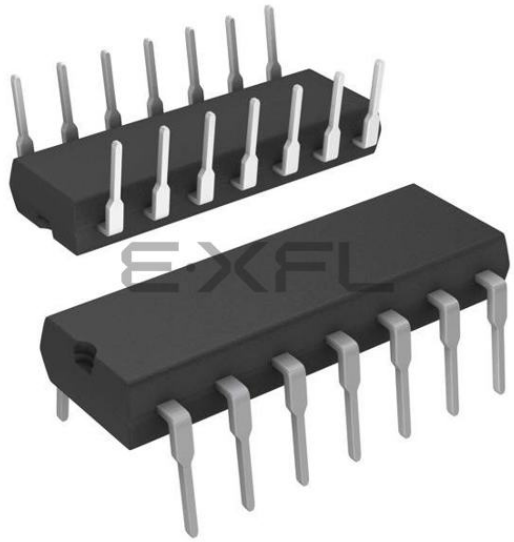


Welcome to E-XFL.COM

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	20MHz
Connectivity	I²C, SPI
Peripherals	Brown-out Detect/Reset, POR, PWM, WDT
Number of I/O	11
Program Memory Size	3.5KB (2K x 14)
Program Memory Type	FLASH
EEPROM Size	-
RAM Size	128 x 8
Voltage - Supply (Vcc/Vdd)	2.3V ~ 5.5V
Data Converters	A/D 8x10b
Oscillator Type	Internal
Operating Temperature	-40°C ~ 85°C (TA)
Mounting Type	Through Hole
Package / Case	14-DIP (0.300", 7.62mm)
Supplier Device Package	14-PDIP
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic16f1503-i-p

2.1 Automatic Interrupt Context Saving

During interrupts, certain registers are automatically saved in shadow registers and restored when returning from the interrupt. This saves stack space and user code. See **Section 7.5 "Automatic Context Saving"**, for more information.

2.2 16-Level Stack with Overflow and Underflow

These devices have a hardware stack memory 15 bits wide and 16 words deep. A Stack Overflow or Underflow will set the appropriate bit (STKOVF or STKUNF) in the PCON register, and if enabled, will cause a software Reset. See **Section 3.5 "Stack"** for more details.

2.3 File Select Registers

There are two 16-bit File Select Registers (FSR). FSRs can access all file registers and program memory, which allows one Data Pointer for all memory. When an FSR points to program memory, there is one additional instruction cycle in instructions using INDF to allow the data to be fetched. General purpose memory can now also be addressed linearly, providing the ability to access contiguous data larger than 80 bytes. There are also new instructions to support the FSRs. See **Section 3.6 "Indirect Addressing"** for more details.

2.4 Instruction Set

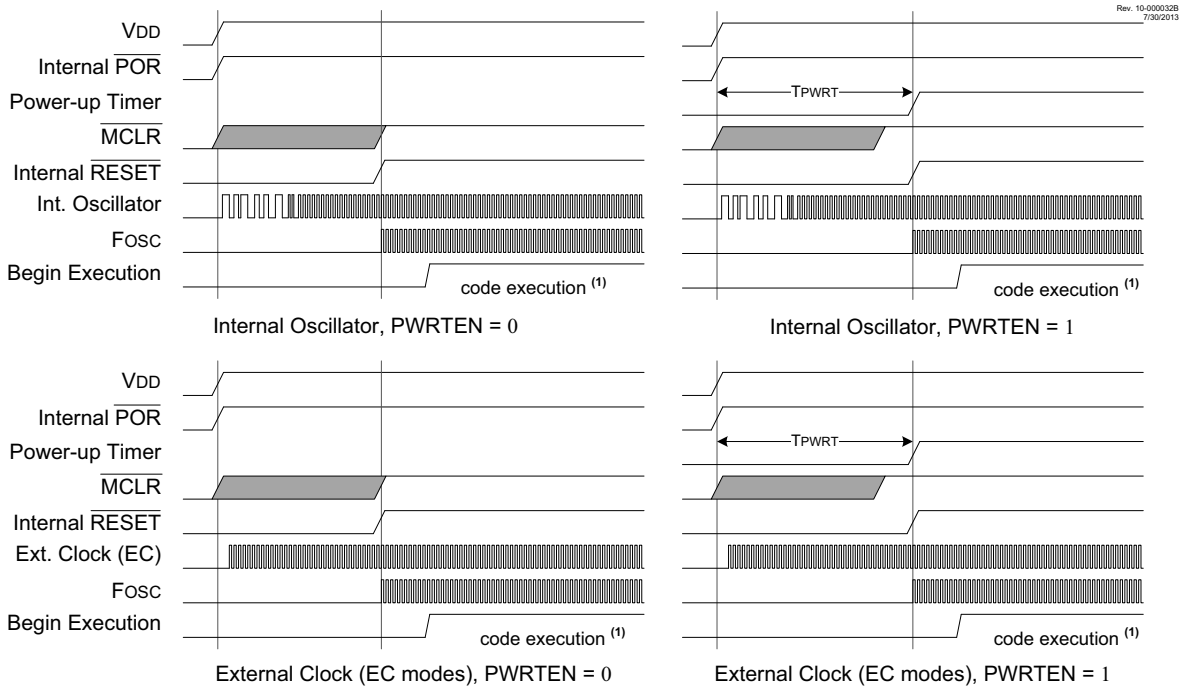
There are 49 instructions for the enhanced mid-range CPU to support the features of the CPU. See **Section 27.0 "Instruction Set Summary"** for more details.

TABLE 3-5: SPECIAL FUNCTION REGISTER SUMMARY (CONTINUED)

