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	48MHz
Connectivity	I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	16
Program Memory Size	32KB (16K x 16)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	3.8K x 8
Voltage - Supply (Vcc/Vdd)	2.15V ~ 3.6V
Data Converters	A/D 10x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	28-SSOP (0.209", 5.30mm Width)
Supplier Device Package	28-SSOP
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18f25j11t-i-ss

4.7 Ultra Low-Power Wake-up

The Ultra Low-Power Wake-up (ULPWU) on RA0 allows a slow falling voltage to generate an interrupt without excess current consumption.

Follow these steps to use this feature:

1. Configure a remappable output pin to output the ULPOUT signal.
2. Map an INTx interrupt-on-change input function to the same pin as used for the ULPOUT output function. Alternatively, in step 1, configure ULPOUT to output onto a PORTB interrupt-on-change pin.
3. Charge the capacitor on RA0 by configuring the RA0 pin to an output and setting it to '1'.
4. Enable interrupt for the corresponding pin selected in step 2.
5. Stop charging the capacitor by configuring RA0 as an input.
6. Discharge the capacitor by setting the ULPEN and ULPSINK bits in the WDTCON register.
7. Configure Sleep mode.
8. Enter Sleep mode.

When the voltage on RA0 drops below V_{IL}, an interrupt will be generated, which will cause the device to wake-up and execute the next instruction.

This feature provides a low-power technique for periodically waking up the device from Sleep mode. The time-out is dependent on the discharge time of the RC circuit on RA0.

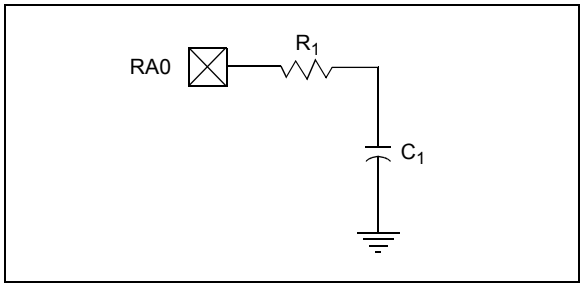
When the ULPWU module causes the device to wake-up from Sleep mode, the WDTCON<ULPLVL> bit is set. When the ULPWU module causes the device to wake-up from Deep Sleep, the DSULP (DSWAKEL<5>) bit is set. Software can check these bits upon wake-up to determine the wake-up source. Also in Sleep mode, only the remappable output function, ULPWU, will output this bit value to an RPn pin for externally detecting wake-up events.

See Example 4-1 for initializing the ULPWU module.

Note:	For module-related bit definitions, see the WDTCON register in **Section 26.2 "Watchdog Timer (WDT)"** and the DSWAKEL register (Register 4-6).

A series resistor between RA0 and the external capacitor provides overcurrent protection for the RA0/AN0/C1INA/ULPWU/RP0 pin and can allow for software calibration of the time-out (see Figure 4-9).

FIGURE 4-9: SERIAL RESISTOR



A timer can be used to measure the charge time and discharge time of the capacitor. The charge time can then be adjusted to provide the desired interrupt delay. This technique will compensate for the affects of temperature, voltage and component accuracy. The peripheral can also be configured as a simple Programmable Low-Voltage Detect (LVD) or temperature sensor.

Note:	For more information, refer to *AN879,* "*Using the Microchip Ultra Low-Power Wake-up Module*" application note (DS00879).

EXAMPLE 4-1: ULTRA LOW-POWER WAKE-UP INITIALIZATION

```
//*****************************************************************************
//Configure a remappable output pin with interrupt capability
//for ULPWU function (RP21 => RD4/INT1 in this example)
//*****************************************************************************
RPOR21 = 13;// ULPWU function mapped to RP21/RD4
RPINR1 = 21;// INT1 mapped to RP21 (RD4)

//***************************
//Charge the capacitor on RA0
//***************************
TRISAbits.TRISA0 = 0;
LATAbits.LATA0 = 1;
for(i = 0; i < 10000; i++) Nop();

//*********************************
//Stop Charging the capacitor on RA0
//*********************************
TRISAbits.TRISA0 = 1;

//****************************************
//Enable the Ultra Low Power Wakeup module
//and allow capacitor discharge
//****************************************
WDTCONbits.ULPEN = 1;
WDTCONbits.ULPSINK = 1;

//****************************************
//For Sleep,  Enable Interrupt for ULPW.
//****************************************
INTCON3bits.INT1IF = 0;
INTCON3bits.INT1IE = 1;

//********************
//Configure Sleep Mode
//********************
//For Sleep
OSCCONbits.IDLEN = 0;

//For Deep Sleep
OSCCONbits.IDLEN = 0;// enable deep sleep
DSCONHbits.DSEN = 1;// Note: must be set just before executing Sleep();
//*****************
//Enter Sleep Mode
//*****************
Sleep();
    // for sleep, execution will resume here
    // for deep sleep, execution will restart at reset vector (use WDTCONbits.DS to detect)
```

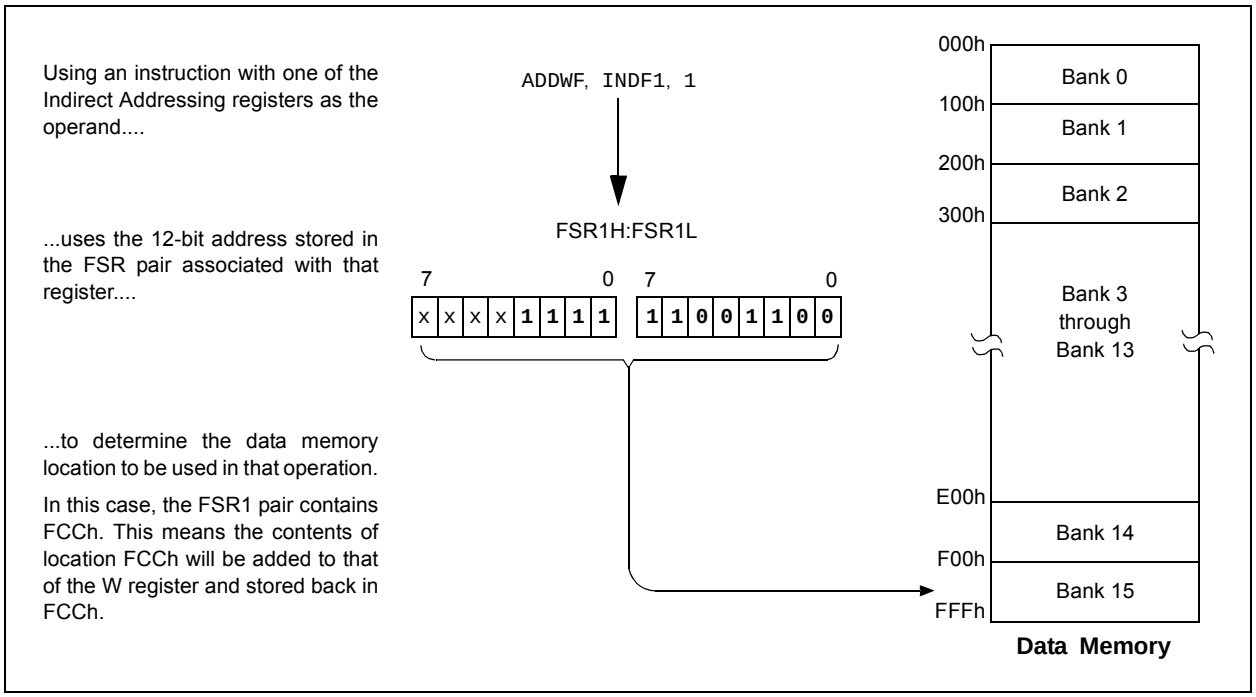
6.4.3.1 FSR Registers and the INDF Operand (INDF)

