



Welcome to [E-XFL.COM](https://www.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, I ² C, SPI, UART/USART |
| Peripherals | Brown-out Detect/Reset, LVD, POR, PWM, WDT |
| Number of I/O | 33 |
| Program Memory Size | 16KB (8K x 16) |
| Program Memory Type | FLASH |
| EEPROM Size | 256 x 8 |
| RAM Size | 768 x 8 |
| Voltage - Supply (Vcc/Vdd) | 4.2V ~ 5.5V |
| Data Converters | A/D 8x10b |
| Oscillator Type | External |
| Operating Temperature | -40°C ~ 85°C (TA) |
| Mounting Type | Surface Mount |
| Package / Case | 44-LCC (J-Lead) |
| Supplier Device Package | 44-PLCC (16.59x16.59) |
| Purchase URL | https://www.e-xfl.com/product-detail/microchip-technology/pic18f448-i-l |

Note the following details of the code protection feature on Microchip devices:

- Microchip products meet the specification contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is one of the most secure families of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the Microchip products in a manner outside the operating specifications contained in Microchip's Data Sheets. Most likely, the person doing so is engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as "unbreakable."

Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our products. Attempts to break Microchip's code protection feature may be a violation of the Digital Millennium Copyright Act. If such acts allow unauthorized access to your software or other copyrighted work, you may have a right to sue for relief under that Act.

Information contained in this publication regarding device applications and the like is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION, INCLUDING BUT NOT LIMITED TO ITS CONDITION, QUALITY, PERFORMANCE, MERCHANTABILITY OR FITNESS FOR PURPOSE. Microchip disclaims all liability arising from this information and its use. Use of Microchip devices in life support and/or safety applications is entirely at the buyer's risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights.

Trademarks

The Microchip name and logo, the Microchip logo, Accuron, dsPIC, KEELOQ, microID, MPLAB, PIC, PICmicro, PICSTART, PRO MATE, PowerSmart, rfPIC and SmartShunt are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.


Amplab, FilterLab, Migratable Memory, MXDEV, MXLAB, SEEVAL, SmartSensor and The Embedded Control Solutions Company are registered trademarks of Microchip Technology Incorporated in the U.S.A.

Analog-for-the-Digital Age, Application Maestro, CodeGuard, dsPICDEM, dsPICDEM.net, dsPICworks, ECAN, ECONOMONITOR, FanSense, FlexROM, fuzzyLAB, In-Circuit Serial Programming, ICSP, ICEPIC, Linear Active Thermistor, Mindi, MiWi, MPASM, MPLIB, MPLINK, PICkit, PICDEM, PICDEM.net, PICLAB, PICtail, PowerCal, PowerInfo, PowerMate, PowerTool, REAL ICE, rLAB, rfPICDEM, Select Mode, Smart Serial, SmartTel, Total Endurance, UNI/O, WiperLock and ZENA are trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

SQTP is a service mark of Microchip Technology Incorporated in the U.S.A.

All other trademarks mentioned herein are property of their respective companies.

© 2006, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

 Printed on recycled paper.

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
== ISO/TS 16949:2002 ==

Microchip received ISO/TS-16949:2002 certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona, Gresham, Oregon and Mountain View, California. The Company's quality system processes and procedures are for its PICmicro® 8-bit MCUs, KEELOQ® code hopping devices, Serial EEPROMs, microperipherals, nonvolatile memory and analog products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001:2000 certified.

PIC18FXX8

TABLE 1-2: PIC18FXX8 PINOUT I/O DESCRIPTIONS (CONTINUED)

| Pin Name | Pin Number | | | | Pin Type | Buffer Type | Description |
|--|---------------|---------------|------|------|----------------|------------------|---|
| | PIC18F248/258 | PIC18F448/458 | | | | | |
| | SPDIP, SOIC | PDIP | TQFP | PLCC | | | |
| | | | | | | | PORTB is a bidirectional I/O port. PORTB can be software programmed for internal weak pull-ups on all inputs. |
| RB0/INT0 RB0 INT0 | 21 | 33 | 8 | 36 | I/O I | TTL ST | Digital I/O. External interrupt 0. |
| RB1/INT1 RB1 INT1 | 22 | 34 | 9 | 37 | I/O I | TTL ST | Digital I/O. External interrupt 1. |
| RB2/CANTX/INT2 RB2 CANTX INT2 | 23 | 35 | 10 | 38 | I/O O I | TTL TTL ST | Digital I/O. Transmit signal for CAN bus. External interrupt 2. |
| RB3/CANRX RB3 CANRX | 24 | 36 | 11 | 39 | I/O I | TTL TTL | Digital I/O. Receive signal for CAN bus. |
| RB4 | 25 | 37 | 14 | 41 | I/O | TTL | Digital I/O. Interrupt-on-change pin. |
| RB5/PGM RB5 PGM | 26 | 38 | 15 | 42 | I/O I | TTL ST | Digital I/O. Interrupt-on-change pin. Low-voltage ICSP™ programming enable. |
| RB6/PGC RB6 PGC | 27 | 39 | 16 | 43 | I/O I | TTL ST | Digital I/O. In-Circuit Debugger pin. Interrupt-on-change pin. ICSP programming clock. |
| RB7/PGD RB7 PGD | 28 | 40 | 17 | 44 | I/O I/O | TTL ST | Digital I/O. In-Circuit Debugger pin. Interrupt-on-change pin. ICSP programming data. |

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 VDD)

