Welcome to E-XFL.COM

What is "Embedded - Microcontrollers"?

"Embedded - Microcontrollers" refer to small, integrated circuits designed to perform specific tasks within larger systems. These microcontrollers are essentially compact computers on a single chip, containing a processor core, memory, and programmable input/output peripherals. They are called "embedded" because they are embedded within electronic devices to control various functions, rather than serving as standalone computers. Microcontrollers are crucial in modern electronics, providing the intelligence and control needed for a wide range of applications.

Applications of "Embedded - Microcontrollers"

Details	
Product Status	Obsolete
Core Processor	PIC
Core Size	8-Bit
Speed	40MHz
Connectivity	CANbus, EBI/EMI, I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, LVD, POR, PWM, WDT
Number of I/O	68
Program Memory Size	48KB (24K x 16)
Program Memory Type	FLASH
EEPROM Size	1K x 8
RAM Size	3.25K x 8
Voltage - Supply (Vcc/Vdd)	2V ~ 5.5V
Data Converters	A/D 12x10b
Oscillator Type	External
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Surface Mount
Package / Case	80-TQFP
Supplier Device Package	80-TQFP (12x12)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18lf8585t-i-pt

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

Register	Applicable Devices		Power-on Reset, Brown-out Reset	MCLR Resets WDT Reset RESET Instruction Stack Resets	Wake-up via WDT or Interrupt
RXFCON0(7)	PIC18F6X8X	PIC18F8X8X	0000 0000	0000 0000	uuuu uuuu
RXFBCON7(7)	PIC18F6X8X	PIC18F8X8X	0000 0000	0000 0000	uuuu uuuu
RXFBCON6(7)	PIC18F6X8X	PIC18F8X8X	0000 0000	0000 0000	uuuu uuuu
RXFBCON5(7)	PIC18F6X8X	PIC18F8X8X	0000 0000	0000 0000	uuuu uuuu
RXFBCON4(7)	PIC18F6X8X	PIC18F8X8X	0000 0000	0000 0000	uuuu uuuu
RXFBCON3(7)	PIC18F6X8X	PIC18F8X8X	0000 0000	0000 0000	uuuu uuuu
RXFBCON2(7)	PIC18F6X8X	PIC18F8X8X	0001 0001	0001 0001	uuuu uuuu
RXFBCON1(7)	PIC18F6X8X	PIC18F8X8X	0001 0001	0001 0001	uuuu uuuu
RXFBCON0(7)	PIC18F6X8X	PIC18F8X8X	0000 0000	0000 0000	uuuu uuuu
RXF15EIDL(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF15EIDH(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF15SIDL(7)	PIC18F6X8X	PIC18F8X8X	xxx- x-xx	uuu- u-uu	uuu- u-uu
RXF15SIDH(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF14EIDL(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF14EIDH(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF14SIDL(7)	PIC18F6X8X	PIC18F8X8X	xxx- x-xx	uuu- u-uu	uuu- u-uu
RXF14SIDH(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF13EIDL(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF13EIDH(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF13SIDL(7)	PIC18F6X8X	PIC18F8X8X	xxx- x-xx	uuu- u-uu	uuu- u-uu
RXF13SIDH(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF12EIDL(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF12EIDH(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF12SIDL(7)	PIC18F6X8X	PIC18F8X8X	xxx- x-xx	uuu- u-uu	uuu- u-uu
RXF12SIDH(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF11EIDL(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF11EIDH(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF11SIDL(7)	PIC18F6X8X	PIC18F8X8X	xxx- x-xx	uuu- u-uu	uuu- u-uu
RXF11SIDH(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
RXF10EIDL(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	-uuu uuuu
RXF10EIDH(7)	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	-uuu uuuu

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

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

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

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

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

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

6: Bit 6 of PORTA, LATA and TRISA are not available on all devices. When unimplemented, they read '0'.

7: This register reads all '0's until ECAN is set up in Mode 1 or Mode 2.

4.6 Instruction Flow/Pipelining

An "Instruction Cycle" consists of four Q cycles (Q1, Q2, Q3 and Q4). The instruction fetch and execute are pipelined such that fetch takes one instruction cycle, while decode and execute takes another instruction cycle. However, due to the pipelining each instruction effectively executes in one cycle. If an instruction causes the program counter to change (e.g., GOTO), then two cycles are required to complete the instruction (Example 4-2).

A fetch cycle begins with the program counter (PC) incrementing in Q1.

In the execution cycle, the fetched instruction is latched into the "Instruction Register" (IR) in cycle Q1. This instruction is then decoded and executed during the Q2, Q3, and Q4 cycles. Data memory is read during Q2 (operand read) and written during Q4 (destination write).

EXAMPLE 4-2: INSTRUCTION PIPELINE FLOW

	TCY0	TCY1	TCY2	TCY3	TCY4	TCY5
1. MOVLW 55h	Fetch 1	Execute 1				
2. MOVWF PORTB		Fetch 2	Execute 2			
3. BRA SUB_1			Fetch 3	Execute 3		
4. BSF PORTA, 3 (Forced NOP)				Fetch 4	Flush (NOP)	
5. Instruction @ address SUB_1					Fetch SUB_1	Execute SUB_1

All instructions are single cycle except for any program branches. These take two cycles since the fetch instruction is "flushed" from the pipeline while the new instruction is being fetched and then executed.

4.7 Instructions in Program Memory

The program memory is addressed in bytes. Instructions are stored as two bytes or four bytes in program memory. The Least Significant Byte (LSB) of an instruction word is always stored in a program memory location with an even address (LSB = 0). Figure 4-6 shows an example of how instruction words are stored in the program memory. To maintain alignment with instruction boundaries, the PC increments in steps of 2 and the LSB will always read '0' (see **Section 4.4 "PCL, PCLATH and PCLATU"**).

The CALL and GOTO instructions have an absolute program memory address embedded into the instruction. Since instructions are always stored on word boundaries, the data contained in the instruction is a word address. The word address is written to PC<20:1> which accesses the desired byte address in program memory. Instruction #2 in Figure 4-6 shows how the instruction "GOTO 000006h" is encoded in the program memory. Program branch instructions which encode a relative address offset operate in the same manner. The offset value stored in a branch instruction represents the number of single-word instructions that the PC will be offset by. **Section 25.0 "Instruction Set Summary"** provides further details of the instruction set.

FIGURE 4-6: INSTRUCTIONS IN PROGRAM MEMORY

			Program Memory Byte Locations →	LSB = 1	LSB = 0	Word Address ↓
						000000h
						000002h
						000004h
						000006h
Instruction 1:	MOVLW	055h		0Fh	55h	000008h
Instruction 2:	GOTO	000006h		0EFh	03h	00000Ah
				0F0h	00h	00000Ch
Instruction 3:	MOVFF	123h, 456h		0C1h	23h	00000Eh
				0F4h	56h	000010h
						000012h
						000014h

4.13 Status Register

The Status register, shown in Register 4-3, contains the arithmetic status of the ALU. The Status register can be the destination for any instruction as with any other register. If the Status register is the destination for an instruction that affects the Z, DC, C, OV or N bits, then the write to these five bits is disabled. These bits are set or cleared according to the device logic. Therefore, the result of an instruction with the Status register as destination may be different than intended.

