Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Active
Core Processor	PIC
Core Size	8-Bit
Speed	64MHz
Connectivity	I²C, LINbus, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, DMA, HLVD, POR, PWM, WDT
Number of I/O	25
Program Memory Size	128KB (64K x 16)
Program Memory Type	FLASH
EEPROM Size	1K x 8
RAM Size	8K x 8
Voltage - Supply (Vcc/Vdd)	2.3V ~ 5.5V
Data Converters	A/D 24x12b; D/A 1x5b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Surface Mount
Package / Case	28-SSOP (0.209", 5.30mm Width)
Supplier Device Package	28-SSOP
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18f27k42-e-ss

12.0 8x8 HARDWARE MULTIPLIER

12.1 Introduction

All PIC18 devices include an 8x8 hardware multiplier as part of the ALU. The multiplier performs an unsigned operation and yields a 16-bit result that is stored in the product register pair, PRODH:PRODL. The multiplier's operation does not affect any flags in the STATUS register.

Making multiplication a hardware operation allows it to be completed in a single instruction cycle. This has the advantages of higher computational throughput and reduced code size for multiplication algorithms and allows the PIC18 devices to be used in many applications previously reserved for digital signal processors. A comparison of various hardware and software multiply operations, along with the savings in memory and execution time, is shown in Table 12-1.

12.2 Operation

Example 12-1 shows the instruction sequence for an 8x8 unsigned multiplication. Only one instruction is required when one of the arguments is already loaded in the WREG register.

Example 12-2 shows the sequence to do an 8x8 signed multiplication. To account for the sign bits of the arguments, each argument's Most Significant bit (MSb) is tested and the appropriate subtractions are done.

EXAMPLE 12-1: 8x8 UNSIGNED MULTIPLY ROUTINE

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

EXAMPLE 12-2: 8x8 SIGNED MULTIPLY ROUTINE

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

TABLE 12-1: PERFORMANCE COMPARISON FOR VARIOUS MULTIPLY OPERATIONS

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

TABLE 15-6: EXAMPLE DMA USE CASE TABLE

Source Module	Source Register(s)	Destination Module	Destination Register(s)	DCHxSIRQ	Comment
Signal Measurement Timer (SMT)	SMTxCPW[U:H:L]	GPR	GPR[x,y,z]	SMTxPWAIF	Store Captured Pulse-width values
	SMTxCPR[U:H:L]			SMTxPRAIF	Store Captured Period values
GPR/SFR/Program Flash/Data EEPROM	MEMORY[x,y]	TMR0	TMR0[H:L]	TMR0IF	Use as a Timer0 reload for custom 16-bit value
GPR/SFR/ Program Flash/Data EEPROM	MEMORY[x]	TMR0	PR0	ANY	Update TMR0 frequency based on a specific trigger
GPR/SFR/ Program Flash/Data EEPROM	MEMORY[x,y]	TMR1	TMR1[H:L]	TMR1IF	Use as a Timer1 reload for custom 16-bit value
TMR1	TMR1[H:L]	GPR	GPR[x,y]	TMR1GIF	Use TMR1 Gate interrupt flag to read data out of TMR1 register
GPR/SFR/ Program Flash/Data EEPROM	MEMORY[x]	TMR2	PR2	TMR2IF	
GPR/SFR/ Program Flash/Data EEPROM	MEMORY[x,y,z]	TMR2 CCP or PWM	PR2 CCPR[H:L] or PWMDC[H:L]	ANY	Frequency generator with 50% duty cycle look up table
CCP	CCPR[H:L]	GPR	GPR[x,y]	CCPxIF	Move data from CCP 16b Capture
GPR/SFR/ Program Flash/Data EEPROM	MEMORY[x,y]	CCP	CCPR[H:L]	ANY	Load Compare value or PWM values into the CCP
GPR/SFR/ Program Flash/Data EEPROM	MEMORY [x,y,z,u,v,w]	CCPx CCPy CCPz	CCPxR[H:L] CCPyR[H:L] CCPzR[H:L]	ANY	Update multiple PWM values at the same time e.g. 3-phase motor control
GPR/SFR/ Program Flash/Data EEPROM	MEMORY[x,y,z]	NCO	NCOxINC[U:H:L]	ANY	Frequency Generator look-up table
GPR/SFR/ Program Flash/Data EEPROM	MEMORY[x]	DAC	DACxCON0	ANY	Update DAC values
GPR/SFR/ Program Flash/Data EEPROM	MEMORY[x]	OSCTUNE	OSCTUNE	ANY	Automated Frequency dithering

21.6.2 TIMER1/3/5 GATE SOURCE SELECTION

The gate source for Timer1/3/5 can be selected using the GSS<4:0> bits of the TMRxGATE register (Register 21-4). The polarity selection for the gate source is controlled by the TxGPOL bit of the TxGCON register (Register 21-2).

Any of the above mentioned signals can be used to trigger the gate. The output of the CMPx can be synchronized to the Timer1/3/5 clock or left asynchronous. For more information see **Section 38.3.1 "Comparator Output Synchronization"**.

21.6.3 TIMER1/3/5 GATE TOGGLE MODE

When Timer1/3/5 Gate Toggle mode is enabled, it is possible to measure the duration between every rising and falling edge of the gate signal.

The Timer1/3/5 gate source is routed through a flip-flop that changes state on every incrementing edge of the signal. See Figure 21-5 for timing details.

Timer1/3/5 Gate Toggle mode is enabled by setting the GTM bit of the TxGCON register. When the GTM bit is cleared, the flip-flop is cleared and held clear. This is necessary in order to control which edge is measured.

Note:	Enabling Toggle mode at the same time as changing the gate polarity may result in indeterminate operation.

