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	CANbus, I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, LVD, POR, PWM, WDT
Number of I/O	68
Program Memory Size	32KB (16K x 16)
Program Memory Type	OTP
EEPROM Size	-
RAM Size	1.5K x 8
Voltage - Supply (Vcc/Vdd)	4.2V ~ 5.5V
Data Converters	A/D 16x10b
Oscillator Type	External
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	80-TQFP
Supplier Device Package	80-TQFP (12x12)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18c858t-i-pt

FIGURE 1-2: PIC18C858 BLOCK DIAGRAM

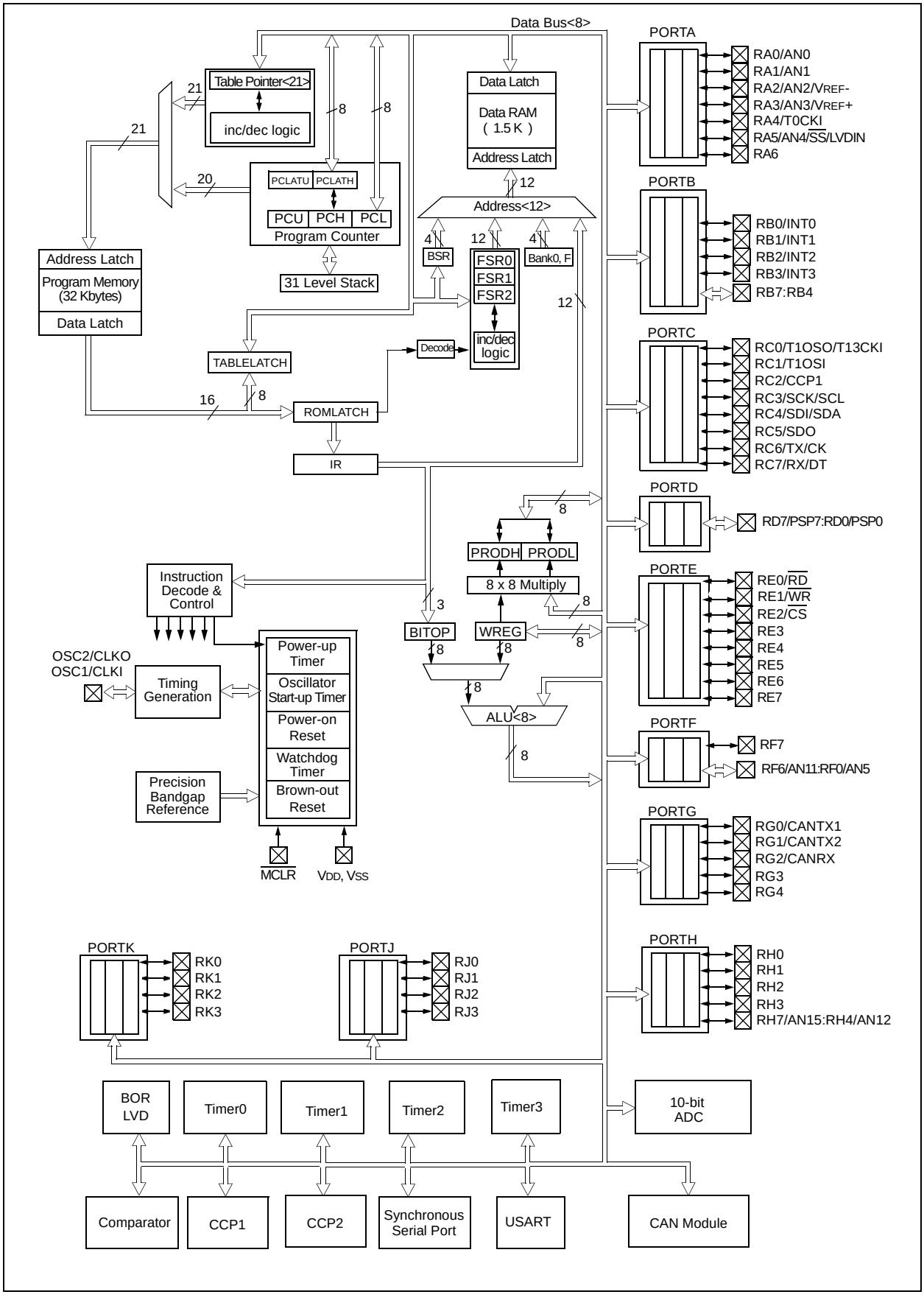


TABLE 1-2: PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number				Pin Type	Buffer Type	Description
	PIC18C658		PIC18C858				
	TQFP	PLCC	TQFP	PLCC			
							PORTE is a bi-directional I/O port
RE0/$\overline{RD}$	2	11	4	15			
RE0					I/O	ST	Digital I/O
$\overline{RD}$					I	TTL	Read control for parallel slave port (See $\overline{WR}$ and $\overline{CS}$ pins)
RE1/$\overline{WR}$	1	10	3	14			
RE1					I/O	ST	Digital I/O
$\overline{WR}$					I	TTL	Write control for parallel slave port (See $\overline{CS}$ and $\overline{RD}$ pins)
RE2/$\overline{CS}$	64	9	78	9			
RE2					I/O	ST	Digital I/O
$\overline{CS}$					I	TTL	Chip select control for parallel slave port (See $\overline{RD}$ and $\overline{WR}$)
RE3	63	8	77	8	I/O	ST	Digital I/O
RE4	62	7	76	7	I/O	ST	Digital I/O
RE5	61	6	75	6	I/O	ST	Digital I/O
RE6	60	5	74	5	I/O	ST	Digital I/O
RE7/CCP2	59	4	73	4			
RE7					I/O	ST	Digital I/O
CCP2					I/O	ST	Capture2 input, Compare2 output, PWM2 output

Legend: TTL = TTL compatible input
ST = Schmitt Trigger input with CMOS levels
I = Input
P = Power
CMOS = CMOS compatible input or output
Analog = Analog input
O = Output
OD = Open Drain (no P diode to V_{DD})

TABLE 3-3: INITIALIZATION CONDITIONS FOR ALL REGISTERS (CONTINUED)

