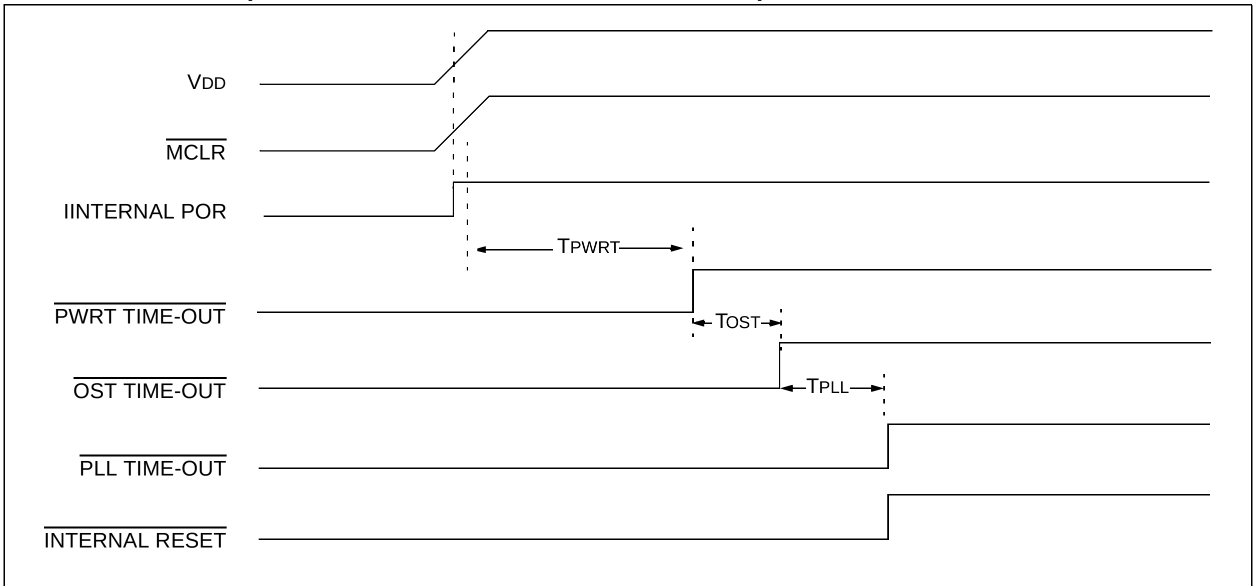


FIGURE 3-7: TIME-OUT SEQUENCE ON POR W/ PLL ENABLED ($\overline{MCLR}$ TIED TO VDD VIA 1 kΩ RESISTOR)



Note: TOST = 1024 clock cycles.
TPLL ≈ 2 ms max. First three stages of the PWRT timer.

4.0 MEMORY ORGANIZATION

There are three memory blocks in PIC18F6585/8585/6680/8680 devices. They are:

- Program Memory
- Data RAM
- Data EEPROM

Data and program memory use separate busses which allows for concurrent access of these blocks. Additional detailed information for Flash program memory and data EEPROM is provided in **Section 5.0 “Flash Program Memory”** and **Section 7.0 “Data EEPROM Memory”**, respectively.

In addition to on-chip Flash, the PIC18F8X8X devices are also capable of accessing external program memory through an external memory bus. Depending on the selected operating mode (discussed in **Section 4.1.1 “PIC18F8X8X Program Memory Modes”**), the controllers may access either internal or external program memory exclusively, or both internal and external memory in selected blocks. Additional information on the external memory interface is provided in **Section 6.0 “External Memory Interface”**.

4.1 Program Memory Organization

A 21-bit program counter is capable of addressing the 2-Mbyte program memory space. Accessing a location between the physically implemented memory and the 2-Mbyte address will cause a read of all ‘0’s (a NOP instruction).

The PIC18F6585 and PIC18F8585 each have 48 Kbytes of on-chip Flash memory, while the PIC18F6680 and PIC18F8680 have 64 Kbytes of Flash. This means that PIC18FX585 devices can store internally up to 24,576 single-word instructions and PIC18FX680 devices can store up to 32,768 single-word instructions.

The Reset vector address is at 0000h and the interrupt vector addresses are at 0008h and 0018h.

Figure 4-1 shows the program memory map for PIC18F6585/8585 devices while Figure 4-2 shows the program memory map for PIC18F6680/8680 devices.

4.1.1 PIC18F8X8X PROGRAM MEMORY MODES

PIC18F8X8X devices differ significantly from their PIC18 predecessors in their utilization of program memory. In addition to available on-chip Flash program memory, these controllers can also address up to 2 Mbytes of external program memory through the external memory interface. There are four distinct operating modes available to the controllers:

- Microprocessor (MP)
- Microprocessor with Boot Block (MPBB)
- Extended Microcontroller (EMC)
- Microcontroller (MC)

The Program Memory mode is determined by setting the two Least Significant bits of the CONFIG3L configuration byte, as shown in Register 4-1. (See also **Section 24.1 “Configuration Bits”** for additional details on the device configuration bits.)

The Program Memory modes operate as follows:

- The **Microprocessor Mode** permits access only to external program memory; the contents of the on-chip Flash memory are ignored. The 21-bit program counter permits access to a 2-MByte linear program memory space.
- The **Microprocessor with Boot Block Mode** accesses on-chip Flash memory from addresses 000000h to 0007FFh. Above this, external program memory is accessed all the way up to the 2-MByte limit. Program execution automatically switches between the two memories as required.
- The **Microcontroller Mode** accesses only on-chip Flash memory. Attempts to read above the physical limit of the on-chip Flash (0BFFFh for the PIC18F8585, 0FFFFh for the PIC18F8680) causes a read of all ‘0’s (a NOP instruction). The Microcontroller mode is the only operating mode available to PIC18F6X8X devices.
- The **Extended Microcontroller Mode** allows access to both internal and external program memories as a single block. The device can access its entire on-chip Flash memory; above this, the device accesses external program memory up to the 2-MByte program space limit. As with Boot Block mode, execution automatically switches between the two memories as required.

In all modes, the microcontroller has complete access to data RAM and EEPROM.

Figure 4-3 compares the memory maps of the different Program Memory modes. The differences between on-chip and external memory access limitations are more fully explained in Table 4-1.

FIGURE 4-1: INTERNAL PROGRAM MEMORY MAP AND STACK FOR PIC18F6585/8585

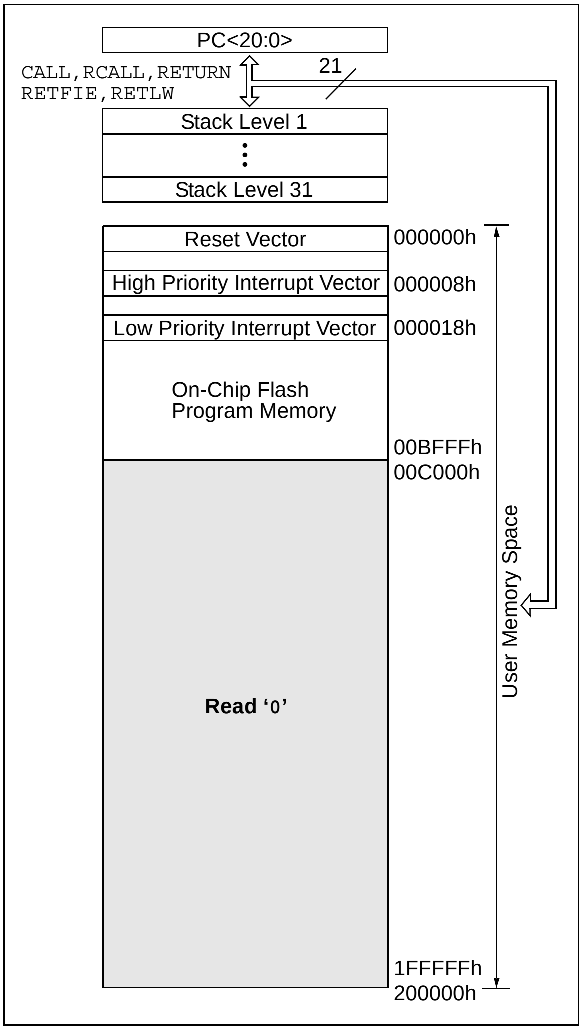


FIGURE 4-2: INTERNAL PROGRAM MEMORY MAP AND STACK FOR PIC18F6680/8680

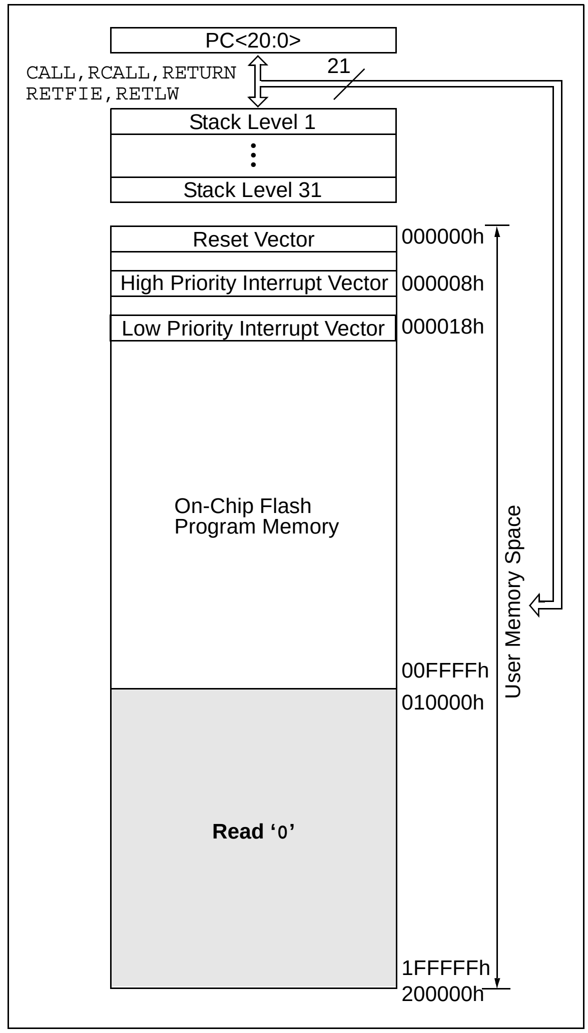


TABLE 4-1: MEMORY ACCESS FOR PIC18F8X8X PROGRAM MEMORY MODES

Operating Mode	Internal Program Memory			External Program Memory		
	Execution From	Table Read From	Table Write To	Execution From	Table Read From	Table Write To
Microprocessor	No Access	No Access	No Access	Yes	Yes	Yes
Microprocessor w/ Boot Block	Yes	Yes	Yes	Yes	Yes	Yes
Microcontroller	Yes	Yes	Yes	No Access	No Access	No Access
Extended Microcontroller	Yes	Yes	Yes	Yes	Yes	Yes

REGISTER 4-1: CONFIG3L CONFIGURATION BYTE

R/P-1	U-0	U-0	U-0	U-0	U-0	R/P-1	R/P-1
WAIT	—	—	—	—	—	PM1	PM0
bit 7							bit 0

bit 7 **WAIT:** External Bus Data Wait Enable bit
1 = Wait selections unavailable, device will not wait
0 = Wait programmed by WAIT1 and WAIT0 bits of MEMCOM register (MEMCOM<5:4>)

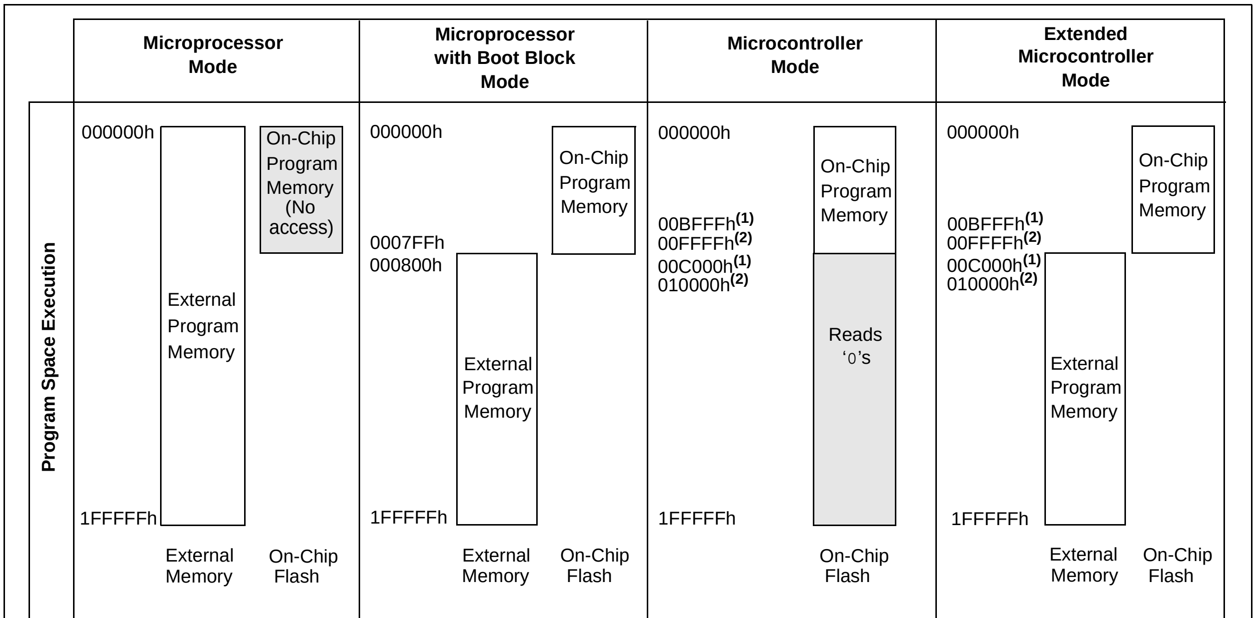
bit 6-2 **Unimplemented:** Read as '0'

bit 1-0 **PM1:PM0:** Processor Data Memory Mode Select bits
11 = Microcontroller mode
10 = Microprocessor mode
01 = Microcontroller with Boot Block mode
00 = Extended Microcontroller mode

Legend:

R = Readable bit	P = Programmable bit	U = Unimplemented bit, read as '0'	
- n = Value after erase	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

FIGURE 4-3: MEMORY MAPS FOR PIC18F8X8X PROGRAM MEMORY MODES



Note 1: PIC18F6585 and PIC18F8585.
2: PIC18F6680 and PIC18F8680.

4.2 Return Address Stack

The return address stack allows any combination of up to 31 program calls and interrupts to occur. The PC (Program Counter) 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, with the stack pointer initialized to 00000b after all Resets. There is no RAM associated with stack pointer 00000b. This is only a Reset value. During a `CALL` type instruction causing a push onto the stack, the stack pointer is first incremented and the RAM location pointed to by the stack pointer is written with the contents of the PC. During a `RETURN` type instruction causing a pop from the stack, the contents of the RAM location pointed to by the STKPTR are transferred to the PC and then the stack pointer is decremented.

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 SFR registers. Data can also be pushed to or popped from the stack, using the top-of-stack SFRs. Status bits indicate if the stack pointer is at or beyond the 31 levels provided.

4.2.1 TOP-OF-STACK ACCESS

The top of the stack is readable and writable. Three register locations, TOSU, TOSH and TOSL, hold the contents of the stack location pointed to by the STKPTR register. 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 and TOSL registers. These values can be placed on a user defined software stack. At return time, the software can replace the TOSU, TOSH and TOSL and do a return.

The user must disable the global interrupt enable bits during this time to prevent inadvertent stack operations.

4.2.2 RETURN STACK POINTER (STKPTR)

The STKPTR register contains the stack pointer value, the STKFUL (Stack Full) status bit, and the STKUNF (Stack Underflow) status bits. Register 4-2 shows the STKPTR register. The value of the stack pointer can be 0 through 31. The stack pointer increments when values are pushed onto the stack and decrements when values are popped off the stack. At Reset, the stack pointer value will be '`0`'. The user may read and write the stack pointer value. This feature can be used by a Real-Time Operating System for return stack maintenance.

After the PC is pushed onto the stack 31 times (without popping any values off the stack), the STKFUL bit is set. The STKFUL bit can only be cleared in 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. Refer to **Section 25.0 "Instruction Set Summary"** for a description of the device configuration bits. If STVREN is set (default), the 31st push will push the (PC + 2) value onto the stack, set the STKFUL bit and reset the device. The STKFUL bit will remain set and the stack pointer will be set to '`0`'.

If STVREN is cleared, the STKFUL bit will be set on the 31st push and the stack pointer will increment to 31. Any additional pushes will not overwrite the 31st push and STKPTR will remain at 31.

When the stack has been popped enough times to unload the stack, the next pop will return a value of zero to the PC and sets the STKUNF bit while the stack pointer remains at '`0`'. The STKUNF bit will remain set until cleared in software or a POR occurs.

Note:	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.

REGISTER 4-2: STKPTR REGISTER

R/C-0	R/C-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
STKFUL(1)	STKUNF(1)	—	SP4	SP3	SP2	SP1	SP0
bit 7							bit 0

bit 7 **STKFUL:** Stack Full Flag bit
`1` = Stack became full or overflowed
`0` = Stack has not become full or overflowed

bit 6 **STKUNF:** Stack Underflow Flag bit
`1` = Stack underflow occurred
`0` = Stack underflow did not occur

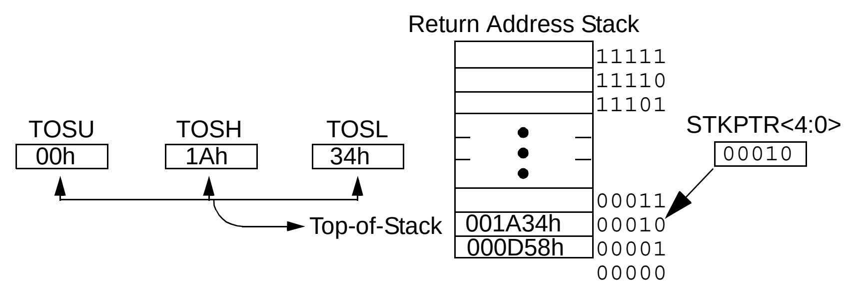
bit 5 **Unimplemented:** Read as '`0`'

bit 4-0 **SP4:SP0:** Stack Pointer Location bits

Note 1: Bit 7 and bit 6 can only be cleared in user software or by a POR.

Legend:			
C = Clearable bit	R = Readable bit	U = Unimplemented bit, read as '0'	W = Writable bit
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

FIGURE 4-4: RETURN ADDRESS STACK AND ASSOCIATED REGISTERS



4.2.3 `PUSH` AND `POP` INSTRUCTIONS

Since the Top-of-Stack (TOS) 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 option. To push the current PC value onto the stack, a `PUSH` instruction can be executed. This will increment the stack pointer and load the current PC value onto the stack. TOSU, TOSH and TOSL can then be modified to place a return address on the stack.

The ability to pull the TOS value off of the stack and replace it with the value that was previously pushed onto the stack, without disturbing normal execution, is achieved by using the `POP` instruction. The `POP` instruction discards the current TOS by decrementing the stack pointer. The previous value pushed onto the stack then becomes the TOS value.

4.2.4 STACK FULL/UNDERFLOW RESETS

These Resets are enabled by programming the STVREN configuration bit. When the STVREN bit is disabled, a full or underflow condition will set the appropriate STKFUL or STKUNF bit, but not cause a device Reset. When the STVREN bit is enabled, a full or underflow condition will set the appropriate STKFUL or STKUNF bit and then cause a device Reset. The STKFUL or STKUNF bits are only cleared by the user software or a POR Reset.

4.3 Fast Register Stack

A “fast interrupt return” option is available for interrupts. A fast register stack is provided for the Status, WREG and BSR registers and is only one in depth. The stack is not readable or writable and is loaded with the current value of the corresponding register when the processor vectors for an interrupt. The values in the registers are then loaded back into the working registers if the `FAST RETURN` instruction is used to return from the interrupt.

A low or high priority interrupt source will push values into the stack registers. If both low and high priority interrupts are enabled, the stack registers cannot be used reliably for 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.

If high priority interrupts are not disabled during low priority interrupts, users must save the key registers in software during a low priority 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 `FAST CALL` instruction must be executed.

Example 4-1 shows a source code example that uses the fast register stack.

EXAMPLE 4-1: 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
```

4.4 PCL, PCLATH and PCLATU

The program counter (PC) specifies the address of the instruction to fetch for execution. The PC is 21 bits wide. The low byte is called the PCL register; this register is readable and writable. The high byte is called the PCH register. This register contains the PC<15:8> bits and is not directly readable or writable; updates to the PCH register may be performed through the PCLATH register. The upper byte is called PCU. This register contains the PC<20:16> bits and is not directly readable or writable; updates to the PCU register may be performed through the PCLATU register.

The PC addresses bytes in the program memory. To prevent the PC from becoming misaligned with word instructions, the LSB of the PCL is fixed to a value of ‘`0`’. The PC increments by 2 to address sequential instructions in the program memory.

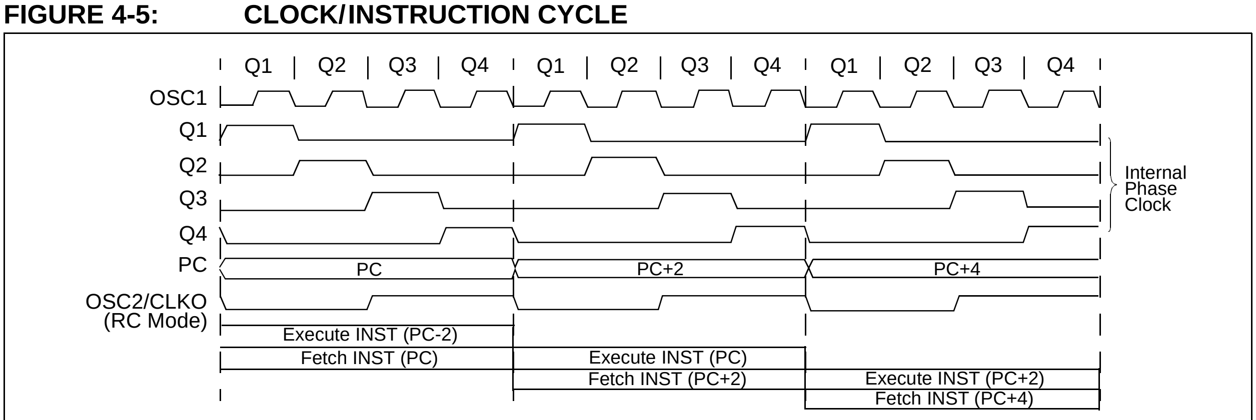
The `CALL`, `RCALL`, `GOTO` and program branch instructions write to the program counter directly. For these instructions, the contents of PCLATH and PCLATU are not transferred to the program counter.

The contents of PCLATH and PCLATU will be transferred to the program counter by an operation that writes PCL. Similarly, the upper two bytes of the program counter will be transferred to PCLATH and PCLATU by an operation that reads PCL. This is useful for computed offsets to the PC (see **Section 4.8.1 “Computed GOTO”**).

4.5 Clocking Scheme/Instruction Cycle

The clock input (from OSC1) is internally divided by four to generate four non-overlapping quadrature clocks, namely Q1, Q2, Q3 and Q4. Internally, the program counter (PC) is incremented every Q1, the instruction is fetched from the program memory and latched into the instruction register in Q4. The instruction is decoded and executed during the following Q1 through Q4. The clocks and instruction execution flow are shown in Figure 4-5.

FIGURE 4-5: CLOCK/INSTRUCTION CYCLE



4.6 Instruction Flow/Pipelining

An "Instruction Cycle" consists of four Q cycles (Q1, Q2, Q3 and Q4). The instruction fetch and execute are pipelined such that fetch takes one instruction cycle, while decode and execute takes another instruction cycle. However, due to the pipelining each instruction effectively executes in one cycle. If an instruction causes the program counter to change (e.g., GOTO), then two cycles are required to complete the instruction (Example 4-2).

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).

EXAMPLE 4-2: INSTRUCTION PIPELINE FLOW

	TCY0	TCY1	TCY2	TCY3	TCY4	TCY5
1. MOVLW 55h	Fetch 1	Execute 1				
2. MOVWF PORTB		Fetch 2	Execute 2			
3. BRA SUB_1			Fetch 3	Execute 3		
4. BSF PORTA, 3 (Forced NOP)				Fetch 4	Flush (NOP)	
5. Instruction @ address SUB_1					Fetch SUB_1	Execute SUB_1

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.

4.7 Instructions in Program Memory

The program memory is addressed in bytes. Instructions are stored as two bytes or four bytes in program memory. The Least Significant Byte (LSB) of an instruction word is always stored in a program memory location with an even address (LSB = 0). Figure 4-6 shows an example of how instruction words are stored in the program memory. To maintain alignment with instruction boundaries, the PC increments in steps of 2 and the LSB will always read '0' (see **Section 4.4 "PCL, PCLATH and PCLATU"**).

The CALL and GOTO instructions have an absolute program memory address embedded into the instruction. Since instructions are always stored on word boundaries, the data contained in the instruction is a word address. The word address is written to PC<20:1> which accesses the desired byte address in program memory. Instruction #2 in Figure 4-6 shows how the instruction "GOTO 000006h" is encoded in the program memory. Program branch instructions which encode a relative address offset operate in the same manner. The offset value stored in a branch instruction represents the number of single-word instructions that the PC will be offset by. **Section 25.0 "Instruction Set Summary"** provides further details of the instruction set.

FIGURE 4-6: INSTRUCTIONS IN PROGRAM MEMORY

Program Memory Byte Locations →

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

4.7.1 TWO-WORD INSTRUCTIONS

The PIC18F6585/8585/6680/8680 devices have four two-word instructions: MOVFF, CALL, GOTO and LFSR. The second word of these instructions has the 4 MSBs set to '1's and is a special kind of NOP instruction. The lower 12 bits of the second word contain data to be used by the instruction. If the first word of the instruction is executed, the data in the second word is accessed. If the second word of the instruction is executed by itself (first word was skipped), it will execute as a NOP. This action is necessary when the two-word instruction is preceded by a conditional instruction that changes the PC. A program example that demonstrates this concept is shown in Example 4-3. Refer to **Section 25.0 "Instruction Set Summary"** for further details of the instruction set.

EXAMPLE 4-3: TWO-WORD INSTRUCTIONS

CASE 1:

Object Code	Source Code
0110 0110 0000 0000	TSTFSZ REG1 ; is RAM location 0?
1100 0001 0010 0011	MOVFF REG1, REG2 ; No, execute 2-word instruction
1111 0100 0101 0110	; 2nd operand holds address of REG2
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
1111 0100 0101 0110	; 2nd operand becomes NOP
0010 0100 0000 0000	ADDWF REG3 ; continue code

4.8 Look-up Tables

Look-up tables are implemented two ways. These are:

- Computed GOTO
- Table Reads

4.8.1 COMPUTED GOTO

A computed GOTO is accomplished by adding an offset to the program counter (ADDWF PCL).

A look-up table can be formed with an ADDWF PCL instruction and a group of RETLW 0xnn instructions. WREG 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 0xnn instructions that returns the value 0xnn to the calling function.

The offset value (value in WREG) specifies the number of bytes that the program counter should advance.

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

4.8.2 TABLE READS/TABLE WRITES

A better method of storing data in program memory allows 2 bytes of data to be stored in each instruction location.

Look-up table data may be stored 2 bytes per program word by using table reads and writes. The Table Pointer (TBLPTR) specifies the byte address and the Table Latch (TABLAT) contains the data that is read from, or written to program memory. Data is transferred to/from program memory, one byte at a time.

A description of the table read/table write operation is shown in **Section 5.0 "Flash Program Memory"**.

4.9 Data Memory Organization

The data memory is implemented as static RAM. Each register in the data memory has a 12-bit address, allowing up to 4096 bytes of data memory. Figure 4-7 shows the data memory organization for the PIC18F6585/8585/6680/8680 devices.

The data memory map is divided into 16 banks that contain 256 bytes each. The lower 4 bits of the Bank Select Register (BSR<3:0>) select which bank will be accessed. The upper 4 bits for the BSR are not implemented.

The data memory contains Special Function Registers (SFR) and General Purpose Registers (GPR). The SFRs are used for control and status of the controller and peripheral functions, while GPRs are used for data storage and scratch pad operations in the user's application. The SFRs start at the last location of Bank 15 (0FFFh) and extend downwards. Any remaining space beyond the SFRs in the Bank may be implemented as GPRs. GPRs start at the first location of Bank 0 and grow upwards. Any read of an unimplemented location will read as '`0`'s.

The entire data memory may be accessed directly or indirectly. Direct addressing may require the use of the BSR register. Indirect addressing requires the use of a File Select Register (FSRn) and a corresponding Indirect File Operand (INDFn). Each FSR holds a 12-bit address value that can be used to access any location in the data memory map without banking.

The instruction set and architecture allow operations across all banks. This may be accomplished by indirect addressing or by the use of the `MOVFF` instruction. The `MOVFF` instruction is a two-word/two-cycle instruction that moves a value from one register to another.

To ensure that commonly used registers (SFRs and select GPRs) can be accessed in a single cycle regardless of the current BSR values, an Access Bank is implemented. A segment of Bank 0 and a segment of Bank 15 comprise the Access RAM. **Section 4.10 "Access Bank"** provides a detailed description of the Access RAM.

4.9.1 GENERAL PURPOSE REGISTER FILE

The register file can be accessed either directly or indirectly. Indirect addressing operates using a File Select Register and corresponding Indirect File Operand. The operation of indirect addressing is shown in **Section 4.12 "Indirect Addressing, INDF and FSR Registers"**.

Enhanced MCU devices may have banked memory in the GPR area. GPRs are not initialized by a Power-on Reset and are unchanged on all other Resets.

Data RAM is available for use as general purpose registers by all instructions. The top section of Bank 15 (0F60h to 0FFFh) contains SFRs. All other banks of data memory contain GPR registers, starting with Bank 0.

4.9.2 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. A list of these registers is given in Table 4-2 and Table 4-3.

The SFRs can be classified into two sets: those associated with the "core" function and those related to the peripheral functions. Those registers related to the "core" are described in this section, while those related to the operation of the peripheral features are described in the section of that peripheral feature. The SFRs are typically distributed among the peripherals whose functions they control.

The unused SFR locations are unimplemented and read as '`0`'s. The addresses for the SFRs are listed in Table 4-2.

FIGURE 4-7: DATA MEMORY MAP FOR PIC18FXX80/XX85 DEVICES

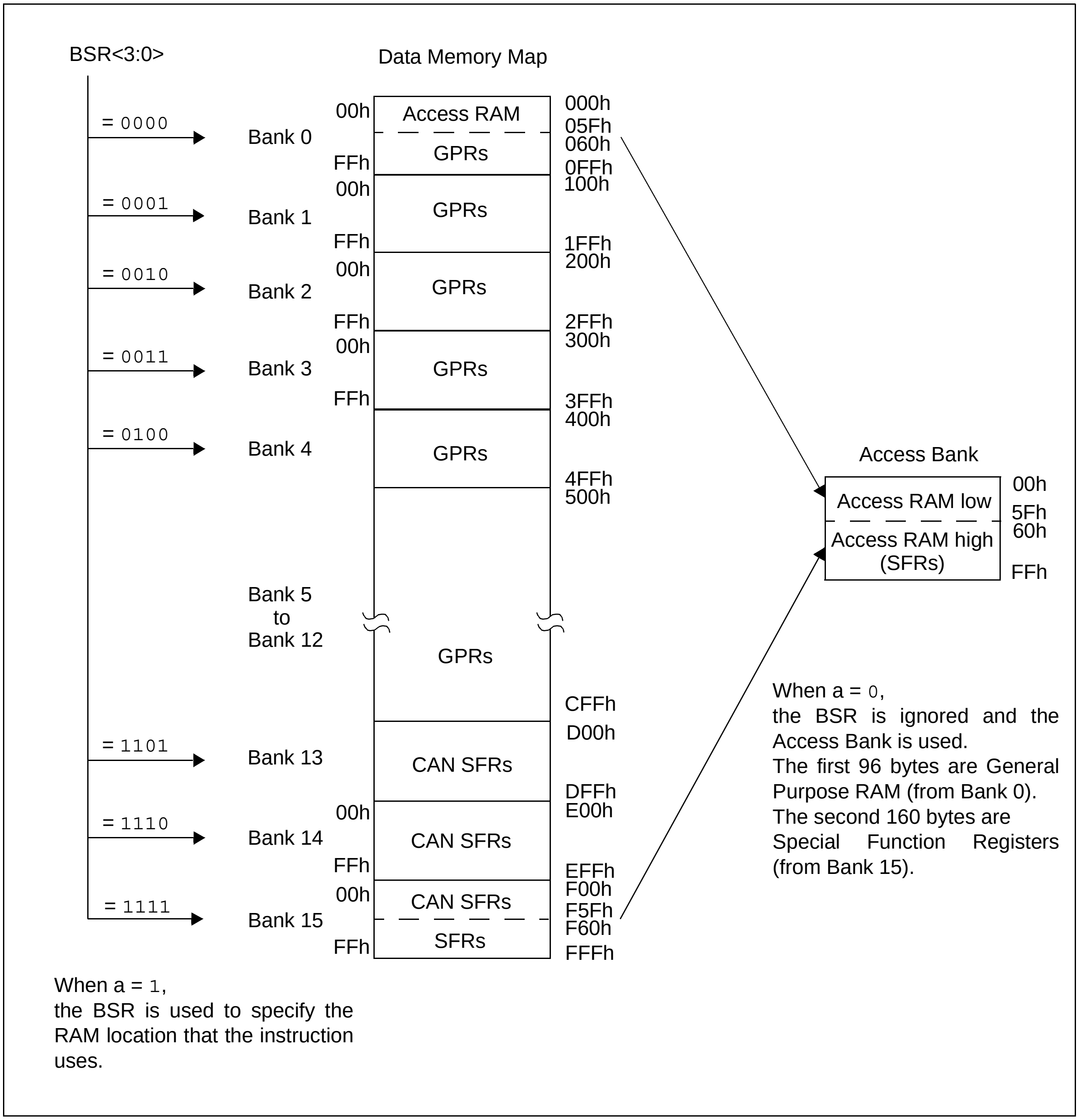


TABLE 4-2: SPECIAL FUNCTION REGISTER MAP

Address	Name	Address	Name	Address	Name	Address	Name
FFFh	TOSU	FDFh	INDF2(3)	FBFh	CCPR1H	F9Fh	IPR1
FFEh	TOSH	FDEh	POSTINC2(3)	FBEh	CCPR1L	F9Eh	PIR1
FFDh	TOSL	FDDh	POSTDEC2(3)	FBDh	CCP1CON	F9Dh	PIE1
FFCh	STKPTR	FDCh	PREINC2(3)	FBCh	CCPR2H	F9Ch	MEMCON(2)
FFBh	PCLATU	FDBh	PLUSW2(3)	FBBh	CCPR2L	F9Bh	—(1)
FFAh	PCLATH	FDAh	FSR2H	FBAh	CCP2CON	F9Ah	TRISJ(2)
FF9h	PCL	FD9h	FSR2L	FB9h	—(1)	F99h	TRISH(2)
FF8h	TBLPTRU	FD8h	STATUS	FB8h	—(1)	F98h	TRISG
FF7h	TBLPTRH	FD7h	TMR0H	FB7h	—(1)	F97h	TRISF
FF6h	TBLPTRL	FD6h	TMR0L	FB6h	ECCP1AS	F96h	TRISE
FF5h	TABLAT	FD5h	T0CON	FB5h	CVRCON	F95h	TRISD
FF4h	PRODH	FD4h	—(1)	FB4h	CMCON	F94h	TRISC
FF3h	PRODL	FD3h	OSCCON	FB3h	TMR3H	F93h	TRISB
FF2h	INTCON	FD2h	LVDCON	FB2h	TMR3L	F92h	TRISA
FF1h	INTCON2	FD1h	WDTCON	FB1h	T3CON	F91h	LATJ(2)
FF0h	INTCON3	FD0h	RCON	FB0h	PSPCON	F90h	LATH(2)
FEFh	INDF0(3)	FCFh	TMR1H	FAFh	SPBRG	F8Fh	LATG
FEEh	POSTINC0(3)	FCEh	TMR1L	FAEh	RCREG	F8Eh	LATF
FEDh	POSTDEC0(3)	FCDh	T1CON	FADh	TXREG	F8Dh	LATE
FECh	PREINC0(3)	FCCh	TMR2	FACh	TXSTA	F8Ch	LATD
FEBh	PLUSW0(3)	FCBh	PR2	FABh	RCSTA	F8Bh	LATC
FEAh	FSR0H	FCAh	T2CON	FAAh	EEADRH	F8Ah	LATB
FE9h	FSR0L	FC9h	SSPBUF	FA9h	EEADR	F89h	LATA
FE8h	WREG	FC8h	SSPADD	FA8h	EEDATA	F88h	PORTJ(2)
FE7h	INDF1(3)	FC7h	SSPSTAT	FA7h	EECON2	F87h	PORTH(2)
FE6h	POSTINC1(3)	FC6h	SSPCON1	FA6h	EECON1	F86h	PORTG
FE5h	POSTDEC1(3)	FC5h	SSPCON2	FA5h	IPR3	F85h	PORTF
FE4h	PREINC1(3)	FC4h	ADRESH	FA4h	PIR3	F84h	PORTE
FE3h	PLUSW1(3)	FC3h	ADRESL	FA3h	PIE3	F83h	PORTD
FE2h	FSR1H	FC2h	ADCON0	FA2h	IPR2	F82h	PORTC
FE1h	FSR1L	FC1h	ADCON1	FA1h	PIR2	F81h	PORTB
FE0h	BSR	FC0h	ADCON2	FA0h	PIE2	F80h	PORTA

Note 1: Unimplemented registers are read as '0'.
2: This register is not available on PIC18F6X8X devices.
3: This is not a physical register.

TABLE 4-2: SPECIAL FUNCTION REGISTER MAP (CONTINUED)

Address	Name	Address	Name	Address	Name	Address	Name
F7Fh	SPBRGH	F5Fh	CANCON_RO0	F3Fh	CANCON_RO2	F1Fh	RXM1EIDL
F7Eh	BAUDCON	F5Eh	CANSTAT_RO0	F3Eh	CANSTAT_RO2	F1Eh	RXM1EIDH
F7Dh	—(1)	F5Dh	RXB1D7	F3Dh	TXB1D7	F1Dh	RXM1SIDL
F7Ch	—(1)	F5Ch	RXB1D6	F3Ch	TXB1D6	F1Ch	RXM1SIDH
F7Bh	—(1)	F5Bh	RXB1D5	F3Bh	TXB1D5	F1Bh	RXM0EIDL
F7Ah	—(1)	F5Ah	RXB1D4	F3Ah	TXB1D4	F1Ah	RXM0EIDH
F79h	ECCP1DEL	F59h	RXB1D3	F39h	TXB1D3	F19h	RXM0SIDL
F78h	—(1)	F58h	RXB1D2	F38h	TXB1D2	F18h	RXM0SIDH
F77h	ECANCON	F57h	RXB1D1	F37h	TXB1D1	F17h	RXF5EIDL
F76h	TXERRCNT	F56h	RXB1D0	F36h	TXB1D0	F16h	RXF5EIDH
F75h	RXERRCNT	F55h	RXB1DLC	F35h	TXB1DLC	F15h	RXF5SIDL
F74h	COMSTAT	F54h	RXB1EIDL	F34h	TXB1EIDL	F14h	RXF5SIDH
F73h	CIOCON	F53h	RXB1EIDH	F33h	TXB1EIDH	F13h	RXF4EIDL
F72h	BRGCON3	F52h	RXB1SIDL	F32h	TXB1SIDL	F12h	RXF4EIDH
F71h	BRGCON2	F51h	RXB1SIDH	F31h	TXB1SIDH	F11h	RXF4SIDL
F70h	BRGCON1	F50h	RXB1CON	F30h	TXB1CON	F10h	RXF4SIDH
F6Fh	CANCON	F4Fh	CANCON_RO1	F2Fh	CANCON_RO3	F0Fh	RXF3EIDL
F6Eh	CANSTAT	F4Eh	CANSTAT_RO1	F2Eh	CANSTAT_RO3	F0Eh	RXF3EIDH
F6Dh	RXB0D7	F4Dh	TXB0D7	F2Dh	TXB2D7	F0Dh	RXF3SIDL
F6Ch	RXB0D6	F4Ch	TXB0D6	F2Ch	TXB2D6	F0Ch	RXF3SIDH
F6Bh	RXB0D5	F4Bh	TXB0D5	F2Bh	TXB2D5	F0Bh	RXF2EIDL
F6Ah	RXB0D4	F4Ah	TXB0D4	F2Ah	TXB2D4	F0Ah	RXF2EIDH
F69h	RXB0D3	F49h	TXB0D3	F29h	TXB2D3	F09h	RXF2SIDL
F68h	RXB0D2	F48h	TXB0D2	F28h	TXB2D2	F08h	RXF2SIDH
F67h	RXB0D1	F47h	TXB0D1	F27h	TXB2D1	F07h	RXF1EIDL
F66h	RXB0D0	F46h	TXB0D0	F26h	TXB2D0	F06h	RXF1EIDH
F65h	RXB0DLC	F45h	TXB0DLC	F25h	TXB2DLC	F05h	RXF1SIDL
F64h	RXB0EIDL	F44h	TXB0EIDL	F24h	TXB2EIDL	F04h	RXF1SIDH
F63h	RXB0EIDH	F43h	TXB0EIDH	F23h	TXB2EIDH	F03h	RXF0EIDL
F62h	RXB0SIDL	F42h	TXB0SIDL	F22h	TXB2SIDL	F02h	RXF0EIDH
F61h	RXB0SIDH	F41h	TXB0SIDH	F21h	TXB2SIDH	F01h	RXF0SIDL
F60h	RXB0CON	F40h	TXB0CON	F20h	TXB2CON	F00h	RXF0SIDH

Note 1: Unimplemented registers are read as ‘0’.
2: This register is not available on PIC18F6X8X devices.
3: This is not a physical register.

TABLE 4-2: SPECIAL FUNCTION REGISTER MAP (CONTINUED)

Address	Name	Address	Name	Address	Name	Address	Name
EFFh	—(1)	EDFh	—(1)	EBFh	—(1)	E9Fh	—(1)
EFEh	—(1)	EDEh	—(1)	EBEh	—(1)	E9Eh	—(1)
EFDh	—(1)	EDDh	—(1)	EBDh	—(1)	E9Dh	—(1)
EFCh	—(1)	EDCh	—(1)	EBCh	—(1)	E9Ch	—(1)
EFBh	—(1)	EDBh	—(1)	EBBh	—(1)	E9Bh	—(1)
EFAh	—(1)	EDAh	—(1)	EBAh	—(1)	E9Ah	—(1)
EF9h	—(1)	ED9h	—(1)	EB9h	—(1)	E99h	—(1)
EF8h	—(1)	ED8h	—(1)	EB8h	—(1)	E98h	—(1)
EF7h	—(1)	ED7h	—(1)	EB7h	—(1)	E97h	—(1)
EF6h	—(1)	ED6h	—(1)	EB6h	—(1)	E96h	—(1)
EF5h	—(1)	ED5h	—(1)	EB5h	—(1)	E95h	—(1)
EF4h	—(1)	ED4h	—(1)	EB4h	—(1)	E94h	—(1)
EF3h	—(1)	ED3h	—(1)	EB3h	—(1)	E93h	—(1)
EF2h	—(1)	ED2h	—(1)	EB2h	—(1)	E92h	—(1)
EF1h	—(1)	ED1h	—(1)	EB1h	—(1)	E91h	—(1)
EF0h	—(1)	ED0h	—(1)	EB0h	—(1)	E90h	—(1)
EEFh	—(1)	ECFh	—(1)	EAFh	—(1)	E8Fh	—(1)
EEEh	—(1)	ECEh	—(1)	EAEh	—(1)	E8Eh	—(1)
EEDh	—(1)	ECDh	—(1)	EADh	—(1)	E8Dh	—(1)
EECh	—(1)	ECCh	—(1)	EACh	—(1)	E8Ch	—(1)
EEBh	—(1)	ECBh	—(1)	EABh	—(1)	E8Bh	—(1)
EEAh	—(1)	ECAh	—(1)	EAAh	—(1)	E8Ah	—(1)
EE9h	—(1)	EC9h	—(1)	EA9h	—(1)	E89h	—(1)
EE8h	—(1)	EC8h	—(1)	EA8h	—(1)	E88h	—(1)
EE7h	—(1)	EC7h	—(1)	EA7h	—(1)	E87h	—(1)
EE6h	—(1)	EC6h	—(1)	EA6h	—(1)	E86h	—(1)
EE5h	—(1)	EC5h	—(1)	EA5h	—(1)	E85h	—(1)
EE4h	—(1)	EC4h	—(1)	EA4h	—(1)	E84h	—(1)
EE3h	—(1)	EC3h	—(1)	EA3h	—(1)	E83h	—(1)
EE2h	—(1)	EC2h	—(1)	EA2h	—(1)	E82h	—(1)
EE1h	—(1)	EC1h	—(1)	EA1h	—(1)	E81h	—(1)
EE0h	—(1)	EC0h	—(1)	EA0h	—(1)	E80h	—(1)

Note 1: Unimplemented registers are read as '0'.
2: This register is not available on PIC18F6X8X devices.
3: This is not a physical register.

TABLE 4-2: SPECIAL FUNCTION REGISTER MAP (CONTINUED)

Address	Name	Address	Name	Address	Name	Address	Name
E7Fh	CANCON_RO4	E5Fh	CANCON_RO6	E3Fh	CANCON_RO8	E1Fh	—(1)
E7Eh	CANSTAT_RO4	E5Eh	CANSTAT_RO6	E3Eh	CANSTAT_RO8	E1Eh	—(1)
E7Dh	B5D7	E5Dh	B3D7	E3Dh	B1D7	E1Dh	—(1)
E7Ch	B5D6	E5Ch	B3D6	E3Ch	B1D6	E1Ch	—(1)
E7Bh	B5D5	E5Bh	B3D5	E3Bh	B1D5	E1Bh	—(1)
E7Ah	B5D4	E5Ah	B3D4	E3Ah	B1D4	E1Ah	—(1)
E79h	B5D3	E59h	B3D3	E39h	B1D3	E19h	—(1)
E78h	B5D2	E58h	B3D2	E38h	B1D2	E18h	—(1)
E77h	B5D1	E57h	B3D1	E37h	B1D1	E17h	—(1)
E76h	B5D0	E56h	B3D0	E36h	B1D0	E16h	—(1)
E75h	B5DLC	E55h	B3DLC	E35h	B1DLC	E15h	—(1)
E74h	B5EIDL	E54h	B3EIDL	E34h	B1EIDL	E14h	—(1)
E73h	B5EIDH	E53h	B3EIDH	E33h	B1EIDH	E13h	—(1)
E72h	B5SIDL	E52h	B3SIDL	E32h	B1SIDL	E12h	—(1)
E71h	B5SIDH	E51h	B3SIDH	E31h	B1SIDH	E11h	—(1)
E70h	B5CON	E50h	B3CON	E30h	B1CON	E10h	—(1)
E6Fh	CANCON_RO5	E4Fh	CANCON_RO7	E2Fh	CANCON_RO9	E0Fh	—(1)
E6Eh	CANSTAT_RO5	E4Eh	CANSTAT_RO7	E2Eh	CANSTAT_RO9	E0Eh	—(1)
E6Dh	B4D7	E4Dh	B2D7	E2Dh	B0D7	E0Dh	—(1)
E6Ch	B4D6	E4Ch	B2D6	E2Ch	B0D6	E0Ch	—(1)
E6Bh	B4D5	E4Bh	B2D5	E2Bh	B0D5	E0Bh	—(1)
E6Ah	B4D4	E4Ah	B2D4	E2Ah	B0D4	E0Ah	—(1)
E69h	B4D3	E49h	B2D3	E29h	B0D3	E09h	—(1)
E68h	B4D2	E48h	B2D2	E28h	B0D2	E08h	—(1)
E67h	B4D1	E47h	B2D1	E27h	B0D1	E07h	—(1)
E66h	B4D0	E46h	B2D0	E26h	B0D0	E06h	—(1)
E65h	B4DLC	E45h	B2DLC	E25h	B0DLC	E05h	—(1)
E64h	B4EIDL	E44h	B2EIDL	E24h	B0EIDL	E04h	—(1)
E63h	B4EIDH	E43h	B2EIDH	E23h	B0EIDH	E03h	—(1)
E62h	B4SIDL	E42h	B2SIDL	E22h	B0SIDL	E02h	—(1)
E61h	B4SIDH	E41h	B2SIDH	E21h	B0SIDH	E01h	—(1)
E60h	B4CON	E40h	B2CON	E20h	B0CON	E00h	—(1)

Note 1: Unimplemented registers are read as '0'.
2: This register is not available on PIC18F6X8X devices.
3: This is not a physical register.

TABLE 4-2: SPECIAL FUNCTION REGISTER MAP (CONTINUED)

Address	Name	Address	Name	Address	Name	Address	Name
DFFh	—(1)	DDFh	—(1)	DBFh	—(1)	D9Fh	—(1)
DFEh	—(1)	DDEh	—(1)	DBEh	—(1)	D9Eh	—(1)
DFDh	—(1)	DDDh	—(1)	DBDh	—(1)	D9Dh	—(1)
DFCh	TXBIE	DDCh	—(1)	DBCh	—(1)	D9Ch	—(1)
DFBh	—(1)	DDBh	—(1)	DBBh	—(1)	D9Bh	—(1)
DFAh	BIE0	DDAh	—(1)	DBAh	—(1)	D9Ah	—(1)
DF9h	—(1)	DD9h	—(1)	DB9h	—(1)	D99h	—(1)
DF8h	BSEL0	DD8h	SDFLC	DB8h	—(1)	D98h	—(1)
DF7h	—(1)	DD7h	—(1)	DB7h	—(1)	D97h	—(1)
DF6h	—(1)	DD6h	—(1)	DB6h	—(1)	D96h	—(1)
DF5h	—(1)	DD5h	RXFCON1	DB5h	—(1)	D95h	—(1)
DF4h	—(1)	DD4h	RXFCON0	DB4h	—(1)	D94h	—(1)
DF3h	MSEL3	DD3h	—(1)	DB3h	—(1)	D93h	RXF15EIDL
DF2h	MSEL2	DD2h	—(1)	DB2h	—(1)	D92h	RXF15EIDH
DF1h	MSEL1	DD1h	—(1)	DB1h	—(1)	D91h	RXF15SIDL
DF0h	MSEL0	DD0h	—(1)	DB0h	—(1)	D90h	RXF15SIDH
DEFh	—(1)	DCFh	—(1)	DAFh	—(1)	D8Fh	—(1)
DEEh	—(1)	DCEh	—(1)	DAEh	—(1)	D8Eh	—(1)
DEDh	—(1)	DCDh	—(1)	DADh	—(1)	D8Dh	—(1)
DECh	—(1)	DCCh	—(1)	DACh	—(1)	D8Ch	—(1)
DEBh	—(1)	DCBh	—(1)	DABh	—(1)	D8Bh	RXF14EIDL
DEAh	—(1)	DCAh	—(1)	DAAh	—(1)	D8Ah	RXF14EIDH
DE9h	—(1)	DC9h	—(1)	DA9h	—(1)	D89h	RXF14SIDL
DE8h	—(1)	DC8h	—(1)	DA8h	—(1)	D88h	RXF14SIDH
DE7h	RXFBCON7	DC7h	—(1)	DA7h	—(1)	D87h	RXF13EIDL
DE6h	RXFBCON6	DC6h	—(1)	DA6h	—(1)	D86h	RXF13EIDH
DE5h	RXFBCON5	DC5h	—(1)	DA5h	—(1)	D85h	RXF13SIDL
DE4h	RXFBCON4	DC4h	—(1)	DA4h	—(1)	D84h	RXF13SIDH
DE3h	RXFBCON3	DC3h	—(1)	DA3h	—(1)	D83h	RXF12EIDL
DE2h	RXFBCON2	DC2h	—(1)	DA2h	—(1)	D82h	RXF12EIDH
DE1h	RXFBCON1	DC1h	—(1)	DA1h	—(1)	D81h	RXF12SIDL
DE0h	RXFBCON0	DC0h	—(1)	DA0h	—(1)	D80h	RXF12SIDH

Note 1: Unimplemented registers are read as '0'.
2: This register is not available on PIC18F6X8X devices.
3: This is not a physical register.

TABLE 4-2: SPECIAL FUNCTION REGISTER MAP (CONTINUED)

Address	Name
D7Fh	—(1)
D7Eh	—(1)
D7Dh	—(1)
D7Ch	—(1)
D7Bh	RXF11EIDL
D7Ah	RXF11EIDH
D79h	RXF11SIDL
D78h	RXF11SIDH
D77h	RXF10EIDL
D76h	RXF10EIDH
D75h	RXF10SIDL
D74h	RXF10SIDH
D73h	RXF9EIDL
D72h	RXF9EIDH
D71h	RXF9SIDL
D70h	RXF9SIDH
D6Fh	—(1)
D6Eh	—(1)
D6Dh	—(1)
D6Ch	—(1)
D6Bh	RXF8EIDL
D6Ah	RXF8EIDH
D69h	RXF8SIDL
D68h	RXF8SIDH
D67h	RXF7EIDL
D66h	RXF7EIDH
D65h	RXF7SIDL
D64h	RXF7SIDH
D63h	RXF6EIDL
D62h	RXF6EIDH
D61h	RXF6SIDL
D60h	RXF6SIDH

Note 1: Unimplemented registers are read as '0'.
2: This register is not available on PIC18F6X8X devices.
3: This is not a physical register.

TABLE 4-3: REGISTER FILE SUMMARY

File Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Details on page:
TOSU	—	—	—	Top-of-Stack Upper Byte (TOS<20:16>)					---0 0000	36, 54
TOSH	Top-of-Stack High Byte (TOS<15:8>)								0000 0000	36, 54
TOSL	Top-of-Stack Low Byte (TOS<7:0>)								0000 0000	36, 54
STKPTR	STKFUL	STKUNF	—	Return Stack Pointer					00-0 0000	36, 55
PCLATU	—	—	bit 21	Holding Register for PC<20:16>					--00 0000	36, 56
PCLATH	Holding Register for PC<15:8>								0000 0000	36, 56
PCL	PC Low Byte (PC<7:0>)								0000 0000	36, 56
TBLPTRU	—	—	bit 21(2)	Program Memory Table Pointer Upper Byte (TBLPTR<20:16>)					--00 0000	36, 86
TBLPTRH	Program Memory Table Pointer High Byte (TBLPTR<15:8>)								0000 0000	36, 86
TBLPTRL	Program Memory Table Pointer Low Byte (TBLPTR<7:0>)								0000 0000	36, 86
TABLAT	Program Memory Table Latch								0000 0000	36, 86
PRODH	Product Register High Byte								xxxx xxxx	36, 107
PRODL	Product Register Low Byte								xxxx xxxx	36, 107
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	36, 111
INTCON2	RBPU	INTEDG0	INTEDG1	INTEDG2	INTEDG3	TMR0IP	INT3IP	RBIP	1111 1111	36, 112
INTCON3	INT2IP	INT1IP	INT3IE	INT2IE	INT1IE	INT3IF	INT2IF	INT1IF	1100 0000	36, 113
INDF0	Uses contents of FSR0 to address data memory – value of FSR0 not changed (not a physical register)								n/a	79
POSTINC0	Uses contents of FSR0 to address data memory – value of FSR0 post-incremented (not a physical register)								n/a	79
POSTDEC0	Uses contents of FSR0 to address data memory – value of FSR0 post-decremented (not a physical register)								n/a	79
PREINC0	Uses contents of FSR0 to address data memory – value of FSR0 pre-incremented (not a physical register)								n/a	79
PLUSW0	Uses contents of FSR0 to address data memory – value of FSR0 pre-incremented (not a physical register) – value of FSR0 offset by value in WREG								n/a	79
FSR0H	—	—	—	—	Indirect Data Memory Address Pointer 0 High Byte				---- 0000	36, 79
FSR0L	Indirect Data Memory Address Pointer 0 Low Byte								xxxx xxxx	36, 79
WREG	Working Register								xxxx xxxx	36
INDF1	Uses contents of FSR1 to address data memory – value of FSR1 not changed (not a physical register)								n/a	79
POSTINC1	Uses contents of FSR1 to address data memory – value of FSR1 post-incremented (not a physical register)								n/a	79
POSTDEC1	Uses contents of FSR1 to address data memory – value of FSR1 post-decremented (not a physical register)								n/a	79
PREINC1	Uses contents of FSR1 to address data memory – value of FSR1 pre-incremented (not a physical register)								n/a	79
PLUSW1	Uses contents of FSR1 to address data memory – value of FSR1 pre-incremented (not a physical register) – value of FSR1 offset by value in WREG								n/a	79
FSR1H	—	—	—	—	Indirect Data Memory Address Pointer 1 High Byte				---- 0000	37, 79
FSR1L	Indirect Data Memory Address Pointer 1 Low Byte								xxxx xxxx	37, 79
BSR	—	—	—	—	Bank Select Register				---- 0000	37, 78
INDF2	Uses contents of FSR2 to address data memory – value of FSR2 not changed (not a physical register)								n/a	79
POSTINC2	Uses contents of FSR2 to address data memory – value of FSR2 post-incremented (not a physical register)								n/a	79
POSTDEC2	Uses contents of FSR2 to address data memory – value of FSR2 post-decremented (not a physical register)								n/a	79
PREINC2	Uses contents of FSR2 to address data memory – value of FSR2 pre-incremented (not a physical register)								n/a	79
PLUSW2	Uses contents of FSR2 to address data memory – value of FSR2 pre-incremented (not a physical register) – value of FSR2 offset by value in WREG								n/a	79
FSR2H	—	—	—	—	Indirect Data Memory Address Pointer 2 High Byte				---- 0000	37, 79
FSR2L	Indirect Data Memory Address Pointer 2 Low Byte								xxxx xxxx	37, 79

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

Note 1: RA6 and associated bits are configured as port pins in RCIO and ECIO Oscillator mode only and read '0' in all other oscillator modes.

2: Bit 21 of the TBLPTRU allows access to the device configuration bits.

3: These registers are unused on PIC18F6X80 devices; always maintain these clear.

4: These bits have multiple functions depending on the CAN module mode selection.

5: Meaning of this register depends on whether this buffer is configured as transmit or receive.

6: RG5 is available as an input when MCLR is disabled.

7: This register reads all '0's until the ECAN module is set up in Mode 1 or Mode 2.

TABLE 4-3: REGISTER FILE SUMMARY (CONTINUED)

File Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Details on page:
STATUS	—	—	—	N	OV	Z	DC	C	---x xxxx	37, 81
TMR0H	Timer0 Register High Byte								0000 0000	37, 157
TMR0L	Timer0 Register Low Byte								xxxx xxxx	37, 157
T0CON	TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0	1111 1111	37, 155
OSCCON	—	—	—	—	LOCK	PLLEN	SCS1	SCS	---- 0000	27, 37
LVDCON	—	—	IRVST	LVDEN	LVDL3	LVDL2	LVDL1	LVDL0	--00 0101	37, 271
WDTCON	—	—	—	—	—	—	—	SWDTE	---- ---0	37, 355
RCON	IPEN	—	—	$\overline{\text{RI}}$	$\overline{\text{TO}}$	$\overline{\text{PD}}$	$\overline{\text{POR}}$	$\overline{\text{BOR}}$	0--1 11qq	37, 82, 123
TMR1H	Timer1 Register High Byte								xxxx xxxx	37, 159
TMR1L	Timer1 Register Low Byte								xxxx xxxx	37, 159
T1CON	RD16	—	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON	0-00 0000	37, 159
TMR2	Timer2 Register								0000 0000	37, 162
PR2	Timer2 Period Register								1111 1111	37, 163
T2CON	—	T2OUTPS3	T2OUTPS2	T2OUTPS1	T2OUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-000 0000	37, 162
SSPBUF	SSP Receive Buffer/Transmit Register								xxxx xxxx	37, 189
SSPADD	SSP Address Register in I^2C Slave mode. SSP Baud Rate Reload Register in I^2C Master mode.								0000 0000	37, 198
SSPSTAT	SMP	CKE	D/$\overline{\text{A}}$	P	S	R/$\overline{\text{W}}$	UA	BF	0000 0000	37, 199
SSPCON1	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	0000 0000	37, 191
SSPCON2	GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN	0000 0000	37, 201
ADRESH	A/D Result Register High Byte								xxxx xxxx	38, 257
ADRESL	A/D Result Register Low Byte								xxxx xxxx	38, 257
ADCON0	—	—	CHS3	CHS2	CHS1	CHS0	GO/$\overline{\text{DONE}}$	ADON	--00 0000	38, 249
ADCON1	—	—	VCFG1	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0	--00 0000	38, 257
ADCON2	ADFM	—	ACQT2	ACQT1	ACQT0	ADCS2	ADCS1	ADCS0	0-00 0000	38, 251
CCPR1H	Enhanced Capture/Compare/PWM Register 1 High Byte								xxxx xxxx	38, 173
CCPR1L	Enhanced Capture/Compare/PWM Register 1 Low Byte								xxxx xxxx	38, 172
CCP1CON	P1M1	P1M0	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0	0000 0000	38, 172
CCPR2H	Capture/Compare/PWM Register 2 High Byte								xxxx xxxx	38, 172
CCPR2L	Capture/Compare/PWM Register 2 Low Byte								xxxx xxxx	38, 172
CCP2CON	—	—	DC2B1	DC2B0	CCP2M3	CCP2M2	CCP2M1	CCP2M0	--00 0000	38, 172
ECCP1AS	ECCPASE	ECCPAS2	ECCPAS1	ECCPAS0	PSSAC1	PSSAC0	PSSBD1	PSSBD0	0000 0000	38, 172
CVRCON	CVREN	CVROE	CVRR	CVRSS	CVR3	CVR2	CVR1	CVR0	0000 0000	38, 265
CMCON	C2OUT	C1OUT	C2INV	C1INV	CIS	CM2	CM1	CM0	0000 0000	38, 259
TMR3H	Timer3 Register High Byte								xxxx xxxx	38, 164
TMR3L	Timer3 Register Low Byte								xxxx xxxx	38, 164
T3CON	RD16	T3CCP2	T3CKPS1	T3CKPS0	T3CCP1	$\overline{\text{T3SYNC}}$	TMR3CS	TMR3ON	0000 0000	38, 164
PSPCON	IBF	OBF	IBOV	PSPMODE	—	—	—	—	0000 ----	38, 153

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

Note 1: RA6 and associated bits are configured as port pins in RCIO and ECIO Oscillator mode only and read '0' in all other oscillator modes.

2: Bit 21 of the TBLPTRU allows access to the device configuration bits.

3: These registers are unused on PIC18F6X80 devices; always maintain these clear.

4: These bits have multiple functions depending on the CAN module mode selection.

5: Meaning of this register depends on whether this buffer is configured as transmit or receive.

6: RG5 is available as an input when $\overline{\text{MCLR}}$ is disabled.

7: This register reads all '0's until the ECAN module is set up in Mode 1 or Mode 2.

TABLE 4-3: REGISTER FILE SUMMARY (CONTINUED)

File Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Details on page:
SPBRG	USART Baud Rate Generator								0000 0000	38, 239
RCREG	USART Receive Register								0000 0000	38, 241
TXREG	USART Transmit Register								0000 0000	38, 239
TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	0000 0010	38, 230
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 000x	38, 231
EEADRH	—	—	—	—	—	—	EE Adr Register High		---- --00	38, 105
EEADR	Data EEPROM Address Register								0000 0000	38, 105
EEDATA	Data EEPROM Data Register								0000 0000	38, 105
EECON2	Data EEPROM Control Register 2 (not a physical register)								---- ----	38, 105
EECON1	EEPGD	CFGS	—	FREE	WRERR	WREN	WR	RD	00-0 x000	38, 102
IPR3	IRXIP	WAKIP	ERRIP	TXB2IP/ TXBnIP	TXB1IP	TXB0IP	RXB1IP/ RXBnIP	RXB0IP/ FIFOWMIP	1111 1111	39, 122
PIR3	IRXIF	WAKIF	ERRIF	TXB2IF/ TXBnIF	TXB1IF	TXB0IF	RXB1IF/ RXBnIF	RXB0IF/ FIFOWMIF	0000 0000	39, 116
PIE3	IRXIE	WAKIE	ERRIE	TXB2IE/ TXBnIE	TXB1IE	TXB0IE	RXB1IE/ RXBnIE	RXB0IE/ FIFOWMIE	0000 0000	39, 119
IPR2	—	CMIP	—	EEIP	BCLIP	LVDIP	TMR3IP	CCP2IP	-1-1 1111	39, 121
PIR2	—	CMIF	—	EEIF	BCLIF	LVDIF	TMR3IF	CCP2IF	-0-0 0000	39, 115
PIE2	—	CMIE	—	EEIE	BCLIE	LVDIE	TMR3IE	CCP2IE	-0-0 0000	39, 118
IPR1	PSPIP	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	0111 1111	39, 120
PIR1	PSPIF	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	39, 114
PIE1	PSPIE	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	39, 117
MEMCON(3)	EBDIS	—	WAIT1	WAIT0	—	—	WM1	WM0	0-00 --00	39, 94
TRISJ(3)	Data Direction Control Register for PORTJ								1111 1111	39, 151
TRISH(3)	Data Direction Control Register for PORTH								1111 1111	39, 148
TRISG	—	—	—	Data Direction Control Register for PORTG					---1 1111	39, 145
TRISF	Data Direction Control Register for PORTF								1111 1111	39, 141
TRISE	Data Direction Control Register for PORTE								1111 1111	39, 138
TRISD	Data Direction Control Register for PORTD								1111 1111	39, 135
TRISC	Data Direction Control Register for PORTC								1111 1111	39, 131
TRISB	Data Direction Control Register for PORTB								1111 1111	39, 128
TRISA	—	TRISA6(1)	Data Direction Control Register for PORTA						-111 1111	39, 125
LATJ(3)	Read PORTJ Data Latch, Write PORTJ Data Latch								xxxx xxxx	39, 151
LATH(3)	Read PORTH Data Latch, Write PORTH Data Latch								xxxx xxxx	39, 148
LATG	—	—	—	Read PORTG Data Latch, Write PORTG Data Latch					---x xxxx	39, 145
LATF	Read PORTF Data Latch, Write PORTF Data Latch								xxxx xxxx	39, 141
LATE	Read PORTE Data Latch, Write PORTE Data Latch								xxxx xxxx	39, 138
LATD	Read PORTD Data Latch, Write PORTD Data Latch								xxxx xxxx	39, 133
LATC	Read PORTC Data Latch, Write PORTC Data Latch								xxxx xxxx	39, 131
LATB	Read PORTB Data Latch, Write PORTB Data Latch								xxxx xxxx	39, 128
LATA	—	LATA6(1)	Read PORTA Data Latch, Write PORTA Data Latch(1)						-xxx xxxx	39, 125

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

Note 1: RA6 and associated bits are configured as port pins in RCIO and ECIO Oscillator mode only and read '0' in all other oscillator modes.

2: Bit 21 of the TBLPTRU allows access to the device configuration bits.

3: These registers are unused on PIC18F6X80 devices; always maintain these clear.

4: These bits have multiple functions depending on the CAN module mode selection.

5: Meaning of this register depends on whether this buffer is configured as transmit or receive.

6: RG5 is available as an input when $\overline{MCLR}$ is disabled.

7: This register reads all '0's until the ECAN module is set up in Mode 1 or Mode 2.

TABLE 4-3: REGISTER FILE SUMMARY (CONTINUED)

File Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Details on page:
PORTJ[3]	Read PORTJ pins, Write PORTJ Data Latch								xxxx xxxx	40, 151
PORTH[3]	Read PORTH pins, Write PORTH Data Latch								xxxx xxxx	40, 148
PORTG	—	—	RG5[6]	Read PORTG pins, Write PORTG Data Latch					--0x xxxx	40, 145
PORTF	Read PORTF pins, Write PORTF Data Latch								xxxx xxxx	40, 141
PORTE	Read PORTE pins, Write PORTE Data Latch								xxxx xxxx	40, 136
PORTD	Read PORTD pins, Write PORTD Data Latch								xxxx xxxx	40, 133
PORTC	Read PORTC pins, Write PORTC Data Latch								xxxx xxxx	40, 131
PORTB	Read PORTB pins, Write PORTB Data Latch								xxxx xxxx	40, 128
PORTA	—	RA6[1]	Read PORTA pins, Write PORTA Data Latch[1]						-x0x 0000	40, 125
SPBRGH	Enhanced USART Baud Rate Generator High Byte								0000 0000	40, 233
BAUDCON	—	RCIDL	—	SCKP	BRG16	—	WUE	ABDEN	-1-0 0-00	40, 233
ECCP1DEL	PRSEN	PDC6	PDC5	PDC4	PDC3	PDC2	PDC1	PDC0	0000 0000	40, 187
TXERRCNT	TEC7	TEC6	TEC5	TEC4	TEC3	TEC2	TEC1	TEC0	0000 0000	40, 288
RXERRCNT	REC7	REC6	REC5	REC4	REC3	REC2	REC1	REC0	0000 0000	40, 296
COMSTAT Mode 0	RXB0OVFL	RXB1OVFL	TXBO	TXBP	RXBP	TXWARN	RXWARN	EWARN	0000 0000	40, 284
COMSTAT Mode 1	—	RXBnOVFL	TXBO	TXBP	RXBP	TXWARN	RXWARN	EWARN	-000 0000	40, 284
COMSTAT Mode 2	$\overline{\text{FIFOEMPTY}}$	RXBnOVFL	TXBO	TXBP	RXBP	TXWARN	RXWARN	EWARN	0000 0000	40, 284
CIOCON	TX2SRC	TX2EN	ENDRHI	CANCAP	—	—	—	—	0000 ----	40, 318
BRGCON3	WAKDIS	WAKFIL	—	—	—	SEG2PH2	SEG2PH1	SEG2PH0	00-- -000	40, 317
BRGCON2	SEG2PHT	SAM	SEG1PH2	SEG1PH1	SEG1PH0	PRSEG2	PRSEG1	PRSEG0	0000 0000	40, 317
BRGCON1	SJW1	SJW0	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0	0000 0000	40, 317
CANCON Mode 0	REQOP2	REQOP1	REQOP0	ABAT	WIN2	WIN1	WIN0	—	1000 000-	40, 239
CANCON Mode 1	REQOP2	REQOP1	REQOP0	ABAT	—	—	—	—	1000 ----	40, 239
CANCON Mode 2	REQOP2	REQOP1	REQOP0	ABAT	FP3	FP2	FP1	FP0	1000 0000	40, 239
CANSTAT Mode 0	OPMODE2	OPMODE1	OPMODE0	—	ICODE2	ICODE1	ICODE0	—	000- 0000	40, 239
CANSTAT Modes 0, 1	OPMODE2	OPMODE1	OPMODE0	EICODE4	EICODE3	EICODE2	EICODE1	EICODE0	0000 0000	40, 239
ECANCON	MDSEL1	MDSEL0	FIFOWM	EWIN4	EWIN3	EWIN2	EWIN1	EWIN0	0001 0000	40, 323
RXB0D7	RXB0D77	RXB0D76	RXB0D75	RXB0D74	RXB0D73	RXB0D72	RXB0D71	RXB0D70	xxxx xxxx	40, 230
RXB0D6	RXB0D67	RXB0D66	RXB0D65	RXB0D64	RXB0D63	RXB0D62	RXB0D61	RXB0D60	xxxx xxxx	40, 230
RXB0D5	RXB0D57	RXB0D56	RXB0D55	RXB0D54	RXB0D53	RXB0D52	RXB0D51	RXB0D50	xxxx xxxx	40, 230
RXB0D4	RXB0D47	RXB0D46	RXB0D45	RXB0D44	RXB0D43	RXB0D42	RXB0D41	RXB0D40	xxxx xxxx	40, 230
RXB0D3	RXB0D37	RXB0D36	RXB0D35	RXB0D34	RXB0D33	RXB0D32	RXB0D31	RXB0D30	xxxx xxxx	40, 230
RXB0D2	RXB0D27	RXB0D26	RXB0D25	RXB0D24	RXB0D23	RXB0D22	RXB0D21	RXB0D20	xxxx xxxx	40, 230
RXB0D1	RXB0D17	RXB0D16	RXB0D15	RXB0D14	RXB0D13	RXB0D12	RXB0D11	RXB0D10	xxxx xxxx	40, 230
RXB0D0	RXB0D07	RXB0D06	RXB0D05	RXB0D04	RXB0D03	RXB0D02	RXB0D01	RXB0D00	xxxx xxxx	40, 230

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

Note 1: RA6 and associated bits are configured as port pins in RCIO and ECIO Oscillator mode only and read '0' in all other oscillator modes.

2: Bit 21 of the TBLPTRU allows access to the device configuration bits.

3: These registers are unused on PIC18F6X80 devices; always maintain these clear.

4: These bits have multiple functions depending on the CAN module mode selection.

5: Meaning of this register depends on whether this buffer is configured as transmit or receive.

6: RG5 is available as an input when $\overline{\text{MCLR}}$ is disabled.

7: This register reads all '0's until the ECAN module is set up in Mode 1 or Mode 2.

TABLE 4-3: REGISTER FILE SUMMARY (CONTINUED)

File Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Details on page:
RXB0DLC	—	RXRTR	RB1	RB0	DLC3	DLC2	DLC1	DLC0	-xxx xxxx	40, 230
RXB0EIDL	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	xxxx xxxx	41, 230
RXB0EIDH	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	xxxx xxxx	41, 230
RXB0SIDL	SID2	SID1	SID0	SRR	EXID	—	EID17	EID16	xxxx x-xx	41, 230
RXB0SIDH	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	xxxx xxxx	41, 230
RXB0CON Mode 0	RXFUL	RXM1	RXM0(4)	—(4)	RXRTRR0(4)	RXB0DBEN(4)	JTOFF(4)	FILHIT0(4)	000- 0000	41, 230
RXB0CON Mode 1, 2	RXFUL	RXM1	RTRR0(4)	FILHIT4(4)	FILHIT3(4)	FILHIT2(4)	FILHIT1(4)	FILHIT0(4)	0000 0000	41, 230
RXB1D7	RXB1D77	RXB1D76	RXB1D75	RXB1D74	RXB1D73	RXB1D72	RXB1D71	RXB1D70	xxxx xxxx	41, 230
RXB1D6	RXB1D67	RXB1D66	RXB1D65	RXB1D64	RXB1D63	RXB1D62	RXB1D61	RXB1D60	xxxx xxxx	41, 230
RXB1D5	RXB1D57	RXB1D56	RXB1D55	RXB1D54	RXB1D53	RXB1D52	RXB1D51	RXB1D50	xxxx xxxx	41, 230
RXB1D4	RXB1D47	RXB1D46	RXB1D45	RXB1D44	RXB1D43	RXB1D42	RXB1D41	RXB1D40	xxxx xxxx	41, 230
RXB1D3	RXB1D37	RXB1D36	RXB1D35	RXB1D34	RXB1D33	RXB1D32	RXB1D31	RXB1D30	xxxx xxxx	41, 230
RXB1D2	RXB1D27	RXB1D26	RXB1D25	RXB1D24	RXB1D23	RXB1D22	RXB1D21	RXB1D20	xxxx xxxx	41, 230
RXB1D1	RXB1D17	RXB1D16	RXB1D15	RXB1D14	RXB1D13	RXB1D12	RXB1D11	RXB1D10	xxxx xxxx	41, 230
RXB1D0	RXB1D07	RXB1D06	RXB1D05	RXB1D04	RXB1D03	RXB1D02	RXB1D01	RXB1D00	xxxx xxxx	41, 230
RXB1DLC	—	RXRTR	RB1	RB0	DLC3	DLC2	DLC1	DLC0	-xxx xxxx	41, 230
RXB1EIDL	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	xxxx xxxx	41, 230
RXB1EIDH	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	xxxx xxxx	41, 230
RXB1SIDL	SID2	SID1	SID0	SRR	EXID	—	EID17	EID16	xxxx x-xx	41, 230
RXB1SIDH	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	xxxx xxxx	41, 230
RXB1CON Mode 0	RXFUL	RXM1	RXM0(4)	—(4)	RXRTRR0(4)	FILHIT2(4)	FILHIT1(4)	FILHIT0(4)	000- 0000	41, 230
RXB1CON Mode 1, 2	RXFUL	RXM1	RTRRO(4)	FILHIT4(4)	FILHIT3(4)	FILHIT2(4)	FILHIT1(4)	FILHIT0(4)	0000 0000	41, 230
TXB0D7	TXB0D77	TXB0D76	TXB0D75	TXB0D74	TXB0D73	TXB0D72	TXB0D71	TXB0D70	xxxx xxxx	41, 230
TXB0D6	TXB0D67	TXB0D66	TXB0D65	TXB0D64	TXB0D63	TXB0D62	TXB0D61	TXB0D60	xxxx xxxx	41, 230
TXB0D5	TXB0D57	TXB0D56	TXB0D55	TXB0D54	TXB0D53	TXB0D52	TXB0D51	TXB0D50	xxxx xxxx	41, 230
TXB0D4	TXB0D47	TXB0D46	TXB0D45	TXB0D44	TXB0D43	TXB0D42	TXB0D41	TXB0D40	xxxx xxxx	41, 230
TXB0D3	TXB0D37	TXB0D36	TXB0D35	TXB0D34	TXB0D33	TXB0D32	TXB0D31	TXB0D30	xxxx xxxx	41, 230
TXB0D2	TXB0D27	TXB0D26	TXB0D25	TXB0D24	TXB0D23	TXB0D22	TXB0D21	TXB0D20	xxxx xxxx	41, 230
TXB0D1	TXB0D17	TXB0D16	TXB0D15	TXB0D14	TXB0D13	TXB0D12	TXB0D11	TXB0D10	xxxx xxxx	41, 230
TXB0D0	TXB0D07	TXB0D06	TXB0D05	TXB0D04	TXB0D03	TXB0D02	TXB0D01	TXB0D00	xxxx xxxx	41, 230
TXB0DLC	—	TXRTR	—	—	DLC3	DLC2	DLC1	DLC0	-x-- xxxx	41, 230
TXB0EIDL	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	xxxx xxxx	41, 230
TXB0EIDH	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	xxxx xxxx	41, 230
TXB0SIDL	SID2	SID1	SID0	—	EXIDE	—	EID17	EID16	xx-x x-xx	41, 230
TXB0SIDH	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	xxxx xxxx	42, 230
TXB0CON Mode 0	—	TXABT	TXLARB	TXERR	TXREQ	—	TXPRI1	TXPRI0	-000 0-00	42, 230
TXB0CON Mode 1, 2	TXBIF	TXABT	TXLARB	TXERR	TXREQ	—	TXPRI1	TXPRI0	0000 0-00	42, 230

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

Note 1: RA6 and associated bits are configured as port pins in RCIO and ECIO Oscillator mode only and read '0' in all other oscillator modes.

2: Bit 21 of the TBLPTRU allows access to the device configuration bits.

3: These registers are unused on PIC18F6X80 devices; always maintain these clear.

4: These bits have multiple functions depending on the CAN module mode selection.

5: Meaning of this register depends on whether this buffer is configured as transmit or receive.

6: RG5 is available as an input when $\overline{\text{MCLR}}$ is disabled.

7: This register reads all '0's until the ECAN module is set up in Mode 1 or Mode 2.

TABLE 4-3: REGISTER FILE SUMMARY (CONTINUED)

File Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Details on page:
TXB1D7	TXB1D77	TXB1D76	TXB1D75	TXB1D74	TXB1D73	TXB1D72	TXB1D71	TXB1D70	xxxx xxxx	42, 230
TXB1D6	TXB1D67	TXB1D66	TXB1D65	TXB1D64	TXB1D63	TXB1D62	TXB1D61	TXB1D60	xxxx xxxx	42, 230
TXB1D5	TXB1D57	TXB1D56	TXB1D55	TXB1D54	TXB1D53	TXB1D52	TXB1D51	TXB1D50	xxxx xxxx	42, 230
TXB1D4	TXB1D47	TXB1D46	TXB1D45	TXB1D44	TXB1D43	TXB1D42	TXB1D41	TXB1D40	xxxx xxxx	42, 230
TXB1D3	TXB1D37	TXB1D36	TXB1D35	TXB1D34	TXB1D33	TXB1D32	TXB1D31	TXB1D30	xxxx xxxx	42, 230
TXB1D2	TXB1D27	TXB1D26	TXB1D25	TXB1D24	TXB1D23	TXB1D22	TXB1D21	TXB1D20	xxxx xxxx	42, 230
TXB1D1	TXB1D17	TXB1D16	TXB1D15	TXB1D14	TXB1D13	TXB1D12	TXB1D11	TXB1D10	xxxx xxxx	42, 230
TXB1D0	TXB1D07	TXB1D06	TXB1D05	TXB1D04	TXB1D03	TXB1D02	TXB1D01	TXB1D00	xxxx xxxx	42, 230
TXB1DLC	—	TXRTR	—	—	DLC3	DLC2	DLC1	DLC0	-x-- xxxx	42, 230
TXB1EIDL	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	xxxx xxxx	42, 230
TXB1EIDH	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	xxxx xxxx	42, 230
TXB1SIDL	SID2	SID1	SID0	—	EXIDE	—	EID17	EID16	xx-x x-xx	42, 230
TXB1SIDH	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	xxxx xxxx	42, 230
TXB1CON Mode 0	—	TXABT	TXLARB	TXERR	TXREQ	—	TXPRI1	TXPRI0	-000 0-00	42, 230
TXB1CON Mode 1, 2	TXBIF	TXABT	TXLARB	TXERR	TXREQ	—	TXPRI1	TXPRI0	0000 0-00	42, 230
TXB2D7	TXB2D77	TXB2D76	TXB2D75	TXB2D74	TXB2D73	TXB2D72	TXB2D71	TXB2D70	xxxx xxxx	42, 230
TXB2D6	TXB2D67	TXB2D66	TXB2D65	TXB2D64	TXB2D63	TXB2D62	TXB2D61	TXB2D60	xxxx xxxx	42, 230
TXB2D5	TXB2D57	TXB2D56	TXB2D55	TXB2D54	TXB2D53	TXB2D52	TXB2D51	TXB2D50	xxxx xxxx	42, 230
TXB2D4	TXB2D47	TXB2D46	TXB2D45	TXB2D44	TXB2D43	TXB2D42	TXB2D41	TXB2D40	xxxx xxxx	42, 230
TXB2D3	TXB2D37	TXB2D36	TXB2D35	TXB2D34	TXB2D33	TXB2D32	TXB2D31	TXB2D30	xxxx xxxx	42, 230
TXB2D2	TXB2D27	TXB2D26	TXB2D25	TXB2D24	TXB2D23	TXB2D22	TXB2D21	TXB2D20	xxxx xxxx	42, 230
TXB2D1	TXB2D17	TXB2D16	TXB2D15	TXB2D14	TXB2D13	TXB2D12	TXB2D11	TXB2D10	xxxx xxxx	42, 230
TXB2D0	TXB2D07	TXB2D06	TXB2D05	TXB2D04	TXB2D03	TXB2D02	TXB2D01	TXB2D00	xxxx xxxx	42, 230
TXB2DLC	—	TXRTR	—	—	DLC3	DLC2	DLC1	DLC0	-x-- xxxx	42, 230
TXB2EIDL	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	xxxx xxxx	42, 230
TXB2EIDH	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	xxxx xxxx	42, 230
TXB2SIDL	SID2	SID1	SID0	—	EXIDE	—	EID17	EID16	xxx- x-xx	42, 230
TXB2SIDH	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	xxxx xxxx	42, 230
TXB2CON Mode 0	—	TXABT	TXLARB	TXERR	TXREQ	—	TXPRI1	TXPRI0	-000 0-00	42, 230
TXB2CON Mode 1, 2	TXBIF	TXABT	TXLARB	TXERR	TXREQ	—	TXPRI1	TXPRI0	0000 0-00	42, 230
RXM1EIDL	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	xxxx xxxx	42, 230
RXM1EIDH	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	xxxx xxxx	43, 230
RXM1SIDL	SID2	SID1	SID0	—	EXIDEN	—	EID17	EID16	xx-x 0-xx	43, 230
RXM1SIDH	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	xxxx xxxx	43, 230
RXM0EIDL	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	xxxx xxxx	43, 230
RXM0EIDH	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	xxxx xxxx	43, 230
RXM0SIDL	SID2	SID1	SID0	—	EXIDM	—	EID17	EID16	xx-x 0-xx	43, 230
RXM0SIDH	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	xxxx xxxx	43, 230
RXF15EIDL(7)	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	xxxx xxxx	47, 230

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

Note 1: RA6 and associated bits are configured as port pins in RCIO and ECIO Oscillator mode only and read '0' in all other oscillator modes.

2: Bit 21 of the TBLPTRU allows access to the device configuration bits.

3: These registers are unused on PIC18F6X80 devices; always maintain these clear.

4: These bits have multiple functions depending on the CAN module mode selection.

5: Meaning of this register depends on whether this buffer is configured as transmit or receive.

6: RG5 is available as an input when $\overline{MCLR}$ is disabled.

7: This register reads all '0's until the ECAN module is set up in Mode 1 or Mode 2.

TABLE 4-3: REGISTER FILE SUMMARY (CONTINUED)

File Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Details on page:
RXF15EIDH(7)	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	xxxx xxxx	47, 230
RXF15SIDL(7)	SID2	SID1	SID0	—	EXIDEN	—	EID17	EID16	xx-x x-xx	47, 230
RXF15SIDH(7)	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	xxxx xxxx	47, 230
RXF14EIDL(7)	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	xxxx xxxx	47, 230
RXF14EIDH(7)	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	xxxx xxxx	47, 230
RXF14SIDL(7)	SID2	SID1	SID0	—	EXIDEN	—	EID17	EID16	xx-x x-xx	47, 230
RXF14SIDH(7)	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	xxxx xxxx	47, 230
RXF13EIDL(7)	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	xxxx xxxx	47, 230
RXF13EIDH(7)	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	xxxx xxxx	47, 230
RXF13SIDL(7)	SID2	SID1	SID0	—	EXIDEN	—	EID17	EID16	xx-x x-xx	47, 230
RXF13SIDH(7)	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	xxxx xxxx	47, 230
RXF12EIDL(7)	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	xxxx xxxx	47, 230
RXF12EIDH(7)	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	xxxx xxxx	47, 230
RXF12SIDL(7)	SID2	SID1	SID0	—	EXIDEN	—	EID17	EID16	xx-x x-xx	47, 230
RXF12SIDH(7)	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	xxxx xxxx	47, 230
RXF11EIDL(7)	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	xxxx xxxx	47, 230
RXF11EIDH(7)	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	xxxx xxxx	47, 230
RXF11SIDL(7)	SID2	SID1	SID0	—	EXIDEN	—	EID17	EID16	xx-x x-xx	47, 230
RXF11SIDH(7)	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	xxxx xxxx	47, 230
RXF10EIDL(7)	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	xxxx xxxx	47, 230
RXF10EIDH(7)	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	xxxx xxxx	47, 230
RXF10SIDL(7)	SID2	SID1	SID0	—	EXIDEN	—	EID17	EID16	xx-x x-xx	48, 230
RXF10SIDH(7)	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	xxxx xxxx	48, 230
RXF9EIDL(7)	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	xxxx xxxx	47, 230
RXF9EIDH(7)	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	xxxx xxxx	48, 230
RXF9SIDL(7)	SID2	SID1	SID0	—	EXIDEN	—	EID17	EID16	xx-x x-xx	48, 230
RXF9SIDH(7)	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	xxxx xxxx	48, 230
RXF8EIDL(7)	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	xxxx xxxx	48, 230
RXF8EIDH(7)	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	xxxx xxxx	48, 230
RXF8SIDL(7)	SID2	SID1	SID0	—	EXIDEN	—	EID17	EID16	xx-x x-xx	48, 230
RXF8SIDH(7)	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	xxxx xxxx	48, 230
RXF7EIDL(7)	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	xxxx xxxx	48, 230
RXF7EIDH(7)	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	xxxx xxxx	48, 230
RXF7SIDL(7)	SID2	SID1	SID0	—	EXIDEN	—	EID17	EID16	xx-x x-xx	48, 230
RXF7SIDH(7)	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	xxxx xxxx	48, 230
RXF6EIDL(7)	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	xxxx xxxx	48, 230
RXF6EIDH(7)	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	xxxx xxxx	48, 230
RXF6SIDL(7)	SID2	SID1	SID0	—	EXIDEN	—	EID17	EID16	xx-x x-xx	48, 230
RXF6SIDH(7)	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	xxxx xxxx	48, 230
RXF5EIDL	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	xxxx xxxx	43, 230
RXF5EIDH	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	xxxx xxxx	43, 230

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

Note 1: RA6 and associated bits are configured as port pins in RCIO and ECIO Oscillator mode only and read '0' in all other oscillator modes.

2: Bit 21 of the TBLPTRU allows access to the device configuration bits.

3: These registers are unused on PIC18F6X80 devices; always maintain these clear.

4: These bits have multiple functions depending on the CAN module mode selection.

5: Meaning of this register depends on whether this buffer is configured as transmit or receive.

6: RG5 is available as an input when $\overline{\text{MCLR}}$ is disabled.

7: This register reads all '0's until the ECAN module is set up in Mode 1 or Mode 2.

TABLE 4-3: REGISTER FILE SUMMARY (CONTINUED)

File Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Details on page:
RXF5SIDL	SID2	SID1	SID0	—	EXIDEN	—	EID17	EID16	xx-x x-xx	43, 230
RXF5SIDH	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	xxxx xxxx	43, 230
RXF4EIDL	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	xxxx xxxx	43, 230
RXF4EIDH	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	xxxx xxxx	43, 230
RXF4SIDL	SID2	SID1	SID0	—	EXIDEN	—	EID17	EID16	xx-x x-xx	43, 230
RXF4SIDH	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	xxxx xxxx	43, 230
RXF3EIDL	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	xxxx xxxx	43, 230
RXF3EIDH	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	xxxx xxxx	43, 230
RXF3SIDL	SID2	SID1	SID0	—	EXIDEN	—	EID17	EID16	xx-x x-xx	43, 230
RXF3SIDH	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	xxxx xxxx	43, 230
RXF2EIDL	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	xxxx xxxx	43, 230
RXF2EIDH	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	xxxx xxxx	43, 230
RXF2SIDL	SID2	SID1	SID0	—	EXIDEN	—	EID17	EID16	xx-x x-xx	43, 230
RXF2SIDH	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	xxxx xxxx	43, 230
RXF1EIDL	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	xxxx xxxx	43, 230
RXF1EIDH	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	xxxx xxxx	43, 230
RXF1SIDL	SID2	SID1	SID0	—	EXIDEN	—	EID17	EID16	xx-x x-xx	43, 230
RXF1SIDH	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	xxxx xxxx	43, 230
RXF0EIDL	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	xxxx xxxx	43, 230
RXF0EIDH	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	xxxx xxxx	43, 230
RXF0SIDL	SID2	SID1	SID0	—	EXIDEN	—	EID17	EID16	xx-x x-xx	43, 230
RXF0SIDH	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	xxxx xxxx	43, 230
B5D7(7)	B5D77	B5D76	B5D75	B5D74	B5D73	B5D72	B5D71	B5D70	xxxx xxxx	44, 230
B5D6(7)	B5D67	B5D66	B5D65	B5D64	B5D63	B5D62	B5D61	B5D60	xxxx xxxx	44, 230
B5D5(7)	B5D57	B5D56	B5D55	B5D54	B5D53	B5D52	B5D51	B5D50	xxxx xxxx	44, 230
B5D4(7)	B5D47	B5D46	B5D45	B5D44	B5D43	B5D42	B5D41	B5D40	xxxx xxxx	44, 230
B5D3(7)	B5D37	B5D36	B5D35	B5D34	B5D33	B5D32	B5D31	B5D30	xxxx xxxx	44, 230
B5D2(7)	B5D27	B5D26	B5D25	B5D24	B5D23	B5D22	B5D21	B5D20	xxxx xxxx	44, 230
B5D1(7)	B5D17	B5D16	B5D15	B5D14	B5D13	B5D12	B5D11	B5D10	xxxx xxxx	44, 230
B5D0(7)	B5D07	B5D06	B5D05	B5D04	B5D03	B5D02	B5D01	B5D00	xxxx xxxx	44, 230
B5DLC(7)	—	RXRTR	RB1	RB0	DLC3	DLC2	DLC1	DLC0	-xxx xxxx	44, 230
B5EIDL(7)	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	xxxx xxxx	44, 230
B5EIDH(7)	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	xxxx xxxx	44, 230
B5SIDL(7)	SID2	SID1	SID0	SRR	EXID/ EXIDE(5)	—	EID17	EID16	xxxx x-xx	44, 230
B5SIDH(7)	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	xxxx xxxx	44, 230
B5CON(5, 7)	RXFUL/ TXBIF	RXM1/ TXABT	RTRRO/ TXLARB	FILHIT4/ TXERR	FILHIT3/ TXREQ	FILHIT2/ RTREN	FILHIT1/ TXPRI1	FILHIT0/ TXPRI0	0000 0000	44, 230
B4D7(7)	B4D77	B4D76	B4D75	B4D74	B4D73	B4D72	B4D71	B4D70	xxxx xxxx	44, 230
B4D6(7)	B4D67	B4D66	B4D65	B4D64	B4D63	B4D62	B4D61	B4D60	xxxx xxxx	44, 230
B4D5(7)	B4D57	B4D56	B4D55	B4D54	B4D53	B4D52	B4D51	B4D50	xxxx xxxx	44, 230
B4D4(7)	B4D47	B4D46	B4D45	B4D44	B4D43	B4D42	B4D41	B4D40	xxxx xxxx	44, 230

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

Note 1: RA6 and associated bits are configured as port pins in RCIO and ECIO Oscillator mode only and read '0' in all other oscillator modes.

2: Bit 21 of the TBLPTRU allows access to the device configuration bits.

3: These registers are unused on PIC18F6X80 devices; always maintain these clear.

4: These bits have multiple functions depending on the CAN module mode selection.

5: Meaning of this register depends on whether this buffer is configured as transmit or receive.

6: RG5 is available as an input when $\overline{\text{MCLR}}$ is disabled.

7: This register reads all '0's until the ECAN module is set up in Mode 1 or Mode 2.

TABLE 4-3: REGISTER FILE SUMMARY (CONTINUED)

File Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Details on page:
B4D3(7)	B4D37	B4D36	B4D35	B4D34	B4D33	B4D32	B4D31	B4D30	xxxx xxxx	44, 230
B4D2(7)	B4D27	B4D26	B4D25	B4D24	B4D23	B4D22	B4D21	B4D20	xxxx xxxx	44, 230
B4D1(7)	B4D17	B4D16	B4D15	B4D14	B4D13	B4D12	B4D11	B4D10	xxxx xxxx	44, 230
B4D0(7)	B4D07	B4D06	B4D05	B4D04	B4D03	B4D02	B4D01	B4D00	xxxx xxxx	44, 230
B4DLC(7)	—	RXRTR	RB1	RB0	DLC3	DLC2	DLC1	DLC0	-xxx xxxx	44, 230
B4EIDL(7)	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	xxxx xxxx	44, 230
B4EIDH(7)	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	xxxx xxxx	44, 230
B4SIDL(7)	SID2	SID1	SID0	SRR	EXID/ EXIDE(5)	—	EID17	EID16	xxxx x-xx	44, 230
B4SIDH(7)	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	xxxx xxxx	44, 230
B4CON(5, 7)	RXFUL/ TXB3IF	RXM1/ TXABT	RTRRO/ TXLARB	FILHIT4/ TXERR	FILHIT3/ TXREQ	FILHIT2/ RTREN	FILHIT1/ TXPRI1	FILHIT0/ TXPRI0	0000 0000	44, 230
B3D7(7)	B3D77	B3D76	B3D75	B3D74	B3D73	B3D72	B3D71	B3D70	xxxx xxxx	44, 230
B3D6(7)	B3D67	B3D66	B3D65	B3D64	B3D63	B3D62	B3D61	B3D60	xxxx xxxx	44, 230
B3D5(7)	B3D57	B3D56	B3D55	B3D54	B3D53	B3D52	B3D51	B3D50	xxxx xxxx	44, 230
B3D4(7)	B3D47	B3D46	B3D45	B3D44	B3D43	B3D42	B3D41	B3D40	xxxx xxxx	45, 230
B3D3(7)	B3D37	B3D36	B3D35	B3D34	B3D33	B3D32	B3D31	B3D30	xxxx xxxx	45, 230
B3D2(7)	B3D27	B3D26	B3D25	B3D24	B3D23	B3D22	B3D21	B3D20	xxxx xxxx	45, 230
B3D1(7)	B3D17	B3D16	B3D15	B3D14	B3D13	B3D12	B3D11	B3D10	xxxx xxxx	45, 230
B3D0(7)	B3D07	B3D06	B3D05	B3D04	B3D03	B3D02	B3D01	B3D00	xxxx xxxx	45, 230
B3DLC(7)	—	RXRTR	RB1	RB0	DLC3	DLC2	DLC1	DLC0	-xxx xxxx	45, 230
B3EIDL(7)	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	xxxx xxxx	45, 230
B3EIDH(7)	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	xxxx xxxx	45, 230
B3SIDL(7)	SID2	SID1	SID0	SRR	EXID/ EXIDE(5)	—	EID17	EID16	xxxx x-xx	45, 230
B3SIDH(7)	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	xxxx xxxx	45, 230
B3CON(5, 7)	RXFUL/ TXBIF	RXM1/ TXABT	RTRRO/ TXLARB	FILHIT4/ TXERR	FILHIT3/ TXREQ	FILHIT2/ RTREN	FILHIT1/ TXPRI1	FILHIT0/ TXPRI0	0000 0000	45, 230
B2D7(7)	B2D77	B2D76	B2D75	B2D74	B2D73	B2D72	B2D71	B2D70	xxxx xxxx	45, 230
B2D6(7)	B2D67	B2D66	B2D65	B2D64	B2D63	B2D62	B2D61	B2D60	xxxx xxxx	45, 230
B2D5(7)	B2D57	B2D56	B2D55	B2D54	B2D53	B2D52	B2D51	B2D50	xxxx xxxx	45, 230
B2D4(7)	B2D47	B2D46	B2D45	B2D44	B2D43	B2D42	B2D41	B2D40	xxxx xxxx	45, 230
B2D3(7)	B2D37	B2D36	B2D35	B2D34	B2D33	B2D32	B2D31	B2D30	xxxx xxxx	45, 230
B2D2(7)	B2D27	B2D26	B2D25	B2D24	B2D23	B2D22	B2D21	B2D20	xxxx xxxx	45, 230
B2D1(7)	B2D17	B2D16	B2D15	B2D14	B2D13	B2D12	B2D11	B2D10	xxxx xxxx	45, 230
B2D0(7)	B2D07	B2D06	B2D05	B2D04	B2D03	B2D02	B2D01	B2D00	xxxx xxxx	45, 230
B2DLC(7)	—	RXRTR	RB1	RB0	DLC3	DLC2	DLC1	DLC0	-xxx xxxx	45, 230
B2EIDL(7)	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	xxxx xxxx	45, 230
B2EIDH(7)	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	xxxx xxxx	45, 230
B2SIDL(7)	SID2	SID1	SID0	SRR	EXID/ EXIDE(5)	—	EID17	EID16	xxxx x-xx	45, 230
B2SIDH(7)	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	xxxx xxxx	45, 230

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

Note 1: RA6 and associated bits are configured as port pins in RCIO and ECIO Oscillator mode only and read '0' in all other oscillator modes.

2: Bit 21 of the TBLPTRU allows access to the device configuration bits.

3: These registers are unused on PIC18F6X80 devices; always maintain these clear.

4: These bits have multiple functions depending on the CAN module mode selection.

5: Meaning of this register depends on whether this buffer is configured as transmit or receive.

6: RG5 is available as an input when $\overline{MCLR}$ is disabled.

7: This register reads all '0's until the ECAN module is set up in Mode 1 or Mode 2.

TABLE 4-3: REGISTER FILE SUMMARY (CONTINUED)

File Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Details on page:
B2CON(5, 7)	RXFUL/ TXBIF	RXM1/ TXABT	RTRRO/ TXLARB	FILHIT4/ TXERR	FILHIT3/ TXREQ	FILHIT2/ RTREN	FILHIT1/ TXPRI1	FILHIT0/ TXPRI0	0000 0000	45, 230
B1D7(7)	B1D77	B1D76	B1D75	B1D74	B1D73	B1D72	B1D71	B1D70	xxxx xxxx	45, 230
B1D6(7)	B1D67	B1D66	B1D65	B1D64	B1D63	B1D62	B1D61	B1D60	xxxx xxxx	45, 230
B1D5(7)	B1D57	B1D56	B1D55	B1D54	B1D53	B1D52	B1D51	B1D50	xxxx xxxx	45, 230
B1D4(7)	B1D47	B1D46	B1D45	B1D44	B1D43	B1D42	B1D41	B1D40	xxxx xxxx	45, 230
B1D3(7)	B1D37	B1D36	B1D35	B1D34	B1D33	B1D32	B1D31	B1D30	xxxx xxxx	45, 230
B1D2(7)	B1D27	B1D26	B1D25	B1D24	B1D23	B1D22	B1D21	B1D20	xxxx xxxx	45, 230
B1D1(7)	B1D17	B1D16	B1D15	B1D14	B1D13	B1D12	B1D11	B1D10	xxxx xxxx	46, 230
B1D0(7)	B1D07	B1D06	B1D05	B1D04	B1D03	B1D02	B1D01	B1D00	xxxx xxxx	46, 230
B1DLC(7)	—	RXRTR	RB1	RB0	DLC3	DLC2	DLC1	DLC0	-xxx xxxx	46, 230
B1EIDL(7)	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	xxxx xxxx	46, 230
B1EIDH(7)	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	xxxx xxxx	46, 230
B1SIDL(7)	SID2	SID1	SID0	SRR	EXID	—	EID17	EID16	xxxx x-xx	46, 230
B1SIDH(7)	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	xxxx xxxx	46, 230
B1CON(5, 7)	RXFUL/ TXBIF	RXM1/ TXABT	RTRRO/ TXLARB	FILHIT4/ TXERR	FILHIT3/ TXREQ	FILHIT2/ RTREN	FILHIT1/ TXPRI1	FILHIT0/ TXPRI0	0000 0000	46, 230
B0D7(7)	B0D77	B0D76	B0D75	B0D74	B0D73	B0D72	B0D71	B0D70	xxxx xxxx	46, 230
B0D6(7)	B0D67	B0D66	B0D65	B0D64	B0D63	B0D62	B0D61	B0D60	xxxx xxxx	46, 230
B0D5(7)	B0D57	B0D56	B0D55	B0D54	B0D53	B0D52	B0D51	B0D50	xxxx xxxx	46, 230
B0D4(7)	B0D47	B0D46	B0D45	B0D44	B0D43	B0D42	B0D41	B0D40	xxxx xxxx	46, 230
B0D3(7)	B0D37	B0D36	B0D35	B0D34	B0D33	B0D32	B0D31	B0D30	xxxx xxxx	46, 230
B0D2(7)	B0D27	B0D26	B0D25	B0D24	B0D23	B0D22	B0D21	B0D20	xxxx xxxx	46, 230
B0D1(7)	B0D17	B0D16	B0D15	B0D14	B0D13	B0D12	B0D11	B0D10	xxxx xxxx	46, 230
B0D0(7)	B0D07	B0D06	B0D05	B0D04	B0D03	B0D02	B0D01	B0D00	xxxx xxxx	46, 230
B0DLC(7)	—	RTR	RB1	RB0	DLC3	DLC2	DLC1	DLC0	-xxx xxxx	46, 230
B0EIDL(7)	EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0	xxxx xxxx	46, 230
B0EIDH(7)	EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8	xxxx xxxx	46, 230
B0SIDL(7)	SID2	SID1	SID0	SRR	EXID	—	EID17	EID16	xxxx x-xx	46, 230
B0SIDH(7)	SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3	xxxx xxxx	46, 230
B0CON(5, 7)	RXFUL/ TXBIF	RXM1/ TXABT	RTRRO/ TXLARB	FILHIT4/ TXERR	FILHIT3/ TXREQ	FILHIT2/ RTREN	FILHIT1/ TXPRI1	FILHIT0/ TXPRI0	0000 0000	46, 230
TXBIE(7)	—	—	—	TXB2IE	TXB1IE	TXB0IE	—	—	---0 00--	46, 230
BIE0(7)	B5IE	B4IE	B3IE	B2IE	B1IE	B0IE	RXB1IE	RXB0IE	0000 0000	46, 230
BSEL0(7)	B5TXEN	B4TXEN	B3TXEN	B2TXEN	B1TXEN	B0TXEN	—	—	0000 00--	46, 230
MSEL3(7)	FIL15_1	FIL15_0	FIL14_1	FIL14_0	FIL13_1	FIL13_0	FIL12_1	FIL12_0	0000 0000	46, 230
MSEL2(7)	FIL11_1	FIL11_0	FIL10_1	FIL10_0	FIL9_1	FIL9_0	FIL8_1	FIL8_0	0000 0000	46, 230
MSEL1(7)	FIL7_1	FIL7_0	FIL6_1	FIL6_0	FIL5_1	FIL5_0	FIL4_1	FIL4_0	0000 0101	46, 230
MSEL0(7)	FIL3_1	FIL3_0	FIL2_1	FIL2_0	FIL1_1	FIL1_0	FIL0_1	FIL0_0	0101 0000	46, 230
SDFLC(7)	—	—	—	DFLC4	DFLC3	DFLC2	DFLC1	DFLC0	---0 0000	46, 230
RXFCON1(7)	RXF15EN	RXF14EN	RXF13EN	RXF12EN	RXF11EN	RXF10EN	RXF9EN	RXF8EN	0000 0000	46, 230
RXFCON0(7)	RXF7EN	RXF6EN	RXF5EN	RXF4EN	RXF3EN	RXF2EN	RXF1EN	RXF0EN	0011 1111	47, 230

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

Note 1: RA6 and associated bits are configured as port pins in RCIO and ECIO Oscillator mode only and read '0' in all other oscillator modes.

2: Bit 21 of the TBLPTRU allows access to the device configuration bits.

3: These registers are unused on PIC18F6X80 devices; always maintain these clear.

4: These bits have multiple functions depending on the CAN module mode selection.

5: Meaning of this register depends on whether this buffer is configured as transmit or receive.

6: RG5 is available as an input when $\overline{\text{MCLR}}$ is disabled.

7: This register reads all '0's until the ECAN module is set up in Mode 1 or Mode 2.

TABLE 4-3: REGISTER FILE SUMMARY (CONTINUED)

File Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Details on page:
RXFBCON7(7)	F15BP_3	F15BP_2	F15BP_1	F15BP_0	F14BP_3	F14BP_2	F14BP_1	F14BP_01	0000 0000	47, 230
RXFBCON6(7)	F13BP_3	F13BP_2	F13BP_1	F13BP_0	F12BP_3	F12BP_2	F12BP_1	F12BP_01	0000 0000	47, 230
RXFBCON5(7)	F11BP_3	F11BP_2	F11BP_1	F11BP_0	F10BP_3	F10BP_2	F10BP_1	F10BP_01	0000 0000	47, 230
RXFBCON4(7)	F9BP_3	F9BP_2	F9BP_1	F9BP_0	F8BP_3	F8BP_2	F8BP_1	F8BP_01	0000 0000	47, 230
RXFBCON3(7)	F7BP_3	F7BP_2	F7BP_1	F7BP_0	F6BP_3	F6BP_2	F6BP_1	F6BP_01	0000 0000	47, 230
RXFBCON2(7)	F5BP_3	F5BP_2	F5BP_1	F5BP_0	F4BP_3	F4BP_2	F4BP_1	F4BP_01	0000 0000	47, 230
RXFBCON1(7)	F3BP_3	F3BP_2	F3BP_1	F3BP_0	F2BP_3	F2BP_2	F2BP_1	F2BP_01	0000 0000	47, 230
RXFBCON0(7)	F1BP_3	F1BP_2	F1BP_1	F1BP_0	F0BP_3	F0BP_2	F0BP_1	F0BP_01	0000 0000	47, 230

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

Note 1: RA6 and associated bits are configured as port pins in RCIO and ECIO Oscillator mode only and read '0' in all other oscillator modes.

2: Bit 21 of the TBLPTRU allows access to the device configuration bits.

3: These registers are unused on PIC18F6X80 devices; always maintain these clear.

4: These bits have multiple functions depending on the CAN module mode selection.

5: Meaning of this register depends on whether this buffer is configured as transmit or receive.

6: RG5 is available as an input when $\overline{\text{MCLR}}$ is disabled.

7: This register reads all '0's until the ECAN module is set up in Mode 1 or Mode 2.

4.10 Access Bank

The Access Bank is an architectural enhancement which is very useful for C compiler code optimization. The techniques used by the C compiler may also be useful for programs written in assembly.

This data memory region can be used for:

- Intermediate computational values
- Local variables of subroutines
- Faster context saving/switching of variables
- Common variables
- Faster evaluation/control of SFRs (no banking)

The Access Bank is comprised of the upper 160 bytes in Bank 15 (SFRs) and the lower 96 bytes in Bank 0. These two sections will be referred to as Access RAM High and Access RAM Low, respectively. Figure 4-7 indicates the Access RAM areas.

A bit in the instruction word specifies if the operation is to occur in the bank specified by the BSR register or in the Access Bank. This bit is denoted by the 'a' bit (for access bit).

When forced in the Access Bank (a = `0`), the last address in Access RAM Low is followed by the first address in Access RAM High. Access RAM High maps the Special Function Registers so that these registers can be accessed without any software overhead. This is useful for testing status flags and modifying control bits.

4.11 Bank Select Register (BSR)

The need for a large general purpose memory space dictates a RAM banking scheme. The data memory is partitioned into sixteen banks. When using direct addressing, the BSR should be configured for the desired bank.

BSR<3:0> holds the upper 4 bits of the 12-bit RAM address. The BSR<7:4> bits will always read '`0`'s and writes will have no effect.

A `MOVLB` instruction has been provided in the instruction set to assist in selecting banks.

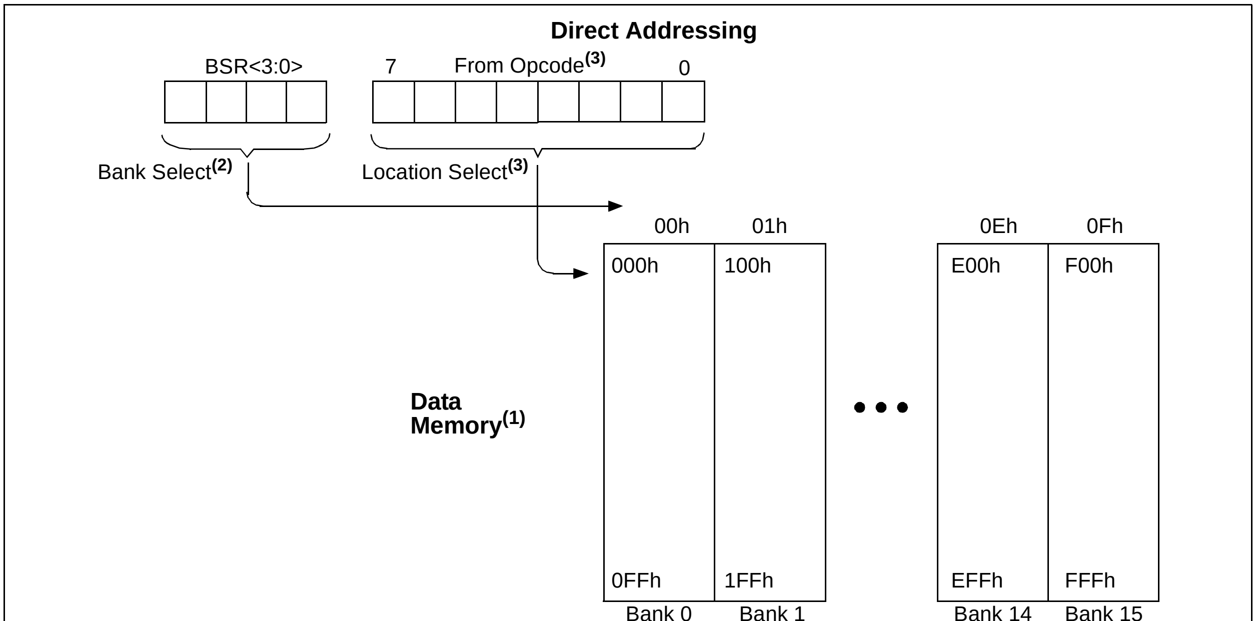
If the currently selected bank is not implemented, any read will return all '`0`'s and all writes are ignored. The Status register bits will be set/cleared as appropriate for the instruction performed.

Each Bank extends up to 0FFh (256 bytes). All data memory is implemented as static RAM.

A `MOVFF` instruction ignores the BSR since the 12-bit addresses are embedded into the instruction word.

Section 4.12 "Indirect Addressing, INDF and FSR Registers" provides a description of indirect addressing which allows linear addressing of the entire RAM space.

FIGURE 4-8: DIRECT ADDRESSING



Note 1: For register file map detail, see Table 4-2.

2: The access 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.

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

4.12 Indirect Addressing, INDF and FSR Registers

Indirect addressing is a mode of addressing data memory where the data memory address in the instruction is not fixed. An FSR register is used as a pointer to the data memory location that is to be read or written. Since this pointer is in RAM, the contents can be modified by the program. This can be useful for data tables in the data memory and for software stacks. Figure 4-9 shows the operation of indirect addressing. This shows the moving of the value to the data memory address specified by the value of the FSR register.

Indirect addressing is possible by using one of the INDF registers. Any instruction using the INDF register actually accesses the register pointed to by the File Select Register, FSR. Reading the INDF register itself, indirectly (FSR = `0`), will read 00h. Writing to the INDF register indirectly, results in a no operation. The FSR register contains a 12-bit address which is shown in Figure 4-10.

The INDFn register is not a physical register. Addressing INDFn actually addresses the register whose address is contained in the FSRn register (FSRn is a pointer). This is indirect addressing.

Example 4-4 shows a simple use of indirect addressing to clear the RAM in Bank 1 (locations 100h-1FFh) in a minimum number of instructions.

EXAMPLE 4-4: HOW TO CLEAR RAM (BANK 1) USING INDIRECT ADDRESSING

```
        LFSR   FSR0, 100h  ;
NEXT    CLRF   POSTINC0    ; Clear INDF
                           ; register and
                           ; inc pointer
        BTFSS  FSR0H, 1    ; All done with
                           ; Bank1?
        BRA    NEXT        ; NO, clear next
CONTINUE                   ; YES, continue
```

There are three Indirect Addressing registers. To address the entire data memory space (4096 bytes), these registers are 12-bits wide. To store the 12 bits of addressing information, two 8-bit registers are required. These Indirect Addressing registers are:

1. FSR0: composed of FSR0H:FSR0L
2. FSR1: composed of FSR1H:FSR1L
3. FSR2: composed of FSR2H:FSR2L

In addition, there are registers INDF0, INDF1 and INDF2 which are not physically implemented. Reading or writing to these registers activates indirect addressing with the value in the corresponding FSR register being the address of the data. If an instruction writes a value to INDF0, the value will be written to the address pointed to by FSR0H:FSR0L. A read from INDF1 reads the data from the address pointed to by FSR1H:FSR1L. INDFn can be used in code anywhere an operand can be used.

If INDF0, INDF1, or INDF2 are read indirectly via an FSR, all '`0`'s are read (zero bit is set). Similarly, if INDF0, INDF1, or INDF2 are written to indirectly, the operation will be equivalent to a `NOP` instruction and the Status bits are not affected.

4.12.1 INDIRECT ADDRESSING OPERATION

Each FSR register has an INDF register associated with it plus four additional register addresses. Performing an operation on one of these five registers determines how the FSR will be modified during indirect addressing.

When data access is done to one of the five INDFn locations, the address selected will configure the FSRn register to:

- Do nothing to FSRn after an indirect access (no change) – INDFn.
- Auto-decrement FSRn after an indirect access (post-decrement) – POSTDECn.
- Auto-increment FSRn after an indirect access (post-increment) – POSTINCn.
- Auto-increment FSRn before an indirect access (pre-increment) – PREINCn.
- Use the value in the WREG register as an offset to FSRn. Do not modify the value of the WREG or the FSRn register after an indirect access (no change) – PLUSWn.

When using the auto-increment or auto-decrement features, the effect on the FSR is not reflected in the Status register. For example, if the indirect address causes the FSR to equal '`0`', the Z bit will not be set.

Incrementing or decrementing an FSR affects all 12 bits. That is, when FSRnL overflows from an increment, FSRnH will be incremented automatically.

Adding these features allows the FSRn to be used as a stack pointer in addition to its uses for table operations in data memory.

Each FSR has an address associated with it that performs an indexed indirect access. When a data access to this INDFn location (PLUSWn) occurs, the FSRn is configured to add the signed value in the WREG register and the value in FSR to form the address before an indirect access. The FSR value is not changed.

If an FSR register contains a value that points to one of the INDFn, an indirect read will read 00h (zero bit is set), while an indirect write will be equivalent to a `NOP` (Status bits are not affected).

If an indirect addressing operation is done where the target address is an FSRnH or FSRnL register, the write operation will dominate over the pre- or post-increment/decrement functions.

FIGURE 4-9: INDIRECT ADDRESSING OPERATION

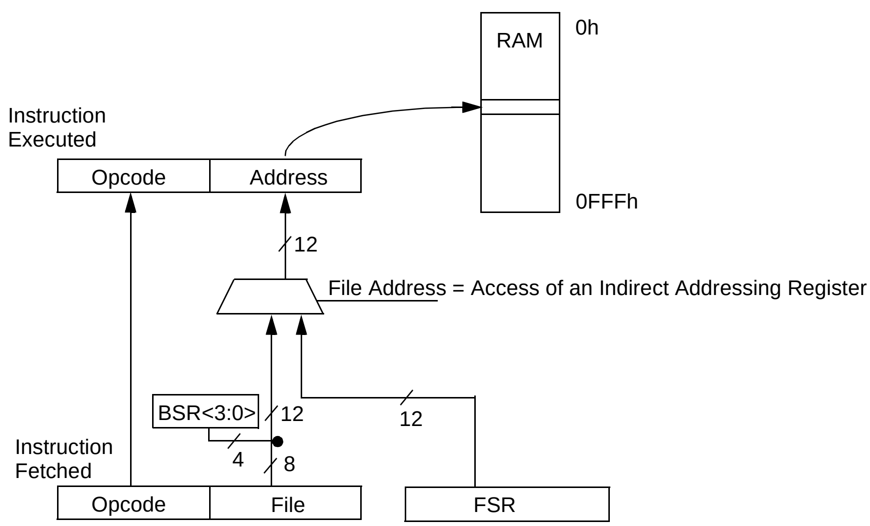
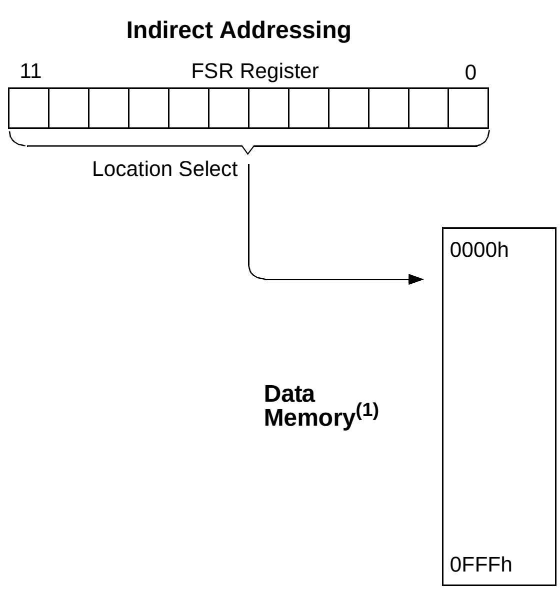


FIGURE 4-10: INDIRECT ADDRESSING



Note 1: For register file map detail, see Table 4-2.

4.13 Status Register

The Status register, shown in Register 4-3, contains the arithmetic status of the ALU. The Status register can be the destination for any instruction as with any other register. If the Status register is the destination for an instruction that affects the Z, DC, C, OV or N bits, then the write to these five bits is disabled. These bits are set or cleared according to the device logic. Therefore, the result of an instruction with the Status register as destination may be different than intended.

For example, `CLRF STATUS` will clear the upper three bits and set the Z bit. This leaves the Status register as `000u u1uu` (where `u` = unchanged).

It is recommended, therefore, 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 from the Status register. For other instructions not affecting any status bits, see Table 25-2.

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

REGISTER 4-3: STATUS REGISTER

U-0	U-0	U-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
—	—	—	N	OV	Z	DC	C
bit 7							bit 0

bit 7-5 **Unimplemented:** Read as '0'

bit 4 **N:** Negative bit

This bit is used for signed arithmetic (2's complement). It indicates whether the result was negative (ALU MSB = 1).

1 = Result was negative
0 = Result was positive

bit 3 **OV:** Overflow bit

This bit is used for signed arithmetic (2's complement). It indicates an overflow of the 7-bit magnitude which causes the sign bit (bit 7) to change state.

1 = Overflow occurred for signed arithmetic (in this arithmetic operation)
0 = No overflow occurred

bit 2 **Z:** Zero bit

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

bit 1 **DC:** Digit carry/$\overline{\text{borrow}}$ bit

For `ADDWF`, `ADDLW`, `SUBLW`, and `SUBWF` instructions:

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

Note: For $\overline{\text{borrow}}$, the polarity is reversed. A subtraction is executed by adding the 2's complement of the second operand. For rotate (`RRF`, `RLF`) instructions, this bit is loaded with either the bit 4 or bit 3 of the source register.

bit 0 **C:** Carry/$\overline{\text{borrow}}$ bit

For `ADDWF`, `ADDLW`, `SUBLW`, and `SUBWF` instructions:

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: For $\overline{\text{borrow}}$, the polarity is reversed. A subtraction is executed by adding the 2's complement of the second operand. For rotate (`RRF`, `RLF`) instructions, this bit is loaded with either the high or low-order bit of the source register.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

4.14 RCON Register

The Reset Control (RCON) register contains flag bits that allow differentiation between the sources of a device Reset. These flags include the $\overline{TO}$, $\overline{PD}$, $\overline{POR}$, $\overline{BOR}$ and $\overline{RI}$ bits. This register is readable and writable.

> **Note 1:** It is recommended that the $\overline{POR}$ bit be set after a Power-on Reset has been detected so that subsequent Power-on Resets may be detected.
>
> **2:** Brown-out Reset is said to have occurred when $\overline{BOR}$ is '0' and $\overline{POR}$ is '1' (assuming that $\overline{POR}$ was set to '1' by software immediately after POR).

REGISTER 4-4: RCON REGISTER

R/W-0	U-0	U-0	R/W-1	R/W-1	R/W-1	R/W-0	R/W-0
IPEN	—	—	$\overline{RI}$	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	$\overline{BOR}$
bit 7							bit 0

bit 7 **IPEN:** Interrupt Priority Enable bit
1 = Enable priority levels on interrupts
0 = Disable priority levels on interrupts (PIC16CXXX Compatibility mode)

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

bit 4 **$\overline{RI}$:** RESET Instruction Flag bit
1 = The RESET instruction was not executed
0 = The RESET instruction was executed causing a device Reset (must be set in software after a Brown-out Reset occurs)

bit 3 **$\overline{TO}$:** Watchdog Time-out Flag bit
1 = After power-up, CLRWDT instruction, or SLEEP instruction
0 = A WDT time-out occurred

bit 2 **$\overline{PD}$:** Power-down Detection Flag bit
1 = After power-up or by the CLRWDT instruction
0 = By execution of the SLEEP instruction

bit 1 **$\overline{POR}$:** Power-on Reset Status bit
1 = A Power-on Reset has not occurred
0 = A Power-on Reset occurred (must be set in software after a Power-on Reset occurs)

bit 0 **$\overline{BOR}$:** Brown-out Reset Status bit
1 = A Brown-out Reset has not occurred
0 = A Brown-out Reset occurred (must be set in software after a Brown-out Reset occurs)

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

5.0 FLASH PROGRAM MEMORY

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

A read from program memory is executed on one byte at a time. A write to program memory is executed on blocks of 8 bytes at a time. Program memory is erased in blocks of 64 bytes at a time. A bulk erase operation cannot be issued from user code.

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

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

5.1 Table Reads and Table Writes

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

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

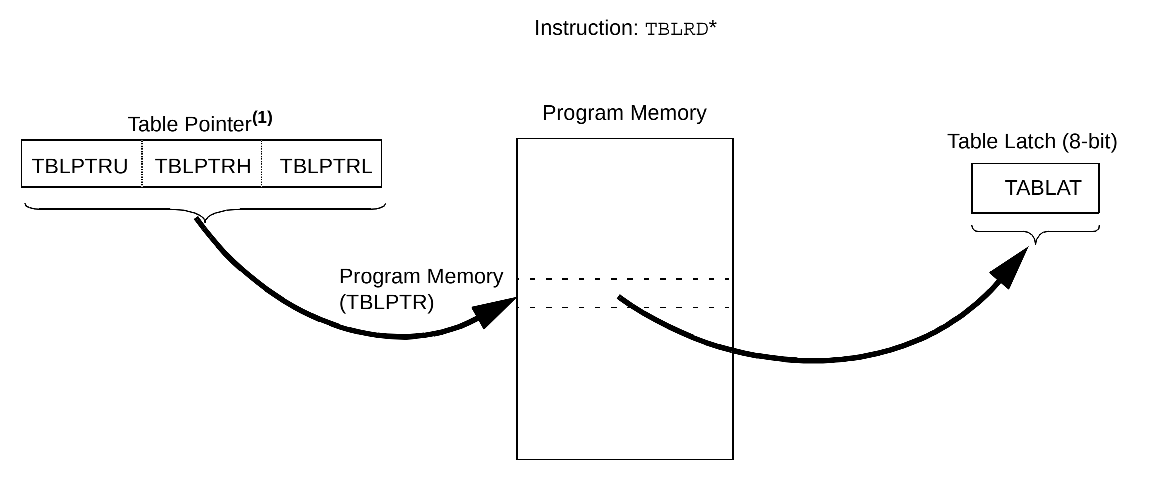
The program memory space is 16 bits wide, while the data RAM space is 8-bits wide. Table reads and table writes move data between these two memory spaces through an 8-bit register (TABLAT).

Table read operations retrieve data from program memory and places it into the data RAM space. Figure 5-1 shows the operation of a table read with program memory and data RAM.

Table write operations store data from the data memory space into holding registers in program memory. The procedure to write the contents of the holding registers into program memory is detailed in **Section 5.5 "Writing to Flash Program Memory"**. Figure 5-2 shows the operation of a table write with program memory and data RAM.

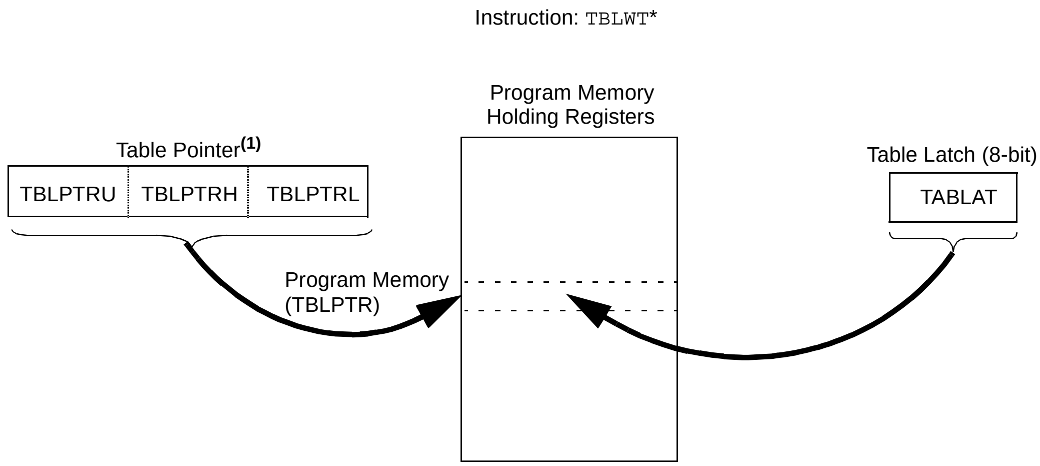
Table operations work with byte entities. A table block containing data, rather than program instructions, is not required to be word aligned. Therefore, a table block can start and end at any byte address. If a table write is being used to write executable code into program memory, program instructions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

FIGURE 5-1: TABLE READ OPERATION



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

FIGURE 5-2: TABLE WRITE OPERATION



Note 1: Table Pointer actually points to one of eight holding registers, the address of which is determined by TBLPTRL<2:0>. The process for physically writing data to the program memory array is discussed in **Section 5.5 "Writing to Flash Program Memory"**.

5.2 Control Registers

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

- EECON1 register
- EECON2 register
- TABLAT register
- TBLPTR registers

5.2.1 EECON1 AND EECON2 REGISTERS

EECON1 is the control register for memory accesses.

EECON2 is not a physical register. Reading EECON2 will read all '0's. The EECON2 register is used exclusively in the memory write and erase sequences.

Control bit EEPGD determines if the access will be a program or data EEPROM memory access. When clear, any subsequent operations will operate on the data EEPROM memory. When set, any subsequent operations will operate on the program memory.

Control bit CFGS determines if the access will be to the configuration/calibration registers or to program memory/data EEPROM memory. When set, subsequent operations will operate on configuration registers regardless of EEPGD (see **Section 24.0 "Special Features of the CPU"**). When clear, memory selection access is determined by EEPGD.

The FREE bit, when set, will allow a program memory erase operation. When the FREE bit is set, the erase operation is initiated on the next WR command. When FREE is clear, only writes are enabled.

The WREN bit, when set, will allow a write operation. On power-up, the WREN bit is clear. The WRERR bit is set when a write operation is interrupted by a $\overline{\text{MCLR}}$ Reset or a WDT Time-out Reset during normal operation. In these situations, the user can check the WRERR bit and rewrite the location. It is necessary to reload the data and address registers (EEDATA and EEADR) due to Reset values of zero.

The WR control bit initiates write operations. The bit cannot be cleared, only set in software; it is cleared in hardware at the completion of the write operation. The inability to clear the WR bit in software prevents the accidental or premature termination of a write operation.

Note:	Interrupt flag bit, EEIF in the PIR2 register, is set when the write is complete. It must be cleared in software.

REGISTER 5-1: EECON1 REGISTER (ADDRESS FA6h)

R/W-x	R/W-x	U-0	R/W-0	R/W-x	R/W-0	R/S-0	R/S-0
EEPGD	CFGS	—	FREE	WRERR	WREN	WR	RD
bit 7							bit 0

bit 7 **EEPGD:** Flash Program or Data EEPROM Memory Select bit
1 = Access Flash program memory
0 = Access data EEPROM memory

bit 6 **CFGS:** Flash Program/Data EEPROM or Configuration Select bit
1 = Access configuration registers
0 = Access Flash program or data EEPROM memory

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

bit 4 **FREE:** Flash Row Erase Enable bit
1 = Erase the program memory row addressed by TBLPTR on the next WR command (cleared by completion of erase operation)
0 = Perform write only

bit 3 **WRERR:** Flash Program/Data EEPROM Error Flag bit
1 = A write operation is prematurely terminated (any Reset during self-timed programming in normal operation)
0 = The write operation completed

Note: When a WRERR occurs, the EEPGD and CFGS bits are not cleared. This allows tracing of the error condition.

bit 2 **WREN:** Flash Program/Data EEPROM Write Enable bit
1 = Allows write cycles
0 = Inhibits write to the EEPROM

bit 1 **WR:** Write Control bit
1 = Initiates a data EEPROM erase/write cycle or a program memory erase cycle or write cycle. (The operation is self-timed and the bit is cleared by hardware once write is complete. The WR bit can only be set (not cleared) in software.)
0 = Write cycle to the EEPROM is complete

bit 0 **RD:** Read Control bit
1 = Initiates an EEPROM read. (Read takes one cycle. RD is cleared in hardware. The RD bit can only be set (not cleared) in software. RD bit cannot be set when EEPGD = 1.)
0 = Does not initiate an EEPROM read

Legend:		
R = Readable bit	U = Unimplemented bit, read as '0'	
W = Writable bit	S = Settable bit	- n = Value after erase
'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

5.2.2 TABLAT – TABLE LATCH REGISTER

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

5.2.3 TBLPTR – TABLE POINTER REGISTER

The Table Pointer (TBLPTR) addresses a byte within the program memory. The TBLPTR is comprised of three SFR registers: Table Pointer Upper Byte, Table Pointer High Byte and Table Pointer Low Byte (TBLPTRU:TBLPTRH:TBLPTRL). These three registers join to form a 22-bit wide pointer. The low-order 21 bits allow the device to address up to 2 Mbytes of program memory space. The 22nd bit allows access to the device ID, the user ID and the configuration bits.

The Table Pointer, TBLPTR, is used by the `TBLRD` and `TBLWT` instructions. These instructions can update the TBLPTR in one of four ways based on the table operation. These operations are shown in Table 5-1. These operations on the TBLPTR only affect the low-order 21 bits.

5.2.4 TABLE POINTER BOUNDARIES

TBLPTR is used in reads, writes and erases of the Flash program memory.

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

When a `TBLWT` is executed, the three LSbs of the Table Pointer (TBLPTR<2:0>) determine which of the eight program memory holding registers is written to. When the timed write to program memory (long write) begins, the 19 MSbs of the Table Pointer (TBLPTR<21:3>) will determine which program memory block of 8 bytes is written to. For more detail, see **Section 5.5 "Writing to Flash Program Memory"**.

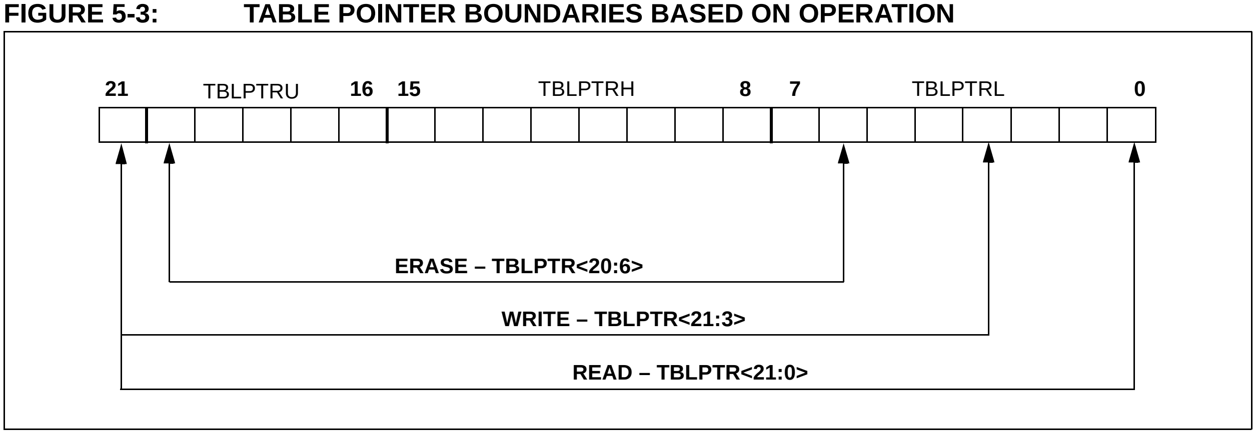
When an erase of program memory is executed, the 16 MSbs of the Table Pointer (TBLPTR<21:6>) point to the 64-byte block that will be erased. The Least Significant bits (TBLPTR<5:0>) are ignored.

Figure 5-3 describes the relevant boundaries of TBLPTR based on Flash program memory operations.

TABLE 5-1: TABLE POINTER OPERATIONS WITH `TBLRD` AND `TBLWT` INSTRUCTIONS

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

FIGURE 5-3: TABLE POINTER BOUNDARIES BASED ON OPERATION



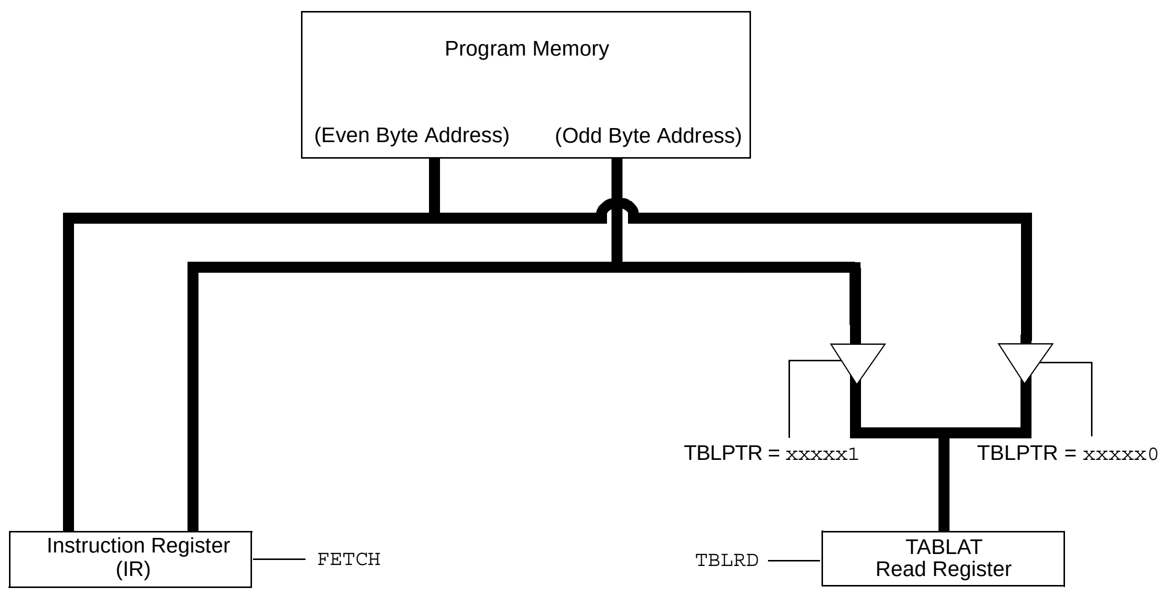
5.3 Reading the Flash Program Memory

The TBLRD instruction is used to retrieve data from program memory and places it into data RAM. Table reads from program memory are performed one byte at a time.

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

The 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 5-4 shows the interface between the internal program memory and the TABLAT.

FIGURE 5-4: READS FROM FLASH PROGRAM MEMORY



EXAMPLE 5-1: READING A FLASH PROGRAM MEMORY WORD

```
            MOVLW     upper(CODE_ADDR)          ; Load TBLPTR with the base
            MOVWF     TBLPTRU                   ; address of the word
            MOVLW     high(CODE_ADDR)
            MOVWF     TBLPTRH
            MOVLW     low(CODE_ADDR_LOW)
            MOVWF     TBLPTRL
READ_WORD
            TBLRD*+                             ; read into TABLAT and increment
            MOVF      TABLAT, W                 ; get data
            MOVWF     LSB
            TBLRD*+                             ; read into TABLAT and increment
            MOVF      TABLAT, W                 ; get data
            MOVWF     MSB
```

5.4 Erasing Flash Program Memory

The minimum erase block is 32 words or 64 bytes. 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.

When initiating an erase sequence from the microcontroller itself, a block of 64 bytes of program memory is erased. The Most Significant 16 bits of the TBLPTR<21:6> point to the block being erased. TBLPTR<5:0> are ignored.

The EECON1 register commands the erase operation. The EEPGD bit must be set to point to the Flash program memory. The WREN bit must be set to enable write operations. The FREE bit is set to select an erase operation.

For protection, the write initiate sequence for EECON2 must be used.

A long write is necessary for erasing the internal Flash. Instruction execution is halted while in a long write cycle. The long write will be terminated by the internal programming timer.

5.4.1 FLASH PROGRAM MEMORY ERASE SEQUENCE

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

1. Load table pointer with address of row being erased.
2. Set the EECON1 register for the erase operation:
   - set EEPGD bit to point to program memory;
   - clear the CFGS bit to access program memory;
   - set WREN bit to enable writes;
   - set FREE bit to enable the erase.
3. Disable interrupts.
4. Write 55h to EECON2.
5. Write 0AAh to EECON2.
6. Set the WR bit. This will begin the row erase cycle.
7. The CPU will stall for duration of the erase (about 2 ms using internal timer).
8. Execute a `NOP`.
9. Re-enable interrupts.

EXAMPLE 5-2: ERASING A FLASH PROGRAM MEMORY ROW

```
                MOVLW    upper(CODE_ADDR)    ; load TBLPTR with the base
                MOVWF    TBLPTRU             ; address of the memory block
                MOVLW    high(CODE_ADDR)
                MOVWF    TBLPTRH
                MOVLW    low(CODE_ADDR)
                MOVWF    TBLPTRL
    ERASE_ROW
                BSF      EECON1, EEPGD       ; point to Flash program memory
                BCF      EECON1, CFGS        ; access Flash program memory
                BSF      EECON1, WREN        ; enable write to memory
                BSF      EECON1, FREE        ; enable Row Erase operation
                BCF      INTCON, GIE         ; disable interrupts
                MOVLW    55h
                MOVWF    EECON2              ; write 55h
  Required      MOVLW    0AAh
  Sequence      MOVWF    EECON2              ; write 0AAh
                BSF      EECON1, WR          ; start erase (CPU stall)
                NOP
                BSF      INTCON, GIE         ; re-enable interrupts
```


5.5 Writing to Flash Program Memory

The minimum programming block is 4 words or 8 bytes. Word or byte programming is not supported.

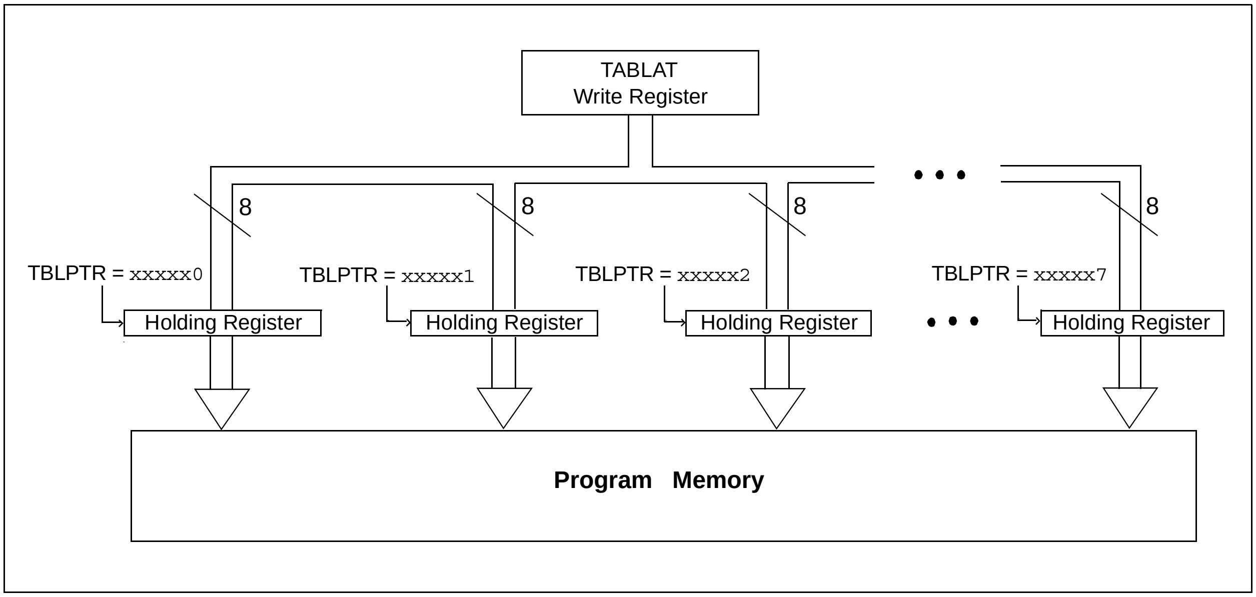
Table writes are used internally to load the holding registers needed to program the Flash memory. There are eight holding registers used by the table writes for programming.

Since the Table Latch (TABLAT) is only a single byte, the `TBLWT` instruction has to be executed 8 times for each programming operation. All of the table write operations will essentially be short writes because only the holding registers are written. At the end of updating eight registers, the EECON1 register must be written to, to start the programming operation with a long write.

The long write is necessary for programming the internal Flash. Instruction execution is halted while in a long write cycle. The long write will be terminated by the internal programming timer.

The EEPROM on-chip 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 for byte or word operations.

FIGURE 5-5: TABLE WRITES TO FLASH PROGRAM MEMORY



5.5.1 FLASH PROGRAM MEMORY WRITE SEQUENCE

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

1. Read 64 bytes into RAM.
2. Update data values in RAM as necessary.
3. Load table pointer with address being erased.
4. Do the row erase procedure.
5. Load table pointer with address of first byte being written.
6. Write the first 8 bytes into the holding registers with auto-increment.
7. Set the EECON1 register for the write operation:
   - set EEPGD bit to point to program memory;
   - clear the CFGS bit to access program memory;
   - set WREN to enable byte writes.
8. Disable interrupts.
9. Write 55h to EECON2.
10. Write 0AAh to EECON2.
11. Set the WR bit. This will begin the write cycle.
12. The CPU will stall for duration of the write (about 5 ms using internal timer).
13. Execute a `NOP`.
14. Re-enable interrupts.
15. Repeat steps 6-14 seven times to write 64 bytes.
16. Verify the memory (table read).

This procedure will require about 40 ms to update one row of 64 bytes of memory. An example of the required code is given in Example 5-3.

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

EXAMPLE 5-3: WRITING TO FLASH PROGRAM MEMORY

```
               MOVLW    D'64                ; number of bytes in erase block
               MOVWF    COUNTER
               MOVLW    high(BUFFER_ADDR)   ; point to buffer
               MOVWF    FSR0H
               MOVLW    low(BUFFER_ADDR)
               MOVWF    FSR0L
               MOVLW    upper(CODE_ADDR)    ; Load TBLPTR with the base
               MOVWF    TBLPTRU             ; address of the memory block
               MOVLW    high(CODE_ADDR)
               MOVWF    TBLPTRH
               MOVLW    low(CODE_ADDR)
               MOVWF    TBLPTRL
READ_BLOCK
               TBLRD*+                      ; read into TABLAT, and inc
               MOVF     TABLAT, W           ; get data
               MOVWF    POSTINC0            ; store data
               DECFSZ   COUNTER             ; done?
               BRA      READ_BLOCK          ; repeat
MODIFY_WORD
               MOVLW    high(DATA_ADDR)     ; point to buffer
               MOVWF    FSR0H
               MOVLW    low(DATA_ADDR)
               MOVWF    FSR0L
               MOVLW    low(NEW_DATA)       ; update buffer word
               MOVWF    POSTINC0
               MOVLW    high(NEW_DATA)
               MOVWF    INDF0
```


EXAMPLE 5-3: WRITING TO FLASH PROGRAM MEMORY (CONTINUED)