Address	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
Bank 10											
50Ch to 51Fh	—	Unimplemented								—	—
Bank 11											
58Ch to 59Fh	—	Unimplemented								—	—
Bank 12											
60Ch to 610h	—	Unimplemented								—	—
611h	PWM1DCL	PWM1DCL<7:6>		—	—	—	—	—	—	00-- ----	00-- ----
612h	PWM1DCH	PWM1DCH<7:0>								xxxx xxxx	uuuu uuuu
613h	PWM1CON0	PWM1EN	PWM1OE	PWM1OUT	PWM1POL	—	—	—	—	0000 ----	0000 ----
614h	PWM2DCL	PWM2DCL<7:6>		—	—	—	—	—	—	00-- ----	00-- ----
615h	PWM2DCH	PWM2DCH<7:0>								xxxx xxxx	uuuu uuuu
616h	PWM2CON0	PWM2EN	PWM2OE	PWM2OUT	PWM2POL	—	—	—	—	0000 ----	0000 ----
617h	PWM3DCL	PWM3DCL<7:6>		—	—	—	—	—	—	00-- ----	00-- ----
618h	PWM3DCH	PWM3DCH<7:0>								xxxx xxxx	uuuu uuuu
619h	PWM3CON0	PWM3EN	PWM3OE	PWM3OUT	PWM3POL	—	—	—	—	0000 ----	0000 ----
61Ah	PWM4DCL	PWM4DCL<7:6>		—	—	—	—	—	—	00-- ----	00-- ----
61Bh	PWM4DCH	PWM4DCH<7:0>								xxxx xxxx	uuuu uuuu
61Ch	PWM4CON0	PWM4EN	PWM4OE	PWM4OUT	PWM4POL	—	—	—	—	0000 ----	0000 ----
61Dh to 61Fh	—	Unimplemented								—	—
Bank 13											
68Ch to 690h	—	Unimplemented								—	—
691h	CWG1DBR	—	—	CWG1DBR<5:0>						--00 0000	--00 0000
692h	CWG1DBF	—	—	CWG1DBF<5:0>						--xx xxxx	--xx xxxx
693h	CWG1CON0	G1EN	G1OEB	G1OEA	G1POLB	G1POLA	—	—	G1CS0	0000 0--0	0000 0--0
694h	CWG1CON1	G1ASDLB<1:0>		G1ASDLA<1:0>		—	G1IS<2:0>			0000 -000	0000 -000
695h	CWG1CON2	G1ASE	G1ARSEN	—	—	G1ASDSC2	G1ASDSC1	G1ASDSFLT	G1ASDSCLC2	00-- 0000	00-- 0000
696h to 69Fh	—	Unimplemented								—	—

Legend: x = unknown, u = unchanged, q = value depends on condition, - = unimplemented, r = reserved. Shaded locations are unimplemented, read as '0'.
Note 1: PIC16F1503 only.
2: Unimplemented, read as '1'.

FIGURE 6-3: RESET START-UP SEQUENCE



Note 1: Code execution begins 10 FOSC cycles after the FOSC clock is released.

EXAMPLE 10-2: ERASING ONE ROW OF PROGRAM MEMORY

```
; This row erase routine assumes the following:
; 1. A valid address within the erase row is loaded in ADDRH:ADDRL
; 2. ADDRH and ADDRL are located in shared data memory 0x70 - 0x7F (common RAM)


        BCF         INTCON,GIE      ; Disable ints so required sequences will execute properly
        BANKSEL     PMADRL
        MOVF        ADDRL,W         ; Load lower 8 bits of erase address boundary
        MOVWF       PMADRL
        MOVF        ADDRH,W         ; Load upper 6 bits of erase address boundary
        MOVWF       PMADRH
        BCF         PMCON1,CFGS     ; Not configuration space
        BSF         PMCON1,FREE     ; Specify an erase operation
        BSF         PMCON1,WREN     ; Enable writes

Required
Sequence
        MOVLW       55h             ; Start of required sequence to initiate erase
        MOVWF       PMCON2          ; Write 55h
        MOVLW       0AAh            ;
        MOVWF       PMCON2          ; Write AAh
        BSF         PMCON1,WR       ; Set WR bit to begin erase
        NOP                         ; NOP instructions are forced as processor starts
        NOP                         ; row erase of program memory.
                                    ;
                                    ; The processor stalls until the erase process is complete
                                    ; after erase processor continues with 3rd instruction

        BCF         PMCON1,WREN     ; Disable writes
        BSF         INTCON,GIE      ; Enable interrupts
```

EXAMPLE 10-3: WRITING TO FLASH PROGRAM MEMORY (16 WRITE LATCHES)

```
; This write routine assumes the following:
; 1. 32 bytes of data are loaded, starting at the address in DATA_ADDR
; 2. Each word of data to be written is made up of two adjacent bytes in DATA_ADDR,
; stored in little endian format
; 3. A valid starting address (the least significant bits = 00000) is loaded in ADDRH:ADDRL
; 4. ADDRH and ADDRL are located in shared data memory 0x70 - 0x7F (common RAM)
;
        BCF         INTCON,GIE      ; Disable ints so required sequences will execute properly
        BANKSEL     PMADRH          ; Bank 3
        MOVF        ADDRH,W         ; Load initial address
        MOVWF       PMADRH          ;
        MOVF        ADDRL,W         ;
        MOVWF       PMADRL          ;
        MOVLW       LOW DATA_ADDR   ; Load initial data address
        MOVWF       FSR0L           ;
        MOVLW       HIGH DATA_ADDR  ; Load initial data address
        MOVWF       FSR0H           ;
        BCF         PMCON1,CFGS     ; Not configuration space
        BSF         PMCON1,WREN     ; Enable writes
        BSF         PMCON1,LWLO     ; Only Load Write Latches
LOOP
        MOVIW       FSR0++          ; Load first data byte into lower
        MOVWF       PMDATL          ;
        MOVIW       FSR0++          ; Load second data byte into upper
        MOVWF       PMDATH          ;

        MOVF        PMADRL,W        ; Check if lower bits of address are '00000'
        XORLW       0x0F            ; Check if we're on the last of 16 addresses
        ANDLW       0x0F            ;
        BTFSC       STATUS,Z        ; Exit if last of 16 words,
        GOTO        START_WRITE     ;

        MOVLW       55h             ; Start of required write sequence:
        MOVWF       PMCON2          ; Write 55h
        MOVLW       0AAh            ;
        MOVWF       PMCON2          ; Write AAh
        BSF         PMCON1,WR       ; Set WR bit to begin write
        NOP                         ; NOP instructions are forced as processor
                                    ; loads program memory write latches
        NOP                         ;

        INCF        PMADRL,F        ; Still loading latches Increment address
        GOTO        LOOP            ; Write next latches

START_WRITE
        BCF         PMCON1,LWLO     ; No more loading latches - Actually start Flash program
                                    ; memory write

Required Sequence
        MOVLW       55h             ; Start of required write sequence:
        MOVWF       PMCON2          ; Write 55h
        MOVLW       0AAh            ;
        MOVWF       PMCON2          ; Write AAh
        BSF         PMCON1,WR       ; Set WR bit to begin write
        NOP                         ; NOP instructions are forced as processor writes
                                    ; all the program memory write latches simultaneously
        NOP                         ; to program memory.
                                    ; After NOPs, the processor
                                    ; stalls until the self-write process in complete
                                    ; after write processor continues with 3rd instruction
        BCF         PMCON1,WREN     ; Disable writes
        BSF         INTCON,GIE      ; Enable interrupts
```

REGISTER 10-5: PMCON1: PROGRAM MEMORY CONTROL 1 REGISTER

U-1	R/W-0/0	R/W-0/0	R/W/HC-0/0	R/W/HC-x/q[2]	R/W-0/0	R/S/HC-0/0	R/S/HC-0/0
—[1]	CFGS	LWLO	FREE	WRERR	WREN	WR	RD
bit 7							bit 0

