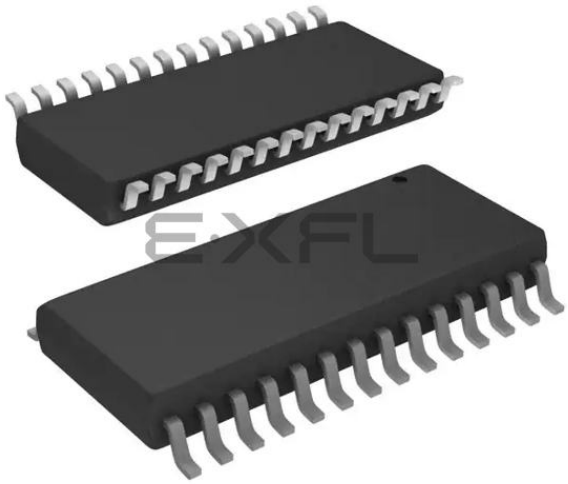


Welcome to E-XFL.COM

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, LINbus, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, DMA, HLVD, POR, PWM, WDT
Number of I/O	25
Program Memory Size	16KB (8K x 16)
Program Memory Type	FLASH
EEPROM Size	256 x 8
RAM Size	1K x 8
Voltage - Supply (Vcc/Vdd)	1.8V ~ 3.6V
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-SOIC (0.295", 7.50mm Width)
Supplier Device Package	28-SOIC
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18lf24k42t-i-so

TABLE 4-5: SPECIAL FUNCTION REGISTER MAP FOR PIC18(L)F24/25K42 DEVICES BANK 61

3DFFh	—	3DDFh	U2FIFO	3DBFh	—	3D9Fh	—	3D7Fh	—	3D5Fh	I2C2CON2	3D3Fh	—	3D1Fh	—
3DFEh	—	3DDEh	U2BRGH	3DBEh	—	3D9Eh	—	3D7Eh	—	3D5Eh	I2C2CON1	3D3Eh	—	3D1Eh	—
3DFDh	—	3DDDh	U2BRGL	3DBDh	—	3D9Dh	—	3D7Dh	—	3D5Dh	I2C2CON0	3D3Dh	—	3D1Dh	—
3DFCh	—	3DDCh	U2CON2	3DBCh	—	3D9Ch	—	3D7Ch	I2C1BTO	3D5Ch	I2C2ADR3	3D3Ch	—	3D1Ch	SPI1CLK
3DFBh	—	3DDBh	U2CON1	3DBBh	—	3D9Bh	—	3D7Bh	I2C1CLK	3D5Bh	I2C2ADR2	3D3Bh	—	3D1Bh	SPI1INTE
3DFAh	U1ERRIE	3DDAh	U2CON0	3DBAh	—	3D9Ah	—	3D7Ah	I2C1PIE	3D5Ah	I2C2ADR1	3D3Ah	—	3D1Ah	SPI1INTF
3DF9h	U1ERRIR	3DD9h	—	3DB9h	—	3D99h	—	3D79h	I2C1PIR	3D59h	I2C2ADR0	3D39h	—	3D19h	SPI1BAUD
3DF8h	U1UIR	3DD8h	U2P3L	3DB8h	—	3D98h	—	3D78h	I2C1STAT1	3D58h	I2C2ADB1	3D38h	—	3D18h	SPI1TWIDTH
3DF7h	U1FIFO	3DD7h	—	3DB7h	—	3D97h	—	3D77h	I2C1STAT0	3D57h	I2C2ADB0	3D37h	—	3D17h	SPI1STATUS
3DF6h	U1BRGH	3DD6h	U2P2L	3DB6h	—	3D96h	—	3D76h	I2C1ERR	3D56h	I2C2CNT	3D36h	—	3D16h	SPI1CON2
3DF5h	U1BRGL	3DD5h	—	3DB5h	—	3D95h	—	3D75h	I2C1CON2	3D55h	I2C2TXB	3D35h	—	3D15h	SPI1CON1
3DF4h	U1CON2	3DD4h	U2P1L	3DB4h	—	3D94h	—	3D74h	I2C1CON1	3D54h	I2C2RXB	3D34h	—	3D14h	SPI1CON0
3DF3h	U1CON1	3DD3h	—	3DB3h	—	3D93h	—	3D73h	I2C1CON0	3D53h	—	3D33h	—	3D13h	SPI1TCNTH
3DF2h	U1CON0	3DD2h	U2TXB	3DB2h	—	3D92h	—	3D72h	I2C1ADR3	3D52h	—	3D32h	—	3D12h	SPI1TCNTL
3DF1h	U1P3H	3DD1h	—	3DB1h	—	3D91h	—	3D71h	I2C1ADR2	3D51h	—	3D31h	—	3D11h	SPI1TXB
3DF0h	U1P3L	3DD0h	U2RXB	3DB0h	—	3D90h	—	3D70h	I2C1ADR1	3D50h	—	3D30h	—	3D10h	SPI1RXB
3DEFh	U1P2H	3DCFh	—	3DAFh	—	3D8Fh	—	3D6Fh	I2C1ADR0	3D4Fh	—	3D2Fh	—	3D0Fh	—
3DEEh	U1P2L	3DCEh	—	3DAEh	—	3D8Eh	—	3D6Eh	I2C1ADB1	3D4Eh	—	3D2Eh	—	3D0Eh	—
3DEDh	U1P1H	3DCDh	—	3DADh	—	3D8Dh	—	3D6Dh	I2C1ADB0	3D4Dh	—	3D2Dh	—	3D0Dh	—
3DECh	U1P1L	3DCCh	—	3DACh	—	3D8Ch	—	3D6Ch	I2C1CNT	3D4Ch	—	3D2Ch	—	3D0Ch	—
3DEBh	U1TXCHK	3DCBh	—	3DABh	—	3D8Bh	—	3D6Bh	I2C1TXB	3D4Bh	—	3D2Bh	—	3D0Bh	—
3DEAh	U1TXB	3DCAh	—	3DAAh	—	3D8Ah	—	3D6Ah	I2C1RXB	3D4Ah	—	3D2Ah	—	3D0Ah	—
3DE9h	U1RXCHK	3DC9h	—	3DA9h	—	3D89h	—	3D69h	—	3D49h	—	3D29h	—	3D09h	—
3DE8h	U1RXB	3DC8h	—	3DA8h	—	3D88h	—	3D68h	—	3D48h	—	3D28h	—	3D08h	—
3DE7h	—	3DC7h	—	3DA7h	—	3D87h	—	3D67h	—	3D47h	—	3D27h	—	3D07h	—
3DE6h	—	3DC6h	—	3DA6h	—	3D86h	—	3D66h	I2C2BTO	3D46h	—	3D26h	—	3D06h	—
3DE5h	—	3DC5h	—	3DA5h	—	3D85h	—	3D65h	I2C2CLK	3D45h	—	3D25h	—	3D05h	—
3DE4h	—	3DC4h	—	3DA4h	—	3D84h	—	3D64h	I2C2PIE	3D44h	—	3D24h	—	3D04h	—
3DE3h	—	3DC3h	—	3DA3h	—	3D83h	—	3D63h	I2C2PIR	3D43h	—	3D23h	—	3D03h	—
3DE2h	U2ERRIE	3DC2h	—	3DA2h	—	3D82h	—	3D62h	I2C2STAT1	3D42h	—	3D22h	—	3D02h	—
3DE1h	U2ERRIR	3DC1h	—	3DA1h	—	3D81h	—	3D61h	I2C2STAT0	3D41h	—	3D21h	—	3D01h	—
3DE0h	U2UIR	3DC0h	—	3DA0h	—	3D80h	—	3D60h	I2C2ERR	3D40h	—	3D20h	—	3D00h	—

Legend: Unimplemented data memory locations and registers, read as '0'.