For example, `CLRF STATUS` will clear the upper three bits and set the Z bit. This leaves the Status register as `000u u1uu` (where `u` = unchanged).

It is recommended, therefore, that only `BCF`, `BSF`, `SWAPF`, `MOVFF` and `MOVWF` instructions are used to alter the Status register because these instructions do not affect the Z, C, DC, OV or N bits from the Status register. For other instructions not affecting any status bits, see Table 25-2.

Note:	The C and DC bits operate as a $\overline{\text{borrow}}$ and $\overline{\text{digit borrow}}$ bit respectively, in subtraction.

REGISTER 4-3: STATUS REGISTER

U-0	U-0	U-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
—	—	—	N	OV	Z	DC	C
bit 7							bit 0

bit 7-5 **Unimplemented:** Read as '0'

bit 4 **N:** Negative bit

This bit is used for signed arithmetic (2's complement). It indicates whether the result was negative (ALU MSB = 1).

1 = Result was negative
0 = Result was positive

bit 3 **OV:** Overflow bit

This bit is used for signed arithmetic (2's complement). It indicates an overflow of the 7-bit magnitude which causes the sign bit (bit 7) to change state.

1 = Overflow occurred for signed arithmetic (in this arithmetic operation)
0 = No overflow occurred

bit 2 **Z:** Zero bit

1 = The result of an arithmetic or logic operation is zero
0 = The result of an arithmetic or logic operation is not zero

bit 1 **DC:** Digit carry/$\overline{\text{borrow}}$ bit

For `ADDWF`, `ADDLW`, `SUBLW`, and `SUBWF` instructions:

1 = A carry-out from the 4th low order bit of the result occurred
0 = No carry-out from the 4th low order bit of the result

Note: For $\overline{\text{borrow}}$, the polarity is reversed. A subtraction is executed by adding the 2's complement of the second operand. For rotate (`RRF`, `RLF`) instructions, this bit is loaded with either the bit 4 or bit 3 of the source register.

bit 0 **C:** Carry/$\overline{\text{borrow}}$ bit

For `ADDWF`, `ADDLW`, `SUBLW`, and `SUBWF` instructions:

1 = A carry-out from the Most Significant bit of the result occurred
0 = No carry-out from the Most Significant bit of the result occurred

Note: For $\overline{\text{borrow}}$, the polarity is reversed. A subtraction is executed by adding the 2's complement of the second operand. For rotate (`RRF`, `RLF`) instructions, this bit is loaded with either the high or low-order bit of the source register.

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

7.0 DATA EEPROM MEMORY

The data EEPROM is readable and writable during normal operation over the entire VDD range. The data memory is not directly mapped in the register file space. Instead, it is indirectly addressed through the Special Function Registers (SFR).

There are five SFRs used to read and write the program and data EEPROM memory. These registers are:

- EECON1
- EECON2
- EEDATA
- EEADR
- EEADRH

The EEPROM data memory allows byte read and write. When interfacing to the data memory block, EEDATA holds the 8-bit data for read/write and EEADR holds the address of the EEPROM location being accessed. These devices have 1024 bytes of data EEPROM with an address range from 0h to 3FFh.

The EEPROM data memory is rated for high erase/write cycles. A byte write automatically erases the location and writes the new data (erase-before-write). The write time is controlled by an on-chip timer. The write time will vary with voltage and temperature as well as from chip to chip. Please refer to parameter D122 (Electrical Characteristics, **Section 27.0 "Electrical Characteristics"**) for exact limits.

7.1 EEADRH:EEADR

The address register pair, EEADRH:EEADR, can address up to a maximum of 1024 bytes of data EEPROM.

7.2 EECON1 and EECON2 Registers

EECON1 is the control register for EEPROM memory accesses.

EECON2 is not a physical register. Reading EECON2 will read all '0's. The EECON2 register is used exclusively in the EEPROM write sequence.

Control bits RD and WR initiate read and write operations, respectively. These bits cannot be cleared, only set in software. They are cleared in hardware at the completion of the read or write operation. The inability to clear the WR bit in software prevents the accidental or premature termination of a write operation.

The WREN bit, when set, will allow a write operation. On power-up, the WREN bit is clear. The WRERR bit is set when a write operation is interrupted by a $\overline{\text{MCLR}}$ Reset or a WDT Time-out Reset during normal operation. In these situations, the user can check the WRERR bit and rewrite the location. It is necessary to reload the data and address registers (EEDATA and EEADR) due to the Reset condition forcing the contents of the registers to zero.

Note:	Interrupt flag bit, EEIF in the PIR2 register, is set when write is complete. It must be cleared in software.

7.5 Write Verify

Depending on the application, good programming practice may dictate that the value written to the memory should be verified against the original value. This should be used in applications where excessive writes can stress bits near the specification limit.

7.6 Protection Against Spurious Write

There are conditions when the device may not want to write to the data EEPROM memory. To protect against spurious EEPROM writes, various mechanisms have been built-in. On power-up, the WREN bit is cleared. Also, the Power-up Timer (72 ms duration) prevents EEPROM write.

The write initiate sequence and the WREN bit together help prevent an accidental write during brown-out, power glitch, or software malfunction.

7.7 Operation During Code-Protect

Data EEPROM memory has its own code-protect mechanism. External read and write operations are disabled if either of these mechanisms are enabled.

The microcontroller itself can both read and write to the internal data EEPROM regardless of the state of the code-protect configuration bit. Refer to **Section 24.0 "Special Features of the CPU"** for additional information.

7.8 Using the Data EEPROM

The data EEPROM is a high endurance, byte addressable array that has been optimized for the storage of frequently changing information (e.g., program variables or other data that are updated often). Frequently changing values will typically be updated more often than specification D124. If this is not the case, an array refresh must be performed. For this reason, variables that change infrequently (such as constants, IDs, calibration, etc.) should be stored in Flash program memory.

A simple data EEPROM refresh routine is shown in Example 7-3.

Note:	If data EEPROM is only used to store constants and/or data that changes rarely, an array refresh is likely not required. See specification D124.