```
ERASE_BLOCK
               MOVLW     upper(CODE_ADDR)         ; load TBLPTR with the base
               MOVWF     TBLPTRU                  ; address of the memory block
               MOVLW     high(CODE_ADDR)
               MOVWF     TBLPTRH
               MOVLW     low(CODE_ADDR)
               MOVWF     TBLPTRL
               BSF       EECON1, EEPGD            ; point to Flash program memory
               BCF       EECON1, CFGS             ; access Flash program memory
               BSF       EECON1, WREN             ; enable write to memory
               BSF       EECON1, FREE             ; enable Row Erase operation
               BCF       INTCON, GIE              ; disable interrupts
               MOVLW     55h
               MOVWF     EECON2                   ; write 55H
    Required   MOVLW     0AAh
    Sequence   MOVWF     EECON2                   ; write AAH
               BSF       EECON1, WR               ; start erase (CPU stall)
               NOP
               BSF       INTCON, GIE              ; re-enable interrupts
               TBLRD*-                            ; dummy read decrement
WRITE_BUFFER_BACK
               MOVLW     8                        ; number of write buffer groups of 8 bytes
               MOVWF     COUNTER_HI
               MOVLW     high(BUFFER_ADDR)        ; point to buffer
               MOVWF     FSR0H
               MOVLW     low(BUFFER_ADDR)
               MOVWF     FSR0L
PROGRAM_LOOP
               MOVLW     8                        ; number of bytes in holding register
               MOVWF     COUNTER
WRITE_WORD_TO_HREGS
               MOVFW     POSTINC0, W              ; get low byte of buffer data
               MOVWF     TABLAT                   ; present data to table latch
               TBLWT+*                            ; write data, perform a short write
                                                  ; to internal TBLWT holding register.
               DECFSZ    COUNTER                  ; loop until buffers are full
               BRA       WRITE_WORD_TO_HREGS
PROGRAM_MEMORY
               BSF       EECON1, EEPGD            ; point to Flash program memory
               BCF       EECON1, CFGS             ; access Flash program memory
               BSF       EECON1, WREN             ; enable write to memory
               BCF       INTCON, GIE              ; disable interrupts
               MOVLW     55h
               MOVWF     EECON2                   ; write 55h
    Required   MOVLW     0AAh
    Sequence   MOVWF     EECON2                   ; write 0AAh
               BSF       EECON1, WR               ; start program (CPU stall)
               NOP
               BSF       INTCON, GIE              ; re-enable interrupts
               DECFSZ    COUNTER_HI               ; loop until done
               BRA       PROGRAM_LOOP
               BCF       EECON1, WREN             ; disable write to memory
```

5.5.2 WRITE VERIFY

Depending on the application, good programming practice may dictate that the value written to the memory should be verified against the original value. This should be used in applications where excessive writes can stress bits near the specification limit.

5.5.3 UNEXPECTED TERMINATION OF WRITE OPERATION

If a write is terminated by an unplanned event, such as loss of power or an unexpected Reset, the memory location just programmed should be verified and reprogrammed if needed. The WRERR bit is set when a write operation is interrupted by a $\overline{\text{MCLR}}$ Reset or a WDT Time-out Reset during normal operation. In these situations, users can check the WRERR bit and rewrite the location.

5.5.4 PROTECTION AGAINST SPURIOUS WRITES

To protect against spurious writes to Flash program memory, the write initiate sequence must also be followed. See **Section 24.0 "Special Features of the CPU"** for more detail.

5.6 Flash Program Operation During Code Protection

See **Section 24.0 "Special Features of the CPU"** for details on code protection of Flash program memory.

TABLE 5-2: REGISTERS ASSOCIATED WITH PROGRAM FLASH MEMORY

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
TBLPTRU	—	—	bit 21	Program Memory Table Pointer Upper Byte (TBLPTR<20:16>)					`--00 0000`	`--00 0000`
TBPLTRH	Program Memory Table Pointer High Byte (TBLPTR<15:8>)								`0000 0000`	`0000 0000`
TBLPTRL	Program Memory Table Pointer High Byte (TBLPTR<7:0>)								`0000 0000`	`0000 0000`
TABLAT	Program Memory Table Latch								`0000 0000`	`0000 0000`
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INTE	RBIE	TMR0IF	INTF	RBIF	`0000 0000`	`0000 0000`
EECON2	EEPROM Control Register 2 (not a physical register)								—	—
EECON1	EEPGD	CFGS	—	FREE	WRERR	WREN	WR	RD	`xx-0 x000`	`uu-0 u000`
IPR2	—	CMIP	—	EEIP	BCLIP	LVDIP	TMR3IP	CCP2IP	`-1-1 1111`	`-1-1 1111`
PIR2	—	CMIF	—	EEIF	BCLIF	LVDIF	TMR3IF	CCP2IF	`-0-0 0000`	`-0-0 0000`
PIE2	—	CMIE	—	EEIE	BCLIE	LVDIE	TMR3IE	CCP2IE	`-0-0 0000`	`-0-0 0000`

Legend: `x` = unknown, `u` = unchanged, `r` = reserved, – = unimplemented, read as '`0`'.
Shaded cells are not used during Flash/EEPROM access.

6.0 EXTERNAL MEMORY INTERFACE

Note:	The external memory interface is not implemented on PIC18F6X8X (64/68-pin) devices.

The external memory interface is a feature of the PIC18F8X8X devices that allows the controller to access external memory devices (such as Flash, EPROM, SRAM, etc.) as program memory.

The physical implementation of the interface uses 27 pins. These pins are reserved for external address/ data bus functions; they are multiplexed with I/O port pins on four ports. Three I/O ports are multiplexed with the address/data bus, while the fourth port is multiplexed with the bus control signals. The I/O port functions are enabled when the EBDIS bit in the MEMCON register is set (see Register 6-1). A list of the multiplexed pins and their functions is provided in Table 6-1.

As implemented in the PIC18F8X8X devices, the interface operates in a similar manner to the external memory interface introduced on PIC18C601/801 microcontrollers. The most notable difference is that the interface on PIC18F8X8X devices only operates in 16-bit modes. The 8-bit mode is not supported.

For a more complete discussion of the operating modes that use the external memory interface, refer to **Section 4.1.1 "PIC18F8X8X Program Memory Modes"**.

6.1 Program Memory Modes and the External Memory Interface

As previously noted, PIC18F8X8X controllers are capable of operating in any one of four program memory modes using combinations of on-chip and external program memory. The functions of the multiplexed port pins depend on the program memory mode selected as well as the setting of the EBDIS bit.

In **Microprocessor Mode**, the external bus is always active and the port pins have only the external bus function.

In **Microcontroller Mode,** the bus is not active and the pins have their port functions only. Writes to the MEMCOM register are not permitted.

In **Microprocessor with Boot Block** or **Extended Microcontroller Mode,** the external program memory bus shares I/O port functions on the pins. When the device is fetching or doing table read/table write operations on the external program memory space, the pins will have the external bus function. If the device is fetching and accessing internal program memory locations only, the EBDIS control bit will change the pins from external memory to I/O port functions. When EBDIS = `0`, the pins function as the external bus. When EBDIS = `1`, the pins function as I/O ports.

REGISTER 6-1: MEMCON REGISTER

R/W-0	U-0	R/W-0	R/W-0	U-0	U-0	R/W-0	R/W-0
EBDIS(1)	—	WAIT1	WAIT0	—	—	WM1	WM0
bit 7							bit 0

bit 7 **EBDIS**: External Bus Disable bit(1)
1 = External system bus disabled, all external bus drivers are mapped as I/O ports
0 = External system bus enabled and I/O ports are disabled

Note 1: This bit is ignored when device is accessing external memory either to fetch an instruction or perform `TBLRD/TBLWT`.

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

bit 5-4 **WAIT<1:0>**: Table Reads and Writes Bus Cycle Wait Count bits
11 = Table reads and writes will wait 0 T_{CY}
10 = Table reads and writes will wait 1 T_{CY}
01 = Table reads and writes will wait 2 T_{CY}
00 = Table reads and writes will wait 3 T_{CY}

bit 3-2 **Unimplemented**: Read as '0'

bit 1-0 **WM<1:0>**: `TBLWT` Operation with 16-bit Bus bits
1x = Word Write mode: LSB and MSB word output, $\overline{WRH}$ active when MSB written
01 = Byte Select mode: TABLAT data copied on both MS and LS Byte, $\overline{WRH}$ and ($\overline{UB}$ or $\overline{LB}$) will activate
00 = Byte Write mode: TABLAT data copied on both MS and LS Byte, $\overline{WRH}$ or $\overline{WRL}$ will activate

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

Note: The MEMCON register is held in Reset in Microcontroller mode.

If the device fetches or accesses external memory while EBDIS = 1, the pins will switch to external bus. If the EBDIS bit is set by a program executing from external memory, the action of setting the bit will be delayed until the program branches into the internal memory. At that time, the pins will change from external bus to I/O ports.

When the device is executing out of internal memory (with EBDIS = 0) in Microprocessor with Boot Block mode or Extended Microcontroller mode, the control signals will be in inactive. They will go to a state where the AD<15:0>, A<19:16> are tri-state; the $\overline{OE}$, $\overline{WRH}$, $\overline{WRL}$, $\overline{UB}$ and $\overline{LB}$ signals are '1'; and ALE and BA0 are '0'.

TABLE 6-1: PIC18F8X8X EXTERNAL BUS – I/O PORT FUNCTIONS

Name	Port	Bit	Function
RD0/AD0	PORTD	bit 0	Input/Output or System Bus Address bit 0 or Data bit 0
RD1/AD1	PORTD	bit 1	Input/Output or System Bus Address bit 1 or Data bit 1
RD2/AD2	PORTD	bit 2	Input/Output or System Bus Address bit 2 or Data bit 2
RD3/AD3	PORTD	bit 3	Input/Output or System Bus Address bit 3 or Data bit 3
RD4/AD4	PORTD	bit 4	Input/Output or System Bus Address bit 4 or Data bit 4
RD5/AD5	PORTD	bit 5	Input/Output or System Bus Address bit 5 or Data bit 5
RD6/AD6	PORTD	bit 6	Input/Output or System Bus Address bit 6 or Data bit 6
RD7/AD7	PORTD	bit 7	Input/Output or System Bus Address bit 7 or Data bit 7
RE0/AD8	PORTE	bit 0	Input/Output or System Bus Address bit 8 or Data bit 8
RE1/AD9	PORTE	bit 1	Input/Output or System Bus Address bit 9 or Data bit 9
RE2/AD10	PORTE	bit 2	Input/Output or System Bus Address bit 10 or Data bit 10
RE3/AD11	PORTE	bit 3	Input/Output or System Bus Address bit 11 or Data bit 11
RE4/AD12	PORTE	bit 4	Input/Output or System Bus Address bit 12 or Data bit 12
RE5/AD13	PORTE	bit 5	Input/Output or System Bus Address bit 13 or Data bit 13
RE6/AD14	PORTE	bit 6	Input/Output or System Bus Address bit 14 or Data bit 14
RE7/AD15	PORTE	bit 7	Input/Output or System Bus Address bit 15 or Data bit 15
RH0/A16	PORTH	bit 0	Input/Output or System Bus Address bit 16
RH1/A17	PORTH	bit 1	Input/Output or System Bus Address bit 17
RH2/A18	PORTH	bit 2	Input/Output or System Bus Address bit 18
RH3/A19	PORTH	bit 3	Input/Output or System Bus Address bit 19
RJ0/ALE	PORTJ	bit 0	Input/Output or System Bus Address Latch Enable (ALE) Control pin
RJ1/$\overline{OE}$	PORTJ	bit 1	Input/Output or System Bus Output Enable ($\overline{OE}$) Control pin
RJ2/$\overline{WRL}$	PORTJ	bit 2	Input/Output or System Bus Write Low ($\overline{WRL}$) Control pin
RJ3/$\overline{WRH}$	PORTJ	bit 3	Input/Output or System Bus Write High ($\overline{WRH}$) Control pin
RJ4/BA0	PORTJ	bit 4	Input/Output or System Bus Byte Address bit 0
RJ5/$\overline{CE}$	PORTJ	bit 5	Input/Output or Chip Enable
RJ6/$\overline{LB}$	PORTJ	bit 6	Input/Output or System Bus Lower Byte Enable ($\overline{LB}$) Control pin
RJ7/$\overline{UB}$	PORTJ	bit 7	Input/Output or System Bus Upper Byte Enable ($\overline{UB}$) Control pin

6.2 16-bit Mode

The external memory interface implemented in PIC18F8X8X devices operates only in 16-bit mode. The mode selection is not software configurable but is programmed via the configuration bits.

The WM<1:0> bits in the MEMCON register determine three types of connections in 16-bit mode. They are referred to as:

- 16-bit Byte Write
- 16-bit Word Write
- 16-bit Byte Select

These three different configurations allow the designer maximum flexibility in using 8-bit and 16-bit memory devices.

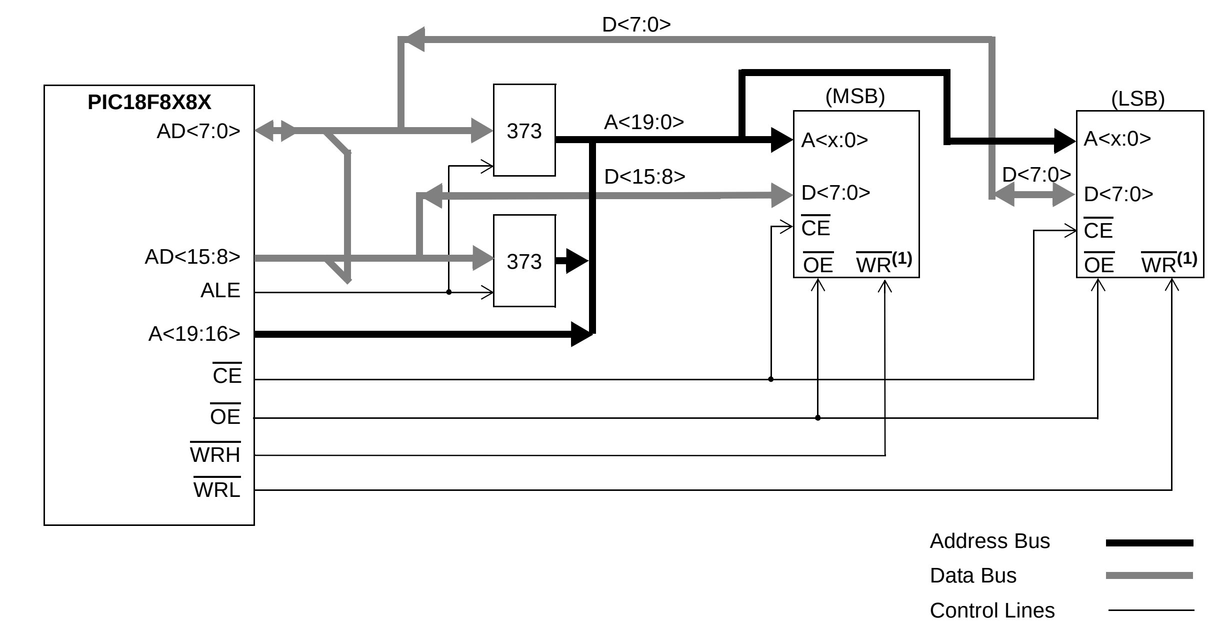
For all 16-bit modes, the Address Latch Enable (ALE) pin indicates that the Address bits (A<15:0>) are available on the external memory interface bus. Following the address latch, the Output Enable signal ($\overline{OE}$) will enable both bytes of program memory at once to form a 16-bit instruction word.

In Byte Select mode, JEDEC standard Flash memories will require BA0 for the byte address line, and one I/O line to select between Byte and Word mode. The other 16-bit modes do not need BA0. JEDEC standard static RAM memories will use the $\overline{UB}$ or $\overline{LB}$ signals for byte selection.

6.2.1 16-BIT BYTE WRITE MODE

Figure 6-1 shows an example of 16-bit Byte Write mode for PIC18F8X8X devices.

FIGURE 6-1: 16-BIT BYTE WRITE MODE EXAMPLE

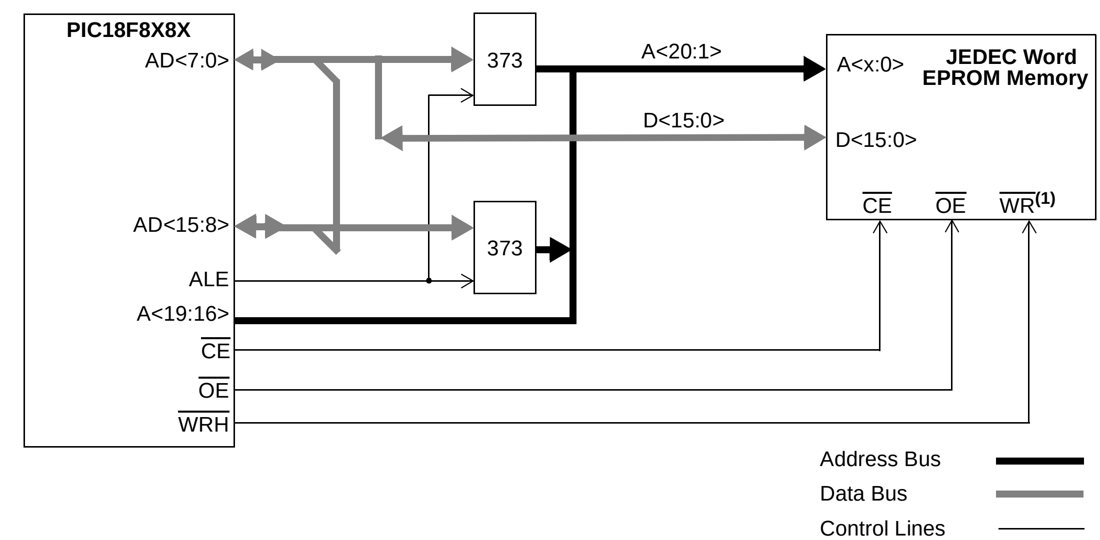


Note 1: This signal only applies to table writes. See **Section 5.1 "Table Reads and Table Writes"**.

6.2.2 16-BIT WORD WRITE MODE

Figure 6-2 shows an example of 16-bit Word Write mode for PIC18F8X8X devices.

FIGURE 6-2: 16-BIT WORD WRITE MODE EXAMPLE

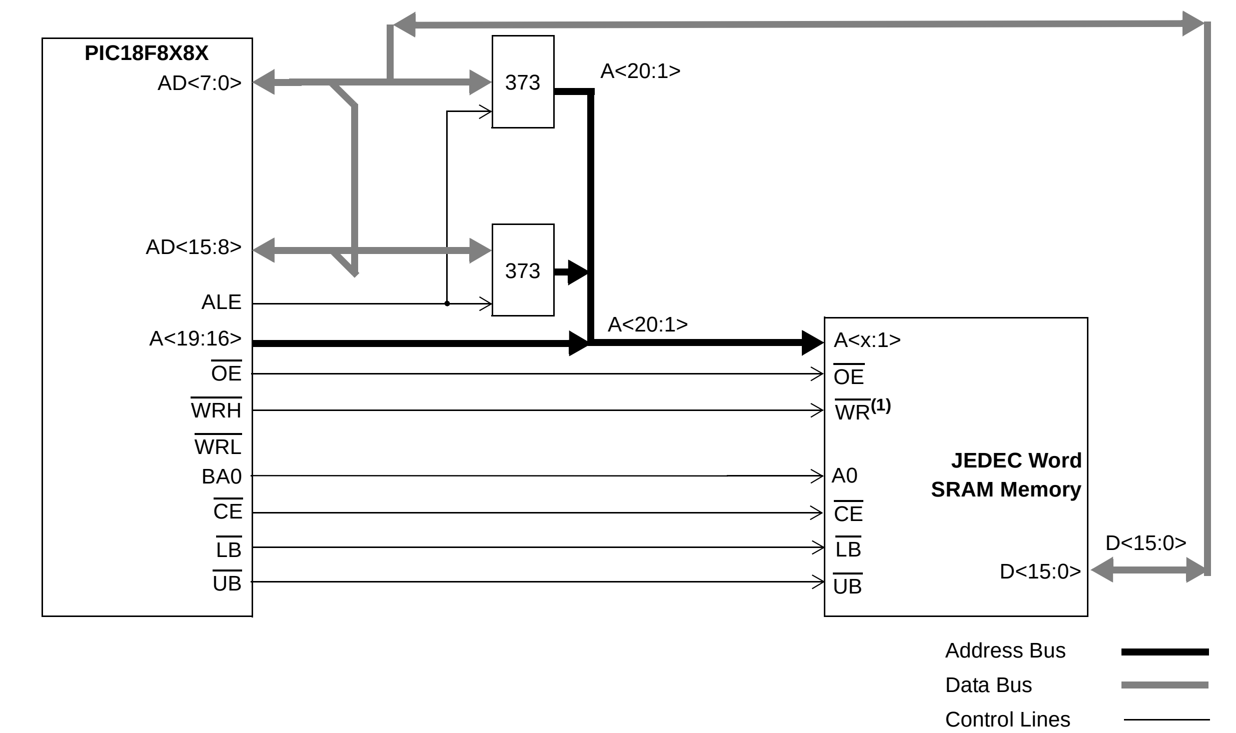


Note 1: This signal only applies to table writes. See **Section 5.1 "Table Reads and Table Writes"**.

6.2.3 16-BIT BYTE SELECT MODE

Figure 6-3 shows an example of 16-bit Byte Select mode for PIC18F8X8X devices.

FIGURE 6-3: 16-BIT BYTE SELECT MODE EXAMPLE

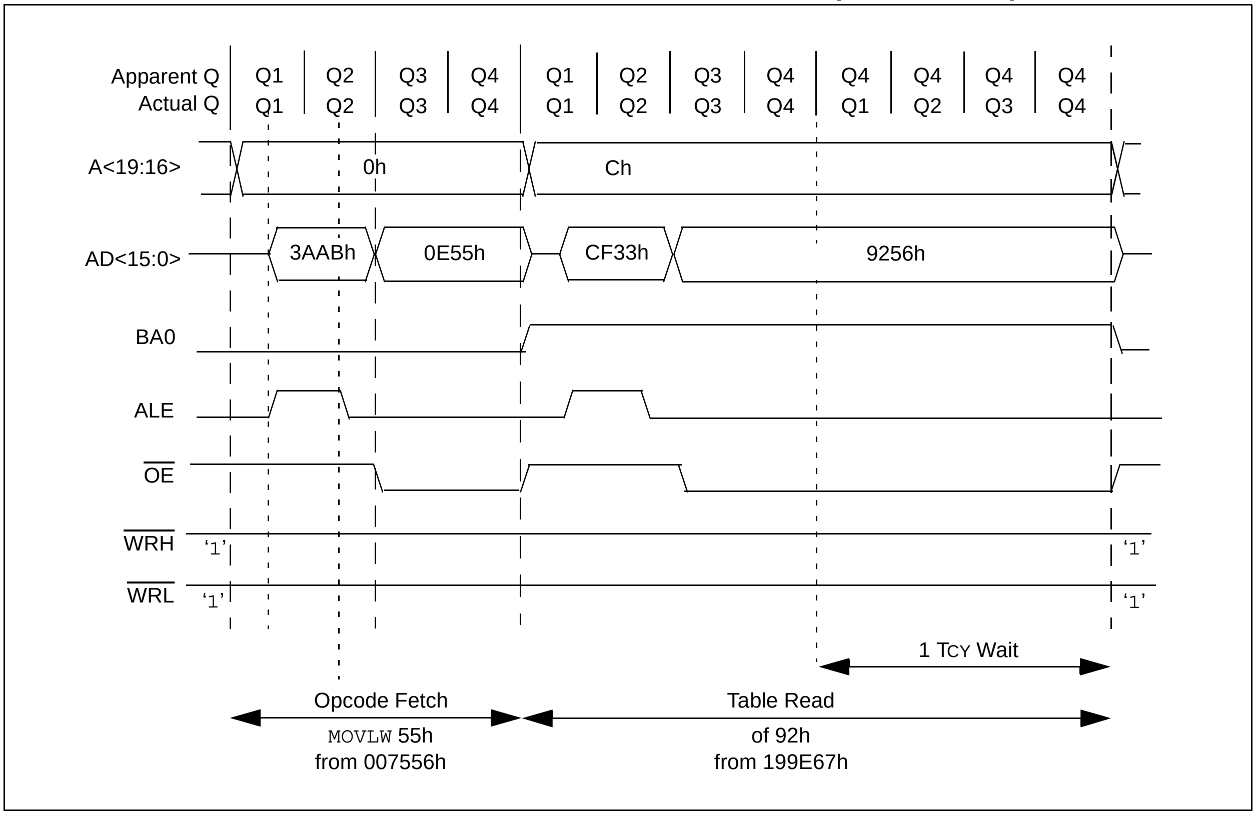


Note 1: This signal only applies to table writes. See **Section 5.1 "Table Reads and Table Writes"**.

6.2.4 16-BIT MODE TIMING

Figure 6-4 shows the 16-bit mode external bus timing for PIC18F8X8X devices.

FIGURE 6-4: EXTERNAL PROGRAM MEMORY BUS TIMING (16-BIT MODE)



NOTES:

7.0 DATA EEPROM MEMORY

The data EEPROM is readable and writable during normal operation over the entire VDD range. The data memory is not directly mapped in the register file space. Instead, it is indirectly addressed through the Special Function Registers (SFR).

There are five SFRs used to read and write the program and data EEPROM memory. These registers are:

- EECON1
- EECON2
- EEDATA
- EEADR
- EEADRH

The EEPROM data memory allows byte read and write. When interfacing to the data memory block, EEDATA holds the 8-bit data for read/write and EEADR holds the address of the EEPROM location being accessed. These devices have 1024 bytes of data EEPROM with an address range from 0h to 3FFh.

The EEPROM data memory is rated for high erase/write cycles. A byte write automatically erases the location and writes the new data (erase-before-write). The write time is controlled by an on-chip timer. The write time will vary with voltage and temperature as well as from chip to chip. Please refer to parameter D122 (Electrical Characteristics, **Section 27.0 "Electrical Characteristics"**) for exact limits.

7.1 EEADRH:EEADR

The address register pair, EEADRH:EEADR, can address up to a maximum of 1024 bytes of data EEPROM.

7.2 EECON1 and EECON2 Registers

EECON1 is the control register for EEPROM memory accesses.

EECON2 is not a physical register. Reading EECON2 will read all '0's. The EECON2 register is used exclusively in the EEPROM write sequence.

Control bits RD and WR initiate read and write operations, respectively. These bits cannot be cleared, only set in software. They are cleared in hardware at the completion of the read or write operation. The inability to clear the WR bit in software prevents the accidental or premature termination of a write operation.

The WREN bit, when set, will allow a write operation. On power-up, the WREN bit is clear. The WRERR bit is set when a write operation is interrupted by a MCLR Reset or a WDT Time-out Reset during normal operation. In these situations, the user can check the WRERR bit and rewrite the location. It is necessary to reload the data and address registers (EEDATA and EEADR) due to the Reset condition forcing the contents of the registers to zero.

Note:	Interrupt flag bit, EEIF in the PIR2 register, is set when write is complete. It must be cleared in software.

REGISTER 7-1: EECON1 REGISTER (ADDRESS FA6h)

R/W-x	R/W-x	U-0	R/W-0	R/W-x	R/W-0	R/S-0	R/S-0
EEPGD	CFGS	—	FREE	WRERR	WREN	WR	RD
bit 7							bit 0

bit 7 **EEPGD:** Flash Program or Data EEPROM Memory Select bit
1 = Access Flash program memory
0 = Access data EEPROM memory

bit 6 **CFGS:** Flash Program/Data EE or Configuration Select bit
1 = Access configuration or calibration registers
0 = Access Flash program or data EEPROM memory

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

bit 4 **FREE:** Flash Row Erase Enable bit
1 = Erase the program memory row addressed by TBLPTR on the next WR command (cleared by completion of erase operation)
0 = Perform write only

bit 3 **WRERR:** Flash Program/Data EE Error Flag bit
1 = A write operation is prematurely terminated (any $\overline{\text{MCLR}}$ or any WDT Reset during self-timed programming in normal operation)
0 = The write operation completed

Note: When a WRERR occurs, the EEPGD or FREE bits are not cleared. This allows tracing of the error condition.

bit 2 **WREN:** Flash Program/Data EE Write Enable bit
1 = Allows write cycles
0 = Inhibits write to the EEPROM

bit 1 **WR:** Write Control bit
1 = Initiates a data EEPROM erase/write cycle or a program memory erase cycle or write cycle. (The operation is self-timed and the bit is cleared by hardware once write is complete. The WR bit can only be set (not cleared) in software.)
0 = Write cycle to the EEPROM is complete

bit 0 **RD:** Read Control bit
1 = Initiates an EEPROM read. (Read takes one cycle. RD is cleared in hardware. The RD bit can only be set (not cleared) in software. RD bit cannot be set when EEPGD = 1.)
0 = Does not initiate an EEPROM read

Legend:		
R = Readable bit	U = Unimplemented bit, read as '0'	
W = Writable bit	S = Settable bit	- n = Value after erase
'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

7.3 Reading the Data EEPROM Memory

To read a data memory location, the user must write the address to the EEADR register, clear the EEPGD control bit (EECON1<7>), clear the CFGS control bit (EECON1<6>) and then set control bit, RD (EECON1<0>). The data is available for the very next instruction cycle; therefore, the EEDATA register can be read by the next instruction. EEDATA will hold this value until another read operation or until it is written to by the user (during a write operation).

EXAMPLE 7-1: DATA EEPROM READ

```
MOVLW   DATA_EE_ADR_HI      ;
MOVWF   EEADRH              ;
MOVLW   DATA_EE_ADDR_LOW    ;
MOVWF   EEADR               ; Data Memory Address to read
BCF     EECON1, EEPGD       ; Point to DATA memory
BCF     EECON1, CFGS        ; Access program Flash or Data EEPROM memory
BSF     EECON1, RD          ; EEPROM Read
MOVF    EEDATA, W           ; W = EEDATA
```

7.4 Writing to the Data EEPROM Memory

To write an EEPROM data location, the address must first be written to the EEADRH:EEADR register pair and the data written to the EEDATA register. Then the sequence in Example 7-2 must be followed to initiate the write cycle.

The write will not initiate if the above sequence is not exactly followed (write 55h to EECON2, write 0AAh to EECON2, then set WR bit) for each byte. It is strongly recommended that interrupts be disabled during this code segment.

Additionally, the WREN bit in EECON1 must be set to enable writes. This mechanism prevents accidental writes to data EEPROM due to unexpected code execution (i.e., runaway programs). The WREN bit should be kept clear at all times except when updating the EEPROM. The WREN bit is not cleared by hardware.

After a write sequence has been initiated, EECON1, EEADRH:EEADR and EDATA cannot be modified. The WR bit will be inhibited from being set unless the WREN bit is set. The WREN bit must be set on a previous instruction. Both WR and WREN cannot be set with the same instruction.

At the completion of the write cycle, the WR bit is cleared in hardware and the EEPROM Write Complete Interrupt Flag bit (EEIF) is set. The user may either enable this interrupt or poll this bit. EEIF must be cleared by software.

EXAMPLE 7-2: DATA EEPROM WRITE

```
           MOVLW   DATA_EE_ADDR_HI     ;
           MOVWF   EEADRH              ;
           MOVLW   DATA_EE_ADDR_LOW    ;
           MOVWF   EEADR               ; Data Memory Address to read
           MOVLW   DATA_EE_DATA        ;
           MOVWF   EEDATA              ; Data Memory Value to write
           BCF     EECON1, EEPGD       ; Point to DATA memory
           BCF     EECON1, CFGS        ; Access program Flash or Data EEPROM memory
           BSF     EECON1, WREN        ; Enable writes

           BCF     INTCON, GIE         ; Disable interrupts
Required   MOVLW   55h                 ;
Sequence   MOVWF   EECON2              ; Write 55h
           MOVLW   0AAh                ;
           MOVWF   EECON2              ; Write 0AAh
           BSF     EECON1, WR          ; Set WR bit to begin write
           BSF     INTCON, GIE         ; Enable interrupts

           .                           ; user code execution
           .
           .
           BCF     EECON1, WREN        ; Disable writes on write complete (EEIF set)
```

7.5 Write Verify

Depending on the application, good programming practice may dictate that the value written to the memory should be verified against the original value. This should be used in applications where excessive writes can stress bits near the specification limit.

7.6 Protection Against Spurious Write

There are conditions when the device may not want to write to the data EEPROM memory. To protect against spurious EEPROM writes, various mechanisms have been built-in. On power-up, the WREN bit is cleared. Also, the Power-up Timer (72 ms duration) prevents EEPROM write.

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

7.7 Operation During Code-Protect

Data EEPROM memory has its own code-protect mechanism. External read and write operations are disabled if either of these mechanisms are enabled.

The microcontroller itself can both read and write to the internal data EEPROM regardless of the state of the code-protect configuration bit. Refer to **Section 24.0 “Special Features of the CPU”** for additional information.

7.8 Using the Data EEPROM

The data EEPROM is a high endurance, byte addressable array that has been optimized for the storage of frequently changing information (e.g., program variables or other data that are updated often). Frequently changing values will typically be updated more often than specification D124. If this is not the case, an array refresh must be performed. For this reason, variables that change infrequently (such as constants, IDs, calibration, etc.) should be stored in Flash program memory.

A simple data EEPROM refresh routine is shown in Example 7-3.

Note:	If data EEPROM is only used to store constants and/or data that changes rarely, an array refresh is likely not required. See specification D124.

EXAMPLE 7-3: DATA EEPROM REFRESH ROUTINE

```
        CLRF    EEADRH                ;
        CLRF    EEADR                 ; Start at address 0
        BCF     EECON1, CFGS          ; Set for memory
        BCF     EECON1, EEPGD         ; Set for Data EEPROM
        BCF     INTCON, GIE           ; Disable interrupts
        BSF     EECON1, WREN          ; Enable writes
Loop                                  ; Loop to refresh array
        BSF     EECON1, RD            ; Read current address
        MOVLW   55h                   ;
        MOVWF   EECON2                ; Write 55h
        MOVLW   0AAh                  ;
        MOVWF   EECON2                ; Write 0AAh
        BSF     EECON1, WR            ; Set WR bit to begin write
        BTFSC   EECON1, WR            ; Wait for write to complete
        BRA     $-2
        INCFSZ  EEADR, F              ; Increment address
        BRA     Loop                  ; Not zero, do it again

        INCFS2  EEADRH, F             ;
        BRA     Loop                  ;
        BCF     EECON1, WREN          ; Disable writes
        BSF     INTCON, GIE           ; Enable interrupts
```

TABLE 7-1: REGISTERS ASSOCIATED WITH DATA EEPROM MEMORY

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
INTCON	GIE/ GIEH	PEIE/ GIEL	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u
EEADRH	—	—	—	—	—	—	EE Addr High		---- --00	---- --00
EEADR	EEPROM Address Register								0000 0000	0000 0000
EEDATA	EEPROM Data Register								0000 0000	0000 0000
EECON2	EEPROM Control Register 2 (not a physical register)								—	—
EECON1	EEPGD	CFGS	—	FREE	WRERR	WREN	WR	RD	xx-0 x000	uu-0 u000
IPR2	—	CMIP	—	EEIP	BCLIP	LVDIP	TMR3IP	CCP2IP	-1-1 1111	---1 1111
PIR2	—	CMIF	—	EEIF	BCLIF	LVDIF	TMR3IF	CCP2IF	-0-0 0000	---0 0000
PIE2	—	CMIE	—	EEIE	BCLIE	LVDIE	TMR3IE	CCP2IE	-0-0 0000	---0 0000

Legend: x = unknown, u = unchanged, r = reserved, - = unimplemented, read as '0'.
Shaded cells are not used during Flash/EEPROM access.

NOTES:

8.0 8 x 8 HARDWARE MULTIPLIER

8.1 Introduction

An 8 x 8 hardware multiplier is included in the ALU of the PIC18F6585/8585/6680/8680 devices. By making the multiply a hardware operation, it completes in a single instruction cycle. This is an unsigned multiply that gives a 16-bit result. The result is stored in the 16-bit product register pair (PRODH:PRODL). The multiplier does not affect any flags in the ALUSTA register.

Making the 8 x 8 multiplier execute in a single cycle gives the following advantages:

- Higher computational throughput
- Reduces code size requirements for multiply algorithms

The performance increase allows the device to be used in applications previously reserved for Digital Signal Processors.

Table 8-1 shows a performance comparison between enhanced devices using the single-cycle hardware multiply and performing the same function without the hardware multiply.

8.2 Operation

Example 8-1 shows the sequence to do an 8 x 8 unsigned multiply. Only one instruction is required when one argument of the multiply is already loaded in the WREG register.

Example 8-2 shows the sequence to do an 8 x 8 signed multiply. 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 8-1: 8 x 8 UNSIGNED MULTIPLY ROUTINE

```
MOVF    ARG1, W         ;
MULWF   ARG2            ; ARG1 * ARG2 ->
                        ; PRODH:PRODL
```

EXAMPLE 8-2: 8 x 8 SIGNED MULTIPLY ROUTINE

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

TABLE 8-1: PERFORMANCE COMPARISON

Routine	Multiply Method	Program Memory (Words)	Cycles (Max)	Time		
				@ 40 MHz	@ 10 MHz	@ 4 MHz
8 x 8 unsigned	Without hardware multiply	13	69	6.9 μs	27.6 μs	69 μs
	Hardware multiply	1	1	100 ns	400 ns	1 μs
8 x 8 signed	Without hardware multiply	33	91	9.1 μs	36.4 μs	91 μs
	Hardware multiply	6	6	600 ns	2.4 μs	6 μs
16 x 16 unsigned	Without hardware multiply	21	242	24.2 μs	96.8 μs	242 μs
	Hardware multiply	24	24	2.4 μs	9.6 μs	24 μs
16 x 16 signed	Without hardware multiply	52	254	25.4 μs	102.6 μs	254 μs
	Hardware multiply	36	36	3.6 μs	14.4 μs	36 μs

Example 8-3 shows the sequence to do a 16 x 16 unsigned multiply. Equation 8-1 shows the algorithm that is used. The 32-bit result is stored in four registers, RES3:RES0.

EQUATION 8-1: 16 x 16 UNSIGNED MULTIPLICATION ALGORITHM

RES3:RES0 = ARG1H:ARG1L • ARG2H:ARG2L
= (ARG1H • ARG2H • 216) +
(ARG1H • ARG2L • 28) +
(ARG1L • ARG2H • 28) +
(ARG1L • ARG2L)

EXAMPLE 8-3: 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          ; Add cross
    MOVF    PRODH, W      ; products
    ADDWFC  RES2          ;
    CLRF    WREG          ;
    ADDWFC  RES3          ;
;
    MOVF    ARG1H, W      ;
    MULWF   ARG2L         ; ARG1H * ARG2L ->
                          ; PRODH:PRODL
    MOVF    PRODL, W      ;
    ADDWF   RES1          ; Add cross
    MOVF    PRODH, W      ; products
    ADDWFC  RES2          ;
    CLRF    WREG          ;
    ADDWFC  RES3          ;
```

Example 8-4 shows the sequence to do a 16 x 16 signed multiply. Equation 8-2 shows the algorithm used. The 32-bit result is stored in four registers, RES3:RES0. To account for the sign bits of the arguments, each argument pairs' Most Significant bit (MSb) is tested and the appropriate subtractions are done.

EQUATION 8-2: 16 x 16 SIGNED MULTIPLICATION ALGORITHM

RES3:RES0
= ARG1H:ARG1L • ARG2H:ARG2L
= (ARG1H • ARG2H • 216) +
(ARG1H • ARG2L • 28) +
(ARG1L • ARG2H • 28) +
(ARG1L • ARG2L) +
(-1 • ARG2H<7> • ARG1H:ARG1L • 216) +
(-1 • ARG1H<7> • ARG2H:ARG2L • 216)

EXAMPLE 8-4: 16 x 16 SIGNED MULTIPLY ROUTINE

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

9.0 INTERRUPTS

The PIC18F6585/8585/6680/8680 devices have multiple interrupt sources and an interrupt priority feature that allows each interrupt source to be assigned a high or a low priority level. The high priority interrupt vector is at 000008h while the low priority interrupt vector is at 000018h. High priority interrupt events will override any low priority interrupts that may be in progress.

There are thirteen registers which are used to control interrupt operation. They are:

- RCON
- INTCON
- INTCON2
- INTCON3
- PIR1, PIR2, PIR3
- PIE1, PIE2, PIE3
- IPR1, IPR2, IPR3

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.

Each interrupt source (except INT0) has three bits to control its operation. The functions of these bits 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

The interrupt priority feature is enabled by setting the IPEN bit (RCON<7>). When interrupt priority is enabled, there are two bits which enable interrupts globally. Setting the GIEH bit (INTCON<7>) enables all interrupts that have the priority bit set. Setting the GIEL bit (INTCON<6>) enables all interrupts that have the priority bit cleared. When the interrupt flag, enable bit and appropriate global interrupt enable bit are set, the interrupt will vector immediately to address 000008h or 000018h depending on the priority level. Individual interrupts can be disabled through their corresponding enable bits.

When the IPEN bit is cleared (default state), the interrupt priority feature is disabled and interrupts are compatible with PICmicro® mid-range devices. In Compatibility mode, the interrupt priority bits for each source have no effect. INTCON<6> is the PEIE bit which enables/disables all peripheral interrupt sources. INTCON<7> is the GIE bit which enables/disables all interrupt sources. All interrupts branch to address 000008h in Compatibility mode.

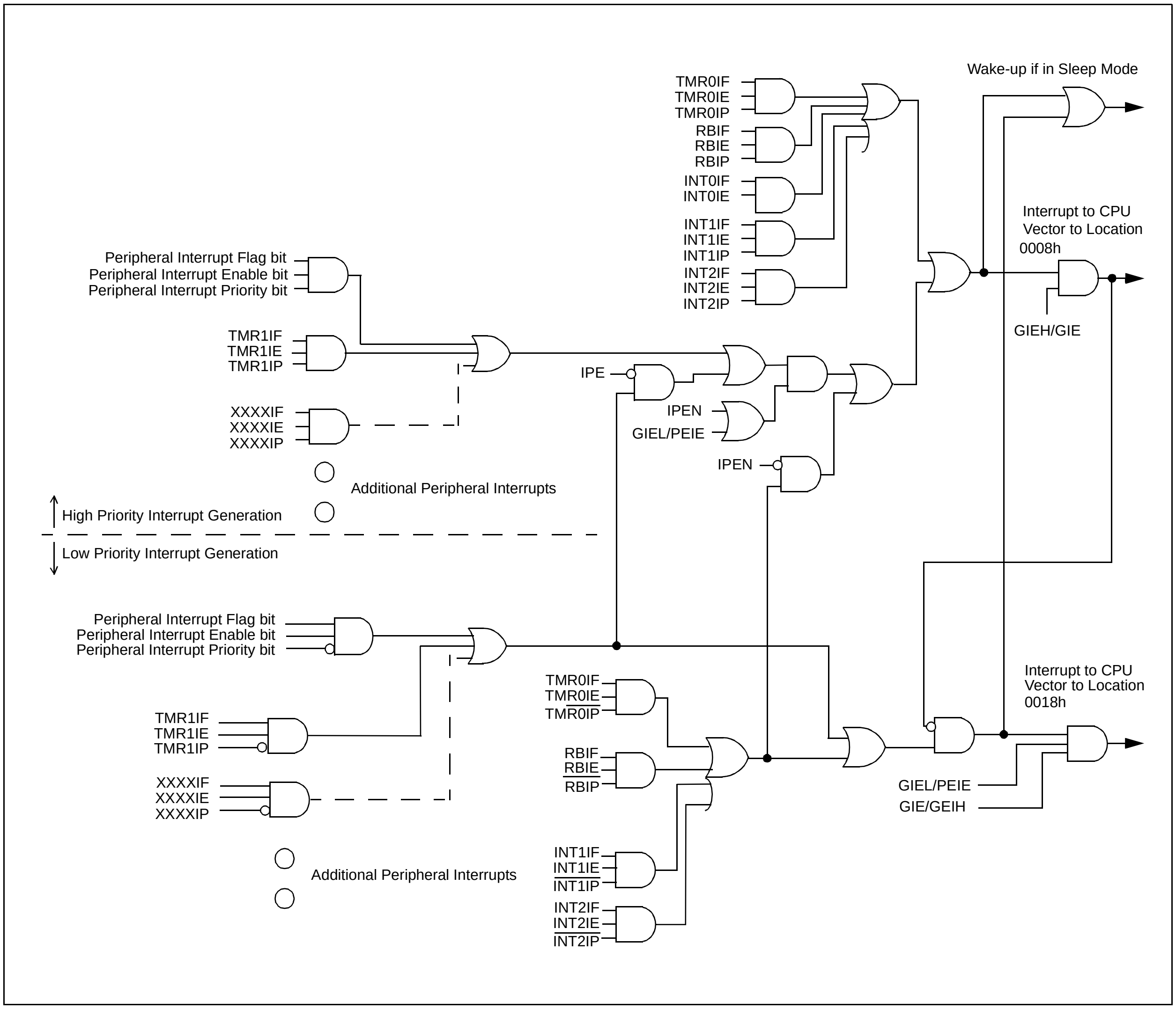
When an interrupt is responded to, the global interrupt enable bit is cleared to disable further interrupts. If the IPEN bit is cleared, this is the GIE bit. If interrupt priority levels are used, this will be either the GIEH or GIEL bit. High priority interrupt sources can interrupt a low priority interrupt.

The return address is pushed onto the stack and the PC is loaded with the interrupt vector address (000008h or 000018h). Once in the Interrupt Service Routine, the source(s) of the interrupt can be determined by polling the interrupt flag bits. The interrupt flag bits must be cleared in software before re-enabling interrupts to avoid recursive interrupts.

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

For external interrupt events, such as the INT pins or the PORTB input change interrupt, the interrupt latency will be three to four instruction cycles. The exact latency is the same for one- or two-cycle instructions. Individual interrupt flag bits are set regardless of the status of their corresponding enable bit or the GIE bit.

FIGURE 9-1: INTERRUPT LOGIC



9.1 INTCON Registers

The INTCON registers are readable and writable registers which contain various enable, priority and flag bits.

Note: Interrupt flag bits are set when an interrupt condition occurs regardless of the state of its corresponding enable bit or the global enable bit. User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt. This feature allows for software polling.

REGISTER 9-1: INTCON REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF
bit 7							bit 0

bit 7 **GIE/GIEH:** Global Interrupt Enable bit
When IPEN (RCON<7>) = 0:
1 = Enables all unmasked interrupts
0 = Disables all interrupts
When IPEN (RCON<7>) = 1:
1 = Enables all high priority interrupts
0 = Disables all interrupts

bit 6 **PEIE/GIEL:** Peripheral Interrupt Enable bit
When IPEN (RCON<7>) = 0:
1 = Enables all unmasked peripheral interrupts
0 = Disables all peripheral interrupts
When IPEN (RCON<7>) = 1:
1 = Enables all low priority peripheral interrupts
0 = Disables all low priority peripheral interrupts

bit 5 **TMR0IE:** TMR0 Overflow Interrupt Enable bit
1 = Enables the TMR0 overflow interrupt
0 = Disables the TMR0 overflow interrupt

bit 4 **INT0IE:** INT0 External Interrupt Enable bit
1 = Enables the INT0 external interrupt
0 = Disables the INT0 external interrupt

bit 3 **RBIE:** RB Port Change Interrupt Enable bit
1 = Enables the RB port change interrupt
0 = Disables the RB port change interrupt

bit 2 **TMR0IF:** TMR0 Overflow Interrupt Flag bit
1 = TMR0 register has overflowed (must be cleared in software)
0 = TMR0 register did not overflow

bit 1 **INT0IF:** INT0 External Interrupt Flag bit
1 = The INT0 external interrupt occurred (must be cleared in software)
0 = The INT0 external interrupt did not occur

bit 0 **RBIF:** RB Port Change Interrupt Flag bit
1 = At least one of the RB7:RB4 pins changed state (must be cleared in software)
0 = None of the RB7:RB4 pins have changed state

Note: A mismatch condition will continue to set this bit. Reading PORTB will end the mismatch condition and allow the bit to be cleared.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
- n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

REGISTER 9-2: INTCON2 REGISTER

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
$\overline{\text{RBPU}}$	INTEDG0	INTEDG1	INTEDG2	INTEDG3	TMR0IP	INT3IP	RBIP
bit 7							bit 0

bit 7 **$\overline{\text{RBPU}}$**: PORTB Pull-up Enable bit
1 = All PORTB pull-ups are disabled
0 = PORTB pull-ups are enabled by individual port latch values

bit 6 **INTEDG0**: External Interrupt 0 Edge Select bit
1 = Interrupt on rising edge
0 = Interrupt on falling edge

bit 5 **INTEDG1**: External Interrupt 1 Edge Select bit
1 = Interrupt on rising edge
0 = Interrupt on falling edge

bit 4 **INTEDG2**: External Interrupt 2 Edge Select bit
1 = Interrupt on rising edge
0 = Interrupt on falling edge

bit 3 **INTEDG3**: External Interrupt 3 Edge Select bit
1 = Interrupt on rising edge
0 = Interrupt on falling edge

bit 2 **TMR0IP**: TMR0 Overflow Interrupt Priority bit
1 = High priority
0 = Low priority

bit 1 **INT3IP:** INT3 External Interrupt Priority bit
1 = High priority
0 = Low priority

bit 0 **RBIP**: RB Port Change Interrupt Priority bit
1 = High priority
0 = Low priority

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
- n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

Note:	Interrupt flag bits are set when an interrupt condition occurs regardless of the state of its corresponding enable bit or the global enable bit. User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt. This feature allows for software polling.

REGISTER 9-3: INTCON3 REGISTER

R/W-1	R/W-1	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
INT2IP	INT1IP	INT3IE	INT2IE	INT1IE	INT3IF	INT2IF	INT1IF
bit 7							bit 0

bit 7 **INT2IP:** INT2 External Interrupt Priority bit
1 = High priority
0 = Low priority

bit 6 **INT1IP:** INT1 External Interrupt Priority bit
1 = High priority
0 = Low priority

bit 5 **INT3IE:** INT3 External Interrupt Enable bit
1 = Enables the INT3 external interrupt
0 = Disables the INT3 external interrupt

bit 4 **INT2IE:** INT2 External Interrupt Enable bit
1 = Enables the INT2 external interrupt
0 = Disables the INT2 external interrupt

bit 3 **INT1IE:** INT1 External Interrupt Enable bit
1 = Enables the INT1 external interrupt
0 = Disables the INT1 external interrupt

bit 2 **INT3IF:** INT3 External Interrupt Flag bit
1 = The INT3 external interrupt occurred (must be cleared in software)
0 = The INT3 external interrupt did not occur

bit 1 **INT2IF:** INT2 External Interrupt Flag bit
1 = The INT2 external interrupt occurred (must be cleared in software)
0 = The INT2 external interrupt did not occur

bit 0 **INT1IF:** INT1 External Interrupt Flag bit
1 = The INT1 external interrupt occurred (must be cleared in software)
0 = The INT1 external interrupt did not occur

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
- n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

Note: Interrupt flag bits are set when an interrupt condition occurs regardless of the state of its corresponding enable bit or the global enable bit. User software should ensure the appropriate interrupt flag bits are clear prior to enabling an interrupt. This feature allows for software polling.

9.2 PIR Registers

The PIR registers contain the individual flag bits for the peripheral interrupts. Due to the number of peripheral interrupt sources, there are three Peripheral Interrupt Flag registers (PIR1, PIR2 and PIR3).

> **Note 1:** Interrupt flag bits are set when an interrupt condition occurs regardless of the state of its corresponding enable bit or the global enable bit, GIE (INTCON<7>).
>
> **2:** User software should ensure the appropriate interrupt flag bits are cleared prior to enabling an interrupt, and after servicing that interrupt.

REGISTER 9-4: PIR1: PERIPHERAL INTERRUPT REQUEST (FLAG) REGISTER 1

R/W-0	R/W-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
PSPIF[(1)]	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF
bit 7							bit 0

bit 7 **PSPIF:** Parallel Slave Port Read/Write Interrupt Flag bit[(1)]
1 = A read or a write operation has taken place (must be cleared in software)
0 = No read or write has occurred

bit 6 **ADIF**: A/D Converter Interrupt Flag bit
1 = An A/D conversion completed (must be cleared in software)
0 = The A/D conversion is not complete

bit 5 **RCIF**: USART Receive Interrupt Flag bit
1 = The USART receive buffer, RCREG, is full (cleared when RCREG is read)
0 = The USART receive buffer is empty

bit 4 **TXIF**: USART Transmit Interrupt Flag bit
1 = The USART transmit buffer, TXREG, is empty (cleared when TXREG is written)
0 = The USART transmit buffer is full

bit 3 **SSPIF**: Master Synchronous Serial Port Interrupt Flag bit
1 = The transmission/reception is complete (must be cleared in software)
0 = Waiting to transmit/receive

bit 2 **CCP1IF**: Enhanced CCP1 Interrupt Flag bit
Capture mode:
1 = A TMR1 register capture occurred (must be cleared in software)
0 = No TMR1 register capture occurred
Compare mode:
1 = A TMR1 register compare match occurred (must be cleared in software)
0 = No TMR1 register compare match occurred
PWM mode:
Unused in this mode.

bit 1 **TMR2IF:** TMR2 to PR2 Match Interrupt Flag bit
1 = TMR2 to PR2 match occurred (must be cleared in software)
0 = No TMR2 to PR2 match occurred

bit 0 **TMR1IF:** TMR1 Overflow Interrupt Flag bit
1 = TMR1 register overflowed (must be cleared in software)
0 = TMR1 register did not overflow

Note 1: Available in Microcontroller mode only.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
- n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

REGISTER 9-5: PIR2: PERIPHERAL INTERRUPT REQUEST (FLAG) REGISTER 2

U-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	CMIF	—	EEIF	BCLIF	LVDIF	TMR3IF	CCP2IF
bit 7							bit 0

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

bit 6 **CMIF**: Comparator Interrupt Flag bit
1 = The comparator input has changed (must be cleared in software)
0 = The comparator input has not changed

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

bit 4 **EEIF**: Data EEPROM/Flash Write Operation Interrupt Flag bit
1 = The write operation is complete (must be cleared in software)
0 = The write operation is not complete, or has not been started

bit 3 **BCLIF**: Bus Collision Interrupt Flag bit
1 = A bus collision occurred while the SSP module (configured in I^2C Master mode) was transmitting (must be cleared in software)
0 = No bus collision occurred

bit 2 **LVDIF**: Low-Voltage Detect Interrupt Flag bit
1 = A low-voltage condition occurred (must be cleared in software)
0 = The device voltage is above the Low-Voltage Detect trip point

bit 1 **TMR3IF**: TMR3 Overflow Interrupt Flag bit
1 = TMR3 register overflowed (must be cleared in software)
0 = TMR3 register did not overflow

bit 0 **CCP2IF**: CCP2 Interrupt Flag bit
Capture mode:
1 = A TMR1 or TMR3 register capture occurred (must be cleared in software)
0 = No TMR1 or TMR3 register capture occurred
Compare mode:
1 = A TMR1 or TMR3 register compare match occurred (must be cleared in software)
0 = No TMR1 or TMR3 register compare match occurred
PWM mode:
Unused in this mode.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

REGISTER 9-6: PIR3: PERIPHERAL INTERRUPT REQUEST (FLAG) REGISTER 3

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
IRXIF	WAKIF	ERRIF	TXB2IF/ TXBnIF	TXB1IF(1)	TXB0IF(1)	RXB1IF/ RXBnIF	RXB0IF/ FIFOWMIF
bit 7							bit 0

bit 7 **IRXIF:** CAN Invalid Received Message Interrupt Flag bit
1 = An invalid message has occurred on the CAN bus
0 = No invalid message on CAN bus

bit 6 **WAKIF:** CAN bus Activity Wake-up Interrupt Flag bit
1 = Activity on CAN bus has occurred
0 = No activity on CAN bus

bit 5 **ERRIF:** CAN bus Error Interrupt Flag bit
1 = An error has occurred in the CAN module (multiple sources)
0 = No CAN module errors

bit 4 When CAN is in Mode 0:
TXB2IF: CAN Transmit Buffer 2 Interrupt Flag bit
1 = Transmit Buffer 2 has completed transmission of a message and may be reloaded
0 = Transmit Buffer 2 has not completed transmission of a message
When CAN is in Mode 1 or 2:
TXBnIF: Any Transmit Buffer Interrupt Flag bit
1 = One or more transmit buffers has completed transmission of a message and may be reloaded (TXBIE or BIE0<7:2> must be non-zero)
0 = No message was transmitted

bit 3 **TXB1IF:** CAN Transmit Buffer 1 Interrupt Flag bit(1)
1 = Transmit Buffer 1 has completed transmission of a message and may be reloaded
0 = Transmit Buffer 1 has not completed transmission of a message

bit 2 **TXB0IF:** CAN Transmit Buffer 0 Interrupt Flag bit(1)
1 = Transmit Buffer 0 has completed transmission of a message and may be reloaded
0 = Transmit Buffer 0 has not completed transmission of a message

bit 1 When CAN is in Mode 0:
RXB1IF: CAN Receive Buffer 1 Interrupt Flag bit
1 = Receive Buffer 1 has received a new message
0 = Receive Buffer 1 has not received a new message
When CAN is in Mode 1 or 2:
RXBnIF: CAN Receive Buffer Interrupt Flag bit
1 = One or more receive buffers has received a new message
0 = No receive buffer has received a new message

bit 0 When CAN is in Mode 0:
RXB0IF: CAN Receive Buffer 0 Interrupt Flag bit(1)
1 = Receive Buffer 0 has received a new message
0 = Receive Buffer 0 has not received a new message
When CAN is in Mode 1:
Unimplemented: Read as '0'
When CAN is in Mode 2:
FIFOWMIF: FIFO Watermark Interrupt Flag bit
1 = FIFO high watermark is reached
0 = FIFO high watermark is not reached

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

9.3 PIE Registers

The PIE registers contain the individual enable bits for the peripheral interrupts. Due to the number of peripheral interrupt sources, there are three Peripheral Interrupt Enable registers (PIE1, PIE2 and PIE3). When the IPEN bit (RCON<7>) is '0', the PEIE bit must be set to enable any of these peripheral interrupts.

REGISTER 9-7: PIE1: PERIPHERAL INTERRUPT ENABLE REGISTER 1

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PSPIE[(1)]	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE
bit 7							bit 0

bit 7 **PSPIE:** Parallel Slave Port Read/Write Interrupt Enable bit[(1)]
1 = Enables the PSP read/write interrupt
0 = Disables the PSP read/write interrupt

Note 1: Available in Microcontroller mode only.

bit 6 **ADIE**: A/D Converter Interrupt Enable bit
1 = Enables the A/D interrupt
0 = Disables the A/D interrupt

bit 5 **RCIE**: USART Receive Interrupt Enable bit
1 = Enables the USART receive interrupt
0 = Disables the USART receive interrupt

bit 4 **TXIE**: USART Transmit Interrupt Enable bit
1 = Enables the USART transmit interrupt
0 = Disables the USART transmit interrupt

bit 3 **SSPIE**: Master Synchronous Serial Port Interrupt Enable bit
1 = Enables the MSSP interrupt
0 = Disables the MSSP interrupt

bit 2 **CCP1IE**: Enhanced CCP1 Interrupt Enable bit
1 = Enables the CCP1 interrupt
0 = Disables the CCP1 interrupt

bit 1 **TMR2IE**: TMR2 to PR2 Match Interrupt Enable bit
1 = Enables the TMR2 to PR2 match interrupt
0 = Disables the TMR2 to PR2 match interrupt

bit 0 **TMR1IE**: TMR1 Overflow Interrupt Enable bit
1 = Enables the TMR1 overflow interrupt
0 = Disables the TMR1 overflow interrupt

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

REGISTER 9-8: PIE2: PERIPHERAL INTERRUPT ENABLE REGISTER 2

U-0	R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	CMIE	—	EEIE	BCLIE	LVDIE	TMR3IE	CCP2IE
bit 7							bit 0

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

bit 6 **CMIE**: Comparator Interrupt Enable bit
1 = Enables the comparator interrupt
0 = Disables the comparator interrupt

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

bit 4 **EEIE**: Data EEPROM/Flash Write Operation Interrupt Enable bit
1 = Enables the write operation interrupt
0 = Disables the write operation interrupt

bit 3 **BCLIE**: Bus Collision Interrupt Enable bit
1 = Enables the bus collision interrupt
0 = Disables the bus collision interrupt

bit 2 **LVDIE**: Low-Voltage Detect Interrupt Enable bit
1 = Enables the Low-Voltage Detect interrupt
0 = Disables the Low-Voltage Detect interrupt

bit 1 **TMR3IE**: TMR3 Overflow Interrupt Enable bit
1 = Enables the TMR3 overflow interrupt
0 = Disables the TMR3 overflow interrupt

bit 0 **CCP2IE**: CCP2 Interrupt Enable bit
1 = Enables the CCP2 interrupt
0 = Disables the CCP2 interrupt

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

REGISTER 9-9: PIE3: PERIPHERAL INTERRUPT ENABLE REGISTER 3

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
IRXIE	WAKIE	ERRIE	TXB2IE/ TXBnIE	TXB1IE[(1)]	TXB0IE[(1)]	RXB1IE/ RXBnIE	RXB0IE/ FIFOWMIE
bit 7							bit 0

bit 7 **IRXIE:** CAN Invalid Received Message Interrupt Enable bit
1 = Enable invalid message received interrupt
0 = Disable invalid message received interrupt

bit 6 **WAKIE:** CAN bus Activity Wake-up Interrupt Enable bit
1 = Enable bus activity wake-up interrupt
0 = Disable bus activity wake-up interrupt

bit 5 **ERRIE:** CAN bus Error Interrupt Enable bit
1 = Enable CAN bus error interrupt
0 = Disable CAN bus error interrupt

bit 4 When CAN is in Mode 0:
TXB2IE: CAN Transmit Buffer 2 Interrupt Enable bit
1 = Enable Transmit Buffer 2 interrupt
0 = Disable Transmit Buffer 2 interrupt
When CAN is in Mode 1 or 2:
TXBnIE: CAN Transmit Buffer Interrupts Enable bit
1 = Enable transmit buffer interrupt; individual interrupt is enabled by TXBIE and BIE0
0 = Disable all transmit buffer interrupts

bit 3 **TXB1IE:** CAN Transmit Buffer 1 Interrupt Enable bit[(1)]
1 = Enable Transmit Buffer 1 interrupt
0 = Disable Transmit Buffer 1 interrupt

bit 2 **TXB0IE:** CAN Transmit Buffer 0 Interrupt Enable bit[(1)]
1 = Enable Transmit Buffer 0 interrupt
0 = Disable Transmit Buffer 0 interrupt

bit 1 When CAN is in Mode 0:
RXB1IE: CAN Receive Buffer 1 Interrupt Enable bit
1 = Enable Receive Buffer 1 interrupt
0 = Disable Receive Buffer 1 interrupt
When CAN is in Mode 1 or 2:
RXBnIE: CAN Receive Buffer Interrupts Enable bit
1 = Enable receive buffer interrupt; individual interrupt is enabled by BIE0
0 = Disable all receive buffer interrupts

bit 0 When CAN is in Mode 0:
RXB0IE: CAN Receive Buffer 0 Interrupt Enable bit
1 = Enable Receive Buffer 0 interrupt
0 = Disable Receive Buffer 0 interrupt
When CAN is in Mode 1:
Unimplemented: Read as '0'
When CAN is in Mode 2:
FIFOWMIE: FIFO Watermark Interrupt Enable bit
1 = Enable FIFO watermark interrupt
0 = Disable FIFO watermark interrupt

Note 1: In CAN Mode 1 and 2, this bit is forced to '0'.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

9.4 IPR Registers

The IPR registers contain the individual priority bits for the peripheral interrupts. Due to the number of peripheral interrupt sources, there are three Peripheral Interrupt Priority registers (IPR1, IPR2 and IPR3). The operation of the priority bits requires that the Interrupt Priority Enable (IPEN) bit be set.

REGISTER 9-10: IPR1: PERIPHERAL INTERRUPT PRIORITY REGISTER 1

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
PSPIP(1)	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP
bit 7							bit 0

bit 7 **PSPIP:** Parallel Slave Port Read/Write Interrupt Priority bit(1)
1 = High priority
0 = Low priority

Note 1: Available in Microcontroller mode only.

bit 6 **ADIP**: A/D Converter Interrupt Priority bit
1 = High priority
0 = Low priority

bit 5 **RCIP**: USART Receive Interrupt Priority bit
1 = High priority
0 = Low priority

bit 4 **TXIP**: USART Transmit Interrupt Priority bit
1 = High priority
0 = Low priority

bit 3 **SSPIP**: Master Synchronous Serial Port Interrupt Priority bit
1 = High priority
0 = Low priority

bit 2 **CCP1IP**: CCP1 Interrupt Priority bit
1 = High priority
0 = Low priority

bit 1 **TMR2IP**: TMR2 to PR2 Match Interrupt Priority bit
1 = High priority
0 = Low priority

bit 0 **TMR1IP**: TMR1 Overflow Interrupt Priority bit
1 = High priority
0 = Low priority

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
- n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

REGISTER 9-11: IPR2: PERIPHERAL INTERRUPT PRIORITY REGISTER 2

U-0	R/W-1	U-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
—	CMIP	—	EEIP	BCLIP	LVDIP	TMR3IP	CCP2IP
bit 7							bit 0

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

bit 6 **CMIP**: Comparator Interrupt Priority bit
1 = High priority
0 = Low priority

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

bit 4 **EEIP**: Data EEPROM/Flash Write Operation Interrupt Priority bit
1 = High priority
0 = Low priority

bit 3 **BCLIP**: Bus Collision Interrupt Priority bit
1 = High priority
0 = Low priority

bit 2 **LVDIP**: Low-Voltage Detect Interrupt Priority bit
1 = High priority
0 = Low priority

bit 1 **TMR3IP**: TMR3 Overflow Interrupt Priority bit
1 = High priority
0 = Low priority

bit 0 **CCP2IP**: CCP2 Interrupt Priority bit
1 = High priority
0 = Low priority

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

REGISTER 9-12: IPR3: PERIPHERAL INTERRUPT PRIORITY REGISTER 3

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
IRXIP	WAKIP	ERRIP	TXB2IP/ TXBnIP	TXB1IP(1)	TXB0IP(1)	RXB1IP/ RXBnIP	RXB0IP/ FIFOWMIP
bit 7							bit 0

bit 7 **IRXIP:** CAN Invalid Received Message Interrupt Priority bit
1 = High priority
0 = Low priority

bit 6 **WAKIP:** CAN bus Activity Wake-up Interrupt Priority bit
1 = High priority
0 = Low priority

bit 5 **ERRIP:** CAN bus Error Interrupt Priority bit
1 = High priority
0 = Low priority

bit 4 When CAN is in Mode 0:
TXB2IP: CAN Transmit Buffer 2 Interrupt Priority bit
1 = High priority
0 = Low priority
When CAN is in Mode 1 or 2:
TXBnIP: CAN Transmit Buffer Interrupt Priority bit
1 = High priority
0 = Low priority

bit 3 **TXB1IP:** CAN Transmit Buffer 1 Interrupt Priority bit(1)
1 = High priority
0 = Low priority

bit 2 **TXB0IP:** CAN Transmit Buffer 0 Interrupt Priority bit(1)
1 = High priority
0 = Low priority

bit 1 When CAN is in Mode 0:
RXB1IP: CAN Receive Buffer 1 Interrupt Priority bit
1 = High priority
0 = Low priority
When CAN is in Mode 1 or 2:
RXBnIP: CAN Receive Buffer Interrupts Priority bit
1 = High priority
0 = Low priority

bit 0 When CAN is in Mode 0:
RXB0IP: CAN Receive Buffer 0 Interrupt Priority bit
1 = High priority
0 = Low priority
When CAN is in Mode 1:
Unimplemented: Read as '0'
When CAN is in Mode 2:
FIFOWMIP: FIFO Watermark Interrupt Priority bit
1 = High priority
0 = Low priority

Note 1: In CAN Mode 1 and 2, this bit is forced to '0'.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

9.5 RCON Register

The RCON register contains the IPEN bit which is used to enable prioritized interrupts. The functions of the other bits in this register are discussed in more detail in **Section 4.14 “RCON Register”**.

REGISTER 9-13: RCON REGISTER

R/W-0	U-0	U-0	R/W-1	R-1	R-1	R/W-0	R/W-0
IPEN	—	—	$\overline{RI}$	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	$\overline{BOR}$
bit 7							bit 0

bit 7 **IPEN:** Interrupt Priority Enable bit
1 = Enable priority levels on interrupts
0 = Disable priority levels on interrupts (PIC16 Compatibility mode)

bit 6-5 **Unimplemented:** Read as ‘0’

bit 4 **$\overline{RI}$:** RESET Instruction Flag bit
For details of bit operation, see Register 4-4.

bit 3 **$\overline{TO}$:** Watchdog Time-out Flag bit
For details of bit operation, see Register 4-4.

bit 2 **$\overline{PD}$:** Power-down Detection Flag bit
For details of bit operation, see Register 4-4.

bit 1 **$\overline{POR}$:** Power-on Reset Status bit
For details of bit operation, see Register 4-4.

bit 0 **$\overline{BOR}$:** Brown-out Reset Status bit
For details of bit operation, see Register 4-4.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
- n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

9.6 INT0 Interrupt

External interrupts on the RB0/INT0, RB1/INT1, RB2/INT2 and RB3/INT3 pins are edge-triggered: either rising if the corresponding INTEDGx bit is set in the INTCON2 register, or falling if the INTEDGx bit is clear. When a valid edge appears on the RBx/INTx pin, the corresponding flag bit, INTxF, is set. This interrupt can be disabled by clearing the corresponding enable bit, INTxE. Flag bit, INTxF, must be cleared in software in the Interrupt Service Routine before re-enabling the interrupt. All external interrupts (INT0, INT1, INT2 and INT3) can wake-up the processor from Sleep if bit INTxIE was set prior to going into Sleep. If the global interrupt enable bit GIE is set, the processor will branch to the interrupt vector following wake-up.

The interrupt priority for INT, INT2 and INT3 is determined by the value contained in the interrupt priority bits: INT1IP (INTCON3<6>), INT2IP (INTCON3<7>) and INT3IP (INTCON2<1>). There is no priority bit associated with INT0; it is always a high priority interrupt source.

9.7 TMR0 Interrupt

In 8-bit mode (which is the default), an overflow in the TMR0 register (0FFh → 00h) will set flag bit TMR0IF. In 16-bit mode, an overflow in the TMR0H:TMR0L registers (0FFFFh → 0000h) will set flag bit, TMR0IF. The interrupt can be enabled/disabled by setting/clearing enable bit, TMR0IE (INTCON<5>). Interrupt priority for Timer0 is determined by the value contained in the interrupt priority bit, TMR0IP (INTCON2<2>). See **Section 11.0 “Timer0 Module”** for further details on the Timer0 module.

9.8 PORTB Interrupt-on-Change

An input change on PORTB<7:4> sets flag bit RBIF (INTCON<0>). The interrupt can be enabled/disabled by setting/clearing enable bit, RBIE (INTCON<3>). Interrupt priority for PORTB interrupt-on-change is determined by the value contained in the interrupt priority bit, RBIP (INTCON2<0>).

9.9 Context Saving During Interrupts

During an interrupt, the return PC value is saved on the stack. Additionally, the WREG, Status and BSR registers are saved on the fast return stack. If a fast return from interrupt is not used (See **Section 4.3 “Fast Register Stack”**), the user may need to save the WREG, Status and BSR registers in software. Depending on the user’s application, other registers may also need to be saved. Example 9-1 saves and restores the WREG, Status and BSR registers during an Interrupt Service Routine.

EXAMPLE 9-1: 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 located anywhere
;
; USER ISR CODE
;
MOVFF    BSR_TEMP, BSR             ; Restore BSR
MOVF     W_TEMP, W                 ; Restore WREG
MOVFF    STATUS_TEMP, STATUS       ; Restore STATUS
```

10.0 I/O PORTS

Depending on the device selected, there are either seven or nine I/O ports available on PIC18F6X8X/8X8X devices. Some of their pins are multiplexed with one or more alternate functions from the other peripheral features on the device. In general, when a peripheral is enabled, that pin may not be used as a general purpose I/O pin.

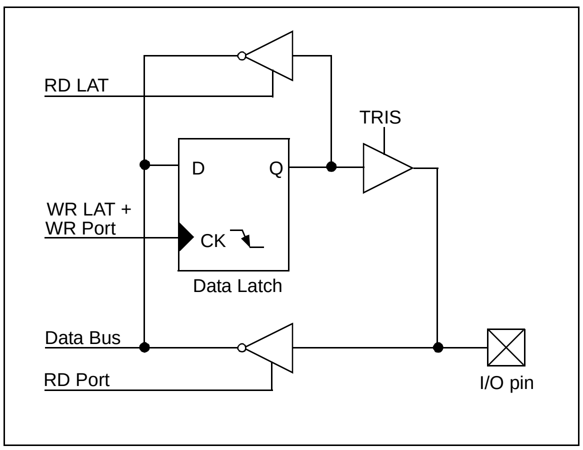
Each port has three registers for its operation. These registers are:

- TRIS register (data direction register)
- PORT register (reads the levels on the pins of the device)
- LAT register (output latch)

The Data Latch register (LAT) is useful for read-modify-write operations on the value that the I/O pins are driving.

A simplified version of a generic I/O port and its operation is shown in Figure 10-1.

FIGURE 10-1: SIMPLIFIED BLOCK DIAGRAM OF PORT/LAT/TRIS OPERATION



10.1 PORTA, TRISA and LATA Registers

PORTA is a 7-bit wide, bidirectional port. The corresponding data direction register is TRISA. Setting a TRISA bit (= `1`) will make the corresponding PORTA pin an input (i.e., put the corresponding output driver in a high-impedance mode). Clearing a TRISA bit (= `0`) will make the corresponding PORTA pin an output (i.e., put the contents of the output latch on the selected pin).

Reading the PORTA register reads the status of the pins, whereas writing to it will write to the port latch.

The Data Latch register (LATA) is also memory mapped. Read-modify-write operations on the LATA register read and write the latched output value for PORTA.

The RA4 pin is multiplexed with the Timer0 module clock input to become the RA4/T0CKI pin. The RA4/T0CKI pin is a Schmitt Trigger input and an open-drain output. All other RA port pins have TTL input levels and full CMOS output drivers.

The RA6 pin is only enabled as a general I/O pin in ECIO and RCIO Oscillator modes.

The other PORTA pins are multiplexed with analog inputs and the analog VREF+ and VREF- inputs. The operation of each pin is selected by clearing/setting the control bits in the ADCON1 register (A/D Control Register 1).

Note:	On a Power-on Reset, RA5 and RA3:RA0 are configured as analog inputs and read as '`0`'. RA6 and RA4 are configured as digital inputs.

The TRISA register controls the direction of the RA pins even when they are being used as analog inputs. The user must ensure the bits in the TRISA register are maintained set when using them as analog inputs.

EXAMPLE 10-1: INITIALIZING PORTA

```
CLRF   PORTA    ; Initialize PORTA by
                ; clearing output
                ; data latches
CLRF   LATA     ; Alternate method
                ; to clear output
                ; data latches
MOVLW  0Fh      ; Configure A/D
MOVWF  ADCON1   ; for digital inputs
MOVLW  0CFh     ; Value used to
                ; initialize data
                ; direction
MOVWF  TRISA    ; Set RA<3:0> as inputs
                ; RA<5:4> as outputs
```


FIGURE 10-2: BLOCK DIAGRAM OF RA3:RA0 AND RA5 PINS

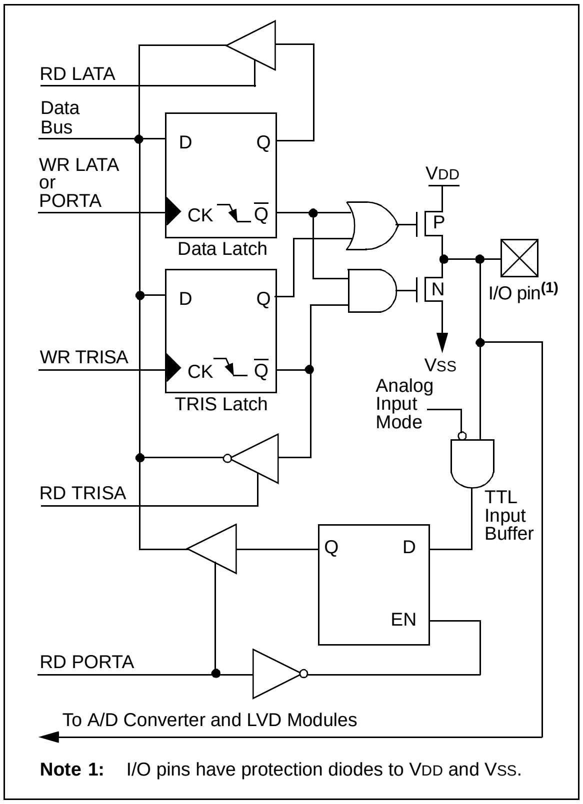


FIGURE 10-3: BLOCK DIAGRAM OF RA4/T0CKI PIN

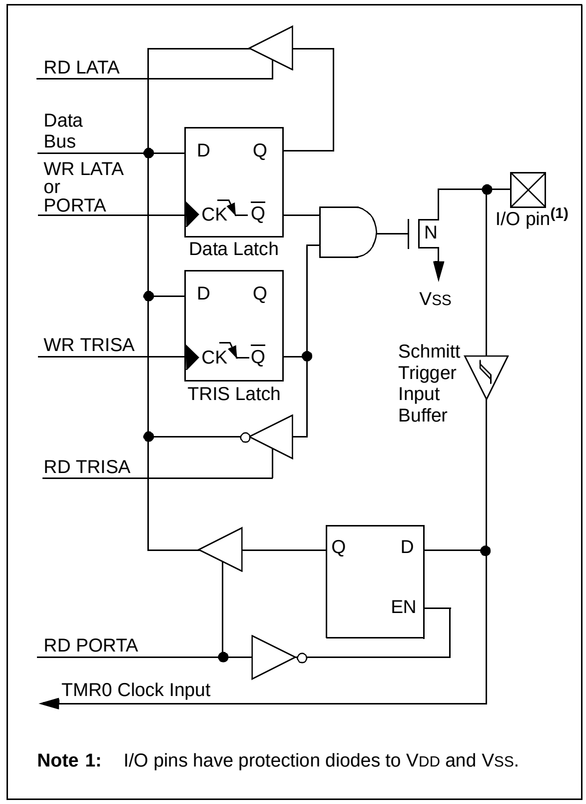


FIGURE 10-4: BLOCK DIAGRAM OF RA6 PIN (WHEN ENABLED AS I/O)

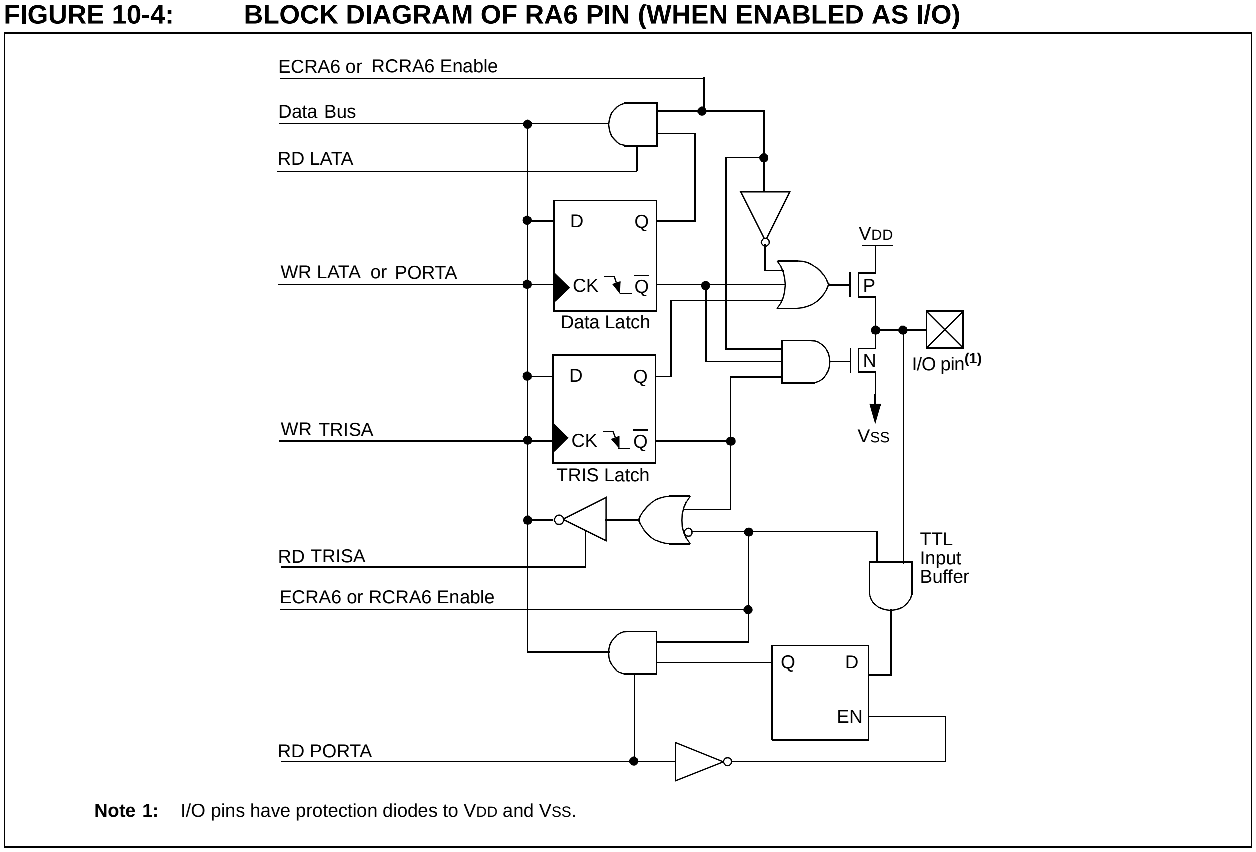


TABLE 10-1: PORTA FUNCTIONS

Name	Bit#	Buffer	Function
RA0/AN0	bit 0	TTL	Input/output or analog input.
RA1/AN1	bit 1	TTL	Input/output or analog input.
RA2/AN2/VREF-	bit 2	TTL	Input/output or analog input or VREF-.
RA3/AN3/VREF+	bit 3	TTL	Input/output or analog input or VREF+.
RA4/T0CKI	bit 4	ST/OD	Input/output or external clock input for Timer0. Output is open-drain type.
RA5/AN4/LVDIN	bit 5	TTL	Input/output or slave select input for synchronous serial port or analog input, or Low-Voltage Detect input.
OSC2/CLKO/RA6	bit 6	TTL	OSC2 or clock output, or I/O pin.

Legend: TTL = TTL input, ST = Schmitt Trigger input

TABLE 10-2: SUMMARY OF REGISTERS ASSOCIATED WITH PORTA

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other Resets
PORTA	—	RA6	RA5	RA4	RA3	RA2	RA1	RA0	-x0x 0000	-u0u 0000
LATA	—	LATA Data Output Register							-xxx xxxx	-uuu uuuu
TRISA	—	PORTA Data Direction Register							-111 1111	-111 1111
ADCON1	—	—	VCFG1	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0	--00 0000	--00 0000

Legend: x = unknown, u = unchanged, – = unimplemented locations read as '0'.
Shaded cells are not used by PORTA.

10.2 PORTB, TRISB and LATB Registers

PORTB is an 8-bit wide, bidirectional port. The corresponding data direction register is TRISB. Setting a TRISB bit (= `1`) will make the corresponding PORTB pin an input (i.e., put the corresponding output driver in a high-impedance mode). Clearing a TRISB bit (= `0`) will make the corresponding PORTB pin an output (i.e., put the contents of the output latch on the selected pin).

The Data Latch register (LATB) is also memory mapped. Read-modify-write operations on the LATB register read and write the latched output value for PORTB.

EXAMPLE 10-2: INITIALIZING PORTB

```
CLRF    PORTB   ; Initialize PORTB by
                ; clearing output
                ; data latches
CLRF    LATB    ; Alternate method
                ; to clear output
                ; data latches
MOVLW   0CFh    ; Value used to
                ; initialize data
                ; direction
MOVWF   TRISB   ; Set RB<3:0> as inputs
                ; RB<5:4> as outputs
                ; RB<7:6> as inputs
```

Each of the PORTB pins has a weak internal pull-up. A single control bit can turn on all the pull-ups. This is performed by clearing bit RBPU (INTCON2<7>). The weak pull-up is automatically turned off when the port pin is configured as an output. The pull-ups are disabled on a Power-on Reset.

Note:	On a Power-on Reset, these pins are configured as digital inputs.

Four of the PORTB pins (RB3:RB0) are the external interrupt pins, INT3 through INT0. In order to use these pins as external interrupts, the corresponding TRISB bit must be set to '`1`'.

The other four PORTB pins (RB7:RB4) have an interrupt-on-change feature. Only pins configured as inputs can cause this interrupt to occur (i.e., any RB7:RB4 pin configured as an output is excluded from the interrupt-on-change comparison). The input pins (of RB7:RB4) are compared with the old value latched on the last read of PORTB. The "mismatch" outputs of RB7:RB4 are OR'ed together to generate the RB port change interrupt with flag bit, RBIF (INTCON<0>).

This interrupt can wake the device from Sleep. The user, in the Interrupt Service Routine, can clear the interrupt in the following manner:

a) Any read or write of PORTB (except with the `MOVFF` instruction). This will end the mismatch condition.

b) Clear flag bit RBIF.

A mismatch condition will continue to set flag bit, RBIF. Reading PORTB will end the mismatch condition and allow flag bit RBIF to be cleared.

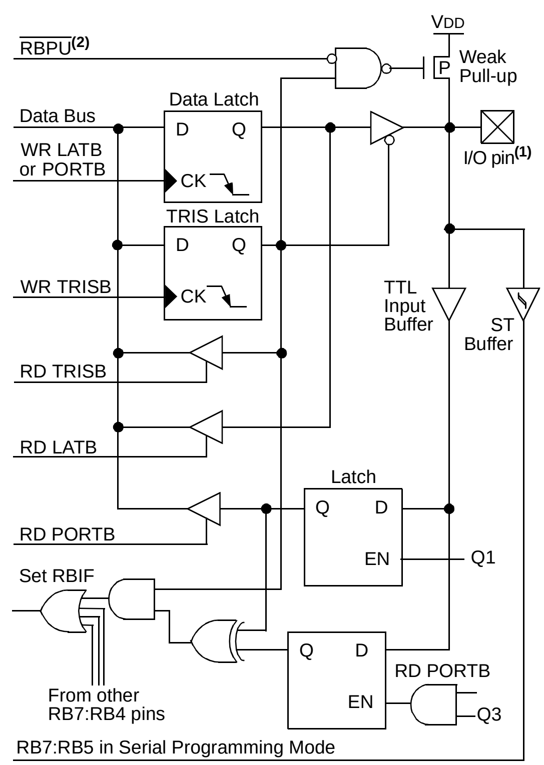
The interrupt-on-change feature is recommended for wake-up on key depression operation and operations where PORTB is only used for the interrupt-on-change feature. Polling of PORTB is not recommended while using the interrupt-on-change feature.

For PIC18FXX85 devices, RB3 can be configured by the configuration bit, CCP2MX, as the alternate peripheral pin for the CCP2 module. This is only available when the device is configured in Microprocessor, Microprocessor with Boot Block, or Extended Microcontroller Operating modes.

The RB5 pin is used as the LVP programming pin. When the LVP configuration bit is programmed, this pin loses the I/O function and becomes a programming test function.

Note:	When LVP is enabled, the weak pull-up on RB5 is disabled.

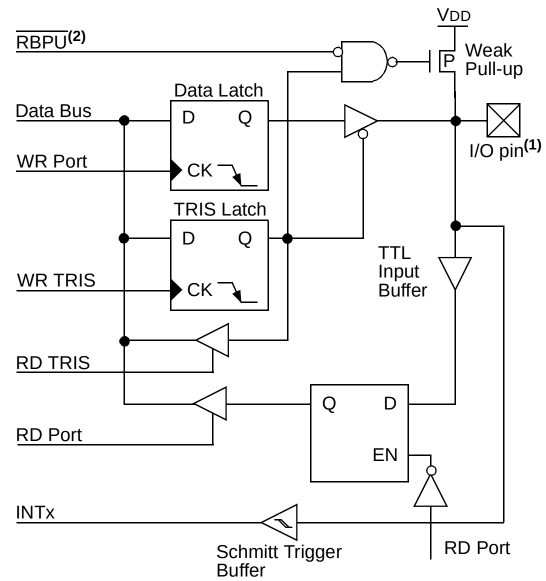
FIGURE 10-5: BLOCK DIAGRAM OF RB7:RB4 PINS



Note 1: I/O pins have diode protection to VDD and VSS.

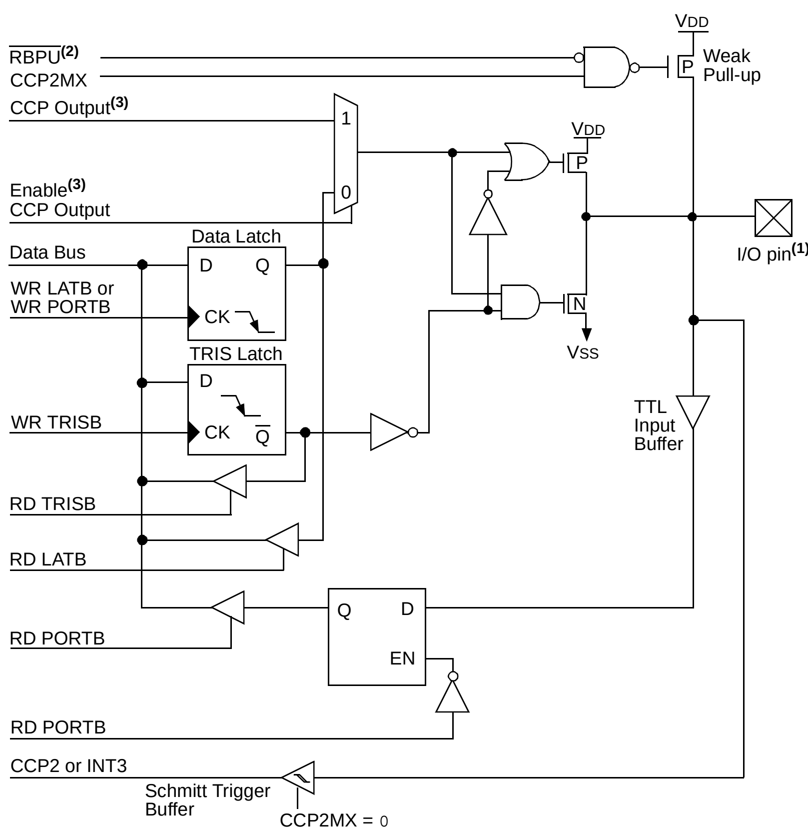
2: To enable weak pull-ups, set the appropriate TRIS bit(s) and clear the RBPU bit (INTCON2<7>).

FIGURE 10-6: BLOCK DIAGRAM OF RB2:RB0 PINS



Note 1: I/O pins have diode protection to VDD and VSS.
2: To enable weak pull-ups, set the appropriate TRIS bit(s) and clear the $\overline{\text{RBPU}}$ bit (OPTION_REG<7>).

FIGURE 10-7: BLOCK DIAGRAM OF RB3 PIN



Note 1: I/O pin has diode protection to VDD and VSS.
2: To enable weak pull-ups, set the appropriate TRIS bit(s) and clear the $\overline{\text{RBPU}}$ bit (INTCON2<7>).
3: For PIC18FXX85 parts, the CCP2 input/output is multiplexed with RB3 if the CCP2MX bit is enabled (= 0) in the Configuration register and the device is operating in Microprocessor, Microprocessor with Boot Block or Extended Microcontroller mode.

TABLE 10-3: PORTB FUNCTIONS

Name	Bit#	Buffer	Function
RB0/INT0	bit 0	TTL/ST(1)	Input/output pin or external interrupt input 0. Internal software programmable weak pull-up.
RB1/INT1	bit 1	TTL/ST(1)	Input/output pin or external interrupt input 1. Internal software programmable weak pull-up.
RB2/INT2	bit 2	TTL/ST(1)	Input/output pin or external interrupt input 2. Internal software programmable weak pull-up.
RB3/INT3/CCP2(3)	bit 3	TTL/ST(4)	Input/output pin or external interrupt input 3. Capture 2 input/ Compare 2 output/PWM output (when CCP2MX configuration bit is enabled, all PIC18FXX85 operating modes except Microcontroller mode). Internal software programmable weak pull-up.
RB4/KBI0	bit 4	TTL	Input/output pin (with interrupt-on-change). Internal software programmable weak pull-up.
RB5/KBI1/PGM	bit 5	TTL/ST(2)	Input/output pin (with interrupt-on-change). Internal software programmable weak pull-up. Low-voltage ICSP enable pin.
RB6/KBI2/PGC	bit 6	TTL/ST(2)	Input/output pin (with interrupt-on-change). Internal software programmable weak pull-up. Serial programming clock.
RB7/KBI3/PGD	bit 7	TTL/ST(2)	Input/output pin (with interrupt-on-change). Internal software programmable weak pull-up. Serial programming data.

Legend: TTL = TTL input, ST = Schmitt Trigger input

Note 1: This buffer is a Schmitt Trigger input when configured as the external interrupt.

2: This buffer is a Schmitt Trigger input when used in Serial Programming mode.

3: RC1 is the alternate assignment for CCP2 when CCP2MX is not set (all operating modes except Microcontroller mode).

4: This buffer is a Schmitt Trigger input when configured as the CCP2 input.

TABLE 10-4: SUMMARY OF REGISTERS ASSOCIATED WITH PORTB

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other Resets
PORTB	RB7	RB6	RB5	RB4	RB3	RB2	RB1	RB0	xxxx xxxx	uuuu uuuu
LATB	LATB Data Output Register								xxxx xxxx	uuuu uuuu
TRISB	PORTB Data Direction Register								1111 1111	1111 1111
INTCON	GIE/ GIEH	PEIE/ GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 0000	0000 0000
INTCON2	RBPU	INTEDG0	INTEDG1	INTEDG2	INTEDG3	TMR0IP	INT3IP	RBIP	1111 1111	1111 1111
INTCON3	INT2IP	INT1IP	INT3IE	INT2IE	INT1IE	INT3IF	INT2IF	INT1IF	1100 0000	1100 0000

Legend: x = unknown, u = unchanged. Shaded cells are not used by PORTB.

10.3 PORTC, TRISC and LATC Registers

PORTC is an 8-bit wide, bidirectional port. The corresponding data direction register is TRISC. Setting a TRISC bit (= `1`) will make the corresponding PORTC pin an input (i.e., put the corresponding output driver in a high-impedance mode). Clearing a TRISC bit (= `0`) will make the corresponding PORTC pin an output (i.e., put the contents of the output latch on the selected pin).

The Data Latch register (LATC) is also memory mapped. Read-modify-write operations on the LATC register read and write the latched output value for PORTC.

PORTC is multiplexed with several peripheral functions (Table 10-5). PORTC pins have Schmitt Trigger input buffers.

When enabling peripheral functions, care should be taken in defining TRIS bits for each PORTC pin. Some peripherals override the TRIS bit to make a pin an output, while other peripherals override the TRIS bit to make a pin an input. The user should refer to the corresponding peripheral section for the correct TRIS bit settings.

Note:	On a Power-on Reset, these pins are configured as digital inputs.

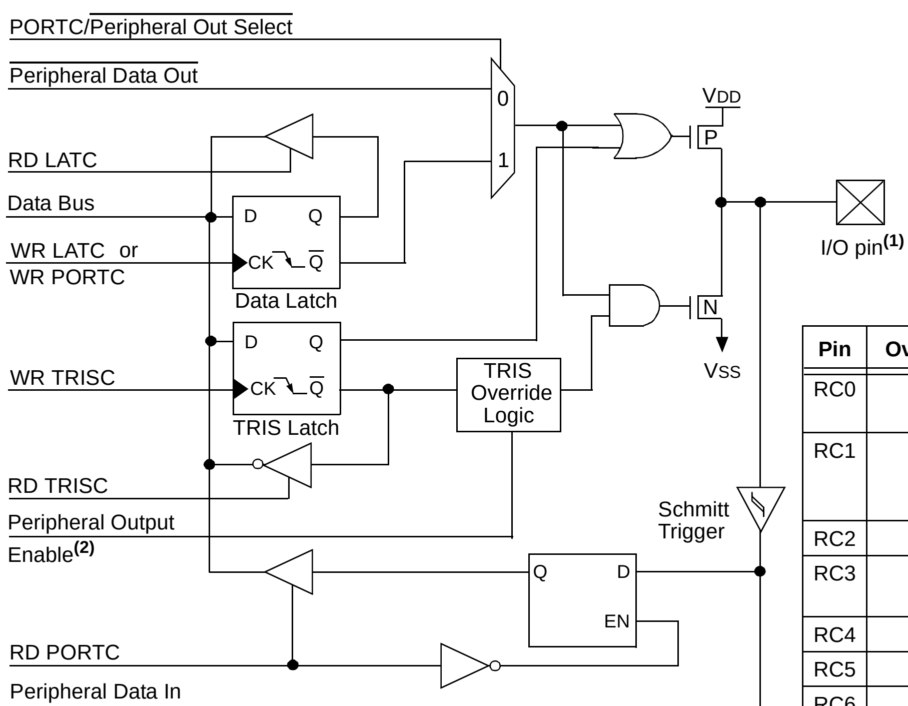
The pin override value is not loaded into the TRIS register. This allows read-modify-write of the TRIS register without concern due to peripheral overrides.

RC1 is normally configured by configuration bit, CCP2MX, as the default peripheral pin of the CCP2 module (default/erased state, CCP2MX = `1`).

EXAMPLE 10-3: INITIALIZING PORTC

```
CLRF    PORTC   ; Initialize PORTC by
                ; clearing output
                ; data latches
CLRF    LATC    ; Alternate method
                ; to clear output
                ; data latches
MOVLW   0CFh    ; Value used to
                ; initialize data
                ; direction
MOVWF   TRISC   ; Set RC<3:0> as inputs
                ; RC<5:4> as outputs
                ; RC<7:6> as inputs
```

FIGURE 10-8: PORTC BLOCK DIAGRAM (PERIPHERAL OUTPUT OVERRIDE)



TRIS OVERRIDE

Pin	Override	Peripheral
RC0	Yes	Timer1 Osc for Timer1/Timer3
RC1	Yes	Timer1 Osc for Timer1/Timer3, CCP2 I/O
RC2	Yes	CCP1 I/O
RC3	Yes	SPI/I^{2}C Master Clock
RC4	Yes	I^{2}C Data Out
RC5	Yes	SPI Data Out
RC6	Yes	USART Async Xmit, Sync Clock
RC7	Yes	USART Sync Data Out

Note 1: I/O pins have diode protection to VDD and VSS.

2: Peripheral output enable is only active if peripheral select is active.

TABLE 10-5: PORTC FUNCTIONS

Name	Bit#	Buffer Type	Function
RC0/T1OSO/T13CKI	bit 0	ST	Input/output port pin, Timer1 oscillator output or Timer1/Timer3 clock input.
RC1/T1OSI/CCP2[(1)]	bit 1	ST	Input/output port pin, Timer1 oscillator input or Capture 2 input/ Compare 2 output/PWM output (when CCP2MX configuration bit is disabled).
RC2/CCP1/P1A	bit 2	ST	Input/output port pin or Capture 1 input/Compare 1 output/ PWM1 output.
RC3/SCK/SCL	bit 3	ST	RC3 can also be the synchronous serial clock for both SPI and I^2C modes.
RC4/SDI/SDA	bit 4	ST	RC4 can also be the SPI data in (SPI mode) or data I/O (I^2C mode).
RC5/SDO	bit 5	ST	Input/output port pin or synchronous serial port data output.
RC6/TX/CK	bit 6	ST	Input/output port pin, addressable USART asynchronous transmit or addressable USART synchronous clock.
RC7/RX/DT	bit 7	ST	Input/output port pin, addressable USART asynchronous receive or addressable USART synchronous data.

Legend: ST = Schmitt Trigger input

Note 1: RB3 is the alternate assignment for CCP2 when CCP2MX is set.

TABLE 10-6: SUMMARY OF REGISTERS ASSOCIATED WITH PORTC

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other Resets
PORTC	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0	xxxx xxxx	uuuu uuuu
LATC	LATC Data Output Register								xxxx xxxx	uuuu uuuu
TRISC	PORTC Data Direction Register								1111 1111	1111 1111

Legend: x = unknown, u = unchanged

10.4 PORTD, TRISD and LATD Registers

PORTD is an 8-bit wide, bidirectional port. The corresponding data direction register is TRISD. Setting a TRISD bit (= `1`) will make the corresponding PORTD pin an input (i.e., put the corresponding output driver in a high-impedance mode). Clearing a TRISD bit (= `0`) will make the corresponding PORTD pin an output (i.e., put the contents of the output latch on the selected pin).

The Data Latch register (LATD) is also memory mapped. Read-modify-write operations on the LATD register read and write the latched output value for PORTD.

PORTD is an 8-bit port with Schmitt Trigger input buffers. Each pin is individually configurable as an input or output.

Note:	On a Power-on Reset, these pins are configured as digital inputs.

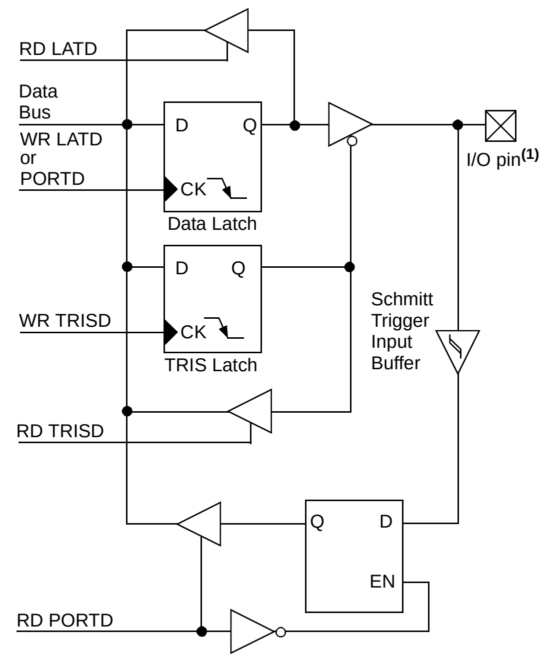
On PIC18F8X8X devices, PORTD is multiplexed with the system bus as the external memory interface; I/O port functions are only available when the system bus is disabled by setting the EBDIS bit in the MEMCOM register (MEMCON<7>). When operating as the external memory interface, PORTD is the low-order byte of the multiplexed address/data bus (AD7:AD0).

PORTD can also be configured as an 8-bit wide microprocessor port (Parallel Slave Port) by setting control bit, PSPMODE (TRISE<4>). In this mode, the input buffers are TTL. See **Section 10.10 "Parallel Slave Port (PSP)"** for additional information.

EXAMPLE 10-4: INITIALIZING PORTD

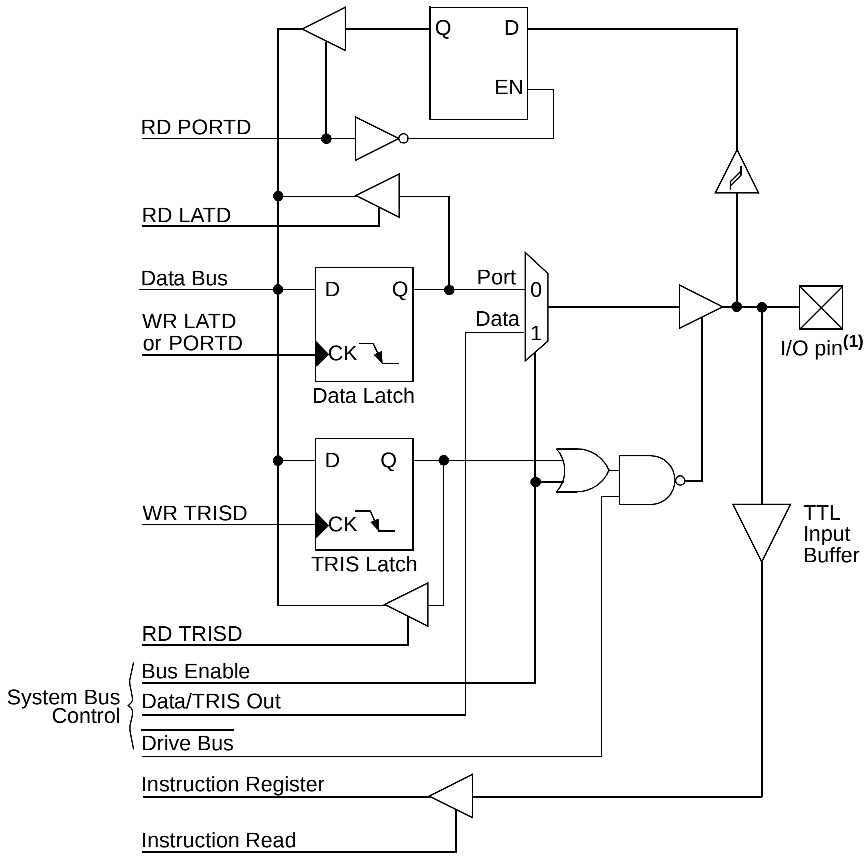
```
CLRF   PORTD   ; Initialize PORTD by
               ; clearing output
               ; data latches
CLRF   LATD    ; Alternate method
               ; to clear output
               ; data latches
MOVLW  0CFh    ; Value used to
               ; initialize data
               ; direction
MOVWF  TRISD   ; Set RD<3:0> as inputs
               ; RD<5:4> as outputs
               ; RD<7:6> as inputs
```

FIGURE 10-9: PORTD BLOCK DIAGRAM IN I/O PORT MODE



Note 1: I/O pins have diode protection to VDD and VSS.

FIGURE 10-10: PORTD BLOCK DIAGRAM IN SYSTEM BUS MODE (PIC18F8X8X ONLY)



Note 1: I/O pins have protection diodes to VDD and VSS.

TABLE 10-7: PORTD FUNCTIONS

Name	Bit#	Buffer Type	Function
RD0/PSP0/AD0(2)	bit 0	ST/TTL(1)	Input/output port pin, Parallel Slave Port bit 0 or address/data bus bit 0.
RD1/PSP1/AD1(2)	bit 1	ST/TTL(1)	Input/output port pin, Parallel Slave Port bit 1 or address/data bus bit 1.
RD2/PSP2/AD2(2)	bit 2	ST/TTL(1)	Input/output port pin, Parallel Slave Port bit 2 or address/data bus bit 2.
RD3/PSP3/AD3(2)	bit 3	ST/TTL(1)	Input/output port pin, Parallel Slave Port bit 3 or address/data bus bit 3.
RD4/PSP4/AD4(2)	bit 4	ST/TTL(1)	Input/output port pin, Parallel Slave Port bit 4 or address/data bus bit 4.
RD5/PSP5/AD5(2)	bit 5	ST/TTL(1)	Input/output port pin, Parallel Slave Port bit 5 or address/data bus bit 5.
RD6/PSP6/AD6(2)	bit 6	ST/TTL(1)	Input/output port pin, Parallel Slave Port bit 6 or address/data bus bit 6.
RD7/PSP7/AD7(2)	bit 7	ST/TTL(1)	Input/output port pin, Parallel Slave Port bit 7 or address/data bus bit 7.

Legend: ST = Schmitt Trigger input, TTL = TTL input

Note 1: Input buffers are Schmitt Triggers when in I/O mode and TTL buffers when in System Bus or Parallel Slave Port mode.

2: Available in PIC18F8X8X devices only.

TABLE 10-8: SUMMARY OF REGISTERS ASSOCIATED WITH PORTD

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other Resets
PORTD	RD7	RD6	RD5	RD4	RD3	RD2	RD1	RD0	xxxx xxxx	uuuu uuuu
LATD	LATD Data Output Register								xxxx xxxx	uuuu uuuu
TRISD	PORTD Data Direction Register								1111 1111	1111 1111
PSPCON	IBF	OBF	IBOV	PSPMODE	—	—	—	—	0000 ----	0000 ----
MEMCON	EBDIS	—	WAIT1	WAIT0	—	—	WM1	WM0	0-00 --00	0-00 --00

Legend: x = unknown, u = unchanged, – = unimplemented, read as '0'. Shaded cells are not used by PORTD.

10.5 PORTE, TRISE and LATE Registers

PORTE is an 8-bit wide, bidirectional port. The corresponding data direction register is TRISE. Setting a TRISE bit (= `1`) will make the corresponding PORTE pin an input (i.e., put the corresponding output driver in a high-impedance mode). Clearing a TRISE bit (= `0`) will make the corresponding PORTE pin an output (i.e., put the contents of the output latch on the selected pin).

Read-modify-write operations on the LATE register read and write the latched output value for PORTE.

PORTE is an 8-bit port with Schmitt Trigger input buffers. Each pin is individually configurable as an input or output. PORTE is multiplexed with the Enhanced CCP module (Table 10-9).

On PIC18F8X8X devices, PORTE is also multiplexed with the system bus as the external memory interface; the I/O bus is available only when the system bus is disabled by setting the EBDIS bit in the MEMCON register (MEMCON<7>). If the device is configured in Microprocessor or Extended Microcontroller mode, then the PORTE<7:0> becomes the high byte of the address/data bus for the external program memory interface. In Microcontroller mode, the PORTE<2:0> pins become the control inputs for the Parallel Slave Port when bit PSPMODE (PSPCON<4>) is set. (Refer to **Section 4.1.1 "PIC18F8X8X Program Memory Modes"** for more information on program memory modes.)

When the Parallel Slave Port is active, three PORTE pins (RE0/$\overline{\text{RD}}$/AD8, RE1/$\overline{\text{WR}}$/AD9 and RE2/$\overline{\text{CS}}$/AD10) function as its control inputs. This automatically occurs when the PSPMODE bit (PSPCON<4>) is set. Users must also make certain that bits TRISE<2:0> are set to configure the pins as digital inputs and the ADCON1 register is configured for digital I/O. The PORTE PSP control functions are summarized in Table 10-9.

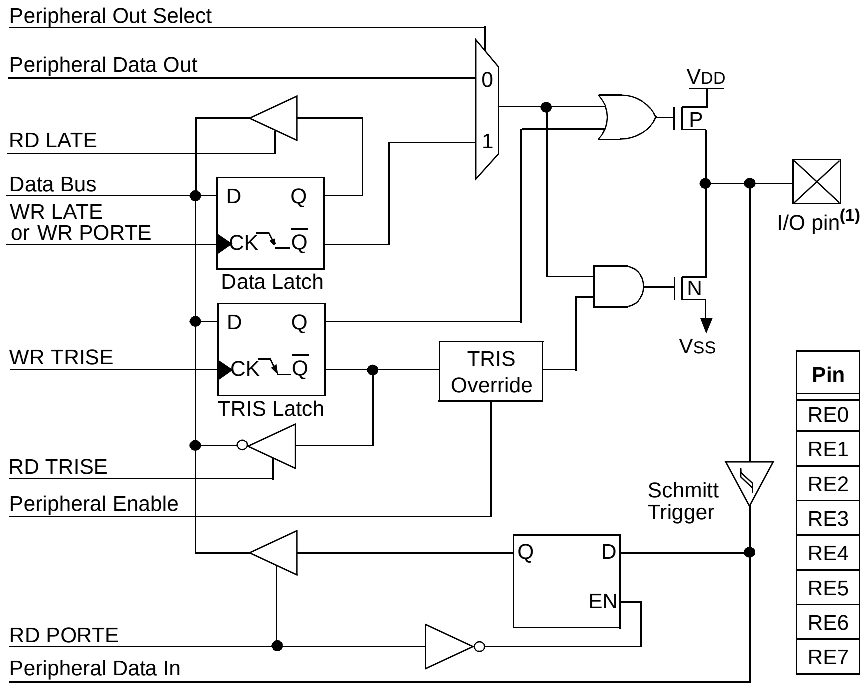
Pin RE7 can be configured as the alternate peripheral pin for the CCP2 module when the device is operating in Microcontroller mode. This is done by clearing the configuration bit, CCP2MX, in configuration register, CONFIG3H (CONFIG3H<0>).

Note:	For PIC18F8X8X (80-pin) devices operating in other than Microcontroller mode, PORTE defaults to the system bus on Power-on Reset.

EXAMPLE 10-5: INITIALIZING PORTE

```
CLRF    PORTE   ; Initialize PORTE by
                ; clearing output
                ; data latches
CLRF    LATE    ; Alternate method
                ; to clear output
                ; data latches
MOVLW   03h     ; Value used to
                ; initialize data
                ; direction
MOVWF   TRISE   ; Set RE1:RE0 as inputs
                ; RE7:RE2 as outputs
```


FIGURE 10-11: PORTE BLOCK DIAGRAM IN I/O MODE

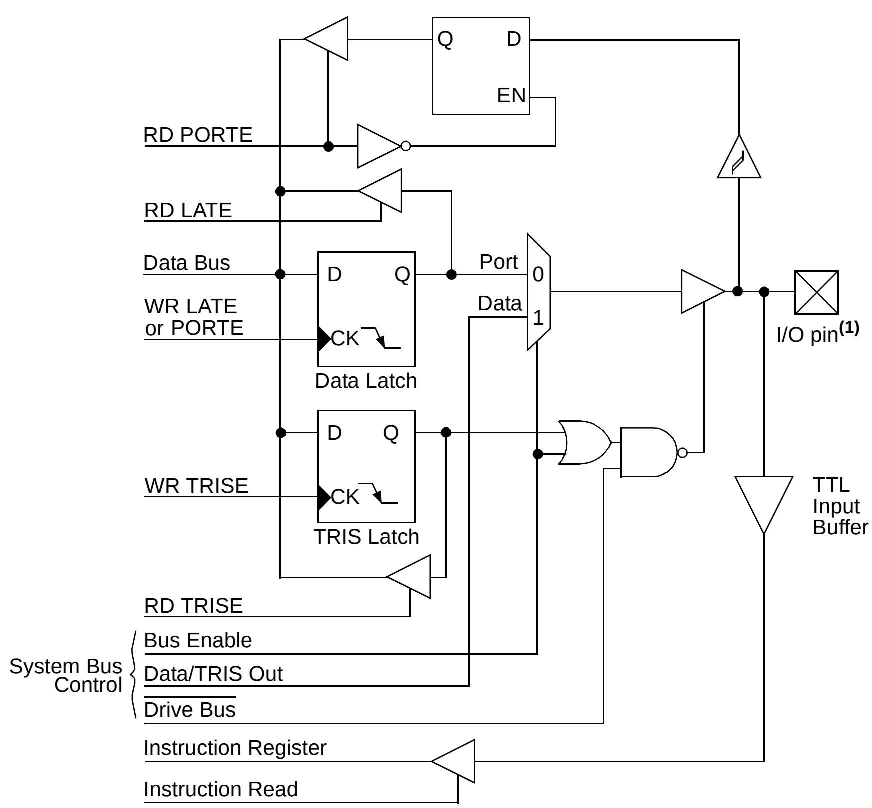


TRIS OVERRIDE

Pin	Override	Peripheral
RE0	Yes	External Bus
RE1	Yes	External Bus
RE2	Yes	External Bus
RE3	Yes	External Bus
RE4	Yes	External Bus
RE5	Yes	External Bus
RE6	Yes	External Bus
RE7	Yes	External Bus

Note 1: I/O pins have diode protection to V_{DD} and V_{SS}.

FIGURE 10-12: PORTE BLOCK DIAGRAM IN SYSTEM BUS MODE (PIC18F8X8X ONLY)



Note 1: I/O pins have protection diodes to V_{DD} and V_{SS}.

TABLE 10-9: PORTE FUNCTIONS

Name	Bit#	Buffer Type	Function
RE0/$\overline{RD}$/AD8(2)	bit 0	ST/TTL(1)	Input/output port pin, read control for Parallel Slave Port or address/data bit 8. For $\overline{RD}$ (PSP Control mode): `1` = Not a read operation `0` = Read operation, reads PORTD register (if chip selected)
RE1/$\overline{WR}$/AD9(2)	bit 1	ST/TTL(1)	Input/output port pin, write control for Parallel Slave Port or address/data bit 9. For $\overline{WR}$ (PSP Control mode): `1` = Not a write operation `0` = Write operation, writes PORTD register (if chip selected)
RE2/$\overline{CS}$/AD10(2)	bit 2	ST/TTL(1)	Input/output port pin, chip select control for Parallel Slave Port or address/data bit 10. For $\overline{CS}$ (PSP Control mode): `1` = Device is not selected `0` = Device is selected
RE3/AD11(2)	bit 3	ST/TTL(1)	Input/output port pin or address/data bit 11.
RE4/AD12(2)	bit 4	ST/TTL(1)	Input/output port pin or address/data bit 12.
RE5/AD13/(2)P1C(3)	bit 5	ST/TTL(1)	Input/output port pin, address/data bit 13 or ECCP1 PWM output C.
RE6/AD14/(2)P1B(3)	bit 6	ST/TTL(1)	Input/output port pin, address/data bit 13 or ECCP1 PWM output B.
RE7/CCP2/AD15(2)	bit 7	ST/TTL(1)	Input/output port pin, Capture 2 input/Compare 2 output/PWM output (PIC18F8X20 devices in Microcontroller mode only) or address/data bit 15.

Legend: ST = Schmitt Trigger input, TTL = TTL input

Note 1: Input buffers are Schmitt Triggers when in I/O or CCP mode, and TTL buffers when in System Bus or PSP Control mode.

2: Available in PIC18F8X8X devices only.

3: On PIC18F8X8X devices, these pins may be moved to RHY or RH6 by changing the ECCPMX configuration bit.

TABLE 10-10: SUMMARY OF REGISTERS ASSOCIATED WITH PORTE

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
TRISE	PORTE Data Direction Control Register								`1111 1111`	`1111 1111`
PORTE	Read PORTE pin/Write PORTE Data Latch								`xxxx xxxx`	`uuuu uuuu`
LATE	Read PORTE Data Latch/Write PORTE Data Latch								`xxxx xxxx`	`uuuu uuuu`
MEMCON	EBDIS	—	WAIT1	WAIT0	—	—	WM1	WM0	`0-00 --00`	`0000 --00`
PSPCON	IBF	OBF	IBOV	PSPMODE	—	—	—	—	`0000 ----`	`0000 ----`

Legend: `x` = unknown, `u` = unchanged. Shaded cells are not used by PORTE.

10.6 PORTF, LATF and TRISF Registers

PORTF is an 8-bit wide, bidirectional port. The corresponding data direction register is TRISF. Setting a TRISF bit (= 1) will make the corresponding PORTF pin an input (i.e., put the corresponding output driver in a high-impedance mode). Clearing a TRISF bit (= 0) will make the corresponding PORTF pin an output (i.e., put the contents of the output latch on the selected pin).

Read-modify-write operations on the LATF register read and write the latched output value for PORTF.

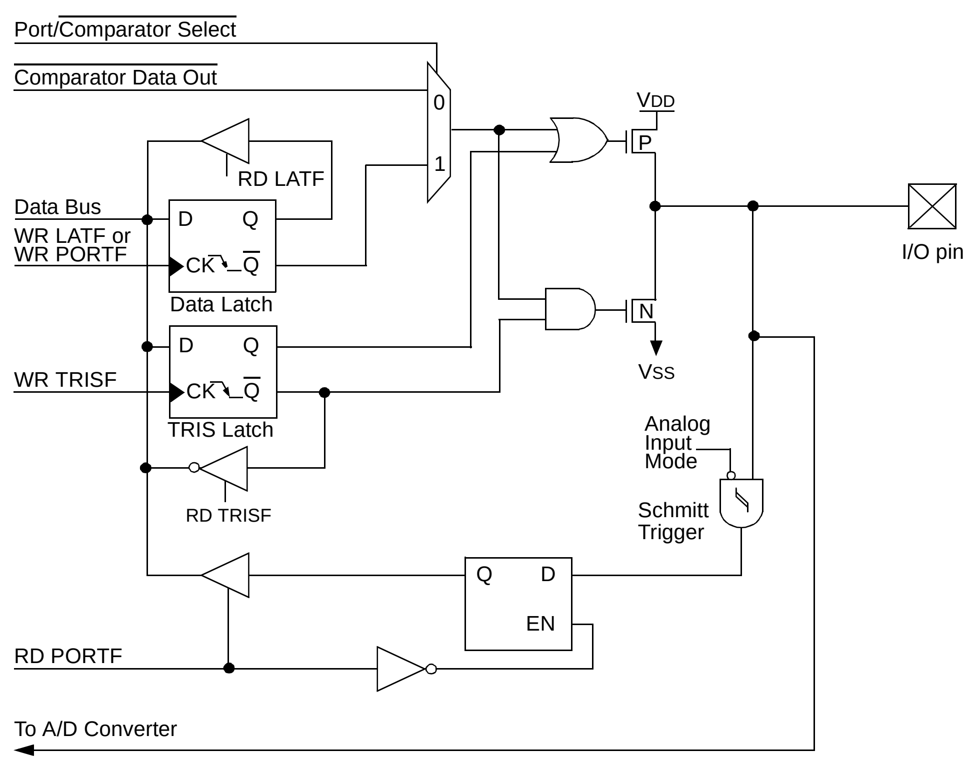
PORTF is multiplexed with several analog peripheral functions, including the A/D converter inputs and comparator inputs, outputs, and voltage reference.

Note 1:	On a Power-on Reset, the RF6:RF0 pins are configured as inputs and read as '0'.
2:	To configure PORTF as digital I/O, turn off comparators and set ADCON1 value.

EXAMPLE 10-6: INITIALIZING PORTF

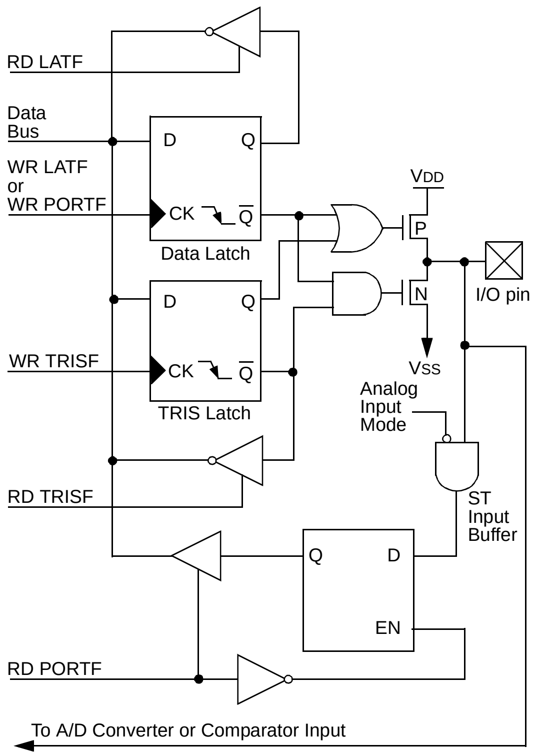
```
CLRF    PORTF      ; Initialize PORTF by
                   ; clearing output
                   ; data latches
CLRF    LATF       ; Alternate method
                   ; to clear output
                   ; data latches
MOVLW   07h        ;
MOVWF   CMCON      ; Turn off comparators
MOVLW   0Fh        ;
MOVWF   ADCON1     ; Set PORTF as digital I/O
MOVLW   0CFh       ; Value used to
                   ; initialize data
                   ; direction
MOVWF   TRISF      ; Set RF3:RF0 as inputs
                   ; RF5:RF4 as outputs
                   ; RF7:RF6 as inputs
```

FIGURE 10-13: PORTF RF1/AN6/C2OUT AND RF2/AN7/C1OUT PINS BLOCK DIAGRAM



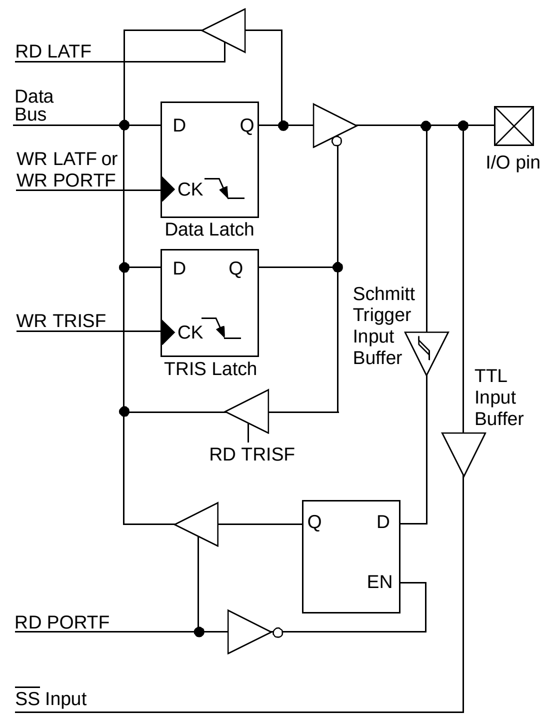
Note 1: I/O pins have diode protection to VDD and VSS.

FIGURE 10-14: RF6:RF3 AND RF0 PINS BLOCK DIAGRAM



Note 1: I/O pins have diode protection to VDD and VSS.

FIGURE 10-15: RF7 PIN BLOCK DIAGRAM



Note: I/O pins have diode protection to VDD and VSS.

TABLE 10-11: PORTF FUNCTIONS

Name	Bit#	Buffer Type	Function
RF0/AN5	bit 0	ST	Input/output port pin or analog input.
RF1/AN6/C2OUT	bit 1	ST	Input/output port pin, analog input or comparator 2 output.
RF2/AN7/C1OUT	bit 2	ST	Input/output port pin, analog input or comparator 1 output.
RF3/AN8/C2IN+	bit 3	ST	Input/output port pin, analog input or comparator 2 input (+).
RF4/AN9/C2IN-	bit 4	ST	Input/output port pin, analog input or comparator 2 input (-).
RF5/AN10/ C1IN+/CVREF	bit 5	ST	Input/output port pin, analog input, comparator 1 input (+) or comparator reference output.
RF6/AN11/C1IN-	bit 6	ST	Input/output port pin, analog input or comparator 1 input (-).
RF7/$\overline{SS}$	bit 7	ST/TTL	Input/output port pin or slave select pin for synchronous serial port.

Legend: ST = Schmitt Trigger input, TTL = TTL input

TABLE 10-12: SUMMARY OF REGISTERS ASSOCIATED WITH PORTF

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
TRISF	PORTF Data Direction Control Register								1111 1111	1111 1111
PORTF	Read PORTF pin/Write PORTF Data Latch								xxxx xxxx	uuuu uuuu
LATF	Read PORTF Data Latch/Write PORTF Data Latch								0000 0000	uuuu uuuu
ADCON1	—	—	VCFG1	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0	--00 0000	--00 0000
CMCON	C2OUT	C1OUT	C2INV	C1INV	CIS	CM2	CM1	CM0	0000 0000	0000 0000
CVRCON	CVREN	CVROE	CVRR	CVRSS	CVR3	CVR2	CVR1	CVR0	0000 0000	0000 0000

Legend: x = unknown, u = unchanged. Shaded cells are not used by PORTF.

10.7 PORTG, TRISG and LATG Registers

PORTG is a 6-bit wide port with 5 bidirectional pins and 1 unidirectional pin. The corresponding data direction register is TRISG. Setting a TRISG bit (= 1) will make the corresponding PORTG pin an input (i.e., put the corresponding output driver in a high-impedance mode). Clearing a TRISG bit (= 0) will make the corresponding PORTG pin an output (i.e., put the contents of the output latch on the selected pin).

The Data Latch register (LATG) is also memory mapped. Read-modify-write operations on the LATG register read and write the latched output value for PORTG.

Pins RG0-RG2 on PORTG are multiplexed with the CAN peripheral. Refer to **Section 23.0 "ECAN Module"** for proper settings of TRISG when CAN is enabled. RG5 is multiplexed with MCLR/VPP. Refer to Register 24-5 for more information.

When enabling peripheral functions, care should be taken in defining TRIS bits for each PORTG pin. Some peripherals override the TRIS bit to make a pin an output, while other peripherals override the TRIS bit to make a pin an input. The user should refer to the corresponding peripheral section for the correct TRIS bit settings.

Note:	On a Power-on Reset, these pins are configured as digital inputs.

The pin override value is not loaded into the TRIS register. This allows read-modify-write of the TRIS register without concern due to peripheral overrides.

EXAMPLE 10-7: INITIALIZING PORT