PIC18FXX8

TABLE 4-2: REGISTER FILE SUMMARY (CONTINUED)

| File Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 | Value on POR, BOR | Details on Page: |
|------------|---------|---------|---------|---------|---------|---------|---------|---------|-------------------|------------------|
| CANSTATRO1 | OPMODE2 | OPMODE1 | OPMODE0 | — | ICODE2 | ICODE1 | ICODE0 | — | xxx - xxx - | 33, 202 |
| RXB1D7 | RXB1D77 | RXB1D76 | RXB1D75 | RXB1D74 | RXB1D73 | RXB1D72 | RXB1D71 | RXB1D70 | xxxx xxxx | 34, 214 |
| RXB1D6 | RXB1D67 | RXB1D66 | RXB1D65 | RXB1D64 | RXB1D63 | RXB1D62 | RXB1D61 | RXB1D60 | xxxx xxxx | 34, 214 |
| RXB1D5 | RXB1D57 | RXB1D56 | RXB1D55 | RXB1D54 | RXB1D53 | RXB1D52 | RXB1D51 | RXB1D50 | xxxx xxxx | 34, 214 |
| RXB1D4 | RXB1D47 | RXB1D46 | RXB1D45 | RXB1D44 | RXB1D43 | RXB1D42 | RXB1D41 | RXB1D40 | xxxx xxxx | 34, 214 |
| RXB1D3 | RXB1D37 | RXB1D36 | RXB1D35 | RXB1D34 | RXB1D33 | RXB1D32 | RXB1D31 | RXB1D30 | xxxx xxxx | 34, 214 |
| RXB1D2 | RXB1D27 | RXB1D26 | RXB1D25 | RXB1D24 | RXB1D23 | RXB1D22 | RXB1D21 | RXB1D20 | xxxx xxxx | 34, 214 |
| RXB1D1 | RXB1D17 | RXB1D16 | RXB1D15 | RXB1D14 | RXB1D13 | RXB1D12 | RXB1D11 | RXB1D10 | xxxx xxxx | 34, 214 |
| RXB1D0 | RXB1D07 | RXB1D06 | RXB1D05 | RXB1D04 | RXB1D03 | RXB1D02 | RXB1D01 | RXB1D00 | xxxx xxxx | 34, 214 |
| RXB1DLC | — | RXRTR | RB1 | RB0 | DLC3 | DLC2 | DLC1 | DLC0 | -xxx xxxx | 34, 213 |
| RXB1EIDL | EID7 | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0 | xxxx xxxx | 34, 213 |
| RXB1EIDH | EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8 | xxxx xxxx | 34, 212 |
| RXB1SIDL | SID2 | SID1 | SID0 | SRR | EXID | — | EID17 | EID16 | xxxx x-xx | 34, 212 |
| RXB1SIDH | SID10 | SID9 | SID8 | SID7 | SID6 | SID5 | SID4 | SID3 | xxxx xxxx | 34, 212 |
| RXB1CON | RXFUL | RXM1 | RXM0 | — | RXRTRRO | FILHIT2 | FILHIT1 | FILHIT0 | 000 - 0000 | 34, 211 |
| CANSTATRO2 | OPMODE2 | OPMODE1 | OPMODE0 | — | ICODE2 | ICODE1 | ICODE0 | — | xxx - xxx - | 33, 202 |
| TXB0D7 | TXB0D77 | TXB0D76 | TXB0D75 | TXB0D74 | TXB0D73 | TXB0D72 | TXB0D71 | TXB0D70 | xxxx xxxx | 34, 208 |
| TXB0D6 | TXB0D67 | TXB0D66 | TXB0D65 | TXB0D64 | TXB0D63 | TXB0D62 | TXB0D61 | TXB0D60 | xxxx xxxx | 34, 208 |
| TXB0D5 | TXB0D57 | TXB0D56 | TXB0D55 | TXB0D54 | TXB0D53 | TXB0D52 | TXB0D51 | TXB0D50 | xxxx xxxx | 34, 208 |
| TXB0D4 | TXB0D47 | TXB0D46 | TXB0D45 | TXB0D44 | TXB0D43 | TXB0D42 | TXB0D41 | TXB0D40 | xxxx xxxx | 34, 208 |
| TXB0D3 | TXB0D37 | TXB0D36 | TXB0D35 | TXB0D34 | TXB0D33 | TXB0D32 | TXB0D31 | TXB0D30 | xxxx xxxx | 34, 208 |
| TXB0D2 | TXB0D27 | TXB0D26 | TXB0D25 | TXB0D24 | TXB0D23 | TXB0D22 | TXB0D21 | TXB0D20 | xxxx xxxx | 34, 208 |
| TXB0D1 | TXB0D17 | TXB0D16 | TXB0D15 | TXB0D14 | TXB0D13 | TXB0D12 | TXB0D11 | TXB0D10 | xxxx xxxx | 34, 208 |
| TXB0D0 | TXB0D07 | TXB0D06 | TXB0D05 | TXB0D04 | TXB0D03 | TXB0D02 | TXB0D01 | TXB0D00 | xxxx xxxx | 34, 208 |
| TXB0DLC | — | TXRTR | — | — | DLC3 | DLC2 | DLC1 | DLC0 | -x-- xxxx | 34, 209 |
| TXB0EIDL | EID7 | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0 | xxxx xxxx | 34, 208 |
| TXB0EIDH | EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8 | xxxx xxxx | 34, 207 |
| TXB0SIDL | SID2 | SID1 | SID0 | — | EXIDE | — | EID17 | EID16 | xxx - x-xx | 34, 207 |
| TXB0SIDH | SID10 | SID9 | SID8 | SID7 | SID6 | SID5 | SID4 | SID3 | xxxx xxxx | 35, 207 |
| TXB0CON | — | TXABT | TXLARB | TXERR | TXREQ | — | TXPRI1 | TXPRI0 | -000 0-00 | 35, 206 |
| CANSTATRO3 | OPMODE2 | OPMODE1 | OPMODE0 | — | ICODE2 | ICODE1 | ICODE0 | — | xxx - xxx - | 33, 202 |
| TXB1D7 | TXB1D77 | TXB1D76 | TXB1D75 | TXB1D74 | TXB1D73 | TXB1D72 | TXB1D71 | TXB1D70 | xxxx xxxx | 35, 208 |
| TXB1D6 | TXB1D67 | TXB1D66 | TXB1D65 | TXB1D64 | TXB1D63 | TXB1D62 | TXB1D61 | TXB1D60 | xxxx xxxx | 35, 208 |
| TXB1D5 | TXB1D57 | TXB1D56 | TXB1D55 | TXB1D54 | TXB1D53 | TXB1D52 | TXB1D51 | TXB1D50 | xxxx xxxx | 35, 208 |
| TXB1D4 | TXB1D47 | TXB1D46 | TXB1D45 | TXB1D44 | TXB1D43 | TXB1D42 | TXB1D41 | TXB1D40 | xxxx xxxx | 35, 208 |
| TXB1D3 | TXB1D37 | TXB1D36 | TXB1D35 | TXB1D34 | TXB1D33 | TXB1D32 | TXB1D31 | TXB1D30 | xxxx xxxx | 35, 208 |
| TXB1D2 | TXB1D27 | TXB1D26 | TXB1D25 | TXB1D24 | TXB1D23 | TXB1D22 | TXB1D21 | TXB1D20 | xxxx xxxx | 35, 208 |
| TXB1D1 | TXB1D17 | TXB1D16 | TXB1D15 | TXB1D14 | TXB1D13 | TXB1D12 | TXB1D11 | TXB1D10 | xxxx xxxx | 35, 208 |
| TXB1D0 | TXB1D07 | TXB1D06 | TXB1D05 | TXB1D04 | TXB1D03 | TXB1D02 | TXB1D01 | TXB1D00 | xxxx xxxx | 35, 208 |
| TXB1DLC | — | TXRTR | — | — | DLC3 | DLC2 | DLC1 | DLC0 | -x-- xxxx | 35, 209 |
| TXB1EIDL | EID7 | EID6 | EID5 | EID4 | EID3 | EID2 | EID1 | EID0 | xxxx xxxx | 35, 208 |
| TXB1EIDH | EID15 | EID14 | EID13 | EID12 | EID11 | EID10 | EID9 | EID8 | xxxx xxxx | 35, 207 |
| TXB1SIDL | SID2 | SID1 | SID0 | — | EXIDE | — | EID17 | EID16 | xxx - x-xx | 35, 207 |
| TXB1SIDH | SID10 | SID9 | SID8 | SID7 | SID6 | SID5 | SID4 | SID3 | xxxx xxxx | 35, 207 |
| TXB1CON | — | TXABT | TXLARB | TXERR | TXREQ | — | TXPRI1 | TXPRI0 | 0000 0000 | 35, 206 |

