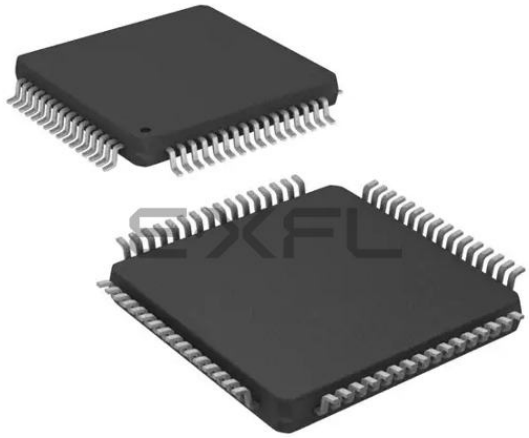


Welcome to E-XFL.COM

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	64MHz
Connectivity	I²C, IrDA, LINbus, SPI, UART/USART, USB
Peripherals	Brown-out Detect/Reset, HLVD, LCD, POR, PWM, WDT
Number of I/O	52
Program Memory Size	32KB (16K x 16)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	4K x 8
Voltage - Supply (Vcc/Vdd)	2V ~ 3.6V
Data Converters	A/D 16x10b/12b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	64-TQFP
Supplier Device Package	64-TQFP (10x10)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18f65j94-i-pt

TABLE 1-3: DEVICE FEATURES FOR THE 100-PIN DEVICES

Features	PIC18F95J94	PIC18F96J94	PIC18F97J94
Operating Frequency	DC – 64 MHz		
Program Memory (Bytes)	32 K	64K	128K
	(Up to 2 Mbytes with Extended Memory)		
Program Memory (Instructions)	16,384	32,768	65,536
Data Memory (Bytes)	4K	4K	4K
Interrupt Sources	42	48	
I/O Ports	Ports A, B, C, D, E, F, G, H, J, K, L		
Parallel Communications	Parallel Slave Port (PSP)		
Timers	8		
Comparators	3		
LCD	480 pixels		
CTMU	Yes		
RTCC	Yes		
Enhanced Capture/Compare/PWM Modules	3 ECCPs and 7 CCPs		
Serial Communications	Two MSSPs, Four Enhanced USARTs (EUSART) and USB		
12-Bit Analog-to-Digital Module	24 Input Channels		
Resets (and Delays)	POR, BOR, CM RESET Instruction, Stack Full, Stack Underflow, $\overline{\text{MCLR}}$, WDT (PWRT, OST)		
Instruction Set	75 Instructions, 83 with Extended Instruction Set Enabled		
Packages	100-Pin TQFP		

REGISTER 5-1: RCON: RESET CONTROL REGISTER

R/W-0	U-0	R/W-1	R/W-1	R-1	R-1	R/W-0(1)	R/W-0
IPEN	—	$\overline{CM}$	$\overline{RI}$	$\overline{TO}$	$\overline{PD}$	$\overline{POR}$	$\overline{BOR}$
bit 7							bit 0

Legend:	HC = Hardware Clearable bit		
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 **IPEN:** Interrupt Priority Enable Register bit
1 = Prioritized interrupts are enabled
0 = Prioritized interrupts are disabled

bit 6 **Unimplemented**: Read as '0'

bit 5 **$\overline{CM}$:** Configuration Mismatch Flag bit
1 = A Configuration Mismatch Reset has not occurred
0 = A Configuration Mismatch Reset occurred; must be set in software once the Reset occurs

bit 4 **$\overline{RI}$:** RESET Instruction Flag bit
1 = The RESET instruction was not executed (set by firmware only)
0 = The RESET instruction was executed, causing a device Reset (must be set in software after a Brown-out Reset occurs)

bit 3 **$\overline{TO}$:** Watchdog Time-out Flag bit
1 = Set by power-up, CLRWDT instruction or SLEEP instruction
0 = A WDT time-out occurred

bit 2 **$\overline{PD}$:** Power-down Detection Flag bit
1 = Set by power-up or by the CLRWDT instruction
0 = Set by execution of the SLEEP instruction

bit 1 **$\overline{POR}$:** Power-on Reset Status bit(1)
1 = A Power-on Reset has not occurred (set by firmware only)
0 = A Power-on Reset occurred (must be set in software after a Power-on Reset occurs)

bit 0 **$\overline{BOR}$:** Brown-out Reset Status bit
1 = A Brown-out Reset has not occurred (set by firmware only)
0 = A Brown-out Reset occurred (must be set in software after a Brown-out Reset occurs)

Note 1: Brown-out Reset is said to have occurred when $\overline{BOR}$ is '0' and $\overline{POR}$ is '1' (assuming that $\overline{POR}$ was set to '1' by software immediately after a Power-on Reset).

TABLE 5-3: INITIALIZATION CONDITIONS FOR ALL REGISTERS (CONTINUED)

