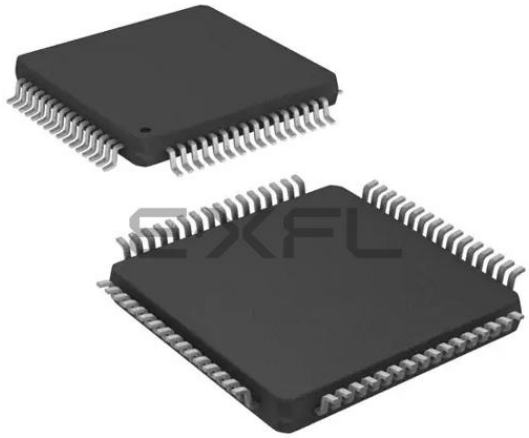


Welcome to E-XFL.COM

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	40MHz
Connectivity	CANbus, I²C, SPI, UART/USART
Peripherals	Brown-out Detect/Reset, LVD, POR, PWM, WDT
Number of I/O	52
Program Memory Size	64KB (32K x 16)
Program Memory Type	FLASH
EEPROM Size	1K x 8
RAM Size	3.25K 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	64-TQFP
Supplier Device Package	64-TQFP (10x10)
Purchase URL	https://www.e-xfl.com/product-detail/microchip-technology/pic18f6680t-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
FSR1H	PIC18F6X8X	PIC18F8X8X	---- xxxx	---- uuuu	---- uuuu
FSR1L	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
BSR	PIC18F6X8X	PIC18F8X8X	---- 0000	---- 0000	---- uuuu
INDF2	PIC18F6X8X	PIC18F8X8X	N/A	N/A	N/A
POSTINC2	PIC18F6X8X	PIC18F8X8X	N/A	N/A	N/A
POSTDEC2	PIC18F6X8X	PIC18F8X8X	N/A	N/A	N/A
PREINC2	PIC18F6X8X	PIC18F8X8X	N/A	N/A	N/A
PLUSW2	PIC18F6X8X	PIC18F8X8X	N/A	N/A	N/A
FSR2H	PIC18F6X8X	PIC18F8X8X	---- xxxx	---- uuuu	---- uuuu
FSR2L	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
STATUS	PIC18F6X8X	PIC18F8X8X	---x xxxx	---u uuuu	---u uuuu
TMR0H	PIC18F6X8X	PIC18F8X8X	0000 0000	uuuu uuuu	uuuu uuuu
TMR0L	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
T0CON	PIC18F6X8X	PIC18F8X8X	1111 1111	1111 1111	uuuu uuuu
OSCCON	PIC18F6X8X	PIC18F8X8X	---- 0000	---- 0000	---- uuuu
LVDCON	PIC18F6X8X	PIC18F8X8X	--00 0101	--00 0101	--uu uuuu
WDTCON	PIC18F6X8X	PIC18F8X8X	---- ---0	---- ---0	---- ---u
RCON(4)	PIC18F6X8X	PIC18F8X8X	0--q 11qq	0--q qquu	u--u qquu
TMR1H	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
TMR1L	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
T1CON	PIC18F6X8X	PIC18F8X8X	0-00 0000	u-uu uuuu	u-uu uuuu
TMR2	PIC18F6X8X	PIC18F8X8X	0000 0000	0000 0000	uuuu uuuu
PR2	PIC18F6X8X	PIC18F8X8X	1111 1111	1111 1111	1111 1111
T2CON	PIC18F6X8X	PIC18F8X8X	-000 0000	-000 0000	-uuu uuuu
SSPBUF	PIC18F6X8X	PIC18F8X8X	xxxx xxxx	uuuu uuuu	uuuu uuuu
SSPADD	PIC18F6X8X	PIC18F8X8X	0000 0000	0000 0000	uuuu uuuu
SSPSTAT	PIC18F6X8X	PIC18F8X8X	0000 0000	0000 0000	uuuu uuuu
SSPCON1	PIC18F6X8X	PIC18F8X8X	0000 0000	0000 0000	uuuu uuuu
SSPCON2	PIC18F6X8X	PIC18F8X8X	0000 0000	0000 0000	uuuu 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.

4.9 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 4-7 shows the data memory organization for the PIC18F6585/8585/6680/8680 devices.

The data memory map is divided into 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 SFRs are used for control and status of the controller and peripheral functions, while GPRs are used for data storage and scratch pad operations in the user's application. The SFRs start at the last location of Bank 15 (0FFFh) and extend downwards. Any remaining space beyond the SFRs in the Bank may be implemented as GPRs. GPRs 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 a File Select Register (FSRn) and a corresponding Indirect File Operand (INDFn). 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 (SFRs and select GPRs) 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 "Access Bank"** 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 using a File Select Register and corresponding Indirect File Operand. The operation of indirect addressing is shown in **Section 4.12 "Indirect Addressing, INDF and FSR Registers"**.

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

Data RAM is available for use as general purpose registers by all instructions. The top section of Bank 15 (0F60h to 0FFFh) contains SFRs. All other banks of data memory contain GPR registers, starting with Bank 0.

4.9.2 SPECIAL FUNCTION REGISTERS

The Special Function Registers (SFRs) 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 and Table 4-3.

The SFRs 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 SFRs are typically distributed among the peripherals whose functions they control.

The unused SFR locations are unimplemented and read as '0's. The addresses for the SFRs are listed in Table 4-2.