21.6.4 TIMER1/3/5 GATE SINGLE-PULSE MODE

When Timer1/3/5 Gate Single-Pulse mode is enabled, it is possible to capture a single-pulse gate event. Timer1/3/5 Gate Single-Pulse mode is first enabled by setting the GSPM bit in the TxGCON register. Next, the GGO/DONE bit in the TxGCON register must be set. The Timer1/3/5 will be fully enabled on the next incrementing edge of the gate signal. On the next trailing edge of the pulse, the GGO/DONE bit will automatically be cleared. No other gate events will be allowed to increment Timer1/3/5 until the GGO/DONE bit is once again set in software.

Clearing the TxGSPM bit of the TxGCON register will also clear the GGO/DONE bit. See Figure 21-6 for timing details.

Enabling the Toggle mode and the Single-Pulse mode simultaneously will permit both sections to work together. This allows the period on the Timer1/3/5 gate source to be measured. See Figure 21-7 for timing details.

21.6.5 TIMER1/3/5 GATE VALUE STATUS

When Timer1/3/5 Gate Value Status is utilized, it is possible to read the most current level of the gate signal. The value is stored in the GVAL bit in the TxGCON register. The GVAL bit is valid even when the Timer1/3/5 gate is not enabled (GE bit is cleared).

21.6.6 TIMER1/3/5 GATE EVENT INTERRUPT

When Timer1/3/5 Gate Event Interrupt is enabled, it is possible to generate an interrupt upon the completion of a gate event. When the falling edge of GVAL occurs, the TMRxGIF flag bit in the respective PIR register will be set. If the TMRxGIE bit in the respective PIE register is set, then an interrupt will be recognized.

The TMRxGIF flag bit operates even when the Timer1/3/5 gate is not enabled (GE bit is cleared).

For more information on selecting high or low priority status for the Timer1/3/5 Gate Event Interrupt see **Section 9.0 "Interrupt Controller"**.

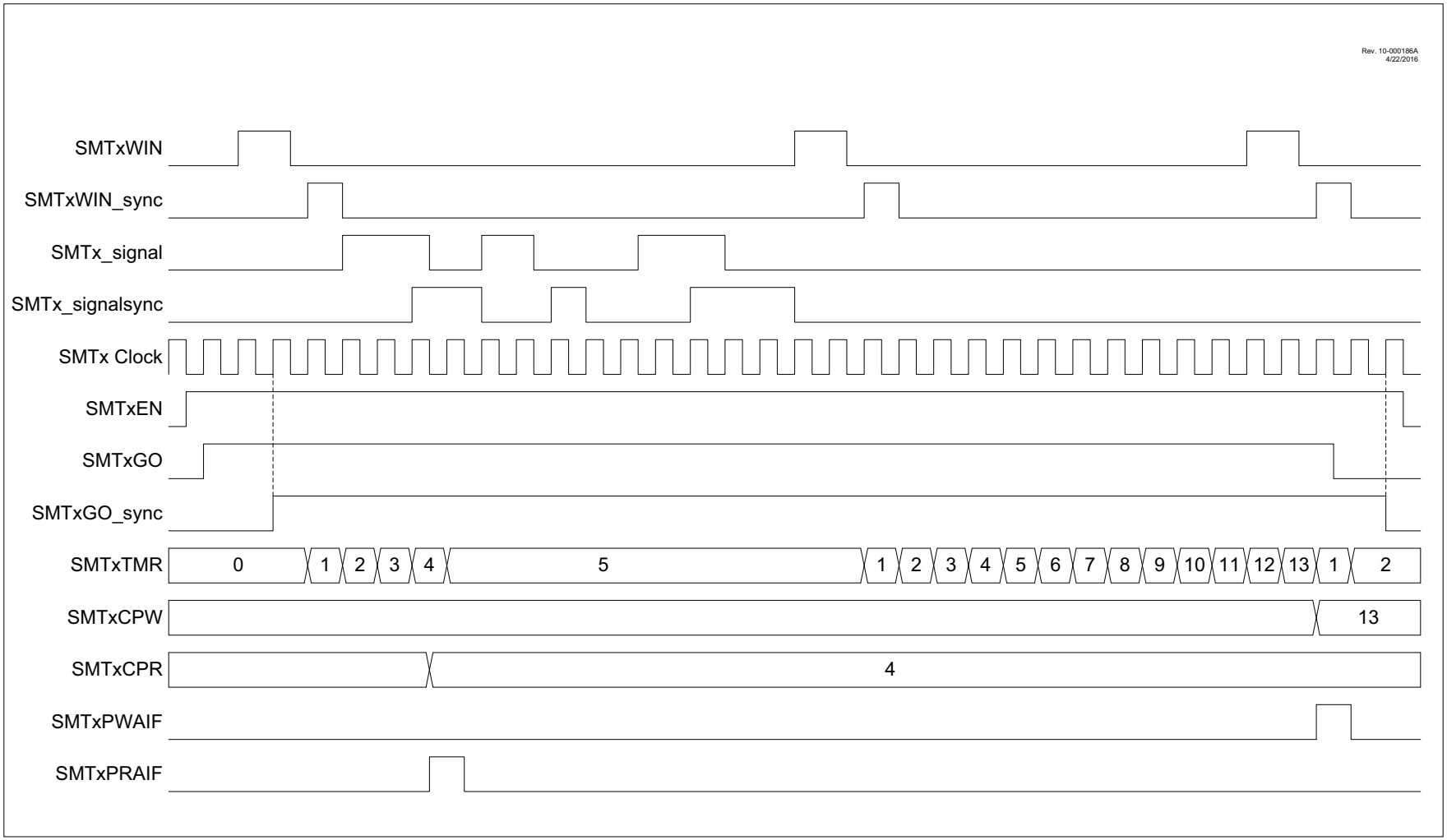
TABLE 22-3: SUMMARY OF REGISTERS ASSOCIATED WITH TIMER2

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
TxPR	Timer2 Module Period Register								320*
TxTMR	Holding Register for the 8-bit T2TMR Register								320*
TxCON	ON	CKPS<2:0>			OUTPS<3:0>				338
TxCLK	—	—	—	—	—	CS<2:0>			335
TxRST	—	—	—	—	RSEL<3:0>				336
TxHLT	$\overline{\text{PSYNC}}$	CPOL	$\overline{\text{CSYNC}}$	MODE<4:0>					339

Legend: — = unimplemented location, read as '0'. Shaded cells are not used for Timer2 module.

