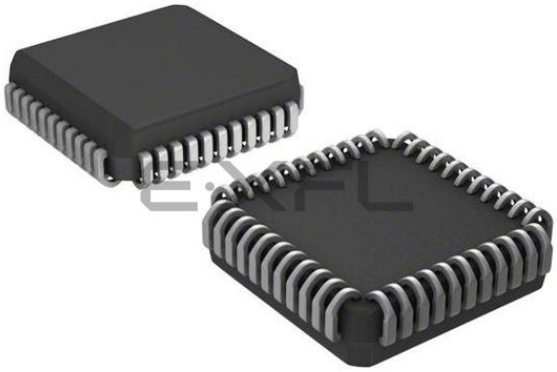


Welcome to E-XFL.COM

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.

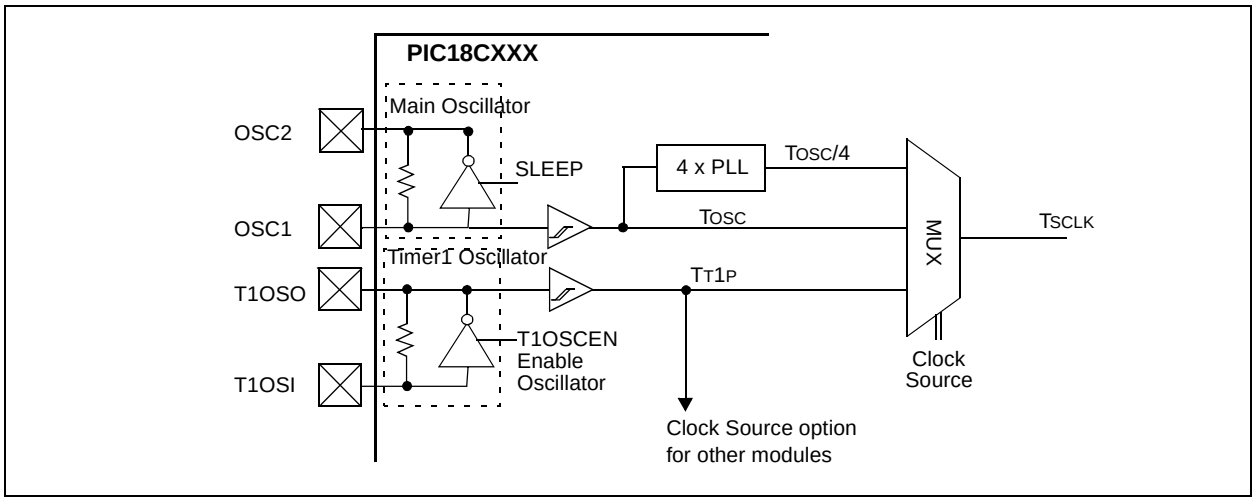
Applications of "Embedded - Microcontrollers"

Details	
Product Status	Obsolete
Core Processor	PIC
Core Size	8-Bit
Speed	40MHz
Connectivity	I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, LVD, POR, PWM, WDT
Number of I/O	33
Program Memory Size	16KB (8K x 16)
Program Memory Type	OTP
EEPROM Size	-
RAM Size	512 x 8
Voltage - Supply (Vcc/Vdd)	4.2V ~ 5.5V
Data Converters	A/D 8x10b
Oscillator Type	External
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	44-LCC (J-Lead)
Supplier Device Package	44-PLCC (16.59x16.59)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18c442t-i-l

2.6 Oscillator Switching Feature

The PIC18CXX2 devices include a feature that allows the system clock source to be switched from the main oscillator to an alternate low frequency clock source. For the PIC18CXX2 devices, this alternate clock source is the Timer1 oscillator. If a low frequency crystal (32 kHz, for example) has been attached to the Timer1 oscillator pins and the Timer1 oscillator has been enabled, the device can switch to a low power execution mode. Figure 2-7 shows a block diagram of the system clock sources. The clock switching feature is enabled by programming the Oscillator Switching Enable ($\overline{\text{OSCSEN}}$) bit in Configuration Register1H to a '0'. Clock switching is disabled in an erased device. See Section 9.0 for further details of the Timer1 oscillator. See Section 18.0 for Configuration Register details.

FIGURE 2-7: DEVICE CLOCK SOURCES



2.6.1 SYSTEM CLOCK SWITCH BIT

The system clock source switching is performed under software control. The system clock switch bit, SCS (OSCCON<0>) controls the clock switching. When the SCS bit is'0', the system clock source comes from the main oscillator that is selected by the FOSC configuration bits in Configuration Register1H. When the SCS bit is set, the system clock source will come from the Timer1 oscillator. The SCS bit is cleared on all forms of RESET.

Note:	The Timer1 oscillator must be enabled and operating to switch the system clock source. The Timer1 oscillator is enabled by setting the T1OSCEN bit in the Timer1 control register (T1CON). If the Timer1 oscillator is not enabled, then any write to the SCS bit will be ignored (SCS bit forced cleared) and the main oscillator will continue to be the system clock source.

REGISTER 2-1: OSCCON REGISTER

U-0	U-0	U-0	U-0	U-0	U-0	U-0	R/W-1
—	—	—	—	—	—	—	SCS
bit 7							bit 0

bit 7-1 **Unimplemented:** Read as '0'

bit 0 **SCS:** System Clock Switch bit
When $\overline{\text{OSCSEN}}$ configuration bit = '0' and T1OSCEN bit is set:
1 = Switch to Timer1 oscillator/clock pin
0 = Use primary oscillator/clock input pin
When $\overline{\text{OSCSEN}}$ and T1OSCEN are in other states:
bit is forced clear

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

4.10 Access Bank

The Access Bank is an architectural enhancement, which is very useful for C compiler code optimization. The techniques used by the C compiler may also be useful for programs written in assembly.

This data memory region can be used for:

- Intermediate computational values
- Local variables of subroutines
- Faster context saving/switching of variables
- Common variables
- Faster evaluation/control of SFRs (no banking)

The Access Bank is comprised of the upper 128 bytes in Bank 15 (SFRs) and the lower 128 bytes in Bank 0. These two sections will be referred to as Access RAM High and Access RAM Low, respectively. Figure 4-6 and Figure 4-7 indicate the Access RAM areas.

A bit in the instruction word specifies if the operation is to occur in the bank specified by the BSR register or in the Access Bank. This bit is denoted by the 'a' bit (for access bit).

When forced in the Access Bank (a = '0'), the last address in Access RAM Low is followed by the first address in Access RAM High. Access RAM High maps the Special Function registers, so that these registers can be accessed without any software overhead. This is useful for testing status flags and modifying control bits.