Register	Applicable Devices			Power-on Reset, Brown-out Reset	$\overline{\text{MCLR}}$ Resets, WDT Reset, RESET Instruction, Stack Resets	Wake-up via WDT or Interrupt
LCDREFL	64-pin	80-pin	100-pin	0000 -000	uuuu -uuu	uuuu -uuu
LCDSE7	64-pin	80-pin	100-pin	0000 0000	uuuu uuuu	uuuu uuuu
LCDSE6	64-pin	80-pin	100-pin	0000 0000	uuuu uuuu	uuuu uuuu
LCDSE5	64-pin	80-pin	100-pin	0000 0000	uuuu uuuu	uuuu uuuu
LCDSE4	64-pin	80-pin	100-pin	0000 0000	uuuu uuuu	uuuu uuuu
LCDSE3	64-pin	80-pin	100-pin	0000 0000	uuuu uuuu	uuuu uuuu
LCDSE2	64-pin	80-pin	100-pin	0000 0000	uuuu uuuu	uuuu uuuu
LCDSE1	64-pin	80-pin	100-pin	0000 0000	uuuu uuuu	uuuu uuuu
LCDSE0	64-pin	80-pin	100-pin	0000 0000	uuuu uuuu	uuuu uuuu
LCDDATA63	64-pin	80-pin	100-pin	0000 0000	uuuu uuuu	uuuu uuuu
LCDDATA62	64-pin	80-pin	100-pin	0000 0000	uuuu uuuu	uuuu uuuu
LCDDATA61	64-pin	80-pin	100-pin	0000 0000	uuuu uuuu	uuuu uuuu
LCDDATA60	64-pin	80-pin	100-pin	0000 0000	uuuu uuuu	uuuu uuuu
LCDDATA59	64-pin	80-pin	100-pin	0000 0000	uuuu uuuu	uuuu uuuu
LCDDATA58	64-pin	80-pin	100-pin	0000 0000	uuuu uuuu	uuuu uuuu
LCDDATA57	64-pin	80-pin	100-pin	0000 0000	uuuu uuuu	uuuu uuuu
LCDDATA56	64-pin	80-pin	100-pin	0000 0000	uuuu uuuu	uuuu uuuu
LCDDATA55	64-pin	80-pin	100-pin	0000 0000	uuuu uuuu	uuuu uuuu
LCDDATA54	64-pin	80-pin	100-pin	0000 0000	uuuu uuuu	uuuu uuuu
LCDDATA53	64-pin	80-pin	100-pin	0000 0000	uuuu uuuu	uuuu uuuu
LCDDATA52	64-pin	80-pin	100-pin	0000 0000	uuuu uuuu	uuuu uuuu
LCDDATA51	64-pin	80-pin	100-pin	0000 0000	uuuu uuuu	uuuu uuuu
LCDDATA50	64-pin	80-pin	100-pin	0000 0000	uuuu uuuu	uuuu uuuu
LCDDATA49	64-pin	80-pin	100-pin	0000 0000	uuuu uuuu	uuuu uuuu
LCDDATA48	64-pin	80-pin	100-pin	0000 0000	uuuu uuuu	uuuu uuuu
LCDDATA47	64-pin	80-pin	100-pin	0000 0000	uuuu uuuu	uuuu uuuu
LCDDATA46	64-pin	80-pin	100-pin	0000 0000	uuuu uuuu	uuuu uuuu
LCDDATA45	64-pin	80-pin	100-pin	0000 0000	uuuu uuuu	uuuu uuuu
LCDDATA44	64-pin	80-pin	100-pin	0000 0000	uuuu uuuu	uuuu uuuu
LCDDATA43	64-pin	80-pin	100-pin	0000 0000	uuuu uuuu	uuuu uuuu
LCDDATA42	64-pin	80-pin	100-pin	0000 0000	uuuu uuuu	uuuu uuuu
LCDDATA41	64-pin	80-pin	100-pin	0000 0000	uuuu uuuu	uuuu uuuu
LCDDATA40	64-pin	80-pin	100-pin	0000 0000	uuuu uuuu	uuuu uuuu

Legend: u = unchanged; x = unknown; - = unimplemented bit, read as '0'; q = value depends on condition. Shaded cells indicate that conditions do not apply for the designated device.

Note 1: 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.

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

4: See Table 5-2 for Reset value for specific condition.

5: Bits 7,6 are unimplemented on 64 and 80-pin devices.

6: If the V_{BAT} is always powered, the DSGPx register values will remain unchanged after the first POR.

7.4 Erasing Flash Program Memory

The minimum erase block is 256 words or 512 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 microcontroller itself, a block of 512 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 A FLASH PROGRAM MEMORY ROW

```
                    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 Row 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
```

REGISTER 10-13: PIE4: PERIPHERAL INTERRUPT ENABLE REGISTER 4

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
CCP10IE	CCP9IE	CCP8IE	CCP7IE	CCP6IE	CCP5IE	CCP4IE	ECCP3IE
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 **CCP10IE:** CCP10 Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 6 **CCP9IE:** CCP9 Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 5 **CCP8IE:** CCP8 Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 4 **CCP7IE:** CCP7 Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 3 **CCP6IE:** CCP6 Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 2 **CCP5IE:** CCP5 Interrupt Flag bit
1 = Enabled
0 = Disabled

bit 1 **CCP4IE:** CCP4 Interrupt Flag bit
1 = Enabled
0 = Disabled

bit 0 **ECCP3IE:** ECCP3 Interrupt Flag bit
1 = Enabled
0 = Disabled

REGISTER 11-1: PADCFG1: PAD CONFIGURATION REGISTER 1(1)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
RDPU	REPU	RFPU	RGPU	RHPU	RJPU	RKPU	RLPU
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 **RDPU:** PORTD Pull-up Enable bit
1 = PORTD pull-ups are enabled for any input pad
0 = All PORTD pull-ups are disabled

bit 6 **REPU:** PORTE Pull-up Enable bit
1 = PORTE pull-ups are enabled for any input pad
0 = All PORTE pull-ups are disabled

bit 5 **RFPU:** PORTF Pull-up Enable bit
1 = PORTF pull-ups are enabled for any input pad
0 = All PORTF pull-ups are disabled

bit 4 **RGPU:** PORTG Pull-up Enable bit
1 = PORTG pull-ups are enabled for any input pad
0 = All PORTG pull-ups are disabled

bit 3 **RHPU:** PORTH Pull-up Enable bit
1 = PORTH pull-ups are enabled for any input pad
0 = All PORTH pull-ups are disabled

bit 2 **RJPU:** PORTJ Pull-up Enable bit
1 = PORTJ pull-ups are enabled for any input pad
0 = All PORTJ pull-ups are disabled

bit 1 **RKPU:** PORTK Pull-up Enable bit
1 = PORTK pull-ups are enabled for any input pad
0 = All PORTK pull-ups are disabled

bit 0 **RLPU:** PORTL Pull-up Enable bit
1 = PORTL pull-ups are enabled for any input pad
0 = All PORTL pull-ups are disabled

Note 1: If a particular PORT is not available on a package, the corresponding RnPU register bit will be unimplemented and read back as '0'.