Register	Applicable Devices		Power-on Reset, Brown-out Reset	$\overline{\text{MCLR}}$ Reset WDT Reset RESET Instruction Stack Resets	Wake-up via WDT or Interrupt
RXM1SIDL	658	858	xxx- --xx	uuu- --uu	uuu- --uu
RXM1SIDH	658	858	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXM0EIDL	658	858	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXM0EIDH	658	858	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXM0SIDL	658	858	xxx- --xx	uuu- --uu	uuu- --uu
RXM0SIDH	658	858	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF5EIDL	658	858	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF5EIDH	658	858	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF5SIDL	658	858	xxx- x-xx	uuu- u-uu	uuu- u-uu
RXF5SIDH	658	858	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF4EIDL	658	858	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF4EIDH	658	858	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF4SIDL	658	858	xxx- x-xx	uuu- u-uu	uuu- u-uu
RXF4SIDH	658	858	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF3EIDL	658	858	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF3EIDH	658	858	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF3SIDL	658	858	xxx- x-xx	uuu- u-uu	uuu- u-uu
RXF3SIDH	658	858	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF2EIDL	658	858	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF2EIDH	658	858	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF2SIDL	658	858	xxx- x-xx	uuu- u-uu	uuu- u-uu
RXF2SIDH	658	858	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF1EIDL	658	858	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF1EIDH	658	858	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF1SIDL	658	858	xxx- x-xx	uuu- u-uu	uuu- u-uu
RXF1SIDH	658	858	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF0EIDL	658	858	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF0EIDH	658	858	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF0SIDL	658	858	xxx- x-xx	uuu- u-uu	uuu- u-uu
RXF0SIDH	658	858	xxxx xxxx	uuuu uuuu	uuuu uuuu

Legend: u = unchanged, x = unknown, - = unimplemented bit, read as '0', q = value depends on condition

Note 1: One or more bits in the INTCONx or PIRx registers will be affected (to cause wake-up).

2: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the PC is loaded with the interrupt vector (0008h or 0018h).

3: When the wake-up is due to an interrupt and the GIEL or GIEH bit is set, the TOSU, TOSH and TOSL are updated with the current value of the PC. The STKPTR is modified to point to the next location in the hardware stack.

4: See Table 3-2 for RESET value for specific condition.

5: Bit 6 of PORTA, LATA, and TRISA are enabled in ECIO and RCIO oscillator modes only. In all other oscillator modes, they are disabled and read '0'.

6: The long write enable is only reset on a POR or $\overline{\text{MCLR}}$.

7: Available on PIC18C858 only.

4.2 Return Address Stack

The return address stack allows any combination of up to 31 program calls and interrupts to occur. The PC (Program Counter) is pushed onto the stack when a PUSH, CALL or RCALL instruction is executed, or an interrupt is acknowledged. The PC value is pulled off the stack on a RETURN, RETLW or a RETFIE instruction. PCLATU and PCLATH are not affected by any of the return instructions.

The stack operates as a 31 word by 21-bit stack memory and a 5-bit stack pointer, with the stack pointer initialized to 00000b after all RESETs. There is no RAM associated with stack pointer 00000b. This is only a RESET value. During a CALL type instruction causing a push onto the stack, the stack pointer is first incremented and the RAM location pointed to by the stack pointer is written with the contents of the PC. During a RETURN type instruction causing a pop from the stack, the contents of the RAM location indicated by the STKPTR is transferred to the PC and then the stack pointer is decremented.

The stack space is not part of either program or data space. The stack pointer is readable and writable, and the data on the top of the stack is readable and writable through SFR registers. Status bits indicate if the stack pointer is at or beyond the 31 levels provided.

4.2.1 TOP-OF-STACK ACCESS

The top of the stack is readable and writable. Three register locations, TOSU, TOSH and TOSL allow access to the contents of the stack location indicated by the STKPTR register. This allows users to implement a software stack if necessary. After a CALL, RCALL or interrupt, the software can read the pushed value by reading the TOSU, TOSH and TOSL registers. These values can be placed on a user defined software stack. At return time, the software can replace the TOSU, TOSH and TOSL and do a return.

The user should disable the global interrupt enable bits during this time to prevent inadvertent stack operations.

4.2.2 RETURN STACK POINTER (STKPTR)

The STKPTR register contains the stack pointer value, the STKFUL (stack full) status bit, and the STKUNF (stack underflow) status bits. Register 4-1 shows the STKPTR register. The value of the stack pointer can be 0 through 31. The stack pointer increments when values are pushed onto the stack and decrements when values are popped off the stack. At RESET, the stack pointer value will be 0. The user may read and write the stack pointer value. This feature can be used by a Real Time Operating System for return stack maintenance.

After the PC is pushed onto the stack 31 times (without popping any values off the stack), the STKFUL bit is set. The STKFUL bit can only be cleared in software or by a POR.

The action that takes place when the stack becomes full depends on the state of the STVREN (stack overflow RESET enable) configuration bit. Refer to Section 18 for a description of the device configuration bits. If STVREN is set (default) the 31st push will push the (PC + 2) value onto the stack, set the STKFUL bit, and reset the device. The STKFUL bit will remain set and the stack pointer will be set to 0.

If STVREN is cleared, the STKFUL bit will be set on the 31st push and the stack pointer will increment to 31. The 32nd push will overwrite the 31st push (and so on), while STKPTR remains at 31.

When the stack has been popped enough times to unload the stack, the next pop will return a value of zero to the PC and sets the STKUNF bit, while the stack pointer remains at 0. The STKUNF bit will remain set until cleared in software or a POR occurs.

Note:	Returning a value of zero to the PC on an underflow has the effect of vectoring the program to the RESET vector, where the stack conditions can be verified and appropriate actions can be taken.

REGISTER 4-1: STKPTR - STACK POINTER REGISTER

R/C-0	R/C-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
STKFUL	STKUNF	—	SP4	SP3	SP2	SP1	SP0
bit 7							bit 0

bit 7 **STKFUL**: Stack Full Flag bit
1 = Stack became full or overflowed
0 = Stack has not become full or overflowed

bit 6 **STKUNF**: Stack Underflow Flag bit
1 = Stack underflow occurred
0 = Stack underflow did not occur

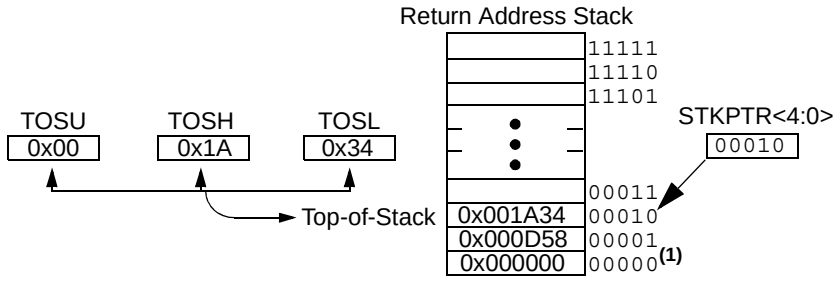
bit 5 **Unimplemented**: Read as '0'

bit 4-0 **SP4:SP0**: Stack Pointer Location bits

Note: Bit 7 and bit 6 can only be cleared in user software or by a POR.