4.11 Bank Select Register (BSR)

The need for a large general purpose memory space dictates a RAM banking scheme. The data memory is partitioned into sixteen banks. When using direct addressing, the BSR should be configured for the desired bank.

BSR<3:0> holds the upper 4 bits of the 12-bit RAM address. The BSR<7:4> bits will always read '0's, and writes will have no effect.

A `MOVLB` instruction has been provided in the instruction set to assist in selecting banks.

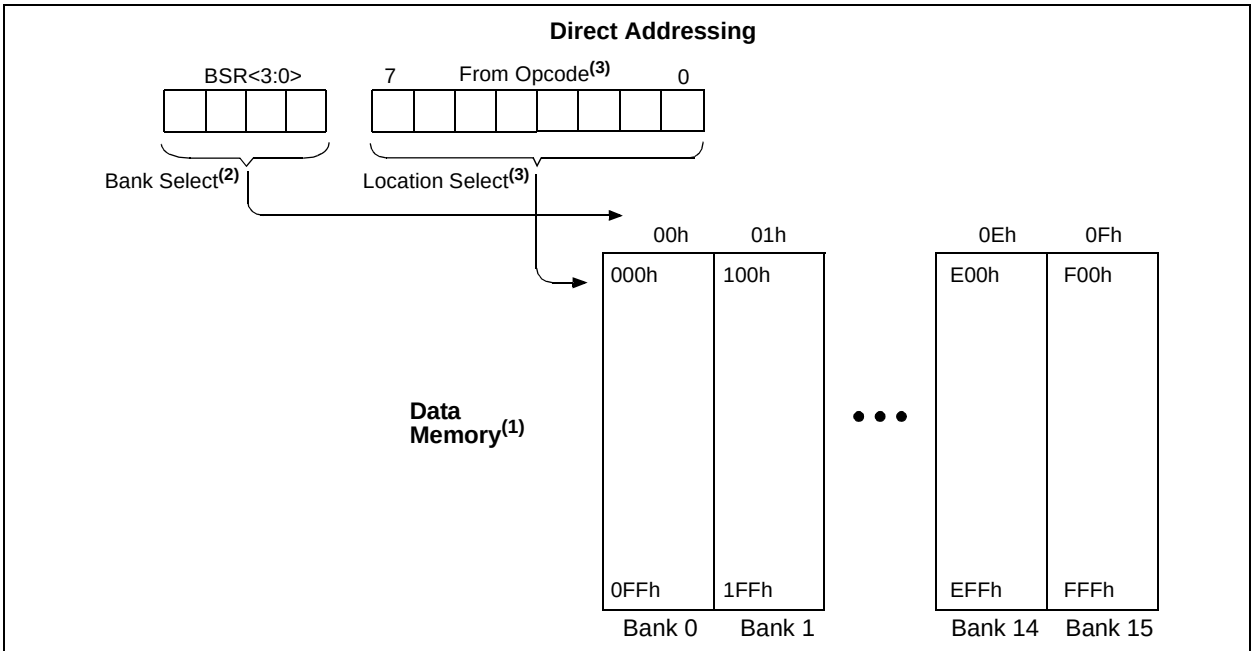
If the currently selected bank is not implemented, any read will return all '0's and all writes are ignored. The STATUS register bits will be set/cleared as appropriate for the instruction performed.

Each Bank extends up to FFh (256 bytes). All data memory is implemented as static RAM.

A `MOVFF` instruction ignores the BSR, since the 12-bit addresses are embedded into the instruction word.

Section 4.12 provides a description of indirect addressing, which allows linear addressing of the entire RAM space.

FIGURE 4-8: DIRECT ADDRESSING



Note 1: For register file map detail, see Table 4-1.

2: The access bit of the instruction can be used to force an override of the selected bank (BSR<3:0>) to the registers of the Access Bank.

3: The `MOVFF` instruction embeds the entire 12-bit address in the instruction.

NOTES:

REGISTER 8-1: TRISE REGISTER

R-0	R-0	R/W-0	R/W-0	U-0	R/W-1	R/W-1	R/W-1
IBF	OBF	IBOV	PSPMODE	—	TRISE2	TRISE1	TRISE0
bit 7							bit 0

bit 7 **IBF:** Input Buffer Full Status bit
1 = A word has been received and waiting to be read by the CPU
0 = No word has been received

bit 6 **OBF**: Output Buffer Full Status bit
1 = The output buffer still holds a previously written word
0 = The output buffer has been read

bit 5 **IBOV**: Input Buffer Overflow Detect bit (in Microprocessor mode)
1 = A write occurred when a previously input word has not been read (must be cleared in software)
0 = No overflow occurred

bit 4 **PSPMODE**: Parallel Slave Port Mode Select bit
1 = Parallel Slave Port mode
0 = General purpose I/O mode

bit 3 **Unimplemented:** Read as '0'

bit 2 **TRISE2**: RE2 Direction Control bit
1 = Input
0 = Output

bit 1 **TRISE1**: RE1 Direction Control bit
1 = Input
0 = Output

bit 0 **TRISE0**: RE0 Direction Control bit
1 = Input
0 = Output

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

TABLE 10-2: REGISTERS ASSOCIATED WITH TIMER1 AS A TIMER/COUNTER

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other RESETS
INTCON	GIE/GIEH	PEIE/ GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
PIR1	PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
PIE1	PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
IPR1	PSPIP(1)	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	0000 0000	0000 0000
TMR1L	Holding Register for the Least Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	uuuu uuuu
TMR1H	Holding Register for the Most Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	uuuu uuuu
T1CON	RD16	—	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON	--00 0000	--uu uuuu

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used by the Timer1 module.

Note 1: The PSPIF, PSPIE and PSPIP bits are reserved on the PIC18C2X2 devices. Always maintain these bits clear.

13.1 CCP1 Module

Capture/Compare/PWM Register 1 (CCPR1) is comprised of two 8-bit registers: CCPR1L (low byte) and CCPR1H (high byte). The CCP1CON register controls the operation of CCP1. All are readable and writable.

13.2 CCP2 Module

Capture/Compare/PWM Register2 (CCPR2) is comprised of two 8-bit registers: CCPR2L (low byte) and CCPR2H (high byte). The CCP2CON register controls the operation of CCP2. All are readable and writable.

TABLE 13-1: CCP MODE - TIMER RESOURCE

CCP Mode	Timer Resource
Capture	Timer1 or Timer3
Compare	Timer1 or Timer3
PWM	Timer2

TABLE 13-2: INTERACTION OF TWO CCP MODULES

CCPx Mode	CCPy Mode	Interaction
Capture	Capture	TMR1 or TMR3 time-base. Time-base can be different for each CCP.
Capture	Compare	The compare could be configured for the special event trigger, which clears either TMR1, or TMR3, depending upon which time-base is used.
Compare	Compare	The compare(s) could be configured for the special event trigger, which clears TMR1, or TMR3, depending upon which time-base is used.
PWM	PWM	The PWMs will have the same frequency and update rate (TMR2 interrupt).
PWM	Capture	None.
PWM	Compare	None.