TABLE 4-2: SPECIAL FUNCTION REGISTER MAP (CONTINUED)

Address	Name	Address	Name	Address	Name	Address	Name
EFFh	—(1)	EDFh	—(1)	EBFh	—(1)	E9Fh	—(1)
EFEh	—(1)	EDEh	—(1)	EBEh	—(1)	E9Eh	—(1)
EFDh	—(1)	EDDh	—(1)	EBDh	—(1)	E9Dh	—(1)
EFCh	—(1)	EDCh	—(1)	EBCh	—(1)	E9Ch	—(1)
EFBh	—(1)	EDBh	—(1)	EBBh	—(1)	E9Bh	—(1)
EFAh	—(1)	EDAh	—(1)	EBAh	—(1)	E9Ah	—(1)
EF9h	—(1)	ED9h	—(1)	EB9h	—(1)	E99h	—(1)
EF8h	—(1)	ED8h	—(1)	EB8h	—(1)	E98h	—(1)
EF7h	—(1)	ED7h	—(1)	EB7h	—(1)	E97h	—(1)
EF6h	—(1)	ED6h	—(1)	EB6h	—(1)	E96h	—(1)
EF5h	—(1)	ED5h	—(1)	EB5h	—(1)	E95h	—(1)
EF4h	—(1)	ED4h	—(1)	EB4h	—(1)	E94h	—(1)
EF3h	—(1)	ED3h	—(1)	EB3h	—(1)	E93h	—(1)
EF2h	—(1)	ED2h	—(1)	EB2h	—(1)	E92h	—(1)
EF1h	—(1)	ED1h	—(1)	EB1h	—(1)	E91h	—(1)
EF0h	—(1)	ED0h	—(1)	EB0h	—(1)	E90h	—(1)
EEFh	—(1)	ECFh	—(1)	EAFh	—(1)	E8Fh	—(1)
EEEh	—(1)	ECEh	—(1)	EAEh	—(1)	E8Eh	—(1)
EEDh	—(1)	ECDh	—(1)	EADh	—(1)	E8Dh	—(1)
EECh	—(1)	ECCh	—(1)	EACh	—(1)	E8Ch	—(1)
EEBh	—(1)	ECBh	—(1)	EABh	—(1)	E8Bh	—(1)
EEAh	—(1)	ECAh	—(1)	EAAh	—(1)	E8Ah	—(1)
EE9h	—(1)	EC9h	—(1)	EA9h	—(1)	E89h	—(1)
EE8h	—(1)	EC8h	—(1)	EA8h	—(1)	E88h	—(1)
EE7h	—(1)	EC7h	—(1)	EA7h	—(1)	E87h	—(1)
EE6h	—(1)	EC6h	—(1)	EA6h	—(1)	E86h	—(1)
EE5h	—(1)	EC5h	—(1)	EA5h	—(1)	E85h	—(1)
EE4h	—(1)	EC4h	—(1)	EA4h	—(1)	E84h	—(1)
EE3h	—(1)	EC3h	—(1)	EA3h	—(1)	E83h	—(1)
EE2h	—(1)	EC2h	—(1)	EA2h	—(1)	E82h	—(1)
EE1h	—(1)	EC1h	—(1)	EA1h	—(1)	E81h	—(1)
EE0h	—(1)	EC0h	—(1)	EA0h	—(1)	E80h	—(1)

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

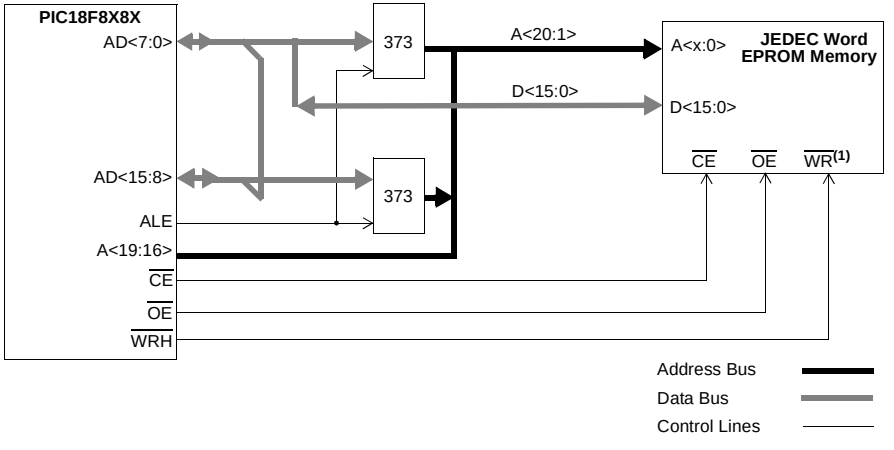
2: This register is not available on PIC18F6X8X devices.

3: This is not a physical register.

6.2.2 16-BIT WORD WRITE MODE

Figure 6-2 shows an example of 16-bit Word Write mode for PIC18F8X8X devices.

FIGURE 6-2: 16-BIT WORD WRITE MODE EXAMPLE



Note 1: This signal only applies to table writes. See **Section 5.1 "Table Reads and Table Writes"**.

8.0 8 x 8 HARDWARE MULTIPLIER

8.1 Introduction

An 8 x 8 hardware multiplier is included in the ALU of the PIC18F6585/8585/6680/8680 devices. By making the multiply a hardware operation, it completes in a single instruction cycle. This is an unsigned multiply that gives a 16-bit result. The result is stored in the 16-bit product register pair (PRODH:PRODL). The multiplier does not affect any flags in the ALUSTA register.

Making the 8 x 8 multiplier execute in a single cycle gives the following advantages:

- Higher computational throughput
- Reduces code size requirements for multiply algorithms

The performance increase allows the device to be used in applications previously reserved for Digital Signal Processors.

Table 8-1 shows a performance comparison between enhanced devices using the single-cycle hardware multiply and performing the same function without the hardware multiply.

8.2 Operation

Example 8-1 shows the sequence to do an 8 x 8 unsigned multiply. Only one instruction is required when one argument of the multiply is already loaded in the WREG register.

Example 8-2 shows the sequence to do an 8 x 8 signed multiply. To account for the sign bits of the arguments, each argument's Most Significant bit (MSb) is tested and the appropriate subtractions are done.