Legend:		
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as ‘0’
S = Bit can only be set	x = Bit is unknown	-n/n = Value at POR and BOR/Value at all other Resets
‘1’ = Bit is set	‘0’ = Bit is cleared	HC = Bit is cleared by hardware

bit 7 **Unimplemented:** Read as ‘1’

bit 6 **CFGS:** Configuration Select bit
- 1 = Access Configuration, User ID and Device ID Registers
- 0 = Access Flash program memory

bit 5 **LWLO:** Load Write Latches Only bit[3]
- 1 = Only the addressed program memory write latch is loaded/updated on the next WR command
- 0 = The addressed program memory write latch is loaded/updated and a write of all program memory write latches will be initiated on the next WR command

bit 4 **FREE:** Program Flash Erase Enable bit
- 1 = Performs an erase operation on the next WR command (hardware cleared upon completion)
- 0 = Performs a write operation on the next WR command

bit 3 **WRERR:** Program/Erase Error Flag bit
- 1 = Condition indicates an improper program or erase sequence attempt or termination (bit is set automatically on any set attempt (write ‘1’) of the WR bit).
- 0 = The program or erase operation completed normally.

bit 2 **WREN:** Program/Erase Enable bit
- 1 = Allows program/erase cycles
- 0 = Inhibits programming/erasing of program Flash

bit 1 **WR:** Write Control bit
- 1 = Initiates a program Flash program/erase operation.
  The operation is self-timed and the bit is cleared by hardware once operation is complete.
  The WR bit can only be set (not cleared) in software.
- 0 = Program/erase operation to the Flash is complete and inactive.

bit 0 **RD:** Read Control bit
- 1 = Initiates a program Flash read. Read takes one cycle. RD is cleared in hardware. The RD bit can only be set (not cleared) in software.
- 0 = Does not initiate a program Flash read.

Note 1: Unimplemented bit, read as ‘1’.
2: The WRERR bit is automatically set by hardware when a program memory write or erase operation is started (WR = 1).
3: The LWLO bit is ignored during a program memory erase operation (FREE = 1).

15.3 Register Definitions: ADC Control

REGISTER 15-1: ADCON0: ADC CONTROL REGISTER 0

U-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
—	CHS<4:0>					GO/$\overline{\text{DONE}}$	ADON
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 **Unimplemented:** Read as '0'

bit 6-2 **CHS<4:0>:** Analog Channel Select bits

00000 = AN0
00001 = AN1
00010 = AN2
00011 = AN3
00100 = AN4
00101 = AN5
00110 = AN6
00111 = AN7
01000 = Reserved. No channel connected.
•
•
•
11100 = Reserved. No channel connected.
11101 = Temperature Indicator(1)
11110 = DAC (Digital-to-Analog Converter)(3)
11111 = FVR (Fixed Voltage Reference) Buffer 1 Output(2)

bit 1 **GO/$\overline{\text{DONE}}$:** ADC Conversion Status bit

1 = ADC conversion cycle in progress. Setting this bit starts an ADC conversion cycle.
This bit is automatically cleared by hardware when the ADC conversion has completed.
0 = ADC conversion completed/not in progress

bit 0 **ADON:** ADC Enable bit

1 = ADC is enabled
0 = ADC is disabled and consumes no operating current

Note 1: See **Section 14.0 "Temperature Indicator Module"** for more information.
2: See **Section 13.0 "Fixed Voltage Reference (FVR)"** for more information.
3: See **Section 16.0 "5-Bit Digital-to-Analog Converter (DAC) Module"** for more information.

16.1 Output Voltage Selection

The DAC has 32 voltage level ranges. The 32 levels are set with the DACR<4:0> bits of the DACxCON1 register.

The DAC output voltage can be determined by using Equation 16-1.

16.2 Ratiometric Output Level

The DAC output value is derived using a resistor ladder with each end of the ladder tied to a positive and negative voltage reference input source. If the voltage of either input source fluctuates, a similar fluctuation will result in the DAC output value.

The value of the individual resistors within the ladder can be found in Table 28-14.

16.3 DAC Voltage Reference Output

The unbuffered DAC voltage can be output to the DACxOUTn pin(s) by setting the respective DACOEn bit(s) of the DACxCON0 register. Selecting the DAC reference voltage for output on either DACxOUTn pin automatically overrides the digital output buffer, the weak pull-up and digital input threshold detector functions of that pin.

Reading the DACxOUTn pin when it has been configured for DAC reference voltage output will always return a '0'.

Note:	The unbuffered DAC output (DACxOUTn) is not intended to drive an external load.

16.4 Operation During Sleep

When the device wakes up from Sleep through an interrupt or a Watchdog Timer time-out, the contents of the DACxCON0 register are not affected. To minimize current consumption in Sleep mode, the voltage reference should be disabled.

16.5 Effects of a Reset

A device Reset affects the following:

- DACx is disabled.
- DACx output voltage is removed from the DACxOUTn pin(s).
- The DACR<4:0> range select bits are cleared.

EQUATION 16-1: DAC OUTPUT VOLTAGE

IF DACEN = 1

$$DACx\_output = \left((V_{SOURCE+} - V_{SOURCE-}) \times \frac{DACR[4:0]}{2^5} \right) + V_{SOURCE-}$$

Note: See the DACxCON0 register for the available $V_{SOURCE+}$ and $V_{SOURCE-}$ selections.

19.5.2.1 T1G Pin Gate Operation

The T1G pin is one source for Timer1 gate control. It can be used to supply an external source to the Timer1 gate circuitry.

19.5.2.2 Timer0 Overflow Gate Operation

When Timer0 increments from FFh to 00h, a low-to-high pulse will automatically be generated and internally supplied to the Timer1 gate circuitry.

19.5.3 TIMER1 GATE TOGGLE MODE

When Timer1 Gate Toggle mode is enabled, it is possible to measure the full-cycle length of a Timer1 gate signal, as opposed to the duration of a single level pulse.

The Timer1 gate source is routed through a flip-flop that changes state on every incrementing edge of the signal. See Figure 19-4 for timing details.

Timer1 Gate Toggle mode is enabled by setting the T1GTM bit of the T1GCON register. When the T1GTM bit is cleared, the flip-flop is cleared and held clear. This is necessary in order to control which edge is measured.

Note:	Enabling Toggle mode at the same time as changing the gate polarity may result in indeterminate operation.

