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	64KB (32K x 16)
Program Memory Type	FLASH
EEPROM Size	1K x 8
RAM Size	4K 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-SSOP (0.209", 5.30mm Width)
Supplier Device Package	28-SSOP
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18lf26k83t-i-ss

TABLE 4-7: SPECIAL FUNCTION REGISTER MAP FOR PIC18(L)F25/26K83 DEVICES BANK 59

3BFFh	DMA1SIRQ	3BDFh	DMA2SIRQ	3BBFh	—	3B9Fh	—	3B7Fh	—	3B5Fh	—	3B3Fh	—	3B1Fh	—
3BFEh	DMA1AIRQ	3BDEh	DMA2AIRQ	3BBEh	—	3B9Eh	—	3B7Eh	—	3B5Eh	—	3B3Eh	—	3B1Eh	—
3BFDh	DMA1CON1	3BDDh	DMA2CON1	3BBDh	—	3B9Dh	—	3B7Dh	—	3B5Dh	—	3B3Dh	—	3B1Dh	—
3BFCh	DMA1CON0	3BDCh	DMA2CON0	3BBCh	—	3B9Ch	—	3B7Ch	—	3B5Ch	—	3B3Ch	—	3B1Ch	—
3BFBh	DMA1SSAU	3BDBh	DMA2SSAU	3BBBh	—	3B9Bh	—	3B7Bh	—	3B5Bh	—	3B3Bh	—	3B1Bh	—
3BFAh	DMA1SSAH	3BDAh	DMA2SSAH	3BBAh	—	3B9Ah	—	3B7Ah	—	3B5Ah	—	3B3Ah	—	3B1Ah	—
3BF9h	DMA1SSAL	3BD9h	DMA2SSAL	3BB9h	—	3B99h	—	3B79h	—	3B59h	—	3B39h	—	3B19h	—
3BF8h	DMA1SSZH	3BD8h	DMA2SSZH	3BB8h	—	3B98h	—	3B78h	—	3B58h	—	3B38h	—	3B18h	—
3BF7h	DMA1SSZL	3BD7h	DMA2SSZL	3BB7h	—	3B97h	—	3B77h	—	3B57h	—	3B37h	—	3B17h	—
3BF6h	DMA1SPTRU	3BD6h	DMA2SPTRU	3BB6h	—	3B96h	—	3B76h	—	3B56h	—	3B36h	—	3B16h	—
3BF5h	DMA1SPTRH	3BD5h	DMA2SPTRH	3BB5h	—	3B95h	—	3B75h	—	3B55h	—	3B35h	—	3B15h	—
3BF4h	DMA1SPTRL	3BD4h	DMA2SPTRL	3BB4h	—	3B94h	—	3B74h	—	3B54h	—	3B34h	—	3B14h	—
3BF3h	DMA1SCNTH	3BD3h	DMA2SCNTH	3BB3h	—	3B93h	—	3B73h	—	3B53h	—	3B33h	—	3B13h	—
3BF2h	DMA1SCNTL	3BD2h	DMA2SCNTL	3BB2h	—	3B92h	—	3B72h	—	3B52h	—	3B32h	—	3B12h	—
3BF1h	DMA1DSAH	3BD1h	DMA2DSAH	3BB1h	—	3B91h	—	3B71h	—	3B51h	—	3B31h	—	3B11h	—
3BF0h	DMA1DSAL	3BD0h	DMA2DSAL	3BB0h	—	3B90h	—	3B70h	—	3B50h	—	3B30h	—	3B10h	—
3BEFh	DMA1DSZH	3BCFh	DMA2DSZH	3BAFh	—	3B8Fh	—	3B6Fh	—	3B4Fh	—	3B2Fh	—	3B0Fh	—
3BEEh	DMA1DSZL	3BCEh	DMA2DSZL	3BAEh	—	3B8Eh	—	3B6Eh	—	3B4Eh	—	3B2Eh	—	3B0Eh	—
3BEDh	DMA1DPTRH	3BCDh	DMA2DPTRH	3BADh	—	3B8Dh	—	3B6Dh	—	3B4Dh	—	3B2Dh	—	3B0Dh	—
3BECh	DMA1DPTRL	3BCCh	DMA2DPTRL	3BACh	—	3B8Ch	—	3B6Ch	—	3B4Ch	—	3B2Ch	—	3B0Ch	—
3BEBh	DMA1DCNTH	3BCBh	DMA2DCNTH	3BABh	—	3B8Bh	—	3B6Bh	—	3B4Bh	—	3B2Bh	—	3B0Bh	—
3BEAh	DMA1DCNTL	3BCAh	DMA2DCNTL	3BAAh	—	3B8Ah	—	3B6Ah	—	3B4Ah	—	3B2Ah	—	3B0Ah	—
3BE9h	DMA1BUF	3BC9h	DMA2BUF	3BA9h	—	3B89h	—	3B69h	—	3B49h	—	3B29h	—	3B09h	—
3BE8h	—	3BC8h	—	3BA8h	—	3B88h	—	3B68h	—	3B48h	—	3B28h	—	3B08h	—
3BE7h	—	3BC7h	—	3BA7h	—	3B87h	—	3B67h	—	3B47h	—	3B27h	—	3B07h	—
3BE6h	—	3BC6h	—	3BA6h	—	3B86h	—	3B66h	—	3B46h	—	3B26h	—	3B06h	—
3BE5h	—	3BC5h	—	3BA5h	—	3B85h	—	3B65h	—	3B45h	—	3B25h	—	3B05h	—
3BE4h	—	3BC4h	—	3BA4h	—	3B84h	—	3B64h	—	3B44h	—	3B24h	—	3B04h	—
3BE3h	—	3BC3h	—	3BA3h	—	3B83h	—	3B63h	—	3B43h	—	3B23h	—	3B03h	—
3BE2h	—	3BC2h	—	3BA2h	—	3B82h	—	3B62h	—	3B42h	—	3B22h	—	3B02h	—
3BE1h	—	3BC1h	—	3BA1h	—	3B81h	—	3B61h	—	3B41h	—	3B21h	—	3B01h	—
3BE0h	—	3BC0h	—	3BA0h	—	3B80h	—	3B60h	—	3B40h	—	3B20h	—	3B00h	—