```
CLRF   PORTG    ; Initialize PORTG by
                ; clearing output
                ; data latches
CLRF   LATG     ; Alternate method
                ; to clear output
                ; data latches
MOVLW  04h      ; Value used to
                ; initialize data
                ; direction
MOVWF  TRISG    ; Set RG1:RG0 as outputs
                ; RG2 as input
                ; RG4:RG3 as inputs
```

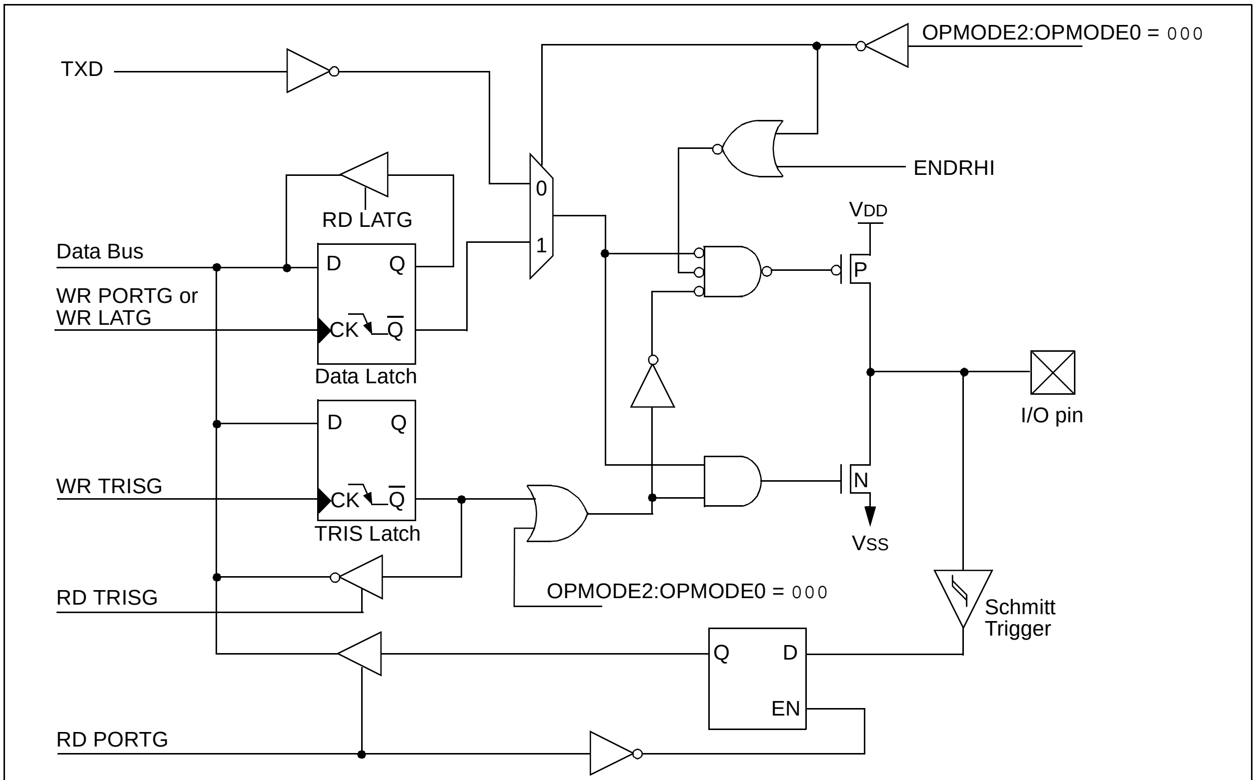
Note 1: On a Power-on Reset, RG5 is enabled as a digital input only if Master Clear functionality is disabled (MCLRE = 0).

2: If the device Master Clear is disabled, verify that either of the following is done to ensure proper entry into ICSP mode:

a) disable Low-Voltage Programming (CONFIG4L<2> = 0); or

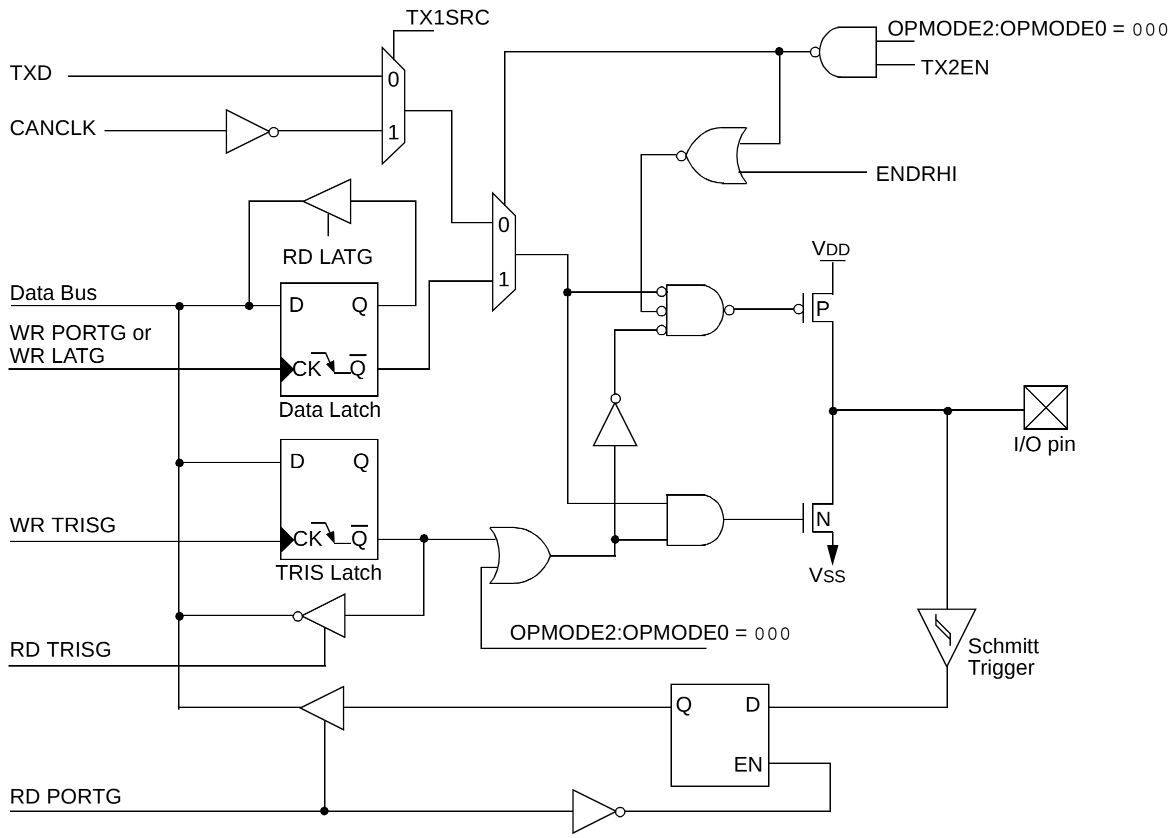
b) make certain that RB5/KBI1/PGM is held low during entry into ICSP.

FIGURE 10-16: RG0/CANTX1 PIN BLOCK DIAGRAM



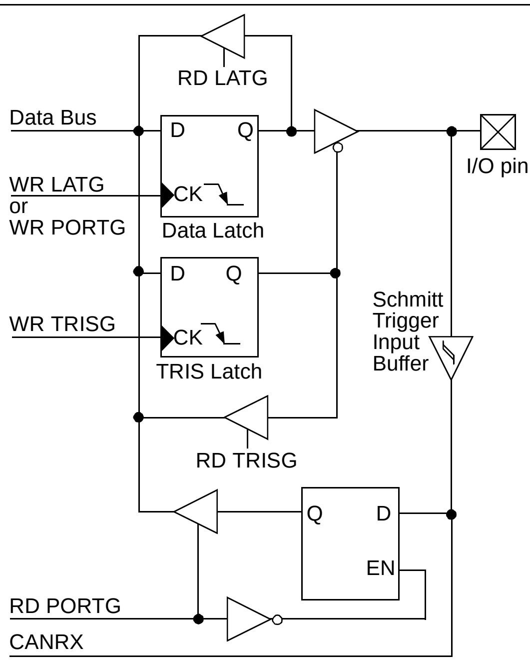
Note: I/O pins have diode protection to VDD and VSS.

FIGURE 10-17: RG1/CANTX2 PIN BLOCK DIAGRAM



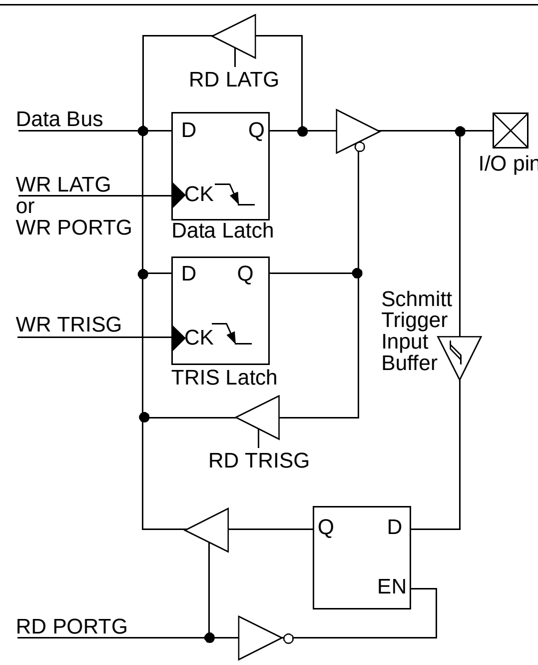
Note: I/O pins have diode protection to VDD and VSS.

FIGURE 10-18: RG2/CANRX PIN BLOCK DIAGRAM



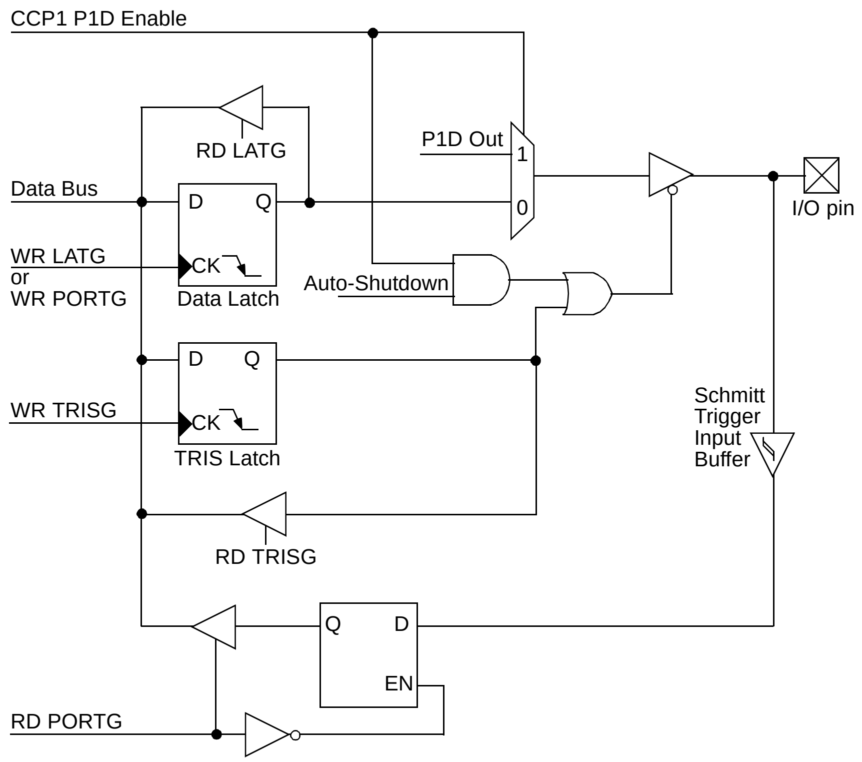
Note: I/O pins have diode protection to VDD and VSS.

FIGURE 10-19: RG3 PIN BLOCK DIAGRAM



Note: I/O pins have diode protection to VDD and VSS.

FIGURE 10-20: RG4/P1D PIN BLOCK DIAGRAM



Note: I/O pins have diode protection to VDD and VSS.

FIGURE 10-21: RG5/$\overline{\text{MCLR}}$/VPP PIN BLOCK DIAGRAM

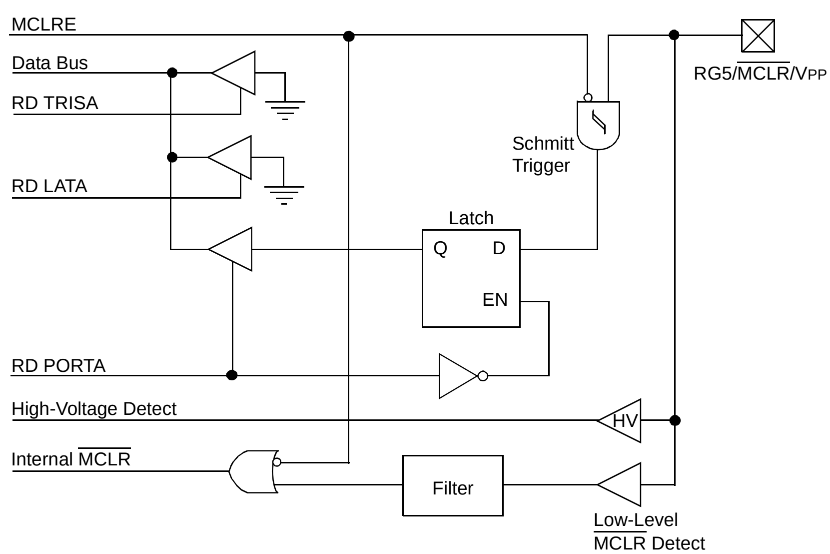


TABLE 10-13: PORTG FUNCTIONS

Name	Bit#	Buffer Type	Function
RG0/CANTX1	bit 0	ST	Input/output port pin or CAN bus transmit output.
RG1/CANTX2	bit 1	ST	Input/output port pin, CAN bus complimentary transmit output or CAN bus bit time clock.
RG2/CANRX	bit 2	ST	Input/output port pin or CAN bus receive.
RG3	bit 3	ST	Input/output port pin.
RG4/P1D	bit 4	ST	Input/output port pin or ECCP1 PWM output D.
RG5/$\overline{\text{MCLR}}$/VPP	bit 5	ST	Master Clear input or programming voltage input (if $\overline{\text{MCLR}}$ is enabled). Input only port pin or programming voltage input (if $\overline{\text{MCLR}}$ is disabled).

Legend: ST = Schmitt Trigger input

TABLE 10-14: SUMMARY OF REGISTERS ASSOCIATED WITH PORTG

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other Resets
PORTG	—	—	RG5(1)	Read PORTF pin/Write PORTF Data Latch					--0x xxxx	--0u uuuu
LATG	—	—	—	LATG Data Output Register					---x xxxx	---u uuuu
TRISG	—	—	—	Data Direction Control Register for PORTG					---1 1111	---1 1111

Legend: x = unknown, u = unchanged

Note 1: RG5 is available as an input only when $\overline{\text{MCLR}}$ is disabled.

10.8 PORTH, LATH and TRISH Registers

Note:	PORTH is available only on PIC18F8X8X devices.

PORTH is an 8-bit wide, bidirectional I/O port. The corresponding data direction register is TRISH. Setting a TRISH bit (= 1) will make the corresponding PORTH pin an input (i.e., put the corresponding output driver in a high-impedance mode). Clearing a TRISH bit (= 0) will make the corresponding PORTH pin an output (i.e., put the contents of the output latch on the selected pin).

Read-modify-write operations on the LATH register read and write the latched output value for PORTH.

Pins RH7:RH4 are multiplexed with analog inputs AN15:AN12. Pins RH3:RH0 are multiplexed with the system bus as the external memory interface; they are the high-order address bits, A19:A16. By default, pins RH7:RH4 are enabled as A/D inputs and pins RH3:RH0 are enabled as the system address bus. Register ADCON1 configures RH7:RH4 as I/O or A/D inputs. Register MEMCON configures RH3:RH0 as I/O or system bus pins.

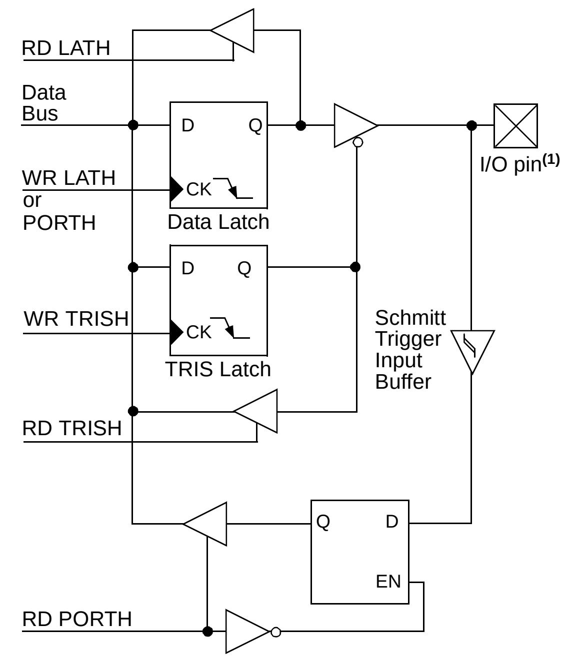
Pins RH7 and RH6 can be configured as the alternate peripheral pins for CCP1 PWM output P1B and P1C, respectively. This is done by clearing the configuration bit ECCPMX, in configuration register CONFIG3H (CONFIG3H<1>).

Note 1:	On Power-on Reset, PORTH pins RH7:RH4 default to A/D inputs and read as '0'.
2:	On Power-on Reset, PORTH pins RH3:RH0 default to system bus signals.

EXAMPLE 10-8: INITIALIZING PORTH

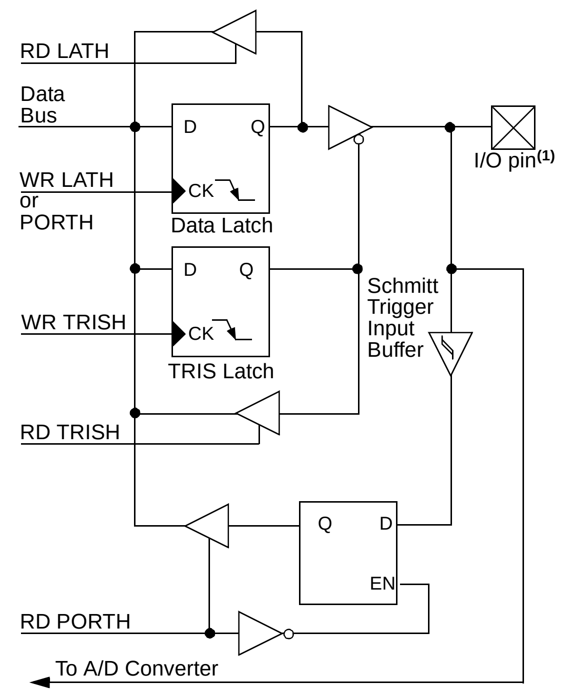
```
CLRF    PORTH       ; Initialize PORTH by
                    ; clearing output
                    ; data latches
CLRF    LATH        ; Alternate method
                    ; to clear output
                    ; data latches
MOVLW   0Fh         ;
MOVWF   ADCON1      ;
MOVLW   0CFh        ; Value used to
                    ; initialize data
                    ; direction
MOVWF   TRISH       ; Set RH3:RH0 as inputs
                    ; RH5:RH4 as outputs
                    ; RH7:RH6 as inputs
```

FIGURE 10-22: RH3:RH0 PINS BLOCK DIAGRAM IN I/O MODE



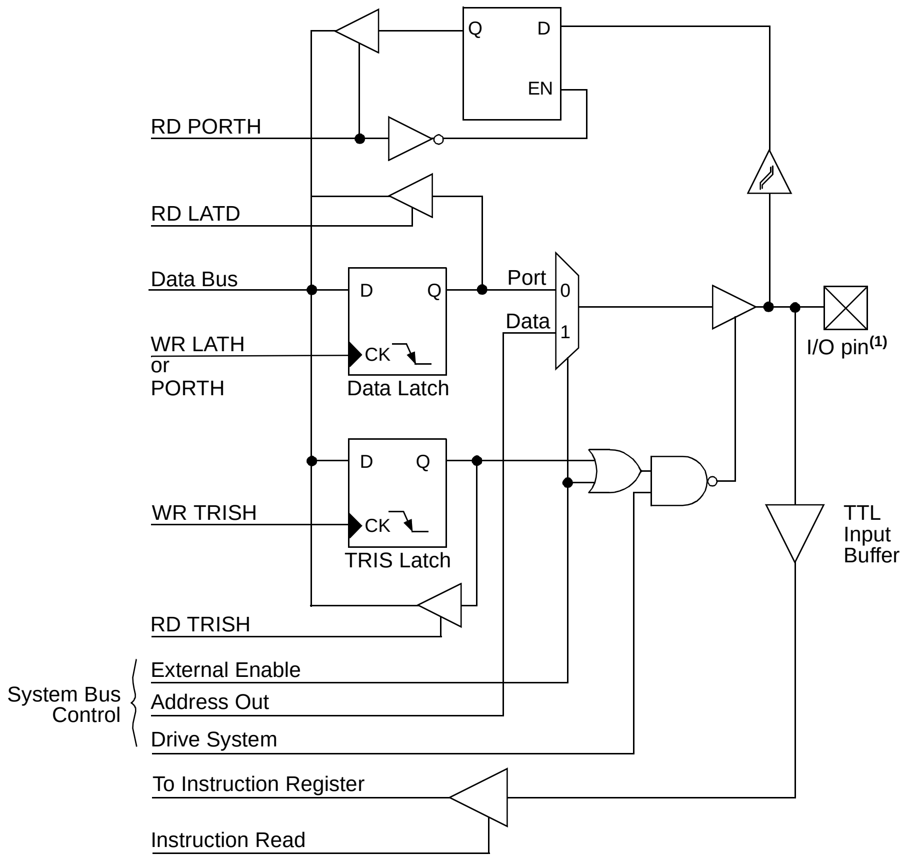
Note 1: I/O pins have diode protection to VDD and VSS.

FIGURE 10-23: RH7:RH4 PINS BLOCK DIAGRAM IN I/O MODE



Note 1: I/O pins have diode protection to VDD and VSS.

FIGURE 10-24: RH3:RH0 PINS BLOCK DIAGRAM IN SYSTEM BUS MODE



Note 1: I/O pins have diode protection to VDD and VSS.

TABLE 10-15: PORTH FUNCTIONS

Name	Bit#	Buffer Type	Function
RH0/A16	bit 0	ST/TTL(1)	Input/output port pin or address bit 16 for external memory interface.
RH1/A17	bit 1	ST/TTL(1)	Input/output port pin or address bit 17 for external memory interface.
RH2/A18	bit 2	ST/TTL(1)	Input/output port pin or address bit 18 for external memory interface.
RH3/A19	bit 3	ST/TTL(1)	Input/output port pin or address bit 19 for external memory interface.
RH4/AN12	bit 4	ST	Input/output port pin or analog input channel 12.
RH5/AN13	bit 5	ST	Input/output port pin or analog input channel 13.
RH6/AN14/P1C(2)	bit 6	ST	Input/output port pin or analog input channel 14.
RH7/AN15/P1B(2)	bit 7	ST	Input/output port pin or analog input channel 15.

Legend: ST = Schmitt Trigger input, TTL = TTL input

Note 1: Input buffers are Schmitt Triggers when in I/O mode and TTL buffers when in System Bus or Parallel Slave Port mode.

2: Alternate pin assignment when ECCPMX configuration bit is cleared.

TABLE 10-16: SUMMARY OF REGISTERS ASSOCIATED WITH PORTH

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
TRISH	PORTH Data Direction Control Register								1111 1111	1111 1111
PORTH	Read PORTH pin/Write PORTH Data Latch								xxxx xxxx	uuuu uuuu
LATH	Read PORTH Data Latch/Write PORTH Data Latch								xxxx xxxx	uuuu uuuu
ADCON1	—	—	VCFG1	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0	--00 0000	--00 0000
MEMCON(1)	EBDIS	—	WAIT1	WAIT0	—	—	WM1	WM0	0-00 --00	0-00 --00

Legend: x = unknown, u = unchanged, – = unimplemented. Shaded cells are not used by PORTH.

Note 1: This register is held in Reset in Microcontroller mode.

10.9 PORTJ, TRISJ and LATJ Registers

Note:	PORTJ is available only on PIC18F8X8X devices.

PORTJ is an 8-bit wide, bidirectional port. The corresponding data direction register is TRISJ. Setting a TRISJ bit (= 1) will make the corresponding PORTJ pin an input (i.e., put the corresponding output driver in a high-impedance mode). Clearing a TRISJ bit (= 0) will make the corresponding PORTJ pin an output (i.e., put the contents of the output latch on the selected pin).

The Data Latch register (LATJ) is also memory mapped. Read-modify-write operations on the LATJ register read and write the latched output value for PORTJ.

PORTJ is multiplexed with the system bus as the external memory interface; I/O port functions are only available when the system bus is disabled. When operating as the external memory interface, PORTJ provides the control signal to external memory devices. The RJ5 pin is not multiplexed with any system bus functions.

When enabling peripheral functions, care should be taken in defining TRIS bits for each PORTJ pin. Some peripherals override the TRIS bit to make a pin an output, while other peripherals override the TRIS bit to make a pin an input. The user should refer to the corresponding peripheral section for the correct TRIS bit settings.

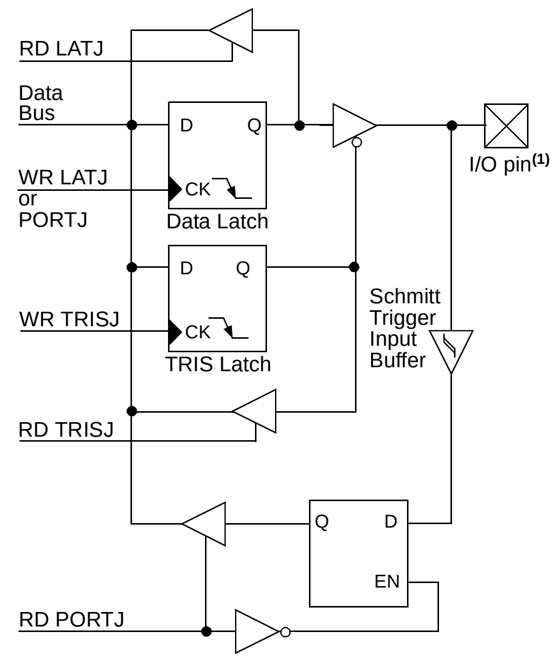
Note:	On a Power-on Reset, these pins are configured as digital inputs.

The pin override value is not loaded into the TRIS register. This allows read-modify-write of the TRIS register without concern due to peripheral overrides.

EXAMPLE 10-9: INITIALIZING PORTJ

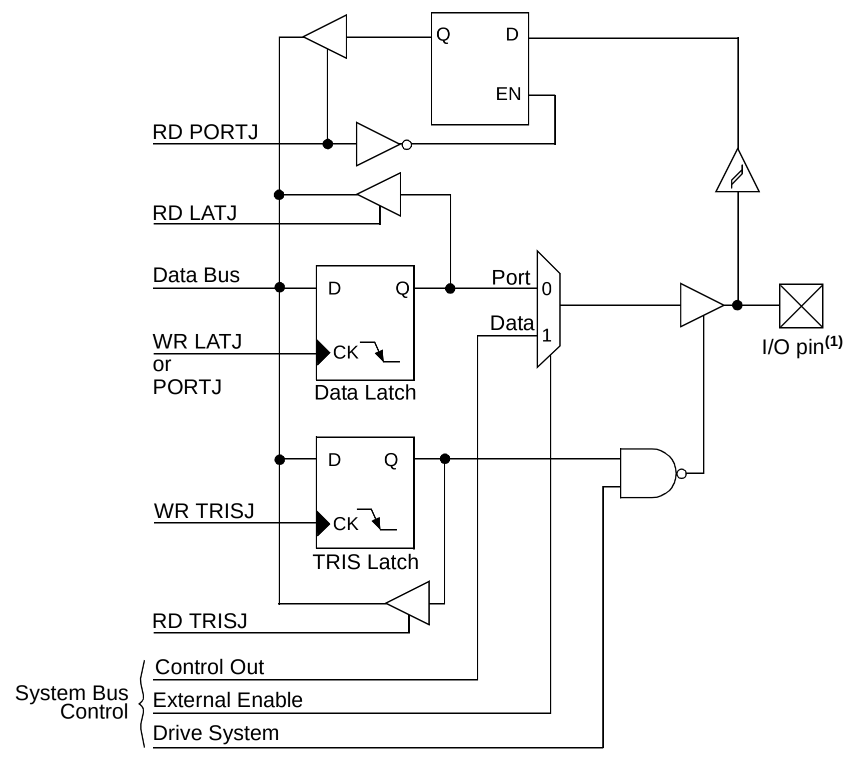
```
CLRF   PORTJ   ; Initialize PORTG by
               ; clearing output
               ; data latches
CLRF   LATJ    ; Alternate method
               ; to clear output
               ; data latches
MOVLW  0CFh    ; Value used to
               ; initialize data
               ; direction
MOVWF  TRISJ   ; Set RJ3:RJ0 as inputs
               ; RJ5:RJ4 as output
               ; RJ7:RJ6 as inputs
```

FIGURE 10-25: PORTJ BLOCK DIAGRAM IN I/O MODE



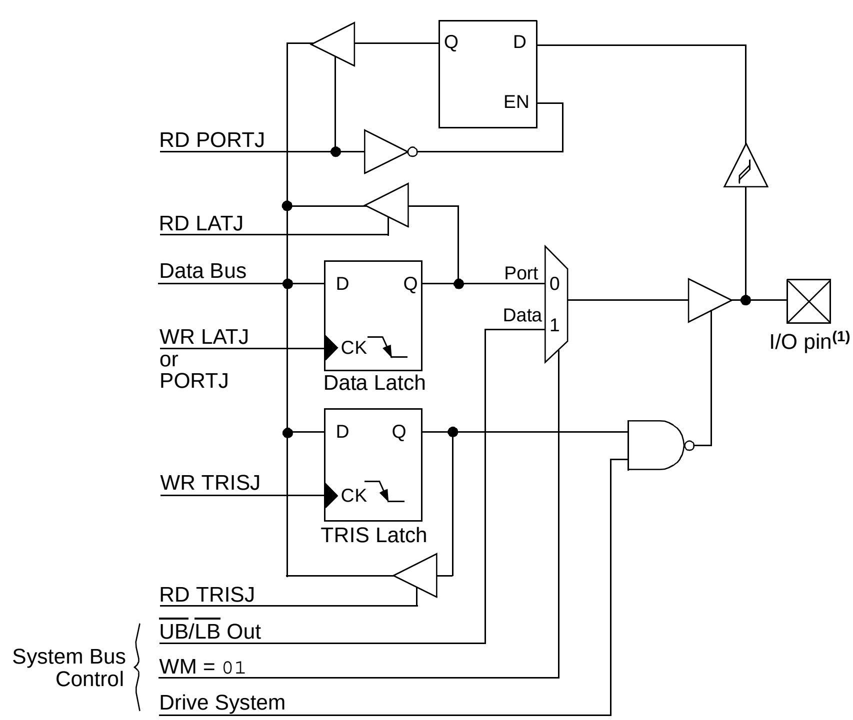
Note 1: I/O pins have diode protection to VDD and VSS.

FIGURE 10-26: RJ5:RJ0 PINS BLOCK DIAGRAM IN SYSTEM BUS MODE



Note 1: I/O pins have diode protection to VDD and VSS.

FIGURE 10-27: RJ7:RJ6 PINS BLOCK DIAGRAM IN SYSTEM BUS MODE



Note 1: I/O pins have diode protection to VDD and VSS.

TABLE 10-17: PORTJ FUNCTIONS

Name	Bit#	Buffer Type	Function
RJ0/ALE	bit 0	ST	Input/output port pin or address latch enable control for external memory interface.
RJ1/$\overline{OE}$	bit 1	ST	Input/output port pin or output enable control for external memory interface.
RJ2/$\overline{WRL}$	bit 2	ST	Input/output port pin or write low byte control for external memory interface.
RJ3/$\overline{WRH}$	bit 3	ST	Input/output port pin or write high byte control for external memory interface.
RJ4/BA0	bit 4	ST	Input/output port pin or byte address 0 control for external memory interface.
RJ5/$\overline{CE}$	bit 5	ST	Input/output port pin or external memory chip enable.
RJ6/$\overline{LB}$	bit 6	ST	Input/output port pin or lower byte select control for external memory interface.
RJ7/$\overline{UB}$	bit 7	ST	Input/output port pin or upper byte select control for external memory interface.

Legend: ST = Schmitt Trigger input

TABLE 10-18: SUMMARY OF REGISTERS ASSOCIATED WITH PORTJ

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other Resets
PORTJ	Read PORTJ pin/Write PORTJ Data Latch								xxxx xxxx	uuuu uuuu
LATJ	LATJ Data Output Register								xxxx xxxx	uuuu uuuu
TRISJ	Data Direction Control Register for PORTJ								1111 1111	1111 1111

Legend: x = unknown, u = unchanged

10.10 Parallel Slave Port (PSP)

PORTD also operates as an 8-bit wide Parallel Slave Port, or microprocessor port, when control bit PSPMODE (TRISE<4>) is set. It is asynchronously readable and writable by the external world through RD control input pin, RE0/RD/AD8 and WR control input pin, RE1/WR/AD9.

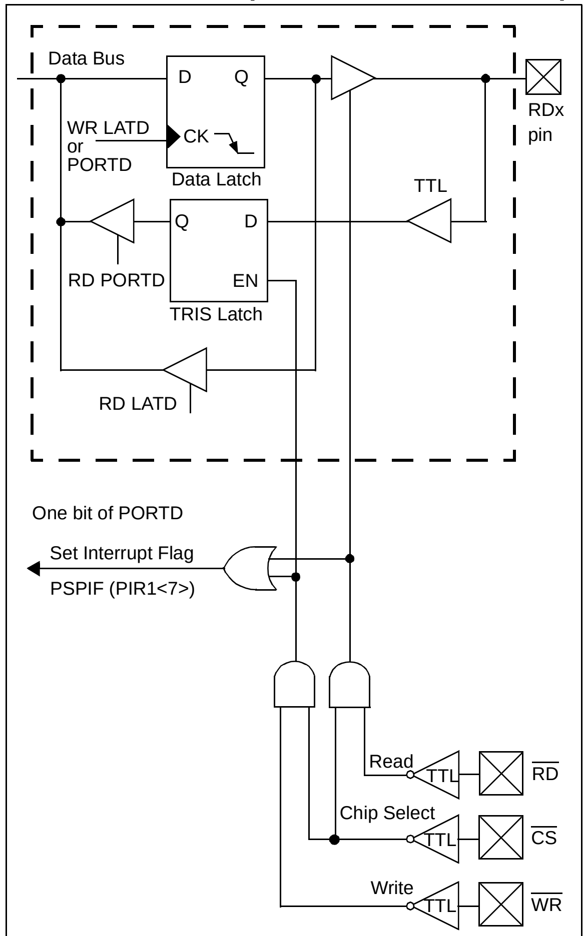
Note:	For PIC18F8X8X devices, the Parallel Slave Port is available only in Microcontroller mode.

The PSP can directly interface to an 8-bit microprocessor data bus. The external microprocessor can read or write the PORTD latch as an 8-bit latch. Setting bit PSPMODE enables port pin RE0/RD/AD8 to be the RD input, RE1/WR/AD9 to be the WR input and RE2/CS/AD10 to be the CS (chip select) input. For this functionality, the corresponding data direction bits of the TRISE register (TRISE<2:0>) must be configured as inputs (set). The A/D port configuration bits PCFG2:PCFG0 (ADCON1<2:0>) must be set, which will configure pins RE2:RE0 as digital I/O.

A write to the PSP occurs when both the CS and WR lines are first detected low. A read from the PSP occurs when both the CS and RD lines are first detected low.

The PORTE I/O pins become control inputs for the microprocessor port when bit PSPMODE (PSPCON<4>) is set. In this mode, the user must make sure that the TRISE<2:0> bits are set (pins are configured as digital inputs) and the ADCON1 is configured for digital I/O. In this mode, the input buffers are TTL.

FIGURE 10-28: PORTD AND PORTE BLOCK DIAGRAM (PARALLEL SLAVE PORT)



Note: I/O pin has protection diodes to VDD and VSS.

REGISTER 10-1: PSPCON REGISTER

R-0	R-0	R/W-0	R/W-0	U-0	U-0	U-0	U-0
IBF	OBF	IBOV	PSPMODE	—	—	—	—
bit 7							bit 0

bit 7 **IBF:** Input Buffer Full Status bit
1 = A data byte has been received and is waiting to be read by the CPU
0 = No data byte has been received

bit 6 **OBF:** Output Buffer Full Status bit
1 = The output buffer still holds a previously written data byte
0 = The output buffer has been read

bit 5 **IBOV:** Input Buffer Overflow Detect bit
1 = A write occurred when a previously input data byte has not been read (must be cleared in software)
0 = No overflow occurred

bit 4 **PSPMODE:** Parallel Slave Port Mode Select bit
1 = Parallel Slave Port mode
0 = General Purpose I/O mode

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

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

FIGURE 10-29: PARALLEL SLAVE PORT WRITE WAVEFORMS

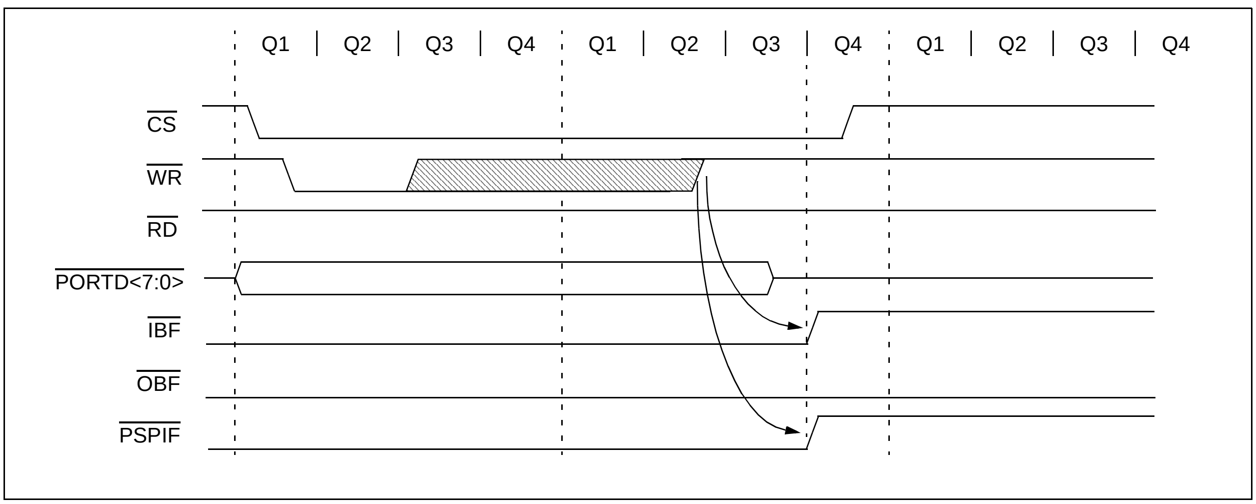


FIGURE 10-30: PARALLEL SLAVE PORT READ WAVEFORMS

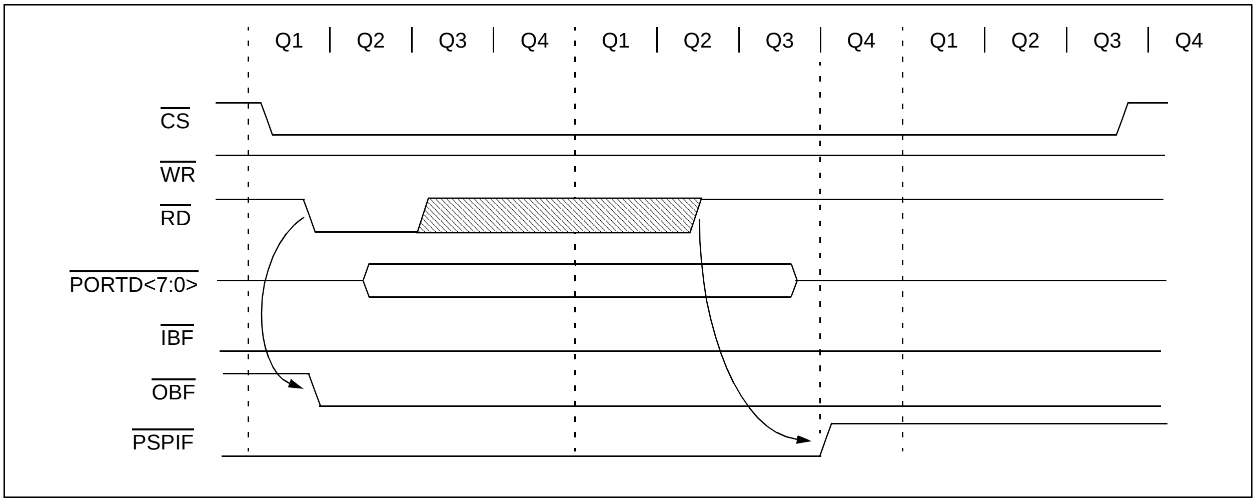


TABLE 10-19: REGISTERS ASSOCIATED WITH PARALLEL SLAVE PORT

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other Resets
PORTD	Port Data Latch when Written; Port pins when Read								xxxx xxxx	uuuu uuuu
LATD	LATD Data Output bits								xxxx xxxx	uuuu uuuu
TRISD	PORTD Data Direction bits								1111 1111	1111 1111
PORTE	RE7/CCP2/ AD15	RE6/AD14/ P1B	RE5/AD13/ P1C	RE4/ AD12	RE3/ AD11	RE2/$\overline{CS}$(1)/ AD10	RE1/$\overline{WR}$(1)/ AD9	RE0/$\overline{RD}$(1)/ AD8	xxxx xxxx	uuuu uuuu
LATE	LATE Data Output bits								xxxx xxxx	uuuu uuuu
TRISE	PORTE Data Direction bits								1111 1111	1111 1111
PSPCON	IBF	OBF	IBOV	PSPMODE	—	—	—	—	0000 ----	0000 ----
INTCON	GIE/ GIEH	PEIE/ GIEL	TMR0IF	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 0000	0000 0000
PIR1	PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
PIE1	PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
IPR1	PSPIP(1)	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	1111 1111	1111 1111

Legend: x = unknown, u = unchanged, – = unimplemented, read as '0'. Shaded cells are not used by the Parallel Slave Port.

Note 1: Enabled only in Microcontroller mode.

11.0 TIMER0 MODULE

The Timer0 module has the following features:

- Software selectable as an 8-bit or 16-bit timer/ counter
- Readable and writable
- Dedicated 8-bit software programmable prescaler
- Clock source selectable to be external or internal
- Interrupt-on-overflow from 0FFh to 00h in 8-bit mode and 0FFFFh to 0000h in 16-bit mode
- Edge select for external clock

Figure 11-1 shows a simplified block diagram of the Timer0 module in 8-bit mode and Figure 11-2 shows a simplified block diagram of the Timer0 module in 16-bit mode.

The T0CON register (Register 11-1) is a readable and writable register that controls all the aspects of Timer0, including the prescale selection.

Note:	Timer0 is enabled on POR.

REGISTER 11-1: T0CON: TIMER0 CONTROL REGISTER

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0
bit 7							bit 0

bit 7 **TMR0ON:** Timer0 On/Off Control bit
1 = Enables Timer0
0 = Stops Timer0

bit 6 **T08BIT**: Timer0 8-bit/16-bit Control bit
1 = Timer0 is configured as an 8-bit timer/counter
0 = Timer0 is configured as a 16-bit timer/counter

bit 5 **T0CS**: Timer0 Clock Source Select bit
1 = Transition on T0CKI pin
0 = Internal instruction cycle clock (CLKO)

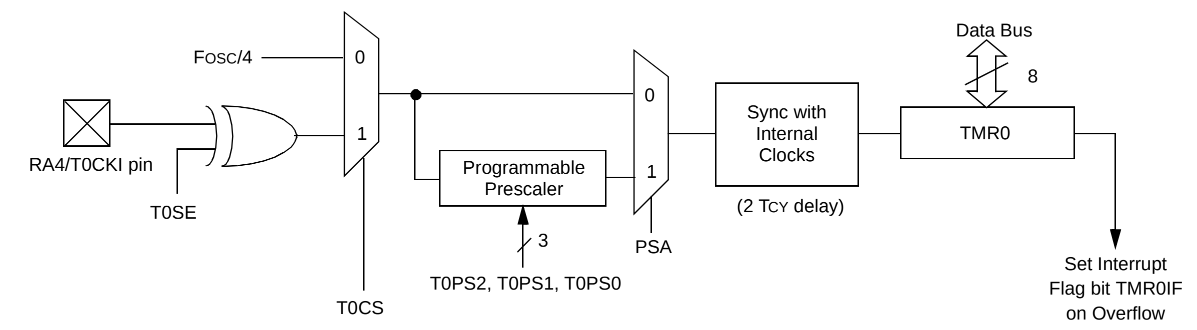
bit 4 **T0SE**: Timer0 Source Edge Select bit
1 = Increment on high-to-low transition on T0CKI pin
0 = Increment on low-to-high transition on T0CKI pin

bit 3 **PSA**: Timer0 Prescaler Assignment bit
1 = TImer0 prescaler is not assigned. Timer0 clock input bypasses prescaler.
0 = Timer0 prescaler is assigned. Timer0 clock input comes from prescaler output.

bit 2-0 **T0PS2:T0PS0**: Timer0 Prescaler Select bits
111 = 1:256 prescale value
110 = 1:128 prescale value
101 = 1:64 prescale value
100 = 1:32 prescale value
011 = 1:16 prescale value
010 = 1:8 prescale value
001 = 1:4 prescale value
000 = 1:2 prescale value

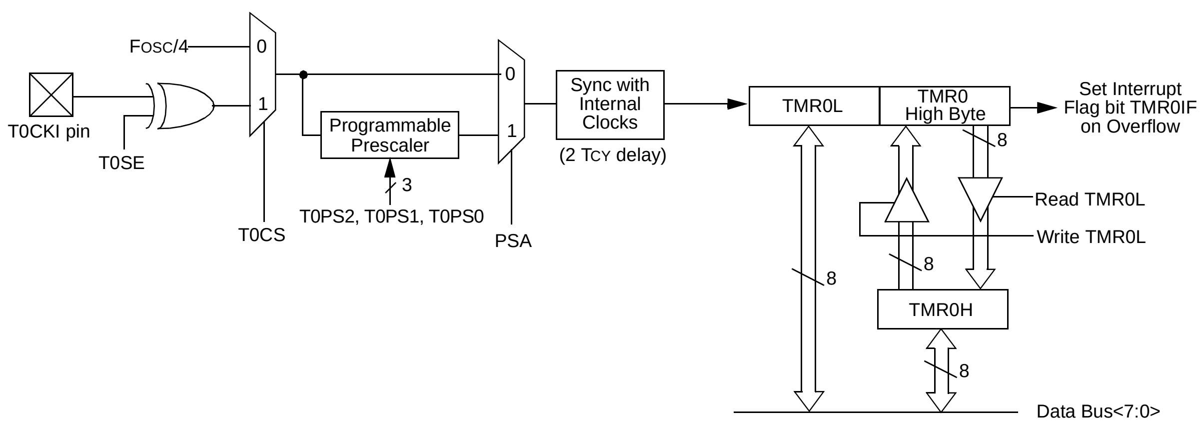
Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
- n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

FIGURE 11-1: TIMER0 BLOCK DIAGRAM IN 8-BIT MODE



Note: Upon Reset, Timer0 is enabled in 8-bit mode with clock input from T0CKI max. prescale.

FIGURE 11-2: TIMER0 BLOCK DIAGRAM IN 16-BIT MODE



Note: Upon Reset, Timer0 is enabled in 8-bit mode with clock input from T0CKI max. prescale.

11.1 Timer0 Operation

Timer0 can operate as a timer or as a counter.

Timer mode is selected by clearing the T0CS bit. In Timer mode, the Timer0 module will increment every instruction cycle (without prescaler). If the TMR0 register is written, the increment is inhibited for the following two instruction cycles. The user can work around this by writing an adjusted value to the TMR0 register.

Counter mode is selected by setting the T0CS bit. In Counter mode, Timer0 will increment either on every rising or falling edge of pin RA4/T0CKI. The incrementing edge is determined by the Timer0 Source Edge Select bit (T0SE). Clearing the T0SE bit selects the rising edge. Restrictions on the external clock input are discussed below.

When an external clock input is used for Timer0, it must meet certain requirements. The requirements ensure the external clock can be synchronized with the internal phase clock (TOSC). Also, there is a delay in the actual incrementing of Timer0 after synchronization.

11.2 Prescaler

An 8-bit counter is available as a prescaler for the Timer0 module. The prescaler is not readable or writable.

The PSA and T0PS2:T0PS0 bits determine the prescaler assignment and prescale ratio.

Clearing bit PSA will assign the prescaler to the Timer0 module. When the prescaler is assigned to the Timer0 module, prescale values of 1:2, 1:4, ..., 1:256 are selectable.

When assigned to the Timer0 module, all instructions writing to the TMR0 register (e.g., `CLRF TMR0, MOVWF TMR0, BSF TMR0, x`, ..., etc.) will clear the prescaler count.

Note:	Writing to TMR0 when the prescaler is assigned to Timer0 will clear the prescaler count but will not change the prescaler assignment.

11.2.1 SWITCHING PRESCALER ASSIGNMENT

The prescaler assignment is fully under software control (i.e., it can be changed “on-the-fly” during program execution).

11.3 Timer0 Interrupt

The TMR0 interrupt is generated when the TMR0 register overflows from 0FFh to 00h in 8-bit mode, or 0FFFFh to 0000h in 16-bit mode. This overflow sets the TMR0IF bit. The interrupt can be masked by clearing the TMR0IE bit. The TMR0IE bit must be cleared in software by the Timer0 module Interrupt Service Routine before re-enabling this interrupt. The TMR0 interrupt cannot awaken the processor from Sleep since the timer is shut-off during Sleep.

11.4 16-Bit Mode Timer Reads and Writes

TMR0H is not the high byte of the timer/counter in 16-bit mode, but is actually a buffered version of the high byte of Timer0 (refer to Figure 11-2). The high byte of the Timer0 counter/timer is not directly readable nor writable. TMR0H is updated with the contents of the high byte of Timer0 during a read of TMR0L. This provides the ability to read all 16 bits of Timer0 without having to verify that the read of the high and low byte were valid due to a rollover between successive reads of the high and low byte.

A write to the high byte of Timer0 must also take place through the TMR0H buffer register. Timer0 high byte is updated with the contents of TMR0H when a write occurs to TMR0L. This allows all 16 bits of Timer0 to be updated at once.

TABLE 11-1: REGISTERS ASSOCIATED WITH TIMER0

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other Resets
TMR0L	Timer0 Module Low Byte Register								xxxx xxxx	uuuu uuuu
TMR0H	Timer0 Module High Byte Register								0000 0000	0000 0000
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
T0CON	TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0	1111 1111	1111 1111
TRISA	—	PORTA Data Direction Register							-111 1111	-111 1111

Legend: x = unknown, u = unchanged, – = unimplemented locations, read as ‘0’. Shaded cells are not used by Timer0.

NOTES:

12.0 TIMER1 MODULE

The Timer1 module timer/counter has the following features:

- 16-bit timer/counter (two 8-bit registers; TMR1H and TMR1L)
- Readable and writable (both registers)
- Internal or external clock select
- Interrupt on overflow from 0FFFFh to 0000h
- Reset from CCP module special event trigger

Figure 12-1 is a simplified block diagram of the Timer1 module.

Register 12-1 details the Timer1 Control register. This register controls the operating mode of the Timer1 module and contains the Timer1 Oscillator Enable bit (T1OSCEN). Timer1 can be enabled or disabled by setting or clearing control bit, TMR1ON (T1CON<0>).

REGISTER 12-1: T1CON: TIMER1 CONTROL REGISTER

R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
RD16	—	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON
bit 7							bit 0

bit 7 **RD16:** 16-bit Read/Write Mode Enable bit
1 = Enables register read/write of Timer1 in one 16-bit operation
0 = Enables register read/write of Timer1 in two 8-bit operations

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

bit 5-4 **T1CKPS1:T1CKPS0**: Timer1 Input Clock Prescale Select bits
11 = 1:8 prescale value
10 = 1:4 prescale value
01 = 1:2 prescale value
00 = 1:1 prescale value

bit 3 **T1OSCEN:** Timer1 Oscillator Enable bit
1 = Timer1 oscillator is enabled
0 = Timer1 oscillator is shut-off
The oscillator inverter and feedback resistor are turned off to eliminate power drain.

bit 2 **$\overline{\text{T1SYNC}}$:** Timer1 External Clock Input Synchronization Select bit
When TMR1CS = 1:
1 = Do not synchronize external clock input
0 = Synchronize external clock input
When TMR1CS = 0:
This bit is ignored. Timer1 uses the internal clock when TMR1CS = 0.

bit 1 **TMR1CS:** Timer1 Clock Source Select bit
1 = External clock from pin RC0/T1OSO/T13CKI (on the rising edge)
0 = Internal clock (FOSC/4)

bit 0 **TMR1ON:** Timer1 On bit
1 = Enables Timer1
0 = Stops Timer1

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

12.1 Timer1 Operation

Timer1 can operate in one of these modes:

- As a timer
- As a synchronous counter
- As an asynchronous counter

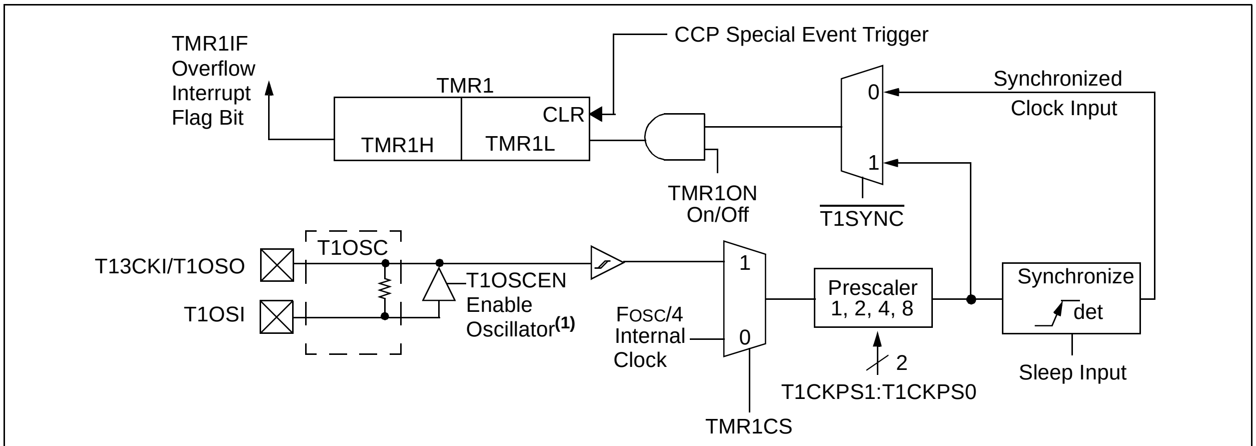
The operating mode is determined by the clock select bit, TMR1CS (T1CON<1>).

When TMR1CS = 0, Timer1 increments every instruction cycle. When TMR1CS = 1, Timer1 increments on every rising edge of the external clock input or the Timer1 oscillator if enabled.

When the Timer1 oscillator is enabled (T1OSCEN is set), the RC1/T1OSI and RC0/T1OSO/T13CKI pins become inputs. That is, the TRISC<1:0> value is ignored and the pins are read as '0'.

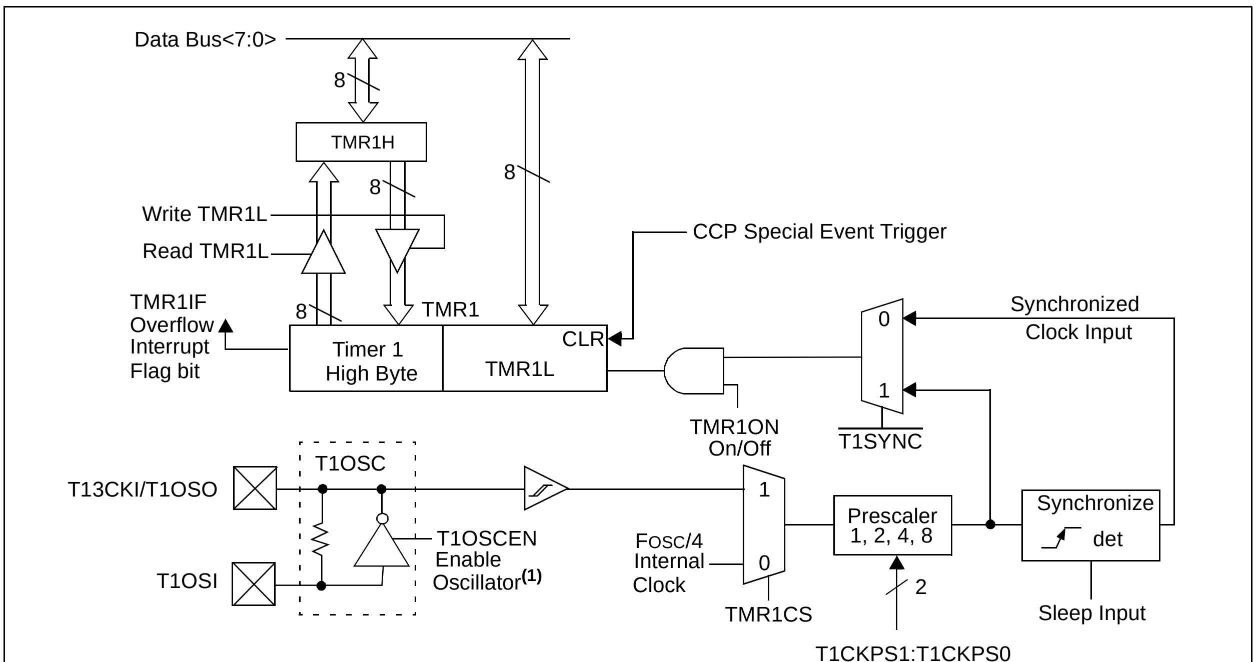
Timer1 also has an internal "Reset input". This Reset can be generated by the CCP module (**Section 15.0 "Capture/Compare/PWM (CCP) Modules"**).

FIGURE 12-1: TIMER1 BLOCK DIAGRAM



Note 1: When enable bit T1OSCEN is cleared, the inverter and feedback resistor are turned off. This eliminates power drain.

FIGURE 12-2: TIMER1 BLOCK DIAGRAM: 16-BIT READ/WRITE MODE



Note 1: When enable bit T1OSCEN is cleared, the inverter and feedback resistor are turned off. This eliminates power drain.

12.2 Timer1 Oscillator

A crystal oscillator circuit is built-in between pins T1OSI (input) and T1OSO (amplifier output). It is enabled by setting control bit, T1OSCEN (T1CON<3>). The oscillator is a low-power oscillator rated up to 200 kHz. It will continue to run during Sleep. It is primarily intended for a 32 kHz crystal. Table 12-1 shows the capacitor selection for the Timer1 oscillator.

The user must provide a software time delay to ensure proper start-up of the Timer1 oscillator.

TABLE 12-1: CAPACITOR SELECTION FOR THE ALTERNATE OSCILLATOR

Osc Type	Freq	C1	C2
LP	32 kHz	TBD(1)	TBD(1)
Crystal to be Tested:			
32.768 kHz	Epson C-001R32.768K-A		± 20 PPM

Note 1: Microchip suggests 33 pF as a starting point in validating the oscillator circuit.

2: Higher capacitance increases the stability of the oscillator but also increases the start-up time.

3: Since each resonator/crystal has its own characteristics, the user should consult the resonator/crystal manufacturer for appropriate values of external components.

4: Capacitor values are for design guidance only.

12.3 Timer1 Interrupt

The TMR1 register pair (TMR1H:TMR1L) increments from 0000h to 0FFFFh and rolls over to 0000h. The TMR1 interrupt, if enabled, is generated on overflow which is latched in interrupt flag bit, TMR1IF (PIR1<0>). This interrupt can be enabled/disabled by setting/clearing TMR1 interrupt enable bit, TMR1IE (PIE1<0>).

12.4 Resetting Timer1 Using a CCP Trigger Output

If the CCP module is configured in Compare mode to generate a “special event trigger” (CCP1M3:CCP1M0 = `1011`), this signal will reset Timer1 and start an A/D conversion (if the A/D module is enabled).

Note: The special event triggers from the CCP1 module will not set interrupt flag bit TMR1IF (PIR1<0>).

Timer1 must be configured for either Timer or Synchronized Counter mode to take advantage of this feature. If Timer1 is running in Asynchronous Counter mode, this Reset operation may not work.

In the event that a write to Timer1 coincides with a special event trigger from CCP1, the write will take precedence.

In this mode of operation, the CCPR1H:CCPR1L register pair effectively becomes the period register for Timer1.

12.5 Timer1 16-Bit Read/Write Mode

Timer1 can be configured for 16-bit reads and writes (see Figure 12-2). When the RD16 control bit (T1CON<7>) is set, the address for TMR1H is mapped to a buffer register for the high byte of Timer1. A read from TMR1L will load the contents of the high byte of Timer1 into the Timer1 high byte buffer. This provides the user with the ability to accurately read all 16 bits of Timer1 without having to determine whether a read of the high byte, followed by a read of the low byte, is valid due to a rollover between reads.

A write to the high byte of Timer1 must also take place through the TMR1H Buffer register. Timer1 high byte is updated with the contents of TMR1H when a write occurs to TMR1L. This allows a user to write all 16 bits to both the high and low bytes of Timer1 at once.

The high byte of Timer1 is not directly readable or writable in this mode. All reads and writes must take place through the Timer1 High Byte Buffer register. Writes to TMR1H do not clear the Timer1 prescaler. The prescaler is only cleared on writes to TMR1L.

TABLE 12-2: REGISTERS ASSOCIATED WITH TIMER1 AS A TIMER/COUNTER

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other Resets
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 0000	0000 0000
PIR1	PSPIF	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
PIE1	PSPIE	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
IPR1	PSPIP	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	0111 1111	0111 1111
TMR1L	Holding Register for the Least Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	uuuu uuuu
TMR1H	Holding Register for the Most Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	uuuu uuuu
T1CON	RD16	—	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON	0-00 0000	u-uu uuuu

Legend: x = unknown, u = unchanged, – = unimplemented, read as ‘0’. Shaded cells are not used by the Timer1 module.

13.0 TIMER2 MODULE

The Timer2 module timer has the following features:

- 8-bit timer (TMR2 register)
- 8-bit period register (PR2)
- Readable and writable (both registers)
- Software programmable prescaler (1:1, 1:4, 1:16)
- Software programmable postscaler (1:1 to 1:16)
- Interrupt on TMR2 match of PR2
- SSP module optional use of TMR2 output to generate clock shift

Timer2 has a control register shown in Register 13-1. Timer2 can be shut-off by clearing control bit, TMR2ON (T2CON<2>), to minimize power consumption. Figure 13-1 is a simplified block diagram of the Timer2 module. Register 13-1 shows the Timer2 Control register. The prescaler and postscaler selection of Timer2 are controlled by this register.

13.1 Timer2 Operation

Timer2 can be used as the PWM time base for the PWM mode of the CCP module. The TMR2 register is readable and writable and is cleared on any device Reset. The input clock (FOSC/4) has a prescale option of 1:1, 1:4 or 1:16, selected by control bits, T2CKPS1:T2CKPS0 (T2CON<1:0>). The match output of TMR2 goes through a 4-bit postscaler (which gives a 1:1 to 1:16 scaling inclusive) to generate a TMR2 interrupt latched in flag bit, TMR2IF (PIR1<1>).

The prescaler and postscaler counters are cleared when any of the following occurs:

- a write to the TMR2 register
- a write to the T2CON register
- any device Reset (Power-on Reset, $\overline{\text{MCLR}}$ Reset, Watchdog Timer Reset, or Brown-out Reset)

TMR2 is not cleared when T2CON is written.

REGISTER 13-1: T2CON: TIMER2 CONTROL REGISTER

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	T2OUTPS3	T2OUTPS2	T2OUTPS1	T2OUTPS0	TMR2ON	T2CKPS1	T2CKPS0
bit 7							bit 0

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

bit 6-3 **T2OUTPS3:T2OUTPS0**: Timer2 Output Postscale Select bits
0000 = 1:1 postscale
0001 = 1:2 postscale
•
•
•
1111 = 1:16 postscale

bit 2 **TMR2ON**: Timer2 On bit
1 = Timer2 is on
0 = Timer2 is off

bit 1-0 **T2CKPS1:T2CKPS0**: Timer2 Clock Prescale Select bits
00 = Prescaler is 1
01 = Prescaler is 4
1x = Prescaler is 16

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

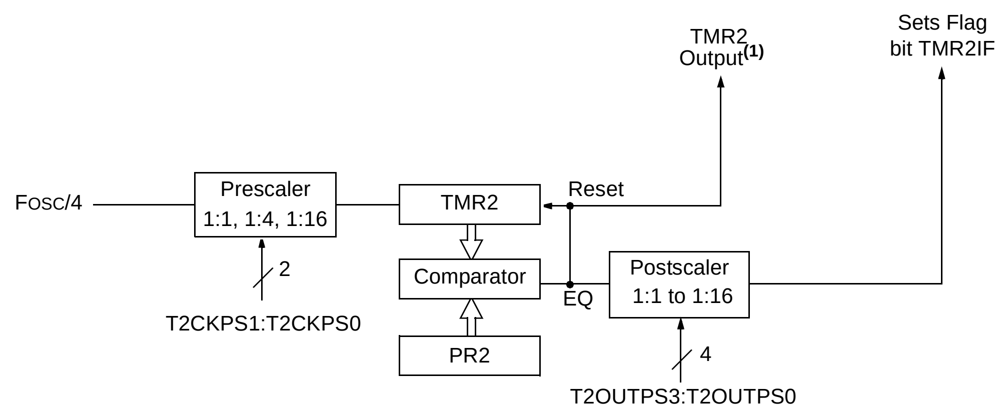
13.2 Timer2 Interrupt

The Timer2 module has an 8-bit period register, PR2. Timer2 increments from 00h until it matches PR2 and then resets to 00h on the next increment cycle. PR2 is a readable and writable register. The PR2 register is initialized to 0FFh upon Reset.

13.3 Output of TMR2

The output of TMR2 (before the postscaler) is fed to the synchronous serial port module which optionally uses it to generate the shift clock.

FIGURE 13-1: TIMER2 BLOCK DIAGRAM



Note 1: TMR2 register output can be software selected by the SSP module as a baud clock.

TABLE 13-1: REGISTERS ASSOCIATED WITH TIMER2 AS A TIMER/COUNTER

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other Resets
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 0000	0000 0000
PIR1	PSPIF	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
PIE1	PSPIE	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
IPR1	PSPIP	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	1111 1111	1111 1111
TMR2	Timer2 Module Register								0000 0000	0000 0000
T2CON	—	T2OUTPS3	T2OUTPS2	T2OUTPS1	T2OUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-000 0000	-000 0000
PR2	Timer2 Period Register								1111 1111	1111 1111

Legend: x = unknown, u = unchanged, – = unimplemented, read as ‘0’. Shaded cells are not used by the Timer2 module.

14.0 TIMER3 MODULE

The Timer3 module timer/counter has the following features:

- 16-bit timer/counter (two 8-bit registers; TMR3H and TMR3L)
- Readable and writable (both registers)
- Internal or external clock select
- Interrupt on overflow from FFFFh to 0000h
- Reset from CCP module trigger

Figure 14-1 is a simplified block diagram of the Timer3 module.

Register 14-1 shows the Timer3 Control register. This register controls the operating mode of the Timer3 module and sets the Enhanced CCP1 and CCP2 clock source.

Register 12-1 shows the Timer1 Control register. This register controls the operating mode of the Timer1 module, as well as containing the Timer1 oscillator enable bit (T1OSCEN) which can be a clock source for Timer3.

REGISTER 14-1: T3CON: TIMER3 CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
RD16	T3CCP2	T3CKPS1	T3CKPS0	T3CCP1	$\overline{\text{T3SYNC}}$	TMR3CS	TMR3ON
bit 7							bit 0

bit 7 **RD16:** 16-bit Read/Write Mode Enable bit
1 = Enables register read/write of Timer3 in one 16-bit operation
0 = Enables register read/write of Timer3 in two 8-bit operations

bit 6, 3 **T3CCP2:T3CCP1:** Timer3 and Timer1 to CCPx Enable bits
1x = Timer3 is the clock source for compare/capture of CCP1 and CCP2 modules
01 = Timer3 is the clock source for compare/capture of CCP2 module,
Timer1 is the clock source for compare/capture of CCP1 module
00 = Timer1 is the clock source for compare/capture of CCP1 and CCP2 modules

bit 5-4 **T3CKPS1:T3CKPS0**: Timer3 Input Clock Prescale Select bits
11 = 1:8 prescale value
10 = 1:4 prescale value
01 = 1:2 prescale value
00 = 1:1 prescale value

bit 2 **$\overline{\text{T3SYNC}}$:** Timer3 External Clock Input Synchronization Control bit
(Not usable if the system clock comes from Timer1/Timer3.)
<u>When TMR3CS = 1:</u>
1 = Do not synchronize external clock input
0 = Synchronize external clock input
<u>When TMR3CS = 0:</u>
This bit is ignored. Timer3 uses the internal clock when TMR3CS = 0.

bit 1 **TMR3CS:** Timer3 Clock Source Select bit
1 = External clock input from Timer1 oscillator or T13CKI
(on the rising edge after the first falling edge)
0 = Internal clock (FOSC/4)

bit 0 **TMR3ON:** Timer3 On bit
1 = Enables Timer3
0 = Stops Timer3

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
- n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

14.1 Timer3 Operation

Timer3 can operate in one of these modes:

- As a timer
- As a synchronous counter
- As an asynchronous counter

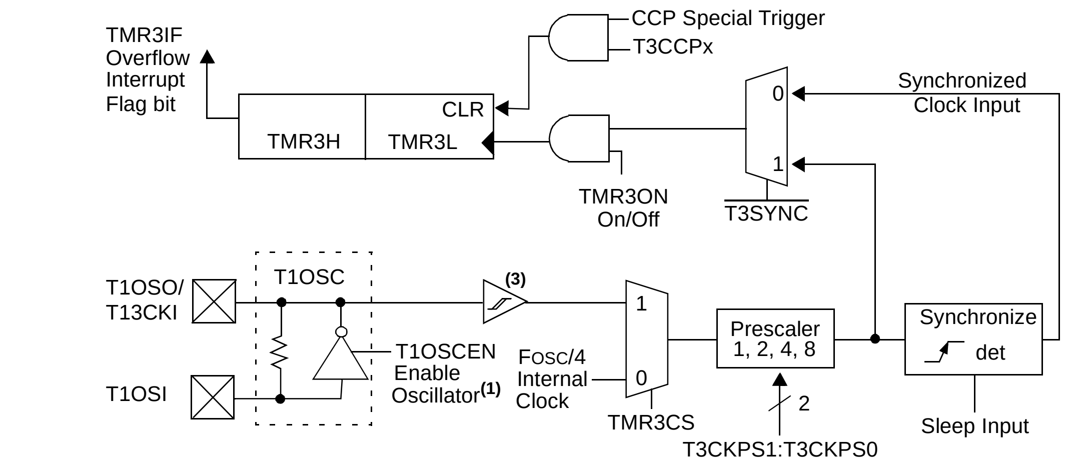
The operating mode is determined by the clock select bit, TMR3CS (T3CON<1>).

When TMR3CS = `0`, Timer3 increments every instruction cycle. When TMR3CS = `1`, Timer3 increments on every rising edge of the Timer1 external clock input or the Timer1 oscillator if enabled.

When the Timer1 oscillator is enabled (T1OSCEN is set), the RC1/T1OSI and RC0/T1OSO/T13CKI pins become inputs. That is, the TRISC<1:0> value is ignored and the pins are read as '`0`'.

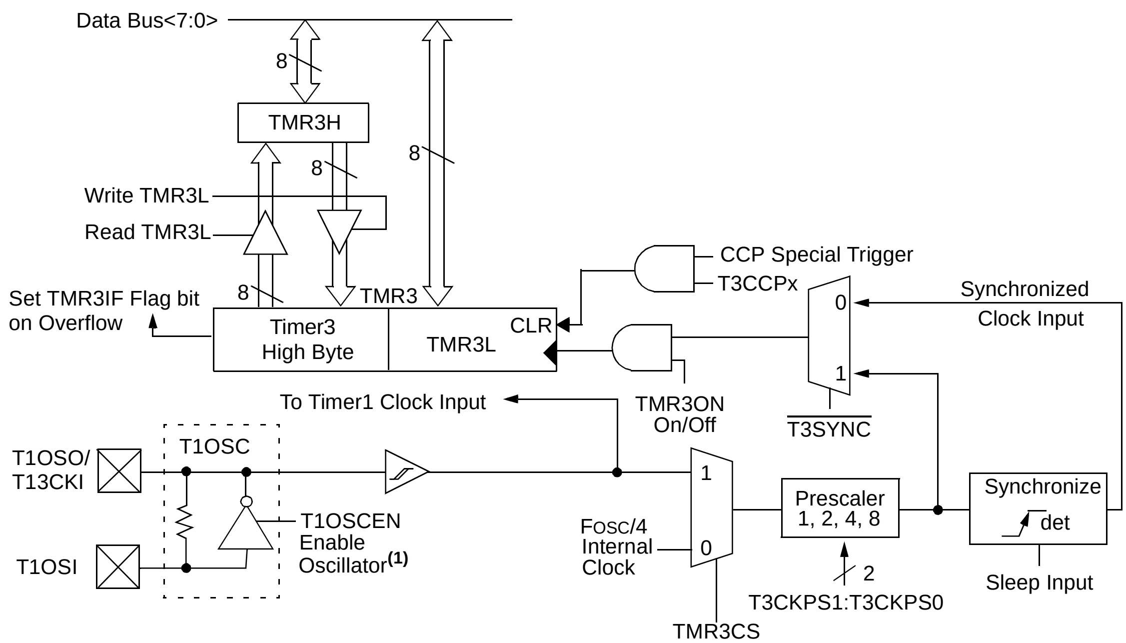
Timer3 also has an internal "Reset input". This Reset can be generated by the CCP module (**Section 14.0 "Timer3 Module"**).

FIGURE 14-1: TIMER3 BLOCK DIAGRAM



Note 1: When enable bit T1OSCEN is cleared, the inverter and feedback resistor are turned off. This eliminates power drain.

FIGURE 14-2: TIMER3 BLOCK DIAGRAM CONFIGURED IN 16-BIT READ/WRITE MODE



Note 1: When the T1OSCEN bit is cleared, the inverter and feedback resistor are turned off. This eliminates power drain.

14.2 Timer1 Oscillator

The Timer1 oscillator may be used as the clock source for Timer3. The Timer1 oscillator is enabled by setting the T1OSCEN (T1CON<3>) bit. The oscillator is a low-power oscillator rated up to 200 kHz. See **Section 12.0 "Timer1 Module"** for further details.

14.3 Timer3 Interrupt

The TMR3 register pair (TMR3H:TMR3L) increments from 0000h to 0FFFFh and rolls over to 0000h. The TMR3 interrupt, if enabled, is generated on overflow which is latched in interrupt flag bit, TMR3IF (PIR2<1>). This interrupt can be enabled/disabled by setting/clearing TMR3 interrupt enable bit, TMR3IE (PIE2<1>).

14.4 Resetting Timer3 Using a CCP Trigger Output

If the CCP module is configured in Compare mode to generate a "special event trigger" (CCP1M3:CCP1M0 = 1011), this signal will reset Timer3.

Note:	The special event triggers from the CCP module will not set interrupt flag bit, TMR3IF (PIR1<0>).

Timer3 must be configured for either Timer or Synchronized Counter mode to take advantage of this feature. If Timer3 is running in Asynchronous Counter mode, this Reset operation may not work. In the event that a write to Timer3 coincides with a special event trigger from CCP1, the write will take precedence. In this mode of operation, the CCPR1H:CCPR1L register pair effectively becomes the period register for Timer3.

TABLE 14-1: REGISTERS ASSOCIATED WITH TIMER3 AS A TIMER/COUNTER

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other Resets
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 0000	0000 0000
PIR2	—	CMIF	—	EEIF	BCLIF	LVDIF	TMR3IF	CCP2IF	-0-0 0000	-0-0 0000
PIE2	—	CMIE	—	EEIE	BCLIE	LVDIE	TMR3IE	CCP2IE	-0-0 0000	-0-0 0000
IPR2	—	CMIP	—	EEIP	BCLIP	LVDIP	TMR3IP	CCP2IP	-1-1 1111	-1-1 1111
TMR3L	Holding Register for the Least Significant Byte of the 16-bit TMR3 Register								xxxx xxxx	uuuu uuuu
TMR3H	Holding Register for the Most Significant Byte of the 16-bit TMR3 Register								xxxx xxxx	uuuu uuuu
T1CON	RD16	—	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON	0-00 0000	u-uu uuuu
T3CON	RD16	T3CCP2	T3CKPS1	T3CKPS0	T3CCP1	$\overline{\text{T3SYNC}}$	TMR3CS	TMR3ON	0000 0000	uuuu uuuu

Legend: x = unknown, u = unchanged, – = unimplemented, read as '0'. Shaded cells are not used by the Timer3 module.

15.0 CAPTURE/COMPARE/PWM (CCP) MODULES

PIC18FXX80/XX85 devices contain a total of two CCP modules: CCP1 and CCP2. CCP1 is an enhanced version of the CCP2 module. CCP1 is fully backward compatible with the CCP2 module.

The CCP1 module differs from CCP2 in the following respect:

- CCP1 contains a special trigger event that may reset Timer1 or the Timer3 register pair
- CCP1 contains “CAN Message Time-Stamp Trigger”
- CCP1 contains enhanced PWM output with programmable dead band and auto-shutdown functionality

Additionally, the CCP2 special event trigger may be used to start an A/D conversion if the A/D module is enabled.

To avoid duplicate information, this section describes basic CCP module operation that applies to both CCP1 and CCP2. Enhanced CCP functionality of the CCP1 module is described in **Section 16.0 “Enhanced Capture/Compare/PWM (ECCP) Module”**.

The control registers for the CCP1 and CCP2 modules are shown in Register 15-1 and Register 15-2, respectively. Table 15-2 details the interactions of the CCP and ECCP modules.

REGISTER 15-1: CCP1CON REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
P1M1	P1M0	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0
bit 7							bit 0

bit 7-6 **P1M1:P1M0:** Enhanced PWM Output Configuration bits
<u>If CCP1M<3:2> = 00, 01, 10:</u>
xx =P1A assigned as capture/compare input; P1B, P1C, P1D assigned as port pins
<u>If CCP1M<3:2> = 11:</u>
00 = Single output; P1A modulated; P1B, P1C, P1D assigned as port pins
01 = Full-bridge output forward; P1D modulated; P1A active; P1B, P1C inactive
10 = Half-bridge output; P1A, P1B modulated with dead-band control; P1C, P1D assigned as port pins
11 = Full-bridge output reverse; P1B modulated; P1C active; P1A, P1D inactive

bit 5-4 **DC1B1:DC1B0**: PWM Duty Cycle bit 1 and bit 0
<u>Capture mode:</u>
Unused.
<u>Compare mode:</u>
Unused.
<u>PWM mode:</u>
These bits are the two LSbs of the 10-bit PWM duty cycle. The eight MSbs of the duty cycle are found in CCPR1L.

bit 3-0 **CCP1M3:CCP1M0**: Enhanced CCP Mode Select bits
0000 = Capture/Compare/PWM off (resets CCP1 module)
0001 = Reserved
0010 = Compare mode, toggle output on match
0011 = Reserved
0100 = Capture mode, every falling edge
0101 = Capture mode, every rising edge
0110 = Capture mode, every 4th rising edge
0111 = Capture mode, every 16th rising edge
1000 = Compare mode, initialize CCP pin low, on compare match force CCP pin high
1001 = Compare mode, initialize CCP pin high, on compare match force CCP pin low
1010 = Compare mode, generate software interrupt only, CCP pin is unaffected
1011 = Compare mode, trigger special event, resets TMR1 or TMR3
1100 = PWM mode; P1A, P1C active-high; P1B, P1D active-high
1101 = PWM mode; P1A, P1C active-high; P1B, P1D active-low
1110 = PWM mode; P1A, P1C active-low; P1B, P1D active-high
1111 = PWM mode; P1A, P1C active-low; P1B, P1D active-low

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
- n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

REGISTER 15-2: CCP2CON REGISTER

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	DC2B1	DC2B0	CCP2M3	CCP2M2	CCP2M1	CCP2M0
bit 7							bit 0

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

bit 5-4 **DC2B1:DC2B0**: PWM Duty Cycle bit 1 and bit 0

Capture mode:
Unused.

Compare mode:
Unused.

PWM mode:
These bits are the two LSbs of the 10-bit PWM duty cycle. The eight MSbs of the duty cycle are found in CCPR2L.

bit 3-0 **CCP2M3:CCP2M0**: CCP2 Mode Select bits

`0000` = Capture/Compare/PWM off (resets CCP2 module)
`0001` = Reserved
`0010` = Compare mode, toggle output on match
`0011` = Reserved
`0100` = Capture mode, every falling edge
`0101` = Capture mode, every rising edge
`0110` = Capture mode, every 4th rising edge
`0111` = Capture mode, every 16th rising edge
`1000` = Compare mode, initialize CCP pin low, on compare match force CCP pin high
`1001` = Compare mode, initialize CCP pin high, on compare match force CCP pin low
`1010` = Compare mode, generate software interrupt only, CCP pin is unaffected
`1011` = Compare mode, trigger special event, resets TMR1 or TMR3 and starts A/D conversion if A/D module is enabled
`11xx` = PWM mode

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

15.1 CCP Module

Both CCP1 and CCP2 are comprised of two 8-bit registers: CCPRxL (low byte) and CCPRxH (high byte), $1 \leq x \leq 2$. The CCPxCON register controls the operation of CCPx. All are readable and writable.

Table 15-1 shows the timer resources of the CCP module modes.

TABLE 15-1: CCP MODE – TIMER RESOURCE

CCP Mode	Timer Resource
Capture	Timer1 or Timer3
Compare	Timer1 or Timer3
PWM	Timer2

15.2 Capture Mode

In Capture mode, CCPRxH:CCPRxL captures the 16-bit value of the TMR1 or TMR3 register when an event occurs on pin CCPn. An event is defined as:

- every falling edge
- every rising edge
- every 4th rising edge
- every 16th rising edge

An event is selected by control bits CCPxM3:CCPxM0 (CCPxCON<3:0>). When a capture is made, the interrupt request flag bit, CCPxIF (PIR registers), is set. It must be cleared in software. If another capture occurs before the value in register CCPRx is read, the old captured value will be lost.

15.2.1 CCP PIN CONFIGURATION

In Capture mode, the CCPx pin should be configured as an input by setting the appropriate TRIS bit.

> **Note:** If the CCPx is configured as an output, a write to the port can cause a capture condition.

15.2.2 TIMER1/TIMER3 MODE SELECTION

The timer used with each CCP module is selected in the T3CCP2:T3CCP1 bits of the T3CON register. The timers used with the capture feature (either Timer1 or Timer3) must be running in Timer mode or Synchronized Counter mode. In Asynchronous Counter mode, the capture operation may not work.

TABLE 15-2: INTERACTION OF CCP MODULES

CCP1 Mode	CCP2 Mode	Interaction
Capture	Capture	TMR1 or TMR3 time base. Time base can be different for each CCP.
Capture	Compare	The compare could be configured for the special event trigger which clears either TMR1 or TMR3 depending upon which time base is used.
Compare	Compare	The compare(s) could be configured for the special event trigger which clears TMR1 or TMR3 depending upon which time base is used.
PWM	PWM	The PWMs will have the same frequency and update rate (TMR2 interrupt).
PWM	Capture	None.
PWM	Compare	None.

15.2.3 SOFTWARE INTERRUPT

When the Capture mode is changed, a false capture interrupt may be generated. The user should keep bit CCPxIE (PIE registers) clear to avoid false interrupts and should clear the flag bit, CCPxIF, following any such change in operating mode.

15.2.4 CCP PRESCALER

There are four prescaler settings specified by bits CCPxM3:CCPxM0. Whenever the CCPx module is turned off, or the CCPx module is not in Capture mode, the prescaler counter is cleared. This means that any Reset will clear the prescaler counter.

Switching from one capture prescaler to another may generate an interrupt. The prescaler counter will not be cleared; therefore, the first capture may be from a non-zero prescaler. Example 15-1 shows the recommended method for switching between capture prescalers. This example also clears the prescaler counter and will not generate the "false" interrupt.

15.2.5 CAN MESSAGE TIME-STAMP

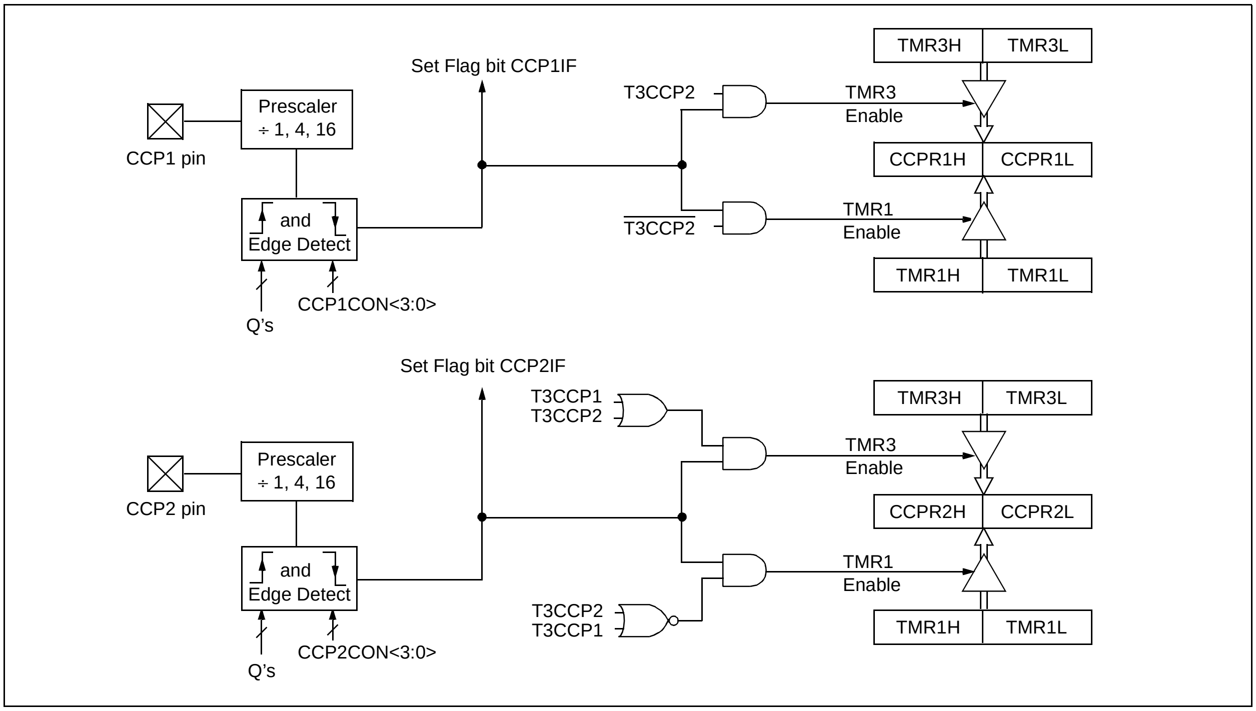
The CAN capture event occurs when a message is received in any of the receive buffers. When configured, the CAN module provides the trigger to the CCP1 module to cause a capture event. This feature is provided to time-stamp the received CAN messages.

This feature is enabled by setting the CANCAP bit of the CAN I/O Control register (CIOCON<4>). The message receive signal from the CAN module then takes the place of the events on RC2/CCP1.

EXAMPLE 15-1: CHANGING BETWEEN CAPTURE PRESCALERS

```
CLRF    CCP1CON       ; Turn CCP module off
MOVLW   NEW_CAPT_PS   ; Load WREG with the
                      ; new prescaler mode
                      ; value and CCP ON
MOVWF   CCP1CON       ; Load CCP1CON with
                      ; this value
```

FIGURE 15-1: CAPTURE MODE OPERATION BLOCK DIAGRAM



15.3 Compare Mode

In Compare mode, the 16-bit CCPRx register value is constantly compared against either the TMR1 register pair value or the TMR3 register pair value. When a match occurs, the CCPx pin can have one of the following actions:

- Driven high
- Driven low
- Toggle output (high-to-low or low-to-high)
- Remains unchanged

The action on the pin is based on the value of control bits, CCPxM3:CCPxM0. At the same time, interrupt flag bit, CCPxIF, is set.

When configured to drive the CCP pin, the CCP1 pin cannot be changed; CCP1 module controls the pin.

15.3.1 CCP PIN CONFIGURATION

The user must configure the CCPx pin as an output by clearing the appropriate TRIS bit.

By default, the CCP2 pin is multiplexed with RC1. Alternately, it can also be multiplexed with either RB3 or RE7. This is done by changing the CCP2MX configuration bit.

Note:	Clearing the CCPxCON register will force the CCPx compare output latch to the default low level. This is not the data latch.

15.3.2 TIMER1/TIMER3 MODE SELECTION

The timer used with each CCP module is selected in the T3CCP2:T3CCP1 bits of the T3CON register. Timer1 and/or Timer3 must be running in Timer mode, or Synchronized Counter mode, if the CCP module is using the compare feature. In Asynchronous Counter mode, the compare operation may not work.

15.3.3 SOFTWARE INTERRUPT MODE

When generate software interrupt is chosen, the CCPx pin is not affected. Only a CCP interrupt is generated (if enabled).

15.3.4 SPECIAL EVENT TRIGGER

In this mode, an internal hardware trigger is generated which may be used to initiate an action.

The special event trigger output of CCP1 resets either the TMR1 or TMR3 register pair. This allows the CCPR1 register to effectively be a 16-bit programmable period register for TMR1 or TMR3.

Additionally, the CCP2 special event trigger will start an A/D conversion if the A/D module is enabled.

Note:	The special event trigger from the CCPx module will not set the Timer1 or Timer3 interrupt flag bits.

FIGURE 15-2: COMPARE MODE OPERATION BLOCK DIAGRAM

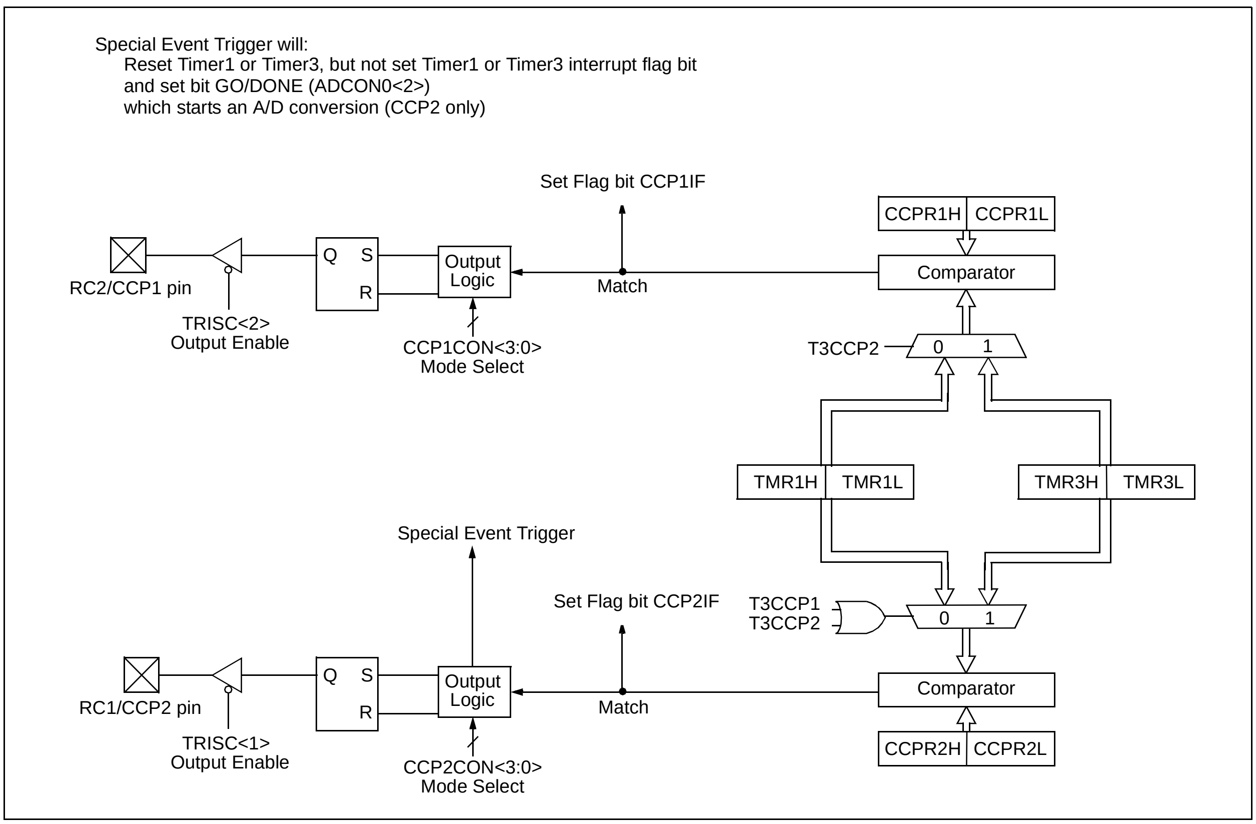


TABLE 15-3: REGISTERS ASSOCIATED WITH CAPTURE, COMPARE, TIMER1 AND TIMER3

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other Resets
INTCON	GIE/ GIEH	PEIE/ GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
PIR1	PSPIF	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
PIE1	PSPIE	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
IPR1	PSPIP	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	1111 1111	1111 1111
TRISD	PORTD Data Direction Register								1111 1111	1111 1111
TMR1L	Holding Register for the Least Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	uuuu uuuu
TMR1H	Holding Register for the Most Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	uuuu uuuu
T1CON	RD16	—	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON	0-00 0000	u-uu uuuu
CCPR1L	Capture/Compare/PWM Register 1 (LSB)								xxxx xxxx	uuuu uuuu
CCPR1H	Capture/Compare/PWM Register 1 (MSB)								xxxx xxxx	uuuu uuuu
CCP1CON	P1M1	P1M0	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0	0000 0000	0000 0000
PIR2	—	CMIF	—	EEIF	BCLIF	LVDIF	TMR3IF	CCP2IF	-0-0 0000	-0-0 0000
PIE2	—	CMIE	—	EEIE	BCLIE	LVDIE	TMR3IE	CCP2IE	-0-0 0000	-0-0 0000
IPR2	—	CMIP	—	EEIP	BCLIP	LVDIP	TMR3IP	CCP2IP	-1-1 1111	-1-1 1111
TMR3L	Holding Register for the Least Significant Byte of the 16-bit TMR3 Register								xxxx xxxx	uuuu uuuu
TMR3H	Holding Register for the Most Significant Byte of the 16-bit TMR3 Register								xxxx xxxx	uuuu uuuu
T3CON	RD16	T3CCP2	T3CKPS1	T3CKPS0	T3CCP1	$\overline{\text{T3SYNC}}$	TMR3CS	TMR3ON	0000 0000	uuuu uuuu

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used by capture and Timer1.

15.4 PWM Mode

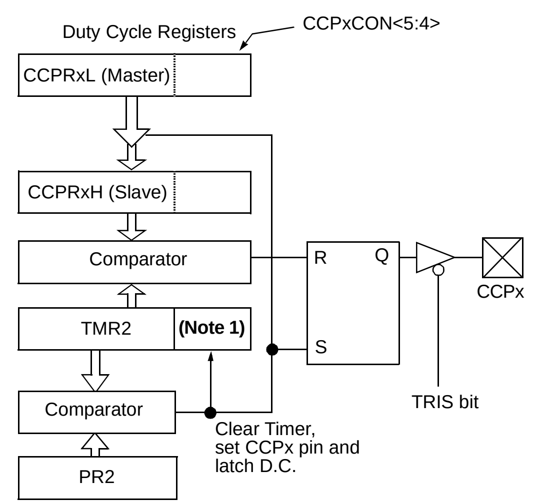
In Pulse Width Modulation (PWM) mode, the CCPx pin produces up to a 10-bit resolution PWM output. For PWM mode to function properly, the TRIS bit for the CCPx pin must be cleared to make it an output.

Note:	Clearing the CCPxCON register will force the CCPx PWM output latch to the default low level. This is not the port data latch.

Figure 15-3 shows a simplified block diagram of the CCP module in PWM mode.

For a step-by-step procedure on how to set up the CCP module for PWM operation, see **Section 15.4.3 "Setup for PWM Operation"**.

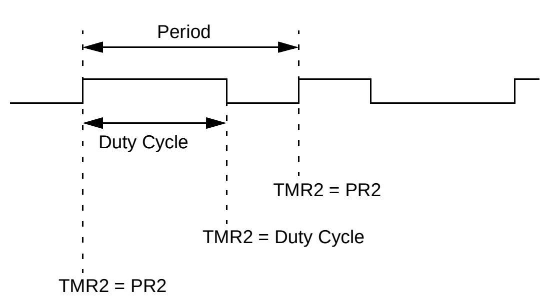
FIGURE 15-3: SIMPLIFIED PWM BLOCK DIAGRAM



Note 1: 8-bit timer is concatenated with 2-bit internal Q clock or 2 bits of the prescaler to create 10-bit time base.

A PWM output (Figure 15-4) has a time base (period) and a time that the output stays high (duty cycle). The frequency of the PWM is the inverse of the period (1/period).

FIGURE 15-4: PWM OUTPUT



15.4.1 PWM PERIOD

The PWM period is specified by writing to the PR2 register. The PWM period can be calculated using the following formula.

EQUATION 15-1:

$$\text{PWM Period} = [(PR2) + 1] \bullet 4 \bullet T_{OSC} \bullet (\text{TMR2 Prescale Value})$$

PWM frequency is defined as 1/[PWM period].

When TMR2 is equal to PR2, the following three events occur on the next increment cycle:

- TMR2 is cleared
- The CCP1 pin is set (exception: if PWM duty cycle = 0%, the CCP1 pin will not be set)
- The PWM duty cycle is latched from CCPR1L into CCPR1H

Note:	The Timer2 postscaler (see **Section 13.0 "Timer2 Module"**) 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.

15.4.2 PWM DUTY CYCLE

The PWM duty cycle is specified by writing to the CCPRxL register and to the CCPxCON<5:4> bits. Up to 10-bit resolution is available. The CCPRxL contains the eight MSbs and the CCPxCON<5:4> contain the two LSbs. This 10-bit value is represented by CCPRxL:CCPxCON<5:4>. The following equation is used to calculate the PWM duty cycle in time.

EQUATION 15-2:

$$\text{PWM Duty Cycle} = (\text{CCPRxL:CCPxCON}<5:4>) \bullet T_{OSC} \bullet (\text{TMR2 Prescale Value})$$

CCPRxL and CCPxCON<5:4> can be written to at any time but the duty cycle value is not latched into CCPRxH until after a match between PR2 and TMR2 occurs (i.e., the period is complete). In PWM mode, CCPRxH is a read-only register.

The CCPRxH register and a 2-bit internal latch are used to double-buffer the PWM duty cycle. This double-buffering is essential for glitchless PWM operation.

When the CCPRxH and 2-bit latch match TMR2, concatenated with an internal 2-bit Q clock or 2 bits of the TMR2 prescaler, the CCPx pin is cleared.

The maximum PWM resolution (bits) for a given PWM frequency is given by the following equation.

EQUATION 15-3:

$$\text{PWM Resolution (max)} = \frac{\log\left(\frac{F_{OSC}}{F_{PWM}}\right)}{\log(2)} \text{bits}$$

Note:	If the PWM duty cycle value is longer than the PWM period, the CCP1 pin will not be cleared.

15.4.3 SETUP FOR PWM OPERATION

The following steps should be taken when configuring the CCP module for PWM operation:

1. Set the PWM period by writing to the PR2 register.
2. Set the PWM duty cycle by writing to the CCPRxL register and CCPxCON<5:4> bits.
3. Make the CCPx pin an output by clearing corresponding TRIS bit.
4. Set the TMR2 prescale value and enable Timer2 by writing to T2CON.
5. Configure the CCPx module for PWM operation.

TABLE 15-4: EXAMPLE PWM FREQUENCIES AND RESOLUTIONS AT 40 MHz

PWM Frequency	2.44 kHz	9.76 kHz	39.06 kHz	156.3 kHz	312.5 kHz	416.6 kHz
Timer Prescaler (1, 4, 16)	16	4	1	1	1	1
PR2 Value	0FFh	0FFh	0FFh	3Fh	1Fh	17h
Maximum Resolution (bits)	10	10	10	8	7	5.5

TABLE 15-5: REGISTERS ASSOCIATED WITH PWM AND TIMER2

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other Resets
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
PIR1	PSPIF	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
PIE1	PSPIE	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
IPR1	PSPIP	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	1111 1111	1111 1111
TRISC	PORTC Data Direction Register								1111 1111	1111 1111
TMR2	Timer2 Module Register								0000 0000	0000 0000
PR2	Timer2 Module Period Register								1111 1111	1111 1111
T2CON	—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-000 0000	-000 0000
CCPR1L	Capture/Compare/PWM Register 1 (LSB)								xxxx xxxx	uuuu uuuu
CCPR1H	Capture/Compare/PWM Register 1 (MSB)								xxxx xxxx	uuuu uuuu
CCP1CON	P1M1	P1M0	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0	0000 0000	0000 0000
CCPR2L	Capture/Compare/PWM Register 2 (LSB)								xxxx xxxx	uuuu uuuu
CCPR2H	Capture/Compare/PWM Register 2 (MSB)								xxxx xxxx	uuuu uuuu
CCP2CON	—	—	DC2B1	DC2B0	CCP2M3	CCP2M2	CCP2M1	CCP2M0	--00 0000	--00 0000

Legend: x = unknown, u = unchanged, – = unimplemented, read as '0'. Shaded cells are not used by PWM and Timer2.

16.0 ENHANCED CAPTURE/ COMPARE/PWM (ECCP) MODULE

The CCP1 module is implemented as a standard CCP module with enhanced PWM capabilities. These capabilities allow for 2 or 4 output channels, user selectable polarity, dead-band control, and automatic shutdown and restart and are discussed in detail in **Section 16.2 "Enhanced PWM Mode"**.

The control register for CCP1 is shown in Register 16-1.

In addition to the expanded functions of the CCP1CON register, the CCP1 module has two additional registers associated with enhanced PWM operation and auto-shutdown features:

- ECCP1DEL
- ECCP1AS

REGISTER 16-1: CCP1CON REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
P1M1	P1M0	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0
bit 7							bit 0

bit 7-6 **P1M1:P1M0:** Enhanced PWM Output Configuration bits

If CCP1M<3:2> = `00`, `01`, `10`:
`xx` = P1A assigned as capture/compare input; P1B, P1C, P1D assigned as port pins

If CCP1M<3:2> = `11`:
`00` = Single output; P1A modulated, P1B, P1C, P1D assigned as port pins
`01` = Full-bridge output forward; P1D modulated; P1A active; P1B, P1C inactive
`10` = Half-bridge output; P1A, P1B modulated with dead-band control; P1C, P1D assigned as port pins
`11` = Full-bridge output reverse; P1B modulated; P1C active; P1A, P1D inactive

bit 5-4 **DC1B1:DC1B0**: PWM Duty Cycle bit 1 and bit 0

Capture mode:
Unused.
Compare mode:
Unused.
PWM mode:
These bits are the two LSbs of the 10-bit PWM duty cycle. The eight MSbs of the duty cycle are found in CCPR1L.

bit 3-0 **CCP1M3:CCP1M0**: Enhanced CCP Mode Select bits

`0000` = Capture/Compare/PWM off (resets CCP1 module)
`0001` = Reserved
`0010` = Compare mode, toggle output on match
`0011` = Capture mode, CAN message time-stamp
`0100` = Capture mode, every falling edge
`0101` = Capture mode, every rising edge
`0110` = Capture mode, every 4th rising edge
`0111` = Capture mode, every 16th rising edge
`1000` = Compare mode, initialize CCP pin low, on compare match, force CCP pin high
`1001` = Compare mode, initialize CCP pin high, on compare match, force CCP pin low
`1010` = Compare mode, generate software interrupt only, CCP pin is unaffected
`1011` = Compare mode, trigger special event, resets TMR1 or TMR3
`1100` = PWM mode; P1A, P1C active-high; P1B, P1D active-high
`1101` = PWM mode; P1A, P1C active-high; P1B, P1D active-low
`1110` = PWM mode; P1A, P1C active-low; P1B, P1D active-high
`1111` = PWM mode; P1A, P1C active-low; P1B, P1D active-low

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

16.1 ECCP Outputs

The enhanced CCP module may have up to four outputs depending on the selected operating mode. These outputs, designated P1A through P1D, are multiplexed with I/O pins RC2, RE6, RE5 and RG4. The pin assignments are summarized in Table 16-1.

To configure I/O pins as PWM outputs, the proper PWM mode must be selected by setting the P1Mx and CCP1Mx bits (CCP1CON<7:6> and <3:0>, respectively). The appropriate TRIS direction bits for the port pins must also be set as outputs.

TABLE 16-1: PIN ASSIGNMENTS FOR VARIOUS ECCP MODES

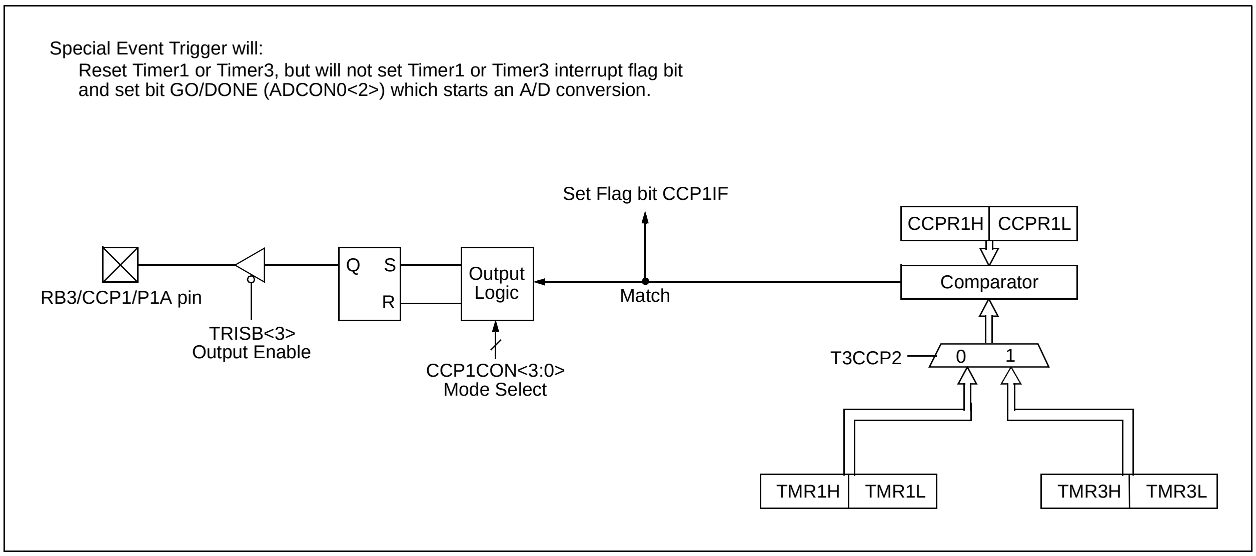
ECCP Mode	CCP1CON Configuration	RC2	RE6	RE5	RG4
Compatible CCP	00xx11xx	CCP1	RE6	RE5	RG4
Dual PWM	10xx11xx	P1A	P1B(2)	RE5	RG4
Quad PWM	x1xx11xx	P1A	P1B(2)	P1C(2)	P1D

Legend: x = Don't care. Shaded cells indicate pin assignments not used by ECCP in a given mode.

Note 1: TRIS register values must be configured appropriately.

2: On PIC18F8X8X devices, these pins can be alternately multiplexed with RH7 or RH6 by changing the ECCPMX configuration bit.

FIGURE 16-1: COMPARE MODE OPERATION BLOCK DIAGRAM



16.2 Enhanced PWM Mode

The Enhanced PWM mode provides additional PWM output options for a broader range of control applications. The module is a backward compatible version of the standard CCP module and offers up to four outputs, designated P1A through P1D. Users are also able to select the polarity of the signal (either active-high or active-low). The module's output mode and polarity are configured by setting the P1M1:P1M0 and CCP1M3:CCP1M0 bits of the CCP1CON register (CCP1CON<7:6> and CCP1CON<3:0>, respectively).

Figure 16-2 shows a simplified block diagram of PWM operation. All control registers are double-buffered and are loaded at the beginning of a new PWM cycle (the period boundary when Timer2 resets) in order to prevent glitches on any of the outputs. The exception is the PWM Delay register, ECCP1DEL, which is loaded at either the duty cycle boundary or the boundary period (whichever comes first). Because of the buffering, the module waits until the assigned timer resets instead of starting immediately. This means that enhanced PWM waveforms do not exactly match the standard PWM waveforms, but are instead offset by one full instruction cycle (4 TOSC).

As before, the user must manually configure the appropriate TRIS bits for output.

16.2.1 PWM PERIOD

The PWM period is specified by writing to the PR2 register. The PWM period can be calculated using the following equation.

EQUATION 16-1:

$$\text{PWM Period} = [(\text{PR2}) + 1] \bullet 4 \bullet T_{OSC} \bullet (\text{TMR2 Prescale Value})$$

PWM frequency is defined as 1/[PWM period]. When TMR2 is equal to PR2, the following three events occur on the next increment cycle:

- TMR2 is cleared
- The CCP1 pin is set (if PWM duty cycle = 0%, the CCP1 pin will not be set)
- The PWM duty cycle is copied from CCPR1L into CCPR1H

Note:	The Timer2 postscaler (see **Section 13.0 "Timer2 Module"**) 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.

16.2.2 PWM DUTY CYCLE

The PWM duty cycle is specified by writing to the CCPR1L register and to the CCP1CON<5:4> bits. Up to 10-bit resolution is available. The CCPR1L contains the eight MSbs and the CCP1CON<5:4> contains the two LSbs. This 10-bit value is represented by CCPR1L:CCP1CON<5:4>. The PWM duty cycle is calculated by the following equation.

EQUATION 16-2:

$$\text{PWM Duty Cycle} = (\text{CCPR1L:CCP1CON<5:4>}) \bullet T_{OSC} \bullet (\text{TMR2 Prescale Value})$$

CCPR1L and CCP1CON<5:4> can be written to at any time, but the duty cycle value is not copied into CCPR1H until a match between PR2 and TMR2 occurs (i.e., the period is complete). In PWM mode, CCPR1H is a read-only register.

The CCPR1H register and a 2-bit internal latch are used to double-buffer the PWM duty cycle. This double-buffering is essential for glitchless PWM operation. When the CCPR1H and 2-bit latch match TMR2, concatenated with an internal 2-bit Q clock or two bits of the TMR2 prescaler, the CCP1 pin is cleared. The maximum PWM resolution (bits) for a given PWM frequency is given by the following equation:

EQUATION 16-3:

$$\text{PWM Resolution (max)} = \frac{\log\left(\frac{F_{OSC}}{F_{PWM}}\right)}{\log(2)} \text{ bits}$$

Note:	If the PWM duty cycle value is longer than the PWM period, the CCP1 pin will not be cleared.

16.2.3 PWM OUTPUT CONFIGURATIONS

The P1M1:P1M0 bits in the CCP1CON register allow one of four configurations:

- Single Output
- Half-Bridge Output
- Full-Bridge Output, Forward mode
- Full-Bridge Output, Reverse mode

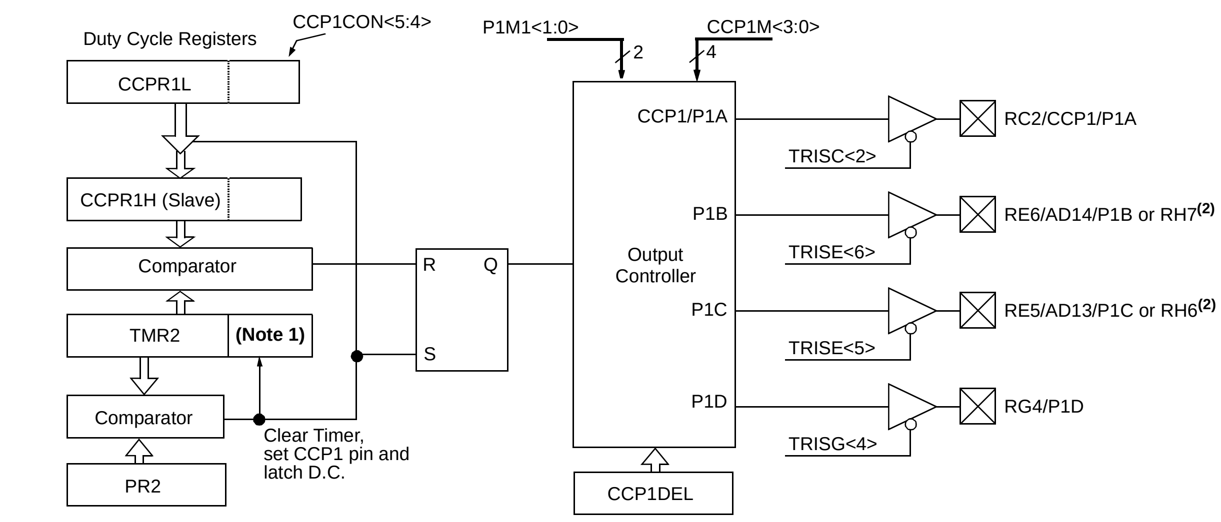
The Single Output mode is the standard PWM mode discussed in **Section 16.2 "Enhanced PWM Mode"**. The Half-Bridge and Full-Bridge Output modes are covered in detail in the sections that follow.

The general relationship of the outputs in all configurations is summarized in Figure 16-3.

TABLE 16-2: EXAMPLE PWM FREQUENCIES AND RESOLUTIONS AT 40 MHz

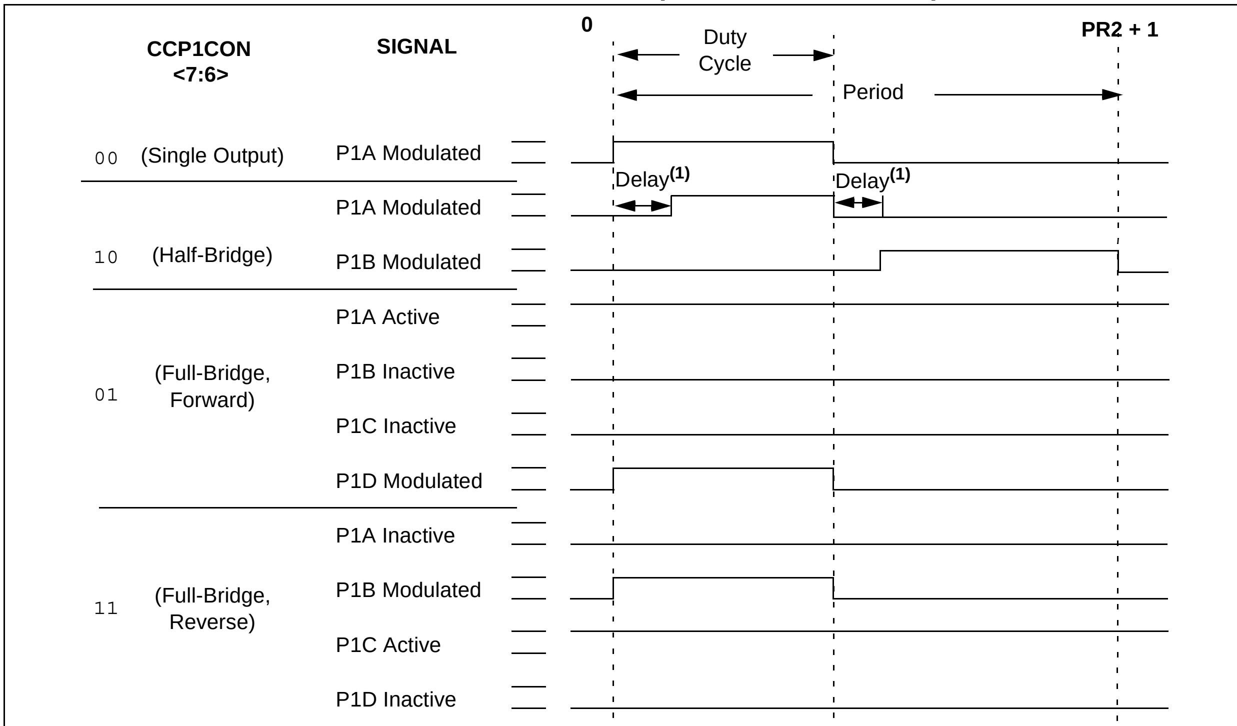
PWM Frequency	2.44 kHz	9.77 kHz	39.06 kHz	156.25 kHz	312.50 kHz	416.67 kHz
Timer Prescaler (1, 4, 16)	16	4	1	1	1	1
PR2 Value	FFh	FFh	FFh	3Fh	1Fh	17h
Maximum Resolution (bits)	10	10	10	8	7	6.58

FIGURE 16-2: SIMPLIFIED BLOCK DIAGRAM OF THE ENHANCED PWM MODULE



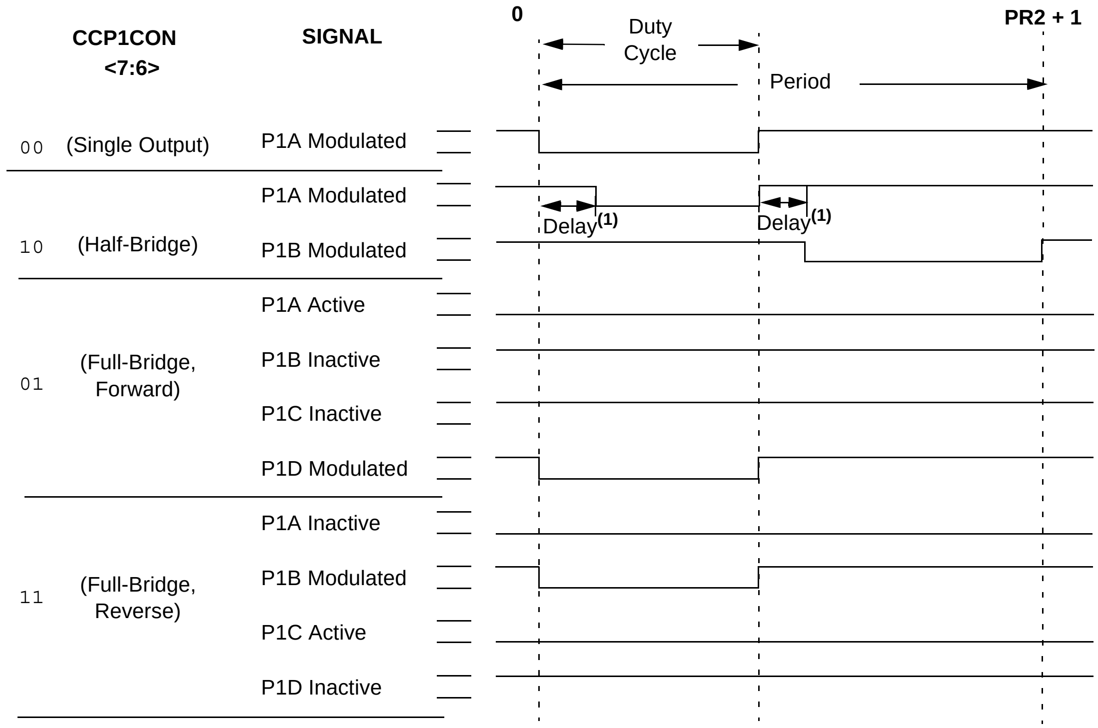
Note 1: The 8-bit timer TMR2 register is concatenated with the 2-bit internal Q clock or 2 bits of the prescaler to create the 10-bit time base.
2: Alternate setting controlled by the ECCPMX bit (PIC18F8X8X devices only).

FIGURE 16-3: PWM OUTPUT RELATIONSHIPS (ACTIVE-HIGH STATE)



Note 1: Dead-band delay is programmed using the ECCP1DEL register (**Section 16.2.6 "Programmable Dead-Band Delay"**).

FIGURE 16-4: PWM OUTPUT RELATIONSHIPS (ACTIVE-LOW STATE)



Note 1: Dead-band delay is programmed using the ECCP1DEL register (**Section 16.2.6 "Programmable Dead-Band Delay"**).

Relationships:

- Period = 4 * TOSC * (PR2 + 1) * (TMR2 prescale value)
- Duty Cycle = TOSC * (CCPR1L<7:0>:CCP1CON<5:4>) * (TMR2 prescale value)
- Delay = 4 * TOSC * (PWM1CON<6:0>)

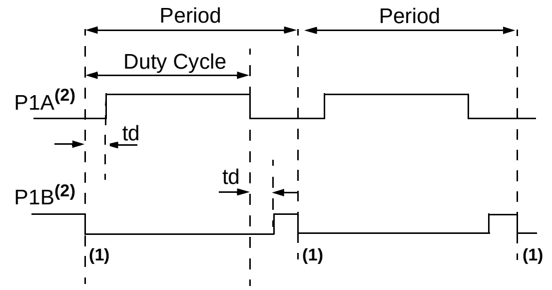
16.2.4 HALF-BRIDGE MODE

In the Half-Bridge Output mode, two pins are used as outputs to drive push-pull loads. The PWM output signal is output on the P1A pin while the complementary PWM output signal is output on the P1B pin (Figure 16-5). This mode can be used for half-bridge applications, as shown in Figure 16-6, or for full-bridge applications where four power switches are being modulated with two PWM signals.

In Half-Bridge Output mode, the programmable dead-band delay can be used to prevent shoot-through current in half-bridge power devices. The value of bits PDC6:PDC0 sets the number of instruction cycles before the output is driven active. If the value is greater than the duty cycle, the corresponding output remains inactive during the entire cycle. See **Section 16.2.6 “Programmable Dead-Band Delay”** for more details of the dead-band delay operations.

Since the P1A and P1B outputs are multiplexed with the PORTC<2> and PORTE<6> data latches, the TRISC<2> and TRISE<6> bits must be cleared to configure P1A and P1B as outputs.

FIGURE 16-5: HALF-BRIDGE PWM OUTPUT

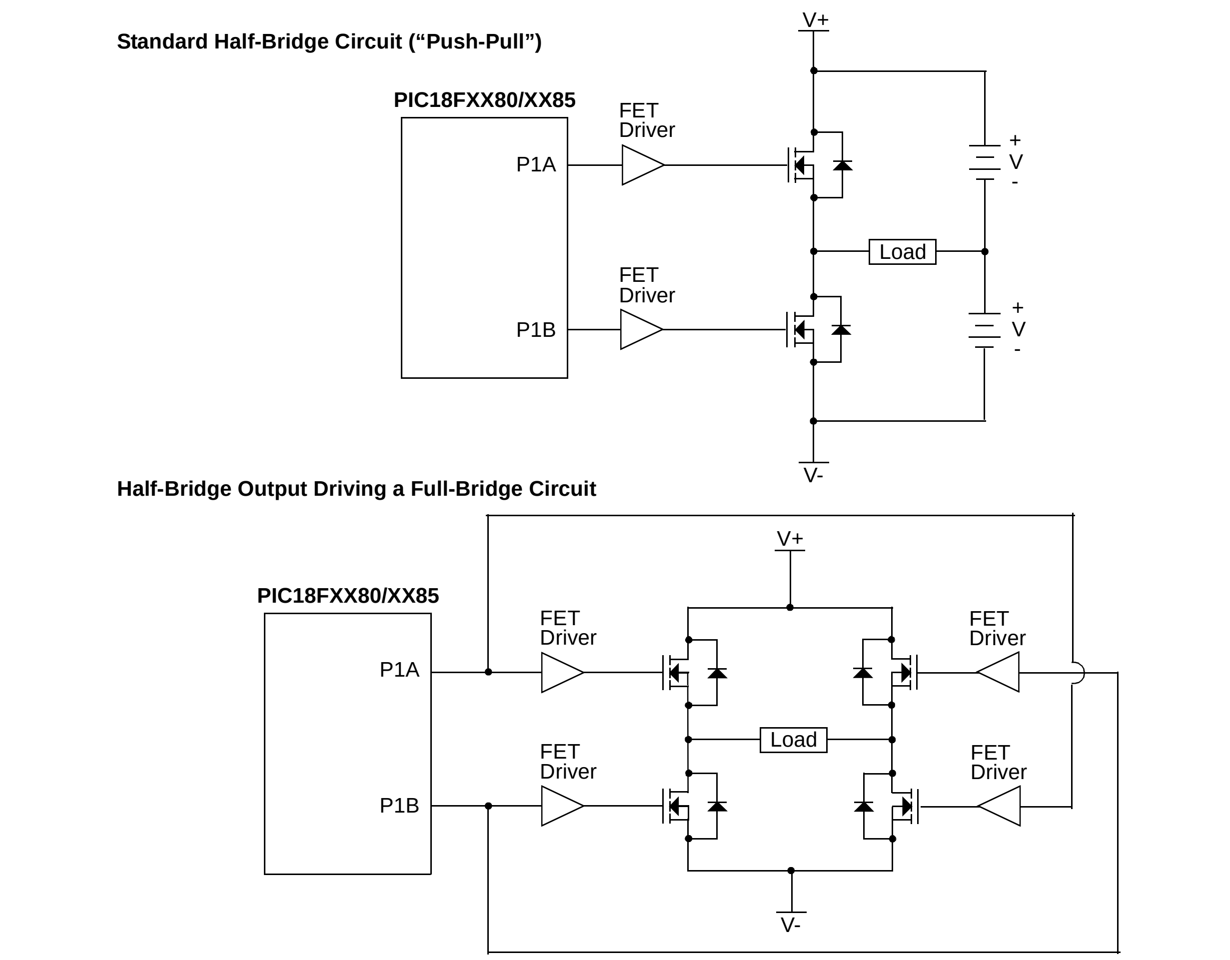


td = Dead-band Delay

Note 1: At this time, the TMR2 register is equal to the PR2 register.

2: Output signals are shown as active-high.

FIGURE 16-6: EXAMPLES OF HALF-BRIDGE OUTPUT MODE APPLICATIONS

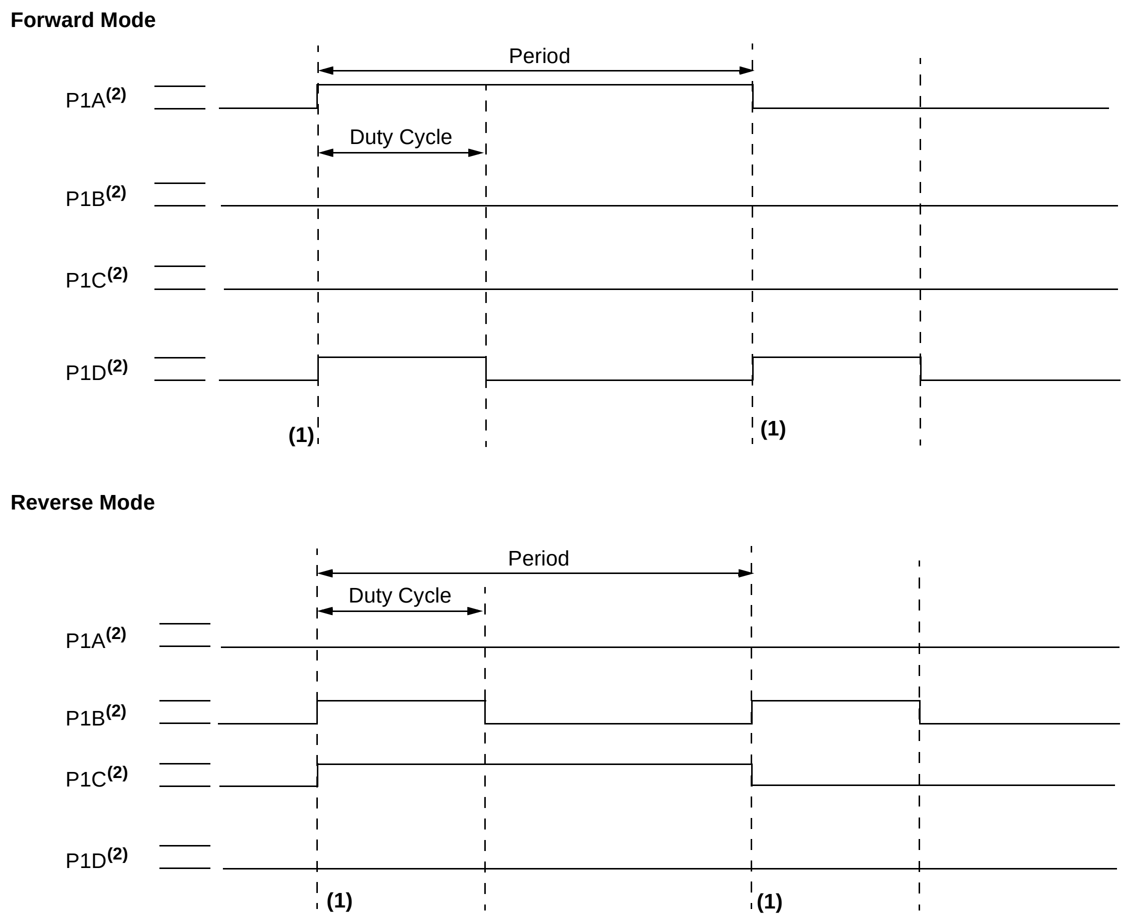


16.2.5 FULL-BRIDGE MODE

In Full-Bridge Output mode, four pins are used as outputs; however, only two outputs are active at a time. In the Forward mode, pin P1A is continuously active and pin P1D is modulated. In the Reverse mode, pin PGC is continuously active and pin P1B is modulated. These are illustrated in Figure 16-7.

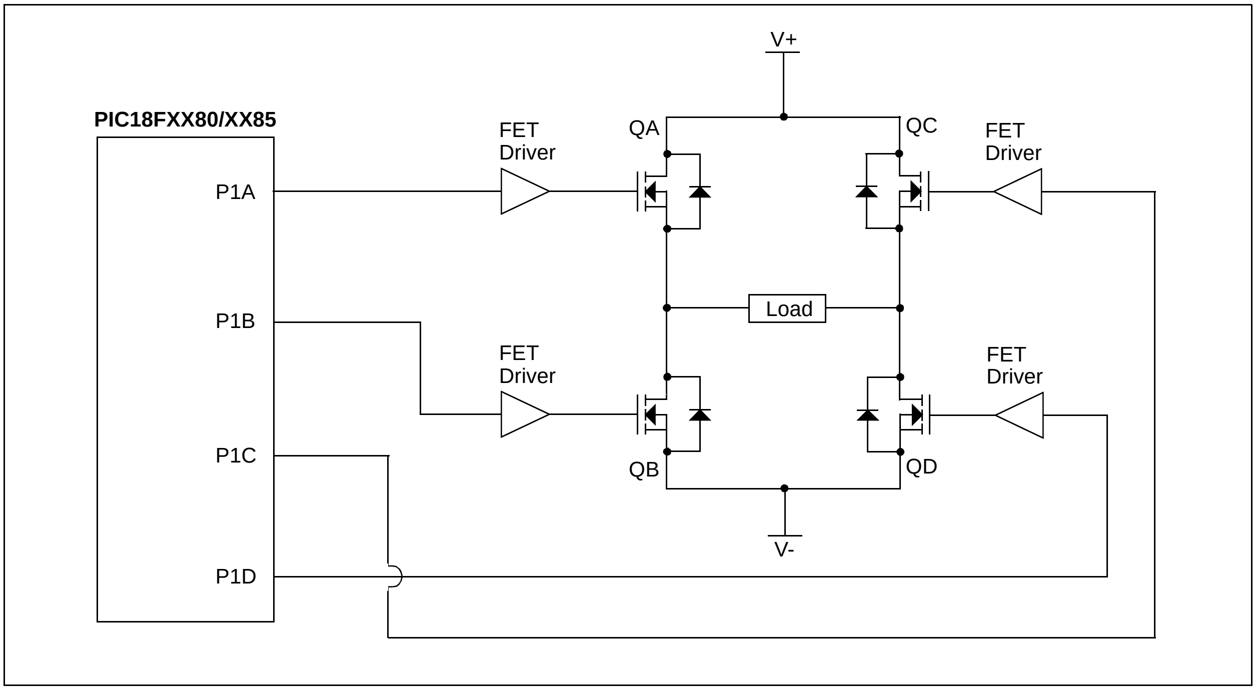
P1A, P1B, P1C and P1D outputs are multiplexed with the PORTC<2>, PORTE<6:5> and PORTG<4> data latches. The TRISC<2>, TRISC<6:5> and TRISG<4> bits must be cleared to make the P1A, P1B, P1C and P1D pins outputs.

FIGURE 16-7: FULL-BRIDGE PWM OUTPUT



Note 1: At this time, the TMR2 register is equal to the PR2 register.
2: Output signal is shown as active-high.

FIGURE 16-8: EXAMPLE OF FULL-BRIDGE APPLICATION



16.2.5.1 Direction Change in Full-Bridge Mode

In the Full-Bridge Output mode, the P1M1 bit in the CCP1CON register allows the user to control the forward/reverse direction. When the application firmware changes this direction control bit, the module will assume the new direction on the next PWM cycle.

Just before the end of the current PWM period, the modulated outputs (P1B and P1D) are placed in their inactive state while the unmodulated outputs (P1A and P1C) are switched to drive in the opposite direction. This occurs in a time interval of (4 TOSC * (Timer2 Prescale value)) before the next PWM period begins. The Timer2 prescaler will be either 1, 4 or 16, depending on the value of the T2CKPS bit (T2CON<1:0>). During the interval from the switch of the unmodulated outputs to the beginning of the next period, the modulated outputs (P1B and P1D) remain inactive. This relationship is shown in Figure 16-9.

Note that in the Full-Bridge Output mode, the CCP1 module does not provide any dead-band delay. In general, since only one output is modulated at all times, dead-band delay is not required. However, there is a situation where a dead-band delay might be required. This situation occurs when both of the following conditions are true:

1. The direction of the PWM output changes when the duty cycle of the output is at or near 100%.
2. The turn off time of the power switch, including the power device and driver circuit, is greater than the turn on time.

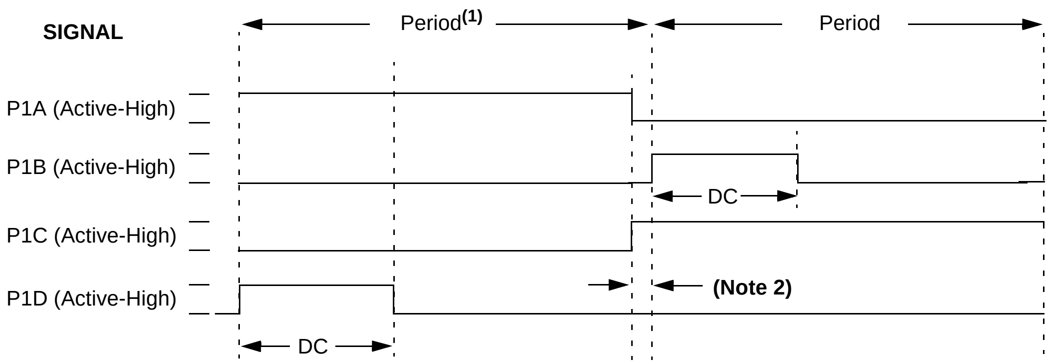
Figure 16-10 shows an example where the PWM direction changes from forward to reverse at a near 100% duty cycle. At time t1, the output P1A and P1D become inactive while output P1C becomes active. In this example, since the turn off time of the power devices is longer than the turn on time, a shoot-through current may flow through power devices QC and QD (see Figure 16-8) for the duration of ‘t’. The same phenomenon will occur to power devices QA and QB for PWM direction change from reverse to forward.

If changing PWM direction at high duty cycle is required for an application, one of the following requirements must be met:

1. Reduce PWM for a PWM period before changing directions.
2. Use switch drivers that can drive the switches off faster than they can drive them on.

Other options to prevent shoot-through current may exist.

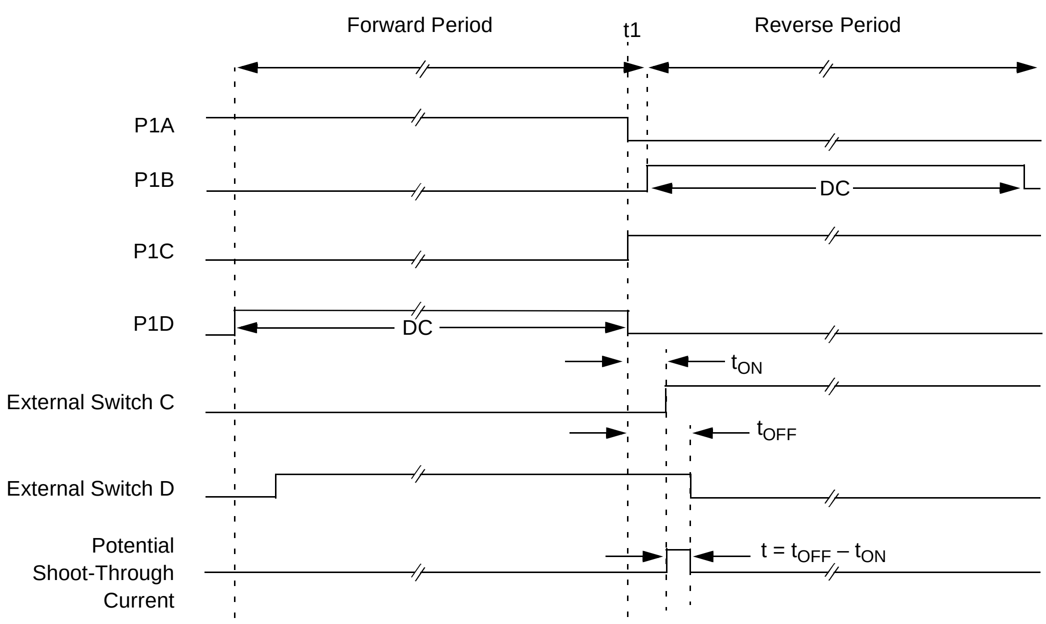
FIGURE 16-9: PWM DIRECTION CHANGE



Note 1: The direction bit in the CCP1 Control register (CCP1CON<7>) is written any time during the PWM cycle.

2: When changing directions, the P1A and P1C signals switch before the end of the current PWM cycle at intervals of 4 TOSC, 16 TOSC or 64 TOSC, depending on the Timer2 prescaler value. The modulated P1B and P1D signals are inactive at this time.

FIGURE 16-10: PWM DIRECTION CHANGE AT NEAR 100% DUTY CYCLE



Note 1: All signals are shown as active-high.

2: t_{ON} is the turn on delay of power switch QC and its driver.

3: t_{OFF} is the turn off delay of power switch QD and its driver.

16.2.6 PROGRAMMABLE DEAD-BAND DELAY

In half-bridge applications where all power switches are modulated at the PWM frequency at all times, the power switches normally require more time to turn off than to turn on. If both the upper and lower power switches are switched at the same time (one turned on and the other turned off), both switches may be on for a short period of time until one switch completely turns off. During this brief interval, a very high current (shoot-through current) may flow through both power switches, shorting the bridge supply. To avoid this potentially destructive shoot-through current from flowing during switching, turning on either of the power switches is normally delayed to allow the other switch to completely turn off.

In the Half-Bridge Output mode, a digitally programmable dead-band delay is available to avoid shoot-through current from destroying the bridge power switches. The delay occurs at the signal transition from the non-active state to the active state. See Figure 16-5 for an illustration. The lower seven bits of the ECCP1DEL register (Register 16-2) set the delay period in terms of microcontroller instruction cycles (T_{CY} or 4 T_{OSC}).

16.2.7 ENHANCED PWM AUTO-SHUTDOWN

When the CCP1 is programmed for any of the enhanced PWM modes, the active output pins may be configured for auto-shutdown. Auto-shutdown immediately places the enhanced PWM output pins into a defined shutdown state when a shutdown event occurs.

A shutdown event can be caused by either of the two comparator modules or a low level on the RB0 pin (or any combination of these three sources). The comparators may be used to monitor a voltage input proportional to a current being monitored in the bridge circuit. If the voltage exceeds a threshold, the comparator switches state and triggers a shutdown. Alternatively, a low digital signal on the RB0 pin can also trigger a shutdown. The auto-shutdown feature can be disabled by not selecting any auto-shutdown sources. The auto-shutdown sources to be used are selected using the ECCPAS2:ECCPAS0 bits (bits <6:4> of the ECCP1AS register).

When a shutdown occurs, the output pins are asynchronously placed in their shutdown states, specified by the PSSAC1:PSSAC0 and PSSBD1:PSSBD0 bits (ECCP1AS<3:0>). Each pin pair (P1A/P1C and P1B/P1D) may be set to drive high, drive low, or be tri-stated (not driving). The ECCPASE bit (ECCP1AS<7>) is also set to hold the enhanced PWM outputs in their shutdown states.

The ECCPASE bit is set by hardware when a shutdown event occurs. If automatic restarts are not enabled, the ECCPASE bit is cleared by firmware when the cause of the shutdown clears. If automatic restarts are enabled, the ECCPASE bit is automatically cleared when the cause of the auto-shutdown has cleared.

If the ECCPASE bit is set when a PWM period begins, the PWM outputs remain in their shutdown state for that entire PWM period. When the ECCPASE bit is cleared, the PWM outputs will return to normal operation at the beginning of the next PWM period.

Note:	Writing to the ECCPASE bit is disabled while a shutdown condition is active.

REGISTER 16-2: ECCP1DEL: ECCP1 DELAY REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PRSEN	PDC6	PDC5	PDC4	PDC3	PDC2	PDC1	PDC0
bit 7							bit 0

bit 7 **PRSEN:** PWM Restart Enable bit
1 = Upon auto-shutdown, the ECCPASE bit clears automatically once the shutdown event goes away; the PWM restarts automatically
0 = Upon auto-shutdown, ECCPASE must be cleared in software to restart the PWM

bit 6-0 **PDC<6:0>:** PWM Delay Count bits
Number of F_{OSC}/4 (4 * T_{OSC}) cycles between the scheduled time when a PWM signal should transition active and the actual time it transitions active.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
- n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

REGISTER 16-3: ECCP1AS: ENHANCED CAPTURE/COMPARE/PWM AUTO-SHUTDOWN CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ECCPASE	ECCPAS2	ECCPAS1	ECCPAS0	PSSAC1	PSSAC0	PSSBD1	PSSBD0
bit 7							bit 0

bit 7 **ECCPASE:** ECCP Auto-Shutdown Event Status bit

`0` = ECCP outputs are operating
`1` = A shutdown event has occurred; ECCP outputs are in shutdown state

bit 6-4 **ECCPAS<2:0>:** ECCP Auto-Shutdown Source Select bits

`000` = Auto-shutdown is disabled
`001` = Comparator 1 output
`010` = Comparator 2 output
`011` = Either Comparator 1 or 2
`100` = RB0
`101` = RB0 or Comparator 1
`110` = RB0 or Comparator 2
`111` = RB0 or Comparator 1 or Comparator 2

bit 3-2 **PSSACn:** Pins A and C Shutdown State Control bits

`00` = Drive pins A and C to '0'
`01` = Drive pins A and C to '1'
`1x` = Pins A and C tri-state

bit 1-0 **PSSBDn:** Pins B and D Shutdown State Control bits

`00` = Drive pins B and D to '0'
`01` = Drive pins B and D to '1'
`1x` = Pins B and D tri-state

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

16.2.7.1 Auto-Shutdown and Automatic Restart

The auto-shutdown feature can be configured to allow automatic restarts of the module following a shutdown event. This is enabled by setting the PRSEN bit of the ECCP1DEL register (ECCP1DEL<7>).

In Shutdown mode with PRSEN = `1` (Figure 16-11), the ECCPASE bit will remain set for as long as the cause of the shutdown continues. When the shutdown condition clears, the ECCPASE bit is cleared. If PRSEN = `0` (Figure 16-12), once a shutdown condition occurs, the ECCPASE bit will remain set until it is cleared by firmware. Once ECCPASE is cleared, the enhanced PWM will resume at the beginning of the next PWM period.

Note:	Writing to the ECCPASE bit is disabled while a shutdown condition is active.

Independent of the PRSEN bit setting, if the auto-shutdown source is one of the comparators, the shutdown condition is a level. The ECCPASE bit cannot be cleared as long as the cause of the shutdown persists.

The Auto-Shutdown mode can be forced by writing a '`1`' to the ECCPASE bit.

16.2.8 START-UP CONSIDERATIONS

When the ECCP module is used in the PWM mode, the application hardware must use the proper external pull-up and/or pull-down resistors on the PWM output pins. When the microcontroller is released from Reset, all of the I/O pins are in the high-impedance state. The external circuits must keep the power switch devices in the off state until the microcontroller drives the I/O pins with the proper signal levels or activates the PWM output(s).

The CCP1M1:CCP1M0 bits (CCP1CON<1:0>) allow the user to choose whether the PWM output signals are active-high or active-low for each pair of PWM output pins (P1A/P1C and P1B/P1D). The PWM output polarities must be selected before the PWM pins are configured as outputs. Changing the polarity configuration while the PWM pins are configured as outputs is not recommended since it may result in damage to the application circuits.

The P1A, P1B, P1C and P1D output latches may not be in the proper states when the PWM module is initialized. Enabling the PWM pins for output at the same time as the ECCP module may cause damage to the application circuit. The ECCP module must be enabled in the proper Output mode and complete a full PWM cycle before configuring the PWM pins as outputs. The completion of a full PWM cycle is indicated by the TMR2IF bit being set as the second PWM period begins.

FIGURE 16-11: PWM AUTO-SHUTDOWN (PRSEN = `1`, AUTO-RESTART ENABLED)

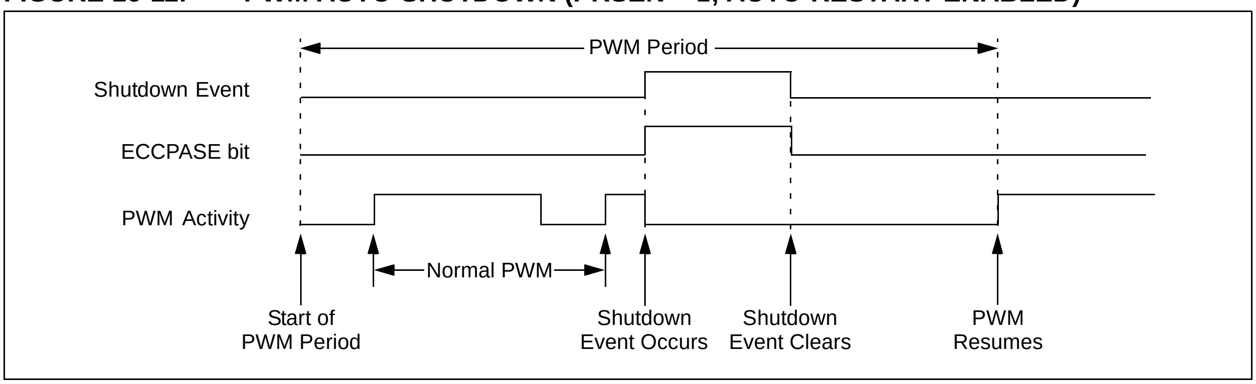
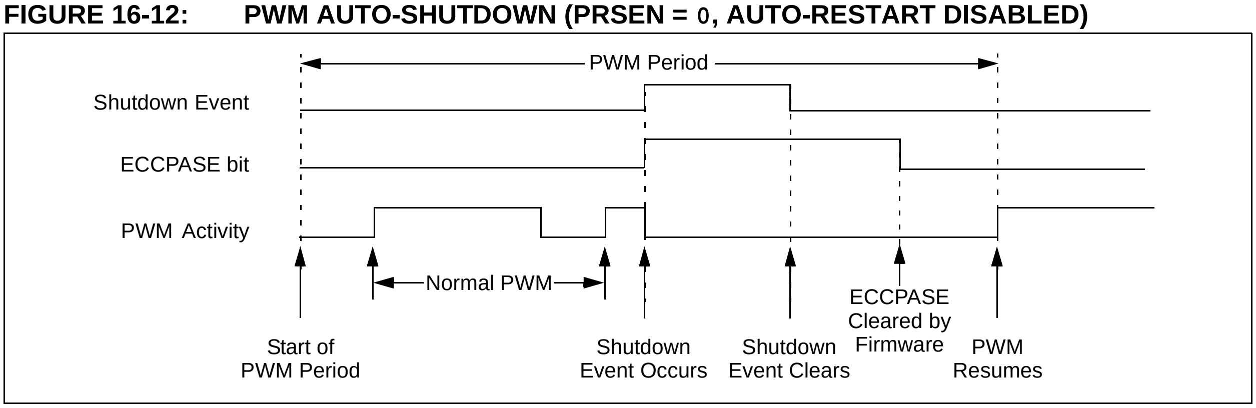


FIGURE 16-12: PWM AUTO-SHUTDOWN (PRSEN = `0`, AUTO-RESTART DISABLED)



16.2.9 SETUP FOR PWM OPERATION

The following steps should be taken when configuring the ECCP1 module for PWM operation:

1. Configure the PWM pins, P1A and P1B (and P1C and P1D, if used), as inputs by setting the corresponding TRISB bits.
2. Set the PWM period by loading the PR2 register.
3. Configure the ECCP1 module for the desired PWM mode and configuration by loading the CCP1CON register with the appropriate values:
   - Select one of the available output configurations and direction with the P1M1:P1M0 bits.
   - Select the polarities of the PWM output signals with the CCP1M3:CCP1M0 bits.
4. Set the PWM duty cycle by loading the CCPR1L register and CCP1CON<5:4> bits.
5. For Half-Bridge Output mode, set the dead-band delay by loading ECCP1DEL<6:0> with the appropriate value.
6. If auto-shutdown operation is required, load the ECCPAS register:
   - Select the auto-shutdown sources using the ECCPAS<2:0> bits.
   - Select the shutdown states of the PWM output pins using PSSAC1:PSSAC0 and PSSBD1:PSSBD0 bits.
   - Set the ECCPASE bit (ECCPAS<7>).
   - Configure the comparators using the CMCON register.
   - Configure the comparator inputs as analog inputs.
7. If auto-restart operation is required, set the PRSEN bit (ECCP1DEL<7>).
8. Configure and start TMR2:
   - Clear the TMR2 interrupt flag bit by clearing the TMR2IF bit (PIR1<1>).
   - Set the TMR2 prescale value by loading the T2CKPS bits (T2CON<1:0>).
   - Enable Timer2 by setting the TMR2ON bit (T2CON<2>).
9. Enable PWM outputs after a new PWM cycle has started:
   - Wait until TMR2 overflows (TMR2IF bit is set).
   - Enable the CCP1/P1A, P1B, P1C and/or P1D pin outputs by clearing the respective TRISB bits.
   - Clear the ECCPASE bit (ECCP1AS<7>).

16.2.10 EFFECTS OF A RESET

Both Power-on and subsequent Resets will force all ports to Input mode and the CCP registers to their Reset states.

This forces the Enhanced CCP module to reset to a state compatible with the standard CCP module.

TABLE 16-3: REGISTERS ASSOCIATED WITH PWM AND TIMER2

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other Resets
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
PIR1	PSPIF	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
PIE1	PSPIE	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
IPR1	PSPIP	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	1111 1111	1111 1111
TRISC	PORTC Data Direction Register								1111 1111	1111 1111
TRISE	PORTE Data Direction Register								1111 1111	1111 1111
TRISG	—	—	—	PORTG Data Direction Register					---1 1111	---1 1111
TMR2	Timer2 Module Register								0000 0000	0000 0000
PR2	Timer2 Module Period Register								1111 1111	1111 1111
T2CON	—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-000 0000	-000 0000
CCPR1L	Capture/Compare/PWM Register 1 (LSB)								xxxx xxxx	uuuu uuuu
CCPR1H	Capture/Compare/PWM Register 1 (MSB)								xxxx xxxx	uuuu uuuu
CCP1CON	P1M1	P1M0	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0	0000 0000	0000 0000
ECCP1AS	ECCPASE	ECCPAS2	ECCPAS1	ECCPAS0	PSSAC1	PSSAC0	PSSBD1	PSSBD0	0000 0000	0000 0000
ECCP1DEL	PRSEN	PDC6	PDC5	PDC4	PDC3	PDC2	PDC1	PDC0	0000 0000	uuuu uuuu

Legend: x = unknown, u = unchanged, – = unimplemented, read as '0'. Shaded cells are not used by PWM and Timer2.

NOTES:

17.0 MASTER SYNCHRONOUS SERIAL PORT (MSSP) MODULE

17.1 Master SSP (MSSP) Module Overview

The Master Synchronous Serial Port (MSSP) module is a serial interface, useful for communicating with other peripheral or microcontroller devices. These peripheral devices may be serial EEPROMs, shift registers, display drivers, A/D converters, etc. The MSSP module can operate in one of two modes:

- Serial Peripheral Interface (SPI)
- Inter-Integrated Circuit (I^{2}C)
  - Full Master mode
  - Slave mode (with general address call)

The I^{2}C interface supports the following modes in hardware:

- Master mode
- Multi-Master mode
- Slave mode

17.2 Control Registers

The MSSP module has three associated registers. These include a status register (SSPSTAT) and two control registers (SSPCON1 and SSPCON2). The use of these registers and their individual configuration bits differ significantly depending on whether the MSSP module is operated in SPI or I^{2}C mode.

Additional details are provided under the individual sections.

17.3 SPI Mode

The SPI mode allows 8 bits of data to be synchronously transmitted and received simultaneously. All four modes of SPI are supported. To accomplish communication, typically three pins are used:

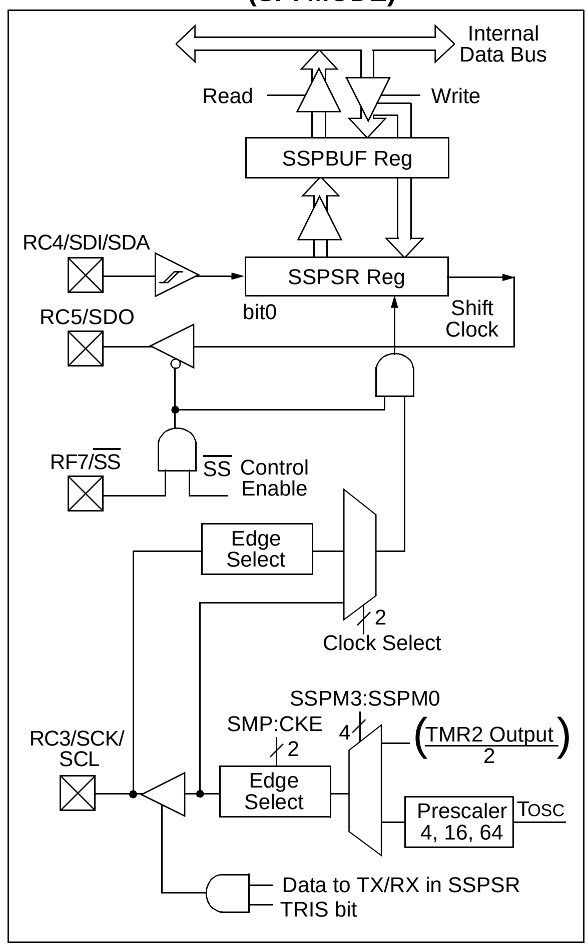
- Serial Data Out (SDO) – RC5/SDO
- Serial Data In (SDI) – RC4/SDI/SDA
- Serial Clock (SCK) – RC3/SCK/SCL

Additionally, a fourth pin may be used when in a Slave mode of operation:

- Slave Select ($\overline{SS}$) – RF7/$\overline{SS}$

Figure 17-1 shows the block diagram of the MSSP module when operating in SPI mode.

FIGURE 17-1: MSSP BLOCK DIAGRAM (SPI MODE)



17.3.1 REGISTERS

The MSSP module has four registers for SPI mode operation. These are:

- MSSP Control Register 1 (SSPCON1)
- MSSP Status Register (SSPSTAT)
- Serial Receive/Transmit Buffer Register (SSPBUF)
- MSSP Shift Register (SSPSR) – Not directly accessible

SSPCON1 and SSPSTAT are the control and status registers in SPI mode operation. The SSPCON1 register is readable and writable. The lower 6 bits of the SSPSTAT are read-only. The upper two bits of the SSPSTAT are read/write.

SSPSR is the shift register used for shifting data in or out. SSPBUF is the buffer register to which data bytes are written to or read from.

In receive operations, SSPSR and SSPBUF together create a double-buffered receiver. When SSPSR receives a complete byte, it is transferred to SSPBUF and the SSPIF interrupt is set.

During transmission, the SSPBUF is not double-buffered. A write to SSPBUF will write to both SSPBUF and SSPSR.

REGISTER 17-1: SSPSTAT: MSSP STATUS REGISTER (SPI MODE)

R/W-0	R/W-0	R-0	R-0	R-0	R-0	R-0	R-0
SMP	CKE	D/$\overline{A}$	P	S	R/$\overline{W}$	UA	BF
bit 7							bit 0

bit 7 **SMP:** Sample bit

<u>SPI Master mode:</u>
1 = Input data sampled at end of data output time
0 = Input data sampled at middle of data output time

<u>SPI Slave mode:</u>
SMP must be cleared when SPI is used in Slave mode.

bit 6 **CKE:** SPI Clock Edge Select bit

<u>When CKP = 0:</u>
1 = Data transmitted on rising edge of SCK
0 = Data transmitted on falling edge of SCK

<u>When CKP = 1:</u>
1 = Data transmitted on falling edge of SCK
0 = Data transmitted on rising edge of SCK

bit 5 **D/$\overline{A}$:** Data/$\overline{\text{Address}}$ bit
Used in I^{2}C mode only.

bit 4 **P:** Stop bit
Used in I^{2}C mode only. This bit is cleared when the MSSP module is disabled, SSPEN is cleared.

bit 3 **S:** Start bit
Used in I^{2}C mode only.

bit 2 **R/$\overline{W}$:** Read/$\overline{\text{Write}}$ bit Information
Used in I^{2}C mode only.

bit 1 **UA:** Update Address bit
Used in I^{2}C mode only.

bit 0 **BF:** Buffer Full Status bit (Receive mode only)
1 = Receive complete, SSPBUF is full
0 = Receive not complete, SSPBUF is empty

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
- n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

REGISTER 17-2: SSPCON1: MSSP CONTROL REGISTER 1 (SPI MODE)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0
bit 7							bit 0

bit 7 **WCOL:** Write Collision Detect bit (Transmit mode only)
1 = The SSPBUF register is written while it is still transmitting the previous word (must be cleared in software)
0 = No collision

bit 6 **SSPOV:** Receive Overflow Indicator bit
SPI Slave mode:
1 = A new byte is received while the SSPBUF register is still holding the previous data. In case of overflow, the data in SSPSR is lost. Overflow can only occur in Slave mode.The user must read the SSPBUF, even if only transmitting data, to avoid setting overflow (must be cleared in software).
0 = No overflow

Note: In Master mode, the overflow bit is not set since each new reception (and transmission) is initiated by writing to the SSPBUF register.

bit 5 **SSPEN:** Synchronous Serial Port Enable bit
1 = Enables serial port and configures SCK, SDO, SDI, and $\overline{SS}$ as serial port pins
0 = Disables serial port and configures these pins as I/O port pins

Note: When enabled, these pins must be properly configured as input or output.

bit 4 **CKP:** Clock Polarity Select bit
1 = Idle state for clock is a high level
0 = Idle state for clock is a low level

bit 3-0 **SSPM3:SSPM0:** Synchronous Serial Port Mode Select bits
0101 = SPI Slave mode, clock = SCK pin, $\overline{SS}$ pin control disabled, $\overline{SS}$ can be used as I/O pin
0100 = SPI Slave mode, clock = SCK pin, $\overline{SS}$ pin control 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: Bit combinations not specifically listed here are either reserved or implemented in I^2C mode only.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
- n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

17.3.2 OPERATION

When initializing the SPI, several options need to be specified. This is done by programming the appropriate control bits (SSPCON1<5:0> and SSPSTAT<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)

The MSSP consists of a Transmit/Receive Shift register (SSPSR) and a Buffer register (SSPBUF). The SSPSR shifts the data in and out of the device, MSb first. The SSPBUF holds the data that was written to the SSPSR, until the received data is ready. Once the 8 bits of data have been received, that byte is moved to the SSPBUF register. Then the Buffer Full detect bit, BF (SSPSTAT<0>) and the interrupt flag bit, SSPIF, are set. This double-buffering of the received data (SSPBUF) allows the next byte to start reception before reading the data that was just received. Any write to the SSPBUF register during transmission/reception of data will be ignored and the Write Collision detect bit, WCOL (SSPCON1<7>), will be set. User software must clear the WCOL bit so that it can be determined if the following write(s) to the SSPBUF register completed successfully.

When the application software is expecting to receive valid data, the SSPBUF should be read before the next byte of data to transfer is written to the SSPBUF. Buffer Full bit, BF (SSPSTAT<0>), indicates when SSPBUF has been loaded with the received data (transmission is complete). When the SSPBUF 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. The SSPBUF must be read and/or written. 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. Example 17-1 shows the loading of the SSPBUF (SSPSR) for data transmission.

The SSPSR is not directly readable or writable and can only be accessed by addressing the SSPBUF register. Additionally, the MSSP Status register (SSPSTAT) indicates the various status conditions.

EXAMPLE 17-1: LOADING THE SSPBUF (SSPSR) REGISTER

