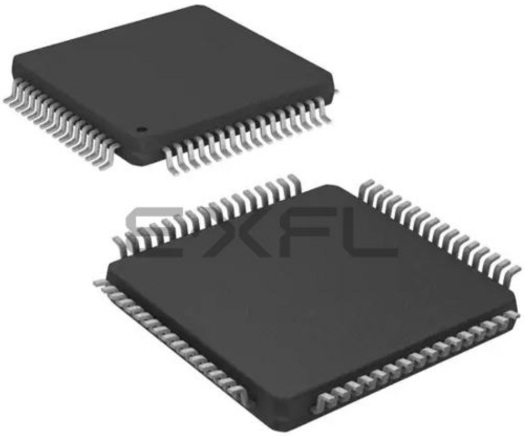
Microchip Technology - PIC18F66K22-I/PT Datasheet



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, LVD, POR, PWM, WDT
Number of I/O	53
Program Memory Size	64KB (32K x 16)
Program Memory Type	FLASH
EEPROM Size	1K x 8
RAM Size	4K x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 5.5V
Data Converters	A/D 16x12b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	64-TQFP
Supplier Device Package	64-TQFP (10x10)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18f66k22-i-pt

1.3 Details on Individual Family Members

Devices in the PIC18F87K22 family are available in 64-pin and 80-pin packages. Block diagrams for the two groups are shown in Figure 1-1 and Figure 1-2.

The devices are differentiated from each other in these ways:

- Flash Program Memory:
  - PIC18FX5K22 (PIC18F65K22 and PIC18F85K22) – 32 Kbytes
  - PIC18FX6K22 (PIC18F66K22 and PIC18F86K22) – 64 Kbytes
  - PIC18FX7K22 (PIC18F67K22 and PIC18F87K22) – 128 Kbytes
- Data RAM:
  - All devices except PIC18FX5K22 – 4 Kbytes
  - PIC18FX5K22 – 2 Kbytes
- I/O Ports:
  - PIC18F6XK22 (64-pin devices) – 7 bidirectional ports
  - PIC18F8XK22 (80-pin devices) – 9 bidirectional ports
- CCP modules:
  - PIC18FX5K22 (PIC18F65K22 and PIC18F85K22) – 5 CCP modules
  - PIC18FX6K22 and PIC18FX7K22 (PIC18F66K22, PIC18F86K22, PIC18F67K22, and PIC18F87K22) – 7 CCP modules
- Timer modules:
  - PIC18FX5K22 (PIC18F65K22 and PIC18F85K22) – Four 8-bit timer/counters and four 16-bit timer/counters
  - PIC18FX6K22 and PIC18FX7K22 (PIC18F66K22, PIC18F86K22, PIC18F67K22, and PIC18F87K22) – Six 8-bit timer/counters and five 16-bit timer/counters
- A/D Channels:
  - PIC18F6XK22 (64-pin devices) – 24 channels
  - PIC18F8XK22 (80-pin devices) – 16 channels

All other features for devices in this family are identical. These are summarized in Table 1-1 and Table 1-2.

The pinouts for all devices are listed in Table 1-3 and Table 1-4.

TABLE 1-4: PIC18F8XK22 PINOUT I/O DESCRIPTIONS

Pin Name	Pin Number	Pin Type	Buffer Type	Description
	TQFP			
$\overline{\text{MCLR}}$/RG5	9			Master Clear (input) or programming voltage (input).
RG5		I	ST	This pin is an active-low Reset to the device.
$\overline{\text{MCLR}}$		I	ST	General purpose, input only pin.
OSC1/CLKI/RA7	49			Oscillator crystal or external clock input.
OSC1		I	CMOS	Oscillator crystal input.
CLKI		I	CMOS	External clock source input. Always associated with pin function, OSC1. (See related OSC1/CLKI, OSC2/CLKO pins.)
RA7		I/O	TTL	General purpose I/O pin.
OSC2/CLKO/RA6	50			Oscillator crystal or clock output.
OSC2		O	—	Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode.
CLKO		O	—	In certain oscillator modes, OSC2 pin outputs CLKO, which has 1/4 the frequency of OSC1 and denotes the instruction cycle rate.
RA6		I/O	TTL	General purpose I/O pin.

Legend:

TTL	= TTL compatible input	CMOS	= CMOS compatible input or output
ST	= Schmitt Trigger input with CMOS levels	Analog	= Analog input
I	= Input	O	= Output
P	= Power	OD	= Open-Drain (no P diode to VDD)
I^2C	= I^2C™/SMBus		

Note 1: Default assignment for ECCP2 when the CCP2MX Configuration bit is set.

2: Alternate assignment for ECCP2 when the CCP2MX Configuration bit is cleared.

3: Not available on PIC18F65K22 and PIC18F85K22 devices.

4: PSP is available only in Microcontroller mode.

5: The CC6, CCP7, CCP8 and CCP9 pin placement depends on the setting of the ECCPMX Configuration bit (CONFIG3H<1>).

6.4.3.2 FSR Registers and POSTINC, POSTDEC, PREINC and PLUSW

In addition to the INDF operand, each FSR register pair also has four additional indirect operands. Like INDF, these are “virtual” registers that cannot be indirectly read or written to. Accessing these registers actually accesses the associated FSR register pair, but also performs a specific action on its stored value.

These operands are:

- POSTDEC – Accesses the FSR value, then automatically decrements it by ‘1’ afterwards
- POSTINC – Accesses the FSR value, then automatically increments it by ‘1’ afterwards
- PREINC – Increments the FSR value by ‘1’, then uses it in the operation
- PLUSW – Adds the signed value of the W register (range of -127 to 128) to that of the FSR and uses the new value in the operation

In this context, accessing an INDF register uses the value in the FSR registers without changing them. Similarly, accessing a PLUSW register gives the FSR value, offset by the value in the W register, with neither value actually changed in the operation. Accessing the other virtual registers changes the value of the FSR registers.

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

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

6.4.3.3 Operations by FSRs on FSRs

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

As a specific case, assume that the FSR0H:FSR0L registers contain FE7h, the address of INDF1. Attempts to read the value of the INDF1, using INDF0 as an operand, will return 00h. Attempts to write to INDF1, using INDF0 as the operand, will result in a NOP.

On the other hand, using the virtual registers to write to an FSR pair may not occur as planned. In these cases, the value will be written to the FSR pair, but without any incrementing or decrementing. Thus, writing to INDF2 or POSTDEC2 will write the same value to the FSR2H:FSR2L.

Since the FSRs are physical registers mapped in the SFR space, they can be manipulated through all direct operations. Users should proceed cautiously when working on these registers, however, particularly if their code uses Indirect Addressing.

Similarly, operations by Indirect Addressing are generally permitted on all other SFRs. Users should exercise the appropriate caution, so that they do not inadvertently change settings that might affect the operation of the device.

6.5 Program Memory and the Extended Instruction Set

The operation of program memory is unaffected by the use of the extended instruction set.

Enabling the extended instruction set adds five additional two-word commands to the existing PIC18 instruction set: ADDFSR, CALLW, MOVSF, MOVSS and SUBFSR. These instructions are executed as described in **Section 6.2.4 “Two-Word Instructions”**.

TABLE 12-16: SUMMARY OF REGISTERS ASSOCIATED WITH PORTH

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PORTH[1]	RH7	RH6	RH5	RH4	RH3	RH2	RH1	RH0
LATH[1]	LATH7	LATH6	LATH5	LATH4	LATH3	LATH2	LATH1	LATH0
TRISH[1]	TRISH7	TRISH6	TRISH5	TRISH4	TRISH3	TRISH2	TRISH1	TRISH0
ANCON1	ANSEL15	ANSEL14	ANSEL13	ANSEL12	ANSEL11	ANSEL10	ANSEL9	ANSEL8
ANCON2	ANSEL23	ANSEL22	ANSEL21	ANSEL20	ANSEL19	ANSEL18	ANSEL17	ANSEL16
ODCON2	CCP10OD[2]	CCP9OD[2]	CCP8OD	CCP7OD	CCP6OD	CCP5OD	CCP4OD	CCP3OD

Note 1: Unimplemented on 64-pin devices (PIC18F6XK22), read as ‘0’.