EXAMPLE 8-1: 8 x 8 UNSIGNED MULTIPLY ROUTINE

```
MOVF    ARG1, W         ;
MULWF   ARG2            ; ARG1 * ARG2 ->
                        ; PRODH:PRODL
```

EXAMPLE 8-2: 8 x 8 SIGNED MULTIPLY ROUTINE

```
MOVF    ARG1, W         ;
MULWF   ARG2            ; ARG1 * ARG2 ->
                        ; PRODH:PRODL
BTFSC   ARG2, SB        ; Test Sign Bit
SUBWF   PRODH           ; PRODH = PRODH
                        ;         - ARG1
MOVF    ARG2, W         ;
BTFSC   ARG1, SB        ; Test Sign Bit
SUBWF   PRODH           ; PRODH = PRODH
                        ;         - ARG2
```

TABLE 8-1: PERFORMANCE COMPARISON

Routine	Multiply Method	Program Memory (Words)	Cycles (Max)	Time		
				@ 40 MHz	@ 10 MHz	@ 4 MHz
8 x 8 unsigned	Without hardware multiply	13	69	6.9 µs	27.6 µs	69 µs
	Hardware multiply	1	1	100 ns	400 ns	1 µs
8 x 8 signed	Without hardware multiply	33	91	9.1 µs	36.4 µs	91 µs
	Hardware multiply	6	6	600 ns	2.4 µs	6 µs
16 x 16 unsigned	Without hardware multiply	21	242	24.2 µs	96.8 µs	242 µs
	Hardware multiply	24	24	2.4 µs	9.6 µs	24 µs
16 x 16 signed	Without hardware multiply	52	254	25.4 µs	102.6 µs	254 µs
	Hardware multiply	36	36	3.6 µs	14.4 µs	36 µs

TABLE 10-1: PORTA FUNCTIONS

Name	Bit#	Buffer	Function
RA0/AN0	bit 0	TTL	Input/output or analog input.
RA1/AN1	bit 1	TTL	Input/output or analog input.
RA2/AN2/VREF-	bit 2	TTL	Input/output or analog input or VREF-.
RA3/AN3/VREF+	bit 3	TTL	Input/output or analog input or VREF+.
RA4/T0CKI	bit 4	ST/OD	Input/output or external clock input for Timer0. Output is open-drain type.
RA5/AN4/LVDIN	bit 5	TTL	Input/output or slave select input for synchronous serial port or analog input, or Low-Voltage Detect input.
OSC2/CLKO/RA6	bit 6	TTL	OSC2 or clock output, or I/O pin.

Legend: TTL = TTL input, ST = Schmitt Trigger input

TABLE 10-2: SUMMARY OF REGISTERS ASSOCIATED WITH PORTA

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
PORTA	—	RA6	RA5	RA4	RA3	RA2	RA1	RA0	-x0x 0000	-u0u 0000
LATA	—	LATA Data Output Register							-xxx xxxx	-uuu uuuu
TRISA	—	PORTA Data Direction Register							-111 1111	-111 1111
ADCON1	—	—	VCFG1	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0	--00 0000	--00 0000

Legend: x = unknown, u = unchanged, – = unimplemented locations read as '0'.
Shaded cells are not used by PORTA.

10.6 PORTF, LATF and TRISF Registers

PORTF is an 8-bit wide, bidirectional port. The corresponding data direction register is TRISF. Setting a TRISF bit (= 1) will make the corresponding PORTF pin an input (i.e., put the corresponding output driver in a high-impedance mode). Clearing a TRISF bit (= 0) will make the corresponding PORTF pin an output (i.e., put the contents of the output latch on the selected pin).

Read-modify-write operations on the LATF register read and write the latched output value for PORTF.

PORTF is multiplexed with several analog peripheral functions, including the A/D converter inputs and comparator inputs, outputs, and voltage reference.

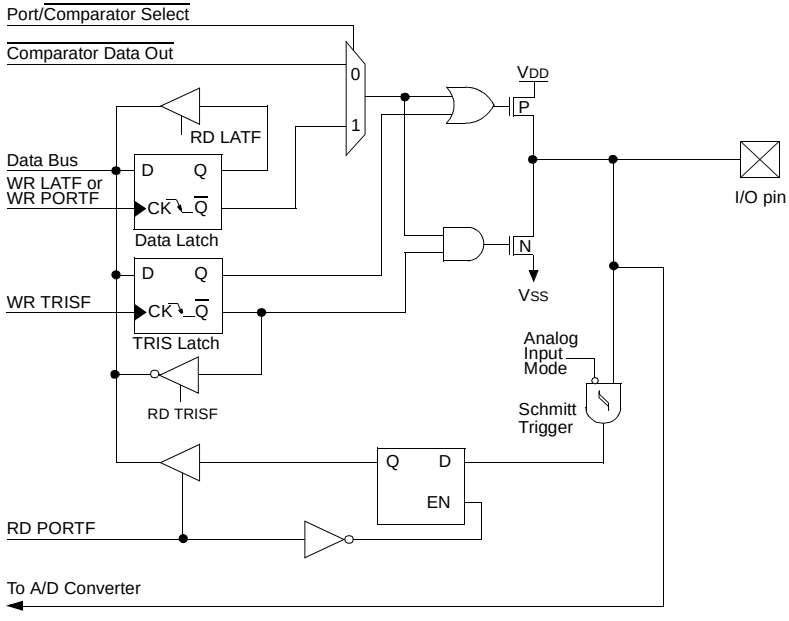
Note 1: On a Power-on Reset, the RF6:RF0 pins are configured as inputs and read as '0'.

2: To configure PORTF as digital I/O, turn off comparators and set ADCON1 value.

EXAMPLE 10-6: INITIALIZING PORTF

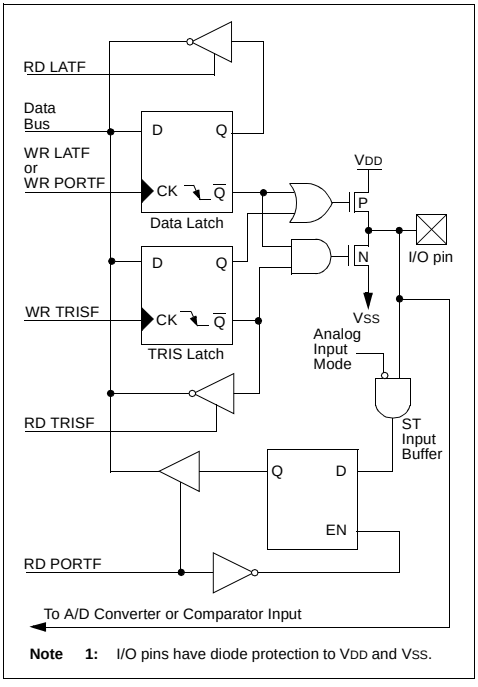
```
CLRF    PORTF       ; Initialize PORTF by
                    ; clearing output
                    ; data latches
CLRF    LATF        ; Alternate method
                    ; to clear output
                    ; data latches
MOVLW   07h         ;
MOVWF   CMCON       ; Turn off comparators
MOVLW   0Fh         ;
MOVWF   ADCON1      ; Set PORTF as digital I/O
MOVLW   0CFh        ; Value used to
                    ; initialize data
                    ; direction
MOVWF   TRISF       ; Set RF3:RF0 as inputs
                    ; RF5:RF4 as outputs
                    ; RF7:RF6 as inputs
```