```
LOOP    BTFSS    SSPSTAT, BF     ;Has data been received(transmit complete)?
        BRA      LOOP            ;No
        MOVF     SSPBUF, W       ;WREG reg = contents of SSPBUF
        MOVWF    RXDATA          ;Save in user RAM, if data is meaningful
        MOVF     TXDATA, W       ;W reg = contents of TXDATA
        MOVWF    SSPBUF          ;New data to xmit
```


17.3.3 ENABLING SPI I/O

To enable the serial port, SSP Enable bit, SSPEN (SSPCON1<5>), must be set. To reset or reconfigure SPI mode, clear the SSPEN bit, reinitialize the SSPCON registers and then set the SSPEN bit. This configures the SDI, SDO, SCK and $\overline{SS}$ pins as serial port pins. For the pins to behave as the serial port function, some must have their data direction bits (in the TRIS register) appropriately programmed as follows:

- SDI is automatically controlled by the SPI module
- SDO must have TRISC<5> bit cleared
- SCK (Master mode) must have TRISC<3> bit cleared
- SCK (Slave mode) must have TRISC<3> bit set
- $\overline{SS}$ must have TRISF<7> bit set

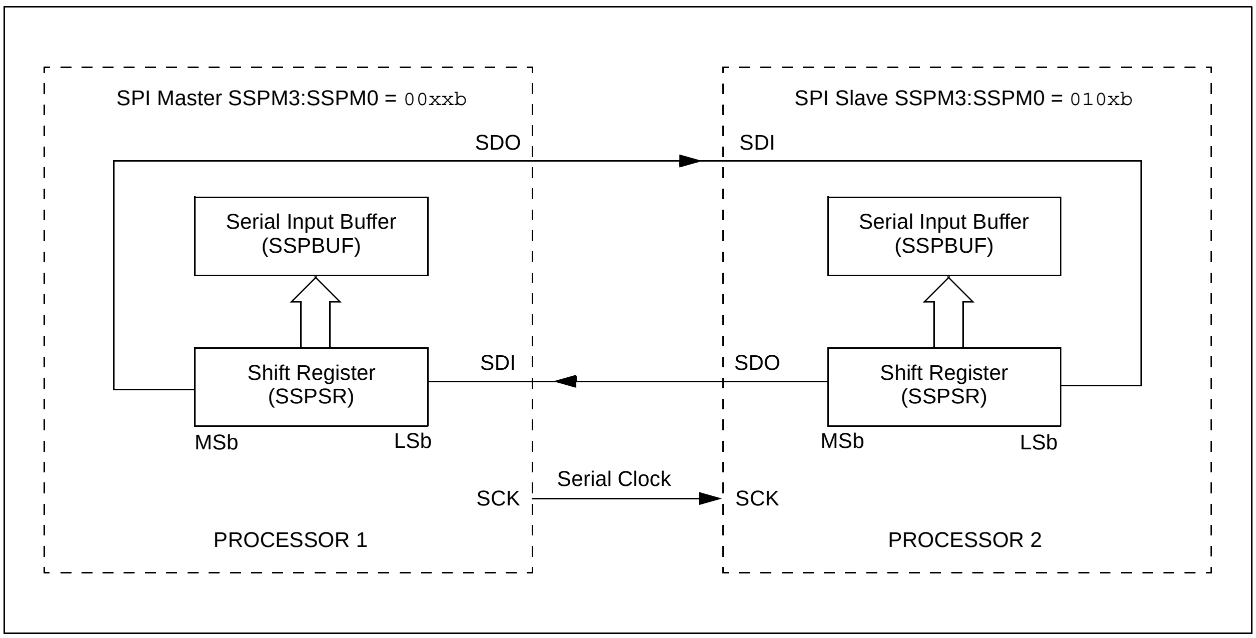
Any serial port function that is not desired may be overridden by programming the corresponding data direction (TRIS) register to the opposite value.

17.3.4 TYPICAL CONNECTION

Figure 17-2 shows a typical connection between two microcontrollers. The master controller (Processor 1) initiates the data transfer by sending the SCK signal. Data is shifted out of both shift registers on their programmed clock edge and latched on the opposite edge of the clock. Both processors should be programmed to the same Clock Polarity (CKP), then both controllers would send and receive data at the same time. Whether the data is meaningful (or dummy data) depends on the application software. This leads to three scenarios for data transmission:

- Master sends data - Slave sends dummy data
- Master sends data - Slave sends data
- Master sends dummy data - Slave sends data

FIGURE 17-2: SPI MASTER/SLAVE CONNECTION



17.3.5 MASTER MODE

The master can initiate the data transfer at any time because it controls the SCK. The master determines when the slave (Processor 2, Figure 17-2) is to broadcast data by the software protocol.

In Master mode, the data is transmitted/received as soon as the SSPBUF 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 SSPBUF register as if a normal received byte (interrupts and status bits appropriately set). This could be useful in receiver applications as a "Line Activity Monitor" mode.

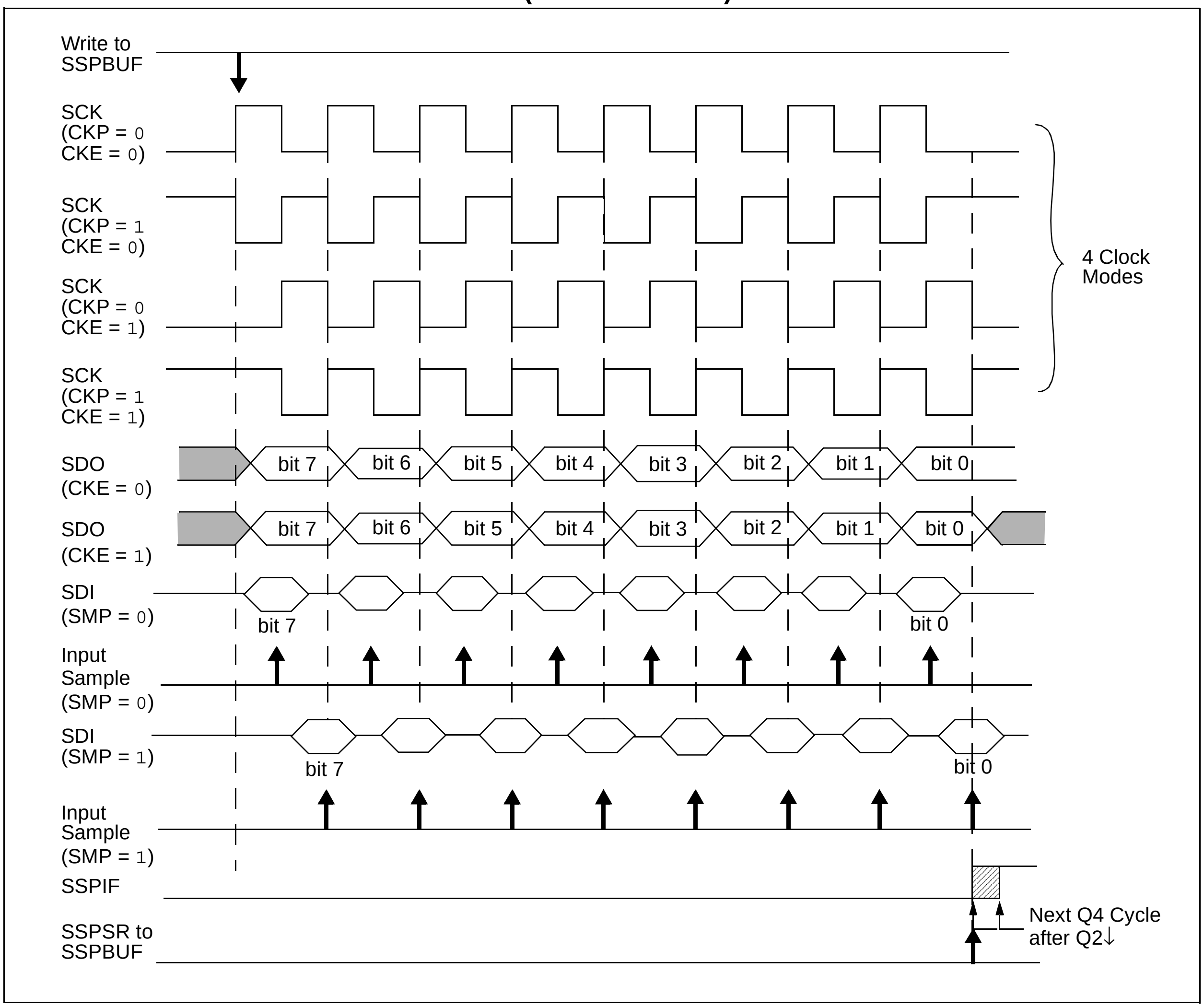
The clock polarity is selected by appropriately programming the CKP bit (SSPCON1<4>). This then, would give waveforms for SPI communication, as shown in Figure 17-3, Figure 17-5 and Figure 17-6, where the MSB is transmitted first. In Master mode, the SPI clock rate (bit rate) is user programmable to be one of the following:

- FOSC/4 (or TCY)
- FOSC/16 (or 4 • TCY)
- FOSC/64 (or 16 • TCY)
- Timer2 output/2

This allows a maximum data rate (at 40 MHz) of 10.00 Mbps.

Figure 17-3 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 SSPBUF is loaded with the received data is shown.

FIGURE 17-3: SPI MODE WAVEFORM (MASTER MODE)



17.3.6 SLAVE MODE

In Slave mode, the data is transmitted and received as the external clock pulses appear on SCK. When the last bit is latched, the SSPIF interrupt flag bit is set.

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. When a byte is received, the device will wake-up from Sleep.

17.3.7 SLAVE SELECT SYNCHRONIZATION

The SS pin allows a Synchronous Slave mode. The SPI must be in Slave mode with SS pin control enabled (SSPCON1<3:0> = 04h). The pin must not be driven low for the SS pin to function as an input. The data latch must be high. When the SS pin is low, transmission and reception are enabled and the SDO pin is driven. When the 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 SS pin control enabled (SSPCON<3:0> = `0100`), the SPI module will reset if the SS pin is set to V_{DD}.
>
> **2:** If the SPI is used in Slave mode with CKE set, then the SS pin control must be enabled.

When the SPI module resets, the bit counter is forced to '`0`'. This can be done by either forcing the SS pin to a high level or clearing the SSPEN bit.

To emulate two-wire communication, the SDO pin can be connected to the SDI pin. When the SPI needs to operate as a receiver, the SDO pin can be configured as an input. This disables transmissions from the SDO. The SDI can always be left as an input (SDI function) since it cannot create a bus conflict.

FIGURE 17-4: SLAVE SYNCHRONIZATION WAVEFORM

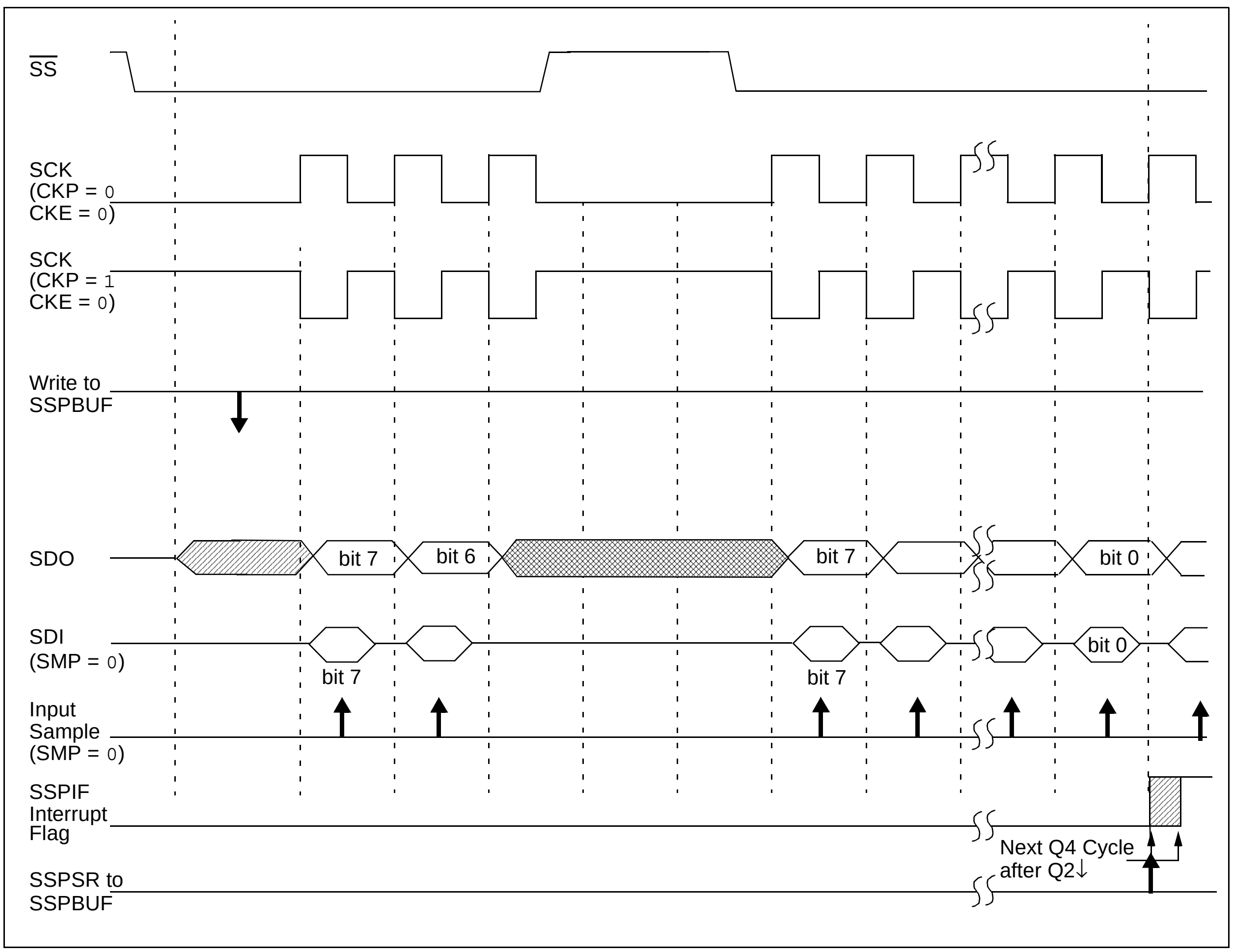


FIGURE 17-5: SPI MODE WAVEFORM (SLAVE MODE WITH CKE = 0)

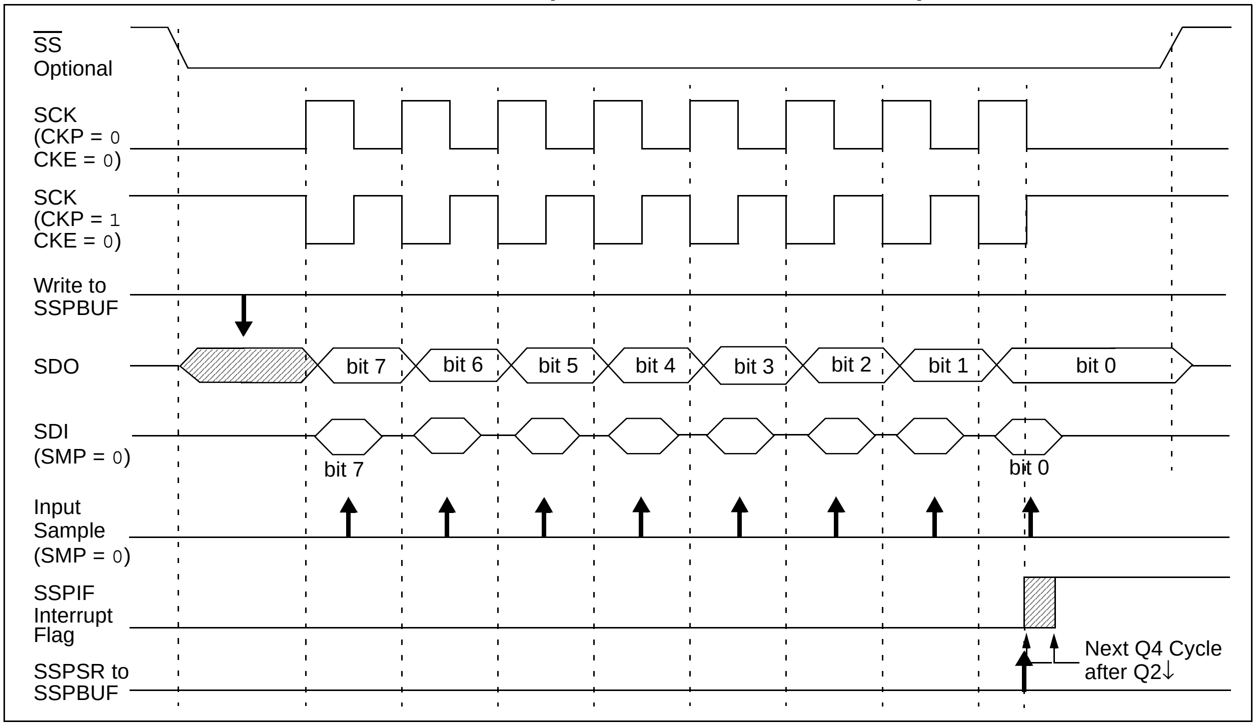
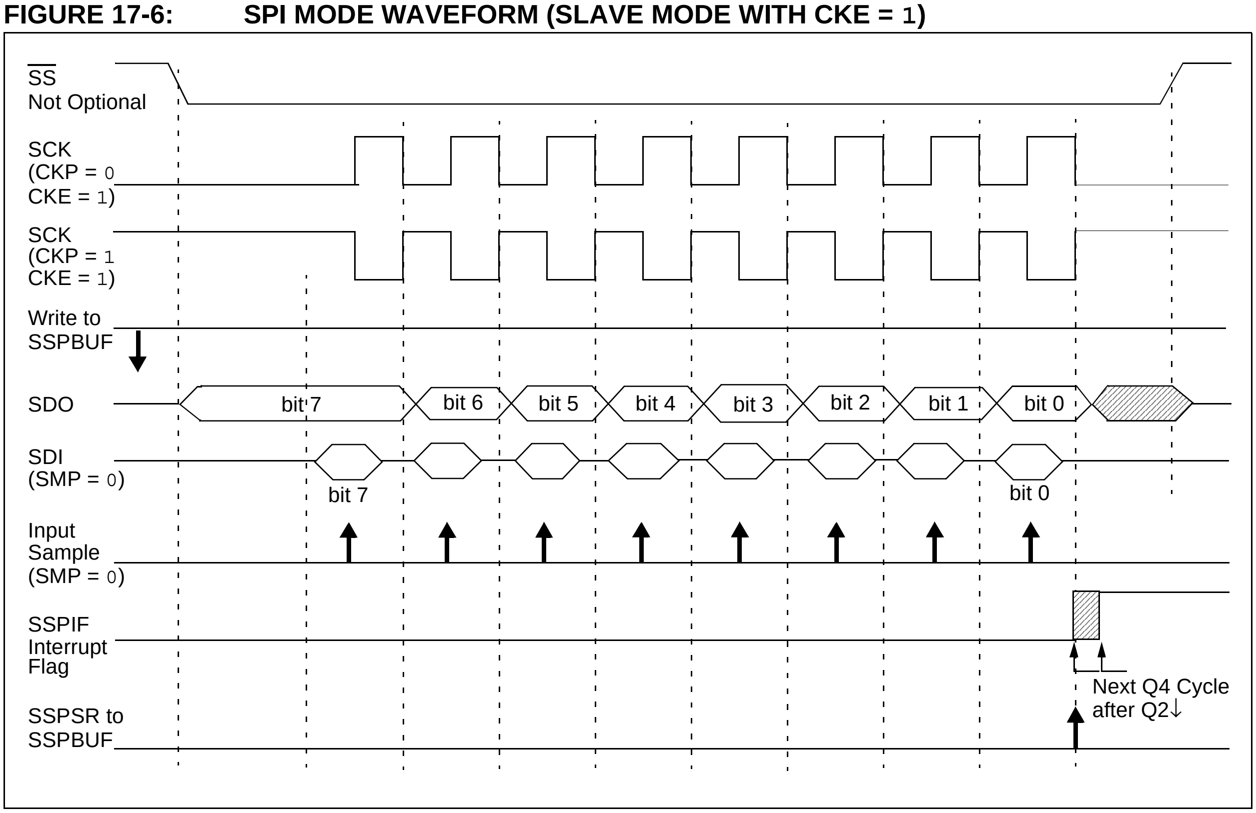


FIGURE 17-6: SPI MODE WAVEFORM (SLAVE MODE WITH CKE = 1)



17.3.8 SLEEP OPERATION

In Master mode, all module clocks are halted and the transmission/reception will remain in that state until the device wakes from Sleep. After the device returns to normal mode, the module will continue to transmit/receive data.

In 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 8 bits have been received, the MSSP interrupt flag bit will be set and if enabled, will wake the device from Sleep.

17.3.9 EFFECTS OF A RESET

A Reset disables the MSSP module and terminates the current transfer.

17.3.10 BUS MODE COMPATIBILITY

Table 17-1 shows the compatibility between the standard SPI modes and the states of the CKP and CKE control bits.

TABLE 17-1: SPI BUS MODES

Standard SPI Mode Terminology	Control Bits State	
	CKP	CKE
0, 0	0	1
0, 1	0	0
1, 0	1	1
1, 1	1	0

There is also a SMP bit which controls when the data is sampled.

TABLE 17-2: REGISTERS ASSOCIATED WITH SPI OPERATION

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other Resets
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 0000	0000 0000
PIR1	PSPIF	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
PIE1	PSPIE	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
IPR1	PSPIP	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	1111 1111	1111 1111
TRISC	PORTC Data Direction Register								1111 1111	1111 1111
TRISF	TRISF7	TRISF6	TRISF5	TRISF4	TRISF3	TRISF2	TRISF1	TRISF0	1111 1111	uuuu uuuu
SSPBUF	Synchronous Serial Port Receive Buffer/Transmit Register								xxxx xxxx	uuuu uuuu
SSPCON	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	0000 0000	0000 0000
SSPSTAT	SMP	CKE	D/$\overline{A}$	P	S	R/$\overline{W}$	UA	BF	0000 0000	0000 0000

Legend: x = unknown, u = unchanged, – = unimplemented, read as '0'. Shaded cells are not used by the MSSP in SPI mode.

17.4 I²C Mode

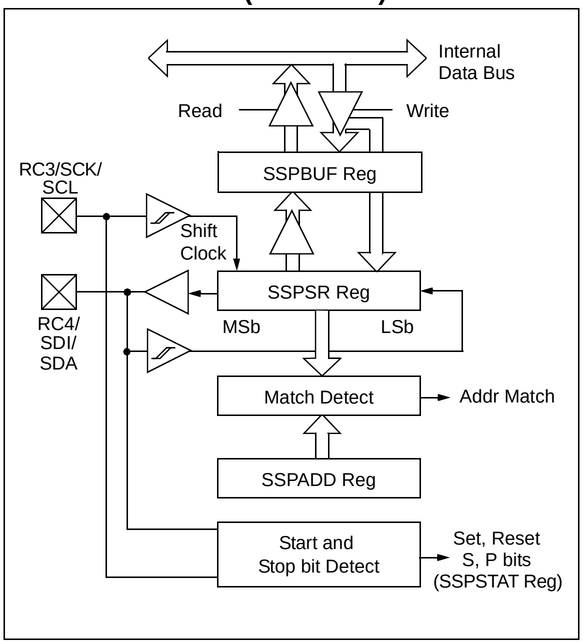
The MSSP module in I²C mode fully implements all master and slave functions (including general call support) and provides interrupts on Start and Stop bits in hardware to determine a free bus (multi-master function). The MSSP module implements the standard mode specifications, as well as 7-bit and 10-bit addressing.

Two pins are used for data transfer:

- Serial clock (SCL) – RC3/SCK/SCL
- Serial data (SDA) – RC4/SDI/SDA

The user must configure these pins as inputs or outputs through the TRISC<4:3> bits.

FIGURE 17-7: MSSP BLOCK DIAGRAM (I²C MODE)



17.4.1 REGISTERS

The MSSP module has six registers for I²C operation. These are:

- MSSP Control Register 1 (SSPCON1)
- MSSP Control Register 2 (SSPCON2)
- MSSP Status Register (SSPSTAT)
- Serial Receive/Transmit Buffer (SSPBUF)
- MSSP Shift Register (SSPSR) – Not directly accessible
- MSSP Address Register (SSPADD)

SSPCON, SSPCON2 and SSPSTAT are the control and status registers in I²C mode operation. The SSPCON and SSPCON2 registers are readable and writable. The lower six bits of the SSPSTAT are read-only. The upper two bits of the SSPSTAT are read/write.

SSPSR is the shift register used for shifting data in or out. SSPBUF is the buffer register to which data bytes are written to or read from.

SSPADD register holds the slave device address when the SSP is configured in I²C Slave mode. When the SSP is configured in Master mode, the lower seven bits of SSPADD act as the Baud Rate Generator reload value.

In receive operations, SSPSR and SSPBUF together create a double-buffered receiver. When SSPSR receives a complete byte, it is transferred to SSPBUF and the SSPIF interrupt is set.

During transmission, the SSPBUF is not double-buffered. A write to SSPBUF will write to both SSPBUF and SSPSR.

REGISTER 17-3: SSPSTAT: MSSP STATUS REGISTER (I^2C MODE)

R/W-0	R/W-0	R-0	R-0	R-0	R-0	R-0	R-0
SMP	CKE	D/$\overline{\text{A}}$	P	S	R/$\overline{\text{W}}$	UA	BF
bit 7							bit 0

bit 7 **SMP:** Slew Rate Control bit
In Master or Slave mode:
1 = Slew rate control disabled for Standard Speed mode (100 kHz and 1 MHz)
0 = Slew rate control enabled for High-Speed mode (400 kHz)

bit 6 **CKE:** SMBus Select bit
In Master or Slave mode:
1 = Enable SMBus specific inputs
0 = Disable SMBus specific inputs

bit 5 **D/$\overline{\text{A}}$:** Data/$\overline{\text{Address}}$ bit
In Master mode:
Reserved.
In Slave mode:
1 = Indicates that the last byte received or transmitted was data
0 = Indicates that the last byte received or transmitted was address

bit 4 **P:** Stop bit
1 = Indicates that a Stop bit has been detected last
0 = Stop bit was not detected last

Note: This bit is cleared on Reset and when SSPEN is cleared.

bit 3 **S:** Start bit
1 = Indicates that a Start bit has been detected last
0 = Start bit was not detected last

Note: This bit is cleared on Reset and when SSPEN is cleared.

bit 2 **R/$\overline{\text{W}}$:** Read/$\overline{\text{Write}}$ bit Information (I^2C mode only)
In Slave mode:
1 = Read
0 = Write

Note: This bit holds the R/$\overline{\text{W}}$ bit information following the last address match. This bit is only valid from the address match to the next Start bit, Stop bit or not $\overline{\text{ACK}}$ bit.

In Master mode:
1 = Transmit is in progress
0 = Transmit is not in progress

Note: ORing this bit with SEN, RSEN, PEN, RCEN or ACKEN will indicate if the MSSP is in Idle mode.

bit 1 **UA:** Update Address bit (10-bit Slave mode only)
1 = Indicates that the user needs to update the address in the SSPADD register
0 = Address does not need to be updated

bit 0 **BF:** Buffer Full Status bit
In Transmit mode:
1 = Receive complete, SSPBUF is full
0 = Receive not complete, SSPBUF is empty
In Receive mode:
1 = Data transmit in progress (does not include the $\overline{\text{ACK}}$ and Stop bits), SSPBUF is full
0 = Data transmit complete (does not include the $\overline{\text{ACK}}$ and Stop bits), SSPBUF is empty

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
- n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

REGISTER 17-4: SSPCON1: MSSP CONTROL REGISTER 1 (I^2C MODE)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0
bit 7							bit 0

bit 7 **WCOL:** Write Collision Detect bit

In Master Transmit mode:

1 = A write to the SSPBUF register was attempted while the I^2C conditions were not valid for a transmission to be started (must be cleared in software)

0 = No collision

In Slave Transmit mode:

1 = The SSPBUF register is written while it is still transmitting the previous word (must be cleared in software)

0 = No collision

In Receive mode (Master or Slave modes):

This is a “don’t care” bit.

bit 6 **SSPOV:** Receive Overflow Indicator bit

In Receive mode:

1 = A byte is received while the SSPBUF register is still holding the previous byte (must be cleared in software)

0 = No overflow

In Transmit mode:

This is a “don’t care” bit in Transmit mode.

bit 5 **SSPEN:** Synchronous Serial Port Enable bit

1 = Enables the serial port and configures the SDA and SCL pins as the serial port pins

0 = Disables serial port and configures these pins as I/O port pins

Note: When enabled, the SDA and SCL pins must be properly configured as input or output.

bit 4 **CKP:** SCK Release Control bit

In Slave mode:

1 = Release clock

0 = Holds clock low (clock stretch), used to ensure data setup time

In Master mode:

Unused in this mode.

bit 3-0 **SSPM3:SSPM0:** Synchronous Serial Port Mode Select bits

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

1011 = I^2C Firmware Controlled Master mode (slave Idle)

1000 = I^2C Master mode, clock = FOSC/(4 * (SSPADD + 1))

0111 = I^2C Slave mode, 10-bit address

0110 = I^2C Slave mode, 7-bit address

Note: Bit combinations not specifically listed here are either reserved or implemented in SPI mode only.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
- n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

REGISTER 17-5: SSPCON2: MSSP CONTROL REGISTER 2 (I^2C MODE)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN
bit 7							bit 0

bit 7 **GCEN:** General Call Enable bit (Slave mode only)
1 = Enable interrupt when a general call address (0000h) is received in the SSPSR
0 = General call address disabled

bit 6 **ACKSTAT:** Acknowledge Status bit (Master Transmit mode only)
1 = Acknowledge was not received from slave
0 = Acknowledge was received from slave

bit 5 **ACKDT:** Acknowledge Data bit (Master Receive mode only)
1 = Not Acknowledge
0 = Acknowledge

Note: Value that will be transmitted when the user initiates an Acknowledge sequence at the end of a receive.

bit 4 **ACKEN:** Acknowledge Sequence Enable bit (Master Receive mode only)
1 = Initiate Acknowledge sequence on SDA and SCL pins and transmit ACKDT data bit. Automatically cleared by hardware.
0 = Acknowledge sequence Idle

bit 3 **RCEN:** Receive Enable bit (Master Mode only)
1 = Enables Receive mode for I^2C
0 = Receive Idle

bit 2 **PEN:** Stop Condition Enable bit (Master mode only)
1 = Initiate Stop condition on SDA and SCL pins. Automatically cleared by hardware.
0 = Stop condition Idle

bit 1 **RSEN:** Repeated Start Condition Enabled bit (Master mode only)
1 = Initiate Repeated Start condition on SDA and SCL pins. Automatically cleared by hardware.
0 = Repeated Start condition Idle

bit 0 **SEN:** Start Condition Enabled/Stretch Enabled bit
In Master mode:
1 = Initiate Start condition on SDA and SCL pins. Automatically cleared by hardware.
0 = Start condition Idle
In Slave mode:
1 = Clock stretching is enabled for both slave transmit and slave receive (stretch enabled)
0 = Clock stretching is disabled

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
- n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

Note: For bits ACKEN, RCEN, PEN, RSEN, SEN: If the I^2C module is not in the Idle mode, this bit may not be set (no spooling) and the SSPBUF may not be written (or writes to the SSPBUF are disabled).

17.4.2 OPERATION

The MSSP module functions are enabled by setting MSSP Enable bit, SSPEN (SSPCON<5>).

The SSPCON1 register allows control of the I^{2}C operation. Four mode selection bits (SSPCON<3:0>) allow one of the following I^{2}C modes to be selected:

- I^{2}C Master mode, clock = OSC/4 (SSPADD + 1)
- I^{2}C Slave mode (7-bit address)
- I^{2}C Slave mode (10-bit address)
- I^{2}C Slave mode (7-bit address) with Start and Stop bit interrupts enabled
- I^{2}C Slave mode (10-bit address) with Start and Stop bit interrupts enabled
- I^{2}C Firmware Controlled Master mode, slave is Idle

Selection of any I^{2}C mode with the SSPEN bit set, forces the SCL and SDA pins to be open-drain, provided these pins are programmed to inputs by setting the appropriate TRISC bits. To ensure proper operation of the module, pull-up resistors must be provided externally to the SCL and SDA pins.

17.4.3 SLAVE MODE

In Slave mode, the SCL and SDA pins must be configured as inputs (TRISC<4:3> set). The MSSP module will override the input state with the output data when required (slave-transmitter).

The I^{2}C Slave mode hardware will always generate an interrupt on an address match. Through the mode select bits, the user can also choose to interrupt on Start and Stop bits

When an address is matched or the data transfer after an address match is received, the hardware automatically will generate the Acknowledge ($\overline{\text{ACK}}$) pulse and load the SSPBUF register with the received value currently in the SSPSR register.

Any combination of the following conditions will cause the MSSP module not to give this $\overline{\text{ACK}}$ pulse:

- The buffer full bit BF (SSPSTAT<0>) was set before the transfer was received.
- The overflow bit SSPOV (SSPCON<6>) was set before the transfer was received.

In this case, the SSPSR register value is not loaded into the SSPBUF but bit SSPIF (PIR1<3>) is set. The BF bit is cleared by reading the SSPBUF register while bit SSPOV is cleared through software.

The SCL clock input must have a minimum high and low for proper operation. The high and low times of the I^{2}C specification, as well as the requirement of the MSSP module, are shown in timing parameter #100 and parameter #101.

17.4.3.1 Addressing

Once the MSSP module has been enabled, it waits for a Start condition to occur. Following the Start condition, the 8 bits are shifted into the SSPSR register. All incoming bits are sampled with the rising edge of the clock (SCL) line. The value of register SSPSR<7:1> is compared to the value of the SSPADD register. The address is compared on the falling edge of the eighth clock (SCL) pulse. If the addresses match and the BF and SSPOV bits are clear, the following events occur:

1. The SSPSR register value is loaded into the SSPBUF register.
2. The buffer full bit BF is set.
3. An $\overline{\text{ACK}}$ pulse is generated.
4. MSSP interrupt flag bit, SSPIF (PIR1<3>), is set (interrupt is generated, if enabled) on the falling edge of the ninth SCL pulse.

In 10-bit Address mode, two address bytes need to be received by the slave. The five Most Significant bits (MSbs) of the first address byte specify if this is a 10-bit address. Bit R/$\overline{\text{W}}$ (SSPSTAT<2>) must specify a write so the slave device will receive the second address byte. For a 10-bit address, the first byte would equal '`11110 A9 A8 0`', where '`A9`' and '`A8`' are the two MSbs of the address. The sequence of events for 10-bit address is as follows, with steps 7 through 9 for the slave-transmitter:

1. Receive first (high) byte of address (bits SSPIF, BF and bit UA (SSPSTAT<1>) are set).
2. Update the SSPADD register with second (low) byte of address (clears bit UA and releases the SCL line).
3. Read the SSPBUF register (clears bit BF) and clear flag bit SSPIF.
4. Receive second (low) byte of address (bits SSPIF, BF, and UA are set).
5. Update the SSPADD register with the first (high) byte of address. If match releases SCL line, this will clear bit UA.
6. Read the SSPBUF register (clears bit BF) and clear flag bit SSPIF.
7. Receive Repeated Start condition.
8. Receive first (high) byte of address (bits SSPIF and BF are set).
9. Read the SSPBUF register (clears bit BF) and clear flag bit SSPIF.

17.4.3.2 Reception

When the R/$\overline{\text{W}}$ bit of the address byte is clear and an address match occurs, the R/$\overline{\text{W}}$ bit of the SSPSTAT register is cleared. The received address is loaded into the SSPBUF register and the SDA line is held low ($\overline{\text{ACK}}$).

When the address byte overflow condition exists, then the no Acknowledge ($\overline{\text{ACK}}$) pulse is given. An overflow condition is defined as either bit BF (SSPSTAT<0>) is set or bit SSPOV (SSPCON1<6>) is set.

An MSSP interrupt is generated for each data transfer byte. Flag bit SSPIF (PIR1<3>) must be cleared in software. The SSPSTAT register is used to determine the status of the byte.

If SEN is enabled (SSPCON2<0> = `1`), RC3/SCK/SCL will be held low (clock stretch) following each data transfer. The clock must be released by setting bit CKP (SSPCON<4>). See **Section 17.4.4 “Clock Stretching”** for more detail.

17.4.3.3 Transmission

When the R/$\overline{\text{W}}$ bit of the incoming address byte is set and an address match occurs, the R/$\overline{\text{W}}$ bit of the SSPSTAT register is set. The received address is loaded into the SSPBUF register. The $\overline{\text{ACK}}$ pulse will be sent on the ninth bit and pin RC3/SCK/SCL is held low, regardless of SEN (see **Section 17.4.4 “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 SSPBUF register which also loads the SSPSR register. Then pin RC3/SCK/SCL should be enabled by setting bit CKP (SSPCON1<4>). 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 (Figure 17-9).

The $\overline{\text{ACK}}$ pulse from the master-receiver is latched on the rising edge of the ninth SCL input pulse. If the SDA line is high (not $\overline{\text{ACK}}$), then the data transfer is complete. In this case, when the $\overline{\text{ACK}}$ is latched by the slave, the slave logic is reset (resets SSPSTAT register) and the slave monitors for another occurrence of the Start bit. If the SDA line was low ($\overline{\text{ACK}}$), the next transmit data must be loaded into the SSPBUF register. Again, pin RC3/SCK/SCL must be enabled by setting bit CKP.

An MSSP interrupt is generated for each data transfer byte. The SSPIF bit must be cleared in software and the SSPSTAT register is used to determine the status of the byte. The SSPIF bit is set on the falling edge of the ninth clock pulse.

FIGURE 17-8: I²C SLAVE MODE TIMING WITH SEN = 0 (RECEPTION, 7-BIT ADDRESS)

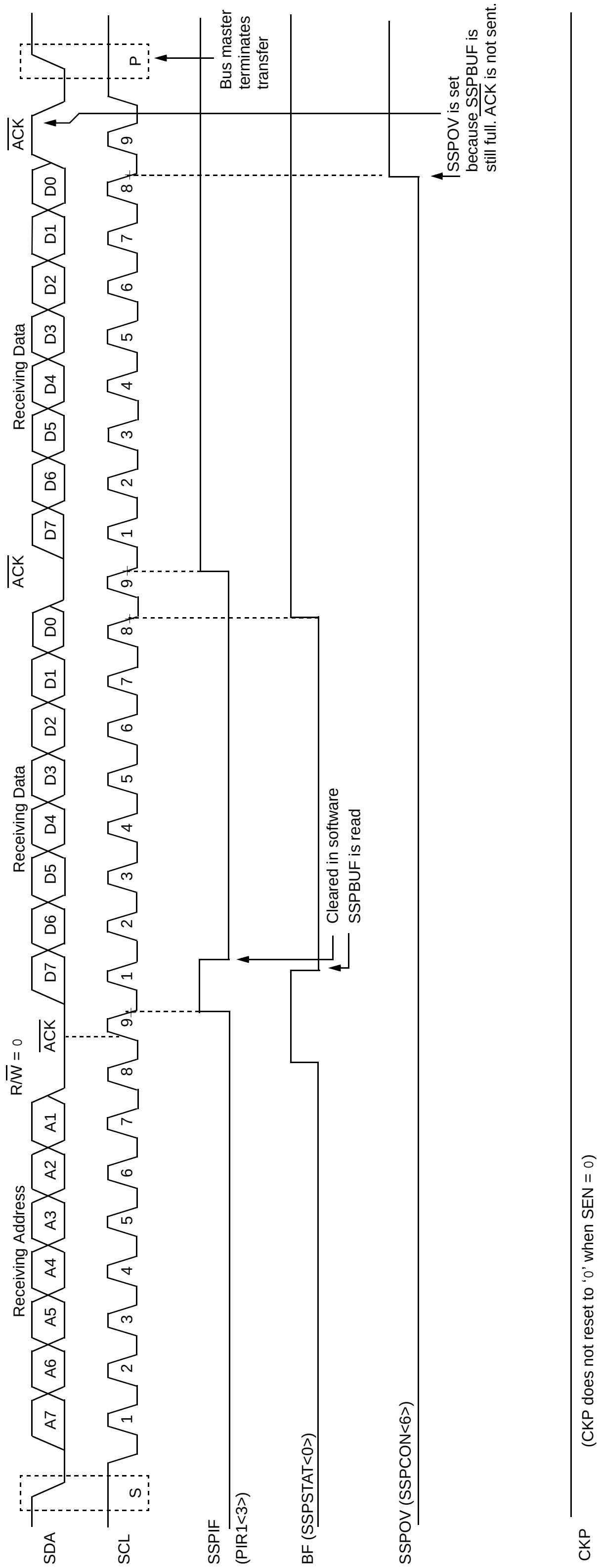


FIGURE 17-9: I²C SLAVE MODE TIMING (TRANSMISSION, 7-BIT ADDRESS)

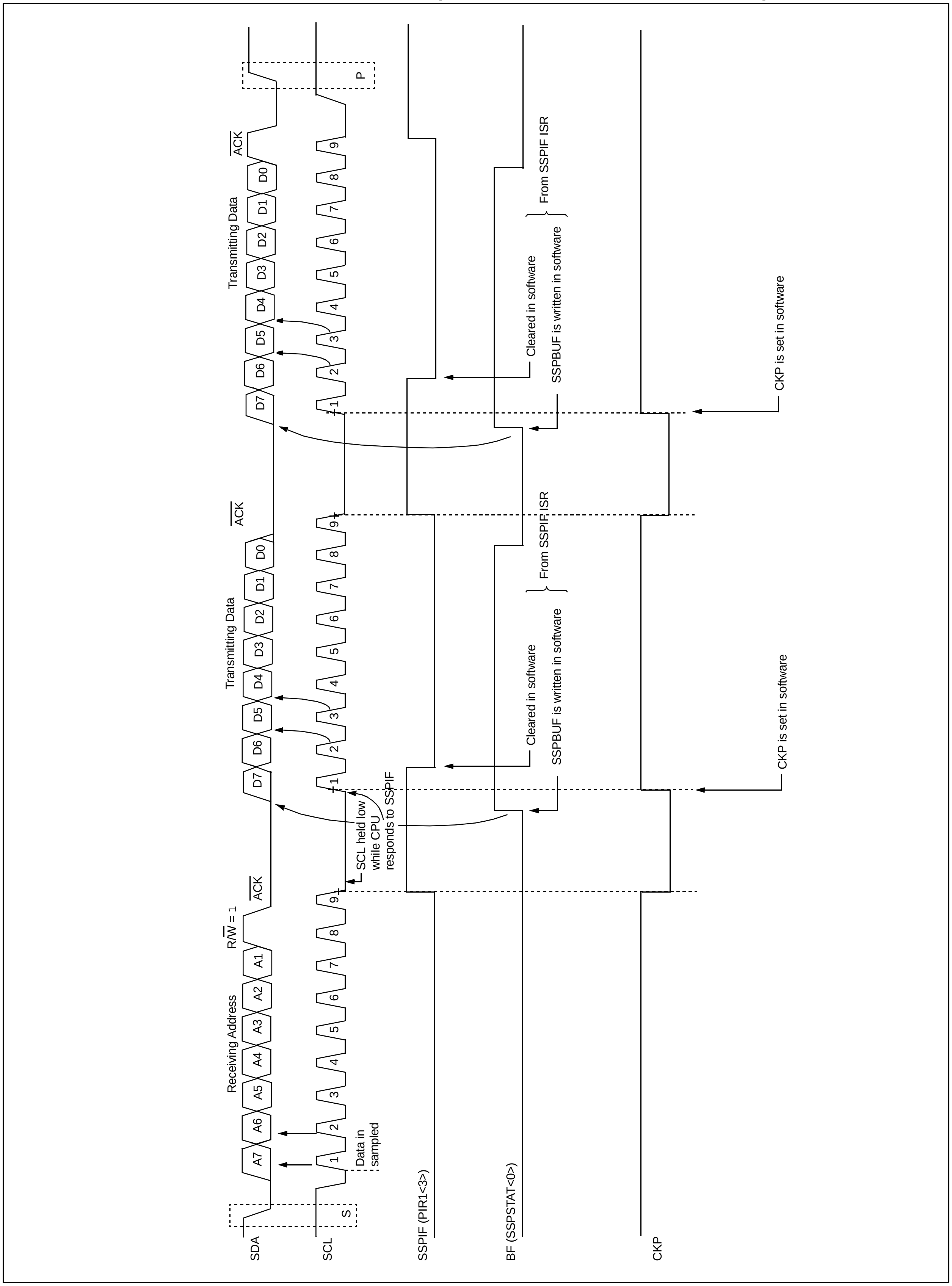


FIGURE 17-10: I²C SLAVE MODE TIMING WITH SEN = 0 (RECEPTION, 10-BIT ADDRESS)

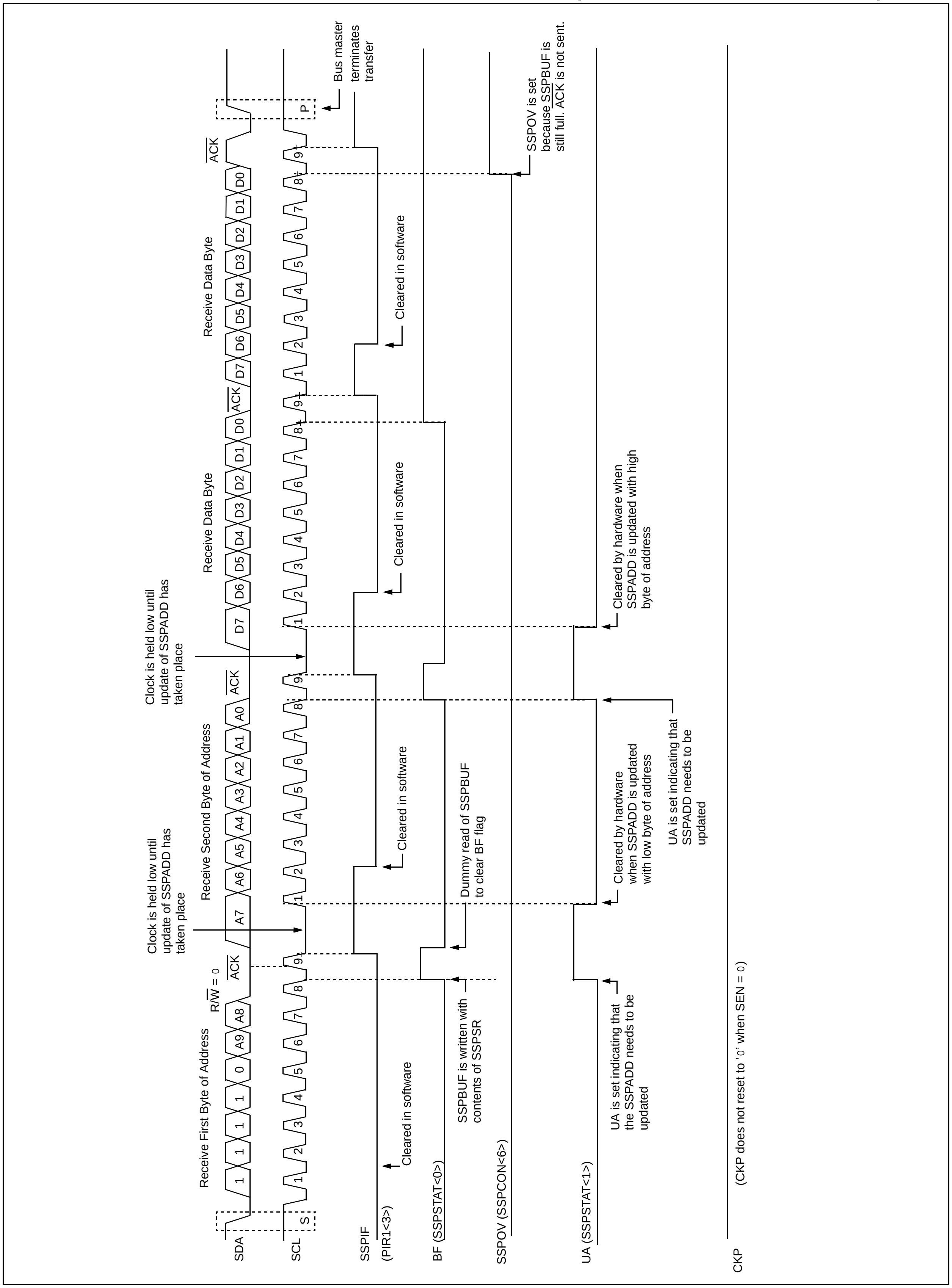
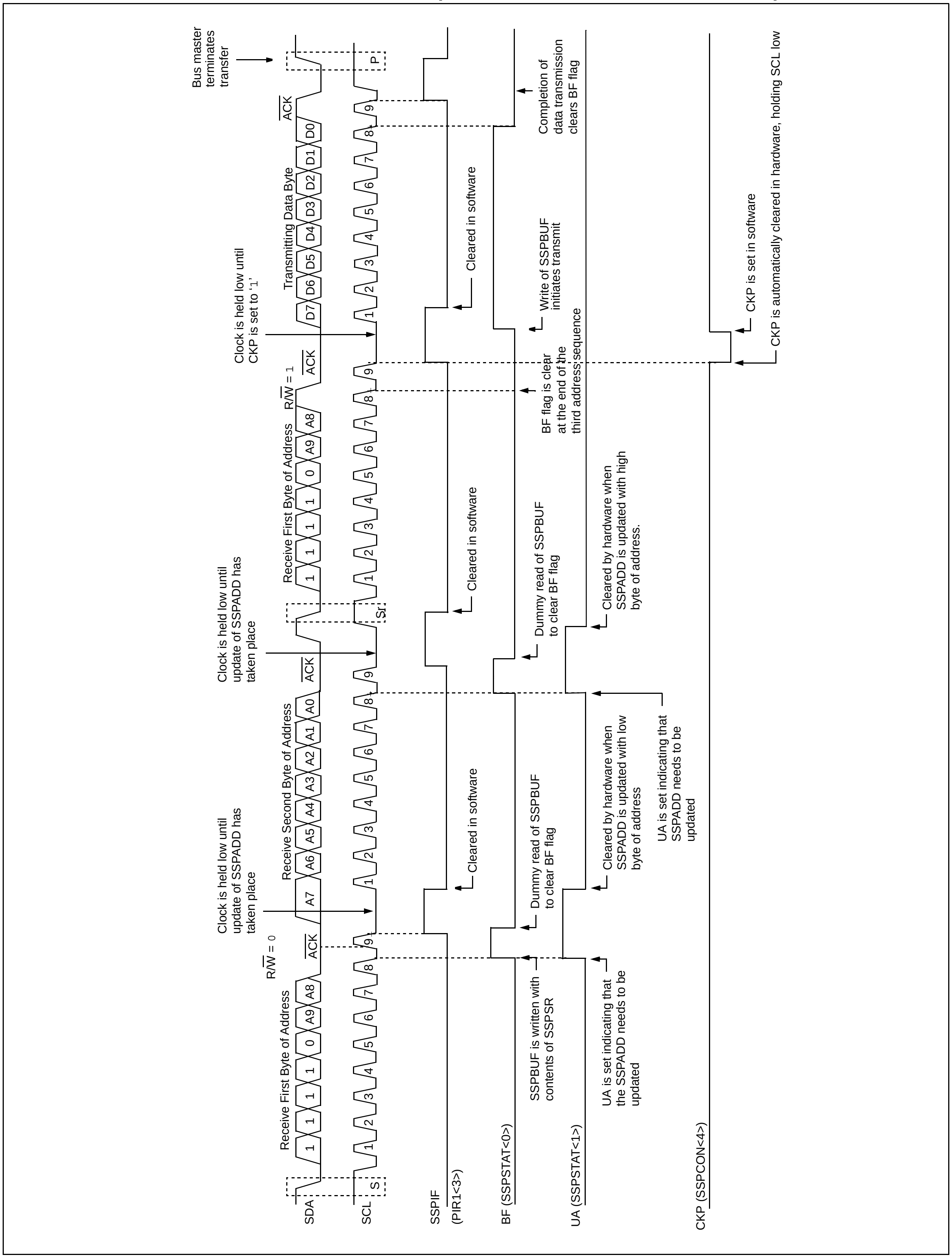


FIGURE 17-11: I²C SLAVE MODE TIMING (TRANSMISSION, 10-BIT ADDRESS)



17.4.4 CLOCK STRETCHING

Both 7- and 10-bit Slave modes implement automatic clock stretching during a transmit sequence.

The SEN bit (SSPCON2<0>) allows clock stretching to be enabled during receives. Setting SEN will cause the SCL pin to be held low at the end of each data receive sequence.

17.4.4.1 Clock Stretching for 7-bit Slave Receive Mode (SEN = 1)

In 7-bit Slave Receive mode, on the falling edge of the ninth clock at the end of the $\overline{ACK}$ sequence if the BF bit is set, the CKP bit in the SSPCON1 register is automatically cleared, forcing the SCL output to be held low. The CKP being cleared to '0' will assert the SCL line low. The CKP bit must be set in the user's ISR before reception is allowed to continue. By holding the SCL line low, the user has time to service the ISR and read the contents of the SSPBUF before the master device can initiate another receive sequence. This will prevent buffer overruns from occurring (see Figure 17-13).

> **Note 1:** If the user reads the contents of the SSPBUF before the falling edge of the ninth clock, thus clearing the BF bit, the CKP bit will not be cleared and clock stretching will not occur.
>
> **2:** The CKP bit can be set in software regardless of the state of the BF bit. The user should be careful to clear the BF bit in the ISR before the next receive sequence in order to prevent an overflow condition.

17.4.4.2 Clock Stretching for 10-bit Slave Receive Mode (SEN = 1)

In 10-bit Slave Receive mode, during the address sequence, clock stretching automatically takes place but CKP is not cleared. During this time, if the UA bit is set after the ninth clock, clock stretching is initiated. The UA bit is set after receiving the upper byte of the 10-bit address and following the receive of the second byte of the 10-bit address with the R/$\overline{W}$ bit cleared to '0'. The release of the clock line occurs upon updating SSPADD. Clock stretching will occur on each data receive sequence as described in 7-bit mode.

> **Note:** If the user polls the UA bit and clears it by updating the SSPADD register before the falling edge of the ninth clock occurs and if the user hasn't cleared the BF bit by reading the SSPBUF register before that time, then the CKP bit will still NOT be asserted low. Clock stretching on the basis of the state of the BF bit only occurs during a data sequence, not an address sequence.

17.4.4.3 Clock Stretching for 7-bit Slave Transmit Mode

7-bit Slave Transmit mode implements clock stretching by clearing the CKP bit after the falling edge of the ninth clock, if the BF bit is clear. This occurs regardless of the state of the SEN bit.

The user's ISR must set the CKP bit before transmission is allowed to continue. By holding the SCL line low, the user has time to service the ISR and load the contents of the SSPBUF before the master device can initiate another transmit sequence (see Figure 17-9).

> **Note 1:** If the user loads the contents of SSPBUF, setting the BF bit before the falling edge of the ninth clock, the CKP bit will not be cleared and clock stretching will not occur.
>
> **2:** The CKP bit can be set in software regardless of the state of the BF bit.

17.4.4.4 Clock Stretching for 10-bit Slave Transmit Mode

In 10-bit Slave Transmit mode, clock stretching is controlled during the first two address sequences by the state of the UA bit, just as it is in 10-bit Slave Receive mode. The first two addresses are followed by a third address sequence which contains the high order bits of the 10-bit address and the R/$\overline{W}$ bit set to '1'. After the third address sequence is performed, the UA bit is not set, the module is now configured in Transmit mode, and clock stretching is controlled by the BF flag as in 7-bit Slave Transmit mode (see Figure 17-11).

17.4.4.5 Clock Synchronization and the CKP bit

When the CKP bit is cleared, the SCL output is forced to '0'. However, setting 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 deasserted SCL. This ensures that a write to the CKP bit will not violate the minimum high time requirement for SCL (see Figure 17-12).

FIGURE 17-12: CLOCK SYNCHRONIZATION TIMING

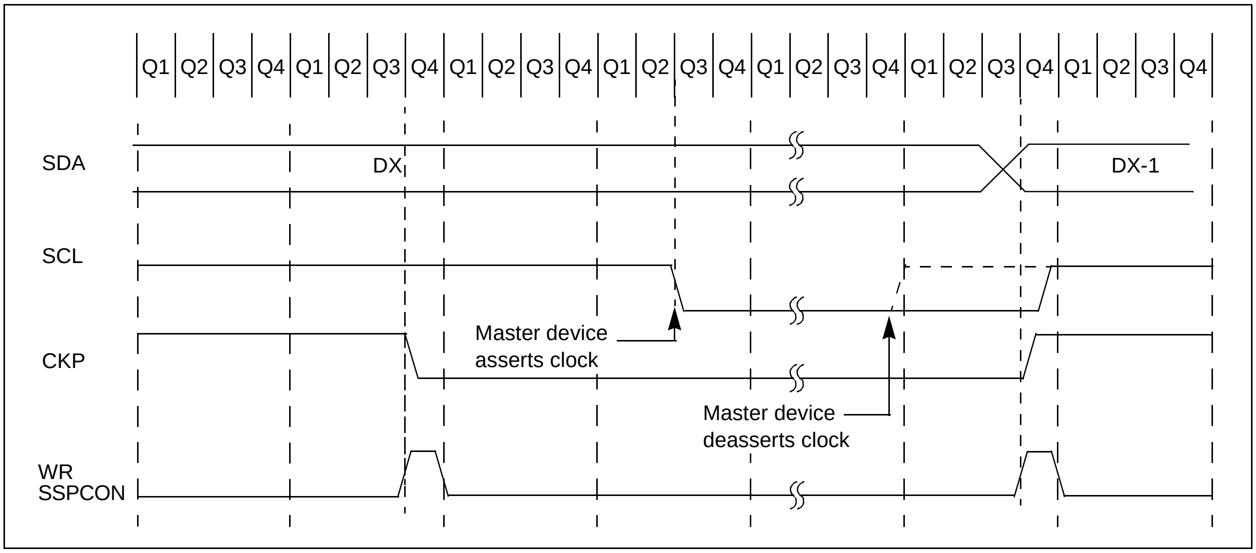


FIGURE 17-13: I²C SLAVE MODE TIMING WITH SEN = 1 (RECEPTION, 7-BIT ADDRESS)

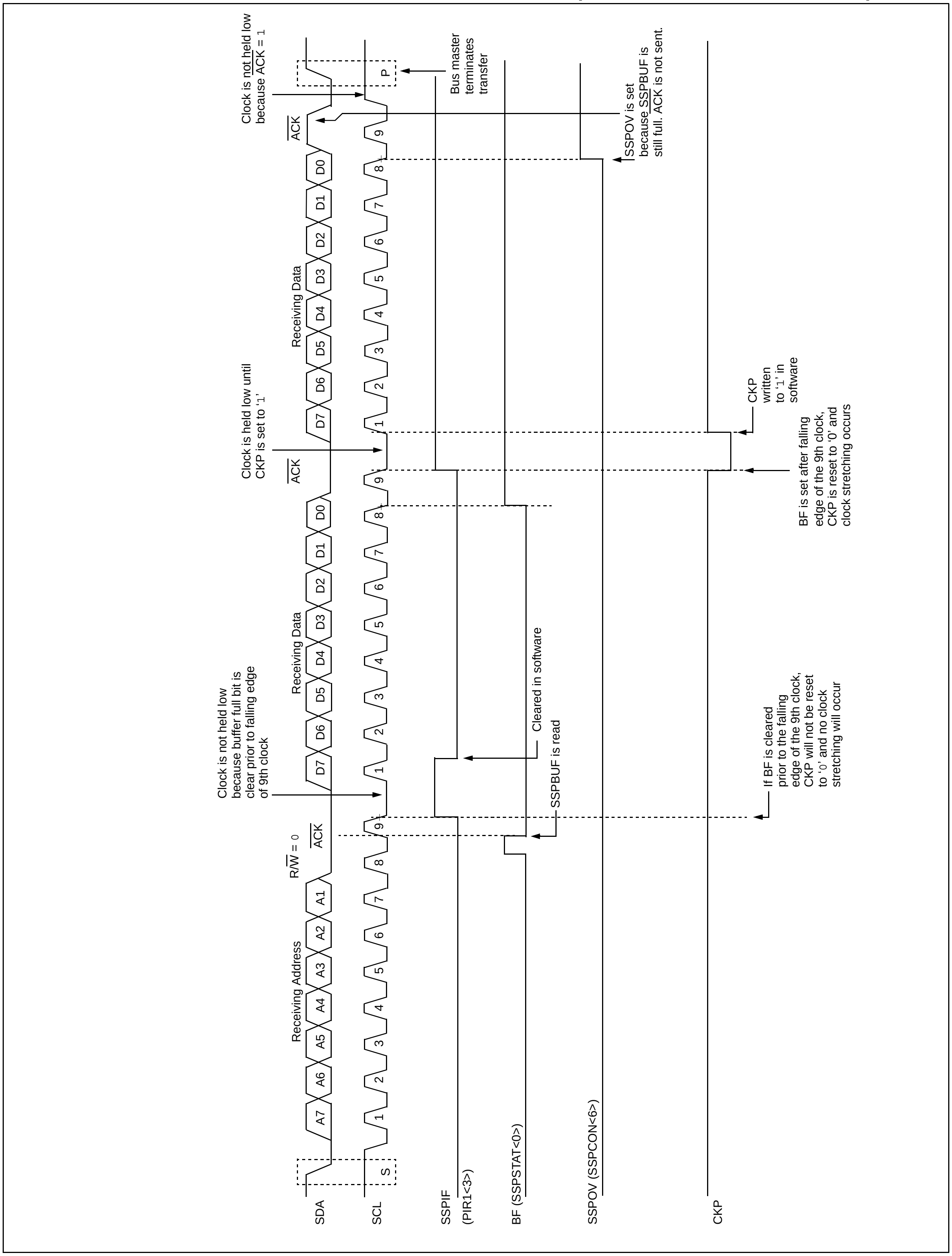
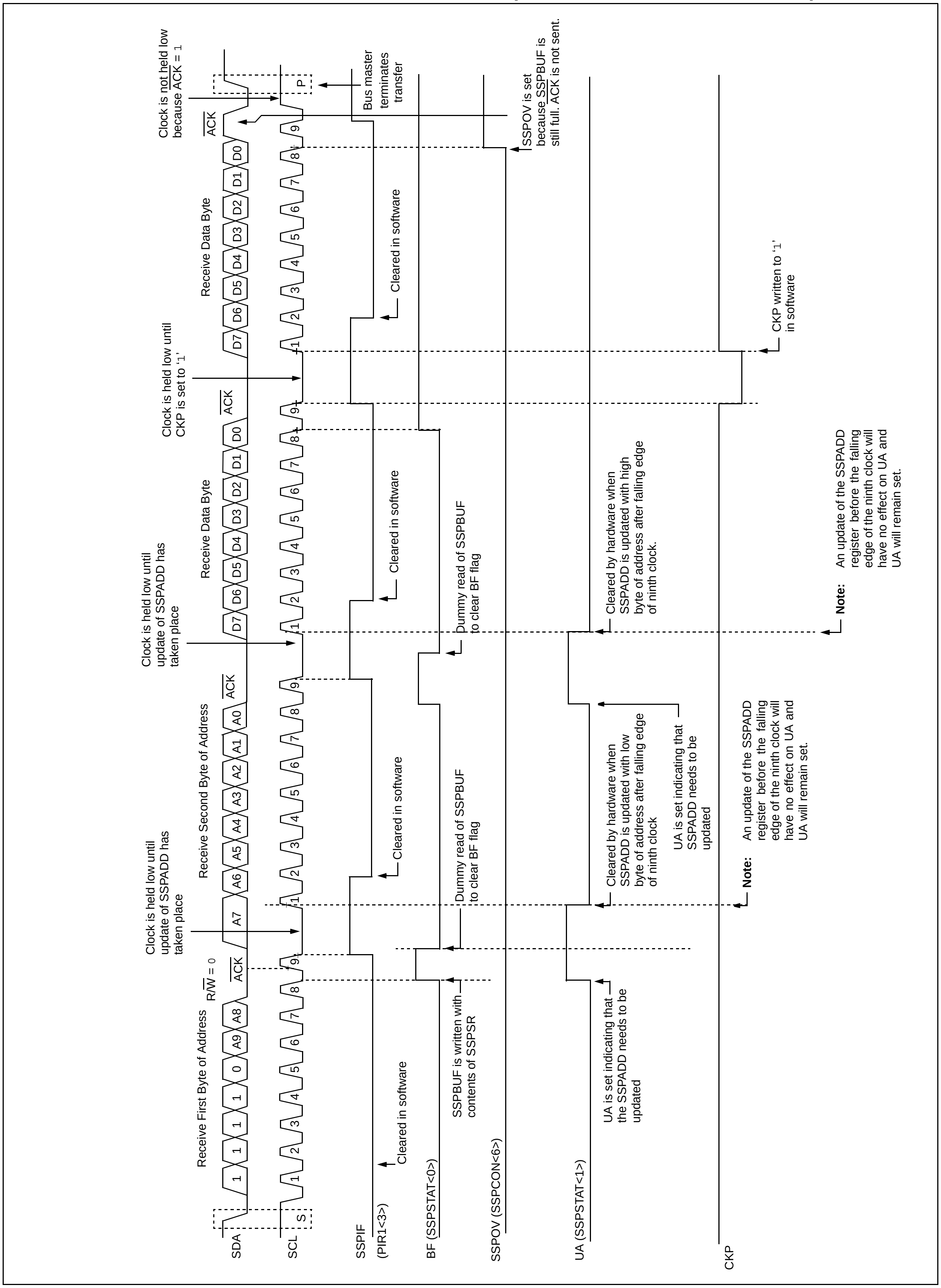


FIGURE 17-14: I²C SLAVE MODE TIMING SEN = 1 (RECEPTION, 10-BIT ADDRESS)



17.4.5 GENERAL CALL ADDRESS SUPPORT

The addressing procedure for the I²C bus is such that the first byte after the Start condition usually determines which device will be the slave addressed by the master. 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 one of eight addresses reserved for specific purposes by the I²C protocol. It consists of all '0's with R/$\overline{W}$ = 0.

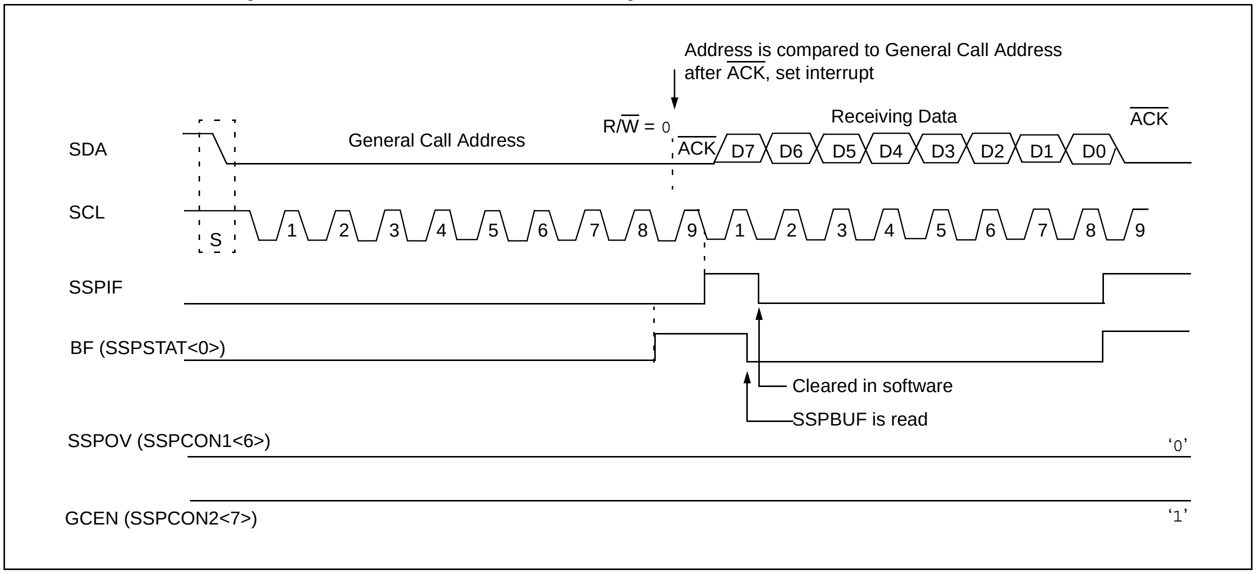
The general call address is recognized when the General Call Enable bit (GCEN) is enabled (SSPCON2<7> is set). Following a Start bit detect, 8 bits are shifted into the SSPSR and the address is compared against the SSPADD. It is also compared to the general call address and fixed in hardware.

If the general call address matches, the SSPSR is transferred to the SSPBUF, the BF flag bit is set (eighth bit) and on the falling edge of the ninth bit ($\overline{ACK}$ bit), the SSPIF interrupt flag bit is set.

When the interrupt is serviced, the source for the interrupt can be checked by reading the contents of the SSPBUF. The value can be used to determine if the address was device specific or a general call address.

In 10-bit mode, the SSPADD is required to be updated for the second half of the address to match and the UA bit is set (SSPSTAT<1>). If the general call address is sampled when the GCEN bit is set while the slave is configured in 10-bit Address mode, then the second half of the address is not necessary, the UA bit will not be set and the slave will begin receiving data after the Acknowledge (Figure 17-15).

FIGURE 17-15: SLAVE MODE GENERAL CALL ADDRESS SEQUENCE (7 OR 10-BIT ADDRESS MODE)



17.4.6 MASTER MODE

Master mode is enabled by setting and clearing the appropriate SSPM bits in SSPCON1 and by setting the SSPEN bit. In Master mode, the SCL and SDA lines are manipulated by the MSSP hardware.

Master mode of operation is supported by interrupt generation on the detection of the Start and Stop conditions. The Stop (P) and Start (S) bits are cleared from a Reset or when the MSSP module is disabled. Control of the I^2C bus may be taken when the P bit is set or the bus is Idle, with both the S and P bits clear.

In Firmware Controlled Master mode, user code conducts all I^2C bus operations based on Start and Stop bit conditions.

Once Master mode is enabled, the user has six options.

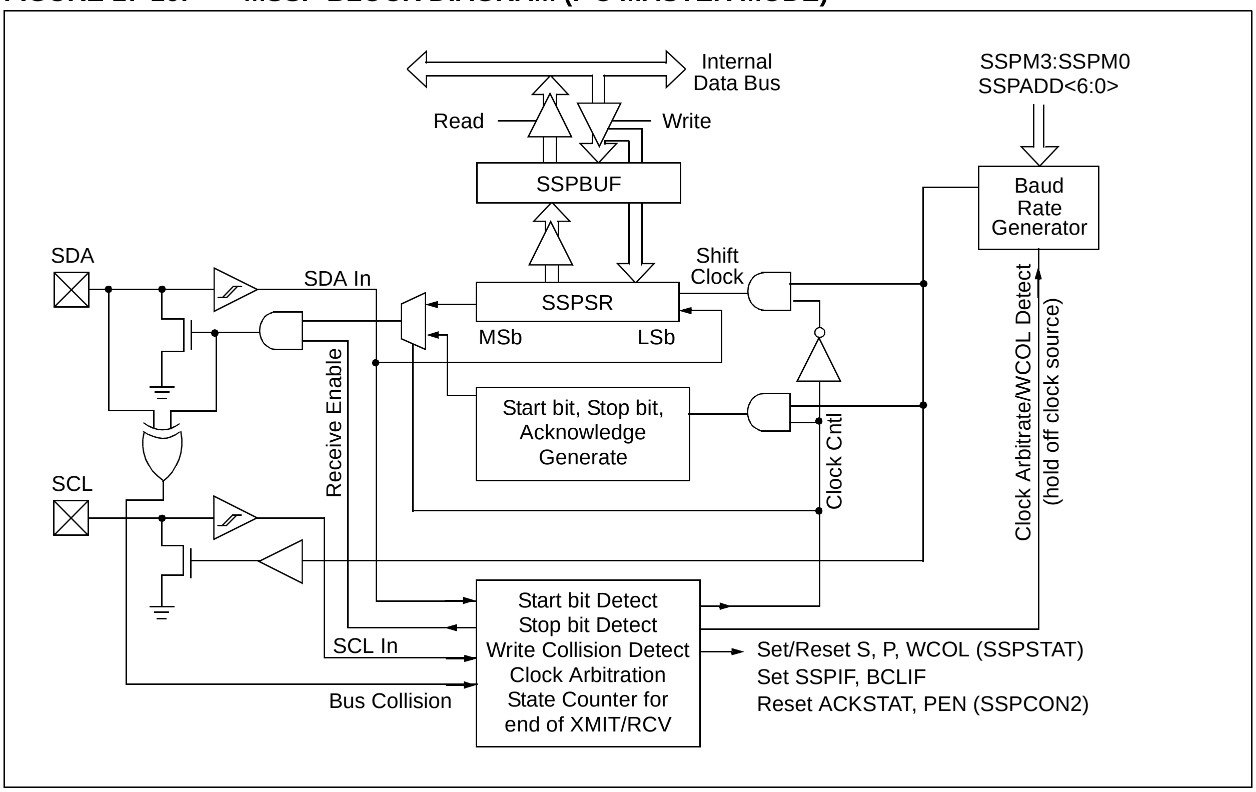
1. Assert a Start condition on SDA and SCL.
2. Assert a Repeated Start condition on SDA and SCL.
3. Write to the SSPBUF register initiating transmission of data/address.
4. Configure the I^2C port to receive data.
5. Generate an Acknowledge condition at the end of a received byte of data.
6. Generate a Stop condition on SDA and SCL.

Note: The MSSP module, when configured in I^2C Master mode, does not allow queueing of events. For instance, the user is not allowed to initiate a Start condition and immediately write the SSPBUF register to initiate transmission before the Start condition is complete. In this case, the SSPBUF will not be written to and the WCOL bit will be set, indicating that a write to the SSPBUF did not occur.

The following events will cause SSP interrupt flag bit, SSPIF, to be set (SSP interrupt if enabled):

- Start Condition
- Stop Condition
- Data Transfer Byte Transmitted/Received
- Acknowledge Transmit
- Repeated Start

FIGURE 17-16: MSSP BLOCK DIAGRAM (I^2C MASTER MODE)



17.4.6.1 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 Read/$\overline{\text{Write}}$ ($R/\overline{W}$) bit. In this case, the $R/\overline{W}$ bit will be logic '0'. Serial data is transmitted 8 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 a receive bit. Serial data is received via SDA while SCL outputs the serial clock. Serial data is received 8 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.

The Baud Rate Generator used for the SPI mode operation is used to set the SCL clock frequency for either 100 kHz, 400 kHz or 1 MHz I²C operation. See **Section 17.4.7 "Baud Rate Generator"** for more detail.

A typical transmit sequence would go as follows:

1. The user generates a Start condition by setting the Start enable bit, SEN (SSPCON2<0>).
2. SSPIF is set. The MSSP module will wait the required start time before any other operation takes place.
3. The user loads the SSPBUF with the slave address to transmit.
4. Address is shifted out the SDA pin until all 8 bits are transmitted.
5. The MSSP module shifts in the ACK bit from the slave device and writes its value into the SSPCON2 register (SSPCON2<6>).
6. The MSSP module generates an interrupt at the end of the ninth clock cycle by setting the SSPIF bit.
7. The user loads the SSPBUF with eight bits of data.
8. Data is shifted out the SDA pin until all 8 bits are transmitted.
9. The MSSP module shifts in the ACK bit from the slave device and writes its value into the SSPCON2 register (SSPCON2<6>).
10. The MSSP module generates an interrupt at the end of the ninth clock cycle by setting the SSPIF bit.
11. The user generates a Stop condition by setting the Stop enable bit PEN (SSPCON2<2>).
12. Interrupt is generated once the Stop condition is complete.

17.4.7 BAUD RATE GENERATOR

In I²C Master mode, the Baud Rate Generator (BRG) reload value is placed in the lower 7 bits of the SSPADD register (Figure 17-17). When a write occurs to SSPBUF, the Baud Rate Generator will automatically begin counting. The BRG counts down to '0' and stops until another reload has taken place. The BRG count is decremented twice per instruction cycle (TCY) on the Q2 and Q4 clocks. In I²C Master mode, the BRG is reloaded automatically.

Once the given operation is complete (i.e., transmission of the last data bit is followed by ACK), the internal clock will automatically stop counting and the SCL pin will remain in its last state.

Table 17-3 demonstrates clock rates based on instruction cycles and the BRG value loaded into SSPADD.

FIGURE 17-17: BAUD RATE GENERATOR BLOCK DIAGRAM

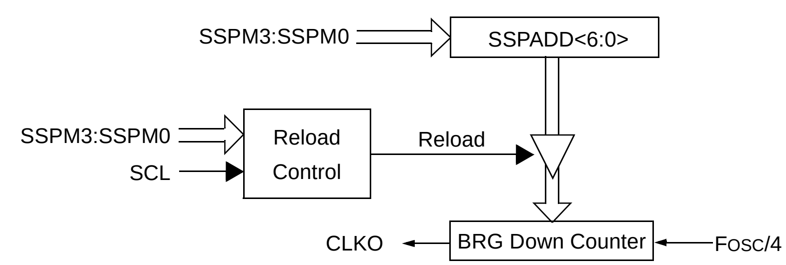


TABLE 17-3: I²C CLOCK RATE w/BRG

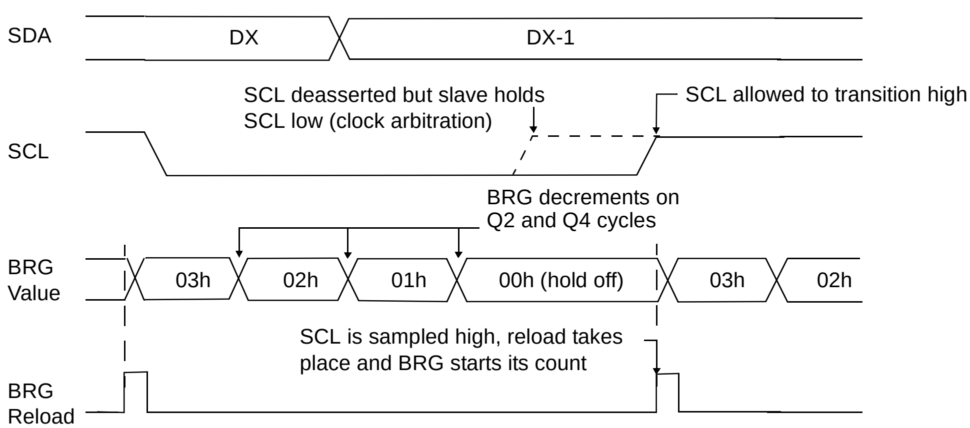
FCY	FCY*2	BRG Value	FSCL (2 Rollovers of BRG)
10 MHz	20 MHz	19h	400 kHz(1)
10 MHz	20 MHz	20h	312.5 kHz
10 MHz	20 MHz	64h	100 kHz
4 MHz	8 MHz	0Ah	400 kHz(1)
4 MHz	8 MHz	0Dh	308 kHz
4 MHz	8 MHz	28h	100 kHz
1 MHz	2 MHz	03h	333 kHz(1)
1 MHz	2 MHz	0Ah	100 kHz
1 MHz	2 MHz	00h	1 MHz(1)

Note 1: The I²C interface does not conform to the 400 kHz I²C specification (which applies to rates greater than 100 kHz) in all details but may be used with care where higher rates are required by the application.

17.4.7.1 Clock Arbitration

Clock arbitration occurs when the master, during any receive, transmit or Repeated Start/Stop condition, deasserts 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 SSPADD<6:0> and begins counting. This ensures that the SCL high time will always be at least one BRG rollover count in the event that the clock is held low by an external device (Figure 17-18).

FIGURE 17-18: BAUD RATE GENERATOR TIMING WITH CLOCK ARBITRATION



17.4.8 I²C MASTER MODE START CONDITION TIMING

To initiate a Start condition, the user sets the Start Condition Enable bit, SEN (SSPCON2<0>). If the SDA and SCL pins are sampled high, the Baud Rate Generator is reloaded with the contents of SSPADD<6:0> and starts its count. If SCL and SDA are both sampled high when the Baud Rate Generator times out (TBRG), the SDA pin is driven low. The action of the SDA being driven low while SCL is high is the Start condition and causes the S bit (SSPSTAT<3>) to be set. Following this, the Baud Rate Generator is reloaded with the contents of SSPADD<6:0> and resumes its count. When the Baud Rate Generator times out (TBRG), the SEN bit (SSPCON2<0>) will be automatically cleared by hardware, the Baud Rate Generator is suspended, leaving the SDA line held low and the Start condition is complete.

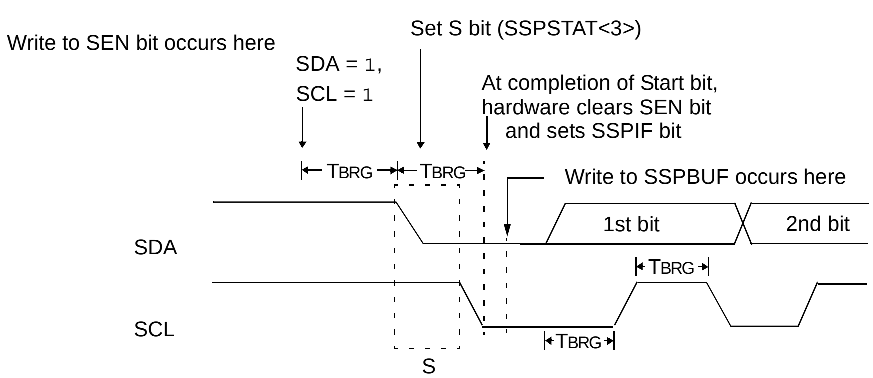
Note:	If at the beginning of the Start condition, the SDA and SCL pins are already sampled low or if during the Start condition, the SCL line is sampled low before the SDA line is driven low, a bus collision occurs, the Bus Collision Interrupt Flag, BCLIF, is set, the Start condition is aborted and the I²C module is reset into its Idle state.

17.4.8.1 WCOL Status Flag

If the user writes the SSPBUF when a Start sequence is in progress, the WCOL is set and the contents of the buffer are unchanged (the write doesn’t occur).

Note:	Because queueing of events is not allowed, writing to the lower 5 bits of SSPCON2 is disabled until the Start condition is complete.

FIGURE 17-19: FIRST START BIT TIMING



17.4.9 I²C MASTER MODE REPEATED START CONDITION TIMING

A Repeated Start condition occurs when the RSEN bit (SSPCON2<1>) is programmed high and the I²C logic module is in the Idle state. 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 with the contents of SSPADD<5:0> and begins counting. The SDA pin is released (brought high) for one Baud Rate Generator count (TBRG). When the Baud Rate Generator times out, if SDA is sampled high, the SCL pin will be deasserted (brought high). When SCL is sampled high, the Baud Rate Generator is reloaded with the contents of SSPADD<6:0> and begins counting. SDA and SCL must be sampled high for one TBRG. This action is then followed by assertion of the SDA pin (SDA = 0) for one TBRG while SCL is high. Following this, the RSEN bit (SSPCON2<1>) will be automatically cleared and the Baud Rate Generator will not be reloaded, leaving the SDA pin held low. As soon as a Start condition is detected on the SDA and SCL pins, the S bit (SSPSTAT<3>) will be set. The SSPIF bit will not be set until the Baud Rate Generator has timed out.

> **Note 1:** If RSEN is programmed while any other event is in progress, it will not take effect.
>
> **2:** A bus collision during the Repeated Start condition occurs if:
> - SDA is sampled low when SCL goes from low-to-high.
> - SCL goes low before SDA is asserted low. This may indicate that another master is attempting to transmit a data '1'.

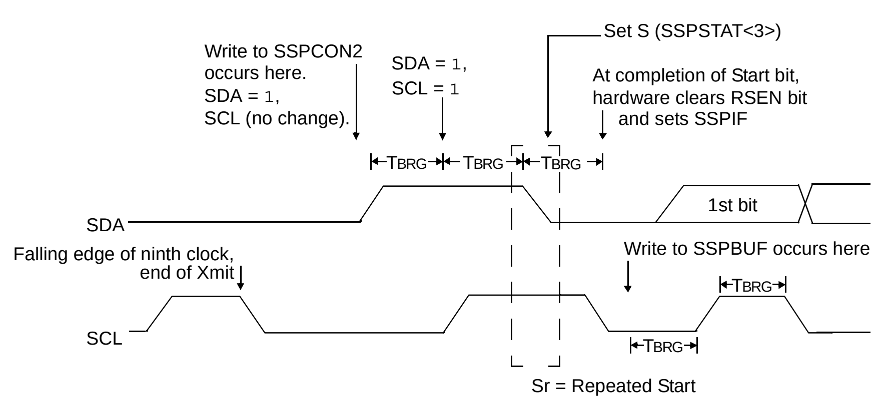
Immediately following the SSPIF bit getting set, the user may write the SSPBUF with the 7-bit address in 7-bit mode, or the default first address in 10-bit mode. After the first eight bits are transmitted and an ACK is received, the user may then transmit an additional eight bits of address (10-bit mode) or eight bits of data (7-bit mode).

17.4.9.1 WCOL Status Flag

If the user writes the SSPBUF when a Repeated Start sequence is in progress, the WCOL is set and the contents of the buffer are unchanged (the write doesn't occur).

> **Note:** Because queueing of events is not allowed, writing of the lower 5 bits of SSPCON2 is disabled until the Repeated Start condition is complete.

FIGURE 17-20: REPEAT START CONDITION WAVEFORM



17.4.10 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 SSPBUF 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 (see data hold time specification parameter #106). SCL is held low for one Baud Rate Generator rollover count (TBRG). Data should be valid before SCL is released high (see data setup time specification parameter #107). When the SCL pin is released high, it is held that way for TBRG. The data on the SDA pin must remain stable for that duration and some hold time after the next falling edge of SCL. After the eighth bit is shifted out (the falling edge of the eighth clock), the BF flag is cleared and the master releases SDA. This allows the slave device being addressed to respond with an $\overline{\text{ACK}}$ bit during the ninth bit time if an address match occurred, or if data was received properly. The status of $\overline{\text{ACK}}$ is written into the ACKDT bit on the falling 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 SSPIF bit is set and the master clock (Baud Rate Generator) is suspended until the next data byte is loaded into the SSPBUF, leaving SCL low and SDA unchanged (Figure 17-21).

After the write to the SSPBUF, each bit of the address will be shifted out on the falling edge of SCL until all seven address bits and the R/$\overline{\text{W}}$ bit are completed. On the falling edge of the eighth clock, the master will deassert the SDA pin, allowing the slave to respond with an Acknowledge. On the falling edge of the ninth clock, the master will sample the SDA pin to see if the address was recognized by a slave. The status of the $\overline{\text{ACK}}$ bit is loaded into the ACKSTAT status bit (SSPCON2<6>). Following the falling edge of the ninth clock transmission of the address, the SSPIF is set, the BF flag is cleared and the Baud Rate Generator is turned off until another write to the SSPBUF takes place, holding SCL low and allowing SDA to float.

17.4.10.1 BF Status Flag

In Transmit mode, the BF bit (SSPSTAT<0>) is set when the CPU writes to SSPBUF and is cleared when all 8 bits are shifted out.

17.4.10.2 WCOL Status Flag

If the user writes the SSPBUF when a transmit is already in progress (i.e., SSPSR is still shifting out a data byte), the WCOL is set and the contents of the buffer are unchanged (the write doesn't occur).

WCOL must be cleared in software.

17.4.10.3 ACKSTAT Status Flag

In Transmit mode, the ACKSTAT bit (SSPCON2<6>) is cleared when the slave has sent an Acknowledge ($\overline{\text{ACK}}$ = 0) and is set when the slave does not Acknowledge ($\overline{\text{ACK}}$ = 1). A slave sends an Acknowledge when it has recognized its address (including a general call) or when the slave has properly received its data.

17.4.11 I²C MASTER MODE RECEPTION

Master mode reception is enabled by programming the receive enable bit, RCEN (SSPCON2<3>).

Note:	The MSSP module must be in an Idle state before the RCEN bit is set or the RCEN bit will be disregarded.

The Baud Rate Generator begins counting and on each rollover, the state of the SCL pin changes (high-to-low/low-to-high) and data is shifted into the SSPSR. After the falling edge of the eighth clock, the receive enable flag is automatically cleared, the contents of the SSPSR are loaded into the SSPBUF, the BF flag bit is set, the SSPIF flag bit is set and the Baud Rate Generator is suspended from counting, holding SCL low. The MSSP is now in Idle state awaiting the next command. When the buffer is read by the CPU, the BF flag bit is automatically cleared. The user can then send an Acknowledge bit at the end of reception by setting the Acknowledge sequence enable bit, ACKEN (SSPCON2<4>).

17.4.11.1 BF Status Flag

In receive operation, the BF bit is set when an address or data byte is loaded into SSPBUF from SSPSR. It is cleared when the SSPBUF register is read.

17.4.11.2 SSPOV Status Flag

In receive operation, the SSPOV bit is set when 8 bits are received into the SSPSR and the BF flag bit is already set from a previous reception.

17.4.11.3 WCOL Status Flag

If the user writes the SSPBUF when a receive is already in progress (i.e., SSPSR is still shifting in a data byte), the WCOL bit is set and the contents of the buffer are unchanged (the write doesn't occur).

FIGURE 17-21: I²C MASTER MODE WAVEFORM (TRANSMISSION, 7 OR 10-BIT ADDRESS)

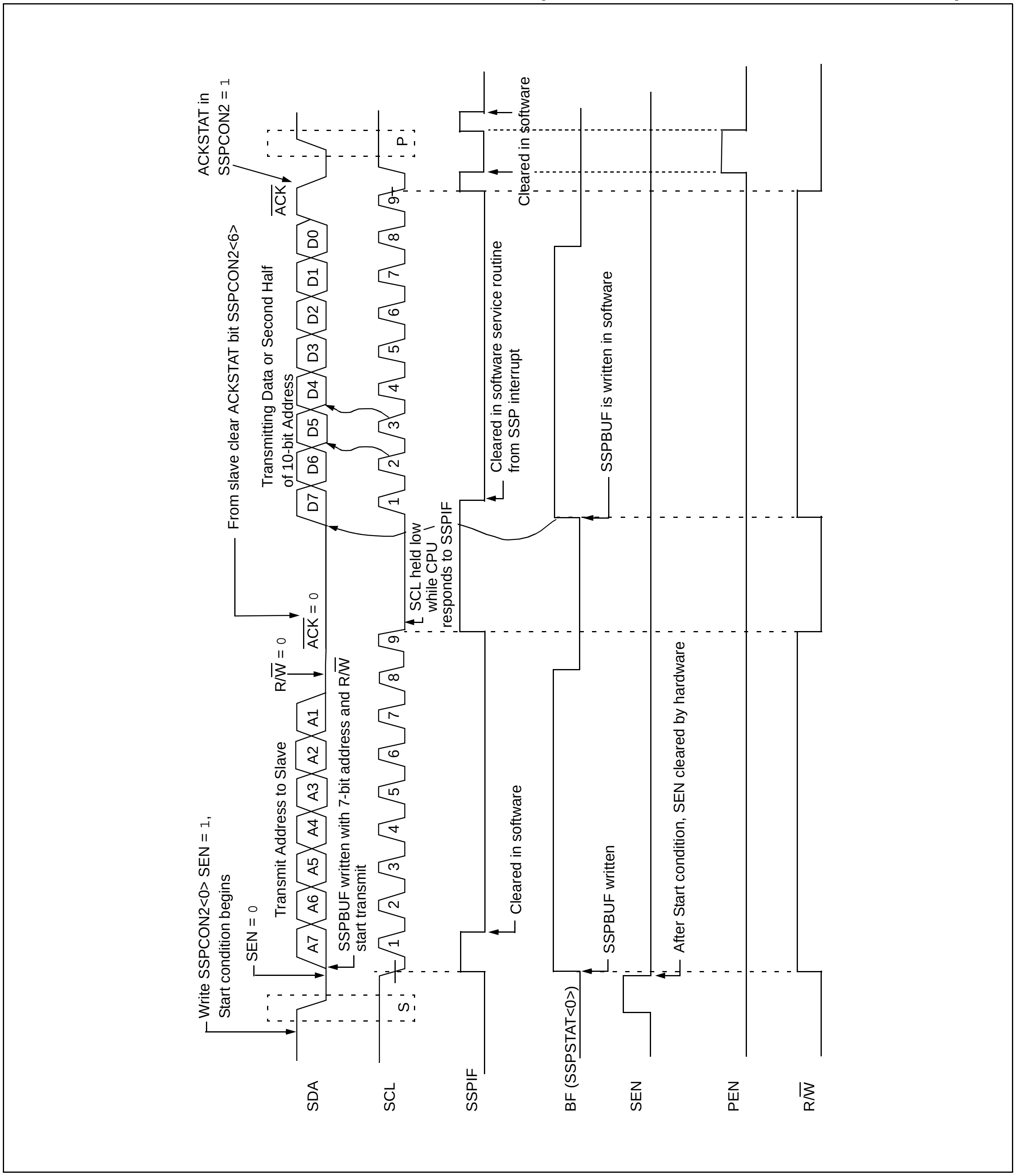
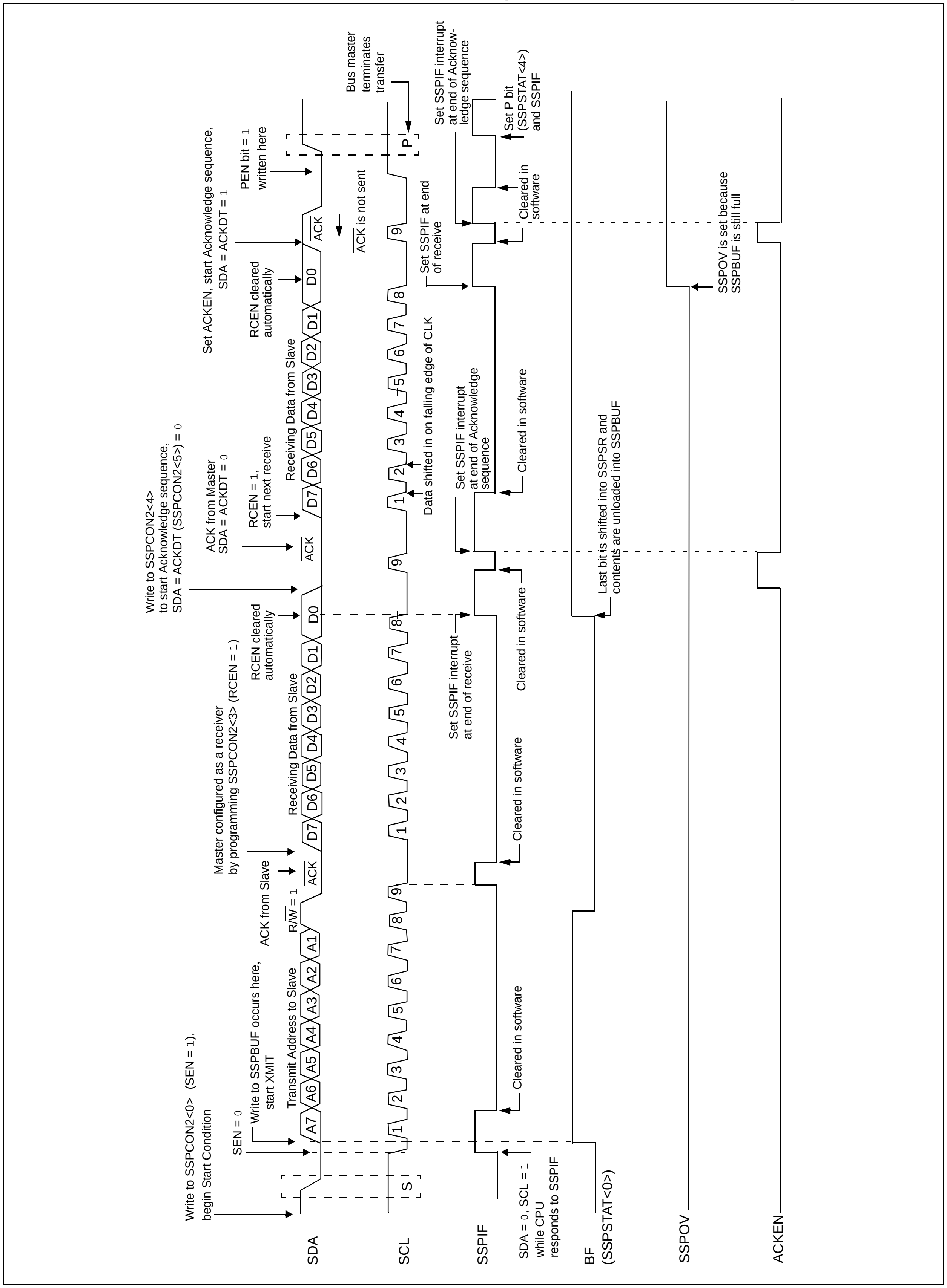


FIGURE 17-22: I²C MASTER MODE WAVEFORM (RECEPTION, 7-BIT ADDRESS)



17.4.12 ACKNOWLEDGE SEQUENCE TIMING

An Acknowledge sequence is enabled by setting the Acknowledge Sequence Enable bit, ACKEN (SSPCON2<4>). When this bit is set, the SCL pin is pulled low and the contents of the Acknowledge data bit are presented on the SDA pin. If the user wishes to generate an Acknowledge, then the ACKDT bit should be cleared. If not, the user should set the ACKDT bit before starting an Acknowledge sequence. The Baud Rate Generator then counts for one rollover period (TBRG) and the SCL pin is deasserted (pulled high). When the SCL pin is sampled high (clock arbitration), the Baud Rate Generator counts for TBRG. The SCL pin is then pulled low. Following this, the ACKEN bit is automatically cleared, the Baud Rate Generator is turned off and the MSSP module then goes into Idle mode (Figure 17-23).

17.4.12.1 WCOL Status Flag

If the user writes the SSPBUF when an Acknowledge sequence is in progress, then WCOL is set and the contents of the buffer are unchanged (the write doesn't occur).

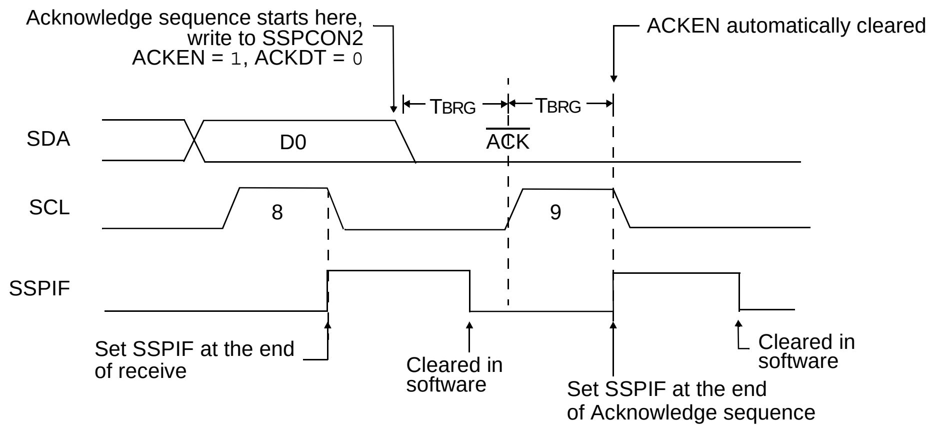
17.4.13 STOP CONDITION TIMING

A Stop bit is asserted on the SDA pin at the end of a receive/transmit by setting the Stop Sequence Enable bit, PEN (SSPCON2<2>). At the end of a receive/transmit, the SCL line is held low after the falling edge of the ninth clock. When the PEN bit is set, the master will assert the SDA line low. When the SDA line is sampled low, the Baud Rate Generator is reloaded and counts down to '0'. When the Baud Rate Generator times out, the SCL pin will be brought high and one TBRG (Baud Rate Generator rollover count) later, the SDA pin will be deasserted. When the SDA pin is sampled high while SCL is high, the P bit (SSPSTAT<4>) is set. A TBRG later, the PEN bit is cleared and the SSPIF bit is set (Figure 17-24).

17.4.13.1 WCOL Status Flag

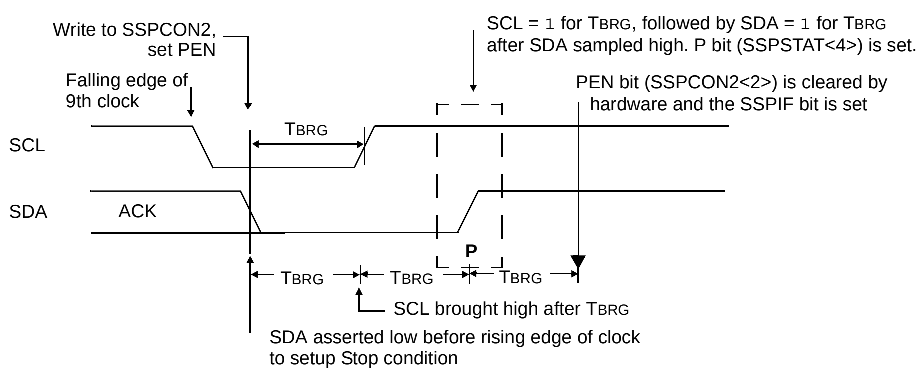
If the user writes the SSPBUF when a Stop sequence is in progress, then the WCOL bit is set and the contents of the buffer are unchanged (the write doesn't occur).

FIGURE 17-23: ACKNOWLEDGE SEQUENCE WAVEFORM



Note: TBRG = one Baud Rate Generator period.

FIGURE 17-24: STOP CONDITION RECEIVE OR TRANSMIT MODE



Note: TBRG = one Baud Rate Generator period.

17.4.14 SLEEP OPERATION

While in Sleep mode, the I²C 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).

17.4.15 EFFECT OF A RESET

A Reset disables the MSSP module and terminates the current transfer.

17.4.16 MULTI-MASTER MODE

In Multi-Master mode, the interrupt generation on the detection of the Start and Stop conditions allows the determination of when the bus is free. The Stop (P) and Start (S) bits are cleared from a Reset or when the MSSP module is disabled. Control of the I²C bus may be taken when the P bit (SSPSTAT<4>) 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 in hardware with the result placed in the BCLIF bit.

The states where arbitration can be lost are:

- Address Transfer
- Data Transfer
- A Start Condition
- A Repeated Start Condition
- An Acknowledge Condition

17.4.17 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 = 0, then a bus collision has taken place. The master will set the Bus Collision Interrupt Flag, BCLIF and reset the I²C port to its Idle state (Figure 17-25).

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 SSPBUF can be written to. When the user services the bus collision Interrupt Service Routine and if the I²C bus is free, the user can resume communication by asserting a Start condition.

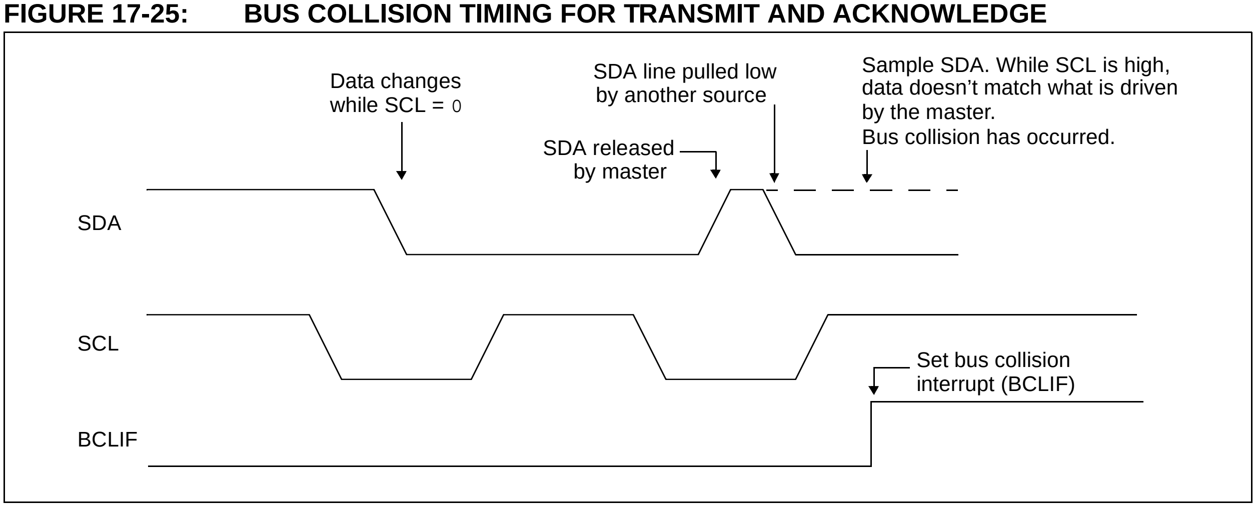
If a Start, Repeated Start, Stop, or Acknowledge condition was in progress when the bus collision occurred, the condition is aborted, the SDA and SCL lines are deasserted, and the respective control bits in the SSPCON2 register are cleared. When the user services the bus collision Interrupt Service Routine and if the I²C bus is free, the user can resume communication by asserting a Start condition.

The master will continue to monitor the SDA and SCL pins. If a Stop condition occurs, the SSPIF bit will be set.

A write to the SSPBUF will start the transmission of data at the first data bit regardless of where the transmitter left off when the bus collision occurred.

In Multi-Master mode, the interrupt generation on the detection of Start and Stop conditions allows the determination of when the bus is free. Control of the I²C bus can be taken when the P bit is set in the SSPSTAT register or the bus is Idle and the S and P bits are cleared.

FIGURE 17-25: BUS COLLISION TIMING FOR TRANSMIT AND ACKNOWLEDGE



17.4.17.1 Bus Collision During a Start Condition

During a Start condition, a bus collision occurs if:

a) SDA or SCL are sampled low at the beginning of the Start condition (Figure 17-26).

b) SCL is sampled low before SDA is asserted low (Figure 17-27).

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 BCLIF flag is set, and
- the MSSP module is reset to its Idle state (Figure 17-26).

The Start condition begins with the SDA and SCL pins deasserted. When the SDA pin is sampled high, the Baud Rate Generator is loaded from SSPADD<6:0> and counts down to '0'. If the SCL pin is sampled low while SDA is high, a bus collision occurs because it is assumed that another master is attempting to drive a data '1' during the Start condition.

If the SDA pin is sampled low during this count, the BRG is reset and the SDA line is asserted early (Figure 17-28). 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 '0' and during this time, if the SCL pins are sampled as '0', a bus collision does not occur. At the end of the BRG count, the SCL pin is asserted low.

> **Note:** The reason that bus collision is not a factor during a Start condition is that no two bus masters can assert a Start condition at the exact same time. Therefore, one master will always assert SDA before the other. This condition does not cause a bus collision because the two masters must be allowed to arbitrate the first address following the Start condition. If the address is the same, arbitration must be allowed to continue into the data portion, Repeated Start or Stop conditions.

FIGURE 17-26: BUS COLLISION DURING START CONDITION (SDA ONLY)

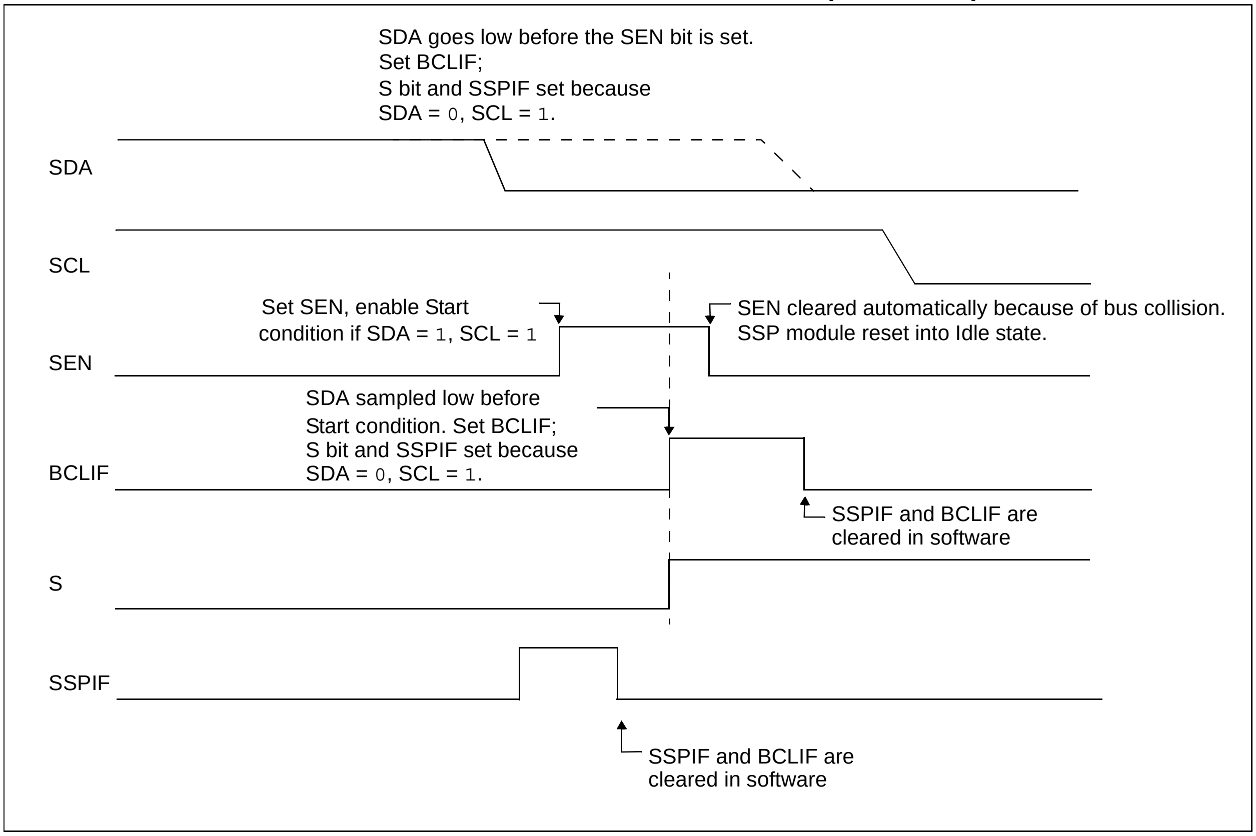


FIGURE 17-27: BUS COLLISION DURING START CONDITION (SCL = 0)

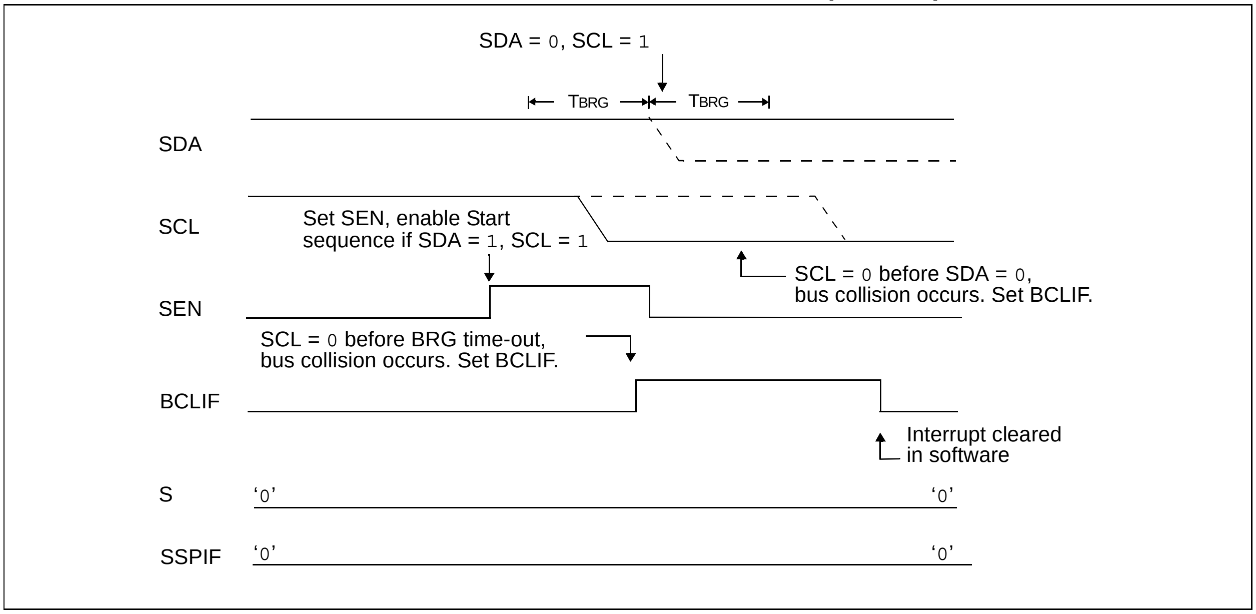
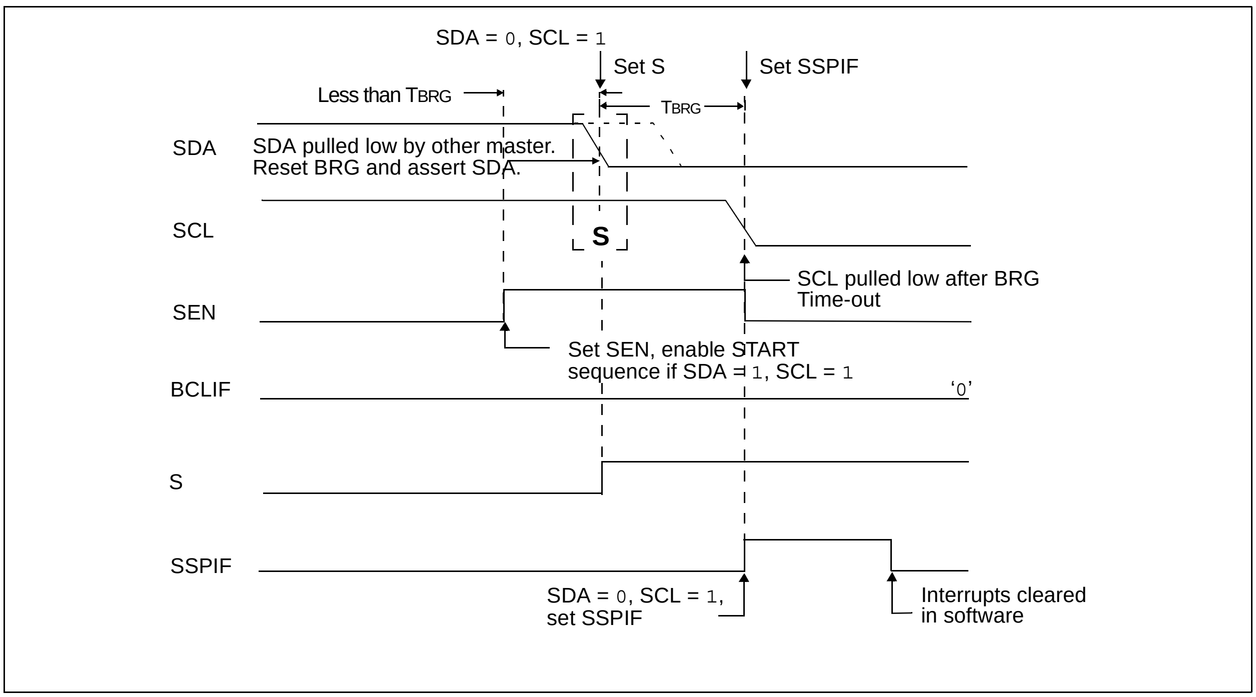


FIGURE 17-28: BRG RESET DUE TO SDA ARBITRATION DURING START CONDITION



17.4.17.2 Bus Collision During a Repeated Start Condition

During a Repeated Start condition, a bus collision occurs if:

a) A low level is sampled on SDA when SCL goes from low level to high level.
b) SCL goes low before SDA is asserted low, indicating that another master is attempting to transmit a data '1'.

When the user deasserts SDA and the pin is allowed to float high, the BRG is loaded with SSPADD<6:0> and counts down to '0'. 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', see Figure 17-29). If SDA is sampled high, the BRG is reloaded and begins counting. If SDA goes from high to low before the BRG times out, no bus collision occurs because no two masters can assert SDA at exactly the same time.

If SCL goes from high to low before the BRG times out and SDA has not already been asserted, a bus collision occurs. In this case, another master is attempting to transmit a data '1' during the Repeated Start condition (see Figure 17-30).

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 17-29: BUS COLLISION DURING A REPEATED START CONDITION (CASE 1)

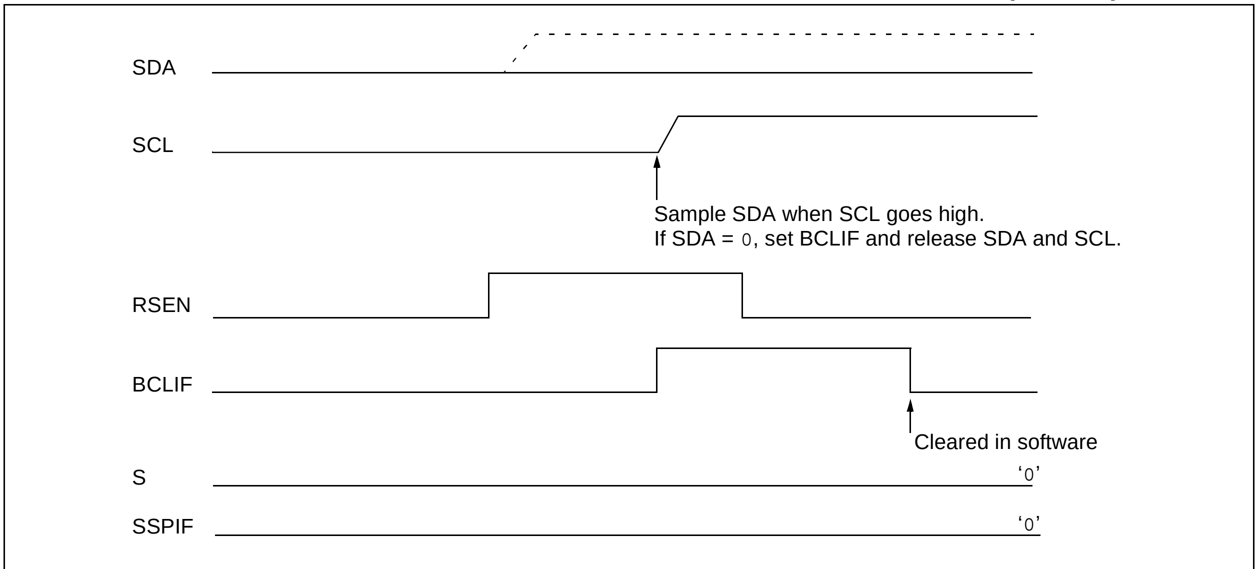
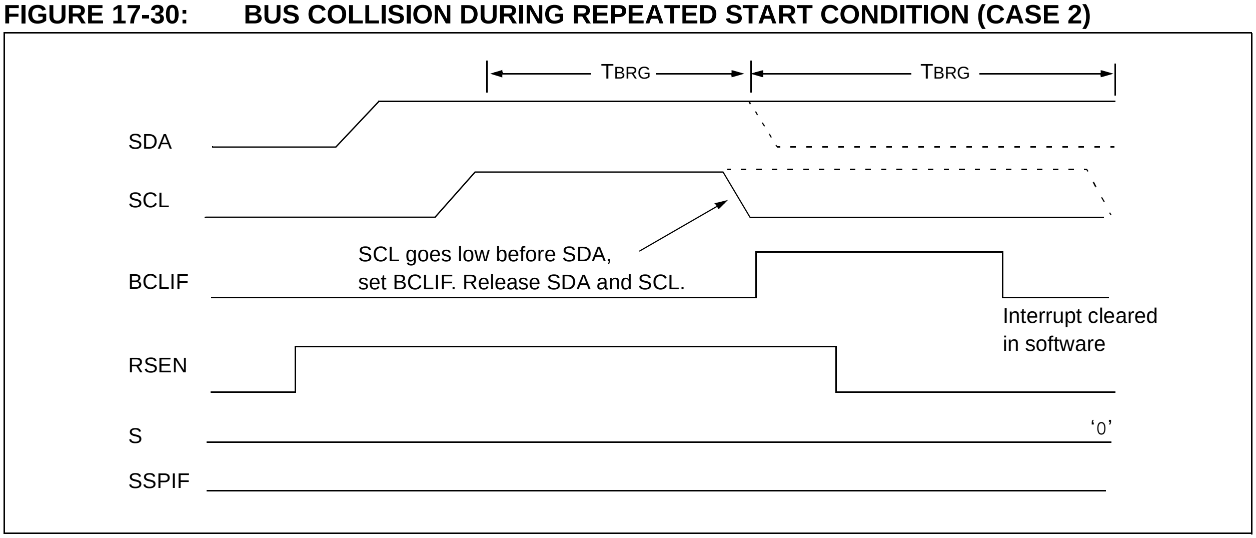


FIGURE 17-30: BUS COLLISION DURING REPEATED START CONDITION (CASE 2)



17.4.17.3 Bus Collision During a Stop Condition

Bus collision occurs during a Stop condition if:

a) After the SDA pin has been deasserted and allowed to float high, SDA is sampled low after the BRG has timed out.

b) After the SCL pin is deasserted, SCL is sampled low before SDA goes high.

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 SSPADD<6:0> and counts down to '0'. 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 17-31). 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 17-32).

FIGURE 17-31: BUS COLLISION DURING A STOP CONDITION (CASE 1)

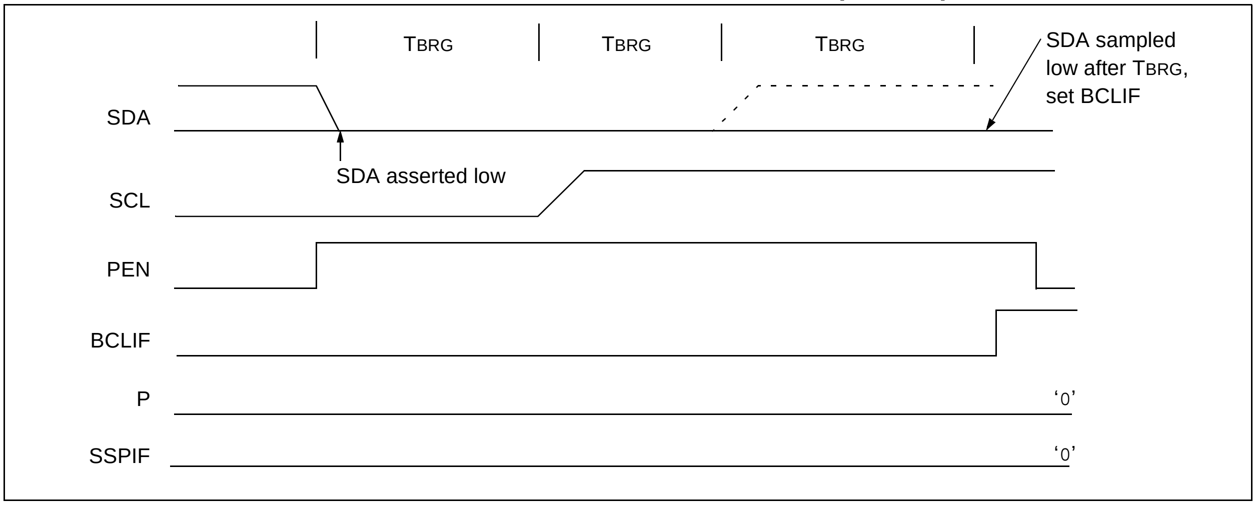
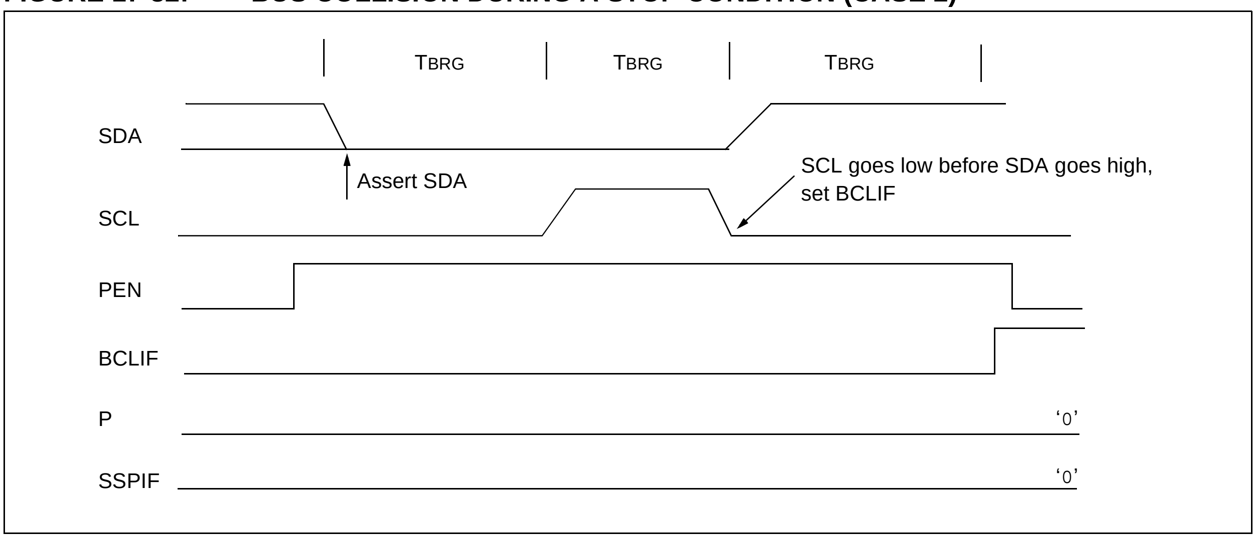


FIGURE 17-32: BUS COLLISION DURING A STOP CONDITION (CASE 2)



NOTES:

18.0 ENHANCED UNIVERSAL SYNCHRONOUS ASYNCHRONOUS RECEIVER TRANSMITTER (USART)

The Universal Synchronous Asynchronous Receiver Transmitter (USART) module is one of the two serial I/O modules. (USART is also known as a Serial Communications Interface or SCI.) The USART can be configured as a full-duplex asynchronous system that can communicate with peripheral devices, such as CRT terminals and personal computers. It can also be configured as a half-duplex synchronous system that can communicate with peripheral devices, such as A/D or D/A integrated circuits, serial EEPROMs, etc.

The Enhanced USART module implements additional features, including automatic baud rate detection and calibration, automatic wake-up on sync break reception and 12-bit break character transmit. These make it ideally suited for use in Local Interconnect Network bus (LIN bus) systems.

The USART can be configured in the following modes:

- Asynchronous (full-duplex) with:
  - Auto-wake-up on character reception
  - Auto-baud calibration
  - 12-bit break character transmission
- Synchronous – Master (half-duplex) with selectable clock polarity
- Synchronous – Slave (half-duplex) with selectable clock polarity

In order to configure pins RC6/TX/CK and RC7/RX/DT as the Universal Synchronous Asynchronous Receiver Transmitter:

- SPEN (RCSTA<7>) bit must be set (= `1`),
- TRISC<6> bit must be set (= `1`), and
- TRISC<7> bit must be set (= `1`).

Note:	The USART control will automatically reconfigure the pin from input to output as needed.

The operation of the Enhanced USART module is controlled through three registers:

- Transmit Status and Control (TXSTA)
- Receive Status and Control (RCSTA)
- Baud Rate Control (BAUDCON)

These are detailed on the following pages in Register 18-1, Register 18-2 and Register 18-3, respectively.

REGISTER 18-1: TXSTA: TRANSMIT STATUS AND CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-1	R/W-0
CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D
bit 7							bit 0

bit 7 **CSRC:** Clock Source Select bit
Asynchronous mode:
Don't care.
Synchronous mode:
1 = Master mode (clock generated internally from BRG)
0 = Slave mode (clock from external source)

bit 6 **TX9:** 9-bit Transmit Enable bit
1 = Selects 9-bit transmission
0 = Selects 8-bit transmission

bit 5 **TXEN:** Transmit Enable bit
1 = Transmit enabled
0 = Transmit disabled

Note: SREN/CREN overrides TXEN in Sync mode.

bit 4 **SYNC:** USART Mode Select bit
1 = Synchronous mode
0 = Asynchronous mode

bit 3 **SENDB:** Send Break Character bit
Asynchronous mode:
1 = Send sync break on next transmission (cleared by hardware upon completion)
0 = Sync break transmission completed
Synchronous mode:
Don't care.

bit 2 **BRGH:** High Baud Rate Select bit
Asynchronous mode:
1 = High speed
0 = Low speed
Synchronous mode:
Unused in this mode.

bit 1 **TRMT:** Transmit Shift Register Status bit
1 = TSR empty
0 = TSR full

bit 0 **TX9D:** 9th bit of Transmit Data
Can be address/data bit or a parity bit.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

REGISTER 18-2: RCSTA: RECEIVE STATUS AND CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0	R-0	R-x
SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D
bit 7							bit 0

bit 7 **SPEN:** Serial Port Enable bit
1 = Serial port enabled (configures RX/DT and TX/CK pins as serial port pins)
0 = Serial port disabled (held in Reset)

bit 6 **RX9:** 9-bit Receive Enable bit
1 = Selects 9-bit reception
0 = Selects 8-bit reception

bit 5 **SREN:** Single Receive Enable bit
Asynchronous mode:
Don't care.
Synchronous mode – Master:
1 = Enables single receive
0 = Disables single receive
This bit is cleared after reception is complete.
Synchronous mode – Slave:
Don't care.

bit 4 **CREN:** Continuous Receive Enable bit
Asynchronous mode:
1 = Enables receiver
0 = Disables receiver
Synchronous mode:
1 = Enables continuous receive until enable bit CREN is cleared (CREN overrides SREN)
0 = Disables continuous receive

bit 3 **ADDEN:** Address Detect Enable bit
Asynchronous mode 9-bit (RX9 = 1):
1 = Enables address detection, enables interrupt and loads the receive buffer when RSR<8> is set
0 = Disables address detection, all bytes are received and ninth bit can be used as parity bit
Asynchronous mode 9-bit (RX9 = 0):
Don't care.

bit 2 **FERR:** Framing Error bit
1 = Framing error (can be updated by reading RCREG register and receiving next valid byte)
0 = No framing error

bit 1 **OERR:** Overrun Error bit
1 = Overrun error (can be cleared by clearing bit CREN)
0 = No overrun error

bit 0 **RX9D:** 9th bit of Received Data
This can be an address/data bit or a parity bit and must be calculated by user firmware.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
- n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

REGISTER 18-3: BAUDCON: BAUD RATE CONTROL REGISTER

U-0	R-1	U-0	R/W-0	R/W-0	U-0	R/W-0	R/W-0
—	RCIDL	—	SCKP	BRG16	—	WUE	ABDEN
bit 7							bit 0

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

bit 6 **RCIDL**: Receive Operation Idle Status bit
1 = Receive operation is Idle
0 = Receive operation is active

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

bit 4 **SCKP**: Synchronous Clock Polarity Select bit
Asynchronous mode:
Unused in this mode.
Synchronous mode:
1 = Idle state for clock (CK) is a high level
0 = Idle state for clock (CK) is a low level

bit 3 **BRG16:** 16-bit Baud Rate Register Enable bit
1 = 16-bit Baud Rate Generator – SPBRGH and SPBRG
0 = 8-bit Baud Rate Generator – SPBRG only (Compatible mode), SPBRGH value ignored

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

bit 1 **WUE:** Wake-up Enable bit
Asynchronous mode:
1 = USART will continue to sample the RX pin – interrupt generated on falling edge; bit cleared in hardware on following rising edge
0 = RX pin not monitored or rising edge detected
Synchronous mode:
Unused in this mode.

bit 0 **ABDEN**: Auto-Baud Detect Enable bit
Asynchronous mode:
1 = Enable baud rate measurement on the next character – requires reception of a sync field (55h); cleared in hardware upon completion
0 = Baud rate measurement disabled or completed
Synchronous mode:
Unused in this mode.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

18.1 USART Baud Rate Generator (BRG)

The BRG is a dedicated 8-bit or 16-bit generator that supports both the Asynchronous and Synchronous modes of the USART. By default, the BRG operates in 8-bit mode; setting the BRG16 bit (BAUDCON<3>) selects 16-bit mode.

The SPBRGH:SPBRG register pair controls the period of a free-running timer. In Asynchronous mode, bits BRGH (TXSTA<2>) and BRG16 also control the baud rate. In Synchronous mode, bit BRGH is ignored. Table 18-1 shows the formula for computation of the baud rate for different USART modes which only apply in Master mode (internally generated clock).

Given the desired baud rate and FOSC, the nearest integer value for the SPBRGH:SPBRG registers can be calculated using the formulas in Table 18-1. From this, the error in baud rate can be determined. An example calculation is shown in Example 18-1. Typical baud rates and error values for the various Asynchronous modes are shown in Table 18-2. It may be advantageous to use the high baud rate (BRGH = 1) or the 16-bit BRG to reduce the baud rate error, or achieve a slow baud rate for a fast oscillator frequency.

Writing a new value to the SPBRGH:SPBRG registers causes the BRG timer to be reset (or cleared). This ensures the BRG does not wait for a timer overflow before outputting the new baud rate.

18.1.1 SAMPLING

The data on the RC7/RX/DT pin is sampled three times by a majority detect circuit to determine if a high or a low level is present at the RX pin.

TABLE 18-1: BAUD RATE FORMULAS

Configuration Bits			BRG/USART Mode	Baud Rate Formula
SYNC	BRG16	BRGH		
0	0	0	8-bit/Asynchronous	FOSC/[64 (n + 1)]
0	0	1	8-bit/Asynchronous	FOSC/[16 (n + 1)]
0	1	0	16-bit/Asynchronous	
0	1	1	16-bit/Asynchronous	FOSC/[4 (n + 1)]
1	0	x	8-bit/Synchronous	
1	1	x	16-bit/Synchronous	

Legend: x = Don't care, n = Value of SPBRGH:SPBRG register pair

EXAMPLE 18-1: CALCULATING BAUD RATE ERROR

For a device with FOSC of 16 MHz, desired baud rate of 9600, Asynchronous mode, 8-bit BRG:

Desired Baud Rate = FOSC/(64 ([SPBRGH:SPBRG] + 1))

Solving for SPBRGH:SPBRG:

X = ((FOSC/Desired Baud Rate)/64) – 1
= ((16000000/9600)/64) – 1
= [25.042] = 25

Calculated Baud Rate= 16000000/(64 (25 + 1))
= 9615

Error = (Calculated Baud Rate – Desired Baud Rate)/Desired Baud Rate
= (9615 – 9600)/9600 = 0.16%

TABLE 18-2: REGISTERS ASSOCIATED WITH BAUD RATE GENERATOR

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other Resets
TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	0000 0010	0000 0010
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 000x	0000 000x
BAUDCON	—	RCIDL	—	SCKP	BRG16	—	WUE	ABDEN	-1-0 0-00	-1-0 0-00
SPBRGH	Baud Rate Generator Register, High Byte								0000 0000	0000 0000
SPBRG	Baud Rate Generator Register, Low Byte								0000 0000	0000 0000

Legend: x = unknown, - = unimplemented, read as '0'. Shaded cells are not used by the BRG.

TABLE 18-3: BAUD RATES FOR ASYNCHRONOUS MODES

BAUD RATE (K)	SYNC = 0, BRGH = 0, BRG16 = 0											
	FOSC = 40.000 MHz			FOSC = 20.000 MHz			FOSC = 10.000 MHz			FOSC = 8.000 MHz		
	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)
0.3	—	—	—	—	—	—	—	—	—	—	—	—
1.2	—	—	—	1.221	1.73	255	1.202	0.16	129	1201	-0.16	103
2.4	2.441	1.73	255	2.404	0.16	129	2.404	0.16	64	2403	-0.16	51
9.6	9.615	0.16	64	9.766	1.73	31	9.766	1.73	15	9615	-0.16	12
19.2	19.531	1.73	31	19.531	1.73	15	19.531	1.73	7	—	—	—
57.6	56.818	-1.36	10	62.500	8.51	4	52.083	-9.58	2	—	—	—
115.2	125.000	8.51	4	104.167	-9.58	2	78.125	-32.18	1	—	—	—

BAUD RATE (K)	SYNC = 0, BRGH = 0, BRG16 = 0								
	FOSC = 4.000 MHz			FOSC = 2.000 MHz			FOSC = 1.000 MHz		
	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)
0.3	0.300	0.16	207	300	-0.16	103	300	-0.16	51
1.2	1.202	0.16	51	1201	-0.16	25	1201	-0.16	12
2.4	2.404	0.16	25	2403	-0.16	12	—	—	—
9.6	8.929	-6.99	6	—	—	—	—	—	—
19.2	20.833	8.51	2	—	—	—	—	—	—
57.6	62.500	8.51	0	—	—	—	—	—	—
115.2	62.500	-45.75	0	—	—	—	—	—	—

BAUD RATE (K)	SYNC = 0, BRGH = 1, BRG16 = 0											
	FOSC = 40.000 MHz			FOSC = 20.000 MHz			FOSC = 10.000 MHz			FOSC = 8.000 MHz		
	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)
0.3	—	—	—	—	—	—	—	—	—	—	—	—
1.2	—	—	—	—	—	—	—	—	—	—	—	—
2.4	—	—	—	—	—	—	2.441	1.73	255	2403	-0.16	207
9.6	9.766	1.73	255	9.615	0.16	129	9.615	0.16	64	9615	-0.16	51
19.2	19.231	0.16	129	19.231	0.16	64	19.531	1.73	31	19230	-0.16	25
57.6	58.140	0.94	42	56.818	-1.36	21	56.818	-1.36	10	55555	3.55	8
115.2	113.636	-1.36	21	113.636	-1.36	10	125.000	8.51	4	—	—	—

BAUD RATE (K)	SYNC = 0, BRGH = 1, BRG16 = 0								
	FOSC = 4.000 MHz			FOSC = 2.000 MHz			FOSC = 1.000 MHz		
	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)
0.3	—	—	—	—	—	—	300	-0.16	207
1.2	1.202	0.16	207	1201	-0.16	103	1201	-0.16	51
2.4	2.404	0.16	103	2403	-0.16	51	2403	-0.16	25
9.6	9.615	0.16	25	9615	-0.16	12	—	—	—
19.2	19.231	0.16	12	—	—	—	—	—	—
57.6	62.500	8.51	3	—	—	—	—	—	—
115.2	125.000	8.51	1	—	—	—	—	—	—

TABLE 18-3: BAUD RATES FOR ASYNCHRONOUS MODES (CONTINUED)

BAUD RATE (K)	SYNC = 0, BRGH = 0, BRG16 = 1											
	FOSC = 40.000 MHz			FOSC = 20.000 MHz			FOSC = 10.000 MHz			FOSC = 8.000 MHz		
	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)
0.3	0.300	0.00	8332	0.300	0.02	4165	0.300	0.02	2082	300	-0.04	1665
1.2	1.200	0.02	2082	1.200	-0.03	1041	1.200	-0.03	520	1201	-0.16	415
2.4	2.402	0.06	1040	2.399	-0.03	520	2.404	0.16	259	2403	-0.16	207
9.6	9.615	0.16	259	9.615	0.16	129	9.615	0.16	64	9615	-0.16	51
19.2	19.231	0.16	129	19.231	0.16	64	19.531	1.73	31	19230	-0.16	25
57.6	58.140	0.94	42	56.818	-1.36	21	56.818	-1.36	10	55555	3.55	8
115.2	113.636	-1.36	21	113.636	-1.36	10	125.000	8.51	4	—	—	—

BAUD RATE (K)	SYNC = 0, BRGH = 0, BRG16 = 1								
	FOSC = 4.000 MHz			FOSC = 2.000 MHz			FOSC = 1.000 MHz		
	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)
0.3	0.300	0.04	832	300	-0.16	415	300	-0.16	207
1.2	1.202	0.16	207	1201	-0.16	103	1201	-0.16	51
2.4	2.404	0.16	103	2403	-0.16	51	2403	-0.16	25
9.6	9.615	0.16	25	9615	-0.16	12	—	—	—
19.2	19.231	0.16	12	—	—	—	—	—	—
57.6	62.500	8.51	3	—	—	—	—	—	—
115.2	125.000	8.51	1	—	—	—	—	—	—

BAUD RATE (K)	SYNC = 0, BRGH = 1, BRG16 = 1 or SYNC = 1, BRG16 = 1											
	FOSC = 40.000 MHz			FOSC = 20.000 MHz			FOSC = 10.000 MHz			FOSC = 8.000 MHz		
	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)
0.3	0.300	0.00	33332	0.300	0.00	16665	0.300	0.00	8332	300	-0.01	6665
1.2	1.200	0.00	8332	1.200	0.02	4165	1.200	0.02	2082	1200	-0.04	1665
2.4	2.400	0.02	4165	2.400	0.02	2082	2.402	0.06	1040	2400	-0.04	832
9.6	9.606	0.06	1040	9.596	-0.03	520	9.615	0.16	259	9615	-0.16	207
19.2	19.193	-0.03	520	19.231	0.16	259	19.231	0.16	129	19230	-0.16	103
57.6	57.803	0.35	172	57.471	-0.22	86	58.140	0.94	42	57142	0.79	34
115.2	114.943	-0.22	86	116.279	0.94	42	113.636	-1.36	21	117647	-2.12	16

BAUD RATE (K)	SYNC = 0, BRGH = 1, BRG16 = 1 or SYNC = 1, BRG16 = 1								
	FOSC = 4.000 MHz			FOSC = 2.000 MHz			FOSC = 1.000 MHz		
	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)
0.3	0.300	0.01	3332	300	-0.04	1665	300	-0.04	832
1.2	1.200	0.04	832	1201	-0.16	415	1201	-0.16	207
2.4	2.404	0.16	415	2403	-0.16	207	2403	-0.16	103
9.6	9.615	0.16	103	9615	-0.16	51	9615	-0.16	25
19.2	19.231	0.16	51	19230	-0.16	25	19230	-0.16	12
57.6	58.824	2.12	16	55555	3.55	8	—	—	—
115.2	111.111	-3.55	8	—	—	—	—	—	—

18.1.2 AUTO-BAUD RATE DETECT

The enhanced USART module supports the automatic detection and calibration of baud rate. This feature is active only in Asynchronous mode and while the WUE bit is clear.

The automatic baud rate measurement sequence (Figure 18-1) begins whenever a Start bit is received and the ABDEN bit is set. The calculation is self-averaging.

In the Auto-Baud Rate 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. In ABD mode, the internal Baud Rate Generator is used as a counter to time the bit period of the incoming serial byte stream.

Once the ABDEN bit is set, the state machine will clear the BRG and look for a Start bit. The auto-baud detect must receive a byte with the value 55h (ASCII "U", which is also the LIN bus sync character) in order to calculate the proper bit rate. The measurement is taken over both a low and a high bit time in order to minimize any effects caused by asymmetry of the incoming signal. After a Start bit, the SPBRG begins counting up using the preselected clock source on the first rising edge of RX. After eight bits on the RX pin or the fifth rising edge, an accumulated value totalling the proper BRG period is left in the SPBRGH:SPBRG registers. Once the 5th edge is seen (should correspond to the Stop bit), the ABDEN bit is automatically cleared.

While calibrating the baud rate period, the BRG registers are clocked at 1/8th the preconfigured clock rate. Note that the BRG clock will be configured by the BRG16 and BRGH bits. Independent of the BRG16 bit setting, both the SPBRG and SPBRGH will be used as a 16-bit counter. This allows the user to verify that no carry occurred for 8-bit modes by checking for 00h in the SPBRGH register. Refer to Table 18-4 for counter clock rates to the BRG.

While the ABD sequence takes place, the USART state machine is held in Idle. The RCIF interrupt is set once the fifth rising edge on RX is detected. The value in the RCREG needs to be read to clear the RCIF interrupt. RCREG content should be discarded.

Note 1: If the WUE bit is set with the ABDEN bit, auto-baud rate detection will occur on the byte *following* the break character.

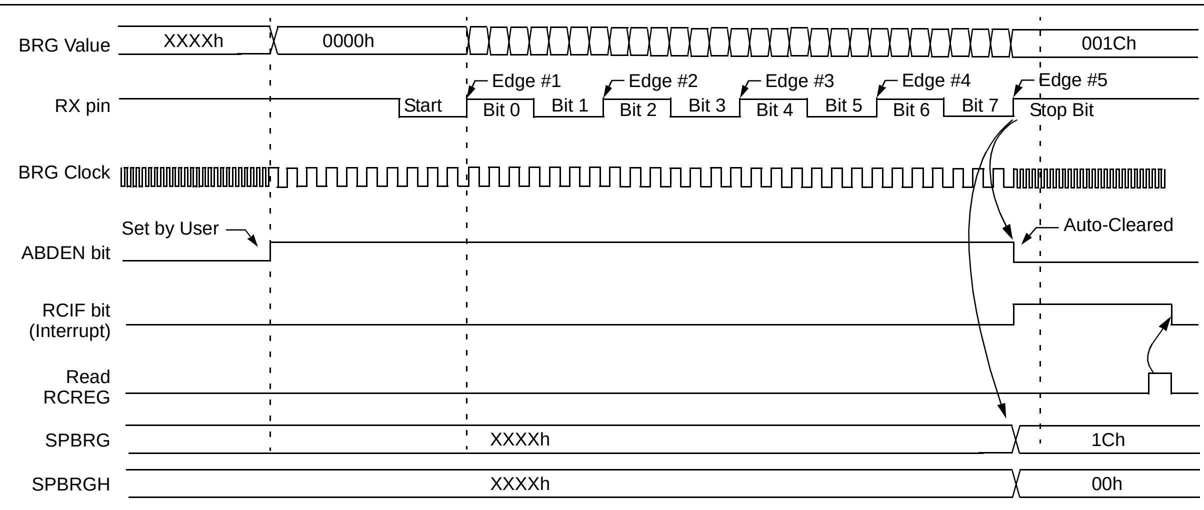
2: It is up to the user to determine that the incoming character baud rate is within the range of the selected BRG clock source. Some combinations of oscillator frequency and USART baud rates are not possible due to bit error rates. Overall system timing and communication baud rates must be taken into consideration when using the auto-baud rate detection feature.

TABLE 18-4: BRG COUNTER CLOCK RATES

BRG16	BRGH	BRG Counter Clock
0	0	FOSC/512
0	1	FOSC/128
1	0	FOSC/128
1	1	FOSC/32

Note: During the ABD sequence, SPBRG and SPBRGH are both used as a 16-bit counter independent of BRG16 setting.

FIGURE 18-1: AUTOMATIC BAUD RATE CALCULATION



Note 1: The ABD sequence requires the USART module to be configured in Asynchronous mode and WUE = 0.

18.2 USART Asynchronous Mode

The Asynchronous mode of operation is selected by clearing the SYNC bit (TXSTA<4>). In this mode, the USART uses standard Non-Return-to-Zero (NRZ) format (one Start bit, eight or nine data bits and one Stop bit). The most common data format is 8 bits. An on-chip dedicated 8-bit/16-bit Baud Rate Generator can be used to derive standard baud rate frequencies from the oscillator.

The USART transmits and receives the LSb first. The USART's transmitter and receiver are functionally independent but use the same data format and baud rate. The Baud Rate Generator produces a clock, either x16 or x64 of the bit shift rate depending on the BRGH and BRG16 bits (TXSTA<2> and BAUDCON<3>). Parity is not supported by the hardware but can be implemented in software and stored as the 9th data bit.

Asynchronous mode is available in all low-power modes; it is available in Sleep mode only when auto-wake-up on sync break is enabled. When in PRI_IDLE mode, no changes to the Baud Rate Generator values are required; however, other low-power mode clocks may operate at another frequency than the primary clock. Therefore, the Baud Rate Generator values may need to be adjusted.

When operating in Asynchronous mode, the USART module consists of the following important elements:

- Baud Rate Generator
- Sampling Circuit
- Asynchronous Transmitter
- Asynchronous Receiver
- Auto-Wake-up on Sync Break Character
- 12-bit Break Character Transmit
- Auto-Baud Rate Detection

18.2.1 USART ASYNCHRONOUS TRANSMITTER

The USART transmitter block diagram is shown in Figure 18-2. The heart of the transmitter is the Transmit (Serial) Shift register (TSR). The Shift register obtains its data from the read/write transmit buffer, TXREG. The TXREG register is loaded with data in software. The TSR register is not loaded until the Stop bit has been transmitted from the previous load. As soon as the Stop bit is transmitted, the TSR is loaded with new data from the TXREG register (if available).

Once the TXREG register transfers the data to the TSR register (occurs in one TCY), the TXREG register is empty and flag bit TXIF (PIR1<4>) is set. This interrupt can be enabled/disabled by setting/clearing enable bit TXIE (PIE1<4>). Flag bit TXIF will be set regardless of the state of enable bit TXIE and cannot be cleared in software. Flag bit TXIF is not cleared immediately upon loading the Transmit Buffer register, TXREG. TXIF becomes valid in the second instruction cycle following the load instruction. Polling TXIF immediately following a load of TXREG will return invalid results.

While flag bit TXIF indicates the status of the TXREG register, another bit, TRMT (TXSTA<1>), shows the status of the TSR register. Status bit TRMT is a read-only bit which is set when the TSR register is empty. No interrupt logic is tied to this bit, so the user has to poll this bit in order to determine if the TSR register is empty.

> **Note 1:** The TSR register is not mapped in data memory so it is not available to the user.
>
> **2:** Flag bit TXIF is set when enable bit TXEN is set.

To set up an Asynchronous Transmission:

1. Initialize the SPBRGH:SPBRG registers for the appropriate baud rate. Set or clear the BRGH and BRG16 bits, as required, to achieve the desired baud rate.

> **Note:** When BRGH and BRG16 bits are set, SPBRGH:SPBRG must be more than '1'.

2. Enable the asynchronous serial port by clearing bit SYNC and setting bit SPEN.
3. If interrupts are desired, set enable bit TXIE.
4. If 9-bit transmission is desired, set transmit bit TX9. Can be used as address/data bit.
5. Enable the transmission by setting bit TXEN which will also set bit TXIF.
6. If 9-bit transmission is selected, the ninth bit should be loaded in bit TX9D.
7. Load data to the TXREG register (starts transmission).

If using interrupts, ensure that the GIE and PEIE bits in the INTCON register (INTCON<7:6>) are set.

FIGURE 18-2: USART TRANSMIT BLOCK DIAGRAM

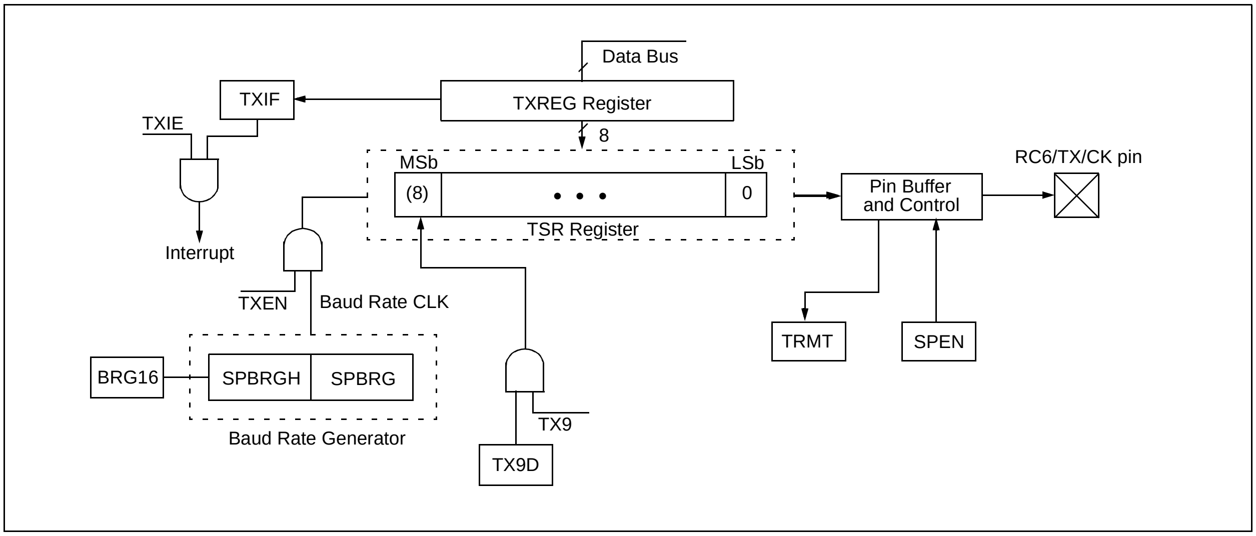


FIGURE 18-3: ASYNCHRONOUS TRANSMISSION

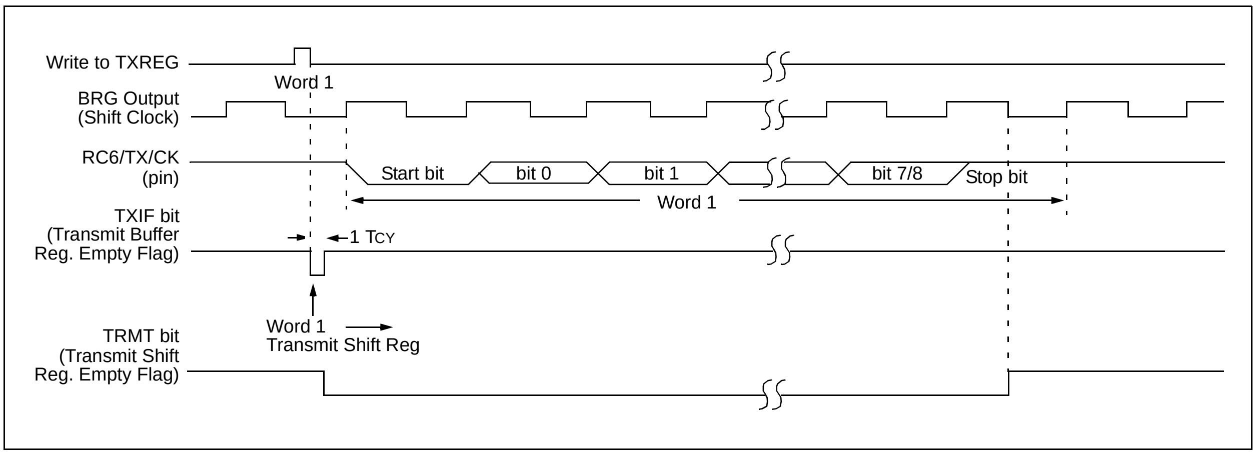
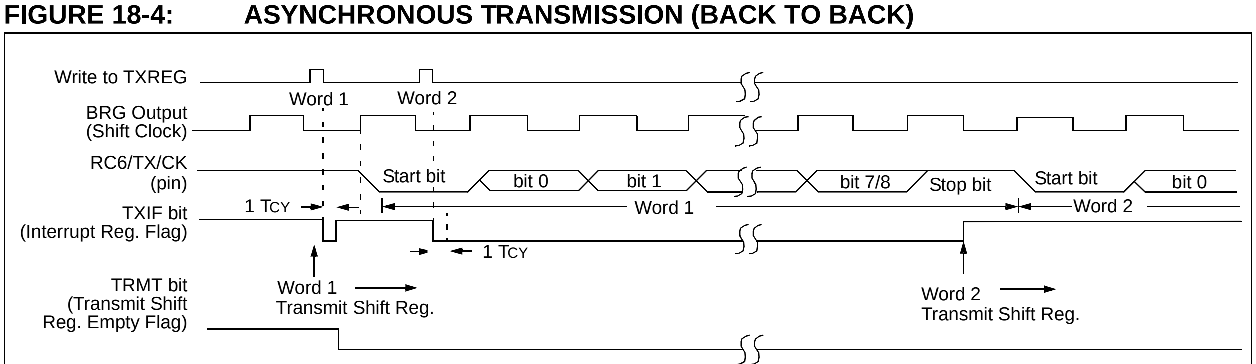


FIGURE 18-4: ASYNCHRONOUS TRANSMISSION (BACK TO BACK)



Note: This timing diagram shows two consecutive transmissions.

TABLE 18-5: REGISTERS ASSOCIATED WITH ASYNCHRONOUS TRANSMISSION

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other Resets
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
PIR1	PSPIF	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
PIE1	PSPIE	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
IPR1	PSPIP	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	1111 1111	1111 1111
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 000x	0000 000x
TXREG	USART Transmit Register								0000 0000	0000 0000
TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	0000 0010	0000 0010
BAUDCON	—	RCIDL	—	SCKP	BRG16	—	WUE	ABDEN	-1-1 0-00	-1-1 0-00
SPBRGH	Baud Rate Generator Register, High Byte								0000 0000	0000 0000
SPBRG	Baud Rate Generator Register, Low Byte								0000 0000	0000 0000

Legend: x = unknown, – = unimplemented locations read as '0'. Shaded cells are not used for asynchronous transmission.

18.2.2 USART ASYNCHRONOUS RECEIVER

The receiver block diagram is shown in Figure 18-5. The data is received on the RC7/RX/DT pin and drives the data recovery block. The data recovery block is actually a high-speed shifter operating at x16 times the baud rate, whereas the main receive serial shifter operates at the bit rate or at FOSC. This mode would typically be used in RS-232 systems.

To set up an asynchronous reception:

1. Initialize the SPBRGH:SPBRG registers for the appropriate baud rate. Set or clear the BRGH and BRG16 bits, as required, to achieve the desired baud rate.
2. Enable the asynchronous serial port by clearing bit SYNC and setting bit SPEN.
3. If interrupts are desired, set enable bit RCIE.
4. If 9-bit reception is desired, set bit RX9.
5. Enable the reception by setting bit CREN.
6. Flag bit RCIF will be set when reception is complete and an interrupt will be generated if enable bit RCIE was set.
7. Read the RCSTA register to get the 9th bit (if enabled) and determine if any error occurred during reception.
8. Read the 8-bit received data by reading the RCREG register.
9. If any error occurred, clear the error by clearing enable bit CREN.
10. If using interrupts, ensure that the GIE and PEIE bits in the INTCON register (INTCON<7:6>) are set.

18.2.3 SETTING UP 9-BIT MODE WITH ADDRESS DETECT

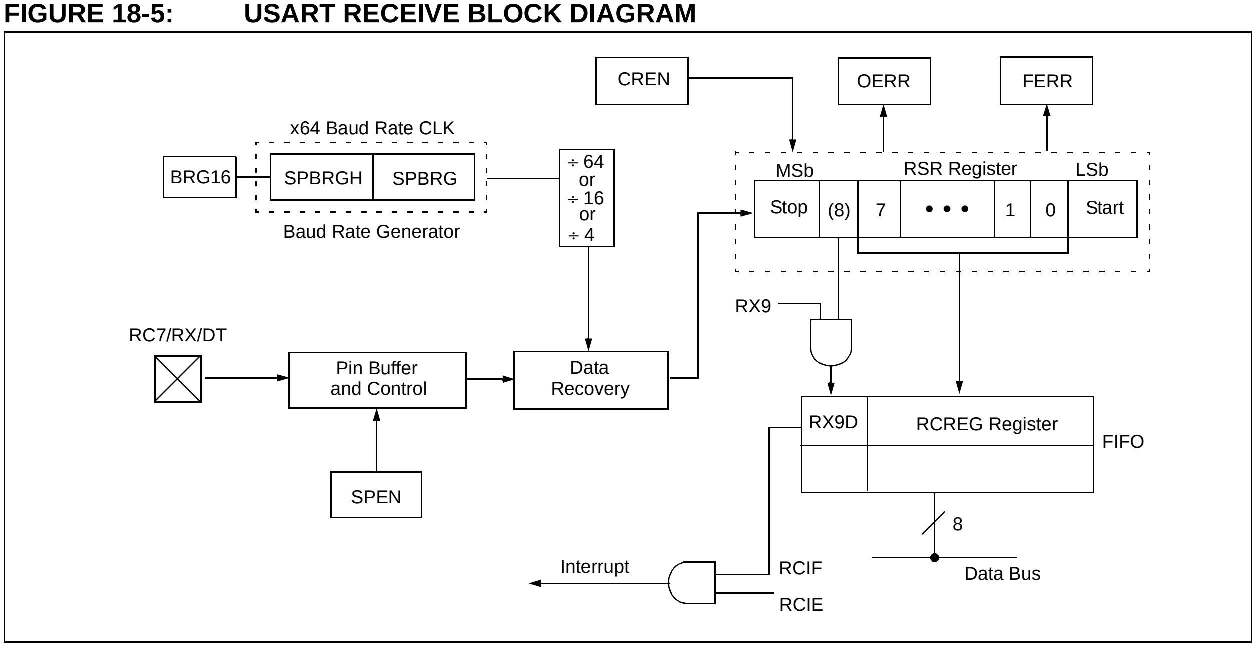
This mode would typically be used in RS-485 systems. To set up an asynchronous reception with address detect enable:

1. Initialize the SPBRGH:SPBRG registers for the appropriate baud rate. Set or clear the BRGH and BRG16 bits, as required, to achieve the desired baud rate..

Note:	When BRGH and BRG16 bits are set, SPBRGH:SPBRG must be more than '1'.

2. Enable the asynchronous serial port by clearing the SYNC bit and setting the SPEN bit.
3. If interrupts are required, set the RCEN bit and select the desired priority level with the RCIP bit.
4. Set the RX9 bit to enable 9-bit reception.
5. Set the ADDEN bit to enable address detect.
6. Enable reception by setting the CREN bit.
7. The RCIF bit will be set when reception is complete. The interrupt will be Acknowledged if the RCIE and GIE bits are set.
8. Read the RCSTA register to determine if any error occurred during reception, as well as read bit 9 of data (if applicable).
9. Read RCREG to determine if the device is being addressed.
10. If any error occurred, clear the CREN bit.
11. If the device has been addressed, clear the ADDEN bit to allow all received data into the receive buffer and interrupt the CPU.

FIGURE 18-5: USART RECEIVE BLOCK DIAGRAM

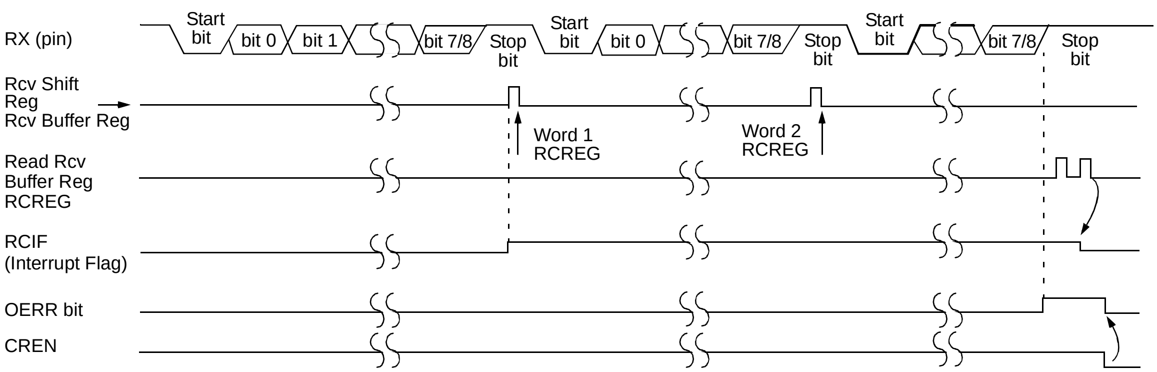


To set up an asynchronous transmission:

1. Initialize the SPBRG register for the appropriate baud rate. If a high-speed baud rate is desired, set bit BRGH (see **Section 18.1 "USART Baud Rate Generator (BRG)"**).
2. Enable the asynchronous serial port by clearing bit SYNC and setting bit SPEN.
3. If interrupts are desired, set enable bit TXIE.
4. If 9-bit transmission is desired, set transmit bit TX9. Can be used as address/data bit.
5. Enable the transmission by setting bit TXEN which will also set bit TXIF.
6. If 9-bit transmission is selected, the ninth bit should be loaded in bit TX9D.
7. Load data to the TXREG register (starts transmission).

If using interrupts, ensure that the GIE and PEIE bits in the INTCON register (INTCON<7:6>) are set.

FIGURE 18-6: ASYNCHRONOUS RECEPTION



Note: This timing diagram shows three words appearing on the RX input. The RCREG (receive buffer) is read after the third word causing the OERR (overrun) bit to be set.

TABLE 18-6: REGISTERS ASSOCIATED WITH ASYNCHRONOUS RECEPTION

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other Resets
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
PIR1	PSPIF	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
PIE1	PSPIE	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
IPR1	PSPIP	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	1111 1111	1111 1111
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 000x	0000 000x
RCREG	USART Receive Register								0000 0000	0000 0000
TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	0000 0010	0000 0010
BAUDCON	—	RCIDL	—	SCKP	BRG16	—	WUE	ABDEN	-1-1 0-00	-1-1 0-00
SPBRGH	Baud Rate Generator Register, High Byte								0000 0000	0000 0000
SPBRG	Baud Rate Generator Register, Low Byte								0000 0000	0000 0000

Legend: x = unknown, – = unimplemented locations read as '0'. Shaded cells are not used for asynchronous reception.

18.2.4 AUTO-WAKE-UP ON SYNC BREAK CHARACTER

During Sleep mode, all clocks to the USART are suspended. Because of this, the Baud Rate Generator is inactive and a proper byte reception cannot be performed. The auto-wake-up feature allows the controller to wake-up due to activity on the RX/DT line while the USART is operating in Asynchronous mode.

The auto-wake-up feature is enabled by setting the WUE bit (BAUDCON<1>). Once set, the typical receive sequence on RX/DT is disabled and the USART 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.)

Following a wake-up event, the module generates an RCIF interrupt. The interrupt is generated synchronously to the Q clocks in normal operating modes (Figure 18-7) and asynchronously, if the device is in Sleep mode (Figure 18-8). The interrupt condition is cleared by reading the RCREG register.

The WUE bit is automatically cleared once a low-to-high transition is observed on the RX line following the wake-up event. At this point, the USART module is in Idle mode and returns to normal operation. This signals to the user that the sync break event is over.

18.2.4.1 Special Considerations Using Auto-Wake-up

Since auto-wake-up functions by sensing rising edge transitions on RX/DT, information with any state changes before the Stop bit may signal a false end-of-character and cause data or framing errors. To work properly, therefore, the initial character in the transmission must be all '0's. This can be 00h (8 bytes) for standard RS-232 devices or 000h (12 bits) for LIN bus.

Oscillator start-up time must also be considered, especially in applications using oscillators with longer start-up intervals (i.e., XT or HS 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 USART.

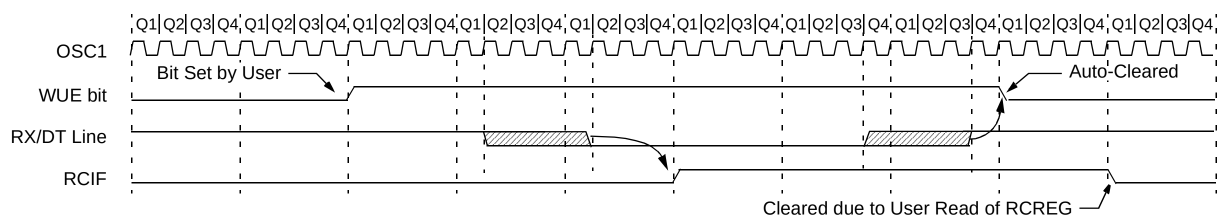
18.2.4.2 Special Considerations Using the WUE Bit

The timing of WUE and RCIF events may cause some confusion when it comes to determining the validity of received data. As noted, setting the WUE bit places the USART in an Idle mode. The wake-up event causes a receive interrupt by setting the RCIF bit. The WUE bit is cleared after this when a rising edge is seen on RX/DT. The interrupt condition is then cleared by reading the RCREG register. Ordinarily, the data in RCREG will be dummy data and should be discarded.

The fact that the WUE bit has been cleared (or is still set) and the RCIF flag is set should not be used as an indicator of the integrity of the data in RCREG. Users should consider implementing a parallel method in firmware to verify received data integrity.

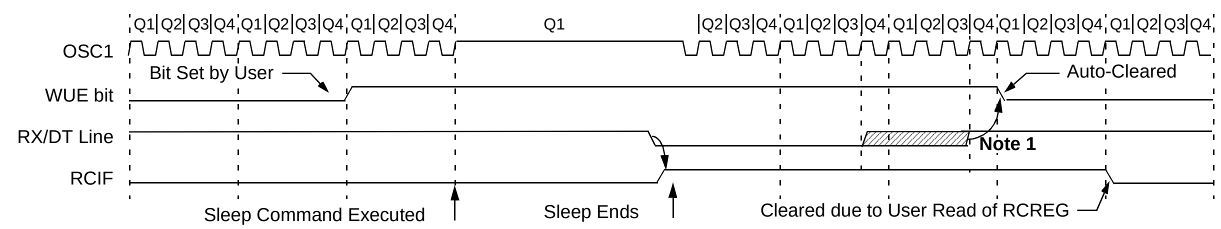
To assure that no actual data is lost, check the RCIDL bit to verify that a receive operation is not in process. If a receive operation is not occurring, the WUE bit may then be set just prior to entering the Sleep mode.

FIGURE 18-7: AUTO-WAKE-UP BIT (WUE) TIMINGS DURING NORMAL OPERATION



Note: The USART remains in Idle while the WUE bit is set.

FIGURE 18-8: AUTO-WAKE-UP BIT (WUE) TIMINGS DURING SLEEP



Note 1: If the wake-up event requires long oscillator warm-up time, the auto-clear of the WUE bit can occur while the *stposc* signal is still active. This sequence should not depend on the presence of Q clocks.

2: The USART remains in Idle while the WUE bit is set.

18.2.5 BREAK CHARACTER SEQUENCE

The enhanced USART module has the capability of sending the special break character sequences that are required by the LIN bus standard. The break character transmit consists of a Start bit, followed by twelve '0' bits and a Stop bit. The frame break character is sent whenever the SENDB and TXEN bits (TXSTA<3> and TXSTA<5>) are set while the Transmit Shift register is loaded with data. Note that the value of data written to TXREG 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).

Note that the data value written to the TXREG for the break character is ignored. The write simply serves the purpose of initiating the proper sequence.

The TRMT bit indicates when the transmit operation is active or Idle, just as it does during normal transmission. See Figure 18-9 for the timing of the break character sequence.

18.2.5.1 Break and Sync Transmit Sequence

The following sequence will send 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 USART for the desired mode.
2. Set the TXEN and SENDB bits to set up the break character.
3. Load the TXREG with a dummy character to initiate transmission (the value is ignored).
4. Write '55h' to TXREG to load the sync character into the transmit FIFO buffer.
5. After the break has been sent, the SENDB bit is reset by hardware. The sync character now transmits in the preconfigured mode.

When the TXREG becomes empty, as indicated by the TXIF, the next data byte can be written to TXREG.

18.2.6 RECEIVING A BREAK CHARACTER

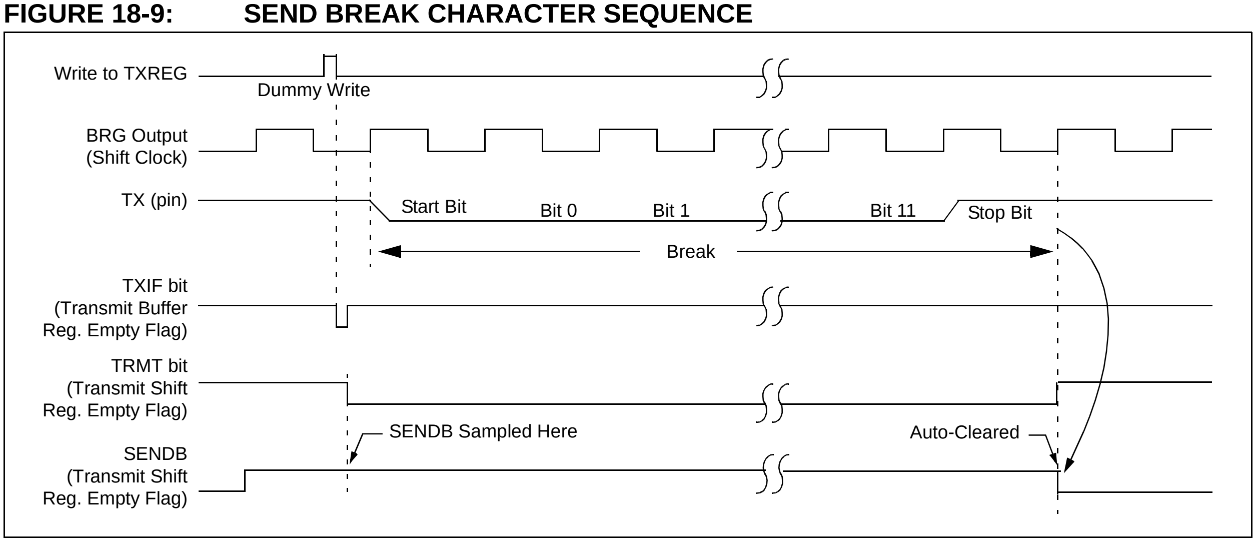
The enhanced USART module can receive a break character in two ways.

The first method forces the configuration of the baud rate at a frequency of 9/13 the typical speed. This allows for the Stop bit transition to be at the correct sampling location (13 bits for break versus Start bit and 8 data bits for typical data).

The second method uses the auto-wake-up feature described in **Section 18.2.4 "Auto-Wake-up on Sync Break Character"**. By enabling this feature, the USART will sample the next two transitions on RX/DT, cause an RCIF 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 rate detect feature. For both methods, the user can set the ABD bit once the TXIF interrupt is observed.

FIGURE 18-9: SEND BREAK CHARACTER SEQUENCE



18.3 USART Synchronous Master Mode

The Synchronous Master mode is entered by setting the CSRC bit (TXSTA<7>). In this mode, the data is transmitted in a half-duplex manner (i.e., transmission and reception do not occur at the same time). When transmitting data, the reception is inhibited and vice versa. Synchronous mode is entered by setting bit SYNC (TXSTA<4>). In addition, enable bit, SPEN (RCSTA<7>), is set in order to configure the RC6/TX/CK and RC7/RX/DT I/O pins to CK (clock) and DT (data) lines, respectively.

The Master mode indicates that the processor transmits the master clock on the CK line. Clock polarity is selected with the SCKP bit (BAUDCON<5>); setting SCKP sets the Idle state on CK as high, while clearing the bit sets the Idle state as low. This option is provided to support Microwire devices with this module.

18.3.1 USART SYNCHRONOUS MASTER TRANSMISSION

The USART transmitter block diagram is shown in Figure 18-2. The heart of the transmitter is the Transmit (Serial) Shift Register (TSR). The Shift register obtains its data from the Read/Write Transmit Buffer register, TXREG. The TXREG register is loaded with data in software. The TSR register is not loaded until the last bit has been transmitted from the previous load. As soon as the last bit is transmitted, the TSR is loaded with new data from the TXREG (if available).

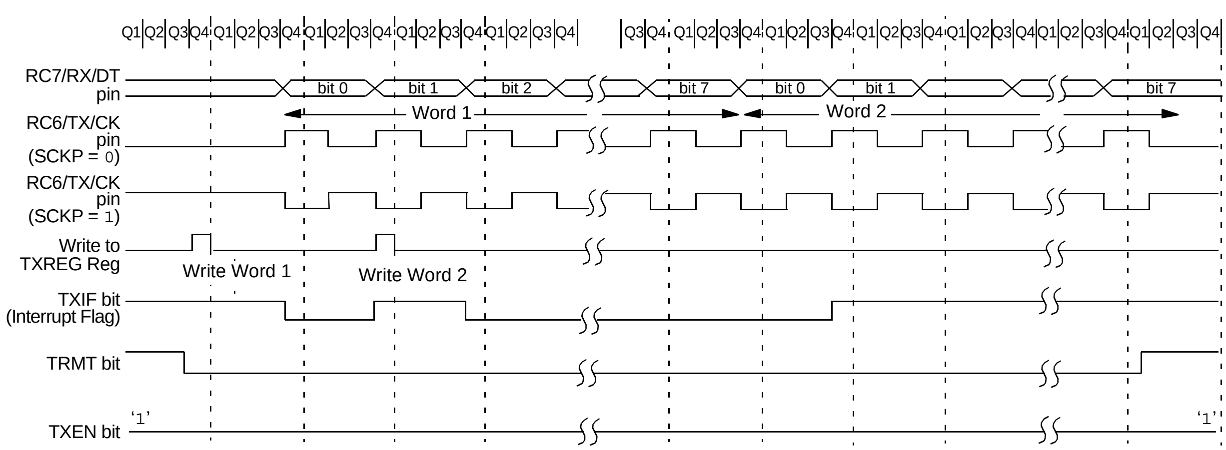
Once the TXREG register transfers the data to the TSR register (occurs in one TCYCLE), the TXREG is empty and interrupt bit TXIF (PIR1<4>) is set. The interrupt can be enabled/disabled by setting/clearing enable bit, TXIE (PIE1<4>). Flag bit TXIF will be set regardless of the state of enable bit TXIE and cannot be cleared in software. It will reset only when new data is loaded into the TXREG register.

While flag bit TXIF indicates the status of the TXREG register, another bit, TRMT (TXSTA<1>), shows the status of the TSR register. TRMT is a read-only bit which is set when the TSR is empty. No interrupt logic is tied to this bit so the user must poll this bit in order to determine if the TSR register is empty. The TSR is not mapped in data memory so it is not available to the user.

To set up a synchronous master transmission:

1. Initialize the SPBRGH:SPBRG registers for the appropriate baud rate. Set or clear the BRGH and BRG16 bits, as required, to achieve the desired baud rate.
2. Enable the synchronous master serial port by setting bits SYNC, SPEN and CSRC.
3. If interrupts are desired, set enable bit TXIE.
4. If 9-bit transmission is desired, set bit TX9.
5. Enable the transmission by setting bit TXEN.
6. If 9-bit transmission is selected, the ninth bit should be loaded in bit TX9D.
7. Start transmission by loading data to the TXREG register.
8. If using interrupts, ensure that the GIE and PEIE bits in the INTCON register (INTCON<7:6>) are set.

FIGURE 18-10: SYNCHRONOUS TRANSMISSION



Note: Sync Master mode, SPBRG = 0, continuous transmission of two 8-bit words.

FIGURE 18-11: SYNCHRONOUS TRANSMISSION (THROUGH TXEN)

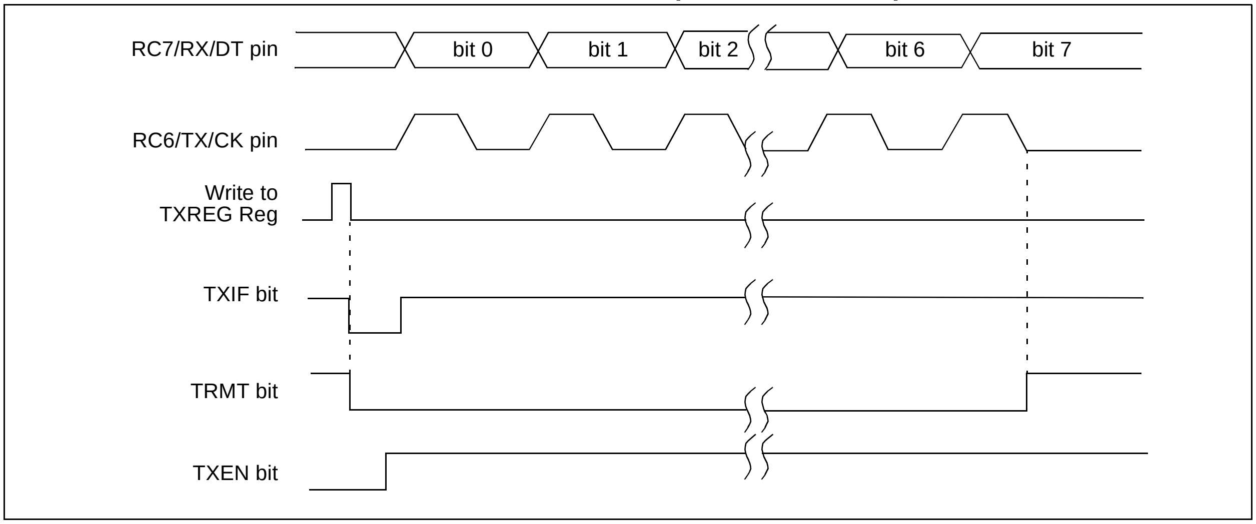


TABLE 18-7: REGISTERS ASSOCIATED WITH SYNCHRONOUS MASTER TRANSMISSION

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other Resets
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 0000	0000 0000
PIR1	PSPIF	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
PIE1	PSPIE	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
IPR1	PSPIP	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	1111 1111	1111 1111
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 000x	0000 000x
TXREG	USART Transmit Register								0000 0000	0000 0000
TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	0000 0010	0000 0010
BAUDCON	—	RCIDL	—	SCKP	BRG16	—	WUE	ABDEN	-1-0 0-00	-1-0 0-00
SPBRGH	Baud Rate Generator Register, High Byte								0000 0000	0000 0000
SPBRG	Baud Rate Generator Register, Low Byte								0000 0000	0000 0000

Legend: x = unknown, - = unimplemented, read as '0'. Shaded cells are not used for synchronous master transmission.

18.3.2 USART SYNCHRONOUS MASTER RECEPTION

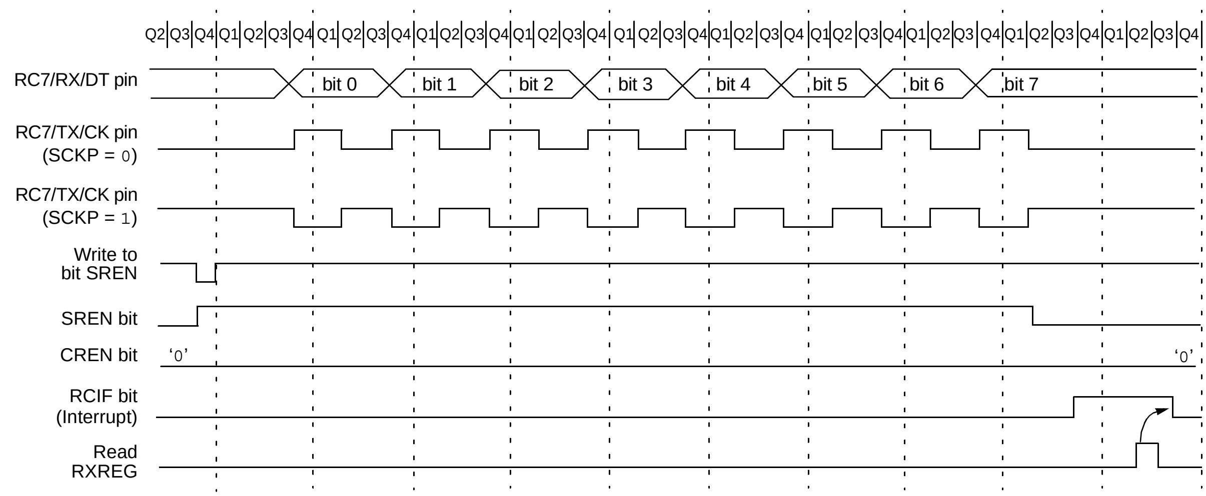
Once Synchronous mode is selected, reception is enabled by setting either the Single Receive Enable bit, SREN (RCSTA<5>), or the Continuous Receive Enable bit, CREN (RCSTA<4>). Data is sampled on the RC7/RX/DT pin on the falling edge of the clock.

If enable bit SREN is set, only a single word is received. If enable bit CREN is set, the reception is continuous until CREN is cleared. If both bits are set, then CREN takes precedence.

To set up a synchronous master reception:

1. Initialize the SPBRGH:SPBRG registers for the appropriate baud rate. Set or clear the BRGH and BRG16 bits, as required, to achieve the desired baud rate.
2. Enable the synchronous master serial port by setting bits SYNC, SPEN and CSRC.
3. Ensure bits CREN and SREN are clear.
4. If interrupts are desired, set enable bit RCIE.
5. If 9-bit reception is desired, set bit RX9.
6. If a single reception is required, set bit SREN. For continuous reception, set bit CREN.
7. Interrupt flag bit RCIF will be set when reception is complete and an interrupt will be generated if the enable bit RCIE was set.
8. Read the RCSTA register to get the 9th bit (if enabled) and determine if any error occurred during reception.
9. Read the 8-bit received data by reading the RCREG register.
10. If any error occurred, clear the error by clearing bit CREN.
11. If using interrupts, ensure that the GIE and PEIE bits in the INTCON register (INTCON<7:6>) are set.

FIGURE 18-12: SYNCHRONOUS RECEPTION (MASTER MODE, SREN)



Note: Timing diagram demonstrates Sync Master mode with bit SREN = 1 and bit BRGH = 0.

TABLE 18-8: REGISTERS ASSOCIATED WITH SYNCHRONOUS MASTER RECEPTION

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other Resets
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 0000	0000 0000
PIR1	PSPIF	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
PIE1	PSPIE	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
IPR1	PSPIP	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	1111 1111	1111 1111
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 000x	0000 000x
RCREG	USART Receive Register								0000 0000	0000 0000
TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	0000 0010	0000 0010
BAUDCON	—	RCIDL	—	SCKP	BRG16	—	WUE	ABDEN	-1-0 0-00	-1-0 0-00
SPBRGH	Baud Rate Generator Register, High Byte								0000 0000	0000 0000
SPBRG	Baud Rate Generator Register, Low Byte								0000 0000	0000 0000

Legend: x = unknown, - = unimplemented, read as '0'. Shaded cells are not used for synchronous master reception.

18.4 USART Synchronous Slave Mode

Synchronous Slave mode is entered by clearing bit CSRC (TXSTA<7>). This mode differs from the Synchronous Master mode in that the shift clock is supplied externally at the RC6/TX/CK pin (instead of being supplied internally in Master mode). This allows the device to transfer or receive data while in any low-power mode.

18.4.1 USART SYNCHRONOUS SLAVE TRANSMIT

The operation of the Synchronous Master and Slave modes are identical except in the case of the Sleep mode.

If two words are written to the TXREG and then the `SLEEP` instruction is executed, the following will occur:

a) The first word will immediately transfer to the TSR register and transmit.
b) The second word will remain in TXREG register.
c) Flag bit TXIF will not be set.
d) When the first word has been shifted out of TSR, the TXREG register will transfer the second word to the TSR and flag bit TXIF will now be set.
e) If enable bit TXIE is set, the interrupt will wake the chip from Sleep. If the global interrupt is enabled, the program will branch to the interrupt vector.

To set up a synchronous slave transmission:

1. Enable the synchronous slave serial port by setting bits SYNC and SPEN and clearing bit CSRC.
2. Clear bits CREN and SREN.
3. If interrupts are desired, set enable bit TXIE.
4. If 9-bit transmission is desired, set bit TX9.
5. Enable the transmission by setting enable bit TXEN.
6. If 9-bit transmission is selected, the ninth bit should be loaded in bit TX9D.
7. Start transmission by loading data to the TXREG register.
8. If using interrupts, ensure that the GIE and PEIE bits in the INTCON register (INTCON<7:6>) are set.

TABLE 18-9: REGISTERS ASSOCIATED WITH SYNCHRONOUS SLAVE TRANSMISSION

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other Resets
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
PIR1	PSPIF	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
PIE1	PSPIE	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
IPR1	PSPIP	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	1111 1111	1111 1111
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 000x	0000 000x
TXREG	USART Transmit Register								0000 0000	0000 0000
TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	0000 0010	0000 0010
BAUDCON	—	RCIDL	—	SCKP	BRG16	—	WUE	ABDEN	-1-1 0-00	-1-1 0-00
SPBRGH	Baud Rate Generator Register, High Byte								0000 0000	0000 0000
SPBRG	Baud Rate Generator Register, Low Byte								0000 0000	0000 0000

Legend: x = unknown, - = unimplemented, read as '0'. Shaded cells are not used for synchronous slave transmission.

18.4.2 USART SYNCHRONOUS SLAVE RECEPTION

The operation of the Synchronous Master and Slave modes is identical, except in the case of Sleep or any Idle mode and bit SREN, which is a "don't care" in Slave mode.

If receive is enabled by setting the CREN bit prior to entering Sleep or any Idle mode, then a word may be received while in this low-power mode. Once the word is received, the RSR register will transfer the data to the RCREG register; if the RCIE enable bit is set, the interrupt generated will wake the chip from low-power mode. If the global interrupt is enabled, the program will branch to the interrupt vector.

To set up a synchronous slave reception:

1. Enable the synchronous master serial port by setting bits SYNC and SPEN and clearing bit CSRC.
2. If interrupts are desired, set enable bit RCIE.
3. If 9-bit reception is desired, set bit RX9.
4. To enable reception, set enable bit CREN.
5. Flag bit RCIF will be set when reception is complete. An interrupt will be generated if enable bit RCIE was set.
6. Read the RCSTA register to get the 9th bit (if enabled) and determine if any error occurred during reception.
7. Read the 8-bit received data by reading the RCREG register.
8. If any error occurred, clear the error by clearing bit CREN.
9. If using interrupts, ensure that the GIE and PEIE bits in the INTCON register (INTCON<7:6>) are set.

TABLE 18-10: REGISTERS ASSOCIATED WITH SYNCHRONOUS SLAVE RECEPTION

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other Resets
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 0000	0000 0000
PIR1	PSPIF	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
PIE1	PSPIE	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
IPR1	PSPIP	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	1111 1111	1111 1111
RCSTA	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	0000 000x	0000 000x
RCREG	USART Receive Register								0000 0000	0000 0000
TXSTA	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	0000 0010	0000 0010
BAUDCON	—	RCIDL	—	SCKP	BRG16	—	WUE	ABDEN	-1-0 0-00	-1-0 0-00
SPBRGH	Baud Rate Generator Register, High Byte								0000 0000	0000 0000
SPBRG	Baud Rate Generator Register, Low Byte								0000 0000	0000 0000

Legend: x = unknown, - = unimplemented, read as '0'. Shaded cells are not used for synchronous slave reception.

19.0 10-BIT ANALOG-TO-DIGITAL CONVERTER (A/D) MODULE

The Analog-to-Digital (A/D) converter module has 12 inputs for the PIC18F6X8X devices and 16 inputs for the PIC18F8X8X devices. This module allows conversion of an analog input signal to a corresponding 10-bit digital number.

A new feature for the A/D converter is the addition of programmable acquisition time. This feature allows the user to select a new channel for conversion and to set the GO/DONE bit immediately. When the GO/DONE bit is set, the selected channel is sampled for the programmed acquisition time before a conversion is actually started. This removes the firmware overhead that may have been required to allow for an acquisition (sampling) period (see Register 19-3 and **Section 19.4 "Selecting the A/D Conversion Clock"**).

The module has five registers:

- A/D Result High Register (ADRESH)
- A/D Result Low Register (ADRESL)
- A/D Control Register 0 (ADCON0)
- A/D Control Register 1 (ADCON1)
- A/D Control Register 2 (ADCON2)

The ADCON0 register, shown in Register 19-1, controls the operation of the A/D module. The ADCON1 register, shown in Register 19-2, configures the functions of the port pins. The ADCON2 register, shown in Register 19-3, configures the A/D clock source, programmed acquisition time and justification.

REGISTER 19-1: ADCON0 REGISTER

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	CHS3	CHS2	CHS1	CHS0	GO/DONE	ADON
bit 7							bit 0

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

bit 5-2 **CHS3:CHS0:** Analog Channel Select bits
0000 = Channel 0 (AN0)
0001 = Channel 1 (AN1)
0010 = Channel 2 (AN2)
0011 = Channel 3 (AN3)
0100 = Channel 4 (AN4)
0101 = Channel 5 (AN5)
0110 = Channel 6 (AN6)
0111 = Channel 7 (AN7)
1000 = Channel 8 (AN8)
1001 = Channel 9 (AN9)
1010 = Channel 10 (AN10)
1011 = Channel 11 (AN11)
1100 = Channel 12 (AN12)(1)
1101 = Channel 13 (AN13)(1)
1110 = Channel 14 (AN14)(1)
1111 = Channel 15 (AN15)(1)

bit 1 **GO/DONE:** A/D Conversion Status bit
When ADON = 1:
1 = A/D conversion in progress. This bit is automatically cleared when the A/D conversion is complete.
0 = A/D Idle

bit 0 **ADON:** A/D On bit
1 = A/D converter module is enabled
0 = A/D converter module is disabled and consumes no current

Note 1: These channels are only available on PIC18F8X8X devices.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

REGISTER 19-2: ADCON1 REGISTER

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	VCFG1	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0
bit 7							bit 0

bit 7-6 **Unimplemented:** Read as ‘0’

bit 5-4 **VCFG1:VCFG0:** Voltage Reference Configuration bits

	A/D VREF+	A/D VREF-
00	AVDD	AVSS
01	External VREF+	AVSS
10	AVDD	External VREF-
11	External VREF+	External VREF-

bit 3-0 **PCFG3:PCFG0:** A/D Port Configuration Control bits

	AN15	AN14	AN13	AN12	AN11	AN10	AN9	AN8	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0
0000	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A	A
0001	D	D	A	A	A	A	A	A	A	A	A	A	A	A	A	A
0010	D	D	D	A	A	A	A	A	A	A	A	A	A	A	A	A
0011	D	D	D	D	A	A	A	A	A	A	A	A	A	A	A	A
0100	D	D	D	D	D	A	A	A	A	A	A	A	A	A	A	A
0101	D	D	D	D	D	D	A	A	A	A	A	A	A	A	A	A
0110	D	D	D	D	D	D	D	A	A	A	A	A	A	A	A	A
0111	D	D	D	D	D	D	D	D	A	A	A	A	A	A	A	A
1000	D	D	D	D	D	D	D	D	D	A	A	A	A	A	A	A
1001	D	D	D	D	D	D	D	D	D	D	A	A	A	A	A	A
1010	D	D	D	D	D	D	D	D	D	D	D	A	A	A	A	A
1011	D	D	D	D	D	D	D	D	D	D	D	D	A	A	A	A
1100	D	D	D	D	D	D	D	D	D	D	D	D	D	A	A	A
1101	D	D	D	D	D	D	D	D	D	D	D	D	D	D	A	A
1110	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	A
1111	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D	D

A = Analog input D = Digital I/O
Shaded cells = Additional channels available on the PIC18F8X8X devices

Legend:

R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
- n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

Note: Channels AN15 through AN12 are not available on the 68-pin devices.

REGISTER 19-3: ADCON2 REGISTER

R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
ADFM	—	ACQT2	ACQT1	ACQT0	ADCS2	ADCS1	ADCS0
bit 7							bit 0

bit 7 **ADFM:** A/D Result Format Select bit
1 = Right justified
0 = Left justified

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

bit 5-3 **ACQT2:ACQT0:** A/D Acquisition Time Select bits
000 = 0 TAD(1)
001 = 2 TAD
010 = 4 TAD
011 = 6 TAD
100 = 8 TAD
101 = 12 TAD
110 = 16 TAD
111 = 20 TAD

bit 2-0 **ADCS2:ADCS0:** A/D Conversion Clock Select bits
000 = FOSC/2
001 = FOSC/8
010 = FOSC/32
011 = FRC (clock derived from A/D RC oscillator)(1)
100 = FOSC/4
101 = FOSC/16
110 = FOSC/64
111 = FRC (clock derived from A/D RC oscillator)(1)

Note 1: If the A/D FRC clock source is selected, a delay of one TCY (instruction cycle) is added before the A/D clock starts. This allows the SLEEP instruction to be executed before starting a conversion.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

The analog reference voltage is software selectable to either the device's positive and negative supply voltage (AVDD and AVSS) or the voltage level on the RA3/AN3/VREF+ and RA2/AN2/VREF- pins.

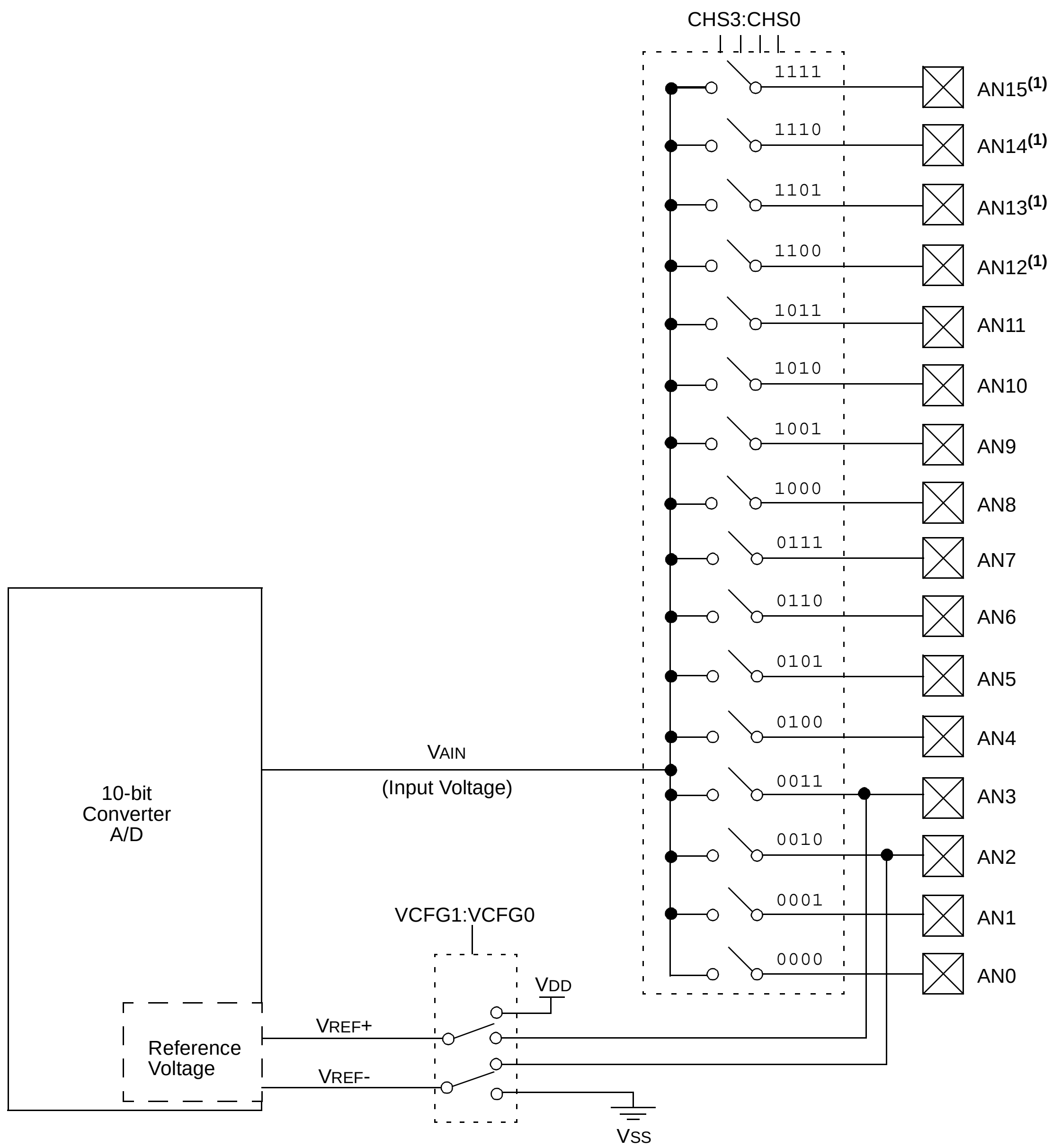
The A/D converter has a unique feature of being able to operate while the device is in Sleep mode. To operate in Sleep, the A/D conversion clock must be derived from the A/D's internal RC oscillator.

The output of the sample and hold is the input into the converter which generates the result via successive approximation.

A device Reset forces all registers to their Reset state. This forces the A/D module to be turned off and any conversion in progress is aborted.

Each port pin associated with the A/D converter can be configured as an analog input or as a digital I/O. The ADRESH and ADRESL registers contain the result of the A/D conversion. When the A/D conversion is complete, the result is loaded into the ADRESH/ADRESL registers, the GO/DONE bit (ADCON0 register) is cleared and A/D interrupt flag bit ADIF is set. The block diagram of the A/D module is shown in Figure 19-1.

FIGURE 19-1: A/D BLOCK DIAGRAM



Note 1: Channels AN15 through AN12 are not available on the PIC18F6X8X.

2: I/O pins have diode protection to VDD and VSS.

The value in the ADRESH/ADRESL registers is not modified for a Power-on Reset. The ADRESH/ ADRESL registers will contain unknown data after a Power-on Reset.

After the A/D module has been configured as desired, the selected channel must be acquired before the conversion is started. The analog input channels must have their corresponding TRIS bits selected as an input. To determine acquisition time, see **Section 19.1 "A/D Acquisition Requirements"**. After this acquisition time has elapsed, the A/D conversion can be started. An acquisition time can be programmed to occur between setting the GO/$\overline{DONE}$ bit and the actual start of the conversion.

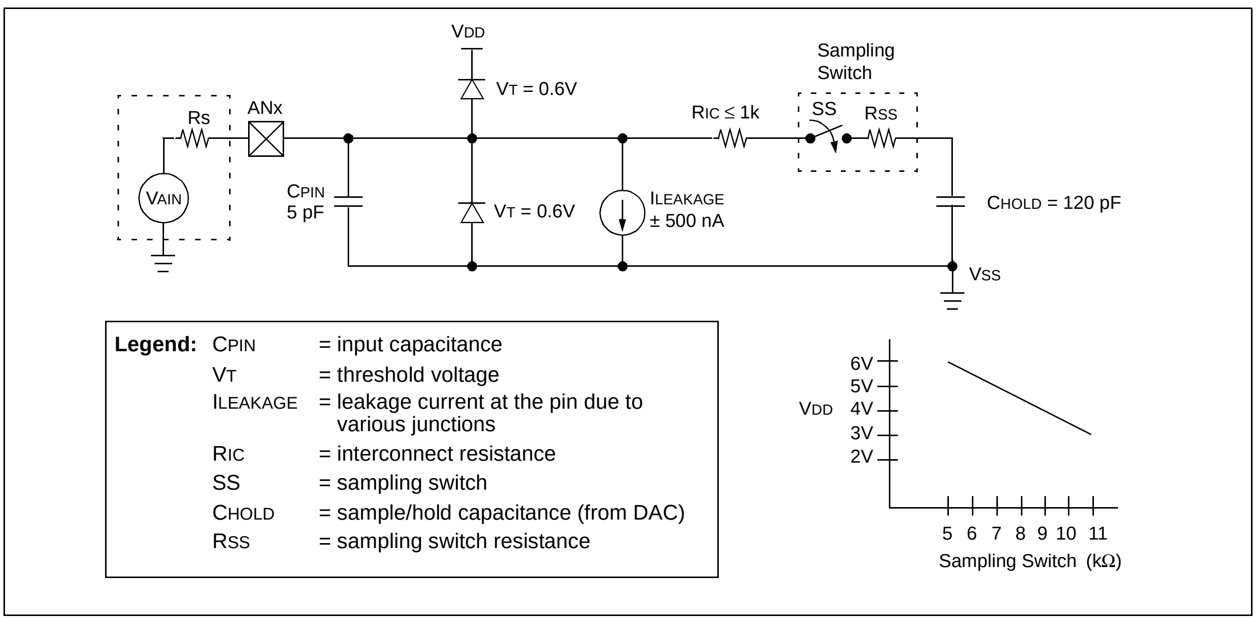
The following steps should be followed to do an A/D conversion:

1. Configure the A/D module:
   - Configure analog pins, voltage reference and digital I/O (ADCON1)
   - Select A/D input channel (ADCON0)
   - Select A/D acquisition time (ADCON2)
   - Select A/D conversion clock (ADCON2)
   - Turn on A/D module (ADCON0)
2. Configure A/D interrupt (if desired):
   - Clear ADIF bit
   - Set ADIE bit
   - Set GIE bit
3. Wait the required acquisition time (if required).
4. Start conversion:
   - Set GO/$\overline{DONE}$ bit (ADCON0 register)
5. Wait for A/D conversion to complete by either:
   - Polling for the GO/$\overline{DONE}$ bit to be cleared

   or

   - Waiting for the A/D interrupt
6. Read A/D Result registers (ADRESH:ADRESL); clear bit ADIF if required.
7. For next conversion, go to step 1 or step 2 as required. The A/D conversion time per bit is defined as T_{AD}. A minimum wait of 2 T_{AD} is required before next acquisition starts.

FIGURE 19-2: ANALOG INPUT MODEL



19.1 A/D Acquisition Requirements

For the A/D converter 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 19-2. The source impedance (R_S) and the internal sampling switch (R_{SS}) impedance directly affect the time required to charge the capacitor C_{HOLD}. The sampling switch (R_{SS}) impedance varies over the device voltage (V_{DD}). The source impedance affects the offset voltage at the analog input (due to pin leakage current). **The maximum recommended impedance for analog sources is 2.5 kΩ**. After the analog input channel is selected (changed), this acquisition must be done before the conversion can be started.

Note:	When the conversion is started, the holding capacitor is disconnected from the input pin.

To calculate the minimum acquisition time, Equation 19-1 may be used. This equation assumes that 1/2 LSb error is used (1024 steps for the A/D). The 1/2 LSb error is the maximum error allowed for the A/D to meet its specified resolution.

Example 19-1 shows the calculation of the minimum required acquisition time, T_{ACQ}. This calculation is based on the following application system assumptions:

C_{HOLD}	=	120 pF
Rs	=	2.5 kΩ
Conversion Error	≤	1/2 LSb
V_{DD}	=	5V → Rss = 7 kΩ
Temperature	=	50°C (system max.)
V_{HOLD}	=	0V @ time = 0

19.2 A/D VREF+ and VREF- References

If external voltage references are used instead of the internal AV_{DD} and AV_{SS} sources, the source impedance of the $V_{REF}+$ and $V_{REF}-$ voltage sources must be considered. During acquisition, currents supplied by these sources are insignificant. However, during conversion, the A/D module sinks and sources current through the reference sources. The effect of this current, as specified in parameter A50, along with source impedance must be considered to meet specified A/D resolution.

Note:	When using external voltage references with the A/D converter, the source impedance of the external voltage references must be less than 20Ω to obtain the specified A/D resolution. Higher reference source impedances will increase both offset and gain errors. Resistive voltage dividers will not provide a sufficiently low source impedance. To maintain the best possible performance in A/D conversions, external V_{REF} inputs should be buffered with an operational amplifier or other low output impedance circuit.

EQUATION 19-1: ACQUISITION TIME

$$T_{ACQ} = \text{Amplifier Settling Time} + \text{Holding Capacitor Charging Time} + \text{Temperature Coefficient}$$
$$= T_{AMP} + T_C + T_{COFF}$$

EQUATION 19-2: A/D MINIMUM CHARGING TIME

$$V_{HOLD} = (V_{REF} - (V_{REF}/2048)) \cdot (1 - e^{(-T_C/C_{HOLD}(R_{IC} + R_{SS} + R_S))})$$

or

$$T_C = -(120\ pF)(1\ k\Omega + R_{SS} + R_S)\ \ln(1/2047)$$

EXAMPLE 19-1: CALCULATING THE MINIMUM REQUIRED ACQUISITION TIME

$T_{ACQ} = T_{AMP} + T_C + T_{COFF}$

Temperature coefficient is only required for temperatures > 25°C.

$T_{ACQ} = 2\ \mu s + T_C + [(Temp - 25°C)(0.05\ \mu s/°C)]$

$T_C = -C_{HOLD}\ (R_{IC} + R_{SS} + R_S)\ \ln(1/2047)$
$-120\ pF\ (1\ k\Omega + 7\ k\Omega + 2.5\ k\Omega)\ \ln(0.0004885)$
$-120\ pF\ (10.5\ k\Omega)\ \ln(0.0004885)$
$-1.26\ \mu s\ (-7.6241)$
$9.61\ \mu s$

$T_{ACQ} = 2\ \mu s + 9.61\ \mu s + [(50°C - 25°C)(0.05\ \mu s/°C)]$
$11.61\ \mu s + 1.25\ \mu s$
$12.86\ \mu s$

19.3 Selecting and Configuring Automatic Acquisition Time

The ADCON2 register allows the user to select an acquisition time that occurs each time the GO/$\overline{\text{DONE}}$ bit is set.

When the GO/$\overline{\text{DONE}}$ bit is set, sampling is stopped and a conversion begins. The user is responsible for ensuring the required acquisition time has passed between selecting the desired input channel and setting the GO/$\overline{\text{DONE}}$ bit. This occurs when the ACQT2:ACQT0 bits (ADCON2<5:3>) remain in their Reset state ('000') and is compatible with devices that do not offer programmable acquisition times.

If desired, the ACQT bits can be set to select a programmable acquisition time for the A/D module. When the GO/$\overline{\text{DONE}}$ bit is set, the A/D module continues to sample the input for the selected acquisition time, then automatically begins a conversion. Since the acquisition time is programmed, there may be no need to wait for an acquisition time between selecting a channel and setting the GO/$\overline{\text{DONE}}$ bit.

In either case, when the conversion is completed, the GO/$\overline{\text{DONE}}$ bit is cleared, the ADIF flag is set, and the A/D begins sampling the currently selected channel again. If an acquisition time is programmed, there is nothing to indicate if the acquisition time has ended or if the conversion has begun.

19.4 Selecting the A/D Conversion Clock

The A/D conversion time per bit is defined as TAD. The A/D conversion requires 11 TAD per 10-bit conversion. The source of the A/D conversion clock is software selectable. There are seven possible options for TAD:

- 2 TOSC
- 4 TOSC
- 8 TOSC
- 16 TOSC
- 32 TOSC
- 64 TOSC
- Internal RC Oscillator

For correct A/D conversions, the A/D conversion clock (TAD) must be as short as possible but greater than the minimum TAD (approximately 2 µs, see parameter 130 for more information).

Table 19-1 shows the resultant TAD times derived from the device operating frequencies and the A/D clock source selected.

19.5 Configuring Analog Port Pins

The ADCON1, TRISA, TRISF and TRISH registers control the operation of the A/D port pins. The port pins needed as analog inputs must have their corresponding TRIS bits set (input). If the TRIS bit is cleared (output), the digital output level (VOH or VOL) will be converted.

The A/D operation is independent of the state of the CHS3:CHS0 bits and the TRIS bits.

> **Note 1:** When reading the port register, all pins configured as analog input channels will read as cleared (a low level). Pins configured as digital inputs will convert an analog input. Analog levels on a digitally configured input will not affect the conversion accuracy.
>
> **2:** Analog levels on any pin defined as a digital input may cause the input buffer to consume current out of the device's specification limits.

TABLE 19-1: TAD vs. DEVICE OPERATING FREQUENCIES

AD Clock Source (TAD)		Maximum Device Frequency	
Operation	**ADCS2:ADCS0**	**PIC18FXX80/XX85**	**PIC18LFXX80/XX85**
2 TOSC	000	1.25 MHz	666 kHz
4 TOSC	100	2.50 MHz	1.33 MHz
8 TOSC	001	5.00 MHz	2.66 MHz
16 TOSC	101	10.0 MHz	5.33 MHz
32 TOSC	010	20.0 MHz	10.65 MHz
64 TOSC	110	40.0 MHz	21.33 MHz
RC(3)	x11	1.00 MHz(1)	1.00 MHz(2)

Note 1: The RC source has a typical TAD time of 4 µs.
2: The RC source has a typical TAD time of 6 µs.
3: For device frequencies above 1 MHz, the device must be in Sleep for the entire conversion or the A/D accuracy may be out of specification.

19.6 A/D Conversions

Figure 19-3 shows the operation of the A/D converter after the GO bit has been set and the ACQT2:ACQT0 bits are cleared. A conversion is started after the following instruction to allow entry into Sleep mode before the conversion begins.

Figure 19-4 shows the operation of the A/D converter after the GO bit has been set, the ACQT2:ACQT0 bits are set to '010' and selecting a 4 TAD acquisition time before the conversion starts.

Clearing the GO/DONE bit during a conversion will abort the current conversion. The A/D Result register pair will not be updated with the partially completed A/D conversion sample. This means the ADRESH:ADRESL registers will continue to contain the value of the last completed conversion (or the last value written to the ADRESH:ADRESL registers).

After the A/D conversion is completed or aborted, a 2 TAD wait is required before the next acquisition can be started. After this wait, acquisition on the selected channel is automatically started.

Note:	The GO/DONE bit should **NOT** be set in the same instruction that turns on the A/D.

19.7 Use of the CCP2 Trigger

An A/D conversion can be started by the "special event trigger" of the CCP2 module. This requires that the CCP2M3:CCP2M0 bits (CCP2CON<3:0>) be programmed as '1011' and that the A/D module is enabled (ADON bit is set). When the trigger occurs, the GO/DONE bit will be set, starting the A/D conversion and the Timer1 (or Timer3) counter will be reset to zero. Timer1 (or Timer3) is reset to automatically repeat the A/D acquisition period with minimal software overhead (moving ADRESH/ADRESL to the desired location). The appropriate analog input channel must be selected and the minimum acquisition done before the "special event trigger" sets the GO/DONE bit (starts a conversion).

If the A/D module is not enabled (ADON is cleared), the "special event trigger" will be ignored by the A/D module but will still reset the Timer1 (or Timer3) counter.

FIGURE 19-3: A/D CONVERSION TAD CYCLES (ACQT<2:0> = 000, TACQ = 0)

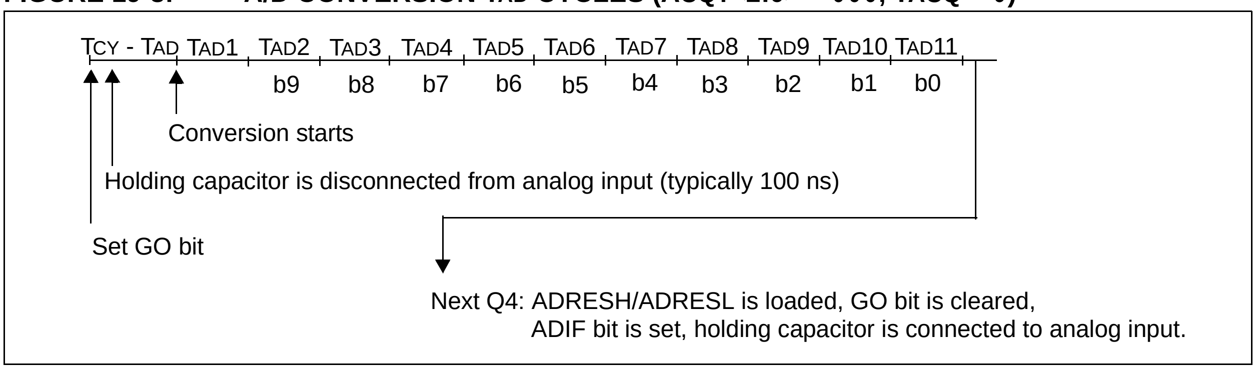


FIGURE 19-4: A/D CONVERSION TAD CYCLES (ACQT<2:0> = 010, TACQ = 4 TAD)

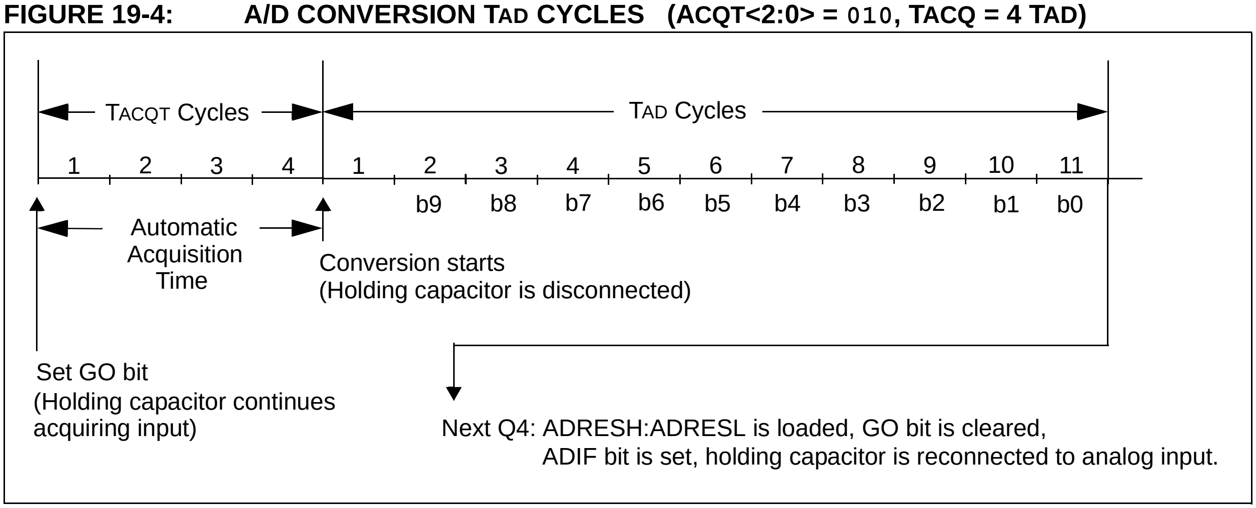


TABLE 19-2: SUMMARY OF A/D REGISTERS

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other Resets
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 0000	0000 0000
PIR1	PSPIF	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
PIE1	PSPIE	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
IPR1	PSPIP	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	1111 1111	1111 1111
PIR2	—	CMIF	—	EEIF	BCLIF	LVDIF	TMR3IF	CCP2IF	-0-0 0000	-0-0 0000
PIE2	—	CMIE	—	EEIE	BCLIE	LVDIE	TMR3IE	CCP2IE	-0-0 0000	-0-0 0000
IPR2	—	CMIP	—	EEIP	BCLIP	LVDIP	TMR3IP	CCP2IP	-1-1 1111	-1-1 1111
ADRESH	A/D Result Register High Byte								xxxx xxxx	uuuu uuuu
ADRESL	A/D Result Register Low Byte								xxxx xxxx	uuuu uuuu
ADCON0	—	—	CHS3	CHS3	CHS1	CHS0	GO/$\overline{\text{DONE}}$	ADON	--00 0000	--00 0000
ADCON1	—	—	VCFG1	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0	--00 0000	--00 0000
ADCON2	ADFM	—	—	—	—	ADCS2	ADCS1	ADCS0	0--- -000	0--- -000
PORTA	—	RA6	RA5	RA4	RA3	RA2	RA1	RA0	--xx xxxx	--uu uuuu
TRISA	—	PORTA Data Direction Register							--11 1111	--11 1111
PORTF	RF7	RF6	RF5	RF4	RF3	RF2	RF1	RF0	xxxx xxxx	uuuu uuuu
LATF	LATF7	LATF6	LATF5	LATF4	LATF3	LATF2	LATF1	LATF0	xxxx xxxx	uuuu uuuu
TRISF	PORTF Data Direction Control Register								1111 1111	1111 1111
PORTH(1)	RH7	RH6	RH5	RH4	RH3	RH2	RH1	RH0	xxxx xxxx	uuuu uuuu
LATH(1)	LATH7	LATH6	LATH5	LATH4	LATH3	LATH2	LATH1	LATH0	xxxx xxxx	uuuu uuuu
TRISH(1)	PORTH Data Direction Control Register								1111 1111	1111 1111

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used for A/D conversion.
Note 1: Only available on PIC18F8X8X devices.

NOTES:

20.0 COMPARATOR MODULE

The comparator module contains two analog comparators. The inputs to the comparators are multiplexed with the RF1 through RF6 pins. The on-chip voltage reference (**Section 21.0 "Comparator Voltage Reference Module"**) can also be an input to the comparators.

The CMCON register, shown in Register 20-1, controls the comparator input and output multiplexers. A block diagram of the various comparator configurations is shown in Figure 20-1.

REGISTER 20-1: CMCON REGISTER

R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
C2OUT	C1OUT	C2INV	C1INV	CIS	CM2	CM1	CM0
bit 7							bit 0

bit 7 **C2OUT**: Comparator 2 Output bit
<u>When C2INV = 0:</u>
1 = C2 VIN+ > C2 VIN-
0 = C2 VIN+ < C2 VIN-
<u>When C2INV = 1:</u>
1 = C2 VIN+ < C2 VIN-
0 = C2 VIN+ > C2 VIN-

bit 6 **C1OUT**: Comparator 1 Output bit
<u>When C1INV = 0:</u>
1 = C1 VIN+ > C1 VIN-
0 = C1 VIN+ < C1 VIN-
<u>When C1INV = 1:</u>
1 = C1 VIN+ < C1 VIN-
0 = C1 VIN+ > C1 VIN-

bit 5 **C2INV**: Comparator 2 Output Inversion bit
1 = C2 output inverted
0 = C2 output not inverted

bit 4 **C1INV**: Comparator 1 Output Inversion bit
1 = C1 output inverted
0 = C1 output not inverted

bit 3 **CIS**: Comparator Input Switch bit
<u>When CM2:CM0 = 110:</u>
1 = C1 VIN- connects to RF5/AN10
C2 VIN- connects to RF3/AN8
0 = C1 VIN- connects to RF6/AN11
C2 VIN- connects to RF4/AN9

bit 2-0 **CM2:CM0**: Comparator Mode bits
Figure 20-1 shows the Comparator modes and CM2:CM0 bit settings.

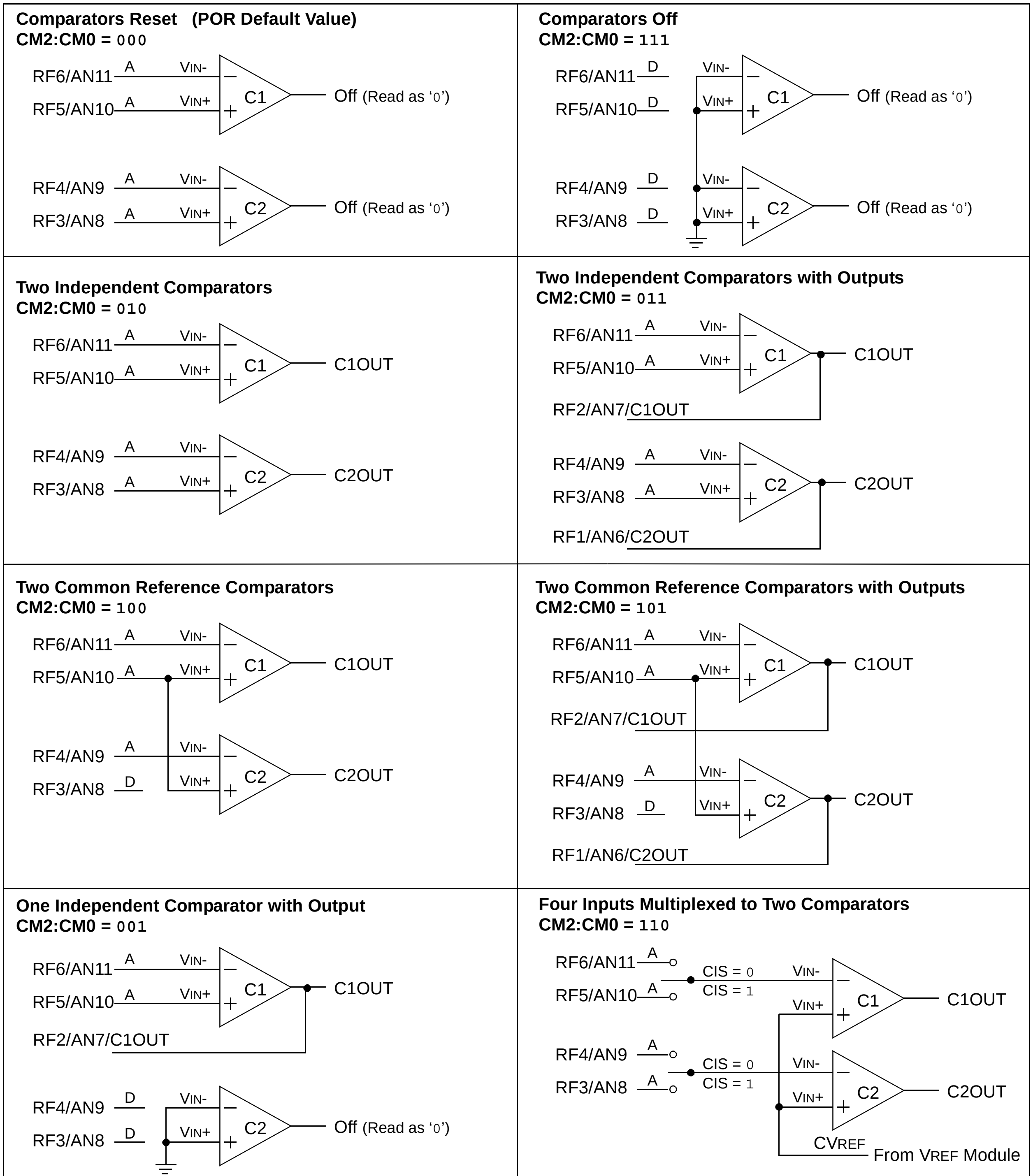
Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

20.1 Comparator Configuration

There are eight modes of operation for the comparators. The CMCON register is used to select these modes. Figure 20-1 shows the eight possible modes. The TRISF register controls the data direction of the comparator pins for each mode. If the Comparator mode is changed, the comparator output level may not be valid for the specified mode change delay shown in **Section 27.0 "Electrical Characteristics"**.

Note:	Comparator interrupts should be disabled during a Comparator mode change. Otherwise, a false interrupt may occur.

FIGURE 20-1: COMPARATOR I/O OPERATING MODES



A = Analog Input, port reads zeros always D = Digital Input CIS (CMCON<3>) = Comparator Input Switch

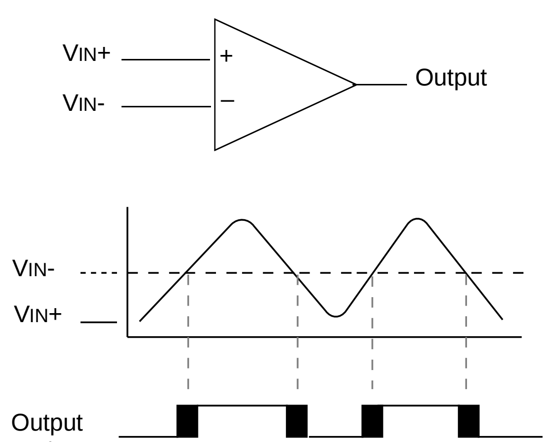
20.2 Comparator Operation

A single comparator is shown in Figure 20-2, along with the relationship between the analog input levels and the digital output. When the analog input at VIN+ is less than the analog input VIN-, the output of the comparator is a digital low level. When the analog input at VIN+ is greater than the analog input VIN-, the output of the comparator is a digital high level. The shaded areas of the output of the comparator in Figure 20-2 represent the uncertainty due to input offsets and response time.

20.3 Comparator Reference

An external or internal reference signal may be used depending on the Comparator Operating mode. The analog signal present at VIN- is compared to the signal at VIN+ and the digital output of the comparator is adjusted accordingly (Figure 20-2).

FIGURE 20-2: SINGLE COMPARATOR



20.3.1 EXTERNAL REFERENCE SIGNAL

When external voltage references are used, the comparator module can be configured to have the comparators operate from the same or different reference sources. However, threshold detector applications may require the same reference. The reference signal must be between VSS and VDD and can be applied to either pin of the comparator(s).

20.3.2 INTERNAL REFERENCE SIGNAL

The comparator module also allows the selection of an internally generated voltage reference for the comparators. **Section 21.0 "Comparator Voltage Reference Module"** contains a detailed description of the comparator voltage reference module that provides this signal. The internal reference signal is used when comparators are in mode CM<2:0> = `110` (Figure 20-1). In this mode, the internal voltage reference is applied to the VIN+ pin of both comparators.

20.4 Comparator Response Time

Response time is the minimum time, after selecting a new reference voltage or input source, before the comparator output has a valid level. If the internal reference is changed, the maximum delay of the internal voltage reference must be considered when using the comparator outputs. Otherwise, the maximum delay of the comparators should be used (**Section 27.0 "Electrical Characteristics"**).

20.5 Comparator Outputs

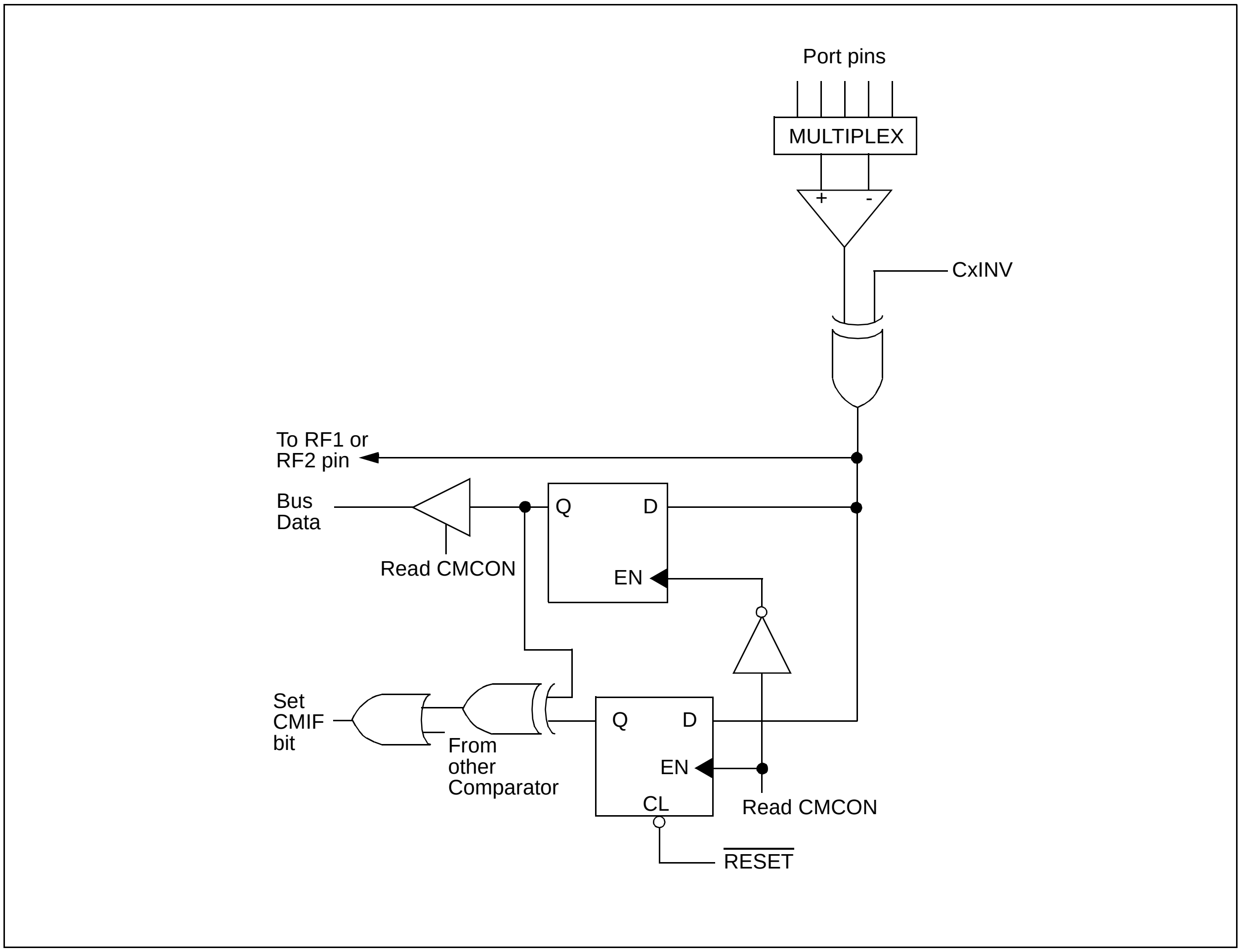
The comparator outputs are read through the CMCON register. These bits are read-only. The comparator outputs may also be directly output to the RF1 and RF2 I/O pins. When enabled, multiplexors in the output path of the RF1 and RF2 pins will switch and the output of each pin will be the unsynchronized output of the comparator. The uncertainty of each of the comparators is related to the input offset voltage and the response time given in the specifications. Figure 20-3 shows the comparator output block diagram.

The TRISA bits will still function as an output enable/disable for the RF1 and RF2 pins while in this mode.

The polarity of the comparator outputs can be changed using the C2INV and C1INV bits (CMCON<4:5>).

Note 1:	When reading the Port register, all pins configured as analog inputs will read as a '`0`'. Pins configured as digital inputs will convert an analog input according to the Schmitt Trigger 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 20-3: COMPARATOR OUTPUT BLOCK DIAGRAM



20.6 Comparator Interrupts

The comparator interrupt flag is set whenever there is a change in the output value of either comparator. Software will need to maintain information about the status of the output bits, as read from CMCON<7:6>, to determine the actual change that occurred. The CMIF bit (PIR registers) is the Comparator Interrupt Flag. The CMIF bit must be reset by clearing it to '0'. Since it is also possible to write a '1' to this register, a simulated interrupt may be initiated.

The CMIE bit (PIE registers) and the PEIE bit (INTCON register) must be set to enable the interrupt. In addition, the GIE bit must also be set. If any of these bits are clear, the interrupt is not enabled, though the CMIF bit will still be set if an interrupt condition occurs.

Note:	If a change in the CMCON register (C1OUT or C2OUT) should occur when a read operation is being executed (start of the Q2 cycle), then the CMIF (PIR registers) interrupt flag may not get set.

The user, in the Interrupt Service Routine, can clear the interrupt in the following manner:

a) Any read or write of CMCON will end the mismatch condition.

b) Clear flag bit CMIF.

A mismatch condition will continue to set flag bit CMIF. Reading CMCON will end the mismatch condition and allow flag bit CMIF to be cleared.

20.7 Comparator Operation During Sleep

When a comparator is active and the device is placed in Sleep mode, the comparator remains active and the interrupt is functional if enabled. This interrupt will wake-up the device from Sleep mode when enabled. While the comparator is powered up, higher Sleep currents than shown in the power-down current specification will occur. Each operational comparator will consume additional current as shown in the comparator specifications. To minimize power consumption while in Sleep mode, turn off the comparators (CM<2:0> = `111`) before entering Sleep. If the device wakes up from Sleep, the contents of the CMCON register are not affected.

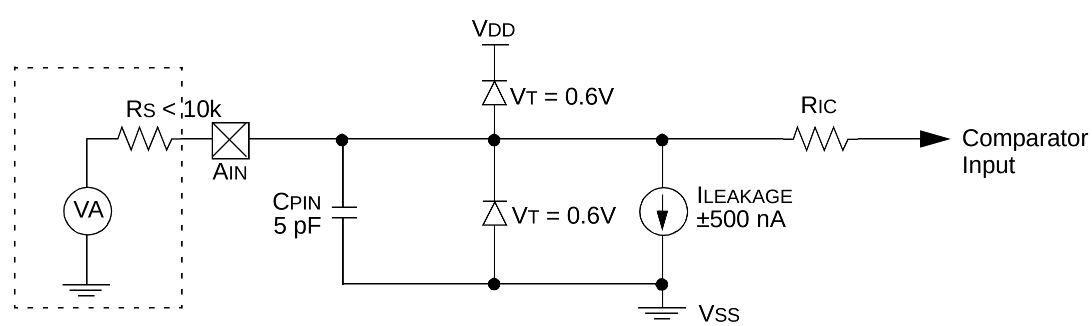
20.8 Effects of a Reset

A device Reset forces the CMCON register to its Reset state, causing the comparator module to be in the Comparator Reset mode (CM<2:0> = `000`). This ensures that all potential inputs are analog inputs. Device current is minimized when analog inputs are present at Reset time. The comparators will be powered down during the Reset interval.

20.9 Analog Input Connection Considerations

A simplified circuit for an analog input is shown in Figure 20-4. Since the analog pins are connected to a digital output, they have reverse biased 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 condition may occur. A maximum source impedance of 10 kΩ is recommended for the analog sources. Any external component connected to an analog input pin, such as a capacitor or a Zener diode, should have very little leakage current.

FIGURE 20-4: COMPARATOR ANALOG INPUT MODEL



Legend:

CPIN	=	Input Capacitance
VT	=	Threshold Voltage
ILEAKAGE	=	Leakage Current at the pin due to various junctions
RIC	=	Interconnect Resistance
RS	=	Source Impedance
VA	=	Analog Voltage

TABLE 20-1: REGISTERS ASSOCIATED WITH COMPARATOR MODULE

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR	Value on all other Resets
CMCON	C2OUT	C1OUT	C2INV	C1INV	CIS	CM2	CM1	CM0	0000 0000	0000 0000
CVRCON	CVREN	CVROE	CVRR	CVRSS	CVR3	CVR2	CVR1	CVR0	0000 0000	0000 0000
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 0000	0000 0000
PIR2	—	CMIF	—	EEIF	BCLIF	LVDIF	TMR3IF	CCP2IF	-0-0 0000	-0-0 0000
PIE2	—	CMIE	—	EEIE	BCLIE	LVDIE	TMR3IE	CCP2IE	-0-0 0000	-0-0 0000
IPR2	—	CMIP	—	EEIP	BCLIP	LVDIP	TMR3IP	CCP2IP	-1-1 1111	-1-1 1111
PORTF	RF7	RF6	RF5	RF4	RF3	RF2	RF1	RF0	xxxx xxxx	uuuu uuuu
LATF	LATF7	LATF6	LATF5	LATF4	LATF3	LATF2	LATF1	LATF0	xxxx xxxx	uuuu uuuu
TRISF	TRISF7	TRISF6	TRISF5	TRISF4	TRISF3	TRISF2	TRISF1	TRISF0	1111 1111	1111 1111

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are unused by the comparator module.

21.0 COMPARATOR VOLTAGE REFERENCE MODULE

The comparator voltage reference is a 16-tap resistor ladder network that provides a selectable voltage reference. The resistor ladder is segmented to provide two ranges of CVREF values and has a power-down function to conserve power when the reference is not being used. The CVRCON register controls the operation of the reference as shown in Register 21-1. The block diagram is given in Figure 21-1.

The comparator reference supply voltage can come from either VDD or VSS, or the external VREF+ and VREF- that are multiplexed with RA3 and RA2. The comparator reference supply voltage is controlled by the CVRSS bit.

21.1 Configuring the Comparator Voltage Reference

The comparator voltage reference can output 16 distinct voltage levels for each range. The equations used to calculate the output of the comparator voltage reference are as follows:

If CVRR = 1:
CVREF = (CVR<3:0>/24) x CVRSRC

If CVRR = 0:
CVREF = (CVDD x 1/4) + (CVR<3:0>/32) x CVRSRC

The settling time of the comparator voltage reference must be considered when changing the CVREF output (**Section 27.0 "Electrical Characteristics"**).

REGISTER 21-1: CVRCON REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CVREN	CVROE	CVRR	CVRSS	CVR3	CVR2	CVR1	CVR0
bit 7							bit 0

bit 7 **CVREN**: Comparator Voltage Reference Enable bit
1 = CVREF circuit powered on
0 = CVREF circuit powered down

bit 6 **CVROE**: Comparator VREF Output Enable bit(1)
1 = CVREF voltage level is also output on the RF5/AN10/C1IN+/CVREF pin
0 = CVREF voltage is disconnected from the RF5/AN10/C1IN+/CVREF pin

bit 5 **CVRR**: Comparator VREF Range Selection bit
1 = 0.00 CVRSRC to 0.625 CVRSRC with CVRSRC/24 step size
0 = 0.25 CVRSRC to 0.71875 CVRSRC with CVRSRC/32 step size

bit 4 **CVRSS**: Comparator VREF Source Selection bit
1 = Comparator reference source, CVRSRC = VREF+ – VREF-
0 = Comparator reference source, CVRSRC = VDD – VSS

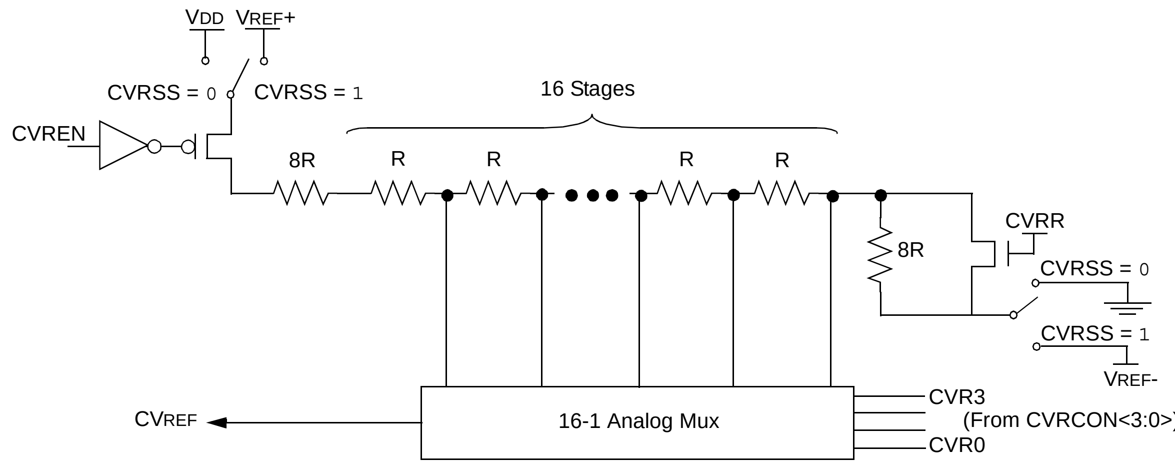
Note: To select (VREF+ – VREF-) as the comparator voltage reference source, the voltage reference configuration bits in the ADCON1 register (ADCON1<5:4>) must also be set to '11'.

bit 3-0 **CVR3:CVR0:** Comparator VREF Value Selection bits ($0 \leq VR3:VR0 \leq 15$)
When CVRR = 1:
CVREF = (CVR<3:0>/24) • (CVRSRC)
When CVRR = 0:
CVREF = 1/4 • (CVRSRC) + (CVR3:CVR0/32) • (CVRSRC)

Note 1: If enabled for output, RF5 must also be configured as an input by setting TRISF<5> to '1'.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

FIGURE 21-1: VOLTAGE REFERENCE BLOCK DIAGRAM



Note: R is defined in **Section 27.0 "Electrical Characteristics"**.

21.2 Voltage Reference Accuracy/Error

The full range of voltage reference cannot be realized due to the construction of the module. The transistors on the top and bottom of the resistor ladder network (Figure 21-1) keep CVREF from approaching the reference source rails. The voltage reference is derived from the reference source; therefore, the CVREF output changes with fluctuations in that source. The tested absolute accuracy of the voltage reference can be found in **Section 27.0 "Electrical Characteristics"**.

21.3 Operation During Sleep

When the device wakes up from Sleep through an interrupt or a Watchdog Timer time-out, the contents of the CVRCON register are not affected. To minimize current consumption in Sleep mode, the voltage reference should be disabled.

21.4 Effects of a Reset

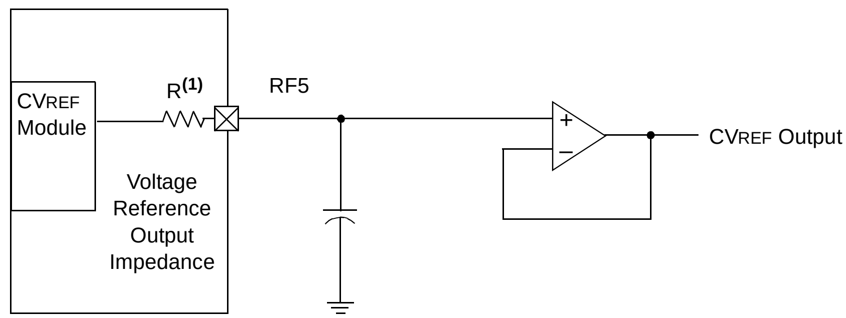
A device Reset disables the voltage reference by clearing bit CVREN (CVRCON<7>). This Reset also disconnects the reference from the RA2 pin by clearing bit CVROE (CVRCON<6>) and selects the high-voltage range by clearing bit CVRR (CVRCON<5>). The VRSS value select bits, CVRCON<3:0>, are also cleared.

21.5 Connection Considerations

The voltage reference module operates independently of the comparator module. The output of the reference generator may be connected to the RF5 pin if the TRISF<5> bit is set and the CVROE bit is set. Enabling the voltage reference output onto the RF5 pin with an input signal present will increase current consumption. Connecting RF5 as a digital output with VRSS enabled will also increase current consumption.

The RF5 pin can be used as a simple D/A output with limited drive capability. Due to the limited current drive capability, a buffer must be used on the voltage reference output for external connections to VREF. Figure 21-2 shows an example buffering technique.

FIGURE 21-2: VOLTAGE REFERENCE OUTPUT BUFFER EXAMPLE



Note 1: R is dependent upon the voltage reference configuration bits CVRCON<3:0> and CVRCON<5>.

TABLE 21-1: REGISTERS ASSOCIATED WITH COMPARATOR VOLTAGE REFERENCE

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR	Value on all other Resets
CVRCON	CVREN	CVROE	CVRR	CVRSS	CVR3	CVR2	CVR1	CVR0	0000 0000	0000 0000
CMCON	C2OUT	C1OUT	C2INV	C1INV	CIS	CM2	CM1	CM0	0000 0000	0000 0000
TRISF	TRISF7	TRISF6	TRISF5	TRISF4	TRISF3	TRISF2	TRISF1	TRISF0	1111 1111	1111 1111

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'.
Shaded cells are not used with the comparator voltage reference.

NOTES:

22.0 LOW-VOLTAGE DETECT

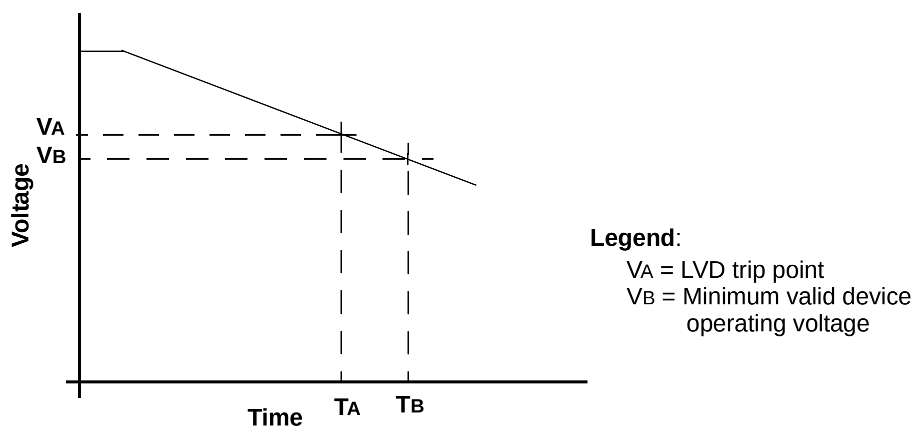
In many applications, the ability to determine if the device voltage (V_{DD}) is below a specified voltage level is a desirable feature. A window of operation for the application can be created where the application software can do “housekeeping tasks” before the device voltage exits the valid operating range. This can be done using the Low-Voltage Detect module.

This module is a software programmable circuitry where a device voltage trip point can be specified. When the voltage of the device becomes lower then the specified point, an interrupt flag is set. If the interrupt is enabled, the program execution will branch to the interrupt vector address and the software can then respond to that interrupt source.

The Low-Voltage Detect circuitry is completely under software control. This allows the circuitry to be “turned off” by the software which minimizes the current consumption for the device.

Figure 22-1 shows a possible application voltage curve (typically for batteries). Over time, the device voltage decreases. When the device voltage equals voltage V_A, the LVD logic generates an interrupt. This occurs at time T_A. The application software then has the time, until the device voltage is no longer in valid operating range, to shut down the system. Voltage point V_B is the minimum valid operating voltage specification. This occurs at time T_B. The difference, $T_B – T_A$, is the total time for shutdown.

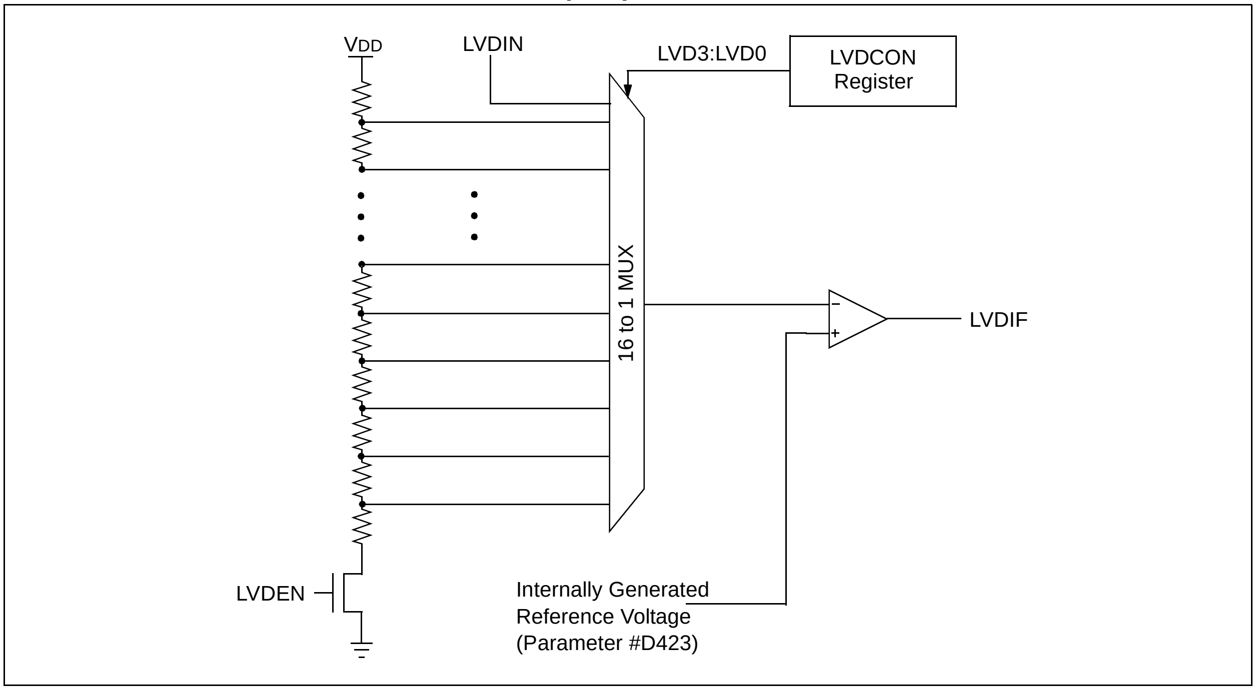
FIGURE 22-1: TYPICAL LOW-VOLTAGE DETECT APPLICATION



The block diagram for the LVD module is shown in Figure 22-2. A comparator uses an internally generated reference voltage as the set point. When the selected tap output of the device voltage crosses the set point (is lower than), the LVDIF bit is set.

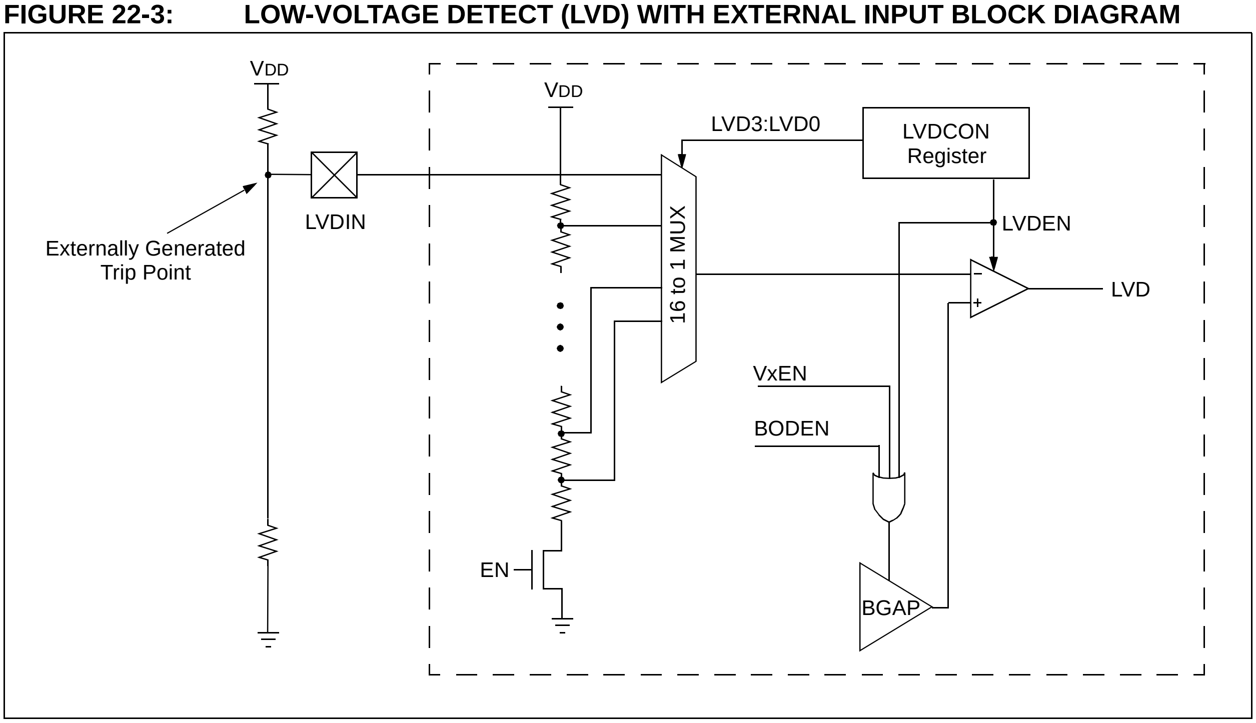
Each node in the resistor divider represents a “trip point” voltage. The “trip point” voltage is the minimum supply voltage level at which the device can operate before the LVD module asserts an interrupt. When the supply voltage is equal to the trip point, the voltage tapped off of the resistor array is equal to the 1.2V internal reference voltage generated by the voltage reference module. The comparator then generates an interrupt signal setting the LVDIF bit. This voltage is software programmable to any one of 16 values (see Figure 22-2). The trip point is selected by programming the LVDL3:LVDL0 bits (LVDCON<3:0>).

FIGURE 22-2: LOW-VOLTAGE DETECT (LVD) BLOCK DIAGRAM



The LVD module has an additional feature that allows the user to supply the trip voltage to the module from an external source. This mode is enabled when bits LVDL3:LVDL0 are set to '1111'. In this state, the comparator input is multiplexed from the external input pin, LVDIN (Figure 22-3). This gives users flexibility because it allows them to configure the Low-Voltage Detect interrupt to occur at any voltage in the valid operating range.

FIGURE 22-3: LOW-VOLTAGE DETECT (LVD) WITH EXTERNAL INPUT BLOCK DIAGRAM



22.1 Control Register

The Low-Voltage Detect Control register controls the operation of the Low-Voltage Detect circuitry.

REGISTER 22-1: LVDCON REGISTER

U-0	U-0	R-0	R/W-0	R/W-0	R/W-1	R/W-0	R/W-1
—	—	IRVST	LVDEN	LVDL3	LVDL2	LVDL1	LVDL0
bit 7							bit 0

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

bit 5 **IRVST:** Internal Reference Voltage Stable Flag bit
1 = Indicates that the Low-Voltage Detect logic will generate the interrupt flag at the specified voltage range
0 = Indicates that the Low-Voltage Detect logic will not generate the interrupt flag at the specified voltage range and the LVD interrupt should not be enabled

bit 4 **LVDEN:** Low-Voltage Detect Power Enable bit
1 = Enables LVD, powers up LVD circuit
0 = Disables LVD, powers down LVD circuit

bit 3-0 **LVDL3:LVDL0:** Low-Voltage Detection Limit bits
1111 = External analog input is used (input comes from the LVDIN pin)
1110 = 4.5V-4.77V
1101 = 4.2V-4.45V
1100 = 4.0V-4.24V
1011 = 3.8V-4.03V
1010 = 3.6V-3.82V
1001 = 3.5V-3.71V
1000 = 3.3V-3.50V
0111 = 3.0V-3.18V
0110 = 2.8V-2.97V
0101 = 2.7V-2.86V
0100 = 2.5V-2.65V
0011 = 2.4V-2.54V
0010 = 2.2V-2.33V
0001 = 2.0V-2.12V
0000 = Reserved

Note: LVDL3:LVDL0 modes which result in a trip point below the valid operating voltage of the device are not tested.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

22.2 Operation

Depending on the power source for the device voltage, the voltage normally decreases relatively slowly. This means that the LVD module does not need to be constantly operating. To decrease the current requirements, the LVD circuitry only needs to be enabled for short periods where the voltage is checked. After doing the check, the LVD module may be disabled.

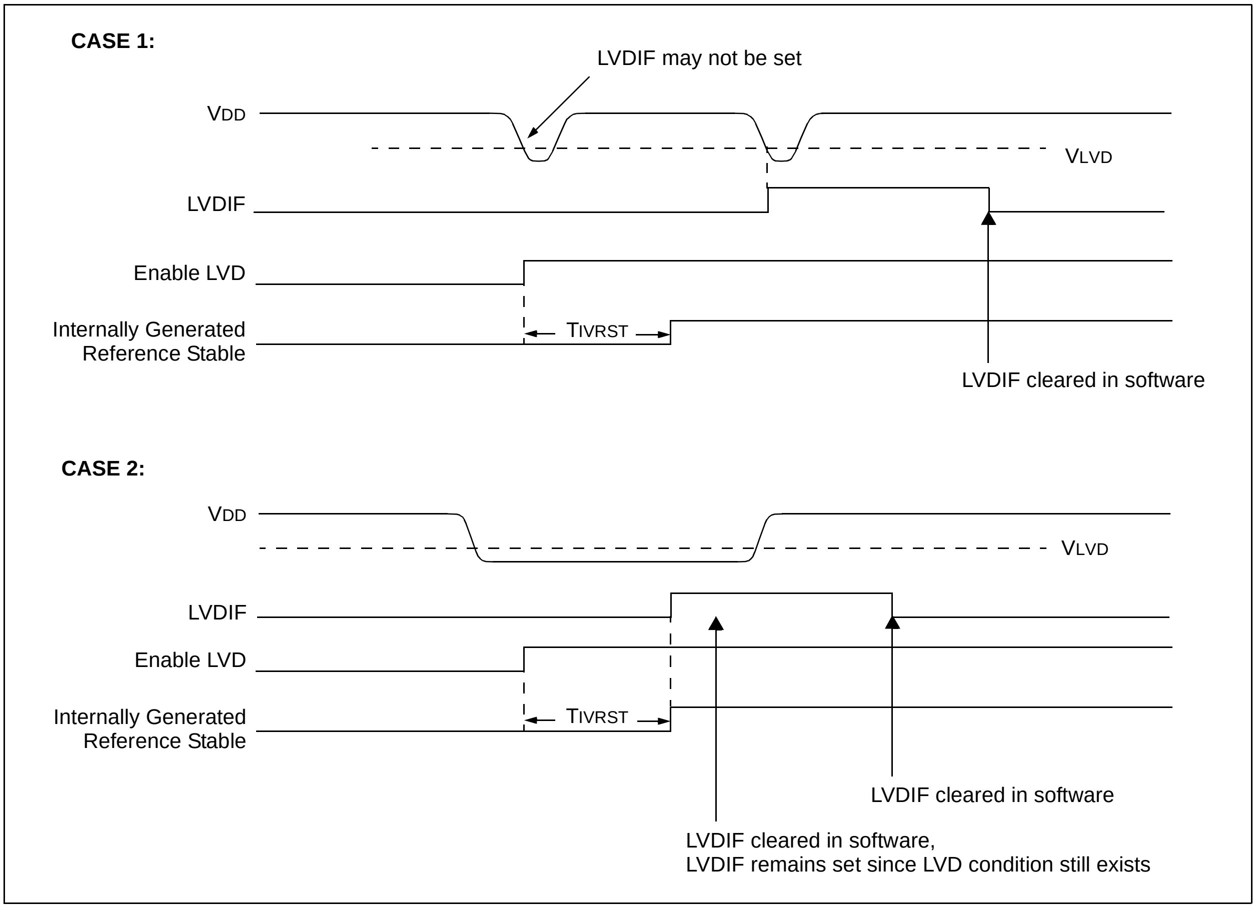
Each time that the LVD module is enabled, the circuitry requires some time to stabilize. After the circuitry has stabilized, all status flags may be cleared. The module will then indicate the proper state of the system.

The following steps are needed to set up the LVD module:

1. Write the value to the LVDL3:LVDL0 bits (LVDCON register) which selects the desired LVD trip point.
2. Ensure that LVD interrupts are disabled (the LVDIE bit is cleared or the GIE bit is cleared).
3. Enable the LVD module (set the LVDEN bit in the LVDCON register).
4. Wait for the LVD module to stabilize (the IRVST bit to become set).
5. Clear the LVD interrupt flag which may have falsely become set until the LVD module has stabilized (clear the LVDIF bit).
6. Enable the LVD interrupt (set the LVDIE and the GIE bits).

Figure 22-4 shows typical waveforms that the LVD module may be used to detect.

FIGURE 22-4: LOW-VOLTAGE DETECT WAVEFORMS



22.2.1 REFERENCE VOLTAGE SET POINT

The internal reference voltage of the LVD module, specified in electrical specification parameter #D423, may be used by other internal circuitry (the Programmable Brown-out Reset). If these circuits are disabled (lower current consumption), the reference voltage circuit requires a time to become stable before a low-voltage condition can be reliably detected. This time is invariant of system clock speed. This start-up time is specified in electrical specification parameter #36. The low-voltage interrupt flag will not be enabled until a stable reference voltage is reached. Refer to the waveform in Figure 22-4.

22.2.2 CURRENT CONSUMPTION

When the module is enabled, the LVD comparator and voltage divider are enabled and will consume static current. The voltage divider can be tapped from multiple places in the resistor array. Total current consumption, when enabled, is specified in electrical specification parameter #D022B.

22.3 Operation During Sleep

When enabled, the LVD circuitry continues to operate during Sleep. If the device voltage crosses the trip point, the LVDIF 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.

22.4 Effects of a Reset

A device Reset forces all registers to their Reset state. This forces the LVD module to be turned off.

NOTES:

23.0 ECAN MODULE

PIC18F6585/8585/6680/8680 devices contain an Enhanced Controller Area Network (ECAN) module. The ECAN module is fully backward compatible with the CAN module available in PIC18CXX8 and PIC18FXX8 devices.

The Controller Area Network (CAN) module is a serial interface which is useful for communicating with other peripherals or microcontroller devices. This interface, or protocol, was designed to allow communications within noisy environments.

The ECAN module is a communication controller, implementing the CAN 2.0A or B protocol as defined in the BOSCH specification. The module will support CAN 1.2, CAN 2.0A, CAN 2.0B Passive and CAN 2.0B Active versions of the protocol. The module implementation is a full CAN system; however, the CAN specification is not covered within this data sheet. Refer to the BOSCH CAN specification for further details.

The module features are as follows:

- Implementation of the CAN protocol CAN 1.2, CAN 2.0A and CAN 2.0B
- DeviceNet™ data bytes filter support
- Standard and extended data frames
- 0-8 bytes data length
- Programmable bit rate up to 1 Mbit/sec
- Fully backward compatible with PIC18XX8 CAN module
- Three modes of operation:
  - Mode 0 – Legacy mode
  - Mode 1 – Enhanced Legacy mode with DeviceNet support
  - Mode 2 – FIFO mode with DeviceNet support
- Support for remote frames with automated handling
- Double-buffered receiver with two prioritized received message storage buffers
- Six buffers programmable as RX and TX message buffers
- 16 full (standard/extended identifier) acceptance filters that can be linked to one of four masks
- Two full acceptance filter masks that can be assigned to any filter
- One full acceptance filter that can be used as either an acceptance filter or acceptance filter mask
- Three dedicated transmit buffers with application specified prioritization and abort capability
- Programmable wake-up functionality with integrated low-pass filter
- Programmable Loopback mode supports self-test operation
- Signaling via interrupt capabilities for all CAN receiver and transmitter error states
- Programmable clock source
- Programmable link to timer module for time-stamping and network synchronization
- Low-Power Sleep mode

23.1 Module Overview

The CAN bus module consists of a protocol engine and message buffering and control. The CAN protocol engine automatically handles all functions for receiving and transmitting messages on the CAN bus. Messages are transmitted by first loading the appropriate data registers. Status and errors can be checked by reading the appropriate registers. Any message detected on the CAN bus is checked for errors and then matched against filters to see if it should be received and stored in one of the two receive registers.

The CAN module supports the following frame types:

- Standard Data Frame
- Extended Data Frame
- Remote Frame
- Error Frame
- Overload Frame Reception
- Interframe Space Generation/Detection

The CAN module uses the RG0/CANTX1, RG1/CANTX2 and RG2/CANRX pins to interface with the CAN bus. In Normal mode, the CAN module automatically overrides the TRISG0 and TRISG1 bits of the CAN module pins.

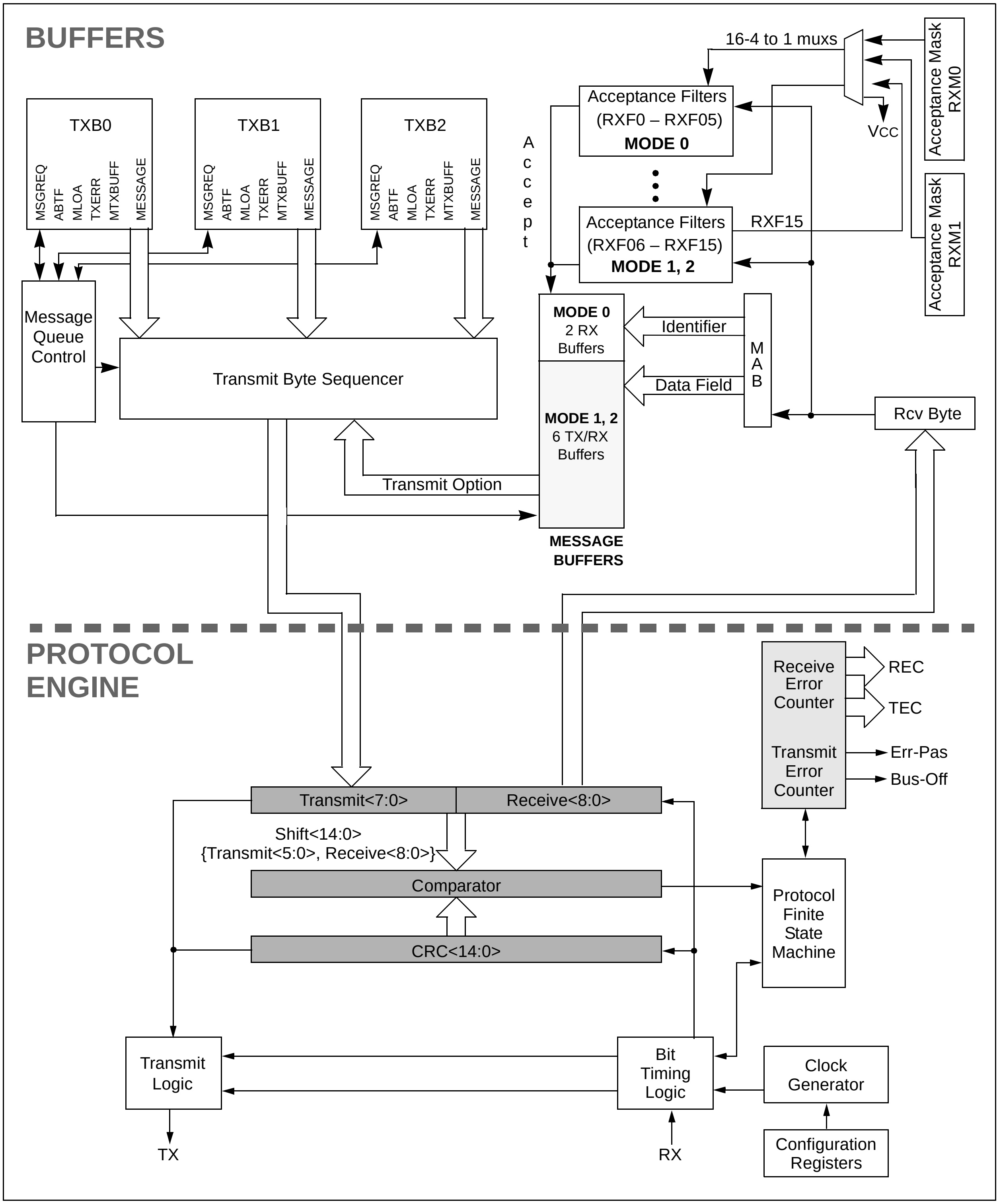
23.1.1 MODULE FUNCTIONALITY

The CAN bus module consists of a protocol engine, message buffering and control (see Figure 23-1). The protocol engine can best be understood by defining the types of data frames to be transmitted and received by the module.

The following sequence illustrates the necessary initialization steps before the ECAN module can be used to transmit or receive a message. Steps can be added or removed depending on the requirements of the application.

1. Ensure that the ECAN module is in Configuration mode.
2. Select ECAN Operational mode.
3. Set up the baud rate registers.
4. Set up the filter and mask registers.
5. Set the ECAN module to Normal mode or any other mode required by the application logic.

FIGURE 23-1: CAN BUFFERS AND PROTOCOL ENGINE BLOCK DIAGRAM



23.2 CAN Module Registers

Note:	Not all CAN registers are available in the Access Bank.

There are many control and data registers associated with the CAN module. For convenience, their descriptions have been grouped into the following sections:

- Control and Status Registers
- Dedicated Transmit Buffer Registers
- Dedicated Receive Buffer Registers
- Programmable TX/RX and Auto RTR Buffers
- Baud Rate Control Registers
- I/O Control Register
- Interrupt Status and Control Registers

Detailed descriptions of each register and their usage are described in the following sections.

23.2.1 CAN CONTROL AND STATUS REGISTERS

The registers described in this section control the overall operation of the CAN module and show its operational status.

REGISTER 23-1: CANCON: CAN CONTROL REGISTER

	R/W-1	R/W-0	R/W-0	R/S-0	R/W-0	R/W-0	R/W-0	U-0
Mode 0	REQOP2	REQOP1	REQOP0	ABAT	WIN2	WIN1	WIN0	—

	R/W-1	R/W-0	R/W-0	R/S-0	U-0	U-0	U-0	U-0
Mode 1	REQOP2	REQOP1	REQOP0	ABAT	—	—	—	—

	R/W-1	R/W-0	R/W-0	R/S-0	R-0	R-0	R-0	R-0
Mode 2	REQOP2	REQOP1	REQOP0	ABAT	FP3	FP2	FP1	FP0
	bit 7							bit 0

bit 7-5 **REQOP2:REQOP0:** Request CAN Operation Mode bits
1xx = Request Configuration mode
011 = Request Listen Only mode
010 = Request Loopback mode
001 = Request Disable mode
000 = Request Normal mode

bit 4 **ABAT:** Abort All Pending Transmissions bit
1 = Abort all pending transmissions (in all transmit buffers)
0 = Transmissions proceeding as normal

bit 3-1 Mode 0:
WIN2:WIN0: Window Address bits
This selects which of the CAN buffers to switch into the access bank area. This allows access to the buffer registers from any data memory bank. After a frame has caused an interrupt, the ICODE2:ICODE0 bits can be copied to the WIN2:WIN0 bits to select the correct buffer. See Example 23-2 for a code example.
111 = Receive Buffer 0
110 = Receive Buffer 0
101 = Receive Buffer 1
100 = Transmit Buffer 0
011 = Transmit Buffer 1
010 = Transmit Buffer 2
001 = Receive Buffer 0
000 = Receive Buffer 0

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

bit 3-0 Mode 1:
Unimplemented: Read as '0'
Mode 2:
FP3:FP0: FIFO Read Pointer bits
These bits point to the message buffer to be read.
0111:0000 = Message buffer to be read
1111:1000 = Reserved

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

REGISTER 23-2: CANSTAT: CAN STATUS REGISTER

Mode 0

R-1	R-0	R-0	R-0	R-0	R-0	R-0	U-0
OPMODE2[(1)]	OPMODE1[(1)]	OPMODE0[(1)]	—	ICODE2	ICODE1	ICODE0	—

Mode 1, 2

R-1	R-0	R-0	R-0	R-0	R-0	R-0	R-0
OPMODE2[(1)]	OPMODE1[(1)]	OPMODE0[(1)]	EICODE4	EICODE3	EICODE2	EICODE1	EICODE0
bit 7							bit 0

bit 7-5 **OPMODE2:OPMODE0:** Operation Mode Status bits[(1)]

`111` = Reserved
`110` = Reserved
`101` = Reserved
`100` = Configuration mode
`011` = Listen Only mode
`010` = Loopback mode
`001` = Disable/Sleep mode
`000` = Normal mode

bit 4 Mode 0:

Unimplemented: Read as '`0`'

bit 3-1 **ICODE2:ICODE0**: Interrupt Code bits in Mode 0

When an interrupt occurs, a prioritized coded interrupt value will be present in these bits. This code indicates the source of the interrupt. By copying ICODE2:ICODE0 to WIN2:WIN0, it is possible to select the correct buffer to map into the Access Bank area. See Example 23-2 for a code example.

	ICODE2:ICODE0 Value
No interrupt	`000`
Error interrupt	`001`
TXB2 interrupt	`010`
TXB1 interrupt	`011`
TXB0 interrupt	`100`
RXB1 interrupt	`101`
RXB0 interrupt	`110`
Wake-up interrupt	`111`

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

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

REGISTER 23-2: CANSTAT: CAN STATUS REGISTER (CONTINUED)

bit 4-0 Mode 1,2:

EICODE4:EICODE0: Interrupt Code bits in Mode 1 and Mode 2

When an interrupt occurs, a prioritized coded interrupt value will be present in these bits. This code indicates the source of the interrupt. Unlike ICODE bits in Mode 0, these bits may not be copied directly to EWIN bits to map interrupted buffer to Access Bank area. If required, user software may maintain a table in program memory to map EICODE bits to EWIN bits and access interrupt buffer in Access Bank area.

	EICODE4:EICODE0 Value
No interrupt	00000
Error interrupt	00010
TXB2 interrupt	00100
TXB1 interrupt	00110
TXB0 interrupt	01000
RXB1 interrupt	10001/10000(2)
RXB0 interrupt	10000
Wake-up interrupt	01110
RX/TX B0 interrupt	10010(2)
RX/TX B1 interrupt	10011(2)
RX/TX B2 interrupt	10100(2)
RX/TX B3 interrupt	10101(2)
RX/TX B4 interrupt	10110(2)
RX/TX B4 interrupt	10111(2)

Note 1: To achieve maximum power saving and/or able to wake-up on CAN bus activity, switch CAN module to Disable mode before putting the device to Sleep.

2: In Mode 2, if the buffer is configured as a receiver, EICODE bits will always contain '10000' upon interrupt.

Legend:	U = Unimplemented bit, read as '0'		- n = Value at POR
C = Clearable bit	R = Readable bit	W = Writable bit	x = Bit is unknown
'1' = Bit is set	'0' = Bit is cleared		

EXAMPLE 23-1: CHANGING TO CONFIGURATION MODE