19.5.4 TIMER1 GATE SINGLE-PULSE MODE

When Timer1 Gate Single-Pulse mode is enabled, it is possible to capture a single pulse gate event. Timer1 Gate Single-Pulse mode is first enabled by setting the T1GSPM bit in the T1GCON register. Next, the T1GGO/$\overline{\text{DONE}}$ bit in the T1GCON register must be set. The Timer1 will be fully enabled on the next incrementing edge. On the next trailing edge of the pulse, the T1GGO/$\overline{\text{DONE}}$ bit will automatically be cleared. No other gate events will be allowed to increment Timer1 until the T1GGO/$\overline{\text{DONE}}$ bit is once again set in software. See Figure 19-5 for timing details.

If the Single Pulse Gate mode is disabled by clearing the T1GSPM bit in the T1GCON register, the T1GGO/$\overline{\text{DONE}}$ bit should also be cleared.

Enabling the Toggle mode and the Single-Pulse mode simultaneously will permit both sections to work together. This allows the cycle times on the Timer1 gate source to be measured. See Figure 19-6 for timing details.

19.5.5 TIMER1 GATE VALUE STATUS

When Timer1 Gate Value Status is utilized, it is possible to read the most current level of the gate control value. The value is stored in the T1GVAL bit in the T1GCON register. The T1GVAL bit is valid even when the Timer1 gate is not enabled (TMR1GE bit is cleared).

19.5.6 TIMER1 GATE EVENT INTERRUPT

When Timer1 Gate Event Interrupt is enabled, it is possible to generate an interrupt upon the completion of a gate event. When the falling edge of T1GVAL occurs, the TMR1GIF flag bit in the PIR1 register will be set. If the TMR1GIE bit in the PIE1 register is set, then an interrupt will be recognized.

The TMR1GIF flag bit operates even when the Timer1 gate is not enabled (TMR1GE bit is cleared).

FIGURE 21-9: SPI MODE WAVEFORM (SLAVE MODE WITH CKE = 0)

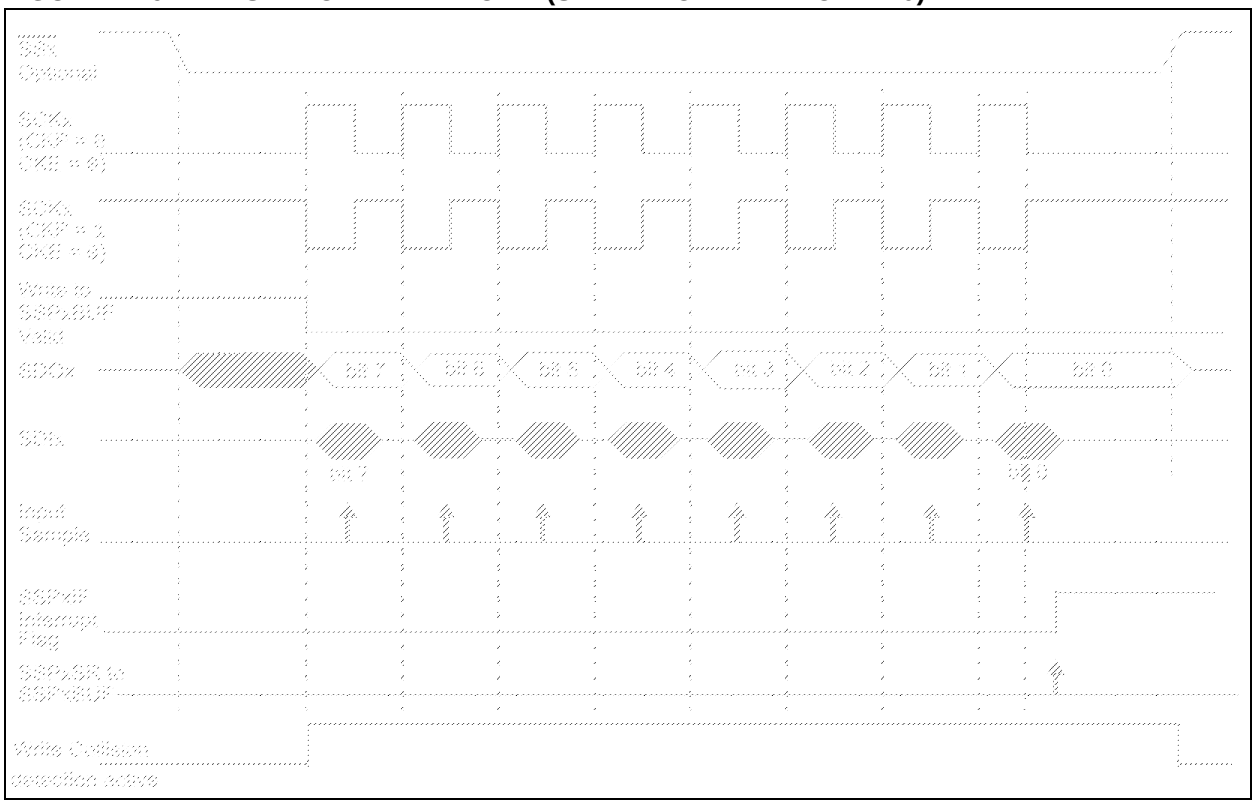


FIGURE 21-10: SPI MODE WAVEFORM (SLAVE MODE WITH CKE = 1)