11.8 PORTG, LATG and TRISG Registers

PORTG width varies depending on pin count. For 64- and 80-pin devices, PORTG is a 6-bit wide, bidirectional port. For 100-pin devices, PORTG is an 8-bit wide bidirectional port. The corresponding Data Direction and Output Latch registers are TRISG and LATG.

PORTG is multiplexed with the EUSART, and CCP, ECCP, Analog, Comparator, RTCC and Timer input functions (Table 11-7). When operating as I/O, all PORTG pins have Schmitt Trigger input buffers. The open-drain functionality for the CCPx and EUSARTx can be configured using ODCONx.

When enabling peripheral functions, care should be taken in defining TRIS bits for each PORTG pin. Some peripherals override the TRIS bit to make a pin an output, while other peripherals override the TRIS bit to make a pin an input. The user should refer to the corresponding peripheral section for the correct TRIS bit settings. The pin override value is not loaded into the TRIS register. This allows read-modify-write of the TRIS register without concern due to peripheral overrides.

EXAMPLE 11-7: INITIALIZING PORTG

```
CLRF    PORTG       ;  Initialize PORTG by
                    ; clearing output
                    ; data latches
BCF     CM1CON, CON ; disable
                    ; comparator 1
CLRF    LATG        ; Alternate method
                    ; to clear output
                    ; data latches
BANKSEL ANCON2      ; Select bank with ACON2 register
MOVLW   0F0h        ; make AN16 to AN19
                    ; digital
MOVWF   ANCON2
BANKSEL TRISG       ; Select bank with TRISG register
MOVLW   04h         ; Value used to
                    ; initialize data
                    ; direction
MOVWF   TRISG       ; Set RG1:RG0 as
                    ; outputs
                    ; RG2 as input
                    ; RG4:RG3 as inputs
```

TABLE 11-7: PORTG FUNCTIONS

Pin Name	Function	TRIS Setting	I/O	I/O Type	Description
RG0/RP46/AN8/ SEG28/COM4	RG0	0	O	DIG	LATG<0> data output; not affected by analog input.
		1	I	ST	PORTG<0> data input; disabled when analog input is enabled.
	RP46	x	x	DIG	Reconfigurable Pin 46 for PPS-Lite; TRIS must be set to match input/ output of module.
	AN8	1	I	ANA	A/D Input Channel 8. Default input configuration on POR; does not affect digital output.
	SEG28	0	O	ANA	LCD Segment 28 output; disables all other pin functions.
	COM4	x	O	ANA	LCD Common 4 output; disables all other outputs.
RG1/RP39/ AN19/SEG29/ COM5	RG1	0	O	DIG	LATG<1> data output; not affected by analog input.
		1	I	ST	PORTG<1> data input; disabled when analog input is enabled.
	RP39	x	x	DIG	Reconfigurable Pin 39 for PPS-Lite; TRIS must be set to match input/ output of module.
	AN19	1	I	ANA	A/D Input Channel 19. Default input configuration on POR; does not affect digital output.
	SEG29	0	O	ANA	LCD Segment 29 output; disables all other pin functions.
	COM5	x	O	ANA	LCD Common 5 output; disables all other outputs.
RG2/RP42/ C3INA/AN18/ SEG30/COM6	RG2	0	O	DIG	LATG<2> data output; not affected by analog input.
		1	I	ST	PORTG<2> data input; disabled when analog input is enabled.
	RP42	x	x	DIG	Reconfigurable Pin 42 for PPS-Lite; TRIS must be set to match input/ output of module.
	C3INA	1	I	ANA	Comparator 3 Input A.
	AN18	1	I	ANA	A/D Input Channel 18. Default input configuration on POR; does not affect digital output.
	SEG30	0	O	ANA	LCD Segment 30 output; disables all other pin functions.
	COM6	x	O	ANA	LCD Common 6 output; disables all other outputs.

Legend: O = Output, I = Input, ANA = Analog Signal, DIG = Digital Output, ST = Schmitt Trigger Buffer Input, x = Don't care (TRIS bit does not affect port direction or is overridden for this option).

REGISTER 13-3: LCDPS: LCD PHASE REGISTER

R/W-0	R/W-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0
WFT	BIASMD	LCDA	WA	LP3	LP2	LP1	LP0
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 **WFT**: Waveform Type Select bit
1 = Type-B waveform (phase changes on each frame boundary)
0 = Type-A waveform (phase changes within each common type)

bit 6 **BIASMD**: Bias Mode Select bit
When LMUX<2:0> = 000 or 011 through 111:
0 = Static Bias mode (LMUX<2:0> = 000) / 1/3 Bias mode (LMUX<2:0> = 011 through 111) (do not set this bit to '1')
When LMUX<2:0> = 001 or 010:
1 = 1/2 Bias mode
0 = 1/3 Bias mode

bit 5 **LCDA**: LCD Active Status bit
1 = LCD driver module is active
0 = LCD driver module is inactive

bit 4 **WA**: LCD Write Allow Status bit
1 = Writes into the LCDDATAx registers is allowed
0 = Writes into the LCDDATAx registers is not allowed

bit 3-0 **LP<3:0>**: LCD Prescaler Select bits
1111 = 1:16
1110 = 1:15
1101 = 1:14
1100 = 1:13
1011 = 1:12
1010 = 1:11
1001 = 1:10
1000 = 1:9
0111 = 1:8
0110 = 1:7
0101 = 1:6
0100 = 1:5
0011 = 1:4
0010 = 1:3
0001 = 1:2
0000 = 1:1

REGISTER 17-18: ALRMMIN: ALARM MINUTES VALUE REGISTER (ALRMVALH when ALRMPTR<1:0> = 00)

U-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
—	MINTEN2	MINTEN1	MINTEN0	MINONE3	MINONE2	MINONE1	MINONE0
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 **Unimplemented:** Read as '0'

bit 6-4 **MINTEN<2:0>:** Binary Coded Decimal Value of Minute's Tens Digit bits
Contains a value from 0 to 5.

bit 3-0 **MINONE<3:0>:** Binary Coded Decimal Value of Minute's Ones Digit bits
Contains a value from 0 to 9.