EXAMPLE 7-3: DATA EEPROM REFRESH ROUTINE

```
        CLRF    EEADRH              ;
        CLRF    EEADR               ; Start at address 0
        BCF     EECON1, CFGS        ; Set for memory
        BCF     EECON1, EEPGD       ; Set for Data EEPROM
        BCF     INTCON, GIE         ; Disable interrupts
        BSF     EECON1, WREN        ; Enable writes
Loop                                ; Loop to refresh array
        BSF     EECON1, RD          ; Read current address
        MOVLW   55h                 ;
        MOVWF   EECON2              ; Write 55h
        MOVLW   0AAh                ;
        MOVWF   EECON2              ; Write 0AAh
        BSF     EECON1, WR          ; Set WR bit to begin write
        BTFSC   EECON1, WR          ; Wait for write to complete
        BRA     $-2
        INCFSZ  EEADR, F            ; Increment address
        BRA     Loop                ; Not zero, do it again

        INCFS2  EEADRH, F           ;
        BRA     Loop                ;
        BCF     EECON1, WREN        ; Disable writes
        BSF     INTCON, GIE         ; Enable interrupts
```

17.3.8 SLEEP OPERATION

In Master mode, all module clocks are halted and the transmission/reception will remain in that state until the device wakes from Sleep. After the device returns to normal mode, the module will continue to transmit/receive data.

In Slave mode, the SPI Transmit/Receive Shift register operates asynchronously to the device. This allows the device to be placed in Sleep mode and data to be shifted into the SPI Transmit/Receive Shift register. When all 8 bits have been received, the MSSP interrupt flag bit will be set and if enabled, will wake the device from Sleep.

17.3.9 EFFECTS OF A RESET

A Reset disables the MSSP module and terminates the current transfer.

17.3.10 BUS MODE COMPATIBILITY

Table 17-1 shows the compatibility between the standard SPI modes and the states of the CKP and CKE control bits.

TABLE 17-1: SPI BUS MODES

Standard SPI Mode Terminology	Control Bits State	
	CKP	CKE
0, 0	0	1
0, 1	0	0
1, 0	1	1
1, 1	1	0

There is also a SMP bit which controls when the data is sampled.

TABLE 17-2: REGISTERS ASSOCIATED WITH SPI OPERATION

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on POR, BOR	Value on all other Resets
INTCON	GIE/GIEH	PEIE/GIEL	TMR0IE	INT0IE	RBIE	TMR0IF	INT0IF	RBIF	0000 0000	0000 0000
PIR1	PSPIF	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
PIE1	PSPIE	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
IPR1	PSPIP	ADIP	RCIP	TXIP	SSPIP	CCP1IP	TMR2IP	TMR1IP	1111 1111	1111 1111
TRISC	PORTC Data Direction Register								1111 1111	1111 1111
TRISF	TRISF7	TRISF6	TRISF5	TRISF4	TRISF3	TRISF2	TRISF1	TRISF0	1111 1111	uuuu uuuu
SSPBUF	Synchronous Serial Port Receive Buffer/Transmit Register								xxxx xxxx	uuuu uuuu
SSPCON	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0	0000 0000	0000 0000
SSPSTAT	SMP	CKE	D/$\overline{A}$	P	S	R/$\overline{W}$	UA	BF	0000 0000	0000 0000

Legend: x = unknown, u = unchanged, – = unimplemented, read as '0'. Shaded cells are not used by the MSSP in SPI mode.

REGISTER 17-4: SSPCON1: MSSP CONTROL REGISTER 1 (I^2C MODE)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0
bit 7							bit 0

bit 7 **WCOL:** Write Collision Detect bit

In Master Transmit mode:

1 = A write to the SSPBUF register was attempted while the I^2C conditions were not valid for a transmission to be started (must be cleared in software)

0 = No collision

In Slave Transmit mode:

1 = The SSPBUF register is written while it is still transmitting the previous word (must be cleared in software)

0 = No collision

In Receive mode (Master or Slave modes):

This is a “don’t care” bit.

bit 6 **SSPOV:** Receive Overflow Indicator bit

In Receive mode:

1 = A byte is received while the SSPBUF register is still holding the previous byte (must be cleared in software)

0 = No overflow

In Transmit mode:

This is a “don’t care” bit in Transmit mode.

bit 5 **SSPEN:** Synchronous Serial Port Enable bit

1 = Enables the serial port and configures the SDA and SCL pins as the serial port pins

0 = Disables serial port and configures these pins as I/O port pins

Note: When enabled, the SDA and SCL pins must be properly configured as input or output.

bit 4 **CKP:** SCK Release Control bit

In Slave mode:

1 = Release clock

0 = Holds clock low (clock stretch), used to ensure data setup time

In Master mode:

Unused in this mode.

bit 3-0 **SSPM3:SSPM0:** Synchronous Serial Port Mode Select bits

1111 = I^2C Slave mode, 10-bit address with Start and Stop bit interrupts enabled

1110 = I^2C Slave mode, 7-bit address with Start and Stop bit interrupts enabled

1011 = I^2C Firmware Controlled Master mode (slave Idle)

1000 = I^2C Master mode, clock = FOSC/(4 * (SSPADD + 1))

0111 = I^2C Slave mode, 10-bit address

0110 = I^2C Slave mode, 7-bit address

Note: Bit combinations not specifically listed here are either reserved or implemented in SPI mode only.

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

17.4.7 BAUD RATE GENERATOR

In I^2C Master mode, the Baud Rate Generator (BRG) reload value is placed in the lower 7 bits of the SSPADD register (Figure 17-17). When a write occurs to SSPBUF, the Baud Rate Generator will automatically begin counting. The BRG counts down to '0' and stops until another reload has taken place. The BRG count is decremented twice per instruction cycle (T_{CY}) on the Q2 and Q4 clocks. In I^2C Master mode, the BRG is reloaded automatically.

Once the given operation is complete (i.e., transmission of the last data bit is followed by $\overline{ACK}$), the internal clock will automatically stop counting and the SCL pin will remain in its last state.

Table 17-3 demonstrates clock rates based on instruction cycles and the BRG value loaded into SSPADD.

FIGURE 17-17: BAUD RATE GENERATOR BLOCK DIAGRAM

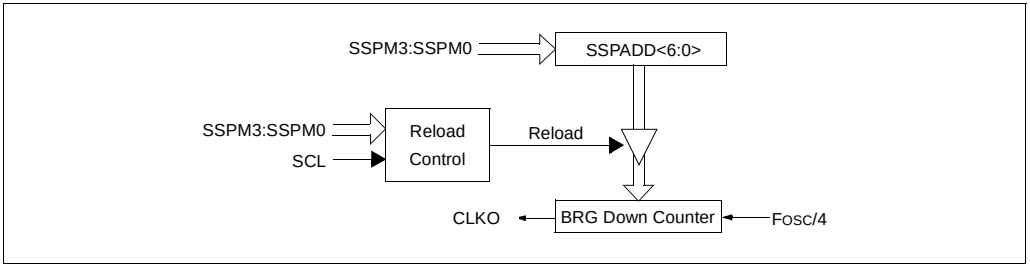


TABLE 17-3: I^2C CLOCK RATE w/BRG

FCY	FCY*2	BRG Value	FSCL (2 Rollovers of BRG)
10 MHz	20 MHz	19h	400 kHz(1)
10 MHz	20 MHz	20h	312.5 kHz
10 MHz	20 MHz	64h	100 kHz
4 MHz	8 MHz	0Ah	400 kHz(1)
4 MHz	8 MHz	0Dh	308 kHz
4 MHz	8 MHz	28h	100 kHz
1 MHz	2 MHz	03h	333 kHz(1)
1 MHz	2 MHz	0Ah	100 kHz
1 MHz	2 MHz	00h	1 MHz(1)

Note 1: The I^2C interface does not conform to the 400 kHz I^2C specification (which applies to rates greater than 100 kHz) in all details but may be used with care where higher rates are required by the application.