At the core of Indirect Addressing are three sets of registers: FSR0, FSR1 and FSR2. Each represents a pair of 8-bit registers, FSRnH and FSRnL. The four upper bits of the FSRnH register are not used, so each FSR pair holds a 12-bit value. This represents a value that can address the entire range of the data memory in a linear fashion. The FSR register pairs, then, serve as pointers to data memory locations.

Indirect Addressing is accomplished with a set of INDF operands, INDF0 through INDF2. These can be presumed to be "virtual" registers: they are mapped in the SFR space but are not physically implemented. Reading or writing to a particular INDF register actually accesses its corresponding FSR register pair. A read from INDF1, for example, reads the data at the address indicated by FSR1H:FSR1L. Instructions that use the INDF registers as operands actually use the contents of their corresponding FSR as a pointer to the instruction's target. The INDF operand is just a convenient way of using the pointer.

Because Indirect Addressing uses a full 12-bit address, data RAM banking is not necessary. Thus, the current contents of the BSR and the Access RAM bit have no effect on determining the target address.

FIGURE 6-8: INDIRECT ADDRESSING



REGISTER 7-1: EECON1: EEPROM CONTROL REGISTER 1 (ACCESS FA6h)

U-0	U-0	R/W-0	R/W-0	R/W-x	R/W-0	R/S-0	U-0
—	—	WPROG	FREE	WRERR	WREN	WR	—
bit 7							bit 0

Legend:	S = Settable bit (cannot be cleared in software)		
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 **Unimplemented:** Read as '0'

bit 5 **WPROG:** One Word-Wide Program bit
1 = Program 2 bytes on the next WR command
0 = Program 64 bytes on the next WR command

bit 4 **FREE:** Flash Erase Enable bit
1 = Perform an erase operation on the next WR command (cleared by hardware after completion of erase)
0 = Perform write only

bit 3 **WRERR:** Flash Program Error Flag bit
1 = A write operation is prematurely terminated (any Reset during self-timed programming in normal operation, or an improper write attempt)
0 = The write operation completed

bit 2 **WREN:** Flash Program Write Enable bit
1 = Allows write cycles to Flash program memory
0 = Inhibits write cycles to Flash program memory

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

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

7.4 Erasing Flash Program Memory

The minimum erase block is 512 words or 1024 bytes. Only through the use of an external programmer, or through ICSP control, can larger blocks of program memory be bulk erased. Word erase in the Flash array is not supported.

When initiating an erase sequence from the micro-controller itself, a block of 1024 bytes of program memory is erased. The Most Significant 12 bits of the TBLPTR<21:10> point to the block being erased. TBLPTR<9:0> are ignored.

The EECON1 register commands the erase operation. The WREN bit must be set to enable write operations. The FREE bit is set to select an erase operation. For protection, the write initiate sequence for EECON2 must be used.

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

7.4.1 FLASH PROGRAM MEMORY ERASE SEQUENCE

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

1. Load Table Pointer register with address of row being erased.
2. Set the WREN and FREE bits (EECON1<2,4>) to enable the erase operation.
3. Disable interrupts.
4. Write 55h to EECON2.
5. Write 0AAh to EECON2.
6. Set the WR bit; this will begin the erase cycle.
7. The CPU will stall for the duration of the erase for TIE (see parameter D133B).
8. Re-enable interrupts.

EXAMPLE 7-2: ERASING FLASH PROGRAM MEMORY

```
                MOVLW   CODE_ADDR_UPPER     ; load TBLPTR with the base
                MOVWF   TBLPTRU             ; address of the memory block
                MOVLW   CODE_ADDR_HIGH
                MOVWF   TBLPTRH
                MOVLW   CODE_ADDR_LOW
                MOVWF   TBLPTRL
    ERASE_ROW
                BSF     EECON1, WREN        ; enable write to memory
                BSF     EECON1, FREE        ; enable Erase operation
                BCF     INTCON, GIE         ; disable interrupts
Required        MOVLW   0x55
Sequence        MOVWF   EECON2              ; write 55h
                MOVLW   0xAA
                MOVWF   EECON2              ; write 0AAh
                BSF     EECON1, WR          ; start erase (CPU stall)
                BSF     INTCON, GIE         ; re-enable interrupts
```

TABLE 10-3: PORTA I/O SUMMARY (CONTINUED)

Pin	Function	TRIS Setting	I/O	I/O Type	Description
RA5/AN4/$\overline{SS1}$/ HLVDIN/RP2	RA5	0	O	DIG	LATA<5> data output; not affected by analog input.
		1	I	TTL	PORTA<5> data input; disabled when analog input enabled.
	AN4	1	I	ANA	A/D input channel 4. Default configuration on POR.
	$\overline{SS1}$	1	I	TTL	Slave select input for MSSP1.
	HLVDIN	1	I	ANA	High/Low-Voltage Detect external trip point reference input.
	RP2	1	I	ST	Remappable Peripheral pin 2 input.
		0	O	DIG	Remappable Peripheral pin 2 output.
OSC2/CLKO/ RA6	OSC2	x	O	ANA	Main oscillator feedback output connection (HS mode).
	CLKO	x	O	DIG	System cycle clock output (FOSC/4) in RC and EC Oscillator modes.
	RA6	1	I	TTL	PORTA<6> data input.
		0	O	DIG	LATA<6> data output.
OSC1/CLKI/RA7	OSC1	1	I	ANA	Main oscillator input connection.
	CLKI	1	I	ANA	Main clock input connection.
	RA7	1	I	TTL	PORTA<6> data input.
		0	O	DIG	LATA<6> data output.

