



Welcome to [E-XFL.COM](https://www.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	I ² C, LINbus, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, LVD, POR, PWM, WDT
Number of I/O	25
Program Memory Size	16KB (8K x 16)
Program Memory Type	FLASH
EEPROM Size	256 x 8
RAM Size	1K x 8
Voltage - Supply (Vcc/Vdd)	2.3V ~ 5.5V
Data Converters	A/D 24x10b; D/A 1x5b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 125°C (TA)
Mounting Type	Surface Mount
Package / Case	28-SOIC (0.295", 7.50mm Width)
Supplier Device Package	28-SOIC
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18f24k40-e-so

REGISTER 3-13: REVISION ID: REVISION ID REGISTER

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

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

Legend:

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

bit 15-12 **Read as '1010'**

These bits are fixed with value '1010' for all devices in this family.

bit 11-6 **MJRREV<5:0>**: Major Revision ID bits

These bits are used to identify a major revision. A major revision is indicated by an all-layer revision (A0, B0, C0, etc.).

Revision A = 6'b00_0000

bit 5-0 **MNRREV<5:0>**: Minor Revision ID bits

These bits are used to identify a minor revision.

REGISTER 4-6: OSCTUNE: HFINTOSC TUNING REGISTER

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

REGISTER 7-3: PMD2: PMD CONTROL REGISTER 2

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

Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

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

'1' = Bit is set

'0' = Bit is cleared

q = Value depends on condition

- bit 7 **Unimplemented:** Read as '0'
- bit 6 **DACMD:** Disable DAC bit
 - 1 = DAC module disabled
 - 0 = DAC module enabled
- bit 5 **ADCMD:** Disable ADC bit
 - 1 = ADC module disabled
 - 0 = ADC module enabled
- bit 4-3 **Unimplemented:** Read as '0'
- bit 2 **CMP2MD:** Disable Comparator CMP2 bit
 - 1 = CMP2 module disabled
 - 0 = CMP2 module enabled
- bit 1 **CMP1MD:** Disable Comparator CMP1 bit
 - 1 = CMP1 module disabled
 - 0 = CMP1 module enabled
- bit 0 **ZCDMD:** Disable Zero-Cross Detect module bit⁽¹⁾
 - 1 = ZCD module disabled
 - 0 = ZCD module enabled

Note 1: Subject to $\overline{\text{ZCD}}$ bit in CONFIG2H.

PIC18(L)F24/25K40

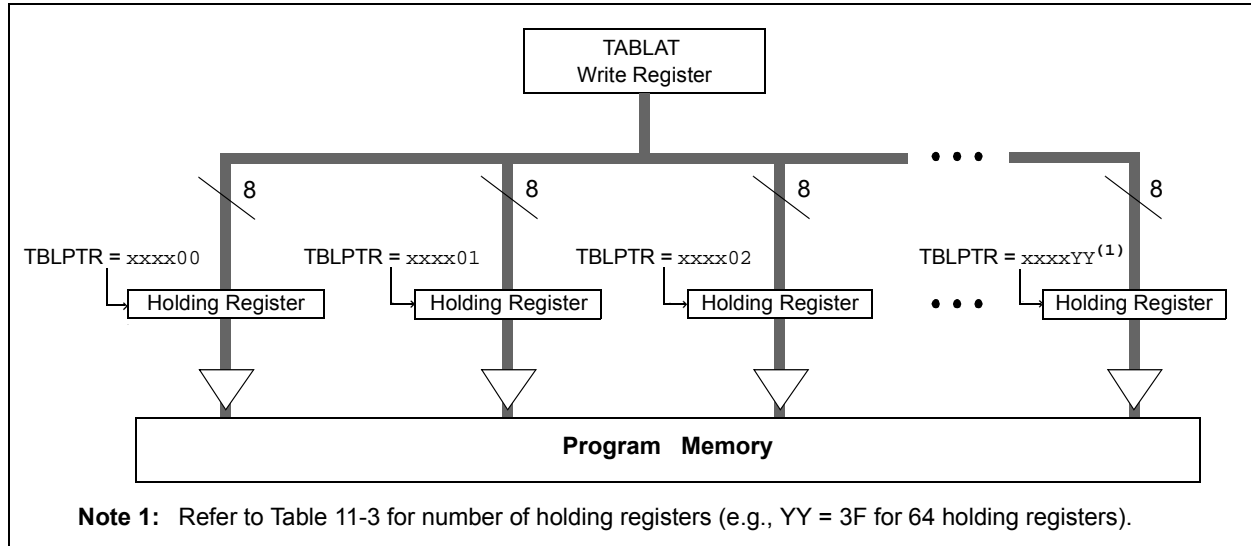
TABLE 10-5: REGISTER FILE SUMMARY FOR PIC18(L)F24/25K40 DEVICES (CONTINUED)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR
F74h	CRCDATL	DATA<7:0>								xxxxxxxx
F73h	ADFLTRH	ADFLTRH<15:8>								xxxxxxxx
F72h	ADFLTRL	ADFLTRL<7:0>								xxxxxxxx
F71h	ADACCH	ADACCH<15:8>								xxxxxxxx
F70h	ADACCL	ADACCL<7:0>								xxxxxxxx
F6Fh	ADERRH	ADERRH<15:8>								00000000
F6Eh	ADERRL	ADERRL<7:0>								00000000
F6Dh	ADUTHH	ADUTHH<15:8>								00000000
F6Ch	ADUTHL	ADUTHL<7:0>								00000000
F6Bh	ADLTHH	ADLTHH<15:8>								00000000
F6Ah	ADLTHL	ADLTHL<7:0>								00000000
F69h	ADSTPTH	ADSTPTH<15:8>								00000000
F68h	ADSTPTL	ADSTPTL<7:0>								00000000
F67h	ADCNT	ADCNT<7:0>								00000000
F66h	ADRPT	ADRPT<7:0>								00000000
F65h	ADSTAT	ADAOV	ADUTHR	ADLTHR	ADMATH	—	ADSTAT<2:0>			0000-000
F64h	ADRESH	ADRESH<7:0>								00000000
F63h	ADRESL	ADRESL<7:0>								00000000
F62h	ADPREVH	ADPREVH<15:8>								00000000
F61h	ADPREVL	ADPREVL<7:0>								00000000
F60h	ADCON0	ADON	ADCONT	—	ADSC	—	ADFM	—	ADGO	00-000-0
F5Fh	ADPCH	—	—	ADPCH<5:0>						--000000
F5Eh	ADPRE	ADPRE<7:0>								00000000
F5Dh	ADCAP	—	—	—	ADCAP<4:0>					---00000
F5Ch	ADACQ	ADACQ<7:0>								00000000
F5Bh	ADCON3	—	ADCALC<2:0>			ADSOI	ADTMD<2:0>			-0000000
F5Ah	ADCON2	ADPSIS	ADCRS<2:0>			ADACLR	ADMD<2:0>			00000000
F59h	ADCON1	ADPPOL	ADIPEN	ADGPOL	—	—	—	—	ADDSEN	000----0
F58h	ADREF	—	—	—	ADNREF	—	—	ADPREF<1:0>		---0--00
F57h	ADCLK	—	—	ADCS<5:0>						--000000
F56h	ADACT	—	—	—	ADACT<4:0>					---00000
F55h	MDCARH	—	—	—	—	—	CHS<2:0>			-----000
F54h	MDCARL	—	—	—	—	—	CLS<2:0>			-----000
F53h	MDSRC	—	—	—	—	SRCS<3:0>				----0000
F52h	MDCON1	—	—	CHPOL	CHSYNC	—	—	CLPOL	CLSYNC	--00--00
F51h	MDCON0	EN	—	OUT	OPOL	—	—	—	MDBIT	0-00---0
F50h	SCANTRIG	—	—	—	—	TSEL<3:0>				----0000
F4Fh	SCANCON0	SCANEN	SCANGO	BUSY	INVALID	INTM	—	MODE<1:0>		00000-00
F4Eh	SCANHADRU	—	—	HADR<21:16>						--111111