19.6 A/D Conversions

Figure 19-3 shows the operation of the A/D converter after the GO bit has been set and the ACQT2:ACQT0 bits are cleared. A conversion is started after the following instruction to allow entry into Sleep mode before the conversion begins.

Figure 19-4 shows the operation of the A/D converter after the GO bit has been set, the ACQT2:ACQT0 bits are set to '010' and selecting a 4 TAD acquisition time before the conversion starts.

Clearing the GO/$\overline{\text{DONE}}$ bit during a conversion will abort the current conversion. The A/D Result register pair will not be updated with the partially completed A/D conversion sample. This means the ADRESH:ADRESL registers will continue to contain the value of the last completed conversion (or the last value written to the ADRESH:ADRESL registers).

After the A/D conversion is completed or aborted, a 2 TAD wait is required before the next acquisition can be started. After this wait, acquisition on the selected channel is automatically started.

Note:	The GO/$\overline{\text{DONE}}$ bit should **NOT** be set in the same instruction that turns on the A/D.

19.7 Use of the CCP2 Trigger

An A/D conversion can be started by the "special event trigger" of the CCP2 module. This requires that the CCP2M3:CCP2M0 bits (CCP2CON<3:0>) be programmed as '1011' and that the A/D module is enabled (ADON bit is set). When the trigger occurs, the GO/$\overline{\text{DONE}}$ bit will be set, starting the A/D conversion and the Timer1 (or Timer3) counter will be reset to zero. Timer1 (or Timer3) is reset to automatically repeat the A/D acquisition period with minimal software overhead (moving ADRESH/ADRESL to the desired location). The appropriate analog input channel must be selected and the minimum acquisition done before the "special event trigger" sets the GO/$\overline{\text{DONE}}$ bit (starts a conversion).

If the A/D module is not enabled (ADON is cleared), the "special event trigger" will be ignored by the A/D module but will still reset the Timer1 (or Timer3) counter.

FIGURE 19-3: A/D CONVERSION TAD CYCLES (ACQT<2:0> = 000, TACQ = 0)

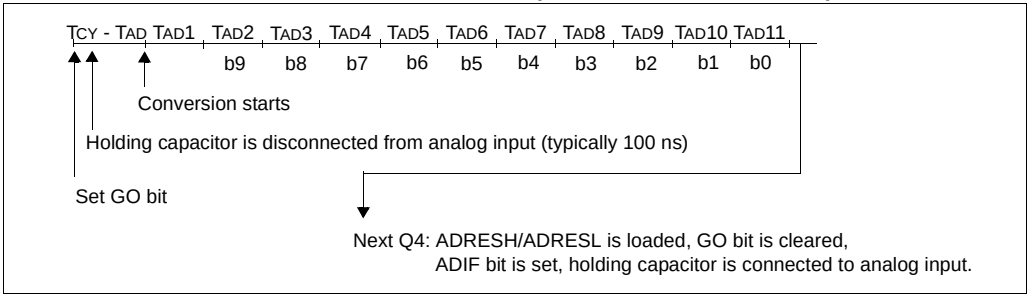
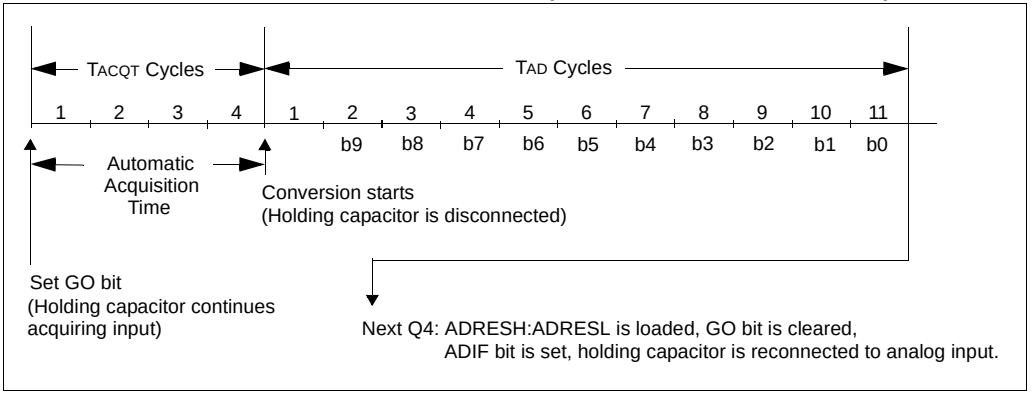


FIGURE 19-4: A/D CONVERSION TAD CYCLES (ACQT<2:0> = 010, TACQ = 4 TAD)



REGISTER 23-49: MSEL1: MASK SELECT REGISTER 1(1)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-1	R/W-0	R/W-1
FIL7_1	FIL7_0	FIL6_1	FIL6_0	FIL5_1	FIL5_0	FIL4_1	FIL4_0
bit 7							bit 0

bit 7-6 **FIL7_1:FIL7_0:** Filter 7 Select bits 1 and 0
11 = No mask
10 = Filter 15
01 = Acceptance Mask 1
00 = Acceptance Mask 0

bit 5-4 **FIL6_1:FIL6_0:** Filter 6 Select bits 1 and 0
11 = No mask
10 = Filter 15
01 = Acceptance Mask 1
00 = Acceptance Mask 0

bit 3-2 **FIL5_1:FIL5_0:** Filter 5 Select bits 1 and 0
11 = No mask
10 = Filter 15
01 = Acceptance Mask 1
00 = Acceptance Mask 0

bit 1-0 **FIL4_1:FIL4_0:** Filter 4 Select bits 1 and 0
11 = No mask
10 = Filter 15
01 = Acceptance Mask 1
00 = Acceptance Mask 0

Note 1: This register is available in Mode 1 and 2 only.