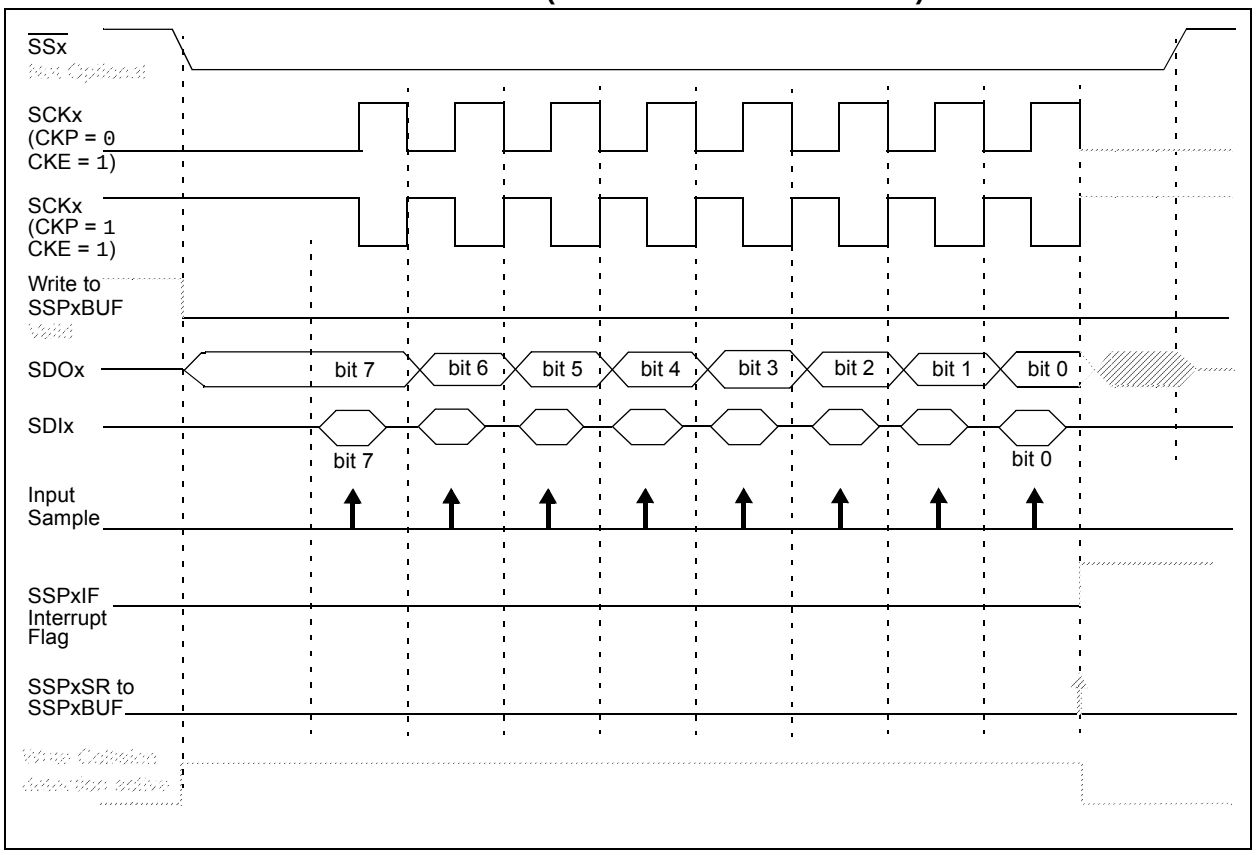
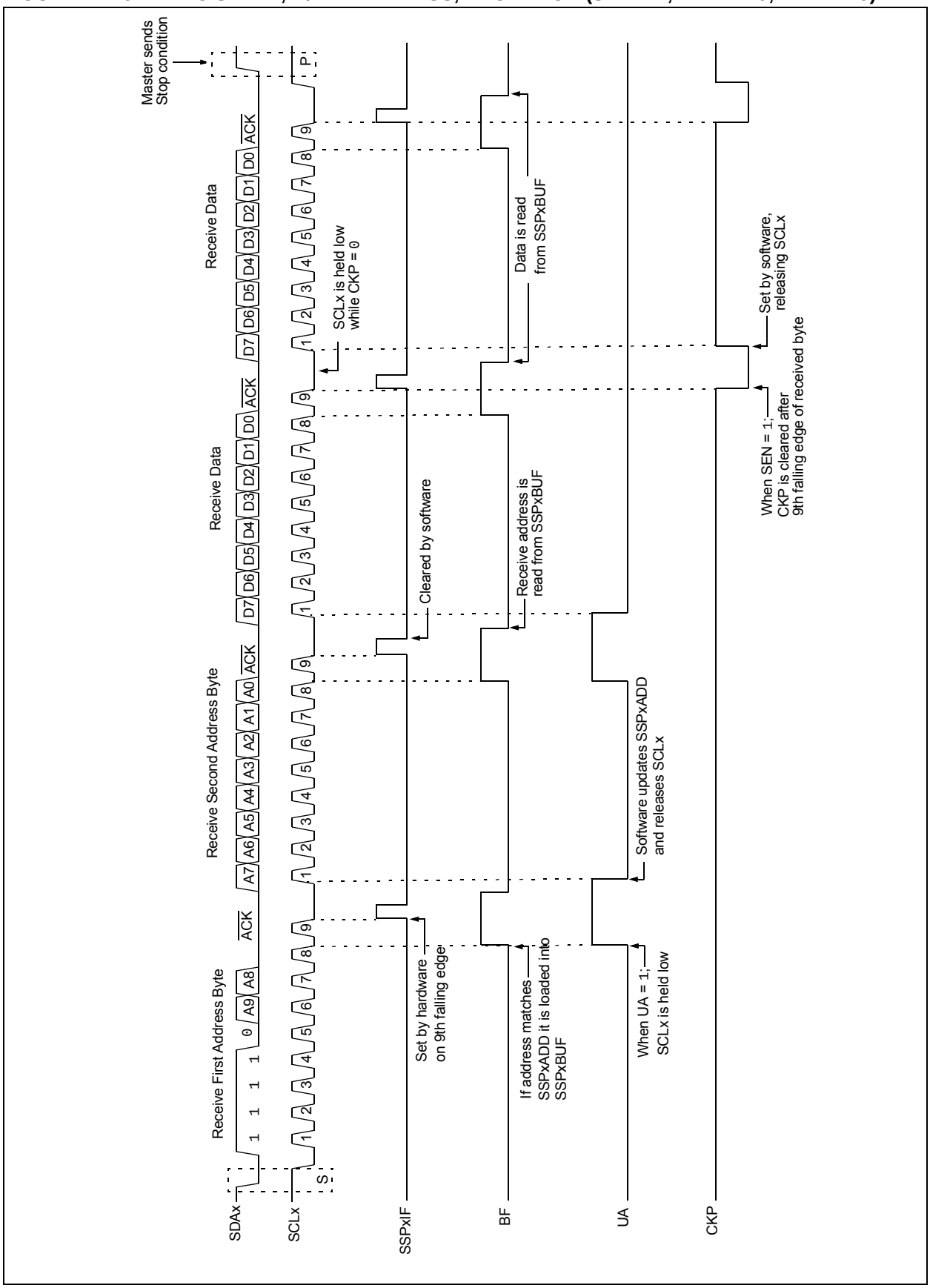


FIGURE 21-20: I²C SLAVE, 10-BIT ADDRESS, RECEPTION (SEN = 1, AHEN = 0, DHEN = 0)



REGISTER 22-2: PWMxDCH: PWM DUTY CYCLE HIGH BITS

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
PWMxDCH<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 **PWMxDCH<7:0>:** PWM Duty Cycle Most Significant bits
These bits are the MSbs of the PWM duty cycle. The two LSbs are found in the PWMxDCL register.

REGISTER 22-3: PWMxDCL: PWM DUTY CYCLE LOW BITS

R/W-x/u	R/W-x/u	U-0	U-0	U-0	U-0	U-0	U-0
PWMxDCL<7:6>		—	—	—	—	—	—
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 **PWMxDCL<7:6>:** PWM Duty Cycle Least Significant bits
These bits are the LSbs of the PWM duty cycle. The MSbs are found in the PWMxDCH register.