7.0 DEVICE CONFIGURATION INFORMATION

The Device Configuration Information (DCI) is a dedicated region in the Program memory space mapped from 3FFF00h to 3FFF09h. The data stored in these locations is ready-only and cannot be erased or modified.

Refer to Table 7-1: Device Configuration Information for the complete DCI table address and description. The DCI holds information about the device which is useful for programming and Bootloader applications. These locations are read-only and cannot be erased or modified.

TABLE 7-1: DEVICE CONFIGURATION INFORMATION

ADDRESS	Name	DESCRIPTION	VALUE	UNITS
3FFF00h-3FFF01h	ERSIZ	Erase Row Size	32	Words
3FFF02h-3FFF03h	WLSIZ	Number of write latches	64	
3FFF04h-3FFF05h	URSIZ	Number of User Rows	See Table 7-2	Rows
3FFF06h-3FFF07h	EESIZ	EE Data memory size	256	Bytes
3FFF08h-3FFF09h	PCNT	Pin Count	28	Pins

TABLE 7-2: MEMORY SIZE AND NUMBER OF USER ROWS

Part Name	Memory size	Number of user rows
PIC18(L)F24K42	8K	256
PIC18(L)F25K42	16K	512

7.1 DIA and DCI Access

The DIA and DCI addresses are read-only and cannot be erased or modified. See **Section 15.2 "Device Information Area, Device Configuration Area, User ID, Device ID and Configuration Word Access"** for more information on accessing these memory locations.

Development tools, such as device programmers and debuggers, may be used to read the DIA and DCI regions, similar to the Device ID and Revision ID.

8.11 Start-up Sequence

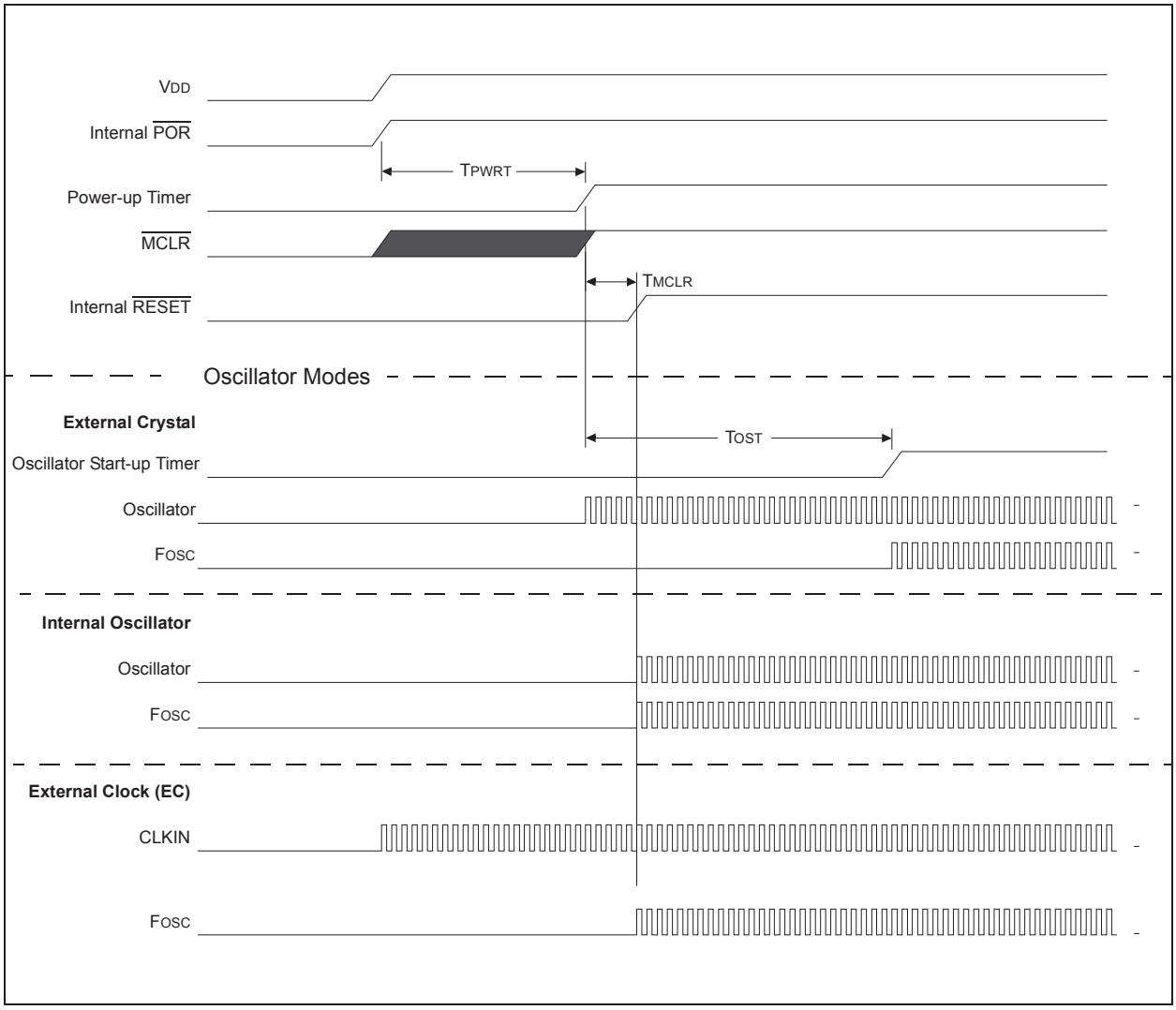
Upon the release of a POR or BOR, the following must occur before the device will begin executing:

1. Power-up Timer runs to completion (if enabled).
2. Oscillator start-up timer runs to completion (if required for selected oscillator source).
3. $\overline{MCLR}$ must be released (if enabled).

The total time out will vary based on oscillator configuration and Power-up Timer configuration. See **Section 9.0 "Oscillator Module (with Fail-Safe Clock Monitor)"** for more information.

The Power-up Timer and oscillator start-up timer run independently of $\overline{MCLR}$ Reset. If $\overline{MCLR}$ is kept low long enough, the Power-up Timer and oscillator Start-up Timer will expire. Upon bringing $\overline{MCLR}$ high, the device will begin execution after 10 FOSC cycles (see Figure 8-4). This is useful for testing purposes or to synchronize more than one device operating in parallel.