```
    ; Request Configuration mode.
    MOVLW   B'10000000'                 ; Set to Configuration Mode.
    MOVWF   CANCON
    ; A request to switch to Configuration mode may not be immediately honored.
    ; Module will wait for CAN bus to be idle before switching to Configuration Mode.
    ; Request for other modes such as Loopback, Disable etc. may be honored immediately.
    ; It is always good practice to wait and verify before continuing.
ConfigWait:
    MOVF    CANSTAT, W                  ; Read current mode state.
    ANDLW   B'10000000'                 ; Interested in OPMODE bits only.
    TSTFSZ  WREG                        ; Is it Configuration mode yet?
    BRA     ConfigWait                  ; No. Continue to wait...
    ; Module is in Configuration mode now.
    ; Modify configuration registers as required.
    ; Switch back to Normal mode to be able to communicate.
```

EXAMPLE 23-2: WIN AND ICODE BITS USAGE IN INTERRUPT SERVICE ROUTINE TO ACCESS TX/RX BUFFERS

```
    ; Save application required context.
    ; Poll interrupt flags and determine source of interrupt
    ; This was found to be CAN interrupt
    ; TempCANCON and TempCANSTAT are variables defined in Access Bank low
    MOVFF   CANCON, TempCANCON          ; Save CANCON.WIN bits
                                        ; This is required to prevent CANCON
                                        ; from corrupting CAN buffer access
                                        ; in-progress while this interrupt
                                        ; occurred
    MOVFF   CANSTAT, TempCANSTAT        ; Save CANSTAT register
                                        ; This is required to make sure that
                                        ; we use same CANSTAT value rather
                                        ; than one changed by another CAN
                                        ; interrupt.
    MOVF    TempCANSTAT, W              ; Retrieve ICODE bits
    ANDLW   B'00001110'
    ADDWF   PCL, F                      ; Perform computed GOTO
                                        ; to corresponding interrupt cause
    BRA     NoInterrupt                 ; 000 = No interrupt
    BRA     ErrorInterrupt              ; 001 = Error interrupt
    BRA     TXB2Interrupt               ; 010 = TXB2 interrupt
    BRA     TXB1Interrupt               ; 011 = TXB1 interrupt
    BRA     TXB0Interrupt               ; 100 = TXB0 interrupt
    BRA     RXB1Interrupt               ; 101 = RXB1 interrupt
    BRA     RXB0Interrupt               ; 110 = RXB0 interrupt
                                        ; 111 = Wake-up on interrupt
WakeupInterrupt
    BCF     PIR3, WAKIF                 ; Clear the interrupt flag
    ;
    ; User code to handle wake-up procedure
    ;
    ;
    ; Continue checking for other interrupt source or return from here
    …
NoInterrupt
    …                                   ; PC should never vector here. User may
                                        ; place a trap such as infinite loop or pin/port
                                        ; indication to catch this error.
```

EXAMPLE 23-2: WIN AND ICODE BITS USAGE IN INTERRUPT SERVICE ROUTINE TO ACCESS TX/RX BUFFERS (CONTINUED)

```
ErrorInterrupt
    BCF     PIR3, ERRIF                  ; Clear the interrupt flag
    …                                    ; Handle error.
    RETFIE
TXB2Interrupt
    BCF     PIR3, TXB2IF                 ; Clear the interrupt flag
    GOTO    AccessBuffer
TXB1Interrupt
    BCF     PIR3, TXB1IF                 ; Clear the interrupt flag
    GOTO    AccessBuffer
TXB0Interrupt
    BCF     PIR3, TXB0IF                 ; Clear the interrupt flag
    GOTO    AccessBuffer
RXB1Interrupt
    BCF     PIR3, RXB1IF                 ; Clear the interrupt flag
    GOTO    Accessbuffer
RXB0Interrupt
    BCF     PIR3, RXB0IF                 ; Clear the interrupt flag
    GOTO    AccessBuffer
AccessBuffer                             ; This is either TX or RX interrupt
    ; Copy CANSTAT.ICODE bits to CANCON.WIN bits
    MOVF    TempCANCON, W                ; Clear CANCON.WIN bits before copying
                                         ; new ones.
    ANDLW   B'11110001'                  ; Use previously saved CANCON value to
                                         ; make sure same value.
    MOVWF   TempCANCON                   ; Copy masked value back to TempCANCON
    MOVF    TempCANSTAT, W               ; Retrieve ICODE bits
    ANDLW   B'00001110'                  ; Use previously saved CANSTAT value
                                         ; to make sure same value.
    IORWF   TempCANCON                   ; Copy ICODE bits to WIN bits.
    MOVFF   TempCANCON, CANCON           ; Copy the result to actual CANCON
    ; Access current buffer…
    ; User code
    ; Restore CANCON.WIN bits
    MOVF    CANCON, W                    ; Preserve current non WIN bits
    ANDLW   B'11110001'
    IORWF   TempCANCON                   ; Restore original WIN bits
    ; Do not need to restore CANSTAT - it is read-only register.
    ; Return from interrupt or check for another module interrupt source
```


REGISTER 23-3: ECANCON: ENHANCED CAN CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-1	R/W-0	R/W-0	R/W-0	R/W-0
MDSEL1(1, 2)	MDSEL0(1, 2)	FIFOWM	EWIN4	EWIN3	EWIN2	EWIN1	EWIN0
bit 7							bit 0

bit 7-6 **MDSEL1:MDSEL0:** Mode Select bits
`00` = Legacy mode (Mode 0, default)
`01` = Enhanced Legacy mode (Mode 1)
`10` = Enhanced FIFO mode (Mode 2)
`11` = Reserved

bit 5 **FIFOWM:** FIFO High Water Mark bit(3)
`1` = Will cause FIFO interrupt when one receive buffer remains(4)
`0` = Will cause FIFO interrupt when four receive buffers remain

bit 4-0 **EWIN4:EWIN0:** Enhanced Window Address bits
These bits map the group of 16 banked CAN SFRs into access bank addresses 0F60-0F6Dh. Exact group of registers to map is determined by binary value of these bits.

Mode 0:
Unimplemented: Read as '`0`'

Mode 1, 2:
`00000` = Acceptance Filters 0, 1, 2 and BRGCON3, 2
`00001` = Acceptance Filters 3, 4, 5 and BRGCON1, CIOCON
`00010` = Acceptance Filter Masks, Error and Interrupt Control
`00011` = Transmit Buffer 0
`00100` = Transmit Buffer 1
`00101` = Transmit Buffer 2
`00110` = Acceptance Filters 6, 7, 8
`00111` = Acceptance Filters 9, 10, 11
`01000` = Acceptance Filters 12, 13, 14
`01001` = Acceptance Filters 15
`01010-01111` = Reserved
`10000` = Receive Buffer 0
`10001` = Receive Buffer 1
`10010` = TX/RX Buffer 0
`10011` = TX/RX Buffer 1
`10100` = TX/RX Buffer 2
`10101` = TX/RX Buffer 3
`10110` = TX/RX Buffer 4
`10111` = TX/RX Buffer 5
`11000-11111` = Reserved

Note 1: These bits can only be changed in Configuration mode. See Register 19-2 to change to Configuration mode.
2: A new mode takes into effect only after Configuration mode is exited.
3: This bit is used in Mode 2 only.
4: FIFO length of 4 or less will cause this bit to be set.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

REGISTER 23-4: COMSTAT: COMMUNICATION STATUS REGISTER

	R/C-0	R/C-0	R-0	R-0	R-0	R-0	R-0	R-0
Mode 0	RXB0OVFL	RXB1OVFL	TXBO	TXBP	RXBP	TXWARN	RXWARN	EWARN
	U-0	R/C-0	R-0	R-0	R-0	R-0	R-0	R-0
Mode 1	—	RXBnOVFL	TXB0	TXBP	RXBP	TXWARN	RXWARN	EWARN
	R/C-0	R/C-0	R-0	R-0	R-0	R-0	R-0	R-0
Mode 2	$\overline{\text{FIFOEMPTY}}$	RXBnOVFL	TXBO	TXBP	RXBP	TXWARN	RXWARN	EWARN
	bit 7							bit 0

bit 7 Mode 0:
RXB0OVFL: Receive Buffer 0 Overflow bit
1 = Receive Buffer 0 overflowed
0 = Receive Buffer 0 has not overflowed
Mode 1:
Unimplemented: Read as '0'
Mode 2:
$\overline{\text{FIFOEMPTY}}$: FIFO Not Empty bit
1 = Receive FIFO is not empty
0 = Receive FIFO is empty

bit 6 Mode 0:
RXB1OVFL: Receive Buffer 1 Overflow bit
1 = Receive Buffer 1 overflowed
0 = Receive Buffer 1 has not overflowed
Mode 1, 2:
RXBnOVFL: Receive Buffer Overflow bit
1 = Receive buffer has overflowed
0 = Receive buffer has not overflowed

bit 5 **TXBO:** Transmitter Bus-Off bit
1 = Transmit error counter > 255
0 = Transmit error counter ≤ 255

bit 4 **TXBP:** Transmitter Bus Passive bit
1 = Transmit error counter > 127
0 = Transmit error counter ≤ 127

bit 3 **RXBP:** Receiver Bus Passive bit
1 = Receive error counter > 127
0 = Receive error counter ≤ 127

bit 2 **TXWARN:** Transmitter Warning bit
1 = 127 ≥ Transmit error counter > 95
0 = Transmit error counter ≤ 95

bit 1 **RXWARN:** Receiver Warning bit
1 = 127 ≥ Receive error counter > 95
0 = Receive error counter ≤ 95

bit 0 **EWARN:** Error Warning bit
This bit is a flag of the RXWARN and TXWARN bits.
1 = The RXWARN or the TXWARN bits are set
0 = Neither the RXWARN or the TXWARN bits are set

Legend:			
C = Clearable bit	R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

23.2.2 DEDICATED CAN TRANSMIT BUFFER REGISTERS

This section describes the dedicated CAN Transmit Buffer registers and their associated control registers.

REGISTER 23-5: TXBnCON: TRANSMIT BUFFER n CONTROL REGISTERS [0 ≤ n ≤ 2]

	U-0	R-0	R-0	R-0	R/W-0	U-0	R/W-0	R/W-0
Mode 0	—	TXABT	TXLARB	TXERR	TXREQ	—	TXPRI1	TXPRI0
	R/C-0	R-0	R-0	R-0	R/W-0	U-0	R/W-0	R/W-0
Mode 1, 2	TXBIF	TXABT	TXLARB	TXERR	TXREQ	—	TXPRI1	TXPRI0
	bit 7							bit 0

bit 7 Mode 0:
Unimplemented: Read as '0'
Mode 1, 2:
TXBIF: Transmit Buffer Interrupt Flag bit
1 = Transmit buffer has completed transmission of message and may be reloaded
0 = Transmit buffer has not completed transmission of a message

bit 6 **TXABT:** Transmission Aborted Status bit(1)
1 = Message was aborted
0 = Message was not aborted

bit 5 **TXLARB:** Transmission Lost Arbitration Status bit(1)
1 = Message lost arbitration while being sent
0 = Message did not lose arbitration while being sent

bit 4 **TXERR:** Transmission Error Detected Status bit(1)
1 = A bus error occurred while the message was being sent
0 = A bus error did not occur while the message was being sent

bit 3 **TXREQ:** Transmit Request Status bit(2)
1 = Requests sending a message. Clears the TXABT, TXLARB, and TXERR bits.
0 = Automatically cleared when the message is successfully sent

Note: Clearing this bit in software while the bit is set, will request a message abort.

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

bit 1-0 **TXPRI1:TXPRI0:** Transmit Priority bits(3)
11 = Priority Level 3 (highest priority)
10 = Priority Level 2
01 = Priority Level 1
00 = Priority Level 0 (lowest priority)

Note 1: This bit is automatically cleared when TXREQ is set.
2: While TXREQ is set, Transmit Buffer registers remain read-only.
3: These bits define the order in which transmit buffers will be transferred. They do not alter the CAN message identifier.

Legend:	U = Unimplemented bit, read as '0'		- n = Value at POR
C = Clearable bit	R = Readable bit	W = Writable bit	x = Bit is unknown
'1' = Bit is set	'0' = Bit is cleared		

REGISTER 23-6: TXBnSIDH: TRANSMIT BUFFER n STANDARD IDENTIFIER REGISTERS, HIGH BYTE [0 ≤ n ≤ 2]

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3
bit 7							bit 0

bit 7-0 **SID10:SID3:** Standard Identifier bits, if EXIDE (TXBnSIDL<3>) = 0;
Extended Identifier bits EID28:EID21, if EXIDE = 1.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
- n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

REGISTER 23-7: TXBnSIDL: TRANSMIT BUFFER n STANDARD IDENTIFIER REGISTERS, LOW BYTE [0 ≤ n ≤ 2]

R/W-x	R/W-x	R/W-x	U-0	R/W-x	U-0	R/W-x	R/W-x
SID2	SID1	SID0	—	EXIDE	—	EID17	EID16
bit 7							bit 0

bit 7-5 **SID2:SID0:** Standard Identifier bits, if EXIDE (TXBnSIDL<3>) = 0;
Extended Identifier bits EID20:EID18, if EXIDE = 1.

bit 4 **Unimplemented:** Read as ‘0’

bit 3 **EXIDE:** Extended Identifier Enable bit
1 = Message will transmit extended ID, SID10:SID0 becomes EID28:EID18
0 = Message will transmit standard ID, EID17:EID0 are ignored

bit 2 **Unimplemented:** Read as ‘0’

bit 1-0 **EID17:EID16:** Extended Identifier bits

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
- n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

REGISTER 23-8: TXBnEIDH: TRANSMIT BUFFER n EXTENDED IDENTIFIER REGISTERS, HIGH BYTE [0 ≤ n ≤ 2]

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8
bit 7							bit 0

bit 7-0 **EID15:EID8:** Extended Identifier bits (not used when transmitting standard identifier message)

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
- n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

REGISTER 23-9: TXBnEIDL: TRANSMIT BUFFER n EXTENDED IDENTIFIER REGISTERS, LOW BYTE [$0 \leq n \leq 2$]

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0
bit 7							bit 0

bit 7-0 **EID7:EID0:** Extended Identifier bits (not used when transmitting standard identifier message)

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
- n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

REGISTER 23-10: TXBnDm: TRANSMIT BUFFER n DATA FIELD BYTE m REGISTERS [$0 \leq n \leq 2$, $0 \leq m \leq 7$]

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
TXBnDm7	TXBnDm6	TXBnDm5	TXBnDm4	TXBnDm3	TXBnDm2	TXBnDm1	TXBnDm0
bit 7							bit 0

bit 7-0 **TXBnDm7:TXBnDm0:** Transmit Buffer n Data Field Byte m bits (where $0 \leq n < 3$ and $0 \leq m < 8$)

Each transmit buffer has an array of registers. For example, Transmit Buffer 0 has 7 registers: TXB0D0 to TXB0D7.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
- n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

REGISTER 23-11: TXBnDLC: TRANSMIT BUFFER n DATA LENGTH CODE REGISTERS [0 ≤ n ≤ 2]

U-0	R/W-x	U-0	U-0	R/W-x	R/W-x	R/W-x	R/W-x
—	TXRTR	—	—	DLC3	DLC2	DLC1	DLC0
bit 7							bit 0

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

bit 6 **TXRTR:** Transmit Remote Frame Transmission Request bit
1 = Transmitted message will have TXRTR bit set
0 = Transmitted message will have TXRTR bit cleared

bit 5-4 **Unimplemented:** Read as '0'

bit 3-0 **DLC3:DLC0:** Data Length Code bits
1111 = Reserved
1110 = Reserved
1101 = Reserved
1100 = Reserved
1011 = Reserved
1010 = Reserved
1001 = Reserved
1000 = Data length = 8 bytes
0111 = Data length = 7 bytes
0110 = Data length = 6 bytes
0101 = Data length = 5 bytes
0100 = Data length = 4 bytes
0011 = Data length = 3 bytes
0010 = Data length = 2 bytes
0001 = Data length = 1 bytes
0000 = Data length = 0 bytes

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

REGISTER 23-12: TXERRCNT: TRANSMIT ERROR COUNT REGISTER

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
TEC7	TEC6	TEC5	TEC4	TEC3	TEC2	TEC1	TEC0
bit 7							bit 0

bit 7-0 **TEC7:TEC0:** Transmit Error Counter bits
This register contains a value which is derived from the rate at which errors occur. When the error count overflows, the bus-off state occurs. When the bus has 128 occurrences of 11 consecutive recessive bits, the counter value is cleared.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

EXAMPLE 23-3: TRANSMITTING A CAN MESSAGE USING BANKED METHOD

```
; Need to transmit Standard Identifier message 123h using TXB0 buffer.
; To successfully transmit, CAN module must be either in Normal or Loopback mode.
; TXB0 buffer is not in access bank.  And since we want banked method, we need to make sure
; that correct bank is selected.
BANKSEL TXB0CON                        ; One BANKSEL in beginning will make sure that we are
                                       ; in correct bank for rest of the buffer access.
; Now load transmit data into TXB0 buffer.
MOVLW   MY_DATA_BYTE1                  ; Load first data byte into buffer
MOVWF   TXB0D0                         ; Compiler will automatically set "BANKED" bit
; Load rest of data bytes - up to 8 bytes into TXB0 buffer.
...
; Load message identifier
MOVLW   60H                            ; Load SID2:SID0, EXIDE = 0
MOVWF   TXB0SIDL
MOVLW   24H                            ; Load SID10:SID3
MOVWF   TXB0SIDH
; No need to load TXB0EIDL:TXB0EIDH, as we are transmitting Standard Identifier Message only.

; Now that all data bytes are loaded, mark it for transmission.
MOVLW   B'00001000'                    ; Normal priority; Request transmission
MOVWF   TXB0CON

; If required, wait for message to get transmitted
BTFSC   TXB0CON, TXREQ                 ; Is it transmitted?
BRA     $-2                            ; No.  Continue to wait...

; Message is transmitted.
```

EXAMPLE 23-4: TRANSMITTING A CAN MESSAGE USING WIN BITS

```
; Need to transmit Standard Identifier message 123h using TXB0 buffer.
; To successfully transmit, CAN module must be either in Normal or Loopback mode.
; TXB0 buffer is not in access bank.  Use WIN bits to map it to RXB0 area.
MOVF    CANCON, W                    ; WIN bits are in lower 4 bits only.  Read CANCON
                                     ; register to preserve all other bits.  If operation
                                     ; mode is already known, there is no need to preserve
                                     ; other bits.
ANDLW   B'11110000'                  ; Clear WIN bits.
IORLW   B'00001000'                  ; Select Transmit Buffer 0
MOVWF   CANCON                       ; Apply the changes.
; Now TXB0 is mapped in place of RXB0.  All future access to RXB0 registers will actually
; yield TXB0 register values.

; Load transmit data into TXB0 buffer.
MOVLW   MY_DATA_BYTE1                ; Load first data byte into buffer
MOVWF   RXB0D0                       ; Access TXB0D0 via RXB0D0 address.
; Load rest of the data bytes - up to 8 bytes into "TXB0" buffer using RXB0 registers.
...
; Load message identifier
MOVLW   60H                          ; Load SID2:SID0, EXIDE = 0
MOVWF   RXB0SIDL
MOVLW   24H                          ; Load SID10:SID3
MOVWF   RXB0SIDH
; No need to load RXB0EIDL:RXB0EIDH, as we are transmitting Standard Identifier Message only.

; Now that all data bytes are loaded, mark it for transmission.
MOVLW   B'00001000'                  ; Normal priority; Request transmission
MOVWF   RXB0CON

; If required, wait for message to get transmitted
BTFSC   RXB0CON, TXREQ               ; Is it transmitted?
BRA     $-2                          ; No.  Continue to wait...