REGISTER 17-19: ALRMSEC: ALARM SECONDS VALUE REGISTER (ALRMVALL when ALRMPTR<1:0> = 00)

U-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
—	SECTEN2	SECTEN1	SECTEN0	SECONE3	SECONE2	SECONE1	SECONE0
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 **Unimplemented:** Read as '0'

bit 6-4 **SECTEN<2:0>:** Binary Coded Decimal Value of Second's Tens Digit bits
Contains a value from 0 to 5.

bit 3-0 **SECONE<3:0>:** Binary Coded Decimal Value of Second's Ones Digit bits
Contains a value from 0 to 9.

17.2.4 LEAP YEAR

Since the year range on the RTCC module is 2000 to 2099, the leap year calculation is determined by any year divisible by four in the above range. Only February is affected in a leap year.

February will have 29 days in a leap year and 28 days in any other year.

17.2.5 GENERAL FUNCTIONALITY

All Timer registers containing a time value of seconds or greater are writable. The user configures the time by writing the required year, month, day, hour, minutes and seconds to the Timer registers, via register pointers. (See **Section 17.2.8 "Register Mapping"**.)

The timer uses the newly written values and proceeds with the count from the required starting point.

The RTCC is enabled by setting the RTCEN bit (RTCCON1<7>). If enabled, while adjusting these registers, the timer still continues to increment. However, any time the MINSEC register is written to, both of the timer prescalers are reset to '0'. This allows fraction of a second synchronization.

The Timer registers are updated in the same cycle as the WRITE instruction's execution by the CPU. The user must ensure that when RTCEN = 1, the updated registers will not be incremented at the same time. This can be accomplished in several ways:

- By checking the RTCSYNC bit (RTCCON1<4>)
- By checking the preceding digits from which a carry can occur
- By updating the registers immediately following the seconds pulse (or an alarm interrupt)

The user has visibility to the half-second field of the counter. This value is read-only and can be reset only by writing to the lower half of the SECONDS register.

17.2.6 SAFETY WINDOW FOR REGISTER READS AND WRITES

The RTCSYNC bit indicates a time window during which the RTCC clock domain registers can be safely read and written without concern about a rollover. When RTCSYNC = 0, the registers can be safely accessed by the CPU.

Whether RTCSYNC = 1 or 0, the user should employ a firmware solution to ensure that the data read did not fall on a rollover boundary, resulting in an invalid or partial read. This firmware solution would consist of reading each register twice and then comparing the two values. If the two values matched, then a rollover did not occur.

17.2.7 WRITE LOCK

In order to perform a write to any of the RTCC Timer registers, the RTCWREN bit (RTCCON1<5>) must be set.

To avoid accidental writes to the RTCC Timer register, it is recommended that the RTCWREN bit (RTCCON1<5>) be kept clear when not writing to the register. For the RTCWREN bit to be set, there is only one instruction cycle time window allowed between the 55h/AA sequence and the setting of RTCWREN. For that reason, it is recommended that users follow the code example in Example 17-1.

EXAMPLE 17-1: SETTING THE RTCWREN BIT

```
movlw   0x55
movwf   EECON2
movlw   0xAA
movwf   EECON2
bsf     RTCCON1,RTCWREN
```

17.2.8 REGISTER MAPPING

To limit the register interface, the RTCC Timer and Alarm Timer registers are accessed through corresponding register pointers. The RTCC Value register window (RTCVALH and RTCVALL) uses the RTCPTRx bits (RTCCON1<1:0>) to select the required Timer register pair.

By reading or writing to the RTCVALH register, the RTCC Pointer value (RTCPTR<1:0>) decrements by '1' until it reaches '00'. When '00' is reached, the MINUTES and SECONDS value is accessible through RTCVALH and RTCVALL until the pointer value is manually changed.