Legend: DIG = Digital level output; TTL = TTL input buffer; ST = Schmitt Trigger input buffer; ANA = Analog level input/output; x = Don't care (TRIS bit does not affect port direction or is overridden for this option)

Note 1: This bit is only available on 44-pin devices.

TABLE 10-4: SUMMARY OF REGISTERS ASSOCIATED WITH PORTA

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on page
PORTA	RA7	RA6	RA5	—	RA3	RA2	RA1	RA0	87
LATA	LAT7	LAT6	LAT5	—	LAT3	LAT2	LAT1	LAT0	87
TRISA	TRIS7	TRIS6	TRISA5	—	TRISA3	TRISA2	TRISA1	TRISA0	87
ANCON0	PCFG7(1)	PCFG6(1)	PCFG5(1)	PCFG4	PCFG3	PCFG2	PCFG1	PCFG0	88
CMxCON	CON	COE	CPOL	EVPOL1	EVPOL0	CREF	CCH1	CCH0	87
CVRCON	CVREN	CVROE	CVRR	CVRSS	CVR3	CVR2	CVR1	CVR0	88

Legend: — = unimplemented, read as '0'. Shaded cells are not used by PORTA.

Note 1: These bits are only available in 44-pin devices.

REGISTER 10-12: RPINR8: PERIPHERAL PIN SELECT INPUT REGISTER 8 (BANKED EEEh)

U-0	U-0	U-0	R/$\overline{W}$-1	R/$\overline{W}$-1	R/$\overline{W}$-1	R/$\overline{W}$-1	R/$\overline{W}$-1
—	—	—	IC2R4	IC2R3	IC2R2	IC2R1	IC2R0
bit 7							bit 0

Legend:	R/$\overline{W}$ = Readable, Writable if IOLOCK = 0		
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 7-5 **Unimplemented:** Read as '0'

bit 4-0 **IC2R<4:0>:** Assign Input Capture 2 (ECCP2) to the Corresponding RPn Pin bits

REGISTER 10-13: RPINR12: PERIPHERAL PIN SELECT INPUT REGISTER 12 (BANKED EF2h)

U-0	U-0	U-0	R/$\overline{W}$-1	R/$\overline{W}$-1	R/$\overline{W}$-1	R/$\overline{W}$-1	R/$\overline{W}$-1
—	—	—	T1GR4	T1GR3	T1GR2	T1GR1	T1GR0
bit 7							bit 0

Legend:	R/$\overline{W}$ = Readable, Writable if IOLOCK = 0		
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 7-5 **Unimplemented:** Read as '0'

bit 4-0 **T1GR<4:0>:** Timer1 Gate Input (T1G) to the Corresponding RPn Pin bits

REGISTER 10-14: RPINR13: PERIPHERAL PIN SELECT INPUT REGISTER 13 (BANKED EF3h)

U-0	U-0	U-0	R/$\overline{W}$-1	R/$\overline{W}$-1	R/$\overline{W}$-1	R/$\overline{W}$-1	R/$\overline{W}$-1
—	—	—	T3GR4	T3GR3	T3GR2	T3GR1	T3GR0
bit 7							bit 0

Legend:	R/$\overline{W}$ = Readable, Writable if IOLOCK = 0		
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 7-5 **Unimplemented:** Read as '0'

bit 4-0 **T3GR<4:0>:** Timer3 Gate Input (T3G) to the Corresponding RPn Pin bits

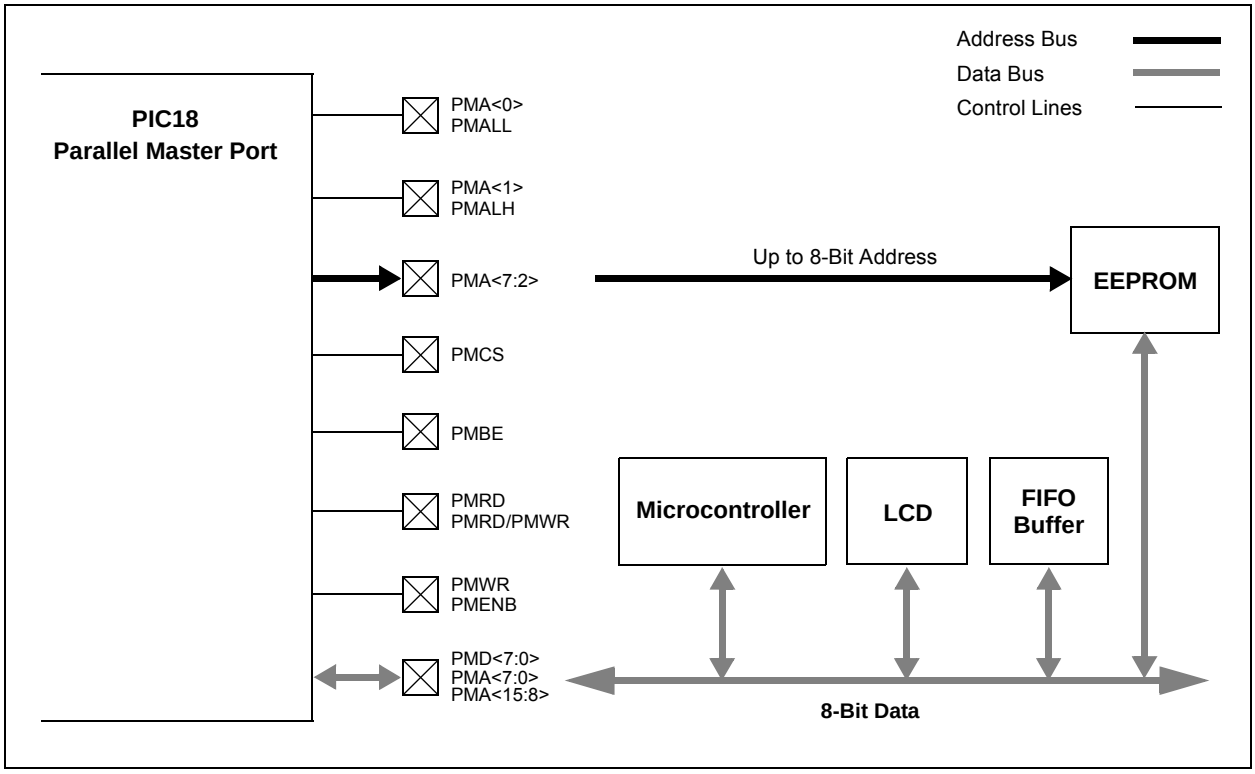
11.0 PARALLEL MASTER PORT (PMP)