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

TABLE 22-3: SUMMARY OF REGISTERS ASSOCIATED WITH PWM

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Register on Page
PR2	Timer2 module Period Register								151*
PWM1CON	PWM1EN	PWM1OE	PWM1OUT	PWM1POL	—	—	—	—	212
PWM1DCH	PWM1DCH<7:0>								213
PWM1DCL	PWM1DCL<7:6>		—	—	—	—	—	—	213
PWM2CON	PWM2EN	PWM2OE	PWM2OUT	PWM2POL	—	—	—	—	212
PWM2DCH	PWM2DCH<7:0>								213
PWM2DCL	PWM2DCL<7:6>		—	—	—	—	—	—	213
PWM3CON	PWM3EN	PWM3OE	PWM3OUT	PWM3POL	—	—	—	—	212
PWM3DCH	PWM3DCH<7:0>								213
PWM3DCL	PWM3DCL<7:6>		—	—	—	—	—	—	213
PWM4CON	PWM4EN	PWM4OE	PWM4OUT	PWM4POL	—	—	—	—	212
PWM4DCH	PWM4DCH<7:0>								213
PWM4DCL	PWM4DCL<7:6>		—	—	—	—	—	—	213
T2CON	—	T2OUTPS<3:0>				TMR2ON	T2CKPS<1:0>		153
TMR2	Timer2 module Register								151*
TRISA	—	—	TRISA5	TRISA4	—(1)	TRISA2	TRISA1	TRISA0	98
TRISC	—	—	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0	102

Legend: - = Unimplemented locations, read as ‘0’, u = unchanged, x = unknown. Shaded cells are not used by the PWM.
* Page provides register information.
Note 1: Unimplemented, read as ‘1’.

27.2 Instruction Descriptions

ADDFSR — Add Literal to FSRn

Syntax:	[*label*] ADDFSR FSRn, k
Operands:	-32 ≤ k ≤ 31 n ∈ [0, 1]
Operation:	FSR(n) + k → FSR(n)
Status Affected:	None
Description:	The signed 6-bit literal ‘k’ is added to the contents of the FSRnH:FSRnL register pair. FSRn is limited to the range 0000h - FFFFh. Moving beyond these bounds will cause the FSR to wrap-around.

ADDLW — Add literal and W

Syntax:	[*label*] ADDLW k
Operands:	0 ≤ k ≤ 255
Operation:	(W) + k → (W)
Status Affected:	C, DC, Z
Description:	The contents of the W register are added to the 8-bit literal ‘k’ and the result is placed in the W register.

ADDWF — Add W and f

Syntax:	[*label*] ADDWF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(W) + (f) → (destination)
Status Affected:	C, DC, Z
Description:	Add the contents of the W register with register ‘f’. If ‘d’ is ‘0’, the result is stored in the W register. If ‘d’ is ‘1’, the result is stored back in register ‘f’.

ADDWFC — ADD W and CARRY bit to f

Syntax:	[*label*] ADDWFC f {,d}
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(W) + (f) + (C) → dest
Status Affected:	C, DC, Z
Description:	Add W, the Carry flag and data memory location ‘f’. If ‘d’ is ‘0’, the result is placed in W. If ‘d’ is ‘1’, the result is placed in data memory location ‘f’.

ANDLW — AND literal with W

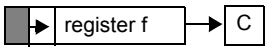
Syntax:	[*label*] ANDLW k
Operands:	0 ≤ k ≤ 255
Operation:	(W) .AND. (k) → (W)
Status Affected:	Z
Description:	The contents of W register are AND’ed with the 8-bit literal ‘k’. The result is placed in the W register.

ANDWF — AND W with f

Syntax:	[*label*] ANDWF f,d
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(W) .AND. (f) → (destination)
Status Affected:	Z
Description:	AND the W register with register ‘f’. If ‘d’ is ‘0’, the result is stored in the W register. If ‘d’ is ‘1’, the result is stored back in register ‘f’.

ASRF — Arithmetic Right Shift

Syntax:	[*label*] ASRF f {,d}
Operands:	0 ≤ f ≤ 127 d ∈ [0,1]
Operation:	(f<7>)→ dest<7> (f<7:1>) → dest<6:0>, (f<0>) → C,
Status Affected:	C, Z
Description:	The contents of register ‘f’ are shifted one bit to the right through the Carry flag. The MSb remains unchanged. If ‘d’ is ‘0’, the result is placed in W. If ‘d’ is ‘1’, the result is stored back in register ‘f’.



28.3 DC Characteristics

TABLE 28-1: SUPPLY VOLTAGE

PIC16LF1503			Standard Operating Conditions (unless otherwise stated)				
PIC16F1503							
Param. No.	Sym.	Characteristic	Min.	Typ†	Max.	Units	Conditions
D001	VDD	**Supply Voltage**					
			VDDMIN		VDDMAX		
			1.8	—	3.6	V	FOSC ≤ 16 MHz
			2.5	—	3.6	V	FOSC ≤ 20 MHz
D001			2.3	—	5.5	V	FOSC ≤ 16 MHz
			2.5	—	5.5	V	FOSC ≤ 20 MHz
D002*	VDR	**RAM Data Retention Voltage(1)**					
			1.5	—	—	V	Device in Sleep mode
D002*			1.7	—	—	V	Device in Sleep mode
D002A*	VPOR	**Power-on Reset Release Voltage(2)**					
			—	1.6	—	V	
D002A*			—	1.6	—	V	
D002B*	VPORR*	**Power-on Reset Rearm Voltage(2)**					
			—	0.8	—	V	
D002B*			—	1.5	—	V	
D003	VFVR	**Fixed Voltage Reference Voltage**					
		1x gain (1.024V nominal)					VDD ≥ 2.5V, -40°C ≤ TA ≤ +85°C
		2x gain (2.048V nominal)	-4	—	+4	%	VDD ≥ 2.5V, -40°C ≤ TA ≤ +85°C
		4x gain (4.096V nominal)	-3	—	+7	%	VDD ≥ 4.75V, -40°C ≤ TA ≤ +85°C
D004*	SVDD	**VDD Rise Rate(2)**	0.05	—	—	V/ms	Ensures that the Power-on Reset signal is released properly.

* These parameters are characterized but not tested.

† 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: This is the limit to which VDD can be lowered in Sleep mode without losing RAM data.

2: See Figure 28-3, POR and POR REARM with Slow Rising VDD.