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	C = Clearable bit

FIGURE 4-2: RETURN ADDRESS STACK AND ASSOCIATED REGISTERS



Note 1: No RAM associated with this address; always maintained '0's.

4.4 PCL, PCLATH and PCLATU

The program counter (PC) specifies the address of the instruction to fetch for execution. The PC is 21-bits wide. The low byte is called the PCL register. This register is readable and writable. The high byte is called the PCH register. This register contains the PC<15:8> bits and is not directly readable or writable. Updates to the PCH register may be performed through the PCLATH register. The upper byte is called PCU. This register contains the PC<20:16> bits and is not directly readable or writable. Updates to the PCU register may be performed through the PCLATU register.

The PC addresses bytes in the program memory. To prevent the PC from becoming misaligned with word instructions, the LSb of PCL is fixed to a value of '0'. The PC increments by 2 to address sequential instructions in the program memory.

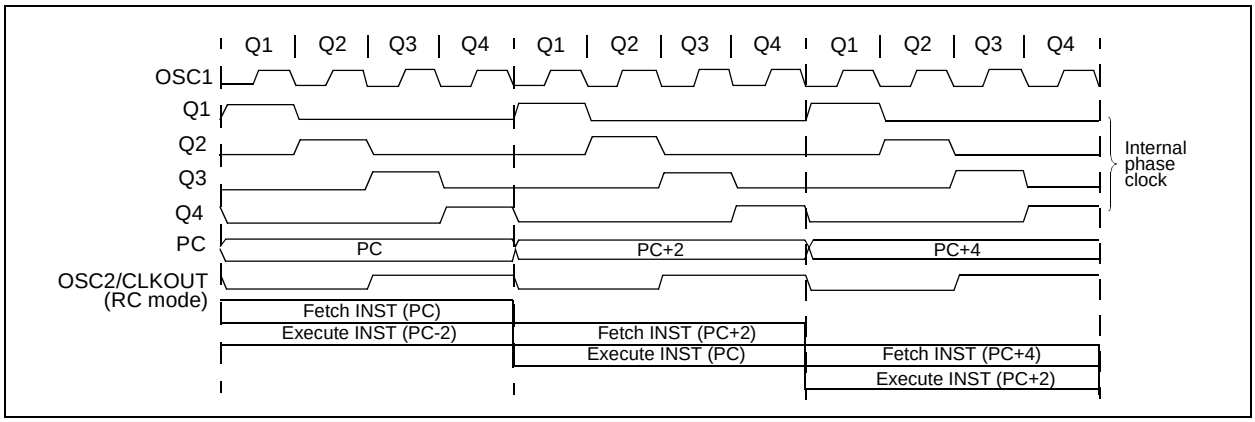
The `CALL, RCALL, GOTO` and program branch instructions write to the program counter directly. For these instructions, the contents of PCLATH and PCLATU are not transferred to the program counter.

The contents of PCLATH and PCLATU will be transferred to the program counter by an operation that writes PCL. Similarly, the upper two bytes of the program counter will be transferred to PCLATH and PCLATU by an operation that reads PCL. This is useful for computed offsets to the PC (See Section 4.8.1).

4.5 Clocking Scheme/Instruction Cycle

The clock input (from OSC1) is internally divided by four to generate four non-overlapping quadrature clocks, namely Q1, Q2, Q3 and Q4. Internally, the program counter (PC) is incremented every Q1, the instruction is fetched from the program memory and latched into the instruction register in Q4. The instruction is decoded and executed during the following Q1 through Q4. The clocks and instruction execution flow are shown in Figure 4-3.

FIGURE 4-3: CLOCK/INSTRUCTION CYCLE



6.0 8 X 8 HARDWARE MULTIPLIER

An 8 x 8 hardware multiplier is included in the ALU of the PIC18CXX8 devices. By making the multiply a hardware operation, it completes in a single instruction cycle. This is an unsigned multiply that gives a 16-bit result. The result is stored into the 16-bit product register pair (PRODH:PRODL). The multiplier does not affect any flags in the STATUS register.

Making the 8 x 8 multiplier execute in a single cycle gives the following advantages:

- Higher computational throughput
- Reduces code size requirements for multiply algorithms