The Parallel Master Port module (PMP) is an 8-bit parallel I/O module, specifically designed to communicate with a wide variety of parallel devices, such as communication peripherals, LCDs, external memory devices and microcontrollers. Because the interface to parallel peripherals varies significantly, the PMP is highly configurable. The PMP module can be configured to serve as either a PMP or as a Parallel Slave Port (PSP).

Key features of the PMP module are:

- Up to 16 bits of Addressing when Using Data/Address Multiplexing
- Up to 8 Programmable Address Lines
- One Chip Select Line
- Programmable Strobe Options:
  - Individual Read and Write Strobes or;
  - Read/$\overline{\text{Write}}$ Strobe with Enable Strobe
- Address Auto-Increment/Auto-Decrement
- Programmable Address/Data Multiplexing
- Programmable Polarity on Control Signals
- Legacy Parallel Slave Port Support
- Enhanced Parallel Slave Support:
  - Address Support
  - 4-Byte Deep, Auto-Incrementing Buffer
- Programmable Wait States
- Selectable Input Voltage Levels

FIGURE 11-1: PMP MODULE OVERVIEW



12.0 TIMER0 MODULE

The Timer0 module incorporates the following features:

- Software selectable operation as a timer or counter in both 8-bit or 16-bit modes
- Readable and writable registers
- Dedicated 8-bit, software programmable prescaler
- Selectable clock source (internal or external)
- Edge select for external clock
- Interrupt-on-overflow

The T0CON register (Register 12-1) controls all aspects of the module's operation, including the prescale selection. It is both readable and writable.

Figure 12-1 provides a simplified block diagram of the Timer0 module in 8-bit mode. Figure 12-2 provides a simplified block diagram of the Timer0 module in 16-bit mode.

REGISTER 12-1: T0CON: TIMER0 CONTROL REGISTER (ACCESS FD5h)

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0
bit 7							bit 0

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 **TMR0ON:** Timer0 On/Off Control bit
1 = Enables Timer0
0 = Stops Timer0

bit 6 **T08BIT**: Timer0 8-Bit/16-Bit Control bit
1 = Timer0 is configured as an 8-bit timer/counter
0 = Timer0 is configured as a 16-bit timer/counter

bit 5 **T0CS**: Timer0 Clock Source Select bit
1 = Transition on T0CKI pin input edge
0 = Internal clock (FOSC/4)

bit 4 **T0SE**: Timer0 Source Edge Select bit
1 = Increment on high-to-low transition on T0CKI pin
0 = Increment on low-to-high transition on T0CKI pin

bit 3 **PSA**: Timer0 Prescaler Assignment bit
1 = Timer0 prescaler is not assigned. Timer0 clock input bypasses prescaler.
0 = Timer0 prescaler is assigned. Timer0 clock input comes from prescaler output.

bit 2-0 **T0PS<2:0>**: Timer0 Prescaler Select bits
111 = 1:256 Prescale value
110 = 1:128 Prescale value
101 = 1:64 Prescale value
100 = 1:32 Prescale value
011 = 1:16 Prescale value
010 = 1:8 Prescale value
001 = 1:4 Prescale value
000 = 1:2 Prescale value

17.1 RTCC MODULE REGISTERS

The RTCC module registers are divided into following categories:

RTCC Control Registers

- RTCCFG
- RTCCAL
- PADCFG1
- ALRMCFG
- ALRMRPT

RTCC Value Registers

- RTCVALH and RTCVALL – Can access the following registers
  - YEAR
  - MONTH
  - DAY
  - WEEKDAY
  - HOUR
  - MINUTE
  - SECOND

Alarm Value Registers

- ALRMVALH and ALRMVALL – Can access the following registers:
  - ALRMMNTH
  - ALRMDAY
  - ALRMWD
  - ALRMHR
  - ALRMMIN
  - ALRMSEC

Note: The RTCVALH and RTCVALL registers can be accessed through RTCRPT<1:0>. ALRMVALH and ALRMVALL can be accessed through ALRMPTR<1:0>.

18.5.2.1 Direction Change in Full-Bridge Mode

In the Full-Bridge mode, the PxM1 bit in the CCPxCON register allows users to control the forward/reverse direction. When the application firmware changes this direction control bit, the module will change to the new direction on the next PWM cycle.

A direction change is initiated in software by changing the PxM1 bit of the CCPxCON register. The following sequence occurs prior to the end of the current PWM period:

- The modulated outputs (PxB and PxD) are placed in their inactive state.
- The associated unmodulated outputs (PxA and PxC) are switched to drive in the opposite direction.
- PWM modulation resumes at the beginning of the next period.

See Figure 18-12 for an illustration of this sequence.

The Full-Bridge mode does not provide a dead-band delay. As one output is modulated at a time, a dead-band delay is generally not required. There is a situation where a dead-band delay is required. This situation occurs when both of the following conditions are true:

1. The direction of the PWM output changes when the duty cycle of the output is at or near 100%.
2. The turn-off time of the power switch, including the power device and driver circuit, is greater than the turn-on time.