2: Unimplemented on PIC18FX5K22 devices, read as ‘0’.

12.10 PORTJ, TRISJ and LATJ Registers

Note: PORTJ is available only on 80-pin devices.

PORTJ is an 8-bit wide, bidirectional port. The corresponding Data Direction and Output Latch registers are TRISJ and LATJ.

All pins on PORTJ are implemented with Schmitt Trigger input buffers. Each pin is individually configurable as an input or output.

Note: These pins are configured as digital inputs on any device Reset.

When the external memory interface is enabled, all of the PORTJ pins function as control outputs for the interface. This occurs automatically when the interface is enabled by clearing the EBDIS control bit (MEMCON<7>). The TRISJ bits are also overridden.

Each of the PORTJ pins has a weak internal pull-up. The pull-ups are provided to keep the inputs at a known state for the external memory interface while powering up. A single control bit can turn off all the pull-ups. This is performed by clearing bit, RJPU (PADCFG1<5>). The weak pull-up is automatically turned off when the port pin is configured as an output. The pull-ups are disabled on any device Reset.

EXAMPLE 12-9: INITIALIZING PORTJ

```
CLRF    PORTJ   ; Initialize PORTJ by
                ; clearing output latches
CLRF    LATJ    ; Alternate method
                ; to clear output latches
MOVLW   0CFh    ; Value used to
                ; initialize data
                ; direction
MOVWF   TRISJ   ; Set RJ3:RJ0 as inputs
                ; RJ5:RJ4 as output
                ; RJ7:RJ6 as inputs
```

14.8.3 TIMER1 GATE TOGGLE MODE

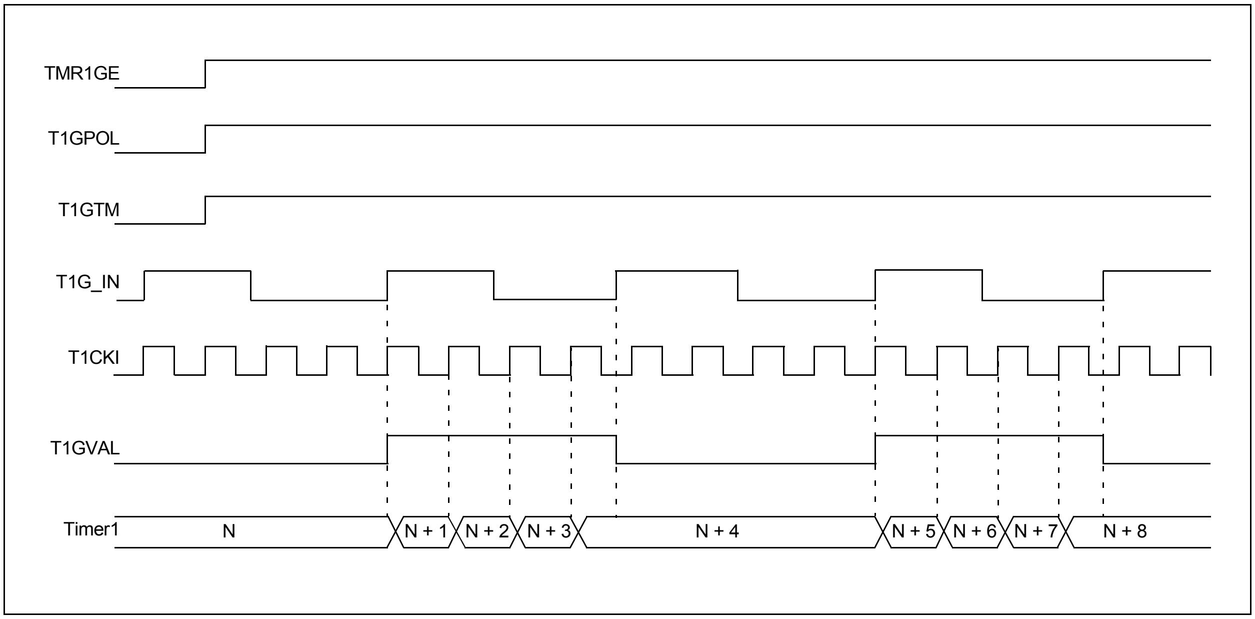
When Timer1 Gate Toggle mode is enabled, it is possible to measure the full cycle length of a Timer1 gate signal, as opposed to the duration of a single level pulse.

The Timer1 gate source is routed through a flip-flop that changes state on every incrementing edge of the signal. (For timing details, see Figure 14-5.)

The T1GVAL bit (T1GCON<2>) indicates when the Toggled mode is active and the timer is counting.

The Timer1 Gate Toggle mode is enabled by setting the T1GTM bit (T1GCON<5>). When T1GTM is cleared, the flip-flop is cleared and held clear. This is necessary in order to control which edge is measured.

FIGURE 14-5: TIMER1 GATE TOGGLE MODE



17.0 TIMER4/6/8/10/12 MODULES

The Timer4/6/8/10/12 timer modules have the following features:

- Eight-bit Timer register (TMRx)
- Eight-bit Period register (PRx)
- Readable and writable (all registers)
- Software programmable prescaler (1:1, 1:4, 1:16)
- Software programmable postscaler (1:1 to 1:16)
- Interrupt on TMRx match of PRx

Timer10 and Timer12 are unimplemented for devices with a program memory of 32 Kbytes (PIC18FX5K22).

> **Note:** Throughout this section, generic references are used for register and bit names that are the same, except for an 'x' variable that indicates the item's association with the Timer4, Timer6, Timer8, Timer10 or Timer12 module. For example, the control register is named TxCON and refers to T4CON, T6CON, T8CON, T10CON and T12CON.

The Timer4/6/8/10/12 modules have a control register, which is shown in Register 17-1. Timer4/6/8/10/12 can be shut off by clearing control bit, TMRxON (TxCON<2>), to minimize power consumption. The prescaler and postscaler selection of Timer4/6/8/10/12 are also controlled by this register. Figure 17-1 is a simplified block diagram of the Timer4/6/8/10/12 modules.

17.1 Timer4/6/8/10/12 Operation

Timer4/6/8/10/12 can be used as the PWM time base for the PWM mode of the ECCP modules. The TMRx registers are readable and writable, and are cleared on any device Reset. The input clock (FOSC/4) has a prescale option of 1:1, 1:4 or 1:16, selected by control bits, TxCKPS<1:0> (TxCON<1:0>). The match output of TMRx goes through a four-bit postscaler (that gives a 1:1 to 1:16 inclusive scaling) to generate a TMRx interrupt, latched in the flag bit, TMRxIF. Table 17-1 shows each module's flag bit.

TABLE 17-1: TIMER4/6/8/10/12 FLAG BITS

Timer Module	Flag Bit PIR5<x>	Timer Module	Flag Bit PIR5<x>
4	0	10	5
6	2	12	6
8	4		

The interrupt can be enabled or disabled by setting or clearing the Timerx Interrupt Enable bit (TMRxIE), shown in Table 17-2.