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

REGISTER 9-15: PIE2: PERIPHERAL INTERRUPT ENABLE REGISTER 2

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
I2C1RXIE	SPI1IE	SPI1TXIE	SPI1RXIE	DMA1AIE	DMA1ORIE	DMA1DCNTIE	DMA1SCNTIE
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 **I2C1RXIE:** I^2C1 Receive Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 6 **SPI1IE:** SPI1 Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 5 **SPI1TXIE:** SPI1 Transmit Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 4 **SPI1RXIE:** SPI1 Receive Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 3 **DMA1AIE:** DMA1 Abort Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 2 **DMA1ORIE:** DMA1 Overrun Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 1 **DMA1DCNTIE:** DMA1 Destination Count Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 0 **DMA1SCNTIE:** DMA1 Source Count Interrupt Enable bit
1 = Enabled
0 = Disabled

REGISTER 9-17: PIE4: PERIPHERAL INTERRUPT ENABLE REGISTER 4

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
INT1IE	CLC1IE	CWG1IE	NCO1IE	CCP1IE	TMR2IE	TMR1GIE	TMR1IE
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 **INT1IE:** External Interrupt 1 Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 6 **CLC1IE:** CLC1 Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 5 **CWG1IE:** CWG1 Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 4 **NCO1IE:** NCO1 Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 3 **CCP1IE:** CCP1 Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 2 **TMR2IE:** TMR2 Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 1 **TMR1GIE:** TMR1 Gate Interrupt Enable bit
1 = Enabled
0 = Disabled

bit 0 **TMR1IE:** TMR1 Interrupt Enable bit
1 = Enabled
0 = Disabled

REGISTER 9-30: IPR7: PERIPHERAL INTERRUPT PRIORITY REGISTER 7

R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1	R/W-1/1
U2IP	U2EIP	U2TXIP	U2RXIP	I2C2EIP	I2C2IP	I2C2TXIP	I2C2RXIP
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 **U2IP:** UART2 Interrupt Priority bit
1 = High priority
0 = Low priority

bit 6 **U2EIP:** UART2 Framing Error Interrupt Priority bit
1 = High priority
0 = Low priority

bit 5 **U2TXIP:** UART2 Transmit Interrupt Priority bit
1 = High priority
0 = Low priority

bit 4 **U2RXIP:** UART2 Receive Interrupt Priority bit
1 = High priority
0 = Low priority

bit 3 **I2C2EIP:** I^2C2 Error Interrupt Priority bit
1 = High priority
0 = Low priority

bit 2 **I2C2IP:** I^2C2 Interrupt Priority bit
1 = High priority
0 = Low priority

bit 1 **I2C2TXIP:** I^2C2 Transmit Interrupt Priority bit
1 = High priority
0 = Low priority

bit 0 **I2C2RXIP:** TMR4 Interrupt Priority bit
1 = High priority
0 = Low priority

EXAMPLE 13-4: WRITING TO PROGRAM FLASH MEMORY

```
            MOVLW   D'64'               ; number of bytes in erase block
            MOVWF   COUNTER
            MOVLW   BUFFER_ADDR_HIGH    ; point to buffer
            MOVWF   FSR0H
            MOVLW   BUFFER_ADDR_LOW
            MOVWF   FSR0L
            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
READ_BLOCK
            TBLRD*+                     ; read into TABLAT, and inc
            MOVF    TABLAT, W           ; get data
            MOVWF   POSTINC0            ; store data
            DECFSZ  COUNTER             ; done?
            BRA     READ_BLOCK          ; repeat
MODIFY_WORD
            MOVLW   BUFFER_ADDR_HIGH    ; point to buffer
            MOVWF   FSR0H
            MOVLW   BUFFER_ADDR_LOW
            MOVWF   FSR0L
            MOVLW   NEW_DATA_LOW        ; update buffer word
            MOVWF   POSTINC0
            MOVLW   NEW_DATA_HIGH
            MOVWF   INDF0
ERASE_BLOCK
            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
            BCF     NVMCON1, REG0       ; point to Program Flash Memory
            BSF     NVMCON1, REG1       ; point to Program Flash Memory
            BSF     NVMCON1, WREN       ; enable write to memory
            BSF     NVMCON1, FREE       ; enable Erase operation
            BCF     INTCON0, GIE        ; disable interrupts
            MOVLW   55h
Required    MOVWF   NVMCON2             ; write 55h
Sequence    MOVLW   AAh
            MOVWF   NVMCON2             ; write 0AAh
            BSF     NVMCON1, WR         ; start erase (CPU stall)
            BSF     INTCON0, GIE        ; re-enable interrupts
            TBLRD*-                     ; dummy read decrement
            MOVLW   BUFFER_ADDR_HIGH    ; point to buffer
            MOVWF   FSR0H
            MOVLW   BUFFER_ADDR_LOW
            MOVWF   FSR0L
WRITE_BUFFER_BACK
            MOVLW   BlockSize           ; number of bytes in holding register
            MOVWF   COUNTER
            MOVLW   D'64'/BlockSize     ; number of write blocks in 64 bytes
            MOVWF   COUNTER2
```

TABLE 14-2: SUMMARY OF REGISTERS ASSOCIATED WITH CRC

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
CRCACCH	ACC<15:8>								209
CRCACCL	ACC<7:0>								210
CRCCON0	EN	GO	BUSY	ACCM	—	—	SHIFTM	FULL	208
CRCCON1	DLEN<3:0>				PLEN<3:0>				208
CRCDATH	DATA<15:8>								209
CRCDATL	DATA<7:0>								209
CRCSHIFTH	SHIFT<15:8>								210
CRCSHIFTL	SHIFT<7:0>								210
CRCXORH	X<15:8>								211
CRCXORL	X<7:1>							—	211
SCANCON0	EN	TRIGEN	SGO	—	—	MREG	BURSTMD	BUSY	212
SCANHADRU	—	—	HADR<21:16>						214
SCANHADRH	HADR<15:8>								215
SCANHADRL	HADR<7:0>								215
SCANLADRU	—	—	LADR<21:16>						213
SCANLADRH	LADR<15:8>								213
SCANLADRL	LADR<7:0>								214
SCANTRIG	—	—	—	—	TSEL<3:0>				216

Legend: — = unimplemented location, read as '0'. Shaded cells are not used for the CRC module.