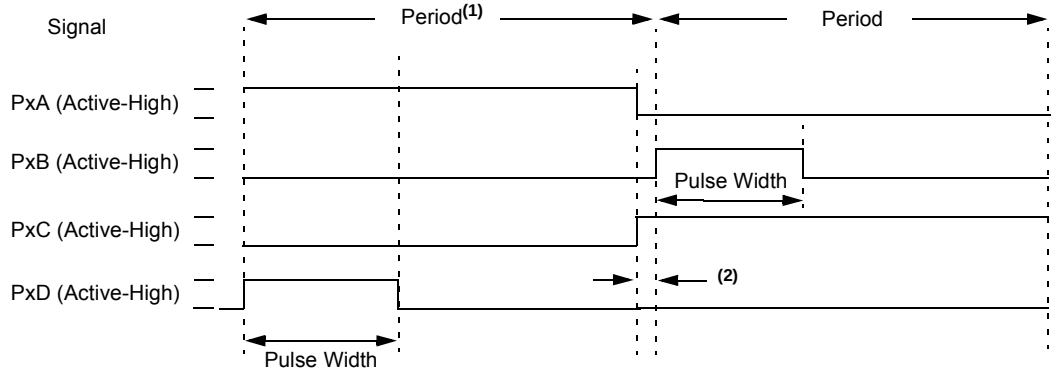
Figure 18-13 shows an example of the PWM direction changing from forward to reverse, at a near 100% duty cycle. In this example, at time, t1, the PxA and PxD outputs become inactive, while the PxC output becomes active. Since the turn-off time of the power devices is longer than the turn-on time, a shoot-through current will flow through power devices, QC and QD (see Figure 18-10), for the duration of 't'. The same phenomenon will occur to power devices, QA and QB, for PWM direction change from reverse to forward.

If changing PWM direction at high duty cycle is required for an application, two possible solutions for eliminating the shoot-through current are:

1. Reduce PWM duty cycle for one PWM period before changing directions.
2. Use switch drivers that can drive the switches off faster than they can drive them on.

Other options to prevent shoot-through current may exist.

FIGURE 18-12: EXAMPLE OF PWM DIRECTION CHANGE



Note 1: The direction bit, PxM1 of the CCPxCON register, is written any time during the PWM cycle.

2: When changing directions, the PxA and PxC signals switch before the end of the current PWM cycle. The modulated PxB and PxD signals are inactive at this time. The length of this time is: (1/FOSC) • TMR2 Prescale Value

FIGURE 20-4: ASYNCHRONOUS TRANSMISSION

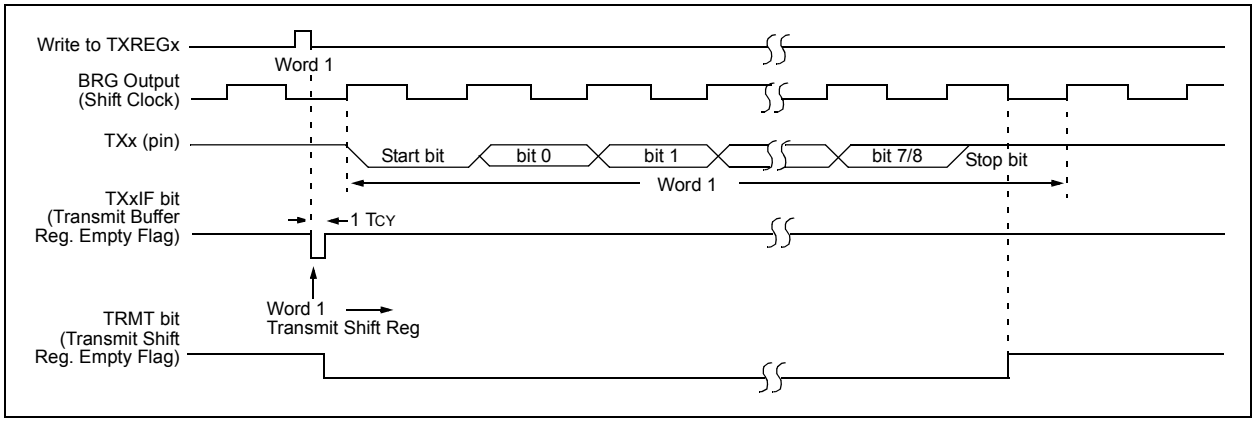
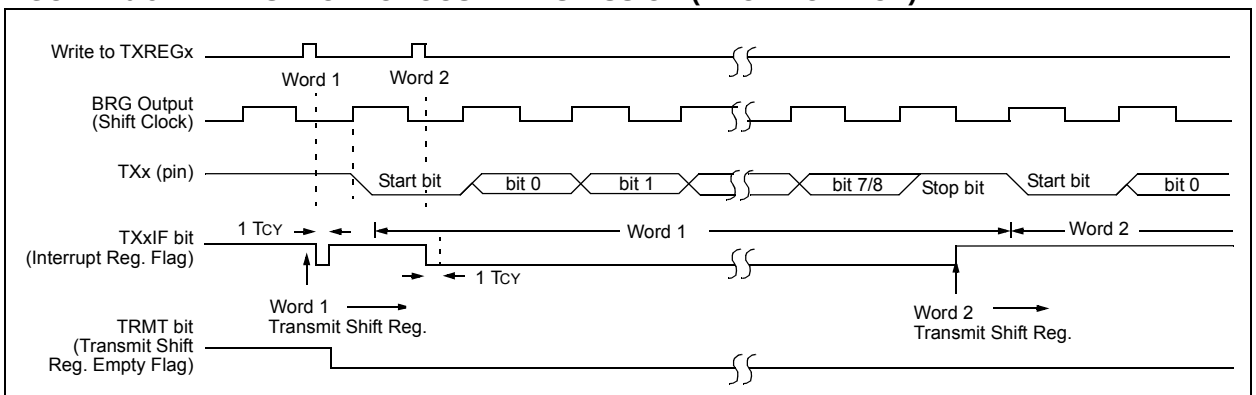


FIGURE 20-5: ASYNCHRONOUS TRANSMISSION (BACK-TO-BACK)



Note: This timing diagram shows two consecutive transmissions.