TABLE 13-3: REGISTERS ASSOCIATED WITH CAPTURE, COMPARE, TIMER1 AND TIMER3

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other RESETS
INTCON	GIE/GIEH	PEIE/ GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 000x	0000 000u
PIR1	PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
PIE1	PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
IPR1	PSPIP(1)	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	0000 0000	0000 0000
TRISC	PORTC Data Direction Register								1111 1111	1111 1111
TMR1L	Holding Register for the Least Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	uuuu uuuu
TMR1H	Holding Register for the Most Significant Byte of the 16-bit TMR1 Register								xxxx xxxx	uuuu uuuu
T1CON	RD16	—	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON	--00 0000	--uu uuuu
CCPR1L	Capture/Compare/PWM Register1 (LSB)								xxxx xxxx	uuuu uuuu
CCPR1H	Capture/Compare/PWM Register1 (MSB)								xxxx xxxx	uuuu uuuu
CCP1CON	—	—	DC1B1	DC1B0	CCP1M3	CCP1M2	CCP1M1	CCP1M0	--00 0000	--00 0000
CCPR2L	Capture/Compare/PWM Register2 (LSB)								xxxx xxxx	uuuu uuuu
CCPR2H	Capture/Compare/PWM Register2 (MSB)								xxxx xxxx	uuuu uuuu
CCP2CON	—	—	DC2B1	DC2B0	CCP2M3	CCP2M2	CCP2M1	CCP2M0	--00 0000	--00 0000
PIR2	—	—	—	—	BCLIF	LVDIF	TMR3IF	CCP2IF	0000 0000	0000 0000
PIE2	—	—	—	—	BCLIE	LVDIE	TMR3IE	CCP2IE	0000 0000	0000 0000
IPR2	—	—	—	—	BCLIP	LVDIP	TMR3IP	CCP2IP	0000 0000	0000 0000
TMR3L	Holding Register for the Least Significant Byte of the 16-bit TMR3 Register								xxxx xxxx	uuuu uuuu
TMR3H	Holding Register for the Most Significant Byte of the 16-bit TMR3 Register								xxxx xxxx	uuuu uuuu
T3CON	RD16	T3CCP2	T3CKPS1	T3CKPS0	T3CCP1	$\overline{\text{T3SYNC}}$	TMR3CS	TMR3ON	-000 0000	-uuu uuuu

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used by Capture and Timer1.

Note 1: The PSPIF, PSPIE and PSPIP bits are reserved on the PIC18C2X2 devices. Always maintain these bits clear.

14.3 SPI Mode

The SPI mode allows 8-bits of data to be synchronously transmitted and received simultaneously. All four modes of SPI are supported. To accomplish communication, typically three pins are used:

- Serial Data Out (SDO) - RC5/SDO
- Serial Data In (SDI) - RC4/SDI/SDA
- Serial Clock (SCK) - RC3/SCK/SCL/LVOIN

Additionally, a fourth pin may be used when in a Slave mode of operation:

- Slave Select ($\overline{SS}$) - RA5/$\overline{SS}$/AN4

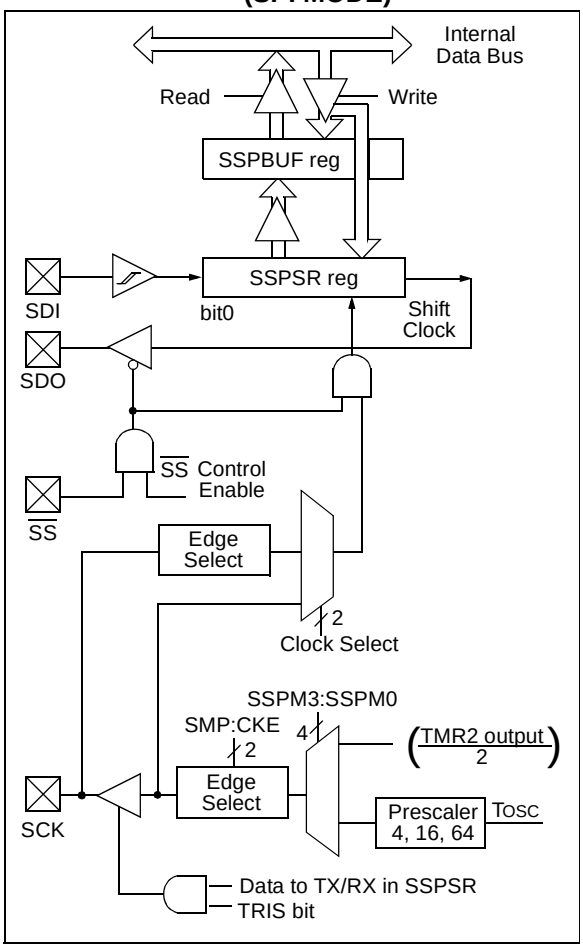
14.3.1 OPERATION

When initializing the SPI, several options need to be specified. This is done by programming the appropriate control bits (SSPCON1<5:0>) and SSPSTAT<7:6>. These control bits allow the following to be specified:

- Master mode (SCK is the clock output)
- Slave mode (SCK is the clock input)
- Clock Polarity (Idle state of SCK)
- Data input sample phase (middle or end of data output time)
- Clock edge (output data on rising/falling edge of SCK)
- Clock Rate (Master mode only)
- Slave Select mode (Slave mode only)

Figure 14-1 shows the block diagram of the MSSP module, when in SPI mode.

FIGURE 14-1: MSSP BLOCK DIAGRAM (SPI MODE)



The MSSP consists of a transmit/receive shift register (SSPSR) and a buffer register (SSPBUF). The SSPSR shifts the data in and out of the device, MSb first. The SSPBUF holds the data that was written to the SSPSR, until the received data is ready. Once the 8-bits of data have been received, that byte is moved to the SSPBUF register. Then the buffer full detect bit, BF (SSPSTAT<0>), and the interrupt flag bit, SSPIF, are set. This double buffering of the received data (SSPBUF) allows the next byte to start reception before reading the data that was just received. Any write to the SSPBUF register during transmission/reception of data will be ignored, and the write collision detect bit, WCOL (SSPCON1<7>), will be set. User software must clear the WCOL bit so that it can be determined if the following write(s) to the SSPBUF register completed successfully.