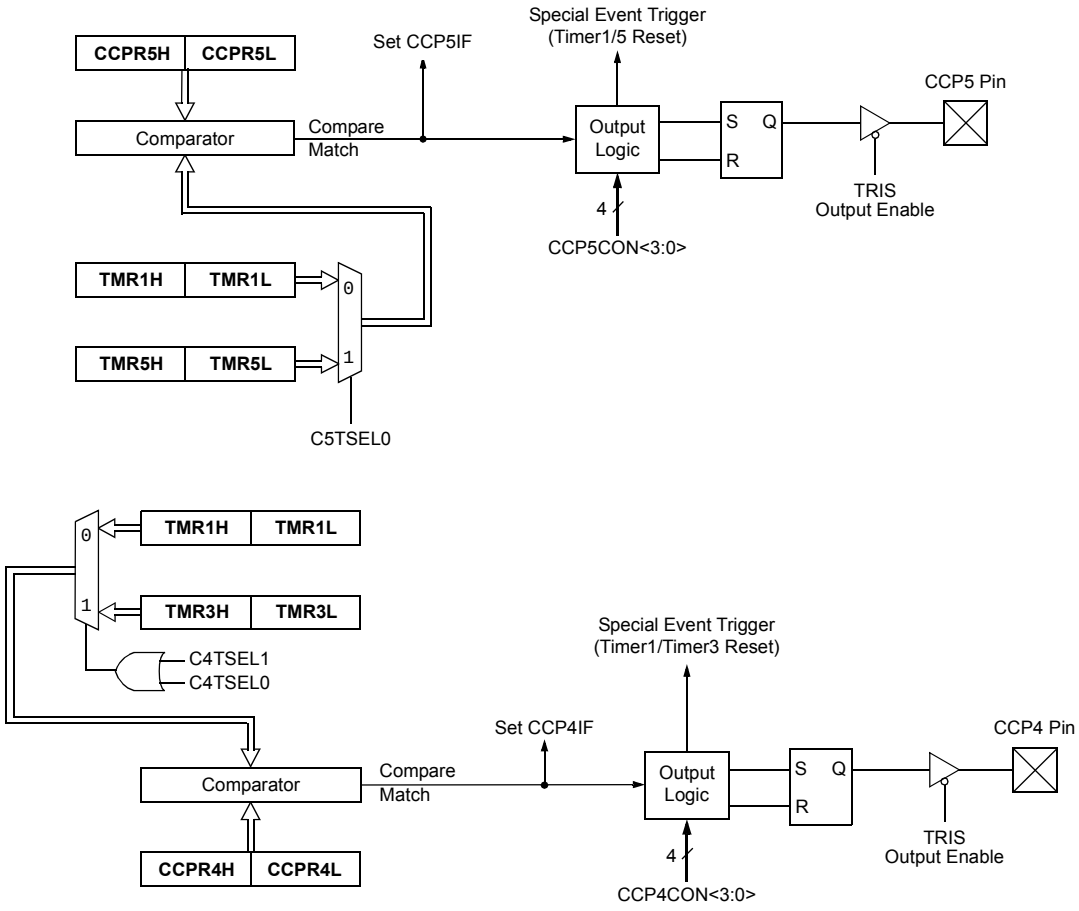
TABLE 17-3: RTCVALH AND RTCVALL REGISTER MAPPING

RTCPTR<1:0>	RTCC Value Register Window	
	RTCVALH	RTCVALL
00	MINUTES	SECONDS
01	WEEKDAY	HOURS
10	MONTH	DAY
11	—	YEAR

The Alarm Value register windows (ALRMVALH and ALRMVALL) use the ALRMPTR bits (ALRMCFG<1:0>) to select the desired Alarm register pair.

By reading or writing to the ALRMVALH register, the Alarm Pointer value, ALRMPTR<1:0>, decrements by '1' until it reaches '00'. When it reaches '00', the ALRMMIN and ALRMSEC values are accessible through ALRMVALH and ALRMVALL until the pointer value is manually changed.

FIGURE 19-2: COMPARE MODE OPERATION BLOCK DIAGRAM



Note: This block diagram uses CCP4 and CCP5 and their appropriate timers as an example. For details on all of the CCP modules and their timer assignments, see Table 19-2 and Table 19-3.

26.4.2 CAPACITANCE CALIBRATION

There is a small amount of capacitance from the internal A/D Converter sample capacitor, as well as stray capacitance from the circuit board traces and pads that affect the precision of capacitance measurements. A measurement of the stray capacitance can be taken by making sure the desired capacitance to be measured has been removed.

After removing the capacitance to be measured:

1. Initialize the A/D Converter and the CTMU.
2. Set EDG1STAT (= 1).
3. Wait for a fixed delay of time, *t*.
4. Clear EDG1STAT.
5. Perform an A/D conversion.
6. Calculate the stray and A/D sample capacitances:

$$C_{OFFSET} = C_{STRAY} + C_{AD} = (I \bullet t)/V$$

Where:

- I is known from the current source measurement step
- t is a fixed delay
- V is measured by performing an A/D conversion

This measured value is then stored and used for calculations of time measurement or subtracted for capacitance measurement. For calibration, it is expected that the capacitance of C_{STRAY} + C_{AD} is approximately known; C_{AD} is approximately 4 pF.

An iterative process may be required to adjust the time, t, that the circuit is charged to obtain a reasonable voltage reading from the A/D Converter. The value of *t* may be determined by setting C_{OFFSET} to a theoretical value and solving for *t*. For example, if C_{STRAY} is theoretically calculated to be 11 pF, and *V* is expected to be 70% of V_{DD} or 2.31V, t would be:

$$(4\ pF + 11\ pF) \bullet 2.31V/0.55\ mA$$

or 63 μs.

See Example 26-3 for a typical routine for CTMU capacitance calibration.

EXAMPLE 26-4: CTMU ROUTINE FOR CAPACITIVE TOUCH SWITCH

```
#include "p18cxxx.h"

#define COUNT 500                       //@ 8MHz = 125uS.
#define DELAY for(i=0;i<COUNT;i++)
#define OPENSW 1000                     //Un-pressed switch value
#define TRIP 300                        //Difference between pressed
                                        //and un-pressed switch
#define HYST 65                         //amount to change
                                        //from pressed to un-pressed
#define PRESSED 1
#define UNPRESSED 0

int main(void)
{
    unsigned int Vread;                 //storage for reading
    unsigned int switchState;
    int i;

    //assume CTMU and A/D have been setup correctly
    //see Example 25-1 for CTMU & A/D setup
    setup();

    CTMUCONbits.CTMUEN = 1;             //Enable the CTMU

    CTMUCONbits.IDISSEN = 1;            //drain charge on the circuit
    DELAY;                              //wait 125us
    CTMUCONbits.IDISSEN = 0;            //end drain of circuit

    CTMUCON3bits.EDG1STAT = 1;          //Begin charging the circuit
                                        //using CTMU current source
    DELAY;                              //wait for 125us
    CTMUCON3bits.EDG1STAT = 0;          //Stop charging circuit

    PIR1bits.ADIF = 0;                  //make sure A/D Int not set
    ADCON1Lbits.SAMP=1;;                //and begin A/D conv.
    while(!PIR1bits.ADIF);              //Wait for A/D convert complete

    Vread = ADRES;                      //Get the value from the A/D

    if(Vread < OPENSW - TRIP)
    {
        switchState = PRESSED;
    }
    else if(Vread > OPENSW - TRIP + HYST)
    {
        switchState = UNPRESSED;
    }
}
```

27.4 Buffer Descriptors and the Buffer Descriptor Table

The registers in Bank 13 are used specifically for endpoint buffer control in a structure known as the Buffer Descriptor Table (BDT). This provides a flexible method for users to construct and control endpoint buffers of various lengths and configuration.

The BDT is composed of Buffer Descriptors (BD) which are used to define and control the actual buffers in the USB RAM space. Each BD, in turn, consists of four registers, where n represents one of the 64 possible BDs (range of 0 to 63):

- BDnSTAT: BD STATUS Register
- BDnCNT: BD Byte Count Register
- BDnADRL: BD Address Low Register
- BDnADRH: BD Address High Register

BDs always occur as a four-byte block in the sequence, BDnSTAT:BDnCNT:BDnADRL:BDnADRH. The address of BDnSTAT is always an offset of (4n – 1, in hexadecimal) from D00h, with n being the buffer descriptor number.

Depending on the buffering configuration used (**Section 27.4.4 "Ping-Pong Buffering"**), there are up to 32, 33 or 64 sets of buffer descriptors. At a minimum, the BDT must be at least 8 bytes long. This is because the USB Specification mandates that every device must have Endpoint 0 with both input and output for initial setup. Depending on the endpoint and buffering configuration, the BDT can be as long as 256 bytes.

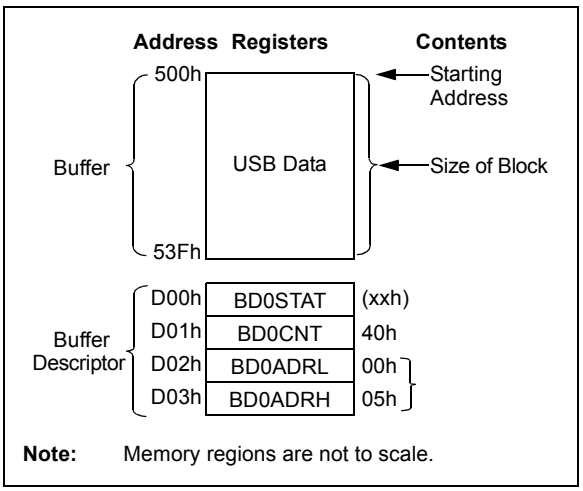
Although they can be thought of as Special Function Registers, the Buffer Descriptor Status and Address registers are not hardware mapped, as conventional microcontroller SFRs in Bank 15 are. If the endpoint corresponding to a particular BD is not enabled, its registers are not used. Instead of appearing as unimplemented addresses, however, they appear as available RAM. Only when an endpoint is enabled by setting the UEPn<1> bit does the memory at those addresses become functional as BD registers. As with any address in the data memory space, the BD registers have an indeterminate value on any device Reset.

Figure 27-5 provides an example of a BD for a 64-byte buffer, starting at 500h. A particular set of BD registers is only valid if the corresponding endpoint has been enabled using the UEPn register. All BD registers are available in USB RAM. The BD for each endpoint should be set up prior to enabling the endpoint.

27.4.1 BD STATUS AND CONFIGURATION

Buffer descriptors not only define the size of an endpoint buffer, but also determine its configuration and control. Most of the configuration is done with the BD STATUS register, BDnSTAT. Each BD has its own unique and correspondingly numbered BDnSTAT register.

FIGURE 27-5: EXAMPLE OF A BUFFER DESCRIPTOR



Note: Memory regions are not to scale.

Unlike other control registers, the bit configuration for the BDnSTAT register is context-sensitive. There are two distinct configurations, depending on whether the microcontroller or the USB module is modifying the BD and buffer at a particular time. Only three bit definitions are shared between the two.

27.4.1.1 Buffer Ownership

Because the buffers and their BDs are shared between the CPU and the USB module, a simple semaphore mechanism is used to distinguish which is allowed to update the BD and associated buffers in memory.

This is done by using the UOWN bit (BDnSTAT<7>) as a semaphore to distinguish which is allowed to update the BD and associated buffers in memory. UOWN is the only bit that is shared between the two configurations of BDnSTAT.

When UOWN is clear, the BD entry is "owned" by the microcontroller core. When the UOWN bit is set, the BD entry and the buffer memory are "owned" by the USB peripheral. The core should not modify the BD or its corresponding data buffer during this time. Note that the microcontroller core can still read BDnSTAT while the SIE owns the buffer and vice versa.

The buffer descriptors have a different meaning based on the source of the register update. Prior to placing ownership with the USB peripheral, the user can configure the basic operation of the peripheral through the BDnSTAT bits. During this time, the byte count and buffer location registers can also be set.

When UOWN is set, the user can no longer depend on the values that were written to the BDs. From this point, the SIE updates the BDs as necessary, overwriting the original BD values. The BDnSTAT register is updated by the SIE with the token PID and the transfer count, BDnCNT, is updated.

29.0 INSTRUCTION SET SUMMARY

The PIC18FXXJ94 of devices incorporates the standard set of 75 PIC18 core instructions, as well as an extended set of eight new instructions for the optimization of code that is recursive or that utilizes a software stack. The extended set is discussed later in this section.

29.1 Standard Instruction Set

The standard PIC18 MCU instruction set adds many enhancements to the previous PIC® MCU 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 four 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 PIC18 instruction set summary in Table 29-2 lists **byte-oriented**, **bit-oriented**, **literal** and **control** operations. Table 29-1 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 four double-word instructions. These instructions were made double-word to contain the required information in 32 bits. In the second word, the 4 MSbs 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 29-1 shows the general formats that the instructions can have. All examples use the convention ‘nnh’ to represent a hexadecimal number.

The Instruction Set Summary, shown in Table 29-2, lists the standard instructions recognized by the Microchip MPASM™ Assembler.

Section 29.1.1 “Standard Instruction Set” provides a description of each instruction.