TABLE 17-2: TIMER4/6/8/10/12 INTERRUPT ENABLE BITS

Timer Module	Flag Bit PIE5<x>	Timer Module	Flag Bit PIE5<x>
4	0	10	5
6	2	12	6
8	4		

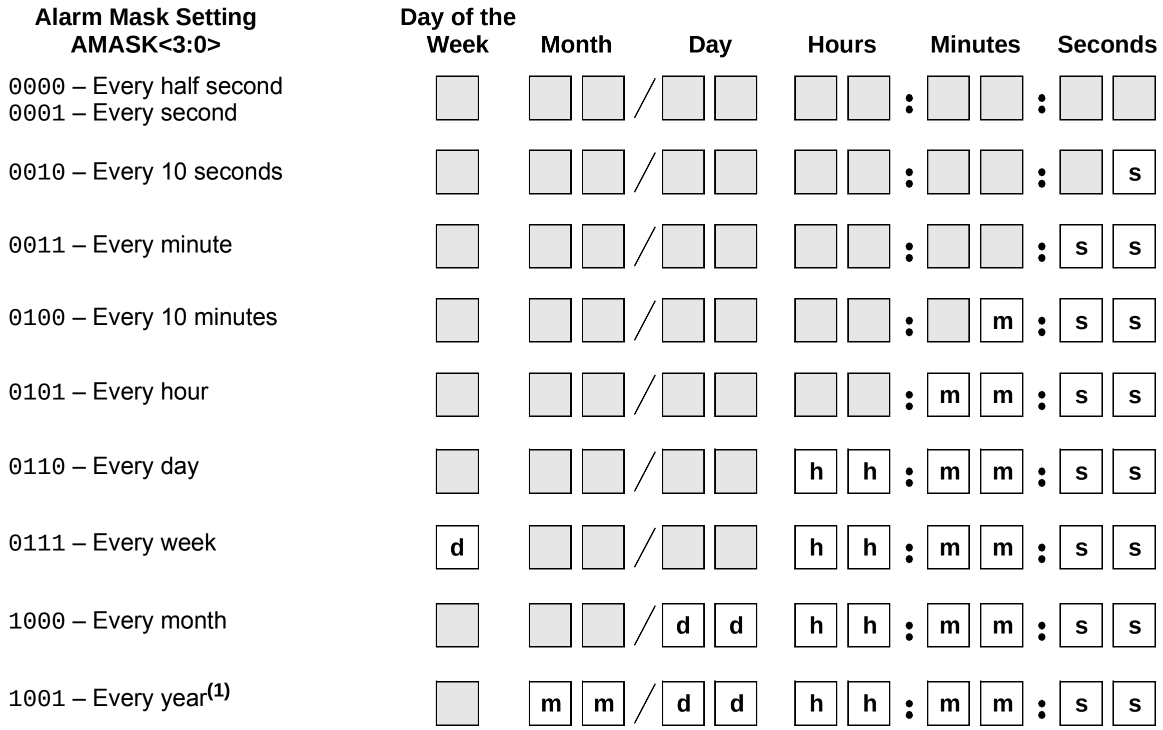
The prescaler and postscaler counters are cleared when any of the following occurs:

- A write to the TMRx register
- A write to the TxCON register
- Any device Reset – Power-on Reset (POR), $\overline{\text{MCLR}}$ Reset, Watchdog Timer Reset (WDTR) or Brown-out Reset (BOR)

A TMRx is not cleared when a TxCON is written.

> **Note:** The CCP and ECCP modules use Timers, 1 through 8, for some modes. The assignment of a particular timer to a CCP/ECCP module is determined by the Timer to CCP enable bits in the CCPTMRSx registers. For more details, see Register 19-2, Register 19-3 and Register 20-2.

FIGURE 18-5: ALARM MASK SETTINGS



Alarm Mask Setting AMASK<3:0>	Day of the Week	Month	Day	Hours	Minutes	Seconds
0000 – Every half second 0001 – Every second		/		:	:	
0010 – Every 10 seconds		/		:	:	s
0011 – Every minute		/		:	:	s s
0100 – Every 10 minutes		/		:	m :	s s
0101 – Every hour		/		:	m m :	s s
0110 – Every day		/		h h :	m m :	s s
0111 – Every week	d	/		h h :	m m :	s s
1000 – Every month		/	d d	h h :	m m :	s s
1001 – Every year[(1)]		m m /	d d	h h :	m m :	s s

Note 1: Annually, except when configured for February 29.

When ALRMCFG = 00 and the CHIME bit = 0 (ALRMCFG<6>), the repeat function is disabled and only a single alarm will occur. The alarm can be repeated up to 255 times by loading the ALRMRPT register with FFh.

After each alarm is issued, the ALRMRPT register is decremented by one. Once the register has reached ‘00’, the alarm will be issued one last time.

After the alarm is issued a last time, the ALRMEN bit is cleared automatically and the alarm is turned off. Indefinite repetition of the alarm can occur if the CHIME bit = 1.

When CHIME = 1, the alarm is not disabled when the ALRMRPT register reaches ‘00’, but it rolls over to FF and continues counting indefinitely.

18.3.2 ALARM INTERRUPT

At every alarm event, an interrupt is generated. Additionally, an alarm pulse output is provided that operates at half the frequency of the alarm.

The alarm pulse output is completely synchronous with the RTCC clock and can be used as a trigger clock to other peripherals. This output is available on the RTCC pin. The output pulse is a clock with a 50% duty cycle and a frequency half that of the alarm event (see Figure 18-6).

The RTCC pin also can output the seconds clock. The user can select between the alarm pulse, generated by the RTCC module, or the seconds clock output.

The RTSECSEL<1:0> bits (PADCFG1<2:1>) select between these two outputs:

- Alarm pulse – RTSECSEL<1:0> = 00
- Seconds clock – RTSECSEL<1:0> = 01

19.0 CAPTURE/COMPARE/PWM (CCP) MODULES

PIC18F87K22 family devices have seven CCP (Capture/Compare/PWM) modules, designated CCP4 through CCP10. All the modules implement standard Capture, Compare and Pulse-Width Modulation (PWM) modes.

Note:	Throughout this section, generic references are used for register and bit names that are the same, except for an ‘x’ variable that indicates the item’s association with the specific CCP module. For example, the control register is named CCPxCON and refers to CCP4CON through CCP10CON.

Each CCP module contains a 16-bit register that can operate as a 16-bit Capture register, a 16-bit Compare register or a PWM Master/Slave Duty Cycle register. For the sake of clarity, all CCP module operation in the following sections is described with respect to CCP4, but is equally applicable to CCP5 through CCP10.

Note:	The CCP9 and CCP10 modules are disabled on the devices with 32 Kbytes of program memory (PIC18FX5K22).

REGISTER 19-1: CCPxCON: CCPx CONTROL REGISTER (CCP4-CCP10 MODULES)(1)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
PxM1	PxM0	DCxB1	DCxB0	CCPxM3(2)	CCPxM2(2)	CCPxM1(2)	CCPxM0(2)
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 7-6 **PxM<1:0>:** PWM Output Configuration bits

If CCPxM<3:2> = 00, 01, 10:
xx = PxA is assigned as a capture/compare input/output; PxB, PxC and PxD are assigned as port pins

If CCPxM<3:2> = 11:
00 = Single output: PxA, PxB, PxC and PxD are controlled by steering
01 = Full-bridge output forward: PxD is modulated; PxA is active; PxB, PxC are inactive
10 = Half-bridge output: PxA, PxB are modulated with dead-band control; PxC and PxD are assigned as port pins
11 = Full-bridge output reverse: PxB is modulated; PxC is active; PxA and PxD are inactive

bit 5-4 **DCxB<1:0>:** PWM Duty Cycle bit 1 and bit 0 for CCPx Module

Capture mode:
Unused.

Compare mode:
Unused.

PWM mode:
These bits are the two Least Significant bits (bit 1 and bit 0) of the 10-bit PWM duty cycle. The eight Most Significant bits (DCx<9:2>) of the duty cycle are found in CCPRxL.

Note 1: The CCP9 and CCP10 modules are not available on devices with 32 Kbytes of program memory (PIC18FX5K22).

2: CCPxM<3:0> = 1011 will only reset the timer and not start AN A/D conversion on CCPx match.