FIGURE 8-4: RESET START-UP SEQUENCE



REGISTER 8-3: PCON1: POWER CONTROL REGISTER 1

U-0	U-0	U-0	U-0	U-0	U-0	R/W/HC-1/u	U-0
—	—	—	—	—	—	$\overline{\text{MEMV}}$	—
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	-m/n = Value at POR and BOR/Value at all other Resets
'1' = Bit is set	'0' = Bit is cleared	q = Value depends on condition

bit 7-2 **Unimplemented:** Read as '0'

bit 1 $\overline{\textbf{MEMV}}$**:** Memory Violation Flag bit
- 1 = No memory violation Reset occurred or set to '1' by firmware
- 0 = A memory violation Reset occurred (set to '0' in hardware when a Memory Violation occurs)

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

TABLE 8-4: SUMMARY OF REGISTERS ASSOCIATED WITH RESETS

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
BORCON	SBOREN	—	—	—	—	—	—	BORRDY	90
PCON0	STKOVF	STKUNF	$\overline{\text{WDTWV}}$	$\overline{\text{RWDT}}$	$\overline{\text{RMCLR}}$	$\overline{\text{RI}}$	$\overline{\text{POR}}$	$\overline{\text{BOR}}$	95
PCON1	—	—	—	—	—	—	$\overline{\text{MEMV}}$	—	96

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

REGISTER 11-21: PIE7: PERIPHERAL INTERRUPT ENABLE REGISTER 7

U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0	U-0	R/W-0/0	R/W-0/0
—	—	INT2IE	CLC2IE	CWG2IE	—	CCP2IE	TMR4IE
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 7-6 **Unimplemented:** Read as '0'

bit 5 **INT2IE:** External Interrupt 2 Enable bit
1 = Enabled
0 = Disabled

bit 4 **CLC2IE:** CLC2 Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 3 **CWG2IE:** CWG2 Interrupt Enable bit
1 = Enabled
0 = Disabled

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

bit 1 **CCP2IE:** CCP2 Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 0 **TMR4IE:** TMR4 Interrupt Enable bit
1 = Enabled
0 = Disabled

17.5 DMA Message Transfers

Once the Enable bit is set to start DMA message transfers, the Source/Destination pointer and counter registers are initialized to the conditions shown in Table 17-3.

TABLE 17-3: DMA INITIAL CONDITIONS

Register	Value loaded
DMAxSPTR<21:0>	DMAxSSA<21:0>
DMAxSCNT<11:0>	DMAxSSZ<11:0>
DMAxDPTR<15:0>	DMAxDSA<15:0>
DMAxDCNT<11:0>	DMAxDSZ<11:0>

During the DMA Operation after each transaction, Table 17-4, and Table 17-5 indicate how the Source/Destination pointer and counter registers are modified

TABLE 17-4: DMA SOURCE POINTER/COUNTER DURING OPERATION

Register	Modified Source Counter/Pointer Value
DMAxSCNT<11:0> != 1	DMAxSCNT = DMAxSCNT -1
	SMODE = 00: DMAxSPTR = DMAxSPTR
	SMODE = 01: DMAxSPTR = DMAxSPTR + 1
	SMODE = 10: DMAxSPTR = DMAxSPTR - 1
DMAxSCNT<11:0> == 1	DMAxSCNT = DMAxSSZ
	DMAxSPTR = DMAxSSA

TABLE 17-5: DMA DESTINATION POINTER/COUNTER DURING OPERATION

Register	Modified Destination Counter/Pointer Value
DMAxDCNT<11:0> != 1	DMAxDCNT = DMAxDCNT -1
	DMODE = 00: DMAxDPTR = DMAxDPTR
	DMODE = 01: DMAxDPTR = DMAxDPTR + 1
	DMODE = 10: DMAxDPTR = DMAxDPTR - 1
DMAxDCNT<11:0> == 1	DMAxDCNT = DMAxDSZ
	DMAxDPTR = DMAxDSA

The following sections discuss how to initiate and terminate DMA transfers.

17.5.1 STARTING DMA MESSAGE TRANSFERS

The DMA can initiate data transactions by either of the following two conditions:

1. User software control
2. Hardware trigger, SIRQ

17.5.1.1 User Software Control

Software starts or stops DMA transaction by setting/clearing the DGO bit. The DGO bit is also used to indicate whether a DMA hardware trigger has been received and a message is in progress.

Note 1: Software start can only occur if the EN bit (DMAxCON1) is set.

2: If the CPU writes to the DGO bit while it is already set, there is no effect on the system, the DMA will continue to operate normally.

19.0 PERIPHERAL PIN SELECT (PPS) MODULE

The Peripheral Pin Select (PPS) module connects peripheral inputs and outputs to the device I/O pins. Only digital signals are included in the selections. All analog inputs and outputs remain fixed to their assigned pins. Input and output selections are independent as shown in the simplified block diagram Figure 19-1.

The peripheral input is selected with the peripheral xxxPPS register (Register 19-1), and the peripheral output is selected with the PORT RxyPPS register (Register 19-2). For example, to select PORTC<7> as the UART1 RX input, set U1RXPPS to `5'b1 0111`, and to select PORTC<6> as the UART1 TX output set RC6PPS to `6'b01 0011`.

19.1 PPS Inputs

Each peripheral has a PPS register with which the inputs to the peripheral are selected. Inputs include the device pins.

Multiple peripherals can operate from the same source simultaneously. Port reads always return the pin level regardless of peripheral PPS selection. If a pin also has analog functions associated, the ANSEL bit for that pin must be cleared to enable the digital input buffer.

Although every peripheral has its own PPS input selection register, the selections are identical for every peripheral as shown in Register 19-1.

Note:	The notation "xxx" in the register name is a place holder for the peripheral identifier. For example, INT0PPS.

19.2 PPS Outputs

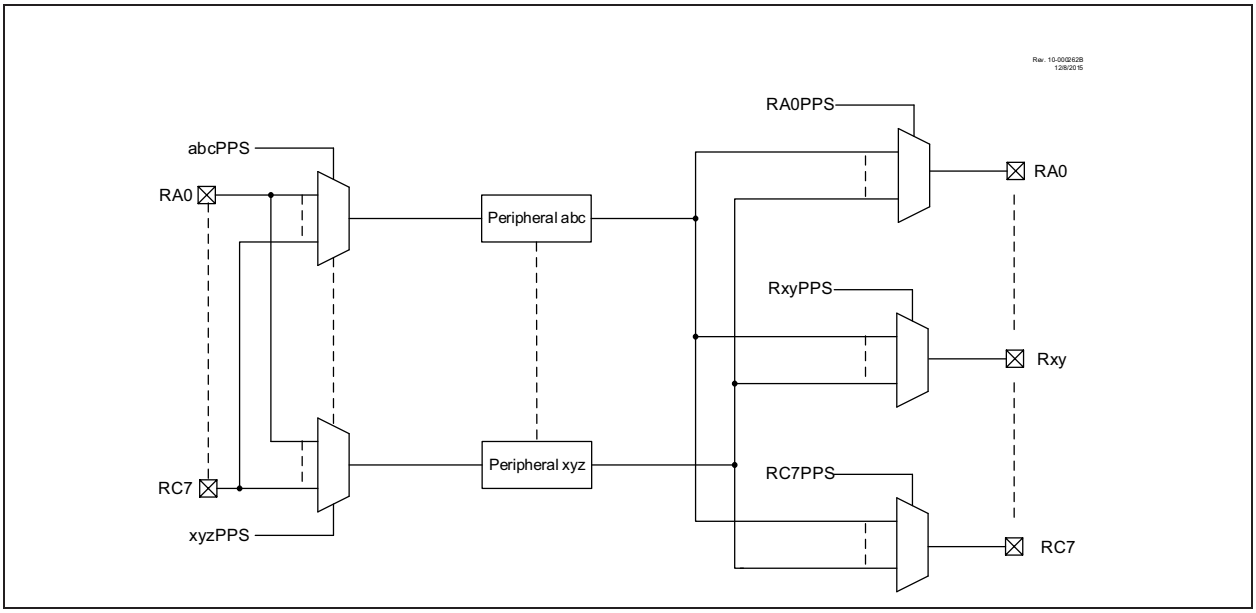
Each I/O pin has a PPS register with which the pin output source is selected. With few exceptions, the port TRIS control associated with that pin retains control over the pin output driver. Peripherals that control the pin output driver as part of the peripheral operation will override the TRIS control as needed. These peripherals include:

- UART
- I^2C

Although every pin has its own PPS peripheral selection register, the selections are identical for every pin as shown in Register 19-2.