FIGURE 10-13: PORTF RF1/AN6/C2OUT AND RF2/AN7/C1OUT PINS BLOCK DIAGRAM



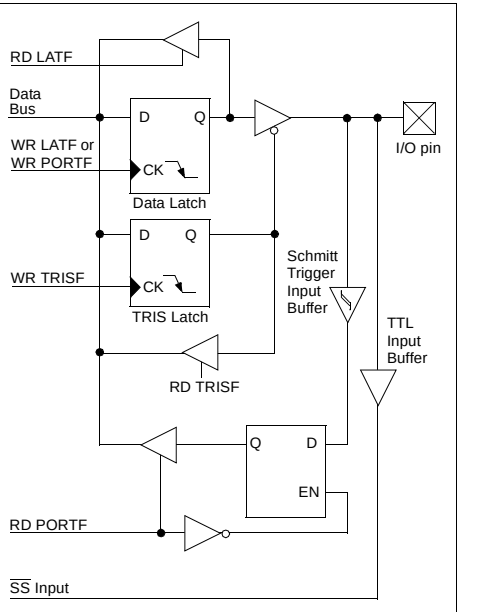
Note 1: I/O pins have diode protection to VDD and VSS.

FIGURE 10-14: RF6:RF3 AND RF0 PINS BLOCK DIAGRAM



Note 1: I/O pins have diode protection to VDD and VSS.

FIGURE 10-15: RF7 PIN BLOCK DIAGRAM



Note: I/O pins have diode protection to VDD and VSS.

REGISTER 15-2: CCP2CON REGISTER

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	DC2B1	DC2B0	CCP2M3	CCP2M2	CCP2M1	CCP2M0
bit 7							bit 0

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

bit 5-4 **DC2B1:DC2B0**: PWM Duty Cycle bit 1 and bit 0

Capture mode:
Unused.

Compare mode:
Unused.

PWM mode:
These bits are the two LSbs of the 10-bit PWM duty cycle. The eight MSbs of the duty cycle are found in CCPR2L.

bit 3-0 **CCP2M3:CCP2M0**: CCP2 Mode Select bits

0000 = Capture/Compare/PWM off (resets CCP2 module)
0001 = Reserved
0010 = Compare mode, toggle output on match
0011 = Reserved
0100 = Capture mode, every falling edge
0101 = Capture mode, every rising edge
0110 = Capture mode, every 4th rising edge
0111 = Capture mode, every 16th rising edge
1000 = Compare mode, initialize CCP pin low, on compare match force CCP pin high
1001 = Compare mode, initialize CCP pin high, on compare match force CCP pin low
1010 = Compare mode, generate software interrupt only, CCP pin is unaffected
1011 = Compare mode, trigger special event, resets TMR1 or TMR3 and starts A/D conversion if A/D module is enabled
11xx = PWM mode

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

17.4.17.2 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 deasserts 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 deasserted 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 17-29). 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 (see Figure 17-30).

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 17-29: BUS COLLISION DURING A REPEATED START CONDITION (CASE 1)

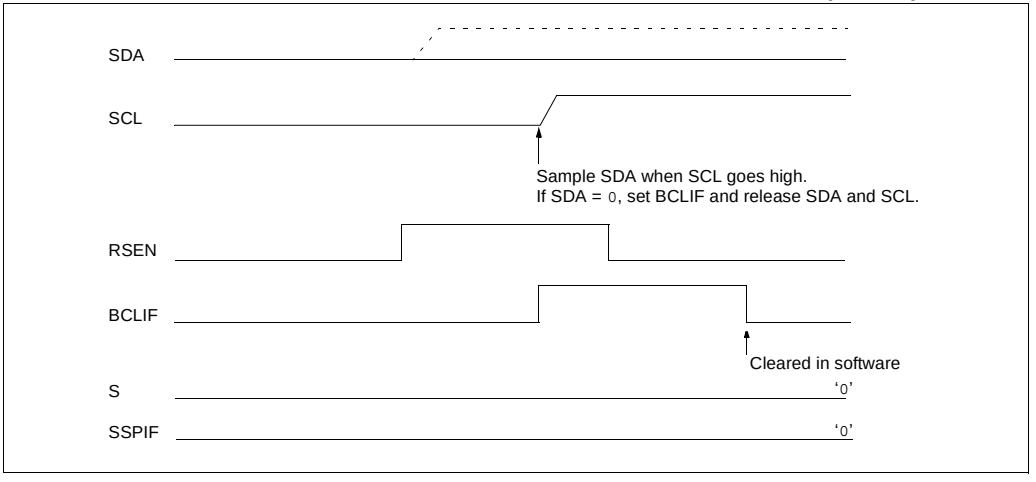
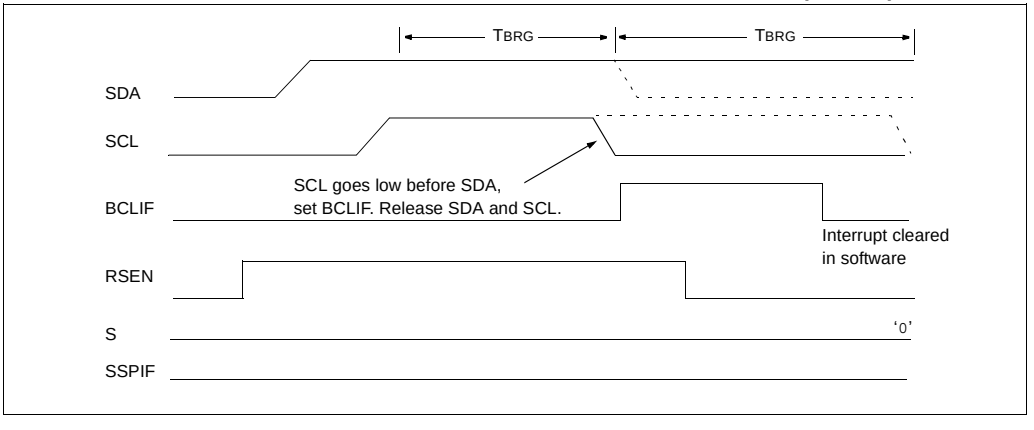


FIGURE 17-30: BUS COLLISION DURING REPEATED START CONDITION (CASE 2)



18.1.2 AUTO-BAUD RATE DETECT

The enhanced USART module supports the automatic detection and calibration of baud rate. This feature is active only in Asynchronous mode and while the WUE bit is clear.