Legend: x = unknown, u = unchanged, - = unimplemented, q = value depends on condition

Note 1: These registers or register bits are not implemented on the PIC18F248 and PIC18F258 and read as '0's.

Note 2: Bit 21 of the TBLPTRU allows access to the device configuration bits.

Note 3: RA6 and associated bits are configured as port pins in RCIO and ECIO Oscillator mode only and read '0' in all other oscillator modes.

6.0 FLASH PROGRAM MEMORY

The Flash program memory is readable, writable and erasable during normal operation over the entire VDD range.

A read from program memory is executed on one byte at a time. A write to program memory is executed on blocks of 8 bytes at a time. Program memory is erased in blocks of 64 bytes at a time. A bulk erase operation may not be issued from user code.

Writing or erasing program memory will cease instruction fetches until the operation is complete. The program memory cannot be accessed during the write or erase, therefore, code cannot execute. An internal programming timer terminates program memory writes and erases.

A value written to program memory does not need to be a valid instruction. Executing a program memory location that forms an invalid instruction results in a NOP.

6.1 Table Reads and Table Writes

In order to read and write program memory, there are two operations that allow the processor to move bytes between the program memory space and the data RAM:

- Table Read (TBLRD)
- Table Write (TBLWT)

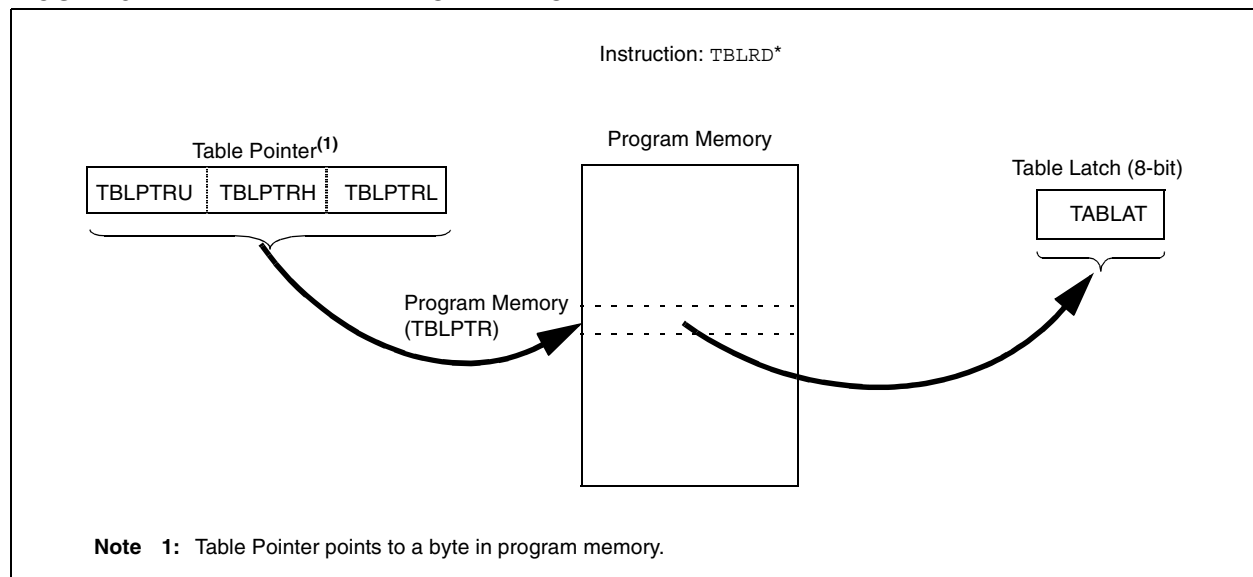
The program memory space is 16 bits wide, while the data RAM space is 8 bits wide. Table reads and table writes move data between these two memory spaces through an 8-bit register (TABLAT).

Table read operations retrieve data from program memory and place it into the data RAM space. Figure 6-1 shows the operation of a table read with program memory and data RAM.