20.2 Capture Mode

In Capture mode, the CCPRxH:CCPRxL register pair captures the 16-bit value of the TMR1 or TMR3 registers when an event occurs on the corresponding ECCPx pin. An event is defined as one of the following:

- Every falling edge
- Every rising edge
- Every fourth rising edge
- Every 16th rising edge

The event is selected by the mode select bits, CCPxM<3:0> (CCPxCON<3:0>). When a capture is made, the interrupt request flag bit, CCPxIF, is set (see Table 20-2). The flag must be cleared by software. If another capture occurs before the value in the CCPRxH/L register is read, the old captured value is overwritten by the new captured value.

TABLE 20-2: ECCP1/2/3 INTERRUPT FLAG BITS

ECCP Module	Flag Bit
1	PIR3<1>
2	PIR3<2>
3	PIR4<0>

20.2.1 ECCP PIN CONFIGURATION

In Capture mode, the appropriate ECCPx pin should be configured as an input by setting the corresponding TRIS direction bit.

Note: If the ECCPx pin is configured as an output, a write to the port can cause a capture condition.

20.2.2 TIMER1/2/3/4/6/8/10/12 MODE SELECTION

The timers that are to be used with the capture feature (Timer1 2, 3, 4, 6, 8, 10 or 12) must be running in Timer mode or Synchronized Counter mode. In Asynchronous Counter mode, the capture operation may not work. The timer to be used with each ECCP module is selected in the CCPTMRS0 register (Register 20-2).

20.2.3 SOFTWARE INTERRUPT

When the Capture mode is changed, a false capture interrupt may be generated. The user should keep the CCPxIE interrupt enable bit clear to avoid false interrupts. The interrupt flag bit, CCPxIF, should also be cleared following any such change in operating mode.

20.2.4 ECCP PRESCALER

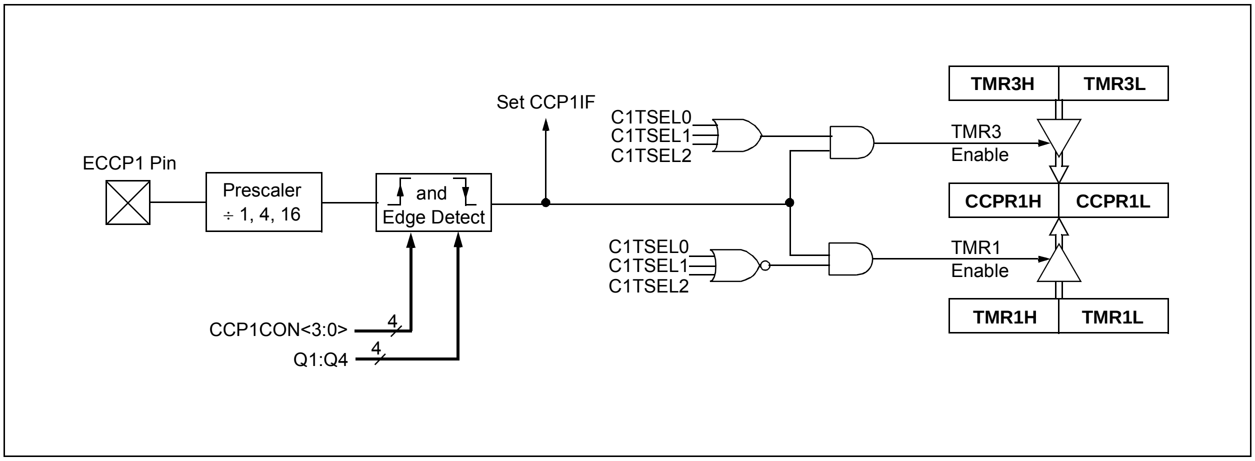
There are four prescaler settings in Capture mode; they are specified as part of the operating mode selected by the mode select bits (CCPxM<3:0>). Whenever the ECCP module is turned off, or Capture mode is disabled, the prescaler counter is cleared. This means that any Reset will clear the prescaler counter.

Switching from one capture prescaler to another may generate an interrupt. Also, the prescaler counter will not be cleared; therefore, the first capture may be from a non-zero prescaler. Example 20-1 provides the recommended method for switching between capture prescalers. This example also clears the prescaler counter and will not generate the “false” interrupt.

EXAMPLE 20-1: CHANGING BETWEEN CAPTURE PRESCALERS

```
CLRF   CCP1CON     ; Turn ECCP module off
MOVLW  NEW_CAPT_PS ; Load WREG with the
                   ; new prescaler mode
                   ; value and ECCP ON
MOVWF  CCP1CON     ; Load CCP1CON with
                   ; this value
```

FIGURE 20-1: CAPTURE MODE OPERATION BLOCK DIAGRAM



21.4 I²C Mode

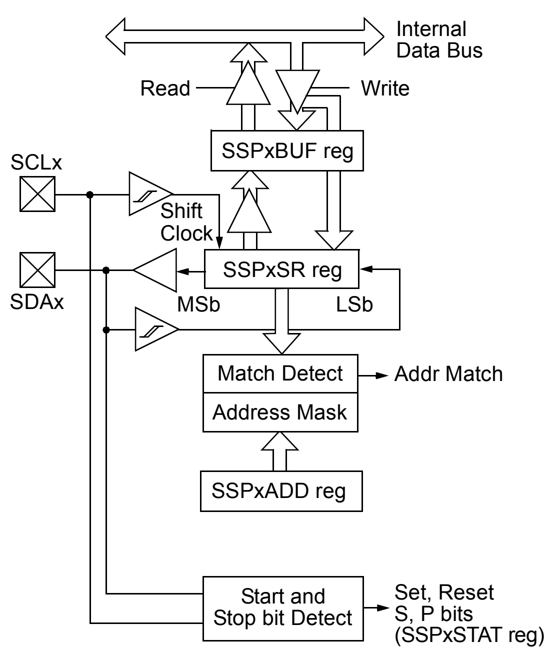
The MSSP module in I²C mode fully implements all master and slave functions (including general call support), and provides interrupts on Start and Stop bits in hardware to determine a free bus (multi-master function). The MSSP module implements the standard mode specifications, as well as 7-bit and 10-bit addressing.

Two pins are used for data transfer:

- Serial Clock (SCLx) – RC3/SCK1/SCL1 or RD6/SCK2/SCL2
- Serial Data (SDAx) – RC4/SDI1/SDA1 or RD5/SDI2/SDA2

The user must configure these pins as inputs by setting the associated TRIS bits.

FIGURE 21-7: MSSP BLOCK DIAGRAM (I²C™ MODE)



Note: Only port I/O names are used in this diagram for the sake of brevity. Refer to the text for a full list of multiplexed functions.

21.4.1 REGISTERS

The MSSP module has seven registers for I²C operation. These are:

- MSSPx Control Register 1 (SSPxCON1)
- MSSPx Control Register 2 (SSPxCON2)
- MSSPx Status Register (SSPxSTAT)
- Serial Receive/Transmit Buffer Register (SSPxBUF)
- MSSPx Shift Register (SSPxSR) – Not directly accessible
- MSSPx Address Register (SSPxADD)
- I²C Slave Address Mask Register (SSPxMSK)

SSPxCON1, SSPxCON2 and SSPxSTAT are the control and status registers in I²C mode operation. The SSPxCON1 and SSPxCON2 registers are readable and writable. The lower 6 bits of the SSPxSTAT are read-only. The upper two bits of the SSPxSTAT are read/write.

SSPxSR is the shift register used for shifting data in or out. SSPxBUF is the buffer register to which data bytes are written to or read from.

SSPxADD contains the slave device address when the MSSP is configured in I²C Slave mode. When the MSSP is configured in Master mode, the lower seven bits of SSPxADD act as the Baud Rate Generator reload value.

SSPxMSK holds the slave address mask value when the module is configured for 7-Bit Address Masking mode. While it is a separate register, it shares the same SFR address as SSPxADD; it is only accessible when the SSPM<3:0> bits are specifically set to permit access. Additional details are provided in **Section 21.4.3.4 “7-Bit Address Masking Mode”**.