14.4.2 GENERAL CALL ADDRESS SUPPORT

The addressing procedure for the I^2C bus is such that the first byte after the START condition usually determines which device will be the slave addressed by the master. The exception is the general call address which can address all devices. When this address is used, all devices should, in theory, respond with an acknowledge.

The general call address is one of eight addresses reserved for specific purposes by the I^2C protocol. It consists of all 0's with R/$\overline{W}$ = 0.

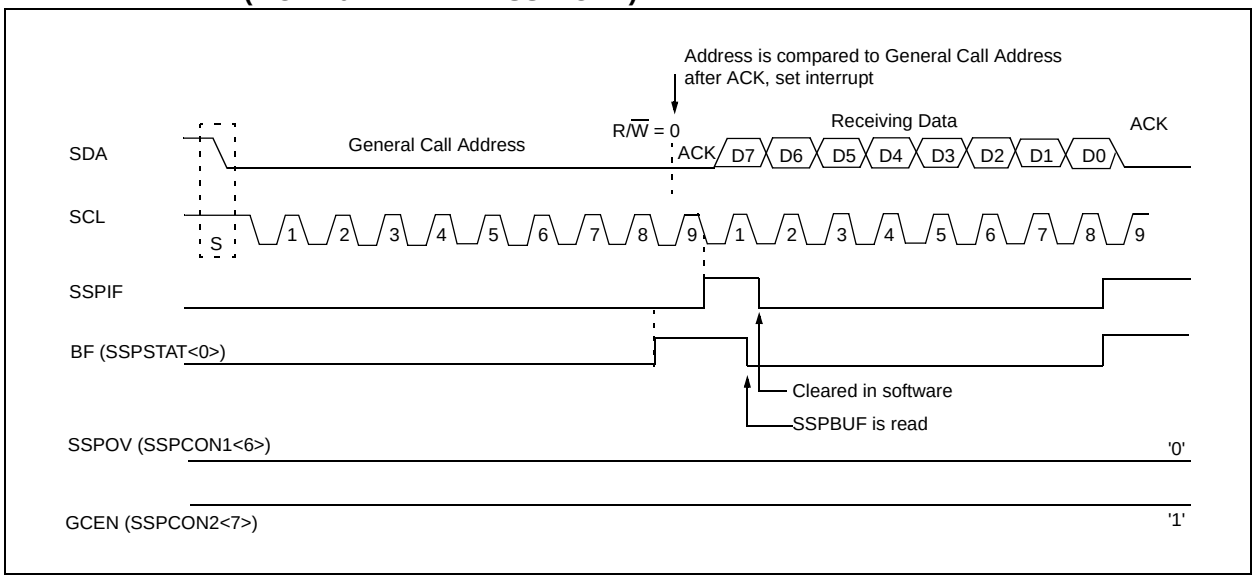
The general call address is recognized when the General Call Enable bit (GCEN) is enabled (SSPCON2<7> is set). Following a START bit detect, 8-bits are shifted into the SSPSR and the address is compared against the SSPADD. It is also compared to the general call address and fixed in hardware.

If the general call address matches, the SSPSR is transferred to the SSPBUF, the BF flag bit is set (eighth bit), and on the falling edge of the ninth bit ($\overline{ACK}$ bit), the SSPIF interrupt flag bit is set.

When the interrupt is serviced, the source for the interrupt can be checked by reading the contents of the SSPBUF. The value can be used to determine if the address was device specific or a general call address.

In 10-bit mode, the SSPADD is required to be updated for the second half of the address to match, and the UA bit is set (SSPSTAT<1>). If the general call address is sampled when the GCEN bit is set, while the slave is configured in 10-bit address mode, then the second half of the address is not necessary, the UA bit will not be set, and the slave will begin receiving data after the Acknowledge (Figure 14-12).

FIGURE 14-12: SLAVE MODE GENERAL CALL ADDRESS SEQUENCE (7 OR 10-BIT ADDRESS MODE)



14.4.3 MASTER MODE

Master mode of operation is supported by interrupt generation on the detection of the START and STOP conditions. The STOP (P) and START (S) bits are cleared from a RESET or when the MSSP module is disabled. Control of the I^2C bus may be taken when the P bit is set, or the bus is idle, with both the S and P bits clear.

In Master mode, the SCL and SDA lines are manipulated by the MSSP hardware.

The following events will cause SSP Interrupt Flag bit, SSPIF, to be set (SSP interrupt, if enabled):

- START condition
- STOP condition
- Data transfer byte transmitted/received
- Acknowledge Transmit
- Repeated START

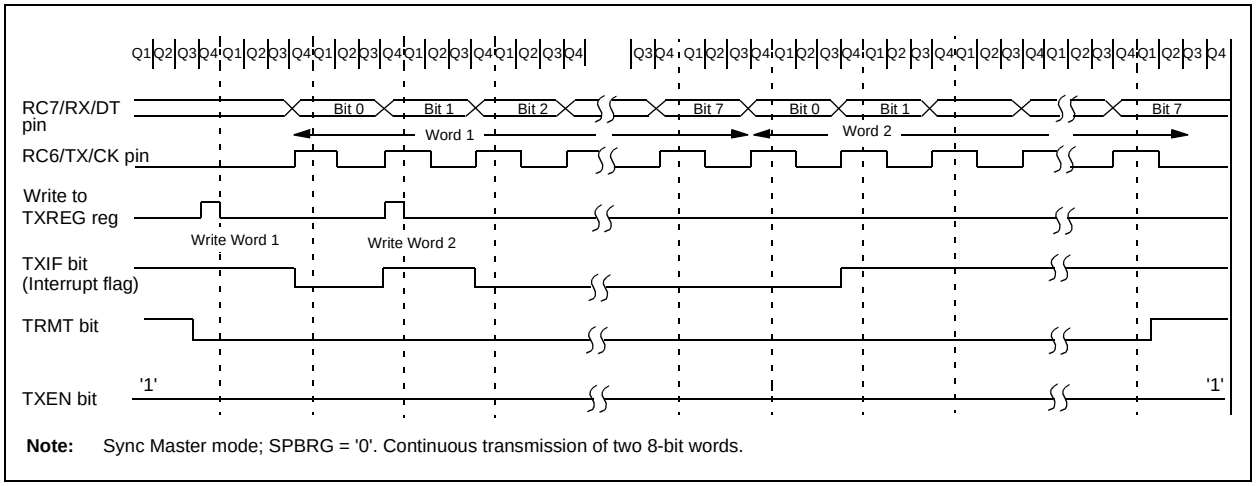
TABLE 15-4: BAUD RATES FOR ASYNCHRONOUS MODE (BRGH = 0)