* Page provides register information.

FIGURE 25-14: TIME OF FLIGHT MODE REPEAT ACQUISITION TIMING DIAGRAM



REGISTER 26-2: CWGxCON1: CWG CONTROL REGISTER 1

U-0	U-0	R-x	U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	IN	—	POLD	POLC	POLB	POLA
bit 7							bit 0

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

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

bit 5 **IN:** CWG Input Value bit (read-only)

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

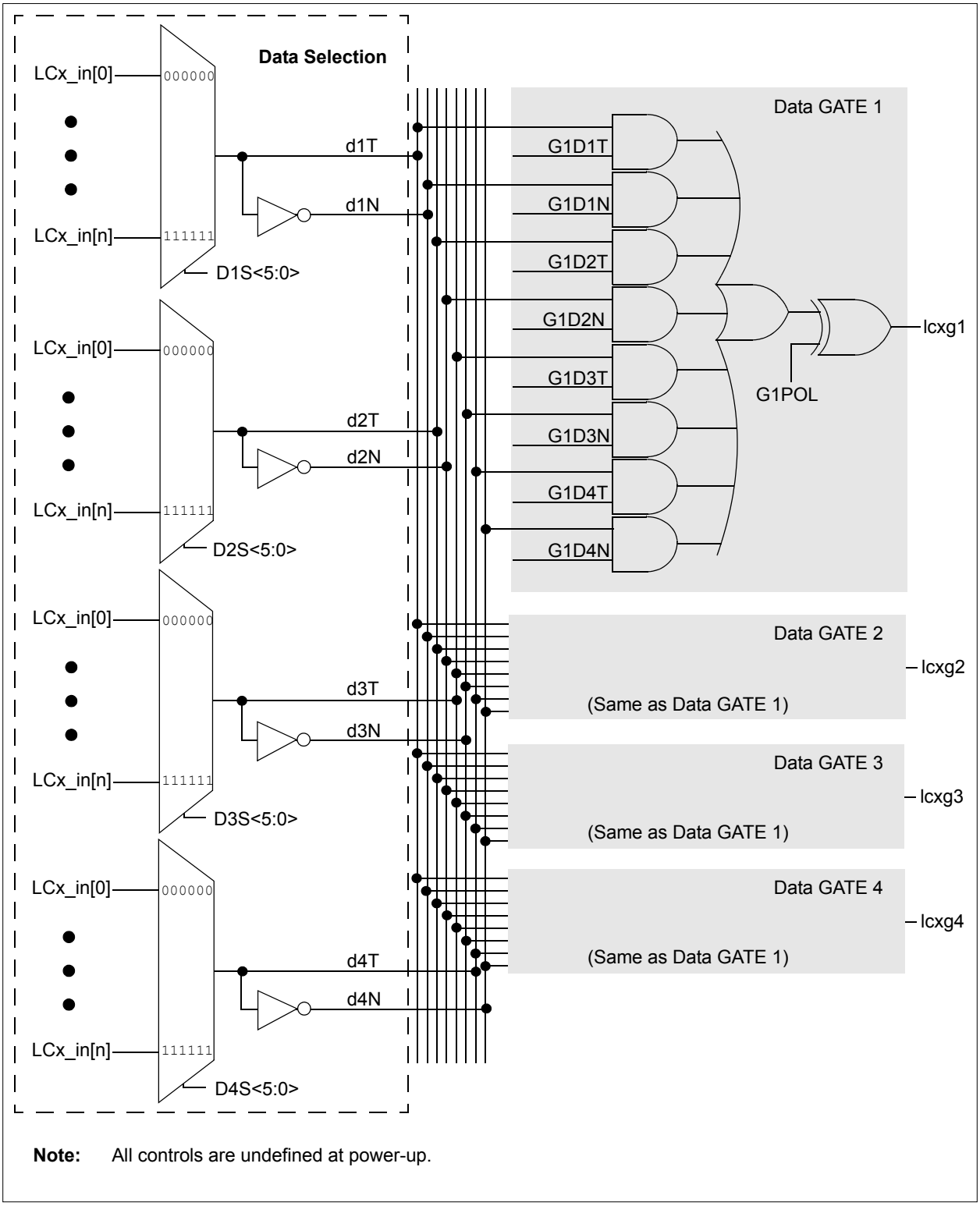
bit 3 **POLD:** CWGxD Output Polarity bit
1 = Signal output is inverted polarity
0 = Signal output is normal polarity

bit 2 **POLC:** CWGxC Output Polarity bit
1 = Signal output is inverted polarity
0 = Signal output is normal polarity

bit 1 **POLB:** CWGxB Output Polarity bit
1 = Signal output is inverted polarity
0 = Signal output is normal polarity

bit 0 **POLA:** CWGxA Output Polarity bit
1 = Signal output is inverted polarity
0 = Signal output is normal polarity

FIGURE 27-2: INPUT DATA SELECTION AND GATING



Note: All controls are undefined at power-up.