In receive operations, SSPxSR and SSPxBUF together, create a double-buffered receiver. When SSPxSR receives a complete byte, it is transferred to SSPxBUF and the SSPxIF interrupt is set.

During transmission, the SSPxBUF is not double-buffered. A write to SSPxBUF will write to both SSPxBUF and SSPxSR.

21.4.7 BAUD RATE

In I^{2}C Master mode, the Baud Rate Generator (BRG) reload value is placed in the lower 7 bits of the SSPxADD register (Figure 21-19). When a write occurs to SSPxBUF, the Baud Rate Generator will automatically begin counting. The BRG counts down to 0 and stops until another reload has taken place. The BRG count is decremented twice per instruction cycle (TCY) on the Q2 and Q4 clocks. In I^{2}C Master mode, the BRG is reloaded automatically.

Once the given operation is complete (i.e., transmission of the last data bit is followed by $\overline{\text{ACK}}$), the internal clock will automatically stop counting and the SCLx pin will remain in its last state.

Table 21-3 demonstrates clock rates based on instruction cycles and the BRG value loaded into SSPxADD. The SSPxADD BRG value of `0x00` is not supported.

21.4.7.1 Baud Rate and Module Interdependence

Because MSSP1 and MSSP2 are independent, they can operate simultaneously in I^{2}C Master mode at different baud rates. This is done by using different BRG reload values for each module.

Because this mode derives its basic clock source from the system clock, any changes to the clock will affect both modules in the same proportion. It may be possible to change one or both baud rates back to a previous value by changing the BRG reload value.

FIGURE 21-19: BAUD RATE GENERATOR BLOCK DIAGRAM

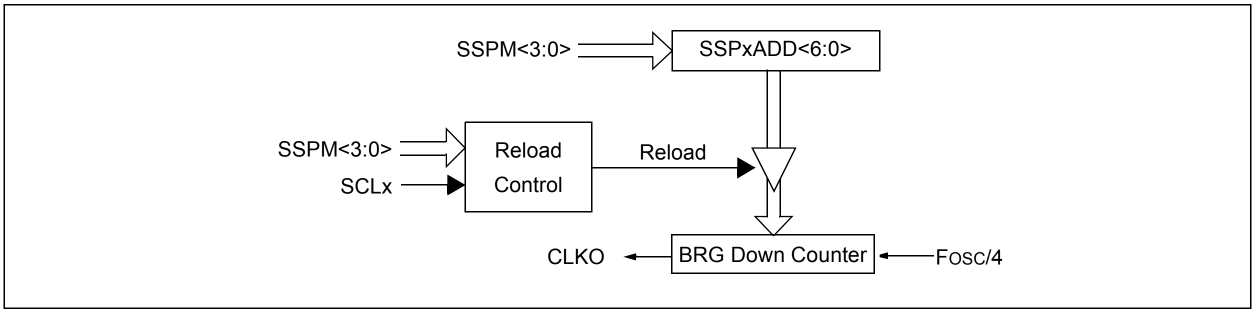


TABLE 21-3: I^{2}C™ CLOCK RATE w/BRG

FOSC	FCY	FCY * 2	BRG Value	FSCL (2 Rollovers of BRG)
40 MHz	10 MHz	20 MHz	18h	400 kHz
40 MHz	10 MHz	20 MHz	1Fh	312.5 kHz
40 MHz	10 MHz	20 MHz	63h	100 kHz
16 MHz	4 MHz	8 MHz	09h	400 kHz
16 MHz	4 MHz	8 MHz	0Ch	308 kHz
16 MHz	4 MHz	8 MHz	27h	100 kHz
4 MHz	1 MHz	2 MHz	02h	333 kHz
4 MHz	1 MHz	2 MHz	09h	100 kHz
16 MHz	4 MHz	8 MHz	03h	1 MHz(1)

Note 1: A minimum of 16 MHz FOSC is required to get 1 MHz I^{2}C.

22.0 ENHANCED UNIVERSAL SYNCHRONOUS ASYNCHRONOUS RECEIVER TRANSMITTER (EUSART)

The Enhanced Universal Synchronous Asynchronous Receiver Transmitter (EUSART) module is one of two serial I/O modules. (Generically, the EUSART is also known as a Serial Communications Interface or SCI.) The EUSART can be configured as a full-duplex, asynchronous system that can communicate with peripheral devices, such as CRT terminals and personal computers. It can also be configured as a half-duplex synchronous system that can communicate with peripheral devices, such as A/D or D/A integrated circuits, serial EEPROMs, etc.

The Enhanced USART module implements additional features, including automatic baud rate detection and calibration, automatic wake-up on Sync Break reception and 12-bit Break character transmit. These make it ideally suited for use in Local Interconnect Network bus (LIN/J2602 bus) systems.

All members of the PIC18F87K22 family are equipped with two independent EUSART modules, referred to as EUSART1 and EUSART2. They can be configured in the following modes:

- Asynchronous (full duplex) with:
  - Auto-wake-up on character reception
  - Auto-baud calibration
  - 12-bit Break character transmission
- Synchronous – Master (half duplex) with selectable clock polarity
- Synchronous – Slave (half duplex) with selectable clock polarity

The pins of EUSART1 and EUSART2 are multiplexed with the functions of PORTC (RC6/TX1/CK1 and RC7/RX1/DT1) and PORTG (RG1/TX2/CK2/AN19/C3OUT and RG2/RX2/DT2/AN18/C3INA), respectively. In order to configure these pins as an EUSART:

- For EUSART1:
  - Bit, SPEN (RCSTA1<7>), must be set (= `1`)
  - Bit, TRISC<7>, must be set (= `1`)
  - Bit, TRISC<6>, must be cleared (= `0`) for Asynchronous and Synchronous Master modes
  - Bit, TRISC<6>, must be set (= `1`) for Synchronous Slave mode
- For EUSART2:
  - Bit, SPEN (RCSTA2<7>), must be set (= `1`)
  - Bit, TRISG<2>, must be set (= `1`)
  - Bit TRISG<1> must be cleared (= `0`) for Asynchronous and Synchronous Master modes
  - Bit, TRISC<6>, must be set (= `1`) for Synchronous Slave mode

Note: The EUSART control will automatically reconfigure the pin from input to output as needed.

The operation of each Enhanced USART module is controlled through three registers:

- Transmit Status and Control (TXSTAx)
- Receive Status and Control (RCSTAx)
- Baud Rate Control (BAUDCONx)

These are detailed on the following pages in Register 22-1, Register 22-2 and Register 22-3, respectively.

Note: Throughout this section, references to register and bit names that may be associated with a specific EUSART module are referred to generically by the use of ‘x’ in place of the specific module number. Thus, “RCSTAx” might refer to the Receive Status register for either EUSART1 or EUSART2.

22.2.5 BREAK CHARACTER SEQUENCE

The EUSART module has the capability of sending the special Break character sequences that are required by the LIN/J2602 bus standard. The Break character transmit consists of a Start bit, followed by twelve ‘0’ bits and a Stop bit. The Frame Break character is sent whenever the SENDB and TXEN bits (TXSTAx<3> and TXSTAx<5>, respectively) are set while the Transmit Shift Register is loaded with data. Note that the value of data written to TXREGx will be ignored and all ‘0’s will be transmitted.

The SENDB bit is automatically reset by hardware after the corresponding Stop bit is sent. This allows the user to preload the transmit FIFO with the next transmit byte following the Break character (typically, the Sync character in the LIN/J2602 specification).

Note that the data value written to the TXREGx for the Break character is ignored. The write simply serves the purpose of initiating the proper sequence.

The TRMT bit indicates when the transmit operation is active or Idle, just as it does during normal transmission. See Figure 22-10 for the timing of the Break character sequence.

22.2.5.1 Break and Sync Transmit Sequence

The following sequence will send a message frame header made up of a Break, followed by an Auto-Baud Sync byte. This sequence is typical of a LIN/J2602 bus master.