TABLE 20-5: REGISTERS ASSOCIATED WITH ASYNCHRONOUS TRANSMISSION

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Reset Values on Page:
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	69
PIR1	PMPIF(1)	ADIF	RC1IF	TX1IF	SSP1IF	CCP1IF	TMR2IF	TMR1IF	72
PIE1	PMPIE(1)	ADIE	RC1IE	TX1IE	SSP1IE	CCP1IE	TMR2IE	TMR1IE	72
IPR1	PMPIP(1)	ADIP	RC1IP	TX1IP	SSP1IP	CCP1IP	TMR2IP	TMR1IP	72
PIR3	SSP2IF	BCL2IF	RC2IF	TX2IF	TMR4IF	CTMUIF	TMR3GIF	RTCCIF	72
PIE3	SSP2IE	BCL2IE	RC2IE	TX2IE	TMR4IE	CTMUIE	TMR3GIE	RTCCIE	72
IPR3	SSP2IP	BCL2IP	RC2IP	TX2IP	TMR4IP	CTMUIP	TMR3GIP	RTCCIP	72
RCSTAx	SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D	72
TXREGx	EUSARTx Transmit Register								72
TXSTAx	CSRC	TX9	TXEN	SYNC	SENDB	BRGH	TRMT	TX9D	72
BAUDCONx	ABDOVF	RCIDL	RXDTP	TXDTP	BRG16	—	WUE	ABDEN	73
SPBRGHx	EUSARTx Baud Rate Generator Register High Byte								72
SPBRGx	EUSARTx Baud Rate Generator Register Low Byte								72
ODCON2	—	—	—	—			U2OD	U1OD	74

Legend: – = unimplemented locations read as '0'. Shaded cells are not used for asynchronous transmission.

Note 1: These bits are only available on 44-pin devices.

25.9 Registers

There are three control registers for the CTMU:

- CTMUCONH
- CTMUCONL
- CTMUICON

The CTMUCONH and CTMUCONL registers (Register 25-1 and Register 25-2) contain control bits for configuring the CTMU module edge source selection, edge source polarity selection, edge sequencing, A/D trigger, analog circuit capacitor discharge and enables. The CTMUICON register (Register 25-3) has bits for selecting the current source range and current source trim.

REGISTER 25-1: CTMUCONH: CTMU CONTROL REGISTER HIGH (ACCESS FB3h)

R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	U-0
CTMUEN	—	CTMUSIDL	TGEN	EDGEN	EDGSEQEN	IDISSEN	—
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 **CTMUEN:** CTMU Enable bit
1 = Module is enabled
0 = Module is disabled

bit 6 **Unimplemented:** Read as ‘0’

bit 5 **CTMUSIDL:** Stop in Idle Mode bit
1 = Discontinue module operation when device enters Idle mode
0 = Continue module operation in Idle mode

bit 4 **TGEN:** Time Generation Enable bit
1 = Enables edge delay generation
0 = Disables edge delay generation

bit 3 **EDGEN:** Edge Enable bit
1 = Edges are not blocked
0 = Edges are blocked

bit 2 **EDGSEQEN:** Edge Sequence Enable bit
1 = Edge 1 event must occur before Edge 2 event can occur
0 = No edge sequence is needed

bit 1 **IDISSEN:** Analog Current Source Control bit
1 = Analog current source output is grounded
0 = Analog current source output is not grounded

bit 0 **Reserved:** Write as ‘0’

REGISTER 25-2: CTMUCONL: CTMU CONTROL REGISTER LOW (ACCESS FB2h)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x	R/W-x
EDG2POL	EDG2SEL1	EDG2SEL0	EDG1POL	EDG1SEL1	EDG1SEL0	EDG2STAT	EDG1STAT
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 **EDG2POL:** Edge 2 Polarity Select bit
1 = Edge 2 programmed for a positive edge response
0 = Edge 2 programmed for a negative edge response

bit 6-5 **EDG2SEL<1:0>:** Edge 2 Source Select bits
11 = CTED1 pin
10 = CTED2 pin
01 = ECCP1 Special Event Trigger
00 = ECCP2 Special Event Trigger

bit 4 **EDG1POL:** Edge 1 Polarity Select bit
1 = Edge 1 programmed for a positive edge response
0 = Edge 1 programmed for a negative edge response

bit 3-2 **EDG1SEL<1:0>:** Edge 1 Source Select bits
11 = CTED1 pin
10 = CTED2 pin
01 = ECCP1 Special Event Trigger
00 = ECCP2 Special Event Trigger

bit 1 **EDG2STAT:** Edge 2 Status bit
1 = Edge 2 event has occurred
0 = Edge 2 event has not occurred

bit 0 **EDG1STAT:** Edge 1 Status bit
1 = Edge 1 event has occurred
0 = Edge 1 event has not occurred

26.2 Watchdog Timer (WDT)

PIC18F46J11 family devices have both a conventional WDT circuit and a dedicated, Deep Sleep capable Watchdog Timer. When enabled, the conventional WDT operates in normal Run, Idle and Sleep modes. This data sheet section describes the conventional WDT circuit.

The dedicated, Deep Sleep capable WDT can only be enabled in Deep Sleep mode. This timer is described in **Section 4.6.4 "Deep Sleep Watchdog Timer (DSWDT)"**.

The conventional WDT is driven by the INTRC oscillator. When the WDT is enabled, the clock source is also enabled. The nominal WDT period is 4 ms and has the same stability as the INTRC oscillator.

The 4 ms period of the WDT is multiplied by a 16-bit postscaler. Any output of the WDT postscaler is selected by a multiplexer, controlled by the WDTPS bits in Configuration Register 2H. Available periods range from about 4 ms to 135 seconds (2.25 minutes depending on voltage, temperature and WDT postscaler). The WDT and postscaler are cleared whenever a SLEEP or CLRWDT instruction is executed, or a clock failure (primary or Timer1 oscillator) has occurred.

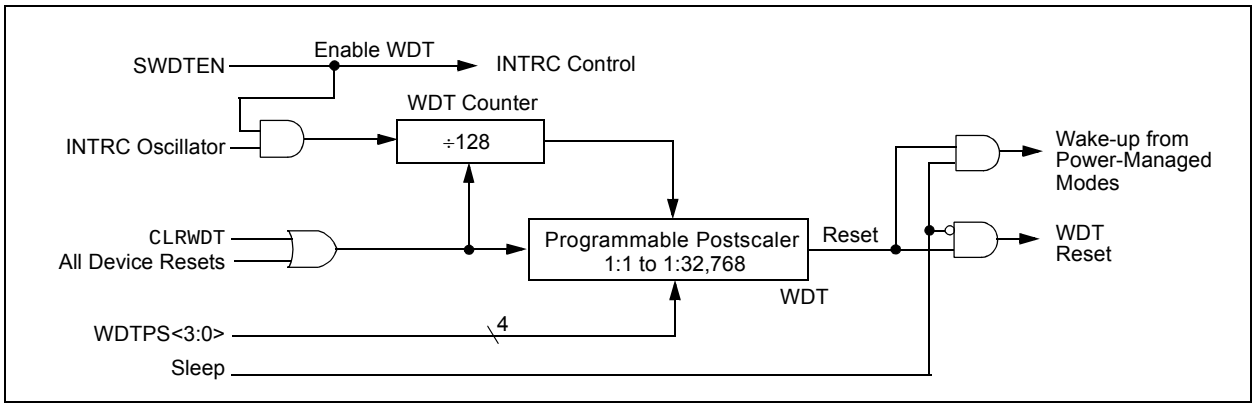
Note 1:	The CLRWDT and SLEEP instructions clear the WDT and postscaler counts when executed.
2:	When a CLRWDT instruction is executed, the postscaler count will be cleared.