REGISTER 27-11: CLCDATA: CLC DATA OUTPUT

U-0	U-0	U-0	U-0	R-0	R-0	R-0	R-0
—	—	—	—	CLC4OUT	CLC3OUT	CLC2OUT	CLC1OUT
bit 7							bit 0

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

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

bit 3 **CLC4OUT:** Mirror copy of OUT bit of CLC4CON register

bit 2 **CLC3OUT:** Mirror copy of OUT bit of CLC3CON register

bit 1 **CLC2OUT:** Mirror copy of OUT bit of CLC2CON register

bit 0 **CLC1OUT:** Mirror copy of OUT bit of CLC1CON register

TABLE 27-3: SUMMARY OF REGISTERS ASSOCIATED WITH CLCx

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
CLCxCON	EN	—	OUT	INTP	INTN	MODE<2:0>			440
CLCxPOL	POL	—	—	—	G4POL	G3POL	G2POL	G1POL	441
CLCxSEL0	—	—	D1S<5:0>						442
CLCxSEL1	—	—	D2S<5:0>						442
CLCxSEL2	—	—	D3S<5:0>						442
CLCxSEL3	—	—	D4S<5:0>						442
CLCxGLS0	G1D4T	G1D4N	G1D3T	G1D3N	G1D2T	G1D2N	G1D1T	G1D1N	443
CLCxGLS1	G2D4T	G2D4N	G2D3T	G2D3N	G2D2T	G2D2N	G2D1T	G2D1N	444
CLCxGLS2	G3D4T	G3D4N	G3D3T	G3D3N	G3D2T	G3D2N	G3D1T	G3D1N	445
CLCxGLS3	G4D4T	G4D4N	G4D3T	G4D3N	G4D2T	G4D2N	G4D1T	G4D1N	446
CLCDATA	—	—	—	—	CLC4OUT	CLC3OUT	CLC2OUT	CLC1OUT	447

Legend: — = unimplemented, read as '0'. Shaded cells are unused by the CLCx modules.

REGISTER 28-3: NCO1ACCL: NCO1 ACCUMULATOR REGISTER – LOW BYTE

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
ACC<7:0>							
bit 7							bit 0

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

bit 7-0 **ACC<7:0>:** NCO1 Accumulator, Low Byte

REGISTER 28-4: NCO1ACCH: NCO1 ACCUMULATOR REGISTER – HIGH BYTE

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
ACC<15:8>							
bit 7							bit 0

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

bit 7-0 **ACC<15:8>:** NCO1 Accumulator, High Byte

31.4 DMX Mode (UART1 only)

DMX is a protocol used in stage and show equipment. This includes lighting, fog machines, motors, etc. The protocol consists of a controller that sends out commands, and receiver such as theater lights that receive these commands. DMX protocol is usually unidirectional, but can be a bidirectional protocol in either Half or Full-duplex modes. An example of Half-duplex mode is the RDM (Remote Device Management) protocol that sits on DMX512A. The controller transmits commands and the receiver receives them. Also there are no error conditions or re-transmit mechanisms.

DMX, or DMX512A as it is known, consists of a “Universe” of 512 channels. This means that one controller can output up to 512 bytes on a single DMX link. Each equipment on the line is programmed to listen to a consecutive sequence of one or more of these bytes.

For example, a fog machine connected to one of the universes may be programmed to receive one byte, starting at byte number 10, and a lighting unit may be programmed to receive four bytes starting at byte number 22.

31.4.1 DMX CONTROLLER

DMX Controller mode is configured with the following settings:

- MODE<3:0> = 1010
- TXEN = 1
- RXEN = 0
- TXPOL = 0
- UxP1 = One less than the number of bytes to transmit (excluding the Start code)
- UxBRGH:L = Value to achieve 250K baud rate
- STP<1:0> = 10 for 2 Stop bits
- RxyPPS = TX pin output code
- ON = 1

Each DMX transmission begins with a Break followed by a byte called the ‘Start Code’. The width of the BREAK is fixed at 25 bit times. The Break is followed by a “Mark After Break” (MAB) Idle period. After this Idle period, the 1st through ‘n’th byte is transmitted, where ‘n-1’ is the value in UxP1. See Figure 31-6.

Software sends the Start Code and the ‘n’ data bytes by writing the UxTXB register with each byte to be sent in the desired order. A UxTXIF value of ‘1’ indicates when the UxTXB is ready to accept the next byte.

The internal byte counter is not accessible to software. Software needs to keep track of the number of bytes written to UxTXB to ensure that no more and no less than ‘n’ bytes are sent because the DMX state machine will automatically insert a Break and reset its internal counter after ‘n’ bytes are written. One way to ensure synchronization between hardware and software is to toggle TXEN after the last byte of the universe is completely free of the transmit shift register as indicated by the TXMTIF bit.

31.4.2 DMX RECEIVER

DMX Receiver mode is configured with the following settings:

- MODE<3:0> = 1010
- TXEN = 0
- RXEN = 1
- RXPOL = 0
- UxP2 = number of first byte to receive
- UxP3 = number of last byte to receive
- UxBRGH:L = Value to achieve 250K baud rate
- STP<1:0> = 10 for 2 Stop bits
- ON = 1
- UxRXPPS = code for desired input pin
- Input pin ANSEL bit = 0

When configured as DMX Receiver, the UART listens for a Break character that is at least 23 bit periods wide. If the Break is shorter than 23 bit times, the Break is ignored and the DMX state machine remains in Idle mode. Upon receiving the Break, the DMX counters will be reset to align with the incoming data stream. Immediately after the Break, the UART will see the “Mark after Break” (MAB). This space is ignored by the UART. The Start Code follows the MAB and will always be stored in the receive FIFO.

After the Start Code, the 1st through 512th byte will be received, but not all of them are stored in the receive FIFO. The UART ignores all received bytes until the ones of interest are received. This is done using the UxP2 and UxP3 registers. The UxP2 register holds the value of the byte number to start the receive process. The byte counter starts at 0 for the first byte after the Start Code. For example, to receive four bytes starting at the 10th byte after the Start Code, write 009h (9 decimal) to UxP2H:L and 00Ch (12 decimal) to UxP3H:L. The receive FIFO is only 2 bytes deep, therefore the bytes must be retrieved by reading UxRXB as they come in to avoid a receive FIFO overrun condition.

Typically two Stop bits are inserted between bytes. If either Stop bit is detected as a ‘0’ then the framing error for that byte will be set.