1. Configure the EUSART for the desired mode.
2. Set the TXEN and SENDB bits to set up the Break character.
3. Load the TXREGx with a dummy character to initiate transmission (the value is ignored).
4. Write ‘55h’ to TXREGx to load the Sync character into the transmit FIFO buffer.
5. After the Break has been sent, the SENDB bit is reset by hardware. The Sync character now transmits in the preconfigured mode.

When the TXREGx becomes empty, as indicated by the TXxIF, the next data byte can be written to TXREGx.

22.2.6 RECEIVING A BREAK CHARACTER

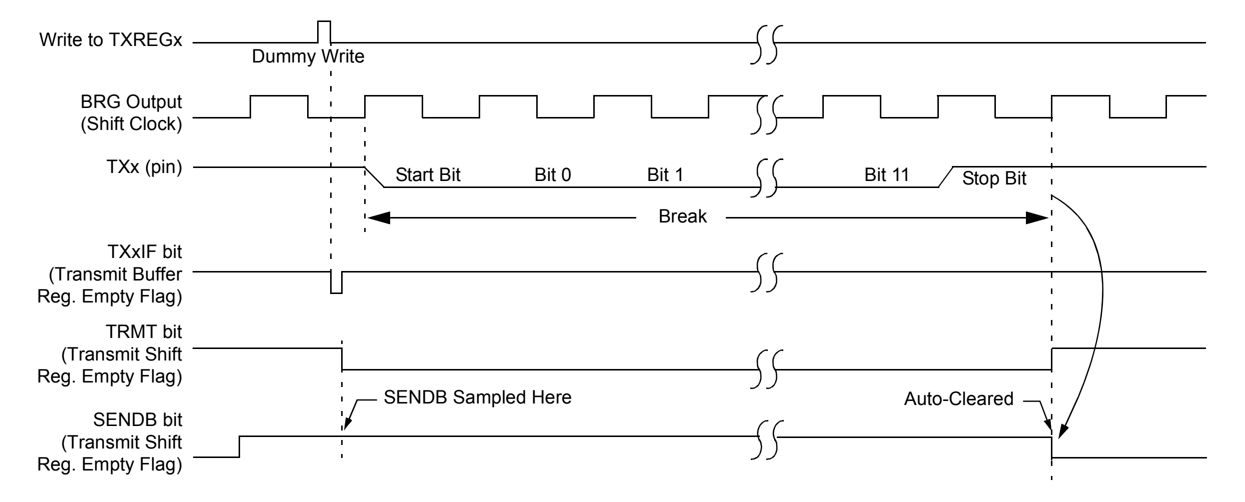
The Enhanced USART module can receive a Break character in two ways.

The first method forces configuration of the baud rate at a frequency of 9/13 the typical speed. This allows for the Stop bit transition to be at the correct sampling location (13 bits for Break versus Start bit and 8 data bits for typical data).

The second method uses the auto-wake-up feature described in **Section 22.2.4 “Auto-Wake-up on Sync Break Character”**. By enabling this feature, the EUSART will sample the next two transitions on RXx/DTx, cause an RCxIF interrupt and receive the next data byte followed by another interrupt.

Note that following a Break character, the user will typically want to enable the Auto-Baud Rate Detect feature. For both methods, the user can set the ABDEN bit once the TXxIF interrupt is observed.

FIGURE 22-10: SEND BREAK CHARACTER SEQUENCE



22.3 EUSART Synchronous Master Mode

The Synchronous Master mode is entered by setting the CSRC bit (TXSTAx<7>). In this mode, the data is transmitted in a half-duplex manner (i.e., transmission and reception do not occur at the same time). When transmitting data, the reception is inhibited and vice versa. Synchronous mode is entered by setting bit, SYNC (TXSTAx<4>). In addition, enable bit, SPEN (RCSTAx<7>), is set in order to configure the TXx and RXx pins to CKx (clock) and DTx (data) lines, respectively.

The Master mode indicates that the processor transmits the master clock on the CKx line. Clock polarity is selected with the TXCKP bit (BAUDCONx<4>). Setting TXCKP sets the Idle state on CKx as high, while clearing the bit sets the Idle state as low. This option is provided to support Microwire devices with this module.

22.3.1 EUSART SYNCHRONOUS MASTER TRANSMISSION

The EUSART transmitter block diagram is shown in Figure 22-3. The heart of the transmitter is the Transmit (Serial) Shift Register (TSR). The Shift register obtains its data from the Read/Write Transmit Buffer register, TXREGx. The TXREGx register is loaded with data in software. The TSR register is not loaded until the last bit has been transmitted from the previous load. As soon as the last bit is transmitted, the TSR is loaded with new data from the TXREGx (if available).

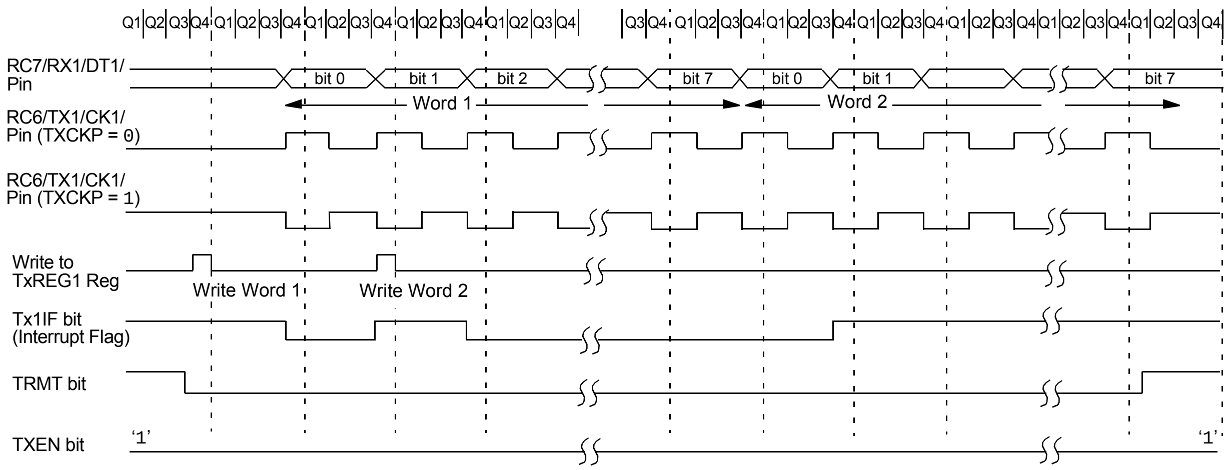
Once the TXREGx register transfers the data to the TSR register (occurs in one TCY), the TXREGx is empty and the TXxIF flag bit is set. The interrupt can be enabled or disabled by setting or clearing the interrupt enable bit, TXxIE. TXxIF is set regardless of the state of enable bit, TXxIE; it cannot be cleared in software. It will reset only when new data is loaded into the TXREGx register.

While flag bit, TXxIF, indicates the status of the TXREGx register, another bit, TRMT (TXSTAx<1>), shows the status of the TSR register. TRMT is a read-only bit which is set when the TSR is empty. No interrupt logic is tied to this bit, so the user must poll this bit in order to determine if the TSR register is empty. The TSR is not mapped in data memory so it is not available to the user.

To set up a Synchronous Master Transmission:

1. Initialize the SPBRGHx:SPBRGx registers for the appropriate baud rate. Set or clear the BRG16 bit, as required, to achieve the desired baud rate.
2. Enable the synchronous master serial port by setting bits, SYNC, SPEN and CSRC.
3. If interrupts are desired, set enable bit, TXxIE.
4. If 9-bit transmission is desired, set bit, TX9.
5. Enable the transmission by setting bit, TXEN.
6. If 9-bit transmission is selected, the ninth bit should be loaded in bit, TX9D.
7. Start transmission by loading data to the TXREGx register.
8. If using interrupts, ensure that the GIE and PEIE bits in the INTCON register (INTCON<7:6>) are set.