Table write operations store data from the data memory space into holding registers in program memory. The procedure to write the contents of the holding registers into program memory is detailed in **Section 6.5 “Writing to Flash Program Memory”**. Figure 6-2 shows the operation of a table write with program memory and data RAM.

Table operations work with byte entities. A table block containing data, rather than program instructions, is not required to be word aligned. Therefore, a table block can start and end at any byte address. If a table write is being used to write executable code into program memory, program instructions will need to be word aligned.

FIGURE 6-1: TABLE READ OPERATION



PIC18FXX8

9.2 PORTB, TRISB and LATB Registers

PORTB is an 8-bit wide, bidirectional 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 high-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 9-2: INITIALIZING PORTB

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

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 $\overline{\text{RBP}}\text{U}$ (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 the PORTB 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 ORed 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:

- Any read or write of PORTB (except with the MOVFF instruction). This will end the mismatch condition.
- 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.

Note 1: While in Low-Voltage ICSP mode, the RB5 pin can no longer be used as a general purpose I/O pin and should not be held low during normal operation to protect against inadvertent ICSP mode entry.

- When using Low-Voltage ICSP Programming (LVP), the pull-up on RB5 becomes disabled. If TRISB bit 5 is cleared, thereby setting RB5 as an output, LATB bit 5 must also be cleared for proper operation.

12.2 Timer1 Oscillator

A crystal oscillator circuit is built in between pins T1OSI (input) and T1OSO (amplifier output). It is enabled by setting control bit T1OSCEN (T1CON register). The oscillator is a low-power oscillator rated up to 50 kHz. It will continue to run during Sleep. It is primarily intended for a 32 kHz crystal. Table 12-1 shows the capacitor selection for the Timer1 oscillator.

The user must provide a software time delay to ensure proper start-up of the Timer1 oscillator.

TABLE 12-1: CAPACITOR SELECTION FOR THE ALTERNATE OSCILLATOR

| Osc Type | Freq | C1 | C2 |
|------------------------------|-----------------------|--------------------|--------------------|
| LP | 32 kHz | TBD ⁽¹⁾ | TBD ⁽¹⁾ |
| Crystal to be Tested: | | | |
| 32.768 kHz | Epson C-001R32.768K-A | ±20 PPM | |

Note 1: Microchip suggests 33 pF as a starting point in validating the oscillator circuit.

- 2:** Higher capacitance increases the stability of the oscillator, but also increases the start-up time.
- 3:** Since each resonator/crystal has its own characteristics, the user should consult the resonator/crystal manufacturer for appropriate values of external components.
- 4:** Capacitor values are for design guidance only.

12.3 Timer1 Interrupt

The TMR1 register pair (TMR1H:TMR1L) increments from 0000h to FFFFh and rolls over to 0000h. The TMR1 Interrupt, if enabled, is generated on overflow which is latched in interrupt flag bit, TMR1IF (PIR registers). This interrupt can be enabled/disabled by setting/clearing TMR1 Interrupt Enable bit, TMR1IE (PIE registers).

12.4 Resetting Timer1 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 Timer1 and start an A/D conversion (if the A/D module is enabled).

Note: The special event triggers from the CCP1 module will not set interrupt flag bit, TMR1IF (PIR registers).

Timer1 must be configured for either Timer or Synchronized Counter mode to take advantage of this feature. If Timer1 is running in Asynchronous Counter mode, this Reset operation may not work.

In the event that a write to Timer1 coincides with a special event trigger from CCP1, the write will take precedence.

In this mode of operation, the CCPR1H:CCPR1L register pair effectively becomes the period register for Timer1.

12.5 Timer1 16-Bit Read/Write Mode

Timer1 can be configured for 16-bit reads and writes (see Figure 12-2). When the RD16 control bit (T1CON register) is set, the address for TMR1H is mapped to a buffer register for the high byte of Timer1. A read from TMR1L will load the contents of the high byte of Timer1 into the Timer1 High Byte Buffer register. This provides the user with the ability to accurately read all 16 bits of Timer1 without having to determine whether a read of the high byte, followed by a read of the low byte, is valid due to a rollover between reads.

A write to the high byte of Timer1 must also take place through the TMR1H Buffer register. Timer1 high byte is updated with the contents of TMR1H when a write occurs to TMR1L. This allows a user to write all 16 bits to both the high and low bytes of Timer1 at once.

The high byte of Timer1 is not directly readable or writable in this mode. All reads and writes must take place through the Timer1 High Byte Buffer register. Writes to TMR1H do not clear the Timer1 prescaler. The prescaler is only cleared on writes to TMR1L.

18.1 USART Baud Rate Generator (BRG)

The BRG supports both the Asynchronous and Synchronous modes of the USART. It is a dedicated 8-bit Baud Rate Generator. The SPBRG register controls the period of a free running, 8-bit timer. In Asynchronous mode, bit BRGH (TXSTA register) also controls the baud rate. In Synchronous mode, bit BRGH is ignored. Table 18-1 shows the formula for computation of the baud rate for different USART modes which only apply in Master mode (internal clock).

Given the desired baud rate and FOSC, the nearest integer value for the SPBRG register can be calculated using the formula in Table 18-1. From this, the error in baud rate can be determined.

Example 18-1 shows the calculation of the baud rate error for the following conditions:

FOSC = 16 MHz
Desired Baud Rate = 9600
BRGH = 0
SYNC = 0

It may be advantageous to use the high baud rate (BRGH = 1) even for slower baud clocks. This is because the $FOSC/(16(X + 1))$ equation can reduce the baud rate error in some cases.

Writing a new value to the SPBRG register causes the BRG timer to be reset (or cleared). This ensures the BRG does not wait for a timer overflow before outputting the new baud rate.

18.1.1 SAMPLING

The data on the RC7/RX/DT pin is sampled three times by a majority detect circuit to determine if a high or a low level is present at the RX pin.