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

REGISTER 23-57: PIE3: PERIPHERAL INTERRUPT ENABLE REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
IRXIE	WAKIE	ERRIE	TXB2IE/ TXBnIE	TXB1IE(1)	TXB0IE(1)	RXB1IE/ RXBnIE	RXB0IE/ FIFOWMIE
bit 7							bit 0

bit 7 **IRXIE:** CAN Invalid Received Message Interrupt Enable bit
1 = Enable invalid message received interrupt
0 = Disable invalid message received interrupt

bit 6 **WAKIE:** CAN bus Activity Wake-up Interrupt Enable bit
1 = Enable bus activity wake-up interrupt
0 = Disable bus activity wake-up interrupt

bit 5 **ERRIE:** CAN bus Error Interrupt Enable bit
1 = Enable CAN bus error interrupt
0 = Disable CAN bus error interrupt

bit 4 When CAN is in Mode 0:
TXB2IE: CAN Transmit Buffer 2 Interrupt Enable bit
1 = Enable Transmit Buffer 2 interrupt
0 = Disable Transmit Buffer 2 interrupt
When CAN is in Mode 1 or 2:
TXBnIE: CAN Transmit Buffer Interrupts Enable bit
1 = Enable transmit buffer interrupt; individual interrupt is enabled by TXBIE and BIE0
0 = Disable all transmit buffer interrupts

bit 3 **TXB1IE:** CAN Transmit Buffer 1 Interrupt Enable bit(1)
1 = Enable Transmit Buffer 1 interrupt
0 = Disable Transmit Buffer 1 interrupt

bit 2 **TXB0IE:** CAN Transmit Buffer 0 Interrupt Enable bit(1)
1 = Enable Transmit Buffer 0 interrupt
0 = Disable Transmit Buffer 0 interrupt

bit 1 When CAN is in Mode 0:
RXB1IE: CAN Receive Buffer 1 Interrupt Enable bit
1 = Enable Receive Buffer 1 interrupt
0 = Disable Receive Buffer 1 interrupt
When CAN is in Mode 1 or 2:
RXBnIE: CAN Receive Buffer Interrupts Enable bit
1 = Enable receive buffer interrupt; individual interrupt is enabled by BIE0
0 = Disable all receive buffer interrupts

bit 0 When CAN is in Mode 0:
RXB0IE: CAN Receive Buffer 0 Interrupt Enable bit
1 = Enable Receive Buffer 0 interrupt
0 = Disable Receive Buffer 0 interrupt
When CAN is in Mode 1:
Unimplemented: Read as '0'
When CAN is in Mode 2:
FIFOWMIE: FIFO Watermark Interrupt Enable bit
1 = Enable FIFO watermark interrupt
0 = Disable FIFO watermark interrupt

Note 1: In CAN Mode 1 and 2, this bit is forced to '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

23.5.3 PROGRAMMABLE TRANSMIT/ RECEIVE BUFFERS

The ECAN module implements six new buffers: B0-B5. These buffers are individually programmable as either transmit or receive buffers. These buffers are available only in Mode 1 and 2. As with dedicated transmit and receive buffers, each of these programmable buffers occupies 14 bytes of SRAM and are mapped into SFR memory map.

Each buffer contains one Control register (BnCON), four Identifier registers (BnSIDL, BnSIDH, BnEIDL, BnEIDH), one Data Length Count register (BnDLC) and eight Data Byte registers (BnDm). Each of these registers contains two sets of control bits. Depending on whether the buffer is configured as transmit or receive, one would use the corresponding control bit set. By default, all buffers are configured as receive buffers. Each buffer can be individually configured as transmit or receive buffers by setting the corresponding TXENn bit in the BSEL0 register.

When configured as transmit buffers, user firmware may access transmit buffers in any order similar to accessing dedicated transmit buffers. In receive configuration, with Mode 1 enabled, user firmware may also access receive buffers in any order required. But in Mode 2, all receive buffers are combined to form a single FIFO. Actual FIFO length is programmable by user firmware. Access to FIFO must be done through the FIFO pointer bits (FP<4:0>) in the CANCON register. It must be noted that there is no hardware protection against out of order FIFO reads.

23.5.4 PROGRAMMABLE AUTO-RTR BUFFERS

In Mode 1 and 2, any of six programmable transmit/ receive buffers may be programmed to automatically respond to predefined RTR messages without user firmware intervention. Automatic RTR handling is enabled by setting the TXnEN bit in the BSEL0 register and the RTREN bit in the BnCON register. After this setup, when an RTR request is received, the TXREQ bit is automatically set and current buffer content is automatically queued for transmission as a RTR response. As with all transmit buffers, once the TXREQ bit is set, buffer registers become read-only and any writes to them will be ignored.

The following outlines the steps required to automatically handle RTR messages:

1. Set buffer to Transmit mode by setting TXnEN bit to '1' in BSEL0 register.
2. At least one acceptance filter must be associated with this buffer and preloaded with expected RTR identifier.
3. Bit RTREN in BnCON register must be set to '1'.
4. Buffer must be preloaded with the data to be sent as a RTR response.

Normally, user firmware will keep Buffer Data registers up to date. If firmware attempts to update buffer while an automatic RTR response is in process of transmission, all writes to buffers are ignored.

23.6 CAN Message Transmission

23.6.1 INITIATING TRANSMISSION

For the MCU to have write access to the message buffer, the TXREQ bit must be clear, indicating that the message buffer is clear of any pending message to be transmitted. At a minimum, the SIDH, SIDL, and DLC registers must be loaded. If data bytes are present in the message, the data registers must also be loaded. If the message is to use extended identifiers, the EIDH:EIDL registers must also be loaded and the EXIDE bit set.

To initiate message transmission, the TXREQ bit must be set for each buffer to be transmitted. When TXREQ is set, the TXABT, TXLARB and TXERR bits will be cleared. To successfully complete the transmission, there must be at least one node with matching baud rate on the network.

Setting the TXREQ bit does not initiate a message transmission, it merely flags a message buffer as ready for transmission. Transmission will start when the device detects that the bus is available. The device will then begin transmission of the highest priority message that is ready.

When the transmission has completed successfully, the TXREQ bit will be cleared, the TXBnIF bit will be set, and an interrupt will be generated if the TXBnIE bit is set.

If the message transmission fails, the TXREQ will remain set, indicating that the message is still pending for transmission and one of the following condition flags will be set. If the message started to transmit but encountered an error condition, the TXERR and the IRXIF bits will be set and an interrupt will be generated. If the message lost arbitration, the TXLARB bit will be set.

23.7 Message Reception

23.7.1 RECEIVING A MESSAGE

Of all receive buffers, the MAB is always committed to receiving the next message from the bus. The MCU can access one buffer while the other buffer is available for message reception, or holding a previously received message.

Note:	The entire contents of the MAB are moved into the receive buffer once a message is accepted. This means that regardless of the type of identifier (standard or extended) and the number of data bytes received, the entire receive buffer is overwritten with the MAB contents. Therefore, the contents of all registers in the buffer must be assumed to have been modified when any message is received.

When a message is moved into either of the receive buffers, the associated RXFUL bit is set. This bit must be cleared by the MCU when it has completed processing the message in the buffer in order to allow a new message to be received into the buffer. This bit provides a positive lockout to ensure that the firmware has finished with the message before the module attempts to load a new message into the receive buffer. If the receive interrupt is enabled, an interrupt will be generated to indicate that a valid message has been received.