Since the DMX sequence always starts with a Break, the software can verify that it is in sync with the sequence by monitoring the RXBKIF flag to ensure that the next byte received after the RXBKIF is processed as the Start Code and subsequent bytes are processed as the expected data.

REGISTER 31-5: UxERRIE: UART ERROR INTERRUPT ENABLE REGISTER

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
TXMTIE	PERIE	ABDOVE	CERIE	FERIE	RXBKIE	RXFOIE	TXCIE
bit 7							bit 0

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

bit 7 **TXMTIE:** Transmit Shift Register Empty Interrupt Enable bit
1 = Interrupt enabled
0 = Interrupt not enabled

bit 6 **PERIE**: Parity Error Interrupt Enable bit
1 = Interrupt enabled
0 = Interrupt not enabled

bit 5 **ABDOVE:** Auto-baud Detect Overflow Interrupt Enable bit
1 = Interrupt enabled
0 = Interrupt not enabled

bit 4 **CERIE**: Checksum Error Interrupt Enable bit
1 = Interrupt enabled
0 = Interrupt not enabled

bit 3 **FERIE:** Framing Error Interrupt Enable bit
1 = Interrupt enabled
0 = Interrupt not enabled

bit 2 **RXBKIE:** Break Reception Interrupt Enable bit
1 = Interrupt enabled
0 = Interrupt not enabled

bit 1 **RXFOIE:** Receive FIFO Overflow Interrupt Enable bit
1 = Interrupt enabled
0 = Interrupt not enabled

bit 0 **TXCIE:** Transmit Collision Interrupt Enable bit
1 = Interrupt enabled
0 = Interrupt not enabled

The SPI transmit output (SDO_out) is available to the remappable PPS SDO pin and internally to the following peripherals:

- Configurable Logic Cell (CLC)
- Data Signal Modulator (DSM)

The SPI bus typically operates with a single master device and one or more slave devices. When multiple slave devices are used, an independent Slave Select connection is required from the master device to each slave device.

The master selects only one slave at a time. Most slave devices have tri-state outputs so their output signal appears disconnected from the bus when they are not selected.

Transmissions typically involve shift registers, eight bits in size, one in the master and one in the slave. With either the master or the slave device, data is always shifted out one bit at a time, with the Most Significant bit (MSb) shifted out first. At the same time, a new bit is shifted into the device. Unlike older Microchip devices, the SPI on the PIC18(L)F2X/4X/5XK42 contains two separate registers for incoming and outgoing data. Both registers also have 2-byte FIFO buffers and allow for DMA bus connections.

Figure 32-2 shows a typical connection between two PIC18F2X/4X/5XK42 devices configured as master and slave devices.

Data is shifted out of the transmit FIFO on the programmed clock edge and into the receive shift register on the opposite edge of the clock.

The master device transmits information on its SDO output pin which is connected to, and received by, the slave's SDI input pin. The slave device transmits information on its SDO output pin, which is connected to, and received by, the master's SDI input pin.

The master device sends out the clock signal. Both the master and the slave devices should be configured for the same clock polarity.

During each SPI clock cycle, a full-duplex data transmission occurs. This means that while the master device is sending out the MSb from its output register (on its SDO pin) and the slave device is reading this bit and saving as the LSb of its input register, that the slave device is also sending out the MSb from its shift register (on its SDO pin) and the master device is reading this bit and saving it as the LSb of its input register.

After eight bits have been shifted out, the master and slave have exchanged register values and stored the incoming data into the receiver FIFOs.

If there is more data to exchange, the registers are loaded with new data and the process repeats itself.

Whether the data is meaningful or not (dummy data) depends on the application software. This leads to three scenarios for data transmission:

- Master sends useful data and slave sends dummy data
- Master sends useful data and slave sends useful data
- Master sends dummy data and slave sends useful data

In this particular SPI module, dummy data may be sent without software involvement, by clearing either the RXR bit (for receiving dummy data) or the TXR bit (for sending dummy data) (see Table 32-1 as well as **Section 32.5 "Master mode"** and **Section 32.6 "Slave Mode"** for further TXR/RXR setting details). This SPI module can send transmissions of any number of bits, and can send information in segments of varying size (from 1-8 bits in width). As such, transmissions may involve any number of clock cycles, depending on the amount of data to be transmitted.

When there is no more data to be transmitted, the master stops sending the clock signal and deselects the slave.

Every slave device connected to the bus that has not been selected through its slave select line disregards the clock and transmission signals and does not transmit out any data of its own.

33.4.3 SLAVE OPERATION IN 7-BIT ADDRESSING MODE

The 8th bit in an address byte transmitted by the master is used to determine if the Master wants to read from or write to the Slave device. If set, it denotes that the Master wants to read from the slave and if cleared it means the master wants to write to the slave device. If there is an address match, the R/W bit is copied to the R/W bit of the I2CxSTAT0 register.

33.4.3.1 Slave Reception (7-bit Addressing Mode)

This section describes the sequence of events for the I^2C module configured as an I^2C slave in 7-bit Addressing mode and is receiving data. Figure 33-6, Figure 33-7, and Figure 33-8 are used as a visual reference for this description.