FIGURE 15-3: DMA COUNTERS BLOCK DIAGRAM

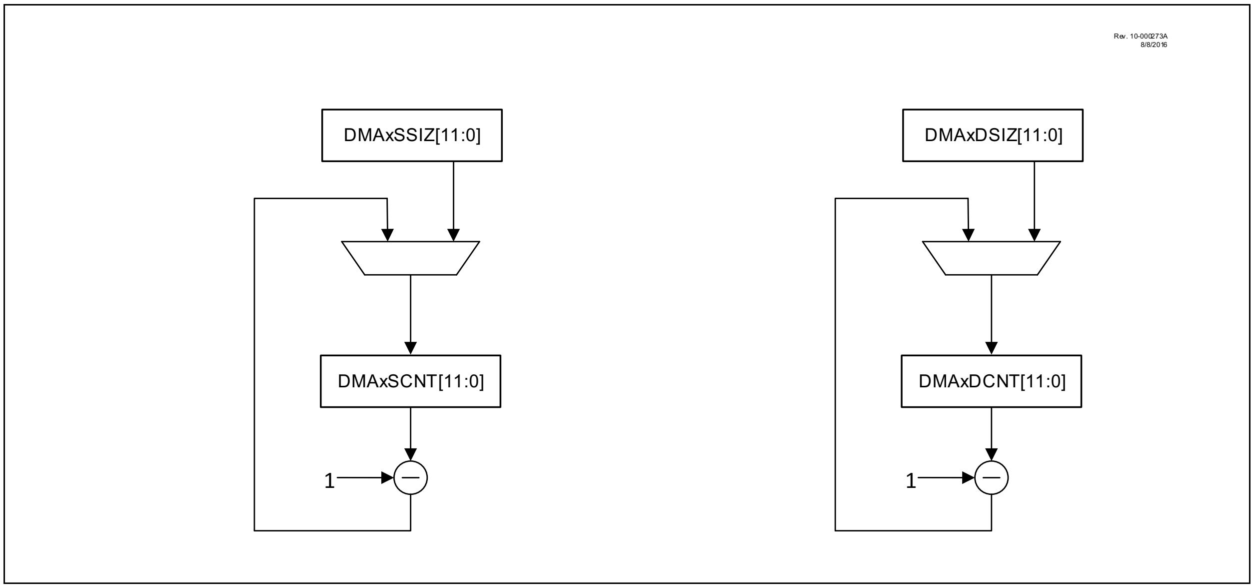


Table 15-2 has a few examples of configuring DMA Message sizes.

TABLE 15-2: EXAMPLE MESSAGE SIZE TABLE

Operation	Example	SCNT	DCNT	Comments
Read from single SFR location to RAM	U1RXB	1	N	N equals the number of bytes desired in the destination buffer. N >= 1.
Write to single SFR location from RAM	U1TXB	N	1	N equals the number of bytes desired in the source buffer. N >= 1.
Read from multiple SFR location	ADRES[H:L]	2	2*N	N equals the number of ADC results to be stored in memory. N>= 1
	TMR1[H:L]	2	2*N	N equals the number of TMR1 Acquisition results to be stored in memory. N>= 1
	SMT1CPR[U:H:L]	3	3*N	N equals the number of Capture Pulse Width measurements to be stored in memory. N>= 1
Write to Multiple SFR registers	PWMDC[H:L]	2*N	2	N equals the number of PWM duty cycle values to be loaded from a memory table. N>= 1
	All ADC registers	N*31	31	Using the DMA to transfer a complete ADC context from RAM to the ADC registers.N>= 1

REGISTER 19-6: PMD5: PMD CONTROL REGISTER 5

U-0	U-0	R/W-0/0	R/W-0/0	U-0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	U2MD	U1MD	—	SPI1MD	I2C2MD	I2C1MD
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 **U2MD:** Disable UART2 bit
1 = UART2 module disabled
0 = UART2 module enabled

bit 4 **U1MD:** Disable UART1 bit
1 = UART1 module disabled
0 = UART1 module enabled

bit 3 **Unimplemented:** Read as ‘0’

bit 2 **SPI1MD:** Disable SPI1 Module bit
1 = SPI1 module disabled
0 = SPI1 module enabled

bit 1 **I2C2MD:** Disable I^2C2 Module bit
1 = I^2C2 module disabled
0 = I^2C2 module enabled

bit 0 **I2C1MD:** Disable I^2C1 Module bit
1 = I^2C1 module disabled
0 = I^2C1 module enabled

23.0 CAPTURE/COMPARE/PWM MODULE

The Capture/Compare/PWM module is a peripheral that allows the user to time and control different events, and to generate Pulse-Width Modulation (PWM) signals. In Capture mode, the peripheral allows the timing of the duration of an event. The Compare mode allows the user to trigger an external event when a predetermined amount of time has expired. The PWM mode can generate pulse-width modulated signals of varying frequency and duty cycle.

This family of devices contains four standard Capture/Compare/PWM modules (CCP1, CCP2, CCP3 and CCP4). Each individual CCP module can select the timer source that controls the module. Each module has an independent timer selection which can be accessed using the CxTSEL bits in the CCPTMRS0 register (Register 23-2). The default timer selection is TMR1 when using Capture/Compare mode and TMR2 when using PWM mode in the CCPx module.

Please note that the Capture/Compare mode operation is described with respect to TMR1 and the PWM mode operation is described with respect to TMR2 in the following sections.

The Capture and Compare functions are identical for all CCP modules.

Note 1: In devices with more than one CCP module, it is very important to pay close attention to the register names used. A number placed after the module acronym is used to distinguish between separate modules. For example, the CCP1CON and CCP2CON control the same operational aspects of two completely different CCP modules.

2: Throughout this section, generic references to a CCP module in any of its operating modes may be interpreted as being equally applicable to CCPx module. Register names, module signals, I/O pins, and bit names may use the generic designator 'x' to indicate the use of a numeral to distinguish a particular module, when required.

23.1 CCP Module Configuration

Each Capture/Compare/PWM module is associated with a control register (CCPxCON), a capture input selection register (CCPxCAP) and a data register (CCPRx). The data register, in turn, is comprised of two 8-bit registers: CCPRxL (low byte) and CCPRxH (high byte).

23.1.1 CCP MODULES AND TIMER RESOURCES

The CCP modules utilize Timers 1 through 6 that vary with the selected mode. Various timers are available to the CCP modules in Capture, Compare or PWM modes, as shown in Table 23-1.

TABLE 23-1: CCP MODE – TIMER RESOURCE

CCP Mode	Timer Resource
Capture	Timer1, Timer3 or Timer5
Compare	
PWM	Timer2, Timer4 or Timer6

The assignment of a particular timer to a module is determined by the timer to CCP enable bits in the CCPTMRS0 register (see Register 23-2) All of the modules may be active at once and may share the same timer resource if they are configured to operate in the same mode (Capture/Compare or PWM) at the same time.

23.1.2 OPEN-DRAIN OUTPUT OPTION

When operating in Output mode (the Compare or PWM modes), the drivers for the CCPx pins can be optionally configured as open-drain outputs. This feature allows the voltage level on the pin to be pulled to a higher level through an external pull-up resistor and allows the output to communicate with external circuits without the need for additional level shifters.