EXAMPLE 18-1: CALCULATING BAUD RATE ERROR

| | | |
|----------------------|---|--|
| Desired Baud Rate | = | $FOSC/(64 (X + 1))$ |
| Solving for X: | | |
| | X | = $((FOSC/Desired\ Baud\ Rate)/64) - 1$ |
| | X | = $((16000000/9600)/64) - 1$ |
| | X | = $[25.042] = 25$ |
| Calculated Baud Rate | = | $16000000/(64 (25 + 1))$ |
| | = | 9615 |
| Error | = | $\frac{(Calculated\ Baud\ Rate - Desired\ Baud\ Rate)}{Desired\ Baud\ Rate}$ |
| | = | $(9615 - 9600)/9600$ |
| | = | 0.16% |

TABLE 18-1: BAUD RATE FORMULA

| SYNC | BRGH = 0 (Low Speed) | BRGH = 1 (High Speed) |
|------|--|---------------------------------|
| 0 | (Asynchronous) Baud Rate = $FOSC/(64 (X + 1))$ | Baud Rate = $FOSC/(16 (X + 1))$ |
| 1 | (Synchronous) Baud Rate = $FOSC/(4 (X + 1))$ | NA |

Legend: X = value in SPBRG (0 to 255)

TABLE 18-2: REGISTERS ASSOCIATED WITH BAUD RATE GENERATOR

| 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 |
|-------|------------------------------|-------|-------|-------|-------|-------|-------|-------|-------------------|---------------------------|
| TXSTA | CSRC | TX9 | TXEN | SYNC | — | BRGH | TRMT | TX9D | 0000 -010 | 0000 -010 |
| RCSTA | SPEN | RX9 | SREN | CREN | ADDEN | FERR | OERR | RX9D | 0000 000x | 0000 000u |
| SPBRG | Baud Rate Generator Register | | | | | | | | 0000 0000 | 0000 0000 |

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

19.2.5 CAN MODULE I/O CONTROL REGISTER

This register controls the operation of the CAN module's I/O pins in relation to the rest of the microcontroller.

REGISTER 19-32: CIOCON: CAN I/O CONTROL REGISTER

| U-0 | U-0 | R/W-0 | R/W-0 | U-0 | U-0 | U-0 | U-0 |
|-------|-----|--------|--------|-------|-----|-----|-----|
| — | — | ENDRHI | CANCAP | — | — | — | — |
| bit 7 | | | | bit 0 | | | |

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

bit 5 **ENDRHI:** Enable Drive High bit
 1 = CANTX pin will drive VDD when recessive
 0 = CANTX pin will tri-state when recessive

bit 4 **CANCAP:** CAN Message Receive Capture Enable bit
 1 = Enable CAN capture, CAN message receive signal replaces input on RC2/CCP1
 0 = Disable CAN capture, RC2/CCP1 input to CCP1 module

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 |

REGISTER 19-34: PIE3: PERIPHERAL INTERRUPT ENABLE REGISTER 3

| R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 | R/W-0 |
|-------|-------|-------|--------|--------|--------|--------|--------|
| IRXIE | WAKIE | ERRIE | TXB2IE | TXB1IE | TXB0IE | RXB1IE | RXB0IE |
| bit 7 | | | | bit 0 | | | |

- bit 7 **IRXIE:** CAN Invalid Received Message Interrupt Enable bit
1 = Enable invalid message received interrupt
0 = Disable invalid message received interrupt
- bit 6 **WAKIE:** CAN bus Activity Wake-up Interrupt Enable bit
1 = Enable bus activity wake-up interrupt
0 = Disable bus activity wake-up interrupt
- bit 5 **ERRIE:** CAN bus Error Interrupt Enable bit
1 = Enable CAN bus error interrupt
0 = Disable CAN bus error interrupt
- bit 4 **TXB2IE:** CAN Transmit Buffer 2 Interrupt Enable bit
1 = Enable Transmit Buffer 2 interrupt
0 = Disable Transmit Buffer 2 interrupt
- bit 3 **TXB1IE:** CAN Transmit Buffer 1 Interrupt Enable bit
1 = Enable Transmit Buffer 1 interrupt
0 = Disable Transmit Buffer 1 interrupt
- bit 2 **TXB0IE:** CAN Transmit Buffer 0 Interrupt Enable bit
1 = Enable Transmit Buffer 0 interrupt
0 = Disable Transmit Buffer 0 interrupt
- bit 1 **RXB1IE:** CAN Receive Buffer 1 Interrupt Enable bit
1 = Enable Receive Buffer 1 interrupt
0 = Disable Receive Buffer 1 interrupt
- bit 0 **RXB0IE:** CAN Receive Buffer 0 Interrupt Enable bit
1 = Enable Receive Buffer 0 interrupt
0 = Disable Receive Buffer 0 interrupt

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 |

24.2.2 WDT POSTSCALER

The WDT has a postscaler that can extend the WDT Reset period. The postscaler is selected at the time of device programming by the value written to the CONFIG2H Configuration register.