Once a message is loaded into any matching buffer, user firmware may determine exactly what filter caused this reception by checking the filter hit bits in the RXBnCON or BnCON registers. In Mode 0, FILHIT<3:0> of RXBnCON serve as filter hit bits. In Mode 1 and 2, FILHIT<4:0> of BnCON serve as filter hit bits. The same registers also indicate whether the current message is RTR frame or not. A received message is considered a standard identifier message if the EXID bit in RXBnSIDL or the BnSIDL register is cleared. Conversely, a set EXID bit indicates an extended identifier message. If the received message is a standard identifier message, user firmware needs to read the SIDL and SIDH registers. In the case of an extended identifier message, firmware should read the SIDL, SIDH, EIDL and EIDH registers. If the RXBnDLC or BnDLC register contain non-zero data count, user firmware should also read the corresponding number of data bytes by accessing the RXBnDm or BnDm registers. When a received message is RTR and if the current buffer is not configured for automatic RTR handling, user firmware must take appropriate action and respond manually.

Each receive buffer contains RXM bits to set special Receive modes. In Mode 0, RXM<1:0> bits in RXBnCON define a total of four Receive modes. In Mode 1 and 2, RXM1 bit in combination with the EXID mask and filter bit define the same four Receive modes. Normally, these bits are set to '00' to enable reception of all valid messages as determined by the appropriate acceptance filters. In this case, the determination of whether or not to receive standard or extended messages is determined by the EXIDE bit in the Acceptance Filter register. In Mode 0, if the RXM bits are set to '01' or '10', the receiver will accept only messages with standard or extended identifiers, respectively. If an acceptance filter has the EXIDE bit set such that it does not correspond with the RXM mode, that acceptance filter is rendered useless. In Mode 1 and 2, setting EXID in the SIDL Mask register will ensure that only standard or extended identifiers are received. These two modes of RXM bits can be used in systems where it is known that only standard or extended messages will be on the bus. If the RXM bits are set to '11' (RXM1 = 1 in Mode 1 and 2), the buffer will receive all messages regardless of the values of the acceptance filters. Also, if a message has an error before the end of frame, that portion of the message assembled in the MAB before the error frame, will be loaded into the buffer. This mode may serve as a valuable debugging tool for a given CAN network. It should not be used in an actual system environment as the actual system will always have some bus errors and all nodes on the bus are expected to ignore them.

In Mode 1 and 2, when a programmable buffer is configured as a transmit buffer and one or more acceptance filters are associated with it, all incoming messages matching this acceptance filter criteria will be discarded. To avoid this scenario, user firmware must make sure that there are no acceptance filters associated with a buffer configured as a transmit buffer.

23.7.2 RECEIVE PRIORITY

When in Mode 0, RXB0 is the higher priority buffer and has two message acceptance filters associated with it. RXB1 is the lower priority buffer and has four acceptance filters associated with it. The lower number of acceptance filters makes the match on RXB0 more restrictive and implies a higher priority for that buffer. Additionally, the RXB0CON register can be configured such that if RXB0 contains a valid message and another valid message is received, an overflow error will not occur and the new message will be moved into RXB1 regardless of the acceptance criteria of RXB1. There are also two programmable acceptance filter masks available, one for each receive buffer (see Section 4.5).

23.10 Synchronization

To compensate for phase shifts between the oscillator frequencies of each of the nodes on the bus, each CAN controller must be able to synchronize to the relevant signal edge of the incoming signal. When an edge in the transmitted data is detected, the logic will compare the location of the edge to the expected time (Sync_Seg). The circuit will then adjust the values of Phase Segment 1 and Phase Segment 2 as necessary. There are two mechanisms used for synchronization.

23.10.1 HARD SYNCHRONIZATION

Hard synchronization is only done when there is a recessive to dominant edge during a bus Idle condition, indicating the start of a message. After hard synchronization, the bit time counters are restarted with Sync_Seg. Hard synchronization forces the edge which has occurred to lie within the synchronization segment of the restarted bit time. Due to the rules of synchronization, if a hard synchronization occurs there will not be a resynchronization within that bit time.

23.10.2 RESYNCHRONIZATION

As a result of resynchronization, Phase Segment 1 may be lengthened or Phase Segment 2 may be shortened. The amount of lengthening or shortening of the phase buffer segments has an upper bound given by the Synchronization Jump Width (SJW). The value of the SJW will be added to Phase Segment 1 (see Figure 23-5) or subtracted from Phase Segment 2 (see Figure 23-6). The SJW is programmable between 1 TQ and 4 TQ.

Clocking information will only be derived from recessive to dominant transitions. The property that only a fixed maximum number of successive bits have the same value, ensures resynchronization to the bit stream during a frame.

The phase error of an edge is given by the position of the edge relative to Sync_Seg, measured in TQ. The phase error is defined in magnitude of TQ as follows:

- e = 0 if the edge lies within Sync_Seg.
- e > 0 if the edge lies before the sample point.
- e < 0 if the edge lies after the sample point of the previous bit.

If the magnitude of the phase error is less than, or equal to the programmed value of the synchronization jump width, the effect of a resynchronization is the same as that of a hard synchronization.

If the magnitude of the phase error is larger than the synchronization jump width, and if the phase error is positive, then Phase Segment 1 is lengthened by an amount equal to the synchronization jump width.

If the magnitude of the phase error is larger than the resynchronization jump width, and if the phase error is negative, then Phase Segment 2 is shortened by an amount equal to the synchronization jump width.

23.10.3 SYNCHRONIZATION RULES

- Only one synchronization within one bit time is allowed.
- An edge will be used for synchronization only if the value detected at the previous sample point (previously read bus value) differs from the bus value immediately after the edge.
- All other recessive to dominant edges fulfilling rules 1 and 2 will be used for resynchronization, with the exception that a node transmitting a dominant bit will not perform a resynchronization as a result of a recessive to dominant edge with a positive phase error.

RETFIE **Return from Interrupt**

Syntax: [*label*] RETFIE [s]

Operands: s ∈ [0,1]

Operation: (TOS) → PC,
1 → GIE/GIEH or PEIE/GIEL,
if s = 1
(WS) → W,
(STATUSS) → STATUS,
(BSRS) → BSR,
PCLATU, PCLATH are unchanged.

Status Affected: GIE/GIEH, PEIE/GIEL.

Encoding:

0000	0000	0001	000s

Description: Return from interrupt. Stack is popped and Top-of-Stack (TOS) is loaded into the PC. Interrupts are enabled by setting either the high or low priority global interrupt enable bit. If 's' = 1, the contents of the shadow registers WS, STATUSS and BSRS are loaded into their corresponding registers, W, Status and BSR. If 's' = 0, no update of these registers occurs (default).