BAUD RATE (K)	Fosc = 40 MHz			Fosc = 20 MHz			Fosc = 16 MHz			Fosc = 10 MHz		
	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)
0.3	NA	—	—	NA	—	—	NA	—	—	NA	—	—
1.2	NA	—	—	1.221	+1.73	255	1.202	+0.16	207	1.202	+0.16	129
2.4	2.44	-1.70	255	2.404	+0.16	129	2.404	+0.16	103	2.404	+0.16	64
9.6	9.62	-0.16	64	9.469	-1.36	32	9.615	+0.16	25	9.766	+1.73	15
19.2	18.94	+1.38	32	19.53	+1.73	15	19.23	+0.16	12	19.53	+1.73	7
76.8	78.13	-1.70	7	78.13	+1.73	3	83.33	+8.51	2	78.13	+1.73	1
96	89.29	+7.52	6	104.2	+8.51	2	NA	—	—	NA	—	—
300	312.50	-4.00	1	312.5	+4.17	0	NA	—	—	NA	—	—
500	625.00	-20.00	0	NA	—	—	NA	—	—	NA	—	—
HIGH	2.44	—	255	312.5	—	0	250	—	0	156.3	—	0
LOW	625.00	—	0	1.221	—	255	0.977	—	255	0.6104	—	255

BAUD RATE (K)	Fosc = 7.15909 MHz			Fosc = 5.0688 MHz			Fosc = 4 MHz			Fosc = 3.579545 MHz		
	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)
0.3	NA	—	—	0.31	+3.13	255	0.3005	-0.17	207	0.301	+0.23	185
1.2	1.203	+0.23	92	1.2	0	65	1.202	+1.67	51	1.190	-0.83	46
2.4	2.380	-0.83	46	2.4	0	32	2.404	+1.67	25	2.432	+1.32	22
9.6	9.322	-2.90	11	9.9	+3.13	7	NA	—	—	9.322	-2.90	5
19.2	18.64	-2.90	5	19.8	+3.13	3	NA	—	—	18.64	-2.90	2
76.8	NA	—	—	79.2	+3.13	0	NA	—	—	NA	—	—
96	NA	—	—	NA	—	—	NA	—	—	NA	—	—
300	NA	—	—	NA	—	—	NA	—	—	NA	—	—
500	NA	—	—	NA	—	—	NA	—	—	NA	—	—
HIGH	111.9	—	0	79.2	—	0	62.500	—	0	55.93	—	0
LOW	0.437	—	255	0.3094	—	255	3.906	—	255	0.2185	—	255

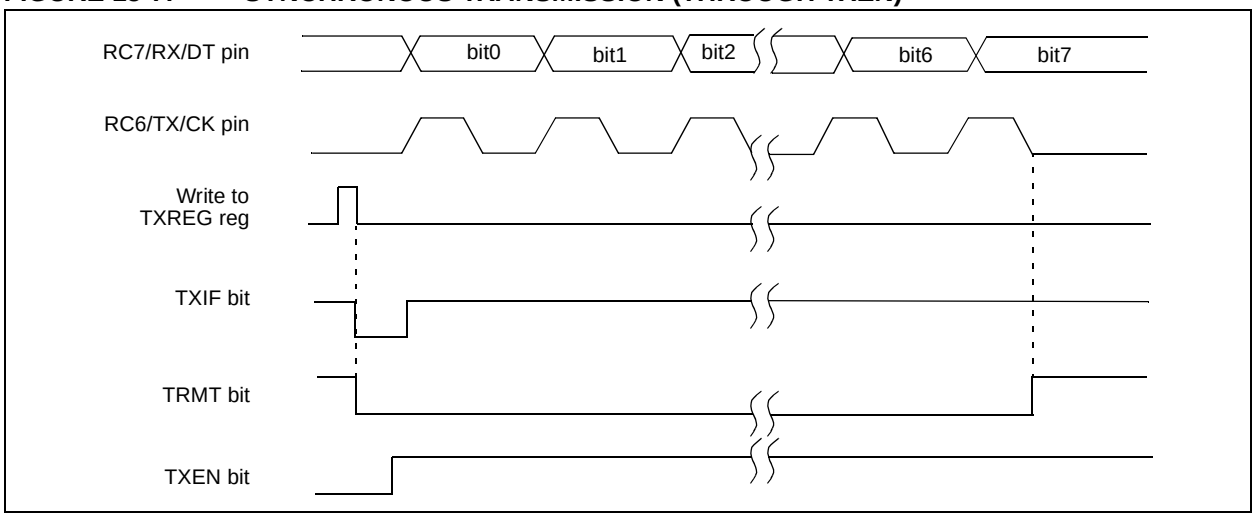
BAUD RATE (K)	Fosc = 1 MHz			Fosc = 32.768 kHz		
	Actual Rate (K)	% Error	SPBRG value (decimal)	Actual Rate (K)	% Error	SPBRG value (decimal)
0.3	0.300	+0.16	51	0.256	-14.67	1
1.2	1.202	+0.16	12	NA	—	—
2.4	2.232	-6.99	6	NA	—	—
9.6	NA	—	—	NA	—	—
19.2	NA	—	—	NA	—	—
76.8	NA	—	—	NA	—	—
96	NA	—	—	NA	—	—
300	NA	—	—	NA	—	—
500	NA	—	—	NA	—	—
HIGH	15.63	—	0	0.512	—	0
LOW	0.0610	—	255	0.0020	—	255

FIGURE 15-6: SYNCHRONOUS TRANSMISSION



Note: Sync Master mode; SPBRG = '0'. Continuous transmission of two 8-bit words.

FIGURE 15-7: SYNCHRONOUS TRANSMISSION (THROUGH TXEN)



To calculate the minimum acquisition time, Equation 16-1 may be used. This equation assumes that 1/2 LSb error is used (1024 steps for the A/D). The 1/2 LSb error is the maximum error allowed for the A/D to meet its specified resolution.