Note:	The notation "Rxy" is a place holder for the pin identifier. For example, RA0PPS.

FIGURE 19-1: SIMPLIFIED PPS BLOCK DIAGRAM



REGISTER 21-5: PMD4: PMD CONTROL REGISTER 4

R/W-0/0	R/W-0/0	R/W-0/0	U-0	U-0	U-0	U-0	U-0
CWG3MD	CWG2MD	CWG1MD	—	—	—	—	—
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	q = Value depends on condition

bit 7 **CWG3MD:** Disable CWG3 Module bit
1 = CWG3 module disabled
0 = CWG3 module enabled

bit 6 **CWG2MD:** Disable CWG2 Module bit
1 = CWG2 module disabled
0 = CWG2 module enabled

bit 5 **CWG1MD:** Disable CWG1 Module bit
1 = CWG1 module disabled
0 = CWG1 module enabled

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

23.0 TIMER1/3/5 MODULE WITH GATE CONTROL

Timer1/3/5 module is a 16-bit timer/counter with the following features:

- 16-bit timer/counter register pair (TMRxH:TMRxL)
- Programmable internal or external clock source
- 2-bit prescaler
- Dedicated Secondary 32 kHz oscillator circuit
- Optionally synchronized comparator out
- Multiple Timer1/3/5 gate (count enable) sources
- Interrupt on overflow
- Wake-up on overflow (external clock, Asynchronous mode only)
- 16-Bit Read/Write Operation
- Time base for the Capture/Compare function with the CCP modules
- Special Event Trigger (with CCP)
- Selectable Gate Source Polarity
- Gate Toggle mode
- Gate Single-pulse mode
- Gate Value Status
- Gate Event Interrupt

Figure 23-1 is a block diagram of the Timer1/3/5 module.

FIGURE 23-1: TIMER1/3/5 BLOCK DIAGRAM

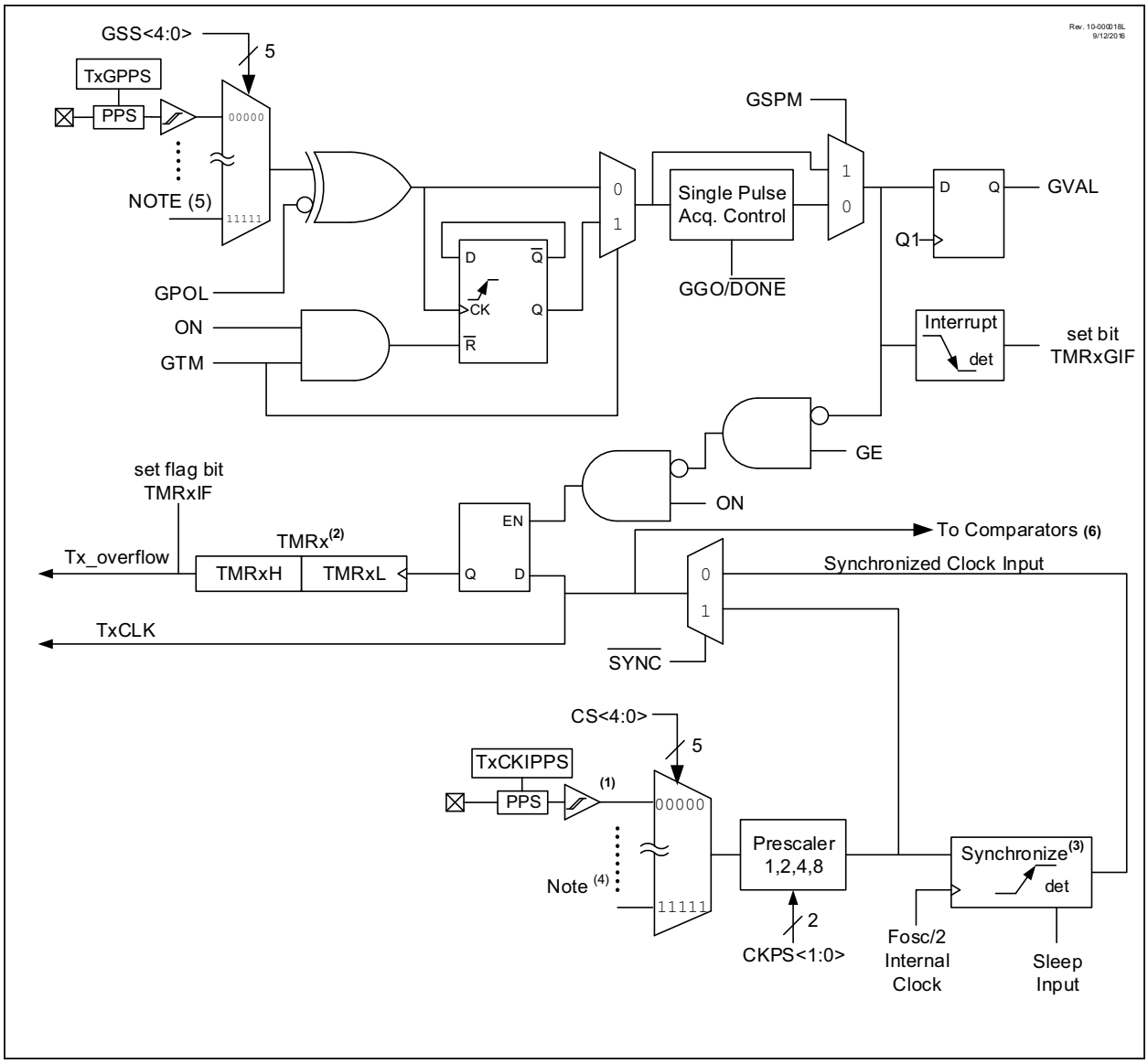
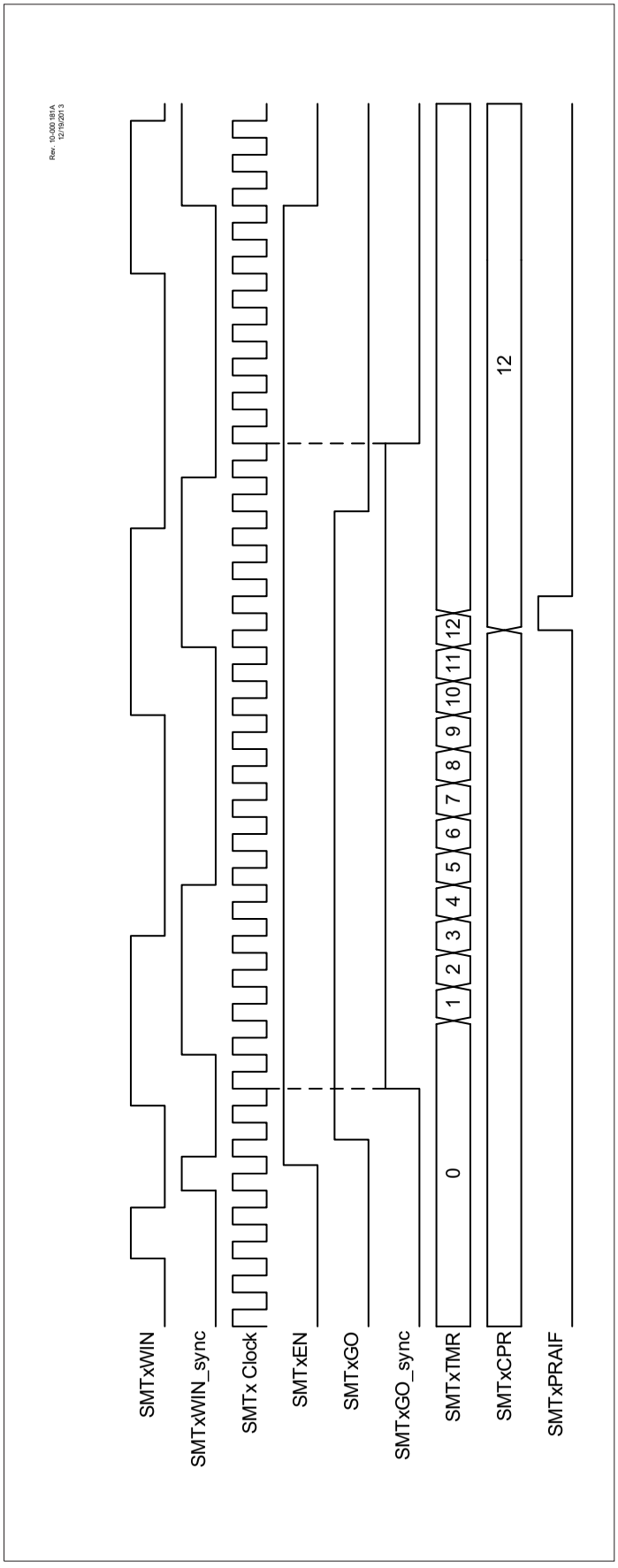