FIGURE 22-11: SYNCHRONOUS TRANSMISSION


Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4
RC7/RX1/DT1/ Pin
bit 0
bit 1
bit 2
bit 7
bit 0
bit 1
bit 7
Word 1
Word 2
RC6/TX1/CK1/ Pin (TXCKP = 0)
RC6/TX1/CK1/ Pin (TXCKP = 1)
Write to TxREG1 Reg
Write Word 1
Write Word 2
Tx1IF bit (Interrupt Flag)
TRMT bit
TXEN bit
'1'
'1'

Note: Sync Master mode, SPBRGx = 0, continuous transmission of two 8-bit words. This example is equally applicable to EUSART2 (RG1/TX2/CK2/AN19/C3OUT and RG2/RX2/DT2/AN18/C3INA).

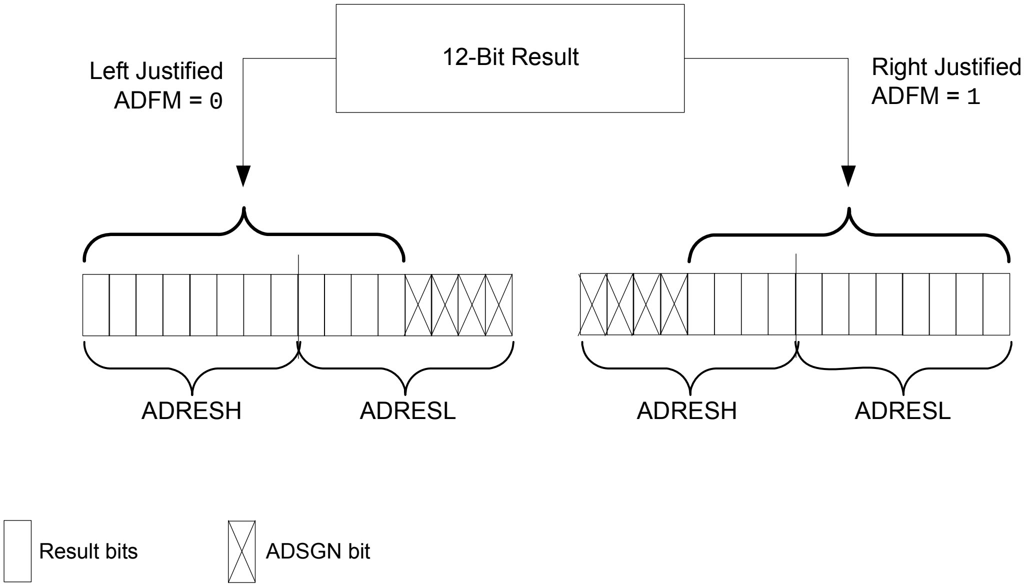
23.2.2 A/D RESULT REGISTERS

The ADRESH:ADRESL register pair is where the 12-bit A/D result and extended sign bits (ADSGN) are loaded at the completion of a conversion. This register pair is 16 bits wide. The A/D module gives the flexibility of left or right justifying the 12-bit result in the 16-Bit Result register. The A/D Format Select bit (ADFM) controls this justification.

Figure 23-3 shows the operation of the A/D result justification and location of the extended sign bits (ADSGN). The extended sign bits allow for easier 16-bit math to be performed on the result. The results are represented as a two's compliment binary value. This means that when sign bits and magnitude bits are considered together in right justification, the ADRESH and ADRESL registers can be read as a single signed integer value.

When the A/D Converter is disabled, these 8-bit registers can be used as two general purpose registers.

FIGURE 23-3: A/D RESULT JUSTIFICATION



Two's Complement Example Results Number Line

Left Justified		Right Justified	
Hex	**Decimal**	**Hex**	**Decimal**
0xFFF0	4095	0x0FFF	4095
0xFFE0	4094	0x0FFE	4094
...	...	...	...
0x0020	2	0x0002	2
0x0010	1	0x0001	1
0x0000	0	0x0000	0
0xFFFF	-1	0xFFFF	-1
0xFFEF	-2	0xFFFE	-2
...	...	...	...
0x001F	-4095	0xF001	-4095
0x000F	-4096	0xF000	-4096

27.2 CTMU Operation

The CTMU works by using a fixed current source to charge a circuit. The type of circuit depends on the type of measurement being made.

In the case of charge measurement, the current is fixed and the amount of time the current is applied to the circuit is fixed. The amount of voltage read by the A/D becomes a measurement of the circuit's capacitance.

In the case of time measurement, the current, as well as the capacitance of the circuit, is fixed. In this case, the voltage read by the A/D is representative of the amount of time elapsed from the time the current source starts and stops charging the circuit.

If the CTMU is being used as a time delay, both capacitance and current source are fixed, as well as the voltage supplied to the comparator circuit. The delay of a signal is determined by the amount of time it takes the voltage to charge to the comparator threshold voltage.

27.2.1 THEORY OF OPERATION

The operation of the CTMU is based on the equation for charge:

$$C = I \bullet \frac{dV}{dT}$$

More simply, the amount of charge measured in coulombs in a circuit is defined as current in amperes (I) multiplied by the amount of time in seconds that the current flows (t). Charge is also defined as the capacitance in farads (C) multiplied by the voltage of the circuit (V). It follows that:

$$I \bullet t = C \bullet V$$

The CTMU module provides a constant, known current source. The A/D Converter is used to measure (V) in the equation, leaving two unknowns: capacitance (C) and time (t). The above equation can be used to calculate capacitance or time, by either the relationship using the known fixed capacitance of the circuit:

$$t = (C \bullet V)/I$$

or by:

$$C = (I \bullet t)/V$$

using a fixed time that the current source is applied to the circuit.

27.2.2 CURRENT SOURCE

At the heart of the CTMU is a precision current source, designed to provide a constant reference for measurements. The level of current is user-selectable across three ranges or a total of two orders of magnitude, with the ability to trim the output in ±2% increments (nominal). The current range is selected by the IRNG<1:0> bits (CTMUICON<1:0>), with a value of '00' representing the lowest range.

Current trim is provided by the ITRIM<5:0> bits (CTMUICON<7:2>). These six bits allow trimming of the current source in steps of approximately 2% per step. Half of the range adjusts the current source positively and the other half reduces the current source. A value of '000000' is the neutral position (no change). A value of '100000' is the maximum negative adjustment (approximately -62%) and '011111' is the maximum positive adjustment (approximately +62%).

27.2.3 EDGE SELECTION AND CONTROL

CTMU measurements are controlled by edge events occurring on the module's two input channels. Each channel, referred to as Edge 1 and Edge 2, can be configured to receive input pulses from one of the edge input pins (CTED1 and CTED2) or CCPx Special Event Triggers. The input channels are level-sensitive, responding to the instantaneous level on the channel rather than a transition between levels. The inputs are selected using the EDG1SEL and EDG2SEL bit pairs (CTMUCONL<3:2, 6:5>).

In addition to source, each channel can be configured for event polarity using the EDGE2POL and EDGE1POL bits (CTMUCONL<7,4>). The input channels can also be filtered for an edge event sequence (Edge 1 occurring before Edge 2) by setting the EDGSEQEN bit (CTMUCONH<2>).

27.2.4 EDGE STATUS

The CTMUCON register also contains two status bits: EDG2STAT and EDG1STAT (CTMUCONL<1:0>). Their primary function is to show if an edge response has occurred on the corresponding channel. The CTMU automatically sets a particular bit when an edge response is detected on its channel. The level-sensitive nature of the input channels also means that the status bits become set immediately if the channel's configuration is changed and matches the channel's current state.

The module uses the edge status bits to control the current source output to external analog modules (such as the A/D Converter). Current is only supplied to external modules when only one (not both) of the status bits is set. Current is shut off when both bits are either set or cleared. This allows the CTMU to measure current only during the interval between edges. After both status bits are set, it is necessary to clear them before another measurement is taken. Both bits should be cleared simultaneously, if possible, to avoid re-enabling the CTMU current source.