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

Note 1: Not available on LF devices.

FIGURE 11-8: TABLE WRITES TO PROGRAM FLASH MEMORY



11.1.6.1 Program Flash Memory Write Sequence

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

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

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

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

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 μ s	6.9 μ s	27.6 μ s	69 μ s
	Hardware multiply	1	1	62.5 ns	100 ns	400 ns	1 μ s
8x8 signed	Without hardware multiply	33	91	5.7 μ s	9.1 μ s	36.4 μ s	91 μ s
	Hardware multiply	6	6	375 ns	600 ns	2.4 μ s	6 μ s
16x16 unsigned	Without hardware multiply	21	242	15.1 μ s	24.2 μ s	96.8 μ s	242 μ s
	Hardware multiply	28	28	1.8 μ s	2.8 μ s	11.2 μ s	28 μ s
16x16 signed	Without hardware multiply	52	254	15.9 μ s	25.4 μ s	102.6 μ s	254 μ s
	Hardware multiply	35	40	2.5 μ s	4.0 μ s	16.0 μ s	40 μ s

14.0 INTERRUPTS

The PIC18(L)F2x/4xK40 devices have multiple interrupt sources and an interrupt priority feature that allows most interrupt sources to be assigned a high or low priority level. The high priority interrupt vector is at 0008h and the low priority interrupt vector is at 0018h. A high priority interrupt event will interrupt a low priority interrupt that may be in progress.

The registers for controlling interrupt operation are:

- INTCON
- PIR1, PIR2, PIR3, PIR4, PIR5, PIR6, PIR7
- PIE1, PIE2, PIE3, PIE4, PIE5, PIE6, PIE7
- IPR1, IPR2, IPR3, IPR4, IPR5, IPR6, IPR7

It is recommended that the Microchip header files supplied with MPLAB® IDE be used for the symbolic bit names in these registers. This allows the assembler/compiler to automatically take care of the placement of these bits within the specified register.

In general, interrupt sources have three bits to control their operation. They are:

- **Flag bit** to indicate that an interrupt event occurred
- **Enable bit** that allows program execution to branch to the interrupt vector address when the flag bit is set
- **Priority bit** to select high priority or low priority

14.1 Mid-Range Compatibility

When the IPEN bit is cleared (default state), the interrupt priority feature is disabled and interrupts are compatible with PIC® microcontroller mid-range devices. In Compatibility mode, the interrupt priority bits of the IPRx registers have no effect. The PEIE/GIEL bit of the INTCON register is the global interrupt enable for the peripherals. The PEIE/GIEL bit disables only the peripheral interrupt sources and enables the peripheral interrupt sources when the GIE/GIEH bit is also set. The GIE/GIEH bit of the INTCON register is the global interrupt enable which enables all non-peripheral interrupt sources and disables all interrupt sources, including the peripherals. All interrupts branch to address 0008h in Compatibility mode.