The automatic baud rate measurement sequence (Figure 18-1) begins whenever a Start bit is received and the ABDEN bit is set. The calculation is self-averaging.

In the Auto-Baud Rate Detect (ABD) mode, the clock to the BRG is reversed. Rather than the BRG clocking the incoming RX signal, the RX signal is timing the BRG. In ABD mode, the internal Baud Rate Generator is used as a counter to time the bit period of the incoming serial byte stream.

Once the ABDEN bit is set, the state machine will clear the BRG and look for a Start bit. The auto-baud detect must receive a byte with the value 55h (ASCII “U”, which is also the LIN bus sync character) in order to calculate the proper bit rate. The measurement is taken over both a low and a high bit time in order to minimize any effects caused by asymmetry of the incoming signal. After a Start bit, the SPBRG begins counting up using the preselected clock source on the first rising edge of RX. After eight bits on the RX pin or the fifth rising edge, an accumulated value totalling the proper BRG period is left in the SPBRGH:SPBRG registers. Once the 5th edge is seen (should correspond to the Stop bit), the ABDEN bit is automatically cleared.

While calibrating the baud rate period, the BRG registers are clocked at 1/8th the preconfigured clock rate. Note that the BRG clock will be configured by the BRG16 and BRGH bits. Independent of the BRG16 bit setting, both the SPBRG and SPBRGH will be used as a 16-bit counter. This allows the user to verify that no carry occurred for 8-bit modes by checking for 00h in the SPBRGH register. Refer to Table 18-4 for counter clock rates to the BRG.

While the ABD sequence takes place, the USART state machine is held in Idle. The RCIF interrupt is set once the fifth rising edge on RX is detected. The value in the RCREG needs to be read to clear the RCIF interrupt. RCREG content should be discarded.

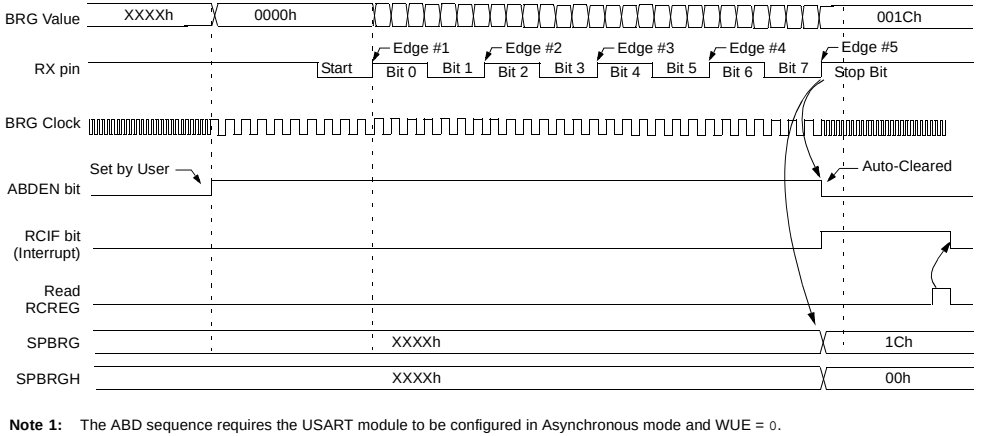
Note 1:	If the WUE bit is set with the ABDEN bit, auto-baud rate detection will occur on the byte *following* the break character.
2:	It is up to the user to determine that the incoming character baud rate is within the range of the selected BRG clock source. Some combinations of oscillator frequency and USART baud rates are not possible due to bit error rates. Overall system timing and communication baud rates must be taken into consideration when using the auto-baud rate detection feature.

TABLE 18-4: BRG COUNTER CLOCK RATES

BRG16	BRGH	BRG Counter Clock
0	0	FOSC/512
0	1	FOSC/128
1	0	FOSC/128
1	1	FOSC/32

Note: During the ABD sequence, SPBRG and SPBRGH are both used as a 16-bit counter independent of BRG16 setting.

FIGURE 18-1: AUTOMATIC BAUD RATE CALCULATION



Note 1: The ABD sequence requires the USART module to be configured in Asynchronous mode and WUE = 0.

18.2.5 BREAK CHARACTER SEQUENCE

The enhanced USART module has the capability of sending the special break character sequences that are required by the LIN bus standard. The break character transmit consists of a Start bit, followed by twelve '0' bits and a Stop bit. The frame break character is sent whenever the SENDB and TXEN bits (TXSTA<3> and TXSTA<5>) are set while the Transmit Shift register is loaded with data. Note that the value of data written to TXREG will be ignored and all '0's will be transmitted.

The SENDB bit is automatically reset by hardware after the corresponding Stop bit is sent. This allows the user to preload the transmit FIFO with the next transmit byte following the break character (typically, the sync character in the LIN specification).

Note that the data value written to the TXREG for the break character is ignored. The write simply serves the purpose of initiating the proper sequence.

The TRMT bit indicates when the transmit operation is active or Idle, just as it does during normal transmission. See Figure 18-9 for the timing of the break character sequence.

18.2.5.1 Break and Sync Transmit Sequence

The following sequence will send a message frame header made up of a break, followed by an auto-baud sync byte. This sequence is typical of a LIN bus master.

1. Configure the USART for the desired mode.
2. Set the TXEN and SENDB bits to set up the break character.
3. Load the TXREG with a dummy character to initiate transmission (the value is ignored).
4. Write '55h' to TXREG to load the sync character into the transmit FIFO buffer.
5. After the break has been sent, the SENDB bit is reset by hardware. The sync character now transmits in the preconfigured mode.

When the TXREG becomes empty, as indicated by the TXIF, the next data byte can be written to TXREG.