REGISTER 27-10: CLCxGLS3: GATE 3 LOGIC SELECT REGISTER

R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u	R/W-x/u
G4D4T	G4D4N	G4D3T	G4D3N	G4D2T	G4D2N	G4D1T	G4D1N
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 **G4D4T:** Gate 3 Data 4 True (non-inverted) bit
1 = CLCIN3 (true) is gated into CLCx Gate 3
0 = CLCIN3 (true) is not gated into CLCx Gate 3

bit 6 **G4D4N:** Gate 3 Data 4 Negated (inverted) bit
1 = CLCIN3 (inverted) is gated into CLCx Gate 3
0 = CLCIN3 (inverted) is not gated into CLCx Gate 3

bit 5 **G4D3T:** Gate 3 Data 3 True (non-inverted) bit
1 = CLCIN2 (true) is gated into CLCx Gate 3
0 = CLCIN2 (true) is not gated into CLCx Gate 3

bit 4 **G4D3N:** Gate 3 Data 3 Negated (inverted) bit
1 = CLCIN2 (inverted) is gated into CLCx Gate 3
0 = CLCIN2 (inverted) is not gated into CLCx Gate 3

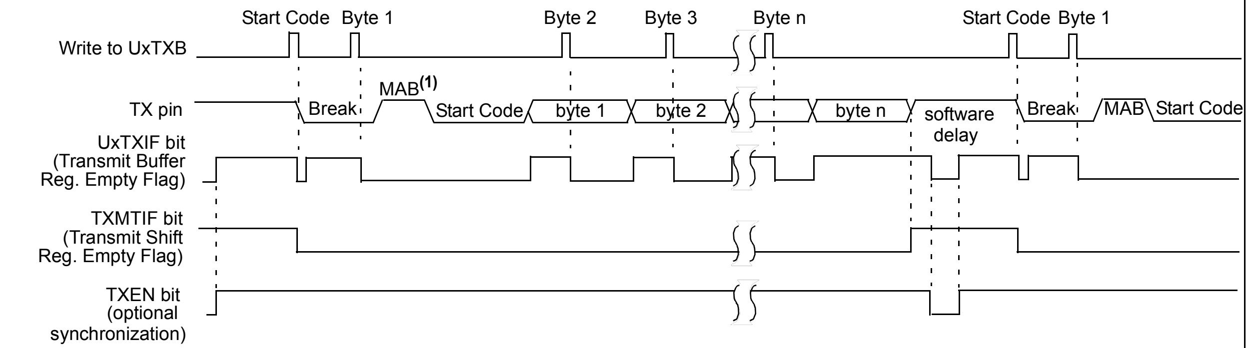
bit 3 **G4D2T:** Gate 3 Data 2 True (non-inverted) bit
1 = CLCIN1 (true) is gated into CLCx Gate 3
0 = CLCIN1 (true) is not gated into CLCx Gate 3

bit 2 **G4D2N:** Gate 3 Data 2 Negated (inverted) bit
1 = CLCIN1 (inverted) is gated into CLCx Gate 3
0 = CLCIN1 (inverted) is not gated into CLCx Gate 3

bit 1 **G4D1T:** Gate 4 Data 1 True (non-inverted) bit
1 = CLCIN0 (true) is gated into CLCx Gate 3
0 = CLCIN0 (true) is not gated into CLCx Gate 3

bit 0 **G4D1N:** Gate 3 Data 1 Negated (inverted) bit
1 = CLCIN0 (inverted) is gated into CLCx Gate 3
0 = CLCIN0 (inverted) is not gated into CLCx Gate 3

FIGURE 31-6: DMX TRANSMIT SEQUENCE



Note 1: The MAB period is fixed at 3-bits period.

31.5 LIN Modes

LIN is a protocol used primarily in automotive applications. The LIN network consists of two kinds of software processes: a Master process and a Slave process. Each network has only one Master process and one or more Slave processes.

From a physical layer point of view, the UART on one processor may be driven by both a Master and a Slave process, as long as only one Master process exists on the network.

A LIN transaction consists of a Master process followed by a Slave process. The Slave process may involve more than one slave where one is transmitting and the other(s) are receiving. The transaction begins by the following Master process transmission sequence:

1. Break
2. Delimiter bit
3. Sync Field
4. PID byte

The PID determines which Slave processes are expected to respond to the Master. When the PID byte is complete, the TX output remains in the Idle state. One or more of the Slave processes may respond to the Master process. If no one responds within the inter-byte period, the Master is free to start another transmission. The inter-byte period is timed by software using a means other than the UART.

The Slave process follows the Master process. When the slave software recognizes the PID then that Slave process responds by either transmitting the required response or by receiving the transmitted data. Only Slave processes send data. Therefore, Slave processes receiving data are receiving that of another Slave process.

When a slave sends data, the slave UART automatically calculates the checksum for the transmitted bytes as they are sent and appends the inverted checksum byte to the slave response.

When a slave receives data, the checksum is accumulated on each byte as it is received using the same algorithm as the sending process. The last byte, which is the inverted checksum value calculated by the sending process, is added to the locally calculated checksum by the UART. The check passes when the result is all '1's, otherwise the check fails and the CERIF bit is set.

Two methods for computing the checksum are available: legacy and enhanced. The legacy checksum includes only the data bytes. The enhanced checksum includes the PID and the data. The C0EN control bit in the UxCON2 register determines the checksum method. Setting C0EN to '1' selects the enhanced method. Software must select the appropriate method before the Start bit of the checksum byte is received.

31.5.1 LIN MASTER/SLAVE MODE

The LIN Master mode includes capabilities to generate Slave processes. The Master process stops at the PID transmission. Any data that is transmitted in Master/Slave mode is done as a Slave process. LIN Master/Slave mode is configured by the following settings:

- MODE<3:0> = 1100
- TXEN = 1
- RXEN = 1
- UxBRGH:L = Value to achieve desired baud rate
- TXPOL = 0 (for high Idle state)
- STP = desired Stop bits selection
- C0EN = desired checksum mode
- RxyPPS = TX pin selection code
- TX pin TRIS control = 0
- ON = 1

Note:	The TXEN bit must be set before the Master process is received and remain set while in LIN mode whether or not the Slave process is a transmitter.