14.2 Interrupt Priority

The interrupt priority feature is enabled by setting the IPEN bit of the INTCON register. When interrupt priority is enabled the GIE/GIEH and PEIE/GIEL Global Interrupt Enable bits of Compatibility mode are replaced by the GIEH high priority, and GIEL low priority, global interrupt enables. When the IPEN bit is set, the GIEH bit of the INTCON register enables all interrupts which have their associated bit in the IPRx register set. When the GIEH bit is cleared, then all interrupt sources including those selected as low priority in the IPRx register are disabled.

When both GIEH and GIEL bits are set, all interrupts selected as low priority sources are enabled.

A high priority interrupt will vector immediately to address 00 0008h and a low priority interrupt will vector to address 00 0018h.

14.3 Interrupt Response

When an interrupt is responded to, the Global Interrupt Enable bit is cleared to disable further interrupts. The GIE/GIEH bit is the Global Interrupt Enable when the IPEN bit is cleared. When the IPEN bit is set, enabling interrupt priority levels, the GIEH bit is the high priority Global Interrupt Enable and the GIEL bit is the low priority Global Interrupt Enable. High priority interrupt sources can interrupt a low priority interrupt. Low priority interrupts are not processed while high priority interrupts are in progress.

The return address is pushed onto the stack and the PC is loaded with the interrupt vector address (0008h or 0018h). Once in the Interrupt Service Routine, the source(s) of the interrupt can be determined by polling the interrupt flag bits in the INTCONx and PIRx registers. The interrupt flag bits must be cleared by software before re-enabling interrupts to avoid repeating the same interrupt.

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

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

Note: Do not use the `MOVFF` instruction to modify any of the interrupt control registers while **any** interrupt is enabled. Doing so may cause erratic microcontroller behavior.

14.8 Register Definitions: Interrupt Control

REGISTER 14-1: INTCON: INTERRUPT CONTROL REGISTER

R/W-0/0	R/W-0/0	R/W-0/0	U-0	U-0	R/W-1/1	R/W-1/1	R/W-1/1
GIE/GIEH	PEIE/GIEL	IPEN	—	—	INT2EDG	INT1EDG	INT0EDG
bit 7						bit 0	

Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

-n = Value at POR

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

- bit 7 **GIE/GIEH:** Global Interrupt Enable bit
If IPEN = 1:
1 = Enables all unmasked interrupts and cleared by hardware for high-priority interrupts only
0 = Disables all interrupts
If IPEN = 0:
1 = Enables all unmasked interrupts and cleared by hardware for all interrupts
0 = Disables all interrupts
- bit 6 **PEIE/GIEL:** Peripheral Interrupt Enable bit
If IPEN = 1:
1 = Enables all low-priority interrupts and cleared by hardware for low-priority interrupts only
0 = Disables all low-priority interrupts
If IPEN = 0:
1 = Enables all unmasked peripheral interrupts
0 = Disables all peripheral interrupts
- bit 5 **IPEN:** Interrupt Priority Enable bit
1 = Enable priority levels on interrupts
0 = Disable priority levels on interrupts
- bit 4-3 **Unimplemented:** Read as '0'
- bit 2 **INT2EDG:** External Interrupt 2 Edge Select bit
1 = Interrupt on rising edge of INT2 pin
0 = Interrupt on falling edge of INT2 pin
- bit 1 **INT1EDG:** External Interrupt 1 Edge Select bit
1 = Interrupt on rising edge of INT1 pin
0 = Interrupt on falling edge of INT1 pin
- bit 0 **INT0EDG:** External Interrupt 0 Edge Select bit
1 = Interrupt on rising edge of INT0 pin
0 = Interrupt on falling edge of INT0 pin

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

15.4 Register Definitions: Port Control

REGISTER 15-1: PORTx: PORTx REGISTER⁽¹⁾

R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u
Rx7	Rx6	Rx5	Rx4	Rx3	Rx2	Rx1	Rx0
bit 7							bit 0

Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

'1' = Bit is set

'0' = Bit is cleared

x = Bit is unknown

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

bit 7-0 **Rx<7:0>**: Rx7:Rx0 Port I/O Value bits

1 = Port pin is $\geq V_{IH}$

0 = Port pin is $\leq V_{IL}$

Note 1: Writes to PORTx are actually written to the corresponding LATx register.
Reads from PORTx register return actual I/O pin values.

TABLE 15-2: PORT REGISTERS

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PORTA	RA7	RA6	RA5	RA4	RA3	RA2	RA1	RA0
PORTB	RB7 ⁽¹⁾	RB6 ⁽¹⁾	RB5	RB4	RB3	RB2	RB1	RB0
PORTC	RC7	RC6	RC5	RC4	RC3	RC2	RC1	RC0
PORTE	—	—	—	—	RE3 ⁽²⁾	—	—	—

Note 1: Bits RB6 and RB7 read '1' while in Debug mode.

2: Bit PORTE3 is read-only, and will read '1' when MCLRE = 1 (Master Clear enabled).

21.5.2 SETUP FOR PWM OPERATION

The following steps should be taken when configuring the CCP module for standard PWM operation:

1. Use the desired output pin RxyPPS control to select CCPx as the source and disable the CCPx pin output driver by setting the associated TRIS bit.
2. Load the PR2 register with the PWM period value.
3. Configure the CCP module for the PWM mode by loading the CCPxCON register with the appropriate values.
4. Load the CCPRxL register, and the CCPRxH register with the PWM duty cycle value and configure the FMT bit of the CCPxCON register to set the proper register alignment.
5. Configure and start Timer2:
 - Clear the TMR2IF interrupt flag bit of the PIR4 register. See Note below.
 - Select the timer clock source to be as Fosc/4 using the TxCLKCON register. This is required for correct operation of the PWM module.
 - Configure the T2CKPS bits of the T2CON register with the Timer prescale value.
 - Enable the Timer by setting the ON bit of the T2CON register.
6. Enable PWM output pin:
 - Wait until the Timer overflows and the TMR2IF bit of the PIR4 register is set. See Note below.
 - Enable the CCPx pin output driver by clearing the associated TRIS bit.