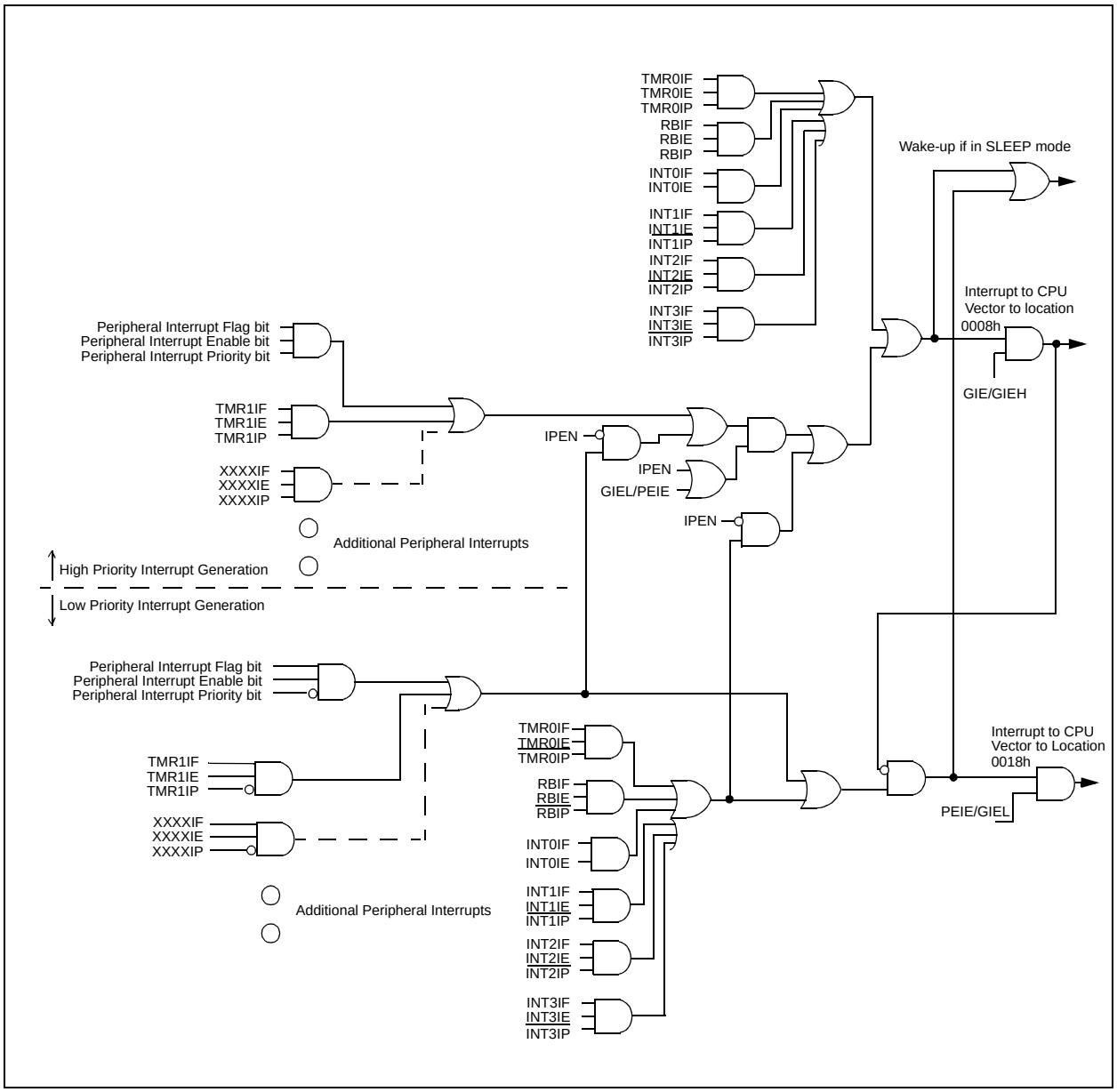
The performance increase allows the device to be used in applications previously reserved for Digital Signal Processors.

Table 6-1 shows a performance comparison between enhanced devices using the single cycle hardware multiply, and performing the same function without the hardware multiply.

TABLE 6-1: PERFORMANCE COMPARISON

Routine	Multiply Method	Program Memory (Words)	Cycles (Max)	Time		
				@ 40 MHz	@ 10 MHz	@ 4 MHz
8 x 8 unsigned	Without hardware multiply	13	69	6.9 µs	27.6 µs	69 µs
	Hardware multiply	1	1	100 ns	400 ns	1 µs
8 x 8 signed	Without hardware multiply	33	91	9.1 µs	36.4 µs	91 µs
	Hardware multiply	6	6	600 ns	2.4 µs	6 µs
16 x 16 unsigned	Without hardware multiply	21	242	24.2 µs	96.8 µs	242 µs
	Hardware multiply	24	24	2.4 µs	9.6 µs	24 µs
16 x 16 signed	Without hardware multiply	52	254	25.4 µs	102.6 µs	254 µs
	Hardware multiply	36	36	3.6 µs	14.4 µs	36 µs

FIGURE 7-1: INTERRUPT LOGIC



REGISTER 7-7: IPR REGISTERS

	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
IPR1	PSPIP	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP
	bit 7							bit 0

	U-0	R/W-1	U-0	U-0	R/W-1	R/W-1	R/W-1	R/W-1
IPR2	—	CMIP	—	—	BCLIP	LVDIP	TMR3IP	CCP2IP
	bit 7							bit 0

	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
IPR3	IVRP	WAKIP	ERRIP	TXB2IP	TXB1IP	TXB0IP	RXB1IP	RXB0IP
	bit 7							bit 0

IPR1

bit 7 **PSPIP:** Parallel Slave Port Read/Write Interrupt Priority bit
1 = High priority
0 = Low priority

bit 6 **ADIP**: A/D Converter Interrupt Priority bit
1 = High priority
0 = Low priority

bit 5 **RCIP**: USART Receive Interrupt Priority bit
1 = High priority
0 = Low priority

bit 4 **TXIP**: USART Transmit Interrupt Priority bit
1 = High priority
0 = Low priority

bit 3 **SSPIP**: Master Synchronous Serial Port Interrupt Priority bit
1 = High priority
0 = Low priority

bit 2 **CCP1IP**: CCP1 Interrupt Priority bit
1 = High priority
0 = Low priority

bit 1 **TMR2IP**: TMR2 to PR2 Match Interrupt Priority bit
1 = High priority
0 = Low priority

bit 0 **TMR1IP**: TMR1 Overflow Interrupt Priority bit
1 = High priority
0 = Low priority

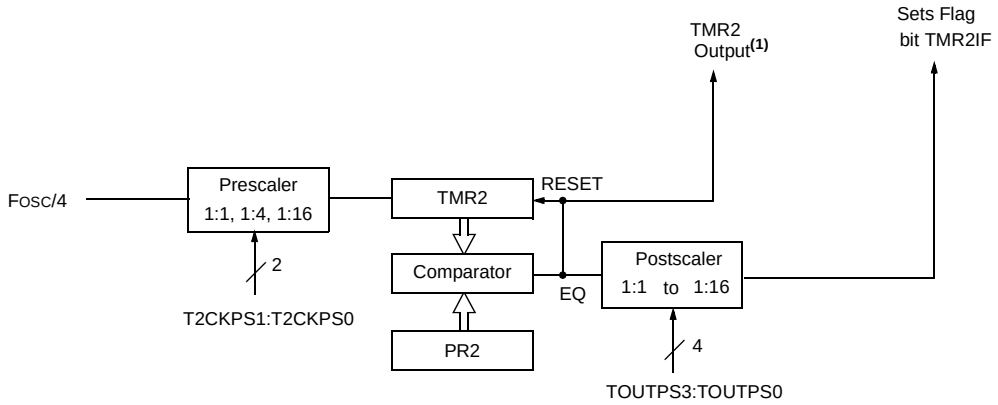
12.2 Timer2 Interrupt

The Timer2 module has an 8-bit period register PR2. Timer2 increments from 00h until it matches PR2 and then resets to 00h on the next increment cycle. PR2 is a readable and writable register. The PR2 register is initialized to FFh upon RESET.

12.3 Output of TMR2

The output of TMR2 (before the postscaler) is a clock input to the Synchronous Serial Port module, which optionally uses it to generate the shift clock.

FIGURE 12-1: TIMER2 BLOCK DIAGRAM



Note 1: TMR2 register output can be software selected by the SSP Module as a baud clock.