32.8.3.1 Shift Register Empty Interrupt

The Shift Register Empty interrupt flag and enable are the SRMTIF and SRMTIE bits, respectively. This interrupt is only available in Master mode and triggers when a data transfer completes and conditions are not present to start a new transfer, as dictated by the TXR and RXR bits (see Table 32-1 for conditions for starting a new Master mode data transfer with different TXR/ RXR settings). This interrupt will be triggered at the end of the last full bit period, after SCK has been low for one ½-baud period. See Figure 32-14 for more details of the timing of this interrupt as well as other interrupts. This bit will not clear itself when the conditions for starting a new transfer occur, and must be cleared in software.

32.8.3.2 Transfer Counter is Zero Interrupt

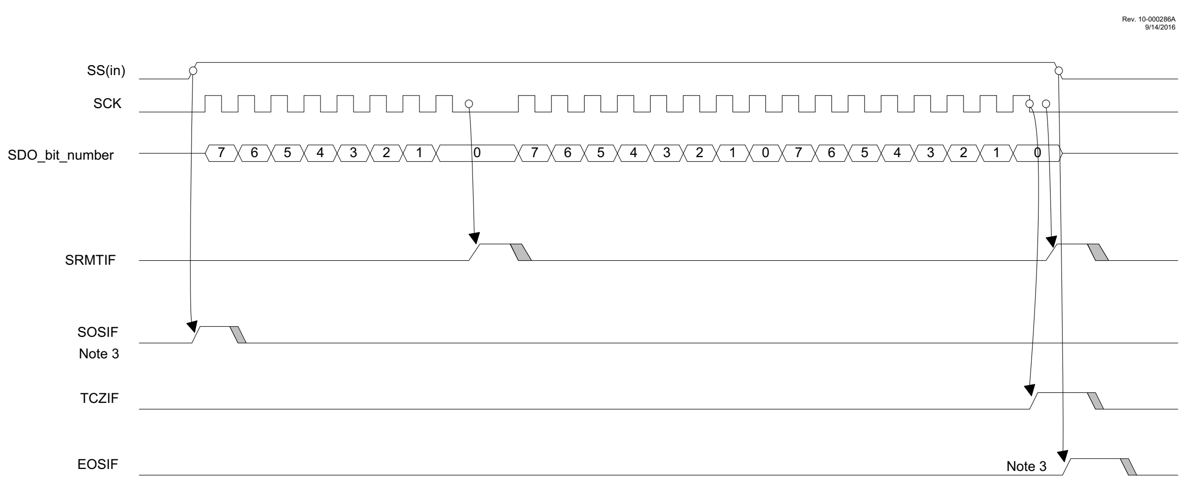
The transfer counter is zero interrupt flag and enable are the TCZIF and TCZIE bits, respectively. This interrupt will trigger when the transfer counter (defined by BMODE, SPIxTCTH/L and SPIxTWIDTH) decrements from one to zero. See Figure 32-14 for more details on the timing of this interrupt as well as other interrupts. This bit must be cleared in software.

Note:	The TCZIF flag only indicates that the transfer counter has decremented from one to zero, and may not indicate that the entire data transfer process is complete. Either poll the BUSY bit of SPIxCON2 and wait for it to be cleared or use the Shift Register Empty Interrupt (SRMTIF) to determine if a data transfer is fully complete.

32.8.3.3 Start of Slave Select and End of Slave Select Interrupts

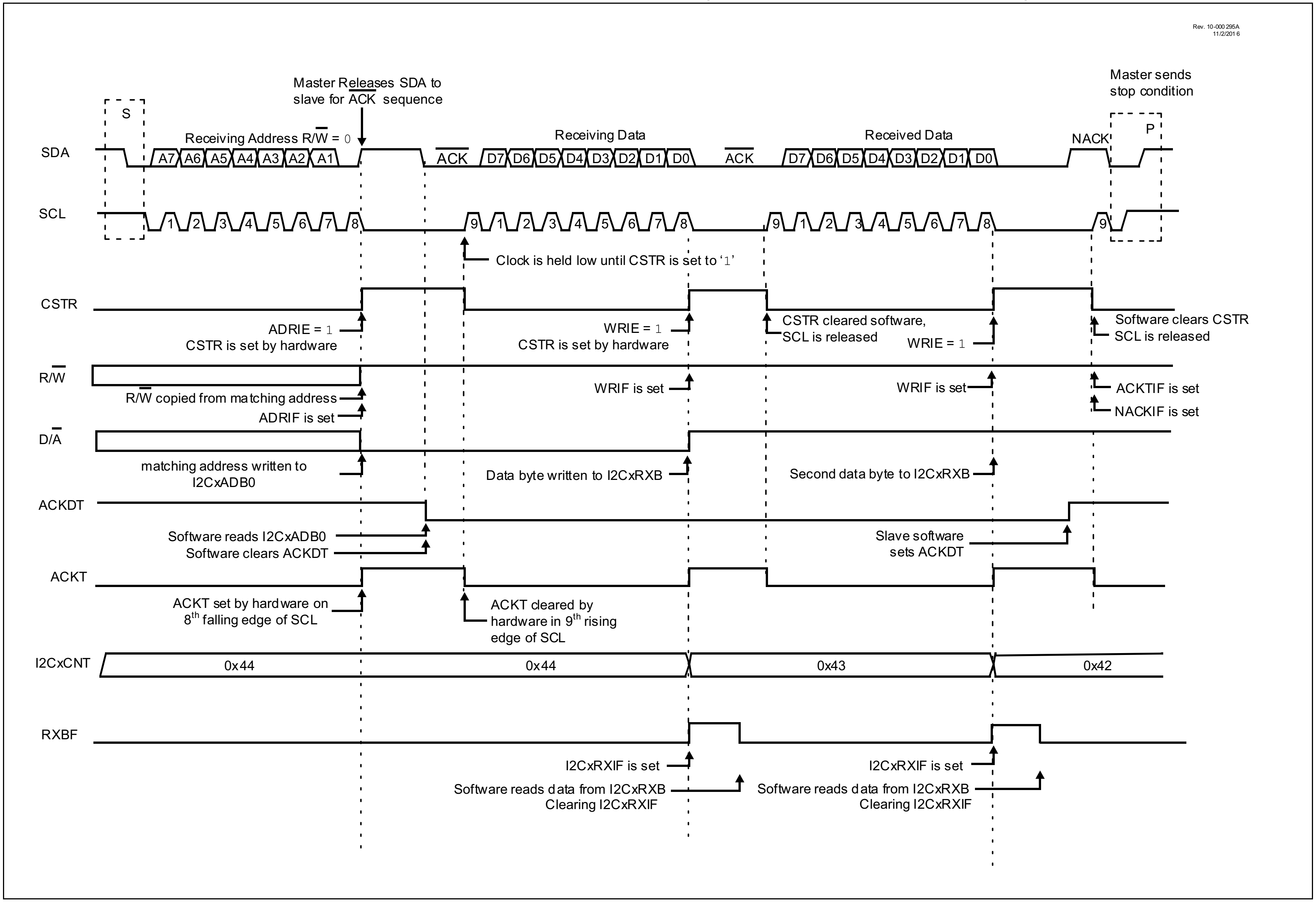
The start of Slave Select interrupt flag and enable are the SOSIF and SOSIE bits, respectively, and the end of Slave Select interrupt flag and enable are similarly designated by the EOSIF and EOSIE bits. These interrupts trigger at the leading and trailing edges of the Slave Select input. Note that the interrupts are active in both master and Slave mode, and will trigger on transitions of the Slave Select input regardless of which mode the SPI is in. In Master mode, PPS should be used to route the Slave Select input to the same pin as the Slave Select output, allowing these interrupts to trigger on changes to the Slave Select output. Also note that in Slave mode, changing the SSET bit can trigger these interrupts, as it changes the effective input value of Slave Select. Both SOSIF and EOSIF must be cleared in software