1. Master asserts Start condition (can also be a restart) on the bus. Start condition Interrupt Flag (SCIF) in I2CxPIR register is set.
2. If Start condition interrupt is enabled (SCIE bit is set), generic interrupt I2CxIF is set.
3. Master transmits eight bits – 7-bit address and R/W = 0.
4. Received address is compared with the values in I2CxADR0/I2CxADR1/I2CxADR2/I2CxADR3 registers. Refer to section **Section 33.4.1 "Slave Addressing Modes"** for slave addressing modes.
5. If address matches; SMA in I2CxSTAT0 register is set, R/W is copied to R/W bit, D/A bit is cleared. If the address does not match; module becomes idle.
6. The matched address data is loaded into I2CxADB0 and ADRIF in I2CxPIR register is set.
7. If Address hold interrupt is enabled (ADRIE = 1), CSTR is set. I2CxIF is set. Slave software can read address from I2CxADB0 and set/clear ACKDT before releasing SCL.
8. If there are any previous error conditions, e.g., Receive buffer overflow or transmit buffer underflow errors, Slave will force a NACK and the module becomes idle.
9. ACKDT value is copied out to SDA for ACK pulse to be read by the Master on the 9th SCL pulse.
10. If the Acknowledge interrupt and hold is enabled (ACKTIE = 1), CSTR is set, I2CxIF is set, then Slave software can read address from I2CxADB0 register and change the value of ACKDT before releasing SCL by clearing CSTR.
11. Master sends first seven SCL pulses of the data byte or a Stop condition (in the case of NACK).
12. If Stop condition; PCIF in I2CxPIR register is set, module becomes idle.
13. If the receive buffer is full from the previous transaction i.e. RXBF = 1 (I2CxRXIF = 1), CSTR is set. Slave software must read data out of I2CxRXB to resume communication.
14. Master sends 8th SCL pulse of the data byte. D/A bit is set, WRIF is set.
15. I2CxRXB is loaded with new data, RXBF bit is set, I2CxRXIF is set.
16. If Data write interrupt and hold is enabled (WRIE = 1), CSTR is set, I2CxIF is set. Slave software can read data from I2CxRXB and set/clear ACKDT before releasing SCL by clearing CSTR.
17. If I2CxCNT = 0, the ACKCNT value is output to the SDA; else, if I2CxCNT!= 0, the ACKDT value is used and the value of I2CxCNT is decremented.
18. The ACK value is copied out to SDA to be read by the Master on the 9th SCL pulse.
19. If I2CxCNT = 0, CNTIF is set.
20. If a NACK was sent, NACKIF is set, module becomes idle.
21. If ACKTIE = 1, CSTR is set, I2CxIF is set. Slave software can read data from I2CxRXB clearing RXBF, before releasing SCL by clearing CSTR.
22. Go to step 11.

FIGURE 33-19: I²C MASTER, 7-BIT ADDRESS, TRANSMISSION WITH STOP

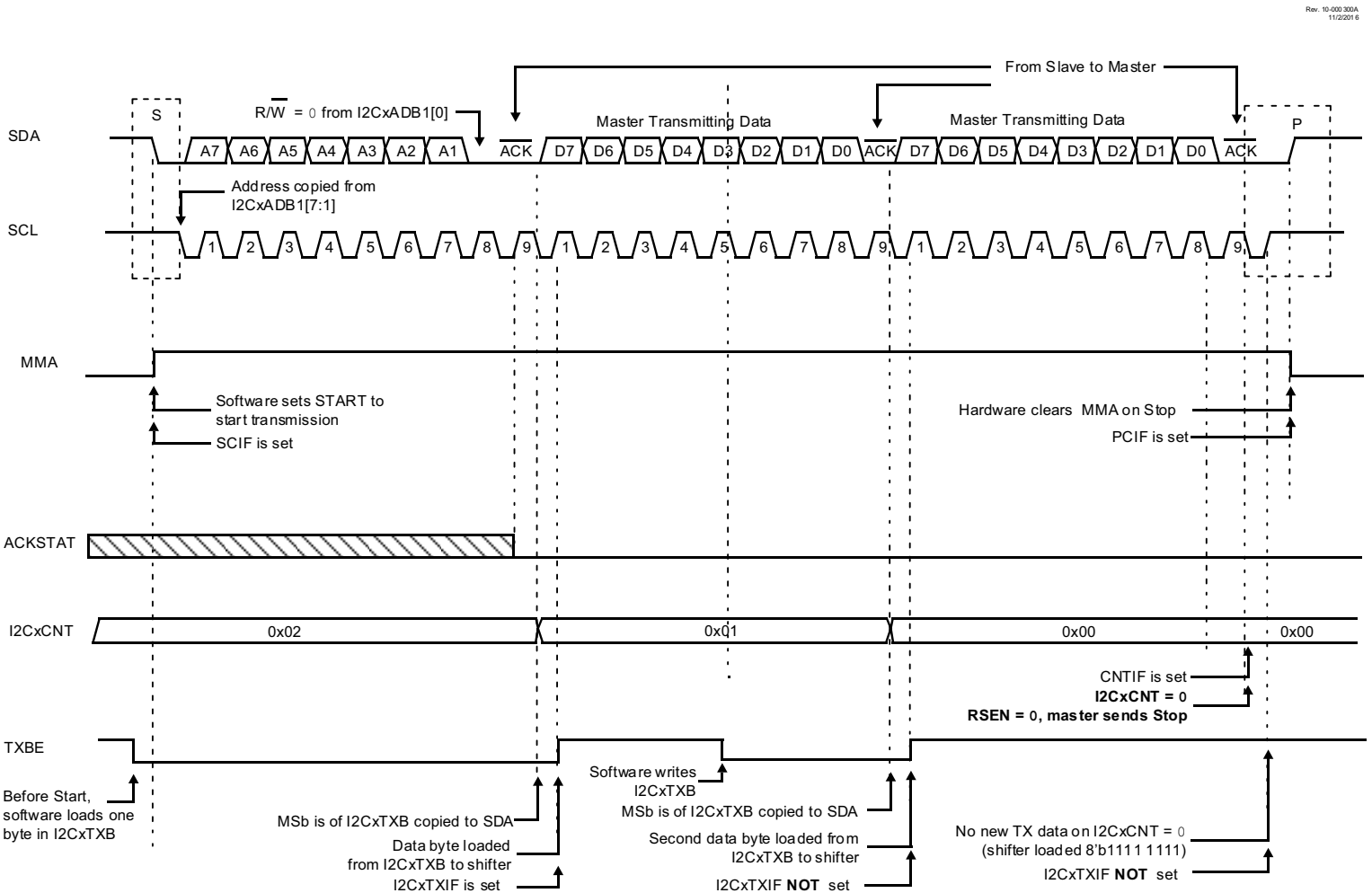
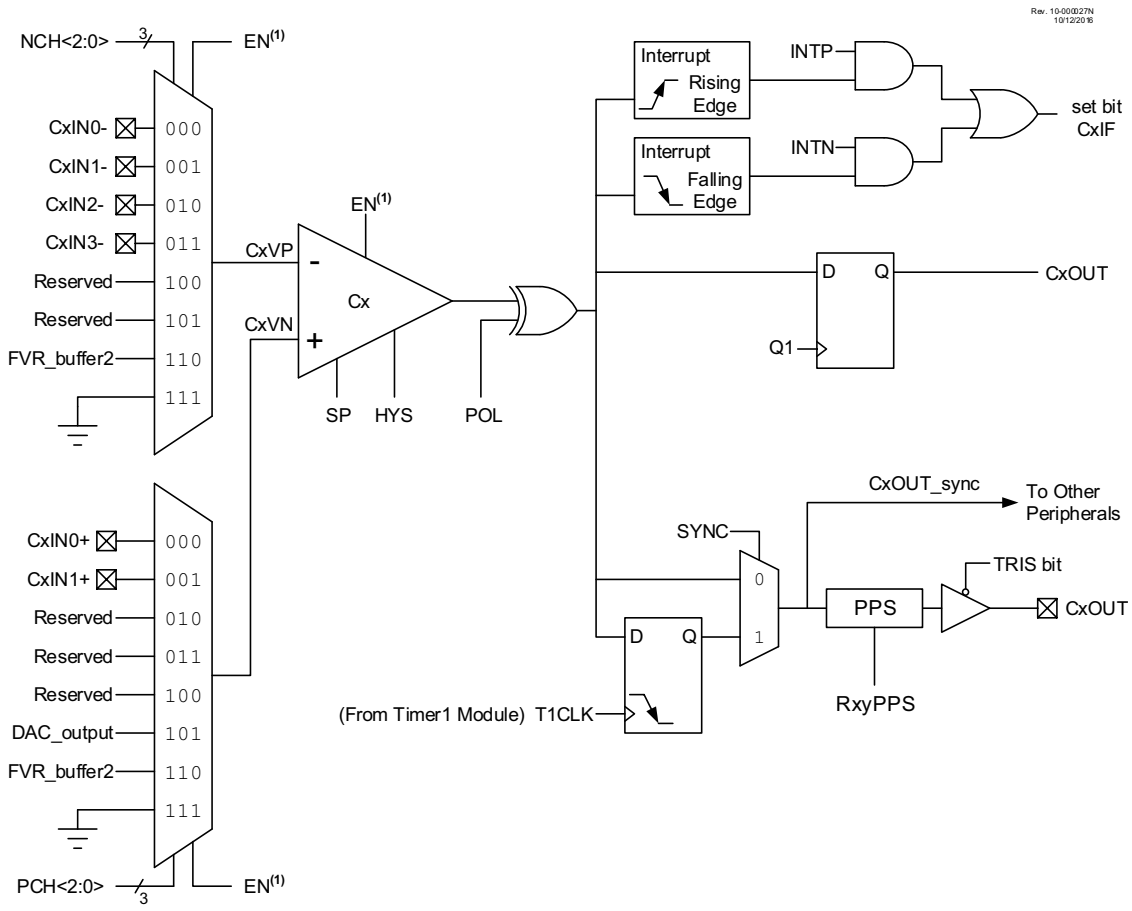