EQUATION 16-1: ACQUISITION TIME

$$\begin{aligned} T_{ACQ} &= \text{Amplifier Settling Time} + \text{Holding Capacitor Charging Time} + \text{Temperature Coefficient} \\ &= T_{AMP} + T_C + T_{COFF} \end{aligned}$$

EQUATION 16-2: A/D MINIMUM CHARGING TIME

$$\begin{aligned} V_{HOLD} &= (V_{REF} - (V_{REF}/2048)) \bullet (1 - e^{(-T_C/C_{HOLD}(R_{IC} + R_{SS} + R_S))}) \\ \text{or} & \\ T_C &= -(120\ pF)(1\ k\Omega + R_{SS} + R_S)\ \ln(1/2047) \end{aligned}$$

Example 16-1 shows the calculation of the minimum required acquisition time TACQ. This calculation is based on the following application system assumptions:

- CHOLD = 120 pF
- Rs = 2.5 kΩ
- Conversion Error ≤ 1/2 LSb
- VDD = 5V → Rss = 7 kΩ
- Temperature = 50°C (system max.)
- VHOLD = 0V @ time = 0

EXAMPLE 16-1: CALCULATING THE MINIMUM REQUIRED ACQUISITION TIME

$T_{ACQ} = T_{AMP} + T_C + T_{COFF}$

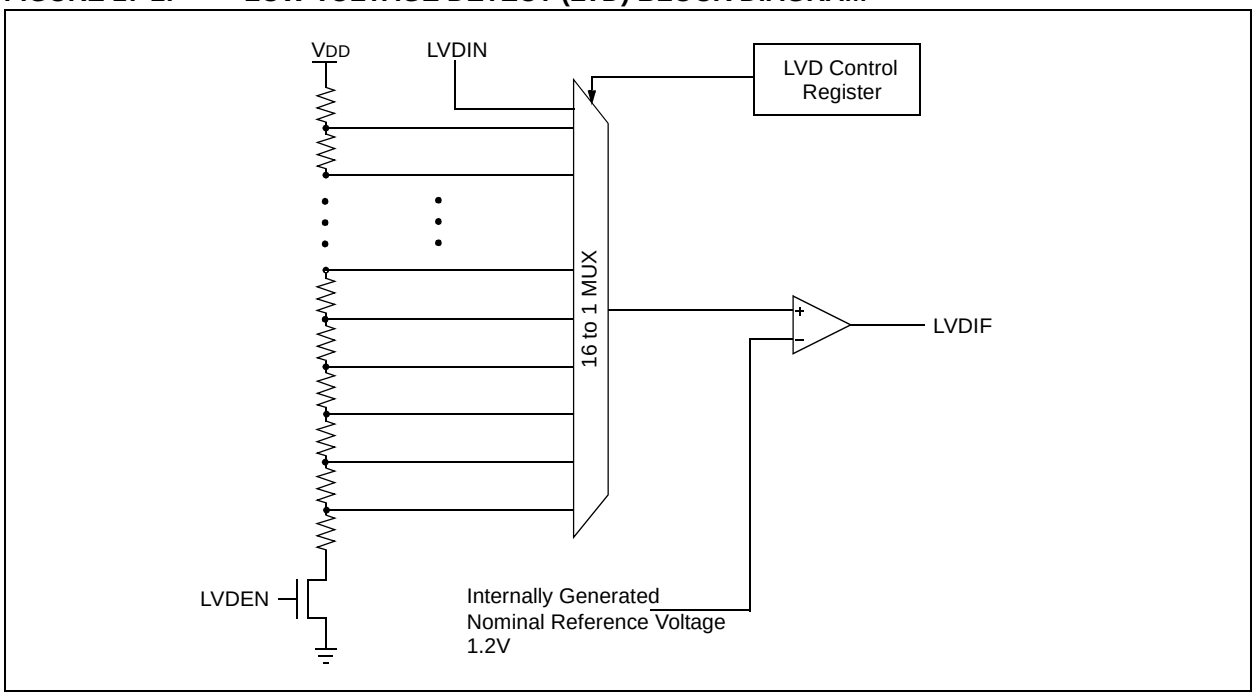
Temperature coefficient is only required for temperatures > 25°C.

$T_{ACQ} = 2\ \mu s + Tc + [(Temp - 25°C)(0.05\ \mu s/°C)]$

$$\begin{aligned} T_C &= -C_{HOLD}\ (R_{IC} + R_{SS} + R_S)\ \ln(1/2047) \\ &\quad -120\ pF\ (1\ k\Omega + 7\ k\Omega + 2.5\ k\Omega)\ \ln(0.0004885) \\ &\quad -120\ pF\ (10.5\ k\Omega)\ \ln(0.0004885) \\ &\quad -1.26\ \mu s\ (-7.6241) \\ &\quad 9.61\ \mu s \end{aligned}$$

$$\begin{aligned} T_{ACQ} &= 2\ \mu s + 9.61\ \mu s + [(50°C - 25°C)(0.05\ \mu s/°C)] \\ &\quad 11.61\ \mu s + 1.25\ \mu s \\ &\quad 12.86\ \mu s \end{aligned}$$

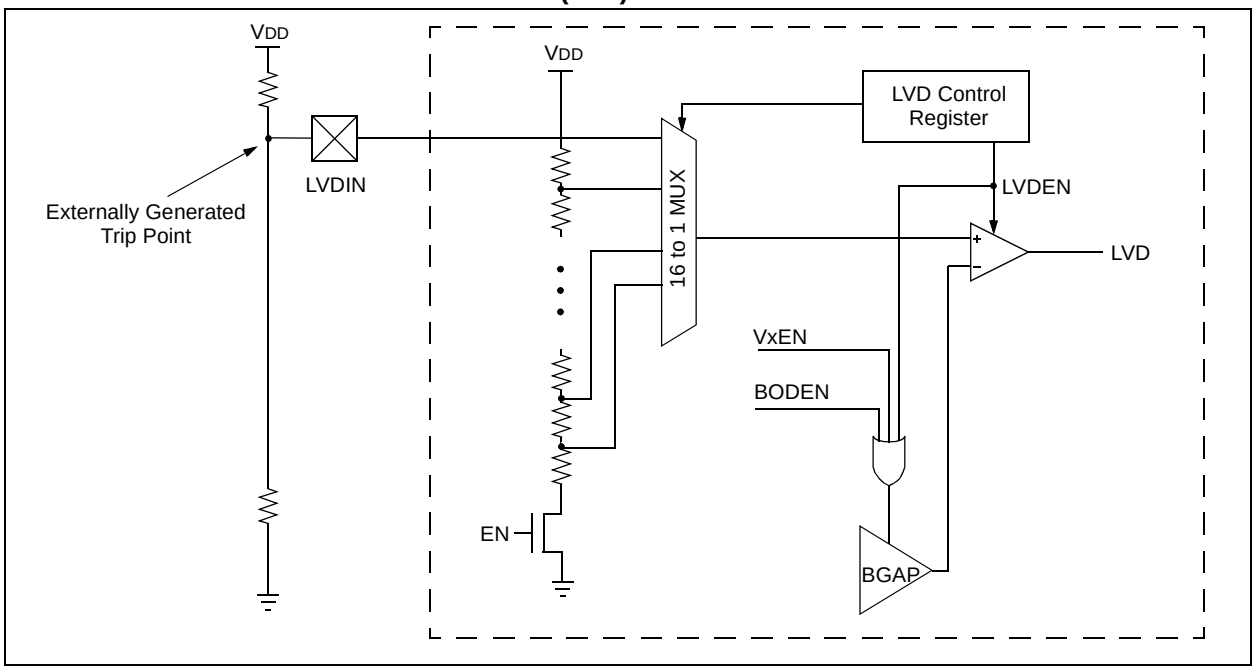
FIGURE 17-2: LOW VOLTAGE DETECT (LVD) BLOCK DIAGRAM



The LVD module has an additional feature that allows the user to supply the trip voltage to the module from an external source. This mode is enabled when bits LVDL3:LVDL0 are set to `1111`. In this state, the comparator input is multiplexed from the external input pin LVDIN (Figure 17-3).