18.2.6 RECEIVING A BREAK CHARACTER

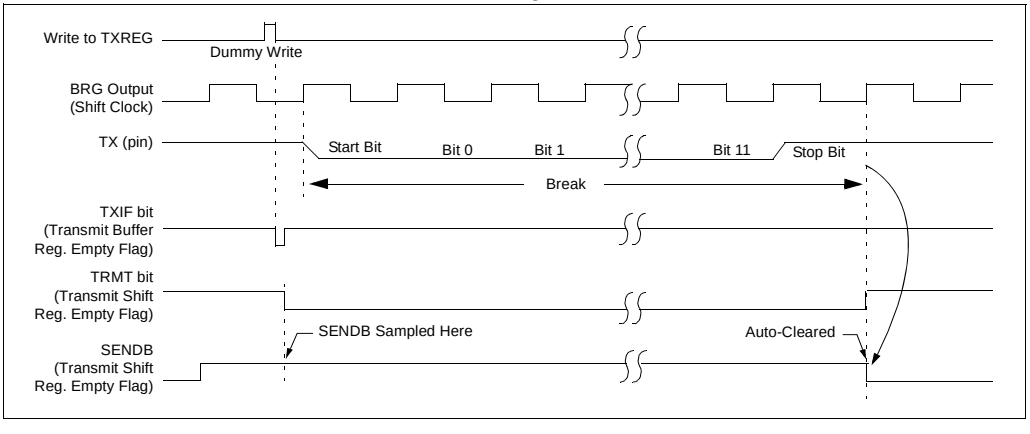
The enhanced USART module can receive a break character in two ways.

The first method forces the configuration of the baud rate at a frequency of 9/13 the typical speed. This allows for the Stop bit transition to be at the correct sampling location (13 bits for break versus Start bit and 8 data bits for typical data).

The second method uses the auto-wake-up feature described in **Section 18.2.4 "Auto-Wake-up on Sync Break Character"**. By enabling this feature, the USART will sample the next two transitions on RX/DT, cause an RCIF interrupt, and receive the next data byte followed by another interrupt.

Note that following a break character, the user will typically want to enable the auto-baud rate detect feature. For both methods, the user can set the ABD bit once the TXIF interrupt is observed.

FIGURE 18-9: SEND BREAK CHARACTER SEQUENCE



NOTES:

REGISTER 23-6: TXBnSIDH: TRANSMIT BUFFER n STANDARD IDENTIFIER REGISTERS, HIGH BYTE [0 ≤ n ≤ 2]

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 EXIDE (TXBnSIDL<3>) = 0;
Extended Identifier bits EID28:EID21, if EXIDE = 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

REGISTER 23-7: TXBnSIDL: TRANSMIT BUFFER n STANDARD IDENTIFIER REGISTERS, LOW BYTE [0 ≤ n ≤ 2]

R/W-x	R/W-x	R/W-x	U-0	R/W-x	U-0	R/W-x	R/W-x
SID2	SID1	SID0	—	EXIDE	—	EID17	EID16
bit 7							bit 0

bit 7-5 **SID2:SID0:** Standard Identifier bits, if EXIDE (TXBnSIDL<3>) = 0;
Extended Identifier bits EID20:EID18, if EXIDE = 1.

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

bit 3 **EXIDE:** Extended Identifier Enable bit
1 = Message will transmit extended ID, SID10:SID0 becomes EID28:EID18
0 = Message will transmit standard ID, EID17:EID0 are ignored

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

bit 1-0 **EID17:EID16:** Extended Identifier bits

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 23-8: TXBnEIDH: TRANSMIT BUFFER n EXTENDED IDENTIFIER REGISTERS, HIGH BYTE [0 ≤ n ≤ 2]

R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
EID15	EID14	EID13	EID12	EID11	EID10	EID9	EID8
bit 7							bit 0

bit 7-0 **EID15:EID8:** Extended Identifier bits (not used when transmitting standard identifier 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

In Mode 1 and 2, there are a total of 16 acceptance filters available and each can be dynamically assigned to any of the receive buffers. A buffer with a lower number has higher priority. Given this, if an incoming message matches with two or more receive buffer acceptance criteria, the buffer with the lower number will be loaded with that message.

23.7.3 ENHANCED FIFO MODE

When configured for Mode 2, two of the dedicated receive buffers, in combination with one or more programmable transmit/receive buffers, are used to create a maximum of 8 buffers deep FIFO (First In First Out) buffer. In this mode, there is no direct correlation between filters and receive buffer registers. Any filter that has been enabled can generate an acceptance. When a message has been accepted, it is stored in the next available receive buffer register and an internal write pointer is incremented. The FIFO can be a maximum of 8 buffers deep. The entire FIFO must consist of contiguous receive buffers. The FIFO head begins at RXB0 buffer and its tail spans toward B5. The maximum length of the FIFO is limited by the presence or absence of the first transmit buffer starting from B0. If a buffer is configured as a transmit buffer, the FIFO length is reduced accordingly. For instance, if B3 is configured as transmit buffer, the actual FIFO will consist of RXB0, RXB1, B0, B1 and B2, a total of 5 buffers. If B0 is configured as a transmit buffer, the FIFO length will be 2. If none of the programmable buffers are configured as a transmit buffer, the FIFO will be 8 buffers deep. A system that requires more transmit buffers should try to locate transmit buffers at the very end of B0-B5 buffers to maximize available FIFO length.

When a message is received in FIFO mode, the Interrupt Flag Code bits (EICODE<4:0>) in the CANSTAT register will have a value of '10000', indicating the FIFO has received a message. FIFO pointer bits FP<3:0> in the CANCON register point to the buffer that contains data not yet read. The FIFO pointer bits, in this sense, serve as the FIFO read pointer. The user should use FP bits and read corresponding buffer data. When receive data is no longer needed, the RXFUL bit in the current buffer must be cleared, causing FP<3:0> to be updated by the module.

To determine whether FIFO is empty or not, the user may use FP<3:0> bits to access RXFUL bit in the current buffer. If RXFUL is cleared, the FIFO is considered to be empty. If it is set, the FIFO may contain one or more messages. In Mode 2, the module also provides a bit called FIFO High Water Mark (FIFOWM) in the ECANCON register. This bit can be used to cause an interrupt whenever the FIFO contains only one or four empty buffers. The FIFO high water mark interrupt can serve as an early warning to a full FIFO condition.

