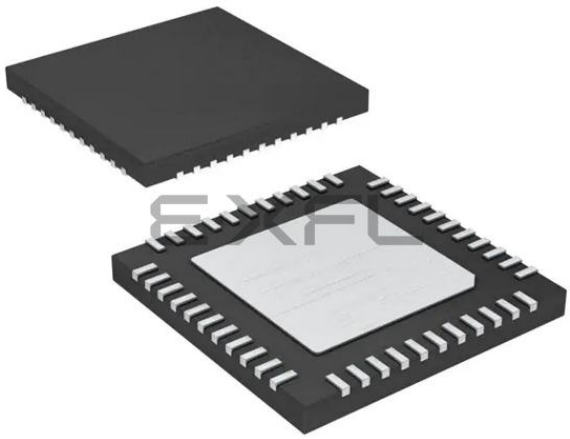


Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

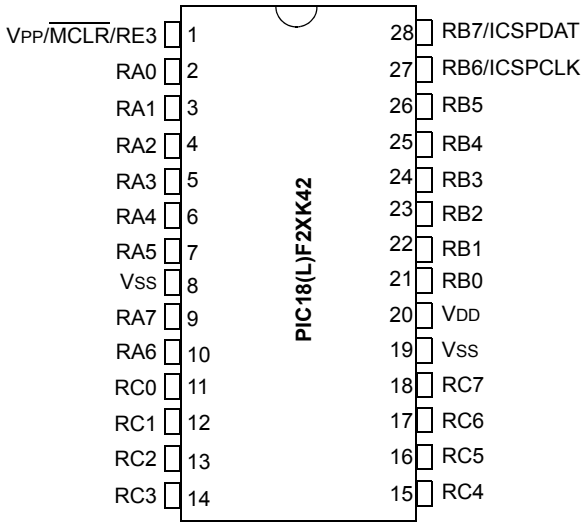
"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" 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	36
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)	1.8V ~ 3.6V
Data Converters	A/D 35x12b; D/A 1x5b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	44-VQFN Exposed Pad
Supplier Device Package	44-QFN (8x8)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18lf47k42t-i-ml

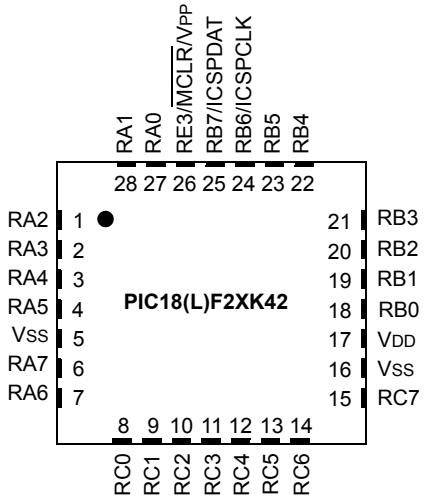
Pin Diagrams

28-pin SPDIP, SOIC, SSOP



Note: See Table 1 for location of all peripheral functions.

28-pin QFN (6x6x0.9mm), UQFN (6x6x0.5mm)



Note 1: See Table 1 for location of all peripheral functions.

2: It is recommended that the exposed bottom pad be connected to VSS, however it must not be the only VSS connection to the device.

4.8.3 MAPPING THE ACCESS BANK IN INDEXED LITERAL OFFSET MODE

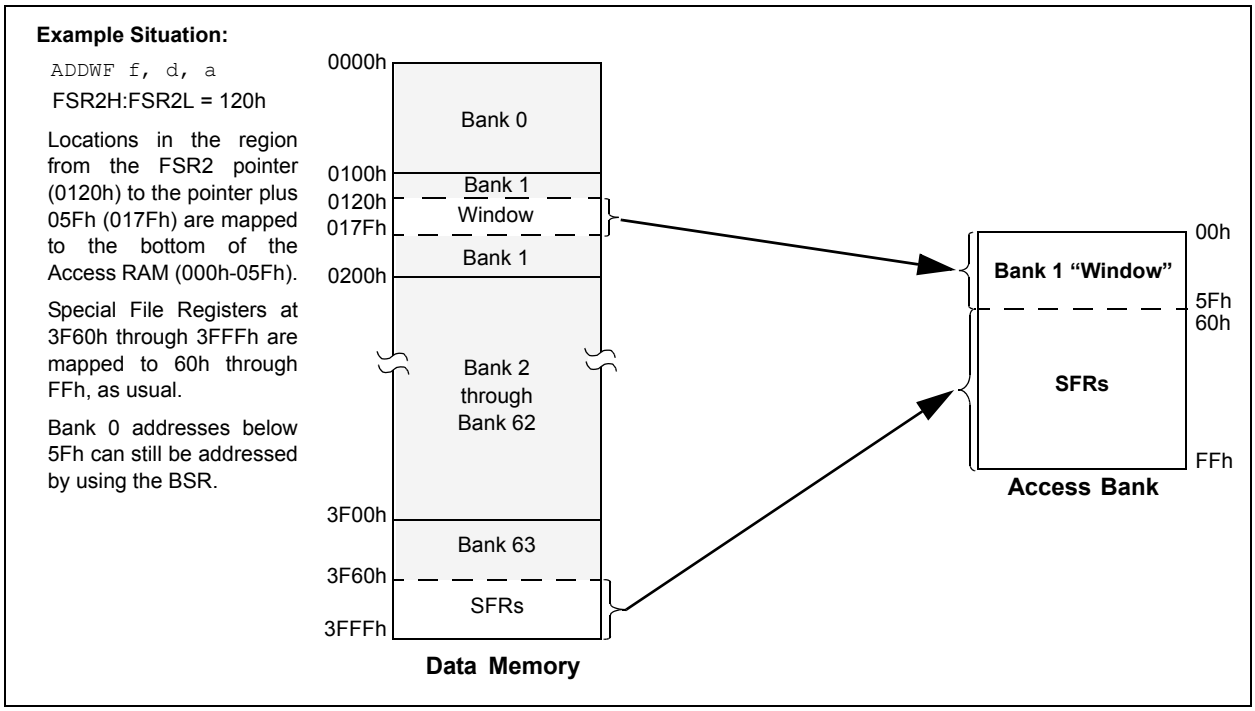
The use of Indexed Literal Offset Addressing mode effectively changes how the first 96 locations of Access RAM (00h to 5Fh) are mapped. Rather than containing just the contents of the bottom section of Bank 0, this mode maps the contents from a user defined “window” that can be located anywhere in the data memory space. The value of FSR2 establishes the lower boundary of the addresses mapped into the window, while the upper boundary is defined by FSR2 plus 95 (5Fh). Addresses in the Access RAM above 5Fh are mapped as previously described (see **Section 4.5.4 “Access Bank”**). An example of Access Bank remapping in this addressing mode is shown in Figure 4-8.