TABLE 33-18: SUMMARY OF REGISTERS FOR I²C 8-BIT MACRO

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on page
I2CxBTO	—	—	—	—	—	BTO<2:0>			582
I2CxCLK	—	—	—	—	—	CLK<2:0>			581
I2CxPIE	CNTIE	ACKTIE	—	WRIE	ADRIE	PCIE	RSCIE	SCIE	588
I2CxPIR	CNTIF	ACKTIF	—	WRIF	ADRIF	PCIF	RSCIF	SCIF	587
I2CxERR	—	BTOIF	BCLIF	NACKIF	—	BTOIE	BCLIE	NACKIE	585
I2CxSTAT0	BFRE	SMA	MMA	R	D	—	—	—	583
I2CxSTAT1	TXWE	—	TXBE	—	RXRE	CLRBF	—	RXBF	584
I2CxCON0	EN	RSEN	S	CSTR	MDR	MODE<2:0>			577
I2CxCON1	ACKCNT	ACKDT	ACKSTAT	ACKT	—	RXOV	TXU	CSD	579
I2CxCON2	ACNT	GCEN	FME	ADB	SDAHT<3:2>		BFRET<1:0>		580
I2CxADR0	ADR<7:0>								589
I2CxADR1	ADR<7:1>							—	590
I2CxADR2	ADR<7:0>								591
I2CxADR3	ADR<7:1>							—	592
I2CxADB0	ADB<7:0>								593
I2CxADB1	ADB<7:0>								594
I2CxCNT	CNT<7:0>								586
I2CxPIR	CNTIF	ACKTIF	—	WRIF	ADRIF	PCIF	RSCIF	SCIF	587
I2CxPIE	CNTIE	ACKTIE	—	WRIE	ADRIE	PCIE	RSCIE	SCIE	588
I2CxADR0	ADR7	ADR6	ADR5	ADR4	ADR3	ADR2	ADR1	ADR0	589
I2CxADR1	ADR14	ADR13	ADR12	ADR11	ADR10	ADR9	ADR8	—	590
I2CxADR2	ADR7	ADR6	ADR5	ADR4	ADR3	ADR2	ADR1	ADR0	591
I2CxADR3	ADR14	ADR13	ADR12	ADR11	ADR10	ADR9	ADR8	—	592
I2CxADB0	ADB7	ADB6	ADB5	ADB4	ADB3	ADB2	ADB1	ADB0	593
I2CxADB1	ADB7	ADB6	ADB5	ADB4	ADB3	ADB2	ADB1	ADB0	594

Legend: — = unimplemented, read as '0'. Shaded cells are unused by the I²C module.