TABLE 12-1: REGISTERS ASSOCIATED WITH TIMER2 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	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
PIE1	PSPIE	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
IPR1	PSPIP	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	0000 0000	0000 0000
TMR2	Timer2 module's register								0000 0000	0000 0000
T2CON	—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-000 0000	-000 0000
PR2	Timer2 Period Register								1111 1111	1111 1111

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used by the Timer2 module.

NOTES:

15.4.16.2 Bus Collision During a Repeated START Condition

During a Repeated START condition, a bus collision occurs if:

a) A low level is sampled on SDA when SCL goes from low level to high level.

b) SCL goes low before SDA is asserted low, indicating that another master is attempting to transmit a data '1'.

When the user de-asserts SDA and the pin is allowed to float high, the BRG is loaded with SSPADD<6:0> and counts down to 0. The SCL pin is then de-asserted, and when sampled high, the SDA pin is sampled.

If SDA is low, a bus collision has occurred (i.e, another master is attempting to transmit a data '0', see Figure 15-24). If SDA is sampled high, the BRG is reloaded and begins counting. If SDA goes from high to low before the BRG times out, no bus collision occurs because no two masters can assert SDA at exactly the same time.

If SCL goes from high to low before the BRG times out and SDA has not already been asserted, a bus collision occurs. In this case, another master is attempting to transmit a data '1' during the Repeated START condition (Figure 15-25).

If at the end of the BRG time-out both SCL and SDA are still high, the SDA pin is driven low and the BRG is reloaded and begins counting. At the end of the count, regardless of the status of the SCL pin, the SCL pin is driven low and the Repeated START condition is complete.

FIGURE 15-24: BUS COLLISION DURING A REPEATED START CONDITION (CASE 1)

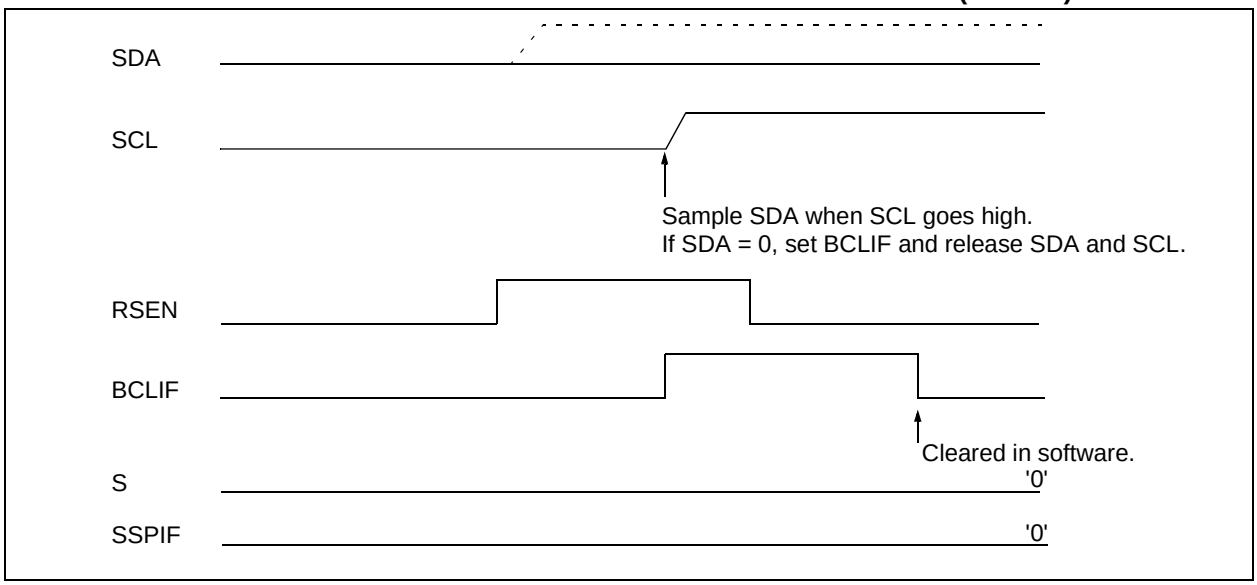
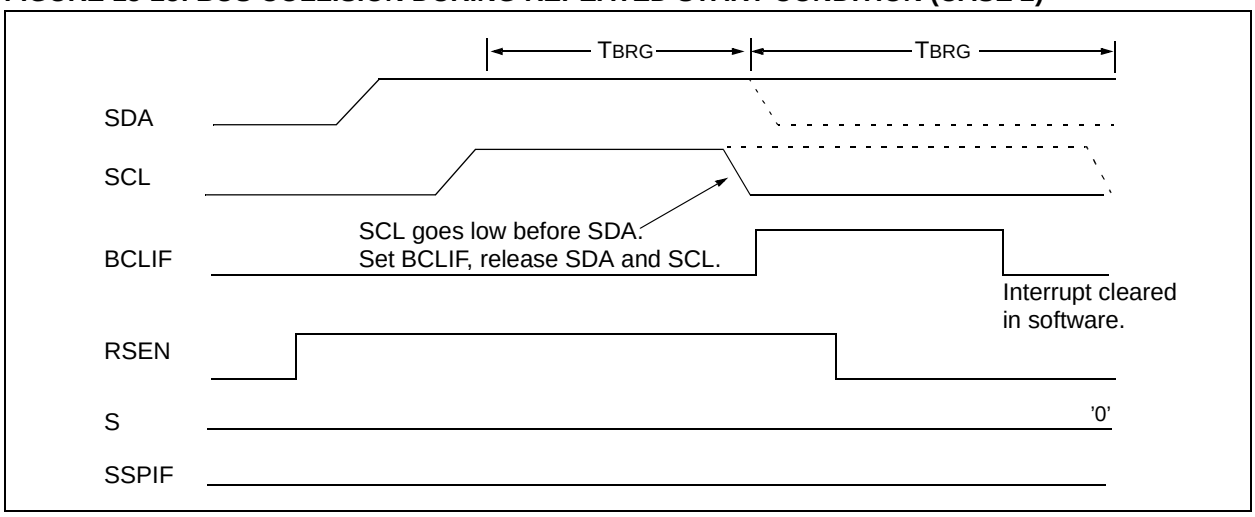


FIGURE 15-25: BUS COLLISION DURING REPEATED START CONDITION (CASE 2)



17.1.2 TRANSMIT/RECEIVE BUFFERS

The PIC18CXX8 has three transmit and two receive buffers, two acceptance masks (one for each receive buffer), and a total of six acceptance filters. Figure 17-1 is a block diagram of these buffers and their connection to the protocol engine.