Remapping of the Access Bank applies *only* to operations using the Indexed Literal Offset mode. Operations that use the BSR (Access RAM bit is ‘1’) will continue to use direct addressing as before.

4.9 PIC18 Instruction Execution and the Extended Instruction Set

Enabling the extended instruction set adds eight additional commands to the existing PIC18 instruction set. These instructions are executed as described in **Section 41.2 “Extended Instruction Set”**.

FIGURE 4-8: REMAPPING THE ACCESS BANK WITH INDEXED LITERAL OFFSET ADDRESSING



7.5 Register Definitions: Oscillator Control

REGISTER 7-1: OSCCON1: OSCILLATOR CONTROL REGISTER 1

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

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

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

bit 6-4 **NOSC<2:0>:** New Oscillator Source Request bits(1,2,3)
The setting requests a source oscillator and PLL combination per Table 7-1.
POR value = RSTOSC (Register 5-1).

bit 3-0 **NDIV<3:0>:** New Divider Selection Request bits(2,3)
The setting determines the new postscaler division ratio per Table 7-1.

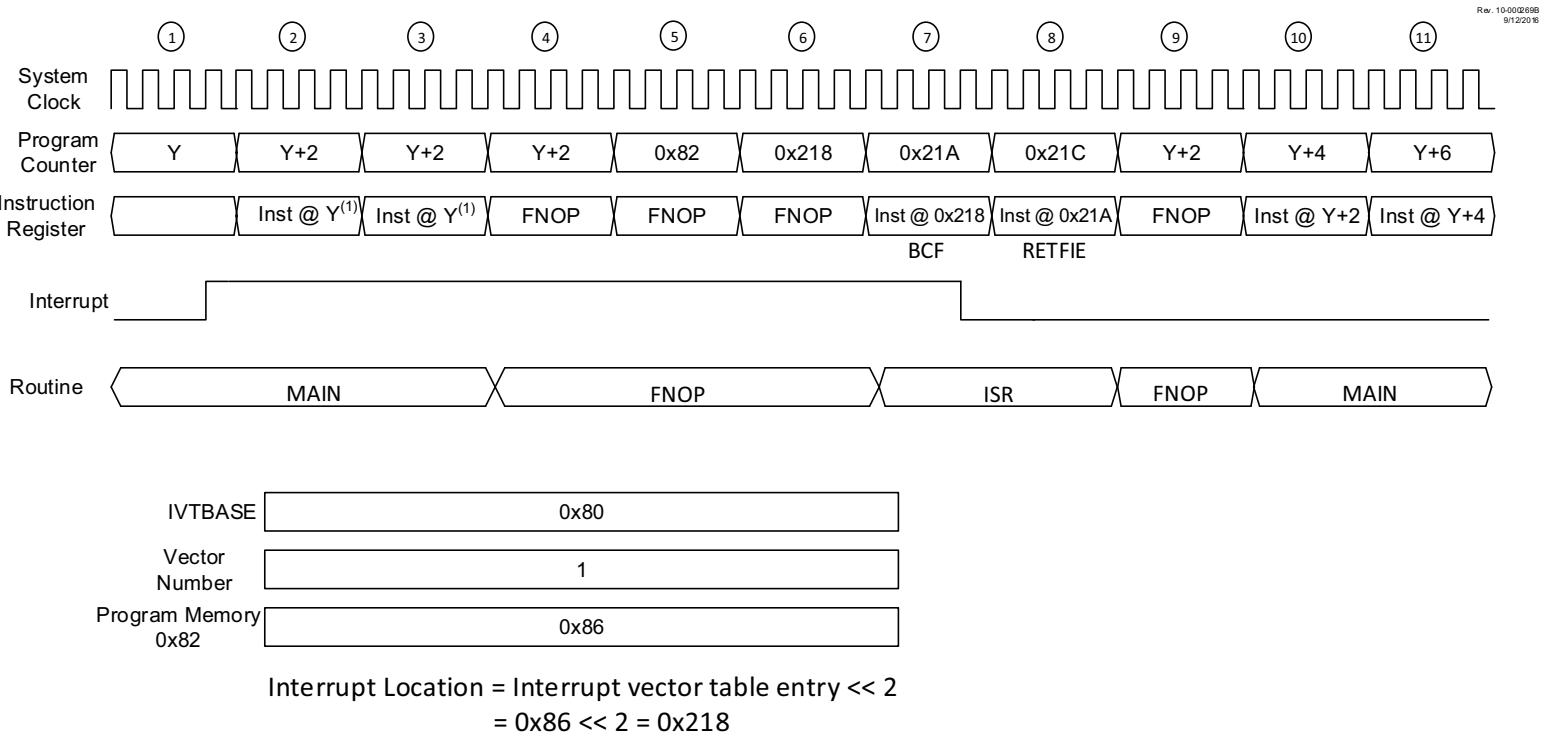
Note 1: The default value (f/f) is determined by the RSTOSC Configuration bits. See Table 7-2 below.
2: If NOSC is written with a reserved value (Table 7-1), the operation is ignored and neither NOSC nor NDIV is written.
3: When CSWEN = 0, this register is read-only and cannot be changed from the POR value.

TABLE 7-2: DEFAULT OSCILLATOR SETTINGS

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

Note 1: EXTOSC must meet the PLL specifications (Table 44-11).

FIGURE 9-8: INTERRUPT TIMING DIAGRAM - TWO WORD INSTRUCTION



Note 1: Instruction @ Y is a Two-cycle instruction.

Example 12-3 shows the sequence to do a 16 x 16 unsigned multiplication. Equation 12-1 shows the algorithm that is used. The 32-bit result is stored in four registers (RES<3:0>).