28.4 AC Characteristics

Timing Parameter Symbology has been created with one of the following formats:

1. TppS2ppS
2. TppS

T			
F	Frequency	T	Time

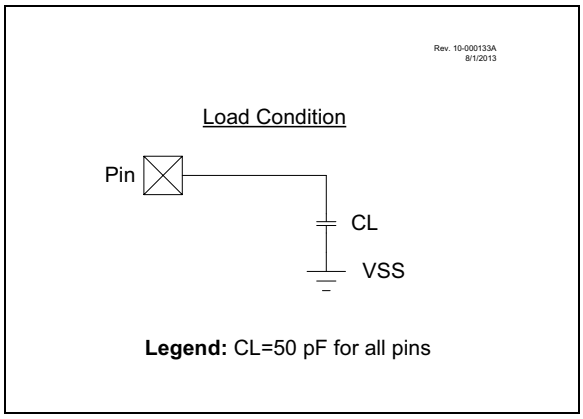
Lowercase letters (pp) and their meanings:

pp			
cc	CCP1	osc	CLKIN
ck	CLKOUT	rd	$\overline{RD}$
cs	$\overline{CS}$	rw	$\overline{RD}$ or $\overline{WR}$
di	SDIx	sc	SCKx
do	SDO	ss	$\overline{SS}$
dt	Data in	t0	T0CKI
io	I/O PORT	t1	T1CKI
mc	$\overline{MCLR}$	wr	$\overline{WR}$

Uppercase letters and their meanings:

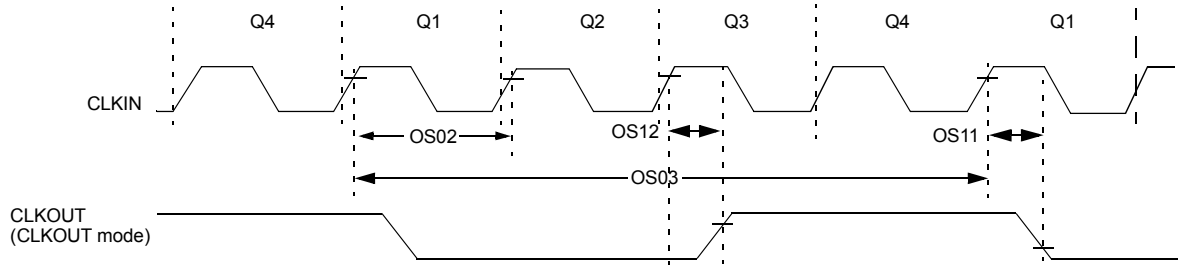
S			
F	Fall	P	Period
H	High	R	Rise
I	Invalid (High-impedance)	V	Valid
L	Low	Z	High-impedance

FIGURE 28-4: LOAD CONDITIONS



Legend: CL=50 pF for all pins

FIGURE 28-5: CLOCK TIMING



Note: See Table 28-9.

TABLE 28-7: CLOCK OSCILLATOR TIMING REQUIREMENTS

Standard Operating Conditions (unless otherwise stated)							
Param. No.	**Sym.**	**Characteristic**	**Min.**	**Typ†**	**Max.**	**Units**	**Conditions**
OS01	FOSC	External CLKIN Frequency(1)	DC	—	0.5	MHz	External Clock (ECL)
			DC	—	4	MHz	External Clock (ECM)
			DC	—	20	MHz	External Clock (ECH)
OS02	TOSC	External CLKIN Period(1)	50	—	∞	ns	External Clock (EC)
OS03	TCY	Instruction Cycle Time(1)	200	TCY	DC	ns	TCY = 4/FOSC

* These parameters are characterized but not tested.

† 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: Instruction cycle period (TCY) equals four times the input oscillator time base period. 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 CLKIN pin. When an external clock input is used, the "max" cycle time limit is "DC" (no clock) for all devices.

TABLE 28-16: DIGITAL-TO-ANALOG CONVERTER (DAC) SPECIFICATIONS(1)

Operating Conditions (unless otherwise stated) V_{DD} = 3.0V, T_A = 25°C							
Param. No.	**Sym.**	**Characteristics**	**Min.**	**Typ.**	**Max.**	**Units**	**Comments**
DAC01*	C_{LSB}	Step Size	—	$V_{DD}/32$	—	V	
DAC02*	C_{ACC}	Absolute Accuracy	—	—	± 1/2	LSb	
DAC03*	C_R	Unit Resistor Value (R)	—	5K	—	Ω	
DAC04*	C_{ST}	Settling Time(2)	—	—	10	μs	

* These parameters are characterized but not tested.

Note 1: See **Section 29.0 "DC and AC Characteristics Graphs and Charts"** for operating characterization.

2: Settling time measured while DACR<4:0> transitions from '00000' to '01111'.