This gives flexibility, because it allows a user to configure the Low Voltage Detect interrupt to occur at any voltage in the valid operating range.

FIGURE 17-3: LOW VOLTAGE DETECT (LVD) WITH EXTERNAL INPUT BLOCK DIAGRAM



19.0 INSTRUCTION SET SUMMARY

The PIC18CXXX instruction set adds many enhancements to the previous PIC instruction sets, while maintaining an easy migration from these PIC MCU instruction sets.

Most instructions are a single program memory word (16-bits), but there are three instructions that require two program memory locations.

Each single word instruction is a 16-bit word divided into an OPCODE, which specifies the instruction type and one or more operands, which further specify the operation of the instruction.

The instruction set is highly orthogonal and is grouped into four basic categories:

- **Byte-oriented** operations
- **Bit-oriented** operations
- **Literal** operations
- **Control** operations

The PIC18CXXX instruction set summary in Table 19-2 lists **byte-oriented**, **bit-oriented**, **literal** and **control** operations. Table 19-1 shows the opcode field descriptions.

Most **byte-oriented** instructions have three operands:

1. The file register (specified by ‘f’)
2. The destination of the result (specified by ‘d’)
3. The accessed memory (specified by ‘a’)

The file register designator 'f' specifies which file register is to be used by the instruction.

The destination designator ‘d’ specifies where the result of the operation is to be placed. If 'd' is zero, the result is placed in the WREG register. If 'd' is one, the result is placed in the file register specified in the instruction.

All **bit-oriented** instructions have three operands:

1. The file register (specified by ‘f’)
2. The bit in the file register (specified by ‘b’)
3. The accessed memory (specified by ‘a’)

The bit field designator 'b' selects the number of the bit affected by the operation, while the file register designator 'f' represents the number of the file in which the bit is located.

The **literal** instructions may use some of the following operands:

- A literal value to be loaded into a file register (specified by ‘k’)
- The desired FSR register to load the literal value into (specified by ‘f’)
- No operand required (specified by ‘—’)

The **control** instructions may use some of the following operands:

- A program memory address (specified by ‘n’)
- The mode of the Call or Return instructions (specified by ‘s’)
- The mode of the Table Read and Table Write instructions (specified by ‘m’)
- No operand required (specified by ‘—’)

All instructions are a single word, except for three double word instructions. These three instructions were made double word instructions so that all the required information is available in these 32-bits. In the second word, the 4 MSb’s are 1’s. If this second word is executed as an instruction (by itself), it will execute as a `NOP`.

All single word instructions are executed in a single instruction cycle, unless a conditional test is true or the program counter is changed as a result of the instruction. In these cases, the execution takes two instruction cycles, with the additional instruction cycle(s) executed as a `NOP`.

The double word instructions execute in two instruction cycles.

One instruction cycle consists of four oscillator periods. Thus, for an oscillator frequency of 4 MHz, the normal instruction execution time is 1 μs. If a conditional test is true, or the program counter is changed as a result of an instruction, the instruction execution time is 2 μs. Two word branch instructions (if true) would take 3 μs.

Figure 19-1 shows the general formats that the instructions can have.

All examples use the format `‘nnh’` to represent a hexadecimal number, where `‘h’` signifies a hexadecimal digit.

The Instruction Set Summary, shown in Table 19-2, lists the instructions recognized by the Microchip assembler (MPASM™).

Section 19.1 provides a description of each instruction.

RETURN **Return from Subroutine**

Syntax: [*label*] RETURN [s]

Operands: s ∈ [0,1]

Operation: (TOS) → PC,
if s = 1
(WS) → WREG,
(STATUSS) → STATUS,
(BSRS) → BSR,
PCLATU, PCLATH are unchanged

Status Affected: None

Encoding:

0000	0000	0001	001s

Description: Return from subroutine. The stack is popped and the top of the stack (TOS) is loaded into the program counter. If 's'= 1, the contents of the shadow registers WS, STATUSS and BSRS are loaded into their corresponding registers, WREG, STATUS and BSR. If 's' = 0, no update of these registers occurs (default).

Words: 1

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	Process Data	pop PC from stack
No operation	No operation	No operation	No operation

Example: RETURN

After Interrupt
PC = TOS

RLCF **Rotate Left f through Carry**

Syntax: [*label*] RLCF f [,d [,a]

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

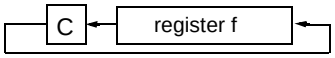
Operation: (f<n>) → dest<n+1>,
(f<7>) → C,
(C) → dest<0>

Status Affected: C,N,Z

Encoding:

0011	01da	ffff	ffff

Description: The contents of register 'f' are rotated one bit to the left through the Carry Flag. If 'd' is 0, the result is placed in WREG. If 'd' is 1, the result is stored back in register 'f' (default). If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' = 1, then the bank will be selected as per the BSR value (default).



Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: RLCF REG, 0, 0

Before Instruction
REG = 1110 0110
C = 0

After Instruction
REG = 1110 0110
WREG = 1100 1100
C = 1

RRNCF **Rotate Right f (no carry)**

Syntax: [*label*] RRNCF f [,d [,a]

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

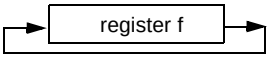
Operation: (f<n>) → dest<n-1>,
(f<0>) → dest<7>

Status Affected: N,Z

Encoding:

0100	00da	ffff	ffff

Description: The contents of register 'f' are rotated one bit to the right. If 'd' is 0, the result is placed in WREG. If 'd' is 1, the result is placed back in register 'f' (default). If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' is 1, then the bank will be selected as per the BSR value (default).



Words: 1

Cycles: 1

Q Cycle Activity:

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
```
WREG    =   ?
REG     =   1101 0111
```
After Instruction
```
WREG    =   1110 1011
REG     =   1101 0111
```

SETF **Set f**

Syntax: [*label*] SETF f [,a]

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

Operation: FFh → f

Status Affected: None

Encoding:

0110	100a	ffff	ffff

Description: The contents of the specified register are set to FFh. If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' is 1, then the bank will be selected as per the BSR value (default).

