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, I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, LVD, POR, PWM, WDT
Number of I/O	68
Program Memory Size	32KB (16K x 16)
Program Memory Type	OTP
EEPROM Size	-
RAM Size	1.5K x 8
Voltage - Supply (Vcc/Vdd)	4.2V ~ 5.5V
Data Converters	A/D 16x10b
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/pic18c858-i-pt

TABLE 1-2: PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number				Pin Type	Buffer Type	Description
	PIC18C658		PIC18C858				
	TQFP	PLCC	TQFP	PLCC			
							PORTE is a bi-directional I/O port
RE0/$\overline{RD}$	2	11	4	15			
RE0					I/O	ST	Digital I/O
$\overline{RD}$					I	TTL	Read control for parallel slave port (See $\overline{WR}$ and $\overline{CS}$ pins)
RE1/$\overline{WR}$	1	10	3	14			
RE1					I/O	ST	Digital I/O
$\overline{WR}$					I	TTL	Write control for parallel slave port (See $\overline{CS}$ and $\overline{RD}$ pins)
RE2/$\overline{CS}$	64	9	78	9			
RE2					I/O	ST	Digital I/O
$\overline{CS}$					I	TTL	Chip select control for parallel slave port (See $\overline{RD}$ and $\overline{WR}$)
RE3	63	8	77	8	I/O	ST	Digital I/O
RE4	62	7	76	7	I/O	ST	Digital I/O
RE5	61	6	75	6	I/O	ST	Digital I/O
RE6	60	5	74	5	I/O	ST	Digital I/O
RE7/CCP2	59	4	73	4			
RE7					I/O	ST	Digital I/O
CCP2					I/O	ST	Capture2 input, Compare2 output, PWM2 output

Legend: TTL = TTL compatible input
ST = Schmitt Trigger input with CMOS levels
I = Input
P = Power
CMOS = CMOS compatible input or output
Analog = Analog input
O = Output
OD = Open Drain (no P diode to V_{DD})

4.7.1 TWO WORD INSTRUCTIONS

The PIC18CXX8 devices have 4 two word instructions: MOVFF, CALL, GOTO and LFSR. The second word of these instructions has the 4 MSB's set to 1's and is a special kind of NOP instruction. The lower 12 bits of the second word contain data to be used by the instruction. If the first word of the instruction is executed, the data in the second word is accessed. If the second word of the instruction is executed by itself (first word was skipped), it will execute as a NOP. This action is necessary when the two word instruction is preceded by a conditional instruction that changes the PC. A program example that demonstrates this concept is shown in Example 4-3. Refer to Section 19.0 for further details of the instruction set.

4.8 Lookup Tables

Lookup tables are implemented two ways. These are:

- Computed GOTO
- Table Reads

4.8.1 COMPUTED GOTO

A computed GOTO is accomplished by adding an offset to the program counter (ADDWF PCL).

A lookup table can be formed with an ADDWF PCL instruction and a group of RETLW 0xnn instructions. WREG is loaded with an offset into the table before executing a call to that table. The first instruction of the called routine is the ADDWF PCL instruction. The next instruction executed will be one of the RETLW 0xnn instructions that returns the value 0xnn to the calling function.

The offset value (value in WREG) specifies the number of bytes that the program counter should advance.

In this method, only one data byte may be stored in each instruction location and room on the return address stack is required.

Warning:	The LSb of PCL is fixed to a value of '0'. Hence, computed GOTO to an odd address is not possible.

4.8.2 TABLE READS/TABLE WRITES

A better method of storing data in program memory allows 2 bytes of data to be stored in each instruction location.

Lookup table data may be stored as 2 bytes per program word by using table reads and writes. The table pointer (TBLPTR) specifies the byte address and the table latch (TABLAT) contains the data that is read from, or written to, program memory. Data is transferred to/from program memory one byte at a time.

A description of the Table Read/Table Write operation is shown in Section 5.0.

EXAMPLE 4-3: TWO WORD INSTRUCTIONS

CASE 1:			
Object Code	**Source Code**		
0110 0110 0000 0000	TSTFSZ	REG1	; is RAM location 0?
1100 0001 0010 0011	MOVFF	REG1, REG2	; No, execute 2-word instruction
1111 0100 0101 0110			; 2nd operand holds address of REG2
0010 0100 0000 0000	ADDWF	REG3	; continue code
CASE 2:			
Object Code	**Source Code**		
0110 0110 0000 0000	TSTFSZ	REG1	; is RAM location 0?
1100 0001 0010 0011	MOVFF	REG1, REG2	; Yes
1111 0100 0101 0110			; 2nd operand becomes NOP
0010 0100 0000 0000	ADDWF	REG3	; continue code

4.9 Data Memory Organization

The data memory is implemented as static RAM. Each register in the data memory has a 12-bit address, allowing up to 4096 bytes of data memory. Figure 4-4 shows the data memory organization for the PIC18CXX8 devices.

The data memory map is divided into as many as 16 banks that contain 256 bytes each. The lower 4 bits of the Bank Select Register (BSR<3:0>) select which bank will be accessed. The upper 4 bits for the BSR are not implemented.

The data memory contains Special Function Registers (SFR) and General Purpose Registers (GPR). The SFR's are used for control and status of the controller and peripheral functions, while GPR's are used for data storage and scratch pad operations in the user's application. The SFR's start at the last location of Bank 15 (0xFFF) and grow downwards. GPR's start at the first location of Bank 0 and grow upwards. Any read of an unimplemented location will read as '0's.

The entire data memory may be accessed directly or indirectly. Direct addressing may require the use of the BSR register. Indirect addressing requires the use of the File Select Register (FSR). Each FSR holds a 12-bit address value that can be used to access any location in the Data Memory map without banking.

The instruction set and architecture allow operations across all banks. This may be accomplished by indirect addressing or by the use of the `MOVFF` instruction. The `MOVFF` instruction is a two word/two cycle instruction that moves a value from one register to another.

To ensure that commonly used registers (SFR's and select GPR's) can be accessed in a single cycle, regardless of the current BSR values, an Access Bank is implemented. A segment of Bank 0 and a segment of Bank 15 comprise the Access RAM. Section 4.10 provides a detailed description of the Access RAM.

4.9.1 GENERAL PURPOSE REGISTER FILE

The register file can be accessed either directly or indirectly. Indirect addressing operates through the File Select Registers (FSR). The operation of indirect addressing is shown in Section 4.12.

Enhanced MCU devices may have banked memory in the GPR area. GPR's are not initialized by a Power-on Reset and are unchanged on all other RESETS.

Data RAM is available for use as GPR registers by all instructions. Bank 15 (0xF00 to 0xFFF) contains SFR's. All other banks of data memory contain GPR registers starting with bank 0.

4.9.2 SPECIAL FUNCTION REGISTERS

The Special Function Registers (SFR's) are registers used by the CPU and Peripheral Modules for controlling the desired operation of the device. These registers are implemented as static RAM. A list of these registers is given in Table 4-2.

The SFR's can be classified into two sets: those associated with the "core" function and those related to the peripheral functions. Those registers related to the "core" are described in this section, while those related to the operation of the peripheral features are described in the section of that peripheral feature.

The SFR's are typically distributed among the peripherals whose functions they control.

The unused SFR locations will be unimplemented and read as '0's. See Table 4-2 for addresses for the SFR's.