TABLE 29-2: PIC18F97J94 FAMILY INSTRUCTION SET

Mnemonic, Operands	Description	Cycles	16-Bit Instruction Word MSb LSb	Status Affected	Notes
BYTE-ORIENTED OPERATIONS					
ADDWF f, d, a	Add WREG and f	1	0010 01da ffff ffff	C, DC, Z, OV, N	1, 2
ADDWFC f, d, a	Add WREG and Carry bit to f	1	0010 00da ffff ffff	C, DC, Z, OV, N	1, 2
ANDWF f, d, a	AND WREG with f	1	0001 01da ffff ffff	Z, N	1, 2
CLRF f, a	Clear f	1	0110 101a ffff ffff	Z	2
COMF f, d, a	Complement f	1	0001 11da ffff ffff	Z, N	1, 2
CPFSEQ f, a	Compare f with WREG, Skip =	1 (2 or 3)	0110 001a ffff ffff	None	4
CPFSGT f, a	Compare f with WREG, Skip >	1 (2 or 3)	0110 010a ffff ffff	None	4
CPFSLT f, a	Compare f with WREG, Skip <	1 (2 or 3)	0110 000a ffff ffff	None	1, 2
DECF f, d, a	Decrement f	1	0000 01da ffff ffff	C, DC, Z, OV, N	1, 2, 3, 4
DECFSZ f, d, a	Decrement f, Skip if 0	1 (2 or 3)	0010 11da ffff ffff	None	1, 2, 3, 4
DCFSNZ f, d, a	Decrement f, Skip if Not 0	1 (2 or 3)	0100 11da ffff ffff	None	1, 2
INCF f, d, a	Increment f	1	0010 10da ffff ffff	C, DC, Z, OV, N	1, 2, 3, 4
INCFSZ f, d, a	Increment f, Skip if 0	1 (2 or 3)	0011 11da ffff ffff	None	4
INFSNZ f, d, a	Increment f, Skip if Not 0	1 (2 or 3)	0100 10da ffff ffff	None	1, 2
IORWF f, d, a	Inclusive OR WREG with f	1	0001 00da ffff ffff	Z, N	1, 2
MOVF f, d, a	Move f	1	0101 00da ffff ffff	Z, N	1
MOVFF f_s, f_d	Move f_s (source) to 1st word f_d (destination) 2nd word	2	1100 ffff ffff ffff 1111 ffff ffff ffff	None	
MOVWF f, a	Move WREG to f	1	0110 111a ffff ffff	None	
MULWF f, a	Multiply WREG with f	1	0000 001a ffff ffff	None	1, 2
NEGF f, a	Negate f	1	0110 110a ffff ffff	C, DC, Z, OV, N	
RLCF f, d, a	Rotate Left f through Carry	1	0011 01da ffff ffff	C, Z, N	1, 2
RLNCF f, d, a	Rotate Left f (No Carry)	1	0100 01da ffff ffff	Z, N	
RRCF f, d, a	Rotate Right f through Carry	1	0011 00da ffff ffff	C, Z, N	
RRNCF f, d, a	Rotate Right f (No Carry)	1	0100 00da ffff ffff	Z, N	
SETF f, a	Set f	1	0110 100a ffff ffff	None	1, 2
SUBFWB f, d, a	Subtract f from WREG with Borrow	1	0101 01da ffff ffff	C, DC, Z, OV, N	
SUBWF f, d, a	Subtract WREG from f	1	0101 11da ffff ffff	C, DC, Z, OV, N	1, 2
SUBWFB f, d, a	Subtract WREG from f with Borrow	1	0101 10da ffff ffff	C, DC, Z, OV, N	
SWAPF f, d, a	Swap Nibbles in f	1	0011 10da ffff ffff	None	4
TSTFSZ f, a	Test f, Skip if 0	1 (2 or 3)	0110 011a ffff ffff	None	1, 2
XORWF f, d, a	Exclusive OR WREG with f	1	0001 10da ffff ffff	Z, N	

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.

SLEEP **Enter Sleep Mode**

Syntax: SLEEP

Operands: None

Operation: 00h → WDT,
0 → WDT postscaler,
1 → $\overline{TO}$,
0 → $\overline{PD}$

Status Affected: $\overline{TO}$, $\overline{PD}$

Encoding:

0000	0000	0000	0011

Description: The Power-Down Status bit ($\overline{PD}$) is cleared. The Time-out Status bit ($\overline{TO}$) is set. The Watchdog Timer and its postscaler are cleared.

The processor is put into Sleep mode with the oscillator stopped.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	Process Data	Go to Sleep

Example: SLEEP

Before Instruction
$\overline{TO}$ = ?
$\overline{PD}$ = ?
After Instruction
$\overline{TO}$ = 1 †
$\overline{PD}$ = 0

† If WDT causes wake-up, this bit is cleared.

SUBFWB **Subtract f from W with Borrow**

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

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

Operation: (W) – (f) – ($\overline{C}$) → dest

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

Encoding:

0101	01da	ffff	ffff

Description: Subtract register 'f' and Carry flag (borrow) from W (2's complement method). If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored in register 'f'.