FIGURE 32-14: TRANSFER AND SLAVE SELECT INTERRUPT TIMINGS



Note 1: SRMTIF available only in Master mode
2: Clearing of interrupt flags is shown for illustration; actual interrupt flags must be cleared in software
3: SOSIF and EOSIF are set according to SS(in), even in Master mode.

FIGURE 33-8: I2C SLAVE, 7-BIT ADDRESS, RECEPTION NO I2CxCNT (ACKTIE = 0, ADRIE = 1, WRIE = 1)



Preliminary

FIGURE 33-10: I²C SLAVE, 7-BIT ADDRESS, TRANSMISSION (NO CLOCK STRETCHING)

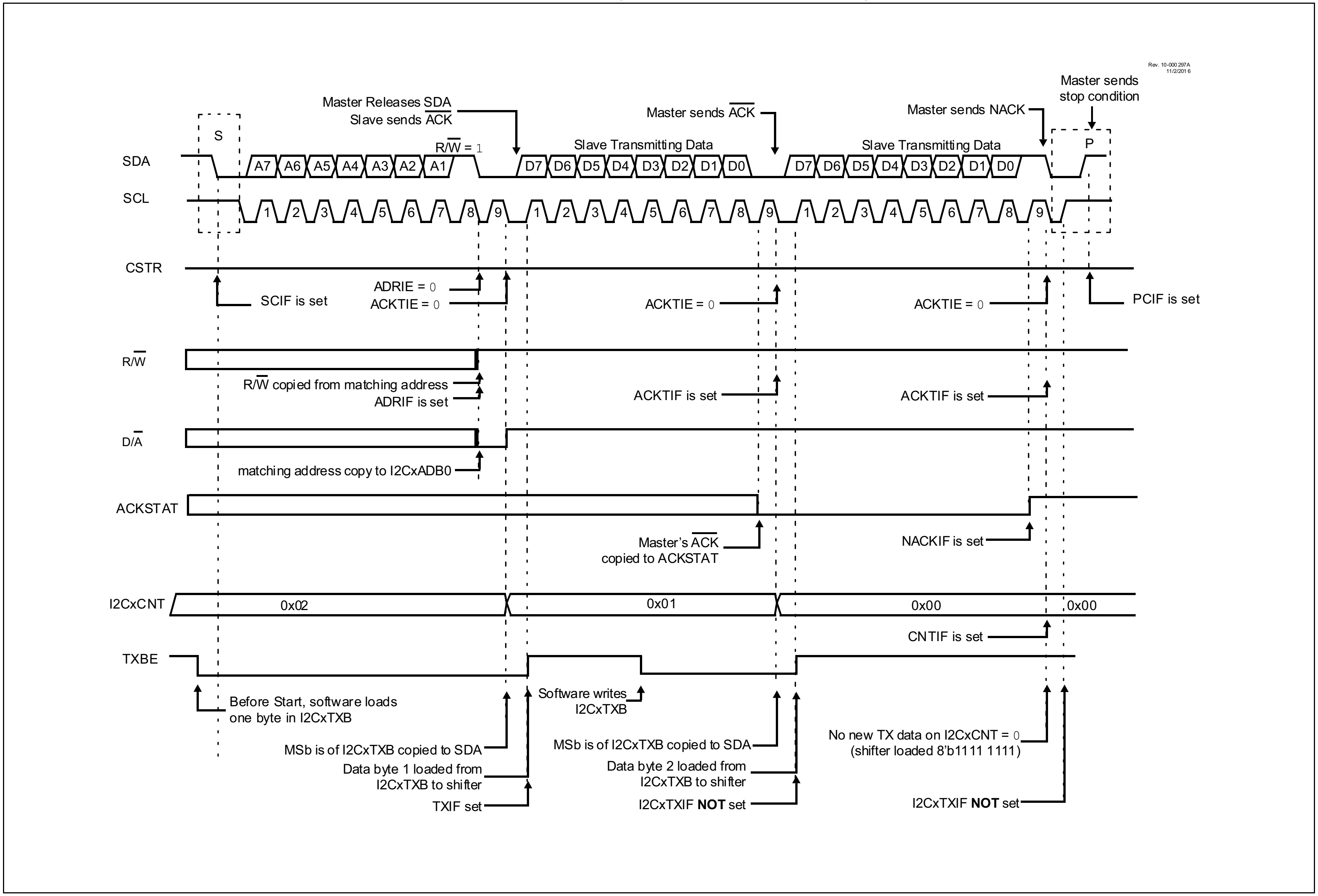
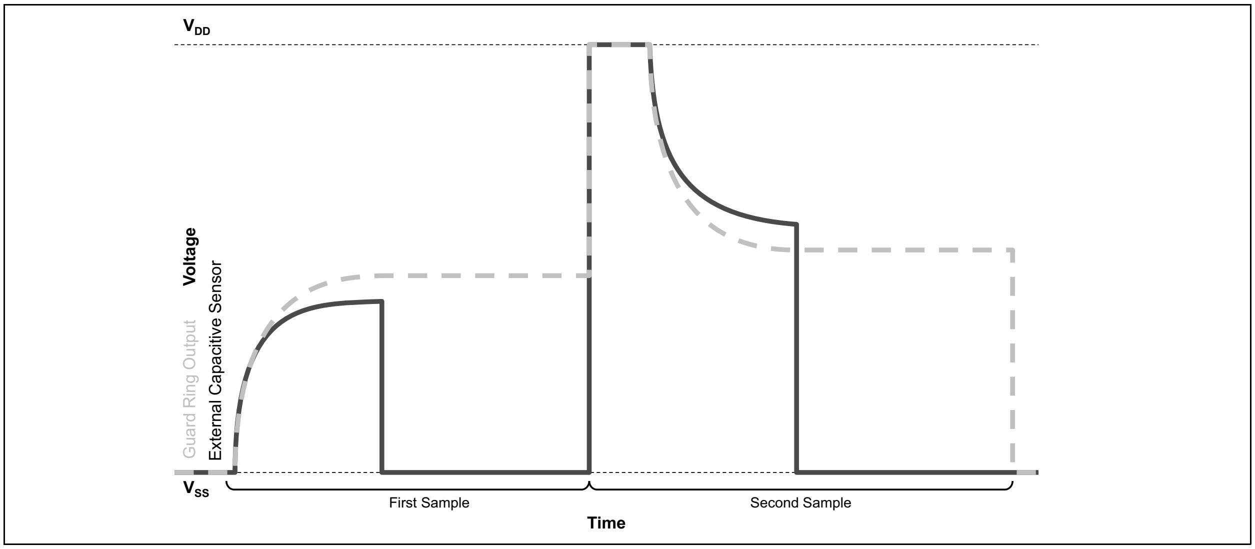