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     =   0x5A
```
After Instruction
```
REG     =   0xFF
```

20.0 DEVELOPMENT SUPPORT

The PIC® microcontrollers are supported with a full range of hardware and software development tools:

- Integrated Development Environment
  - MPLAB® IDE Software
- Assemblers/Compilers/Linkers
  - MPASM™ Assembler
  - MPLAB C17 and MPLAB C18 C Compilers
  - MPLINK™ Object Linker/ MPLIB™ Object Librarian
- Simulators
  - MPLAB SIM Software Simulator
- Emulators
  - MPLAB ICE 2000 In-Circuit Emulator
  - ICEPIC™ In-Circuit Emulator
- In-Circuit Debugger
  - MPLAB ICD for PIC16F87X
- Device Programmers
  - PRO MATE® II Universal Device Programmer
  - PICSTART® Plus Entry-Level Development Programmer
- Low Cost Demonstration Boards
  - PICDEM™ 1 Demonstration Board
  - PICDEM 2 Demonstration Board
  - PICDEM 3 Demonstration Board
  - PICDEM 17 Demonstration Board
  - KEELOQ® Demonstration Board

20.1 MPLAB Integrated Development Environment Software

The MPLAB IDE software brings an ease of software development previously unseen in the 8-bit microcontroller market. The MPLAB IDE is a Windows®-based application that contains:

- An interface to debugging tools
  - simulator
  - programmer (sold separately)
  - emulator (sold separately)
  - in-circuit debugger (sold separately)
- A full-featured editor
- A project manager
- Customizable toolbar and key mapping
- A status bar
- On-line help

The MPLAB IDE allows you to:

- Edit your source files (either assembly or ‘C’)
- One touch assemble (or compile) and download to PIC MCU emulator and simulator tools (automatically updates all project information)
- Debug using:
  - source files
  - absolute listing file
  - machine code

The ability to use MPLAB IDE with multiple debugging tools allows users to easily switch from the cost-effective simulator to a full-featured emulator with minimal retraining.

20.2 MPASM Assembler

The MPASM assembler is a full-featured universal macro assembler for all PIC MCUs.

The MPASM assembler has a command line interface and a Windows shell. It can be used as a stand-alone application on a Windows 3.x or greater system, or it can be used through MPLAB IDE. The MPASM assembler generates relocatable object files for the MPLINK object linker, Intel® standard HEX files, MAP files to detail memory usage and symbol reference, an absolute LST file that contains source lines and generated machine code, and a COD file for debugging.

The MPASM assembler features include:

- Integration into MPLAB IDE projects.
- User-defined macros to streamline assembly code.
- Conditional assembly for multi-purpose source files.
- Directives that allow complete control over the assembly process.

20.3 MPLAB C17 and MPLAB C18 C Compilers

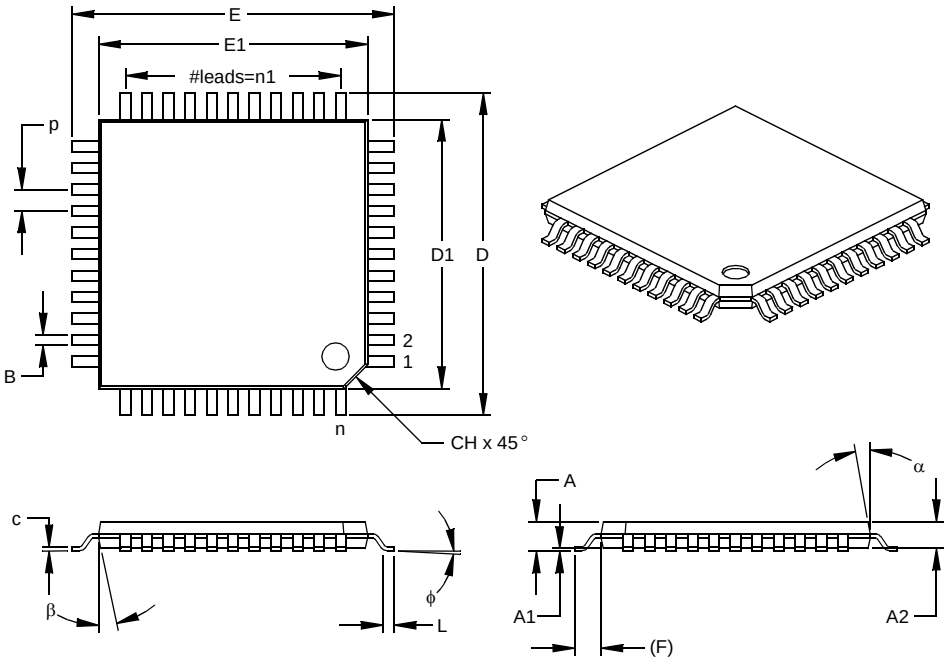
The MPLAB C17 and MPLAB C18 Code Development Systems are complete ANSI ‘C’ compilers for Microchip’s PIC17CXXX and PIC18CXXX family of microcontrollers, respectively. These compilers provide powerful integration capabilities and ease of use not found with other compilers.

For easier source level debugging, the compilers provide symbol information that is compatible with the MPLAB IDE memory display.

NOTES:

44-Lead Plastic Thin Quad Flatpack (PT) 10x10x1 mm Body, 1.0/0.10 mm Lead Form (TQFP)

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



Units		INCHES			MILLIMETERS*		
Dimension Limits		MIN	NOM	MAX	MIN	NOM	MAX
Number of Pins	n		44			44	
Pitch	p		.031			0.80	
Pins per Side	n1		11			11	
Overall Height	A	.039	.043	.047	1.00	1.10	1.20
Molded Package Thickness	A2	.037	.039	.041	0.95	1.00	1.05
Standoff §	A1	.002	.004	.006	0.05	0.10	0.15
Foot Length	L	.018	.024	.030	0.45	0.60	0.75
Footprint (Reference)	(F)		.039		1.00		
Foot Angle	ϕ	0	3.5	7	0	3.5	7
Overall Width	E	.463	.472	.482	11.75	12.00	12.25
Overall Length	D	.463	.472	.482	11.75	12.00	12.25
Molded Package Width	E1	.390	.394	.398	9.90	10.00	10.10
Molded Package Length	D1	.390	.394	.398	9.90	10.00	10.10
Lead Thickness	c	.004	.006	.008	0.09	0.15	0.20
Lead Width	B	.012	.015	.017	0.30	0.38	0.44
Pin 1 Corner Chamfer	CH	.025	.035	.045	0.64	0.89	1.14
Mold Draft Angle Top	α	5	10	15	5	10	15
Mold Draft Angle Bottom	β	5	10	15	5	10	15

* Controlling Parameter
§ Significant Characteristic

Notes:
Dimensions D1 and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010” (0.254mm) per side.
JEDEC Equivalent: MS-026
Drawing No. C04-076

NOTES: