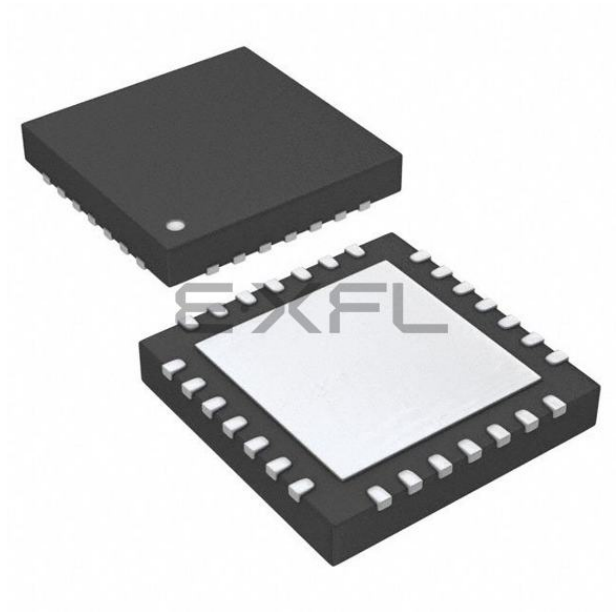


Welcome to E-XFL.COM

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	CANbus, I²C, LINbus, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, DMA, POR, PWM, WDT
Number of I/O	25
Program Memory Size	32KB (16K x 16)
Program Memory Type	FLASH
EEPROM Size	1K x 8
RAM Size	2K x 8
Voltage - Supply (Vcc/Vdd)	2.3V ~ 5.5V
Data Converters	A/D 24x12b; D/A 1x5b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	28-VQFN Exposed Pad
Supplier Device Package	28-QFN (6x6)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18f25k83t-i-ml

9.6 Returning from Interrupt Service Routine (ISR)

The “Return from Interrupt” instruction (RETFIE) is used to mark the end of an ISR.

When RETFIE 1 instruction is executed, the PC is loaded with the saved PC value from the top of the PC stack. Saved context is also restored with the execution of this instruction. Thus, execution returns to the previous state of operation that existed before the interrupt occurred.

When RETFIE 0 instruction is executed, the saved context is not restored back to the registers.

9.7 Interrupt Latency

By assigning each interrupt with a vector address/number (MVECEN = 1), scanning of all interrupts is not necessary to determine the source of the interrupt.

When MVECEN = 1, Vectored interrupt controller requires three clock cycles to vector to the ISR from main routine, thereby removing dependency of interrupt timing on compiled code.

There is a fixed latency of three instruction cycles between the completion of the instruction active when the interrupt occurred and the first instruction of the Interrupt Service Routine. Figure 9-7, Figure 9-8 and Figure 9-9 illustrates the sequence of events when a peripheral interrupt is asserted when the last executed instruction is one-cycle, two-cycle and three-cycle respectively, when MVECEN = 1.

After the Interrupt Flag Status bit is set, the current instruction completes executing. In the first latency cycle, the contents of the PC, STATUS, WREG, BSR, FSR0/1/2, PRODL/H and PCLATH/U registers are context saved and the IVTBASE+ Vector number is calculated. In the second latency cycle, the PC is loaded with the calculated vector table address for the interrupt source and the starting address of the ISR is fetched. In the third latency cycle, the PC is loaded with the ISR address. All the latency cycles are executed as a FNOP instruction.

When MVECEN = 0, Vectored interrupt controller requires two clock cycles to vector to the ISR from main routine. There is a latency of two instruction cycles plus the software latency between the completion of the instruction active when the interrupt occurred and the first instruction of the Interrupt Service Routine.

EXAMPLE 9-3: SETTING UP VECTORED INTERRUPTS USING MPASM

```
ISR_TMR0:  CODE       0x8C0                 ; ISR code at 0x08C0 in PFM
       BANKSEL        PIR0                  ; Select bank for PIR0
       BCF            PIR3, TMR0IF          ; Clear TMR0IF
       BTG            LATC, 0, ACCESS       ; Code to execute in ISR
       RETFIE         1                     ; Return from ISR

InterruptInit:
       BANKSEL        INTCON0               ; Select bank for INTCON0
       BSF            INTCON0, GIEH         ; Enable high priority interrupts
       BSF            INTCON0, GIEL         ; Enable low priority interrupts
       BSF            INTCON0, IPEN_INTCON0 ; Enable interrupt priority

       BANKSEL        PIE0                  ; Select bank for PIE0
       BSF            PIE3, TMR0IE          ; Enable TMR0 interrupt
       BSF            PIE4, TMR1IE          ; Enable TMR1 interrupt

       BCF            IPR3, TMR0IP          ; Make TMR0 interrupt low priority
       RETURN         1

VectorTableInit:
       ; Set IVTBASE (optional - default is 0x000008)
       MOVLW          0x00                  ; This is optional
       MOVWF          IVTBASEU, ACCESS      ; If not included, then the
       MOVLW          0x40                  ; hardware default value of
       MOVWF          IVTBASEH, ACCESS      ; 0x0008 will be taken.
       MOVLW          0x08
       MOVWF          IVTBASEL, ACCESS

       ; TMR0 vector at IVTBASE + 2*(TMR0 vector number i.e. 31) = 0x4046
       MOVLW          0x00                  ; Load TBLPTR with the
       MOVWF          TBLPTRU, ACCESS       ; PFM memory location to be
       MOVLW          0x40                  ; written to.
       MOVWF          TBLPTRH, ACCESS
       MOVLW          0x46
       MOVWF          TBLPTRL, ACCESS

       ; Write the contents of TMR0 vector location
       ; ISR_TMR0_ADDRESS >> 2 = 0x08C0 >> 2 = 0x0230
       MOVLW          0x30                  ; Low byte first
       MOVWF          TABLAT, ACCESS
       TBLWT*+                              ; Write to temp table latch

       MOVLW          0x02                  ; High byte next
       MOVWF          TABLAT, ACCESS
       TBLWT*+                              ; Write to temp table latch

       ; Write to PFM now using NVMCON
       BANKSEL        NVMCON1               ; Select bank for NVMCON1
       MOVLW          0x84                  ; Setting to write to PFM
       MOVWF          NVMCON1

       MOVLW          0x55                  ; Required unlock sequence
       MOVWF          NVMCON2
       MOVLW          0xAA
       MOVWF          NVMCON2
       BSF            NVMCON1, WR           ; Start writing to PFM

       BTFSC          NVMCON1, WR           ; Wait for write to complete
       GOTO           $-2

       RETURN         1
```

REGISTER 9-16: PIE3: PERIPHERAL INTERRUPT ENABLE REGISTER 3

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
TMR0IE	U1IE	U1EIE	U1TXIE	U1RXIE	I2C1EIE	I2C1IE	I2C1TXIE
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
'1' = Bit is set	'0' = Bit is cleared	

bit 7 **TMR0IE:** TMR0 Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 6 **U1IE:** UART1 Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 5 **U1EIE:** UART1 Framing Error Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 4 **U1TXIE:** UART1 Transmit Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 3 **U1RXIE:** UART1 Receive Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 2 **I2C1EIE:** I^2C1 Error Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 1 **I2C1IE:** I^2C1 Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 0 **I2C1TXIE:** I^2C1 Transmit Interrupt Enable bit
1 = Enabled
0 = Disabled

Example 12-3 shows the sequence to do a 16 x 16 unsigned multiplication. Equation 12-1 shows the algorithm that is used. The 32-bit result is stored in four registers (RES<3:0>).

EQUATION 12-1: 16 x 16 UNSIGNED MULTIPLICATION ALGORITHM

$$
\begin{aligned}
RES3:RES0 &= ARG1H:ARG1L \bullet ARG2H:ARG2L \\
&= (ARG1H \bullet ARG2H \bullet 2^{16}) + \\
&\quad (ARG1H \bullet ARG2L \bullet 2^{8}) + \\
&\quad (ARG1L \bullet ARG2H \bullet 2^{8}) + \\
&\quad (ARG1L \bullet ARG2L)
\end{aligned}
$$

EXAMPLE 12-3: 16 x 16 UNSIGNED MULTIPLY ROUTINE

```
    MOVF    ARG1L, W
    MULWF   ARG2L           ; ARG1L * ARG2L->
                            ; PRODH:PRODL
    MOVFF   PRODH, RES1     ;
    MOVFF   PRODL, RES0     ;
;
    MOVF    ARG1H, W
    MULWF   ARG2H           ; ARG1H * ARG2H->
                            ; PRODH:PRODL
    MOVFF   PRODH, RES3     ;
    MOVFF   PRODL, RES2     ;
;
    MOVF    ARG1L, W
    MULWF   ARG2H           ; ARG1L * ARG2H->
                            ; PRODH:PRODL
    MOVF    PRODL, W        ;
    ADDWF   RES1, F         ; Add cross
    MOVF    PRODH, W        ; products
    ADDWFC  RES2, F         ;
    CLRF    WREG            ;
    ADDWFC  RES3, F         ;
;
    MOVF    ARG1H, W        ;
    MULWF   ARG2L           ; ARG1H * ARG2L->
                            ; PRODH:PRODL
    MOVF    PRODL, W        ;
    ADDWF   RES1, F         ; Add cross
    MOVF    PRODH, W        ; products
    ADDWFC  RES2, F         ;
    CLRF    WREG            ;
    ADDWFC  RES3, F         ;
```

Example 12-4 shows the sequence to do a 16 x 16 signed multiply. Equation 12-2 shows the algorithm used. The 32-bit result is stored in four registers (RES<3:0>). To account for the sign bits of the arguments, the MSb for each argument pair is tested and the appropriate subtractions are done.

EQUATION 12-2: 16 x 16 SIGNED MULTIPLICATION ALGORITHM

$$
\begin{aligned}
RES3:RES0 &= ARG1H:ARG1L \bullet ARG2H:ARG2L \\
&= (ARG1H \bullet ARG2H \bullet 2^{16}) + \\
&\quad (ARG1H \bullet ARG2L \bullet 2^{8}) + \\
&\quad (ARG1L \bullet ARG2H \bullet 2^{8}) + \\
&\quad (ARG1L \bullet ARG2L) + \\
&\quad (-1 \bullet ARG2H\langle 7\rangle \bullet ARG1H:ARG1L \bullet 2^{16}) + \\
&\quad (-1 \bullet ARG1H\langle 7\rangle \bullet ARG2H:ARG2L \bullet 2^{16})
\end{aligned}
$$

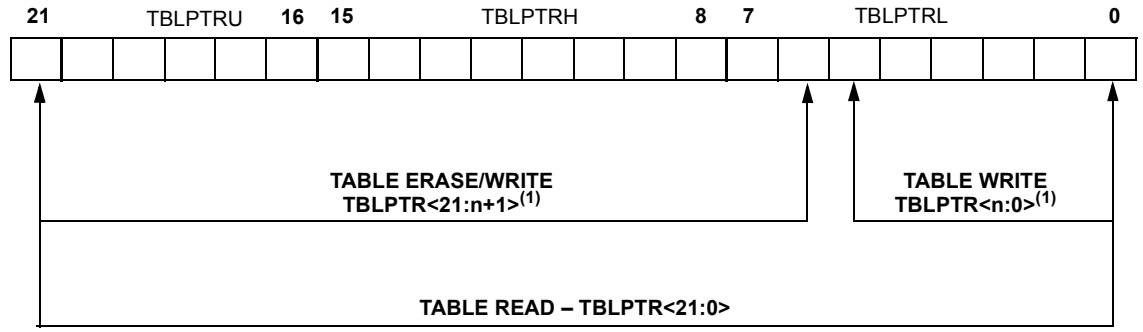
EXAMPLE 12-4: 16 x 16 SIGNED MULTIPLY ROUTINE

```
    MOVF    ARG1L, W
    MULWF   ARG2L           ; ARG1L * ARG2L ->
                            ; PRODH:PRODL
    MOVFF   PRODH, RES1     ;
    MOVFF   PRODL, RES0     ;
;
    MOVF    ARG1H, W
    MULWF   ARG2H           ; ARG1H * ARG2H ->
                            ; PRODH:PRODL
    MOVFF   PRODH, RES3     ;
    MOVFF   PRODL, RES2     ;
;
    MOVF    ARG1L, W
    MULWF   ARG2H           ; ARG1L * ARG2H ->
                            ; PRODH:PRODL
    MOVF    PRODL, W        ;
    ADDWF   RES1, F         ; Add cross
    MOVF    PRODH, W        ; products
    ADDWFC  RES2, F         ;
    CLRF    WREG            ;
    ADDWFC  RES3, F         ;
;
    MOVF    ARG1H, W        ;
    MULWF   ARG2L           ; ARG1H * ARG2L ->
                            ; PRODH:PRODL
    MOVF    PRODL, W        ;
    ADDWF   RES1, F         ; Add cross
    MOVF    PRODH, W        ; products
    ADDWFC  RES2, F         ;
    CLRF    WREG            ;
    ADDWFC  RES3, F         ;
;
    BTFSS   ARG2H, 7        ; ARG2H:ARG2L neg?
    BRA     SIGN_ARG1       ; no, check ARG1
    MOVF    ARG1L, W        ;
    SUBWF   RES2            ;
    MOVF    ARG1H, W        ;
    SUBWFB  RES3
;
SIGN_ARG1
    BTFSS   ARG1H, 7        ; ARG1H:ARG1L neg?
    BRA     CONT_CODE       ; no, done
    MOVF    ARG2L, W        ;
    SUBWF   RES2            ;
    MOVF    ARG2H, W        ;
    SUBWFB  RES3
;
CONT_CODE
    :
```

TABLE 13-3: TABLE POINTER OPERATIONS WITH TBLRD AND TBLWT INSTRUCTIONS

Example	Operation on Table Pointer
TBLRD* TBLWT*	TBLPTR is not modified
TBLRD*+ TBLWT*+	TBLPTR is incremented after the read/write
TBLRD*- TBLWT*-	TBLPTR is decremented after the read/write
TBLRD+* TBLWT+*	TBLPTR is incremented before the read/write

FIGURE 13-3: TABLE POINTER BOUNDARIES BASED ON OPERATION



Note 1: Refer to Table 5-4 for the row size values.

13.3.8 ERASING THE DATA EEPROM MEMORY

Data EEPROM Memory can be erased by writing 0xFF to all locations in the Data EEPROM Memory that needs to be erased.

EXAMPLE 13-7: DATA EEPROM REFRESH ROUTINE

```
        CLRF    NVMADRL             ; Start at address 0
        BCF     NVMCON1, CFGS       ; Set for memory
        BCF     NVMCON1, EEPGD      ; Set for Data EEPROM
        BCF     INTCON0, GIE        ; Disable interrupts
        BSF     NVMCON1, WREN       ; Enable writes
Loop                                ; Loop to refresh array
        BSF     NVMCON1, RD         ; Read current address
        MOVLW   55h                 ;
        MOVWF   NVMCON2             ; Write 55h
        MOVLW   0AAh                ;
        MOVWF   NVMCOM2             ; Write 0AAh
        BSF     NVMCON1, WR         ; Set WR bit to begin write
        BTFSC   NVMCON1, WR         ; Wait for write to complete
        BRA     $-2
        INCFSZ  NVMADRL, F          ; Increment address
        BRA     LOOP                ; Not zero, do it again

        BCF     NVMCON1, WREN       ; Disable writes
        BSF     INTCON0, GIE        ; Enable interrupts
```

15.0 DIRECT MEMORY ACCESS (DMA)

15.1 Introduction

The Direct Memory Access (DMA) module is designed to service data transfers between different memory regions directly without intervention from the CPU. By eliminating the need for CPU-intensive management of handling interrupts intended for data transfers, the CPU now can spend more time on other tasks.

PIC18(L)F25/26K83 family has two DMA modules which can be independently programmed to transfer data between different memory locations, move different data sizes, and use a wide range of hardware triggers to initiate transfers. The two DMA registers can even be programmed to work together, in order to carry out more complex data transfers without CPU overhead.

Key features of the DMA module include:

- Support access to the following memory regions:
  - GPR and SFR space (R/W)
  - Program Flash Memory (R only)
  - Data EEPROM Memory (R only)
- Programmable priority between the DMA and CPU Operations. Refer to **Section 3.1 "System Arbitration"** for details.
- Programmable Source and Destination address modes
  - Fixed address
  - Post-increment address
  - Post-decrement address
- Programmable Source and Destination sizes
- Source and destination pointer register, dynamically updated and reloadable
- Source and destination count register, dynamically updated and reloadable
- Programmable auto-stop based on Source or Destination counter
- Software triggered transfers
- Multiple user selectable sources for hardware triggered transfers
- Multiple user selectable sources for aborting DMA transfers

15.2 DMA Registers

The operation of the DMA module has the following registers:

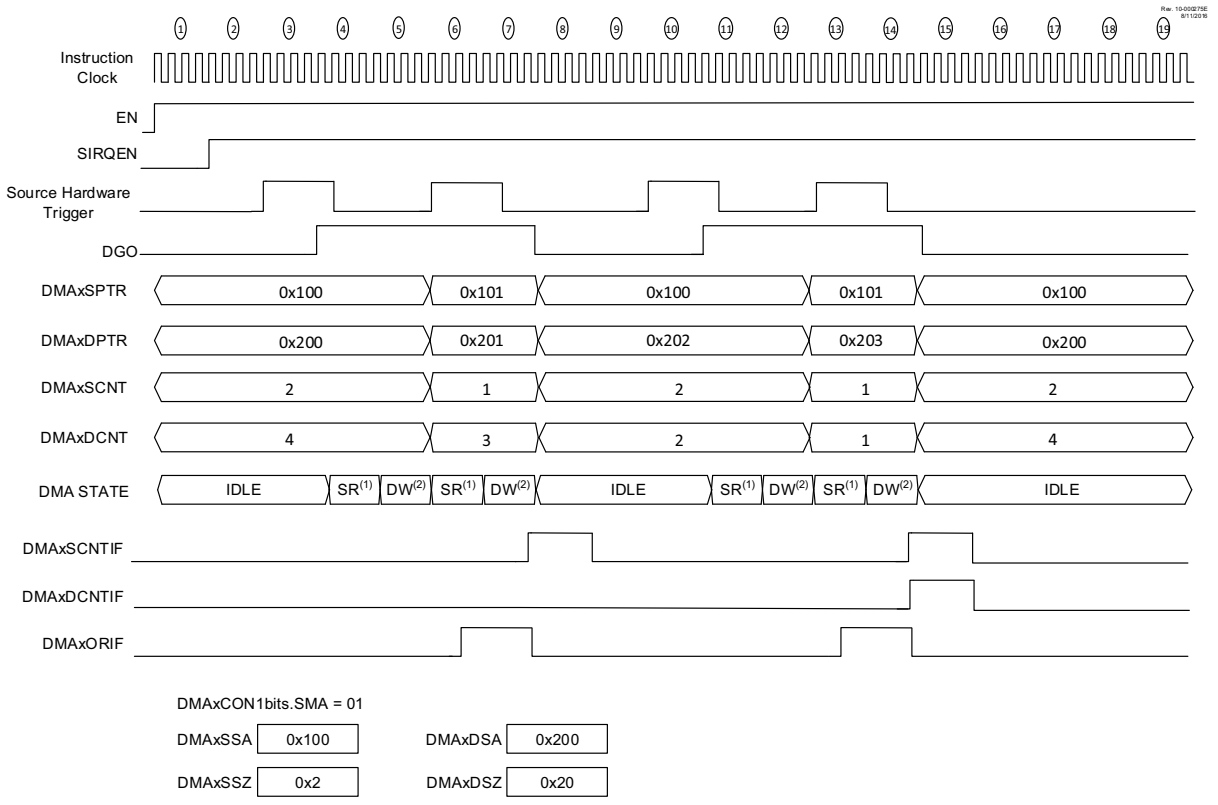
- Control registers (DMAxCON0, DMAxCON1)
- Data buffer register (DMAxBUF)
- Source Start Address Register (DMAxSSAU:H:L)
- Source Pointer Register (DMAxSPTRU:H:L)
- Source Message Size Register (DMAxSSZH:L)
- Source Count Register (DMAxSCNTH:L)
- Destination Start Address Register (DMAxDSAH:L)
- Destination Pointer Register (DMAxDPTRH:L)
- Destination Message Size Register (DMAxDSZH:L)
- Destination Count Register (DMAxDCNTH:L)
- Start Interrupt Request Source Register (DMAxSIRQ)
- Abort Interrupt Request Source Register (DMAxAIRQ)

These registers are detailed in **Section 15.13 "Register definitions: DMA"**.

15.9.5 OVERRUN INTERRUPT

The Overrun Interrupt flag is set if the DMA receives a trigger to start a new message before the current message is completed.

FIGURE 15-9: OVERRUN INTERRUPT



Note 1: SR – Source Read

2: DW – Destination Write

REGISTER 15-15: DMAxDSAH: DMAx DESTINATION START ADDRESS HIGH REGISTER

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
DSA<15:8>							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n/n = Value at POR and BOR/Value at all other Resets	1 = bit is set	0 = bit is cleared	x = bit is unknown u = bit is unchanged

bit 7-0 **DSA<15:8>:** Destination Start Address bits

REGISTER 15-16: DMAxDPTRL: DMAx DESTINATION POINTER LOW REGISTER

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
DPTR<7:0>							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n/n = Value at POR and BOR/Value at all other Resets	1 = bit is set	0 = bit is cleared	x = bit is unknown u = bit is unchanged

bit 7-0 **DPTR<7:0>:** Current Destination Address Pointer

REGISTER 15-17: DMAxDPTRH: DMAx DESTINATION POINTER HIGH REGISTER

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
DPTR<15:8>							
bit 7							bit 0

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
-n/n = Value at POR and BOR/Value at all other Resets	1 = bit is set	0 = bit is cleared	x = bit is unknown u = bit is unchanged

bit 7-0 **DPTR<15:8>:** Current Destination Address Pointer

REGISTER 16-9: RxyI2C: I²C PAD Rxy CONTROL REGISTER

U-0	R/W-0/0	R/W-0/0	R/W-0/0	U-0	U-0	R/W-0/0	R/W-0/0
—	SLEW	PU<1:0>		—	—	TH<1:0>	
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
'1' = Bit is set	'0' = Bit is cleared	HS = Hardware set

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

bit 6 **SLEW**: I²C Specific Slew Rate Limiting is Enabled
- 1 = I²C specific slew rate limiting is enabled. Standard pad slew limiting is disabled. The SLRxy bit is ignored.
- 0 = Standard GPIO Slew Rate; enabled/disabled via SLRxy bit.

bit 5-4 **PU<1:0>**: I²C Pull-up Selection bits
- 11 = Reserved
- 10 = 10x current of standard weak pull-up
- 01 = 2x current of standard weak pull-up
- 00 = Standard GPIO weak pull-up, enabled via WPUxy bit

bit 3-2 **Unimplemented**: Read as '0'

bit 1-0 **TH<1:0>**: I²C Input Threshold Selection bits
- 11 = SMBus 3.0 (1.35 V) input threshold
- 10 = SMBus 2.0 (2.1 V) input threshold
- 01 = I²C specific input thresholds
- 00 = Standard GPIO Input pull-up, enabled via INLVLxy registers

TABLE 16-10: I2C PAD CONTROL REGISTERS

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
RB1I2C	—	SLEW	PU<1:0>		—	—	TH<1:0>	
RB2I2C	—	SLEW	PU<1:0>		—	—	TH<1:0>	
RC3I2C	—	SLEW	PU<1:0>		—	—	TH<1:0>	
RC4I2C	—	SLEW	PU<1:0>		—	—	TH<1:0>	

22.7 Register Definitions: Timer2/4/6 Control

Long bit name prefixes for the Timer2/4/6 peripherals are shown in Table 22-2. Refer to **Section 1.3.2.2 "Long Bit Names"** for more information.

TABLE 22-2: OPERATING MODES

Peripheral	Bit Name Prefix
Timer2	T2
Timer4	T4
Timer6	T6

REGISTER 22-1: TxCLK: TIMERx CLOCK SELECTION REGISTER

U-0	U-0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	—	—	CS<3:0>			
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
'1' = Bit is set	'0' = Bit is cleared	

bit 7-4 **Unimplemented:** Read as '0'

bit 3-0 **CS<3:0>:** Timerx Clock Selection bits

CS<3:0>	T2TMR	TMR4	TMR6
	Clock Source	Clock Source	Clock Source
1111	Reserved	Reserved	Reserved
1110	CLC4_out	CLC4_out	CLC4_out
1101	CLC3_out	CLC3_out	CLC3_out
1100	CLC2_out	CLC2_out	CLC2_out
1011	CLC1_out	CLC1_out	CLC1_out
1010	ZCD_OUT	ZCD_OUT	ZCD_OUT
1001	NCO1OUT	NCO1OUT	NCO1OUT
1000	CLKREF_OUT	CLKREF_OUT	CLKREF_OUT
0111	SOSC	SOSC	SOSC
0110	MFINTOSC (32 kHz)	MFINTOSC (32 kHz)	MFINTOSC (32 kHz)
0101	MFINTOSC (500 kHz)	MFINTOSC (500 kHz)	MFINTOSC (500 kHz)
0100	LFINTOSC	LFINTOSC	LFINTOSC
0011	HFINTOSC	HFINTOSC	HFINTOSC
0010	FOSC	FOSC	FOSC
0001	FOSC/4	FOSC/4	FOSC/4
0000	Pin selected by T2INPPS	Pin selected by T4INPPS	Pin selected by T6INPPS

23.3.1 CCPx PIN CONFIGURATION

The software must configure the CCPx pin as an output by clearing the associated TRIS bit and defining the appropriate output pin through the RxyPPS registers. See **Section 17.0 "Peripheral Pin Select (PPS) Module"** for more details.

Note:	Clearing the CCPxCON register will force the CCPx compare output latch to the default low level. This is not the PORT I/O data latch.

23.3.2 TIMER1 MODE RESOURCE

In Compare mode, Timer1 must be running in either Timer mode or Synchronized Counter mode. The compare operation may not work in Asynchronous Counter mode.

See **Section 21.0 "Timer1/3/5 Module with Gate Control"** for more information on configuring Timer1.

Note:	Clocking Timer1 from the system clock (FOSC) should not be used in Compare mode. In order for Compare mode to recognize the trigger event on the CCPx pin, TImer1 must be clocked from the instruction clock (FOSC/4) or from an external clock source.

23.3.3 AUTO-CONVERSION TRIGGER

All CCPx modes set the CCP interrupt flag (CCPxIF). When this flag is set and a match occurs, an auto-conversion trigger can take place if the CCP module is selected as the conversion trigger source.

Refer to **Section 37.2.5 "Auto-Conversion Trigger"** for more information.

Note:	Removing the match condition by changing the contents of the CCPRxH and CCPRxL register pair, between the clock edge that generates the Auto-conversion Trigger and the clock edge that generates the Timer1 Reset, will preclude the Reset from occurring

23.3.4 COMPARE DURING SLEEP

Since FOSC is shut down during Sleep mode, the Compare mode will not function properly during Sleep, unless the timer is running. The device will wake on interrupt (if enabled).

23.4 PWM Overview

Pulse-Width Modulation (PWM) is a scheme that provides power to a load by switching quickly between fully ON and fully OFF states. The PWM signal resembles a square wave where the high portion of the signal is considered the ON state and the low portion of the signal is considered the OFF state. The high portion, also known as the pulse width, can vary in time and is defined in steps. A larger number of steps applied, which lengthens the pulse width, also supplies more power to the load. Lowering the number of steps applied, which shortens the pulse width, supplies less power. The PWM period is defined as the duration of one complete cycle or the total amount of on and off time combined.

PWM resolution defines the maximum number of steps that can be present in a single PWM period. A higher resolution allows for more precise control of the pulse-width time and in turn the power that is applied to the load.

The term duty cycle describes the proportion of the on time to the off time and is expressed in percentages, where 0% is fully off and 100% is fully on. A lower duty cycle corresponds to less power applied and a higher duty cycle corresponds to more power applied.

Figure 23-3 shows a typical waveform of the PWM signal.

23.4.1 STANDARD PWM OPERATION

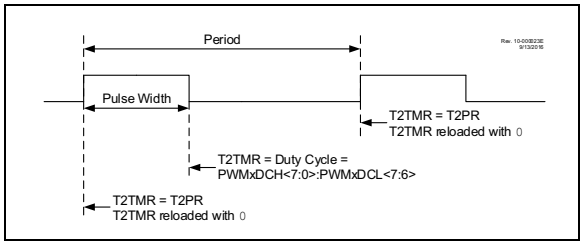
The standard PWM mode generates a Pulse-Width Modulation (PWM) signal on the CCPx pin with up to ten bits of resolution. The period, duty cycle, and resolution are controlled by the following registers:

- T2PR registers
- T2CON registers
- CCPRxL and CCPRxH registers
- CCPxCON registers

It is required to have FOSC/4 as the clock input to TMR2/4/6 for correct PWM operation. Figure 23-4 shows a simplified block diagram of PWM operation.

Note:	The corresponding TRIS bit must be cleared to enable the PWM output on the CCPx pin.

FIGURE 23-3: CCP PWM OUTPUT SIGNAL



REGISTER 25-13: SMTxCPWL: SMT CAPTURED PULSE WIDTH REGISTER – LOW BYTE

R-x/x	R-x/x	R-x/x	R-x/x	R-x/x	R-x/x	R-x/x	R-x/x
SMTxCPW<7:0>							
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
‘1’ = Bit is set	‘0’ = Bit is cleared	

bit 7-0 **SMTxCPW<7:0>**: Significant bits of the SMT PW Latch – Low Byte

REGISTER 25-14: SMTxCPWH: SMT CAPTURED PULSE WIDTH REGISTER – HIGH BYTE

R-x/x	R-x/x	R-x/x	R-x/x	R-x/x	R-x/x	R-x/x	R-x/x
SMTxCPW<15:8>							
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
‘1’ = Bit is set	‘0’ = Bit is cleared	

bit 7-0 **SMTxCPW<15:8>**: Significant bits of the SMT PW Latch – High Byte

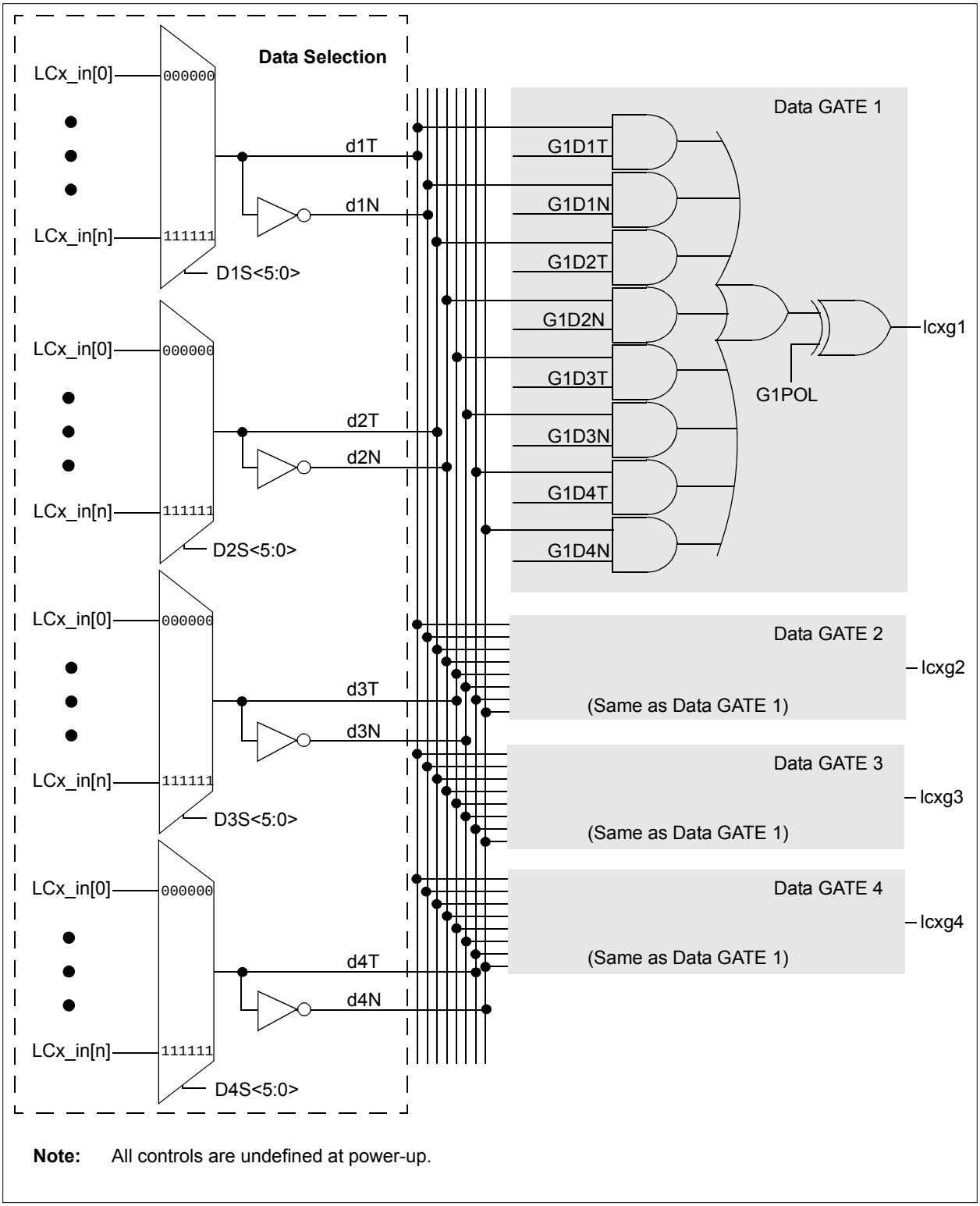
REGISTER 25-15: SMTxCPWU: SMT CAPTURED PULSE WIDTH REGISTER – UPPER BYTE

R-x/x	R-x/x	R-x/x	R-x/x	R-x/x	R-x/x	R-x/x	R-x/x
SMTxCPW<23:16>							
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
‘1’ = Bit is set	‘0’ = Bit is cleared	

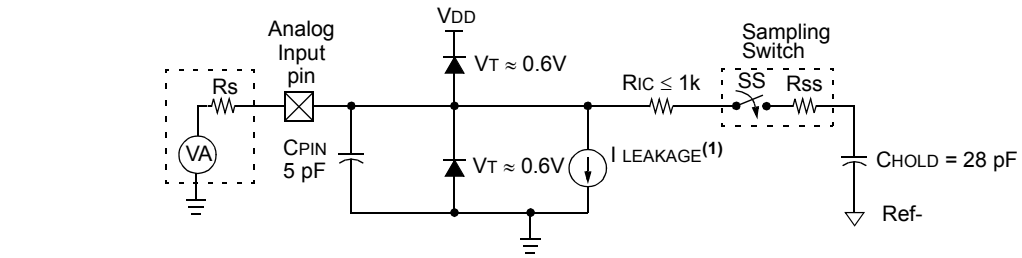
bit 7-0 **SMTxCPW<23:16>**: Significant bits of the SMT PW Latch – Upper Byte

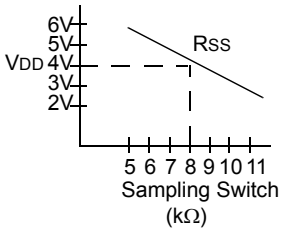
FIGURE 27-2: INPUT DATA SELECTION AND GATING



Note: All controls are undefined at power-up.

FIGURE 37-4: ANALOG INPUT MODEL



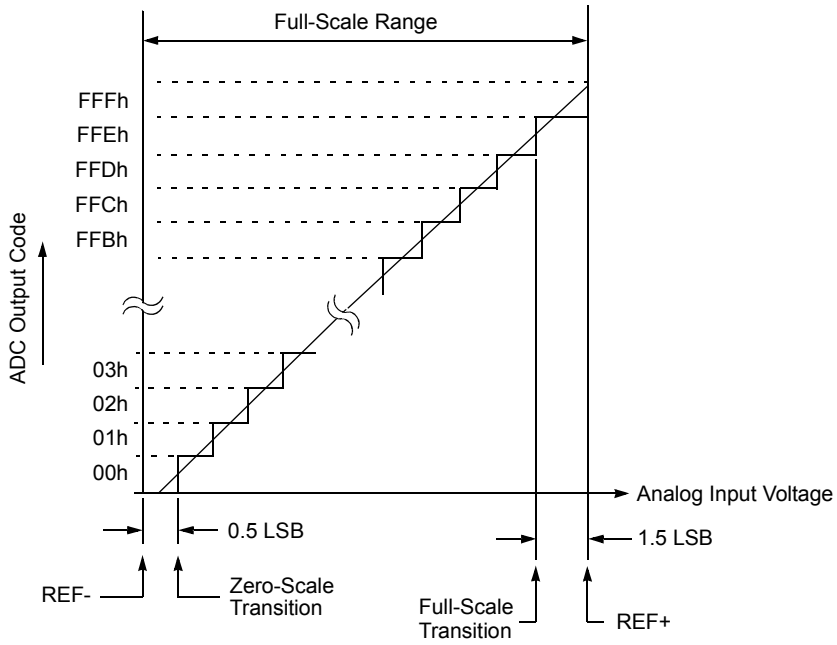


Legend:

CHOLD	= Sample/Hold Capacitance
CPIN	= Input Capacitance
I LEAKAGE	= Leakage current at the pin due to various junctions
RIC	= Interconnect Resistance
RSS	= Resistance of Sampling Switch
SS	= Sampling Switch
VT	= Threshold Voltage

Note 1: Refer to Table 45-4 (parameter D340 and D341).

FIGURE 37-5: ADC TRANSFER FUNCTION



37.7 Register Definitions: ADC Control

REGISTER 37-1: ADCON0: ADC CONTROL REGISTER 0

R/W-0/0	R/W-0/0	U-0	R/W-0/0	U-0	R/W-0/0	U-0	R/W/HC-0
ON	CONT	—	CS	—	FM	—	GO
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
'1' = Bit is set	'0' = Bit is cleared	HC = Bit is cleared by hardware

bit 7 **ON:** ADC Enable bit
1 = ADC is enabled
0 = ADC is disabled

bit 6 **CONT:** ADC Continuous Operation Enable bit
1 = GO is retriggered upon completion of each conversion trigger until ADTIF is set (if ADSOI is set) or until GO is cleared (regardless of the value of ADSOI)
0 = ADC is cleared upon completion of each conversion trigger

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

bit 4 **CS:** ADC Clock Selection bit
1 = Clock supplied from FRC dedicated oscillator
0 = Clock supplied by FOSC, divided according to ADCLK register

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

bit 2 **FM:** ADC results Format/alignment Selection
1 = ADRES and PREV data are right-justified
0 = ADRES and PREV data are left-justified, zero-filled

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

bit 0 **GO:** ADC Conversion Status bit(1)
1 = ADC conversion cycle in progress. Setting this bit starts an ADC conversion cycle. The bit is cleared by hardware as determined by the CONT bit
0 = ADC conversion completed/not in progress

Note 1: This bit requires ON bit to be set.

2: If cleared by software while a conversion is in progress, the results of the conversion up to this point will be transfered to ADRES and the state machine will be reset, but the ADIF interrupt flag bit will not be set; filter and threshold operations will not be performed.

REGISTER 37-13: ADCAP: ADC ADDITIONAL SAMPLE CAPACITOR SELECTION REGISTER

U-0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	—	ADCAP<4:0>				
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
‘1’ = Bit is set	‘0’ = Bit is cleared	

bit 7-5 **Unimplemented**: Read as ‘0’

bit 4-0 **ADCAP<4:0>**: ADC Additional Sample Capacitor Selection bits
11111 = 31 pF
11110 = 30 pF
11101 = 29 pF
•
•
•
00011 = 3 pF
00010 = 2 pF
00001 = 1 pF
00000 = No additional capacitance

REGISTER 37-14: ADRPT: ADC REPEAT SETTING REGISTER

R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
RPT<7:0>							
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’
u = Bit is unchanged	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
‘1’ = Bit is set	‘0’ = Bit is cleared	

bit 7-0 **RPT<7:0>**: ADC Repeat Threshold bits
Counts the number of times that the ADC has been triggered and is used along with CNT to determine when the error threshold is checked when the computation is Low-pass Filter, Burst Average, or Average modes. See Table 37-2 for more details.

40.0 HIGH/LOW-VOLTAGE DETECT (HLVD)

The PIC18(L)F25/26K83 family of devices has a High/Low-Voltage Detect module (HLVD). This is a programmable circuit that sets both a device voltage trip point and the direction of change from that point (positive going, negative going or both). If the device experiences an excursion past the trip point in that direction, an interrupt flag is set. If the interrupt is enabled, the program execution branches to the interrupt vector address and the software responds to the interrupt.

Complete control of the HLVD module is provided through the HLVDCON0 and HLVDCON1 register. This allows the circuitry to be “turned off” by the user under software control, which minimizes the current consumption for the device.

The module’s block diagram is shown in Figure 40-1.

Since the HLVD can be software enabled through the EN bit, setting and clearing the enable bit does not produce a false HLVD event glitch. Each time the HLVD module is enabled, the circuitry requires some time to stabilize. The RDY bit (HLVDCON0<4>) is a read-only bit used to indicate when the band gap reference voltages are stable.

The module can only generate an interrupt after the module is turned ON and the band gap reference voltages are ready.

The INTH and INTL bits determine the overall operation of the module. When INTH is set, the module monitors for rises in V_{DD} above the trip point set by the HLVDCON1 register. When INTL is set, the module monitors for drops in V_{DD} below the trip point set by the HLVDCON1 register. When both the INTH and INTL bits are set, any changes above or below the trip point set by the HLVDCON1 register can be monitored.

The OUT bit can be read to determine if the voltage is greater than or less than the voltage level selected by the HLVDCON1 register.

TABLE 43-1: REGISTER FILE SUMMARY FOR PIC18(L)F25/26K83 DEVICES (CONTINUED)

Addr	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on page
3BC8h - 3AEEh	—	Unimplemented								—
3AEDh	CANRXPPS	—	—	—	CANRXPPS					264
3AECh	—	Unimplemented								—
3AEBh	U2CTSPPS	—	—	—	U2CTSPPS					264
3AEAh	U2RXPPS	—	—	—	U2RXPPS					264
3AE9h	—	Unimplemented								—
3AE8h	U1CTSPPS	—	—	—	U1CTSPPS					264
3AE7h	U1RXPPS	—	—	—	U1RXPPS					264
3AE6h	I2C2SDAPPS	—	—	—	I2C2SDAPPS					264
3AE5h	I2C2SCLPPS	—	—	—	I2C2SCLPPS					264
3AE4h	I2C1SDAPPS	—	—	—	I2C1SDAPPS					264
3AE3h	I2C1SCLPPS	—	—	—	I2C1SCLPPS					264
3AE2h	SPI1SSPPS	—	—	—	SPI1SSPPS					264
3AE1h	SPI1SDIPPS	—	—	—	SPI1SDIPPS					264
3AE0h	SPI1SCKPPS	—	—	—	SPI1SCKPPS					264
3ADFh	ADACTPPS	—	—	—	ADACTPPS					264
3ADEh	CLCIN3PPS	—	—	—	CLCIN3PPS					264
3ADDh	CLCIN2PPS	—	—	—	CLCIN2PPS					264
3ADCh	CLCIN1PPS	—	—	—	CLCIN1PPS					264
3ADBh	CLCIN0PPS	—	—	—	CLCIN0PPS					264
3ADAh	MD1SRCPPS	—	—	—	MD1SRCPPS					264
3AD9h	MD1CARHPPS	—	—	—	MD1CARHPPS					264
3AD8h	MD1CARLPPS	—	—	—	MD1CARLPPS					264
3AD7h	CWG3INPPS	—	—	—	CWG3INPPS					264
3AD6h	CWG2INPPS	—	—	—	CWG2INPPS					264
3AD5h	CWG1INPPS	—	—	—	CWG1INPPS					264
3AD4h	SMT2SIGPPS	—	—	—	SMT2SIGPPS					264
3AD3h	SMT2WINPPS	—	—	—	SMT2WINPPS					264
3AD2h	SMT1SIGPPS	—	—	—	SMT1SIGPPS					264
3AD1h	SMT1WINPPS	—	—	—	SMT1WINPPS					264
3AD0h	CCP4PPS	—	—	—	CCP4PPS					264
3ACFh	CCP3PPS	—	—	—	CCP3PPS					264
3ACEh	CCP2PPS	—	—	—	CCP2PPS					264
3ACDh	CCP1PPS	—	—	—	CCP1PPS					264
3ACCh	T6INPPS	—	—	—	T6INPPS					264
3ACBh	T4INPPS	—	—	—	T4INPPS					264
3ACAh	T2INPPS	—	—	—	T2INPPS					264
3AC9h	T5GPPS	—	—	—	T5GPPS					264
3AC8h	T5CLKIPPS	—	—	—	T5CLKIPPS					264
3AC7h	T3GPPS	—	—	—	T3GPPS					264
3AC6h	T3CLKIPPS	—	—	—	T3CLKIPPS					264
3AC5h	T1GPPS	—	—	—	T1GPPS					264
3AC4h	T1CKIPPS	—	—	—	T1CKIPPS					264
3AC3h	T0CKIPPS	—	—	—	T0CKIPPS					264
3AC2h	INT2PPS	—	—	—	INT2PPS					264
3AC1h	INT1PPS	—	—	—	INT1PPS					264
3AC0h	INT0PPS	—	—	—	INT0PPS					264
3ABFh	PPSLOCK	—	—	—	—	—	—	—	PPSLOCKED	268

Legend: x = unknown, u = unchanged, — = unimplemented, q = value depends on condition
Note 1: Not present in LF devices.