26.2.1 CONTROL REGISTER

The WDTCON register (Register 26-11) is a readable and writable register. The SWDTEN bit enables or disables WDT operation. This allows software to override the WDTEN Configuration bit and enable the WDT only if it has been disabled by the Configuration bit.

LVDSTAT is a read-only status bit that is continuously updated and provides information about the current level of V_{DDCORE}. This bit is only valid when the on-chip voltage regulator is enabled.

FIGURE 26-1: WDT BLOCK DIAGRAM



26.4 Two-Speed Start-up

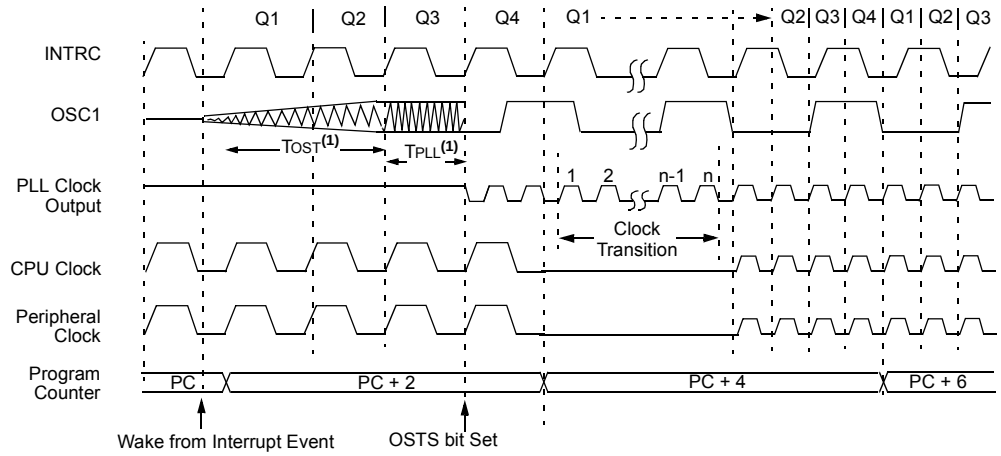
The Two-Speed Start-up feature helps to minimize the latency period, from oscillator start-up to code execution, by allowing the microcontroller to use the INTRC oscillator as a clock source until the primary clock source is available. It is enabled by setting the IESO Configuration bit.

Two-Speed Start-up should be enabled only if the primary oscillator mode is HS or HSPLL (Crystal-Based) modes. Since the EC and ECPLL modes do not require an Oscillator Start-up Timer (OST) delay, Two-Speed Start-up should be disabled.

When enabled, Resets and wake-ups from Sleep mode cause the device to configure itself to run from the internal oscillator block as the clock source, following the time-out of the Power-up Timer after a Power-on Reset is enabled. This allows almost immediate code execution while the primary oscillator starts and the OST is running. Once the OST times out, the device automatically switches to PRI_RUN mode.

In all other power-managed modes, Two-Speed Start-up is not used. The device will be clocked by the currently selected clock source until the primary clock source becomes available. The setting of the IESO bit is ignored.

FIGURE 26-3: TIMING TRANSITION FOR TWO-SPEED START-UP (INTRC TO HSPLL)



Note1: TOST = 1024 TOSC; TPLL = 2 ms (approx). These intervals are not shown to scale.

26.4.1 SPECIAL CONSIDERATIONS FOR USING TWO-SPEED START-UP

While using the INTRC oscillator in Two-Speed Start-up, the device still obeys the normal command sequences for entering power-managed modes, including serial SLEEP instructions (refer to **Section 4.1.4 "Multiple Sleep Commands"**). In practice, this means that user code can change the SCS<1:0> bit settings or issue SLEEP instructions before the OST times out. This would allow an application to briefly wake-up, perform routine "housekeeping" tasks and return to Sleep before the device starts to operate from the primary oscillator.

User code can also check if the primary clock source is currently providing the device clocking by checking the status of the OSTS bit (OSCCON<3>). If the bit is set, the primary oscillator is providing the clock. Otherwise, the internal oscillator block is providing the clock during wake-up from Reset or Sleep mode.

26.5 Fail-Safe Clock Monitor

The Fail-Safe Clock Monitor (FSCM) allows the microcontroller to continue operation in the event of an external oscillator failure by automatically switching the device clock to the internal oscillator block. The FSCM function is enabled by setting the FCMEN Configuration bit.

When FSCM is enabled, the INTRC oscillator runs at all times to monitor clocks to peripherals and provide a backup clock in the event of a clock failure. Clock monitoring (shown in Figure 26-4) is accomplished by creating a sample clock signal, which is the INTRC output divided by 64. This allows ample time between FSCM sample clocks for a peripheral clock edge to occur. The peripheral device clock and the sample clock are presented as inputs to the clock monitor latch. The clock monitor is set on the falling edge of the device clock source but cleared on the rising edge of the sample clock.