Note: In order to send a complete duty cycle and period on the first PWM output, the above steps must be included in the setup sequence. If it is not critical to start with a complete PWM signal on the first output, then step 6 may be ignored.

21.5.3 TIMER2 TIMER RESOURCE

The PWM standard mode makes use of the 8-bit Timer2 timer resources to specify the PWM period.

21.5.4 PWM PERIOD

The PWM period is specified by the PR2 register of Timer2. The PWM period can be calculated using the formula of Equation 21-1.

EQUATION 21-1: PWM PERIOD

$$PWM\ Period = [(PR2) + 1] \cdot 4 \cdot T_{OSC} \cdot (TMR2\ Prescale\ Value)$$

Note 1: $T_{OSC} = 1/F_{OSC}$

When TMR2 is equal to PR2, the following three events occur on the next increment cycle:

- TMR2 is cleared
- The CCPx pin is set. (Exception: If the PWM duty cycle = 0%, the pin will not be set.)
- The PWM duty cycle is transferred from the CCPRxL/H register pair into a 10-bit buffer.

Note: The Timer postscaler (see **Section 20.4 “Timer2 Interrupt”**) is not used in the determination of the PWM frequency.

REGISTER 24-2: CWG1CON1: CWG CONTROL REGISTER 1

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

Legend:

R = Readable bit

W = Writable bit

U = Unimplemented bit, read as '0'

u = Bit is unchanged

x = Bit is unknown

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

'1' = Bit is set

'0' = Bit is cleared

q = Value depends on condition

bit 7-6	Unimplemented: Read as '0'
bit 5	IN: CWG Input Value bit (read-only)
bit 4	Unimplemented: Read as '0'
bit 3	POLD: CWG1D Output Polarity bit 1 = Signal output is inverted polarity 0 = Signal output is normal polarity
bit 2	POLC: CWG1C Output Polarity bit 1 = Signal output is inverted polarity 0 = Signal output is normal polarity
bit 1	POLB: CWG1B Output Polarity bit 1 = Signal output is inverted polarity 0 = Signal output is normal polarity
bit 0	POLA: CWG1A Output Polarity bit 1 = Signal output is inverted polarity 0 = Signal output is normal polarity

25.0 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 MDOOUT 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 25-1 shows a Simplified Block Diagram of the Data Signal Modulator peripheral.

FIGURE 26-7: SPI MODE WAVEFORM (SLAVE MODE WITH CKE = 0)

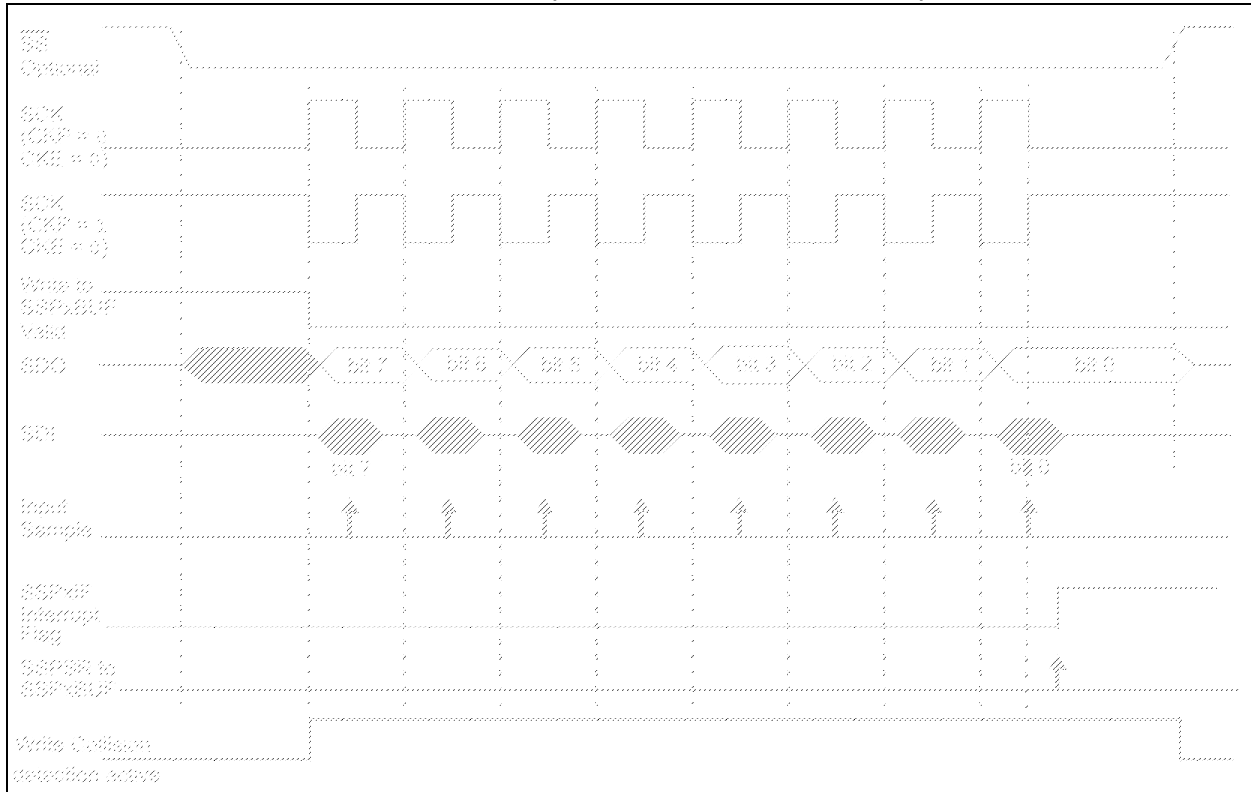


FIGURE 26-8: SPI MODE WAVEFORM (SLAVE MODE WITH CKE = 1)