FIGURE 24-1: WATCHDOG TIMER BLOCK DIAGRAM

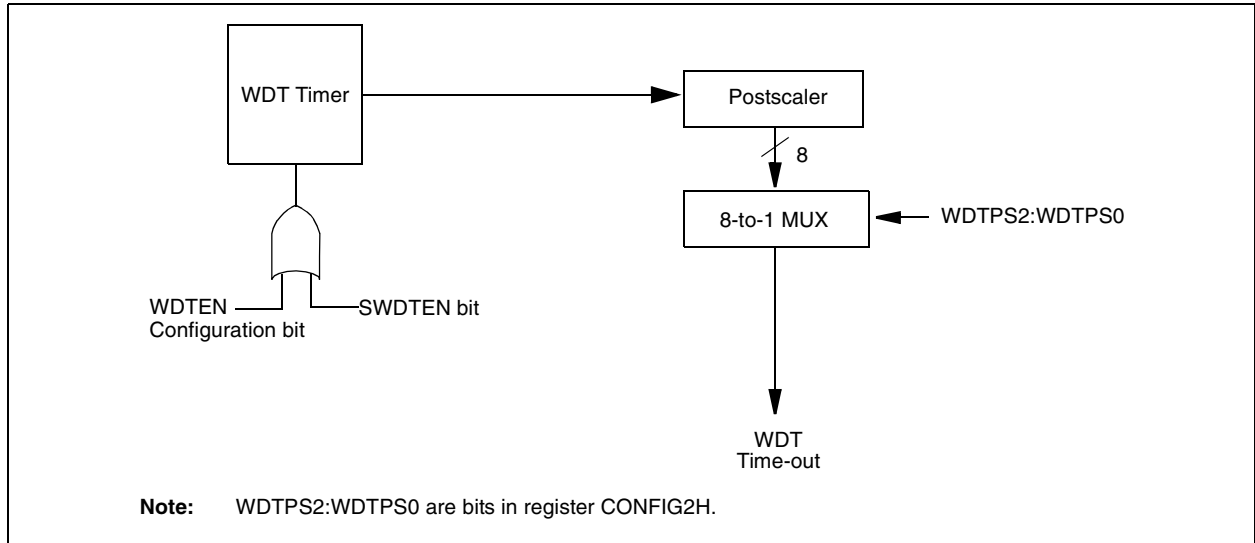


TABLE 24-2: SUMMARY OF WATCHDOG TIMER REGISTERS

| Name | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|----------|-------|-------|-------|-----------------|-----------------|-----------------|------------------|------------------|
| CONFIG2H | — | — | — | — | WDTPS2 | WDTPS1 | WDTPS0 | WDTEN |
| RCON | IPEN | — | — | \overline{RI} | \overline{TO} | \overline{PD} | \overline{POR} | \overline{BOR} |
| WDTCON | — | — | — | — | — | — | — | SWDTEN |

Legend: Shaded cells are not used by the Watchdog Timer.

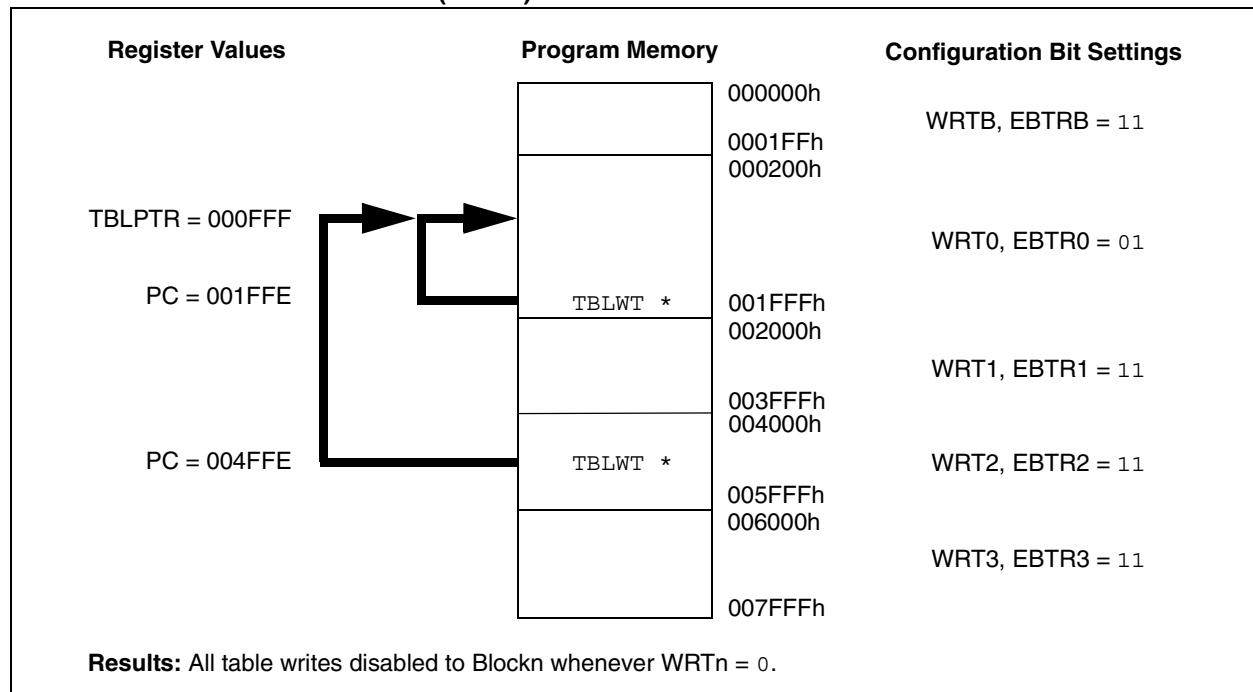
24.4.1 PROGRAM MEMORY CODE PROTECTION

The user memory may be read to or written from any location using the table read and table write instructions. The device ID may be read with table reads. The Configuration registers may be read and written with the table read and table write instructions.

In user mode, the CPn bits have no direct effect. CPn bits inhibit external reads and writes. A block of user memory may be protected from table writes if the WRTn configuration bit is '0'. The EBTRn bits control table reads. For a block of user memory with the EBTRn bit set to '0', a table read instruction that executes from within that block is allowed to read. A table read instruction that executes from a location outside of that block is not allowed to read and will result in reading '0's. Figures 24-4 through 24-6 illustrate table write and table read protection.

Note: Code protection bits may only be written to a '0' from a '1' state. It is not possible to write a '1' to a bit in the '0' state. Code protection bits are only set to '1' by a full chip erase or block erase function. The full chip erase and block erase functions can only be initiated via ICSP or an external programmer.

FIGURE 24-4: TABLE WRITE (WRTn) DISABLED



25.0 INSTRUCTION SET SUMMARY

The PIC18 instruction set adds many enhancements to the previous PICmicro instruction sets, while maintaining an easy migration from these PICmicro instruction sets.

Most instructions are a single program memory word (16 bits) but there are three instructions that require two program memory locations.

Each single-word instruction is a 16-bit word divided into an opcode, which specifies the instruction type and one or more operands, which further specify the operation of the instruction.