TABLE 27-2: PIC18F46J11 FAMILY INSTRUCTION SET (CONTINUED)

Mnemonic, Operands	Description	Cycles	16-Bit Instruction Word				Status Affected	Notes
			MSb			LSb		
LITERAL OPERATIONS								
ADDLW k	Add Literal and WREG	1	0000	1111	kkkk	kkkk	C, DC, Z, OV, N	
ANDLW k	AND Literal with WREG	1	0000	1011	kkkk	kkkk	Z, N	
IORLW k	Inclusive OR Literal with WREG	1	0000	1001	kkkk	kkkk	Z, N	
LFSR f, k	Move Literal (12-bit) 2nd word to FSR(f) 1st word	2	1110 1111	1110 0000	00ff kkkk	kkkk kkkk	None	
MOVLB k	Move Literal to BSR<3:0>	1	0000	0001	0000	kkkk	None	
MOVLW k	Move Literal to WREG	1	0000	1110	kkkk	kkkk	None	
MULLW k	Multiply Literal with WREG	1	0000	1101	kkkk	kkkk	None	
RETLW k	Return with Literal in WREG	2	0000	1100	kkkk	kkkk	None	
SUBLW k	Subtract WREG from Literal	1	0000	1000	kkkk	kkkk	C, DC, Z, OV, N	
XORLW k	Exclusive OR Literal with WREG	1	0000	1010	kkkk	kkkk	Z, N	
DATA MEMORY ↔ PROGRAM MEMORY OPERATIONS								
TBLRD*	Table Read	2	0000	0000	0000	1000	None	
TBLRD*+	Table Read with Post-Increment		0000	0000	0000	1001	None	
TBLRD*-	Table Read with Post-Decrement		0000	0000	0000	1010	None	
TBLRD+*	Table Read with Pre-Increment		0000	0000	0000	1011	None	
TBLWT*	Table Write	2	0000	0000	0000	1100	None	
TBLWT*+	Table Write with Post-Increment		0000	0000	0000	1101	None	
TBLWT*-	Table Write with Post-Decrement		0000	0000	0000	1110	None	
TBLWT+*	Table Write with Pre-Increment		0000	0000	0000	1111	None	

Note 1: When a PORT register is modified as a function of itself (e.g., `MOVF PORTB, 1, 0`), the value used will be that value present on the pins themselves. For example, if the data latch is '1' for a pin configured as input and is driven low by an external device, the data will be written back with a '0'.

2: If this instruction is executed on the TMR0 register (and, where applicable, d = 1), the prescaler will be cleared if assigned.

3: If the Program Counter (PC) is modified or a conditional test is true, the instruction requires two cycles. The second cycle is executed as a `NOP`.

4: Some instructions are two-word instructions. The second word of these instructions will be executed as a `NOP` unless the first word of the instruction retrieves the information embedded in these 16 bits. This ensures that all program memory locations have a valid instruction.

GOTO **Unconditional Branch**

Syntax: GOTO k

Operands: $0 \le k \le 1048575$

Operation: $k \rightarrow PC<20:1>$

Status Affected: None

Encoding:

1st word (k<7:0>)	1110	1111	k_7kkk	$kkkk_0$
2nd word(k<19:8>)	1111	$k_{19}kkk$	kkkk	$kkkk_8$

Description: GOTO allows an unconditional branch anywhere within entire 2-Mbyte memory range. The 20-bit value 'k' is loaded into PC<20:1>. GOTO is always a two-cycle instruction.

Words: 2

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'<7:0>,	No operation	Read literal 'k'<19:8>, Write to PC
No operation	No operation	No operation	No operation

Example: GOTO THERE

After Instruction
PC = Address (THERE)

INCF **Increment f**

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

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

Operation: (f) + 1 → dest

Status Affected: C, DC, N, OV, Z

Encoding:

0010	10da	ffff	ffff

Description: The contents of register 'f' are incremented. If 'd' is '0', the result is placed in W. If 'd' is '1', the result is placed back in register 'f' (default).

If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank (default).

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 27.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: INCF CNT, 1, 0

Before Instruction
CNT = FFh
Z = 0
C = ?
DC = ?
After Instruction
CNT = 00h
Z = 1
C = 1
DC = 1

POP **Pop Top of Return Stack**

Syntax: POP

Operands: None

Operation: (TOS) → bit bucket

Status Affected: None

Encoding:

0000	0000	0000	0110

Description: The TOS value is pulled off the return stack and is discarded. The TOS value then becomes the previous value that was pushed onto the return stack. This instruction is provided to enable the user to properly manage the return stack to incorporate a software stack.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	POP TOS value	No operation

Example:
```
POP
GOTO NEW
```
Before Instruction
TOS = 0031A2h
Stack (1 level down) = 014332h

After Instruction
TOS = 014332h
PC = NEW

PUSH **Push Top of Return Stack**

Syntax: PUSH

Operands: None

Operation: (PC + 2) → TOS

Status Affected: None

Encoding:

0000	0000	0000	0101

Description: The PC + 2 is pushed onto the top of the return stack. The previous TOS value is pushed down on the stack. This instruction allows implementing a software stack by modifying TOS and then pushing it onto the return stack.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	PUSH PC + 2 onto return stack	No operation	No operation

Example:
```
PUSH
```
Before Instruction
TOS = 345Ah
PC = 0124h

After Instruction
PC = 0126h
TOS = 0126h
Stack (1 level down) = 345Ah

TBLRD **Table Read**

Syntax: TBLRD (*; *+; *-; +*)

Operands: None

Operation: if TBLRD *,
(Prog Mem (TBLPTR)) → TABLAT,
TBLPTR – No Change;
if TBLRD *+,
(Prog Mem (TBLPTR)) → TABLAT,
(TBLPTR) + 1 → TBLPTR;
if TBLRD *-,
(Prog Mem (TBLPTR)) → TABLAT,
(TBLPTR) – 1 → TBLPTR;
if TBLRD +*,
(TBLPTR) + 1 → TBLPTR,
(Prog Mem (TBLPTR)) → TABLAT

Status Affected: None

Encoding:

0000	0000	0000	10nn nn=0 * =1 *+ =2 *- =3 +*

Description: This instruction is used to read the contents of Program Memory (P.M.). To address the program memory, a pointer called Table Pointer (TBLPTR) is used.

The TBLPTR (a 21-bit pointer) points to each byte in the program memory. TBLPTR has a 2-Mbyte address range.

TBLPTR<0> = 0: Least Significant Byte of Program Memory Word

TBLPTR<0> = 1: Most Significant Byte of Program Memory Word

The TBLRD instruction can modify the value of TBLPTR as follows:

- no change
- post-increment
- post-decrement
- pre-increment

Words: 1

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	No operation	No operation
No operation	No operation (Read Program Memory)	No operation	No operation (Write TABLAT)

TBLRD **Table Read (Continued)**

Example 1: TBLRD *+

Before Instruction
TABLAT = 55h
TBLPTR = 00A356h
MEMORY(00A356h) = 34h
After Instruction
TABLAT = 34h
TBLPTR = 00A357h

Example 2: TBLRD +*

Before Instruction
TABLAT = AAh
TBLPTR = 01A357h
MEMORY(01A357h) = 12h
MEMORY(01A358h) = 34h
After Instruction
TABLAT = 34h
TBLPTR = 01A358h