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

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

Words: 1

Cycles: 1

Q Cycle Activity:

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

Example 1: SUBFWB REG, 1, 0

Before Instruction
REG = 3
W = 2
C = 1
After Instruction
REG = FF
W = 2
C = 0
Z = 0
N = 1 ; result is negative

Example 2: SUBFWB REG, 0, 0

Before Instruction
REG = 2
W = 5
C = 1
After Instruction
REG = 2
W = 3
C = 1
Z = 0
N = 0 ; result is positive

Example 3: SUBFWB REG, 1, 0

Before Instruction
REG = 1
W = 2
C = 0
After Instruction
REG = 0
W = 2
C = 1
Z = 1 ; result is zero
N = 0

TABLE 30-9: DC CHARACTERISTICS: POWER-DOWN AND SUPPLY CURRENT PIC18F97J94 FAMILY (INDUSTRIAL)

DC CHARACTERISTICS			Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C for industrial			
Param No.	**Sym.**	**Characteristic**	**Min.**	**Max.**	**Units**	**Conditions**
	VIL	**Input Low Voltage**				
		All I/O Ports:				
D031		Schmitt Trigger Buffer	Vss	0.2 VDD	V	2V ≤ VDD ≤ 3.6V
D031A		RC3 and RC4	Vss	0.3 VDD	V	I2C enabled
D031B			Vss	0.8	V	SMBus enabled
D032		$\overline{MCLR}$	Vss	0.2 VDD	V	
D033		OSC1	Vss	0.2 VDD	V	LP, MS, HS modes
D033A		OSC1	Vss	0.2 VDD	V	EC modes
D034		SOSCI	Vss	0.3 VDD	V	
	VIH	**Input High Voltage**				
		All I/O Ports:				
D041		Schmitt Trigger Buffer	0.8 VDD	VDD	V	2V ≤ VDD ≤ 3.6V
D041A		RC3 and RC4	0.7 VDD	VDD	V	I2C enabled
D041B			2.1	VDD	V	SMBus enabled
D042		$\overline{MCLR}$	0.8 VDD	VDD	V	
D043		OSC1	0.9 VDD	VDD	V	RC mode
D043A		OSC1	0.7 VDD	VDD	V	HS mode
D044		SOSCI	0.7 VDD	VDD	V	
	IIL	**Input Leakage Current(1)**				
D060		I/O Ports	±50	±500	nA	Vss ≤ VPIN ≤ VDD Pin at high-impedance
D061		$\overline{MCLR}$	—	±500	nA	Vss ≤ VPIN ≤ VDD
D063		OSC1	—	1	µA	Vss ≤ VPIN ≤ VDD
D070	IPU	**Weak Pull-up Current**				
		Weak Pull-up Current	50	400	µA	VDD = 3.6V, VPIN = Vss
	VOL	**Output Low Voltage**				
D080		I/O Ports:	—	0.4	V	IOL = 6.6 mA, VDD = 3.6V
		All Ports	—	0.4	V	IOL = 5.0 mA, VDD = 2V
D083		OSC2/CLKO	—	0.4	V	IOL = 6.6 mA, VDD = 3.6V
		(EC modes)	—	0.4	V	IOL = 5.0 mA, VDD = 2V
	VOH	**Output High Voltage(1)**				
D090		I/O Ports:	3.0	—	V	IOH = -3.0 mA, VDD = 3.6V
		All Ports	2.4	—	V	IOH = -6.0 mA, VDD = 3.6V
			1.6	—	V	IOH = -1.0 mA, VDD = 2V
D092		OSC2/CLKO	1.4	—	V	IOH = -3.0 mA, VDD = 2V
		(INTOSC, EC modes)				
			2.4	—	V	IOH = -6.0 mA, VDD = 3.6V
			1.4	—	V	IOH = -1.0 mA, VDD = 2V
		Capacitive Loading Specs on Output Pins				
D100	COSC2	OSC2 Pin	—	20	pF	In HS mode when external clock is used to drive OSC1
D101	CIO	All I/O Pins and OSC2	—	50	pF	To meet the AC Timing Specifications
D102	CB	SCLx, SDAx	—	400	pF	I2C Specification

Note 1: Negative current is defined as current sourced by the pin.

31.0 DEVELOPMENT SUPPORT

The PIC® microcontrollers (MCU) and dsPIC® digital signal controllers (DSC) are supported with a full range of software and hardware development tools:

- Integrated Development Environment
  - MPLAB® X IDE Software
- Compilers/Assemblers/Linkers
  - MPLAB XC Compiler
  - MPASM™ Assembler
  - MPLINK™ Object Linker/ MPLIB™ Object Librarian
  - MPLAB Assembler/Linker/Librarian for Various Device Families
- Simulators
  - MPLAB X SIM Software Simulator
- Emulators
  - MPLAB REAL ICE™ In-Circuit Emulator
- In-Circuit Debuggers/Programmers
  - MPLAB ICD 3
  - PICkit™ 3
- Device Programmers
  - MPLAB PM3 Device Programmer
- Low-Cost Demonstration/Development Boards, Evaluation Kits and Starter Kits
- Third-party development tools

31.1 MPLAB X Integrated Development Environment Software

The MPLAB X IDE is a single, unified graphical user interface for Microchip and third-party software, and hardware development tool that runs on Windows®, Linux and Mac OS® X. Based on the NetBeans IDE, MPLAB X IDE is an entirely new IDE with a host of free software components and plug-ins for high-performance application development and debugging. Moving between tools and upgrading from software simulators to hardware debugging and programming tools is simple with the seamless user interface.

With complete project management, visual call graphs, a configurable watch window and a feature-rich editor that includes code completion and context menus, MPLAB X IDE is flexible and friendly enough for new users. With the ability to support multiple tools on multiple projects with simultaneous debugging, MPLAB X IDE is also suitable for the needs of experienced users.

Feature-Rich Editor:

- Color syntax highlighting
- Smart code completion makes suggestions and provides hints as you type
- Automatic code formatting based on user-defined rules
- Live parsing

User-Friendly, Customizable Interface:

- Fully customizable interface: toolbars, toolbar buttons, windows, window placement, etc.
- Call graph window

Project-Based Workspaces:

- Multiple projects
- Multiple tools
- Multiple configurations
- Simultaneous debugging sessions

File History and Bug Tracking:

- Local file history feature
- Built-in support for Bugzilla issue tracker

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Microchip received ISO/TS-16949:2009 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona; Gresham, Oregon and design centers in California and India. The Company's quality system processes and procedures are for its PIC® MCUs and dsPIC® DSCs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
═ ISO/TS 16949 ═

Trademarks

ISBN: 978-1-5224-0896-3