FIGURE 27-11: WINDOWED MEASURE MODE SINGLE ACQUISITION TIMING DIAGRAM



REGISTER 28-2: CWGxCON1: CWG CONTROL REGISTER 1

U-0	U-0	R-x	U-0	R/W-0/0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	IN	—	POLD	POLC	POLB	POLA
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	q = Value depends on condition

bit 7-6 **Unimplemented:** Read as '0'

bit 5 **IN:** CWG Input Value bit (read-only)

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

bit 3 **POLD:** CWGxD Output Polarity bit
1 = Signal output is inverted polarity
0 = Signal output is normal polarity

bit 2 **POLC:** CWGxC Output Polarity bit
1 = Signal output is inverted polarity
0 = Signal output is normal polarity

bit 1 **POLB:** CWGxB Output Polarity bit
1 = Signal output is inverted polarity
0 = Signal output is normal polarity

bit 0 **POLA:** CWGxA Output Polarity bit
1 = Signal output is inverted polarity
0 = Signal output is normal polarity

31.0 ZERO-CROSS DETECTION (ZCD) MODULE

The ZCD module detects when an A/C signal crosses through the ground potential. The actual zero-crossing threshold is the zero-crossing reference voltage, VCPINV, which is typically 0.75V above ground.

The connection to the signal to be detected is through a series current-limiting resistor. The module applies a current source or sink to the ZCD pin to maintain a constant voltage on the pin, thereby preventing the pin voltage from forward biasing the ESD protection diodes. When the applied voltage is greater than the reference voltage, the module sinks current. When the applied voltage is less than the reference voltage, the module sources current. The current source and sink action keeps the pin voltage constant over the full range of the applied voltage. The ZCD module is shown in the simplified block diagram Figure 31-2.

The ZCD module is useful when monitoring an A/C waveform for, but not limited to, the following purposes:

- A/C period measurement
- Accurate long term time measurement
- Dimmer phase delayed drive
- Low EMI cycle switching

31.1 External Resistor Selection

The ZCD module requires a current-limiting resistor in series with the external voltage source. The impedance and rating of this resistor depends on the external source peak voltage. Select a resistor value that will drop all of the peak voltage when the current through the resistor is nominally 300 µA. Refer to Equation 31-1 and Figure 31-1. Make sure that the ZCD I/O pin internal weak pull-up is disabled so it does not interfere with the current source and sink.

EQUATION 31-1: EXTERNAL RESISTOR

$$R_{SERIES} = \frac{V_{PEAK}}{3 \times 10^{-4}}$$

FIGURE 31-1: EXTERNAL VOLTAGE

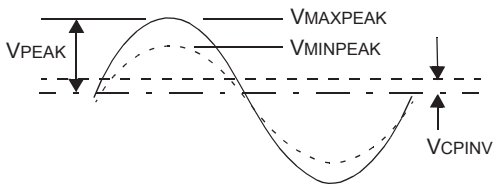
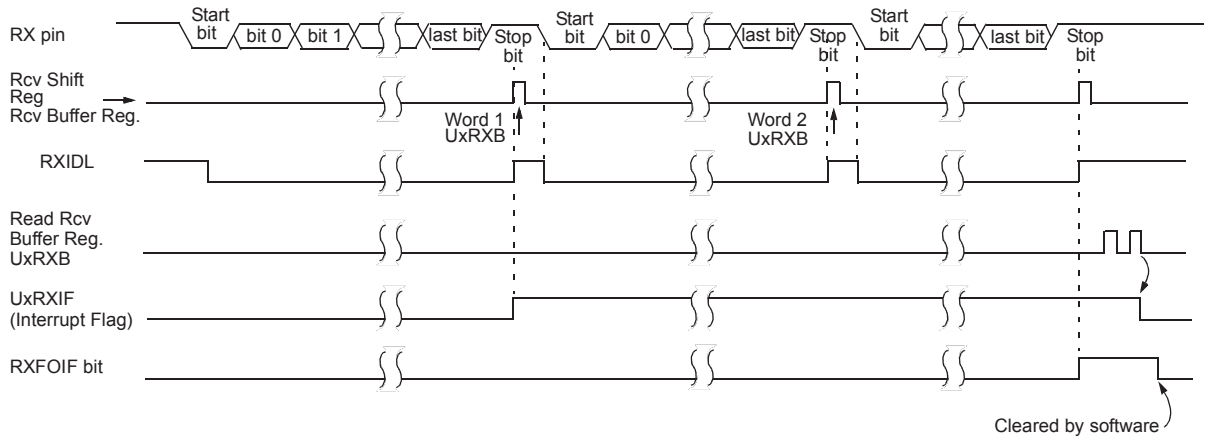


FIGURE 33-5: ASYNCHRONOUS RECEPTION



Note: This timing diagram shows three words appearing on the RX input. The UxRXB (receive buffer) is not read before the third word is received, causing the RXFOIF (FIFO overrun) bit to be set. STPMD = 0, STP<1:0> = 00.