TABLE 4-2: SPECIAL FUNCTION REGISTER MAP

Address	Name	Address	Name	Address	Name	Address	Name
FFFh	TOSU	FDFh	INDF2(2)	FBFh	CCPR1H	F9Fh	IPR1
FFEh	TOSH	FDEh	POSTINC2(2)	FBEh	CCPR1L	F9Eh	PIR1
FFDh	TOSL	FDDh	POSTDEC2(2)	FBDh	CCP1CON	F9Dh	PIE1
FFCh	STKPTR	FDCh	PREINC2(2)	FBCh	CCPR2H	F9Ch	—
FFBh	PCLATU	FDBh	PLUSW2(2)	FBBh	CCPR2L	F9Bh	—
FFAh	PCLATH	FDAh	FSR2H	FBAh	CCP2CON	F9Ah	TRISJ(5)
FF9h	PCL	FD9h	FSR2L	FB9h	—	F99h	TRISH(5)
FF8h	TBLPTRU	FD8h	STATUS	FB8h	—	F98h	TRISG
FF7h	TBLPTRH	FD7h	TMR0H	FB7h	—	F97h	TRISF
FF6h	TBLPTRL	FD6h	TMR0L	FB6h	—	F96h	TRISE
FF5h	TABLAT	FD5h	T0CON	FB5h	CVRCON	F95h	TRISD
FF4h	PRODH	FD4h	—	FB4h	CMCON	F94h	TRISC
FF3h	PRODL	FD3h	OSCCON	FB3h	TMR3H	F93h	TRISB
FF2h	INTCON	FD2h	LVDCON	FB2h	TMR3L	F92h	TRISA
FF1h	INTCON2	FD1h	WDTCON	FB1h	T3CON	F91h	LATJ(5)
FF0h	INTCON3	FD0h	RCON	FB0h	PSPCON	F90h	LATH(5)
FEFh	INDF0(2)	FCFh	TMR1H	FAFh	SPBRG	F8Fh	LATG
FEEh	POSTINC0(2)	FCEh	TMR1L	FAEh	RCREG	F8Eh	LATF
FEDh	POSTDEC0(2)	FCDh	T1CON	FADh	TXREG	F8Dh	LATE
FECh	PREINC0(2)	FCCh	TMR2	FACh	TXSTA	F8Ch	LATD
FEBh	PLUSW0(2)	FCBh	PR2	FABh	RCSTA	F8Bh	LATC
FEAh	FSR0H	FCAh	T2CON	FAAh	—	F8Ah	LATB
FE9h	FSR0L	FC9h	SSPBUF	FA9h	—	F89h	LATA
FE8h	WREG	FC8h	SSPADD	FA8h	—	F88h	PORTJ(5)
FE7h	INDF1(2)	FC7h	SSPSTAT	FA7h	—	F87h	PORTH(5)
FE6h	POSTINC1(2)	FC6h	SSPCON1	FA6h	—	F86h	PORTG
FE5h	POSTDEC1(2)	FC5h	SSPCON2	FA5h	IPR3	F85h	PORTF
FE4h	PREINC1(2)	FC4h	ADRESH	FA4h	PIR3	F84h	PORTE
FE3h	PLUSW1(2)	FC3h	ADRESL	FA3h	PIE3	F83h	PORTD
FE2h	FSR1H	FC2h	ADCON0	FA2h	IPR2	F82h	PORTC
FE1h	FSR1L	FC1h	ADCON1	FA1h	PIR2	F81h	PORTB
FE0h	BSR	FC0h	ADCON2	FA0h	PIE2	F80h	PORTA

Note 1: Unimplemented registers are read as '0'.

2: This is not a physical register.

3: Contents of register is dependent on WIN2:WIN0 bits in CANCON register.

4: CANSTAT register is repeated in these locations to simplify application firmware. Unique names are given for each instance of the CANSTAT register due to the Microchip Header file requirement.

5: Available on PIC18C858 only.

4.12 Indirect Addressing, INDF and FSR Registers

Indirect addressing is a mode of addressing data memory, where the data memory address in the instruction is not fixed. A SFR register is used as a pointer to the data memory location that is to be read or written. Since this pointer is in RAM, the contents can be modified by the program. This can be useful for data tables in the data memory and for software stacks. Figure 4-6 shows the operation of indirect addressing. This shows the moving of the value to the data memory address specified by the value of the FSR register.

Indirect addressing is possible by using one of the INDF registers. Any instruction using the INDF register actually accesses the register indicated by the File Select Register, FSR. Reading the INDF register itself indirectly (FSR = '0') will read 00h. Writing to the INDF register indirectly results in a no-operation. The FSR register contains a 12-bit address, which is shown in Figure 4-6.

The INDFn ($0 \le n \le 2$) register is not a physical register. Addressing INDFn actually addresses the register whose address is contained in the FSRn register (FSRn is a pointer). This is indirect addressing.

Example 4-4 shows a simple use of indirect addressing to clear the RAM in Bank 1 (locations 100h-1FFh) in a minimum number of instructions.

EXAMPLE 4-4: HOW TO CLEAR RAM (BANK 1) USING INDIRECT ADDRESSING

```
     LFSR     FSR0, 0x100  ;
NEXT CLRF     POSTINC0     ; Clear INDF
                           ; register
                           ; & inc pointer
     BTFSS    FSR0H, 1     ; All done
                           ; w/ Bank1?
     GOTO     NEXT         ; NO, clear next
CONTINUE                   ;
     :                     ; YES, continue
```

There are three indirect addressing registers. To address the entire data memory space (4096 bytes), these registers are 12-bit wide. To store the 12-bits of addressing information, two 8-bit registers are required. These indirect addressing registers are:

1. FSR0: composed of FSR0H:FSR0L
2. FSR1: composed of FSR1H:FSR1L
3. FSR2: composed of FSR2H:FSR2L

In addition, there are registers INDF0, INDF1 and INDF2, which are not physically implemented. Reading or writing to these registers activates indirect addressing, with the value in the corresponding FSR register being the address of the data.

If an instruction writes a value to INDF0, the value will be written to the address indicated by FSR0H:FSR0L. A read from INDF1 reads the data from the address indicated by FSR1H:FSR1L. INDFn can be used in code anywhere an operand can be used.

If INDF0, INDF1 or INDF2 are read indirectly via an FSR, all '0's are read (zero bit is set). Similarly, if INDF0, INDF1 or INDF2 are written to indirectly, the operation will be equivalent to a `NOP` instruction and the STATUS bits are not affected.

4.12.1 INDIRECT ADDRESSING OPERATION

Each FSR register has an INDF register associated with it, plus four additional register addresses. Performing an operation on one of these five registers determines how the FSR will be modified during indirect addressing.

When data access is done to one of the five INDFn locations, the address selected will configure the FSRn register to:

- Do nothing to FSRn after an indirect access (no change) - INDFn
- Auto-decrement FSRn after an indirect access (post-decrement) - POSTDECn
- Auto-increment FSRn after an indirect access (post-increment) - POSTINCn
- Auto-increment FSRn before an indirect access (pre-increment) - PREINCn
- Use the value in the WREG register as an offset to FSRn. Do not modify the value of the WREG or the FSRn register after an indirect access (no change) - PLUSWn

When using the auto-increment or auto-decrement features, the effect on the FSR is not reflected in the STATUS register. For example, if the indirect address causes the FSR to equal '0', the Z bit will not be set.

Incrementing or decrementing an FSR affects all 12 bits. That is, when FSRnL overflows from an increment, FSRnH will be incremented automatically.

Adding these features allows the FSRn to be used as a software stack pointer in addition to its uses for table operations in data memory.

Each FSR has an address associated with it that performs an indexed indirect access. When a data access to this INDFn location (PLUSWn) occurs, the FSRn is configured to add the 2's complement value in the WREG register and the value in FSR to form the address before an indirect access. The FSR value is not changed.