FIGURE 17-1: CAN BUFFERS AND PROTOCOL ENGINE BLOCK DIAGRAM

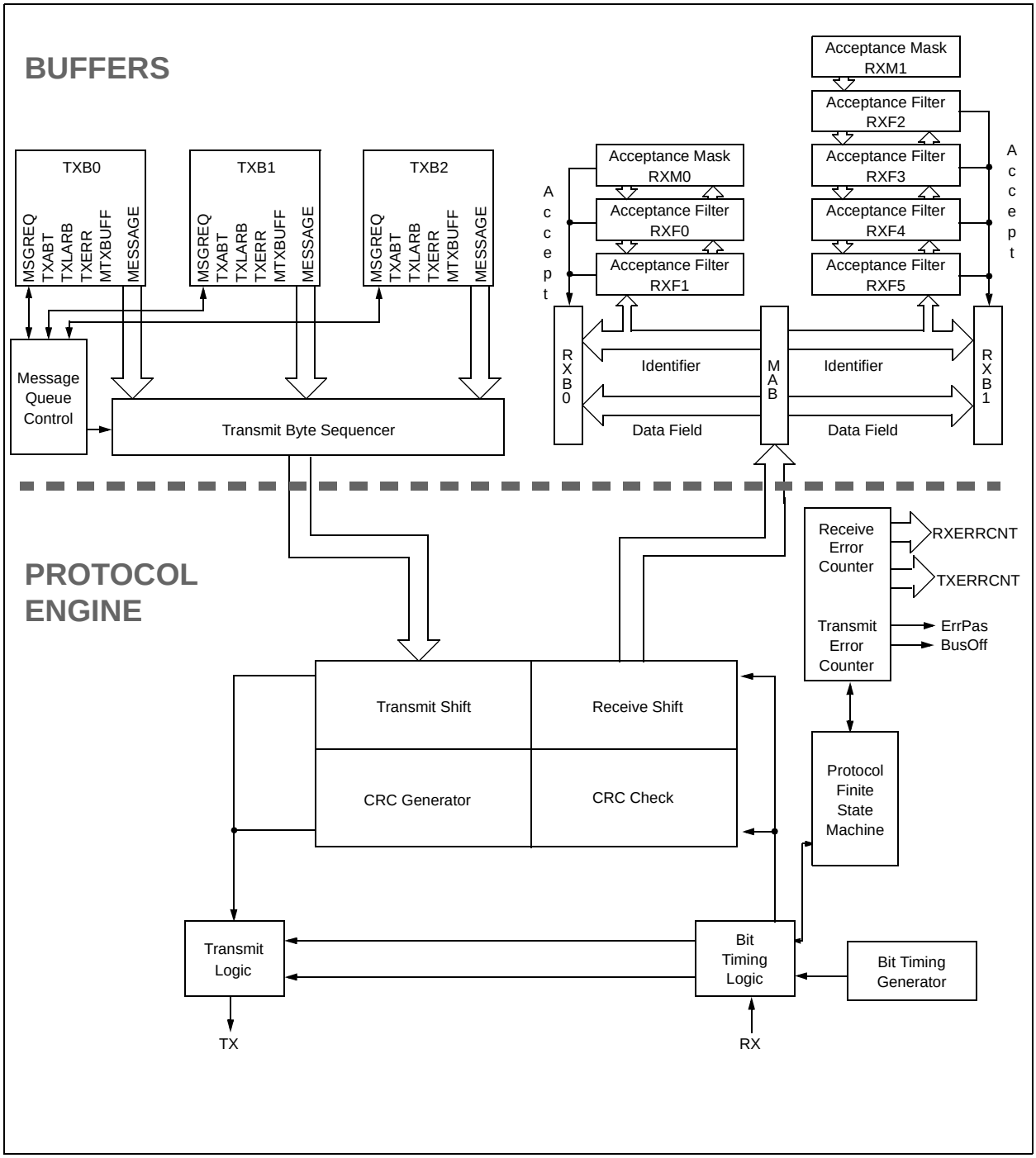
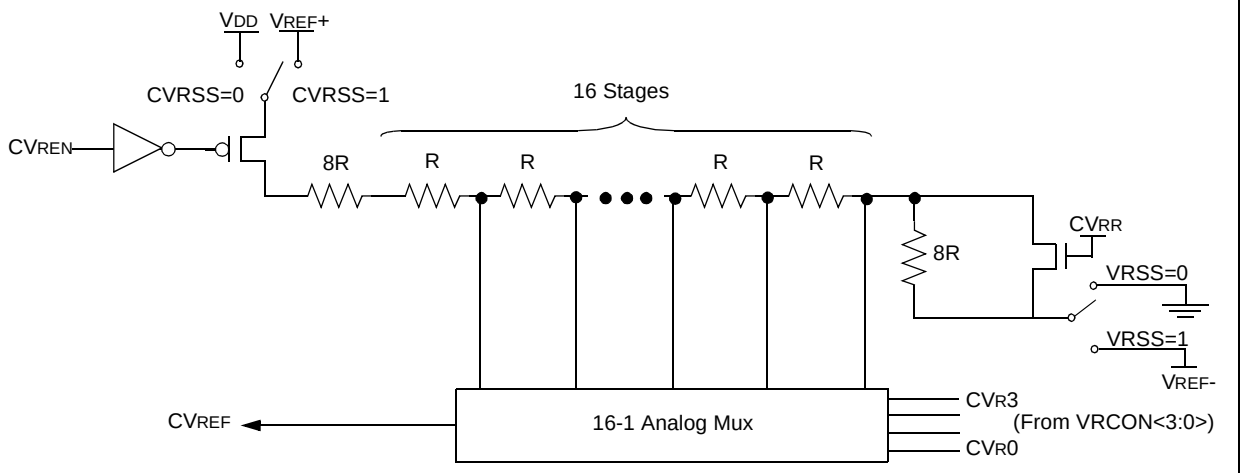


FIGURE 20-1: VOLTAGE REFERENCE BLOCK DIAGRAM



Note: R is defined in Section 25.0.

23.1 Instruction Set

ADDLW **ADD literal to W**

Syntax: [*label*] ADDLW k

Operands: $0 \le k \le 255$

Operation: (WREG) + k → WREG

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

Encoding:

0000	1111	kkkk	kkkk

Description: The contents of WREG are added to the 8-bit literal 'k' and the result is placed in WREG.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: `ADDLW 0x15`

Before Instruction
- WREG = 0x10
- N = ?
- OV = ?
- C = ?
- DC = ?
- Z = ?