Words: 1

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	No operation	No operation	Pop PC from stack Set GIEH or GIEL
No operation	No operation	No operation	No operation

Example:

```
RETFIE  1
```

After Interrupt
PC = TOS
W = WS
BSR = BSRS
STATUS = STATUSS
GIE/GIEH, PEIE/GIEL = 1

RETLW **Return Literal to W**

Syntax: [*label*] RETLW k

Operands: 0 ≤ k ≤ 255

Operation: k → W,
(TOS) → PC,
PCLATU, PCLATH are unchanged

Status Affected: None

Encoding:

0000	1100	kkkk	kkkk

Description: W is loaded with the eight-bit literal 'k'. The program counter is loaded from the top of the stack (the return address). The high address latch (PCLATH) remains unchanged.

Words: 1

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Pop PC from stack, Write to W
No operation	No operation	No operation	No operation

Example:

```
    CALL TABLE  ; W contains table
                ; offset value
                ; W now has
                ; table value
    :
TABLE
    ADDWF PCL   ; W = offset
    RETLW k0    ; Begin table
    RETLW k1    ;
    :
    :
    RETLW kn    ; End of table
```

Before Instruction
W = 0x07
After Instruction
W = value of kn

TABLE 27-1: COMPARATOR SPECIFICATIONS

Operating Conditions: 3.0V < V_{DD} < 5.5V, -40°C < T_A < +125°C, unless otherwise stated							
Param No.	**Sym**	**Characteristics**	**Min**	**Typ**	**Max**	**Units**	**Comments**
D300	V_{IOFF}	Input Offset Voltage	—	± 5.0	± 10	mV	
D301	V_{ICM}	Input Common Mode Voltage	0	—	V_{DD} – 1.5	V	
D302	CMRR	Common Mode Rejection Ratio	55	—	—	dB	
300 300A	T_{RESP}	Response Time(1)	—	150	400 600	ns ns	PIC18FXX8X PIC18LFXX8X
301	T_{MC2OV}	Comparator Mode Change to Output Valid	—	—	10	µs	

Note 1: Response time measured with one comparator input at (V_{DD} – 1.5)/2 while the other input transitions from V_{SS} to V_{DD}.

TABLE 27-2: VOLTAGE REFERENCE SPECIFICATIONS

Operating Conditions: 3.0V < V_{DD} < 5.5V, -40°C < T_A < +125°C, unless otherwise stated							
Param No.	**Sym**	**Characteristics**	**Min**	**Typ**	**Max**	**Units**	**Comments**
D310	V_{RES}	Resolution	V_{DD}/24	—	V_{DD}/32	LSb	
D311	V_{RAA}	Absolute Accuracy	— —	— —	1/4 1/2	LSb LSb	Low Range (VRR = 1) High Range (VRR = 0)
D312	V_{RUR}	Unit Resistor Value (R)	—	2k	—	Ω	
310	T_{SET}	Settling Time(1)	—	—	10	µs	

Note 1: Settling time measured while VRR = 1 and VR<3:0> transitions from 0000 to 1111.

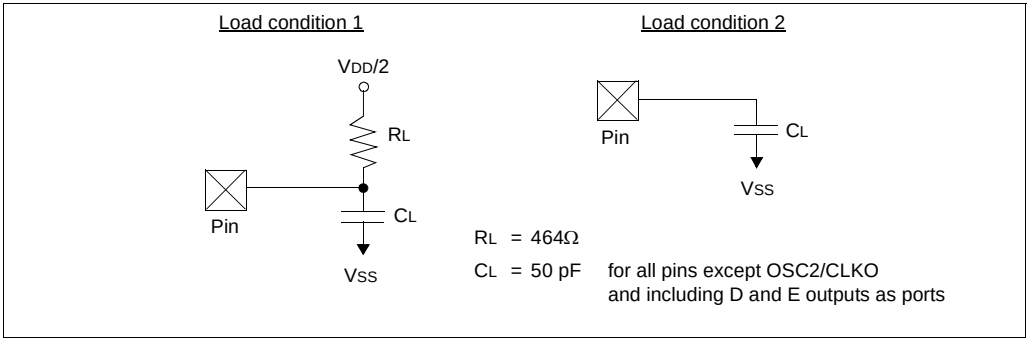
27.4.2 TIMING CONDITIONS

The temperature and voltages specified in Table 27-5 apply to all timing specifications unless otherwise noted. Figure 27-5 specifies the load conditions for the timing specifications.

TABLE 27-5: TEMPERATURE AND VOLTAGE SPECIFICATIONS - AC

AC CHARACTERISTICS	**Standard Operating Conditions (unless otherwise stated)** Operating temperature -40°C ≤ TA ≤ +85°C for industrial -40°C ≤ TA ≤ +125°C for extended Operating voltage VDD range as described in DC spec **Section 27.1** and **Section 27.3**. LC parts operate for industrial temperatures only.

FIGURE 27-5: LOAD CONDITIONS FOR DEVICE TIMING SPECIFICATIONS



27.4.3 TIMING DIAGRAMS AND SPECIFICATIONS

FIGURE 27-6: EXTERNAL CLOCK TIMING (ALL MODES EXCEPT PLL)