FIGURE 38-2: COMPARATOR MODULE SIMPLIFIED BLOCK DIAGRAM



Note 1: When CxON = 0, all multiplexer inputs are disconnected and the Comparator will produce a '0' at the output.

BCF **Bit Clear f**

Syntax: BCF f, b {,a}

Operands: $0 \le f \le 255$
$0 \le b \le 7$
$a \in [0,1]$

Operation: $0 \rightarrow f<b>$

Status Affected: None

Encoding:

1001	bbba	ffff	ffff

Description: Bit 'b' in register 'f' is cleared.
If 'a' is '0', the Access Bank is selected.
If 'a' is '1', the BSR is used to select the GPR bank.
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 41.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write register 'f'

Example: `BCF FLAG_REG, 7, 0`

Before Instruction
FLAG_REG = C7h
After Instruction
FLAG_REG = 47h

BN **Branch if Negative**

Syntax: BN n

Operands: $-128 \le n \le 127$

Operation: if NEGATIVE bit is '1'
$(PC) + 2 + 2n \rightarrow PC$

Status Affected: None

Encoding:

1110	0110	nnnn	nnnn

Description: If the NEGATIVE bit is '1', then the program will branch.
The 2's complement number '2n' is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a 2-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:
If Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	Write to PC
No operation	No operation	No operation	No operation

If No Jump:

Q1	Q2	Q3	Q4
Decode	Read literal 'n'	Process Data	No operation

Example: `HERE BN Jump`

Before Instruction
PC = address `(HERE)`
After Instruction
If NEGATIVE = 1;
PC = address `(Jump)`
If NEGATIVE = 0;
PC = address `(HERE + 2)`

CLRWDT **Clear Watchdog Timer**

Syntax: CLRWDT

Operands: None

Operation: 000h → WDT,
000h → WDT postscaler,
1 → $\overline{TO}$,
1 → $\overline{PD}$

Status Affected: $\overline{TO}$, $\overline{PD}$

Encoding:

0000	0000	0000	0100

Description: CLRWDT instruction resets the Watchdog Timer. It also resets the postscaler of the WDT. Status bits, $\overline{TO}$ and $\overline{PD}$, are set.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	Process Data	No operation

Example: CLRWDT

Before Instruction
WDT Counter = ?
After Instruction
WDT Counter = 00h
WDT Postscaler = 0
$\overline{TO}$ = 1
$\overline{PD}$ = 1

COMF **Complement f**

Syntax: COMF f {,d {,a}}

Operands: 0 ≤ f ≤ 255
d ∈ [0,1]
a ∈ [0,1]

Operation: ($\bar{f}$) → dest

Status Affected: N, Z

Encoding:

0001	11da	ffff	ffff

Description: The contents of register 'f' are complemented. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f' (default).
If 'a' is '0', the Access Bank is selected.
If 'a' is '1', the BSR is used to select the GPR bank.
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 41.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example: COMF REG, 0, 0

Before Instruction
REG = 13h
After Instruction
REG = 13h
W = ECh

41.2.5 SPECIAL CONSIDERATIONS WITH MICROCHIP MPLAB® IDE TOOLS

The latest versions of Microchip's software tools have been designed to fully support the extended instruction set of the PIC18(L)F2x/4x/5xK42 family of devices. This includes the MPLAB C18 C compiler, MPASM assembly language and MPLAB Integrated Development Environment (IDE).

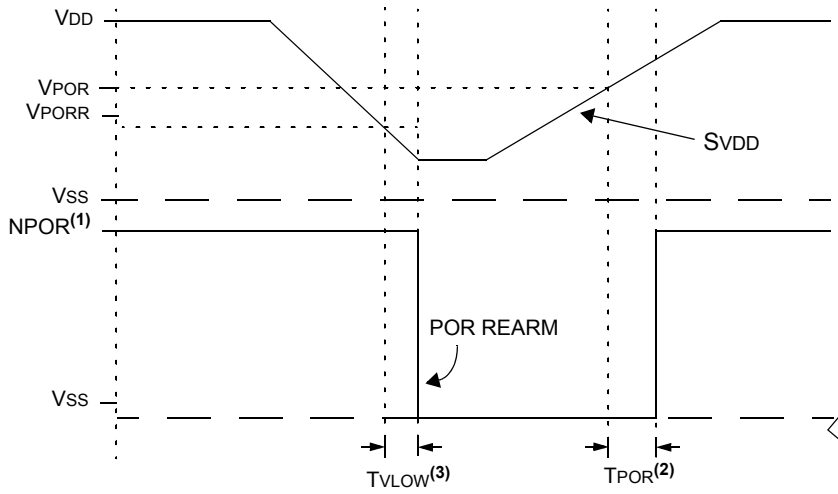
When selecting a target device for software development, MPLAB IDE will automatically set default Configuration bits for that device. The default setting for the XINST Configuration bit is '0', disabling the extended instruction set and Indexed Literal Offset Addressing mode. For proper execution of applications developed to take advantage of the extended instruction set, XINST must be set during programming.

To develop software for the extended instruction set, the user must enable support for the instructions and the Indexed Addressing mode in their language tool(s). Depending on the environment being used, this may be done in several ways:

- A menu option, or dialog box within the environment, that allows the user to configure the language tool and its settings for the project
- A command line option
- A directive in the source code

These options vary between different compilers, assemblers and development environments. Users are encouraged to review the documentation accompanying their development systems for the appropriate information.

FIGURE 44-3: POR AND POR REARM WITH SLOW RISING VDD



Note 1: When NPOR is low, the device is held in Reset.
2: TPOR 1 μs typical.
3: TVLOW 2.7 μs typical.