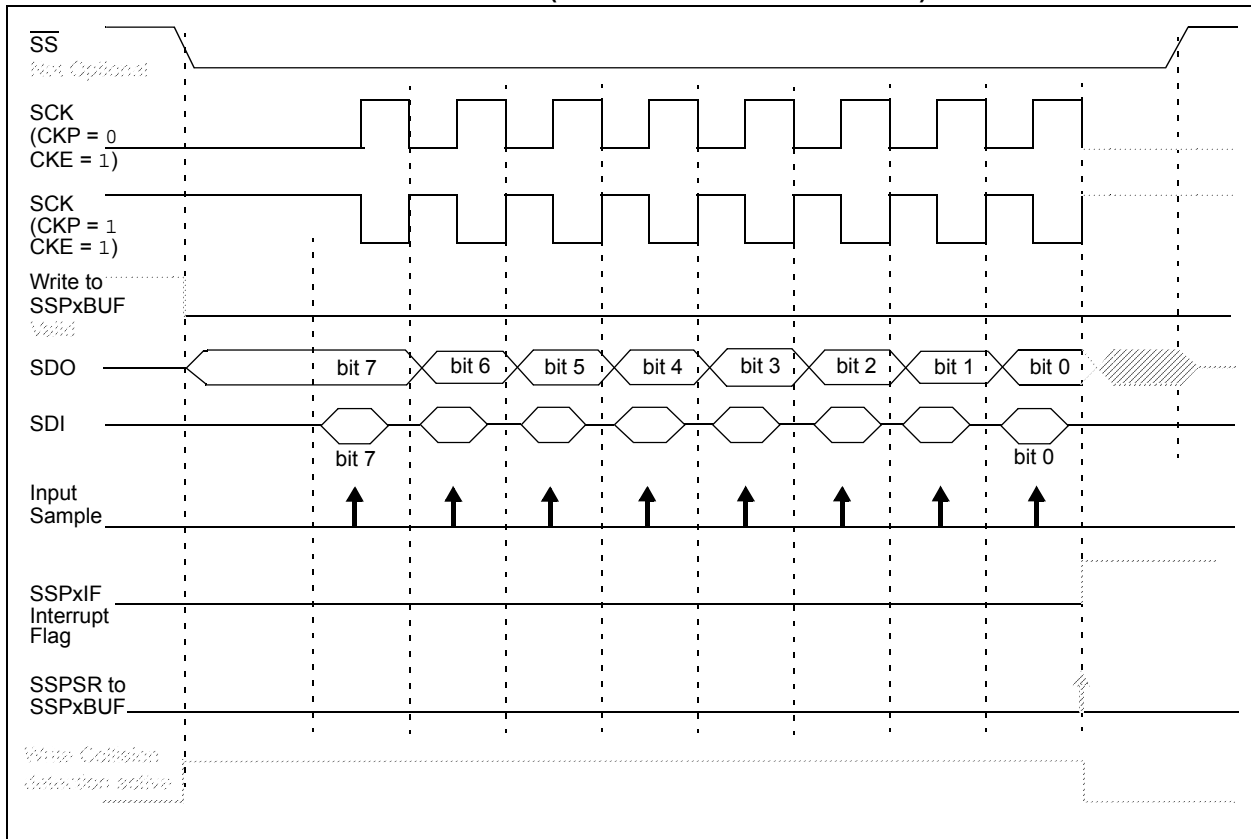
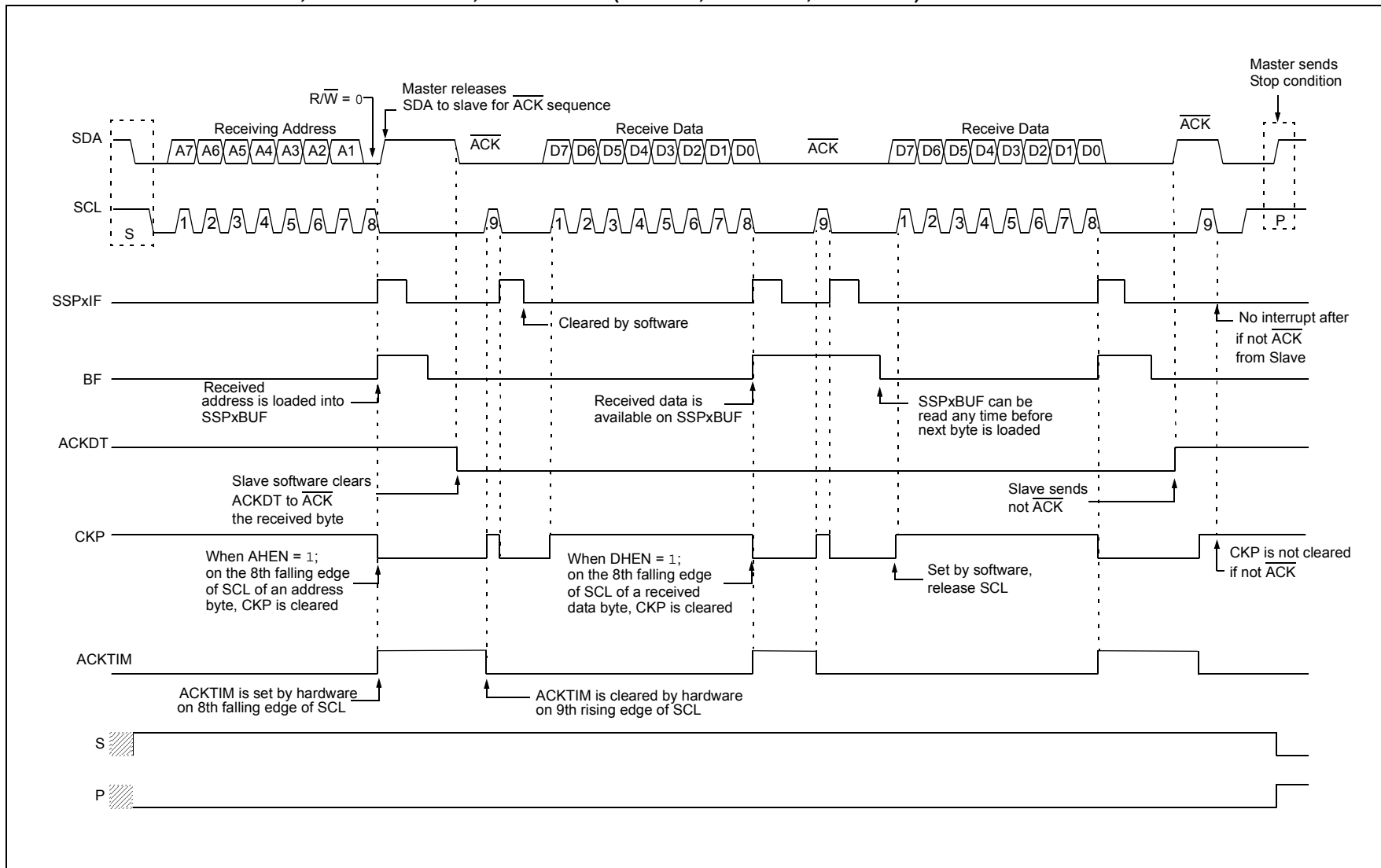