EQUATION 12-1: 16 x 16 UNSIGNED MULTIPLICATION ALGORITHM

$$
\begin{aligned}
RES3:RES0 &= ARG1H:ARG1L \bullet ARG2H:ARG2L \\
&= (ARG1H \bullet ARG2H \bullet 2^{16}) + \\
&\quad (ARG1H \bullet ARG2L \bullet 2^{8}) + \\
&\quad (ARG1L \bullet ARG2H \bullet 2^{8}) + \\
&\quad (ARG1L \bullet ARG2L)
\end{aligned}
$$

EXAMPLE 12-3: 16 x 16 UNSIGNED MULTIPLY ROUTINE

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

Example 12-4 shows the sequence to do a 16 x 16 signed multiply. Equation 12-2 shows the algorithm used. The 32-bit result is stored in four registers (RES<3:0>). To account for the sign bits of the arguments, the MSb for each argument pair is tested and the appropriate subtractions are done.

EQUATION 12-2: 16 x 16 SIGNED MULTIPLICATION ALGORITHM

$$
\begin{aligned}
RES3:RES0 &= ARG1H:ARG1L \bullet ARG2H:ARG2L \\
&= (ARG1H \bullet ARG2H \bullet 2^{16}) + \\
&\quad (ARG1H \bullet ARG2L \bullet 2^{8}) + \\
&\quad (ARG1L \bullet ARG2H \bullet 2^{8}) + \\
&\quad (ARG1L \bullet ARG2L) + \\
&\quad (-1 \bullet ARG2H\langle 7\rangle \bullet ARG1H:ARG1L \bullet 2^{16}) + \\
&\quad (-1 \bullet ARG1H\langle 7\rangle \bullet ARG2H:ARG2L \bullet 2^{16})
\end{aligned}
$$

EXAMPLE 12-4: 16 x 16 SIGNED MULTIPLY ROUTINE

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

13.1.4 NVM UNLOCK SEQUENCE

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

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

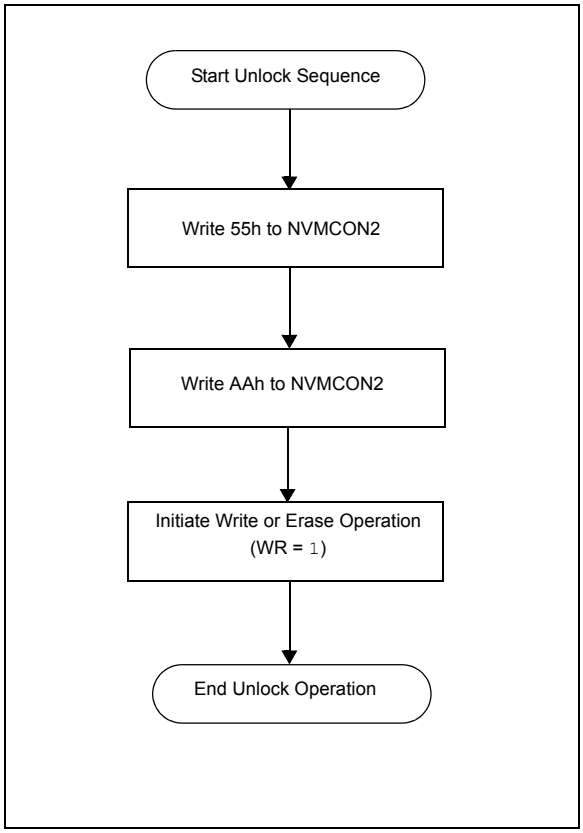
The unlock sequence consists of the following steps and must be completed in order:

- Write 55h to NVMCON2
- Write AAh to NMVCON2
- Set the WR bit of NVMCON1

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

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

FIGURE 13-6: NVM UNLOCK SEQUENCE FLOWCHART



EXAMPLE 13-2: NVM UNLOCK SEQUENCE

```
BCF                 INTCON0,GIE             ; Recommended so sequence is not interrupted
BANKSEL             NVMCON1
BSF                 NVMCON1,WREN            ; Enable write/erase
MOVLW               55h                     ; Load 55h

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

BSF                 INTCON0,GIE             ; Re-enable interrupts
```

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

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

14.5 CRC Check Value

The CRC check value will be located in the CRCACC registers after the CRC calculation has finished. The check value will depend on two mode settings of the CRCCON0 register: ACCM and SHIFTM. When the ACCM bit is set, the CRC module augments the data with a number of zeros equal to the length of the polynomial to align the final check value. When the ACCM bit is not set, the CRC will stop at the end of the data. A number of zeros equal to the length of the polynomial can then be entered into CRCDAT to find the same check value as augmented mode. Alternatively, the expected check value can be entered at this point to make the final result equal '0'.

When the CRC check value is computed with the SHIFTM bit set, selecting LSb first, and the ACCM bit is also set, then the final value in the CRCACC registers will be reversed such that the LSb will be in the MSb position and vice versa. This is the expected check value in bit reversed form. If you are creating a check value to be appended to a data stream, then a bit reversal must be performed on the final value to achieve the correct checksum. You can use the CRC to do this reversal by the following method:

- Save the CRCACC value in user RAM space
- Clear the CRCACC registers
- Clear the CRCXOR registers
- Write the saved CRCACC value to the CRCDAT input.

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

14.6 CRC Interrupt

The CRC will generate an interrupt when the BUSY bit transitions from 1 to 0. The CRCIF Interrupt Flag is set every time the BUSY bit transitions, regardless of whether or not the CRC interrupt is enabled. The CRCIF bit can only be cleared in software.

14.7 Configuring the CRC

The following steps illustrate how to properly configure the CRC.

1. Determine if the automatic program memory scan will be used with the scanner or manual calculation through the SFR interface and perform the actions specified in **Section 14.4 "CRC Data Sources"**, depending on which decision was made.
2. If desired, seed a starting CRC value into the CRCACCH/L registers.
3. Program the CRCXORH/L registers with the desired generator polynomial.
4. Program the DLEN<3:0> bits of the CRCCON1 register with the length of the data word - 1 (refer to Example 14-1). This determines how many times the shifter will shift into the accumulator for each data word.
5. Program the PLEN<3:0> bits of the CRCCON1 register with the length of the polynomial -2 (refer to Example 14-1).
6. Determine whether shifting in trailing zeros is desired and set the ACCM bit of the CRCCON0 register appropriately.
7. Likewise, determine whether the MSb or LSb should be shifted first and write the SHIFTM bit of the CRCCON0 register appropriately.
8. Write the GO bit of the CRCCON0 register to begin the shifting process.