REGISTER 38-27: ADSTPTH: ADC THRESHOLD SETPOINT REGISTER HIGH

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
STPT<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 **STPT<15:8>**: ADC Threshold Setpoint MSB. Upper byte of ADC threshold setpoint, depending on ADCALC, may be used to determine ERR, see Register 38-29 for more details.

REGISTER 38-28: ADSTPTL: ADC THRESHOLD SETPOINT REGISTER LOW

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
STPT<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 **STPT<7:0>**: ADC Threshold Setpoint LSB. Lower byte of ADC threshold setpoint, depending on ADCALC, may be used to determine ERR, see Register 38-30 for more details.

41.6 Operation During Sleep

When enabled, the HLVD circuitry continues to operate during Sleep. If the device voltage crosses the trip point, the HLVDIF bit will be set and the device will wake up from Sleep. Device execution will continue from the interrupt vector address if interrupts have been globally enabled.

41.7 Operation During Idle and Doze Modes

In both Idle and Doze modes, the module is active and events are generated if peripheral is enabled.

41.8 Operation During Freeze

When in Freeze mode, no new event or interrupt can be generated. The state of the LRDY bit is frozen.

Register reads and writes through the CPU interface are allowed.

41.9 Effects of a Reset

A device Reset forces all registers to their Reset state. This forces the HLVD module to be turned off.

TBLWT **Table Write**

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

Operands: None

Operation:
if TBLWT*,
(TABLAT) → Holding Register;
TBLPTR – No Change;
if TBLWT*+,
(TABLAT) → Holding Register;
(TBLPTR) + 1 → TBLPTR;
if TBLWT*-,
(TABLAT) → Holding Register;
(TBLPTR) – 1 → TBLPTR;
if TBLWT+*,
(TBLPTR) + 1 → TBLPTR;
(TABLAT) → Holding Register;

Status Affected: None

Encoding:

0000	0000	0000	11nn nn=0 * =1 *+ =2 *- =3 +*

Description: This instruction uses the three LSBs of TBLPTR to determine which of the eight holding registers the TABLAT is written to. The holding registers are used to program the contents of Program Memory (P.M.). (Refer to **Section 15.1 "Program Flash Memory"** for additional details on programming Flash memory.)
The TBLPTR (a 21-bit pointer) points to each byte in the program memory. TBLPTR has a 2-MByte address range. The LSb of the TBLPTR selects which byte of the program memory location to access.

TBLPTR[0] = 0: Least Significant Byte of Program Memory Word
TBLPTR[0] = 1: Most Significant Byte of Program Memory Word

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

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

Words: 1

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	No operation	No operation
No operation	No operation (Read TABLAT)	No operation	No operation (Write to Holding Register)

TBLWT **Table Write (Continued)**

Example1:
```
TBLWT *+;
```
Before Instruction
TABLAT = 55h
TBLPTR = 00A356h
HOLDING REGISTER (00A356h) = FFh
After Instructions (table write completion)
TABLAT = 55h
TBLPTR = 00A357h
HOLDING REGISTER (00A356h) = 55h

Example 2:
```
TBLWT +*;
```
Before Instruction
TABLAT = 34h
TBLPTR = 01389Ah
HOLDING REGISTER (01389Ah) = FFh
HOLDING REGISTER (01389Bh) = FFh
After Instruction (table write completion)
TABLAT = 34h
TBLPTR = 01389Bh
HOLDING REGISTER (01389Ah) = FFh
HOLDING REGISTER (01389Bh) = 34h

43.2.3 BYTE-ORIENTED AND BIT-ORIENTED INSTRUCTIONS IN INDEXED LITERAL OFFSET MODE

Note: Enabling the PIC18 instruction set extension may cause legacy applications to behave erratically or fail entirely.

In addition to eight new commands in the extended set, enabling the extended instruction set also enables Indexed Literal Offset Addressing mode (**Section 4.8.1 "Indexed Addressing with Literal Offset"**). This has a significant impact on the way that many commands of the standard PIC18 instruction set are interpreted.

When the extended set is disabled, addresses embedded in opcodes are treated as literal memory locations: either as a location in the Access Bank ('a' = 0), or in a GPR bank designated by the BSR ('a' = 1). When the extended instruction set is enabled and 'a' = 0, however, a file register argument of 5Fh or less is interpreted as an offset from the pointer value in FSR2 and not as a literal address. For practical purposes, this means that all instructions that use the Access RAM bit as an argument – that is, all byte-oriented and bit-oriented instructions, or almost half of the core PIC18 instructions – may behave differently when the extended instruction set is enabled.

When the content of FSR2 is 00h, the boundaries of the Access RAM are essentially remapped to their original values. This may be useful in creating backward compatible code. If this technique is used, it may be necessary to save the value of FSR2 and restore it when moving back and forth between C and assembly routines in order to preserve the Stack Pointer. Users must also keep in mind the syntax requirements of the extended instruction set (see **Section 43.2.3.1 "Extended Instruction Syntax with Standard PIC18 Commands"**).

Although the Indexed Literal Offset Addressing mode can be very useful for dynamic stack and pointer manipulation, it can also be very annoying if a simple arithmetic operation is carried out on the wrong register. Users who are accustomed to the PIC18 programming must keep in mind that, when the extended instruction set is enabled, register addresses of 5Fh or less are used for Indexed Literal Offset Addressing.

Representative examples of typical byte-oriented and bit-oriented instructions in the Indexed Literal Offset Addressing mode are provided on the following page to show how execution is affected. The operand conditions shown in the examples are applicable to all instructions of these types.

43.2.3.1 Extended Instruction Syntax with Standard PIC18 Commands

When the extended instruction set is enabled, the file register argument, 'f', in the standard byte-oriented and bit-oriented commands is replaced with the literal offset value, 'k'. As already noted, this occurs only when 'f' is less than or equal to 5Fh. When an offset value is used, it must be indicated by square brackets ("[ ]"). As with the extended instructions, the use of brackets indicates to the compiler that the value is to be interpreted as an index or an offset. Omitting the brackets, or using a value greater than 5Fh within brackets, will generate an error in the MPASM assembler.

If the index argument is properly bracketed for Indexed Literal Offset Addressing, the Access RAM argument is never specified; it will automatically be assumed to be '0'. This is in contrast to standard operation (extended instruction set disabled) when 'a' is set on the basis of the target address. Declaring the Access RAM bit in this mode will also generate an error in the MPASM assembler.

The destination argument, 'd', functions as before.

In the latest versions of the MPASM™ assembler, language support for the extended instruction set must be explicitly invoked. This is done with either the command line option, /y, or the PE directive in the source listing.

43.2.4 CONSIDERATIONS WHEN ENABLING THE EXTENDED INSTRUCTION SET

It is important to note that the extensions to the instruction set may not be beneficial to all users. In particular, users who are not writing code that uses a software stack may not benefit from using the extensions to the instruction set.