After Instruction
- WREG = 0x25
- N = 0
- OV = 0
- C = 0
- DC = 0
- Z = 0

ADDWF **ADD W to f**

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

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

Operation: (WREG) + (f) → dest

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

Encoding:

0010	01da	ffff	ffff

Description: Add WREG to register 'f'. If 'd' is 0, the result is stored 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. If 'a' is 1, the Bank will be selected as per the BSR value.

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example: `ADDWF REG, W`

Before Instruction
- WREG = 0x17
- REG = 0xC2
- N = ?
- OV = ?
- C = ?
- DC = ?
- Z = ?

After Instruction
- WREG = 0xD9
- REG = 0xC2
- N = 1
- OV = 0
- C = 0
- DC = 0
- Z = 0

BTFSC **Bit Test File, Skip if Clear**

Syntax: [*label*] BTFSC f, b [,a]

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

Operation: skip if (f<b>) = 0

Status Affected: None

Encoding:

1011	bbba	ffff	ffff

Description: If bit 'b' in register 'f' is 0, then the next instruction is skipped.
If bit 'b' is 0, then the next instruction fetched during the current instruction execution is discarded, and a NOP is executed instead, making this a two-cycle instruction. If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' is 1, the Bank will be selected as per the BSR value.

Words: 1

Cycles: 1(2)
Note: 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	No operation

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example:
```
HERE    BTFSC   FLAG, 1, ACCESS
FALSE   :
TRUE    :
```

Before Instruction
PC = address (HERE)

After Instruction
If FLAG<1> = 0;
PC = address (TRUE)
If FLAG<1> = 1;
PC = address (FALSE)

BTFSS **Bit Test File, Skip if Set**

Syntax: [*label*] BTFSS f, b [,a]

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

Operation: skip if (f<b>) = 1

Status Affected: None

Encoding:

1010	bbba	ffff	ffff

Description: If bit 'b' in register 'f' is 1 then the next instruction is skipped.
If bit 'b' is 1, then the next instruction fetched during the current instruction execution, is discarded and an NOP is executed instead, making this a two-cycle instruction. If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' is 1, the Bank will be selected as per the BSR value.

Words: 1

Cycles: 1(2)
Note: 3 cycles if skip and followed by a 2-word instruction.

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	No operation

If skip:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation

If skip and followed by 2-word instruction:

Q1	Q2	Q3	Q4
No operation	No operation	No operation	No operation
No operation	No operation	No operation	No operation

Example:
```
HERE    BTFSS   FLAG, 1, ACCESS
FALSE   :
TRUE    :
```

Before Instruction
PC = address (HERE)

After Instruction
If FLAG<1> = 0;
PC = address (FALSE)
If FLAG<1> = 1;
PC = address (TRUE)

BZ **Branch if Zero**

Syntax: [*label*] BZ n

Operands: -128 ≤ n ≤ 127

Operation: if Zero bit is '1'
(PC) + 2 + 2n → PC

Status Affected: None

Encoding:

1110	0000	nnnn	nnnn

Description: If the Zero 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 BZ Jump`

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

CALL **Subroutine Call**

Syntax: [*label*] CALL k [,s]

Operands: 0 ≤ k ≤ 1048575
s ∈ [0,1]

Operation: (PC) + 4 → TOS,
k → PC<20:1>,
if s = 1
(WREG) → WS,
(STATUS) → STATUSS,
(BSR) → BSRS

Status Affected: None

Encoding:

1st word (k<7:0>)	1110	110s	$k_7$kkk	kkkk$_0$
2nd word(k<19:8>)	1111	k_{19}kkk	kkkk	kkkk$_8$

Description: Subroutine call of entire 2M byte memory range. First, return address (PC+ 4) is pushed onto the return stack. If 's' = 1, the WREG, STATUS and BSR registers are also pushed into their respective shadow registers, WS, STATUSS and BSRS. If 's' = 0, no update occurs (default). Then the 20-bit value 'k' is loaded into PC<20:1>. `CALL` is a two-cycle instruction.

Words: 2

Cycles: 2

Q Cycle Activity:

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

Example: `HERE CALL THERE, FAST`

Before Instruction
PC = Address `(HERE)`
After Instruction
PC = Address `(THERE)`
TOS = Address `(HERE + 4)`
WS = WREG
BSRS = BSR
STATUSS = STATUS