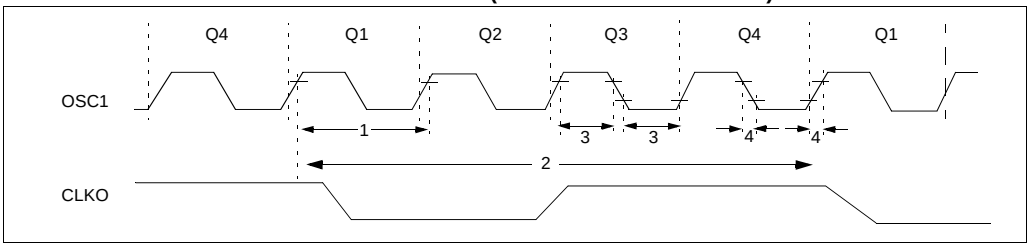


TABLE 27-6: EXTERNAL CLOCK TIMING REQUIREMENTS

Param. No.	Symbol	Characteristic	Min	Max	Units	Conditions
1A	FOSC	External CLKI Frequency(1)	DC	40	MHz	EC, ECIO, -40°C to +85°C
			DC	25	MHz	EC,ECIO, -40°C to +85°C, EMA
		Oscillator Frequency(1)	DC	25	MHz	EC, ECIO, +85°C to +125°C
			DC	16	MHz	EC, ECIO, +85°C to +125°C, EMA
			DC	4	MHz	RC oscillator
			0.1	4	MHz	XT oscillator
			4	25	MHz	HS oscillator, -40°C to +85°C
			4	25	MHz	HS oscillator, -40°C to +85°C, EMA
			4	25	MHz	HS oscillator, +85°C to +125°C
			4	16	MHz	HS oscillator, +85°C to +125°C, EMA
			4	10	MHz	HS + PLL oscillator, -40°C to +85°C
			4	6.25	MHz	HS + PLL oscillator, +85°C to +125°C
			DC	200	kHz	LP oscillator
1	TOSC	External CLKI Period(1)	25	—	ns	EC, ECIO, -40°C to +85°C
		Oscillator Period(1)	40	—	ns	EC,ECIO, -40°C to +85°C, EMA
			40	—	ns	EC, ECIO, +85°C to +125°C
			62.5	—	ns	EC, ECIO, +85°C to +125°C, EMA
			250	—	ns	RC oscillator
			250	10,000	ns	XT oscillator
			40	—	ns	HS oscillator, -40°C to +85°C
			40	—	ns	HS oscillator, -40°C to +85°C, EMA
			40	—	ns	HS oscillator, +85°C to +125°C
			62.5	—	ns	HS oscillator, +85°C to +125°C, EMA
			100	250	ns	HS + PLL oscillator, -40°C to +85°C
			160	250	ns	HS + PLL oscillator, +85°C to +125°C
			5	200	µs	LP oscillator
2	TCY	Instruction Cycle Time(1)	100	—	ns	TCY = 4/FOSC, -40°C to +85°C
			160	—	ns	TCY = 4/FOSC, +85°C to +125°C
3	TOSL, TOSH	External Clock in (OSC1) High or Low Time	30	—	ns	XT oscillator
			2.5	—	µs	LP oscillator
			10	—	ns	HS oscillator
4	TOSR, TOSF	External Clock in (OSC1) Rise or Fall Time	—	20	ns	XT oscillator
			—	50	ns	LP oscillator
			—	7.5	ns	HS oscillator

Note 1: Instruction cycle period (TCY) equals four times the input oscillator time base period for all configurations except PLL. All specified values are based on characterization data for that particular oscillator type under standard operating conditions with the device executing code. Exceeding these specified limits may result in an unstable oscillator operation and/or higher than expected current consumption. All devices are tested to operate at “min.” values with an external clock applied to the OSC1/CLKI pin. When an external clock input is used, the “max.” cycle time limit is “DC” (no clock) for all devices.

FIGURE 28-21: TYPICAL, MINIMUM AND MAXIMUM V_{OH} vs. I_{OH} (V_{DD} = 5V, -40°C TO +125°C)

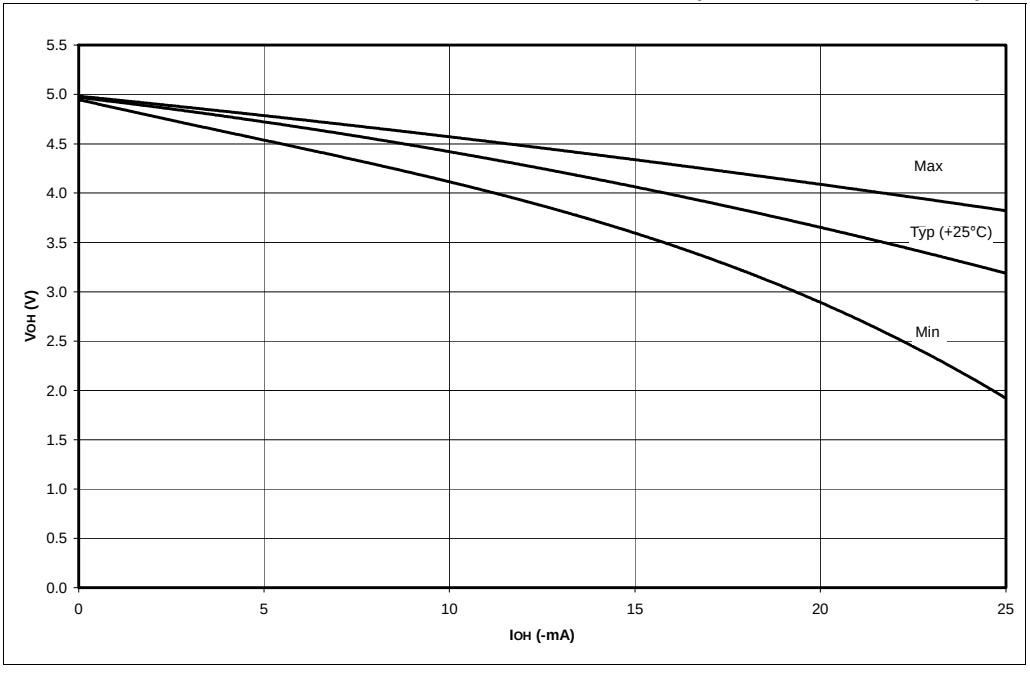
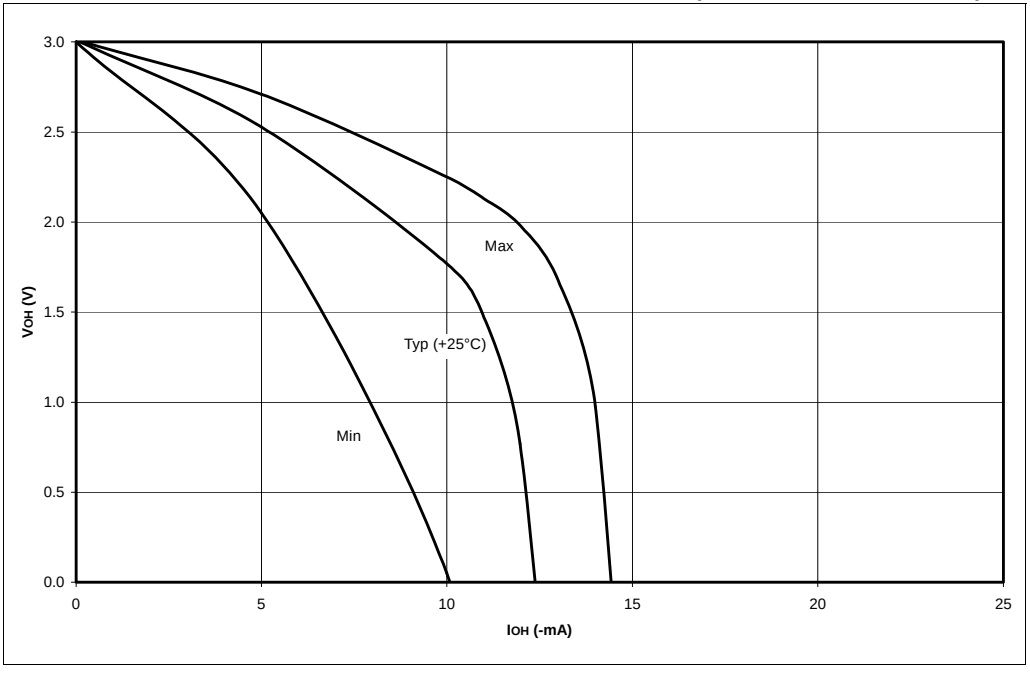


FIGURE 28-22: TYPICAL, MINIMUM AND MAXIMUM V_{OH} vs. I_{OH} (V_{DD} = 3V, -40°C TO +125°C)



INDEX

A

B