Additionally, the Indexed Literal Offset Addressing mode may create issues with legacy applications written to the PIC18 assembler. This is because instructions in the legacy code may attempt to address registers in the Access Bank below 5Fh. Since these addresses are interpreted as literal offsets to FSR2 when the instruction set extension is enabled, the application may read or write to the wrong data addresses.

When porting an application to the PIC18(L)F2x/4xK42, it is very important to consider the type of code. A large, re-entrant application that is written in 'C' and would benefit from efficient compilation will do well when using the instruction set extensions. Legacy applications that heavily use the Access Bank will most likely not benefit from using the extended instruction set.

44.0 REGISTER SUMMARY

TABLE 44-1: REGISTER FILE SUMMARY FOR PIC18(L)F24/25K42 DEVICES

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on page
3FFFh	TOSU	—	—	—	Top of Stack Upper byte					29
3FFEh	TOSH	Top of Stack High byte								29
3FFDh	TOSL	Top of Stack Low byte								30
3FFCh	STKPTR	—	—	—	Stack Pointer					29
3FFBh	PCLATU	—	—	—	Holding Register for PC Upper byte					27
3FFAh	PCLATH	Holding Register for PC High byte								27
3FF9h	PCL	PC Low byte								27
3FF8h	TBLPTRU	—	—	Program Memory Table Pointer Upper byte						196
3FF7h	TBLPTRH	Program Memory Table Pointer High byte								196
3FF6h	TBLPTRL	Program Memory Table Pointer Low byte								196
3FF5h	TABLAT	Table Latch								196
3FF4h	PRODH	Product Register High byte								191
3FF3h	PRODL	Product Register Low byte								191
3FF2h	—	Unimplemented								
3FF1h	PCON1	—	—	—	—	—	—	$\overline{\text{MEMV}}$	—	96
3FF0h	PCON0	STKOVF	STKUNF	$\overline{\text{WDTWV}}$	$\overline{\text{RWDT}}$	$\overline{\text{RMCLR}}$	$\overline{\text{RI}}$	$\overline{\text{POR}}$	$\overline{\text{BOR}}$	95
3FEFh	INDF0	Uses contents of FSR0 to address data memory – value of FSR0 not changed								65
3FEEh	POSTINC0	Uses contents of FSR0 to address data memory – value of FSR0 post-incremented								66
3FEDh	POSTDEC0	Uses contents of FSR0 to address data memory – value of FSR0 post-decremented								66
3FECh	PREINC0	Uses contents of FSR0 to address data memory – value of FSR0 pre-incremented								66
3FEBh	PLUSW0	Uses contents of FSR0 to address data memory – value of FSR0 pre-incremented – value of FSR0 offset by W								66
3FEAh	FSR0H	—	—	Indirect Data Memory Address Pointer 0 High						66
3FE9h	FSR0L	Indirect Data Memory Address Pointer 0 Low								66
3FE8h	WREG	Working Register								
3FE7h	INDF1	Uses contents of FSR1 to address data memory – value of FSR1 not changed								66
3FE6h	POSTINC1	Uses contents of FSR1 to address data memory – value of FSR1 post-incremented								66
3FE5h	POSTDEC1	Uses contents of FSR1 to address data memory – value of FSR1 post-decremented								66
3FE4h	PREINC1	Uses contents of FSR1 to address data memory – value of FSR1 pre-incremented								66
3FE3h	PLUSW1	Uses contents of FSR1 to address data memory – value of FSR1 pre-incremented – value of FSR1 offset by W								66
3FE2h	FSR1H	—	—	Indirect Data Memory Address Pointer 1 High						66
3FE1h	FSR1L	Indirect Data Memory Address Pointer 1 Low								66
3FE0h	BSR	—	—	Bank Select Register						35
3FDFh	INDF2	Uses contents of FSR2 to address data memory – value of FSR2 not changed								66
3FDEh	POSTINC2	Uses contents of FSR2 to address data memory – value of FSR2 post-incremented								66
3FDDh	POSTDEC2	Uses contents of FSR2 to address data memory – value of FSR2 post-decremented								66
3FDCh	PREINC2	Uses contents of FSR2 to address data memory – value of FSR2 pre-incremented								66
3FDBh	PLUSW2	Uses contents of FSR2 to address data memory – value of FSR2 pre-incremented – value of FSR2 offset by W								66
3FDAh	FSR2H	—	—	Indirect Data Memory Address Pointer 2 High						66
3FD9h	FSR2L	Indirect Data Memory Address Pointer 2 Low								66
3FD8h	STATUS	—	$\overline{\text{TO}}$	$\overline{\text{PD}}$	N	OV	Z	DC	C	63
3FD7h	IVTBASEU	—	—	—	BASE20	BASE19	BASE18	BASE17	BASE16	171
3FD6h	IVTBASEH	BASE15	BASE14	BASE13	BASE12	BASE11	BASE10	BASE9	BASE8	171
3FD5h	IVTBASEL	BASE7	BASE6	BASE5	BASE4	BASE3	BASE2	BASE1	BASE0	171
3FD4h	IVTLOCK	—	—	—	—	—	—	—	IVTLOCKED	173
3FD3h	INTCON1	STAT		—	—	—	—	—	—	141
3FD2h	INTCON0	GIE	GIEL	IPEN	—	—	INT2EDG	INT1EDG	INT0EDG	140
3FD1h - 3FCFh	—	Unimplemented								

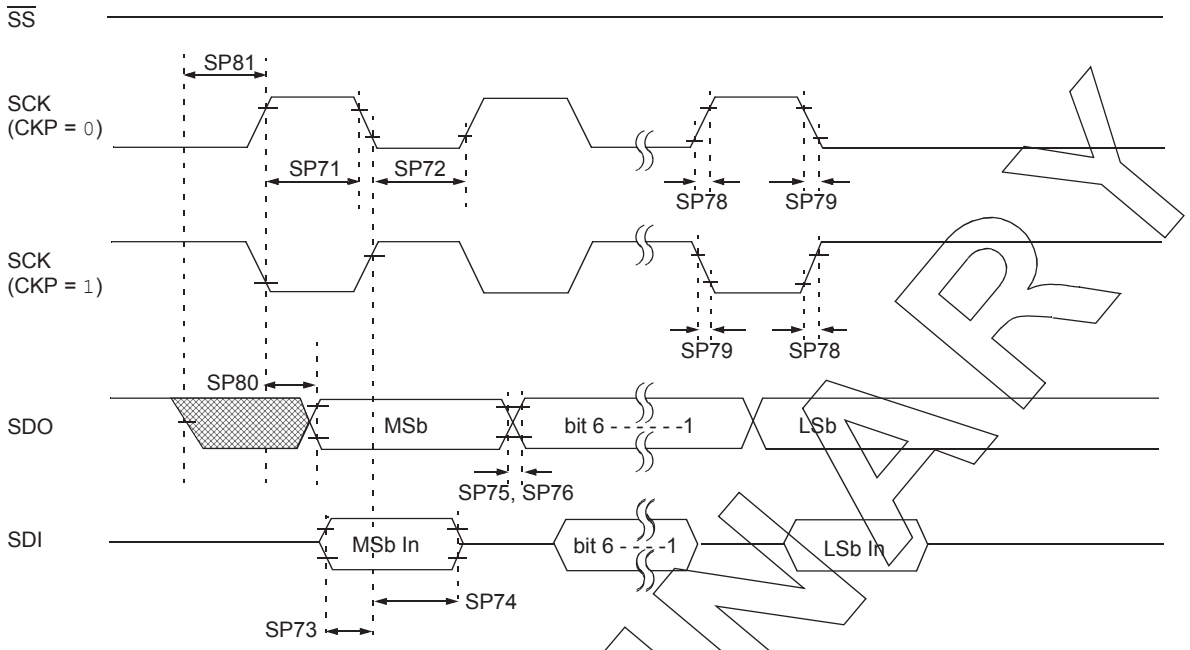
Legend: x = unknown, u = unchanged, — = unimplemented, q = value depends on condition
Note 1: Not present in LF devices.