9a. If manual SFR entry is used, monitor the FULL bit of the CRCCON0 register. When FULL = 0, another word of data can be written to the CRCDATH/L registers, keeping in mind that CRCDATH should be written first if the data has more than eight bits, as the shifter will begin upon the CRCDATL register being written.

9b. If the scanner is used, the scanner will automatically load words into the CRCDATH/L registers as needed, as long as the GO bit is set.

10a. If manual entry is used, monitor the CRCIF (and BUSY bit to determine when the completed CRC calculation can be read from CRCACCH/L registers.

10b. If using the memory scanner, monitor the SCANIF (or the GO bit) for the scanner to finish pushing information into the CRCDAT registers. After the scanner is completed, monitor the BUSY bit to determine that the CRC has been completed and the check value can be read from the CRCACC registers. If both the interrupt flags are set and the BUSY and GO bits are cleared, the completed CRC calculation can be read from the CRCACCH/L registers.

15.3 DMA Organization

The DMA module on the K42 family of devices is designed to move data by using the existing Instruction Bus<16> and Data Bus<8> without the need for any dual-porting of memory or peripheral systems (Figure 15-1). The DMA accesses the required bus when it has been granted to by the System Arbiter.

FIGURE 15-1: DMA FUNCTIONAL BLOCK DIAGRAM

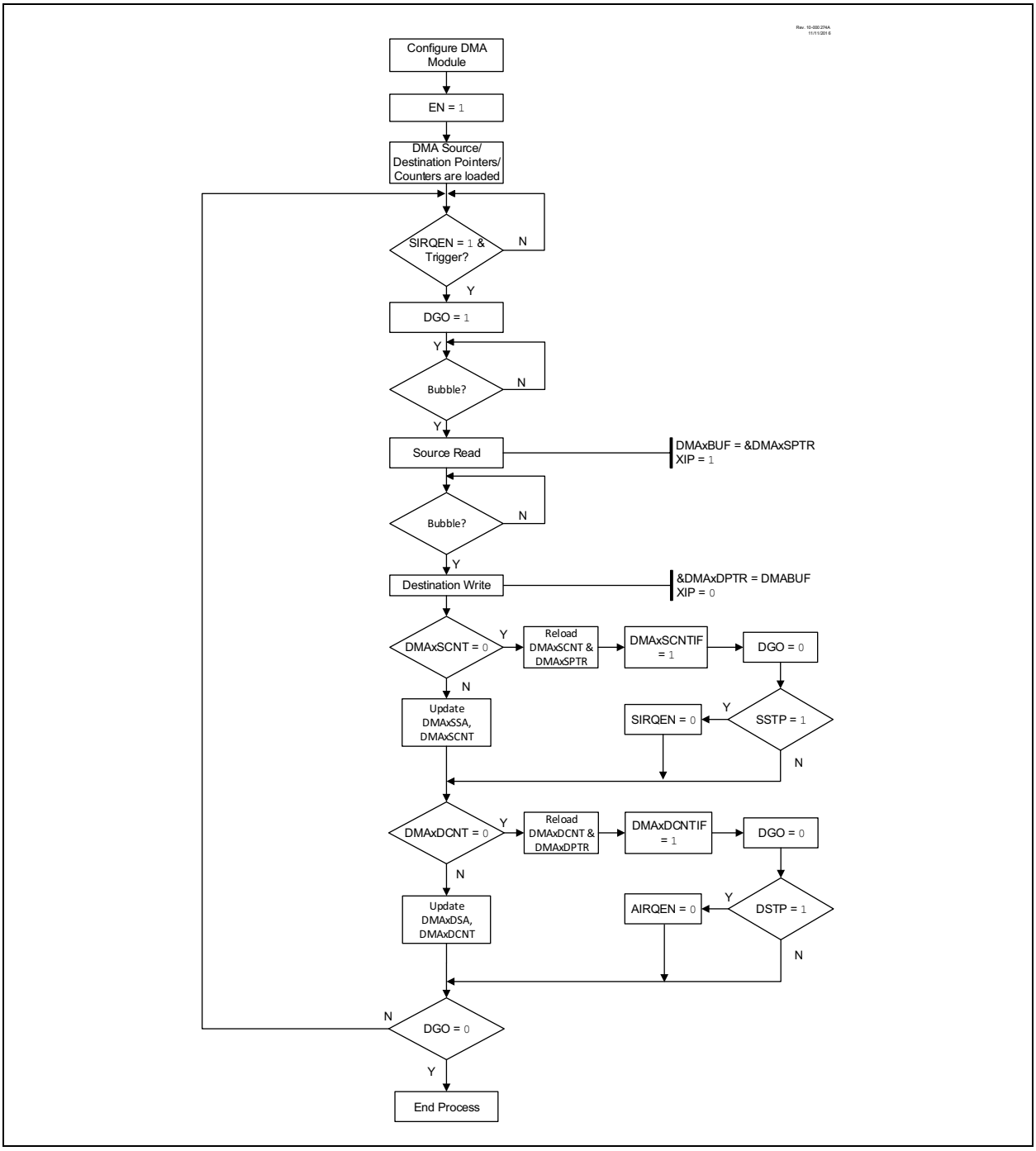


TABLE 15-2: DMAxSIRQ AND DMAxAIRQ INTERRUPT SOURCES

DMAxSIRQ DMAxAIRQ	Trigger Source	Level Triggered
0	Reserved	
1	LVD	No
2	OSF	No
3	CSW	No
4	NVM	No
5	SCAN	No
6	CRC	No
7	IOC	Yes
8	INT0	No
9	ZCD	No
10	AD	No
11	ADT	No
12	CMP1	No
13	SMT1	No
14	SMT1PRA	No
15	SMT1PWA	No
16	DMA1SCNT	No
17	DMA1DCNT	No
18	DMA1OR	No
19	DMA1A	No
20	SPI1RX	Yes
21	SPI1TX	Yes
22	SPI1	Yes
23	I2C1RX	Yes
24	I2C1TX	Yes
25	I2C1	Yes
26	I2C1E	Yes
27	U1RX	Yes
28	U1TX	Yes
29	U1E	Yes
30	U1	No
31	TMR0	No
32	TMR1	No
33	TMR1G	No
34	TMR2	No
35	CCP1	No
36	Reserved	
37	NCO	No
38	CWG1	No
39	CLC1	No
40	INT1	No
41	CMP2	No
42	DMA2SCNT	No
43	DMA2DCNT	No
44	DMA2OR	No
45	DMA2A	No
46	I2C2RX	Yes
47	I2C2TX	Yes
48	I2C2	Yes
49	I2C2E	Yes
50	U2RX	Yes
51	U2TX	Yes
52	U2E	Yes
53	U2	No
54	TMR3	No
55	TMR3G	No
56	TMR4	No
57	CCP2	No
58	Reserved	
59	CWG2	No
60	CLC2	No
61	INT2	No
62	Reserved	
63	Reserved	
64	Reserved	
65	Reserved	
66	Reserved	
67	Reserved	
68	Reserved	
69	Reserved	
70	TMR5	No
71	TMR5G	No
72	TMR6	No
73	CCP3	No
74	CWG3	No
75	CLC3	No
76	Reserved	
77	Reserved	
78	Reserved	
79	Reserved	
80	CCP4	No
81	CLC4	No
82 – 127	Reserved	

Note 1: All trigger sources that are not Level-triggered are Edge-triggered.

REGISTER 21-4: TxGATE: TIMERx GATE ISM REGISTER

U-0	U-0	U-0	R/W-0/u	R/W-0/u	R/W-0/u	R/W-0/u	R/W-0/u
—	—	—	GSS<4:0>				
bit 7							bit 0

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

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

bit 4-0 **GSS<4:0>:** Timerx Gate Source Selection bits

GSS	Timer1	Timer3	Timer5
	Gate Source	Gate Source	Gate Source
11111-11011	Reserved	Reserved	Reserved
11010	CLC4_out	CLC4_out	CLC4_out
11001	CLC3_out	CLC3_out	CLC3_out
11000	CLC2_out	CLC2_out	CLC2_out
10111	CLC1_out	CLC1_out	CLC1_out
10110	ZCDOUT	ZCDOUT	ZCDOUT
10101	CMP2OUT	CMP2OUT	CMP2OUT
10100	CMP1OUT	CMP1OUT	CMP1OUT
10011	NCO1OUT	NCO1OUT	NCO1OUT
10010-10001	Reserved	Reserved	Reserved
10000	PWM8OUT	PWM8OUT	PWM8OUT
01111	PWM7OUT	PWM7OUT	PWM7OUT
01110	PWM6OUT	PWM6OUT	PWM6OUT
01101	PWM5OUT	PWM5OUT	PWM5OUT
01100	CCP4OUT	CCP4OUT	CCP4OUT
01011	CCP3OUT	CCP3OUT	CCP3OUT
01010	CCP2OUT	CCP2OUT	CCP2OUT
01001	CCP1OUT	CCP1OUT	CCP1OUT
01000	SMT1_match	SMT1_match	SMT1_match
00111	TMR6OUT (postscaled)	TMR6OUT (postscaled)	TMR6OUT (postscaled)
00110	TMR5 overflow	TMR5 overflow	Reserved
00101	TMR4OUT (postscaled)	TMR4OUT (postscaled)	TMR4OUT (postscaled)
00100	TMR3 overflow	Reserved	TMR3 overflow
00011	TMR2OUT (postscaled)	TMR2OUT (postscaled)	TMR2OUT (postscaled)
00010	Reserved	TMR1 overflow	TMR1 overflow
00001	TMR0 overflow	TMR0 overflow	TMR0 overflow
00000	Pin selected by T1GPPS	Pin selected by T3GPPS	Pin selected by T5GPPS

23.3.1 CCPx PIN CONFIGURATION

The software must configure the CCPx pin as an output by clearing the associated TRIS bit and defining the appropriate output pin through the RxyPPS registers. See **Section 17.0 "Peripheral Pin Select (PPS) Module"** for more details.

Note:	Clearing the CCPxCON register will force the CCPx compare output latch to the default low level. This is not the PORT I/O data latch.

23.3.2 TIMER1 MODE RESOURCE

In Compare mode, Timer1 must be running in either Timer mode or Synchronized Counter mode. The compare operation may not work in Asynchronous Counter mode.

See **Section 21.0 "Timer1/3/5 Module with Gate Control"** for more information on configuring Timer1.

Note:	Clocking Timer1 from the system clock (FOSC) should not be used in Compare mode. In order for Compare mode to recognize the trigger event on the CCPx pin, TImer1 must be clocked from the instruction clock (FOSC/4) or from an external clock source.

23.3.3 AUTO-CONVERSION TRIGGER

All CCPx modes set the CCP interrupt flag (CCPxIF). When this flag is set and a match occurs, an auto-conversion trigger can take place if the CCP module is selected as the conversion trigger source.

Refer to **Section 36.2.5 "Auto-Conversion Trigger"** for more information.

Note:	Removing the match condition by changing the contents of the CCPRxH and CCPRxL register pair, between the clock edge that generates the Auto-conversion Trigger and the clock edge that generates the Timer1 Reset, will preclude the Reset from occurring

23.3.4 COMPARE DURING SLEEP

Since FOSC is shut down during Sleep mode, the Compare mode will not function properly during Sleep, unless the timer is running. The device will wake on interrupt (if enabled).

23.4 PWM Overview

Pulse-Width Modulation (PWM) is a scheme that provides power to a load by switching quickly between fully ON and fully OFF states. The PWM signal resembles a square wave where the high portion of the signal is considered the ON state and the low portion of the signal is considered the OFF state. The high portion, also known as the pulse width, can vary in time and is defined in steps. A larger number of steps applied, which lengthens the pulse width, also supplies more power to the load. Lowering the number of steps applied, which shortens the pulse width, supplies less power. The PWM period is defined as the duration of one complete cycle or the total amount of on and off time combined.

PWM resolution defines the maximum number of steps that can be present in a single PWM period. A higher resolution allows for more precise control of the pulse-width time and in turn the power that is applied to the load.

The term duty cycle describes the proportion of the on time to the off time and is expressed in percentages, where 0% is fully off and 100% is fully on. A lower duty cycle corresponds to less power applied and a higher duty cycle corresponds to more power applied.

Figure 23-3 shows a typical waveform of the PWM signal.

23.4.1 STANDARD PWM OPERATION

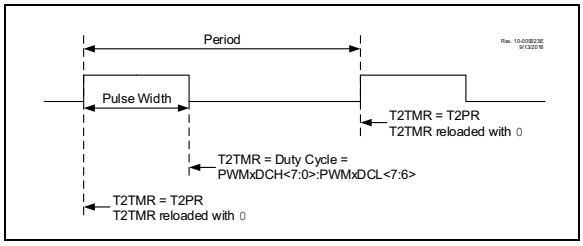
The standard PWM mode generates a Pulse-Width Modulation (PWM) signal on the CCPx pin with up to ten bits of resolution. The period, duty cycle, and resolution are controlled by the following registers:

- T2PR registers
- T2CON registers
- CCPRxL and CCPRxH registers
- CCPxCON registers

It is required to have FOSC/4 as the clock input to TMR2/4/6 for correct PWM operation. Figure 23-4 shows a simplified block diagram of PWM operation.

Note:	The corresponding TRIS bit must be cleared to enable the PWM output on the CCPx pin.

FIGURE 23-3: CCP PWM OUTPUT SIGNAL



25.6.11 WINDOWED COUNTER MODE

This mode counts pulses on the SMT1_signal input, within a window dictated by the SMT1WIN input. It begins counting upon seeing a rising edge of the SMT1WIN input, updates the SMT1CPW register on a falling edge of the SMT1WIN input, and updates the SMT1CPR register on each rising edge of the SMT1WIN input beyond the first. See Figure 25-21 and Figure 25-22.

REGISTER 26-5: CWGxSTR(1): CWG STEERING CONTROL REGISTER

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
OVRD	OVRC	OVRB	OVRA	STRD(2)	STRC(2)	STRB(2)	STRA(2)
bit 7							bit 0

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

bit 7 **OVRD:** Steering Data D bit

bit 6 **OVRC:** Steering Data C bit

bit 5 **OVRB:** Steering Data B bit

bit 4 **OVRA:** Steering Data A bit

bit 3 **STRD:** Steering Enable bit D(2)
- 1 = CWGxD output has the CWG data input waveform with polarity control from POLD bit
- 0 = CWGxD output is assigned to value of OVRD bit

bit 2 **STRC:** Steering Enable bit C(2)
- 1 = CWGxC output has the CWG data input waveform with polarity control from POLC bit
- 0 = CWGxC output is assigned to value of OVRC bit

bit 1 **STRB:** Steering Enable bit B(2)
- 1 = CWGxB output has the CWG data input waveform with polarity control from POLB bit
- 0 = CWGxB output is assigned to value of OVRB bit

bit 0 **STRA:** Steering Enable bit A(2)
- 1 = CWGxA output has the CWG data input waveform with polarity control from POLA bit
- 0 = CWGxA output is assigned to value of OVRA bit

Note 1: The bits in this register apply only when MODE<2:0> = 00x (Register 26-1, Steering modes).

2: This bit is double-buffered when MODE<2:0> = 001.

30.0 DATA SIGNAL MODULATOR (DSM) MODULE

The Data Signal Modulator (DSM) is a peripheral which allows the user to mix a data stream, also known as a modulator signal, with a carrier signal to produce a modulated output.

Both the carrier and the modulator signals are supplied to the DSM module either internally, from the output of a peripheral, or externally through an input pin.

The modulated output signal is generated by performing a logical “AND” operation of both the carrier and modulator signals and then provided to the MDOUT pin.

The carrier signal is comprised of two distinct and separate signals. A carrier high (CARH) signal and a carrier low (CARL) signal. During the time in which the modulator (MOD) signal is in a logic high state, the DSM mixes the carrier high signal with the modulator signal. When the modulator signal is in a logic low state, the DSM mixes the carrier low signal with the modulator signal.

Using this method, the DSM can generate the following types of Key Modulation schemes:

- Frequency-Shift Keying (FSK)
- Phase-Shift Keying (PSK)
- On-Off Keying (OOK)

Additionally, the following features are provided within the DSM module:

- Carrier Synchronization
- Carrier Source Polarity Select
- Programmable Modulator Data
- Modulated Output Polarity Select
- Peripheral Module Disable, which provides the ability to place the DSM module in the lowest power consumption mode

Figure 30-1 shows a Simplified Block Diagram of the Data Signal Modulator peripheral.

FIGURE 31-7: MANCHESTER TIMING

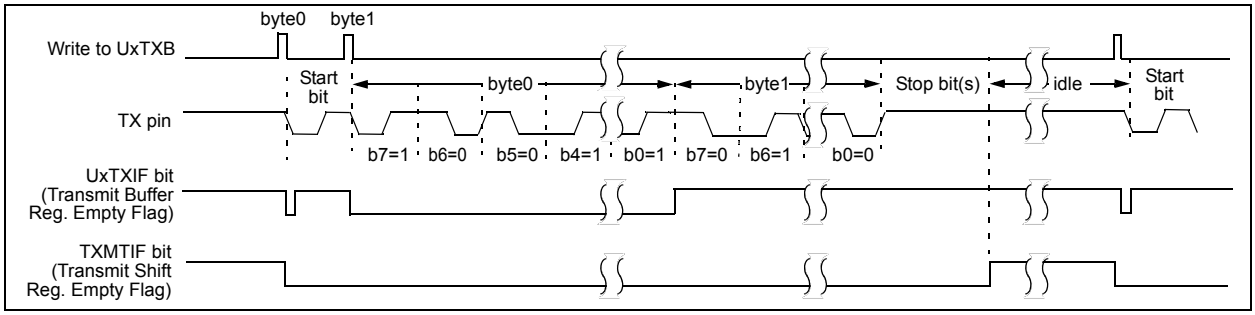


FIGURE 31-8: DALI FRAME TIMING

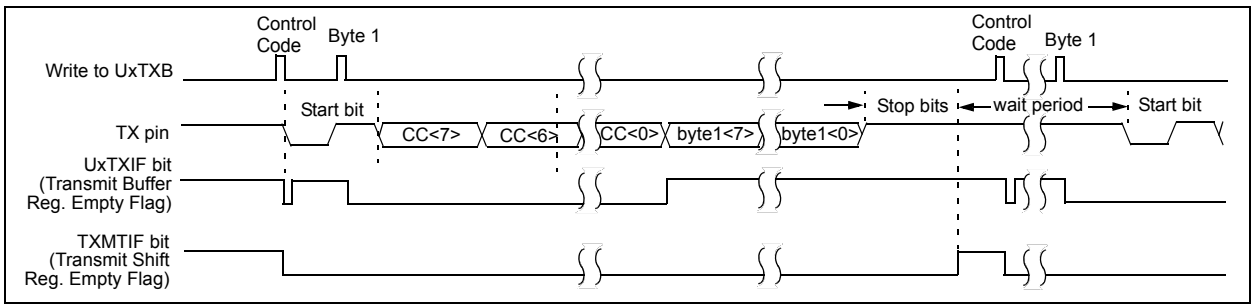
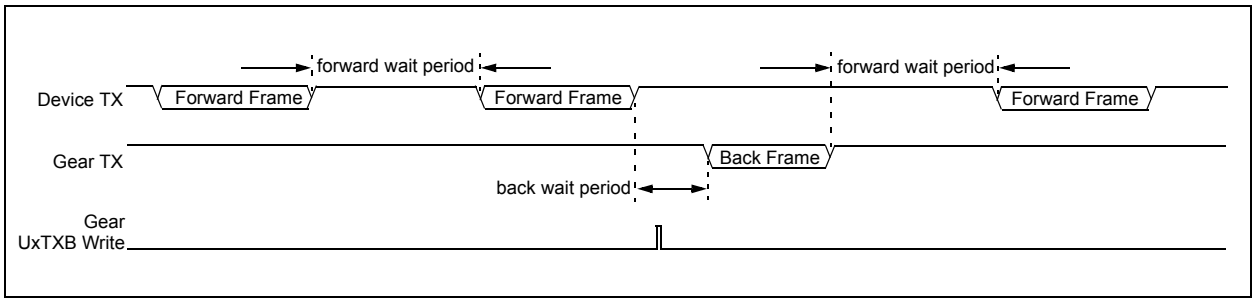


FIGURE 31-9: DALI FORWARD/BACK FRAME TIMING



31.7 General Purpose Manchester (UART1 only)

General purpose Manchester is a subset of the DALI mode. When the UxP1L register is cleared, there is no minimum wait time between frames. This allows full and half-duplex operation because writes to the UxTXB are not held waiting for a receive operation to complete.

General purpose Manchester operation maintains all other aspects of DALI mode such as:

- Single-pulse Start bit
- Most Significant bit first
- No stop periods between back-to-back bytes

General purpose Manchester mode is configured with the following settings:

- MODE<3:0> = 1000
- TXEN = 1
- RXEN = 1
- UxP1 = 0h
- UxBRGH:L = desired baud rate
- TXPOL and RXPOL = desired Idle state
- STP = desired number of stop periods
- RxyPPS = TX pin selection code
- TX pin TRIS control = 0
- RXPPS = RX pin selection code
- RX pin TRIS control = 1
- Input pin ANSEL bit = 0
- ON = 1

The Manchester bit stream timing is shown in Figure 31-7.

32.2 SPI REGISTERS

- SPI Interrupt Flag Register (SPIxINTF)
- SPI Interrupt Enable Register (SPIxINTE)
- SPI Byte Count High and Low Registers (SPIxTCTH/L)
- SPI Bit Count Register (SPIxTWIDTH)
- SPI Baud Rate Register (SPIxBAUD)
- SPI Control Register 0 (SPIxCON0)
- SPI Control Register 1 (SPIxCON1)
- SPI Control Register 2 (SPIxCON2)
- SPI FIFO Status Register (SPIxSTATUS)
- SPI Receiver Buffer Register (SPIxRB)
- SPI Transmit Buffer Register (SPIxTB)
- SPI Clock Select Register (SPIxCLK)

SPIxCON0, SPIxCON1, and SPIxCON2 are control registers for the SPI module.

SPIxSTATUS contains several Status bits that indicate the status of both the SPI module and the receive and transmit FIFOs.

SPIxBAUD and SPIxCLK control the baud rate generator of the SPI module when in Master mode. The SPIxCLK selects the clock source that is used. The SPIxBAUD configures the clock divider used on that clock. More information on the baud rate generator is available in **Section 32.5.6 "Master Mode SPI Clock Configuration"**."

SPIxTxB and SPIxRxB are the transmit and receive buffer registers used to send and receive data on the SPI bus. They both offer indirect access to shift registers that are used for shifting the data in and out. Both registers access the two-byte FIFOs, allowing for multiple transmissions/receptions to be stored between software transfers the data.

The SPIxTCTH:L register pair either count or control the number of bits or bytes in a data transfer. When BMODE = 1, the SPIxTCT value signifies bytes and the SPIxTWIDTH value signifies the number of bits in a byte. When BMODE = 0, the SPIxTCT value is concatenated with the SPIxTWIDTH register to signify bits. In Master Receive-only mode (TXR = 0 and RXR = 1), the data transfer is initiated by writing SPIxTCT with the desired bit or byte value to transfer. In Master Transmit mode (TXR = 1), the data transfer is initiated by writing the SPIxTxB register, in which case the SPIxTCT is a down counter for the bits or bytes transferred.

The SPIxINTF and SPIxINTE are the flags and enables, respectively, for SPI-specific interrupts. They are tied to the SPIxIF flag and SPIxIE enable in the PIR and PIE registers, which is triggered when any interrupt contained in the SPIxINTF/SPIxINTE registers is triggered. The PIR/PIE registers also contain SPIxTXIF/SPIxTXIE bits, which are the interrupt flag and enable for the SPI Transmit Interrupt, as well as the SPIxRXIF/SPIxRXIE bits, which are the interrupt flag and enable for the SPI Receive Interrupt.

REGISTER 36-2: ADCON1: ADC CONTROL REGISTER 1

R/W-0/0	R/W-0/0	R/W-0/0	U-0	U-0	U-0	U-0	R/W-0/0
PPOL	IPEN	GPOL	—	—	—	—	DSEN
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 **PPOL:** Precharge Polarity bit
If PRE>0x00:

PPOL	Action During 1st Precharge Stage	
	External (selected analog I/O pin)	Internal (AD sampling capacitor)
1	Connected to V_{DD}	C_{HOLD} connected to V_{SS}
0	Connected to V_{SS}	C_{HOLD} connected to V_{DD}

Otherwise:
The bit is ignored

bit 6 **IPEN:** A/D Inverted Precharge Enable bit
If DSEN = 1
1 = The precharge and guard signals in the second conversion cycle are the opposite polarity of the first cycle
0 = Both Conversion cycles use the precharge and guards specified by ADPPOL and ADGPOL
Otherwise:
The bit is ignored

bit 5 **GPOL:** Guard Ring Polarity Selection bit
1 = ADC guard Ring outputs start as digital high during Precharge stage
0 = ADC guard Ring outputs start as digital low during Precharge stage

bit 4-1 **Unimplemented:** Read as '0'

bit 0 **DSEN:** Double-sample enable bit
1 = Two conversions are performed on each trigger. Data from the first conversion appears in PREV
0 = One conversion is performed for each trigger

CPFSEQ **Compare f with W, skip if f = W**

Syntax: CPFSEQ 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	001a	ffff	ffff

Description: Compares the contents of data memory location 'f' to the contents of W by performing an unsigned subtraction.
If 'f' = W, then the fetched instruction is discarded and a NOP is executed instead, making this a 2-cycle instruction.
If 'a' is '0', the Access Bank is selected.
If 'a' is '1', the BSR is used to select the GPR bank.
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 41.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1(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: 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 2-cycle instruction.
If 'a' is '0', the Access Bank is selected.
If 'a' is '1', the BSR is used to select the GPR bank.
If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 41.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1(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)

TABLE 42-1: REGISTER FILE SUMMARY FOR PIC18(L)F26/27/45/46/47/55/56/57K42 DEVICES

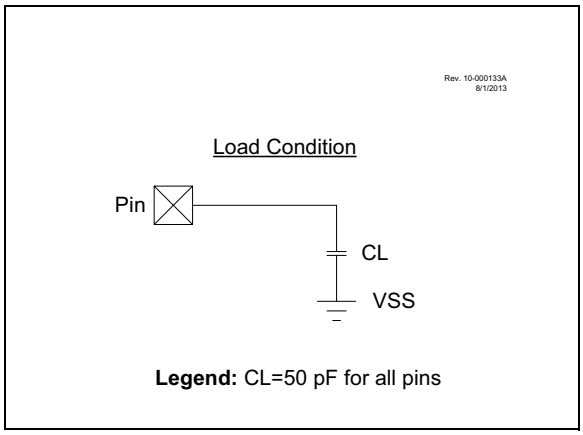
Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on page
3BC8h - 3AEBh	—	Unimplemented								
3AEAh	U2CTSPPS	—	—	—	U2CTSPPS					277
3AE8h	U2RXPPS	—	—	—	U2RXPPS					277
3AE7h	U1CTSPPS	—	—	—	U1CTSPPS					277
3AE5h	U1RXPPS	—	—	—	U1RXPPS					277
3AE4h	I2C2SDAPPS	—	—	—	I2C2SDAPPS					277
3AE3h	I2C2SCLPPS	—	—	—	I2C2SCLPPS					277
3AE2h	I2C1SDAPPS	—	—	—	I2C1SDAPPS					277
3AE1h	I2C1SCLPPS	—	—	—	I2C1SCLPPS					277
3AE0h	SPI1SSPPS	—	—	—	SPI1SSPPS					277
3ADFh	SPI1SDIPPS	—	—	—	SPI1SDIPPS					277
3ADEh	SPI1SCKPPS	—	—	—	SPI1SCKPPS					277
3ADDh	ADACTPPS	—	—	—	ADACTPPS					277
3ADCh	CLCIN3PPS	—	—	—	CLCIN3PPS					277
3ADBh	CLCIN2PPS	—	—	—	CLCIN2PPS					277
3ADAh	CLCIN1PPS	—	—	—	CLCIN1PPS					277
3AD9h	CLCIN0PPS	—	—	—	CLCIN0PPS					277
3AD8h	MD1SRCPPS	—	—	—	MD1SRCPPS					277
3AD7h	MD1CARHPPS	—	—	—	MD1CARHPPS					277
3AD6h	MD1CARLPPS	—	—	—	MD1CARLPPS					277
3AD5h	CWG3INPPS	—	—	—	CWG3INPPS					277
3AD4h	CWG2INPPS	—	—	—	CWG2INPPS					277
3AD3h	CWG1INPPS	—	—	—	CWG1INPPS					277
3AD2h	SMT1SIGPPS	—	—	—	SMT1SIGPPS					277
3AD1h	SMT1WINPPS	—	—	—	SMT1WINPPS					277
3AD0h	CCP4PPS	—	—	—	CCP4PPS					277
3ACFh	CCP3PPS	—	—	—	CCP3PPS					277
3ACEh	CCP2PPS	—	—	—	CCP2PPS					277
3ACDh	CCP1PPS	—	—	—	CCP1PPS					277
3ACCh	T6INPPS	—	—	—	T6INPPS					277
3ACBh	T4INPPS	—	—	—	T4INPPS					277
3ACAh	T2INPPS	—	—	—	T2INPPS					277
3AC9h	T5GPPS	—	—	—	T5GPPS					277
3AC8h	T5CLKIPPS	—	—	—	T5CLKIPPS					277
3AC7h	T3GPPS	—	—	—	T3GPPS					277
3AC6h	T3CLKIPPS	—	—	—	T3CLKIPPS					277
3AC5h	T1GPPS	—	—	—	T1GPPS					277
3AC4h	T1CLKIPPS	—	—	—	T1CLKIPPS					277
3AC3h	T0CLKIPPS	—	—	—	T0CLKIPPS					277
3AC2h	INT2PPS	—	—	—	INT2PPS					277
3AC1h	INT1PPS	—	—	—	INT1PPS					277
3AC0h	INT0PPS	—	—	—	INT0PPS					277
3ABFh	PPSLOCK	—	—	—	—	—	—	—	PPSLOCKED	283
3ABEh	—	Reserved, maintain as '0'								
3ABDh - 3A9Ah	—	Unimplemented								
3A99h	—	Reserved, maintain as '0'								

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

Note 1: Unimplemented in LF devices.
2: Unimplemented in PIC18(L)F26/27K42.
3: Unimplemented on PIC18(L)F26/27/45/46/47K42 devices.
4: Unimplemented in PIC18(L)F45/55K42.

44.4 AC Characteristics

FIGURE 44-4: LOAD CONDITIONS



Legend: CL=50 pF for all pins