If an FSR register contains a value that indicates one of the INDFn, an indirect read will read 00h (zero bit is set), while an indirect write will be equivalent to a `NOP` (STATUS bits are not affected).

If an indirect addressing operation is done where the target address is an FSRnH or FSRnL register, the write operation will dominate over the pre- or post-increment/decrement functions.

8.2 PORTB, TRISB and LATB Registers

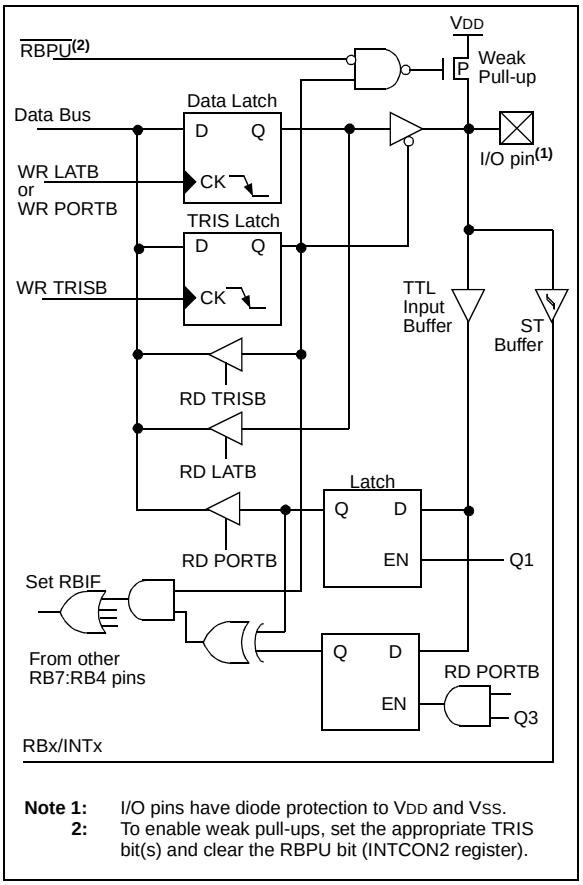
PORTB is an 8-bit wide bi-directional port. The corresponding Data Direction register is TRISB. Setting a TRISB bit (=1) will make the corresponding PORTB pin an input (i.e., put the corresponding output driver in a hi-impedance mode). Clearing a TRISB bit (=0) will make the corresponding PORTB pin an output (i.e., put the contents of the output latch on the selected pin).

Read-modify-write operations on the LATB register read and write the latched output value for PORTB.

EXAMPLE 8-2: INITIALIZING PORTB

```
CLRF    PORTB      ; Initialize PORTB by
                   ; clearing output
                   ; data latches
CLRF    LATB       ; Alternate method
                   ; to clear output
                   ; data latches
MOVLW   0xCF       ; Value used to
                   ; initialize data
                   ; direction
MOVWF   TRISB      ; Set RB3:RB0 as inputs
                   ; RB5:RB4 as outputs
                   ; RB7:RB6 as inputs
```

FIGURE 8-4: RB7:RB4 PINS BLOCK DIAGRAM



Note 1: I/O pins have diode protection to VDD and VSS.
2: To enable weak pull-ups, set the appropriate TRIS bit(s) and clear the RBPU bit (INTCON2 register).

Each of the PORTB pins has a weak internal pull-up. A single control bit can turn on all the pull-ups. This is performed by clearing bit $\overline{\text{RBPU}}$ (INTCON2 register). The weak pull-up is automatically turned off when the port pin is configured as an output. The pull-ups are disabled on a Power-on Reset.

Four of PORTB's pins, RB7:RB4, have an interrupt-on-change feature. Only pins configured as inputs can cause this interrupt to occur (i.e., any RB7:RB4 pin configured as an output is excluded from the interrupt-on-change comparison). The input pins (of RB7:RB4) are compared with the old value latched on the last read of PORTB. The "mismatch" outputs of RB7:RB4 are OR'd together to generate the RB Port Change Interrupt with flag bit RBIF (INTCON register).

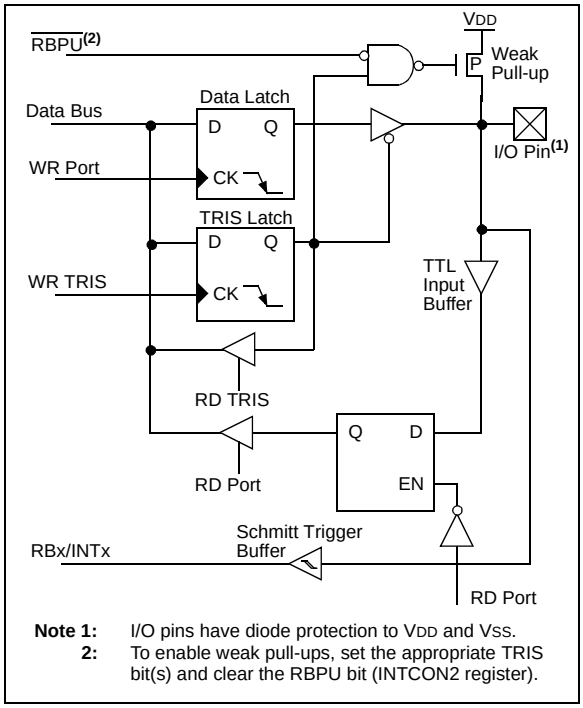
This interrupt can wake the device from SLEEP. The user, in the Interrupt Service Routine, can clear the interrupt in the following manner:

a) Any read or write of PORTB (except with the MOVFF instruction). This will end the mismatch condition.
b) Clear flag bit RBIF.

A mismatch condition will continue to set flag bit RBIF. Reading PORTB will end the mismatch condition and allow flag bit RBIF to be cleared.

The interrupt-on-change feature is recommended for wake-up on key depression operation and operations where PORTB is only used for the interrupt on change feature. Polling of PORTB is not recommended while using the interrupt-on-change feature.

FIGURE 8-5: RB3:RB0 PINS BLOCK DIAGRAM



Note 1: I/O pins have diode protection to VDD and VSS.
2: To enable weak pull-ups, set the appropriate TRIS bit(s) and clear the RBPU bit (INTCON2 register).

13.2 Timer1 Oscillator

The Timer1 oscillator may be used as the clock source for Timer3. The Timer1 oscillator is enabled by setting the T1OSCEN bit (T1CON Register). The oscillator is a low power oscillator rated up to 200 kHz. Refer to "Timer1 Module", Section 11.0 for Timer1 oscillator details.

13.3 Timer3 Interrupt

The TMR3 Register pair (TMR3H:TMR3L) increments from 0000h to FFFFh and rolls over to 0000h. The TMR3 Interrupt, if enabled, is generated on overflow which is latched in interrupt flag bit TMR3IF (PIR Registers). This interrupt can be enabled/disabled by setting/clearing TMR3 interrupt enable bit TMR3IE (PIE Registers).

13.4 Resetting Timer3 Using a CCP Trigger Output

If the CCP module is configured in Compare mode to generate a "special event trigger" (CCP1M3:CCP1M0 = 1011), this signal will reset Timer3.

Note:	The special event triggers from the CCP module will not set interrupt flag bit TMR3IF (PIR registers).