TABLE 44-1: REGISTER FILE SUMMARY FOR PIC18(L)F24/25K42 DEVICES (CONTINUED)

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on page
3F97h	T6RST	—	—	—	RSEL					339
3F96h	T6CLK	—	—	—	—	CS				318
3F95h	T6HLT	PSYNC	CKPOL	CKSYNC	MODE					342
3F94h	T6CON	ON	CKPS		OUTPS					341
3F93h	T6PR	PR6								340
3F92h	T6TMR	TMR6								340
3F91h - 3F80h	—	Unimplemented								
3F7Fh	CCP1CAP	CTS								355
3F7Eh	CCP1CON	EN	—	OUT	FMT	MODE				353
3F7Dh	CCPR1H	RH								356
3F7Ch	CCPR1L	RL								355
3F7Bh	CCP2CAP	CTS								355
3F7Ah	CCP2CON	EN	—	OUT	FMT	MODE				353
3F79h	CCPR2H	RH								356
3F78h	CCPR2L	RL								355
3F77h	CCP3CAP	CTS								355
3F76h	CCP3CON	EN	—	OUT	FMT	MODE				353
3F75h	CCPR3H	RH								356
3F74h	CCPR3L	RL								355
3F73h	CCP4CAP	CTS								355
3F72h	CCP4CON	EN	—	OUT	FMT	MODE				353
3F71h	CCPR4H	RH								356
3F70h	CCPR4L	RL								355
3F6Fh	—	Unimplemented								
3F6Eh	PWM5CON	EN	—	OUT	POL	—	—	—	—	361
3F6Dh	PWM5DCH	DC9	DC8	DC7	DC6	DC5	DC4	DC3	DC2	363
3F6Dh	PWM5DCH	DC8								363
3F6Ch	PWM5DCL	DC1	DC0	—	—	—	—	—	—	363
3F6Ch	PWM5DCL	DC		—	—	—	—	—	—	363
3F6Bh	—	Unimplemented								
3F6Ah	PWM6CON	EN	—	OUT	POL	—	—	—	—	361
3F69h	PWM6DCH	DC9		DC7	DC6	DC5	DC4	DC3	DC2	363
3F69h	PWM6DCH	DC								363
3F68h	PWM6DCL	DC1	DC0	—	—	—	—	—	—	363
3F68h	PWM6DCL	DC		—	—	—	—	—	—	363
3F67h	—	Unimplemented								
3F66h	PWM7CON	EN	—	OUT	POL	—	—	—	—	361
3F65h	PWM7DCH	DC9	DC8	DC7	DC6	DC5	DC4	DC3	DC2	363
3F65h	PWM7DCH	DC								363
3F64h	PWM7DCL	DC1	DC0	—	—	—	—	—	—	363
3F64h	PWM7DCL	DC		—	—	—	—	—	—	363
3F63h	—	Unimplemented								
3F62h	PWM8CON	EN	—	OUT	POL	—	—	—	—	361
3F61h	PWM8DCH	DC9	DC8	DC7	DC6	DC5	DC4	DC3	DC2	363
3F61h	PWM8DCH	DC								363
3F60h	PWM8DCL	DC1	DC0	—	—	—	—	—	—	363
3F60h	PWM8DCL	DC		—	—	—	—	—	—	363
3F5Fh	CCPTMRS1	P8TSEL		P7TSEL		P6TSEL		P5TSEL		362

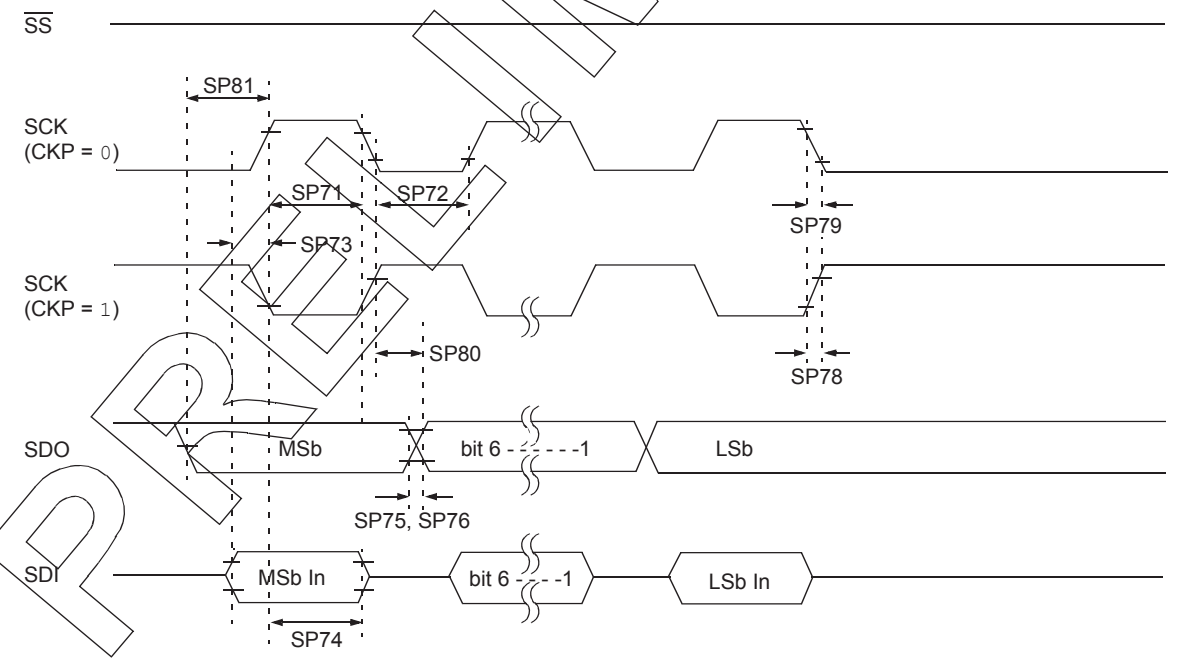
Legend: x = unknown, u = unchanged, — = unimplemented, q = value depends on condition
Note 1: Not present in LF devices.

FIGURE 46-14: SPI MASTER MODE TIMING (CKE = 0, SMP = 0)



Note: Refer to Figure 46-4 for load conditions.

FIGURE 46-15: SPI MASTER MODE TIMING (CKE = 1, SMP = 1)



Note: Refer to Figure 46-4 for load conditions.