In addition to being set by the CTMU hardware, the edge status bits can also be set by software. This permits a user application to manually enable or disable the current source. Setting either (but not both) of the bits enables the current source. Setting or clearing both bits at once disables the source.

BCF — Bit Clear f

Syntax: BCF f, b {,a}

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

Operation: 0 → f<b>

Status Affected: None

Encoding:

1001	bbba	ffff	ffff

Description: Bit ‘b’ in register ‘f’ is cleared.

If ‘a’ is ‘0’, the Access Bank is selected. If ‘a’ is ‘1’, the BSR is used to select the GPR bank.

If ‘a’ is ‘0’ and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \leq 95$ (5Fh). See **Section 29.2.3 “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: $-128 \leq n \leq 127$

Operation: if Negative bit is ‘1’,
(PC) + 2 + 2n → PC

Status Affected: None

Encoding:

1110	0110	nnnn	nnnn

Description: If the Negative bit is ‘1’, then the program will branch.

The 2’s complement number ‘2n’ is added to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC + 2 + 2n. This instruction is then a two-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:
If Jump:

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

If No Jump:

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

Example: `HERE BN Jump`

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

MOVLW Move Literal to W

Syntax: MOVLW k

Operands: 0 ≤ k ≤ 255

Operation: k → W

Status Affected: None

Encoding:

`0000`	`1110`	`kkkk`	`kkkk`

Description: The eight-bit literal ‘k’ is loaded into W.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal ‘k’	Process Data	Write to W

Example: `MOVLW 5Ah`

After Instruction
W = 5Ah

MOVWF Move W to f

Syntax: MOVWF f {,a}

Operands: 0 ≤ f ≤ 255
a ∈ [0,1]

Operation: (W) → f

Status Affected: None

Encoding:

`0110`	`111a`	`ffff`	`ffff`

Description: Move data from W to register ‘f’. Location ‘f’ can be anywhere in the 256-byte bank.

If ‘a’ is ‘0’, the Access Bank is selected. If ‘a’ is ‘1’, the BSR is used to select the GPR bank.

If ‘a’ is ‘0’ and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever f ≤ 95 (5Fh). See **Section 29.2.3 “Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode”** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register ‘f’	Process Data	Write register ‘f’

Example: `MOVWF REG, 0`

Before Instruction
W = 4Fh
REG = FFh
After Instruction
W = 4Fh
REG = 4Fh

SUBFSR	**Subtract Literal from FSR**
Syntax:	SUBFSR f, k
Operands:	0 ≤ k ≤ 63 f ∈ [0, 1, 2]
Operation:	FSRf – k → FSRf
Status Affected:	None
Encoding:	`1110` `1001` `ffkk` `kkkk`
Description:	The 6-bit literal ‘k’ is subtracted from the contents of the FSR specified by ‘f’.
Words:	1
Cycles:	1

Q Cycle Activity:

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

Example: `SUBFSR 2, 23h`

Before Instruction
FSR2 = 03FFh

After Instruction
FSR2 = 03DCh

SUBULNK	**Subtract Literal from FSR2 and Return**
Syntax:	SUBULNK k
Operands:	0 ≤ k ≤ 63
Operation:	FSR2 – k → FSR2, (TOS) → PC
Status Affected:	None
Encoding:	`1110` `1001` `11kk` `kkkk`
Description:	The 6-bit literal ‘k’ is subtracted from the contents of the FSR2. A `RETURN` is then executed by loading the PC with the TOS. The instruction takes two cycles to execute; a `NOP` is performed during the second cycle. This may be thought of as a special case of the `SUBFSR` instruction, where f = 3 (binary ‘`11`’); it operates only on FSR2.
Words:	1
Cycles:	2

Q Cycle Activity:

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

Example: `SUBULNK 23h`

Before Instruction
FSR2 = 03FFh
PC = 0100h

After Instruction
FSR2 = 03DCh
PC = (TOS)

31.2 DC Characteristics: Power-Down and Supply Current PIC18F87K22 Family (Industrial/Extended) (Continued)

PIC18F87K22 Family (Industrial/Extended)		**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial -40°C ≤ TA ≤ +125°C for extended					
Param No.	**Device**	**Typ**	**Max**	**Units**	**Conditions**		
	Supply Current (IDD)[2,3]						
	All devices	5.3	10	μA	-40°C	VDD = 1.8V[4] Regulator Disabled	FOSC = 31 kHz (**RC_RUN** mode, LF-INTOSC)
		5.5	10	μA	+25°C		
		5.5	10	μA	+85°C		
		12	24	μA	+125ºC		
	All devices	10	15	μA	-40°C	VDD = 3.3V[4] Regulator Disabled	
		10	16	μA	+25°C		
		11	17	μA	+85°C		
		15	35	μA	+125ºC		
	All devices	70	180	μA	-40°C	VDD = 5V[5] Regulator Enabled	
		80	185	μA	+25°C		
		90	190	μA	+85°C		
		200	500	μA	+125ºC		
	All devices	410	850	μA	-40°C	VDD = 1.8V[4] Regulator Disabled	FOSC = 1 MHz (**RC_RUN** mode, HF-INTOSC)
		410	800	μA	+25°C		
		410	830	μA	+85°C		
		700	1500	μA	+125ºC		
	All devices	680	990	μA	-40°C	VDD = 3.3V[4] Regulator Disabled	
		680	960	μA	+25°C		
		670	950	μA	+85°C		
		800	1700	μA	+125ºC		
	All devices	760	1400	μA	-40°C	VDD = 5V[5] Regulator Enabled	
		780	1400	μA	+25°C		
		800	1500	μA	+85°C		
		1200	2400	μA	+125ºC		
	All devices	760	1300	μA	-40°C	VDD = 1.8V[4] Regulator Disabled	FOSC = 4 MHz (**RC_RUN** mode, HF-INTOSC)
		760	1400	μA	+25°C		
		770	1500	μA	+85°C		
		800	1700	μA	+125ºC		
	All devices	1.4	2.5	mA	-40°C	VDD = 3.3V[4] Regulator Disabled	
		1.4	2.5	mA	+25°C		
		1.4	2.5	mA	+85°C		
		1.5	3.0	mA	+125ºC		
	All devices	1.5	2.7	mA	-40°C	VDD = 5V[5] Regulator Enabled	
		1.5	2.7	mA	+25°C		
		1.5	2.7	mA	+85°C		
		1.6	3.3	mA	+125ºC		

Note 1: The power-down current in Sleep mode does not depend on the oscillator type. Power-down current is measured with the part in Sleep mode, with all I/O pins in a high-impedance state and tied to VDD or VSS, and all features that add delta current are disabled (such as WDT, SOSC oscillator, BOR, etc.).

2: The supply current is mainly a function of operating voltage, frequency and mode. Other factors, such as I/O pin loading and switching rate, oscillator type and circuit, internal code execution pattern and temperature, also have an impact on the current consumption.
The test conditions for all IDD measurements in Active Operation mode are:
OSC1 = External square wave, from rail-to-rail; all I/O pins tri-stated, pulled to VDD;
$\overline{\text{MCLR}}$ = VDD; WDT enabled/disabled as specified.

3: Standard, low-cost 32 kHz crystals have an operating temperature range of -10°C to +70°C. Extended temperature crystals are available at a much higher cost.

4: Voltage regulator disabled (ENVREG = 0, tied to VSS, $\overline{\text{RETEN}}$ (CONFIG1L<0>) = 1).

5: Voltage regulator enabled (ENVREG = 1, tied to VDD, SRETEN (WDTCON<4>) = 1 and $\overline{\text{RETEN}}$ (CONFIG1L<0>) = 0).

6: 48 MHz, maximum frequency at +125ºC.