TABLE 43-1: REGISTER FILE SUMMARY FOR PIC18(L)F25/26K83 DEVICES (CONTINUED)

Addr	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on page
39D7h - 39D2h	—	Unimplemented								—
39D1h	VREGCON(1)	—	—	—	—	—	—	VREGPM	—	166
39D0h	BORCON	SBOREN	—	—	—	—	—	—	BORRDY	75
39CFh - 39C8h	—	Unimplemented								—
39C7h	PMD7	CANMD	—	—	—	—	—	DMA2MD	DMA1MD	282
39C6h	PMD6	—	SMT2MD	SMT1MD	CLC4MD	CLC3MD	CLC2MD	CLC1MD	DSMMD	281
39C5h	PMD5	—	—	U2MD	U1MD	—	SPI1MD	I2C2MD	I2C1MD	280
39C4h	PMD4	CWG3MD	CWG2MD	CWG1MD	—	—	—	—	—	279
39C3h	PMD3	PWM8MD	PWM7MD	PWM6MD	PWM5MD	CCP4MD	CCP3MD	CCP2MD	CCP1MD	278
39C2h	PMD2	—	DACMD	ADCMD	—	—	CMP2MD	CMP1MD	ZCDMD	277
39C1h	PMD1	NCO1MD	TMR6MD	TMR5MD	TMR4MD	TMR3MD	TMR2MD	TMR1MD	TMR0MD	276
39C0h	PMD0	SYSCMD	FVRMD	HLVDMD	CRCMD	SCANMD	NVMMD	CLKRMD	IOCMD	275
39BFh - 39AAh	—	Unimplemented								—
39A9h	PIR9	—	CLC4IF	CCP4IF	CLC3IF	CWG3IF	CCP3IF	TMR6IF	TMR5IF	136
39A8h	PIR8	TMR5IF	INT2IF	CLC2IF	CWG2IF	CCP2IF	TMR4IF	TMR3GIF	TMR3IF	135
39A7h	PIR7	U2IF	U2EIF	U2TXIF	U2RXIF	I2C2EIF	I2C2IF	I2C2TXIF	I2C2RXIF	134
39A6h	PIR6	DMA2AIF	DMA2ORIF	DMA2DCNTIF	DMA2SCNTIF	SMT2PWAIF	SMT2PRAIF	SMT2IF	C2IF	133
39A5h	PIR5	IRXIF	WAKIF	ERRIF	TXB2IF/ TXBnIF	TXB1IF	TXB0IF	RXB1IF/ RXBnIF	RXB0IF/ FIFOFIF	132
39A4h	PIR4	INT1IF	CLC1IF	CWG1IF	NCO1IF	CCP1IF	TMR2IF	TMR1GIF	TMR1IF	131
39A3h	PIR3	TMR0IF	U1IF	U1EIF	U1TXIF	U1RXIF	I2C1EIF	I2C1IF	I2C1TXIF	130
39A2h	PIR2	I2C1RXIF	SPI1IF	SPI1TXIF	SPI1RXIF	DMA1AIF	DMA1ORIF	DMA1DCNTIF	DMA1SCNTIF	128
39A1h	PIR1	SMT1PWAIF	SMT1PRAIF	SMT1IF	C1IF	ADTIF	ADIF	ZCDIF	INT0IF	128
39A0h	PIR0	IOCIF	CRCIF	SCANIF	NVMIF	CSWIF	OSFIF	HLVDIF	SWIF	127
399Fh - 399Ah	—	Unimplemented								—
3999h	PIE9	—	CLC4IE	CCP4IE	CLC3IE	CWG3IE	CCP3IE	TMR6IE	TMR5IE	146
3998h	PIE8	TMR5IE	INT2IE	CLC2IE	CWG2IE	CCP2IE	TMR4IE	TMR3GIE	TMR3IE	145
3997h	PIE7	U2IE	U2EIE	U2TXIE	U2RXIE	I2C2EIE	I2C2IE	I2C2TXIE	I2C2RXIE	144
3996h	PIE6	DMA2AIE	DMA2ORIE	DMA2DCNTIE	DMA2SCNTIE	SMT2PWAIE	SMT2PRAIE	SMT2IE	C2IE	143
3995h	PIE5	IRXIE	WAKIE	ERRIE	TXB2IE/TXB-nIE	TXB1IE	TXB0IE	RXB1IE/ RXBnIE	RXB0IF/ FIFOFIF	142
3994h	PIE4	INT1IE	CLC1IE	CWG1IE	NCO1IE	CCP1IE	TMR2IE	TMR1GIE	TMR1IE	141
3993h	PIE3	TMR0IE	U1IE	U1EIE	U1TXIE	U1RXIE	I2C1EIE	I2C1IE	I2C1TXIE	140
3992h	PIE2	I2C1RXIE	SPI1IE	SPI1TXIE	SPI1RXIE	DMA1AIE	DMA1ORIE	DMA1DCNTIE	DMA1SCNTIE	139
3991h	PIE1	SMT1PWAIE	SMT1PRAIE	SMT1IE	C1IE	ADTIE	ADIE	ZCDIE	INT0IE	138
3990h	PIE0	IOCIE	CRCIE	SCANIE	NVMIE	CSWIE	OSFIE	HLVDIE	SWIE	137
398Fh - 398Ah	—	Unimplemented								—
3989h	IPR9	—	CLC4IP	CCP4IP	CLC3IP	CWG3IP	CCP3IP	TMR6IP	TMR5IP	156
3988h	IPR8	TMR5IP	INT2IP	CLC2IP	CWG2IP	CCP2IP	TMR4IP	TMR3GIP	TMR3IP	155
3987h	IPR7	U2IP	U2EIP	U2TXIP	U2RXIP	I2C2EIP	I2C2IP	I2C2TXIP	I2C2RXIP	154
3986h	IPR6	DMA2AIP	DMA2ORIP	DMA2DCNTIP	DMA2SCNTIP	SMT2PWAIP	SMT2PRAIP	SMT2IP	C2IP	153
3985h	IPR5	IRXIP	WAKIP	ERRIP	TXB2IP/TXB-nIP	TXB1IP	TXB0IP	RXB1IP/ RXBnIP	RXB0IP/ FIFOFIP	152
3984h	IPR4	INT1IP	CLC1IP	CWG1IP	NCO1IP	CCP1IP	TMR2IP	TMR1GIP	TMR1IP	151
3983h	IPR3	TMR0IP	U1IP	U1EIP	U1TXIP	U1RXIP	I2C1EIP	I2C1IP	I2C1TXIP	150
3982h	IPR2	I2C1RXIP	SPI1IP	SPI1TXIP	SPI1RXIP	DMA1AIP	DMA1ORIP	DMA1DCNTIP	DMA1SCNTIP	149
3981h	IPR1	SMT1PWAIP	SMT1PRAIP	SMT1IP	C1IP	ADTIP	ADIP	ZCDIP	INT0IP	148

Legend: x = unknown, u = unchanged, — = unimplemented, q = value depends on condition
Note 1: Not present in LF devices.