23.7.4 TIME-STAMPING

The CAN module can be programmed to generate a time-stamp for every message that is received. When enabled, the module generates a capture signal for CCP1, which in turn captures the value of either Timer1 or Timer3. This value can be used as the message time-stamp.

To use the time-stamp capability, the CANCAP bit (CIOCAN<4>) must be set. This replaces the capture input for CCP1 with the signal generated from the CAN module. In addition, CCP1CON<3:0> must be set to '0011' to enable the CCP special event trigger for CAN events.

23.8 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 any 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 23-2 that indicates how each bit in the identifier is compared to the masks and filters to determine if a 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 23-2: 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

In Mode 0, 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.

LFSR **Load FSR**

Syntax: [*label*] LFSR f,k

Operands: $0 \le f \le 2$
$0 \le k \le 4095$

Operation: $k \rightarrow FSRf$

Status Affected: None

Encoding:

1110 1111	1110 0000	00ff $k_7$kkk	k_{11}kkk kkkk

Description: The 12-bit literal ‘k’ is loaded into the file select register pointed to by ‘f’.

Words: 2

Cycles: 2

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal ‘k’ MSB	Process Data	Write literal ‘k’ MSB to FSRfH
Decode	Read literal ‘k’ LSB	Process Data	Write literal ‘k’ to FSRfL

Example: `LFSR 2, 0x3AB`

After Instruction
FSR2H = 0x03
FSR2L = 0xAB

MOVF **Move f**

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

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

Operation: $f \rightarrow dest$

Status Affected: N, Z

Encoding:

0101	00da	ffff	ffff

Description: The contents of register ‘f’ are moved to a destination dependent upon the status of ‘d’. If ‘d’ is ‘0’, the result is placed in W. If ‘d’ is ‘1’, the result is placed back in register ‘f’ (default). Location ‘f’ can be anywhere in the 256-byte bank. If ‘a’ is ‘0’, the Access Bank will be selected, overriding the BSR value. If ‘a’ = 1, then the bank will be selected as per the BSR value (default).

Words: 1

Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register ‘f’	Process Data	Write W

Example: `MOVF REG, 0, 0`

Before Instruction
REG = 0x22
W = 0xFF

After Instruction
REG = 0x22
W = 0x22

27.3 DC Characteristics: PIC18FXX8X (Industrial, Extended) PIC18LFXX8X (Industrial) (Continued)

DC CHARACTERISTICS			Standard Operating Conditions (unless otherwise stated) Operating temperature -40°C ≤ TA ≤ +85°C for industrial -40°C ≤ TA ≤ +125°C for extended			
Param No.	**Symbol**	**Characteristic**	**Min**	**Max**	**Units**	**Conditions**
	VOL	**Output Low Voltage**				
D080		I/O ports	—	0.6	V	IOL = 8.5 mA, VDD = 4.5V, -40°C to +85°C
D080A			—	0.6	V	IOL = 7.0 mA, VDD = 4.5V, -40°C to +125°C
D083		OSC2/CLKO (RC mode)	—	0.6	V	IOL = 1.6 mA, VDD = 4.5V, -40°C to +85°C
D083A			—	0.6	V	IOL = 1.2 mA, VDD = 4.5V, -40°C to +125°C
	VOH	**Output High Voltage(3)**				
D090		I/O ports	VDD – 0.7	—	V	IOH = -3.0 mA, VDD = 4.5V, -40°C to +85°C
D090A			VDD – 0.7	—	V	IOH = -2.5 mA, VDD = 4.5V, -40°C to +125°C
D092		OSC2/CLKO (RC mode)	VDD – 0.7	—	V	IOH = -1.3 mA, VDD = 4.5V, -40°C to +85°C
D092A			VDD – 0.7	—	V	IOH = -1.0 mA, VDD = 4.5V, -40°C to +125°C
D150	VOD	**Open-Drain High Voltage**	—	8.5	V	RA4 pin
		Capacitive Loading Specs on Output Pins				
D100(4)	COSC2	OSC2 pin	—	15	pF	In XT, HS and LP modes when external clock is used to drive OSC1
D101	CIO	All I/O pins and OSC2 (in RC mode)	—	50	pF	To meet the AC Timing Specifications
D102	CB	SCL, SDA	—	400	pF	In I^2C mode

Note 1: In RC oscillator configuration, the OSC1/CLKI pin is a Schmitt Trigger input. It is not recommended that the PIC device be driven with an external clock while in RC mode.

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

3: Negative current is defined as current sourced by the pin.

4: Parameter is characterized but not tested.

27.4 AC (Timing) Characteristics

27.4.1 TIMING PARAMETER SYMBOLOGY

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

1. TppS2ppS		3. TCC:ST	(I^2C specifications only)
2. TppS		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 (high-impedance)	V	Valid
L	Low	Z	High-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		

FIGURE 28-11: TYPICAL AND MAXIMUM I_{T1OSC} vs. V_{DD} (TIMER1 AS SYSTEM CLOCK)