FIGURE 29-35: VOH vs. IOH OVER TEMPERATURE, VDD = 5.5V, PIC16F1503 ONLY

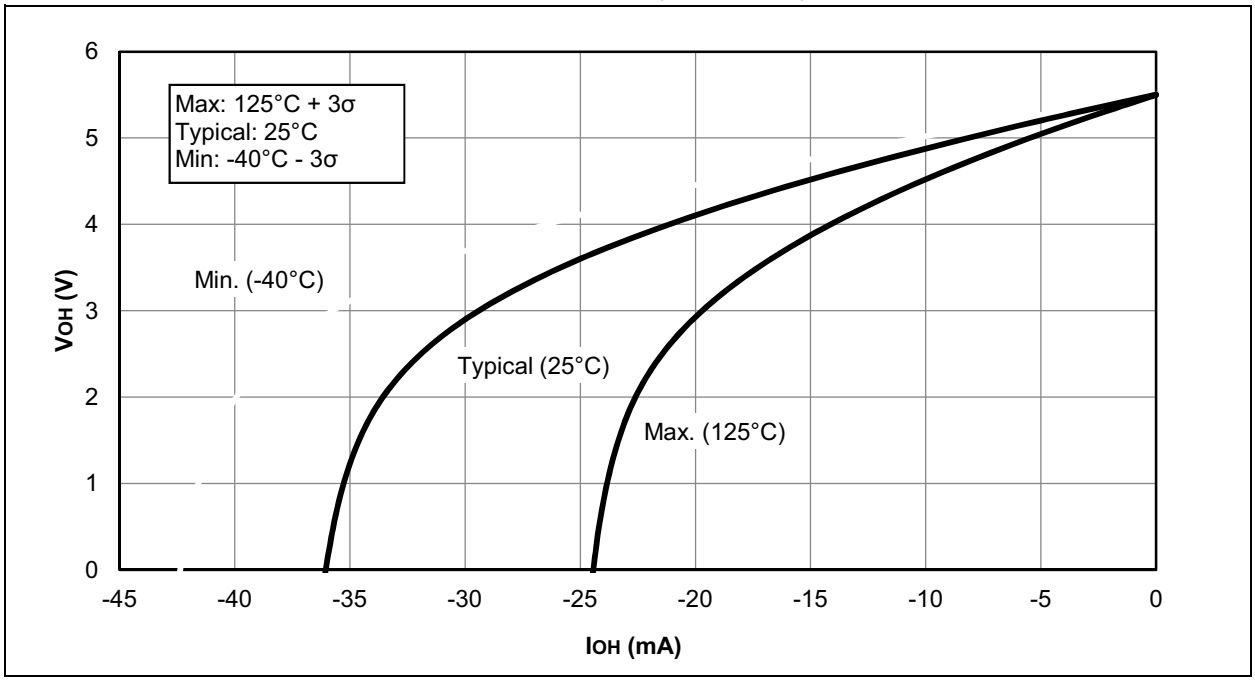
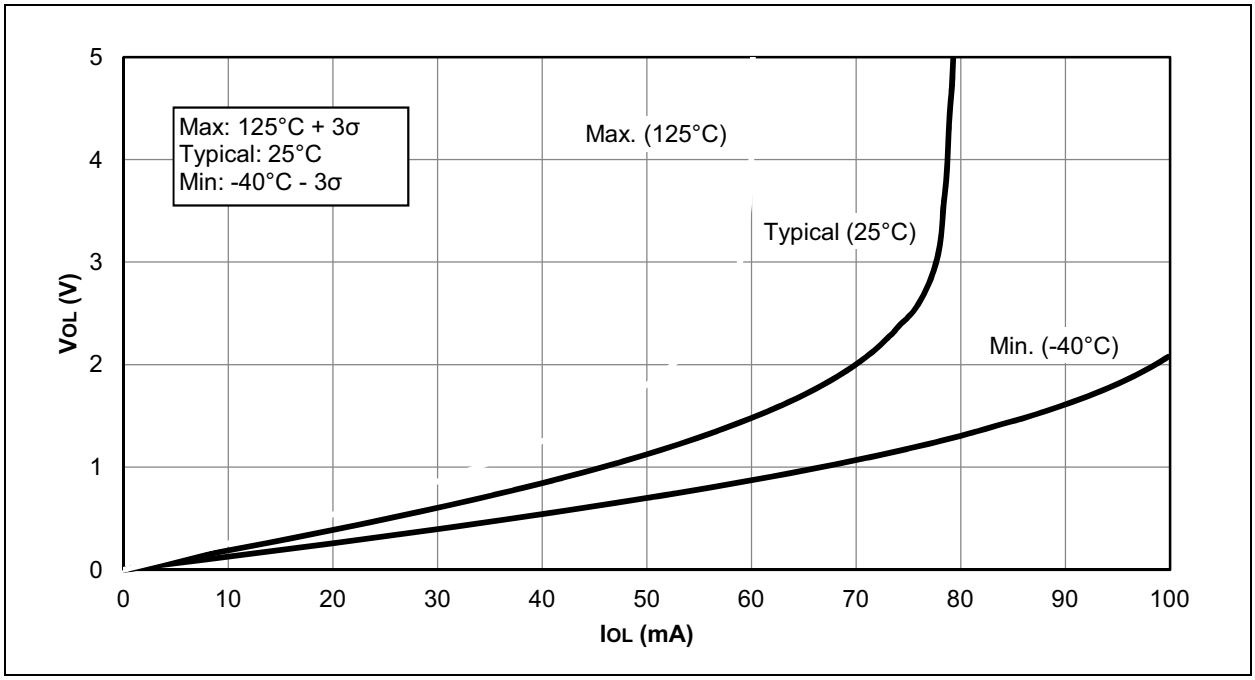


FIGURE 29-36: VOL vs. IOL OVER TEMPERATURE, VDD = 5.5V, PIC16F1503 ONLY



30.0 DEVELOPMENT SUPPORT

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

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

30.1 MPLAB X Integrated Development Environment Software

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

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

Feature-Rich Editor:

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

User-Friendly, Customizable Interface:

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

Project-Based Workspaces:

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

File History and Bug Tracking:

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

30.2 MPLAB XC Compilers

The MPLAB XC Compilers are complete ANSI C compilers for all of Microchip's 8, 16, and 32-bit MCU and DSC devices. These compilers provide powerful integration capabilities, superior code optimization and ease of use. MPLAB XC Compilers run on Windows, Linux or MAC OS X.

For easy source level debugging, the compilers provide debug information that is optimized to the MPLAB X IDE.

The free MPLAB XC Compiler editions support all devices and commands, with no time or memory restrictions, and offer sufficient code optimization for most applications.

MPLAB XC Compilers include an assembler, linker and utilities. The assembler generates relocatable object files that can then be archived or linked with other relocatable object files and archives to create an executable file. MPLAB XC Compiler uses the assembler to produce its object file. Notable features of the assembler include:

- Support for the entire device instruction set
- Support for fixed-point and floating-point data
- Command-line interface
- Rich directive set
- Flexible macro language
- MPLAB X IDE compatibility

30.3 MPASM Assembler

The MPASM Assembler is a full-featured, universal macro assembler for PIC10/12/16/18 MCUs.

The MPASM Assembler generates relocatable object files for the MPLINK Object Linker, Intel® standard HEX files, MAP files to detail memory usage and symbol reference, absolute LST files that contain source lines and generated machine code, and COFF files for debugging.

The MPASM Assembler features include:

- Integration into MPLAB X IDE projects
- User-defined macros to streamline assembly code
- Conditional assembly for multipurpose source files
- Directives that allow complete control over the assembly process

30.4 MPLINK Object Linker/ MPLIB Object Librarian

The MPLINK Object Linker combines relocatable objects created by the MPASM Assembler. It can link relocatable objects from precompiled libraries, using directives from a linker script.

The MPLIB Object Librarian manages the creation and modification of library files of precompiled code. When a routine from a library is called from a source file, only the modules that contain that routine will be linked in with the application. This allows large libraries to be used efficiently in many different applications.

The object linker/library features include:

- Efficient linking of single libraries instead of many smaller files
- Enhanced code maintainability by grouping related modules together
- Flexible creation of libraries with easy module listing, replacement, deletion and extraction

30.5 MPLAB Assembler, Linker and Librarian for Various Device Families

MPLAB Assembler produces relocatable machine code from symbolic assembly language for PIC24, PIC32 and dsPIC DSC devices. MPLAB XC Compiler uses the assembler to produce its object file. The assembler generates relocatable object files that can then be archived or linked with other relocatable object files and archives to create an executable file. Notable features of the assembler include:

- Support for the entire device instruction set
- Support for fixed-point and floating-point data
- Command-line interface
- Rich directive set
- Flexible macro language
- MPLAB X IDE compatibility