Timer3 must be configured for either timer or Synchronized Counter mode to take advantage of this feature. If Timer3 is running in Asynchronous Counter mode, this RESET operation may not work. In the event that a write to Timer3 coincides with a special event trigger from CCP1, the write will take precedence. In this mode of operation, the CCPR1H:CCPR1L registers pair becomes the period register for Timer3. Refer to "Capture/Compare/PWM (CCP) Modules", Section 14.0 for CCP details.

TABLE 13-1: REGISTERS ASSOCIATED WITH TIMER3 AS A TIMER/COUNTER

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 000x	0000 000u
PIR2	—	CMIF	—	—	BCLIF	LVDIF	TMR3IF	CCP2IF	-0-- 0000	-0-- 0000
PIE2	—	CMIE	—	—	BCLIE	LVDIE	TMR3IE	CCP2IE	-0-- 0000	-0-- 0000
IPR2	—	CMIP	—	—	BCLIP	LVDIP	TMR3IP	CCP2IP	-0-- 0000	-0-- 0000
TMR3L	Holding register for the Least Significant Byte of the 16-bit TMR3 register								xxxx xxxx	uuuu uuuu
TMR3H	Holding register for the Most Significant Byte of the 16-bit TMR3 register								xxxx xxxx	uuuu uuuu
T1CON	RD16	—	T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON	0-00 0000	u-uu uuuu
T3CON	RD16	T3CCP2	T3CKPS1	T3CKPS0	T3CCP1	$\overline{\text{T3SYNC}}$	TMR3CS	TMR3ON	0000 0000	uuuu uuuu

Legend: x = unknown, u = unchanged, - = unimplemented, read as '0'. Shaded cells are not used by the Timer1 module.

15.4.16.2 Bus Collision During a Repeated START Condition

During a Repeated START condition, a bus collision occurs if:

a) A low level is sampled on SDA when SCL goes from low level to high level.

b) SCL goes low before SDA is asserted low, indicating that another master is attempting to transmit a data '1'.

When the user de-asserts SDA and the pin is allowed to float high, the BRG is loaded with SSPADD<6:0> and counts down to 0. The SCL pin is then de-asserted, and when sampled high, the SDA pin is sampled.

If SDA is low, a bus collision has occurred (i.e, another master is attempting to transmit a data '0', see Figure 15-24). If SDA is sampled high, the BRG is reloaded and begins counting. If SDA goes from high to low before the BRG times out, no bus collision occurs because no two masters can assert SDA at exactly the same time.

If SCL goes from high to low before the BRG times out and SDA has not already been asserted, a bus collision occurs. In this case, another master is attempting to transmit a data '1' during the Repeated START condition (Figure 15-25).

If at the end of the BRG time-out both SCL and SDA are still high, the SDA pin is driven low and the BRG is reloaded and begins counting. At the end of the count, regardless of the status of the SCL pin, the SCL pin is driven low and the Repeated START condition is complete.

FIGURE 15-24: BUS COLLISION DURING A REPEATED START CONDITION (CASE 1)

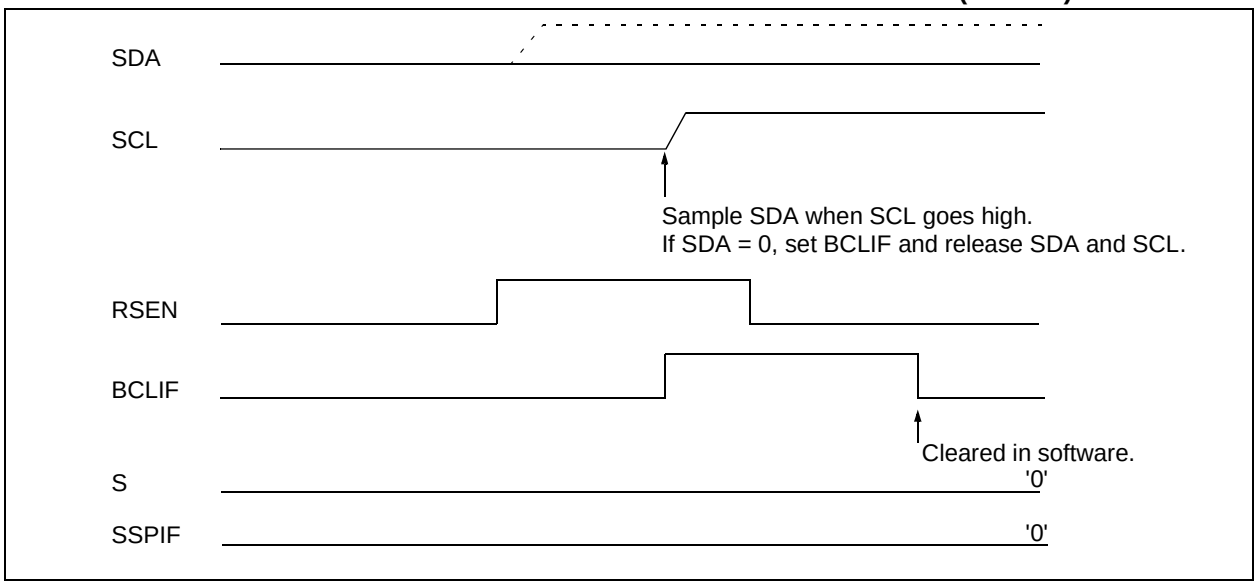
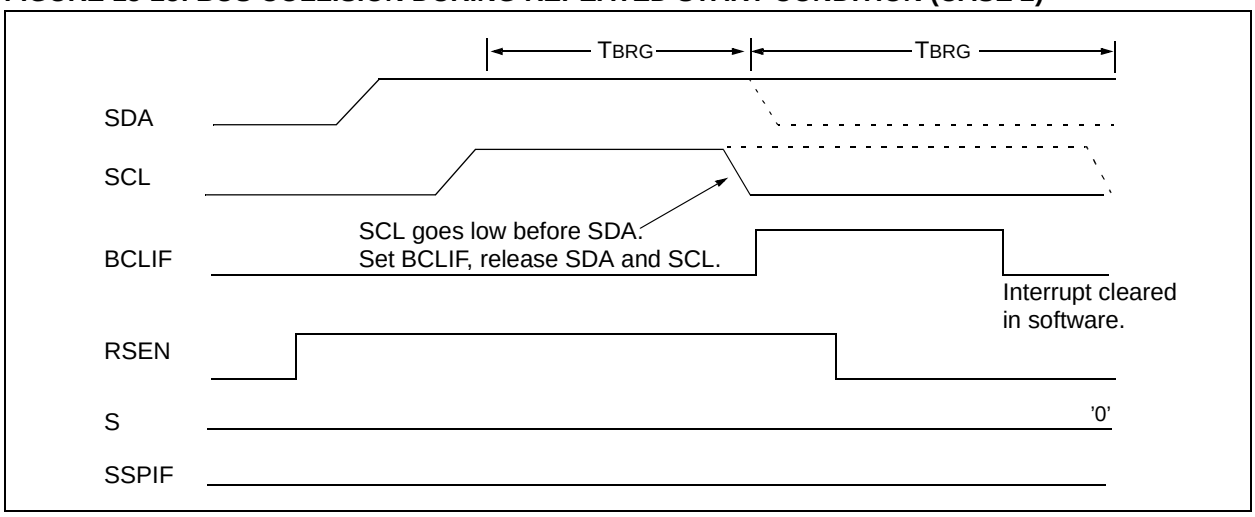


FIGURE 15-25: BUS COLLISION DURING REPEATED START CONDITION (CASE 2)



15.4.16.3 Bus Collision During a STOP Condition

Bus collision occurs during a STOP condition if:

a) After the SDA pin has been de-asserted and allowed to float high, SDA is sampled low after the BRG has timed out.

b) After the SCL pin is de-asserted, SCL is sampled low before SDA goes high.

The STOP condition begins with SDA asserted low. When SDA is sampled low, the SCL pin is allowed to float. When the pin is sampled high (clock arbitration), the baud rate generator is loaded with SSPADD<6:0> and counts down to 0. After the BRG times out, SDA is sampled. If SDA is sampled low, a bus collision has occurred. This is due to another master attempting to drive a data '0' (Figure 15-26). If the SCL pin is sampled low before SDA is allowed to float high, a bus collision occurs. This is another case of another master attempting to drive a data '0' (Figure 15-27).

FIGURE 15-26: BUS COLLISION DURING A STOP CONDITION (CASE 1)

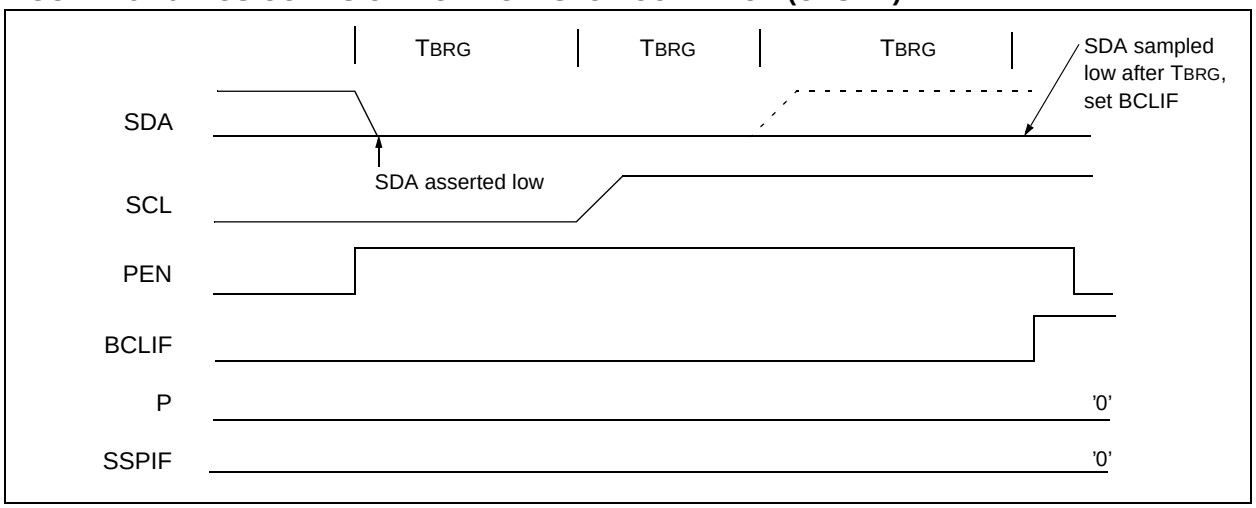
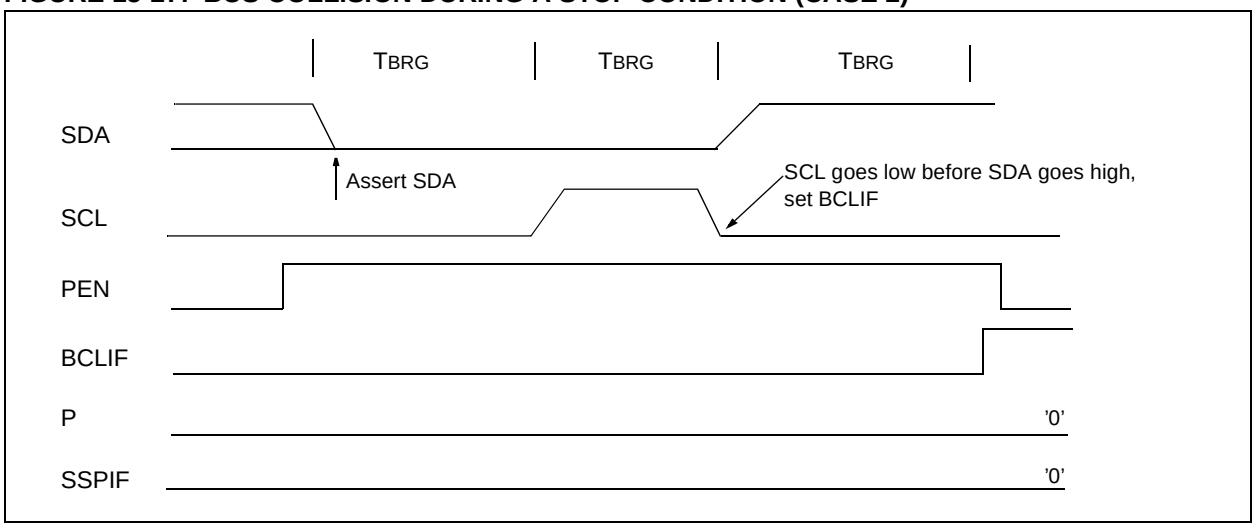


FIGURE 15-27: BUS COLLISION DURING A STOP CONDITION (CASE 2)



REGISTER 17-13: RXB1CON – RECEIVE BUFFER 1 CONTROL REGISTER

R/C-0	R/W-0	R/W-0	U-0	R-0	R-0	R-0	R-0
RXFUL	RXM1	RXM0	—	RXRTRRO	FILHIT2	FILHIT1	FILHIT0
bit 7							bit 0

bit 7 **RXFUL:** Receive Full Status bit
1 = Receive buffer contains a received message
0 = Receive buffer is open to receive a new message

Note: This bit is set by the CAN module and should be cleared by software after the buffer is read.

bit 6-5 **RXM1:RXM0:** Receive Buffer Mode bits
11 = Receive all messages (including those with errors)
10 = Receive only valid messages with extended identifier
01 = Receive only valid messages with standard identifier
00 = Receive all valid messages

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

bit 3 **RXRTRRO:** Receive Remote Transfer Request bit (read only)
1 = Remote transfer request
0 = No remote transfer request

bit 2-0 **FILHIT2:FILHIT0:** Filter Hit bits
These bits indicate which acceptance filter enabled the last message reception into Receive Buffer 1.
111 = Reserved
110 = Reserved
101 = Acceptance Filter 5 (RXF5)
100 = Acceptance Filter 4 (RXF4)
011 = Acceptance Filter 3 (RXF3)
010 = Acceptance Filter 2 (RXF2)
001 = Acceptance Filter 1 (RXF1) only possible when RXB0DBEN bit is set
000 = Acceptance Filter 0 (RXF0) only possible when RXB0DBEN bit is set

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 17-14: RXBnSIDH – RECEIVE BUFFER n STANDARD IDENTIFIER HIGH BYTE REGISTER

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
SID10	SID9	SID8	SID7	SID6	SID5	SID4	SID3
bit 7							bit 0

bit 7-0 **SID10:SID3:** Standard Identifier bits, if EXID = 0 (RXBnSIDL Register).
Extended Identifier bits EID28:EID21, if EXID = 1.

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

17.2.6 CAN MODULE I/O CONTROL REGISTER

This subsection describes the CAN Module I/O Control register.

REGISTER 17-32: CIOCON – CAN I/O CONTROL REGISTER

R/W-0	R/W-0	R/W-0	R/W-0	U-0	U-0	U-0	U-0
TX1SRC	TX1EN	ENDRHI	CANCAP	—	—	—	—
bit 7							bit 0

bit 7 **TX1SRC**: CAN TX1 Pin Data Source
1 = CAN TX1 pin will output the CAN clock
0 = CAN TX1 pin will output $\overline{\text{TXD}}$

bit 6 **TX1EN:** CAN TX1 Pin Enable
1 = CAN TX1 pin will output $\overline{\text{TXD}}$ or CAN clock
0 = CAN TX1 pin will have digital I/O function

bit 5 **ENDRHI:** Enable Drive High
1 = CAN TX0, CAN TX1 pins will drive V_{DD} when recessive
0 = CAN TX0, CAN TX1 pins will tri-state when recessive

bit 4 **CANCAP:** CAN Message Receive Capture Enable
1 = Enable CAN capture
0 = Disable CAN capture

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

17.2.7 CAN INTERRUPT REGISTERS

REGISTER 17-33: PIR3 – PERIPHERAL INTERRUPT FLAG 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
IRXIF	WAKIF	ERRIF	TXB2IF	TXB1IF	TXB0IF	RXB1IF	RXB0IF
bit 7							bit 0

bit 7 **IRXIF:** CAN Invalid Received Message Interrupt Flag bit
1 = An invalid message has occurred on the CAN bus
0 = No invalid message on CAN bus

bit 6 **WAKIF:** CAN Bus Activity Wake-up Interrupt Flag bit
1 = Activity on CAN bus has occurred
0 = No activity on CAN bus

bit 5 **ERRIF:** CAN Bus Error Interrupt Flag bit
1 = An error has occurred in the CAN module (multiple sources)
0 = No CAN module errors

bit 4 **TXB2IF:** CAN Transmit Buffer 2 Interrupt Flag bit
1 = Transmit Buffer 2 has completed transmission of a message, and may be re-loaded
0 = Transmit Buffer 2 has not completed transmission of a message

bit 3 **TXB1IF:** CAN Transmit Buffer 1 Interrupt Flag bit
1 = Transmit Buffer 1 has completed transmission of a message, and may be re-loaded
0 = Transmit Buffer 1 has not completed transmission of a message

bit 2 **TXB0IF:** CAN Transmit Buffer 0 Interrupt Flag bit
1 = Transmit Buffer 0 has completed transmission of a message, and may be re-loaded
0 = Transmit Buffer 0 has not completed transmission of a message

bit 1 **RXB1IF:** CAN Receive Buffer 1 Interrupt Flag bit
1 = Receive Buffer 1 has received a new message
0 = Receive Buffer 1 has not received a new message

bit 0 **RXB0IF:** CAN Receive Buffer 0 Interrupt Flag bit
1 = Receive Buffer 0 has received a new message
0 = Receive Buffer 0 has not received a new message

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

17.6 Message Acceptance Filters and Masks

The Message Acceptance Filters and Masks are used to determine if a message in the message assembly buffer should be loaded into either of the receive buffers. Once a valid message has been received into the MAB, the identifier fields of the message are compared to the filter values. If there is a match, that message will be loaded into the appropriate receive buffer. The filter masks are used to determine which bits in the identifier are examined with the filters. A truth table is shown below in Table 17-2 that indicates how each bit in the identifier is compared to the masks and filters to determine if a the message should be loaded into a receive buffer. The mask essentially determines which bits to apply the acceptance filters to. If any mask bit is set to a zero, then that bit will automatically be accepted, regardless of the filter bit.

TABLE 17-2: FILTER/MASK TRUTH TABLE

Mask bit n	Filter bit n	Message Identifier bit n001	Accept or Reject bit n
0	X	X	Accept
1	0	0	Accept
1	0	1	Reject
1	1	0	Reject
1	1	1	Accept

Legend: X = don't care

As shown in the Receive Buffers Block Diagram (Figure 17-3), acceptance filters RXF0 and RXF1, and filter mask RXM0 are associated with RXB0. Filters RXF2, RXF3, RXF4, and RXF5 and mask RXM1 are associated with RXB1. When a filter matches and a message is loaded into the receive buffer, the filter number that enabled the message reception is loaded into the FILHIT bit(s). For RXB1, the RXB1CON register contains the FILHIT<2:0> bits. They are coded as follows:

- 101 = Acceptance Filter 5 (RXF5)
- 100 = Acceptance Filter 4 (RXF4)
- 011 = Acceptance Filter 3 (RXF3)
- 010 = Acceptance Filter 2 (RXF2)
- 001 = Acceptance Filter 1 (RXF1)
- 000 = Acceptance Filter 0 (RXF0)

Note:	000 and 001 can only occur if the RXB0DBEN bit is set in the RXB0CON register, allowing RXB0 messages to roll over into RXB1.

The coding of the RXB0DBEN bit enables these three bits to be used similarly to the FILHIT bits and to distinguish a hit on filter RXF0 and RXF1, in either RXB0, or after a roll over into RXB1.

- 111 = Acceptance Filter 1 (RXF1)
- 110 = Acceptance Filter 0 (RXF0)
- 001 = Acceptance Filter 1 (RXF1)
- 000 = Acceptance Filter 0

If the RXB0DBEN bit is clear, there are six codes corresponding to the six filters. If the RXB0DBEN bit is set, there are six codes corresponding to the six filters, plus two additional codes corresponding to RXF0 and RXF1 filters that roll over into RXB1.

If more than one acceptance filter matches, the FILHIT bits will encode the binary value of the lowest numbered filter that matched. In other words, if filter RXF2 and filter RXF4 match, FILHIT will be loaded with the value for RXF2. This essentially prioritizes the acceptance filters with a lower number filter having higher priority. Messages are compared to filters in ascending order of filter number.

The mask and filter registers can only be modified when the PIC18CXX8 is in Configuration mode. The mask and filter registers cannot be read outside of Configuration mode. When outside of Configuration mode, all mask and filter registers will be read as '0'.

REGISTER 22-5: CONFIGURATION REGISTER 4 LOW (CONFIG4L: BYTE ADDRESS 0x300006)

U-0	U-0	U-0	U-0	U-0	U-0	R/P-1	R/P-1
—	—	—	—	—	—	Reserved	STVREN
bit 7							bit 0

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

bit 1 **Reserved:** Maintain this bit set

bit 0 **STVREN:** Stack Full/Underflow RESET Enable bit
1 = Stack Full/Underflow will cause RESET
0 = Stack Full/Underflow will not cause RESET

Legend:		
R = Readable bit	P = Programmable bit	U = Unimplemented bit, read as ‘0’
- n = Value when device is unprogrammed		u = Unchanged from programmed state

RCALL **Relative Call**

Syntax: [*label*] RCALL n

Operands: -1024 ≤ n ≤ 1023

Operation: (PC) + 2 → TOS,
(PC) + 2 + 2n → PC

Status Affected: None

Encoding:

1101	1nnn	nnnn	nnnn

Description: Subroutine call with a jump up to 1K from the current location. First, return address (PC+2) is pushed onto the stack. Then, add the 2's complement number '2n' to the PC. Since the PC will have incremented to fetch the next instruction, the new address will be PC+2+2n. This instruction is a two-cycle instruction.

Words: 1

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'n' Push PC to stack	Process Data	Write to PC
No operation	No operation	No operation	No operation

Example: `HERE RCALL` Jump

Before Instruction
PC = Address `(HERE)`

After Instruction
PC = Address `(Jump)`
TOS = Address `(HERE+2)`

RESET **Reset**

Syntax: [*label*] RESET

Operands: None

Operation: Reset all registers and flags that are affected by a MCLR Reset.

Status Affected: All

Encoding:

0000	0000	1111	1111

Description: This instruction provides a way to execute a MCLR Reset in software.

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Start reset	No operation	No operation

Example: `RESET`

After Instruction
Registers = Reset Value
Flags* = Reset Value

RLNCF **Rotate Left f (no carry)**

Syntax: [*label*] RLNCF f [,d [,a]]

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

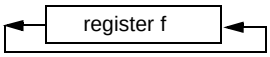
Operation: (f<n>) → dest<n+1>,
(f<7>) → dest<0>

Status Affected: N,Z

Encoding:

0100	01da	ffff	ffff

Description: The contents of register 'f' are rotated one bit to the left. If 'd' is 0 the result is placed in WREG. If 'd' is 1, the result is stored back in register 'f' (default). If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' is 1, the Bank will be selected as per the BSR value.



Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example: RLNCF REG

Before Instruction
REG = 1010 1011
N = ?
Z = ?

After Instruction
REG = 0101 0111
N = 0
Z = 0

RRCF **Rotate Right f through Carry**

Syntax: [*label*] RRCF f [,d [,a]]

Operands: $0 \le f \le 255$
$d \in [0,1]$
$a \in [0,1]$

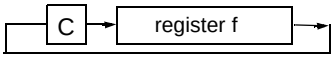
Operation: (f<n>) → dest<n-1>,
(f<0>) → C,
(C) → dest<7>

Status Affected: C,N,Z

Encoding:

0011	00da	ffff	ffff

Description: The contents of register 'f' are rotated one bit to the right through the Carry Flag. If 'd' is 0, the result is placed in WREG. If 'd' is 1, the result is placed back in register 'f' (default). If 'a' is 0, the Access Bank will be selected, overriding the BSR value. If 'a' is 1, the Bank will be selected as per the BSR value.



Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Example: RRCF REG, W

Before Instruction
REG = 1110 0110
C = 0
N = ?
Z = ?

After Instruction
REG = 1110 0110
WREG = 0111 0011
C = 0
N = 0
Z = 0

25.3 AC (Timing) Characteristics

25.3.1 TIMING PARAMETER SYMBOLOGY

The timing parameter symbols have been created following one of the following formats:

1. TppS2ppS
2. TppS
3. TCC:ST (I^2C specifications only)
4. Ts (I^2C specifications only)

T			
F	Frequency	T	Time

Lowercase letters (pp) and their meanings:

pp			
cc	CCP1	osc	OSC1
ck	CLKO	rd	$\overline{RD}$
cs	$\overline{CS}$	rw	$\overline{RD}$ or $\overline{WR}$
di	SDI	sc	SCK
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 (Hi-impedance)	V	Valid
L	Low	Z	Hi-impedance
I^2C only			
AA	output access	High	High
BUF	Bus free	Low	Low

TCC:ST (I^2C specifications only)

CC			
HD	Hold	SU	Setup
ST			
DAT	DATA input hold	STO	STOP condition
STA	START condition		

25.3.2 TIMING CONDITIONS

The temperature and voltages specified in Table 25-3 apply to all timing specifications, unless otherwise noted. Figure 25-4 specifies the load conditions for the timing specifications.

TABLE 25-3: 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 25.1. LC parts operate for industrial temperatures only.

FIGURE 25-4: LOAD CONDITIONS FOR DEVICE TIMING SPECIFICATIONS

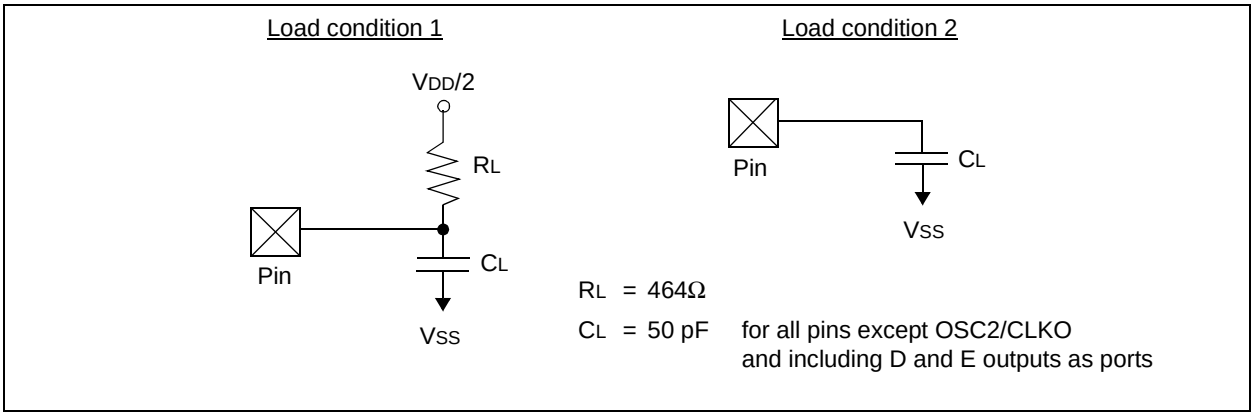
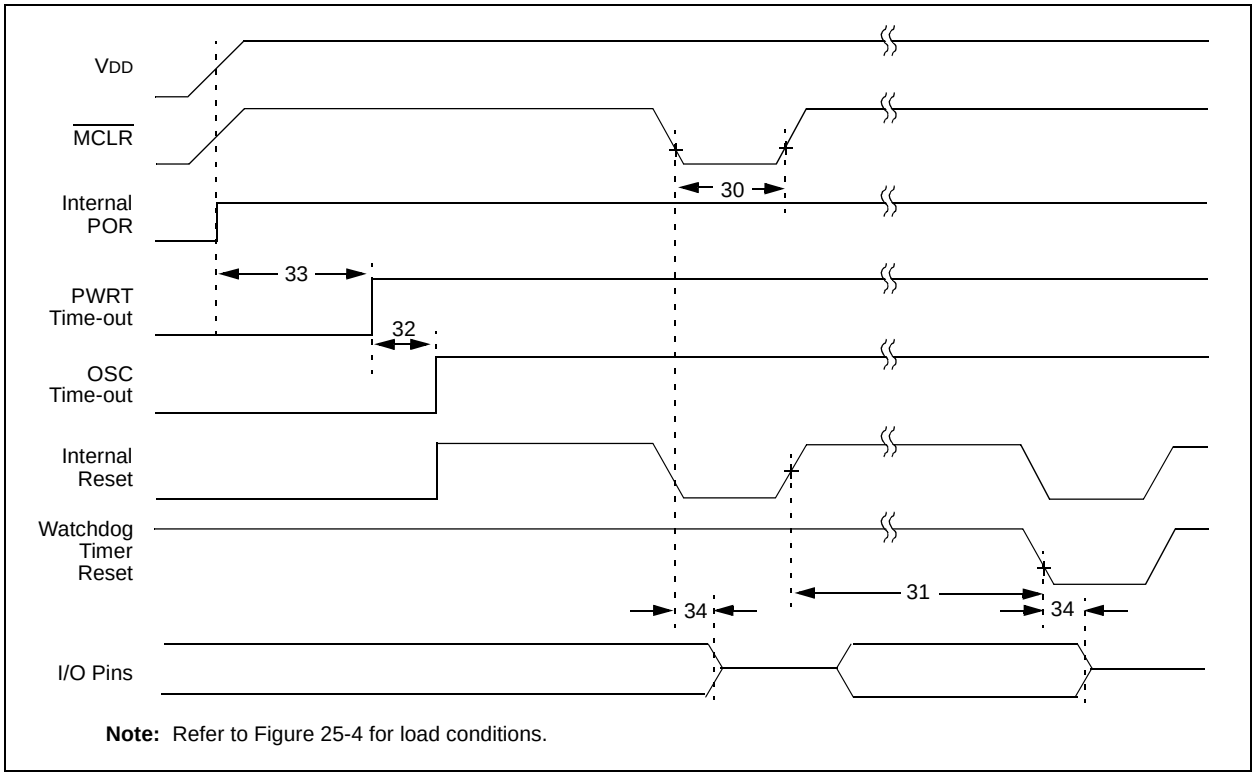


FIGURE 25-7: RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER AND POWER-UP TIMER TIMING



Note: Refer to Figure 25-4 for load conditions.

FIGURE 25-8: BROWN-OUT RESET TIMING

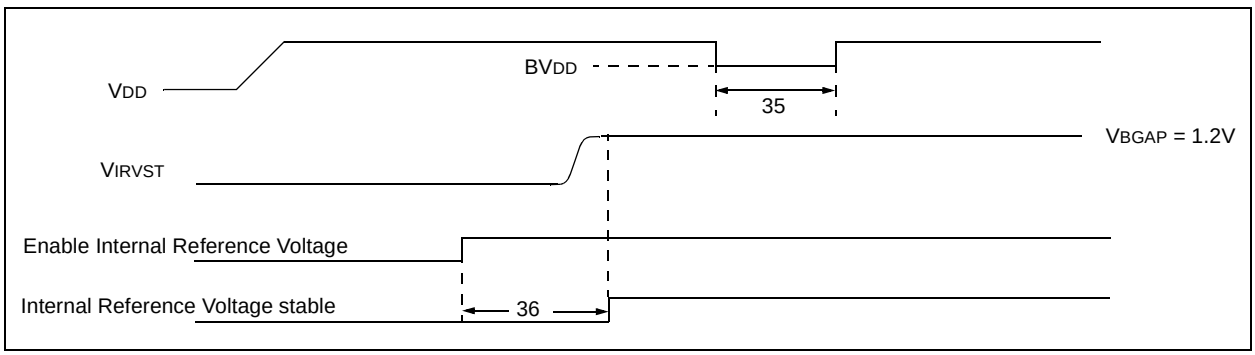
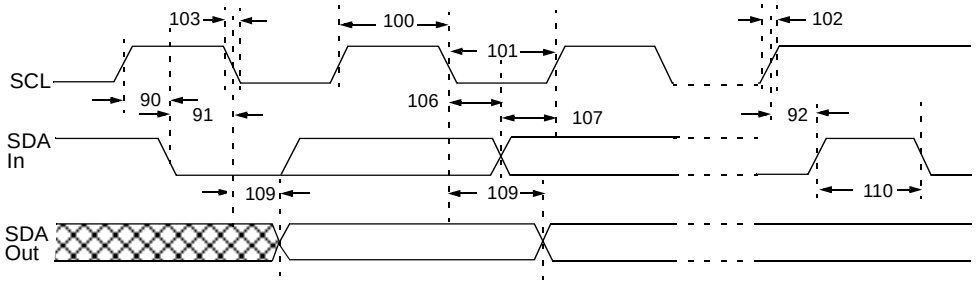


TABLE 25-7: RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER, POWER-UP TIMER AND BROWN-OUT RESET REQUIREMENTS

Param. No.	Symbol	Characteristic	Min	Typ	Max	Units	Conditions
30	TmcL	$\overline{MCLR}$ Pulse Width (low)	2	—	—	µs	
31	TWDT	Watchdog Timer Time-out Period (No Prescaler)	7	18	33	ms	
32	TOST	Oscillation Start-up Timer Period	1024TOSC	—	1024TOSC	—	TOSC = OSC1 period
33	TPWRT	Power up Timer Period	28	72	132	ms	
34	TIOZ	I/O Hi-impedance from $\overline{MCLR}$ Low or Watchdog Timer Reset	—	2	—	µs	
35	TBOR	Brown-out Reset Pulse Width	200	—	—	µs	VDD ≤ BVDD (See D005)
36	TIVRST	Time for Internal Reference Voltage to become stable	—	20	50	µs	

FIGURE 25-19: MASTER SSP I²C BUS DATA TIMING



Note: Refer to Figure 25-4 for load conditions.

TABLE 25-18: MASTER SSP I²C BUS DATA REQUIREMENTS

Param. No.	Symbol	Characteristic		Min	Max	Units	Conditions
100	THIGH	Clock high time	100 kHz mode	2(TOSC)(BRG + 1)	—	ms	
			400 kHz mode	2(TOSC)(BRG + 1)	—	ms	
			1 MHz mode(1)	2(TOSC)(BRG + 1)	—	ms	
101	TLOW	Clock low time	100 kHz mode	2(TOSC)(BRG + 1)	—	ms	
			400 kHz mode	2(TOSC)(BRG + 1)	—	ms	
			1 MHz mode(1)	2(TOSC)(BRG + 1)	—	ms	
102	TR	SDA and SCL rise time	100 kHz mode	—	1000	ns	Cb is specified to be from 10 to 400 pF
			400 kHz mode	20 + 0.1Cb	300	ns	
			1 MHz mode(1)	—	300	ns	
103	TF	SDA and SCL fall time	100 kHz mode	—	300	ns	Cb is specified to be from 10 to 400 pF
			400 kHz mode	20 + 0.1Cb	300	ns	
			1 MHz mode(1)	—	100	ns	
90	TSU:STA	START condition setup time	100 kHz mode	2(TOSC)(BRG + 1)	—	ms	Only relevant for Repeated START condition
			400 kHz mode	2(TOSC)(BRG + 1)	—	ms	
			1 MHz mode(1)	2(TOSC)(BRG + 1)	—	ms	
91	THD:STA	START condition hold time	100 kHz mode	2(TOSC)(BRG + 1)	—	ms	After this period the first clock pulse is generated
			400 kHz mode	2(TOSC)(BRG + 1)	—	ms	
			1 MHz mode(1)	2(TOSC)(BRG + 1)	—	ms	
106	THD:DAT	Data input hold time	100 kHz mode	0	—	ns	
			400 kHz mode	0	0.9	ms	
			1 MHz mode(1)	TBD	—	ns	
107	TSU:DAT	Data input setup time	100 kHz mode	250	—	ns	**(Note 2)**
			400 kHz mode	100	—	ns	
			1 MHz mode(1)	TBD	—	ns	
92	TSU:STO	STOP condition setup time	100 kHz mode	2(TOSC)(BRG + 1)	—	ms	
			400 kHz mode	2(TOSC)(BRG + 1)	—	ms	
			1 MHz mode(1)	2(TOSC)(BRG + 1)	—	ms	
109	TAA	Output valid from clock	100 kHz mode	—	3500	ns	
			400 kHz mode	—	1000	ns	
			1 MHz mode(1)	—	—	ns	
110	TBUF	Bus free time	100 kHz mode	4.7	—	ms	Time the bus must be free before a new transmission can start
			400 kHz mode	1.3	—	ms	
			1 MHz mode(1)	TBD	—	ms	
D102	Cb	Bus capacitive loading		—	400	pF	

Note 1: Maximum pin capacitance = 10 pF for all I²C pins.

2: A fast mode I²C bus device can be used in a standard mode I²C bus system, but parameter #107 ≥ 250 ns must then be met. This will automatically be the case if the device does not stretch the LOW period of the SCL signal. If such a device does stretch the LOW period of the SCL signal, it must output the next data bit to the SDA line. Before the SCL line is released, parameter #102+ parameter #107 = 1000 + 250 = 1250 ns (for 100 kHz mode).