FIGURE 37-9: DIFFERENTIAL CVD WITH GUARD RING OUTPUT WAVEFORM



37.5.5 ADDITIONAL SAMPLE AND HOLD CAPACITANCE

Additional capacitance can be added in parallel with the internal sample and hold capacitor (CHOLD) by using the ADCAP register. This register selects a digitally programmable capacitance which is added to the ADC conversion bus, increasing the effective internal capacitance of the sample and hold capacitor in the ADC module. This is used to improve the match between internal and external capacitance for a better sensing performance. The additional capacitance does not affect analog performance of the ADC because it is not connected during conversion. See Figure 37-10.

REGISTER 39-2: CMxCON1: COMPARATOR x CONTROL REGISTER 1

U-0	U-0	U-0	U-0	U-0	U-0	R/W-0/0	R/W-0/0
—	—	—	—	—	—	INTP	INTN
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 7-2 Unimplemented: Read as ‘0’

bit 1 **INTP**: Comparator Interrupt on Positive-Going Edge Enable bit
1 = The CxIF interrupt flag will be set upon a positive-going edge of the CxOUT bit
0 = No interrupt flag will be set on a positive-going edge of the CxOUT bit

bit 0 **INTN**: Comparator Interrupt on Negative-Going Edge Enable bit
1 = The CxIF interrupt flag will be set upon a negative-going edge of the CxOUT bit
0 = No interrupt flag will be set on a negative-going edge of the CxOUT bit

REGISTER 39-3: CMxNCH: COMPARATOR x INVERTING CHANNEL SELECT REGISTER

U-0	U-0	U-0	U-0	U-0	R/W-0/0	R/W-0/0	R/W-0/0
—	—	—	—	—	NCH<2:0>		
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 7-3 **Unimplemented:** Read as ‘0’

bit 2-0 **NCH<2:0>:** Comparator Inverting Input Channel Select bits
111 = VSS
110 = FVR_Buffer2
101 = NCH not connected
100 = NCH not connected
011 = CxIN3-
010 = CxIN2-
001 = CxIN1-
000 = CxIN0-

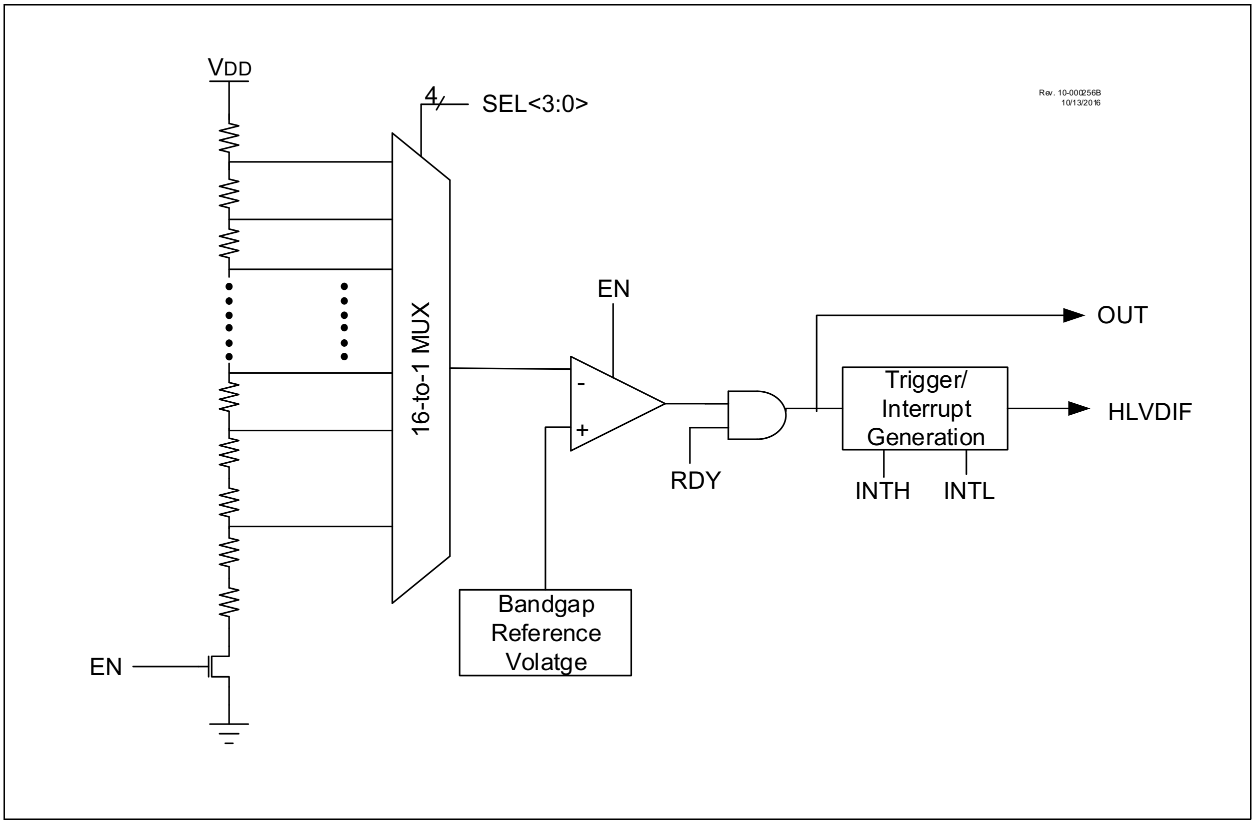
40.1 Operation

When the HLVD module is enabled, a comparator uses an internally generated voltage reference as the set point. The set point is compared with the trip point, where each node in the resistor divider represents a trip point voltage. The “trip point” voltage is the voltage level at which the device detects a high or low-voltage event, depending on the configuration of the module.