; Message is transmitted.
; If required, reset the WIN bits to default state.
```

23.2.3 DEDICATED CAN RECEIVE BUFFER REGISTERS

This section shows the dedicated CAN Receive Buffer registers with their associated control registers.

REGISTER 23-13: RXB0CON: RECEIVE BUFFER 0 CONTROL REGISTER

Mode 0	R/C-0	R/W-0	R/W-0	U-0	R-0	R/W-0	R-0	R-0
	RXFUL	RXM1	RXM0	—	RXRTRRO	RXB0DBEN	JTOFF	FILHIT0
Mode 1, 2	R/C-0	R/W-0	R-0	R-0	R-0	R-0	R-0	R-0
	RXFUL	RXM1	RTRRO	FILHIT4	FILHIT3	FILHIT2	FILHIT1	FILHIT0
	bit 7							bit 0

bit 7 **RXFUL:** Receive Full Status bit
1 = Receive buffer contains a received message
0 = Receive buffer is open to receive a new message

Note: This bit is set by the CAN module upon receiving a message and must be cleared by software after the buffer is read. As long as RXFUL is set, no new message will be loaded and buffer will be considered full.

bit 6 Mode 0:
RXM1: Receive Buffer Mode bit 1; combines with RXM0 to form RXM<1:0> bits (see bit 5)
11 = Receive all messages (including those with errors); filter criteria is ignored
10 = Receive only valid messages with extended identifier; EXIDEN in RXFnSIDL must be ‘1’
01 = Receive only valid messages with standard identifier, EXIDEN in RXFnSIDL must be ‘0’
00 = Receive all valid messages as per EXIDEN bit in RXFnSIDL register

Mode 1, 2:
RXM1: Receive Buffer Mode bit
1 = Receive all messages (including those with errors); acceptance filters are ignored
0 = Receive all valid messages as per acceptance filters

bit 5 Mode 0:
RXM0: Receive Buffer Mode bit 0; combines with RXM1 to form RXM<1:0> bits (see bit 6)

Mode 1, 2:
RTRRO: Remote Transmission Request bit for Received Message (read-only)
1 = A remote transmission request is received
0 = A remote transmission request is not received

bit 4 Mode 0:
Unimplemented: Read as ‘0’

Mode 1, 2:
FILHIT4: Filter Hit bit 4
This bit combines with other bits to form filter acceptance bits <4:0>.

bit 3 Mode 0:
RXRTRRO: Remote Transmission Request bit for Received Message (read-only)
1 = A remote transmission request is received
0 = A remote transmission request is not received

Mode 1, 2:
FILHIT3: Filter Hit bit 3
This bit combines with other bits to form filter acceptance bits <4:0>.

Legend:	U = Unimplemented bit, read as ‘0’		- n = Value at POR
C = Clearable bit	R = Readable bit	W = Writable bit	x = Bit is unknown
‘1’ = Bit is set	‘0’ = Bit is cleared		

REGISTER 23-13: RXB0CON: RECEIVE BUFFER 0 CONTROL REGISTER (CONTINUED)

bit 2 Mode 0:

RXB0DBEN: Receive Buffer 0 Double-Buffer Enable bit

1 = Receive Buffer 0 overflow will write to Receive Buffer 1
0 = No Receive Buffer 0 overflow to Receive Buffer 1

Mode 1, 2:

FILHIT2: Filter Hit bit 2
This bit combines with other bits to form filter acceptance bits <4:0>.

bit 1 Mode 0:

JTOFF: Jump Table Offset bit (read-only copy of RXB0DBEN)

1 = Allows jump table offset between 6 and 7
0 = Allows jump table offset between 1 and 0

Note: This bit allows same filter jump table for both RXB0CON and RXB1CON.

Mode 1, 2:

FILHIT1: Filter Hit bit 1

This bit combines with other bits to form filter acceptance bits <4:0>.

bit 0 Mode 0:

FILHIT0: Filter Hit bit 0

This bit indicates which acceptance filter enabled the message reception into Receive Buffer 0.
1 = Acceptance Filter 1 (RXF1)
0 = Acceptance Filter 0 (RXF0)

Mode 1, 2:

FILHIT0: Filter Hit bit 0

This bit, in combination with FILHIT<4:1>, indicates which acceptance filter enabled the message reception into this receive buffer.
01111 = Acceptance Filter 15 (RXF15)
01110 = Acceptance Filter 14 (RXF14)
...
00000 = Acceptance Filter 0 (RXF0)

Legend:	U = Unimplemented bit, read as ‘0’		- n = Value at POR
C = Clearable bit	R = Readable bit	W = Writable bit	x = Bit is unknown
‘1’ = Bit is set	‘0’ = Bit is cleared		

REGISTER 23-14: RXB1CON: RECEIVE BUFFER 1 CONTROL REGISTER

	R/C-0	R/W-0	R/W-0	U-0	R-0	R/W-0	R-0	R-0
Mode 0	RXFUL	RXM1	RXM0	—	RXRTRRO	FILHIT2	FILHIT1	FILHIT0

	R/C-0	R/W-0	R-0	R-0	R-0	R-0	R-0	R-0
Mode 1, 2	RXFUL	RXM1	RTRRO	FILHIT4	FILHIT3	FILHIT2	FILHIT1	FILHIT0
	bit 7							bit 0

bit 7 **RXFUL:** Receive Full Status bit
1 = Receive buffer contains a received message
0 = Receive buffer is open to receive a new message

Note: This bit is set by the CAN module upon receiving a message and must be cleared by software after the buffer is read. As long as RXFUL is set, no new message will be loaded and buffer will be considered full.

bit 6 Mode 0:
RXM1: Receive Buffer Mode bit 1; combines with RXM0 to form RXM<1:0> bits (see bit 5)
11 = Receive all messages (including those with errors); filter criteria is ignored
10 = Receive only valid messages with extended identifier; EXIDEN in RXFnSIDL must be '1'
01 = Receive only valid messages with standard identifier, EXIDEN in RXFnSIDL must be '0'
00 = Receive all valid messages as per EXIDEN bit in RXFnSIDL register

Mode 1, 2:
RXM1: Receive Buffer Mode bit
1 = Receive all messages (including those with errors); acceptance filters are ignored
0 = Receive all valid messages as per acceptance filters

bit 5 Mode 0:
RXM0: Receive Buffer Mode bit 0; combines with RXM1 to form RXM<1:0> bits (see bit 6)

Mode 1, 2:
RTRRO: Remote Transmission Request bit for Received Message (read-only)
1 = A remote transmission request is received
0 = A remote transmission request is not received

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

Mode 1, 2:
FILHIT4: Filter Hit bit 4
This bit combines with other bits to form filter acceptance bits <4:0>.

bit 3 Mode 0:
RXRTRRO: Remote Transmission Request bit for Received Message (read-only)
1 = A remote transmission request is received
0 = A remote transmission request is not received

Mode 1, 2:
FILHIT3: Filter Hit bit 3
This bit combines with other bits to form filter acceptance bits <4:0>.

bit 2-0 Mode 0:
FILHIT2:FILHIT0: Filter Hit bits
These bits indicate which acceptance filter enabled the last message reception into Receive Buffer 1.
111 = Reserved
110 = Reserved
101 = Acceptance Filter 5 (RXF5)
100 = Acceptance Filter 4 (RXF4)
011 = Acceptance Filter 3 (RXF3)
010 = Acceptance Filter 2 (RXF2)
001 = Acceptance Filter 1 (RXF1), only possible when RXB0DBEN bit is set
000 = Acceptance Filter 0 (RXF0), only possible when RXB0DBEN bit is set

Mode 1, 2:
FILHIT2:FILHIT0 Filter Hit bits <2:0>
These bits, in combination with FILHIT<4:3>, indicate which acceptance filter enabled the message reception into this receive buffer.
01111 = Acceptance Filter 15 (RXF15)
01110 = Acceptance Filter 14 (RXF14)
...
00000 = Acceptance Filter 0 (RXF0)

Legend:	U = Unimplemented bit, read as '0'		- n = Value at POR
C = Clearable bit	R = Readable bit	W = Writable bit	x = Bit is unknown
'1' = Bit is set	'0' = Bit is cleared		

REGISTER 23-15: RXBnSIDH: RECEIVE BUFFER n STANDARD IDENTIFIER REGISTERS, HIGH BYTE [0 ≤ n ≤ 1]

R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x
SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3
bit 7							bit 0

bit 7-0 **SID10:SID3:** Standard Identifier bits, if EXID = 0 (RXBnSIDL<3>);
Extended Identifier bits EID28:EID21, if EXID = 1.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
- n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

REGISTER 23-16: RXBnSIDL: RECEIVE BUFFER n STANDARD IDENTIFIER REGISTERS, LOW BYTE [0 ≤ n ≤ 1]

R-x	R-x	R-x	R-x	R-x	U-0	R-x	R-x
SID2	SID1	SID0	SRR	EXID	—	EID17	EID16
bit 7							bit 0

bit 7-5 **SID2:SID0:** Standard Identifier bits, if EXID = 0;
Extended Identifier bits EID20:EID18, if EXID = 1.

bit 4 **SRR:** Substitute Remote Request bit
This bit is always ‘0’ when EXID = 1 or equal to the value of RXRTRRO (RBXnCON<3>) when EXID = 0.

bit 3 **EXID:** Extended Identifier bit
1 = Received message is an extended data frame, SID10:SID0 are EID28:EID18
0 = Received message is a standard data frame

bit 2 **Unimplemented:** Read as ‘0’

bit 1-0 **EID17:EID16:** Extended Identifier bits

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
- n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

REGISTER 23-17: RXBnEIDH: RECEIVE BUFFER n EXTENDED IDENTIFIER REGISTERS, HIGH BYTE [0 ≤ n ≤ 1]

R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x
EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8
bit 7							bit 0

bit 7-0 **EID15:EID8:** Extended Identifier bits

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
- n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

REGISTER 23-18: RXBnEIDL: RECEIVE BUFFER n EXTENDED IDENTIFIER REGISTERS, LOW BYTE [0 ≤ n ≤ 1]

R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x
EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0
bit 7							bit 0

bit 7-0 **EID7:EID0:** Extended Identifier bits

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
- n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

REGISTER 23-19: RXBnDLC: RECEIVE BUFFER n DATA LENGTH CODE REGISTERS [0 ≤ n ≤ 1]

U-0	R-x	R-x	R-x	R-x	R-x	R-x	R-x
—	RXRTR	RB1	RB0	DLC3	DLC2	DLC1	DLC0
bit 7							bit 0

bit 7 **Unimplemented:** Read as ‘0’

bit 6 **RXRTR:** Receiver Remote Transmission Request bit
1 = Remote transfer request
0 = No remote transfer request

bit 5 **RB1:** Reserved bit 1
Reserved by CAN Spec and read as ‘0’.

bit 4 **RB0:** Reserved bit 0
Reserved by CAN Spec and read as ‘0’.

bit 3-0 **DLC3:DLC0:** Data Length Code bits
1111 = Invalid
1110 = Invalid
1101 = Invalid
1100 = Invalid
1011 = Invalid
1010 = Invalid
1001 = Invalid
1000 = Data length = 8 bytes
0111 = Data length = 7 bytes
0110 = Data length = 6 bytes
0101 = Data length = 5 bytes
0100 = Data length = 4 bytes
0011 = Data length = 3 bytes
0010 = Data length = 2 bytes
0001 = Data length = 1 bytes
0000 = Data length = 0 bytes

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
- n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

REGISTER 23-20: RXBnDm: RECEIVE BUFFER n DATA FIELD BYTE m REGISTERS [0 ≤ n ≤ 1, 0 ≤ m ≤ 7]

R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x
RXBnDm7	RXBnDm6	RXBnDm5	RXBnDm4	RXBnDm3	RXBnDm2	RXBnDm1	RXBnDm0
bit 7							bit 0

bit 7-0 **RXBnDm7:RXBnDm0:** Receive Buffer n Data Field Byte m bits (where $0 \le n < 1$ and $0 < m < 7$)
Each receive buffer has an array of registers. For example, Receive Buffer 0 has 8 registers: RXB0D0 to RXB0D7.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
- n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

REGISTER 23-21: RXERRCNT: RECEIVE ERROR COUNT REGISTER

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
REC7	REC6	REC5	REC4	REC3	REC2	REC1	REC0
bit 7							bit 0

bit 7-0 **REC7:REC0:** Receive Error Counter bits
This register contains the receive error value as defined by the CAN specifications.
When RXERRCNT > 127, the module will go into an error-passive state. RXERRCNT does not have the ability to put the module in “bus-off” state.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
- n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

EXAMPLE 23-5: READING A CAN MESSAGE

```
; Need to read a pending message from RXB0 buffer.
; To receive any message, filter, mask and RXM1:RXM0 bits in RXB0CON registers must be
; programmed correctly.
;
; Make sure that there is a message pending in RXB0.
BTFSS   RXB0CON, RXFUL                ; Does RXB0 contain a message?
BRA     NoMessage                     ; No.  Handle this situation...
; We have verified that a message is pending in RXB0 buffer.
; If this buffer can receive both Standard or Extended Identifier messages,
; identify type of message received.
BTFSS   RXB0SIDL, EXID                ; Is this Extended Identifier?
BRA     StandardMessage               ; No.  This is Standard Identifier message.
                                      ; Yes.  This is Extended Identifier message.
; Read all 29-bits of Extended Identifier message.
...
; Now read all data bytes
MOVFF   RXB0DO, MY_DATA_BYTE1
...
; Once entire message is read, mark the RXB0 that it is read and no longer FULL.
BCF     RXB0CON, RXFUL                ; This will allow CAN Module to load new messages
                                      ; into this buffer.
...
```


23.2.3.1 Programmable TX/RX and Auto RTR Buffers

The ECAN module contains 6 message buffers that can be programmed as transmit or receive buffers. Any of these buffers can also be programmed to automatically handle RTR messages.

Note:	These registers are not used in Mode 0.

REGISTER 23-22: BnCON: TX/RX BUFFER n CONTROL REGISTERS IN RECEIVE MODE [0 ≤ n ≤ 5, TXnEN (BSEL0<n>) = 0][(1)]

R/C-0	R/W-0	R-0	R-0	R-0	R-0	R-0	R-0
RXFUL	RXM1	RTRRO	FILHIT4	FILHIT3	FILHIT2	FILHIT1	FILHIT0
bit 7							bit 0

bit 7 **RXFUL:** Receive Full Status bit[(1)]
1 = Receive buffer contains a received message
0 = Receive buffer is open to receive a new message

Note: This bit is set by the CAN module upon receiving a message and must be cleared by software after the buffer is read. As long as RXFUL is set, no new message will be loaded and buffer will be considered full.

bit 6 **RXM1:** Receive Buffer Mode bit
1 = Receive all messages including partial and invalid (acceptance filters are ignored)
0 = Receive all valid messages as per acceptance filters

bit 5 **RTRRO:** Read-Only Remote Transmission Request bit for Received Message
1 = Received message is a remote transmission request
0 = Received message is not a remote transmission request

bit 4-0 **FILHIT4:FILHIT0:** Filter Hit bits
These bits indicate which acceptance filter enabled the last message reception into this buffer.
01111 = Acceptance Filter 15 (RXF15)
01110 = Acceptance Filter 14 (RXF14)
...
00001 = Acceptance Filter 1 (RXF1)
00000 = Acceptance Filter 0 (RXF0)

Note 1: These registers are available in Mode 1 and 2 only.

Legend:	U = Unimplemented bit, read as ‘0’		- n = Value at POR
C = Clearable bit	R = Readable bit	W = Writable bit	x = Bit is unknown
‘1’ = Bit is set	‘0’ = Bit is cleared		

REGISTER 23-23: BnCON: TX/RX BUFFER n CONTROL REGISTERS IN TRANSMIT MODE [0 ≤ n ≤ 5, TXnEN (BSEL0<n>) = 1][(1)]

R/W-0	R-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
TXBIF	TXABT	TXLARB	TXERR	TXREQ	RTREN	TXPRI1	TXPRI0
bit 7							bit 0

bit 7 **TXBIF:** Transmit Buffer Interrupt Flag bit[(1)]
1 = A message is successfully transmitted
0 = No message was transmitted

bit 6 **TXABT:** Transmission Aborted Status bit[(1)]
1 = Message was aborted
0 = Message was not aborted

bit 5 **TXLARB:** Transmission Lost Arbitration Status bit[(2)]
1 = Message lost arbitration while being sent
0 = Message did not lose arbitration while being sent

bit 4 **TXERR:** Transmission Error Detected Status bit[(2)]
1 = A bus error occurred while the message was being sent
0 = A bus error did not occur while the message was being sent

bit 3 **TXREQ:** Transmit Request Status bit[(3)]
1 = Requests sending a message; clears the TXABT, TXLARB, and TXERR bits
0 = Automatically cleared when the message is successfully sent

Note: Clearing this bit in software while the bit is set will request a message abort.

bit 2 **RTREN:** Automatic Remote Transmission Request Enable bit
1 = When a remote transmission request is received, TXREQ will be automatically set
0 = When a remote transmission request is received, TXREQ will be unaffected

bit 1-0 **TXPRI1:TXPRI0:** Transmit Priority bits[(4)]
11 = Priority Level 3 (highest priority)
10 = Priority Level 2
01 = Priority Level 1
00 = Priority Level 0 (lowest priority)

Note 1: These registers are available in Mode 1 and 2 only.
2: This bit is automatically cleared when TXREQ is set.
3: While TXREQ is set or transmission is in progress, transmit buffer registers remain read-only.
4: These bits set the order in which the transmit buffer will be transferred. They do not alter the CAN message identifier.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
- n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

REGISTER 23-24: BnSIDH: TX/RX BUFFER n STANDARD IDENTIFIER REGISTERS, HIGH BYTE IN RECEIVE MODE [0 ≤ n ≤ 5, TXnEN (BSEL0<n>) = 0][(1)]

R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x
SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3
bit 7							bit 0

bit 7-0 **SID10:SID3:** Standard Identifier bits, if EXIDE (BnSIDL<3>) = 0;
Extended Identifier bits EID28:EID21, if EXIDE = 1.

Note 1: These registers are available in Mode 1 and 2 only.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

REGISTER 23-25: BnSIDH: TX/RX BUFFER n STANDARD IDENTIFIER REGISTERS, HIGH BYTE IN TRANSMIT MODE [0 ≤ n ≤ 5, TXnEN (BSEL0<n>) = 1][(1)]

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3
bit 7							bit 0

bit 7-0 **SID10:SID3:** Standard Identifier bits, if EXIDE (BnSIDL<3>) = 0;
Extended Identifier bits EID28:EID21, if EXIDE = 1.

Note 1: These registers are available in Mode 1 and 2 only.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

REGISTER 23-26: BnSIDL: TX/RX BUFFER n STANDARD IDENTIFIER REGISTERS, LOW BYTE IN RECEIVE MODE [0 ≤ n ≤ 5, TXnEN (BSEL0<n>) = 0][(1)]

R-x	R-x	R-x	R-x	R-x	U-0	R-x	R-x
SID2	SID1	SID0	SRR	EXID	—	EID17	EID16
bit 7							bit 0

bit 7-5 **SID2:SID0:** Standard Identifier bits, if EXID = 0;
Extended Identifier bits EID20:EID18, if EXID = 1.

bit 4 **SRR:** Substitute Remote Transmission Request bit (only when EXID = 1)
1 = Remote transmission request occurred
0 = No remote transmission request occurred

bit 3 **EXID:** Extended Identifier Enable bit
1 = Received message is an extended identifier frame, SID10:SID0 are EID28:EID18
0 = Received message is a standard identifier frame

bit 2 **Unimplemented:** Read as ‘0’

bit 1-0 **EID17:EID16:** Extended Identifier bits

Note 1: These registers are available in Mode 1 and 2 only.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
- n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

REGISTER 23-27: BnSIDL: TX/RX BUFFER n STANDARD IDENTIFIER REGISTERS, LOW BYTE IN TRANSMIT MODE [0 ≤ n ≤ 5, TXnEN (BSEL0<n>) = 1][(1)]

R/W-x	R/W-x	R/W-x	U-0	R/W-x	U-0	R/W-x	R/W-x
SID2	SID1	SID0	—	EXIDE	—	EID17	EID16
bit 7							bit 0

bit 7-5 **SID2:SID0:** Standard Identifier bits, if EXIDE = 0;
Extended Identifier bits EID20:EID18, if EXIDE = 1.

bit 4 **Unimplemented:** Read as ‘0’

bit 3 **EXIDE:** Extended Identifier Enable bit
1 = Received message is an extended identifier frame, SID10:SID0 are EID28:EID18
0 = Received message is a standard identifier frame

bit 2 **Unimplemented:** Read as ‘0’

bit 1-0 **EID17:EID16:** Extended Identifier bits

Note 1: These registers are available in Mode 1 and 2 only.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
- n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

REGISTER 23-28: BnEIDH: TX/RX BUFFER n EXTENDED IDENTIFIER REGISTERS, HIGH BYTE IN RECEIVE MODE [0 ≤ n ≤ 5, TXnEN (BSEL0<n>) = 0](1)

R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x
EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8
bit 7							bit 0

bit 7-0 **EID15:EID8:** Extended Identifier bits

Note 1: These registers are available in Mode 1 and 2 only.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
- n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

REGISTER 23-29: BnEIDH: TX/RX BUFFER n EXTENDED IDENTIFIER REGISTERS, HIGH BYTE IN TRANSMIT MODE [0 ≤ n ≤ 5, TXnEN (BSEL0<n>) = 1] (1)

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8
bit 7							bit 0

bit 7-0 **EID15:EID8:** Extended Identifier bits

Note 1: These registers are available in Mode 1 and 2 only.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
- n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

REGISTER 23-30: BnEIDL: TX/RX BUFFER n EXTENDED IDENTIFIER REGISTERS, LOW BYTE IN RECEIVE MODE [$0 \leq n \leq 5$, TXnEN (BSEL<n>) = 0][(1)]

R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x
EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0
bit 7							bit 0

bit 7-0 **EID7:EID0:** Extended Identifier bits

Note 1: These registers are available in Mode 1 and 2 only.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
- n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

REGISTER 23-31: BnEIDL: TX/RX BUFFER n EXTENDED IDENTIFIER REGISTERS, LOW BYTE IN TRANSMIT MODE [$0 \leq n \leq 5$, TXnEN (BSEL<n>) = 1][(1)]

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0
bit 7							bit 0

bit 7-0 **EID7:EID0:** Extended Identifier bits

Note 1: These registers are available in Mode 1 and 2 only.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
- n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

REGISTER 23-32: BnDm: TX/RX BUFFER n DATA FIELD BYTE m REGISTERS IN RECEIVE MODE [$0 \leq n \leq 5$, $0 \leq m \leq 7$, TXnEN (BSEL<n>) = 0][(1)]

R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x
BnDm7	BnDm6	BnDm5	BnDm4	BnDm3	BnDm2	BnDm1	BnDm0
bit 7							bit 0

bit 7-0 **BnDm7:BnDm0:** Receive Buffer n Data Field Byte m bits (where $0 \leq n < 3$ and $0 < m < 8$)
Each receive buffer has an array of registers. For example, Receive Buffer 0 has 7 registers: B0D0 to B0D7.

Note 1: These registers are available in Mode 1 and 2 only.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

REGISTER 23-33: BnDm: TX/RX BUFFER n DATA FIELD BYTE m REGISTERS IN TRANSMIT MODE [$0 \leq n \leq 5$, $0 \leq m \leq 7$, TXnEN (BSEL<n>) = 1][(1)]

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
BnDm7	BnDm6	BnDm5	BnDm4	BnDm3	BnDm2	BnDm1	BnDm0
bit 7							bit 0

bit 7-0 **BnDm7:BnDm0:** Transmit Buffer n Data Field Byte m bits (where $0 \leq n < 3$ and $0 < m < 8$)
Each transmit buffer has an array of registers. For example, Transmit Buffer 0 has 7 registers: TXB0D0 to TXB0D7.

Note 1: These registers are available in Mode 1 and 2 only.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

REGISTER 23-34: BnDLC: TX/RX BUFFER n DATA LENGTH CODE REGISTERS IN RECEIVE MODE [0 ≤ n ≤ 5, TXnEN (BSEL<n>) = 0][(1)]

U-0	R-x	R-x	R-x	R-x	R-x	R-x	R-x
—	RXRTR	RB1	RB0	DLC3	DLC2	DLC1	DLC0
bit 7							bit 0

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

bit 6 **RXRTR:** Receiver Remote Transmission Request bit
1 = This is a remote transmission request
0 = This is not a remote transmission request

bit 5 **RB1:** Reserved bit 1
Reserved by CAN Spec and read as '0'.

bit 4 **RB0:** Reserved bit 0
Reserved by CAN Spec and read as '0'.

bit 3-0 **DLC3:DLC0:** Data Length Code bits
1111 = Reserved
1110 = Reserved
1101 = Reserved
1100 = Reserved
1011 = Reserved
1010 = Reserved
1001 = Reserved
1000 = Data length = 8 bytes
0111 = Data length = 7 bytes
0110 = Data length = 6 bytes
0101 = Data length = 5 bytes
0100 = Data length = 4 bytes
0011 = Data length = 3 bytes
0010 = Data length = 2 bytes
0001 = Data length = 1 bytes
0000 = Data length = 0 bytes

Note 1: These registers are available in Mode 1 and 2 only.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

REGISTER 23-35: BnDLC: TX/RX BUFFER n DATA LENGTH CODE REGISTERS IN TRANSMIT MODE [0 ≤ n ≤ 5, TXnEN (BSEL<n>) = 1][(1)]

U-0	R/W-x	U-0	U-0	R/W-x	R/W-x	R/W-x	R/W-x
—	TXRTR	—	—	DLC3	DLC2	DLC1	DLC0
bit 7							bit 0

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

bit 6 **TXRTR:** Transmitter Remote Transmission Request bit
1 = Transmitted message will have RTR bit set
0 = Transmitted message will have RTR bit cleared

bit 5-4 **Unimplemented:** Read as '0'

bit 3-0 **DLC3:DLC0:** Data Length Code bits
1111-1001 = Reserved
1000 = Data length = 8 bytes
0111 = Data length = 7 bytes
0110 = Data length = 6 bytes
0101 = Data length = 5 bytes
0100 = Data length = 4 bytes
0011 = Data length = 3 bytes
0010 = Data length = 2 bytes
0001 = Data length = 1 bytes
0000 = Data length = 0 bytes

Note 1: These registers are available in Mode 1 and 2 only.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

REGISTER 23-36: BSEL0: BUFFER SELECT REGISTER 0[(1)]

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0	U-0
B5TXEN	B4TXEN	B3TXEN	B2TXEN	B1TXEN	B0TXEN	—	—
bit 7							bit 0

bit 7-2 **B5TXEN:B0TXEN:** Buffer 5 to Buffer 0 Transmit Enable bit
1 = Buffer is configured in Transmit mode
0 = Buffer is configured in Receive mode

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

Note 1: This register is available in Mode 1 and 2 only.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

23.2.3.2 Message Acceptance Filters and Masks

This subsection describes the message acceptance filters and masks for the CAN receive buffers.

Note:	These registers are writable in Configuration mode only.

REGISTER 23-37: RXFnSIDH: RECEIVE ACCEPTANCE FILTER n STANDARD IDENTIFIER FILTER REGISTERS, HIGH BYTE [0 ≤ n ≤ 15](1)

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3
bit 7							bit 0

bit 7-0 **SID10:SID3:** Standard Identifier Filter bits, if EXIDEN = 0;
Extended Identifier Filter bits EID28:EID21, if EXIDEN = 1.

Note 1: Registers RXF6SIDH:RXF15SIDH are available in Mode 1 and 2 only.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

REGISTER 23-38: RXFnSIDL: RECEIVE ACCEPTANCE FILTER n STANDARD IDENTIFIER FILTER REGISTERS, LOW BYTE [0 ≤ n ≤ 15](1)

R/W-x	R/W-x	R/W-x	U-0	R/W-x	U-0	R/W-x	R/W-x
SID2	SID1	SID0	—	EXIDEN	—	EID17	EID16
bit 7							bit 0

bit 7-5 **SID2:SID0:** Standard Identifier Filter bits, if EXIDEN = 0;
Extended Identifier Filter bits EID20:EID18, if EXIDEN = 1.

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

bit 3 **EXIDEN:** Extended Identifier Filter Enable bit
1 = Filter will only accept extended ID messages
0 = Filter will only accept standard ID messages

Note: In Mode 0, this bit must be set/cleared as required, irrespective of corresponding mask register value.

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

bit 1-0 **EID17:EID16:** Extended Identifier Filter bits

Note 1: Registers RXF6SIDL:RXF15SIDL are available in Mode 1 and 2 only.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

REGISTER 23-39: RXFnEIDH: RECEIVE ACCEPTANCE FILTER n EXTENDED IDENTIFIER REGISTERS, HIGH BYTE [0 ≤ n ≤ 15][(1)]

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8
bit 7							bit 0

bit 7-0 **EID15:EID8:** Extended Identifier Filter bits

Note 1: Registers RXF6EIDH:RXF15EIDH are available in Mode 1 and 2 only.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

REGISTER 23-40: RXFnEIDL: RECEIVE ACCEPTANCE FILTER n EXTENDED IDENTIFIER REGISTERS, LOW BYTE [0 ≤ n ≤ 15][(1)]

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0
bit 7							bit 0

bit 7-0 **EID7:EID0:** Extended Identifier Filter bits

Note 1: Registers RXF6EIDL:RXF15EIDL are available in Mode 1 and 2 only.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

REGISTER 23-41: RXMnSIDH: RECEIVE ACCEPTANCE MASK n STANDARD IDENTIFIER MASK REGISTERS, HIGH BYTE [0 ≤ n ≤ 1]

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3
bit 7							bit 0

bit 7-0 **SID10:SID3:** Standard Identifier Mask bits, or Extended Identifier Mask bits EID28:EID21

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

REGISTER 23-42: RXMnSIDL: RECEIVE ACCEPTANCE MASK n STANDARD IDENTIFIER MASK REGISTERS, LOW BYTE [0 ≤ n ≤ 1]

R/W-x	R/W-x	R/W-x	U-0	R/W-0	U-0	R/W-x	R/W-x
SID2	SID1	SID0	—	EXIDEN(1)	—	EID17	EID16
bit 7							bit 0

bit 7-5 **SID2:SID0:** Standard Identifier Mask bits, or Extended Identifier Mask bits EID20:EID18

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

bit 3 Mode 0:

Unimplemented: Read as '0'

Mode 1, 2:

EXIDEN: Extended Identifier Filter Enable Mask bit(1)

1 = Messages selected by EXIDEN bit in RXFnSIDL will be accepted
0 = Both standard and extended identifier messages will be accepted

Note 1: This bit is available in Mode 1 and 2 only.

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

bit 1-0 **EID17:EID16:** Extended Identifier Mask bits

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

REGISTER 23-43: RXMnEIDH: RECEIVE ACCEPTANCE MASK n EXTENDED IDENTIFIER MASK REGISTERS, HIGH BYTE [0 ≤ n ≤ 1]

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8
bit 7							bit 0

bit 7-0 **EID15:EID8:** Extended Identifier Mask bits

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

REGISTER 23-44: RXMnEIDL: RECEIVE ACCEPTANCE MASK n EXTENDED IDENTIFIER MASK REGISTERS, LOW BYTE [0 ≤ n ≤ 1]

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
EID7	EID6	EID5	EID4	EID3	EID2	EID1	EID0
bit 7							bit 0

bit 7-0 **EID7:EID0:** Extended Identifier Mask bits

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

REGISTER 23-45: SDFLC: STANDARD DATA BYTES FILTER LENGTH COUNT REGISTER(1)

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	FLC4	FLC3	FLC2	FLC1	FLC0
bit 7							bit 0

bit 7-5 **Unimplemented:** Read as '0'

bit 4-0 **FLC4:FLC0:** Filter Length Count bits

Mode 0:

Not used; forced to '00000'.

Mode 1, 2:

00000-10010 = 0 18 bits are available for standard data byte filter. Actual number of bits used depends on DLC3:DLC0 bits (RXBnDLC<3:0> or BnDLC<3:0> if configured as RX buffer) of message being received.

If DLC3:DLC0 = 0000 No bits will be compared with incoming data bits

If DLC3:DLC0 = 0001 Up to 8 data bits of RXFnEID<7:0>, as determined by FLC2:FLC0, will be compared with the corresponding number of data bits of the incoming message

If DLC3:DLC0 = 0010 Up to 16 data bits of RXFnEID<15:0>, as determined by FLC3:FLC0, will be compared with the corresponding number of data bits of the incoming message

If DLC3:DLC0 = 0011 Up to 18 data bits of RXFnEID<17:0>, as determined by FLC4:FLC0, will be compared with the corresponding number of data bits of the incoming message

Note 1: This register is available in Mode 1 and 2 only.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

REGISTER 23-46: RXFCONn: RECEIVE FILTER CONTROL REGISTER n [0 ≤ n ≤ 1](1)

RXFCON0

R/W-0	R/W-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
RXF7EN	RXF6EN	RXF5EN	RXF4EN	RXF3EN	RXF2EN	RXF1EN	RXF0EN

RXFCON1

R/W-0	R/W-0	R/W-0	R/W-1	R/W-0	R/W-0	R/W-0	R/W-0
RXF15EN	RXF14EN	RXF13EN	RXF12EN	RXF11EN	RXF10EN	RXF9EN	RXF8EN
bit 7							bit 0

bit 7-0 **RXFnEN:** Receive Filter n Enable bit

0 = Filter is disabled

1 = Filter is enabled

Note 1: This register is available in Mode 1 and 2 only.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

REGISTER 23-47: RXFBCONn: RECEIVE FILTER BUFFER CONTROL REGISTER n(1)

	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
RXFBCON0	F1BP_3	F1BP_2	F1BP_1	F1BP_0	F0BP_3	F0BP_2	F0BP_1	F0BP_0
	R/W-0	R/W-0	R/W-0	R/W-1	R/W-0	R/W-0	R/W-0	R/W-1
RXFBCON1	F3BP_3	F3BP_2	F3BP_1	F3BP_0	F2BP_3	F2BP_2	F2BP_1	F2BP_0
	R/W-0	R/W-0	R/W-0	R/W-1	R/W-0	R/W-0	R/W-0	R/W-1
RXFBCON2	F5BP_3	F5BP_2	F5BP_1	F5BP_0	F4BP_3	F4BP_2	F4BP_1	F4BP_0
	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
RXFBCON3	F7BP_3	F7BP_2	F7BP_1	F7BP_0	F6BP_3	F6BP_2	F6BP_1	F6BP_0
	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
RXFBCON4	F9BP_3	F9BP_2	F9BP_1	F9BP_0	F8BP_3	F8BP_2	F8BP_1	F8BP_0
	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
RXFBCON5	F11BP_3	F11BP_2	F11BP_1	F11BP_0	F10BP_3	F10BP_2	F10BP_1	F10BP_0
	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
RXFBCON6	F13BP_3	F13BP_2	F13BP_1	F13BP_0	F12BP_3	F12BP_2	F12BP_1	F12BP_0
	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
RXFBCON7	F15BP_3	F15BP_2	F15BP_1	F15BP_0	F14BP_3	F14BP_2	F14BP_1	F14BP_0
	bit 7							bit 0

bit 7-0 **FnBP_3:FnBP_0:** Filter n Buffer Pointer Nibble bits
`0000` = Filter n is associated with RXB0
`0001` = Filter n is associated with RXB1
`0010` = Filter n is associated with B0
`0011` = Filter n is associated with B1
.
.
.
`0111` = Filter n is associated with B5
`1111:1000` = Reserved

Note 1: This register is available in Mode 1 and 2 only.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
- n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

REGISTER 23-48: MSEL0: MASK SELECT REGISTER 0(1)

R/W-0	R/W-1	R/W-0	R/W-1	R/W-0	R/W-0	R/W-0	R/W-0
FIL3_1	FIL3_0	FIL2_1	FIL2_0	FIL1_1	FIL1_0	FIL0_1	FIL0_0
bit 7							bit 0

bit 7-6 **FIL3_1:FIL3_0:** Filter 3 Select bits 1 and 0
11 = No mask
10 = Filter 15
01 = Acceptance Mask 1
00 = Acceptance Mask 0

bit 5-4 **FIL2_1:FIL2_0:** Filter 2 Select bits 1 and 0
11 = No mask
10 = Filter 15
01 = Acceptance Mask 1
00 = Acceptance Mask 0

bit 3-2 **FIL1_1:FIL1_0:** Filter 1 Select bits 1 and 0
11 = No mask
10 = Filter 15
01 = Acceptance Mask 1
00 = Acceptance Mask 0

bit 1-0 **FIL0_1:FIL0_0:** Filter 0 Select bits 1 and 0
11 = No mask
10 = Filter 15
01 = Acceptance Mask 1
00 = Acceptance Mask 0

Note 1: This register is available in Mode 1 and 2 only.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
- n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

REGISTER 23-49: MSEL1: MASK SELECT REGISTER 1(1)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-1	R/W-0	R/W-1
FIL7_1	FIL7_0	FIL6_1	FIL6_0	FIL5_1	FIL5_0	FIL4_1	FIL4_0
bit 7							bit 0

bit 7-6 **FIL7_1:FIL7_0:** Filter 7 Select bits 1 and 0
`11` = No mask
`10` = Filter 15
`01` = Acceptance Mask 1
`00` = Acceptance Mask 0

bit 5-4 **FIL6_1:FIL6_0:** Filter 6 Select bits 1 and 0
`11` = No mask
`10` = Filter 15
`01` = Acceptance Mask 1
`00` = Acceptance Mask 0

bit 3-2 **FIL5_1:FIL5_0:** Filter 5 Select bits 1 and 0
`11` = No mask
`10` = Filter 15
`01` = Acceptance Mask 1
`00` = Acceptance Mask 0

bit 1-0 **FIL4_1:FIL4_0:** Filter 4 Select bits 1 and 0
`11` = No mask
`10` = Filter 15
`01` = Acceptance Mask 1
`00` = Acceptance Mask 0

Note 1: This register is available in Mode 1 and 2 only.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
- n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

REGISTER 23-50: MSEL2: MASK SELECT REGISTER 2(1)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
FIL11_1	FIL11_0	FIL10_1	FIL10_0	FIL9_1	FIL9_0	FIL8_1	FIL8_0
bit 7							bit 0

bit 7-6 **FIL11_1:FIL11_0:** Filter 11 Select bits 1 and 0
11 = No mask
10 = Filter 15
01 = Acceptance Mask 1
00 = Acceptance Mask 0

bit 5-4 **FIL10_1:FIL10_0:** Filter 10 Select bits 1 and 0
11 = No mask
10 = Filter 15
01 = Acceptance Mask 1
00 = Acceptance Mask 0

bit 3-2 **FIL9_1:FIL9_0:** Filter 9 Select bits 1 and 0
11 = No mask
10 = Filter 15
01 = Acceptance Mask 1
00 = Acceptance Mask 0

bit 1-0 **FIL8_1:FIL8_0:** Filter 8 Select bits 1 and 0
11 = No mask
10 = Filter 15
01 = Acceptance Mask 1
00 = Acceptance Mask 0

Note 1: This register is available in Mode 1 and 2 only.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

REGISTER 23-51: MSEL3: MASK SELECT REGISTER 3(1)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
FIL15_1	FIL15_0	FIL14_1	FIL14_0	FIL13_1	FIL13_0	FIL12_1	FIL12_0
bit 7							bit 0

bit 7-6 **FIL15_1:FIL15_0:** Filter 15 Select bits 1 and 0
`11` = No mask
`10` = Filter 15
`01` = Acceptance Mask 1
`00` = Acceptance Mask 0

bit 5-4 **FIL14_1:FIL14_0:** Filter 14 Select bits 1 and 0
`11` = No mask
`10` = Filter 15
`01` = Acceptance Mask 1
`00` = Acceptance Mask 0

bit 3-2 **FIL13_1:FIL13_0:** Filter 13 Select bits 1 and 0
`11` = No mask
`10` = Filter 15
`01` = Acceptance Mask 1
`00` = Acceptance Mask 0

bit 1-0 **FIL12_1:FIL12_0:** Filter 12 Select bits 1 and 0
`11` = No mask
`10` = Filter 15
`01` = Acceptance Mask 1
`00` = Acceptance Mask 0

Note 1: This register is available in Mode 1 and 2 only.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
- n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

23.2.4 CAN BAUD RATE REGISTERS

This subsection describes the CAN Baud Rate registers.

Note: These registers are writable in Configuration mode only.

REGISTER 23-52: BRGCON1: BAUD RATE CONTROL REGISTER 1

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
SJW1	SJW0	BRP5	BRP4	BRP3	BRP2	BRP1	BRP0
bit 7							bit 0

bit 7-6 **SJW1:SJW0:** Synchronized Jump Width bits
`11` = Synchronization jump width time = 4 x T_Q
`10` = Synchronization jump width time = 3 x T_Q
`01` = Synchronization jump width time = 2 x T_Q
`00` = Synchronization jump width time = 1 x T_Q

bit 5-0 **BRP5:BRP0:** Baud Rate Prescaler bits
`111111` = T_Q = (2 x 64)/F_{OSC}
`111110` = T_Q = (2 x 63)/F_{OSC}
.
.
.
`000001` = T_Q = (2 x 2)/F_{OSC}
`000000` = T_Q = (2 x 1)/F_{OSC}

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
- n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

REGISTER 23-53: BRGCON2: BAUD RATE CONTROL REGISTER 2

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
SEG2PHTS	SAM	SEG1PH2	SEG1PH1	SEG1PH0	PRSEG2	PRSEG1	PRSEG0
bit 7							bit 0

bit 7 **SEG2PHTS:** Phase Segment 2 Time Select bit
1 = Freely programmable
0 = Maximum of PHEG1 or Information Processing Time (IPT), whichever is greater

bit 6 **SAM:** Sample of the CAN bus Line bit
1 = Bus line is sampled three times prior to the sample point
0 = Bus line is sampled once at the sample point

bit 5-3 **SEG1PH2:SEG1PH0:** Phase Segment 1 bits
111 = Phase Segment 1 time = 8 x TQ
110 = Phase Segment 1 time = 7 x TQ
101 = Phase Segment 1 time = 6 x TQ
100 = Phase Segment 1 time = 5 x TQ
011 = Phase Segment 1 time = 4 x TQ
010 = Phase Segment 1 time = 3 x TQ
001 = Phase Segment 1 time = 2 x TQ
000 = Phase Segment 1 time = 1 x TQ

bit 2-0 **PRSEG2:PRSEG0:** Propagation Time Select bits
111 = Propagation time = 8 x TQ
110 = Propagation time = 7 x TQ
101 = Propagation time = 6 x TQ
100 = Propagation time = 5 x TQ
011 = Propagation time = 4 x TQ
010 = Propagation time = 3 x TQ
001 = Propagation time = 2 x TQ
000 = Propagation time = 1 x TQ

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
- n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

REGISTER 23-54: BRGCON3: BAUD RATE CONTROL REGISTER 3

R/W-0	R/W-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0
WAKDIS	WAKFIL	—	—	—	SEG2PH2(1)	SEG2PH1(1)	SEG2PH0(1)
bit 7							bit 0

bit 7 **WAKDIS:** Wake-up Disable bit
1 = Disable CAN bus activity wake-up feature
0 = Enable CAN bus activity wake-up feature

bit 6 **WAKFIL:** Selects CAN bus Line Filter for Wake-up bit
1 = Use CAN bus line filter for wake-up
0 = CAN bus line filter is not used for wake-up

bit 5-3 **Unimplemented:** Read as '0'

bit 2-0 **SEG2PH2:SEG2PH0:** Phase Segment 2 Time Select bits(1)
111 = Phase Segment 2 time = 8 x T_Q
110 = Phase Segment 2 time = 7 x T_Q
101 = Phase Segment 2 time = 6 x T_Q
100 = Phase Segment 2 time = 5 x T_Q
011 = Phase Segment 2 time = 4 x T_Q
010 = Phase Segment 2 time = 3 x T_Q
001 = Phase Segment 2 time = 2 x T_Q
000 = Phase Segment 2 time = 1 x T_Q

Note 1: Ignored if SEG2PHTS bit (BRGCON2<7>) is '0'.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

23.2.5 CAN MODULE I/O CONTROL REGISTER

This register controls the operation of the CAN module's I/O pins in relation to the rest of the microcontroller.

REGISTER 23-55: CIOCON: CAN I/O CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	U-0	U-0	U-0	U-0
TX2SRC	TX2EN	ENDRHI	CANCAP	—	—	—	—
bit 7							bit 0

bit 7 **TX2SRC**: CANTX2 Pin Data Source bit
1 = CANTX2 pin will output the CAN clock
0 = CANTX2 pin will output $\overline{\text{CANTX1}}$

bit 6 **TX2EN**: CANTX2 Pin Enable bit
1 = CANTX2 pin will output $\overline{\text{CANTX1}}$ or CAN clock as selected by TX2SRC bit
0 = CANTX2 pin will have digital I/O function

bit 5 **ENDRHI:** Enable Drive High bit(1)
1 = CANTX pin will drive V_{DD} when recessive
0 = CANTX pin will be tri-state when recessive

bit 4 **CANCAP:** CAN Message Receive Capture Enable bit
1 = Enable CAN capture, CAN message receive signal replaces input on RC2/CCP1
0 = Disable CAN capture, RC2/CCP1 input to CCP1 module

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

Note 1: Always set this bit when using differential bus to avoid signal crosstalk in CANTX from other nearby pins.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

23.2.6 CAN INTERRUPT REGISTERS

The registers in this section are the same as described in **Section 9.0 “Interrupts”**. They are duplicated here for convenience.

REGISTER 23-56: PIR3: PERIPHERAL INTERRUPT FLAG REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
IRXIF	WAKIF	ERRIF	TXB2IF/ TXBnIF	TXB1IF(1)	TXB0IF(1)	RXB1IF/ RXBnIF	RXB0IF/ FIFOWMIF
bit 7							bit 0

bit 7 **IRXIF:** CAN Invalid Received Message Interrupt Flag bit
1 = An invalid message has occurred on the CAN bus
0 = No invalid message on CAN bus

bit 6 **WAKIF:** CAN bus Activity Wake-up Interrupt Flag bit
1 = Activity on CAN bus has occurred
0 = No activity on CAN bus

bit 5 **ERRIF:** CAN bus Error Interrupt Flag bit
1 = An error has occurred in the CAN module (multiple sources)
0 = No CAN module errors

bit 4 When CAN is in Mode 0:
TXB2IF: CAN Transmit Buffer 2 Interrupt Flag bit
1 = Transmit Buffer 2 has completed transmission of a message and may be reloaded
0 = Transmit Buffer 2 has not completed transmission of a message
When CAN is in Mode 1 or 2:
TXBnIF: Any Transmit Buffer Interrupt Flag bit
1 = One or more transmit buffers has completed transmission of a message and may be reloaded
0 = No transmit buffer is ready for reload

bit 3 **TXB1IF:** CAN Transmit Buffer 1 Interrupt Flag bit(1)
1 = Transmit Buffer 1 has completed transmission of a message and may be reloaded
0 = Transmit Buffer 1 has not completed transmission of a message

bit 2 **TXB0IF:** CAN Transmit Buffer 0 Interrupt Flag bit(1)
1 = Transmit Buffer 0 has completed transmission of a message and may be reloaded
0 = Transmit Buffer 0 has not completed transmission of a message

bit 1 When CAN is in Mode 0:
RXB1IF: CAN Receive Buffer 1 Interrupt Flag bit
1 = Receive Buffer 1 has received a new message
0 = Receive Buffer 1 has not received a new message
When CAN is in Mode 1 or 2:
RXBnIF: Any Receive Buffer Interrupt Flag bit
1 = One or more receive buffers has received a new message
0 = No receive buffer has received a new message

bit 0 When CAN is in Mode 0:
RXB0IF: CAN Receive Buffer 0 Interrupt Flag bit
1 = Receive Buffer 0 has received a new message
0 = Receive Buffer 0 has not received a new message
When CAN is in Mode 1:
Unimplemented: Read as ‘0’
When CAN is in Mode 2:
FIFOWMIF: FIFO Watermark Interrupt Flag bit
1 = FIFO high watermark is reached
0 = FIFO high watermark is not reached

Note 1: In CAN Mode 1 and 2, this bit is forced to ‘0’.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
- n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

REGISTER 23-57: PIE3: PERIPHERAL INTERRUPT ENABLE REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
IRXIE	WAKIE	ERRIE	TXB2IE/ TXBnIE	TXB1IE(1)	TXB0IE(1)	RXB1IE/ RXBnIE	RXB0IE/ FIFOWMIE

bit 7 bit 0

bit 7 **IRXIE:** CAN Invalid Received Message Interrupt Enable bit
1 = Enable invalid message received interrupt
0 = Disable invalid message received interrupt

bit 6 **WAKIE:** CAN bus Activity Wake-up Interrupt Enable bit
1 = Enable bus activity wake-up interrupt
0 = Disable bus activity wake-up interrupt

bit 5 **ERRIE:** CAN bus Error Interrupt Enable bit
1 = Enable CAN bus error interrupt
0 = Disable CAN bus error interrupt

bit 4 When CAN is in Mode 0:
TXB2IE: CAN Transmit Buffer 2 Interrupt Enable bit
1 = Enable Transmit Buffer 2 interrupt
0 = Disable Transmit Buffer 2 interrupt

When CAN is in Mode 1 or 2:
TXBnIE: CAN Transmit Buffer Interrupts Enable bit
1 = Enable transmit buffer interrupt; individual interrupt is enabled by TXBIE and BIE0
0 = Disable all transmit buffer interrupts

bit 3 **TXB1IE:** CAN Transmit Buffer 1 Interrupt Enable bit(1)
1 = Enable Transmit Buffer 1 interrupt
0 = Disable Transmit Buffer 1 interrupt

bit 2 **TXB0IE:** CAN Transmit Buffer 0 Interrupt Enable bit(1)
1 = Enable Transmit Buffer 0 interrupt
0 = Disable Transmit Buffer 0 interrupt

bit 1 When CAN is in Mode 0:
RXB1IE: CAN Receive Buffer 1 Interrupt Enable bit
1 = Enable Receive Buffer 1 interrupt
0 = Disable Receive Buffer 1 interrupt

When CAN is in Mode 1 or 2:
RXBnIE: CAN Receive Buffer Interrupts Enable bit
1 = Enable receive buffer interrupt; individual interrupt is enabled by BIE0
0 = Disable all receive buffer interrupts

bit 0 When CAN is in Mode 0:
RXB0IE: CAN Receive Buffer 0 Interrupt Enable bit
1 = Enable Receive Buffer 0 interrupt
0 = Disable Receive Buffer 0 interrupt
When CAN is in Mode 1:
Unimplemented: Read as ‘0’
When CAN is in Mode 2:
FIFOWMIE: FIFO Watermark Interrupt Enable bit
1 = Enable FIFO watermark interrupt
0 = Disable FIFO watermark interrupt

Note 1: In CAN Mode 1 and 2, this bit is forced to ‘0’.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’	
- n = Value at POR	‘1’ = Bit is set	‘0’ = Bit is cleared	x = Bit is unknown

REGISTER 23-58: IPR3: PERIPHERAL INTERRUPT PRIORITY REGISTER

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
IRXIP	WAKIP	ERRIP	TXB2IP/ TXBnIP	TXB1IP(1)	TXB0IP(1)	RXB1IP/ RXBnIP	RXB0IP/ FIFOWMIP
bit 7							bit 0

bit 7 **IRXIP:** CAN Invalid Received Message Interrupt Priority bit
1 = High priority
0 = Low priority

bit 6 **WAKIP:** CAN bus Activity Wake-up Interrupt Priority bit
1 = High priority
0 = Low priority

bit 5 **ERRIP:** CAN bus Error Interrupt Priority bit
1 = High priority
0 = Low priority

bit 4 When CAN is in Mode 0:
TXB2IP: CAN Transmit Buffer 2 Interrupt Priority bit
1 = High priority
0 = Low priority

When CAN is in Mode 1 or 2:
TXBnIP: CAN Transmit Buffer Interrupt Priority bit
1 = High priority
0 = Low priority

bit 3 **TXB1IP:** CAN Transmit Buffer 1 Interrupt Priority bit(1)
1 = High priority
0 = Low priority

bit 2 **TXB0IP:** CAN Transmit Buffer 0 Interrupt Priority bit(1)
1 = High priority
0 = Low priority

bit 1 When CAN is in Mode 0:
RXB1IP: CAN Receive Buffer 1 Interrupt Priority bit
1 = High priority
0 = Low priority

When CAN is in Mode 1 or 2:
RXBnIP: CAN Receive Buffer Interrupts Priority bit
1 = High priority
0 = Low priority

bit 0 When CAN is in Mode 0:
RXB0IP: CAN Receive Buffer 0 Interrupt Priority bit
1 = High priority
0 = Low priority

When CAN is in Mode 1:
Unimplemented: Read as '0'

When CAN is in Mode 2:
FIFOWMIP: FIFO Watermark Interrupt Priority bit
1 = High priority
0 = Low priority

Note 1: In CAN Mode 1 and 2, this bit is forced to '0'.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

REGISTER 23-59: TXBIE: TRANSMIT BUFFERS INTERRUPT ENABLE REGISTER(1)

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	U-0	U-0
—	—	—	TXB2IE	TXB1IE	TXB0IE	—	—
bit 7							bit 0

bit 7-5 **Unimplemented:** Read as '0'

bit 4-2 **TX2BIE:TXB0IE:** Transmit Buffer 2-0 Interrupt Enable bit(2)
1 = Transmit buffer interrupt is enabled
0 = Transmit buffer interrupt is disabled

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

Note 1: This register is available in Mode 1 and 2 only.
2: TXBIE in PIE3 register must be set to get an interrupt.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

REGISTER 23-60: BIE0: BUFFER INTERRUPT ENABLE REGISTER 0(1)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
B5IE	B4IE	B3IE	B2IE	B1IE	B0IE	RXB1IE	RXB0IE
bit 7							bit 0

bit 7-2 **B5IE:B0IE:** Programmable Transmit/Receive Buffer 5-0 Interrupt Enable bit(2)
1 = Interrupt is enabled
0 = Interrupt is disabled

bit 1-0 **RXB1IE:RXB0IE:** Dedicated Receive Buffer 1-0 Interrupt Enable bit(2)
1 = Interrupt is enabled
0 = Interrupt is disabled

Note 1: This register is available in Mode 1 and 2 only.
2: Either TXBIE or RXBIE in PIE3 register must be set to get an interrupt.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

TABLE 23-1: CAN CONTROLLER REGISTER MAP

Address[(1)]	Name	Address	Name	Address	Name	Address	Name
F7Fh	SPBRGH[(3)]	F5Fh	CANCON_RO0	F3Fh	CANCON_RO2	F1Fh	RXM1EIDL
F7Eh	BAUDCON[(3)]	F5Eh	CANSTAT_RO0	F3Eh	CANSTAT_RO2	F1Eh	RXM1EIDH
F7Dh	—[(4)]	F5Dh	RXB1D7	F3Dh	TXB1D7	F1Dh	RXM1SIDL
F7Ch	—[(4)]	F5Ch	RXB1D6	F3Ch	TXB1D6	F1Ch	RXM1SIDH
F7Bh	—[(4)]	F5Bh	RXB1D5	F3Bh	TXB1D5	F1Bh	RXM0EIDL
F7Ah	—[(4)]	F5Ah	RXB1D4	F3Ah	TXB1D4	F1Ah	RXM0EIDH
F79h	ECCP1DEL[(3)]	F59h	RXB1D3	F39h	TXB1D3	F19h	RXM0SIDL
F78h	—[(4)]	F58h	RXB1D2	F38h	TXB1D2	F18h	RXM0SIDH
F77h	ECANCON	F57h	RXB1D1	F37h	TXB1D1	F17h	RXF5EIDL
F76h	TXERRCNT	F56h	RXB1D0	F36h	TXB1D0	F16h	RXF5EIDH
F75h	RXERRCNT	F55h	RXB1DLC	F35h	TXB1DLC	F15h	RXF5SIDL
F74h	COMSTAT	F54h	RXB1EIDL	F34h	TXB1EIDL	F14h	RXF5SIDH
F73h	CIOCON	F53h	RXB1EIDH	F33h	TXB1EIDH	F13h	RXF4EIDL
F72h	BRGCON3	F52h	RXB1SIDL	F32h	TXB1SIDL	F12h	RXF4EIDH
F71h	BRGCON2	F51h	RXB1SIDH	F31h	TXB1SIDH	F11h	RXF4SIDL
F70h	BRGCON1	F50h	RXB1CON	F30h	TXB1CON	F10h	RXF4SIDH
F6Fh	CANCON	F4Fh	CANCON_RO1[(2)]	F2Fh	CANCON_RO3[(2)]	F0Fh	RXF3EIDL
F6Eh	CANSTAT	F4Eh	CANSTAT_RO1[(2)]	F2Eh	CANSTAT_RO3[(2)]	F0Eh	RXF3EIDH
F6Dh	RXB0D7	F4Dh	TXB0D7	F2Dh	TXB2D7	F0Dh	RXF3SIDL
F6Ch	RXB0D6	F4Ch	TXB0D6	F2Ch	TXB2D6	F0Ch	RXF3SIDH
F6Bh	RXB0D5	F4Bh	TXB0D5	F2Bh	TXB2D5	F0Bh	RXF2EIDL
F6Ah	RXB0D4	F4Ah	TXB0D4	F2Ah	TXB2D4	F0Ah	RXF2EIDH
F69h	RXB0D3	F49h	TXB0D3	F29h	TXB2D3	F09h	RXF2SIDL
F68h	RXB0D2	F48h	TXB0D2	F28h	TXB2D2	F08h	RXF2SIDH
F67h	RXB0D1	F47h	TXB0D1	F27h	TXB2D1	F07h	RXF1EIDL
F66h	RXB0D0	F46h	TXB0D0	F26h	TXB2D0	F06h	RXF1EIDH
F65h	RXB0DLC	F45h	TXB0DLC	F25h	TXB2DLC	F05h	RXF1SIDL
F64h	RXB0EIDL	F44h	TXB0EIDL	F24h	TXB2EIDL	F04h	RXF1SIDH
F63h	RXB0EIDH	F43h	TXB0EIDH	F23h	TXB2EIDH	F03h	RXF0EIDL
F62h	RXB0SIDL	F42h	TXB0SIDL	F22h	TXB2SIDL	F02h	RXF0EIDH
F61h	RXB0SIDH	F41h	TXB0SIDH	F21h	TXB2SIDH	F01h	RXF0SIDL
F60h	RXB0CON	F40h	TXB0CON	F20h	TXB2CON	F00h	RXF0SIDH

Note 1: Shaded registers are available in Access Bank low area while the rest are available in Bank 15.
2: CANSTAT register is repeated in these locations to simplify application firmware. Unique names are given for each instance of the controller register due to the Microchip header file requirement.
3: These registers are not CAN registers.
4: Unimplemented registers are read as '0'.

TABLE 23-1: CAN CONTROLLER REGISTER MAP (CONTINUED)

Address[1]	Name	Address	Name	Address	Name	Address	Name
EFFh	—(4)	EDFh	—(4)	EBFh	—(4)	E9Fh	—(4)
EFEh	—(4)	EDEh	—(4)	EBEh	—(4)	E9Eh	—(4)
EFDh	—(4)	EDDh	—(4)	EBDh	—(4)	E9Dh	—(4)
EFCh	—(4)	EDCh	—(4)	EBCh	—(4)	E9Ch	—(4)
EFBh	—(4)	EDBh	—(4)	EBBh	—(4)	E9Bh	—(4)
EFAh	—(4)	EDAh	—(4)	EBAh	—(4)	E9Ah	—(4)
EF9h	—(4)	ED9h	—(4)	EB9h	—(4)	E99h	—(4)
EF8h	—(4)	ED8h	—(4)	EB8h	—(4)	E98h	—(4)
EF7h	—(4)	ED7h	—(4)	EB7h	—(4)	E97h	—(4)
EF6h	—(4)	ED6h	—(4)	EB6h	—(4)	E96h	—(4)
EF5h	—(4)	ED5h	—(4)	EB5h	—(4)	E95h	—(4)
EF4h	—(4)	ED4h	—(4)	EB4h	—(4)	E94h	—(4)
EF3h	—(4)	ED3h	—(4)	EB3h	—(4)	E93h	—(4)
EF2h	—(4)	ED2h	—(4)	EB2h	—(4)	E92h	—(4)
EF1h	—(4)	ED1h	—(4)	EB1h	—(4)	E91h	—(4)
EF0h	—(4)	ED0h	—(4)	EB0h	—(4)	E90h	—(4)
EEFh	—(4)	ECFh	—(4)	EAFh	—(4)	E8Fh	—(4)
EEEh	—(4)	ECEh	—(4)	EAEh	—(4)	E8Eh	—(4)
EEDh	—(4)	ECDh	—(4)	EADh	—(4)	E8Dh	—(4)
EECh	—(4)	ECCh	—(4)	EACh	—(4)	E8Ch	—(4)
EEBh	—(4)	ECBh	—(4)	EABh	—(4)	E8Bh	—(4)
EEAh	—(4)	ECAh	—(4)	EAAh	—(4)	E8Ah	—(4)
EE9h	—(4)	EC9h	—(4)	EA9h	—(4)	E89h	—(4)
EE8h	—(4)	EC8h	—(4)	EA8h	—(4)	E88h	—(4)
EE7h	—(4)	EC7h	—(4)	EA7h	—(4)	E87h	—(4)
EE6h	—(4)	EC6h	—(4)	EA6h	—(4)	E86h	—(4)
EE5h	—(4)	EC5h	—(4)	EA5h	—(4)	E85h	—(4)
EE4h	—(4)	EC4h	—(4)	EA4h	—(4)	E84h	—(4)
EE3h	—(4)	EC3h	—(4)	EA3h	—(4)	E83h	—(4)
EE2h	—(4)	EC2h	—(4)	EA2h	—(4)	E82h	—(4)
EE1h	—(4)	EC1h	—(4)	EA1h	—(4)	E81h	—(4)
EE0h	—(4)	EC0h	—(4)	EA0h	—(4)	E80h	—(4)

Note 1: Shaded registers are available in Access Bank low area while the rest are available in Bank 15.

2: CANSTAT register is repeated in these locations to simplify application firmware. Unique names are given for each instance of the controller register due to the Microchip header file requirement.

3: These registers are not CAN registers.

4: Unimplemented registers are read as ‘0’.

TABLE 23-1: CAN CONTROLLER REGISTER MAP (CONTINUED)

Address[(1)]	Name	Address	Name	Address	Name	Address	Name
E7Fh	CANCON_RO4[(2)]	E5Fh	CANCON_RO6[(2)]	E3Fh	CANCON_RO8[(2)]	E1Fh	—[(4)]
E7Eh	CANSTAT_RO4[(2)]	E5Eh	CANSTAT_RO6[(2)]	E3Eh	CANSTAT_RO8[(2)]	E1Eh	—[(4)]
E7Dh	B5D7	E5Dh	B3D7	E3Dh	B1D7	E1Dh	—[(4)]
E7Ch	B5D6	E5Ch	B3D6	E3Ch	B1D6	E1Ch	—[(4)]
E7Bh	B5D5	E5Bh	B3D5	E3Bh	B1D5	E1Bh	—[(4)]
E7Ah	B5D4	E5Ah	B3D4	E3Ah	B1D4	E1Ah	—[(4)]
E79h	B5D3	E59h	B3D3	E39h	B1D3	E19h	—[(4)]
E78h	B5D2	E58h	B3D2	E38h	B1D2	E18h	—[(4)]
E77h	B5D1	E57h	B3D1	E37h	B1D1	E17h	—[(4)]
E76h	B5D0	E56h	B3D0	E36h	B1D0	E16h	—[(4)]
E75h	B5DLC	E55h	B3DLC	E35h	B1DLC	E15h	—[(4)]
E74h	B5EIDL	E54h	B3EIDL	E34h	B1EIDL	E14h	—[(4)]
E73h	B5EIDH	E53h	B3EIDH	E33h	B1EIDH	E13h	—[(4)]
E72h	B5SIDL	E52h	B3SIDL	E32h	B1SIDL	E12h	—[(4)]
E71h	B5SIDH	E51h	B3SIDH	E31h	B1SIDH	E11h	—[(4)]
E70h	B5CON	E50h	B3CON	E30h	B1CON	E10h	—[(4)]
E6Fh	CANCON_RO5	E4Fh	CANCON_RO7	E2Fh	CANCON_RO9	E0Fh	—[(4)]
E6Eh	CANSTAT_RO5	E4Eh	CANSTAT_RO7	E2Eh	CANSTAT_RO9	E0Eh	—[(4)]
E6Dh	B4D7	E4Dh	B2D7	E2Dh	B0D7	E0Dh	—[(4)]
E6Ch	B4D6	E4Ch	B2D6	E2Ch	B0D6	E0Ch	—[(4)]
E6Bh	B4D5	E4Bh	B2D5	E2Bh	B0D5	E0Bh	—[(4)]
E6Ah	B4D4	E4Ah	B2D4	E2Ah	B0D4	E0Ah	—[(4)]
E69h	B4D3	E49h	B2D3	E29h	B0D3	E09h	—[(4)]
E68h	B4D2	E48h	B2D2	E28h	B0D2	E08h	—[(4)]
E67h	B4D1	E47h	B2D1	E27h	B0D1	E07h	—[(4)]
E66h	B4D0	E46h	B2D0	E26h	B0D0	E06h	—[(4)]
E65h	B4DLC	E45h	B2DLC	E25h	B0DLC	E05h	—[(4)]
E64h	B4EIDL	E44h	B2EIDL	E24h	B0EIDL	E04h	—[(4)]
E63h	B4EIDH	E43h	B2EIDH	E23h	B0EIDH	E03h	—[(4)]
E62h	B4SIDL	E42h	B2SIDL	E22h	B0SIDL	E02h	—[(4)]
E61h	B4SIDH	E41h	B2SIDH	E21h	B0SIDH	E01h	—[(4)]
E60h	B4CON	E40h	B2CON	E20h	B0CON	E00h	—[(4)]

Note 1: Shaded registers are available in Access Bank low area while the rest are available in Bank 15.
2: CANSTAT register is repeated in these locations to simplify application firmware. Unique names are given for each instance of the controller register due to the Microchip header file requirement.
3: These registers are not CAN registers.
4: Unimplemented registers are read as '0'.

TABLE 23-1: CAN CONTROLLER REGISTER MAP (CONTINUED)

Address[(1)]	Name	Address	Name	Address	Name	Address	Name
DFFh	—[(4)]	DDFh	—[(4)]	DBFh	—[(4)]	D9Fh	—[(4)]
DFEh	—[(4)]	DDEh	—[(4)]	DBEh	—[(4)]	D9Eh	—[(4)]
DFDh	—[(4)]	DDDh	—[(4)]	DBDh	—[(4)]	D9Dh	—[(4)]
DFCh	TXBIE	DDCh	—[(4)]	DBCh	—[(4)]	D9Ch	—[(4)]
DFBh	—[(4)]	DDBh	—[(4)]	DBBh	—[(4)]	D9Bh	—[(4)]
DFAh	BIE0	DDAh	—[(4)]	DBAh	—[(4)]	D9Ah	—[(4)]
DF9h	—[(4)]	DD9h	—[(4)]	DB9h	—[(4)]	D99h	—[(4)]
DF8h	BSEL0	DD8h	SDFLC	DB8h	—[(4)]	D98h	—[(4)]
DF7h	—[(4)]	DD7h	—[(4)]	DB7h	—[(4)]	D97h	—[(4)]
DF6h	—[(4)]	DD6h	—[(4)]	DB6h	—[(4)]	D96h	—[(4)]
DF5h	—[(4)]	DD5h	RXFCON1	DB5h	—[(4)]	D95h	—[(4)]
DF4h	—[(4)]	DD4h	RXFCON0	DB4h	—[(4)]	D94h	—[(4)]
DF3h	MSEL3	DD3h	—[(4)]	DB3h	—[(4)]	D93h	RXF15EIDL
DF2h	MSEL2	DD2h	—[(4)]	DB2h	—[(4)]	D92h	RXF15EIDH
DF1h	MSEL1	DD1h	—[(4)]	DB1h	—[(4)]	D91h	RXF15SIDL
DF0h	MSEL0	DD0h	—[(4)]	DB0h	—[(4)]	D90h	RXF15SIDH
DEFh	—[(4)]	DCFh	—[(4)]	DAFh	—[(4)]	D8Fh	—[(4)]
DEEh	—[(4)]	DCEh	—[(4)]	DAEh	—[(4)]	D8Eh	—[(4)]
DEDh	—[(4)]	DCDh	—[(4)]	DADh	—[(4)]	D8Dh	—[(4)]
DECh	—[(4)]	DCCh	—[(4)]	DACh	—[(4)]	D8Ch	—[(4)]
DEBh	—[(4)]	DCBh	—[(4)]	DABh	—[(4)]	D8Bh	RXF14EIDL
DEAh	—[(4)]	DCAh	—[(4)]	DAAh	—[(4)]	D8Ah	RXF14EIDH
DE9h	—[(4)]	DC9h	—[(4)]	DA9h	—[(4)]	D89h	RXF14SIDL
DE8h	—[(4)]	DC8h	—[(4)]	DA8h	—[(4)]	D88h	RXF14SIDH
DE7h	RXFBCON7	DC7h	—[(4)]	DA7h	—[(4)]	D87h	RXF13EIDL
DE6h	RXFBCON6	DC6h	—[(4)]	DA6h	—[(4)]	D86h	RXF13EIDH
DE5h	RXFBCON5	DC5h	—[(4)]	DA5h	—[(4)]	D85h	RXF13SIDL
DE4h	RXFBCON4	DC4h	—[(4)]	DA4h	—[(4)]	D84h	RXF13SIDH
DE3h	RXFBCON3	DC3h	—[(4)]	DA3h	—[(4)]	D83h	RXF12EIDL
DE2h	RXFBCON2	DC2h	—[(4)]	DA2h	—[(4)]	D82h	RXF12EIDH
DE1h	RXFBCON1	DC1h	—[(4)]	DA1h	—[(4)]	D81h	RXF12SIDL
DE0h	RXFBCON0	DC0h	—[(4)]	DA0h	—[(4)]	D80h	RXF12SIDH

Note 1: Shaded registers are available in Access Bank low area while the rest are available in Bank 15.

2: CANSTAT register is repeated in these locations to simplify application firmware. Unique names are given for each instance of the controller register due to the Microchip header file requirement.

3: These registers are not CAN registers.

4: Unimplemented registers are read as '0'.

TABLE 23-1: CAN CONTROLLER REGISTER MAP (CONTINUED)

Address(1)	Name
D7Fh	—(4)
D7Eh	—(4)
D7Dh	—(4)
D7Ch	—(4)
D7Bh	RXF11EIDL
D7Ah	RXF11EIDH
D79h	RXF11SIDL
D78h	RXF11SIDH
D77h	RXF10EIDL
D76h	RXF10EIDH
D75h	RXF10SIDL
D74h	RXF10SIDH
D73h	RXF9EIDL
D72h	RXF9EIDH
D71h	RXF9SIDL
D70h	RXF9SIDH
D6Fh	—(4)
D6Eh	—(4)
D6Dh	—(4)
D6Ch	—(4)
D6Bh	RXF8EIDL
D6Ah	RXF8EIDH
D69h	RXF8SIDL
D68h	RXF8SIDH
D67h	RXF7EIDL
D66h	RXF7EIDH
D65h	RXF7SIDL
D64h	RXF7SIDH
D63h	RXF6EIDL
D62h	RXF6EIDH
D61h	RXF6SIDL
D60h	RXF6SIDH

Note 1: Shaded registers are available in Access Bank low area while the rest are available in Bank 15.

2: CANSTAT register is repeated in these locations to simplify application firmware. Unique names are given for each instance of the controller register due to the Microchip header file requirement.

3: These registers are not CAN registers.

4: Unimplemented registers are read as '0'.

23.3 CAN Modes of Operation

The PIC18F6585/8585/6680/8680 has six main modes of operation:

- Configuration mode
- Disable mode
- Normal Operation mode
- Listen Only mode
- Loopback mode
- Error Recognition mode

All modes, except Error Recognition, are requested by setting the REQOP bits (CANCON<7:5>); Error Recognition is requested through the RXM bits of the Receive Buffer register(s). Entry into a mode is Acknowledged by monitoring the OPMODE bits.

When changing modes, the mode will not actually change until all pending message transmissions are complete. Because of this, the user must verify that the device has actually changed into the requested mode before further operations are executed.

23.3.1 CONFIGURATION MODE

The CAN module must be initialized before the activation. This is only possible if the module is in the Configuration mode. The Configuration mode is requested by setting the REQOP2 bit. Only when the status bit, OPMODE2, has a high level can the initialization be performed. Once in Configuration mode, registers such as baud rate control, acceptance mask/filter and ECAN mode selection can be modified. A new ECAN mode selection does not take into effect until Configuration mode is exited. The module is activated by setting the REQOP control bits to zero.

The module will protect the user from accidentally violating the CAN protocol through programming errors. All registers which control the configuration of the module can not be modified while the module is online. The CAN module will not be allowed to enter the Configuration mode while a transmission or reception is taking place. The CAN module will also not be allowed, if the CANRX pin is low (i.e., the CAN bus is busy). The CAN module waits for 11 recessive bits on the CAN bus (bus Idle condition) before switching to Configuration mode. The Configuration mode serves as a lock to protect the following registers:

- Configuration registers
- Functional Mode Selection registers
- Bit Timing registers
- Identifier Acceptance Filter registers
- Identifier Acceptance Mask registers
- Filter and Mask Control registers
- Mask Selection registers

In the Configuration mode, the module will not transmit or receive. The error counters are cleared and the interrupt flags remain unchanged. The programmer will have access to configuration registers that are access restricted in other modes.

23.3.2 DISABLE MODE

In Disable mode, the module will not transmit or receive. The module has the ability to set the WAKIF bit due to bus activity; however, any pending interrupts will remain and the error counters will retain their value.

If REQOP<2:0> is set to '`001`', the module will enter the Module Disable mode. This mode is similar to disabling other peripheral modules by turning off the module enables. This causes the module internal clock to stop unless the module is active (i.e., receiving or transmitting a message). If the module is active, the module will wait for 11 recessive bits on the CAN bus, detect that condition as an Idle bus, then accept the module disable command. OPMODE<2:0> = `001` indicates whether the module successfully went into Module Disable mode.

The WAKIE interrupt is the only module interrupt that is still active in the Module Disable mode. If wake-up from CAN bus activity is required, the CAN module must be put into Disable mode before putting the core to Sleep. If the WAKDIS is cleared and WAKIE is set, the processor will receive an interrupt whenever the module detects recessive to dominant transition. On wake-up, the module will automatically be set to the previous mode of operation. For example, if the module was switched from Normal to Disable mode on bus activity wake-up, the module will automatically enter into Normal mode and the first message that caused the module to wake-up is lost. The module will not generate any error frame. Firmware logic must detect this condition and make sure that retransmission is requested. If the processor receives a wake-up interrupt while it is sleeping, more than one message may get lost. The actual number of messages lost would depend on the processor oscillator start-up time and incoming message bit rate.

The I/O pins will revert to normal I/O function when the module is in the Module Disable mode.

Note:	CAN module must be put in Disable or Configuration mode prior to putting the processor to sleep. Failure to do that may put the CAN module in indeterminate state.

23.3.3 NORMAL MODE

This is the standard operating mode of the PIC18F6585/8585/6680/8680 devices. In this mode, the device actively monitors all bus messages and generates Acknowledge bits, error frames, etc. This is also the only mode in which the PIC18F6585/8585/6680/8680 devices will transmit messages over the CAN bus.

23.3.4 LISTEN ONLY MODE

Listen Only mode provides a means for the PIC18F6585/8585/6680/8680 devices to receive all messages, including messages with errors. This mode can be used for bus monitor applications or for detecting the baud rate in 'hot plugging' situations. For auto-baud detection, it is necessary that there are at least two other nodes which are communicating with each other. The baud rate can be detected empirically by testing different values until valid messages are received. The Listen Only mode is a silent mode, meaning no messages will be transmitted while in this state, including error flags or Acknowledge signals. The filters and masks can be used to allow only particular messages to be loaded into the receive registers, or the filter masks can be set to all zeros to allow a message with any identifier to pass. The error counters are reset and deactivated in this state. The Listen Only mode is activated by setting the mode request bits in the CANCON register.

23.3.5 LOOPBACK MODE

This mode will allow internal transmission of messages from the transmit buffers to the receive buffers without actually transmitting messages on the CAN bus. This mode can be used in system development and testing. In this mode, the ACK bit is ignored and the device will allow incoming messages from itself, just as if they were coming from another node. The Loopback mode is a silent mode, meaning no messages will be transmitted while in this state, including error flags or Acknowledge signals. The CANTX pin will revert to port I/O while the device is in this mode. The filters and masks can be used to allow only particular messages to be loaded into the receive registers. The masks can be set to all zeros to provide a mode that accepts all messages. The Loopback mode is activated by setting the mode request bits in the CANCON register.

23.3.6 ERROR RECOGNITION MODE

The module can be set to ignore all errors and receive any message. In functional Mode 0, the Error Recognition mode is activated by setting the RXM<1:0> bits in the RXBnCON registers to '11'. In this mode, the data which is in the message assembly buffer until the error time, is copied in the receive buffer and can be read via the CPU interface.

23.4 CAN Module Functional Modes

In addition to CAN modes of operation, the ECAN module offers a total of three functional modes. Each of these modes are identified as Mode 0, Mode 1 and Mode 2.

23.4.1 MODE 0 – LEGACY MODE

Mode 0 is designed to be fully compatible with CAN modules used in PIC18CXX8 and PIC18FXX8 devices. This is the default mode of operation on all Reset conditions. As a result, module code written for the PIC18XX8 CAN module may be used on the ECAN module without any code changes.

The following is the list of resources available in Mode 0:

- Three transmit buffers: TXB0, TXB1 and TXB2
- Two receive buffers: RXB0 and RXB1
- Two acceptance masks, one for each receive buffer: RXM0, RXM1
- Six acceptance filters, 2 for RXB0 and 4 for RXB1: RXF0, RXF1, RXF2, RXF3, RXF4, RXF5

23.4.2 MODE 1 – ENHANCED LEGACY MODE

Mode 1 is similar to Mode 0, with the exception that more resources are available in Mode 1. There are 16 acceptance filters and two Acceptance Mask registers. Acceptance Filter 15 can be used as either an acceptance filter or an Acceptance Mask register. In addition to three transmit and two receive buffers, there are six more message buffers. One or more of these additional buffers can be programmed as transmit or receive buffers. These additional buffers can also be programmed to automatically handle RTR messages.

Fourteen of 16 Acceptance Filter registers can be dynamically associated to any receive buffer and Acceptance Mask register. This capability can be used to associate more than one filter to any one buffer.

When a receive buffer is programmed to use standard identifier messages, part of the full Acceptance Filter register can be used as data byte filter. The length of data byte filter is programmable from 0 to 18 bits. This functionality simplifies implementation of high-level protocols, such as DeviceNet.

The following is the list of resources available in Mode 1:

- Three transmit buffers: TXB0, TXB1 and TXB2
- Two receive buffers: RXB0 and RXB1
- Six buffers programmable as TX or RX: B0-B5
- Automatic RTR handling on B0-B5
- Sixteen dynamically assigned acceptance filters: RXF0-RXF15
- Two dedicated Acceptance Mask registers; RXF15 programmable as third mask: RXM0-RXM1, RXF15
- Programmable data filter on standard identifier messages: SDFLC

23.4.3 MODE 2 – ENHANCED FIFO MODE

In Mode 2, two or more receive buffers are used to form the receive FIFO (First In First Out) buffer. There is no one-to-one relation between the receive buffer and Acceptance Filter registers. Any filter that is enabled and linked to any FIFO receive buffer can generate acceptance and cause FIFO to be updated.

FIFO length is user programmable, from 2-8 buffers deep. FIFO length is determined by the very first programmable buffer that is configured as a transmit buffer. For example, if Buffer 2 (B2) is programmed as a transmit buffer, FIFO consists of RXB0, RXB1, B0 and B1 – creating a FIFO length of 4. If all programmable buffers are configured as receive buffers, FIFO will have the maximum length of 8.

The following is the list of resources available in Mode 2:

- Three transmit buffers: TXB0, TXB1 and TXB2
- Two receive buffers: RXB0 and RXB1
- Six buffers programmable as TX or RX; receive buffers form FIFO: B0-B5
- Automatic RTR handling on B0-B5
- Sixteen acceptance filters: RXF0-RXF15
- Two dedicated Acceptance Mask registers; RXF15 programmable as third mask: RXM0-RXM1, RXF15
- Programmable data filter on standard identifier messages: SDFLC, useful for DeviceNet protocol

23.5 CAN Message Buffers

23.5.1 DEDICATED TRANSMIT BUFFERS

The PIC18F6585/8585/6680/8680 devices implement three dedicated transmit buffers – TXB0, TXB1 and TXB2. Each of these buffers occupies 14 bytes of SRAM and are mapped into the SFR memory map. These are the only transmit buffers available in Mode 0. Mode 1 and 2 may access these and other additional buffers.

Each transmit buffer contains one Control register (TXBnCON), four Identifier registers (TXBnSIDL, TXBnSIDH, TXBnEIDL, TXBnEIDH), one Data Length Count register (TXBnDLC) and eight Data Byte registers (TXBnDm).

23.5.2 DEDICATED RECEIVE BUFFERS

The PIC18F6585/8585/6680/8680 devices implement two dedicated receive buffers – RXB0 and RXB1. Each of these buffers occupies 14 bytes of SRAM and are mapped into SFR memory map. These are the only receive buffers available in Mode 0. Mode 1 and 2 may access these and other additional buffers.

Each receive buffer contains one Control register (RXBnCON), four Identifier registers (RXBnSIDL, RXBnSIDH, RXBnEIDL, RXBnEIDH), one Data Length Count register (RXBnDLC) and eight Data Byte registers (RXBnDm).

There is also a separate Message Assembly Buffer (MAB) which acts as an additional receive buffer. MAB is always committed to receiving the next message from the bus and is not directly accessible to user firmware. The MAB assembles all incoming messages one by one. A message is transferred to appropriate receive buffers only if the corresponding acceptance filter criteria is met.

23.5.3 PROGRAMMABLE TRANSMIT/ RECEIVE BUFFERS

The ECAN module implements six new buffers: B0-B5. These buffers are individually programmable as either transmit or receive buffers. These buffers are available only in Mode 1 and 2. As with dedicated transmit and receive buffers, each of these programmable buffers occupies 14 bytes of SRAM and are mapped into SFR memory map.

Each buffer contains one Control register (BnCON), four Identifier registers (BnSIDL, BnSIDH, BnEIDL, BnEIDH), one Data Length Count register (BnDLC) and eight Data Byte registers (BnDm). Each of these registers contains two sets of control bits. Depending on whether the buffer is configured as transmit or receive, one would use the corresponding control bit set. By default, all buffers are configured as receive buffers. Each buffer can be individually configured as transmit or receive buffers by setting the corresponding TXENn bit in the BSEL0 register.

When configured as transmit buffers, user firmware may access transmit buffers in any order similar to accessing dedicated transmit buffers. In receive configuration, with Mode 1 enabled, user firmware may also access receive buffers in any order required. But in Mode 2, all receive buffers are combined to form a single FIFO. Actual FIFO length is programmable by user firmware. Access to FIFO must be done through the FIFO pointer bits (FP<4:0>) in the CANCON register. It must be noted that there is no hardware protection against out of order FIFO reads.

23.5.4 PROGRAMMABLE AUTO-RTR BUFFERS

In Mode 1 and 2, any of six programmable transmit/ receive buffers may be programmed to automatically respond to predefined RTR messages without user firmware intervention. Automatic RTR handling is enabled by setting the TXnEN bit in the BSEL0 register and the RTREN bit in the BnCON register. After this setup, when an RTR request is received, the TXREQ bit is automatically set and current buffer content is automatically queued for transmission as a RTR response. As with all transmit buffers, once the TXREQ bit is set, buffer registers become read-only and any writes to them will be ignored.

The following outlines the steps required to automatically handle RTR messages:

1. Set buffer to Transmit mode by setting TXnEN bit to '1' in BSEL0 register.
2. At least one acceptance filter must be associated with this buffer and preloaded with expected RTR identifier.
3. Bit RTREN in BnCON register must be set to '1'.
4. Buffer must be preloaded with the data to be sent as a RTR response.

Normally, user firmware will keep Buffer Data registers up to date. If firmware attempts to update buffer while an automatic RTR response is in process of transmission, all writes to buffers are ignored.

23.6 CAN Message Transmission

23.6.1 INITIATING TRANSMISSION

For the MCU to have write access to the message buffer, the TXREQ bit must be clear, indicating that the message buffer is clear of any pending message to be transmitted. At a minimum, the SIDH, SIDL, and DLC registers must be loaded. If data bytes are present in the message, the data registers must also be loaded. If the message is to use extended identifiers, the EIDH:EIDL registers must also be loaded and the EXIDE bit set.

To initiate message transmission, the TXREQ bit must be set for each buffer to be transmitted. When TXREQ is set, the TXABT, TXLARB and TXERR bits will be cleared. To successfully complete the transmission, there must be at least one node with matching baud rate on the network.

Setting the TXREQ bit does not initiate a message transmission, it merely flags a message buffer as ready for transmission. Transmission will start when the device detects that the bus is available. The device will then begin transmission of the highest priority message that is ready.

When the transmission has completed successfully, the TXREQ bit will be cleared, the TXBnIF bit will be set, and an interrupt will be generated if the TXBnIE bit is set.

If the message transmission fails, the TXREQ will remain set, indicating that the message is still pending for transmission and one of the following condition flags will be set. If the message started to transmit but encountered an error condition, the TXERR and the IRXIF bits will be set and an interrupt will be generated. If the message lost arbitration, the TXLARB bit will be set.

23.6.2 ABORTING TRANSMISSION

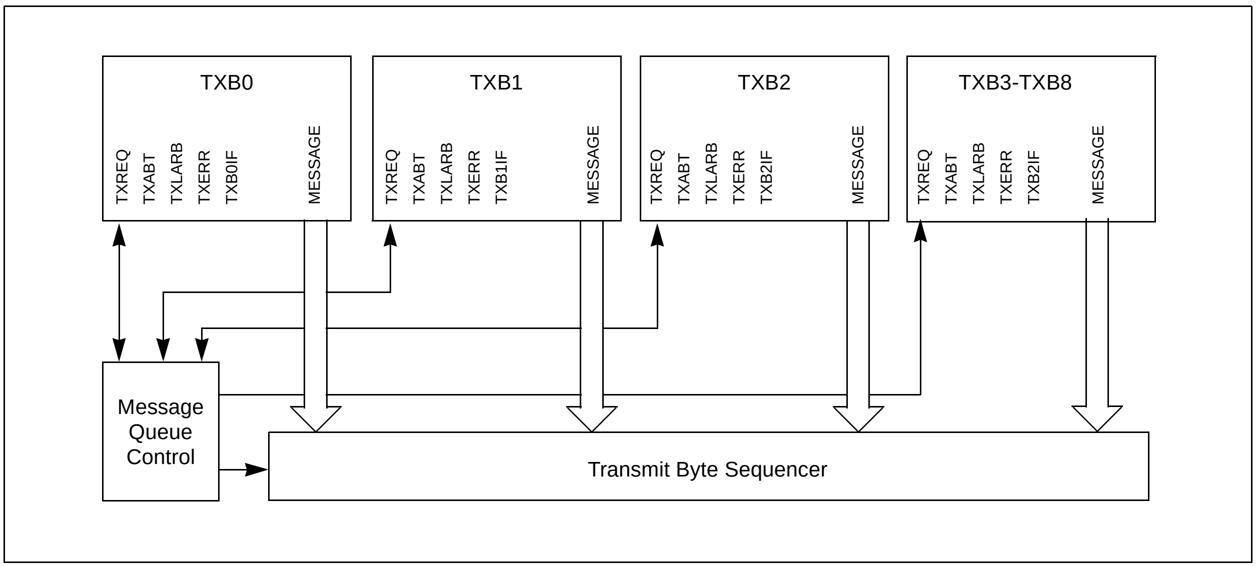
The MCU can request to abort a message by clearing the TXREQ bit associated with the corresponding message buffer (TXBnCON<3> or BnCON<3>). Setting the ABAT bit (CANCON<4>) will request an abort of all pending messages. If the message has not yet started transmission or if the message started but is interrupted by loss of arbitration or an error, the abort will be processed. The abort is indicated when the module sets the TXABT bit for the corresponding buffer (TXBnCON<6> or BnCON<6>). If the message has started to transmit, it will attempt to transmit the current message fully. If the current message is transmitted fully and is not lost to arbitration or an error, the TXABT bit will not be set because the message was transmitted successfully. Likewise, if a message is being transmitted during an abort request and the message is lost to arbitration or an error, the message will not be retransmitted and the TXABT bit will be set, indicating that the message was successfully aborted.

Once an abort is requested by setting ABAT or TXABT bits, it cannot be cleared to cancel the abort request. Only CAN module hardware or a POR condition can clear it.

23.6.3 TRANSMIT PRIORITY

Transmit priority is a prioritization within the PIC18F6585/8585/6680/8680 devices of the pending transmittable messages. This is independent from and not related to any prioritization implicit in the message arbitration scheme built into the CAN protocol. Prior to sending the SOF, the priority of all buffers that are queued for transmission is compared. The transmit buffer with the highest priority will be sent first. If more than one buffer has the same priority setting, the message is transmitted in the order of TXB2, TXB1, TXB0, B5, B4, B3, B2, B1, B0. There are four levels of transmit priority. If TXP bits for a particular message buffer are set to '11', that buffer has the highest possible priority. If TXP bits for a particular message buffer are '00', that buffer has the lowest possible priority.

FIGURE 23-2: TRANSMIT BUFFERS



23.7 Message Reception

23.7.1 RECEIVING A MESSAGE

Of all receive buffers, the MAB is always committed to receiving the next message from the bus. The MCU can access one buffer while the other buffer is available for message reception, or holding a previously received message.

> **Note:** The entire contents of the MAB are moved into the receive buffer once a message is accepted. This means that regardless of the type of identifier (standard or extended) and the number of data bytes received, the entire receive buffer is overwritten with the MAB contents. Therefore, the contents of all registers in the buffer must be assumed to have been modified when any message is received.

When a message is moved into either of the receive buffers, the associated RXFUL bit is set. This bit must be cleared by the MCU when it has completed processing the message in the buffer in order to allow a new message to be received into the buffer. This bit provides a positive lockout to ensure that the firmware has finished with the message before the module attempts to load a new message into the receive buffer. If the receive interrupt is enabled, an interrupt will be generated to indicate that a valid message has been received.

Once a message is loaded into any matching buffer, user firmware may determine exactly what filter caused this reception by checking the filter hit bits in the RXBnCON or BnCON registers. In Mode 0, FILHIT<3:0> of RXBnCON serve as filter hit bits. In Mode 1 and 2, FILHIT<4:0> of BnCON serve as filter hit bits. The same registers also indicate whether the current message is RTR frame or not. A received message is considered a standard identifier message if the EXID bit in RXBnSIDL or the BnSIDL register is cleared. Conversely, a set EXID bit indicates an extended identifier message. If the received message is a standard identifier message, user firmware needs to read the SIDL and SIDH registers. In the case of an extended identifier message, firmware should read the SIDL, SIDH, EIDL and EIDH registers. If the RXBnDLC or BnDLC register contain non-zero data count, user firmware should also read the corresponding number of data bytes by accessing the RXBnDm or BnDm registers. When a received message is RTR and if the current buffer is not configured for automatic RTR handling, user firmware must take appropriate action and respond manually.

Each receive buffer contains RXM bits to set special Receive modes. In Mode 0, RXM<1:0> bits in RXBnCON define a total of four Receive modes. In Mode 1 and 2, RXM1 bit in combination with the EXID mask and filter bit define the same four Receive modes. Normally, these bits are set to '00' to enable reception of all valid messages as determined by the appropriate acceptance filters. In this case, the determination of whether or not to receive standard or extended messages is determined by the EXIDE bit in the Acceptance Filter register. In Mode 0, if the RXM bits are set to '01' or '10', the receiver will accept only messages with standard or extended identifiers, respectively. If an acceptance filter has the EXIDE bit set such that it does not correspond with the RXM mode, that acceptance filter is rendered useless. In Mode 1 and 2, setting EXID in the SIDL Mask register will ensure that only standard or extended identifiers are received. These two modes of RXM bits can be used in systems where it is known that only standard or extended messages will be on the bus. If the RXM bits are set to '11' (RXM1 = 1 in Mode 1 and 2), the buffer will receive all messages regardless of the values of the acceptance filters. Also, if a message has an error before the end of frame, that portion of the message assembled in the MAB before the error frame, will be loaded into the buffer. This mode may serve as a valuable debugging tool for a given CAN network. It should not be used in an actual system environment as the actual system will always have some bus errors and all nodes on the bus are expected to ignore them.

In Mode 1 and 2, when a programmable buffer is configured as a transmit buffer and one or more acceptance filters are associated with it, all incoming messages matching this acceptance filter criteria will be discarded. To avoid this scenario, user firmware must make sure that there are no acceptance filters associated with a buffer configured as a transmit buffer.

23.7.2 RECEIVE PRIORITY

When in Mode 0, RXB0 is the higher priority buffer and has two message acceptance filters associated with it. RXB1 is the lower priority buffer and has four acceptance filters associated with it. The lower number of acceptance filters makes the match on RXB0 more restrictive and implies a higher priority for that buffer. Additionally, the RXB0CON register can be configured such that if RXB0 contains a valid message and another valid message is received, an overflow error will not occur and the new message will be moved into RXB1 regardless of the acceptance criteria of RXB1. There are also two programmable acceptance filter masks available, one for each receive buffer (see Section 4.5).

In Mode 1 and 2, there are a total of 16 acceptance filters available and each can be dynamically assigned to any of the receive buffers. A buffer with a lower number has higher priority. Given this, if an incoming message matches with two or more receive buffer acceptance criteria, the buffer with the lower number will be loaded with that message.

23.7.3 ENHANCED FIFO MODE

When configured for Mode 2, two of the dedicated receive buffers, in combination with one or more programmable transmit/receive buffers, are used to create a maximum of 8 buffers deep FIFO (First In First Out) buffer. In this mode, there is no direct correlation between filters and receive buffer registers. Any filter that has been enabled can generate an acceptance. When a message has been accepted, it is stored in the next available receive buffer register and an internal write pointer is incremented. The FIFO can be a maximum of 8 buffers deep. The entire FIFO must consist of contiguous receive buffers. The FIFO head begins at RXB0 buffer and its tail spans toward B5. The maximum length of the FIFO is limited by the presence or absence of the first transmit buffer starting from B0. If a buffer is configured as a transmit buffer, the FIFO length is reduced accordingly. For instance, if B3 is configured as transmit buffer, the actual FIFO will consist of RXB0, RXB1, B0, B1 and B2, a total of 5 buffers. If B0 is configured as a transmit buffer, the FIFO length will be 2. If none of the programmable buffers are configured as a transmit buffer, the FIFO will be 8 buffers deep. A system that requires more transmit buffers should try to locate transmit buffers at the very end of B0-B5 buffers to maximize available FIFO length.

When a message is received in FIFO mode, the Interrupt Flag Code bits (EICODE<4:0>) in the CANSTAT register will have a value of '10000', indicating the FIFO has received a message. FIFO pointer bits FP<3:0> in the CANCON register point to the buffer that contains data not yet read. The FIFO pointer bits, in this sense, serve as the FIFO read pointer. The user should use FP bits and read corresponding buffer data. When receive data is no longer needed, the RXFUL bit in the current buffer must be cleared, causing FP<3:0> to be updated by the module.

To determine whether FIFO is empty or not, the user may use FP<3:0> bits to access RXFUL bit in the current buffer. If RXFUL is cleared, the FIFO is considered to be empty. If it is set, the FIFO may contain one or more messages. In Mode 2, the module also provides a bit called FIFO High Water Mark (FIFOWM) in the ECANCON register. This bit can be used to cause an interrupt whenever the FIFO contains only one or four empty buffers. The FIFO high water mark interrupt can serve as an early warning to a full FIFO condition.

23.7.4 TIME-STAMPING

The CAN module can be programmed to generate a time-stamp for every message that is received. When enabled, the module generates a capture signal for CCP1, which in turn captures the value of either Timer1 or Timer3. This value can be used as the message time-stamp.

To use the time-stamp capability, the CANCAP bit (CIOCAN<4>) must be set. This replaces the capture input for CCP1 with the signal generated from the CAN module. In addition, CCP1CON<3:0> must be set to '0011' to enable the CCP special event trigger for CAN events.

23.8 Message Acceptance Filters and Masks

The message acceptance filters and masks are used to determine if a message in the message assembly buffer should be loaded into any of the receive buffers. Once a valid message has been received into the MAB, the identifier fields of the message are compared to the filter values. If there is a match, that message will be loaded into the appropriate receive buffer. The filter masks are used to determine which bits in the identifier are examined with the filters. A truth table is shown below in Table 23-2 that indicates how each bit in the identifier is compared to the masks and filters to determine if a message should be loaded into a receive buffer. The mask essentially determines which bits to apply the acceptance filters to. If any mask bit is set to a zero, then that bit will automatically be accepted regardless of the filter bit.

TABLE 23-2: FILTER/MASK TRUTH TABLE

Mask bit n	Filter bit n	Message Identifier bit n001	Accept or Reject bit n
0	x	x	Accept
1	0	0	Accept
1	0	1	Reject
1	1	0	Reject
1	1	1	Accept

Legend: x = don't care

In Mode 0, acceptance filters RXF0 and RXF1 and filter mask RXM0 are associated with RXB0. Filters RXF2, RXF3, RXF4 and RXF5 and mask RXM1 are associated with RXB1.

In Mode 1 and 2, there are an additional 10 acceptance filters, RXF6-RXF15, creating a total of 16 available filters. RXF15 can be used either as an acceptance filter or acceptance mask register. Each of these acceptance filters can be individually enabled or disabled by setting or clearing RXFENn bit in the RXFCONn register. Any of these 16 acceptance filters can be dynamically associated with any of the receive buffers. Actual association is made by setting appropriate bits in the RXFBCONn register. Each RXFBCONn register contains a nibble for each filter. This nibble can be used to associate a specific filter to any of available receive buffers. User firmware may associate more than one filter to any one specific receive buffer.

In addition to dynamic filter to buffer association, in Mode 1 and 2, each filter can also be dynamically associated to available acceptance mask registers. FILn_m bits in the MSELn register can be used to link a specific acceptance filter to an acceptance mask register. As with filter to buffer association, one can also associate more than one mask to a specific acceptance filter.

When a filter matches and a message is loaded into the receive buffer, the filter number that enabled the message reception is loaded into the FILHIT bit(s). In Mode 0 for RXB1, the RXB1CON register contains the FILHIT<2:0> bits. They are coded as follows:

- `101` = Acceptance Filter 5 (RXF5)
- `100` = Acceptance Filter 4 (RXF4)
- `011` = Acceptance Filter 3 (RXF3)
- `010` = Acceptance Filter 2 (RXF2)
- `001` = Acceptance Filter 1 (RXF1)
- `000` = Acceptance Filter 0 (RXF0)

Note:	'`000`' and '`001`' can only occur if the RXB0DBEN bit is set in the RXB0CON register, allowing RXB0 messages to rollover into RXB1.

The coding of the RXB0DBEN bit enables these three bits to be used similarly to the FILHIT bits and to distinguish a hit on filter RXF0 and RXF1, in either RXB0 or after a rollover into RXB1.

- `111` = Acceptance Filter 1 (RXF1)
- `110` = Acceptance Filter 0 (RXF0)
- `001` = Acceptance Filter 1 (RXF1)
- `000` = Acceptance Filter 0

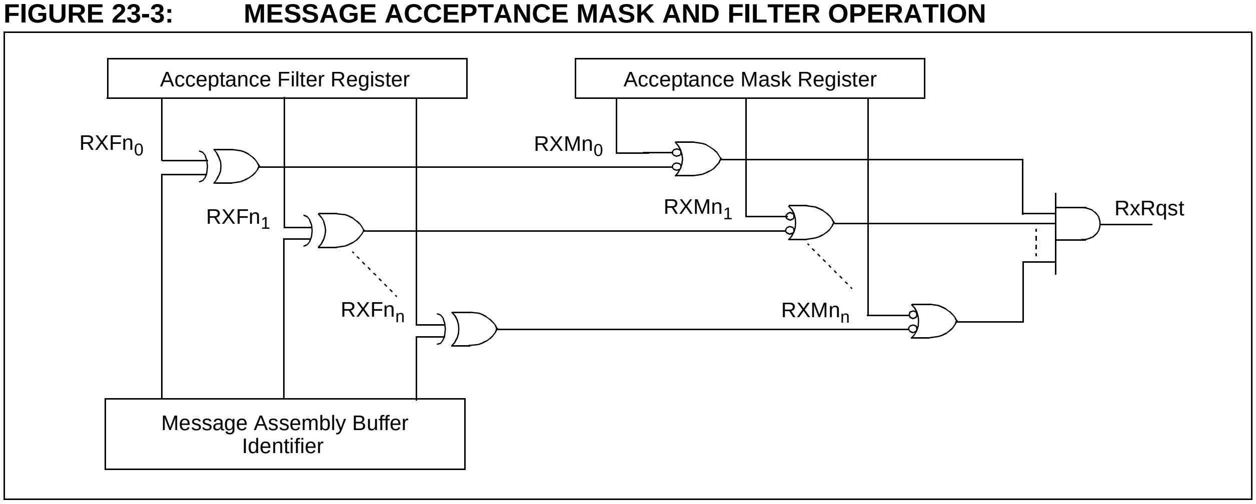
If the RXB0DBEN bit is clear, there are six codes corresponding to the six filters. If the RXB0DBEN bit is set, there are six codes corresponding to the six filters plus two additional codes corresponding to RXF0 and RXF1 filters that rollover into RXB1.

In Mode 1 and 2, each buffer control register contains 5 bits of filter hit bits FILHIT<4:0>. A binary value of '`0`' indicates a hit from RXF0 and 15 indicates RXF15.

If more than one acceptance filter matches, the FILHIT bits will encode the binary value of the lowest numbered filter that matched. In other words, if filter RXF2 and filter RXF4 match, FILHIT will be loaded with the value for RXF2. This essentially prioritizes the acceptance filters with a lower number filter having higher priority. Messages are compared to filters in ascending order of filter number.

The mask and filter registers can only be modified when the PIC18F6585/8585/6680/8680 devices are in Configuration mode.

FIGURE 23-3: MESSAGE ACCEPTANCE MASK AND FILTER OPERATION



23.9 Baud Rate Setting

All nodes on a given CAN bus must have the same nominal bit rate. The CAN protocol uses Non-Return-to-Zero (NRZ) coding which does not encode a clock within the data stream. Therefore, the receive clock must be recovered by the receiving nodes and synchronized to the transmitter's clock.

As oscillators and transmission time may vary from node to node, the receiver must have some type of Phase Lock Loop (PLL) synchronized to data transmission edges to synchronize and maintain the receiver clock. Since the data is NRZ coded, it is necessary to include bit stuffing to ensure that an edge occurs at least every six bit times to maintain the Digital Phase Lock Loop (DPLL) synchronization.

The bit timing of the PIC18F6585/8585/6680/8680 is implemented using a DPLL that is configured to synchronize to the incoming data and provides the nominal timing for the transmitted data. The DPLL breaks each bit time into multiple segments made up of minimal periods of time called the Time Quanta (TQ).

Bus timing functions executed within the bit time frame, such as synchronization to the local oscillator, network transmission delay compensation, and sample point positioning, are defined by the programmable bit timing logic of the DPLL.

All devices on the CAN bus must use the same bit rate. However, all devices are not required to have the same master oscillator clock frequency. For the different clock frequencies of the individual devices, the bit rate has to be adjusted by appropriately setting the baud rate prescaler and number of time quanta in each segment.

The Nominal Bit Rate is the number of bits transmitted per second, assuming an ideal transmitter with an ideal oscillator, in the absence of resynchronization. The nominal bit rate is defined to be a maximum of 1 Mb/s.

The Nominal Bit Time is defined as:

EQUATION 23-1:

$$T_{BIT} = 1/\text{Nominal Bit Rate}$$

The Nominal Bit Time can be thought of as being divided into separate, non-overlapping time segments. These segments (Figure 23-4) include:

- Synchronization Segment (Sync_Seg)
- Propagation Time Segment (Prop_Seg)
- Phase Buffer Segment 1 (Phase_Seg1)
- Phase Buffer Segment 2 (Phase_Seg2)

The time segments (and thus the Nominal Bit Time) are in turn made up of integer units of time called Time Quanta or TQ (see Figure 23-4). By definition, the Nominal Bit Time is programmable from a minimum of 8 TQ to a maximum of 25 TQ. Also by definition, the minimum Nominal Bit Time is 1 µs, corresponding to a maximum 1 Mb/s rate. The actual duration is given by the relationship:

EQUATION 23-2:

$$\text{Nominal Bit Time} = T_Q * (\text{Sync\_Seg} + \text{Prop\_Seg} + \text{Phase\_Seg1} + \text{Phase\_Seg2})$$

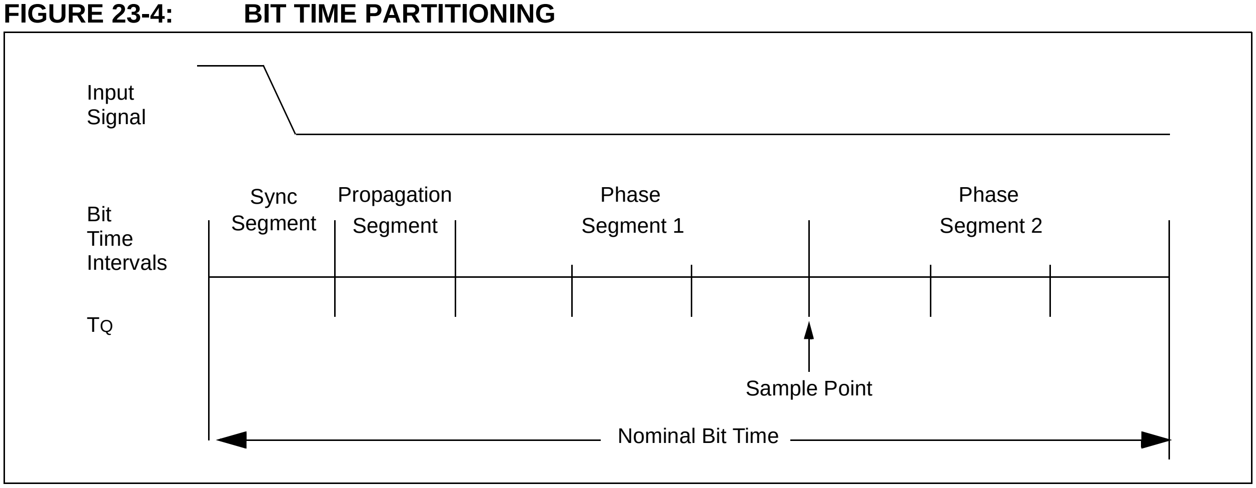
The Time Quantum is a fixed unit derived from the oscillator period. It is also defined by the programmable baud rate prescaler with integer values from 1 to 64 in addition to a fixed divide-by-two for clock generation. Mathematically, this is:

EQUATION 23-3:

$$T_Q\ (\mu s) = (2 * (BRP+1))/F_{OSC}\ (MHz)$$
or
$$T_Q\ (\mu s) = (2 * (BRP+1)) * T_{OSC}\ (\mu s)$$

where FOSC is the clock frequency, TOSC is the corresponding oscillator period, and BRP is an integer (0 through 63) represented by the binary values of BRGCON1<5:0>.

FIGURE 23-4: BIT TIME PARTITIONING



23.9.1 TIME QUANTA

As already mentioned, the Time Quanta is a fixed unit derived from the oscillator period and baud rate prescaler. Its relationship to TBIT and the Nominal Bit Rate is shown in Example 23-6.

EXAMPLE 23-6: CALCULATING TQ, NOMINAL BIT RATE AND NOMINAL BIT TIME

TQ (µs) = (2 * (BRP+1))/FOSC (MHz)

TBIT (µs) = TQ (µs) * number of TQ per bit interval

Nominal Bit Rate (bits/s) = 1/TBIT

CASE 1:

For FOSC = 16 MHz, BRP<5:0> = 00h and Nominal Bit Time = 8 TQ:

TQ = (2*1)/16 = 0.125 µs (125 ns)

TBIT = 8 * 0.125 = 1 µs (10^{-6}s)

Nominal Bit Rate = 1/10^{-6} = 10^{6} bits/s (1 Mb/s)

CASE 2:

For FOSC = 20 MHz, BRP<5:0> = 01h and Nominal Bit Time = 8 TQ:

TQ = (2*2)/20 = 0.2 µs (200 ns)

TBIT = 8 * 0.2 = 1.6 µs (1.6 * 10^{-6}s)

Nominal Bit Rate = 1/1.6 * 10^{-6}s = 625,000 bits/s (625 Kb/s)

CASE 3:

For FOSC = 25 MHz, BRP<5:0> = 3Fh and Nominal Bit Time = 25 TQ:

TQ = (2*64)/25 = 5.12 µs

TBIT = 25 * 5.12 = 128 µs (1.28 * 10^{-4}s)

Nominal Bit Rate = 1/1.28 * 10^{-4} = 7813 bits/s (7.8 Kb/s)

The frequencies of the oscillators in the different nodes must be coordinated in order to provide a system wide specified nominal bit time. This means that all oscillators must have a TOSC that is an integral divisor of TQ. It should also be noted that although the number of TQ is programmable from 4 to 25, the usable minimum is 8 TQ. A bit time of less than 8 TQ in length is not guaranteed to operate correctly.

23.9.2 SYNCHRONIZATION SEGMENT

This part of the bit time is used to synchronize the various CAN nodes on the bus. The edge of the input signal is expected to occur during the sync segment. The duration is 1 TQ.

23.9.3 PROPAGATION SEGMENT

This part of the bit time is used to compensate for physical delay times within the network. These delay times consist of the signal propagation time on the bus line and the internal delay time of the nodes. The length of the Propagation Segment can be programmed from 1 TQ to 8 TQ by setting the PRSEG2:PRSEG0 bits.

23.9.4 PHASE BUFFER SEGMENTS

The phase buffer segments are used to optimally locate the sampling point of the received bit within the nominal bit time. The sampling point occurs between Phase Segment 1 and Phase Segment 2. These segments can be lengthened or shortened by the resynchronization process. The end of Phase Segment 1 determines the sampling point within a bit time. Phase Segment 1 is programmable from 1 TQ to 8 TQ in duration. Phase Segment 2 provides delay before the next transmitted data transition and is also programmable from 1 TQ to 8 TQ in duration. However, due to IPT requirements, the actual minimum length of Phase Segment 2 is 2 TQ, or it may be defined to be equal to the greater of Phase Segment 1 or the Information Processing Time (IPT).

23.9.5 SAMPLE POINT

The sample point is the point of time at which the bus level is read and the value of the received bit is determined. The sampling point occurs at the end of Phase Segment 1. If the bit timing is slow and contains many TQ, it is possible to specify multiple sampling of the bus line at the sample point. The value of the received bit is determined to be the value of the majority decision of three values. The three samples are taken at the sample point and twice before, with a time of TQ/2 between each sample.

23.9.6 INFORMATION PROCESSING TIME

The Information Processing Time (IPT) is the time segment starting at the sample point that is reserved for calculation of the subsequent bit level. The CAN specification defines this time to be less than or equal to 2 TQ. The PIC18F6585/8585/6680/8680 devices define this time to be 2 TQ. Thus, Phase Segment 2 must be at least 2 TQ long.

23.10 Synchronization

To compensate for phase shifts between the oscillator frequencies of each of the nodes on the bus, each CAN controller must be able to synchronize to the relevant signal edge of the incoming signal. When an edge in the transmitted data is detected, the logic will compare the location of the edge to the expected time (Sync_Seg). The circuit will then adjust the values of Phase Segment 1 and Phase Segment 2 as necessary. There are two mechanisms used for synchronization.

23.10.1 HARD SYNCHRONIZATION

Hard synchronization is only done when there is a recessive to dominant edge during a bus Idle condition, indicating the start of a message. After hard synchronization, the bit time counters are restarted with Sync_Seg. Hard synchronization forces the edge which has occurred to lie within the synchronization segment of the restarted bit time. Due to the rules of synchronization, if a hard synchronization occurs there will not be a resynchronization within that bit time.

23.10.2 RESYNCHRONIZATION

As a result of resynchronization, Phase Segment 1 may be lengthened or Phase Segment 2 may be shortened. The amount of lengthening or shortening of the phase buffer segments has an upper bound given by the Synchronization Jump Width (SJW). The value of the SJW will be added to Phase Segment 1 (see Figure 23-5) or subtracted from Phase Segment 2 (see Figure 23-6). The SJW is programmable between 1 TQ and 4 TQ.

Clocking information will only be derived from recessive to dominant transitions. The property that only a fixed maximum number of successive bits have the same value, ensures resynchronization to the bit stream during a frame.

The phase error of an edge is given by the position of the edge relative to Sync_Seg, measured in TQ. The phase error is defined in magnitude of TQ as follows:

- e = 0 if the edge lies within Sync_Seg.
- e > 0 if the edge lies before the sample point.
- e < 0 if the edge lies after the sample point of the previous bit.

If the magnitude of the phase error is less than, or equal to the programmed value of the synchronization jump width, the effect of a resynchronization is the same as that of a hard synchronization.

If the magnitude of the phase error is larger than the synchronization jump width, and if the phase error is positive, then Phase Segment 1 is lengthened by an amount equal to the synchronization jump width.

If the magnitude of the phase error is larger than the resynchronization jump width, and if the phase error is negative, then Phase Segment 2 is shortened by an amount equal to the synchronization jump width.

23.10.3 SYNCHRONIZATION RULES

- Only one synchronization within one bit time is allowed.
- An edge will be used for synchronization only if the value detected at the previous sample point (previously read bus value) differs from the bus value immediately after the edge.
- All other recessive to dominant edges fulfilling rules 1 and 2 will be used for resynchronization, with the exception that a node transmitting a dominant bit will not perform a resynchronization as a result of a recessive to dominant edge with a positive phase error.

FIGURE 23-5: LENGTHENING A BIT PERIOD (ADDING SJW TO PHASE SEGMENT 1)

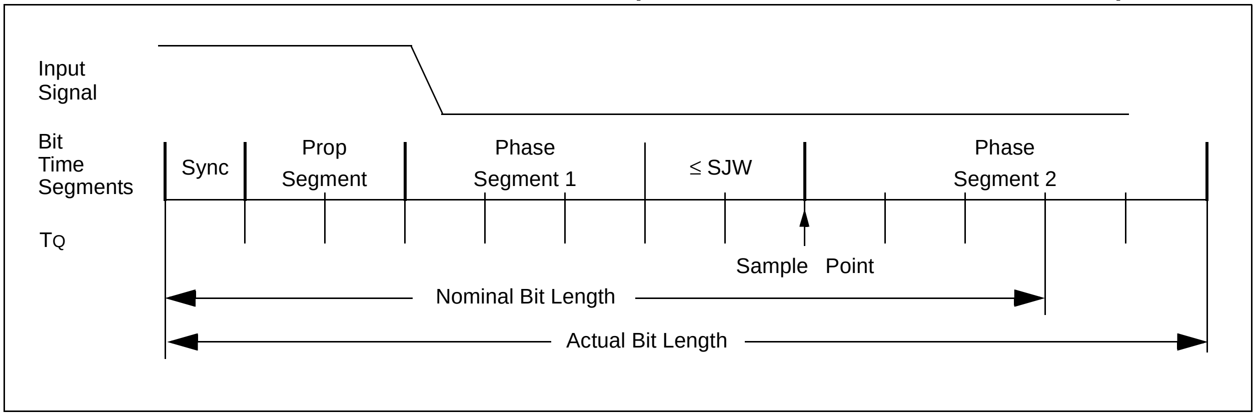
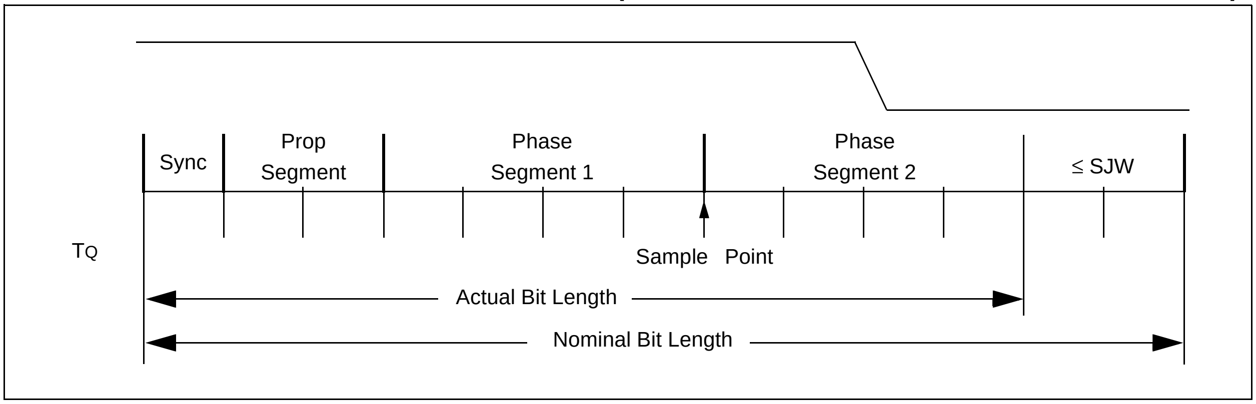


FIGURE 23-6: SHORTENING A BIT PERIOD (SUBTRACTING SJW FROM PHASE SEGMENT 2)



23.11 Programming Time Segments

Some requirements for programming of the time segments:

- Prop_Seg + Phase_Seg 1 ≥ Phase_Seg 2
- Phase_Seg 2 ≥ Sync Jump Width.

For example, assume that a 125 kHz CAN baud rate is desired, using 20 MHz for FOSC. With a TOSC of 50 ns, a baud rate prescaler value of 04h gives a TQ of 500 ns. To obtain a Nominal Bit Rate of 125 kHz, the Nominal Bit Time must be 8 µs or 16 TQ.

Using 1 TQ for the Sync_Seg, 2 TQ for the Prop_Seg and 7 TQ for Phase Segment 1, would place the sample point at 10 TQ after the transition. This leaves 6 TQ for Phase Segment 2.

By the rules above, the Sync Jump Width could be the maximum of 4 TQ. However, normally a large SJW is only necessary when the clock generation of the different nodes is inaccurate or unstable, such as using ceramic resonators. Typically, an SJW of 1 is enough.

23.12 Oscillator Tolerance

As a rule of thumb, the bit timing requirements allow ceramic resonators to be used in applications with transmission rates of up to 125 Kbit/sec. For the full bus speed range of the CAN protocol, a quartz oscillator is required. A maximum node-to-node oscillator variation of 1.7% is allowed.

23.13 Bit Timing Configuration Registers

The Configuration registers (BRGCON1, BRGCON2, BRGCON3) control the bit timing for the CAN bus interface. These registers can only be modified when the PIC18F6585/8585/6680/8680 devices are in Configuration mode.

23.13.1 BRGCON1

The BRP bits control the baud rate prescaler. The SJW<1:0> bits select the synchronization jump width in terms of multiples of TQ.

23.13.2 BRGCON2

The PRSEG bits set the length of the propagation segment in terms of TQ. The SEG1PH bits set the length of Phase Segment 1 in TQ. The SAM bit controls how many times the RXCAN pin is sampled. Setting this bit to a '1' causes the bus to be sampled three times; twice at TQ/2 before the sample point and once at the normal sample point (which is at the end of Phase Segment 1). The value of the bus is determined to be the value read during at least two of the samples. If the SAM bit is set to a '0', then the RXCAN pin is sampled only once at the sample point. The SEG2PHTS bit controls how the length of Phase Segment 2 is determined. If this bit is set to a '1', then the length of Phase Segment 2 is determined by the SEG2PH bits of BRGCON3. If the SEG2PHTS bit is set to a '0', then the length of Phase Segment 2 is the greater of Phase Segment 1 and the information processing time (which is fixed at 2 TQ for the PIC18F6585/8585/6680/8680).

23.13.3 BRGCON3

The PHSEG2<2:0> bits set the length (in TQ) of Phase Segment 2 if the SEG2PHTS bit is set to a '1'. If the SEG2PHTS bit is set to a '0', then the PHSEG2<2:0> bits have no effect.

23.14 Error Detection

The CAN protocol provides sophisticated error detection mechanisms. The following errors can be detected.

23.14.1 CRC ERROR

With the Cyclic Redundancy Check (CRC), the transmitter calculates special check bits for the bit sequence, from the start of a frame until the end of the data field. This CRC sequence is transmitted in the CRC field. The receiving node also calculates the CRC sequence using the same formula and performs a comparison to the received sequence. If a mismatch is detected, a CRC error has occurred and an error frame is generated. The message is repeated.

23.14.2 ACKNOWLEDGE ERROR

In the Acknowledge field of a message, the transmitter checks if the Acknowledge slot (which was sent out as a recessive bit) contains a dominant bit. If not, no other node has received the frame correctly. An Acknowledge error has occurred; an error frame is generated and the message will have to be repeated.

23.14.3 FORM ERROR

If a node detects a dominant bit in one of the four segments, including end of frame, interframe space, Acknowledge delimiter, or CRC delimiter, then a form error has occurred and an error frame is generated. The message is repeated.

23.14.4 BIT ERROR

A bit error occurs if a transmitter sends a dominant bit and detects a recessive bit, or if it sends a recessive bit and detects a dominant bit, when monitoring the actual bus level and comparing it to the just transmitted bit. In the case where the transmitter sends a recessive bit and a dominant bit is detected during the arbitration field and the Acknowledge slot, no bit error is generated because normal arbitration is occurring.

23.14.5 STUFF BIT ERROR

lf between the start of frame and the CRC delimiter, six consecutive bits with the same polarity are detected, the bit stuffing rule has been violated. A stuff bit error occurs and an error frame is generated. The message is repeated.

23.14.6 ERROR STATES

Detected errors are made public to all other nodes via error frames. The transmission of the erroneous message is aborted and the frame is repeated as soon as possible. Furthermore, each CAN node is in one of the three error states “error-active”, “error-passive” or “bus-off” according to the value of the internal error counters. The error-active state is the usual state where the bus node can transmit messages and activate error frames (made of dominant bits) without any restrictions. In the error-passive state, messages and passive error frames (made of recessive bits) may be transmitted. The bus-off state makes it temporarily impossible for the station to participate in the bus communication. During this state, messages can neither be received nor transmitted.

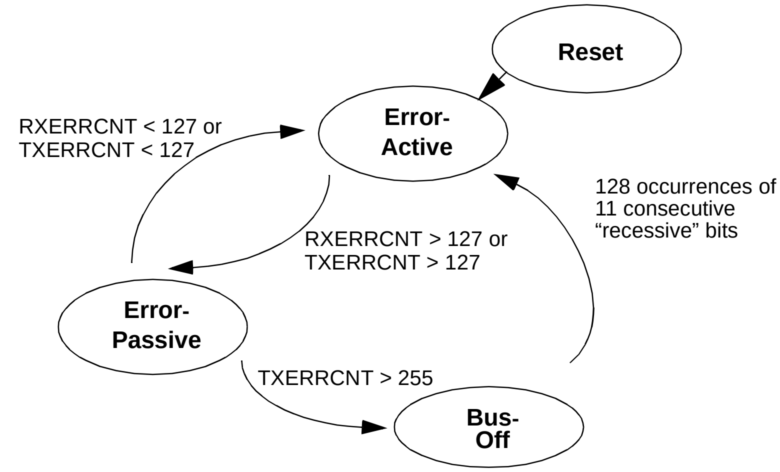
23.14.7 ERROR MODES AND ERROR COUNTERS

The PIC18F6585/8585/6680/8680 devices contain two error counters: the Receive Error Counter (RXERRCNT), and the Transmit Error Counter (TXERRCNT). The values of both counters can be read by the MCU. These counters are incremented or decremented in accordance with the CAN bus specification.

The PIC18F6585/8585/6680/8680 devices are error-active if both error counters are below the error-passive limit of 128. They are error-passive if at least one of the error counters equals or exceeds 128. They go to bus-off if the transmit error counter equals or exceeds the bus-off limit of 256. The devices remain in this state until the bus-off recovery sequence is received. The bus-off recovery sequence consists of 128 occurrences of 11 consecutive recessive bits (see Figure 23-7). Note that the CAN module, after going bus-off, will recover back to error-active without any intervention by the MCU if the bus remains Idle for 128 x 11 bit times. If this is not desired, the error Interrupt Service Routine should address this. The current Error mode of the CAN module can be read by the MCU via the COMSTAT register.

Additionally, there is an error state warning flag bit, EWARN, which is set if at least one of the error counters equals or exceeds the error warning limit of 96. EWARN is reset if both error counters are less than the error warning limit.

FIGURE 23-7: ERROR MODES STATE DIAGRAM



23.15 CAN Interrupts

The module has several sources of interrupts. Each of these interrupts can be individually enabled or disabled. The PIR3 register contains interrupt flags. The PIE3 register contains the enables for the 8 main interrupts. A special set of read-only bits in the CANSTAT register, the ICODE bits, can be used in combination with a jump table for efficient handling of interrupts.

All interrupts have one source with the exception of the error interrupt and buffer interrupts in Mode 1 and 2. Any of the error interrupt sources can set the error interrupt flag. The source of the error interrupt can be determined by reading the Communication Status register, COMSTAT. In Mode 1 and 2, there are two interrupt enable/disable and flag bits – one for all transmit buffers and the other for all receive buffers.

The interrupts can be broken up into two categories: receive and transmit interrupts.

The receive related interrupts are:

- Receive Interrupts
- Wake-up Interrupt
- Receiver Overrun Interrupt
- Receiver Warning Interrupt
- Receiver Error-Passive Interrupt

The transmit related interrupts are:

- Transmit Interrupts
- Transmitter Warning Interrupt
- Transmitter Error-Passive Interrupt
- Bus-Off Interrupt

23.15.1 INTERRUPT CODE BITS

To simplify the interrupt handling process in user firmware, the ECAN module encodes a special set of bits. In Mode 0, these bits are ICODE<2:0> in the CANSTAT register. In Mode 1 and 2, these bits are EICODE<3:0> in the CANSTAT register. Interrupts are internally prioritized such that the higher priority interrupts are assigned lower values. Once the highest priority interrupt condition has been cleared, the code for the next highest priority interrupt that is pending (if any) will be reflected by the ICODE bits. Note that only those interrupt sources that have their associated interrupt enable bit set will be reflected in the ICODE bits.

In Mode 2, when a receive message interrupt occurs, EICODE bits will always consist of ‘`10000`’. User firmware may use FIFO pointer bits to actually access the next available buffer.

23.15.2 TRANSMIT INTERRUPT

When the transmit interrupt is enabled, an interrupt will be generated when the associated transmit buffer becomes empty and is ready to be loaded with a new message. In Mode 0, there are separate interrupt enable/disable and flag bits for each of the three dedicated transmit buffers. The TXBnIF bit will be set to indicate the source of the interrupt. The interrupt is cleared by the MCU resetting the TXBnIF bit to a '0'. In Mode 1 and 2, all transmit buffers share one interrupt enable/disable and flag bits. In Mode 1 and 2, TXBIE in PIE3 and TXBIF in PIR3 indicate when a transmit buffer has completed transmission of its message. TXBnIF, TXBnIE and TXBnIP in PIR3, PIE3 and IPR3, respectively, are not used in Mode 1 and 2. Individual transmit buffer interrupts can be enabled or disabled by setting or clearing TXBIE and BnIE register bits. When a shared interrupt occurs, user firmware must poll the TXREQ bit of all transmit buffers to detect the source of interrupt.

23.15.3 RECEIVE INTERRUPT

When the receive interrupt is enabled, an interrupt will be generated when a message has been successfully received and loaded into the associated receive buffer. This interrupt is activated immediately after receiving the End Of Frame (EOF) field.

In Mode 0, the RXBnIF bit is set to indicate the source of the interrupt. The interrupt is cleared by the MCU resetting the RXBnIF bit to a '0'.

In Mode 1 and 2, all receive buffers share one interrupt. Individual receive buffer interrupts can be controlled by the RXBnIE and BIEn registers. In Mode 1, when a shared receive interrupt occurs, user firmware must poll the RXFUL bit of each receive buffer to detect the source of interrupt. In Mode 2, a receive interrupt indicates that the new message is loaded into FIFO. FIFO can be read by using FIFO pointer bits, FP.

In Mode 2, the FIFOWMIF bit indicates if the FIFO high watermark is reached. The FIFO high watermark is defined by the FIFOWM bit in the ECANCON register.

23.15.4 MESSAGE ERROR INTERRUPT

When an error occurs during transmission or reception of a message, the message error flag, IRXIF, will be set and if the IRXIE bit is set, an interrupt will be generated. This is intended to be used to facilitate baud rate determination when used in conjunction with Listen Only mode.

23.15.5 BUS ACTIVITY WAKE-UP INTERRUPT

When the PIC18F6585/8585/6680/8680 devices are in Sleep mode and the bus activity wake-up interrupt is enabled, an interrupt will be generated and the WAKIF bit will be set when activity is detected on the CAN bus. This interrupt causes the PIC18F6585/8585/6680/8680 devices to exit Sleep mode. The interrupt is reset by the MCU, clearing the WAKIF bit.

23.15.6 ERROR INTERRUPT

When the error interrupt is enabled, an interrupt is generated if an overflow condition occurs or if the error state of the transmitter or receiver has changed. The error flags in COMSTAT will indicate one of the following conditions.

23.15.6.1 Receiver Overflow

An overflow condition occurs when the MAB has assembled a valid received message (the message meets the criteria of the acceptance filters) and the receive buffer associated with the filter is not available for loading of a new message. The associated COMSTAT.RXnOVFL bit will be set to indicate the overflow condition. This bit must be cleared by the MCU.

23.15.6.2 Receiver Warning

The receive error counter has reached the MCU warning limit of 96.

23.15.6.3 Transmitter Warning

The transmit error counter has reached the MCU warning limit of 96.

23.15.6.4 Receiver Bus Passive

The receive error counter has exceeded the error-passive limit of 127 and the device has gone to error-passive state.

23.15.6.5 Transmitter Bus Passive

The transmit error counter has exceeded the error-passive limit of 127 and the device has gone to error-passive state.

23.15.6.6 Bus-Off

The transmit error counter has exceeded 255 and the device has gone to bus-off state.

23.15.6.7 Interrupt Acknowledge

Interrupts are directly associated with one or more status flags in the PIR register. Interrupts are pending as long as one of the flags is set. Once an interrupt flag is set by the device, the flag can not be reset by the microcontroller until the interrupt condition is removed.

NOTES:

24.0 SPECIAL FEATURES OF THE CPU

There are several features intended to maximize system reliability, minimize cost through elimination of external components, provide power saving operating modes and offer code protection. These are:

- OSC Selection
- Reset
  - Power-on Reset (POR)
  - Power-up Timer (PWRT)
  - Oscillator Start-up Timer (OST)
  - Brown-out Reset (BOR)
- Interrupts
- Watchdog Timer (WDT)
- Sleep
- Code Protection
- ID Locations
- In-Circuit Serial Programming

All PIC18F6585/8585/6680/8680 devices have a Watchdog Timer which is permanently enabled via the configuration bits or software controlled. It runs off its own RC oscillator for added reliability. There are two timers that offer necessary delays on power-up. One is the Oscillator Start-up Timer (OST), intended to keep the chip in Reset until the crystal oscillator is stable. The other is the Power-up Timer (PWRT) which provides a fixed delay on power-up only, designed to keep the part in Reset while the power supply stabilizes. With these two timers on-chip, most applications need no external Reset circuitry.

Sleep mode is designed to offer a very low current Power-down mode. The user can wake-up from Sleep through external Reset, Watchdog Timer Wake-up, or through an interrupt. Several oscillator options are also made available to allow the part to fit the application. The RC oscillator option saves system cost, while the LP crystal option saves power. A set of configuration bits is used to select various options.

24.1 Configuration Bits

The configuration bits can be programmed (read as '0') or left unprogrammed (read as '1') to select various device configurations. These bits are mapped, starting at program memory location 300000h.

The user will note that address 300000h is beyond the user program memory space. In fact, it belongs to the configuration memory space (300000h through 3FFFFFh) which can only be accessed using table reads and table writes.

Programming the Configuration registers is done in a manner similar to programming the Flash memory. The EECON1 register WR bit starts a self-timed write to the Configuration register. In normal Operation mode, a `TBLWT` instruction with the TBLPTR pointed to the Configuration register sets up the address and the data for the Configuration register write. Setting the WR bit starts a long write to the Configuration register. The Configuration registers are written a byte at a time. To write or erase a configuration cell, a `TBLWT` instruction can write a '1' or a '0' into the cell.

TABLE 24-1: CONFIGURATION BITS AND DEVICE IDS

File Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Default/ Unprogrammed Value
300001h	CONFIG1H	—	—	$\overline{\text{OSCSEN}}$	—	FOSC3	FOSC2	FOSC1	FOSC0	`--1- 1111`
300002h	CONFIG2L	—	—	—	—	BORV1	BORV0	BODEN	$\overline{\text{PWRTEN}}$	`---- 1111`
300003h	CONFIG2H	—	—	—	WDTPS3	WDTPS2	WDTPS1	WDTPS0	WDTEN	`---1 1111`
300004h(1)	CONFIG3L	WAIT	—	—	—	—	—	PM1	PM0	`1--- --11`
300005h	CONFIG3H	MCLRE	—	—	—	—	—	ECCPMX(4)	CCP2MX	`1--- --11`
300006h	CONFIG4L	$\overline{\text{DEBUG}}$	—	—	—	—	LVP	—	STVREN	`1--- -1-1`
300008h	CONFIG5L	—	—	—	—	CP3(2)	CP2	CP1	CP0	`---- 1111`
300009h	CONFIG5H	CPD	CPB	—	—	—	—	—	—	`11-- ----`
30000Ah	CONFIG6L	—	—	—	—	WRT3(2)	WRT2	WRT1	WRT0	`---- 1111`
30000Bh	CONFIG6H	WRTD	WRTB	WRTC	—	—	—	—	—	`111- ----`
30000Ch	CONFIG7L	—	—	—	—	EBTR3(2)	EBTR2	EBTR1	EBTR0	`---- 1111`
30000Dh	CONFIG7H	—	EBTRB	—	—	—	—	—	—	`-1-- ----`
3FFFFEh	DEVID1	DEV2	DEV1	DEV0	REV4	REV3	REV2	REV1	REV0	**(Note 3)**
3FFFFFh	DEVID2	DEV10	DEV9	DEV8	DEV7	DEV6	DEV5	DEV4	DEV3	`0000 1010`

Legend: `x` = unknown, `u` = unchanged, `-` = unimplemented, `q` = value depends on condition.
Shaded cells are unimplemented, read as '`0`'.

Note 1: Unimplemented in PIC18F6X8X devices; maintain this bit set.
2: Unimplemented in PIC18FX585 devices; maintain this bit set.
3: See Register 24-13 for DEVID1 values.
4: Reserved in PIC18F6X8X devices; maintain this bit set.

REGISTER 24-1: CONFIG1H: CONFIGURATION REGISTER 1 HIGH (BYTE ADDRESS 300001h)

U-0	U-0	R/P-1	U-0	R/P-1	R/P-1	R/P-1	R/P-1
—	—	$\overline{\text{OSCSEN}}$	—	FOSC3	FOSC2	FOSC1	FOSC0
bit 7							bit 0

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

bit 5 **$\overline{\text{OSCSEN}}$**: Oscillator System Clock Switch Enable bit
1 = Oscillator system clock switch option is disabled (main oscillator is source)
0 = Timer1 oscillator system clock switch option is enabled (oscillator switching is enabled)

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

bit 3-0 **FOSC3:FOSC0**: Oscillator Selection bits
1111 = RC oscillator with OSC2 configured as RA6
1110 = HS oscillator with SW enabled 4x PLL
1101 = EC oscillator with OSC2 configured as RA6 and SW enabled 4x PLL
1100 = EC oscillator with OSC2 configured as RA6 and HW enabled 4x PLL
1011 = Reserved; do not use
1010 = Reserved; do not use
1001 = Reserved; do not use
1000 = Reserved; do not use
0111 = RC oscillator with OSC2 configured as RA6
0110 = HS oscillator with HW enabled 4x PLL
0101 = EC oscillator with OSC2 configured as RA6
0100 = EC oscillator with OSC2 configured as divide by 4 clock output
0011 = RC oscillator with OSC2 configured as divide by 4 clock output
0010 = HS oscillator
0001 = XT oscillator
0000 = LP oscillator

Legend:		
R = Readable bit	P = Programmable bit	U = Unimplemented bit, read as '0'
- n = Value when device is unprogrammed		u = Unchanged from programmed state

REGISTER 24-2: CONFIG2L: CONFIGURATION REGISTER 2 LOW (BYTE ADDRESS 300002h)

U-0	U-0	U-0	U-0	R/P-1	R/P-1	R/P-1	R/P-1
—	—	—	—	BORV1	BORV0	BOREN	$\overline{\text{PWRTEN}}$
bit 7							bit 0

bit 7-4 **Unimplemented:** Read as ‘0’

bit 3-2 **BORV1:BORV0:** Brown-out Reset Voltage bits
11 = V_{BOR} set to 2.0V
10 = V_{BOR} set to 2.7V
01 = V_{BOR} set to 4.2V
00 = V_{BOR} set to 4.5V

bit 1 **BOREN:** Brown-out Reset Enable bit
1 = Brown-out Reset enabled
0 = Brown-out Reset disabled

bit 0 **$\overline{\text{PWRTEN}}$:** Power-up Timer Enable bit
1 = PWRT disabled
0 = PWRT enabled

Legend:		
R = Readable bit	P = Programmable bit	U = Unimplemented bit, read as ‘0’
- n = Value when device is unprogrammed		u = Unchanged from programmed state

REGISTER 24-3: CONFIG2H: CONFIGURATION REGISTER 2 HIGH (BYTE ADDRESS 300003h)

U-0	U-0	U-0	R/P-1	R/P-1	R/P-1	R/P-1	R/P-1
—	—	—	WDTPS3	WDTPS2	WDTPS1	WDTPS0	WDTEN
bit 7							bit 0

bit 7-5 **Unimplemented:** Read as '0'

bit 4-1 **WDTPS3:WDTPS0:** Watchdog Timer Postscaler Select bits

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

bit 0 **WDTEN:** Watchdog Timer Enable bit

1 = WDT enabled
0 = WDT disabled (control is placed on the SWDTEN bit)

Legend:		
R = Readable bit	P = Programmable bit	U = Unimplemented bit, read as '0'
- n = Value when device is unprogrammed		u = Unchanged from programmed state

REGISTER 24-4: CONFIG3L: CONFIGURATION REGISTER 3 LOW (BYTE ADDRESS 300004h)[(1)]

R/P-1	U-0	U-0	U-0	U-0	U-0	R/P-1	R/P-1
WAIT	—	—	—	—	—	PM1	PM0
bit 7							bit 0

bit 7 **WAIT:** External Bus Data Wait Enable bit

1 = Wait selections unavailable for table reads and table writes
0 = Wait selections for table reads and table writes are determined by WAIT1:WAIT0 bits (MEMCOM<5:4>)

bit 6-2 **Unimplemented:** Read as '0'

bit 1-0 **PM1:PM0:** Processor Mode Select bits

11 = Microcontroller mode
10 = Microprocessor mode
01 = Microprocessor with Boot Block mode
00 = Extended Microcontroller mode

Note 1: This register is unimplemented for PIC18F6X8X devices; maintain these bits set.

Legend:		
R = Readable bit	P = Programmable bit	U = Unimplemented bit, read as '0'
- n = Value when device is unprogrammed		u = Unchanged from programmed state

REGISTER 24-5: CONFIG3H: CONFIGURATION REGISTER 3 HIGH (BYTE ADDRESS 300005h)

R/P-1	U-0	U-0	U-0	U-0	U-0	R/P-1	R/P-1
MCLRE	—	—	—	—	—	ECCPMX	CCP2MX
bit 7							bit 0

bit 7 **MCLRE:** $\overline{\text{MCLR}}$ Enable bit[1]
1 = $\overline{\text{MCLR}}$ pin enabled, RG5 input pin disabled
0 = RG5 input enabled, $\overline{\text{MCLR}}$ disabled

bit 6-2 **Unimplemented:** Read as '0'

bit 1 **ECCPMX:** CCP1 PWM outputs P1B, P1C mux bit (PIC18F8X8X devices only)[2]
1 = P1B, P1C are multiplexed with RE6, RE5
0 = P1B, P1C are multiplexed with RH7, RH6

bit 0 **CCP2MX:** CCP2 Mux bit
In Microcontroller mode:
1 = CCP2 input/output is multiplexed with RC1
0 = CCP2 input/output is multiplexed with RE7
In Microprocessor, Microprocessor with Boot Block and Extended Microcontroller modes (PIC18F8X8X devices only):
1 = CCP2 input/output is multiplexed with RC1
0 = CCP2 input/output is multiplexed with RB3

Note 1: If $\overline{\text{MCLR}}$ is disabled, either disable low-voltage ICSP or hold RB5/PGM low to ensure proper entry into ICSP mode.
2: Reserved for PIC18F6X8X devices; maintain this bit set.

Legend:		
R = Readable bit	P = Programmable bit	U = Unimplemented bit, read as '0'
- n = Value when device is unprogrammed		u = Unchanged from programmed state

REGISTER 24-6: CONFIG4L: CONFIGURATION REGISTER 4 LOW (BYTE ADDRESS 300006h)

R/P-1	U-0	U-0	U-0	U-0	R/P-1	U-0	R/P-1
DEBUG	—	—	—	—	LVP	—	STVREN
bit 7							bit 0

bit 7 **$\overline{\text{DEBUG}}$:** Background Debugger Enable bit
1 = Background debugger disabled. RB6 and RB7 configured as general purpose I/O pins.
0 = Background debugger enabled. RB6 and RB7 are dedicated to in-circuit debug.

bit 6-3 **Unimplemented:** Read as '0'

bit 2 **LVP:** Low-Voltage ICSP Enable bit
1 = Low-voltage ICSP enabled
0 = Low-voltage ICSP disabled

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

bit 0 **STVREN:** Stack Full/Underflow Reset Enable bit
1 = Stack full/underflow will cause Reset
0 = Stack full/underflow will not cause Reset

Legend:		
R = Readable bit	P = Programmable bit	U = Unimplemented bit, read as '0'
- n = Value when device is unprogrammed		u = Unchanged from programmed state

REGISTER 24-7: CONFIG5L: CONFIGURATION REGISTER 5 LOW (BYTE ADDRESS 300008h)

U-0	U-0	U-0	U-0	R/C-1	R/C-1	R/C-1	R/C-1
—	—	—	—	CP3[(1)]	CP2	CP1	CP0
bit 7							bit 0

bit 7-4 **Unimplemented:** Read as '0'

bit 3 **CP3:** Code Protection bit[(1)]
1 = Block 3 (00C000-00FFFFh) not code-protected
0 = Block 3 (00C000-00FFFFh) code-protected

Note 1: Unimplemented in PIC18FX585 devices; maintain this bit set.

bit 2 **CP2:** Code Protection bit
1 = Block 2 (008000-00BFFFh) not code-protected
0 = Block 2 (008000-00BFFFh) code-protected

bit 1 **CP1:** Code Protection bit
1 = Block 1 (004000-007FFFh) not code-protected
0 = Block 1 (004000-007FFFh) code-protected

bit 0 **CP0:** Code Protection bit
1 = Block 0 (000800-003FFFh) not code-protected
0 = Block 0 (000800-003FFFh) code-protected

Legend:		
R = Readable bit	C = Clearable bit	U = Unimplemented bit, read as '0'
- n = Value when device is unprogrammed		u = Unchanged from programmed state

REGISTER 24-8: CONFIG5H: CONFIGURATION REGISTER 5 HIGH (BYTE ADDRESS 300009h)

R/C-1	R/C-1	U-0	U-0	U-0	U-0	U-0	U-0
CPD	CPB	—	—	—	—	—	—
bit 7							bit 0

bit 7 **CPD:** Data EEPROM Code Protection bit
1 = Data EEPROM not code-protected
0 = Data EEPROM code-protected

bit 6 **CPB:** Boot Block Code Protection bit
1 = Boot block (000000-0007FFh) not code-protected
0 = Boot block (000000-0007FFh) code-protected

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

Legend:		
R = Readable bit	C = Clearable bit	U = Unimplemented bit, read as '0'
- n = Value when device is unprogrammed		u = Unchanged from programmed state

REGISTER 24-9: CONFIG6L: CONFIGURATION REGISTER 6 LOW (BYTE ADDRESS 30000Ah)

U-0	U-0	U-0	U-0	R/C-1	R/C-1	R/C-1	R/C-1
—	—	—	—	WRT3(1)	WRT2	WRT1	WRT0
bit 7							bit 0

bit 7-4 **Unimplemented:** Read as '0'

bit 3 **WRT3:** Write Protection bit(1)
1 = Block 3 (00C000-00FFFFh) not write-protected
0 = Block 3 (00C000-00FFFFh) write-protected

Note 1: Unimplemented in PIC18FX585 devices; maintain this bit set.

bit 2 **WRT2:** Write Protection bit
1 = Block 2 (008000-00BFFFh) not write-protected
0 = Block 2 (008000-00BFFFh) write-protected

bit 1 **WRT1:** Write Protection bit
1 = Block 1 (004000-007FFFh) not write-protected
0 = Block 1 (004000-007FFFh) write-protected

bit 0 **WR0:** Write Protection bit
1 = Block 0 (000800-003FFFh) not write-protected
0 = Block 0 (000800-003FFFh) write-protected

Legend:		
R = Readable bit	P = Programmable bit	U = Unimplemented bit, read as '0'
- n = Value when device is unprogrammed		u = Unchanged from programmed state

REGISTER 24-10: CONFIG6H: CONFIGURATION REGISTER 6 HIGH (BYTE ADDRESS 30000Bh)

R/C-1	R/C-1	R/C-1	U-0	U-0	U-0	U-0	U-0
WRTD	WRTB	WRTC	—	—	—	—	—
bit 7							bit 0

bit 7 **WRTD:** Data EEPROM Write Protection bit
1 = Data EEPROM not write-protected
0 = Data EEPROM write-protected

bit 6 **WRTB:** Boot Block Write Protection bit
1 = Boot block (000000-0007FFh) not write-protected
0 = Boot block (000000-0007FFh) write-protected

bit 5 **WRTC:** Configuration Register Write Protection bit
1 = Configuration registers (300000-3000FFh) not write-protected
0 = Configuration registers (300000-3000FFh) write-protected

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

Legend:		
R = Readable bit	P = Programmable bit	U = Unimplemented bit, read as '0'
- n = Value when device is unprogrammed		u = Unchanged from programmed state

REGISTER 24-11: CONFIG7L: CONFIGURATION REGISTER 7 LOW (BYTE ADDRESS 30000Ch)

U-0	U-0	U-0	U-0	R/C-1	R/C-1	R/C-1	R/C-1
—	—	—	—	EBTR3(1)	EBTR2	EBTR1	EBTR0
bit 7							bit 0

bit 7-4 **Unimplemented:** Read as '0'

bit 3 **EBTR3:** Table Read Protection bit(1)
1 = Block 3 (00C000-00FFFFh) not protected from table reads executed in other blocks
0 = Block 3 (00C000-00FFFFh) protected from table reads executed in other blocks

Note 1: Unimplemented in PIC18FX585 devices; maintain this bit set.

bit 2 **EBTR2:** Table Read Protection bit
1 = Block 2 (008000-00BFFFh) not protected from table reads executed in other blocks
0 = Block 2 (008000-00BFFFh) protected from table reads executed in other blocks

bit 1 **EBTR1:** Table Read Protection bit
1 = Block 1 (004000-007FFFh) not protected from table reads executed in other blocks
0 = Block 1 (004000-007FFFh) protected from table reads executed in other blocks

bit 0 **EBTR0:** Table Read Protection bit
1 = Block 0 (000800-003FFFh) not protected from table reads executed in other blocks
0 = Block 0 (000800-003FFFh) protected from table reads executed in other blocks

Legend:		
R = Readable bit	P = Programmable bit	U = Unimplemented bit, read as '0'
- n = Value when device is unprogrammed		u = Unchanged from programmed state

REGISTER 24-12: CONFIG7H: CONFIGURATION REGISTER 7 HIGH (BYTE ADDRESS 30000Dh)

U-0	R/C-1	U-0	U-0	U-0	U-0	U-0	U-0
—	EBTRB	—	—	—	—	—	—
bit 7							bit 0

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

bit 6 **EBTRB:** Boot Block Table Read Protection bit
1 = Boot block (000000-0007FFh) not protected from table reads executed in other blocks
0 = Boot block (000000-0007FFh) protected from table reads executed in other blocks

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

Legend:		
R = Readable bit	P = Programmable bit	U = Unimplemented bit, read as '0'
- n = Value when device is unprogrammed		u = Unchanged from programmed state

REGISTER 24-13: DEVICE ID REGISTER 1 FOR PIC18FXX8X DEVICES (ADDRESS 3FFFFEh)

R	R	R	R	R	R	R	R
DEV2	DEV1	DEV0	REV4	REV3	REV2	REV1	REV0
bit 7							bit 0

bit 7-5 **DEV2:DEV0:** Device ID bits

000 = PIC18F8680
001 = PIC18F6680
010 = PIC18F8585
011 = PIC18F6585

bit 4-0 **REV4:REV0:** Revision ID bits

These bits are used to indicate the device revision.

Legend:		
R = Readable bit	P = Programmable bit	U = Unimplemented bit, read as ‘0’
- n = Value when device is unprogrammed		u = Unchanged from programmed state

REGISTER 24-14: DEVICE ID REGISTER 2 FOR PIC18FXX8X DEVICES (ADDRESS 3FFFFFh)

R-0	R-0	R-0	R-0	R-1	R-0	R-1	R-0
DEV10	DEV9	DEV8	DEV7	DEV6	DEV5	DEV4	DEV3
bit 7							bit 0

bit 7-0 **DEV10:DEV3:** Device ID bits

These bits are used with the DEV2:DEV0 bits in the Device ID Register 1 to identify the part number.

0000 1010 = PIC18F6585/8585/6680/8680

Legend:		
R = Readable bit	P = Programmable bit	U = Unimplemented bit, read as ‘0’
- n = Value when device is unprogrammed		u = Unchanged from programmed state

24.2 Watchdog Timer (WDT)

The Watchdog Timer is a free-running, on-chip RC oscillator which does not require any external components. This RC oscillator is separate from the RC oscillator of the OSC1/CLKI pin. That means that the WDT will run even if the clock on the OSC1/CLKI and OSC2/CLKO/RA6 pins of the device has been stopped, for example, by execution of a `SLEEP` instruction.

During normal operation, a WDT time-out generates a device Reset (Watchdog Timer Reset). If the device is in Sleep mode, a WDT time-out causes the device to wake-up and continue with normal operation (Watchdog Timer wake-up). The $\overline{TO}$ bit in the RCON register will be cleared upon a WDT time-out.

The Watchdog Timer is enabled/disabled by a device configuration bit. If the WDT is enabled, software execution may not disable this function. When the WDTEN configuration bit is cleared, the SWDTEN bit enables/disables the operation of the WDT.

The WDT time-out period values may be found in **Section 27.0 "Electrical Characteristics"** under parameter #31. Values for the WDT postscaler may be assigned using the configuration bits.

> **Note 1:** The `CLRWDT` and `SLEEP` instructions clear the WDT and the postscaler if assigned to the WDT and prevent it from timing out and generating a device Reset condition.
>
> **2:** When a `CLRWDT` instruction is executed and the postscaler is assigned to the WDT, the postscaler count will be cleared but the postscaler assignment is not changed.

24.2.1 CONTROL REGISTER

Register 24-15 shows the WDTCON register. This is a readable and writable register which contains a control bit that allows software to override the WDT enable configuration bit, only when the configuration bit has disabled the WDT.

REGISTER 24-15: WDTCON REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	R/W-0
—	—	—	—	—	—	—	SWDTEN
bit 7							bit 0

bit 7-1 **Unimplemented**: Read as '0'

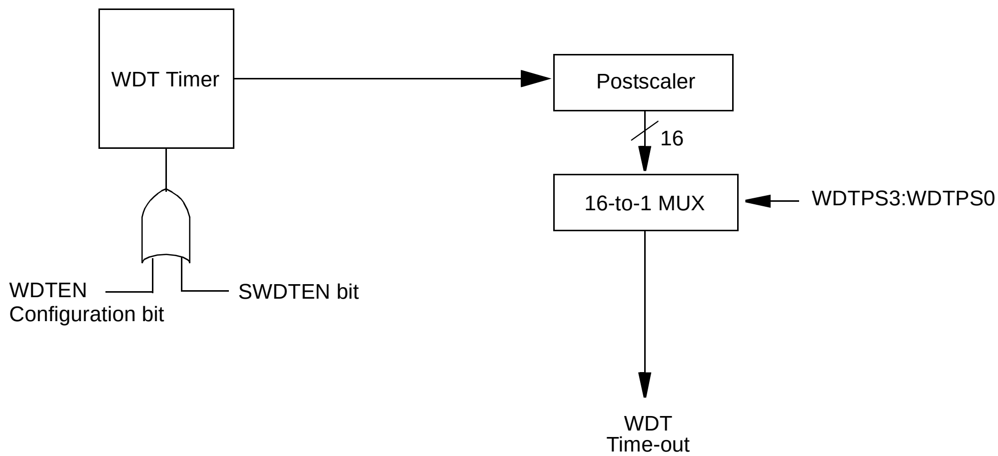
bit 0 **SWDTEN:** Software Controlled Watchdog Timer Enable bit
1 = Watchdog Timer is on
0 = Watchdog Timer is turned off if the WDTEN configuration bit in the Configuration register = 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

24.2.2 WDT POSTSCALER

The WDT has a postscaler that can extend the WDT Reset period. The postscaler is selected at the time of the device programming by the value written to the CONFIG2H Configuration register.

FIGURE 24-1: WATCHDOG TIMER BLOCK DIAGRAM



Note: WDPS3:WDPS0 are bits in register CONFIG2H.

TABLE 24-2: SUMMARY OF WATCHDOG TIMER REGISTERS

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
CONFIG2H	—	—	—	WDTPS3	WDTPS2	WDTPS2	WDTPS0	WDTEN
RCON	IPEN	—	—	$\overline{RI}$	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	$\overline{BOR}$
WDTCON	—	—	—	—	—	—	—	SWDTEN

Legend: Shaded cells are not used by the Watchdog Timer.

24.3 Power-down Mode (Sleep)

Power-down mode is entered by executing a `SLEEP` instruction.

If enabled, the Watchdog Timer will be cleared but keeps running, the $\overline{PD}$ bit (RCON<3>) is cleared, the $\overline{TO}$ (RCON<4>) bit is set and the oscillator driver is turned off. The I/O ports maintain the status they had before the `SLEEP` instruction was executed (driving high, low, or high-impedance).

For lowest current consumption in this mode, place all I/O pins at either V_{DD} or V_{SS}, ensure no external circuitry is drawing current from the I/O pin, power-down the A/D and disable external clocks. Pull all I/O pins that are high-impedance inputs, high or low externally to avoid switching currents caused by floating inputs. The T0CKI input should also be at V_{DD} or V_{SS} for lowest current consumption. The contribution from on-chip pull-ups on PORTB should be considered.

The $\overline{MCLR}$ pin must be at a logic high level (V_{IHMC}).

24.3.1 WAKE-UP FROM SLEEP

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

1. External Reset input on $\overline{MCLR}$ pin.
2. Watchdog Timer Wake-up (if WDT was enabled).
3. Interrupt from INT pin, RB port change or a peripheral interrupt.

The following peripheral interrupts can wake the device from Sleep:

1. PSP read or write.
2. TMR1 interrupt. Timer1 must be operating as an asynchronous counter.
3. TMR3 interrupt. Timer3 must be operating as an asynchronous counter.
4. CCP Capture mode interrupt.
5. Special event trigger (Timer1 in Asynchronous mode using an external clock).
6. MSSP (Start/Stop) bit detect interrupt.
7. MSSP transmit or receive in Slave mode (SPI/I^2C).
8. USART RX or TX (Synchronous Slave mode).
9. A/D conversion (when A/D clock source is RC).
10. EEPROM write operation complete.
11. LVD interrupt.
12. CAN wake-up interrupt.

Other peripherals cannot generate interrupts since during Sleep, no on-chip clocks are present.

External $\overline{MCLR}$ Reset will cause a device Reset. All other events are considered a continuation of program execution and will cause a "wake-up". The $\overline{TO}$ and $\overline{PD}$ bits in the RCON register can be used to determine the cause of the device Reset. The $\overline{PD}$ bit which is set on power-up is cleared when Sleep is invoked. The $\overline{TO}$ bit is cleared if a WDT time-out occurred (and caused wake-up).

When the `SLEEP` instruction is being executed, the next instruction (PC + 2) is pre-fetched. For the device to wake-up through an interrupt event, the corresponding interrupt enable bit must be set (enabled). Wake-up is regardless of the state of the GIE bit. If the GIE bit is clear (disabled), the device continues execution at the instruction after the `SLEEP` instruction. If the GIE bit is set (enabled), the device executes the instruction after the `SLEEP` instruction and then branches to the interrupt address. In cases where the execution of the instruction following `SLEEP` is not desirable, the user should have a `NOP` after the `SLEEP` instruction.

24.3.2 WAKE-UP USING INTERRUPTS

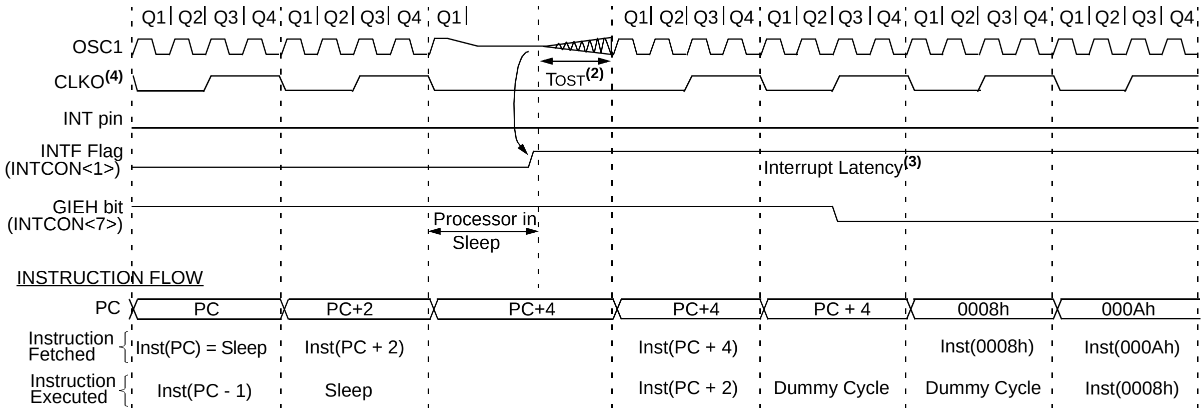
When global interrupts are disabled (GIE cleared) and any interrupt source has both its interrupt enable bit and interrupt flag bit set, one of the following will occur:

- If an interrupt condition (interrupt flag bit and interrupt enable bits are set) occurs **before** the execution of a `SLEEP` instruction, the `SLEEP` instruction will complete as a `NOP`. Therefore, the WDT and WDT postscaler will not be cleared, the $\overline{TO}$ bit will not be set and $\overline{PD}$ bits will not be cleared.
- If the interrupt condition occurs **during or after** the execution of a `SLEEP` instruction, the device will immediately wake-up from Sleep. The `SLEEP` instruction will be completely executed before the wake-up. Therefore, the WDT and WDT postscaler will be cleared, the $\overline{TO}$ bit will be set and the $\overline{PD}$ bit 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`.

To ensure that the WDT is cleared, a `CLRWDT` instruction should be executed before a `SLEEP` instruction.

FIGURE 24-2: WAKE-UP FROM SLEEP THROUGH INTERRUPT(1,2)



Note 1: XT, HS or LP Oscillator mode assumed.

2: GIE = `1` assumed. In this case after wake-up, the processor jumps to the interrupt routine. If GIE = `0`, execution will continue in-line.

3: TOST = 1024 TOSC (drawing not to scale). This delay will not occur for RC and EC Oscillator modes.

4: CLKO is not available in these oscillator modes but shown here for timing reference.

24.4 Program Verification and Code Protection

The overall structure of the code protection on the PIC18 Flash devices differs significantly from other PICmicro® devices.

The user program memory is divided on binary boundaries into four blocks of 16 Kbytes each. The first block is further divided into a boot block of 2048 bytes and a second block (Block 0) of 14 Kbytes.

Each of the blocks has three code protection bits associated with them. They are:

- Code-Protect bit (CPn)
- Write-Protect bit (WRTn)
- External Block Table Read bit (EBTRn)

Figure 24-3 shows the program memory organization for 48 and 64-Kbyte devices and the specific code protection bit associated with each block. The actual locations of the bits are summarized in Table 24-3.

FIGURE 24-3: CODE-PROTECTED PROGRAM MEMORY FOR PIC18FXX8X DEVICES

MEMORY SIZE/DEVICE 48 Kbytes (PIC18FX585	MEMORY SIZE/DEVICE 64 Kbytes (PIC18FX680)	Address Range	Block Code Protection Controlled By:
Boot Block	Boot Block	000000h 0007FFh	CPB, WRTB, EBTRB
Block 0	Block 0	000800h 003FFFh	CP0, WRT0, EBTR0
Block 1	Block 1	004000h 007FFFh	CP1, WRT1, EBTR1
Block 2	Block 2	008000h 00BFFFh	CP2, WRT2, EBTR2
Unimplemented Read ‘0’	Block 3	00C000h 00FFFFh	CP3, WRT3, EBTR3

TABLE 24-3: SUMMARY OF CODE PROTECTION REGISTERS

File Name		Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
300008h	CONFIG5L	—	—	—	—	CP3(1)	CP2	CP1	CP0
300009h	CONFIG5H	CPD	CPB	—	—	—	—	—	—
30000Ah	CONFIG6L	—	—	—	—	WRT3(1)	WRT2	WRT1	WRT0
30000Bh	CONFIG6H	WRTD	WRTB	WRTC	—	—	—	—	—
30000Ch	CONFIG7L	—	—	—	—	EBTR3(1)	EBTR2	EBTR1	EBTR0
30000Dh	CONFIG7H	—	EBTRB	—	—	—	—	—	—

Legend: Shaded cells are unimplemented.

Note 1: Unimplemented in PIC18FX585 devices.

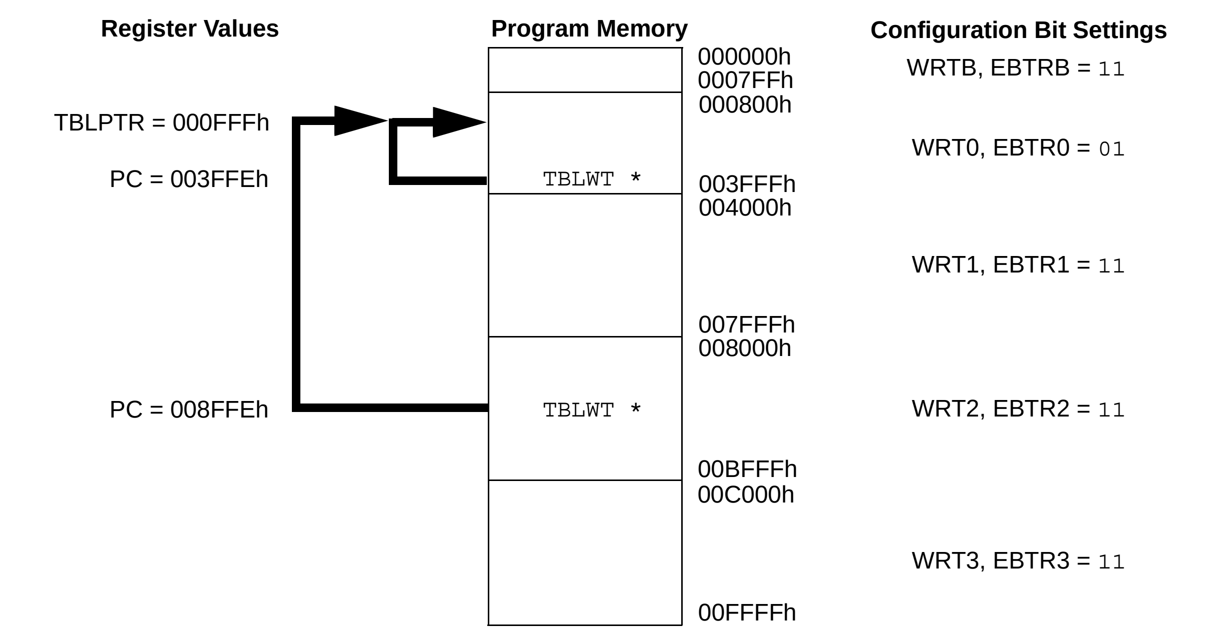
24.4.1 PROGRAM MEMORY CODE PROTECTION

The user memory may be read to or written from any location using the table read and table write instructions. The device ID may be read with table reads. The Configuration registers may be read and written with the table read and table write instructions.

In User mode, the CPn bits have no direct effect. CPn bits inhibit external reads and writes. A block of user memory may be protected from table writes if the WRTn configuration bit is '0'. The EBTRn bits control table reads. For a block of user memory with the EBTRn bit set to '0', a table read instruction that executes from within that block is allowed to read. A table read instruction that executes from a location outside of that block is not allowed to read and will result in reading '0's. Figures 24-4 through 24-6 illustrate table write and table read protection.

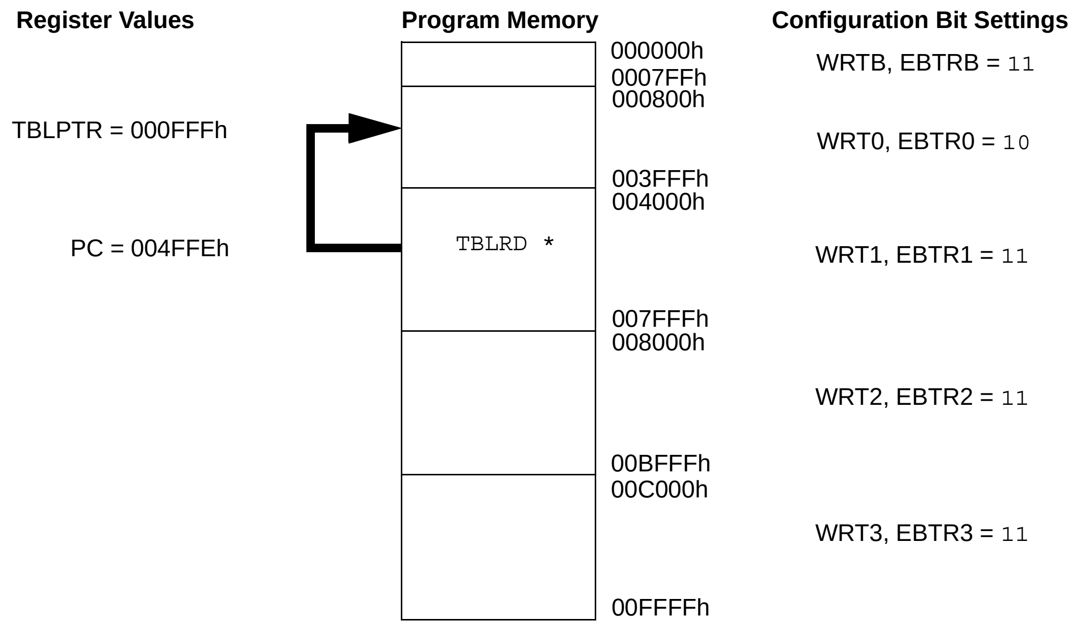
Note:	Code protection bits may only be written to a '0' from a '1' state. It is not possible to write a '1' to a bit in the '0' state. Code protection bits are only set to '1' by a full chip erase or block erase function. The full chip erase and block erase functions can only be initiated via ICSP or an external programmer.

FIGURE 24-4: TABLE WRITE (WRTn) DISALLOWED



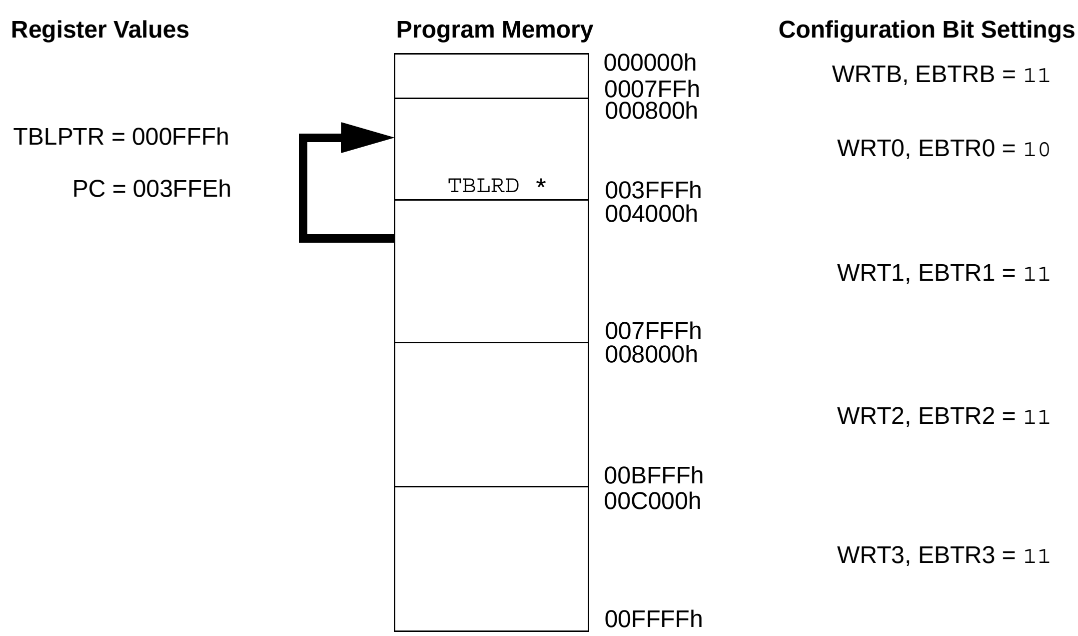
Results: All table writes disabled to Block n whenever WRTn = 0.

FIGURE 24-5: EXTERNAL BLOCK TABLE READ (EBTRn) DISALLOWED



Results: All table reads from external blocks to Block n are disabled whenever EBTRn = 0.
TABLAT register returns a value of '0'.

FIGURE 24-6: EXTERNAL BLOCK TABLE READ (EBTRn) ALLOWED



Results: Table reads permitted within Block n even when EBTRBn = 0.
TABLAT register returns the value of the data at the location TBLPTR.

24.4.2 DATA EEPROM CODE PROTECTION

The entire data EEPROM is protected from external reads and writes by two bits: CPD and WRTD. CPD inhibits external reads and writes of data EEPROM. WRTD inhibits external writes to data EEPROM. The CPU can continue to read and write data EEPROM regardless of the protection bit settings.

24.4.3 CONFIGURATION REGISTER PROTECTION

The Configuration registers can be write-protected. The WRTC bit controls protection of the Configuration registers. In User mode, the WRTC bit is readable only. WRTC can only be written via ICSP or an external programmer.

24.5 ID Locations

Eight memory locations (200000h-200007h) are designated as ID locations where the user can store checksum or other code identification numbers. These locations are accessible during normal execution through the `TBLRD` and `TBLWT` instructions or during program/verify. The ID locations can be read when the device is code-protected.

24.6 In-Circuit Serial Programming

PIC18FXX80/XX85 microcontrollers can be serially programmed while in the end application circuit. This is simply done with two lines for clock and data, and three other lines for power, ground and the programming voltage. This allows customers to manufacture boards with unprogrammed devices and then program the microcontroller just before shipping the product. This also allows the most recent firmware or a custom firmware to be programmed.

24.7 In-Circuit Debugger

When the DEBUG bit in Configuration register, CONFIG4L, is programmed to a '0', the in-circuit debugger functionality is enabled. This function allows simple debugging functions when used with MPLAB® IDE. When the microcontroller has this feature enabled, some of the resources are not available for general use. Table 24-4 shows which features are consumed by the background debugger.

TABLE 24-4: DEBUGGER RESOURCES

I/O pins	RB6, RB7
Stack	2 levels
Program Memory	512 bytes
Data Memory	10 bytes

To use the in-circuit debugger function of the microcontroller, the design must implement In-Circuit Serial Programming connections to $\overline{\text{MCLR}}$/V_{PP}, V_{DD}, GND, RB7 and RB6. This will interface to the in-circuit debugger module available from Microchip or one of the third party development tool companies.

24.8 Low-Voltage ICSP Programming

The LVP bit in Configuration register, CONFIG4L, enables Low-Voltage ICSP Programming. This mode allows the microcontroller to be programmed via ICSP using a VDD source in the operating voltage range. This only means that VPP does not have to be brought to VIHH but can instead be left at the normal operating voltage. In this mode, the RB5/KBI1/PGM pin is dedicated to the programming function and ceases to be a general purpose I/O pin. During programming, VDD is applied to the RG5/MCLR/VPP pin. To enter Programming mode, VDD must be applied to the RB5/KBI1/PGM pin, provided the LVP bit is set. The LVP bit defaults to a '1' from the factory.

> **Note 1:** The High-Voltage Programming mode is always available regardless of the state of the LVP bit, by applying VIHH to the MCLR pin.
>
> **2:** While in Low-Voltage ICSP mode, the RB5 pin can no longer be used as a general purpose I/O pin and should be held low during normal operation.
>
> **3:** When using Low-Voltage ICSP Programming (LVP) and the pull-ups on PORTB are enabled, bit 5 in the TRISB register must be cleared to disable the pull-up on RB5 and ensure the proper operation of the device.
>
> **4:** If the device Master Clear is disabled, verify that either of the following is done to ensure proper entry into ICSP mode:
>
> a) disable Low-Voltage Programming (CONFIG4L<2> = 0); or
>
> b) make certain that RB5/KBI1/PGM is held low during entry into ICSP.

If Low-Voltage Programming mode is not used, the LVP bit can be programmed to a '0' and RB5/KBI1/PGM becomes a digital I/O pin. However, the LVP bit may only be programmed when programming is entered with VIHH on RG5/MCLR/VPP.

It should be noted that once the LVP bit is programmed to '0', only the High-Voltage Programming mode is available and only High-Voltage Programming mode can be used to program the device.

When using low-voltage ICSP, the part must be supplied 4.5V to 5.5V if a bulk erase will be executed. This includes reprogramming of the code-protect bits from an on-state to an off-state. For all other cases of low-voltage ICSP, the part may be programmed at the normal operating voltage. This means unique user IDs or user code can be reprogrammed or added.

NOTES:

25.0 INSTRUCTION SET SUMMARY

The PIC18 instruction set adds many enhancements to the previous PICmicro instruction sets, while maintaining an easy migration from these PICmicro instruction sets.

Most instructions are a single program memory word (16 bits) but there are three 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 25-2 lists **byte-oriented**, **bit-oriented**, **literal** and **control** operations. Table 25-1 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 three double-word instructions. These three instructions were made double-word instructions so that all the required information is available in these 32 bits. In the second word, the 4 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 25-1 shows the general formats that the instructions can have.

All examples use the format '`nnh`' to represent a hexadecimal number, where '`h`' signifies a hexadecimal digit.

The Instruction Set Summary, shown in Table 25-2, lists the instructions recognized by the Microchip Assembler (MPASM™).

Section 25.1 "Instruction Set" provides a description of each instruction.

TABLE 25-1: 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.
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 (0x00 to 0xFF).
fs	12-bit register file address (0x000 to 0xFFF). This is the source address.
fd	12-bit register file address (0x000 to 0xFFF). This is the destination address.
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.
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)
u	Unused or unchanged.
WREG	Working register (accumulator).
x	Don’t care (0 or 1). The assembler will generate code with x = 0. It is the recommended form of use for compatibility with all Microchip software tools.
TBLPTR	21-bit Table Pointer (points to a program memory location).
TABLAT	8-bit Table Latch.
TOS	Top-of-Stack.
PC	Program Counter.
PCL	Program Counter Low Byte.
PCH	Program Counter High Byte.
PCLATH	Program Counter High Byte Latch.
PCLATU	Program Counter Upper Byte Latch.
GIE	Global Interrupt Enable bit.
WDT	Watchdog Timer.
$\overline{\text{TO}}$	Time-out bit.
$\overline{\text{PD}}$	Power-down bit.
C, DC, Z, OV, N	ALU status bits: Carry, Digit Carry, Zero, Overflow, Negative.
[ ]	Optional.
( )	Contents.
→	Assigned to.
< >	Register bit field.
∈	In the set of.
italics	User defined term (font is courier).

FIGURE 25-1: GENERAL FORMAT FOR INSTRUCTIONS

Byte-oriented file register operations — **Example Instruction**

15 … 10	9	8	7 … 0
OPCODE	d	a	f (FILE #)

`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)

15 … 12	11 … 0
OPCODE	f (Source FILE #)
`1111`	f (Destination FILE #)

`MOVFF MYREG1, MYREG2`

f = 12-bit file register address

Bit-oriented file register operations

15 … 12	11 … 9	8	7 … 0
OPCODE	b (BIT #)	a	f (FILE #)

`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

15 … 8	7 … 0
OPCODE	k (literal)

`MOVLW 0x7F`

k = 8-bit immediate value

Control operations

CALL, GOTO and Branch operations

15 … 8	7 … 0
OPCODE	n<7:0> (literal)

15 … 12	11 … 0
`1111`	n<19:8> (literal)

`GOTO Label`

n = 20-bit immediate value

15 … 9	8	7 … 0
OPCODE	S	n<7:0> (literal)

15 … 12	11 … 0
	n<19:8> (literal)

`CALL MYFUNC`

S = Fast bit

15 … 11	10 … 0
OPCODE	n<10:0> (literal)

`BRA MYFUNC`

15 … 8	7 … 0
OPCODE	n<7:0> (literal)

`BC MYFUNC`

TABLE 25-2: PIC18FXXX INSTRUCTION SET

Mnemonic, Operands		Description	Cycles	16-Bit Instruction Word MSb			LSb	Status Affected	Notes
BYTE-ORIENTED FILE REGISTER OPERATIONS									
ADDWF	f, d, a	Add WREG and f	1	0010	01da	ffff	ffff	C, DC, Z, OV, N	1, 2
ADDWFC	f, d, a	Add WREG and Carry bit to f	1	0010	00da	ffff	ffff	C, DC, Z, OV, N	1, 2
ANDWF	f, d, a	AND WREG with f	1	0001	01da	ffff	ffff	Z, N	1,2
CLRF	f, a	Clear f	1	0110	101a	ffff	ffff	Z	2
COMF	f, d, a	Complement f	1	0001	11da	ffff	ffff	Z, N	1, 2
CPFSEQ	f, a	Compare f with WREG, Skip =	1 (2 or 3)	0110	001a	ffff	ffff	None	4
CPFSGT	f, a	Compare f with WREG, Skip >	1 (2 or 3)	0110	010a	ffff	ffff	None	4
CPFSLT	f, a	Compare f with WREG, Skip <	1 (2 or 3)	0110	000a	ffff	ffff	None	1, 2
DECF	f, d, a	Decrement f	1	0000	01da	ffff	ffff	C, DC, Z, OV, N	1, 2, 3, 4
DECFSZ	f, d, a	Decrement f, Skip if 0	1 (2 or 3)	0010	11da	ffff	ffff	None	1, 2, 3, 4
DCFSNZ	f, d, a	Decrement f, Skip if Not 0	1 (2 or 3)	0100	11da	ffff	ffff	None	1, 2
INCF	f, d, a	Increment f	1	0010	10da	ffff	ffff	C, DC, Z, OV, N	1, 2, 3, 4
INCFSZ	f, d, a	Increment f, Skip if 0	1 (2 or 3)	0011	11da	ffff	ffff	None	4
INFSNZ	f, d, a	Increment f, Skip if Not 0	1 (2 or 3)	0100	10da	ffff	ffff	None	1, 2
IORWF	f, d, a	Inclusive OR WREG with f	1	0001	00da	ffff	ffff	Z, N	1, 2
MOVF	f, d, a	Move f	1	0101	00da	ffff	ffff	Z, N	1
MOVFF	f_s, f_d	Move f_s (source) to 1st word f_d (destination) 2nd word	2	1100 1111	ffff ffff	ffff ffff	ffff ffff	None	
MOVWF	f, a	Move WREG to f	1	0110	111a	ffff	ffff	None	
MULWF	f, a	Multiply WREG with f	1	0000	001a	ffff	ffff	None	
NEGF	f, a	Negate f	1	0110	110a	ffff	ffff	C, DC, Z, OV, N	1, 2
RLCF	f, d, a	Rotate Left f through Carry	1	0011	01da	ffff	ffff	C, Z, N	
RLNCF	f, d, a	Rotate Left f (No Carry)	1	0100	01da	ffff	ffff	Z, N	1, 2
RRCF	f, d, a	Rotate Right f through Carry	1	0011	00da	ffff	ffff	C, Z, N	
RRNCF	f, d, a	Rotate Right f (No Carry)	1	0100	00da	ffff	ffff	Z, N	
SETF	f, a	Set f	1	0110	100a	ffff	ffff	None	
SUBFWB	f, d, a	Subtract f from WREG with borrow	1	0101	01da	ffff	ffff	C, DC, Z, OV, N	1, 2
SUBWF	f, d, a	Subtract WREG from f	1	0101	11da	ffff	ffff	C, DC, Z, OV, N	
SUBWFB	f, d, a	Subtract WREG from f with borrow	1	0101	10da	ffff	ffff	C, DC, Z, OV, N	1, 2
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 FILE REGISTER OPERATIONS									
BCF	f, b, a	Bit Clear f	1	1001	bbba	ffff	ffff	None	1, 2
BSF	f, b, a	Bit Set f	1	1000	bbba	ffff	ffff	None	1, 2
BTFSC	f, b, a	Bit Test f, Skip if Clear	1 (2 or 3)	1011	bbba	ffff	ffff	None	3, 4
BTFSS	f, b, a	Bit Test f, Skip if Set	1 (2 or 3)	1010	bbba	ffff	ffff	None	3, 4
BTG	f, d, a	Bit Toggle f	1	0111	bbba	ffff	ffff	None	1, 2

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.

5: If the table write starts the write cycle to internal memory, the write will continue until terminated.

TABLE 25-2: PIC18FXXX INSTRUCTION SET (CONTINUED)

Mnemonic, Operands		Description	Cycles	16-Bit Instruction Word MSb … LSb	Status Affected	Notes
CONTROL OPERATIONS						
BC	n	Branch if Carry	1 (2)	1110 0010 nnnn nnnn	None	
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	n, 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	n	Go to address 1st word	2	1110 1111 kkkk kkkk	None	
		2nd word		1111 kkkk kkkk kkkk		
NOP	—	No Operation	1	0000 0000 0000 0000	None	
NOP	—	No Operation	1	1111 xxxx xxxx xxxx	None	4
POP	—	Pop top of return stack (TOS)	1	0000 0000 0000 0110	None	
PUSH	—	Push top of return stack (TOS)	1	0000 0000 0000 0101	None	
RCALL	n	Relative Call	2	1101 1nnn nnnn nnnn	None	
RESET		Software device Reset	1	0000 0000 1111 1111	All	
RETFIE	s	Return from interrupt enable	2	0000 0000 0001 000s	GIE/GIEH, PEIE/GIEL	
RETLW	k	Return with literal in WREG	2	0000 1100 kkkk kkkk	None	
RETURN	s	Return from Subroutine	2	0000 0000 0001 001s	None	
SLEEP	—	Go into Standby mode	1	0000 0000 0000 0011	$\overline{TO}$, $\overline{PD}$	

Note 1: When a Port register is modified as a function of itself (e.g., `MOVF PORTB, 1, 0`), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.

2: If this instruction is executed on the TMR0 register (and, where applicable, d = 1), the prescaler will be cleared if assigned.

3: If Program Counter (PC) is modified or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a NOP.

4: Some instructions are two-word instructions. The second word of these instructions will be executed as a NOP unless the first word of the instruction retrieves the information embedded in these 16 bits. This ensures that all program memory locations have a valid instruction.

5: If the table write starts the write cycle to internal memory, the write will continue until terminated.

TABLE 25-2: PIC18FXXX INSTRUCTION SET (CONTINUED)

Mnemonic, Operands	Description	Cycles	16-Bit Instruction Word MSb LSb	Status Affected	Notes
LITERAL OPERATIONS					
ADDLW k	Add literal and WREG	1	0000 1111 kkkk kkkk	C, DC, Z, OV, N	
ANDLW k	AND literal with WREG	1	0000 1011 kkkk kkkk	Z, N	
IORLW k	Inclusive OR literal with WREG	1	0000 1001 kkkk kkkk	Z, N	
LFSR f, k	Move literal (12-bit) 2nd word to FSRx 1st word	2	1110 1110 00ff kkkk 1111 0000 kkkk kkkk	None	
MOVLB k	Move literal to BSR<3:0>	1	0000 0001 0000 kkkk	None	
MOVLW k	Move literal to WREG	1	0000 1110 kkkk kkkk	None	
MULLW k	Multiply literal with WREG	1	0000 1101 kkkk kkkk	None	
RETLW k	Return with literal in WREG	2	0000 1100 kkkk kkkk	None	
SUBLW k	Subtract WREG from literal	1	0000 1000 kkkk kkkk	C, DC, Z, OV, N	
XORLW k	Exclusive OR literal with WREG	1	0000 1010 kkkk kkkk	Z, N	
DATA MEMORY ↔ PROGRAM MEMORY OPERATIONS					
TBLRD*	Table Read	2	0000 0000 0000 1000	None	
TBLRD*+	Table Read with post-increment		0000 0000 0000 1001	None	
TBLRD*-	Table Read with post-decrement		0000 0000 0000 1010	None	
TBLRD+*	Table Read with pre-increment		0000 0000 0000 1011	None	
TBLWT*	Table Write	2 (5)	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.

5: If the table write starts the write cycle to internal memory, the write will continue until terminated.

25.1 Instruction Set

ADDLW — ADD literal to W

Syntax:	[*label*] 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 0x15`

Before Instruction
W = 0x10

After Instruction
W = 0x25

ADDWF — ADD W to f

Syntax:	[*label*] ADDWF f [,d [,a] 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 'd' (default). If 'a' is '0', the Access Bank will be selected. If 'a' is '1', the BSR is used.
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 = 0x17
REG = 0xC2

After Instruction
W = 0xD9
REG = 0xC2

ADDWFC **ADD W and Carry bit to f**

Syntax:	[*label*] ADDWFC f [,d [,a]]
Operands:	$0 \leq f \leq 255$ $d \in [0,1]$ $a \in [0,1]$
Operation:	(W) + (f) + (C) → dest
Status Affected:	N,OV, C, DC, Z
Encoding:	`0010` `00da` `ffff` `ffff`
Description:	Add W, the Carry Flag and data memory location 'f'. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed in data memory location 'f'. If 'a' is '0', the Access Bank will be selected. If 'a' is '1', the BSR will not be overridden.
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 = 0x02
W = 0x4D

After Instruction

Carry bit = 0
REG = 0x02
W = 0x50

ANDLW **AND literal with W**

Syntax:	[*label*] ANDLW k
Operands:	$0 \leq k \leq 255$
Operation:	(W) .AND. k → W
Status Affected:	N, Z
Encoding:	`0000` `1011` `kkkk` `kkkk`
Description:	The contents of W are ANDed with the 8-bit literal 'k'. The result is placed in W.
Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Write to W

Example: `ANDLW 0x5F`

Before Instruction

W = 0xA3

After Instruction

W = 0x03

ANDWF **AND W with f**

Syntax:	[*label*] 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 will be selected. If 'a' is '1', the BSR will not be overridden (default).
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 = 0x17
REG = 0xC2

After Instruction

W = 0x02
REG = 0xC2

BC **Branch if Carry**

Syntax:	[*label*] 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 two-cycle instruction.
Words:	1
Cycles:	1(2)

Q Cycle Activity:

If Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	Write to PC
No operation	No operation	No operation	No operation

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	No operation

Example: `HERE BC 5`

Before Instruction

PC = address (`HERE`)

After Instruction

If Carry = 1;
PC = address (`HERE+12`)
If Carry = 0;
PC = address (`HERE+2`)

BCF **Bit Clear f**

Syntax:	[*label*] 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 will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).
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 = 0xC7

After Instruction
FLAG_REG = 0x47

BN **Branch if Negative**

Syntax:	[*label*] 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 two-cycle instruction.
Words:	1
Cycles:	1(2)

Q Cycle Activity:
If Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	Write to PC
No operation	No operation	No operation	No operation

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	No operation

Example: `HERE BN Jump`

Before Instruction
PC = address (`HERE`)

After Instruction
If Negative = 1;
PC = address (`Jump`)
If Negative = 0;
PC = address (`HERE+2`)

BNC — Branch if Not Carry

Syntax:	[*label*] BNC n
Operands:	$-128 \leq n \leq 127$
Operation:	if carry bit is '0' (PC) + 2 + 2n → PC
Status Affected:	None
Encoding:	1110 | 0011 | nnnn | nnnn
Description:	If the Carry bit is '0', then the program will branch. The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC+2+2n. This instruction is then a two-cycle instruction.
Words:	1
Cycles:	1(2)

Q Cycle Activity:
If Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	Write to PC
No operation	No operation	No operation	No operation

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	No operation

<u>Example</u>: `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:	[*label*] BNN n
Operands:	$-128 \leq n \leq 127$
Operation:	if negative bit is '0' (PC) + 2 + 2n → PC
Status Affected:	None
Encoding:	1110 | 0111 | nnnn | nnnn
Description:	If the Negative bit is '0', then the program will branch. The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC+2+2n. This instruction is then a two-cycle instruction.
Words:	1
Cycles:	1(2)

Q Cycle Activity:
If Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	Write to PC
No operation	No operation	No operation	No operation

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	No operation

<u>Example</u>: `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:	[*label*] 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 two-cycle instruction.
Words:	1
Cycles:	1(2)

Q Cycle Activity:

If Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	Write to PC
No operation	No operation	No operation	No operation

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	No operation

Example: `HERE 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:	[*label*] 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 two-cycle instruction.
Words:	1
Cycles:	1(2)

Q Cycle Activity:

If Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	Write to PC
No operation	No operation	No operation	No operation

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	No operation

Example: `HERE BNZ Jump`

Before Instruction
PC = address (HERE)

After Instruction
If Zero = 0;
PC = address (Jump)
If Zero = 1;
PC = address (HERE+2)

BRA Unconditional Branch

Syntax:	[*label*] 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 two-cycle instruction.
Words:	1
Cycles:	2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	Write to PC
No operation	No operation	No operation	No operation

Example: `HERE BRA Jump`

Before Instruction
PC = address `(HERE)`
After Instruction
PC = address `(Jump)`

BSF Bit Set f

Syntax:	[*label*] 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`
Description:	Bit 'b' in register 'f' is set. If 'a' is '0', Access Bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value.
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 = 0x0A
After Instruction
FLAG_REG = 0x8A

BTFSC — Bit Test File, Skip if Clear

Syntax: [*label*] 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 two-cycle instruction. If 'a' is '`0`', the Access Bank will be selected, overriding the BSR value. If 'a' = `1`, then the bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1(2)
Note: 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	No operation

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example:

```
HERE    BTFSC   FLAG, 1, 0
FALSE   :
TRUE    :
```

Before Instruction
PC = address `(HERE)`

After Instruction
If FLAG<1> = 0;
PC = address `(TRUE)`
If FLAG<1> = 1;
PC = address `(FALSE)`

BTFSS — Bit Test File, Skip if Set

Syntax: [*label*] 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 two-cycle instruction. If 'a' is '`0`', the Access Bank will be selected, overriding the BSR value. If 'a' = `1`, then the bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1(2)
Note: 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	No operation

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example:

```
HERE    BTFSS   FLAG, 1, 0
FALSE   :
TRUE    :
```

Before Instruction
PC = address `(HERE)`

After Instruction
If FLAG<1> = 0;
PC = address `(FALSE)`
If FLAG<1> = 1;
PC = address `(TRUE)`

BTG — Bit Toggle f

Syntax:	[*label*] BTG f,b[,a]
Operands:	0 ≤ f ≤ 255 0 ≤ b < 7 a ∈ [0,1]
Operation:	($\overline{\text{f<b>}}$) → 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 will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).

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 [0x75]

After Instruction:
PORTC = 0110 0101 [0x65]

BOV — Branch if Overflow

Syntax:	[*label*] 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 two-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:
If Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	Write to PC
No operation	No operation	No operation	No operation

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	No operation

Example: `HERE BOV Jump`

Before Instruction
PC = address (HERE)

After Instruction
If Overflow = 1;
PC = address (Jump)
If Overflow = 0;
PC = address (HERE+2)

BZ — Branch if Zero

Syntax:	[*label*] BZ n
Operands:	$-128 \le n \le 127$
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 two-cycle instruction.
Words:	1
Cycles:	1(2)

Q Cycle Activity:

If Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	Write to PC
No operation	No operation	No operation	No operation

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	No operation

Example: `HERE 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:	[*label*] CALL k [,s]
Operands:	$0 \le k \le 1048575$ $s \in [0,1]$
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>)	1110	110s	$k_7$kkk	kkkk$_0$
2nd word(k<19:8>)	1111	k_{19}kkk	kkkk	kkkk$_8$

Description:	Subroutine call of entire 2-Mbyte memory range. First, return address (PC+ 4) is pushed onto the return stack. If 's' = 1, the 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 two-cycle instruction.
Words:	2
Cycles:	2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'<7:0>,	Push PC to stack	Read literal 'k'<19:8>, 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:	[*label*] 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 will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).
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 = 0x5A
After Instruction
FLAG_REG = 0x00

CLRWDT **Clear Watchdog Timer**

Syntax:	[*label*] 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 = 0x00
WDT Postscaler = 0
$\overline{TO}$ = 1
$\overline{PD}$ = 1

COMF — **Complement f**

Syntax: [*label*] COMF f [,d [,a]]

Operands: $0 \le f \le 255$
d ∈ [0,1]
a ∈ [0,1]

Operation: $(\overline{f}) \rightarrow$ dest

Status Affected: N, Z

Encoding:

0001	11da	ffff	ffff

Description: The contents of register ‘f’ are complemented. If ‘d’ is ‘0’, the result is stored in W. If ‘d’ is ‘1’, the result is stored back in register ‘f’ (default). If ‘a’ is ‘0’, the Access Bank will be selected, overriding the BSR value. If ‘a’ = 1, then the bank will be selected as per the BSR value (default).

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 = 0x13

After Instruction
REG = 0x13
W = 0xEC

CPFSEQ — **Compare f with W, skip if f = W**

Syntax: [*label*] CPFSEQ f [,a]

Operands: $0 \le f \le 255$
a ∈ [0,1]

Operation: (f) – (W),
skip if (f) = (W)
(unsigned comparison)

Status Affected: None

Encoding:

0110	001a	ffff	ffff

Description: Compares the contents of data memory location ‘f’ to the contents of W by performing an unsigned subtraction.
If ‘f’ = W, then the fetched instruction is discarded and a NOP is executed instead, making this a two-cycle instruction. If ‘a’ is ‘0’, the Access Bank will be selected, overriding the BSR value. If ‘a’ = 1, then the bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1(2)
Note: 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register ‘f’	Process Data	No operation

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example:
```
HERE     CPFSEQ REG, 0
NEQUAL   :
EQUAL    :
```

Before Instruction
PC Address = HERE
W = ?
REG = ?

After Instruction
If REG = W;
PC = Address (EQUAL)
If REG ≠ W;
PC = Address (NEQUAL)

CPFSGT	**Compare f with W, skip if f > W**			
Syntax:	[*label*] CPFSGT f [,a]			
Operands:	$0 \le f \le 255$ $a \in [0,1]$			
Operation:	(f) – (W), skip if (f) > (W) (unsigned comparison)			
Status Affected:	None			
Encoding:	`0110`	`010a`	`ffff`	`ffff`
Description:	Compares the contents of data memory location ‘f’ to the contents of the W by performing an unsigned subtraction. If the contents of ‘f’ are greater than the contents of WREG, then the fetched instruction is discarded and a `NOP` is executed instead, making this a two-cycle instruction. If ‘a’ is ‘`0`’, the Access Bank will be selected, overriding the BSR value. If ‘a’ = `1`, then the bank will be selected as per the BSR value (default).			
Words:	1			
Cycles:	1(2) **Note:** 3 cycles if skip and followed by a 2-word instruction.			

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register ‘f’	Process Data	No operation

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example:

```
HERE      CPFSGT REG, 0
NGREATER  :
GREATER   :
```

Before Instruction
- PC = Address (`HERE`)
- W = ?

After Instruction
- If REG > W;
- PC = Address (`GREATER`)
- If REG ≤ W;
- PC = Address (`NGREATER`)

CPFSLT	**Compare f with W, skip if f < W**			
Syntax:	[*label*] CPFSLT f [,a]			
Operands:	$0 \le f \le 255$ $a \in [0,1]$			
Operation:	(f) – (W), skip if (f) < (W) (unsigned comparison)			
Status Affected:	None			
Encoding:	`0110`	`000a`	`ffff`	`ffff`
Description:	Compares the contents of data memory location ‘f’ to the contents of W by performing an unsigned subtraction. If the contents of ‘f’ are less than the contents of W, then the fetched instruction is discarded and a `NOP` is executed instead, making this a two-cycle instruction. If ‘a’ is ‘`0`’, the Access Bank will be selected. If ’a’ is ‘`1`’, the BSR will not be overridden (default).			
Words:	1			
Cycles:	1(2) **Note:** 3 cycles if skip and followed by a 2-word instruction.			

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register ‘f’	Process Data	No operation

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example:

```
HERE    CPFSLT REG, 1
NLESS   :
LESS    :
```

Before Instruction
- PC = Address (`HERE`)
- W = ?

After Instruction
- If REG < W;
- PC = Address (`LESS`)
- If REG ≥ W;
- PC = Address (`NLESS`)

DAW — Decimal Adjust W Register

Syntax:	[*label*] DAW
Operands:	None

Operation:	If [W<3:0> >9] or [DC = 1] then
	(W<3:0>) + 6 → W<3:0>;
	else
	(W<3:0>) → W<3:0>;
	
	If [W<7:4> >9] or [C = 1] then
	(W<7:4>) + 6 → W<7:4>;
	else
	(W<7:4>) → W<7:4>;

Status Affected:	C
Encoding:	0000 0000 0000 0111
Description:	`DAW` adjusts the eight-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 = 0xA5
C = 0
DC = 0

After Instruction

W = 0x05
C = 1
DC = 0

Example 2:

Before Instruction

W = 0xCE
C = 0
DC = 0

After Instruction

W = 0x34
C = 1
DC = 0

DECF — Decrement f

Syntax:	[*label*] DECF f [,d [,a]]
Operands:	$0 \le f \le 255$ $d \in [0,1]$ $a \in [0,1]$
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 will be selected, overriding the BSR value. If ‘a’ = 1, then the bank will be selected as per the BSR value (default).
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 = 0x01
Z = 0

After Instruction

CNT = 0x00
Z = 1

DECFSZ	**Decrement f, skip if 0**
Syntax:	[*label*] 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 two-cycle instruction. If 'a' is '0', the Access Bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).
Words:	1
Cycles:	1(2) **Note:** 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example:

```
HERE      DECFSZ   CNT, 1, 1
          GOTO     LOOP
CONTINUE
```

Before Instruction
PC = Address (HERE)

After Instruction
CNT = CNT - 1
If CNT = 0;
PC = Address (CONTINUE)
If CNT ≠ 0;
PC = Address (HERE+2)

DCFSNZ	**Decrement f, skip if not 0**
Syntax:	[*label*] 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 two-cycle instruction. If 'a' is '0', the Access Bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).
Words:	1
Cycles:	1(2) **Note:** 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example:

```
HERE    DCFSNZ  TEMP, 1, 0
ZERO    :
NZERO   :
```

Before Instruction
TEMP = ?

After Instruction
TEMP = TEMP - 1,
If TEMP = 0;
PC = Address (ZERO)
If TEMP ≠ 0;
PC = Address (NZERO)

GOTO	Unconditional Branch
Syntax:	[*label*] GOTO k
Operands:	0 ≤ k ≤ 1048575
Operation:	k → PC<20:1>
Status Affected:	None
Encoding: 1st word (k<7:0>) 2nd word(k<19:8>)	`1110` `1111` k_7`kkk` `kkkk`$_0$ `1111` k_{19}`kkk` `kkkk` `kkkk`$_8$
Description:	`GOTO` allows an unconditional branch anywhere within entire 2-Mbyte memory range. The 20-bit value 'k' is loaded into PC<20:1>. `GOTO` is always a two-cycle instruction.
Words:	2
Cycles:	2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'<7:0>,	No operation	Read literal 'k'<19:8>, Write to PC
No operation	No operation	No operation	No operation

Example: `GOTO THERE`

After Instruction
PC = Address `(THERE)`

INCF	Increment f
Syntax:	[*label*] 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 will be selected, overriding the BSR value. If 'a' = `1`, then the bank will be selected as per the BSR value (default).
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 = 0xFF
Z = 0
C = ?
DC = ?

After Instruction
CNT = 0x00
Z = 1
C = 1
DC = 1

INCFSZ — Increment f, skip if 0

Syntax:	[*label*] 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`
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 two-cycle instruction. If 'a' is '`0`', the Access Bank will be selected, overriding the BSR value. If 'a' = `1`, then the bank will be selected as per the BSR value (default).
Words:	1
Cycles:	1(2) **Note:** 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example:

```
HERE    INCFSZ   CNT, 1, 0
NZERO   :
ZERO    :
```

Before Instruction
- PC = Address (`HERE`)

After Instruction
- CNT = CNT + 1
- If CNT = 0;
- PC = Address (`ZERO`)
- If CNT ≠ 0;
- PC = Address (`NZERO`)

INFSNZ — Increment f, skip if not 0

Syntax:	[*label*] 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 two-cycle instruction. If 'a' is '`0`', the Access Bank will be selected, overriding the BSR value. If 'a' = `1`, then the bank will be selected as per the BSR value (default).
Words:	1
Cycles:	1(2) **Note:** 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example:

```
HERE    INFSNZ  REG, 1, 0
ZERO
NZERO
```

Before Instruction
- PC = Address (`HERE`)

After Instruction
- REG = REG + 1
- If REG ≠ 0;
- PC = Address (`NZERO`)
- If REG = 0;
- PC = Address (`ZERO`)

IORLW **Inclusive OR literal with W**

Syntax: [*label*] IORLW k

Operands: $0 \le k \le 255$

Operation: (W) .OR. k → W

Status Affected: N, Z

Encoding:

0000	1001	kkkk	kkkk

Description: The contents of W are OR'ed with the eight-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 0x35`

Before Instruction
W = 0x9A

After Instruction
W = 0xBF

IORWF **Inclusive OR W with f**

Syntax: [*label*] IORWF f [,d [,a]]

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

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 will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).

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 = 0x13
W = 0x91

After Instruction
RESULT = 0x13
W = 0x93

LFSR **Load FSR**

Syntax: [*label*] LFSR f,k

Operands: $0 \le f \le 2$
$0 \le k \le 4095$

Operation: $k \rightarrow$ FSRf

Status Affected: None

Encoding:

1110	1110	00ff	k_{11}kkk
1111	0000	$k_7$kkk	kkkk

Description: The 12-bit literal 'k' is loaded into the file select register pointed to by 'f'.

Words: 2

Cycles: 2

Q Cycle Activity:

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, 0x3AB`

After Instruction
FSR2H = 0x03
FSR2L = 0xAB

MOVF **Move f**

Syntax: [*label*] MOVF f [,d [,a]]

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: $f \rightarrow$ 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 register 'f' (default). Location 'f' can be anywhere in the 256-byte bank. If 'a' is '0', the Access Bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).

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 = 0x22
W = 0xFF

After Instruction
REG = 0x22
W = 0x22

MOVFF **Move f to f**

Syntax: [*label*] MOVFF f_s,f_d

Operands: $0 \leq f_s \leq 4095$
$0 \leq f_d \leq 4095$

Operation: $(f_s) \rightarrow f_d$

Status Affected: None

Encoding:

1st word (source)	`1100`	`ffff`	`ffff`	`ffff`$_s$
2nd word (destin.)	`1111`	`ffff`	`ffff`	`ffff`$_d$

Description: The contents of source register 'f_s' are moved to destination register 'f_d'. Location of source 'f_s' can be anywhere in the 4096-byte data space (000h to FFFh) and location of destination 'f_d' can also be anywhere from 000h to FFFh. 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 = 0x33
REG2 = 0x11

After Instruction
REG1 = 0x33,
REG2 = 0x33

MOVLB **Move literal to low nibble in BSR**

Syntax: [*label*] MOVLB k

Operands: $0 \leq k \leq 255$

Operation: $k \rightarrow BSR$

Status Affected: None

Encoding:

`0000`	`0001`	`kkkk`	`kkkk`

Description: The 8-bit literal 'k' is loaded into the Bank Select Register (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 = 0x02

After Instruction
BSR register = 0x05

MOVLW **Move literal to W**

Syntax:	[*label*] MOVLW k
Operands:	$0 \leq k \leq 255$
Operation:	$k \rightarrow W$
Status Affected:	None
Encoding:	`0000` `1110` `kkkk` `kkkk`
Description:	The eight-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 0x5A`

After Instruction
W = 0x5A

MOVWF **Move W to f**

Syntax:	[*label*] MOVWF f [,a]
Operands:	$0 \leq f \leq 255$ $a \in [0,1]$
Operation:	$(W) \rightarrow f$
Status Affected:	None
Encoding:	`0110` `111a` `ffff` `ffff`
Description:	Move data from W to register 'f'. Location 'f' can be anywhere in the 256-byte bank. If 'a' is '0', the Access Bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).
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 = 0x4F
REG = 0xFF

After Instruction
W = 0x4F
REG = 0x4F

MULLW **Multiply Literal with W**

Syntax:	[*label*] MULLW k
Operands:	0 ≤ k ≤ 255
Operation:	(W) x k → PRODH:PRODL
Status Affected:	None
Encoding:	0000 | 1101 | kkkk | kkkk
Description:	An unsigned multiplication is carried out between the contents of W and the 8-bit literal 'k'. The 16-bit result is placed in 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 0xC4`

Before Instruction

W	=	0xE2
PRODH	=	?
PRODL	=	?

After Instruction

W	=	0xE2
PRODH	=	0xAD
PRODL	=	0x08

MULWF **Multiply W with f**

Syntax:	[*label*] MULWF f [,a]
Operands:	0 ≤ f ≤ 255 a ∈ [0,1]
Operation:	(W) x (f) → PRODH:PRODL
Status Affected:	None
Encoding:	0000 | 001a | ffff | ffff
Description:'	An unsigned multiplication is carried out between the contents of W and the register file location 'f'. The 16-bit result is stored in the PRODH:PRODL register pair. PRODH contains the high byte. Both W and 'f' are unchanged. None of the status flags are affected. Note that neither overflow nor carry is possible in this operation. A zero result is possible but not detected. If 'a' is '0', the Access Bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).
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	=	0xC4
REG	=	0xB5
PRODH	=	?
PRODL	=	?

After Instruction

W	=	0xC4
REG	=	0xB5
PRODH	=	0x8A
PRODL	=	0x94

NEGF **Negate f**

Syntax:	[*label*] NEGF f [,a]
Operands:	$0 \le f \le 255$ $a \in [0,1]$
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 will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value.
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` [0x3A]
After Instruction
REG = `1100 0110` [0xC6]

NOP **No Operation**

Syntax:	[*label*] NOP
Operands:	None
Operation:	No operation
Status Affected:	None
Encoding:	`0000` `0000` `0000` `0000` `1111` `xxxx` `xxxx` `xxxx`
Description:	No operation.
Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	No operation	No operation

Example:

None.

POP — **Pop Top of Return Stack**

Syntax:	[*label*] 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:	[*label*] 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 = 00345Ah
- PC = 000124h

After Instruction
- PC = 000126h
- TOS = 000126h
- Stack (1 level down) = 00345Ah

RCALL	**Relative Call**
Syntax:	[*label*] RCALL n
Operands:	$-1024 \le n \le 1023$
Operation:	(PC) + 2 → TOS, (PC) + 2 + 2n → PC
Status Affected:	None
Encoding:	`1101` `1nnn` `nnnn` `nnnn`
Description:	Subroutine call with a jump up to 1K from the current location. First, return address (PC+2) is pushed onto the stack. Then, add the 2's complement number '2n' to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC+2+2n. This instruction is a two-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:	[*label*] RESET
Operands:	None
Operation:	Reset all registers and flags that are affected by a $\overline{MCLR}$ Reset.
Status Affected:	All
Encoding:	`0000` `0000` `1111` `1111`
Description:	This instruction provides a way to execute a $\overline{MCLR}$ Reset in 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: [*label*] 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: [*label*] 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 eight-bit literal 'k'. The program counter is loaded from the top of the stack (the return address). The high address latch (PCLATH) remains unchanged.

Words: 1

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Pop PC from stack, Write to W
No operation	No operation	No operation	No operation

Example:

```
    CALL TABLE ; W contains table
               ; offset value
               ; W now has
               ; table value
   :
TABLE
    ADDWF PCL  ; W = offset
    RETLW k0   ; Begin table
    RETLW k1   ;
   :
   :
    RETLW kn   ; End of table
```

Before Instruction

W = 0x07

After Instruction

W = value of kn

RETURN — Return from Subroutine

Syntax:	[*label*] 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 Interrupt
PC = TOS

RLCF — Rotate Left f through Carry

Syntax:	[*label*] 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 will be selected, overriding the BSR value. If ‘a’ = `1`, then the bank will be selected as per the BSR value (default).

Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register ‘f’	Process Data	Write to destination

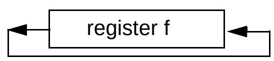
Example: `RLCF REG, 0, 0`

Before Instruction
REG = `1110 0110`
C = 0

After Instruction
REG = `1110 0110`
W = `1100 1100`
C = 1

RLNCF — Rotate Left f (no carry)

Syntax:	[*label*] 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 will be selected, overriding the BSR value. If ‘a’ is ‘1’, then the bank will be selected as per the BSR value (default).



Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register ‘f’	Process Data	Write to destination

<u>Example</u>: `RLNCF REG, 1, 0`

Before Instruction
REG = `1010 1011`
After Instruction
REG = `0101 0111`

RRCF — Rotate Right f through Carry

Syntax:	[*label*] 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 will be selected, overriding the BSR value. If ‘a’ is ‘1’, then the bank will be selected as per the BSR value (default).

Words:	1
Cycles:	1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register ‘f’	Process Data	Write to destination

<u>Example</u>: `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: [*label*] RRNCF f [,d [,a]]

Operands: 0 ≤ f ≤ 255
d ∈ [0,1]
a ∈ [0,1]

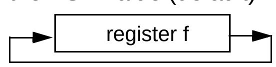
Operation: (f<n>) → dest<n-1>,
(f<0>) → dest<7>

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, overriding the BSR value. If 'a' is '1', then the bank will be selected as per the BSR value (default).



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: [*label*] 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 will be selected, overriding the BSR value. If 'a' is '1', then the bank will be selected as per the BSR value (default).

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 = 0x5A
After Instruction
REG = 0xFF

SLEEP — Enter Sleep mode

Syntax: [*label*] SLEEP

Operands: None

Operation: 00h → WDT,
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.

SUBFWP — Subtract f from W with borrow

Syntax: [*label*] SUBFWB f [,d [,a]]

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

Operation: (W) – (f) – ($\overline{C}$) → dest

Status Affected: N, OV, C, DC, Z

Encoding:

0101	01da	ffff	ffff

Description: Subtract register ‘f’ and carry flag (borrow) from W (2’s complement method). If ‘d’ is ‘0’, the result is stored in W. If ‘d’ is ‘1’, the result is stored in register ‘f’ (default). If ‘a’ is ‘0’, the Access Bank will be selected, overriding the BSR value. If ‘a’ is ‘1’, then the bank will be selected as per the BSR value (default).

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:	[*label*] SUBLW k
Operands:	$0 \le k \le 255$
Operation:	k – (W) → W
Status Affected:	N, OV, C, DC, Z
Encoding:	`0000` `1000` `kkkk` `kkkk`
Description:	W is subtracted from the eight-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 0x02`

Before Instruction
W = 1
C = ?

After Instruction
W = 1
C = 1 ; result is positive
Z = 0
N = 0

Example 2: `SUBLW 0x02`

Before Instruction
W = 2
C = ?

After Instruction
W = 0
C = 1 ; result is zero
Z = 1
N = 0

Example 3: `SUBLW 0x02`

Before Instruction
W = 3
C = ?

After Instruction
W = FF ; (2's complement)
C = 0 ; result is negative
Z = 0
N = 1

SUBWF	Subtract W from f
Syntax:	[*label*] SUBWF f [,d [,a]]
Operands:	$0 \le f \le 255$ $d \in [0,1]$ $a \in [0,1]$
Operation:	(f) – (W) → dest
Status Affected:	N, OV, C, DC, Z
Encoding:	`0101` `11da` `ffff` `ffff`
Description:	Subtract W from register 'f' (2's complement method). If 'd' is '`0`', the result is stored in W. If 'd' is '`1`', the result is stored back in register 'f' (default). If 'a' is '`0`', the Access Bank will be selected, overriding the BSR value. If 'a' is '`1`', then the bank will be selected as per the BSR value (default).
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:	[*label*] 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 will be selected, overriding the BSR value. If 'a' is '1', then the bank will be selected as per the BSR value (default).
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  =  0x19   (0001 1001)
W    =  0x0D   (0000 1101)
C    =  1
```
After Instruction
```
REG  =  0x0C   (0000 1011)
W    =  0x0D   (0000 1101)
C    =  1
Z    =  0
N    =  0     ; result is positive
```

Example 2: `SUBWFB REG, 0, 0`

Before Instruction
```
REG  =  0x1B   (0001 1011)
W    =  0x1A   (0001 1010)
C    =  0
```
After Instruction
```
REG  =  0x1B   (0001 1011)
W    =  0x00
C    =  1
Z    =  1     ; result is zero
N    =  0
```

Example 3: `SUBWFB REG, 1, 0`

Before Instruction
```
REG  =  0x03   (0000 0011)
W    =  0x0E   (0000 1101)
C    =  1
```
After Instruction
```
REG  =  0xF5   (1111 0100)
               ; [2's comp]
W    =  0x0E   (0000 1101)
C    =  0
Z    =  0
N    =  1     ; result is negative
```

SWAPF **Swap f**

Syntax:	[*label*] 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 will be selected, overriding the BSR value. If 'a' is '1', then the bank will be selected as per the BSR value (default).
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  =  0x53
```
After Instruction
```
REG  =  0x35
```

TBLRD	Table Read
Syntax:	[*label*] TBLRD (*; *+; *-; +*)
Operands:	None
Operation:	if TBLRD *, (Prog Mem (TBLPTR)) → TABLAT; TBLPTR – No Change; if TBLRD *+, (Prog Mem (TBLPTR)) → TABLAT; (TBLPTR) + 1 → TBLPTR; if TBLRD *-, (Prog Mem (TBLPTR)) → TABLAT; (TBLPTR) – 1 → TBLPTR; if TBLRD +*, (TBLPTR) + 1 → TBLPTR; (Prog Mem (TBLPTR)) → TABLAT;
Status Affected:	None

Encoding:

0000	0000	0000	10nn nn=0 * =1 *+ =2 *- =3 +*

Description: This instruction is used to read the contents of Program Memory (P.M.). To address the program memory, a pointer called Table Pointer (TBLPTR) is used. The TBLPTR (a 21-bit pointer) points to each byte in the program memory. TBLPTR has a 2-Mbyte address range.

TBLPTR[0] = 0: Least Significant Byte of Program Memory Word

TBLPTR[0] = 1: Most Significant Byte of Program Memory Word

The `TBLRD` instruction can modify the value of TBLPTR as follows:

- no change
- post-increment
- post-decrement
- pre-increment

Words: 1

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	No operation	No operation
No operation	No operation (Read Program Memory)	No operation	No operation (Write TABLAT)

TBLRD Table Read (Continued)

Example1: `TBLRD *+ ;`

Before Instruction

TABLAT	=	0x55
TBLPTR	=	0x00A356
MEMORY(0x00A356)	=	0x34

After Instruction

TABLAT	=	0x34
TBLPTR	=	0x00A357

Example2: `TBLRD +* ;`

Before Instruction

TABLAT	=	0xAA
TBLPTR	=	0x01A357
MEMORY(0x01A357)	=	0x12
MEMORY(0x01A358)	=	0x34

After Instruction

TABLAT	=	0x34
TBLPTR	=	0x01A358

TBLWT **Table Write**

Syntax: [*label*] TBLWT (*; *+; *-; +*)

Operands: None

Operation:
if TBLWT*,
(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 3 LSBs of TBLPTR to determine which of the 8 holding registers the TABLAT is written to. The holding registers are used to program the contents of Program Memory (P.M.). (Refer to **Section 5.0 "Flash Program Memory"** for additional details on programming Flash memory.)
The TBLPTR (a 21-bit pointer) points to each byte in the program memory. TBLPTR has a 2-MBtye 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

TBLWT **Table Write (Continued)**

Words: 1

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	No operation	No operation
No operation	No operation (Read TABLAT)	No operation	No operation (Write to Holding Register)

Example1: `TBLWT *+;`

Before Instruction

TABLAT	=	0x55
TBLPTR	=	0x00A356
HOLDING REGISTER (0x00A356)	=	0xFF

After Instructions (table write completion)

TABLAT	=	0x55
TBLPTR	=	0x00A357
HOLDING REGISTER (0x00A356)	=	0x55

Example 2: `TBLWT +*;`

Before Instruction

TABLAT	=	0x34
TBLPTR	=	0x01389A
HOLDING REGISTER (0x01389A)	=	0xFF
HOLDING REGISTER (0x01389B)	=	0xFF

After Instruction (table write completion)

TABLAT	=	0x34
TBLPTR	=	0x01389B
HOLDING REGISTER (0x01389A)	=	0xFF
HOLDING REGISTER (0x01389B)	=	0x34

TSTFSZ — Test f, skip if 0

Syntax: [*label*] TSTFSZ f [,a]

Operands: $0 \le f \le 255$
$a \in [0,1]$

Operation: skip if f = 0

Status Affected: None

Encoding:

0110	011a	ffff	ffff

Description: If 'f' = 0, the next instruction, fetched during the current instruction execution is discarded and a NOP is executed, making this a two-cycle instruction. If 'a' is '0', the Access Bank will be selected, overriding the BSR value. If 'a' is '1', then the bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1(2)
Note: 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	No operation

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example:

```
HERE     TSTFSZ  CNT, 1
NZERO    :
ZERO     :
```

Before Instruction
PC = Address (HERE)

After Instruction
If CNT = 0x00,
PC = Address (ZERO)
If CNT ≠ 0x00,
PC = Address (NZERO)

XORLW — Exclusive OR literal with W

Syntax: [*label*] XORLW k

Operands: $0 \le k \le 255$

Operation: (W) .XOR. k → W

Status Affected: N, Z

Encoding:

0000	1010	kkkk	kkkk

Description: The contents of W are XORed with the 8-bit literal 'k'. The result is placed in W.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Write to W

Example: `XORLW 0xAF`

Before Instruction
W = 0xB5

After Instruction
W = 0x1A

XORWF	**Exclusive OR W with f**
Syntax:	[*label*] 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 will be selected, overriding the BSR value. If 'a' is '1', then the bank will be selected as per the BSR value (default).
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 = 0xAF
W = 0xB5

After Instruction

REG = 0x1A
W = 0xB5

26.0 DEVELOPMENT SUPPORT

The PICmicro® microcontrollers are supported with a full range of hardware and software development tools:

- Integrated Development Environment
  - MPLAB® IDE Software
- Assemblers/Compilers/Linkers
  - MPASM™ Assembler
  - MPLAB C17 and MPLAB C18 C Compilers
  - MPLINK™ Object Linker/ MPLIB™ Object Librarian
  - MPLAB C30 C Compiler
  - MPLAB ASM30 Assembler/Linker/Library
- Simulators
  - MPLAB SIM Software Simulator
  - MPLAB dsPIC30 Software Simulator
- Emulators
  - MPLAB ICE 2000 In-Circuit Emulator
  - MPLAB ICE 4000 In-Circuit Emulator
- In-Circuit Debugger
  - MPLAB ICD 2
- Device Programmers
  - PRO MATE® II Universal Device Programmer
  - PICSTART® Plus Development Programmer
  - MPLAB PM3 Device Programmer
- Low-Cost Demonstration Boards
  - PICDEM™ 1 Demonstration Board
  - PICDEM.net™ Demonstration Board
  - PICDEM 2 Plus Demonstration Board
  - PICDEM 3 Demonstration Board
  - PICDEM 4 Demonstration Board
  - PICDEM 17 Demonstration Board
  - PICDEM 18R Demonstration Board
  - PICDEM LIN Demonstration Board
  - PICDEM USB Demonstration Board
- Evaluation Kits
  - KEELOQ®
  - PICDEM MSC
  - microID®
  - CAN
  - PowerSmart®
  - Analog

26.1 MPLAB Integrated Development Environment Software

The MPLAB IDE software brings an ease of software development previously unseen in the 8/16-bit microcontroller market. The MPLAB IDE is a Windows® based application that contains:

- An interface to debugging tools
  - simulator
  - programmer (sold separately)
  - emulator (sold separately)
  - in-circuit debugger (sold separately)
- A full-featured editor with color coded context
- A multiple project manager
- Customizable data windows with direct edit of contents
- High-level source code debugging
- Mouse over variable inspection
- Extensive on-line help

The MPLAB IDE allows you to:

- Edit your source files (either assembly or C)
- One touch assemble (or compile) and download to PICmicro emulator and simulator tools (automatically updates all project information)
- Debug using:
  - source files (assembly or C)
  - mixed assembly and C
  - machine code

MPLAB IDE supports multiple debugging tools in a single development paradigm, from the cost effective simulators, through low-cost in-circuit debuggers, to full-featured emulators. This eliminates the learning curve when upgrading to tools with increasing flexibility and power.

26.2 MPASM Assembler

The MPASM assembler is a full-featured, universal macro assembler for all PICmicro 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 IDE projects
- User defined macros to streamline assembly code
- Conditional assembly for multi-purpose source files
- Directives that allow complete control over the assembly process

26.3 MPLAB C17 and MPLAB C18 C Compilers

The MPLAB C17 and MPLAB C18 Code Development Systems are complete ANSI C compilers for Microchip's PIC17CXXX and PIC18CXXX family of microcontrollers. These compilers provide powerful integration capabilities, superior code optimization and ease of use not found with other compilers.

For easy source level debugging, the compilers provide symbol information that is optimized to the MPLAB IDE debugger.

26.4 MPLINK Object Linker/ MPLIB Object Librarian

The MPLINK object linker combines relocatable objects created by the MPASM assembler and the MPLAB C17 and MPLAB C18 C compilers. 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

26.5 MPLAB C30 C Compiler

The MPLAB C30 C compiler is a full-featured, ANSI compliant, optimizing compiler that translates standard ANSI C programs into dsPIC30F assembly language source. The compiler also supports many command line options and language extensions to take full advantage of the dsPIC30F device hardware capabilities and afford fine control of the compiler code generator.

MPLAB C30 is distributed with a complete ANSI C standard library. All library functions have been validated and conform to the ANSI C library standard. The library includes functions for string manipulation, dynamic memory allocation, data conversion, timekeeping and math functions (trigonometric, exponential and hyperbolic). The compiler provides symbolic information for high-level source debugging with the MPLAB IDE.

26.6 MPLAB ASM30 Assembler, Linker and Librarian

MPLAB ASM30 assembler produces relocatable machine code from symbolic assembly language for dsPIC30F devices. MPLAB C30 compiler uses the assembler to produce it's 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 dsPIC30F instruction set
- Support for fixed-point and floating-point data
- Command line interface
- Rich directive set
- Flexible macro language
- MPLAB IDE compatibility

26.7 MPLAB SIM Software Simulator

The MPLAB SIM software simulator allows code development in a PC hosted environment by simulating the PICmicro series microcontrollers on an instruction level. On any given instruction, the data areas can be examined or modified and stimuli can be applied from a file, or user defined key press, to any pin. The execution can be performed in Single-Step, Execute Until Break or Trace mode.

The MPLAB SIM simulator fully supports symbolic debugging using the MPLAB C17 and MPLAB C18 C Compilers, as well as the MPASM assembler. The software simulator offers the flexibility to develop and debug code outside of the laboratory environment, making it an excellent, economical software development tool.

26.8 MPLAB SIM30 Software Simulator

The MPLAB SIM30 software simulator allows code development in a PC hosted environment by simulating the dsPIC30F series microcontrollers on an instruction level. On any given instruction, the data areas can be examined or modified and stimuli can be applied from a file, or user defined key press, to any of the pins.

The MPLAB SIM30 simulator fully supports symbolic debugging using the MPLAB C30 C Compiler and MPLAB ASM30 assembler. The simulator runs in either a Command Line mode for automated tasks, or from MPLAB IDE. This high-speed simulator is designed to debug, analyze and optimize time intensive DSP routines.

26.9 MPLAB ICE 2000 High-Performance Universal In-Circuit Emulator

The MPLAB ICE 2000 universal in-circuit emulator is intended to provide the product development engineer with a complete microcontroller design tool set for PICmicro microcontrollers. Software control of the MPLAB ICE 2000 in-circuit emulator is advanced by the MPLAB Integrated Development Environment, which allows editing, building, downloading and source debugging from a single environment.

The MPLAB ICE 2000 is a full-featured emulator system with enhanced trace, trigger and data monitoring features. Interchangeable processor modules allow the system to be easily reconfigured for emulation of different processors. The universal architecture of the MPLAB ICE in-circuit emulator allows expansion to support new PICmicro microcontrollers.

The MPLAB ICE 2000 in-circuit emulator system has been designed as a real-time emulation system with advanced features that are typically found on more expensive development tools. The PC platform and Microsoft® Windows 32-bit operating system were chosen to best make these features available in a simple, unified application.

26.10 MPLAB ICE 4000 High-Performance Universal In-Circuit Emulator

The MPLAB ICE 4000 universal in-circuit emulator is intended to provide the product development engineer with a complete microcontroller design tool set for high-end PICmicro microcontrollers. Software control of the MPLAB ICE in-circuit emulator is provided by the MPLAB Integrated Development Environment, which allows editing, building, downloading and source debugging from a single environment.

The MPLAB ICD 4000 is a premium emulator system, providing the features of MPLAB ICE 2000, but with increased emulation memory and high-speed performance for dsPIC30F and PIC18XXXX devices. Its advanced emulator features include complex triggering and timing, up to 2 Mb of emulation memory and the ability to view variables in real-time.

The MPLAB ICE 4000 in-circuit emulator system has been designed as a real-time emulation system with advanced features that are typically found on more expensive development tools. The PC platform and Microsoft Windows 32-bit operating system were chosen to best make these features available in a simple, unified application.

26.11 MPLAB ICD 2 In-Circuit Debugger

Microchip's In-Circuit Debugger, MPLAB ICD 2, is a powerful, low-cost, run-time development tool, connecting to the host PC via an RS-232 or high-speed USB interface. This tool is based on the Flash PICmicro MCUs and can be used to develop for these and other PICmicro microcontrollers. The MPLAB ICD 2 utilizes the in-circuit debugging capability built into the Flash devices. This feature, along with Microchip's In-Circuit Serial Programming™ (ICSP™) protocol, offers cost effective in-circuit Flash debugging from the graphical user interface of the MPLAB Integrated Development Environment. This enables a designer to develop and debug source code by setting breakpoints, single-stepping and watching variables, CPU status and peripheral registers. Running at full speed enables testing hardware and applications in real-time. MPLAB ICD 2 also serves as a development programmer for selected PICmicro devices.

26.12 PRO MATE II Universal Device Programmer

The PRO MATE II is a universal, CE compliant device programmer with programmable voltage verification at V_{DDMIN} and V_{DDMAX} for maximum reliability. It features an LCD display for instructions and error messages and a modular detachable socket assembly to support various package types. In Stand-Alone mode, the PRO MATE II device programmer can read, verify and program PICmicro devices without a PC connection. It can also set code protection in this mode.

26.13 MPLAB PM3 Device Programmer

The MPLAB PM3 is a universal, CE compliant device programmer with programmable voltage verification at V_{DDMIN} and V_{DDMAX} 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 PICmicro devices without a PC connection. It can also set code protection in this mode. MPLAB PM3 connects to the host PC via an RS-232 or USB cable. MPLAB PM3 has high-speed communications and optimized algorithms for quick programming of large memory devices and incorporates an SD/MMC card for file storage and secure data applications.

26.14 PICSTART Plus Development Programmer

The PICSTART Plus development programmer is an easy-to-use, low-cost, prototype programmer. It connects to the PC via a COM (RS-232) port. MPLAB Integrated Development Environment software makes using the programmer simple and efficient. The PICSTART Plus development programmer supports most PICmicro devices up to 40 pins. Larger pin count devices, such as the PIC16C92X and PIC17C76X, may be supported with an adapter socket. The PICSTART Plus development programmer is CE compliant.

26.15 PICDEM 1 PICmicro Demonstration Board

The PICDEM 1 demonstration board demonstrates the capabilities of the PIC16C5X (PIC16C54 to PIC16C58A), PIC16C61, PIC16C62X, PIC16C71, PIC16C8X, PIC17C42, PIC17C43 and PIC17C44. All necessary hardware and software is included to run basic demo programs. The sample microcontrollers provided with the PICDEM 1 demonstration board can be programmed with a PRO MATE II device programmer or a PICSTART Plus development programmer. The PICDEM 1 demonstration board can be connected to the MPLAB ICE in-circuit emulator for testing. A prototype area extends the circuitry for additional application components. Features include an RS-232 interface, a potentiometer for simulated analog input, push button switches and eight LEDs.

26.16 PICDEM.net Internet/Ethernet Demonstration Board

The PICDEM.net demonstration board is an Internet/Ethernet demonstration board using the PIC18F452 microcontroller and TCP/IP firmware. The board supports any 40-pin DIP device that conforms to the standard pinout used by the PIC16F877 or PIC18C452. This kit features a user friendly TCP/IP stack, web server with HTML, a 24L256 Serial EEPROM for Xmodem download to web pages into Serial EEPROM, ICSP/MPLAB ICD 2 interface connector, an Ethernet interface, RS-232 interface and a 16 x 2 LCD display. Also included is the book and CD-ROM *"TCP/IP Lean, Web Servers for Embedded Systems,"* by Jeremy Bentham

26.17 PICDEM 2 Plus Demonstration Board

The PICDEM 2 Plus demonstration board supports many 18, 28 and 40-pin microcontrollers, including PIC16F87X and PIC18FXX2 devices. All the necessary hardware and software is included to run the demonstration programs. The sample microcontrollers provided with the PICDEM 2 demonstration board can be programmed with a PRO MATE II device programmer, PICSTART Plus development programmer, or MPLAB ICD 2 with a Universal Programmer Adapter. The MPLAB ICD 2 and MPLAB ICE in-circuit emulators may also be used with the PICDEM 2 demonstration board to test firmware. A prototype area extends the circuitry for additional application components. Some of the features include an RS-232 interface, a 2 x 16 LCD display, a piezo speaker, an on-board temperature sensor, four LEDs and sample PIC18F452 and PIC16F877 Flash microcontrollers.

26.18 PICDEM 3 PIC16C92X Demonstration Board

The PICDEM 3 demonstration board supports the PIC16C923 and PIC16C924 in the PLCC package. All the necessary hardware and software is included to run the demonstration programs.

26.19 PICDEM 4 8/14/18-Pin Demonstration Board

The PICDEM 4 can be used to demonstrate the capabilities of the 8, 14 and 18-pin PIC16XXXX and PIC18XXXX MCUs, including the PIC16F818/819, PIC16F87/88, PIC16F62XA and the PIC18F1320 family of microcontrollers. PICDEM 4 is intended to showcase the many features of these low pin count parts, including LIN and Motor Control using ECCP. Special provisions are made for low-power operation with the supercapacitor circuit and jumpers allow on-board hardware to be disabled to eliminate current draw in this mode. Included on the demo board are provisions for Crystal, RC or Canned Oscillator modes, a five volt regulator for use with a nine volt wall adapter or battery, DB-9 RS-232 interface, ICD connector for programming via ICSP and development with MPLAB ICD 2, 2 x 16 liquid crystal display, PCB footprints for H-Bridge motor driver, LIN transceiver and EEPROM. Also included are: header for expansion, eight LEDs, four potentiometers, three push buttons and a prototyping area. Included with the kit is a PIC16F627A and a PIC18F1320. Tutorial firmware is included along with the User's Guide.

26.20 PICDEM 17 Demonstration Board

The PICDEM 17 demonstration board is an evaluation board that demonstrates the capabilities of several Microchip microcontrollers, including PIC17C752, PIC17C756A, PIC17C762 and PIC17C766. A programmed sample is included. The PRO MATE II device programmer, or the PICSTART Plus development programmer, can be used to reprogram the device for user tailored application development. The PICDEM 17 demonstration board supports program download and execution from external on-board Flash memory. A generous prototype area is available for user hardware expansion.

26.21 PICDEM 18R PIC18C601/801 Demonstration Board

The PICDEM 18R demonstration board serves to assist development of the PIC18C601/801 family of Microchip microcontrollers. It provides hardware implementation of both 8-bit Multiplexed/Demultiplexed and 16-bit Memory modes. The board includes 2 Mb external Flash memory and 128 Kb SRAM memory, as well as serial EEPROM, allowing access to the wide range of memory types supported by the PIC18C601/801.

26.22 PICDEM LIN PIC16C43X Demonstration Board

The powerful LIN hardware and software kit includes a series of boards and three PICmicro microcontrollers. The small footprint PIC16C432 and PIC16C433 are used as slaves in the LIN communication and feature on-board LIN transceivers. A PIC16F874 Flash microcontroller serves as the master. All three microcontrollers are programmed with firmware to provide LIN bus communication.

26.23 PICkit™ 1 Flash Starter Kit

A complete “development system in a box”, the PICkit Flash Starter Kit includes a convenient multi-section board for programming, evaluation and development of 8/14-pin Flash PIC® microcontrollers. Powered via USB, the board operates under a simple Windows GUI. The PICkit 1 Starter Kit includes the User’s Guide (on CD ROM), PICkit 1 tutorial software and code for various applications. Also included are MPLAB® IDE (Integrated Development Environment) software, software and hardware “Tips 'n Tricks for 8-pin Flash PIC® Microcontrollers” Handbook and a USB interface cable. Supports all current 8/14-pin Flash PIC microcontrollers, as well as many future planned devices.

26.24 PICDEM USB PIC16C7X5 Demonstration Board

The PICDEM USB Demonstration Board shows off the capabilities of the PIC16C745 and PIC16C765 USB microcontrollers. This board provides the basis for future USB products.

26.25 Evaluation and Programming Tools

In addition to the PICDEM series of circuits, Microchip has a line of evaluation kits and demonstration software for these products.

- KEELOQ evaluation and programming tools for Microchip’s HCS Secure Data Products
- CAN developers kit for automotive network applications
- Analog design boards and filter design software
- PowerSmart battery charging evaluation/ calibration kits
- IrDA® development kit
- microID development and rfLab™ development software
- SEEVAL® designer kit for memory evaluation and endurance calculations
- PICDEM MSC demo boards for Switching mode power supply, high-power IR driver, delta sigma ADC and flow rate sensor

Check the Microchip web page and the latest Product Selector Guide for the complete list of demonstration and evaluation kits.

NOTES:

27.0 ELECTRICAL CHARACTERISTICS

Absolute Maximum Ratings (†)

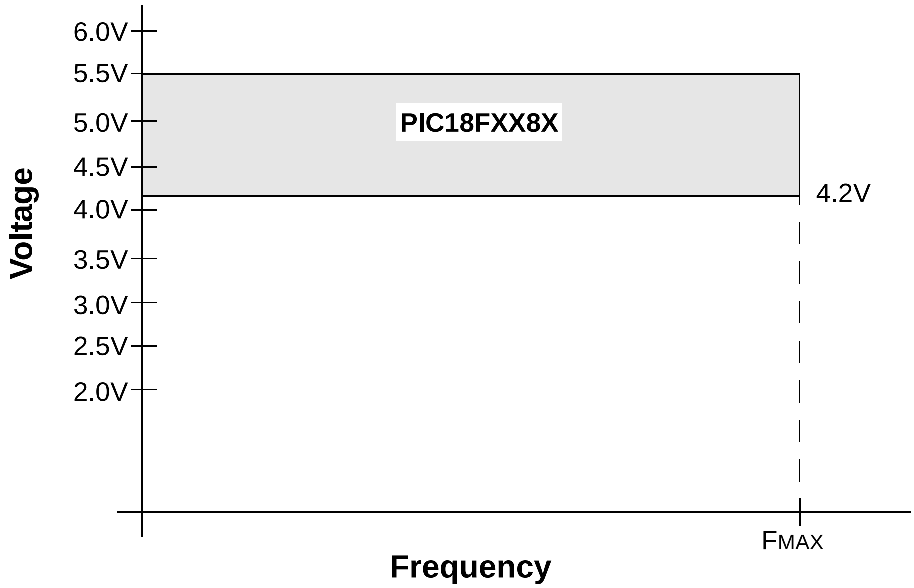
Ambient temperature under bias	-55°C to +125°C
Storage temperature	-65°C to +150°C
Voltage on any pin with respect to V_{SS} (except V_{DD}, $\overline{MCLR}$, and RA4)	-0.3V to (V_{DD} + 0.3V)
Voltage on V_{DD} with respect to V_{SS}	-0.3V to +5.5V
Voltage on $\overline{MCLR}$ with respect to V_{SS} **(Note 2)**	0V to +13.25V
Voltage on RA4 with respect to Vss	0V to +8.5V
Total power dissipation **(Note 1)**	1.0W
Maximum current out of V_{SS} pin	300 mA
Maximum current into V_{DD} pin	250 mA
Input clamp current, I_{IK} (V_I < 0 or V_I > V_{DD})	±20 mA
Output clamp current, I_{OK} (V_O < 0 or V_O > V_{DD})	±20 mA
Maximum output current sunk by any I/O pin	25 mA
Maximum output current sourced by any I/O pin	25 mA
Maximum current sunk by all ports	200 mA
Maximum current sourced by all ports	200 mA

Note 1: Power dissipation is calculated as follows:
$Pdis = V_{DD} \times \{I_{DD} - \sum I_{OH}\} + \sum \{(V_{DD} - V_{OH}) \times I_{OH}\} + \sum(V_{OL} \times I_{OL})$

2: Voltage spikes below V_{SS} at the $\overline{MCLR}$/V_{PP} pin, inducing currents greater than 80 mA, may cause latch-up. Thus, a series resistor of 50-100Ω should be used when applying a "low" level to the $\overline{MCLR}$/V_{PP} pin rather than pulling this pin directly to V_{SS}.

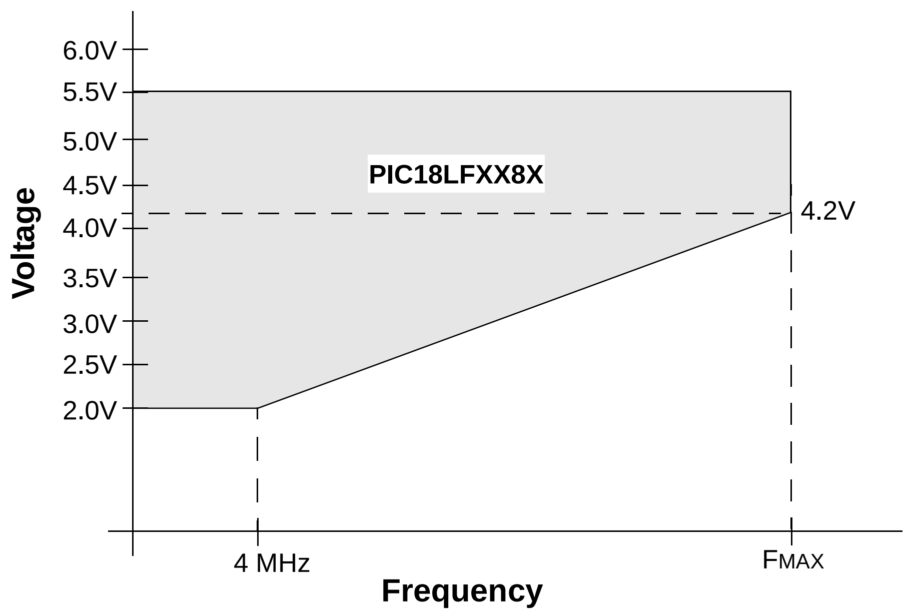
† NOTICE: Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at those or any other conditions above those indicated in the operation listings of this specification is not implied. Exposure to maximum rating conditions for extended periods may affect device reliability.

FIGURE 27-1: PIC18F6585/8585/6680/8680 VOLTAGE-FREQUENCY GRAPH (INDUSTRIAL)



FMAX = 40 MHz for PIC18F6X8X and PIC18F8X8X in Microcontroller mode.
FMAX = 25 MHz for PIC18F8X8X in modes other than Microcontroller mode.

FIGURE 27-2: PIC18LF6585/8585/6680/8680 VOLTAGE-FREQUENCY GRAPH (INDUSTRIAL)

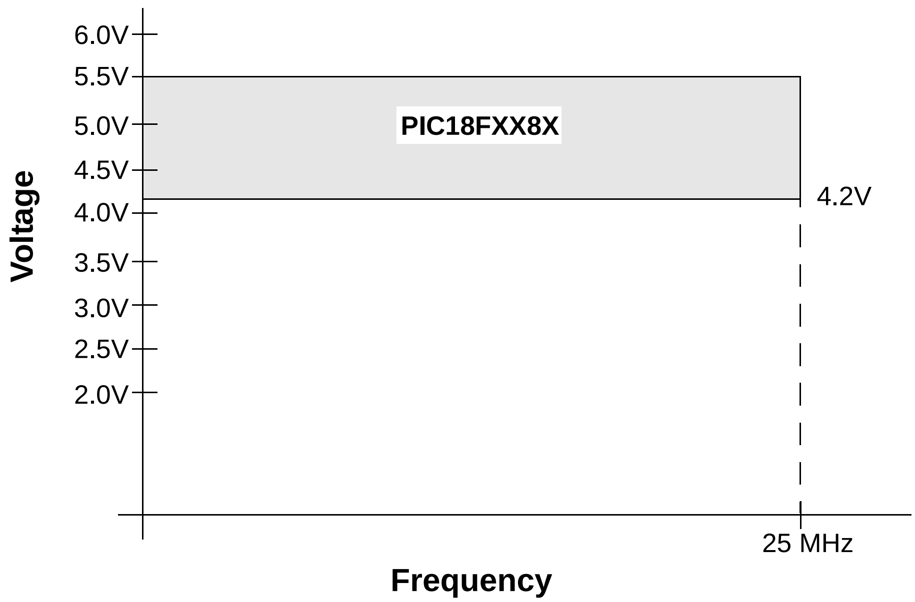


For PIC18F6X8X and PIC18F8X8X in Microcontroller mode:
FMAX = (16.36 MHz/V) (VDDAPPMIN – 2.0V) + 4 MHz, if VDDAPPMIN ≤ 4.2V;
FMAX = 40 MHz, if VDDAPPMIN > 4.2V.

For PIC18F8X8X in modes other than Microcontroller mode:
FMAX = (9.55 MHz/V) (VDDAPPMIN – 2.0V) + 4 MHz, if VDDAPPMIN ≤ 4.2V;
FMAX = 25 MHz, if VDDAPPMIN > 4.2V.

Note: VDDAPPMIN is the minimum voltage of the PICmicro® device in the application.

FIGURE 27-3: PIC18F6585/8585/6680/8680 VOLTAGE-FREQUENCY GRAPH (EXTENDED)



27.1 DC Characteristics: Supply Voltage PIC18FXX8X (Industrial, Extended) PIC18LFXX8X (Industrial)

PIC18LFXX8X (Industrial)			**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial				
PIC18FXX8X (Industrial, Extended)			**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial -40°C ≤ TA ≤ +125°C for extended				
Param. No.	**Symbol**	**Characteristic**	**Min**	**Typ**	**Max**	**Units**	**Conditions**
D001	VDD	**Supply Voltage**					
		PIC18LFXX8X	2.0	—	5.5	V	HS, XT, RC and LP Oscillator mode
		PIC18FXX8X	4.2	—	5.5	V	
D001A	AVDD	**Analog Supply Voltage**	VDD – 0.3	—	VDD + 0.3	V	
D002	VDR	**RAM Data Retention Voltage**(1)	1.5	—	—	V	
D003	VPOR	**VDD Start Voltage** to ensure internal Power-on Reset signal	—	—	0.7	V	See section on Power-on Reset for details
D004	SVDD	**VDD Rise Rate** to ensure internal Power-on Reset signal	0.05	—	—	V/ms	See section on Power-on Reset for details
D005	VBOR	**Brown-out Reset Voltage**					
		BORV1:BORV0 = 11	1.96	—	2.18	V	
		BORV1:BORV0 = 10	2.64	—	2.92	V	
		BORV1:BORV0 = 01	4.11	—	4.55	V	
		BORV1:BORV0 = 00	4.41	—	4.87	V	

Legend: Shading of rows is to assist in readability of the table.

Note 1: This is the limit to which VDD can be lowered in Sleep mode, or during a device Reset, without losing RAM data.

27.2 DC Characteristics: Power-down and Supply Current PIC18FXX8X (Industrial, Extended) PIC18LFXX8X (Industrial)

PIC18LFXX8X (Industrial)	Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C for industrial					
PIC18FXX8X (Industrial, Extended)	**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial -40°C ≤ TA ≤ +125°C for extended					
Param. No.	**Device**	**Typ**	**Max**	**Units**	**Conditions**	
	Power-down Current (IPD)(1)					
D020	PIC18LFXX8X	0.2	1	μA	-40°C	VDD = 2.0V, (Sleep mode)
		0.2	1	μA	+25°C	
		5.0	10	μA	+85°C	
D020A	PIC18LFXX8X	0.4	1	μA	-40°C	VDD = 3.0V, (Sleep mode)
		0.4	1	μA	+25°C	
		3.0	18	μA	+85°C	
D020B	All devices	0.7	2	μA	-40°C	VDD = 5.0V, (Sleep mode)
		0.7	2	μA	+25°C	
		15.0	32	μA	+85°C	

Legend: Shading of rows is to assist in readability of the table.

Note 1: 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 VDD or VSS and all features that add delta current disabled (such as WDT, Timer1 Oscillator, BOR, etc.).

2: The supply current is mainly a function of operating voltage, frequency and mode. Other factors, such as I/O pin loading and switching rate, oscillator type and circuit, internal code execution pattern and temperature, also have an impact on the current consumption.
The test conditions for all IDD measurements in active operation mode are:
OSC1 = external square wave, from rail-to-rail; all I/O pins tri-stated, pulled to VDD;
MCLR = VDD; WDT enabled/disabled as specified.

3: For RC oscillator configurations, current through REXT is not included. The current through the resistor can be estimated by the formula Ir = VDD/2REXT (mA) with REXT in kΩ.

27.2 DC Characteristics: Power-down and Supply Current PIC18FXX8X (Industrial, Extended) PIC18LFXX8X (Industrial) (Continued)

PIC18LFXX8X (Industrial)		Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C for industrial					
PIC18FXX8X (Industrial, Extended)		**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial -40°C ≤ TA ≤ +125°C for extended					
Param. No.	**Device**	**Typ**	**Max**	**Units**	**Conditions**		
	Supply Current (IDD)(2,3)						
D010	PIC18LFXX8X	500	500	µA	-40°C	VDD = 2.0V	FOSC = 1 MHz, EC oscillator
		300	500	µA	+25°C		
		850	1000	µA	+85°C		
	PIC18LFXX8X	500	900	µA	-40°C	VDD = 3.0V	
		500	900	µA	+25°C		
		1	1.5	mA	+85°C		
	All devices	1	2	mA	-40°C	VDD = 5.0V	
		1	2	mA	+25°C		
		1.3	3	mA	+85°C		
	PIC18LFXX8X	1	2	mA	-40°C	VDD = 2.0V	FOSC = 4 MHz, EC oscillator
		1	2	mA	+25°C		
		1.5	2.5	mA	+85°C		
	PIC18LFXX8X	1.5	2	mA	-40°C	VDD = 3.0V	
		1.5	2	mA	+25°C		
		2	2.5	mA	+85°C		
	All devices	3	5	mA	-40°C	VDD = 5.0V	
		3	5	mA	+25°C		
		4	6	mA	+85°C		

Legend: Shading of rows is to assist in readability of the table.

Note 1: 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 VDD or VSS and all features that add delta current disabled (such as WDT, Timer1 Oscillator, BOR, etc.).

2: The supply current is mainly a function of operating voltage, frequency and mode. Other factors, such as I/O pin loading and switching rate, oscillator type and circuit, internal code execution pattern and temperature, also have an impact on the current consumption.
The test conditions for all IDD measurements in active operation mode are:
OSC1 = external square wave, from rail-to-rail; all I/O pins tri-stated, pulled to VDD;
$\overline{MCLR}$ = VDD; WDT enabled/disabled as specified.

3: For RC oscillator configurations, current through REXT is not included. The current through the resistor can be estimated by the formula Ir = VDD/2REXT (mA) with REXT in kΩ.

27.2 DC Characteristics: Power-down and Supply Current PIC18FXX8X (Industrial, Extended) PIC18LFXX8X (Industrial) (Continued)

PIC18LFXX8X (Industrial)	Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C for industrial					
PIC18FXX8X (Industrial, Extended)	**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial -40°C ≤ TA ≤ +125°C for extended					
Param. No.	**Device**	**Typ**	**Max**	**Units**	**Conditions**		
	Supply Current (IDD)(2,3)						
	PIC18FXX8X	13	27	mA	-40°C	VDD = 4.2V	FOSC = 25 MHz, EC oscillator
		15	27	mA	+25°C		
		19	29	mA	+85°C		
	PIC18FXX8X	17	31	mA	-40°C	VDD = 5.0V	
		21	31	mA	+25°C		
		23	34	mA	+85°C		
	PIC18FXX8X	20	34	mA	-40°C	VDD = 4.2V	FOSC = 40 MHz, EC oscillator
		24	34	mA	+25°C		
		29	44	mA	+85°C		
	PIC18FXX8X	28	46	mA	-40°C	VDD = 5.0V	
		33	46	mA	+25°C		
		40	51	mA	+85°C		
D014	PIC18LFXX8X	27	45	µA	-10°C	VDD = 2.0V	FOSC = 32 kHz, Timer1 as clock
		30	50	µA	+25°C		
		32	54	µA	+70°C		
	PIC18LFXX8X	33	55	µA	-10°C	VDD = 3.0V	
		36	60	µA	+25°C		
		39	65	µA	+70°C		
	All devices	75	125	µA	-10°C	VDD = 5.0V	
		90	150	µA	+25°C		
		113	188	µA	+70°C		

Legend: Shading of rows is to assist in readability of the table.

Note 1: 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 VDD or VSS and all features that add delta current disabled (such as WDT, Timer1 Oscillator, BOR, etc.).

2: The supply current is mainly a function of operating voltage, frequency and mode. Other factors, such as I/O pin loading and switching rate, oscillator type and circuit, internal code execution pattern and temperature, also have an impact on the current consumption.
The test conditions for all IDD measurements in active operation mode are:
OSC1 = external square wave, from rail-to-rail; all I/O pins tri-stated, pulled to VDD;
MCLR = VDD; WDT enabled/disabled as specified.

3: For RC oscillator configurations, current through REXT is not included. The current through the resistor can be estimated by the formula Ir = VDD/2REXT (mA) with REXT in kΩ.

27.2 DC Characteristics: Power-down and Supply Current PIC18FXX8X (Industrial, Extended) PIC18LFXX8X (Industrial) (Continued)

PIC18LFXX8X (Industrial)		Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C for industrial					
PIC18FXX8X (Industrial, Extended)		**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial -40°C ≤ TA ≤ +125°C for extended					
Param. No.	**Device**	**Typ**	**Max**	**Units**	**Conditions**		
	Module Differential Currents (ΔIWDT, ΔIBOR, ΔILVD, ΔIOSCB, ΔIAD)						
D022 (ΔIWDT)	**Watchdog Timer**	<1	1.5	μA	-40°C	VDD = 2.0V	
		<1	2	μA	+25°C		
		5	20	μA	+85°C		
		3	10	μA	-40°C	VDD = 3.0V	
		3	20	μA	+25°C		
		10	35	μA	+85°C		
		12	25	μA	-40°C	VDD = 5.0V	
		15	35	μA	+25°C		
		20	50	μA	+85°C		
D022A (ΔIBOR)	**Brown-out Reset**	55	115	μA	-40°C to +85°C	VDD = 3.0V	
		105	175	μA	-40°C to +85°C	VDD = 5.0V	
D022B (ΔILVD)	**Low-Voltage Detect**	45	125	μA	-40°C to +85°C	VDD = 2.0V	
		45	150	μA	-40°C to +85°C	VDD = 3.0V	
		45	225	μA	-40°C to +85°C	VDD = 5.0V	
D025 (ΔIOSCB)	**Timer1 Oscillator**	20	27	μA	-10°C	VDD = 2.0V	32 kHz on Timer1
		20	30	μA	+25°C		
		25	35	μA	+70°C		
		22	60	μA	-10°C	VDD = 3.0V	32 kHz on Timer1
		22	65	μA	+25°C		
		25	75	μA	+70°C		
		30	75	μA	-10°C	VDD = 5.0V	32 kHz on Timer1
		30	85	μA	+25°C		
		35	100	μA	+70°C		
D026 (ΔIAD)	**A/D Converter**	<1	2	μA	+25°C	VDD = 2.0V	A/D on, not converting
		<1	2	μA	+25°C	VDD = 3.0V	
		<1	2	μA	+25°C	VDD = 5.0V	

Legend: Shading of rows is to assist in readability of the table.

Note 1: 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 VDD or VSS and all features that add delta current disabled (such as WDT, Timer1 Oscillator, BOR, etc.).

2: The supply current is mainly a function of operating voltage, frequency and mode. Other factors, such as I/O pin loading and switching rate, oscillator type and circuit, internal code execution pattern and temperature, also have an impact on the current consumption.
The test conditions for all IDD measurements in active operation mode are:
OSC1 = external square wave, from rail-to-rail; all I/O pins tri-stated, pulled to VDD;
MCLR = VDD; WDT enabled/disabled as specified.

3: For RC oscillator configurations, current through REXT is not included. The current through the resistor can be estimated by the formula Ir = VDD/2REXT (mA) with REXT in kΩ.

27.3 DC Characteristics: PIC18FXX8X (Industrial, Extended) PIC18LFXX8X (Industrial)

DC CHARACTERISTICS			Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C for industrial -40°C ≤ TA ≤ +125°C for extended			
Param No.	**Symbol**	**Characteristic**	**Min**	**Max**	**Units**	**Conditions**
	VIL	**Input Low Voltage**				
		I/O ports:				
D030		with TTL buffer	VSS	0.15 VDD	V	VDD < 4.5V
D030A			—	0.8	V	4.5V ≤ VDD ≤ 5.5V
D031		with Schmitt Trigger buffer RC3 and RC4	VSS VSS	0.2 VDD 0.3 VDD	V V	
D032		$\overline{\text{MCLR}}$	VSS	0.2 VDD	V	
D032A		OSC1 (in XT, HS and LP modes) and T1OSI	VSS	0.3 VDD	V	
D033		OSC1 (in RC and EC mode)(1)	VSS	0.2 VDD	V	
	VIH	**Input High Voltage**				
		I/O ports:				
D040		with TTL buffer	0.25 VDD + 0.8V	VDD	V	VDD < 4.5V
D040A			2.0	VDD	V	4.5V ≤ VDD ≤ 5.5V
D041		with Schmitt Trigger buffer RC3 and RC4	0.8 VDD 0.7 VDD	VDD VDD	V V	
D042		$\overline{\text{MCLR}}$, OSC1 (EC mode)	0.8 VDD	VDD	V	
D042A		OSC1 (in XT, HS and LP modes) and T1OSI	0.7 VDD	VDD	V	
D043		OSC1 (RC mode)(1)	0.9 VDD	VDD	V	
	IIL	**Input Leakage Current(2,3)**				
D060		I/O ports	—	±1	μA	VSS ≤ VPIN ≤ VDD, Pin at high-impedance
D061		$\overline{\text{MCLR}}$	—	±5	μA	Vss ≤ VPIN ≤ VDD
D063		OSC1	—	±5	μA	Vss ≤ VPIN ≤ VDD
	IPU	**Weak Pull-up Current**				
D070	IPURB	PORTB weak pull-up current	50	400	μA	VDD = 5V, VPIN = VSS

Note 1: In RC oscillator configuration, the OSC1/CLKI pin is a Schmitt Trigger input. It is not recommended that the PICmicro device be driven with an external clock while in RC mode.

2: The leakage current on the $\overline{\text{MCLR}}$ 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.

3: Negative current is defined as current sourced by the pin.

4: Parameter is characterized but not tested.

27.3 DC Characteristics: PIC18FXX8X (Industrial, Extended) PIC18LFXX8X (Industrial) (Continued)

DC CHARACTERISTICS			Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C for industrial -40°C ≤ TA ≤ +125°C for extended			
Param No.	**Symbol**	**Characteristic**	**Min**	**Max**	**Units**	**Conditions**
	V_{OL}	**Output Low Voltage**				
D080		I/O ports	—	0.6	V	I_{OL} = 8.5 mA, V_{DD} = 4.5V, -40°C to +85°C
D080A			—	0.6	V	I_{OL} = 7.0 mA, V_{DD} = 4.5V, -40°C to +125°C
D083		OSC2/CLKO (RC mode)	—	0.6	V	I_{OL} = 1.6 mA, V_{DD} = 4.5V, -40°C to +85°C
D083A			—	0.6	V	I_{OL} = 1.2 mA, V_{DD} = 4.5V, -40°C to +125°C
	V_{OH}	**Output High Voltage(3)**				
D090		I/O ports	V_{DD} – 0.7	—	V	I_{OH} = -3.0 mA, V_{DD} = 4.5V, -40°C to +85°C
D090A			V_{DD} – 0.7	—	V	I_{OH} = -2.5 mA, V_{DD} = 4.5V, -40°C to +125°C
D092		OSC2/CLKO (RC mode)	V_{DD} – 0.7	—	V	I_{OH} = -1.3 mA, V_{DD} = 4.5V, -40°C to +85°C
D092A			V_{DD} – 0.7	—	V	I_{OH} = -1.0 mA, V_{DD} = 4.5V, -40°C to +125°C
D150	V_{OD}	**Open-Drain High Voltage**	—	8.5	V	RA4 pin
		Capacitive Loading Specs on Output Pins				
D100(4)	C_{OSC2}	OSC2 pin	—	15	pF	In XT, HS and LP modes when external clock is used to drive OSC1
D101	C_{IO}	All I/O pins and OSC2 (in RC mode)	—	50	pF	To meet the AC Timing Specifications
D102	C_B	SCL, SDA	—	400	pF	In I^2C mode

Note 1: In RC oscillator configuration, the OSC1/CLKI pin is a Schmitt Trigger input. It is not recommended that the PICmicro device be driven with an external clock while in RC mode.

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.

3: Negative current is defined as current sourced by the pin.

4: Parameter is characterized but not tested.

TABLE 27-1: COMPARATOR SPECIFICATIONS

Operating Conditions: 3.0V < VDD < 5.5V, -40°C < TA < +125°C, unless otherwise stated							
Param No.	**Sym**	**Characteristics**	**Min**	**Typ**	**Max**	**Units**	**Comments**
D300	VIOFF	Input Offset Voltage	—	± 5.0	± 10	mV	
D301	VICM	Input Common Mode Voltage	0	—	VDD – 1.5	V	
D302	CMRR	Common Mode Rejection Ratio	55	—	—	dB	
300 300A	TRESP	Response Time(1)	—	150	400 600	ns ns	PIC18FXX8X PIC18LFXX8X
301	TMC2OV	Comparator Mode Change to Output Valid	—	—	10	µs	

Note 1: Response time measured with one comparator input at (VDD – 1.5)/2 while the other input transitions from VSS to VDD.

TABLE 27-2: VOLTAGE REFERENCE SPECIFICATIONS

Operating Conditions: 3.0V < VDD < 5.5V, -40°C < TA < +125°C, unless otherwise stated							
Param No.	**Sym**	**Characteristics**	**Min**	**Typ**	**Max**	**Units**	**Comments**
D310	VRES	Resolution	VDD/24	—	VDD/32	LSb	
D311	VRAA	Absolute Accuracy	— —	— —	1/4 1/2	LSb LSb	Low Range (VRR = 1) High Range (VRR = 0)
D312	VRUR	Unit Resistor Value (R)	—	2k	—	Ω	
310	TSET	Settling Time(1)	—	—	10	µs	

Note 1: Settling time measured while VRR = 1 and VR<3:0> transitions from 0000 to 1111.

FIGURE 27-4: LOW-VOLTAGE DETECT CHARACTERISTICS

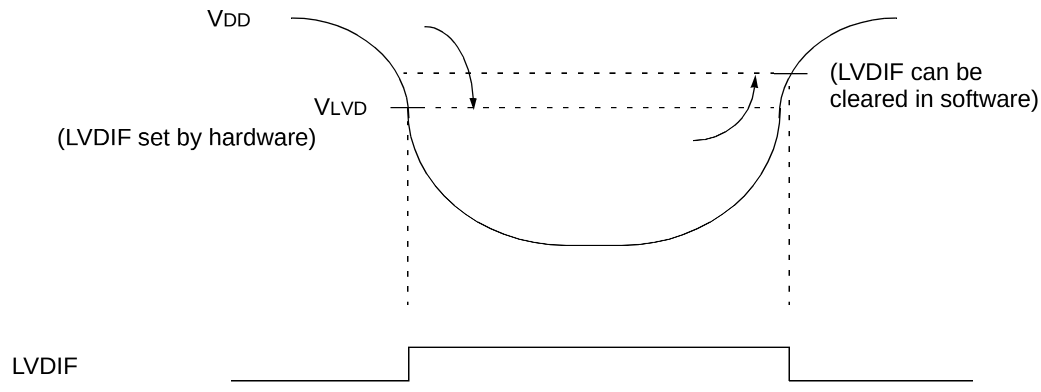


TABLE 27-3: LOW-VOLTAGE DETECT CHARACTERISTICS

				Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C for industrial -40°C ≤ TA ≤ +125°C for extended				
Param No.	**Symbol**	**Characteristic**		**Min**	**Typ†**	**Max**	**Units**	**Conditions**
D420		LVD Voltage on VDD transition high to low	LVV = 0000	—	—	—	V	
			LVV = 0001	1.96	2.06	2.16	V	
			LVV = 0010	2.16	2.27	2.38	V	
			LVV = 0011	2.35	2.47	2.59	V	
			LVV = 0100	2.46	2.58	2.71	V	
			LVV = 0101	2.64	2.78	2.92	V	
			LVV = 0110	2.75	2.89	3.03	V	
			LVV = 0111	2.95	3.1	3.26	V	
			LVV = 1000	3.24	3.41	3.58	V	
			LVV = 1001	3.43	3.61	3.79	V	
			LVV = 1010	3.53	3.72	3.91	V	
			LVV = 1011	3.72	3.92	4.12	V	
			LVV = 1100	3.92	4.13	4.33	V	
			LVV = 1101	4.11	4.33	4.55	V	
			LVV = 1110	4.41	4.64	4.87	V	
D423	VBG	Band Gap Reference Voltage Value		—	1.22	—	V	

† Production tested at TAMB = 25°C. Specifications over temperature limits ensured by characterization.

TABLE 27-4: MEMORY PROGRAMMING REQUIREMENTS

DC Characteristics			Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C for industrial -40°C ≤ TA ≤ +125°C for extended				
Param No.	**Sym**	**Characteristic**	**Min**	**Typ†**	**Max**	**Units**	**Conditions**
		Internal Program Memory Programming Specifications (Note 1)					
D110	VPP	Voltage on $\overline{MCLR}$/VPP pin	9.00	—	13.25	V	**(Note 2)**
D112	IPP	Current into $\overline{MCLR}$/VPP pin	—	—	5	μA	
D113	IDDP	Supply Current during Programming	—	—	10	mA	
		Data EEPROM Memory					
D120	ED	Cell Endurance	100K	1M	—	E/W	-40°C to +85°C
D120A	ED	Cell Endurance	10K	100K	—	E/W	+85°C to +125°C
D121	VDRW	VDD for Read/Write	VMIN	—	5.5	V	Using EECON to read/write, VMIN = Minimum operating voltage
D122	TDEW	Erase/Write Cycle Time	—	4	—	ms	
D123	TRETD	Characteristic Retention	40	—	—	Year	-40°C to +85°C **(Note 3)**
D123A	TRETD	Characteristic Retention	100	—	—	Year	25°C **(Note 3)**
		Program Flash Memory					
D130	EP	Cell Endurance	10K	100K	—	E/W	-40°C to +85°C
D130A	EP	Cell Endurance	1000	10K	—	E/W	+85°C to +125°C
D131	VPR	VDD for Read	VMIN	—	5.5	V	VMIN = Minimum operating voltage
D132	VIE	VDD for Block Erase	4.5	—	5.5	V	Using ICSP port
D132A	VIW	VDD for Externally Timed Erase or Write	4.5	—	5.5	V	Using ICSP port
D132B	VPEW	VDD for Self-timed Write	VMIN	—	5.5	V	VMIN = Minimum operating voltage
D133	TIE	ICSP Block Erase Cycle Time	—	5	—	ms	VDD > 4.5V
D133A	TIW	ICSP Erase or Write Cycle Time (externally timed)	1	—	—	ms	VDD > 4.5V
D133A	TIW	Self-timed Write Cycle Time	—	2.5	—	ms	
D134	TRETD	Characteristic Retention	40	—	—	Year	-40°C to +85°C **(Note 3)**
D134A	TRETD	Characteristic Retention	100	—	—	Year	25°C **(Note 3)**

† Data in "Typ" column is at 5.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

Note 1: These specifications are for programming the on-chip program memory through the use of table write instructions.

2: The pin may be kept in this range at times other than programming but it is not recommended.

3: Retention time is valid provided no other specifications are violated.

27.4 AC (Timing) Characteristics

27.4.1 TIMING PARAMETER SYMBOLOGY

The timing parameter symbols have been created following one of the following formats:

1. TppS2ppS		3. TCC:ST	(I^2C specifications only)
2. TppS		4. Ts	(I^2C specifications only)
T			
F	Frequency	T	Time
Lowercase letters (pp) and their meanings:			
pp			
cc	CCP1	osc	OSC1
ck	CLKO	rd	$\overline{RD}$
cs	$\overline{CS}$	rw	$\overline{RD}$ or $\overline{WR}$
di	SDI	sc	SCK
do	SDO	ss	$\overline{SS}$
dt	Data in	t0	T0CKI
io	I/O port	t1	T1CKI
mc	$\overline{MCLR}$	wr	$\overline{WR}$
Uppercase letters and their meanings:			
S			
F	Fall	P	Period
H	High	R	Rise
I	Invalid (high-impedance)	V	Valid
L	Low	Z	High-impedance
I^2C only			
AA	output access	High	High
BUF	Bus free	Low	Low
TCC:ST (I^2C specifications only)			
CC			
HD	Hold	SU	Setup
ST			
DAT	DATA input hold	STO	Stop condition
STA	Start condition		

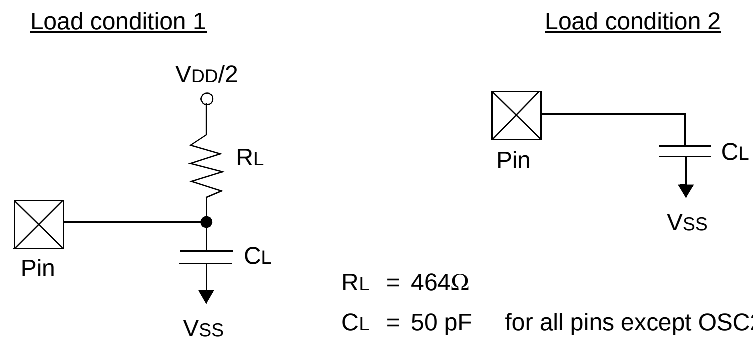
27.4.2 TIMING CONDITIONS

The temperature and voltages specified in Table 27-5 apply to all timing specifications unless otherwise noted. Figure 27-5 specifies the load conditions for the timing specifications.

TABLE 27-5: TEMPERATURE AND VOLTAGE SPECIFICATIONS - AC

AC CHARACTERISTICS	**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ T_A ≤ +85°C for industrial -40°C ≤ T_A ≤ +125°C for extended Operating voltage V_{DD} range as described in DC spec **Section 27.1** and **Section 27.3**. LC parts operate for industrial temperatures only.

FIGURE 27-5: LOAD CONDITIONS FOR DEVICE TIMING SPECIFICATIONS



R_L = 464Ω

C_L = 50 pF for all pins except OSC2/CLKO and including D and E outputs as ports

27.4.3 TIMING DIAGRAMS AND SPECIFICATIONS

FIGURE 27-6: EXTERNAL CLOCK TIMING (ALL MODES EXCEPT PLL)

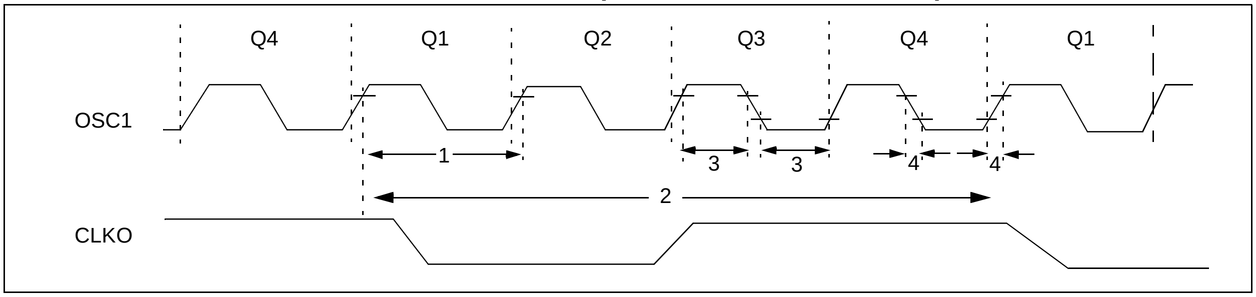


TABLE 27-6: EXTERNAL CLOCK TIMING REQUIREMENTS

Param. No.	Symbol	Characteristic	Min	Max	Units	Conditions
1A	FOSC	External CLKI Frequency(1)	DC	40	MHz	EC, ECIO, -40°C to +85°C
			DC	25	MHz	EC,ECIO, -40°C to +85°C, EMA
		Oscillator Frequency(1)	DC	25	MHz	EC, ECIO, +85°C to +125°C
			DC	16	MHz	EC, ECIO, +85°C to +125°C, EMA
			DC	4	MHz	RC oscillator
			0.1	4	MHz	XT oscillator
			4	25	MHz	HS oscillator, -40°C to +85°C
			4	25	MHz	HS oscillator, -40°C to +85°C, EMA
			4	25	MHz	HS oscillator, +85°C to +125°C
			4	16	MHz	HS oscillator, +85°C to +125°C, EMA
			4	10	MHz	HS + PLL oscillator, -40°C to +85°C
			4	6.25	MHz	HS + PLL oscillator, +85°C to +125°C
			DC	200	kHz	LP oscillator
1	TOSC	External CLKI Period(1)	25	—	ns	EC, ECIO, -40°C to +85°C
		Oscillator Period(1)	40	—	ns	EC,ECIO, -40°C to +85°C, EMA
			40	—	ns	EC, ECIO, +85°C to +125°C
			62.5	—	ns	EC, ECIO, +85°C to +125°C, EMA
			250	—	ns	RC oscillator
			250	10,000	ns	XT oscillator
			40	—	ns	HS oscillator, -40°C to +85°C
			40	—	ns	HS oscillator, -40°C to +85°C, EMA
			40	—	ns	HS oscillator, +85°C to +125°C
			62.5	—	ns	HS oscillator, +85°C to +125°C, EMA
			100	250	ns	HS + PLL oscillator, -40°C to +85°C
			160	250	ns	HS + PLL oscillator, +85°C to +125°C
			5	200	µs	LP oscillator
2	TCY	Instruction Cycle Time(1)	100	—	ns	TCY = 4/FOSC, -40°C to +85°C
			160	—	ns	TCY = 4/FOSC, +85°C to +125°C
3	TOSL, TOSH	External Clock in (OSC1) High or Low Time	30	—	ns	XT oscillator
			2.5	—	µs	LP oscillator
			10	—	ns	HS oscillator
4	TOSR, TOSF	External Clock in (OSC1) Rise or Fall Time	—	20	ns	XT oscillator
			—	50	ns	LP oscillator
			—	7.5	ns	HS oscillator

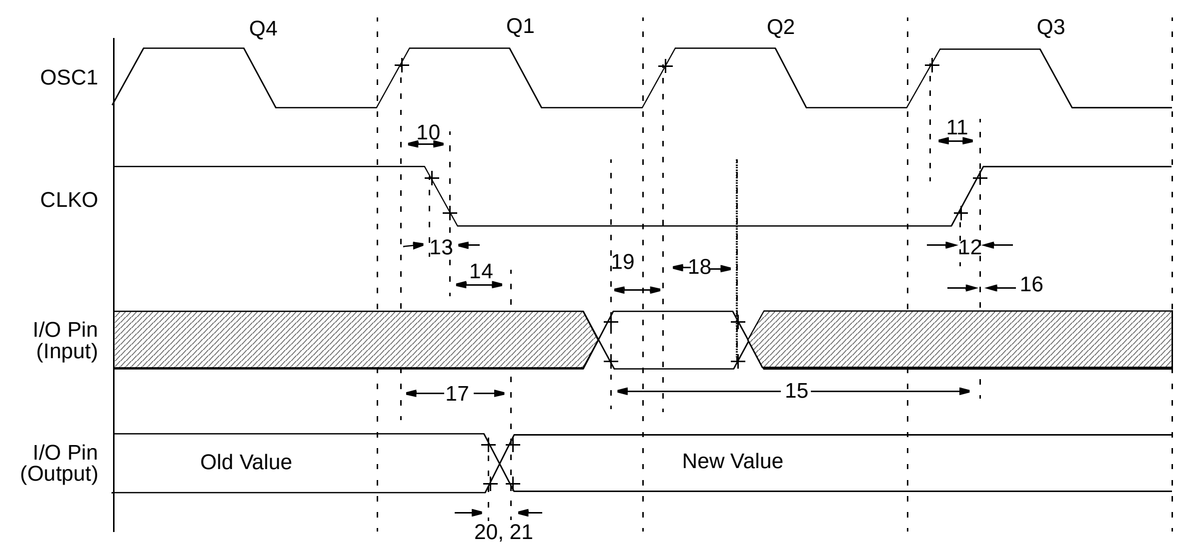
Note 1: Instruction cycle period (TCY) equals four times the input oscillator time base period for all configurations except PLL. All specified values are based on characterization data for that particular oscillator type under standard operating conditions with the device executing code. Exceeding these 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 the OSC1/CLKI pin. When an external clock input is used, the “max.” cycle time limit is “DC” (no clock) for all devices.

TABLE 27-7: PLL CLOCK TIMING SPECIFICATIONS (VDD = 4.2 TO 5.5V)

Param. No.	Sym	Characteristic	Min	Typ†	Max	Units	Conditions
—	FOSC	Oscillator Frequency Range	4	—	10	MHz	HS mode
—	FSYS	On-Chip VCO System Frequency	16	—	40	MHz	HS mode
—	t_{rc}	PLL Start-up Time (Lock Time)	—	—	2	ms	
—	ΔCLK	CLKO Stability (Jitter)	-2	—	+2	%	

† Data in “Typ” column is at 5V, 25°C, unless otherwise stated. These parameters are for design guidance only and are not tested.

FIGURE 27-7: CLKO AND I/O TIMING



Note: Refer to Figure 27-5 for load conditions.

TABLE 27-8: CLKO AND I/O TIMING REQUIREMENTS

Param. No.	Symbol	Characteristic		Min	Typ	Max	Units	Conditions
10	TOSH2CKL	OSC1 ↑ to CLKO ↓		—	75	200	ns	**(1)**
11	TOSH2CKH	OSC1 ↑ to CLKO ↑		—	75	200	ns	**(1)**
12	TCKR	CLKO Rise Time		—	35	100	ns	**(1)**
13	TCKF	CLKO Fall Time		—	35	100	ns	**(1)**
14	TCKL2IOV	CLKO ↓ to Port Out Valid		—	—	0.5 TCY + 20	ns	**(1)**
15	TIOV2CKH	Port In Valid before CLKO ↑		0.25 TCY + 25	—	—	ns	**(1)**
16	TCKH2IOI	Port In Hold after CLKO ↑		0	—	—	ns	**(1)**
17	TOSH2IOV	OSC1 ↑ (Q1 cycle) to Port Out Valid		—	50	150	ns	
18	TOSH2IOI	OSC1 ↑ (Q2 cycle) to Port Input Invalid (I/O in hold time)	PIC18FXX8X	100	—	—	ns	
18A			PIC18LFXX8X	200	—	—	ns	
19	TIOV2OSH	Port Input Valid to OSC1 ↑ (I/O in setup time)		0	—	—	ns	
20	TIOR	Port Output Rise Time	PIC18FXX8X	—	10	25	ns	
20A			PIC18LFXX8X	—	—	60	ns	
21	TIOF	Port Output Fall Time	PIC18FXX8X	—	10	25	ns	
21A			PIC18LFXX8X	—	—	60	ns	
22†	TINP	INT pin High or Low Time		TCY	—	—	ns	
23†	TRBP	RB7:RB4 Change INT High or Low Time		TCY	—	—	ns	
24†	TRCP	RC7:RC4 Change INT High or Low Time		20			ns	

† These parameters are asynchronous events not related to any internal clock edges.

Note 1: Measurements are taken in RC mode, where CLKO output is 4 x TOSC.

FIGURE 27-8: PROGRAM MEMORY READ TIMING DIAGRAM

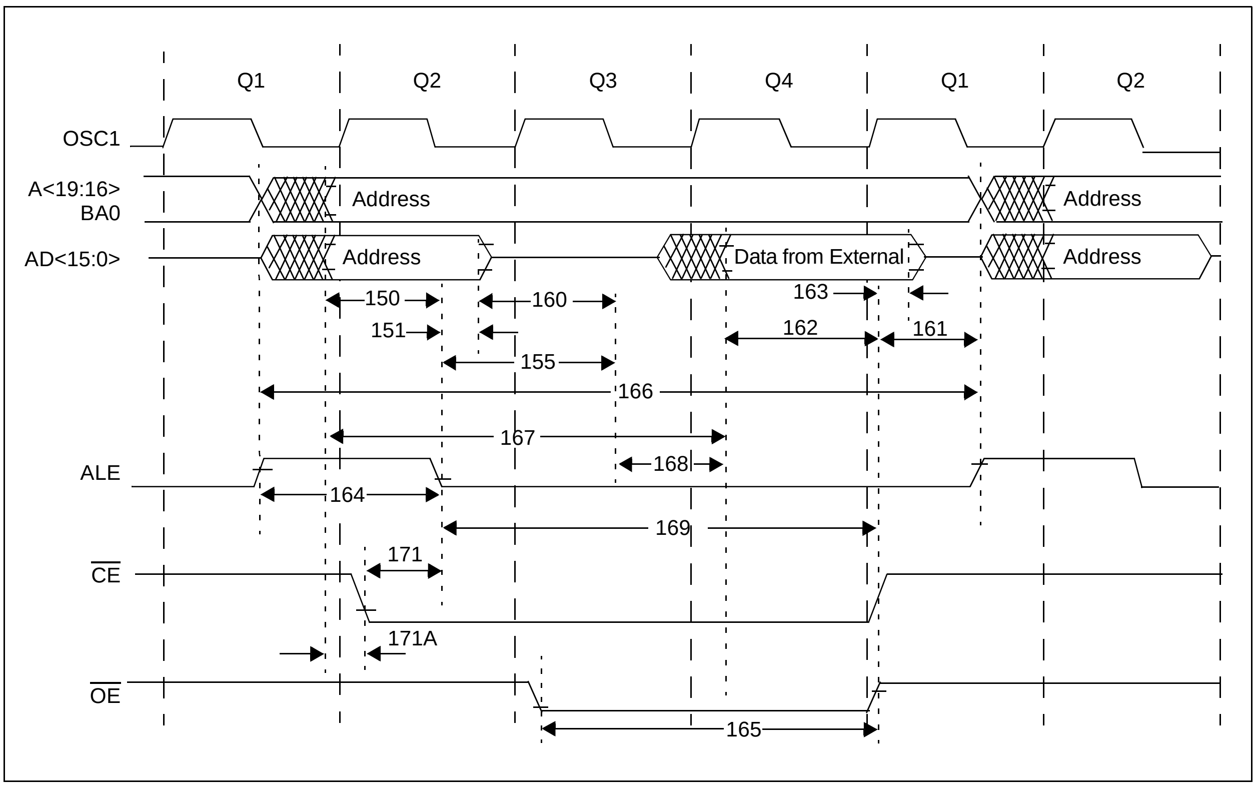


TABLE 27-9: PROGRAM MEMORY READ TIMING REQUIREMENTS (VDD = 4.2 TO 5.5V)

Param. No	Symbol	Characteristics	Min	Typ	Max	Units
150	TADV2ALL	Address Out Valid to ALE ↓ (address setup time)	0.25 TCY – 10	—	—	ns
151	TALL2ADL	ALE ↓ to Address Out Invalid (address hold time)	5	—	—	ns
155	TALL2OEL	ALE ↓ to $\overline{OE}$ ↓	10	0.125 TCY	—	ns
160	TADZ2OEL	AD High-Z to $\overline{OE}$ ↓ (bus release to $\overline{OE}$)	0	—	—	ns
161	TOEH2ADD	$\overline{OE}$ ↑ to AD Driven	0.125 TCY – 5	—	—	ns
162	TADV2OEH	LS Data Valid before $\overline{OE}$ ↑ (data setup time)	20	—	—	ns
163	TOEH2ADL	$\overline{OE}$ ↑ to Data In Invalid (data hold time)	0	—	—	ns
164	TALH2ALL	ALE Pulse Width	—	0.25 TCY	—	ns
165	TOEL2OEH	$\overline{OE}$ Pulse Width	0.5 TCY – 5	0.5 TCY	—	ns
166	TALH2ALH	ALE ↑ to ALE ↑ (cycle time)	—	1 TCY	—	ns
167	TACC	Address Valid to Data Valid	0.75 TCY – 25	—	—	ns
168	TOE	$\overline{OE}$ ↓ to Data Valid		—	0.5 TCY – 25	ns
169	TALL2OEH	ALE ↓ to $\overline{OE}$ ↑	0.625 TCY – 10	—	0.625 TCY + 10	ns
171	TALH2CSL	Chip Select Active to ALE ↓	—	—	10	ns
171A	TUBL2OEH	AD Valid to Chip Select Active	0.25 TCY – 20	—	—	ns

FIGURE 27-9: PROGRAM MEMORY WRITE TIMING DIAGRAM

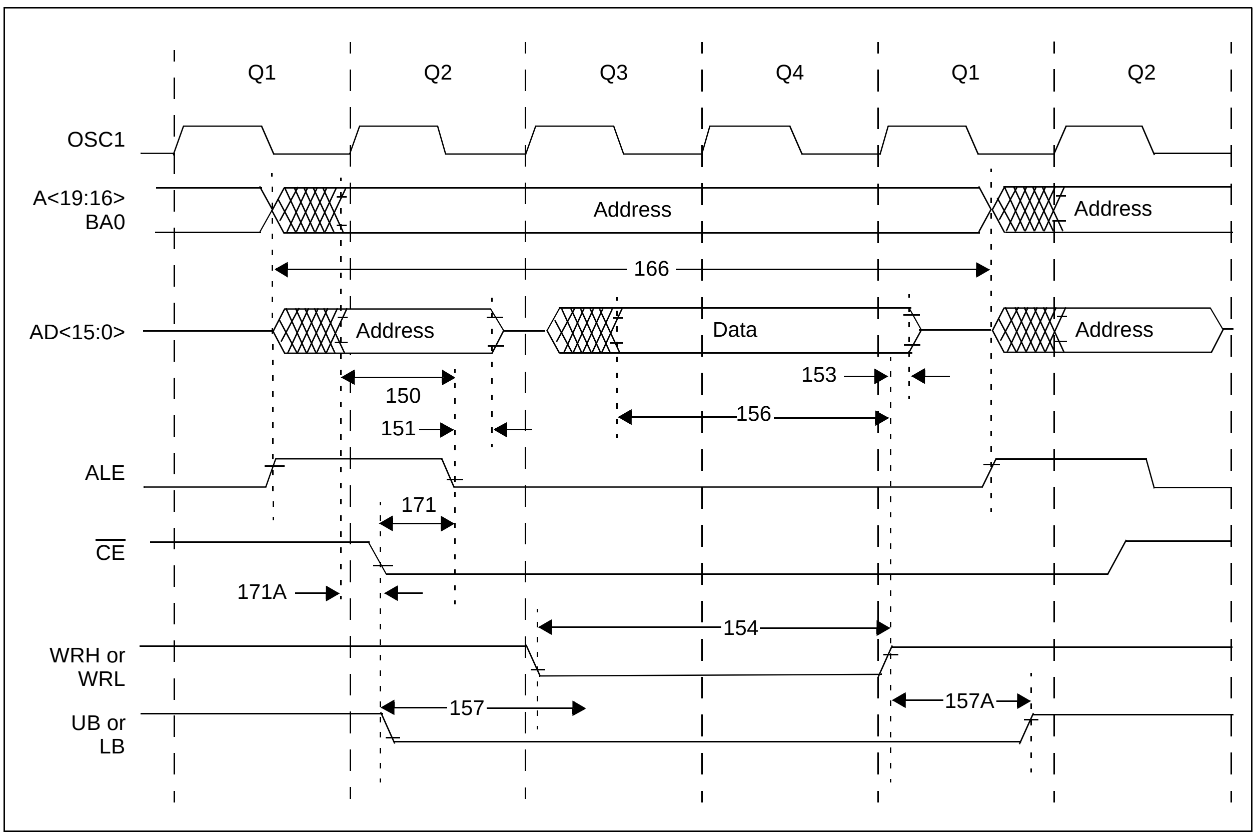
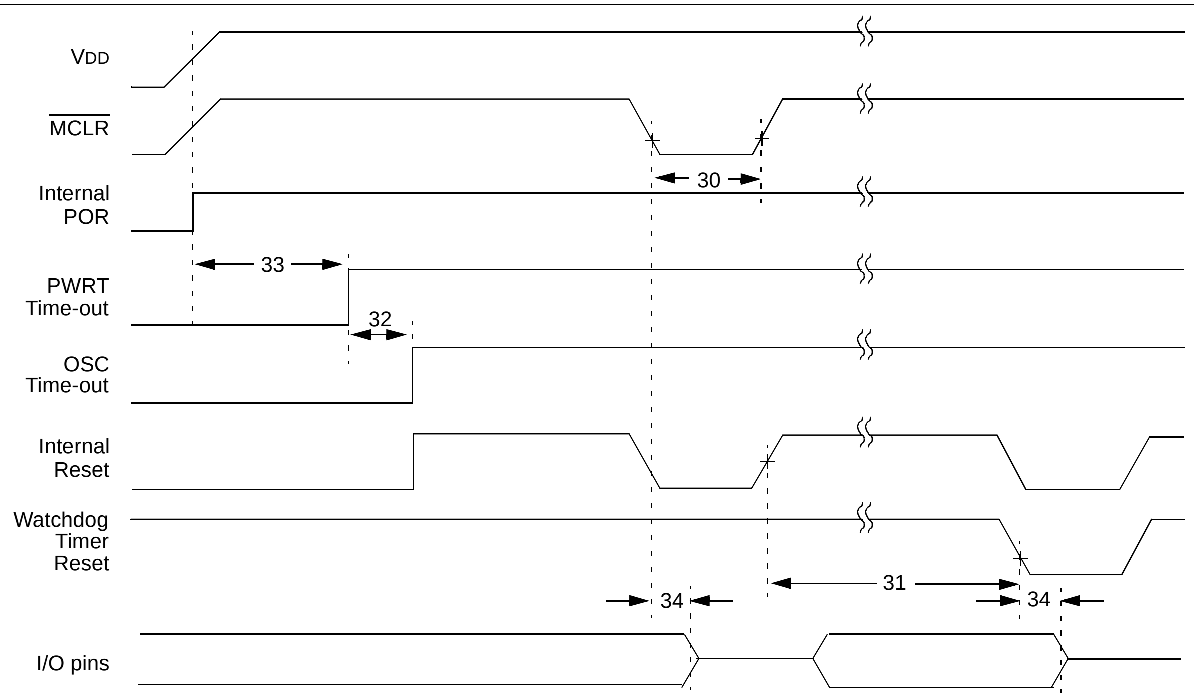


TABLE 27-10: PROGRAM MEMORY WRITE TIMING REQUIREMENTS (VDD = 4.2 TO 5.5V)

Param. No.	Symbol	Characteristics	Min	Typ	Max	Units
150	TADV2ALL	Address Out Valid to ALE ↓ (address setup time)	0.25 TCY – 10	—	—	ns
151	TALL2ADL	ALE ↓ to Address Out Invalid (address hold time)	5	—	—	ns
153	TWRH2ADL	WRn ↑ to Data Out Invalid (data hold time)	5	—	—	ns
154	TWRL	WRn Pulse Width	0.5 TCY – 5	0.5 TCY	—	ns
156	TADV2WRH	Data Valid before WRn ↑ (data setup time)	0.5 TCY – 10	—	—	ns
157	TBSV2WRL	Byte Select Valid before WRn ↓ (byte select setup time)	0.25 TCY	—	—	ns
157A	TWRH2BSI	WRn ↑ to Byte Select Invalid (byte select hold time)	0.125 TCY – 5	—	—	ns
166	TALH2ALH	ALE ↑ to ALE ↑ (cycle time)	—	0.25 TCY	—	ns
171	TALH2CSL	Chip Enable Active to ALE ↓	—	—	10	ns
171A	TUBL2OEH	AD Valid to Chip Enable Active	0.25 TCY – 20	—	—	ns

FIGURE 27-10: RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER AND POWER-UP TIMER TIMING



Note: Refer to Figure 27-5 for load conditions.

FIGURE 27-11: BROWN-OUT RESET TIMING

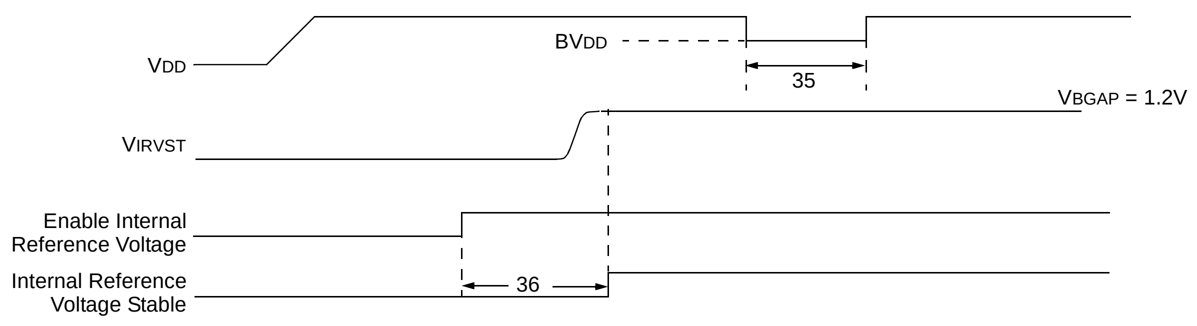
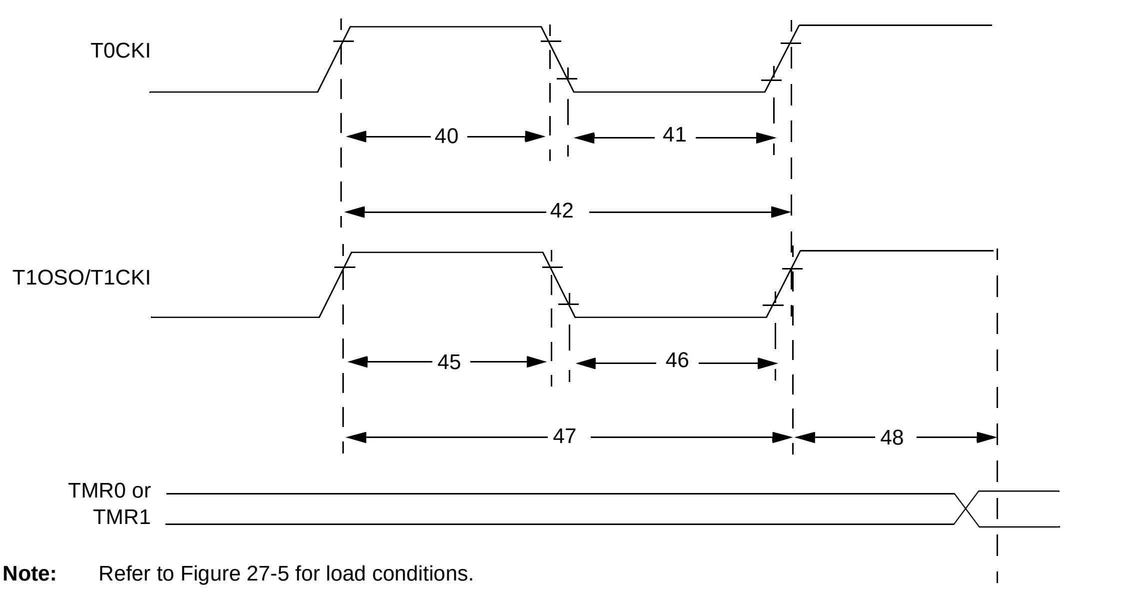


TABLE 27-11: RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER, POWER-UP TIMER AND BROWN-OUT RESET REQUIREMENTS

Param. No.	Symbol	Characteristic	Min	Typ	Max	Units	Conditions
30	TMCL	$\overline{\text{MCLR}}$ Pulse Width (low)	2	—	—	μs	
31	TWDT	Watchdog Timer Time-out Period (No Postscaler)	7	18	33	ms	
32	TOST	Oscillation Start-up Timer Period	1024 TOSC	—	1024 TOSC	—	TOSC = OSC1 period
33	TPWRT	Power up Timer Period	28	72	132	ms	
34	TIOZ	I/O High-Impedance from $\overline{\text{MCLR}}$ Low or Watchdog Timer Reset	—	2	—	μs	
35	TBOR	Brown-out Reset Pulse Width	200	—	—	μs	VDD ≤ BVDD (see)
36	TIVRST	Time for Internal Reference Voltage to become stable	—	20	50	μs	
37	TLVD	Low-Voltage Detect Pulse Width	200	—	—	μs	VDD ≤ VLVD

FIGURE 27-12: TIMER0 AND TIMER1 EXTERNAL CLOCK TIMINGS

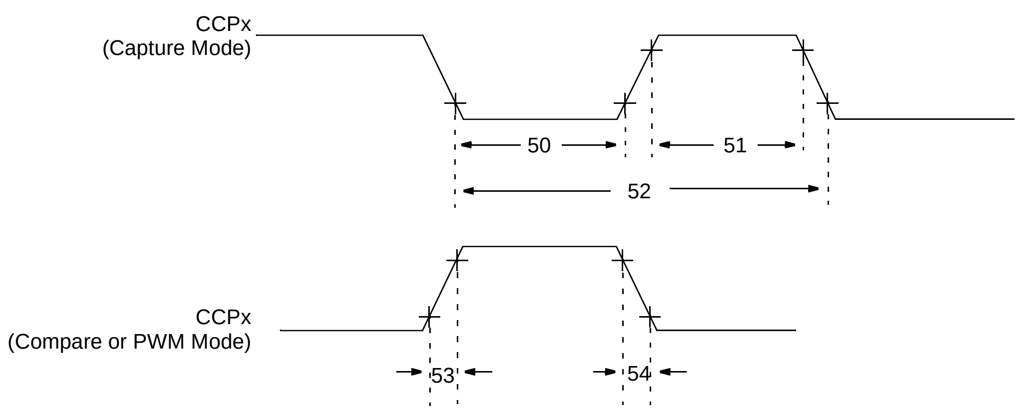


Note: Refer to Figure 27-5 for load conditions.

TABLE 27-12: TIMER0 AND TIMER1 EXTERNAL CLOCK REQUIREMENTS

Param. No.	Symbol	Characteristic			Min	Max	Units	Conditions
40	TT0H	T0CKI High Pulse Width		No prescaler	0.5 TCY + 20	—	ns	
				With prescaler	10	—	ns	
41	TT0L	T0CKI Low Pulse Width		No prescaler	0.5 TCY + 20	—	ns	
				With prescaler	10	—	ns	
42	TT0P	T0CKI Period		No prescaler	TCY + 10	—	ns	
				With prescaler	Greater of: 20 ns or $\frac{T_{CY} + 40}{N}$	—	ns	N = prescale value (1, 2, 4,..., 256)
45	TT1H	T1CKI High Time	Synchronous, no prescaler		0.5 TCY + 20	—	ns	
			Synchronous, with prescaler	PIC18FXX8X	10	—	ns	
				PIC18LFXX8X	25	—	ns	
			Asynchronous	PIC18FXX8X	30	—	ns	
				PIC18LFXX8X	50	—	ns	
46	TT1L	T1CKI Low Time	Synchronous, no prescaler		0.5 TCY + 5	—	ns	
			Synchronous, with prescaler	PIC18FXX8X	10	—	ns	
				PIC18LFXX8X	25	—	ns	
			Asynchronous	PIC18FXX8X	30	—	ns	
				PIC18LFXX8X	TBD	TBD	ns	
47	TT1P	T1CKI Input Period	Synchronous		Greater of: 20 ns or $\frac{T_{CY} + 40}{N}$	—	ns	N = prescale value (1, 2, 4, 8)
			Asynchronous		60	—	ns	
	FT1	T1CKI Oscillator Input Frequency Range			DC	50	kHz	
48	TCKE2TMRI	Delay from External T1CKI Clock Edge to Timer Increment			2 TOSC	7 TOSC	—	

FIGURE 27-13: CAPTURE/COMPARE/PWM TIMINGS (ALL CCP MODULES)

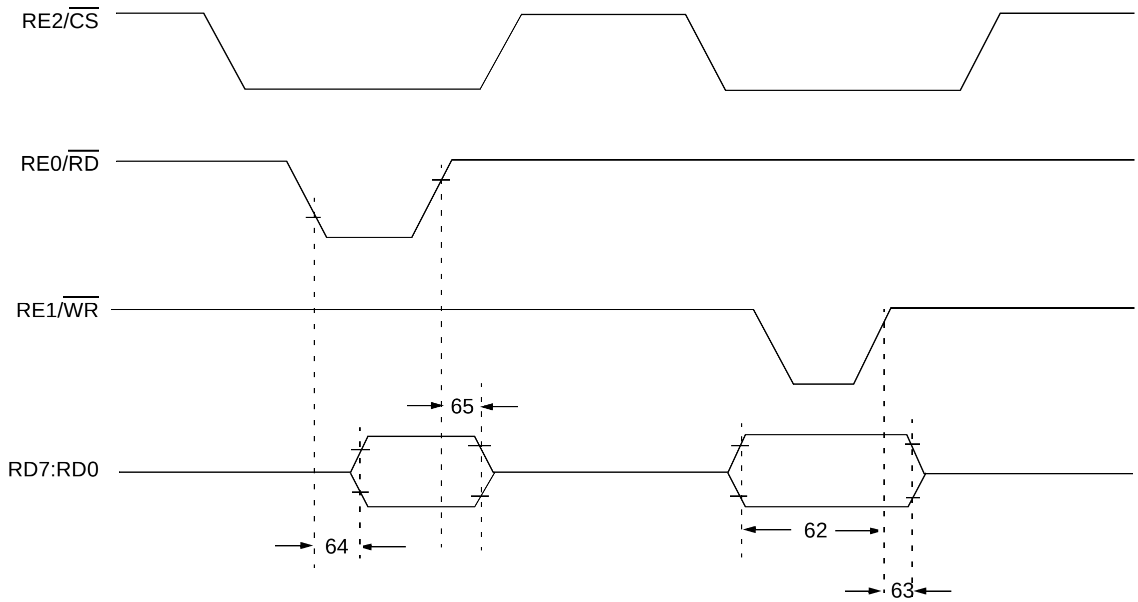


Note: Refer to Figure 27-5 for load conditions.

TABLE 27-13: CAPTURE/COMPARE/PWM REQUIREMENTS (ALL CCP MODULES)

Param. No.	Symbol	Characteristic			Min	Max	Units	Conditions
50	TCCL	CCPx Input Low Time	No prescaler		0.5 TCY + 20	—	ns	
			With prescaler	PIC18FXX8X	10	—	ns	
				PIC18LFXX8X	20	—	ns	
51	TCCH	CCPx Input High Time	No prescaler		0.5 TCY + 20	—	ns	
			With prescaler	PIC18FXX8X	10	—	ns	
				PIC18LFXX8X	20	—	ns	
52	TCCP	CCPx Input Period			$\frac{3\ TCY + 40}{N}$	—	ns	N = prescale value (1,4 or 16)
53	TCCR	CCPx Output Rise Time		PIC18FXX8X	—	25	ns	
				PIC18LFXX8X	—	45	ns	
54	TCCF	CCPx Output Fall Time		PIC18FXX8X	—	25	ns	
				PIC18LFXX8X	—	45	ns	

FIGURE 27-14: PARALLEL SLAVE PORT TIMING (PIC18FXX8X)

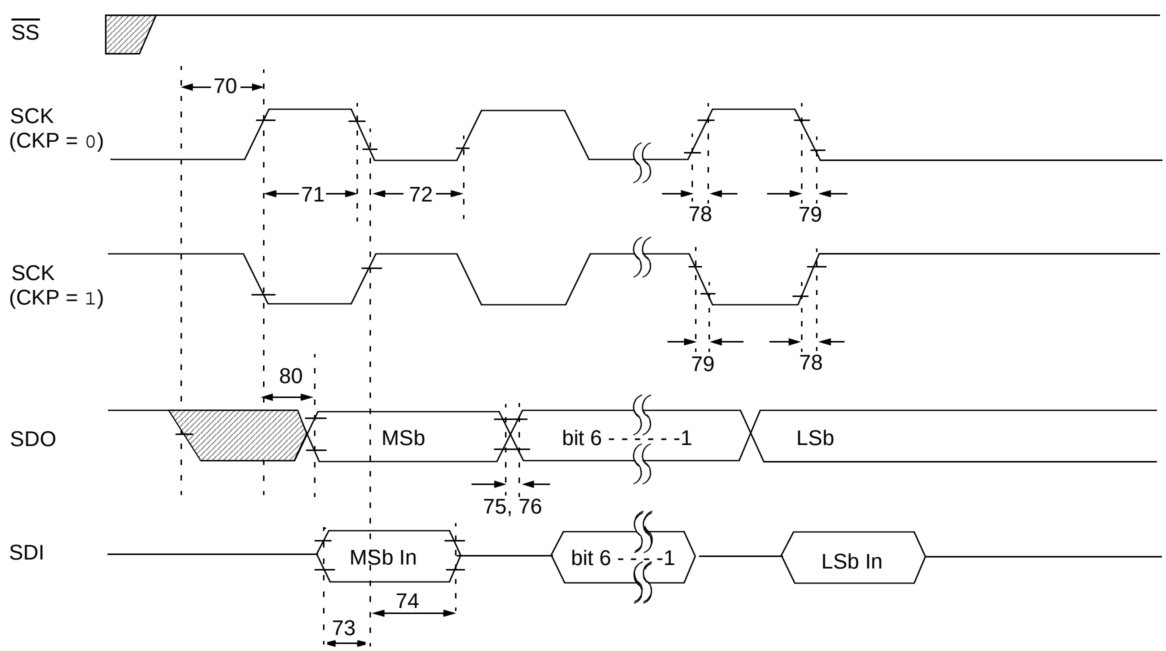


Note: Refer to Figure 27-5 for load conditions.

TABLE 27-14: PARALLEL SLAVE PORT REQUIREMENTS (PIC18FXX8X)

Param. No.	Symbol	Characteristic		Min	Max	Units	Conditions
62	TDTV2WRH	Data In Valid before $\overline{WR}$ ↑ or $\overline{CS}$ ↑ (setup time)		20 25	— —	ns ns	 Extended Temp. range
63	TWRH2DTI	$\overline{WR}$ ↑ or $\overline{CS}$ ↑ to Data–In Invalid (hold time)	PIC18FXX8X	20	—	ns	
			PIC18LFXX8X	35	—	ns	
64	TRDL2DTV	$\overline{RD}$ ↓ and $\overline{CS}$ ↓ to Data–Out Valid		— —	80 90	ns ns	 Extended Temp. range
65	TRDH2DTI	$\overline{RD}$ ↑ or $\overline{CS}$ ↓ to Data–Out Invalid		10	30	ns	
66	TIBFINH	Inhibit of the IBF flag bit being cleared from $\overline{WR}$ ↑ or $\overline{CS}$ ↑		—	3 TCY		

FIGURE 27-15: EXAMPLE SPI MASTER MODE TIMING (CKE = 0)



Note: Refer to Figure 27-5 for load conditions.

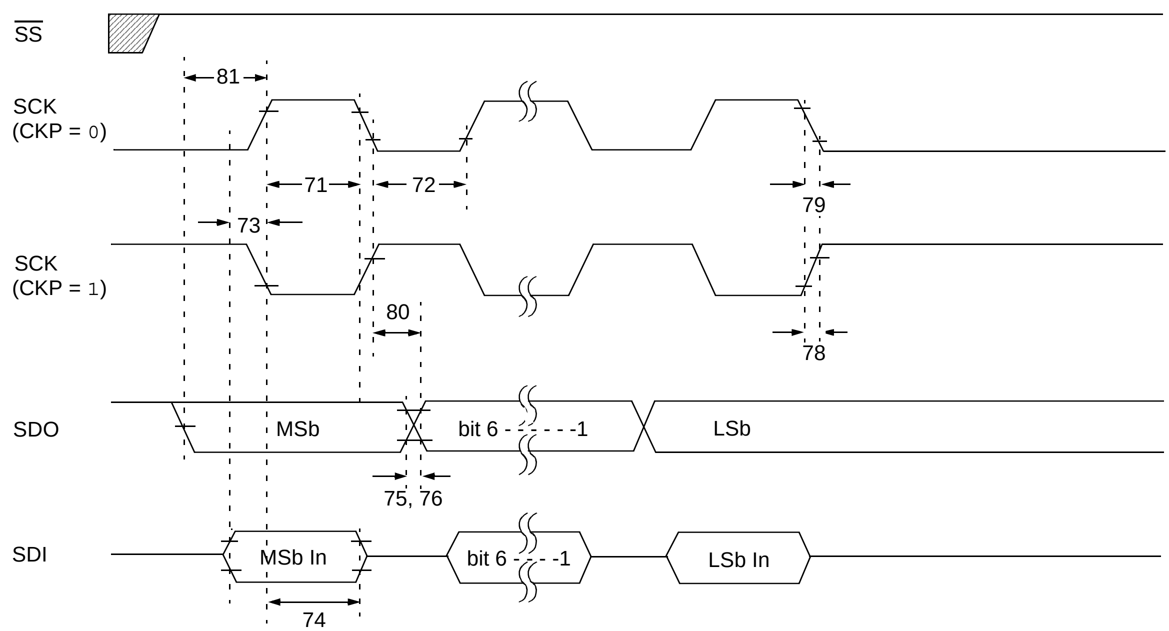
TABLE 27-15: EXAMPLE SPI MODE REQUIREMENTS (MASTER MODE, CKE = 0)

Param. No.	Symbol	Characteristic		Min	Max	Units	Conditions
70	TSSL2SCH, TSSL2SCL	$\overline{SS}$ ↓ to SCK ↓ or SCK ↑ Input		TCY	—	ns	
71	TSCH	SCK Input High Time (Slave mode)	Continuous	1.25 TCY + 30	—	ns	
71A			Single Byte	40	—	ns	**(Note 1)**
72	TSCL	SCK Input Low Time (Slave mode)	Continuous	1.25 TCY + 30	—	ns	
72A			Single Byte	40	—	ns	**(Note 1)**
73	TDIV2SCH, TDIV2SCL	Setup Time of SDI Data Input to SCK Edge		100	—	ns	
73A	TB2B	Last Clock Edge of Byte 1 to the 1st Clock Edge of Byte 2		1.5 TCY + 40	—	ns	**(Note 2)**
74	TSCH2DIL, TSCL2DIL	Hold Time of SDI Data Input to SCK Edge		100	—	ns	
75	TDOR	SDO Data Output Rise Time	PIC18FXX8X	—	25	ns	
			PIC18LFXX8X	—	45	ns	
76	TDOF	SDO Data Output Fall Time		—	25	ns	
78	TSCR	SCK Output Rise Time (Master mode)	PIC18FXX8X	—	25	ns	
			PIC18LFXX8X	—	45	ns	
79	TSCF	SCK Output Fall Time (Master mode)		—	25	ns	
80	TSCH2DOV, TSCL2DOV	SDO Data Output Valid after SCK Edge	PIC18FXX8X	—	50	ns	
			PIC18LFXX8X	—	100	ns	

Note 1: Requires the use of Parameter #73A.

2: Only if Parameter #71A and #72A are used.

FIGURE 27-16: EXAMPLE SPI MASTER MODE TIMING (CKE = 1)



Note: Refer to Figure 27-5 for load conditions.

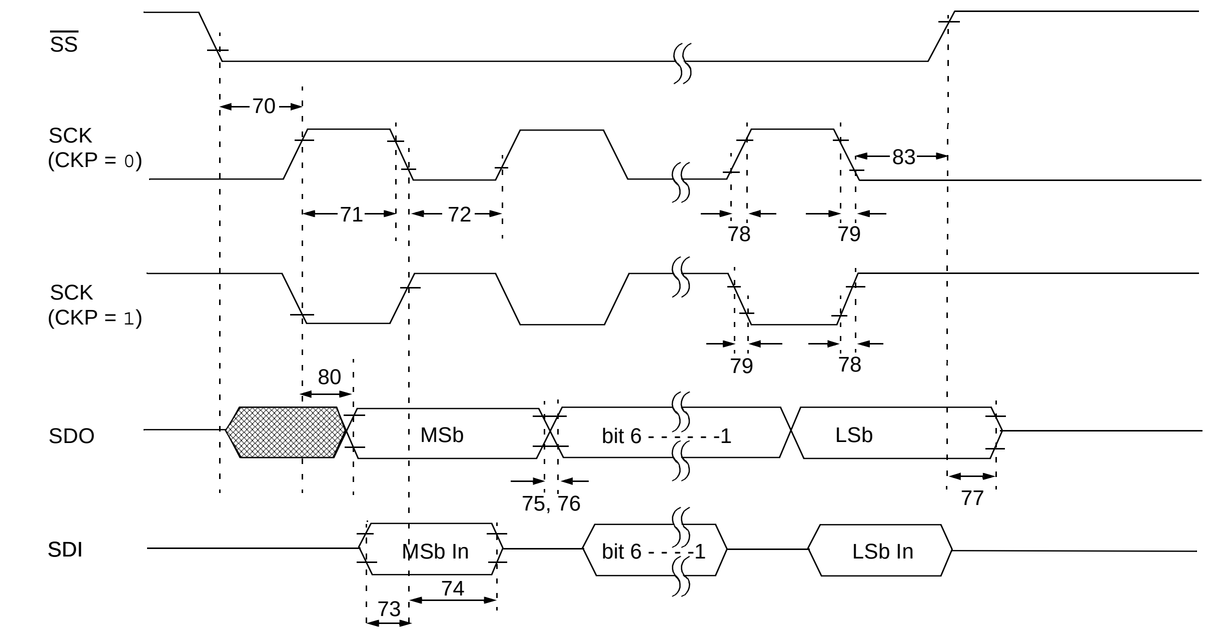
TABLE 27-16: EXAMPLE SPI MODE REQUIREMENTS (MASTER MODE, CKE = 1)

Param. No.	Symbol	Characteristic		Min	Max	Units	Conditions
71	TSCH	SCK Input High Time (Slave mode)	Continuous	1.25 TCY + 30	—	ns	
71A			Single Byte	40	—	ns	**(Note 1)**
72	TSCL	SCK Input Low Time (Slave mode)	Continuous	1.25 TCY + 30	—	ns	
72A			Single Byte	40	—	ns	**(Note 1)**
73	TDIV2SCH, TDIV2SCL	Setup Time of SDI Data Input to SCK Edge		100	—	ns	
73A	TB2B	Last Clock Edge of Byte 1 to the 1st Clock Edge of Byte 2		1.5 TCY + 40	—	ns	**(Note 2)**
74	TSCH2DIL, TSCL2DIL	Hold Time of SDI Data Input to SCK Edge		100	—	ns	
75	TDOR	SDO Data Output Rise Time	PIC18FXX8X	—	25	ns	
			PIC18LFXX8X		45	ns	
76	TDOF	SDO Data Output Fall Time		—	25	ns	
78	TSCR	SCK Output Rise Time (Master mode)	PIC18FXX8X	—	25	ns	
			PIC18LFXX8X		45	ns	
79	TSCF	SCK Output Fall Time (Master mode)		—	25	ns	
80	TSCH2DOV, TSCL2DOV	SDO Data Output Valid after SCK Edge	PIC18FXX8X	—	50	ns	
			PIC18LFXX8X		100	ns	
81	TDOV2SCH, TDOV2SCL	SDO Data Output Setup to SCK Edge		TCY	—	ns	

Note 1: Requires the use of Parameter #73A.

2: Only if Parameter #71A and #72A are used.

FIGURE 27-17: EXAMPLE SPI SLAVE MODE TIMING (CKE = 0)



Note: Refer to Figure 27-5 for load conditions.

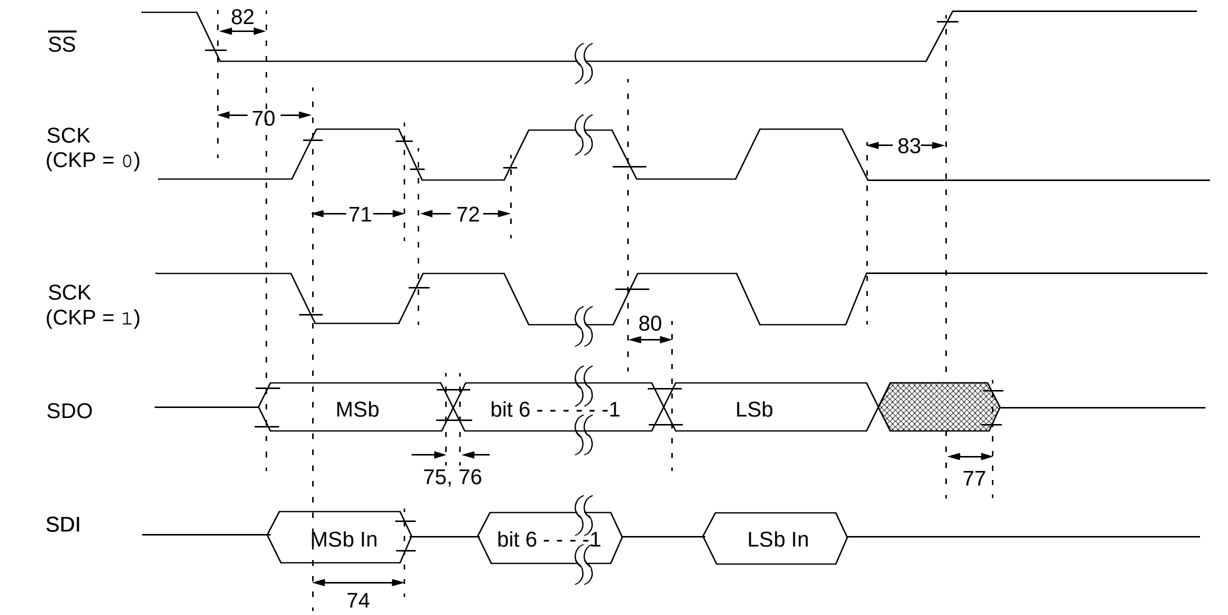
TABLE 27-17: EXAMPLE SPI MODE REQUIREMENTS (SLAVE MODE TIMING, CKE = 0)

Param. No.	Symbol	Characteristic		Min	Max	Units	Conditions
70	TssL2scH, TssL2scL	$\overline{SS}$ ↓ to SCK ↓ or SCK ↑ Input		Tcy	—	ns	
71	TscH	SCK Input High Time (Slave mode)	Continuous	1.25 Tcy + 30	—	ns	
71A			Single Byte	40	—	ns	**(Note 1)**
72	TscL	SCK Input Low Time (Slave mode)	Continuous	1.25 Tcy + 30	—	ns	
72A			Single Byte	40	—	ns	**(Note 1)**
73	TdiV2scH, TdiV2scL	Setup Time of SDI Data Input to SCK Edge		100	—	ns	
73A	Tb2b	Last Clock Edge of Byte 1 to the First Clock Edge of Byte 2		1.5 Tcy + 40	—	ns	**(Note 2)**
74	TscH2diL, TscL2diL	Hold Time of SDI Data Input to SCK Edge		100	—	ns	
75	TdoR	SDO Data Output Rise Time	PIC18FXX8X	—	25	ns	
			PIC18LFXX8X		45	ns	
76	TdoF	SDO Data Output Fall Time		—	25	ns	
77	TssH2doZ	$\overline{SS}$ ↑ to SDO Output High-Impedance		10	50	ns	
78	TscR	SCK oUtput Rise Time (Master mode)	PIC18FXX8X	—	25	ns	
			PIC18LFXX8X		45	ns	
79	TscF	SCK Output Fall Time (Master mode)		—	25	ns	
80	TscH2doV, TscL2doV	SDO Data Output Valid after SCK Edge	PIC18FXX8X	—	50	ns	
			PIC18LFXX8X		100	ns	
83	TscH2ssH, TscL2ssH	$\overline{SS}$ ↑ after SCK Edge		1.5 Tcy + 40	—	ns	

Note 1: Requires the use of Parameter #73A.

2: Only if Parameter #71A and #72A are used.

FIGURE 27-18: EXAMPLE SPI SLAVE MODE TIMING (CKE = 1)



Note: Refer to Figure 27-5 for load conditions.

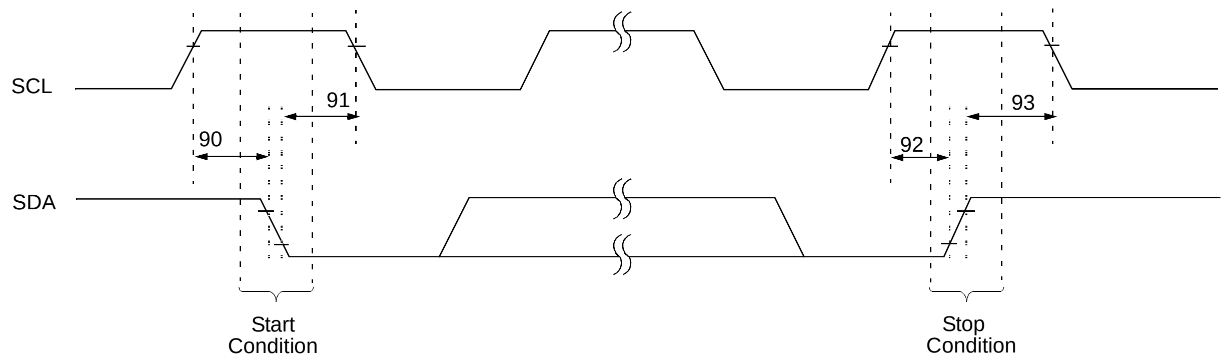
TABLE 27-18: EXAMPLE SPI SLAVE MODE REQUIREMENTS (CKE = 1)

Param. No.	Symbol	Characteristic		Min	Max	Units	Conditions
70	TssL2scH, TssL2scL	$\overline{SS}$ ↓ to SCK ↓ or SCK ↑ Input		TCY	—	ns	
71	TscH	SCK Input High Time (Slave mode)	Continuous	1.25 TCY + 30	—	ns	
71A			Single Byte	40	—	ns	**(Note 1)**
72	TscL	SCK Input Low Time (Slave mode)	Continuous	1.25 TCY + 30	—	ns	
72A			Single Byte	40	—	ns	**(Note 1)**
73A	TB2B	Last Clock Edge of Byte 1 to the First Clock Edge of Byte 2		1.5 TCY + 40	—	ns	**(Note 2)**
74	TscH2diL, TscL2diL	Hold Time of SDI Data Input to SCK Edge		100	—	ns	
75	TdoR	SDO Data Output Rise Time	PIC18FXX8X	—	25	ns	
			PIC18LFXX8X		45	ns	
76	TdoF	SDO Data Output Fall Time		—	25	ns	
77	TssH2doZ	$\overline{SS}$ ↑ to SDO Output High-Impedance		10	50	ns	
78	TscR	SCK Output Rise Time (Master mode)	PIC18FXX8X	—	25	ns	
			PIC18LFXX8X	—	45	ns	
79	TscF	SCK Output Fall Time (Master mode)		—	25	ns	
80	TscH2doV, TscL2doV	SDO Data Output Valid after SCK Edge	PIC18FXX8X	—	50	ns	
			PIC18LFXX8X	—	100	ns	
82	TssL2doV	SDO Data Output Valid after $\overline{SS}$ ↓ Edge	PIC18FXX8X	—	50	ns	
			PIC18LFXX8X	—	100	ns	
83	TscH2ssH, TscL2ssH	$\overline{SS}$ ↑ after SCK Edge		1.5 TCY + 40	—	ns	

Note 1: Requires the use of Parameter #73A.

2: Only if Parameter #71A and #72A are used.

FIGURE 27-19: I²C BUS START/STOP BITS TIMING

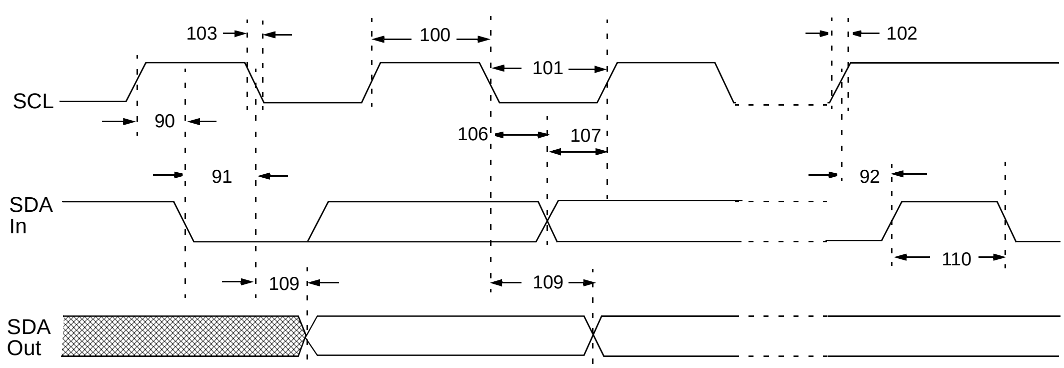


Note: Refer to Figure 27-5 for load conditions.

TABLE 27-19: I²C BUS START/STOP BITS REQUIREMENTS (SLAVE MODE)

Param. No.	Symbol	Characteristic		Min	Max	Units	Conditions
90	TSU:STA	Start Condition	100 kHz mode	4700	—	ns	Only relevant for Repeated Start condition
		Setup Time	400 kHz mode	600	—		
91	THD:STA	Start Condition	100 kHz mode	4000	—	ns	After this period, the first clock pulse is generated
		Hold Time	400 kHz mode	600	—		
92	TSU:STO	Stop Condition	100 kHz mode	4700	—	ns	
		Setup Time	400 kHz mode	600	—		
93	THD:STO	Stop Condition	100 kHz mode	4000	—	ns	
		Hold Time	400 kHz mode	600	—		

FIGURE 27-20: I²C BUS DATA TIMING



Note: Refer to Figure 27-5 for load conditions.

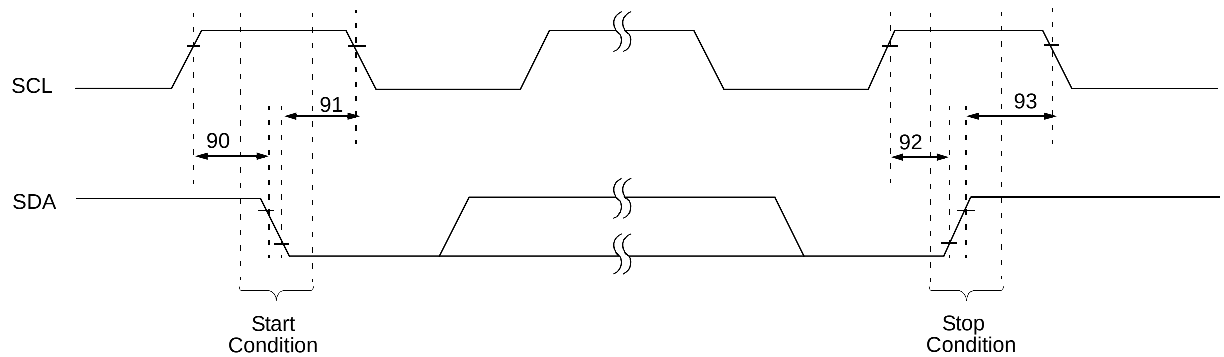
TABLE 27-20: I²C BUS DATA REQUIREMENTS (SLAVE MODE)

Param. No.	Symbol	Characteristic		Min	Max	Units	Conditions
100	THIGH	Clock High Time	100 kHz mode	4.0	—	μs	PIC18FXX8X must operate at a minimum of 1.5 MHz
			400 kHz mode	0.6	—	μs	PIC18FXX8X must operate at a minimum of 10 MHz
			SSP Module	1.5 TCY	—		
101	TLOW	Clock Low Time	100 kHz mode	4.7	—	μs	PIC18FXX8X must operate at a minimum of 1.5 MHz
			400 kHz mode	1.3	—	μs	PIC18FXX8X must operate at a minimum of 10 MHz
			SSP Module	1.5 TCY	—		
102	TR	SDA and SCL Rise Time	100 kHz mode	—	1000	ns	
			400 kHz mode	20 + 0.1 CB	300	ns	CB is specified to be from 10 to 400 pF
103	TF	SDA and SCL Fall Time	100 kHz mode	—	300	ns	
			400 kHz mode	20 + 0.1 CB	300	ns	CB is specified to be from 10 to 400 pF
90	TSU:STA	Start Condition Setup Time	100 kHz mode	4.7	—	μs	Only relevant for Repeated Start condition
			400 kHz mode	0.6	—	μs	
91	THD:STA	Start Condition Hold Time	100 kHz mode	4.0	—	μs	After this period, the first clock pulse is generated
			400 kHz mode	0.6	—	μs	
106	THD:DAT	Data Input Hold Time	100 kHz mode	0	—	ns	
			400 kHz mode	0	0.9	μs	
107	TSU:DAT	Data Input Setup Time	100 kHz mode	250	—	ns	**(Note 2)**
			400 kHz mode	100	—	ns	
92	TSU:STO	Stop Condition Setup Time	100 kHz mode	4.7	—	μs	
			400 kHz mode	0.6	—	μs	
109	TAA	Output Valid from Clock	100 kHz mode	—	3500	ns	**(Note 1)**
			400 kHz mode	—	—	ns	
110	TBUF	Bus Free Time	100 kHz mode	4.7	—	μs	Time the bus must be free before a new transmission can Start
			400 kHz mode	1.3	—	μs	
D102	CB	Bus Capacitive Loading		—	400	pF	

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 I²C bus device can be used in a Standard mode I²C bus system but the requirement, TSU:DAT ≥ 250 ns, must then be met. This will automatically be the case if the device does not stretch the low period of the SCL signal. If such a device does stretch the low period of the SCL signal, it must output the next data bit to the SDA line.
TR max. + TSU:DAT = 1000 + 250 = 1250 ns (according to the Standard mode I²C bus specification) before the SCL line is released.

FIGURE 27-21: MASTER SSP I^{2}C BUS START/STOP BITS TIMING WAVEFORMS



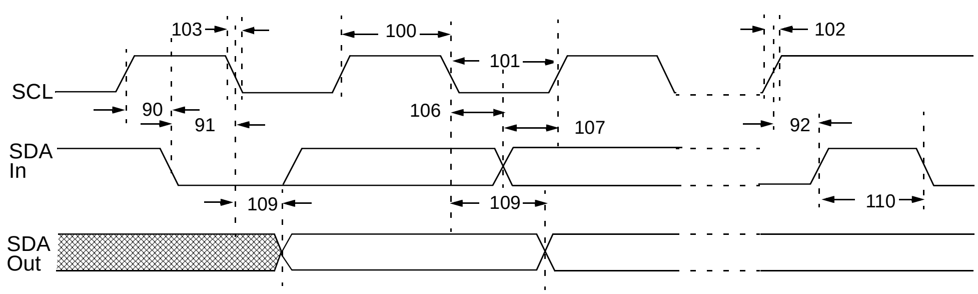
Note: Refer to Figure 27-5 for load conditions.

TABLE 27-21: MASTER SSP I^{2}C BUS START/STOP BITS REQUIREMENTS

Param. No.	Symbol	Characteristic		Min	Max	Units	Conditions
90	TSU:STA	Start Condition Setup Time	100 kHz mode	2(TOSC)(BRG + 1)	—	ns	Only relevant for Repeated Start condition
			400 kHz mode	2(TOSC)(BRG + 1)	—		
			1 MHz mode(1)	2(TOSC)(BRG + 1)	—		
91	THD:STA	Start Condition Hold Time	100 kHz mode	2(TOSC)(BRG + 1)	—	ns	After this period, the first clock pulse is generated
			400 kHz mode	2(TOSC)(BRG + 1)	—		
			1 MHz mode(1)	2(TOSC)(BRG + 1)	—		
92	TSU:STO	Stop Condition Setup Time	100 kHz mode	2(TOSC)(BRG + 1)	—	ns	
			400 kHz mode	2(TOSC)(BRG + 1)	—		
			1 MHz mode(1)	2(TOSC)(BRG + 1)	—		
93	THD:STO	Stop Condition Hold Time	100 kHz mode	2(TOSC)(BRG + 1)	—	ns	
			400 kHz mode	2(TOSC)(BRG + 1)	—		
			1 MHz mode(1)	2(TOSC)(BRG + 1)	—		

Note 1: Maximum pin capacitance = 10 pF for all I^{2}C pins.

FIGURE 27-22: MASTER SSP I^{2}C BUS DATA TIMING



Note: Refer to Figure 27-5 for load conditions.

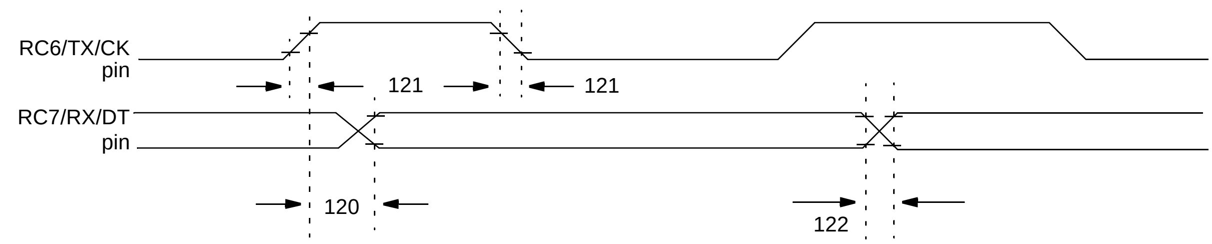
TABLE 27-22: MASTER SSP I²C BUS DATA REQUIREMENTS

Param. No.	Symbol	Characteristic		Min	Max	Units	Conditions
100	THIGH	Clock High Time	100 kHz mode	2(TOSC)(BRG + 1)	—	ms	
			400 kHz mode	2(TOSC)(BRG + 1)	—	ms	
			1 MHz mode(1)	2(TOSC)(BRG + 1)	—	ms	
101	TLOW	Clock Low Time	100 kHz mode	2(TOSC)(BRG + 1)	—	ms	
			400 kHz mode	2(TOSC)(BRG + 1)	—	ms	
			1 MHz mode(1)	2(TOSC)(BRG + 1)	—	ms	
102	TR	SDA and SCL Rise Time	100 kHz mode	—	1000	ns	CB is specified to be from 10 to 400 pF
			400 kHz mode	20 + 0.1 CB	300	ns	
			1 MHz mode(1)	—	300	ns	
103	TF	SDA and SCL Fall Time	100 kHz mode	—	300	ns	CB is specified to be from 10 to 400 pF
			400 kHz mode	20 + 0.1 CB	300	ns	
			1 MHz mode(1)	—	100	ns	
90	TSU:STA	Start Condition Setup Time	100 kHz mode	2(TOSC)(BRG + 1)	—	ms	Only relevant for Repeated Start condition
			400 kHz mode	2(TOSC)(BRG + 1)	—	ms	
			1 MHz mode(1)	2(TOSC)(BRG + 1)	—	ms	
91	THD:STA	Start Condition Hold Time	100 kHz mode	2(TOSC)(BRG + 1)	—	ms	After this period, the first clock pulse is generated
			400 kHz mode	2(TOSC)(BRG + 1)	—	ms	
			1 MHz mode(1)	2(TOSC)(BRG + 1)	—	ms	
106	THD:DAT	Data Input Hold Time	100 kHz mode	0	—	ns	
			400 kHz mode	0	0.9	ms	
			1 MHz mode(1)	TBD	—	ns	
107	TSU:DAT	Data Input Setup Time	100 kHz mode	250	—	ns	**(Note 2)**
			400 kHz mode	100	—	ns	
			1 MHz mode(1)	TBD	—	ns	
92	TSU:STO	Stop Condition Setup Time	100 kHz mode	2(TOSC)(BRG + 1)	—	ms	
			400 kHz mode	2(TOSC)(BRG + 1)	—	ms	
			1 MHz mode(1)	2(TOSC)(BRG + 1)	—	ms	
109	TAA	Output Valid from Clock	100 kHz mode	—	3500	ns	
			400 kHz mode	—	1000	ns	
			1 MHz mode(1)	—	—	ns	
110	TBUF	Bus Free Time	100 kHz mode	4.7	—	ms	Time the bus must be free before a new transmission can start
			400 kHz mode	1.3	—	ms	
			1 MHz mode(1)	TBD	—	ms	
D102	CB	Bus Capacitive Loading		—	400	pF	

Note 1: Maximum pin capacitance = 10 pF for all I²C pins.

2: A Fast mode I²C bus device can be used in a Standard mode I²C bus system, but parameter #107 ≥ 250 ns, must then be met. This will automatically be the case if the device does not stretch the low period of the SCL signal. If such a device does stretch the low period of the SCL signal, it must output the next data bit to the SDA line, parameter #102 + parameter #107 = 1000 + 250 = 1250 ns (for 100 kHz mode), before the SCL line is released.

FIGURE 27-23: USART SYNCHRONOUS TRANSMISSION (MASTER/SLAVE) TIMING

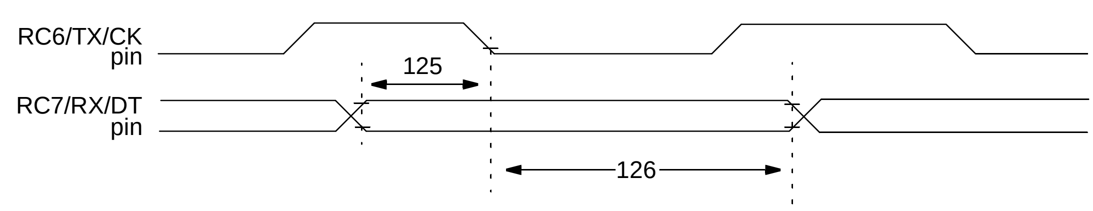


Note: Refer to Figure 27-5 for load conditions.

TABLE 27-23: USART SYNCHRONOUS TRANSMISSION REQUIREMENTS

Param. No.	Symbol	Characteristic		Min	Max	Units	Conditions
120	TCKH2DTV	SYNC XMIT (MASTER & SLAVE) Clock High to Data Out Valid	PIC18FXX8X	—	40	ns	
			PIC18LFXX8X	—	100	ns	
121	TCKRF	Clock Out Rise Time and Fall Time (Master mode)	PIC18FXX8X	—	20	ns	
			PIC18LFXX8X	—	50	ns	
122	TDTRF	Data Out Rise Time and Fall Time	PIC18FXX8X	—	20	ns	
			PIC18LFXX8X	—	50	ns	

FIGURE 27-24: USART SYNCHRONOUS RECEIVE (MASTER/SLAVE) TIMING



Note: Refer to Figure 27-5 for load conditions.

TABLE 27-24: USART SYNCHRONOUS RECEIVE REQUIREMENTS

Param. No.	Symbol	Characteristic	Min	Max	Units	Conditions
125	TDTV2CKL	SYNC RCV (MASTER & SLAVE) Data Hold before CK ↓ (DT hold time)	10	—	ns	
126	TCKL2DTL	Data Hold after CK ↓ (DT hold time)	15	—	ns	

TABLE 27-25: A/D CONVERTER CHARACTERISTICS: PIC18F6585/8585/6680/8680 (INDUSTRIAL, EXTENDED) PIC18LF6585/8585/6680/8680 (INDUSTRIAL)

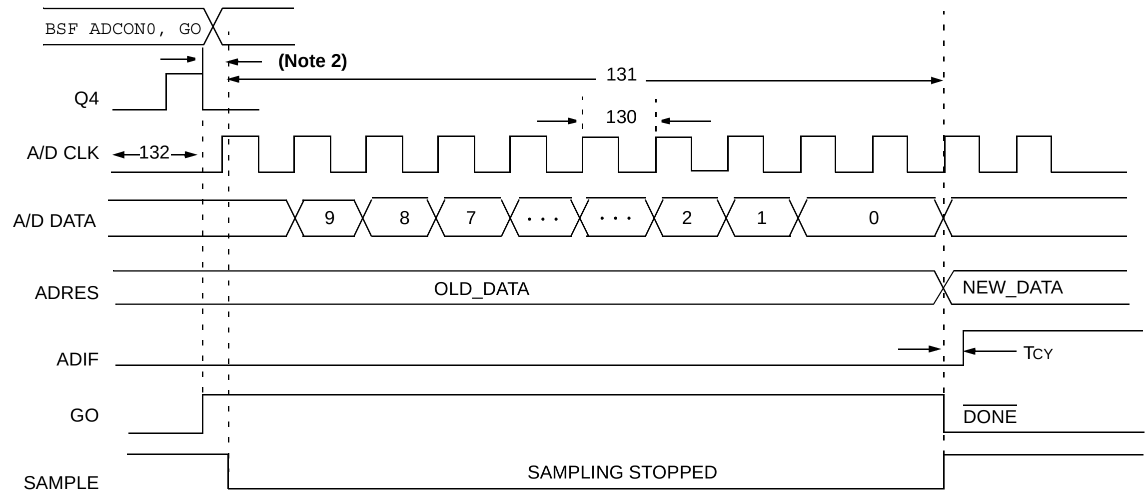
Param No.	Symbol	Characteristic	Min	Typ	Max	Units	Conditions
A01	NR	Resolution	— —	— —	10 TBD	bit bit	VREF = VDD ≥ 3.0V VREF = VDD < 3.0V
A03	EIL	Integral Linearity Error	— —	— —	<±1 TBD	LSb LSb	VREF = VDD ≥ 3.0V VREF = VDD < 3.0V
A04	EDL	Differential Linearity Error	— —	— —	<±1 TBD	LSb LSb	VREF = VDD ≥ 3.0V VREF = VDD < 3.0V
A05	EFS	Full-Scale Error	— —	— —	<±1 TBD	LSb LSb	VREF = VDD ≥ 3.0V VREF = VDD < 3.0V
A06	EOFF	Offset Error	— —	— —	<±1 TBD	LSb LSb	VREF = VDD ≥ 3.0V VREF = VDD < 3.0V
A10	—	Monotonicity	guaranteed(3)			—	VSS ≤ VAIN ≤ VREF
A20 A20A	VREF	Reference Voltage (VREFH – VREFL)	0V 3V	— —	— —	V V	 For 10-bit resolution
A21	VREFH	Reference Voltage High	AVss	—	AVDD + 0.3V	V	
A22	VREFL	Reference Voltage Low	AVss – 0.3V	—	AVDD	V	
A25	VAIN	Analog Input Voltage	AVSS – 0.3V	—	VREF + 0.3V	V	
A30	ZAIN	Recommended Impedance of Analog Voltage Source	—	—	10.0	kΩ	
A40	IAD	A/D Conversion Current (VDD) — PIC18FXX8X	—	180	—	μA	Average current consumption when A/D is on **(Note 1)**
		A/D Conversion Current (VDD) — PIC18LFXX8X	—	90	—	μA	
A50	IREF	VREF Input Current **(Note 2)**	— —	— —	5 150	μA μA	During VAIN acquisition. During A/D conversion cycle.

Note 1: When A/D is off, it will not consume any current other than minor leakage current. The power-down current spec includes any such leakage from the A/D module.
VREF current is from RA2/AN2/VREF- and RA3/AN3/VREF+ pins or AVDD and AVSS pins, whichever is selected as reference input.

2: Vss ≤ VAIN ≤ VREF

3: The A/D conversion result never decreases with an increase in the input voltage and has no missing codes.

FIGURE 27-25: A/D CONVERSION TIMING



Note 1: If the A/D clock source is selected as RC, a time of TCY is added before the A/D clock starts. This allows the `SLEEP` instruction to be executed.

2: This is a minimal RC delay (typically 100 ns) which also disconnects the holding capacitor from the analog input.

TABLE 27-26: A/D CONVERSION REQUIREMENTS

Param. No.	Symbol	Characteristic		Min	Max	Units	Conditions
130	TAD	A/D Clock Period	PIC18FXX8X	1.6	20(5)	μs	TOSC based, VREF ≥ 3.0V
			PIC18LFXX8X	3.0	20(5)	μs	TOSC based, VREF full range
			PIC18FXX8X	2.0	6.0	μs	A/D RC mode
			PIC18LFXX8X	3.0	9.0	μs	A/D RC mode
131	TCNV	Conversion Time (not including acquisition time) **(Note 1)**		11	12	TAD	
132	TACQ	Acquisition Time **(Note 3)**		15 10	— —	μs μs	-40°C ≤ Temp ≤ +125°C 0°C ≤ Temp ≤ +125°C
135	TSWC	Switching Time from Convert → Sample		—	**(Note 4)**		
136	TAMP	Amplifier Settling Time **(Note 2)**		1	—	μs	This may be used if the “new” input voltage has not changed by more than 1 LSb (i.e., 5 mV @ 5.12V) from the last sampled voltage (as stated on CHOLD).

Note 1: ADRES register may be read on the following TCY cycle.

2: See **Section 19.0 “10-bit Analog-to-Digital Converter (A/D) Module”** for minimum conditions when input voltage has changed more than 1 LSb.

3: The time for the holding capacitor to acquire the “New” input voltage when the voltage changes full scale after the conversion (AVDD to AVSS, or AVSS to AVDD). The source impedance (*RS*) on the input channels is 50Ω.

4: On the next Q4 cycle of the device clock.

5: The time of the A/D clock period is dependent on the device frequency and the TAD clock divider.

NOTES:

28.0 DC AND AC CHARACTERISTICS GRAPHS AND TABLES

Note: The graphs and tables provided following this note are a statistical summary based on a limited number of samples and are provided for informational purposes only. The performance characteristics listed herein are not tested or guaranteed. In some graphs or tables, the data presented may be outside the specified operating range (e.g., outside specified power supply range) and therefore, outside the warranted range.

"Typical" represents the mean of the distribution at 25°C. "Maximum" or "minimum" represents (mean + 3σ) or (mean – 3σ) respectively, where σ is a standard deviation, over the whole temperature range.

FIGURE 28-1: TYPICAL IDD vs. FOSC OVER VDD (HS MODE)

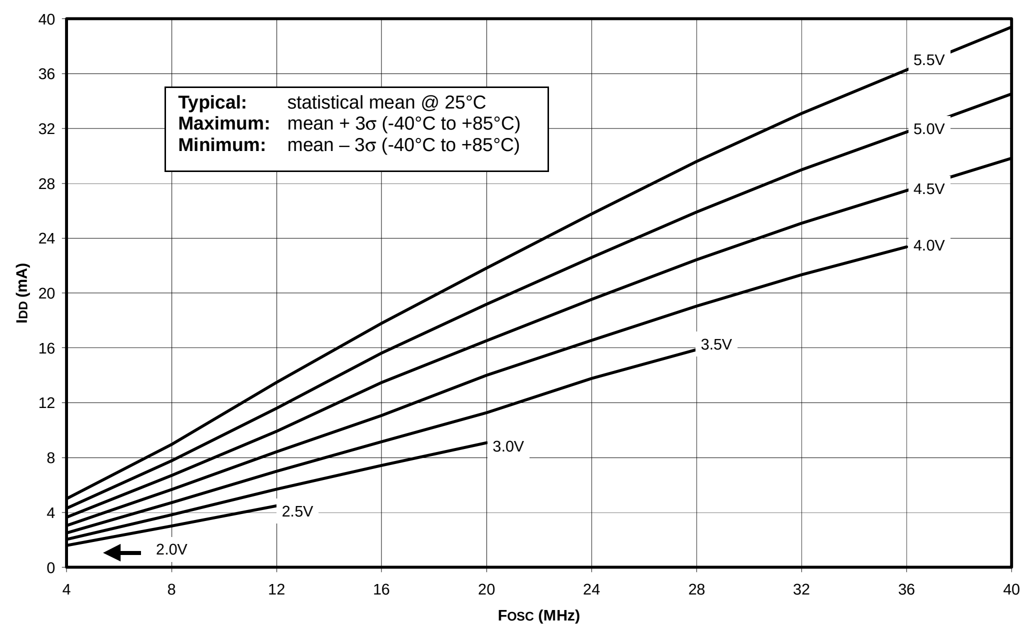


FIGURE 28-2: MAXIMUM IDD vs. FOSC OVER VDD (HS MODE)

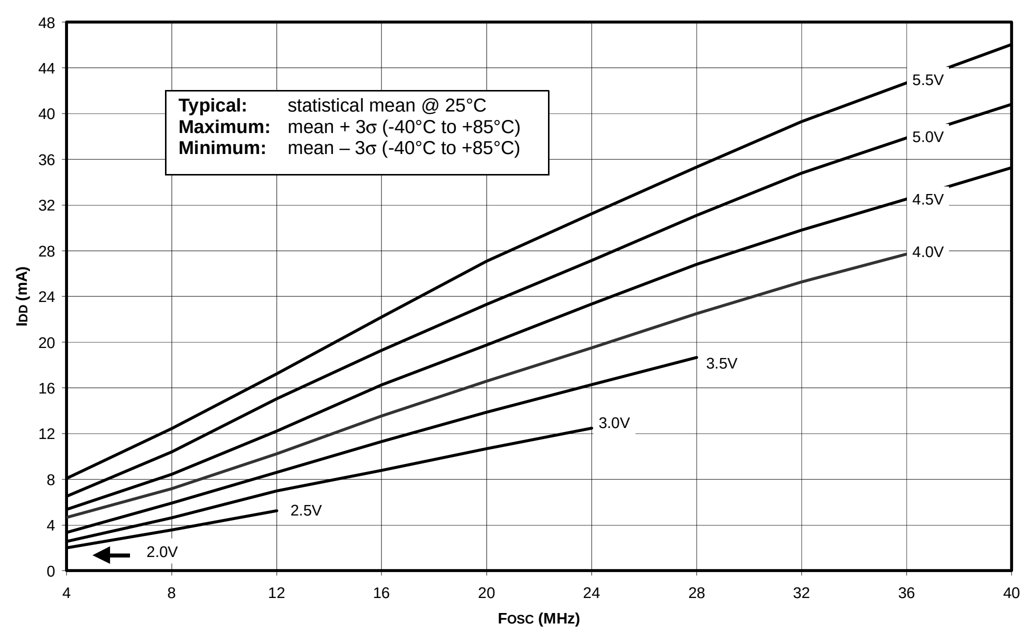


FIGURE 28-3: TYPICAL IDD vs. FOSC OVER VDD (HS/PLL MODE)

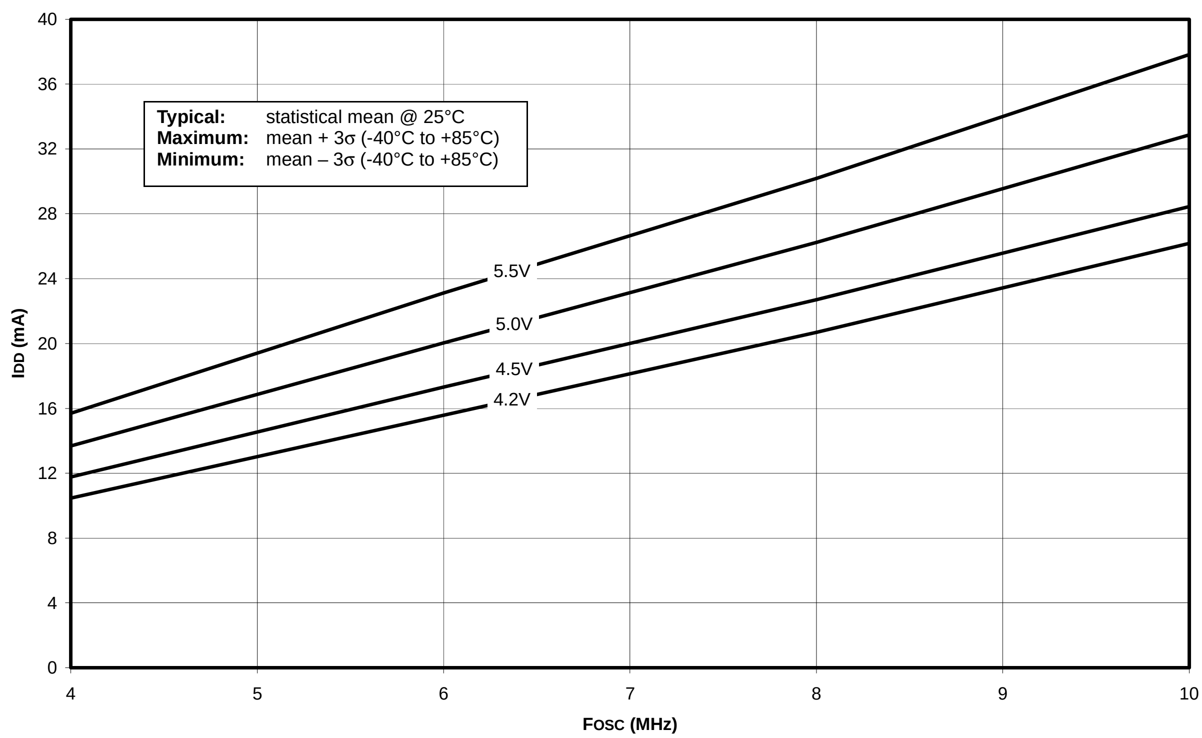


FIGURE 28-4: MAXIMUM IDD vs. FOSC OVER VDD (HS/PLL MODE)

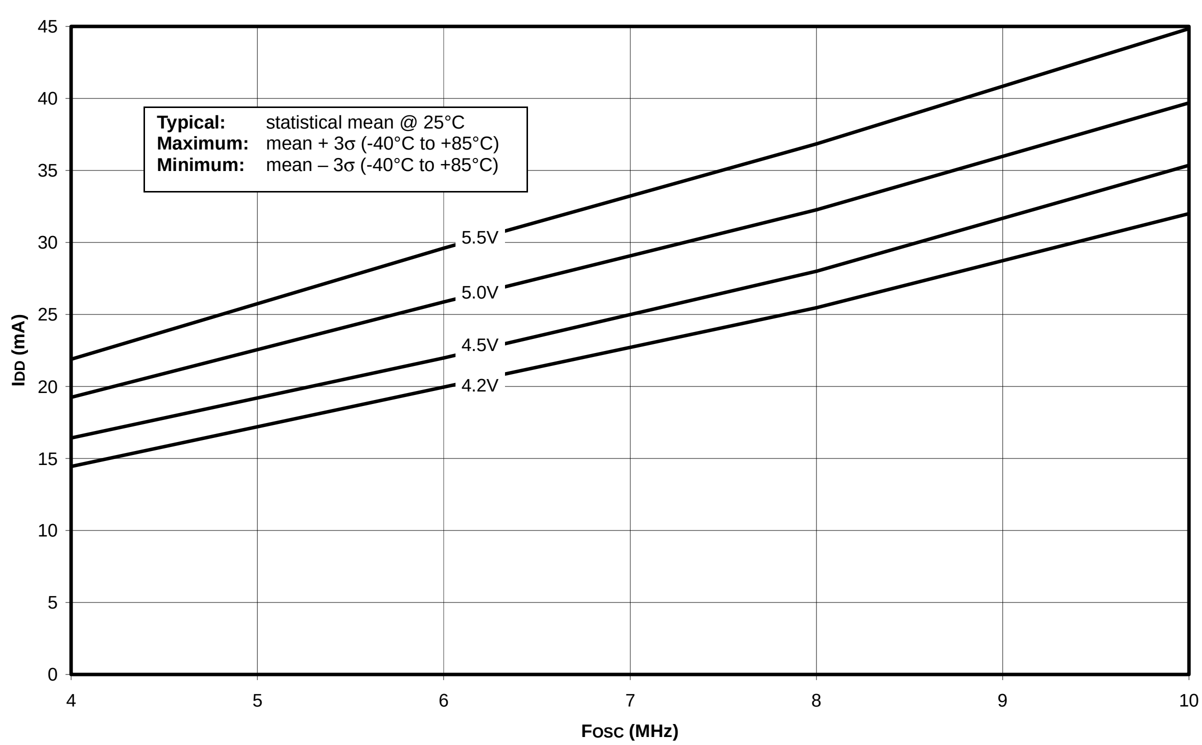


FIGURE 28-5: TYPICAL IDD vs. FOSC OVER VDD (XT MODE)

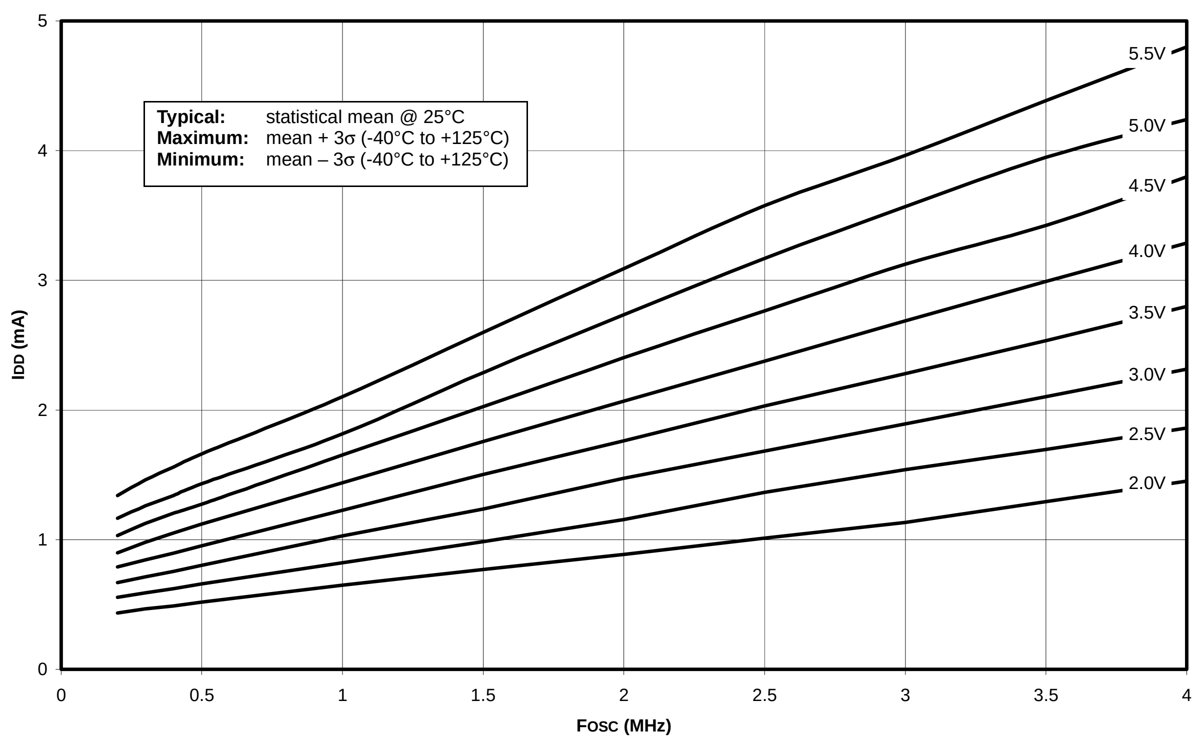


FIGURE 28-6: MAXIMUM IDD vs. FOSC OVER VDD (XT MODE)

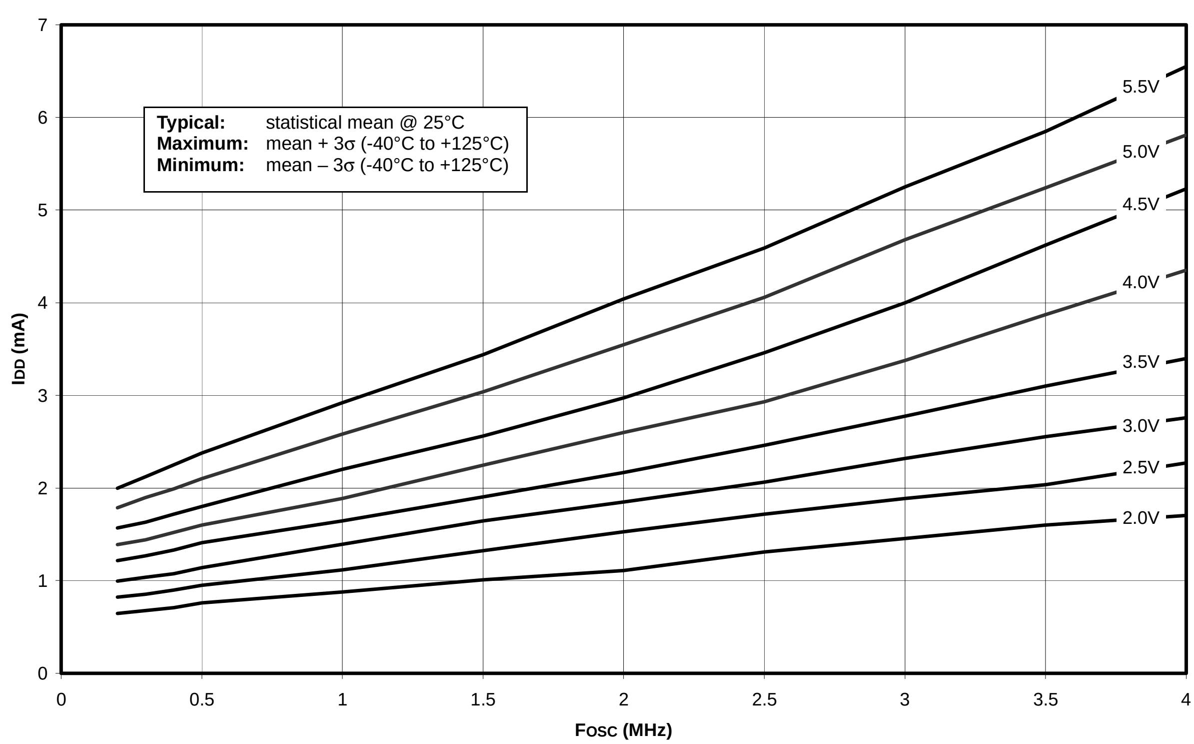


FIGURE 28-7: TYPICAL IDD vs. FOSC OVER VDD (LP MODE)

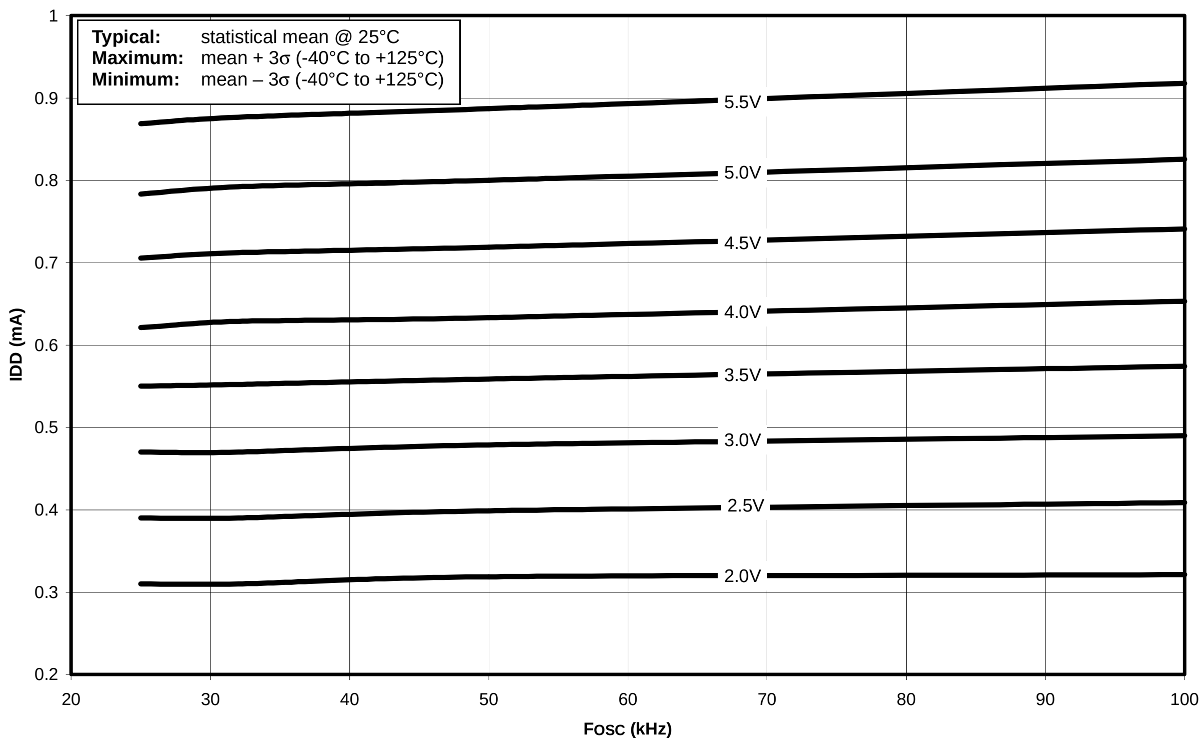


FIGURE 28-8: MAXIMUM IDD vs. FOSC OVER VDD (LP MODE)

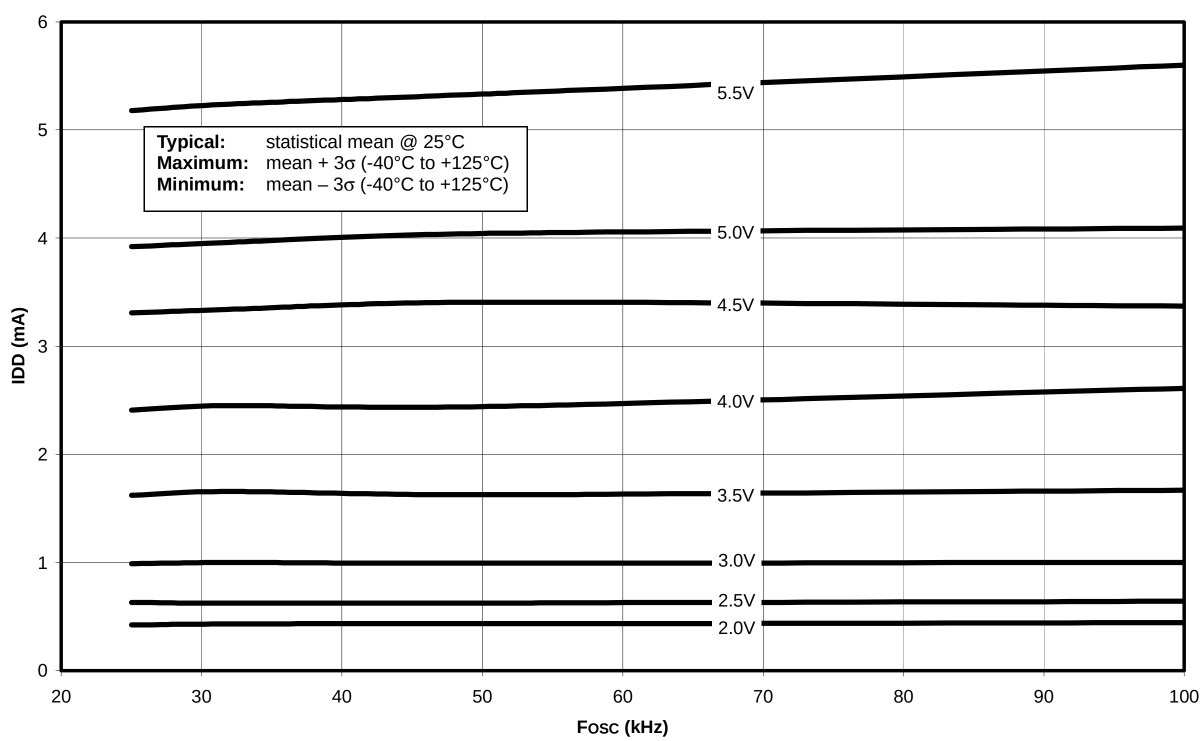


FIGURE 28-9: TYPICAL IDD vs. FOSC OVER VDD (EC MODE)

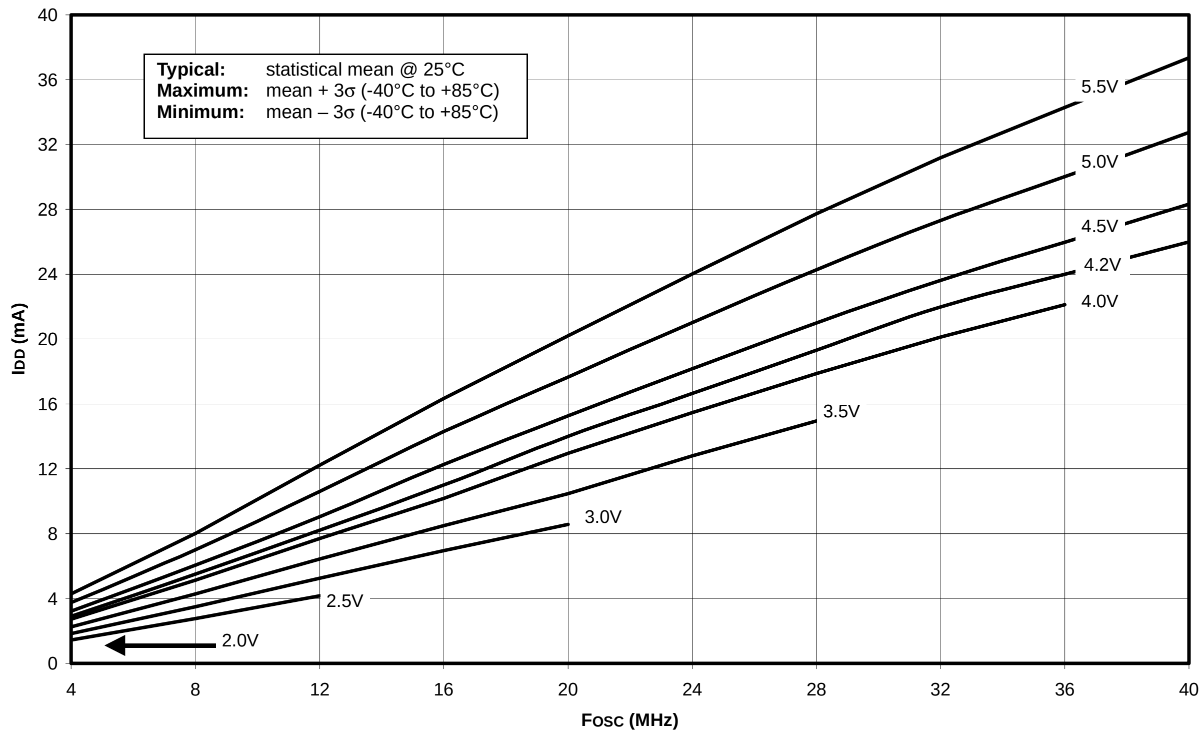


FIGURE 28-10: MAXIMUM IDD vs. FOSC OVER VDD (EC MODE)

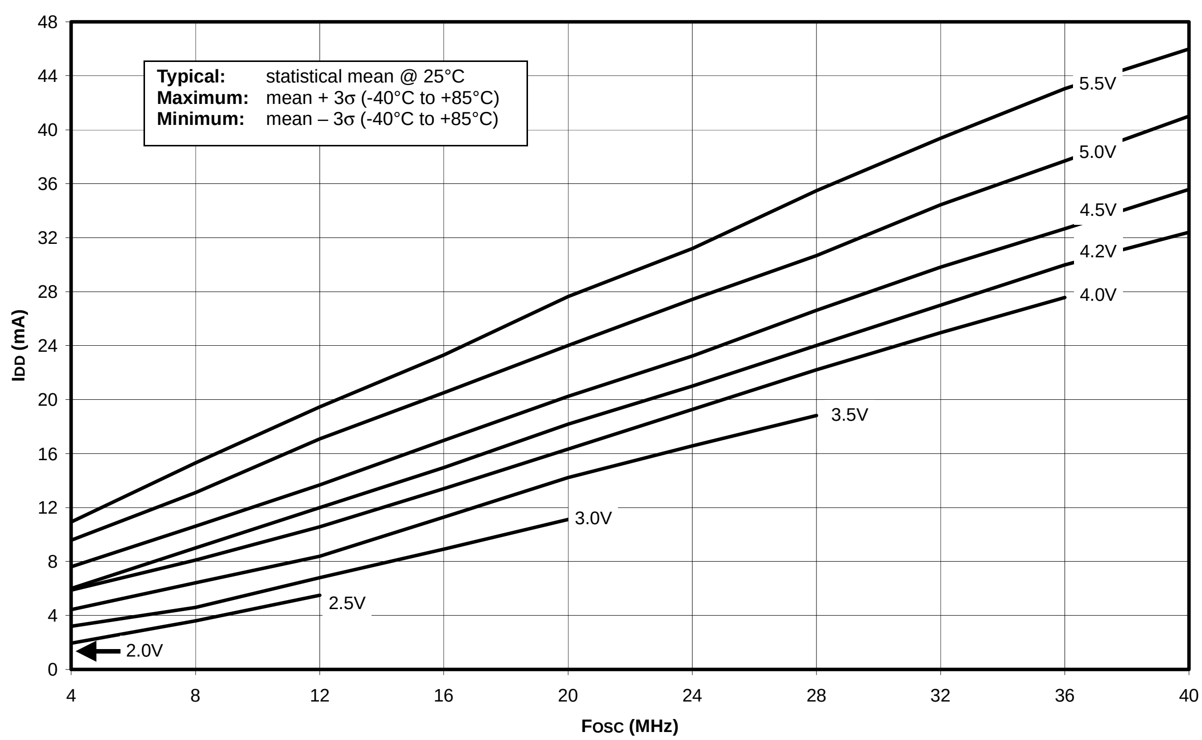


FIGURE 28-11: TYPICAL AND MAXIMUM I_{T1OSC} vs. V_{DD} (TIMER1 AS SYSTEM CLOCK)

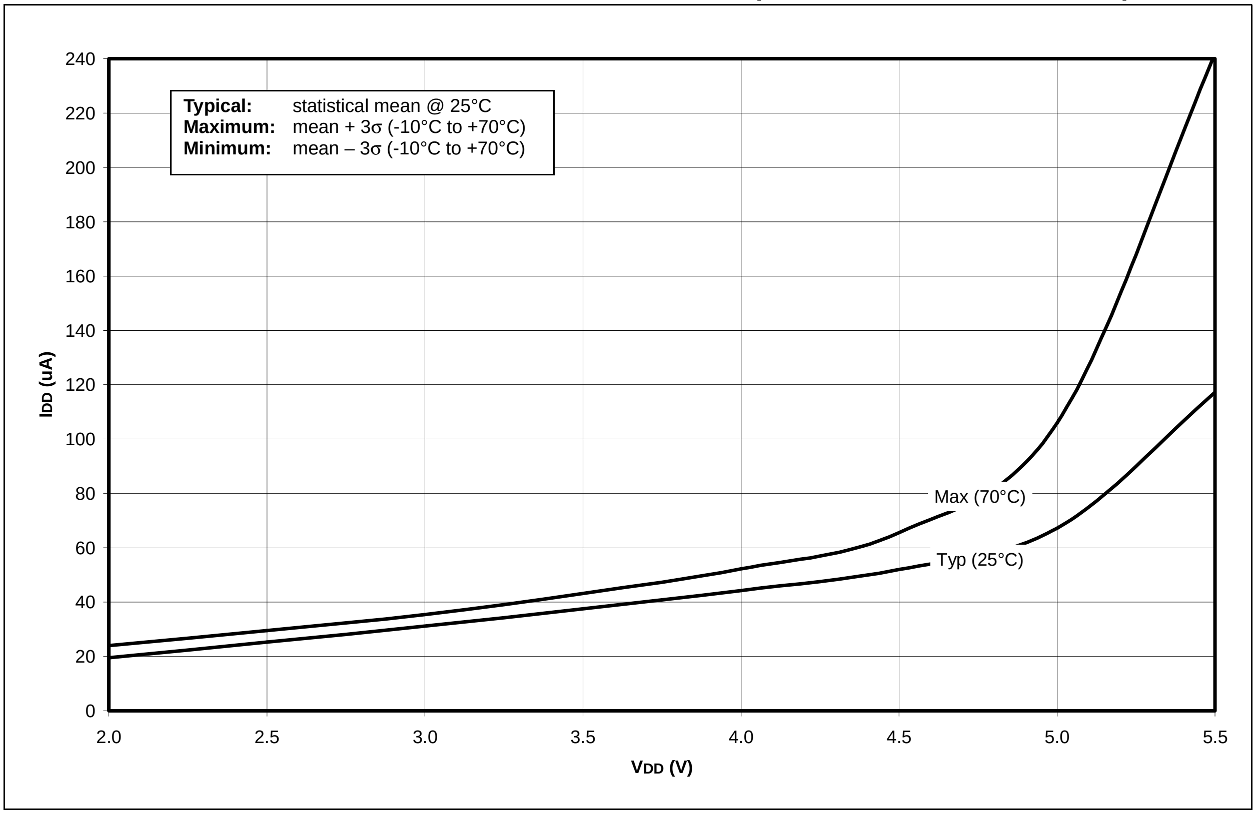


FIGURE 28-12: AVERAGE F_{OSC} vs. V_{DD} FOR VARIOUS R's (RC MODE, C = 20 pF, TEMP = 25°C)

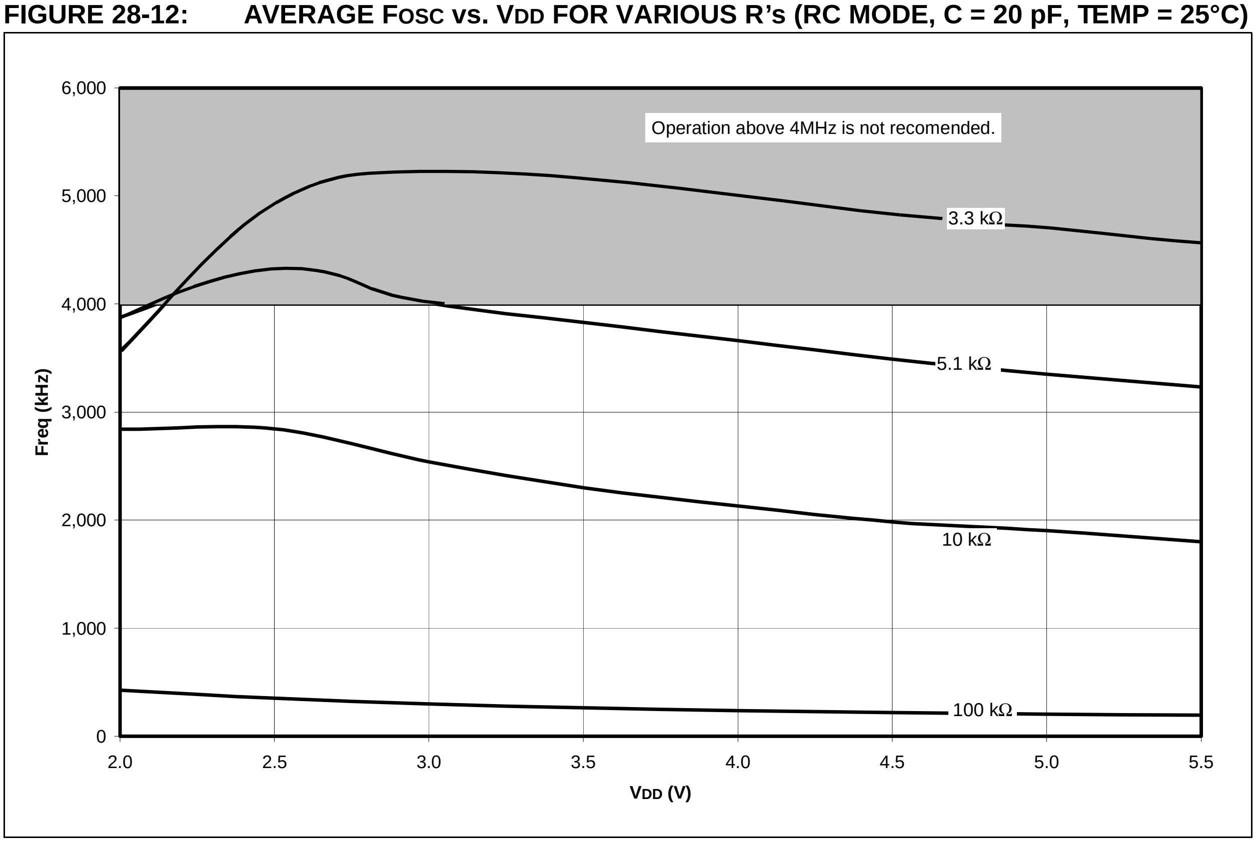


FIGURE 28-13: AVERAGE FOSC vs. VDD FOR VARIOUS R's (RC MODE, C = 100 pF, TEMP = 25°C)

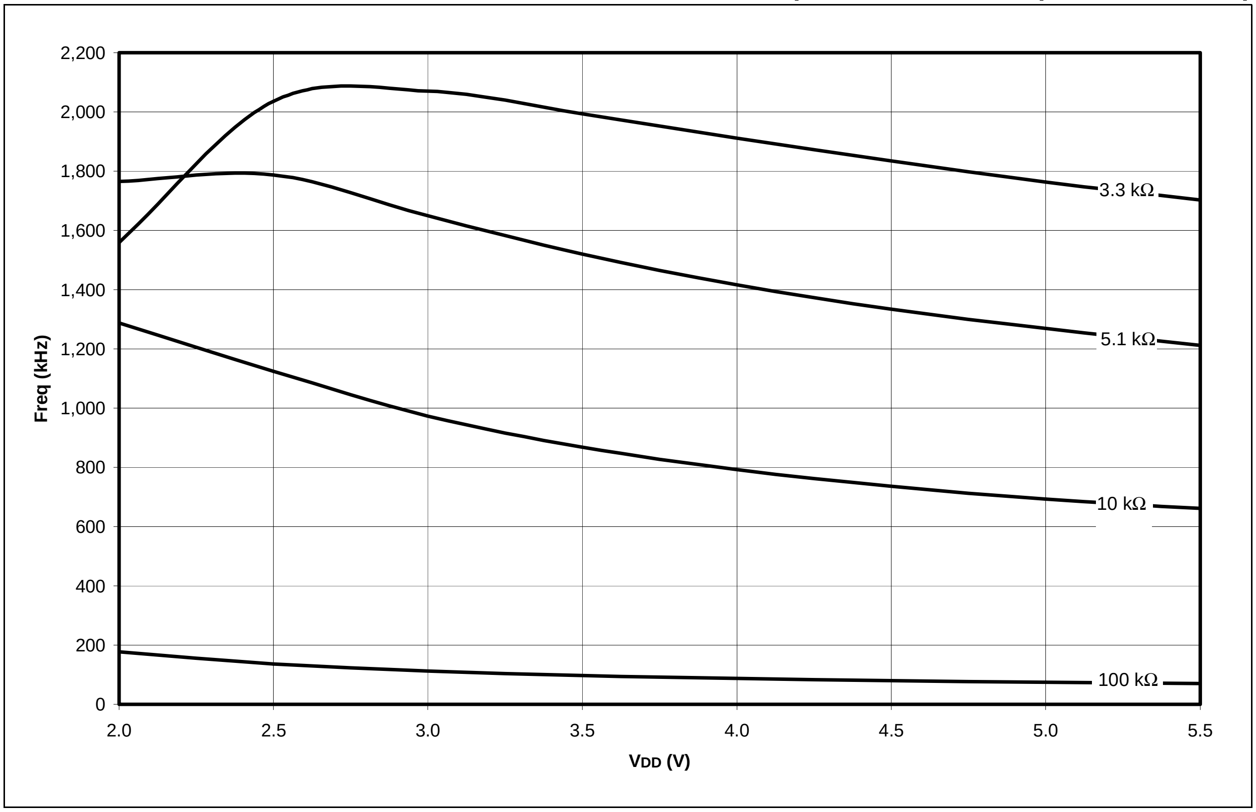


FIGURE 28-14: AVERAGE FOSC vs. VDD FOR VARIOUS R's (RC MODE, C = 300 pF, TEMP = 25°C)

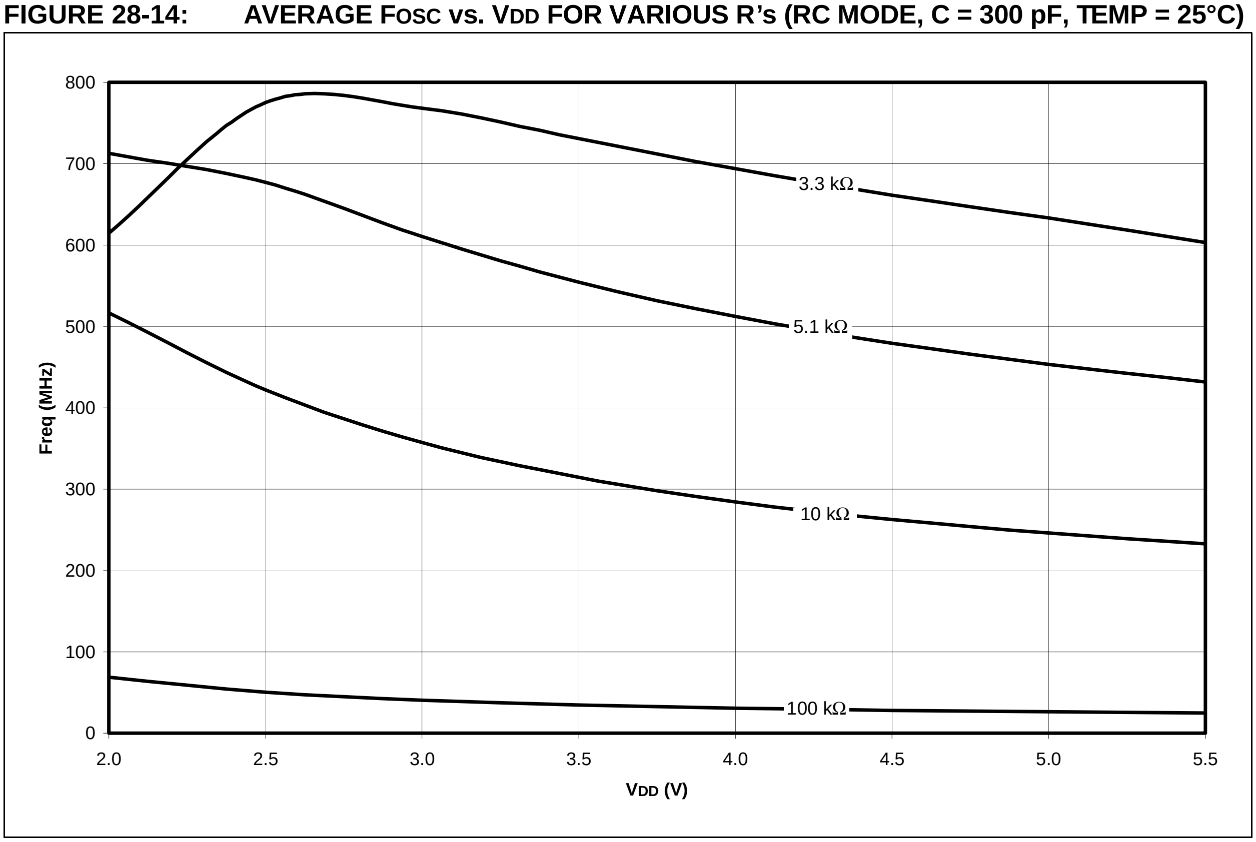


FIGURE 28-15: IPD vs. VDD (SLEEP MODE, ALL PERIPHERALS DISABLED)

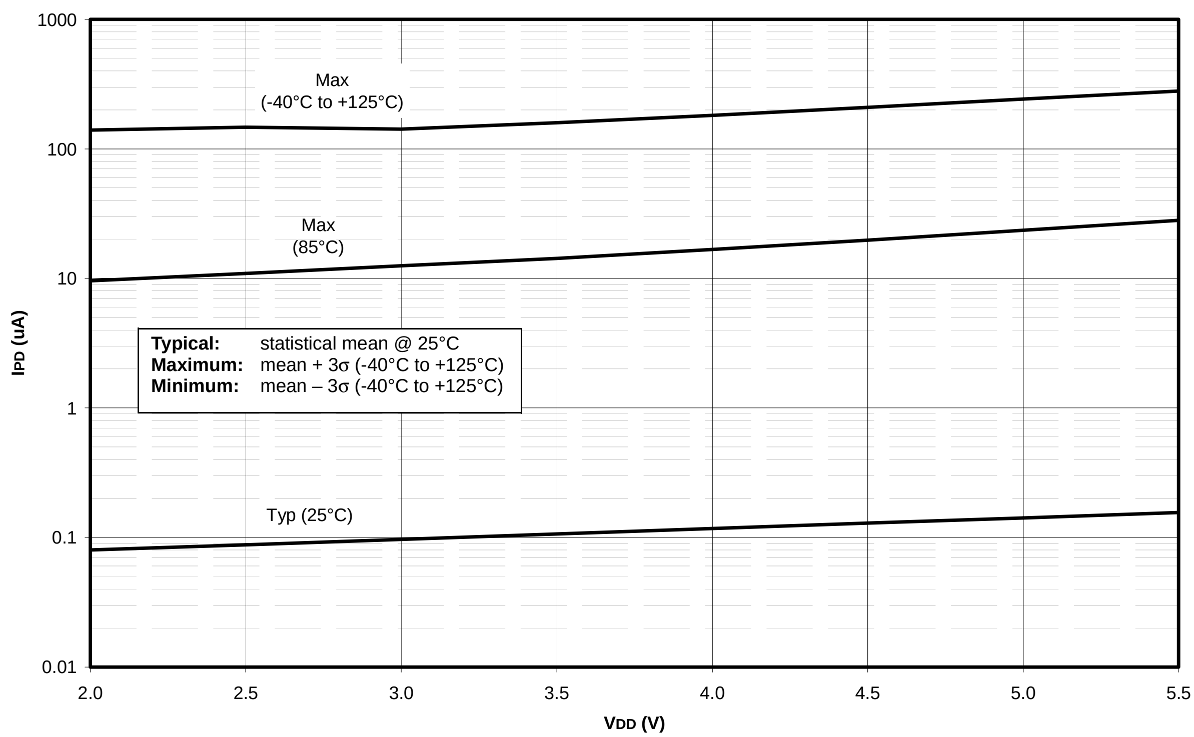


FIGURE 28-16: TYPICAL AND MAXIMUM ΔIBOR vs. VDD OVER TEMPERATURE, VBOR = 2.00V-2.16V

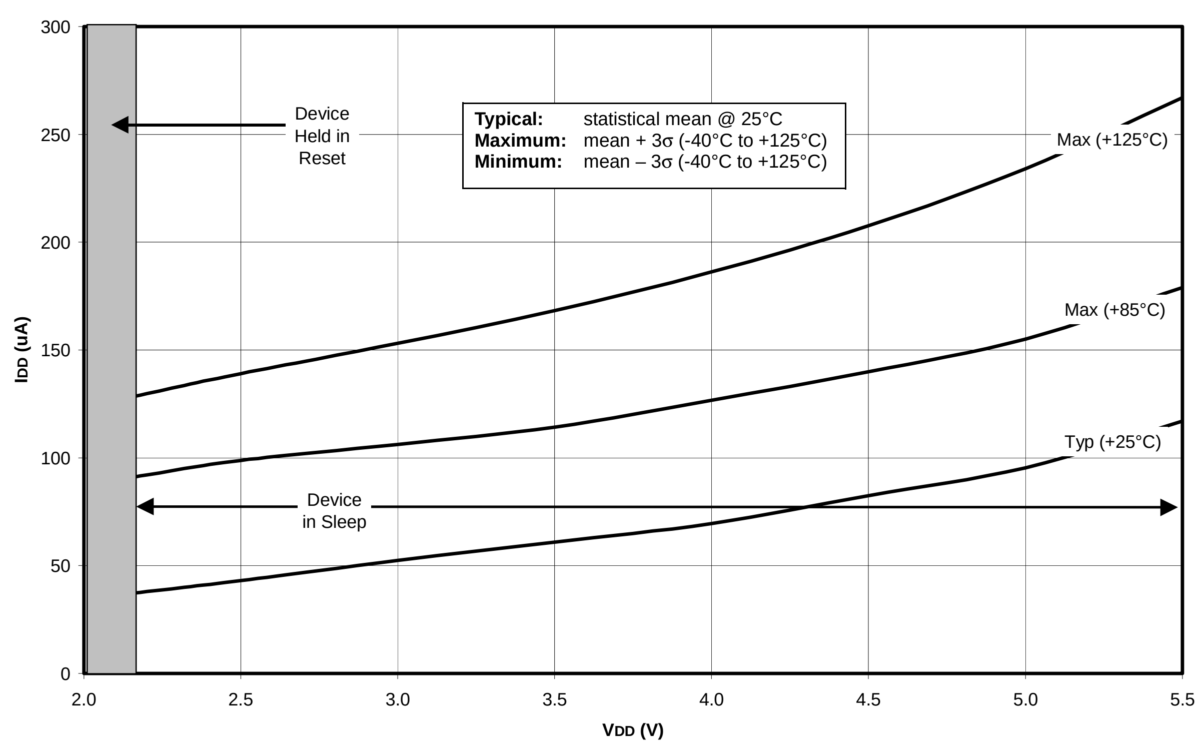


FIGURE 28-17: I_{T1OSC} vs. V_{DD} (SLEEP MODE, TIMER1 AND OSCILLATOR ENABLED)

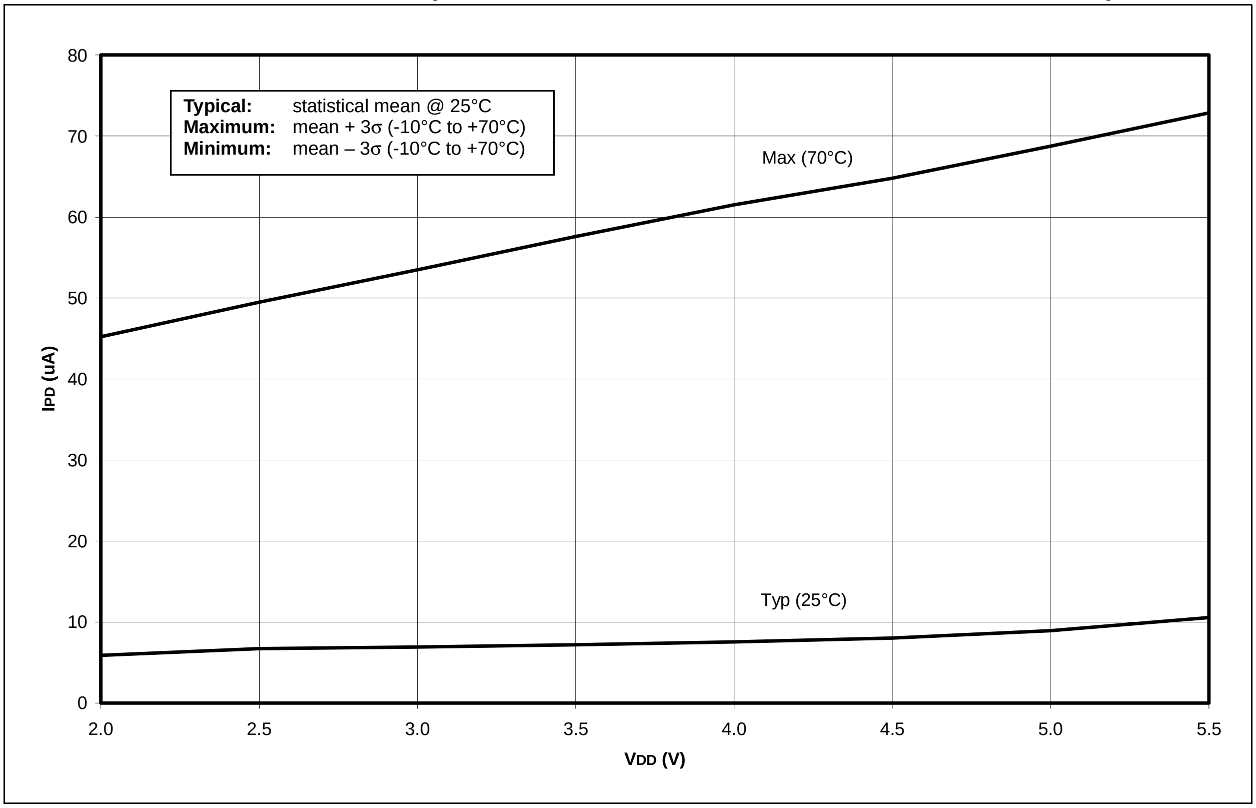


FIGURE 28-18: I_{PD} vs. V_{DD} (SLEEP MODE, WDT ENABLED)

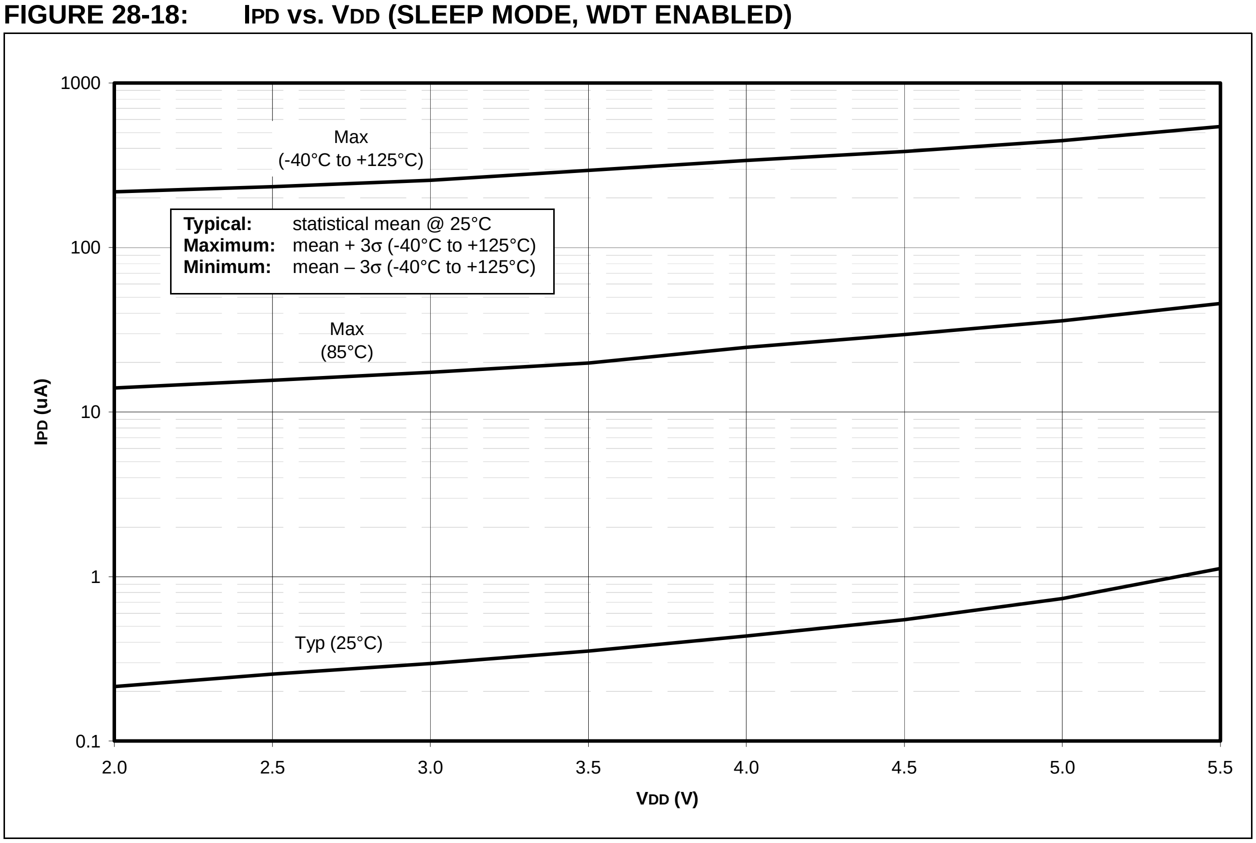


FIGURE 28-19: TYPICAL, MINIMUM AND MAXIMUM WDT PERIOD vs. VDD

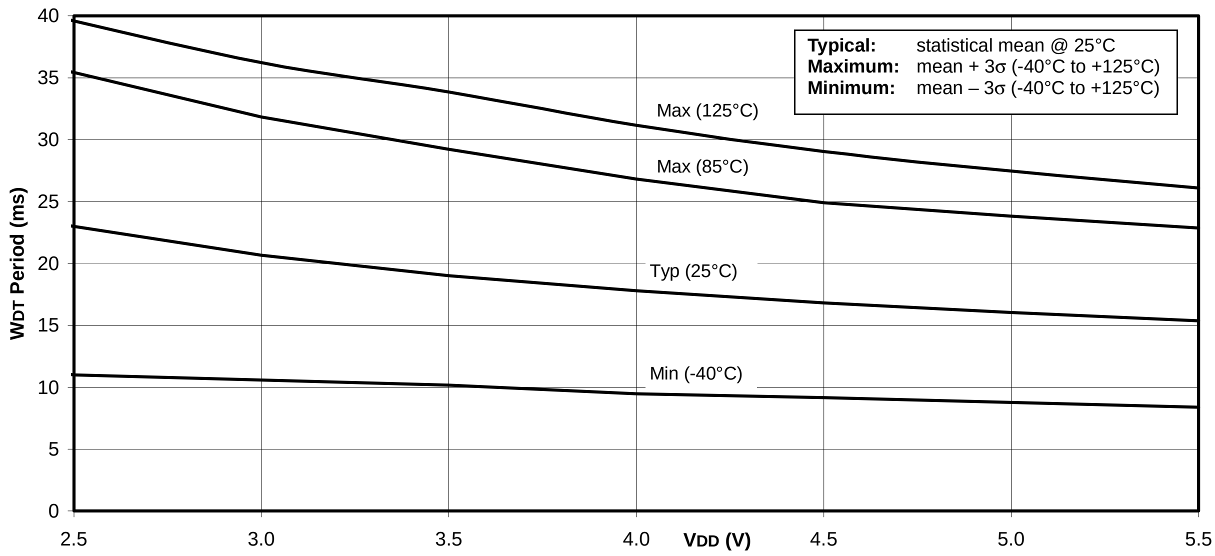


FIGURE 28-20: ΔILVD vs. VDD OVER TEMPERATURE, VLVD = 4.5-4.78V

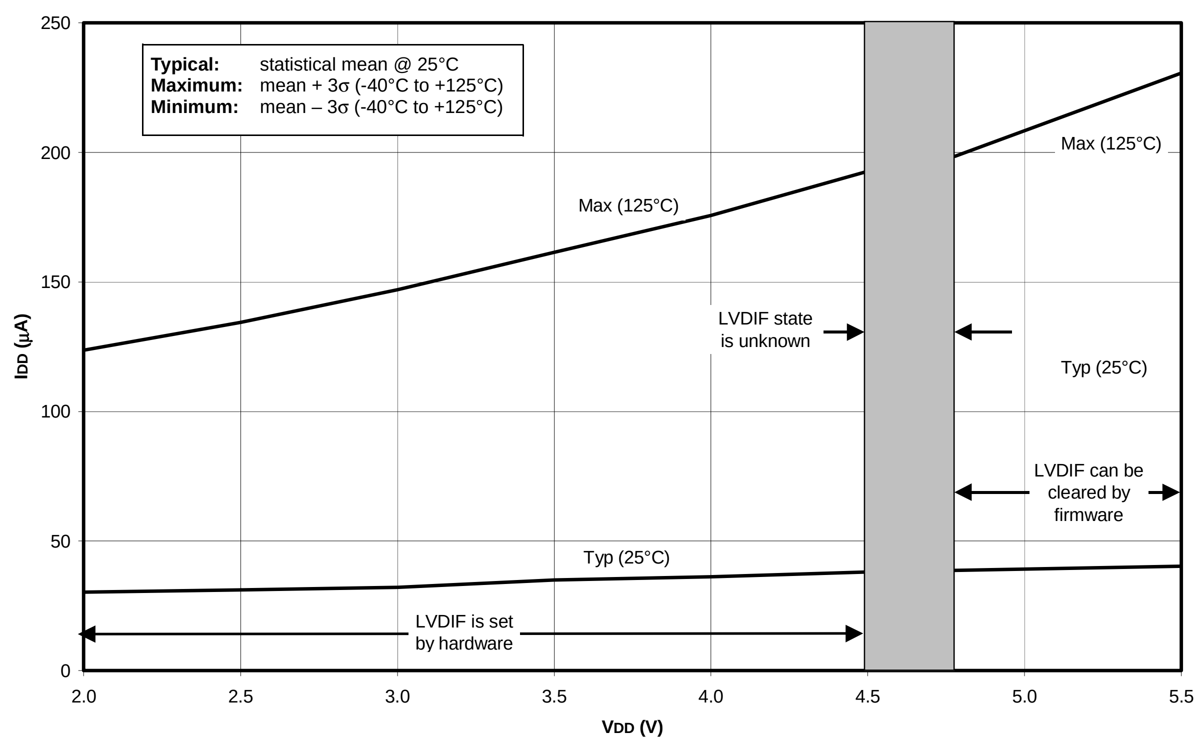


FIGURE 28-21: TYPICAL, MINIMUM AND MAXIMUM V_{OH} vs. I_{OH} (V_{DD} = 5V, -40°C TO +125°C)

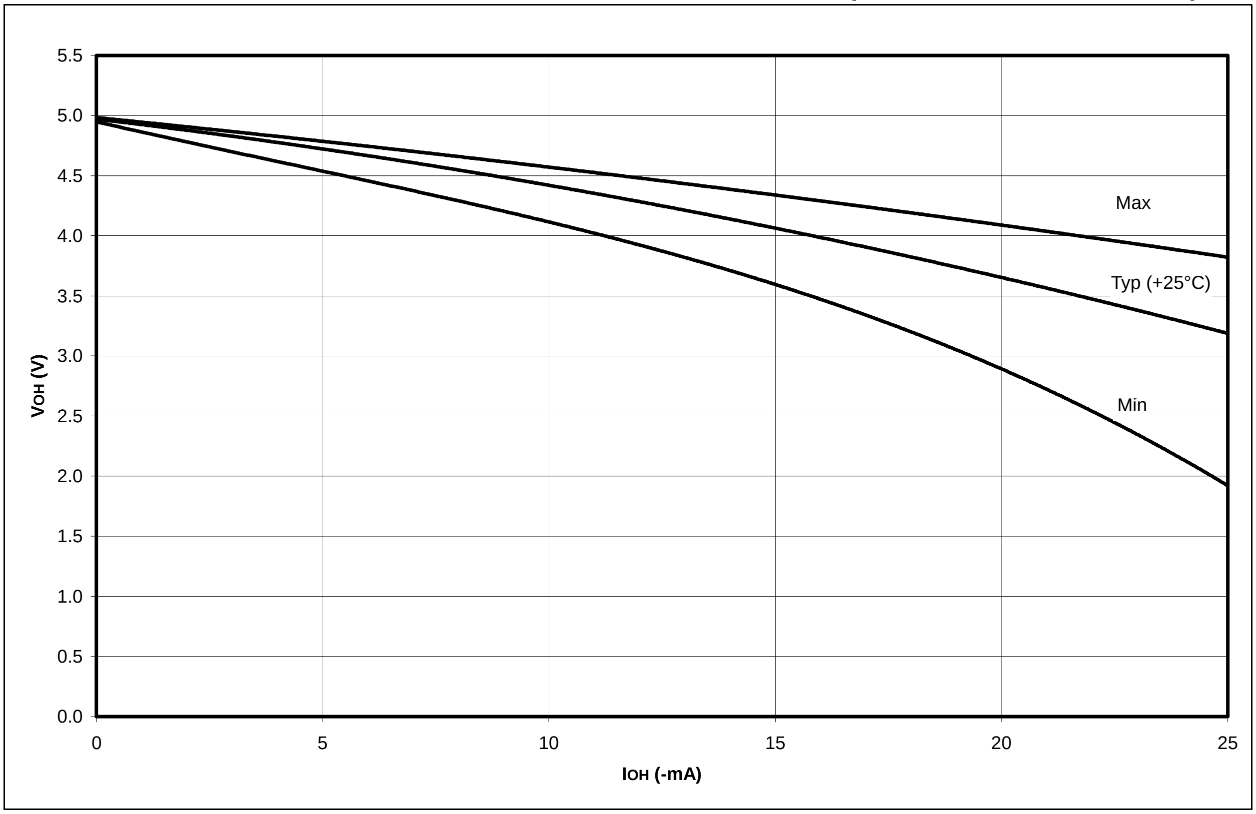


FIGURE 28-22: TYPICAL, MINIMUM AND MAXIMUM V_{OH} vs. I_{OH} (V_{DD} = 3V, -40°C TO +125°C)

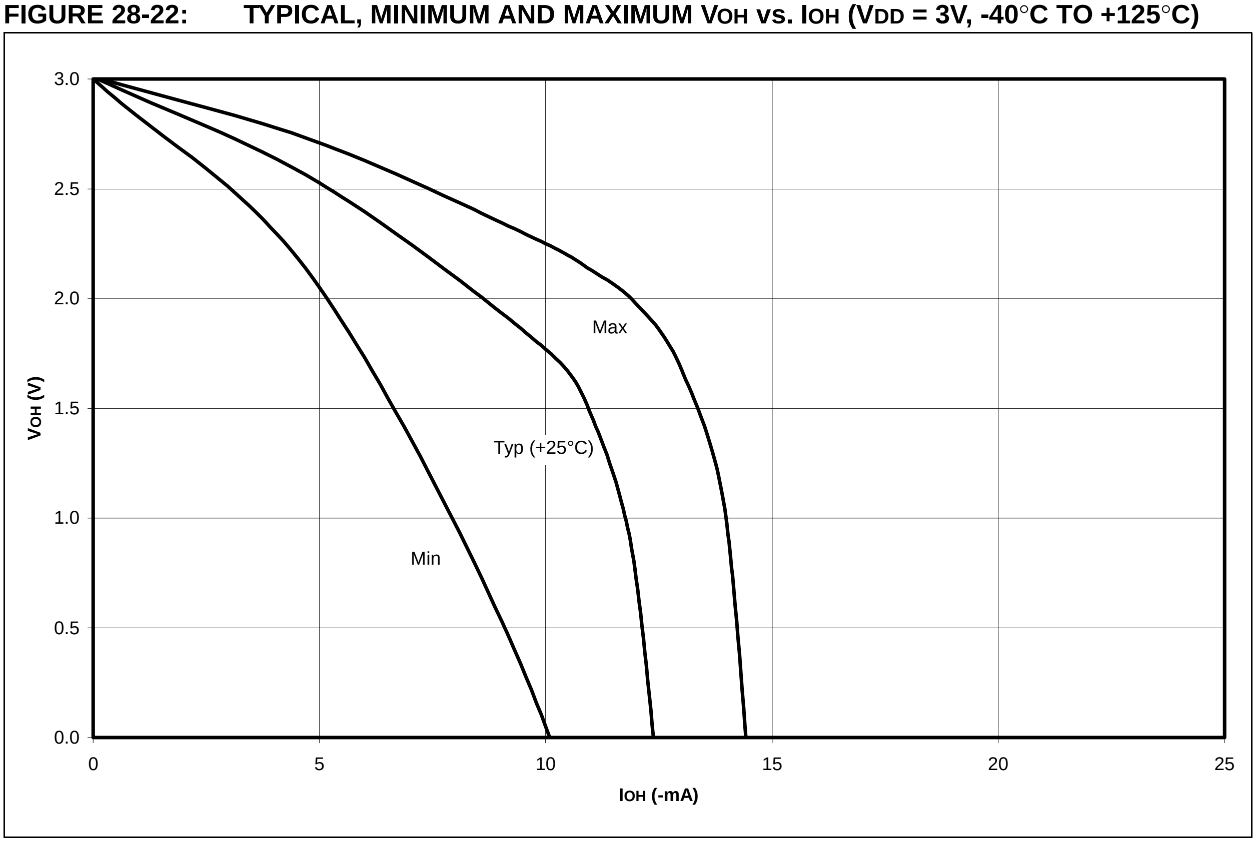


FIGURE 28-23: TYPICAL AND MAXIMUM V_{OL} vs. I_{OL} (V_{DD} = 5V, -40°C TO +125°C)

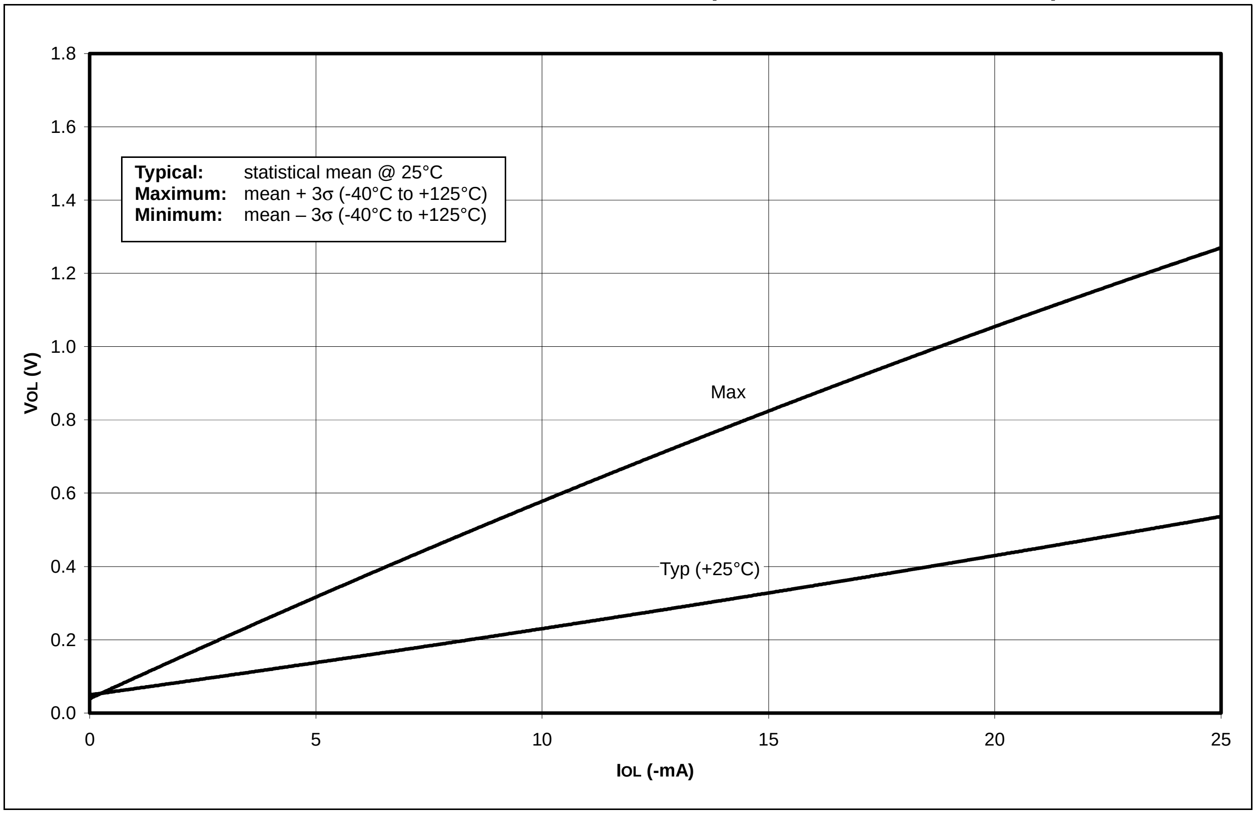


FIGURE 28-24: TYPICAL AND MAXIMUM V_{OL} vs. I_{OL} (V_{DD} = 3V, -40°C TO +125°C)

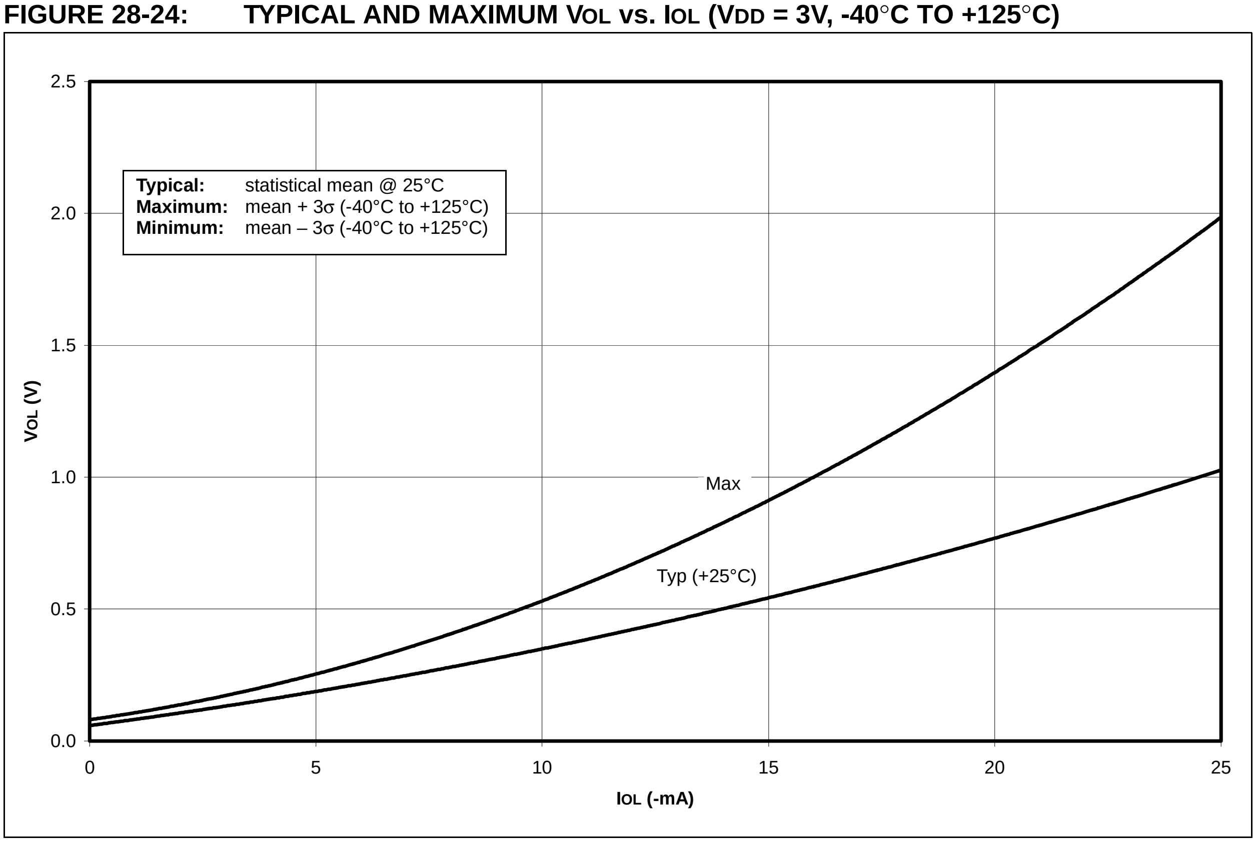


FIGURE 28-25: MINIMUM AND MAXIMUM VIN vs. VDD (ST INPUT, -40°C TO +125°C)

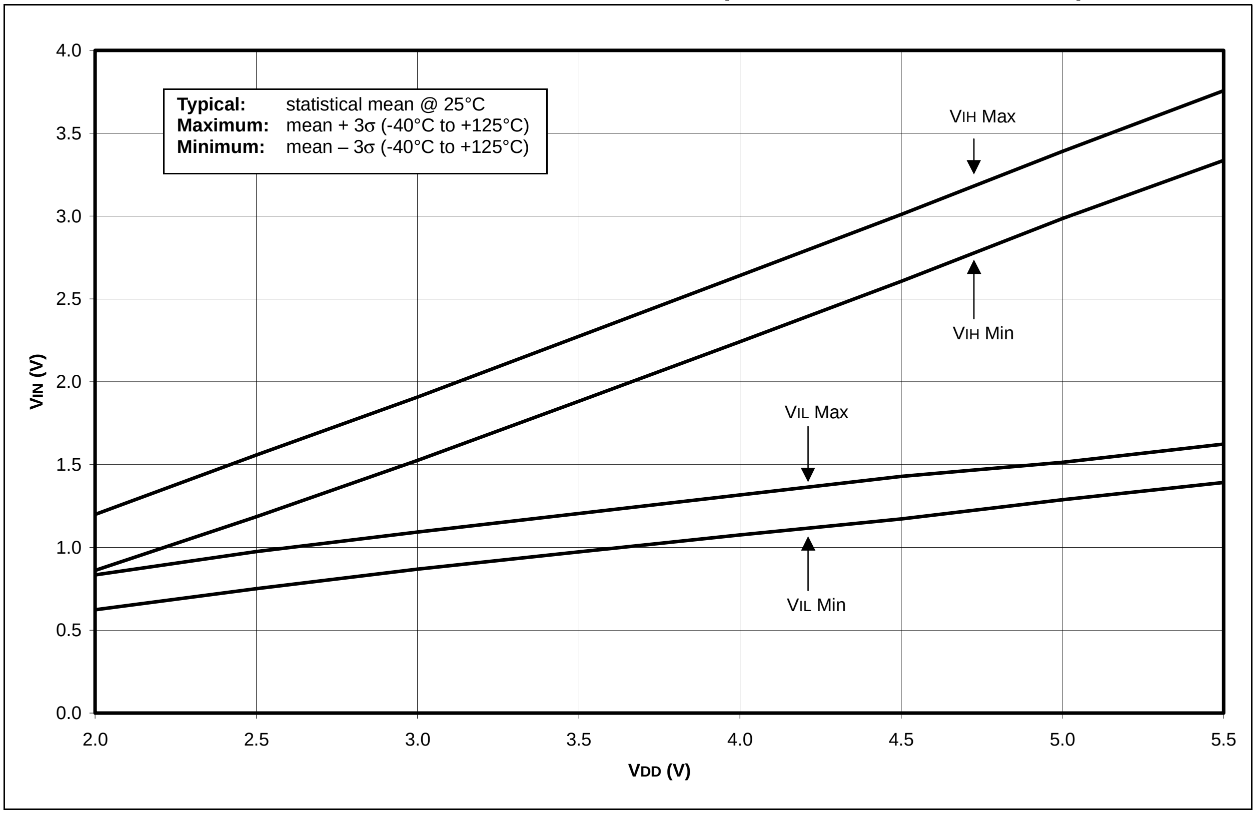


FIGURE 28-26: MINIMUM AND MAXIMUM VIN vs. VDD (TTL INPUT, -40°C TO +125°C)

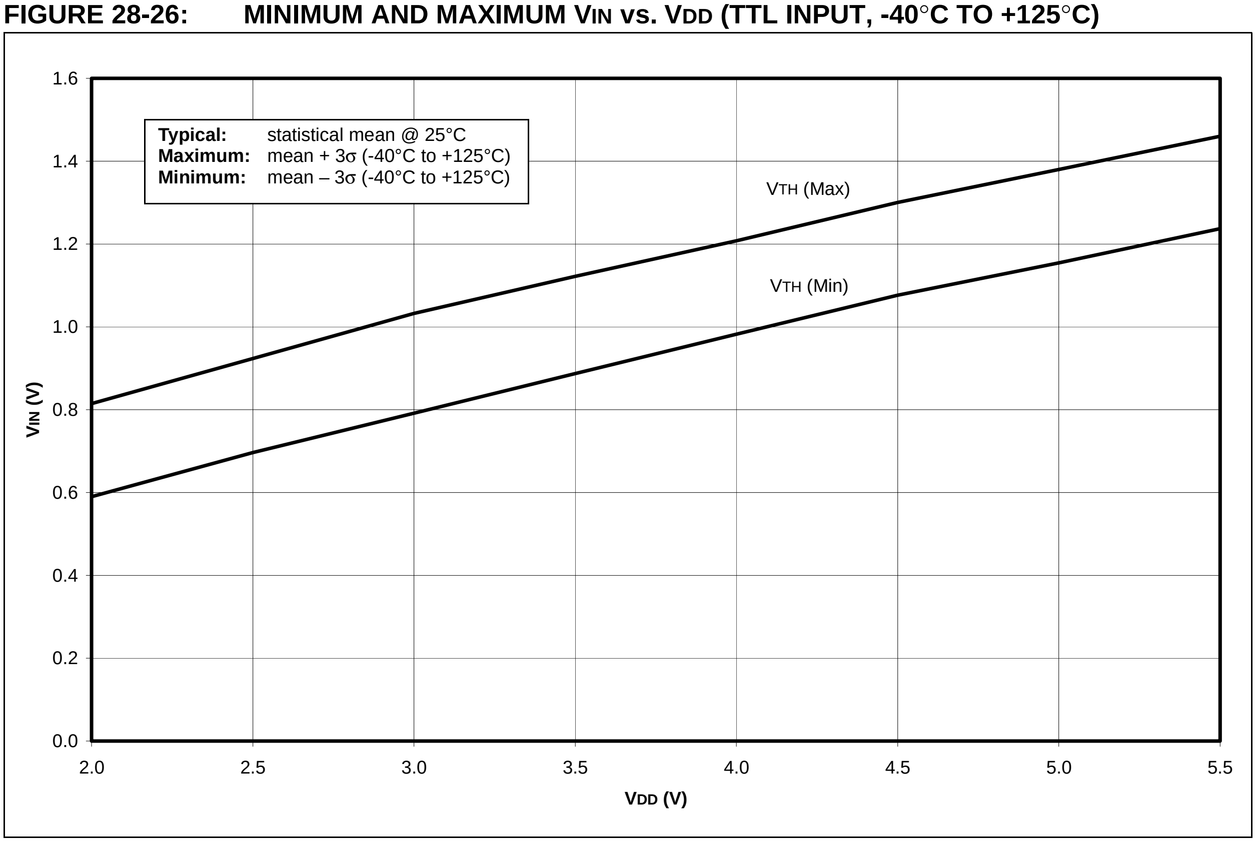


FIGURE 28-27: MINIMUM AND MAXIMUM V_{IN} vs. V_{DD} (I^2C INPUT, -40°C TO +125°C)

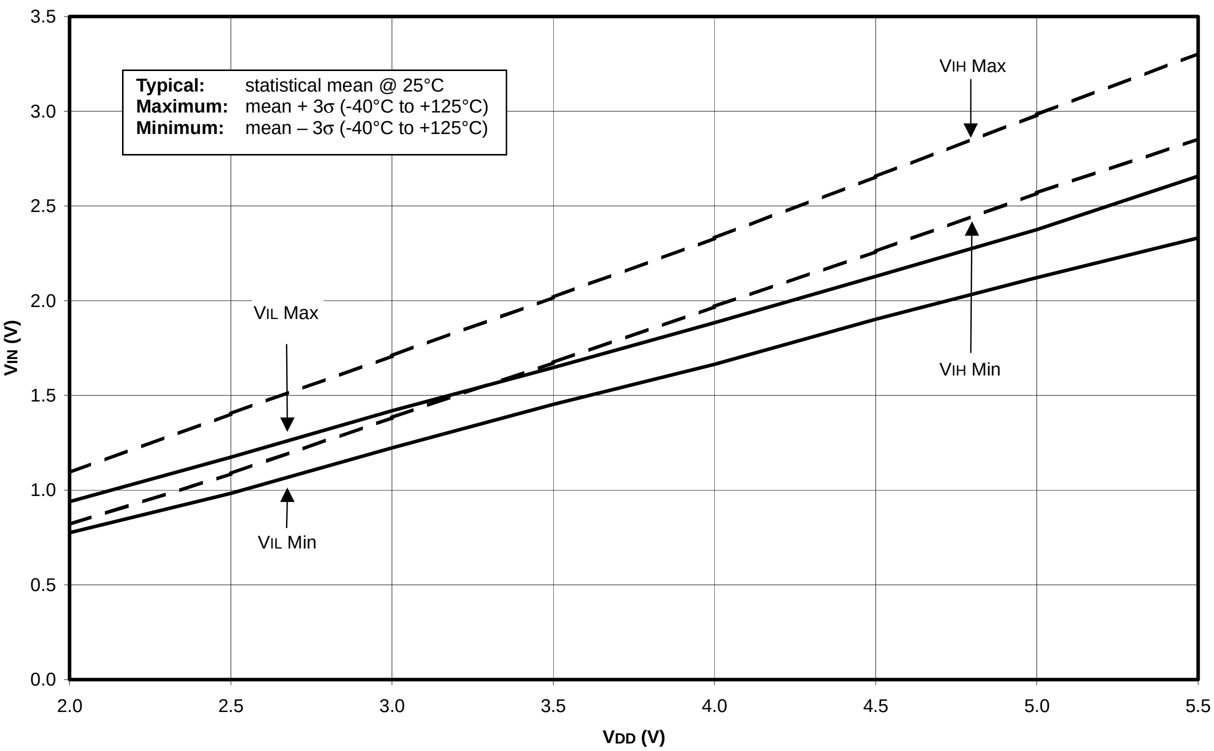


FIGURE 28-28: A/D NONLINEARITY vs. V_{REFH} (V_{DD} = V_{REFH}, -40°C TO +125°C)

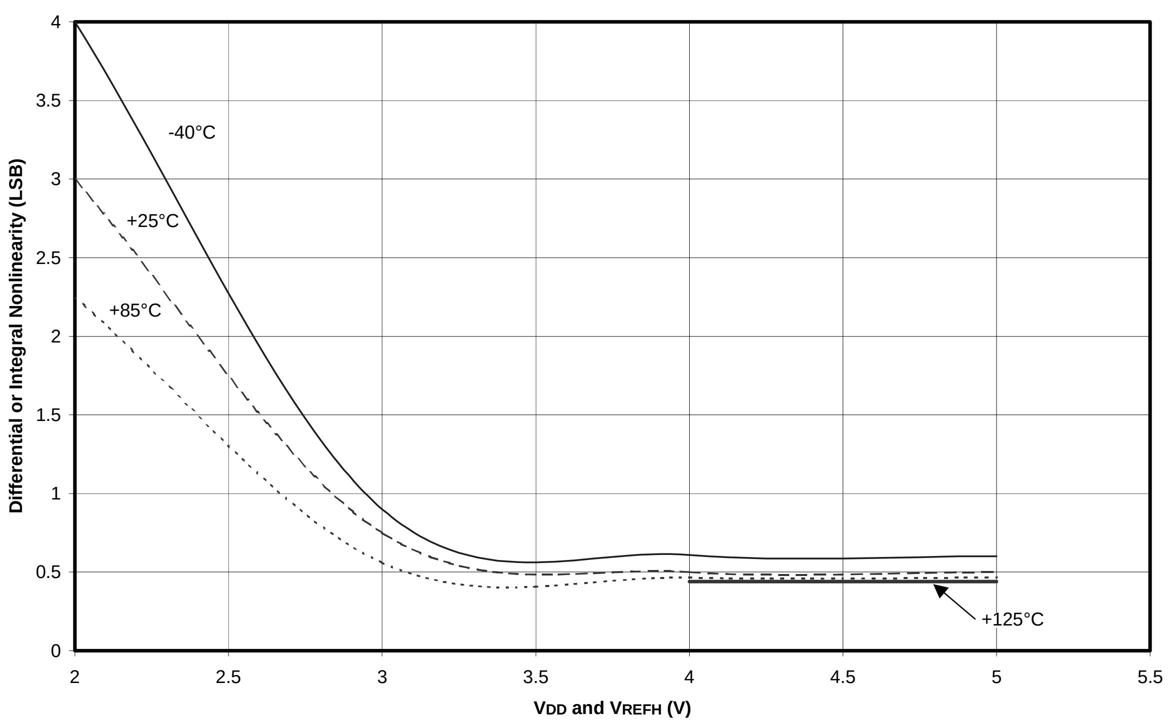
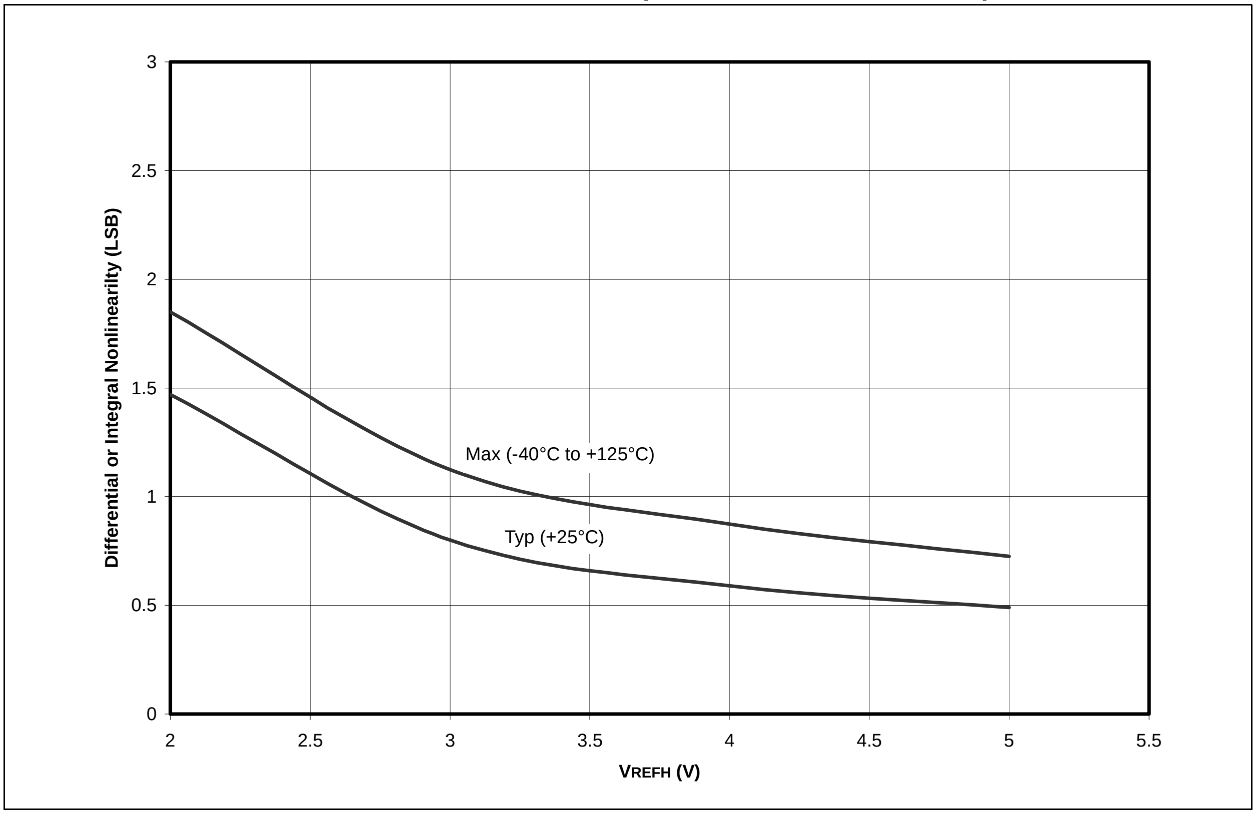


FIGURE 28-29: A/D NON-LINEARITY vs. VREFH (VDD = 5V, -40°C TO +125°C)

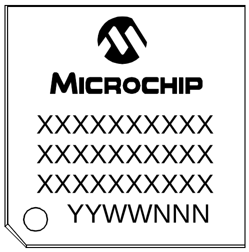


NOTES:

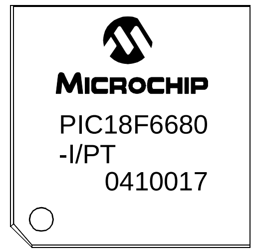
29.0 PACKAGING INFORMATION

29.1 Package Marking Information

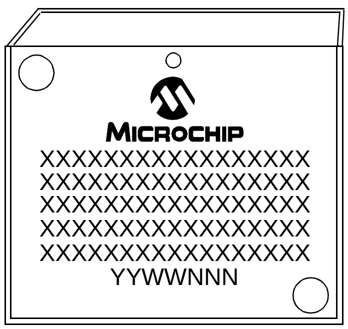
64-Lead TQFP



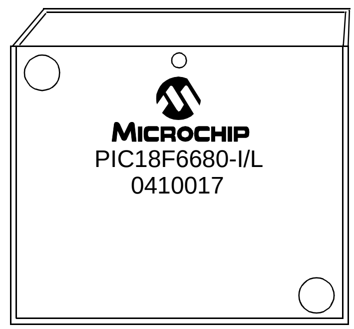
Example



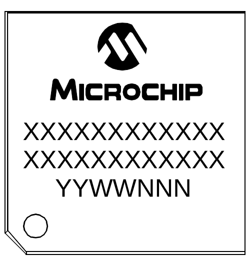
68-Lead PLCC



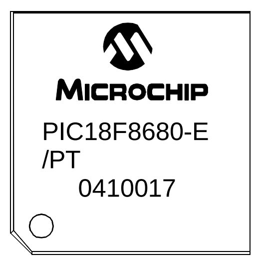
Example



80-Lead TQFP



Example



Legend:	XX...X	Customer specific information*
	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

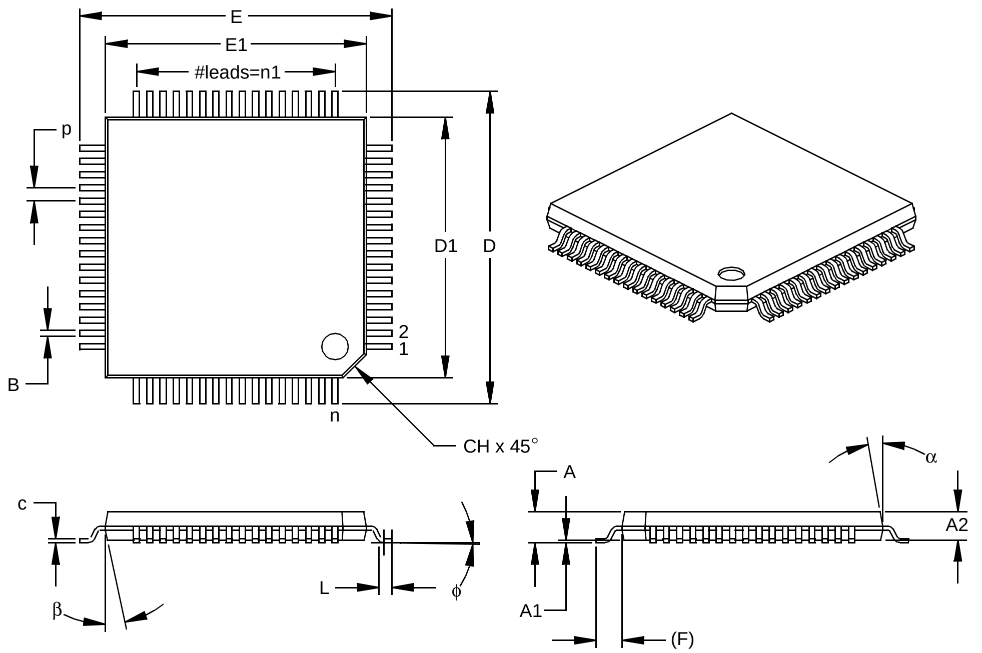
Note: In the event the full Microchip part number cannot be marked on one line, it will be carried over to the next line thus limiting the number of available characters for customer specific information.

* Standard PICmicro device marking consists of Microchip part number, year code, week code, and traceability code. For PICmicro device marking beyond this, certain price adders apply. Please check with your Microchip Sales Office. For QTP devices, any special marking adders are included in QTP price.

29.2 Package Details

The following sections give the technical details of the packages.

64-Lead Plastic Thin Quad Flatpack (PT) 10x10x1 mm Body, 1.0/0.10 mm Lead Form (TQFP)

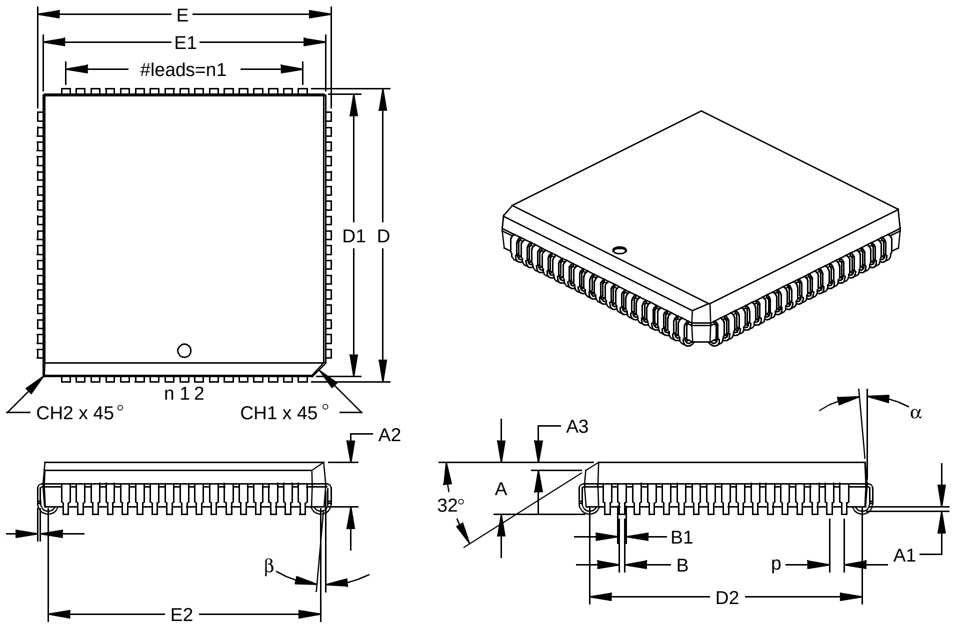


Units		INCHES			MILLIMETERS*		
Dimension Limits		MIN	NOM	MAX	MIN	NOM	MAX
Number of Pins	n		64			64	
Pitch	p		.020			0.50	
Pins per Side	n1		16			16	
Overall Height	A	.039	.043	.047	1.00	1.10	1.20
Molded Package Thickness	A2	.037	.039	.041	0.95	1.00	1.05
Standoff §	A1	.002	.006	.010	0.05	0.15	0.25
Foot Length	L	.018	.024	.030	0.45	0.60	0.75
Footprint (Reference)	(F)		.039			1.00	
Foot Angle	φ	0	3.5	7	0	3.5	7
Overall Width	E	.463	.472	.482	11.75	12.00	12.25
Overall Length	D	.463	.472	.482	11.75	12.00	12.25
Molded Package Width	E1	.390	.394	.398	9.90	10.00	10.10
Molded Package Length	D1	.390	.394	.398	9.90	10.00	10.10
Lead Thickness	c	.005	.007	.009	0.13	0.18	0.23
Lead Width	B	.007	.009	.011	0.17	0.22	0.27
Pin 1 Corner Chamfer	CH	.025	.035	.045	0.64	0.89	1.14
Mold Draft Angle Top	α	5	10	15	5	10	15
Mold Draft Angle Bottom	β	5	10	15	5	10	15

* Controlling Parameter
§ Significant Characteristic

Notes:
Dimensions D1 and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010” (0.254mm) per side.
JEDEC Equivalent: MS-026
Drawing No. C04-085

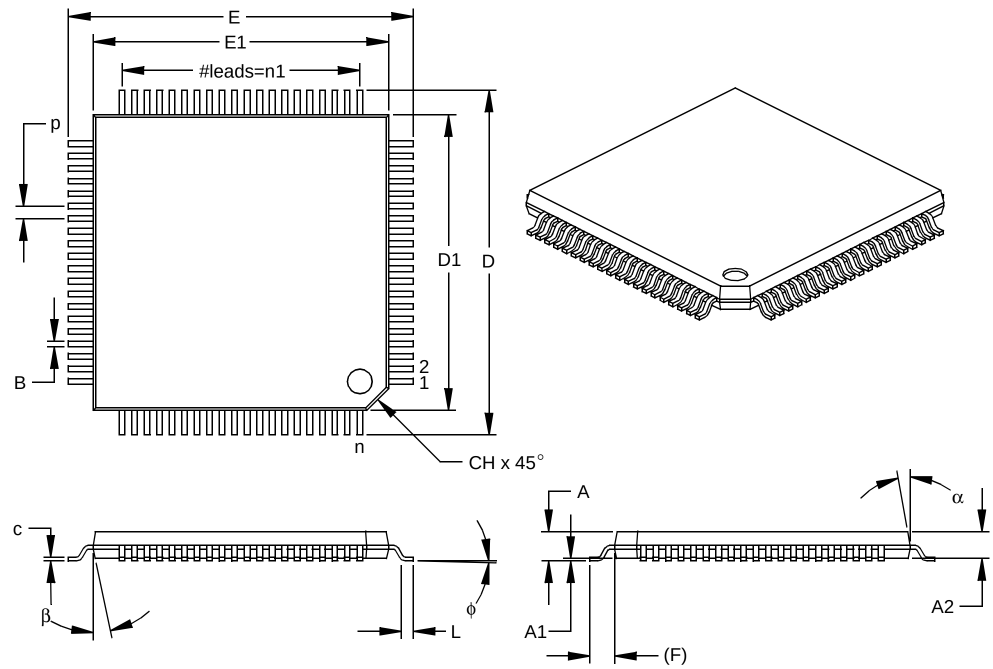
68-Lead Plastic Leaded Chip Carrier (L) – Square (PLCC)



Units		INCHES*			MILLIMETERS		
Dimension Limits		MIN	NOM	MAX	MIN	NOM	MAX
Number of Pins	n		68			68	
Pitch	p		.050			1.27	
Pins per Side	n1		17			17	
Overall Height	A	.165	.173	.180	4.19	4.39	4.57
Molded Package Thickness	A2	.145	.153	.160	3.68	3.87	4.06
Standoff §	A1	.020	.028	.035	0.51	0.71	0.89
Side 1 Chamfer Height	A3	.024	.029	.034	0.61	0.74	0.86
Corner Chamfer 1	CH1	.040	.045	.050	1.02	1.14	1.27
Corner Chamfer (others)	CH2	.000	.005	.010	0.00	0.13	0.25
Overall Width	E	.985	.990	.995	25.02	25.15	25.27
Overall Length	D	.985	.990	.995	25.02	25.15	25.27
Molded Package Width	E1	.950	.954	.958	24.13	24.23	24.33
Molded Package Length	D1	.950	.954	.958	24.13	24.23	24.33
Footprint Width	E2	.890	.920	.930	22.61	23.37	23.62
Footprint Length	D2	.890	.920	.930	22.61	23.37	23.62
Lead Thickness	c	.008	.011	.013	0.20	0.27	0.33
Upper Lead Width	B1	.026	.029	.032	0.66	0.74	0.81
Lower Lead Width	B	.013	.020	.021	0.33	0.51	0.53
Mold Draft Angle Top	α	0	5	10	0	5	10
Mold Draft Angle Bottom	β	0	5	10	0	5	10

* Controlling Parameter
§ Significant Characteristic
Notes:
Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010” (0.254mm) per side.
JEDEC Equivalent: MO-047
Drawing No. C04-049

80-Lead Plastic Thin Quad Flatpack (PT) 12x12x1 mm Body, 1.0/0.10 mm Lead Form (TQFP)



	Units	INCHES			MILLIMETERS*		
	Dimension Limits	MIN	NOM	MAX	MIN	NOM	MAX
Number of Pins	n		80			80	
Pitch	p		.020			0.50	
Pins per Side	n1		20			20	
Overall Height	A	.039	.043	.047	1.00	1.10	1.20
Molded Package Thickness	A2	.037	.039	.041	0.95	1.00	1.05
Standoff §	A1	.002	.004	.006	0.05	0.10	0.15
Foot Length	L	.018	.024	.030	0.45	0.60	0.75
Footprint (Reference)	(F)		.039			1.00	
Foot Angle	φ	0	3.5	7	0	3.5	7
Overall Width	E	.541	.551	.561	13.75	14.00	14.25
Overall Length	D	.541	.551	.561	13.75	14.00	14.25
Molded Package Width	E1	.463	.472	.482	11.75	12.00	12.25
Molded Package Length	D1	.463	.472	.482	11.75	12.00	12.25
Lead Thickness	c	.004	.006	.008	0.09	0.15	0.20
Lead Width	B	.007	.009	.011	0.17	0.22	0.27
Pin 1 Corner Chamfer	CH	.025	.035	.045	0.64	0.89	1.14
Mold Draft Angle Top	α	5	10	15	5	10	15
Mold Draft Angle Bottom	β	5	10	15	5	10	15

* Controlling Parameter
§ Significant Characteristic

Notes:
Dimensions D1 and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010" (0.254mm) per side.
JEDEC Equivalent: MS-026
Drawing No. C04-092

APPENDIX A: REVISION HISTORY

Revision A (February 2003)

Original data sheet for PIC18F6585/8585/6680/8680 family.

Revision B (June 2003)

This revision includes updates to the Special Function Registers in Table 4-2 and Table 23-1 and minor corrections to the data sheet text.

Revision C (February 2004)

This revision includes the DC and AC Characteristics Graphs and Tables. The Electrical Specifications in **Section 27.0 "Electrical Characteristics"** have been updated and there have been minor corrections to the data sheet text.

APPENDIX B: DEVICE DIFFERENCES

The differences between the devices listed in this data sheet are shown in Table B-1.

TABLE B-1: DEVICE DIFFERENCES

Feature	PIC18F6585	PIC18F6680	PIC18F8585	PIC18F8680
On-Chip Program Memory (Kbytes)	48	64	48	64
I/O Ports	Ports A, B, C, D, E, F, G	Ports A, B, C, D, E, F, G	Ports A, B, C, D, E, F, G, H, J	Ports A, B, C, D, E, F, G, H, J
A/D Channels	12	12	16	16
External Memory Interface	No	No	Yes	Yes
Package Types	64-pin TQFP, 68-pin PLCC	64-pin TQFP, 68-pin PLCC	80-pin TQFP	80-pin TQFP

APPENDIX C: CONVERSION CONSIDERATIONS

This appendix discusses the considerations for converting from previous versions of a device to the ones listed in this data sheet. Typically, these changes are due to the differences in the process technology used. An example of this type of conversion is from a PIC17C756 to a PIC18F8720.

Not Applicable

APPENDIX D: MIGRATION FROM MID-RANGE TO ENHANCED DEVICES

A detailed discussion of the differences between the mid-range MCU devices (i.e., PIC16CXXX) and the enhanced devices (i.e., PIC18FXXX) is provided in AN716, “Migrating Designs from PIC16C74A/74B to PIC18C442.” The changes discussed, while device specific, are generally applicable to all mid-range to enhanced device migrations.

This Application Note is available as Literature Number DS00716.

APPENDIX E: MIGRATION FROM HIGH-END TO ENHANCED DEVICES

A detailed discussion of the migration pathway and differences between the high-end MCU devices (i.e., PIC17CXXX) and the enhanced devices (i.e., PIC18FXXXX) is provided in AN726, “PIC17CXXX to PIC18CXXX Migration.” This Application Note is available as Literature Number DS00726.

NOTES:

INDEX

A

B

D

E

L

M

Q

R

U

V

W

X

NOTES:

ON-LINE SUPPORT

Microchip provides on-line support on the Microchip World Wide Web site.

The web site is used by Microchip as a means to make files and information easily available to customers. To view the site, the user must have access to the Internet and a web browser, such as Netscape® or Microsoft® Internet Explorer. Files are also available for FTP download from our FTP site.

Connecting to the Microchip Internet Web Site

The Microchip web site is available at the following URL:

www.microchip.com

The file transfer site is available by using an FTP service to connect to:

ftp://ftp.microchip.com

The web site and file transfer site provide a variety of services. Users may download files for the latest Development Tools, Data Sheets, Application Notes, User's Guides, Articles and Sample Programs. A variety of Microchip specific business information is also available, including listings of Microchip sales offices, distributors and factory representatives. Other data available for consideration is:

- Latest Microchip Press Releases
- Technical Support Section with Frequently Asked Questions
- Design Tips
- Device Errata
- Job Postings
- Microchip Consultant Program Member Listing
- Links to other useful web sites related to Microchip Products
- Conferences for products, Development Systems, technical information and more
- Listing of seminars and events

SYSTEMS INFORMATION AND UPGRADE HOT LINE

The Systems Information and Upgrade Line provides system users a listing of the latest versions of all of Microchip's development systems software products. Plus, this line provides information on how customers can receive the most current upgrade kits. The Hot Line Numbers are:

1-800-755-2345 for U.S. and most of Canada, and

1-480-792-7302 for the rest of the world.

READER RESPONSE

It is our intention to provide you with the best documentation possible to ensure successful use of your Microchip product. If you wish to provide your comments on organization, clarity, subject matter, and ways in which our documentation can better serve you, please FAX your comments to the Technical Publications Manager at (480) 792-4150.

Please list the following information, and use this outline to provide us with your comments about this document.

To: Technical Publications Manager Total Pages Sent ________

RE: Reader Response

From: Name ____

Company ____

Address ____

City / State / ZIP / Country ____

Telephone: (_______) _________ - _________ FAX: (______) _________ - _________

Application (optional):

Would you like a reply?____Y ____N

Device: PIC18F6585/8585/6680/8680 Literature Number: DS30491C

Questions:

1. What are the best features of this document?

____

____

2. How does this document meet your hardware and software development needs?

____

____

3. Do you find the organization of this document easy to follow? If not, why?

____

____

4. What additions to the document do you think would enhance the structure and subject?

____

____

5. What deletions from the document could be made without affecting the overall usefulness?

____

____

6. Is there any incorrect or misleading information (what and where)?

____

____

7. How would you improve this document?

____

____

PIC18F6585/8585/6680/8680 PRODUCT IDENTIFICATION SYSTEM

To order or obtain information, e.g., on pricing or delivery, refer to the factory or the listed sales office.

PART NO.	–	X	/XX	XXX
Device		Temperature Range	Package	Pattern

Device	PIC18FXX8X(1), PIC18FXX8XT(2); VDD range 4.2V to 5.5V PIC18LFXX8X(1), PIC18LFXX8XT(2); VDD range 2.0V to 5.5V
Temperature Range	I = -40°C to +85°C (Industrial) E = -40°C to +125°C (Extended)
Package	PT = TQFP (Thin Quad Flatpack)
Pattern	QTP, SQTP, Code or Special Requirements (blank otherwise)

Examples:

a) PIC18LF6680 - I/PT 301 = Industrial temp., TQFP package, Extended VDD limits, QTP pattern #301.

b) PIC18F8585 - I/PT = Industrial temp., TQFP package, normal VDD limits.

c) PIC18F8680 - E/PT = Extended temp., TQFP package, standard VDD limits.

Note 1: F = Standard Voltage Range
LF = Extended Voltage Range

2: T = in tape and reel

NOTES:

NOTES:

NOTES:

NOTES:

WORLDWIDE SALES AND SERVICE

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200
Fax: 480-792-7277
Technical Support: 480-792-7627
Web Address: http://www.microchip.com

Atlanta
3780 Mansell Road, Suite 130
Alpharetta, GA 30022
Tel: 770-640-0034
Fax: 770-640-0307

Boston
2 Lan Drive, Suite 120
Westford, MA 01886
Tel: 978-692-3848
Fax: 978-692-3821

Chicago
333 Pierce Road, Suite 180
Itasca, IL 60143
Tel: 630-285-0071
Fax: 630-285-0075

Dallas
4570 Westgrove Drive, Suite 160
Addison, TX 75001
Tel: 972-818-7423
Fax: 972-818-2924

Detroit
Tri-Atria Office Building
32255 Northwestern Highway, Suite 190
Farmington Hills, MI 48334
Tel: 248-538-2250
Fax: 248-538-2260

Kokomo
2767 S. Albright Road
Kokomo, IN 46902
Tel: 765-864-8360
Fax: 765-864-8387

Los Angeles
18201 Von Karman, Suite 1090
Irvine, CA 92612
Tel: 949-263-1888
Fax: 949-263-1338

San Jose
1300 Terra Bella Avenue
Mountain View, CA 94043
Tel: 650-215-1444
Fax: 650-961-0286

Toronto
6285 Northam Drive, Suite 108
Mississauga, Ontario L4V 1X5, Canada
Tel: 905-673-0699
Fax: 905-673-6509

ASIA/PACIFIC

Australia
Suite 22, 41 Rawson Street
Epping 2121, NSW
Australia
Tel: 61-2-9868-6733
Fax: 61-2-9868-6755

China - Beijing
Unit 706B
Wan Tai Bei Hai Bldg.
No. 6 Chaoyangmen Bei Str.
Beijing, 100027, China
Tel: 86-10-85282100
Fax: 86-10-85282104

China - Chengdu
Rm. 2401-2402, 24th Floor,
Ming Xing Financial Tower
No. 88 TIDU Street
Chengdu 610016, China
Tel: 86-28-86766200
Fax: 86-28-86766599

China - Fuzhou
Unit 28F, World Trade Plaza
No. 71 Wusi Road
Fuzhou 350001, China
Tel: 86-591-7503506
Fax: 86-591-7503521

China - Hong Kong SAR
Unit 901-6, Tower 2, Metroplaza
223 Hing Fong Road
Kwai Fong, N.T., Hong Kong
Tel: 852-2401-1200
Fax: 852-2401-3431

China - Shanghai
Room 701, Bldg. B
Far East International Plaza
No. 317 Xian Xia Road
Shanghai, 200051
Tel: 86-21-6275-5700
Fax: 86-21-6275-5060

China - Shenzhen
Rm. 1812, 18/F, Building A, United Plaza
No. 5022 Binhe Road, Futian District
Shenzhen 518033, China
Tel: 86-755-82901380
Fax: 86-755-8295-1393

China - Shunde
Room 401, Hongjian Building, No. 2
Fengxiangnan Road, Ronggui Town, Shunde
District, Foshan City, Guangdong 528303, China
Tel: 86-757-28395507 Fax: 86-757-28395571

China - Qingdao
Rm. B505A, Fullhope Plaza,
No. 12 Hong Kong Central Rd.
Qingdao 266071, China
Tel: 86-532-5027355 Fax: 86-532-5027205

India
Divyasree Chambers
1 Floor, Wing A (A3/A4)
No. 11, O'Shaugnessey Road
Bangalore, 560 025, India
Tel: 91-80-22290061 Fax: 91-80-22290062

Japan
Benex S-1 6F
3-18-20, Shinyokohama
Kohoku-Ku, Yokohama-shi
Kanagawa, 222-0033, Japan
Tel: 81-45-471- 6166 Fax: 81-45-471-6122

Korea
168-1, Youngbo Bldg. 3 Floor
Samsung-Dong, Kangnam-Ku
Seoul, Korea 135-882
Tel: 82-2-554-7200 Fax: 82-2-558-5932 or
82-2-558-5934

Singapore
200 Middle Road
#07-02 Prime Centre
Singapore, 188980
Tel: 65-6334-8870 Fax: 65-6334-8850

Taiwan
Kaohsiung Branch
30F - 1 No. 8
Min Chuan 2nd Road
Kaohsiung 806, Taiwan
Tel: 886-7-536-4818
Fax: 886-7-536-4803

Taiwan
Taiwan Branch
11F-3, No. 207
Tung Hua North Road
Taipei, 105, Taiwan
Tel: 886-2-2717-7175 Fax: 886-2-2545-0139

EUROPE

Austria
Durisolstrasse 2
A-4600 Wels
Austria
Tel: 43-7242-2244-399
Fax: 43-7242-2244-393

Denmark
Regus Business Centre
Lautrup hoj 1-3
Ballerup DK-2750 Denmark
Tel: 45-4420-9895 Fax: 45-4420-9910

France
Parc d'Activite du Moulin de Massy
43 Rue du Saule Trapu
Batiment A - ler Etage
91300 Massy, France
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

Germany
Steinheilstrasse 10
D-85737 Ismaning, Germany
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

Italy
Via Quasimodo, 12
20025 Legnano (MI)
Milan, Italy
Tel: 39-0331-742611
Fax: 39-0331-466781

Netherlands
P. A. De Biesbosch 14
NL-5152 SC Drunen, Netherlands
Tel: 31-416-690399
Fax: 31-416-690340

United Kingdom
505 Eskdale Road
Winnersh Triangle
Wokingham
Berkshire, England RG41 5TU
Tel: 44-118-921-5869
Fax: 44-118-921-5820

02/17/04