FIGURE 26-17: I²C SLAVE, 7-BIT ADDRESS, RECEPTION (SEN = 1, AHEN = 1, DHEN = 1)

After the write to the SSPxBUF, each bit of the address will be shifted out on the falling edge of SCL until all seven address bits and the R/W bit are completed. On the falling edge of the eighth clock, the master will release the SDA pin, allowing the slave to respond with an Acknowledge. On the falling edge of the ninth clock, the master will sample the SDA pin to see if the address was recognized by a slave. The status of the $\overline{\text{ACK}}$ bit is loaded into the ACKSTAT Status bit of the SSPxCON2 register. Following the falling edge of the ninth clock transmission of the address, the SSPxIF is set, the BF flag is cleared and the Baud Rate Generator is turned off until another write to the SSPxBUF takes place, holding SCL low and allowing SDA to float.

26.10.6.1 BF Status Flag

In Transmit mode, the BF bit of the SSPxSTAT register is set when the CPU writes to SSPxBUF and is cleared when all eight bits are shifted out.

26.10.6.2 WCOL Status Flag

If the user writes the SSPxBUF when a transmit is already in progress (i.e., SSPSR is still shifting out a data byte), the WCOL bit is set and the contents of the buffer are unchanged (the write does not occur).

The WCOL bit must be cleared by software before the next transmission.

26.10.6.3 ACKSTAT Status Flag

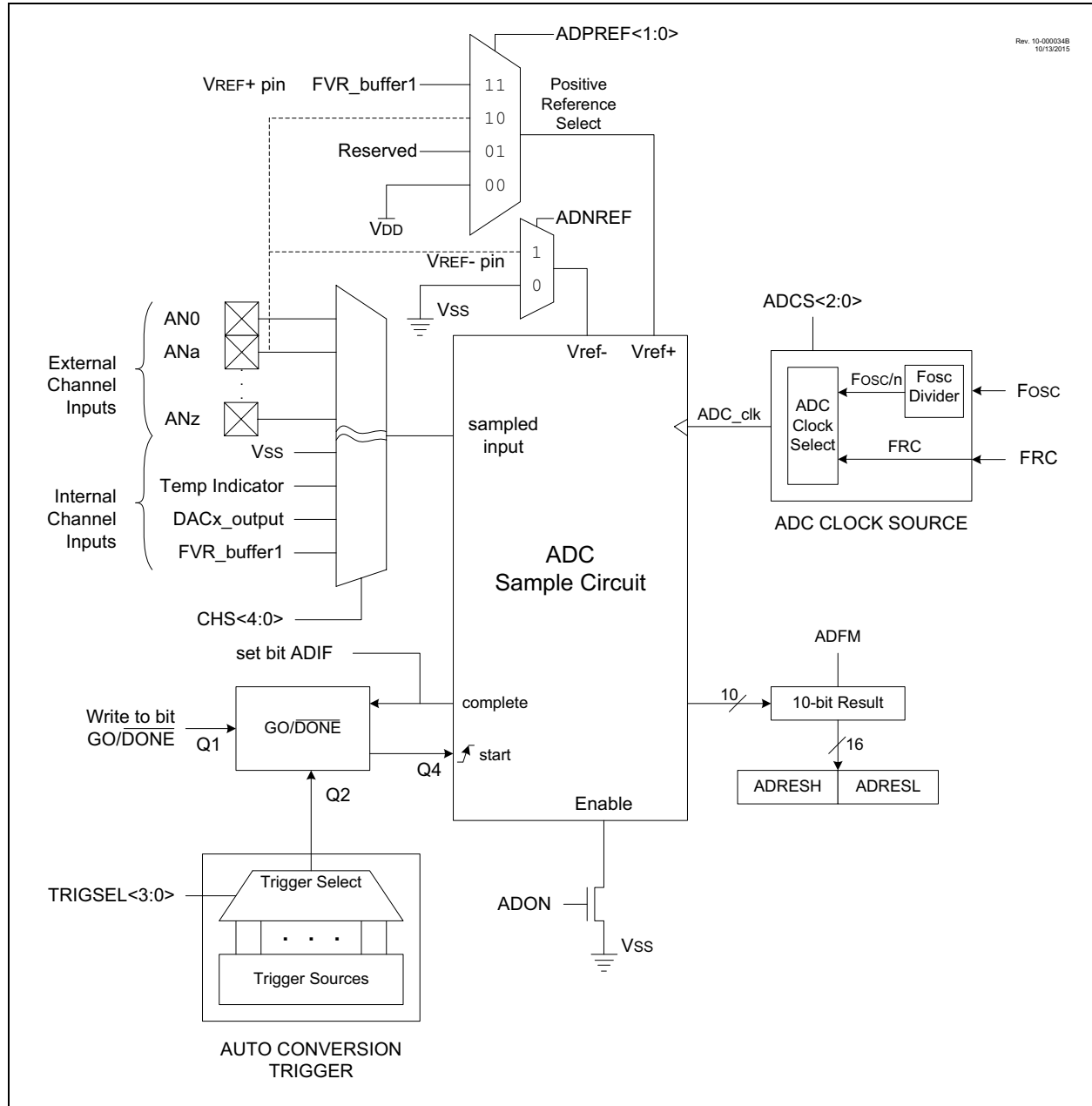
In Transmit mode, the ACKSTAT bit of the SSPxCON2 register is cleared when the slave has sent an Acknowledge ($\overline{\text{ACK}} = 0$) and is set when the slave does not Acknowledge ($\overline{\text{ACK}} = 1$). A slave sends an Acknowledge when it has recognized its address (including a general call), or when the slave has properly received its data.

26.10.6.4 Typical transmit sequence:

1. The user generates a Start condition by setting the SEN bit of the SSPxCON2 register.
2. SSPxIF is set by hardware on completion of the Start.
3. SSPxIF is cleared by software.
4. The MSSP module will wait the required start time before any other operation takes place.
5. The user loads the SSPxBUF with the slave address to transmit.
6. Address is shifted out the SDA pin until all eight bits are transmitted. Transmission begins as soon as SSPxBUF is written to.
7. The MSSP module shifts in the $\overline{\text{ACK}}$ bit from the slave device and writes its value into the ACKSTAT bit of the SSPxCON2 register.
8. The MSSP module generates an interrupt at the end of the ninth clock cycle by setting the SSPxIF bit.

9. The user loads the SSPxBUF with eight bits of data.
10. Data is shifted out the SDA pin until all eight bits are transmitted.
11. The MSSP module shifts in the $\overline{\text{ACK}}$ bit from the slave device and writes its value into the ACKSTAT bit of the SSPxCON2 register.
12. Steps 8-11 are repeated for all transmitted data bytes.
13. The user generates a Stop or Restart condition by setting the PEN or RSEN bits of the SSPxCON2 register. Interrupt is generated once the Stop/Restart condition is complete.

FIGURE 31-1: ADC² BLOCK DIAGRAM



REGISTER 31-4: ADCON3: ADC CONTROL REGISTER 3

U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W/HC-0	R/W-0/0	R/W-0/0	R/W-0/0
—	ADCALC<2:0>			ADSOI	ADTMD<2:0>		
bit 7				bit 0			

Legend:

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

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

bit 6-4 **ADCALC<2:0>:** ADC Error Calculation Mode Select bits

ADCALC	Action During 1st Precharge Stage		Application
	ADDSSEN = 0 Single-Sample Mode	ADDSSEN = 1 CVD Double-Sample Mode ⁽¹⁾	
111	Reserved	Reserved	Reserved
110	Reserved	Reserved	Reserved
101	ADLFTR-ADSTPT	ADFLTR-ADSTPT	Average/filtered value vs. setpoint
100	ADPREV-ADFLTR	ADPREV-ADFLTR	First derivative of filtered value ⁽³⁾ (negative)
011	Reserved	Reserved	Reserved
010	ADRES-ADFLTR	(ADRES-ADPREV)-ADFLTR	Actual result vs. averaged/filtered value
001	ADRES-ADSTPT	(ADRES-ADPREV)-ADSTPT	Actual result vs. setpoint
000	ADRES-ADPREV	ADRES-ADPREV	First derivative of single measurement ⁽²⁾
			Actual CVD result in CVD mode ⁽²⁾

bit 3 **ADSOI:** ADC Stop-on-Interrupt bit

If ADCONT = 1:

- 1 = ADGO is cleared when the threshold conditions are met, otherwise the conversion is retrigged
- 0 = ADGO is not cleared by hardware, must be cleared by software to stop retriggers

bit 2-0 **ADTMD<2:0>:** Threshold Interrupt Mode Select bits

- 111 = Interrupt regardless of threshold test results
- 110 = Interrupt if ADERR>ADUTH
- 101 = Interrupt if ADERR≤ADUTH
- 100 = Interrupt if ADERR<ADLTH or ADERR>ADUTH
- 011 = Interrupt if ADERR>ADLTH and ADERR<ADUTH
- 010 = Interrupt if ADERR≥ADLTH
- 001 = Interrupt if ADERR<ADLTH
- 000 = Never interrupt

Note 1: When ADPSIS = 0, the value of ADRES-ADPREV) is the value of (S2-S1) from Table 31-3.

2: When ADPSIS = 0

3: When ADPSIS = 1.

REGISTER 31-27: ADERRL: ADC SETPOINT ERROR LOW BYTE REGISTER

R-x	R-x	R-x	R-x	R-x	R-x	R-x	R-x
ADERR<7:0>							
bit 7				bit 0			

Legend:

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

bit 7-0 **ADERR<7:0>**: ADC Setpoint Error LSB. Lower byte of ADC Setpoint Error calculation is determined by ADCALC bits of ADCON3, see Register 23-1 for more details.

REGISTER 31-28: ADLTHH: ADC LOWER THRESHOLD HIGH BYTE REGISTER

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

Legend:

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

bit 7-0 **ADLTH<15:8>**: ADC Lower Threshold MSB. ADLTH and ADUTH are compared with ADERR to set the ADUTHR and ADLTHR bits of ADSTAT. Depending on the setting of ADTMD, an interrupt may be triggered by the results of this comparison.


REGISTER 31-29: ADLTHL: ADC LOWER THRESHOLD LOW BYTE REGISTER

R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x	R/W-x/x
ADLTH<7:0>							
bit 7				bit 0			

Legend:

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

bit 7-0 **ADLTH<7:0>**: ADC Lower Threshold LSB. ADLTH and ADUTH are compared with ADERR to set the ADUTHR and ADLTHR bits of ADSTAT. Depending on the setting of ADTMD, an interrupt may be triggered by the results of this comparison.

RRNCF		Rotate Right f (No Carry)											
Syntax:	RRNCF f {,d {,a}}												
Operands:	$0 \leq f \leq 255$ $d \in [0,1]$ $a \in [0,1]$												
Operation:	$(f < n) \rightarrow \text{dest} < n - 1 >$, $(f < 0) \rightarrow \text{dest} < 7 >$												
Status Affected:	N, Z												
Encoding:	<table><tr><td>0100</td><td>00da</td><td>ffff</td><td>ffff</td></tr></table>					0100	00da	ffff	ffff				
0100	00da	ffff	ffff										
Description:	<p>The contents of register 'f' are rotated one bit to the right. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default). If 'a' is '0', the Access Bank will be selected (default), overriding the BSR value. If 'a' is '1', then the bank will be selected as per the BSR value. If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See Section 35.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode” for details.</p> <div></div>												
Words:	1												
Cycles:	1												
Q Cycle Activity:	<table><tr><th>Q1</th><th>Q2</th><th>Q3</th><th>Q4</th></tr><tr><td>Decode</td><td>Read register 'f'</td><td>Process Data</td><td>Write to destination</td></tr></table>					Q1	Q2	Q3	Q4	Decode	Read register 'f'	Process Data	Write to destination
Q1	Q2	Q3	Q4										
Decode	Read register 'f'	Process Data	Write to destination										

Example 1: RRNCF REG, 1, 0

Before Instruction
REG = 1101 0111
After Instruction
REG = 1110 1011

Example 2: RRNCF REG, 0, 0

Before Instruction
W = ?
REG = 1101 0111
After Instruction
W = 1110 1011
REG = 1101 0111

SETF		Set f							
Syntax:	SETF f {,a}								
Operands:	$0 \leq f \leq 255$ $a \in [0,1]$								
Operation:	FFh \rightarrow f								
Status Affected:	None								
Encoding:	<table border="1"><tr><td>0110</td><td>100a</td><td>ffff</td><td>ffff</td></tr></table>					0110	100a	ffff	ffff
0110	100a	ffff	ffff						
Description:	<p>The contents of the specified register are set to FFh.</p> <p>If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.</p> <p>If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See Section 35.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode" for details.</p>								
Words:	1								
Cycles:	1								
Q Cycle Activity:									
	Q1	Q2	Q3	Q4					
	Decode	Read register 'f'	Process Data	Write register 'f'					

Example: SETF REG, 1

Before Instruction
REG = 5Ah
After Instruction
REG = FFh

SUBFSR Subtract Literal from FSR

Syntax: SUBFSR f, k

Operands: $0 \leq k \leq 63$

$f \in [0, 1, 2]$

Operation: $FSR(f) - k \rightarrow FSRf$

Status Affected: None

Encoding:

1110	1001	ffkk	kkkk
------	------	------	------

Description: The 6-bit literal 'k' is subtracted from the contents of the FSR specified by 'f'.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: SUBFSR 2, 23h

Before Instruction

FSR2 = 03FFh

After Instruction

FSR2 = 03DCh

SUBULNK Subtract Literal from FSR2 and Return

Syntax: SUBULNK k

Operands: $0 \leq k \leq 63$

Operation: $FSR2 - k \rightarrow FSR2$
(TOS) $\rightarrow PC$

Status Affected: None

Encoding:

1110	1001	11kk	kkkk
------	------	------	------

Description: The 6-bit literal 'k' is subtracted from the contents of the FSR2. A RETURN is then executed by loading the PC with the TOS. The instruction takes two cycles to execute; a NOP is performed during the second cycle.

This may be thought of as a special case of the SUBFSR instruction, where $f = 3$ (binary '11'); it operates only on FSR2.

Words: 1

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination
No Operation	No Operation	No Operation	No Operation

Example: SUBULNK 23h

Before Instruction

FSR2 = 03FFh

PC = 0100h

After Instruction

FSR2 = 03DCh

PC = (TOS)