25.2 DC Characteristics: PIC18CXX8 (Industrial, Extended) and PIC18LCXX8 (Industrial)

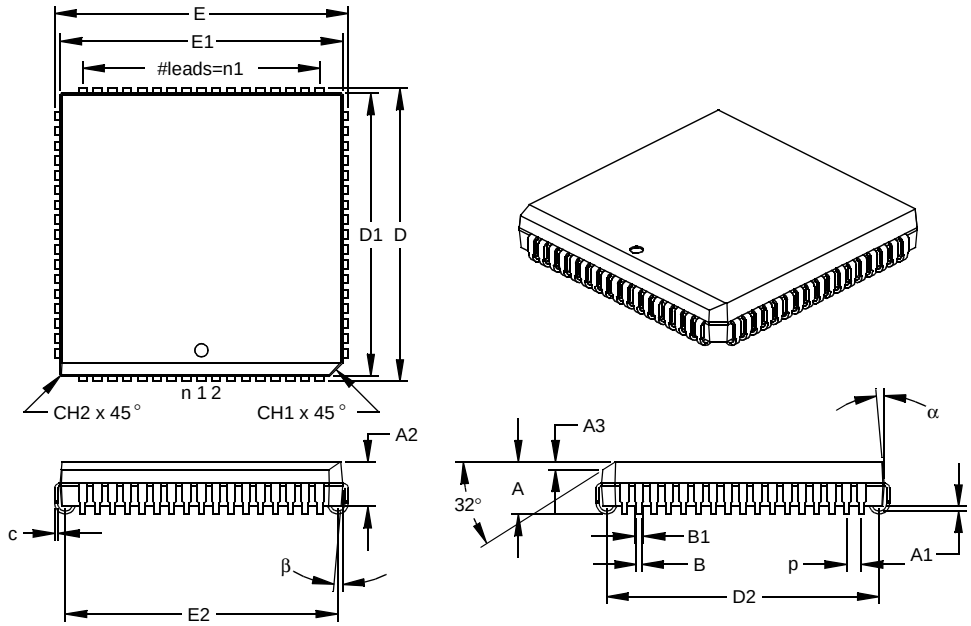
DC CHARACTERISTICS			Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C for industrial -40°C ≤ TA ≤ +125°C for extended			
Param No.	**Symbol**	**Characteristic/ Device**	**Min**	**Max**	**Units**	**Conditions**
	VIL	**Input Low Voltage**				
		I/O ports:				
D030		with TTL buffer	VSS	0.15VDD	V	VDD < 4.5V
D030A			—	0.8	V	4.5V ≤ VDD ≤ 5.5V
D031		with Schmitt Trigger buffer	VSS	0.2VDD	V	
		RC3 and RC4	VSS	0.3VDD	V	
D032		$\overline{MCLR}$	VSS	0.2VDD	V	
D032A		OSC1 (in XT, HS and LP modes) and T1OSI	VSS	0.3VDD	V	
D033		OSC1(in RC mode)(1)	VSS	0.2VDD	V	
	VIH	**Input High Voltage**				
		I/O ports:				
D040		with TTL buffer	0.25VDD + 0.8V	VDD	V	VDD < 4.5V
D040A			2.0	VDD	V	4.5V ≤ VDD ≤ 5.5V
D041		with Schmitt Trigger buffer	0.8VDD	VDD	V	
		RC3 and RC4	0.7VDD	VDD	V	
D042		$\overline{MCLR}$	0.8VDD	VDD	V	
D042A		OSC1 (in XT, HS and LP modes) and T1OSI	0.7VDD	VDD	V	
D043		OSC1 (RC mode)(1)	0.9VDD	VDD	V	
	VHYS	**Hysteresis of Schmitt Trigger Inputs**				
D050			TBD	TBD	V	
	IIL	**Input Leakage Current(2,3)**				
D060		I/O ports	—	±1	µA	VSS ≤ VPIN ≤ VDD, Pin at hi-impedance
D061		$\overline{MCLR}$	—	±5	µA	Vss ≤ VPIN ≤ VDD
D063		OSC1	—	±5	µA	Vss ≤ VPIN ≤ VDD
	IPU	**Weak Pull-up Current**				
D070	IPURB	PORTB weak pull-up current	50	400	µA	VDD = 5V, VPIN = VSS

Note 1: In RC oscillator configuration, the OSC1/CLKI pin is a Schmitt Trigger input. It is not recommended that the PICmicro device be driven with an external clock while in RC mode.

2: The leakage current on the $\overline{MCLR}$ pin is strongly dependent on the applied voltage level. The specified levels represent normal operating conditions. Higher leakage current may be measured at different input voltages.

3: Negative current is defined as current sourced by the pin.

68-Lead Plastic Leaded Chip Carrier (L) – Square (PLCC)



Units		INCHES*			MILLIMETERS		
Dimension Limits		MIN	NOM	MAX	MIN	NOM	MAX
Number of Pins	n		68			68	
Pitch	p		.050			1.27	
Pins per Side	n1		17			17	
Overall Height	A	.165	.173	.180	4.19	4.39	4.57
Molded Package Thickness	A2	.145	.153	.160	3.68	3.87	4.06
Standoff §	A1	.020	.028	.035	0.51	0.71	0.89
Side 1 Chamfer Height	A3	.024	.029	.034	0.61	0.74	0.86
Corner Chamfer 1	CH1	.040	.045	.050	1.02	1.14	1.27
Corner Chamfer (others)	CH2	.000	.005	.010	0.00	0.13	0.25
Overall Width	E	.985	.990	.995	25.02	25.15	25.27
Overall Length	D	.985	.990	.995	25.02	25.15	25.27
Molded Package Width	E1	.950	.954	.958	24.13	24.23	24.33
Molded Package Length	D1	.950	.954	.958	24.13	24.23	24.33
Footprint Width	E2	.890	.920	.930	22.61	23.37	23.62
Footprint Length	D2	.890	.920	.930	22.61	23.37	23.62
Lead Thickness	c	.008	.011	.013	0.20	0.27	0.33
Upper Lead Width	B1	.026	.029	.032	0.66	0.74	0.81
Lower Lead Width	B	.013	.020	.021	0.33	0.51	0.53
Mold Draft Angle Top	α	0	5	10	0	5	10
Mold Draft Angle Bottom	β	0	5	10	0	5	10

* Controlling Parameter
§ Significant Characteristic
Notes:
Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed .010” (0.254mm) per side.
JEDEC Equivalent: MO-047
Drawing No. C04-049

PIC18CXX8 PRODUCT IDENTIFICATION SYSTEM

To order or obtain information, e.g., on pricing or delivery refer to the factory or the listed sales office

PART NO.	X	/XX	XXX
Device	Temperature Range	Package	Pattern

Device	PIC18CXX8[1], PIC18CXX8T[2]; VDD range 4.2V to 5.5V PIC18LCXX5[1], PIC18LCXX8T[2]; VDD range 2.5V to 5.5V
Temperature Range	I = -40°C to +85°C (Industrial) E = -40°C to +125°C (Extended)
Package	CL = Windowed JCERPACK PT = TQFP (Thin Quad Flatpack) L = PLCC
Pattern	QTP, SQTP, Code or Special Requirements (blank otherwise)

Examples:

a) PIC18LC658 - I/L 301 = Industrial temp., PLCC package, Extended VDD limits, QTP pattern #301.

b) PIC18LC858 - I/PT = Industrial temp., TQFP package, Extended VDD limits.

c) PIC18C658 - E/L = Extended temp., PLCC package, normal VDD limits.

Note 1: C = Standard Voltage Range
LC = Wide Voltage Range

2: T = in tape and reel PLCC, and TQFP packages only.

3: CL devices are UV erasable and can be programmed to any device configuration. CL devices meet the electrical requirement of each oscillator type (including LC devices).

* JW Devices are UV erasable and can be programmed to any device configuration. JW Devices meet the electrical requirement of each oscillator type.

Sales and Support

Data Sheets
Products supported by a preliminary Data Sheet may have an errata sheet describing minor operational differences and recommended workarounds. To determine if an errata sheet exists for a particular device, please contact one of the following:

1. Your local Microchip sales office
2. The Microchip Corporate Literature Center U.S. FAX: (480) 792-7277
3. The Microchip Worldwide Site (www.microchip.com)

Please specify which device, revision of silicon and Data Sheet (include Literature #) you are using.

New Customer Notification System
Register on our web site (www.microchip.com/cn) to receive the most current information on our products.