The instruction set is highly orthogonal and is grouped into four basic categories:

- **Byte-oriented** operations
- **Bit-oriented** operations
- **Literal** operations
- **Control** operations

The PIC18 instruction set summary in Table 25-2 lists **byte-oriented**, **bit-oriented**, **literal** and **control** operations. Table 25-1 shows the opcode field descriptions.

Most **byte-oriented** instructions have three operands:

1. The file register (specified by 'f')
2. The destination of the result (specified by 'd')
3. The accessed memory (specified by 'a')

The file register designator 'f' specifies which file register is to be used by the instruction.

The destination designator 'd' specifies where the result of the operation is to be placed. If 'd' is zero, the result is placed in the WREG register. If 'd' is one, the result is placed in the file register specified in the instruction.

All **bit-oriented** instructions have three operands:

1. The file register (specified by 'f')
2. The bit in the file register (specified by 'b')
3. The accessed memory (specified by 'a')

The bit field designator 'b' selects the number of the bit affected by the operation, while the file register designator 'f' represents the number of the file in which the bit is located.

The **literal** instructions may use some of the following operands:

- A literal value to be loaded into a file register (specified by 'k')
- The desired FSR register to load the literal value into (specified by 'f')
- No operand required (specified by '—')

The **control** instructions may use some of the following operands:

- A program memory address (specified by 'n')
- The mode of the **CALL** or **RETURN** instructions (specified by 's')
- The mode of the table read and table write instructions (specified by 'm')
- No operand required (specified by '—')

All instructions are a single word, except for three double-word instructions. These three instructions were made double-word instructions so that all the required information is available in these 32 bits. In the second word, the 4 MSBs are '1's. If this second word is executed as an instruction (by itself), it will execute as a **NOB**.

All single-word instructions are executed in a single instruction cycle, unless a conditional test is true or the program counter is changed as a result of the instruction. In these cases, the execution takes two instruction cycles, with the additional instruction cycle(s) executed as a **NOB**.

The double-word instructions execute in two instruction cycles.

One instruction cycle consists of four oscillator periods. Thus, for an oscillator frequency of 4 MHz, the normal instruction execution time is 1 μ s. If a conditional test is true, or the program counter is changed as a result of an instruction, the instruction execution time is 2 μ s. Two-word branch instructions (if true) would take 3 μ s.

Figure 25-1 shows the general formats that the instructions can have.

All examples use the format 'nnh' to represent a hexadecimal number, where 'h' signifies a hexadecimal digit.

The Instruction Set Summary, shown in Table 25-2, lists the instructions recognized by the Microchip MPASM™ Assembler.

Section 25.2 "Instruction Set" provides a description of each instruction.

25.1 Read-Modify-Write Operations

Any instruction that specifies a file register as part of the instruction performs a Read-Modify-Write (R-M-W) operation. The register is read, the data is modified and the result is stored according to either the instruction or the destination designator 'd'. A read operation is performed on a register even if the instruction writes to that register.

For example, a "**CLRF PORTB**" instruction will read PORTB, clear all the data bits, then write the result back to PORTB. This example would have the unintended result that the condition that sets the RBIF flag would be cleared.

BNC Branch if Not Carry

Syntax: [label] BNC n

Operands: $-128 \leq n \leq 127$

Operation: if Carry bit is '0'
 $(PC) + 2 + 2n \rightarrow PC$

Status Affected: None

Encoding:

| | | | |
|------|------|------|------|
| 1110 | 0011 | nnnn | nnnn |
|------|------|------|------|

Description: If the Carry bit is '0', then the program will branch.
 The 2's complement number '2n' is added 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 then a two-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:

If Jump:

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

If No Jump:

| Q1 | Q2 | Q3 | Q4 |
|--------|------------------|--------------|--------------|
| Decode | Read literal 'n' | Process Data | No operation |

Example: HERE BNC Jump

Before Instruction

PC = address (HERE)

After Instruction

If Carry = 0;
 PC = address (Jump)
 If Carry = 1;
 PC = address (HERE + 2)

BNN Branch if Not Negative

Syntax: [label] BNN n

Operands: $-128 \leq n \leq 127$

Operation: if Negative bit is '0'
 $(PC) + 2 + 2n \rightarrow PC$

Status Affected: None

Encoding:

| | | | |
|------|------|------|------|
| 1110 | 0111 | nnnn | nnnn |
|------|------|------|------|

Description: If the Negative bit is '0', then the program will branch.
 The 2's complement number '2n' is added 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 then a two-cycle instruction.

Words: 1

Cycles: 1(2)

Q Cycle Activity:

If Jump:

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

If No Jump:

| Q1 | Q2 | Q3 | Q4 |
|--------|------------------|--------------|--------------|
| Decode | Read literal 'n' | Process Data | No operation |

Example: HERE BNN Jump

Before Instruction

PC = address (HERE)

After Instruction

If Negative = 0;
 PC = address (Jump)
 If Negative = 1;
 PC = address (HERE + 2)

TBLRD Table Read

Syntax: [*label*] TBLRD (*, *+, *-, +*)

Operands: None

Operation:

```

if TBLRD *,
  (Prog Mem (TBLPTR)) → TABLAT;
  TBLPTR – No Change;
if TBLRD *+,
  (Prog Mem (TBLPTR)) → TABLAT;
  (TBLPTR) + 1 → TBLPTR;
if TBLRD *-,
  (Prog Mem (TBLPTR)) → TABLAT;
  (TBLPTR) – 1 → TBLPTR;
if TBLRD +*,
  (TBLPTR) + 1 → TBLPTR;
  (Prog Mem (TBLPTR)) → TABLAT

```

Status Affected: None

| | | | | |
|-----------|------|------|------|---|
| Encoding: | 0000 | 0000 | 0000 | 10nn nn=0 * =1 *+ =2 *- =3 +* |
|-----------|------|------|------|---|

Description: This instruction is used to read the contents of Program Memory (P.M.). To address the program memory, a pointer called Table Pointer (TBLPTR) is used. The TBLPTR (a 21-bit pointer) points to each byte in the program memory. TBLPTR has a 2-Mbyte address range.