When the supply voltage is equal to the trip point, the voltage tapped off of the resistor array is equal to the internal reference voltage generated by the voltage reference module. The comparator then generates an interrupt signal by setting the HLVDIF bit.

The trip point voltage is software programmable to any of SEL<3:0> bits (HLVDCON1<3:0>).

FIGURE 40-1: HLVD MODULE BLOCK DIAGRAM



40.6 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.

40.7 Operation During Idle and Doze Modes

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

40.8 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.

40.9 Effects of a Reset

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

42.2 Extended Instruction Set

In addition to the standard 75 instructions of the PIC18 instruction set, PIC18(L)F25/26K83 devices also provide an optional extension to the core CPU functionality. The added features include eight additional instructions that augment indirect and indexed addressing operations and the implementation of Indexed Literal Offset Addressing mode for many of the standard PIC18 instructions.

The additional features of the extended instruction set are disabled by default. To enable them, users must set the XINST Configuration bit.

The instructions in the extended set can all be classified as literal operations, which either manipulate the File Select Registers, or use them for indexed addressing. Two of the instructions, `ADDFSR` and `SUBFSR`, each have an additional special instantiation for using FSR2. These versions (`ADDULNK` and `SUBULNK`) allow for automatic return after execution.

The extended instructions are specifically implemented to optimize re-entrant program code (that is, code that is recursive or that uses a software stack) written in high-level languages, particularly C. Among other things, they allow users working in high-level languages to perform certain operations on data structures more efficiently. These include:

- dynamic allocation and deallocation of software stack space when entering and leaving subroutines
- function pointer invocation
- software Stack Pointer manipulation
- manipulation of variables located in a software stack

A summary of the instructions in the extended instruction set is provided in Table 42-3. Detailed descriptions are provided in **Section 42.2.2 “Extended Instruction Set”**. The opcode field descriptions in Table 42-1 apply to both the standard and extended PIC18 instruction sets.

Note:	The instruction set extension and the Indexed Literal Offset Addressing mode were designed for optimizing applications written in C; the user may likely never use these instructions directly in assembler. The syntax for these commands is provided as a reference for users who may be reviewing code that has been generated by a compiler.

42.2.1 EXTENDED INSTRUCTION SYNTAX

Most of the extended instructions use indexed arguments, using one of the File Select Registers and some offset to specify a source or destination register. When an argument for an instruction serves as part of indexed addressing, it is enclosed in square brackets (“[ ]”). This is done to indicate that the argument is used as an index or offset. MPASM™ Assembler will flag an error if it determines that an index or offset value is not bracketed.

When the extended instruction set is enabled, brackets are also used to indicate index arguments in byte-oriented and bit-oriented instructions. This is in addition to other changes in their syntax. For more details, see **Section 42.2.3.1 “Extended Instruction Syntax with Standard PIC18 Commands”**.

Note:	In the past, square brackets have been used to denote optional arguments in the PIC18 and earlier instruction sets. In this text and going forward, optional arguments are denoted by braces (“{ }”).

TABLE 45-4: I/O PORTS

Standard Operating Conditions (unless otherwise stated)							
Param No.	**Sym.**	**Characteristic**	**Min.**	**Typ†**	**Max.**	**Units**	**Conditions**
	V_{IL}	**Input Low Voltage**					
		I/O PORT:					
D300		with TTL buffer	—	—	0.8	V	$4.5V \le V_{DD} \le 5.5V$
D301			—	—	0.15 V_{DD}	V	$1.8V \le V_{DD} \le 4.5V$
D302		with Schmitt Trigger buffer	—	—	0.2 V_{DD}	V	$2.0V \le V_{DD} \le 5.5V$
D303		with I^2C levels	—	—	0.3 V_{DD}	V	
D304		with SMBus 2.0	—	—	0.8	V	$2.7V \le V_{DD} \le 5.5V$
D305		with SMBus 3.0	—	—	0.8	V	$1.8V \le V_{DD} \le 5.5V$
D306		MCLR	—	—	0.2 V_{DD}	V	
	V_{IH}	**Input High Voltage**					
		I/O PORT:					
D320		with TTL buffer	2.0	—	—	V	$4.5V \le V_{DD} \le 5.5V$
D321			0.25 V_{DD} + 0.8	—	—	V	$1.8V \le V_{DD} \le 4.5V$
D322		with Schmitt Trigger buffer	0.8 V_{DD}	—	—	V	$2.0V \le V_{DD} \le 5.5V$
D323		with I^2C levels	0.7 V_{DD}	—	—	V	
D324		with SMBus 2.0	2.1	—	—	V	$2.7V \le V_{DD} \le 5.5V$
D325		with SMBus 3.0	1.35	—	—	V	$1.8V \le V_{DD} \le 5.5V$
D326		MCLR	0.7 V_{DD}	—	—	V	
	I_{IL}	**Input Leakage Current**(1)					
D340		I/O Ports	—	± 5	± 125	nA	$V_{SS} \le V_{PIN} \le V_{DD}$, Pin at high-impedance, 85°C
D341			—	± 5	± 1000	nA	$V_{SS} \le V_{PIN} \le V_{DD}$, Pin at high-impedance, 125°C
D342		MCLR(2)	—	± 50	± 200	nA	$V_{SS} \le V_{PIN} \le V_{DD}$, Pin at high-impedance, 85°C
	I_{PUR}	**Weak Pull-up Current**					
D350			25	120	200	μA	$V_{DD} = 3.0V$, $V_{PIN} = V_{SS}$
	V_{OL}	**Output Low Voltage**					
D360		I/O ports	—	—	0.6	V	$I_{OL} = 10.0mA$, $V_{DD} = 3.0V$
	V_{OH}	**Output High Voltage**					
D370		I/O ports	V_{DD} - 0.7	—	—	V	$I_{OH} = 6.0$ mA, $V_{DD} = 3.0V$
D380	C_{IO}	**All I/O pins**	—	5	50	pF	

† Data in “Typ” column is at 3.0V, 25°C unless otherwise stated. These parameters are for design guidance only and are not tested.

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

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