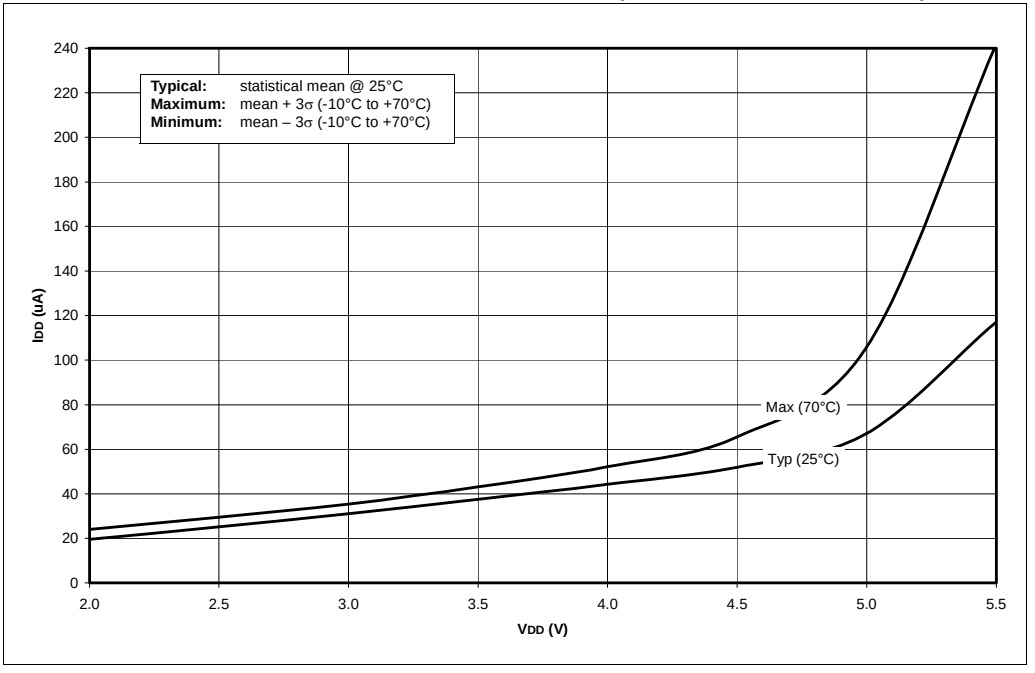
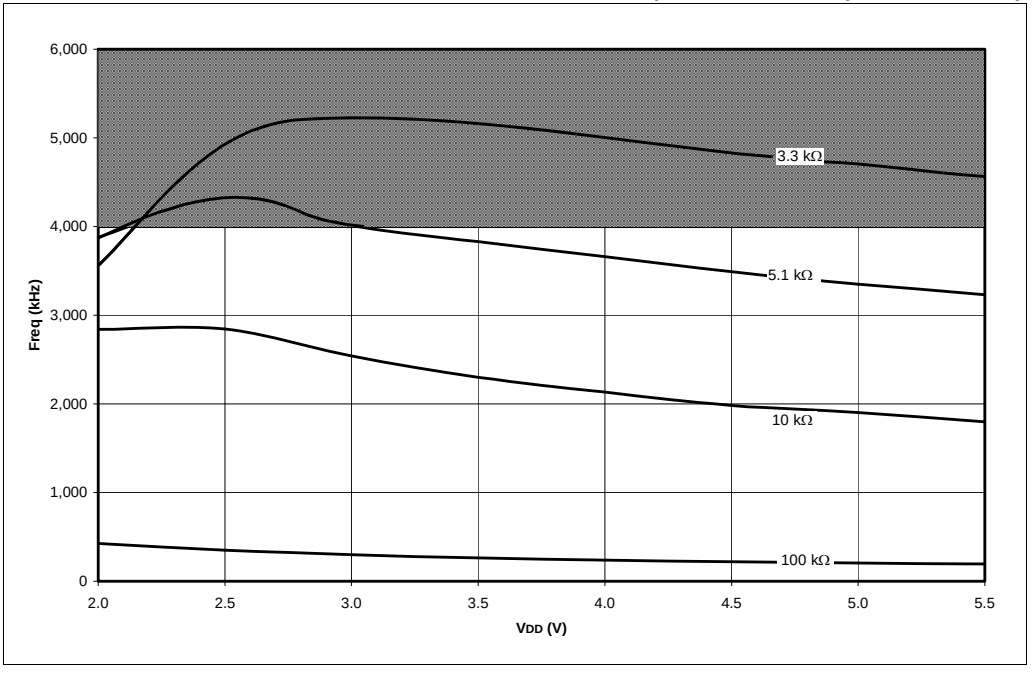


FIGURE 28-12: AVERAGE F_{OSC} vs. V_{DD} FOR VARIOUS R's (RC MODE, C = 20 pF, TEMP = 25°C)



F

Firmware Instructions 365
Flash Program Memory 83
  Associated Registers 92
  Control Registers 84
  Erase Sequence 88
  Erasing 88
  Operation During Code Protection 92
  Reading 87
  Table Pointer Boundaries Based on Operation 86
  Table Pointer Boundaries 86
  Table Reads and Table Writes 83
  Write Sequence 90
  Writing to 89
    Protection Against Spurious Writes 92
    Unexpected Termination 92
    Write Verify 92

G

General Call Address Support 212
GOTO 386

H

Hardware Multiplier 107
  Introduction 107
  Operation 107
  Performance Comparison (table) 107

I

I/O Ports 125
I^{2}C Bus Data Requirements (Slave Mode) 442
I^{2}C Bus Start/Stop Bits Requirements (Slave Mode) 441
I^{2}C Mode
  General Call Address Support 212
  Master Mode
    Operation 214
  Read/$\overline{\text{Write}}$ Bit Information (R/$\overline{\text{W}}$ Bit) 202, 203
  Serial Clock (RC3/SCK/SCL) 203
ID Locations 345, 362
INCF 386
INCFSZ 387
In-Circuit Debugger 362
  Resources (table) 362
In-Circuit Serial Programming (ICSP) 345, 362
Indirect Addressing
  INDF and FSR Registers 79
  Operation 79
Indirect File Operand 59
INFSNZ 387
Instruction Flow/Pipelining 57
Instruction Format 367
Instruction Set 365
  ADDLW 371
  ADDWF 371
  ADDWFC 372
  ANDLW 372
  ANDWF 373
  BC 373
  BCF 374
  BN 374
  BNC 375
  BNN 375
  BNOV 376
  BNZ 376
  BOV 379
  BRA 377
  BSF 377
  BTFSC 378
  BTFSS 378
  BTG 379
  BZ 380
  CALL 380
  CLRF 381
  CLRWDT 381
  COMF 382
  CPFSEQ 382
  CPFSGT 383
  CPFSLT 383
  DAW 384
  DCFSNZ 385
  DECF 384
  DECFSZ 385
  GOTO 386
  INCF 386
  INCFSZ 387
  INFSNZ 387
  IORLW 388
  IORWF 388
  LFSR 389
  MOVF 389
  MOVFF 390
  MOVLB 390
  MOVLW 391
  MOVWF 391
  MULLW 392
  MULWF 392
  NEGF 393
  NOP 393
  POP 394
  PUSH 394
  RCALL 395
  RESET 395
  RETFIE 396
  RETLW 396
  RETURN 397
  RLCF 397
  RLNCF 398
  RRCF 398
  RRNCF 399
  SETF 399