TBLPTR[0] = 0: Least Significant Byte of Program Memory Word

TBLPTR[0] = 1: Most Significant Byte of Program Memory Word

The TBLRD instruction can modify the value of TBLPTR as follows:

- no change
- post-increment
- post-decrement
- pre-increment

Words: 1

Cycles: 2

Q Cycle Activity:

| Q1 | Q2 | Q3 | Q4 |
|--------------|------------------------------------|--------------|-----------------------------|
| Decode | No operation | No operation | No operation |
| No operation | No operation (Read Program Memory) | No operation | No operation (Write TABLAT) |

Example 1: TBLRD *+ ;

Before Instruction

```

TABLAT      = 0x55
TBLPTR      = 0x00A356
MEMORY(0x00A356) = 0x34

```

After Instruction

```

TABLAT      = 0x34
TBLPTR      = 0x00A357

```

Example 2: TBLRD +* ;

Before Instruction

```

TABLAT      = 0xAA
TBLPTR      = 0x01A357
MEMORY(0x01A357) = 0x12
MEMORY(0x01A358) = 0x34

```

After Instruction

```

TABLAT      = 0x34
TBLPTR      = 0x01A358

```

PIC18FXX8

XORWF Exclusive OR W with f

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

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

Operation: (W) .XOR. (f) → dest

Status Affected: N, Z

Encoding:

| | | | |
|------|------|------|------|
| 0001 | 10da | ffff | ffff |
|------|------|------|------|

Description: Exclusive OR the contents of W with register 'f'. If 'd' is '0', the result is stored in W. 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', 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 to destination |

Example: XORWF REG

Before Instruction

REG = 0xAF
W = 0xB5

After Instruction

REG = 0x1A
W = 0xB5

PIC18FXX8

FIGURE 27-4: LOW-VOLTAGE DETECT CHARACTERISTICS

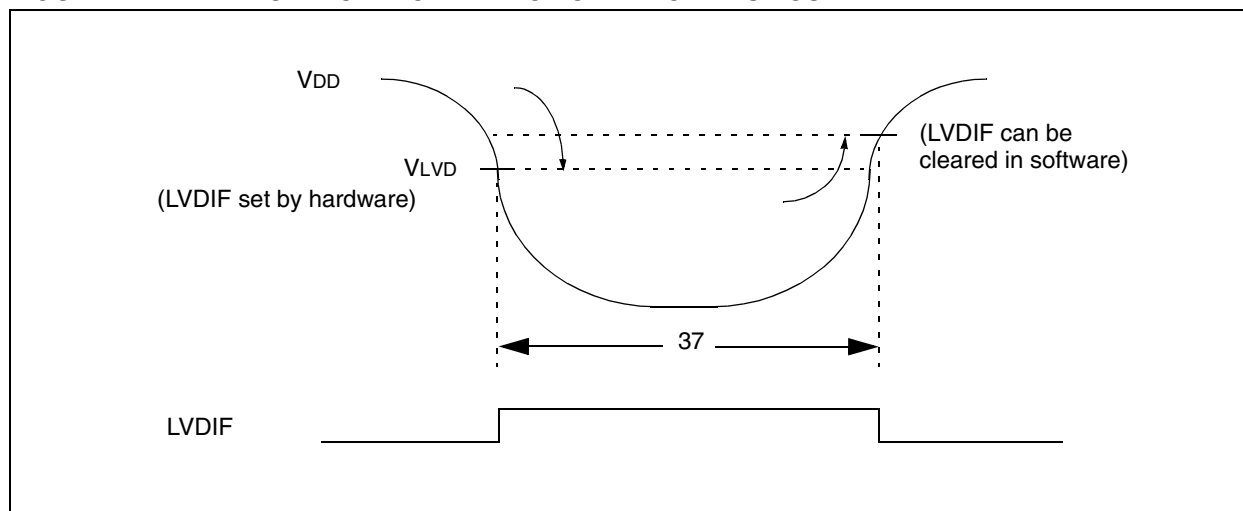


TABLE 27-1: LOW-VOLTAGE DETECT CHARACTERISTICS

| Low-Voltage Detect Characteristics | | Standard Operating Conditions (unless otherwise stated) Operating temperature $-40^{\circ}\text{C} \leq T_A \leq +85^{\circ}\text{C}$ for industrial $-40^{\circ}\text{C} \leq T_A \leq +125^{\circ}\text{C}$ for extended | | | | | | |
|------------------------------------|--------|--|------------|------|------|------|-------|-----------------------------|
| Param No. | Symbol | Characteristic | | Min | Typ | Max | Units | Conditions |
| D420 | VLVD | LVD Voltage | LVV = 0001 | 1.96 | 2.06 | 2.16 | V | $T \geq 25^{\circ}\text{C}$ |
| | | | LVV = 0010 | 2.16 | 2.27 | 2.38 | V | $T \geq 25^{\circ}\text{C}$ |
| | | | LVV = 0011 | 2.35 | 2.47 | 2.59 | V | $T \geq 25^{\circ}\text{C}$ |
| | | | LVV = 0100 | 2.43 | 2.58 | 2.69 | V | |
| | | | LVV = 0101 | 2.64 | 2.78 | 2.92 | V | |
| | | | LVV = 0110 | 2.75 | 2.89 | 3.03 | V | |
| | | | LVV = 0111 | 2.95 | 3.1 | 3.26 | V | |
| | | | LVV = 1000 | 3.24 | 3.41 | 3.58 | V | |
| | | | LVV = 1001 | 3.43 | 3.61 | 3.79 | V | |
| | | | LVV = 1010 | 3.53 | 3.72 | 3.91 | V | |
| | | | LVV = 1011 | 3.72 | 3.92 | 4.12 | V | |
| | | | LVV = 1100 | 3.92 | 4.13 | 4.34 | V | |
| | | | LVV = 1101 | 4.07 | 4.33 | 4.59 | V | |
| | | | LVV = 1110 | 4.36 | 4.64 | 4.92 | V | |

FIGURE 27-8: RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER AND POWER-UP TIMER TIMING

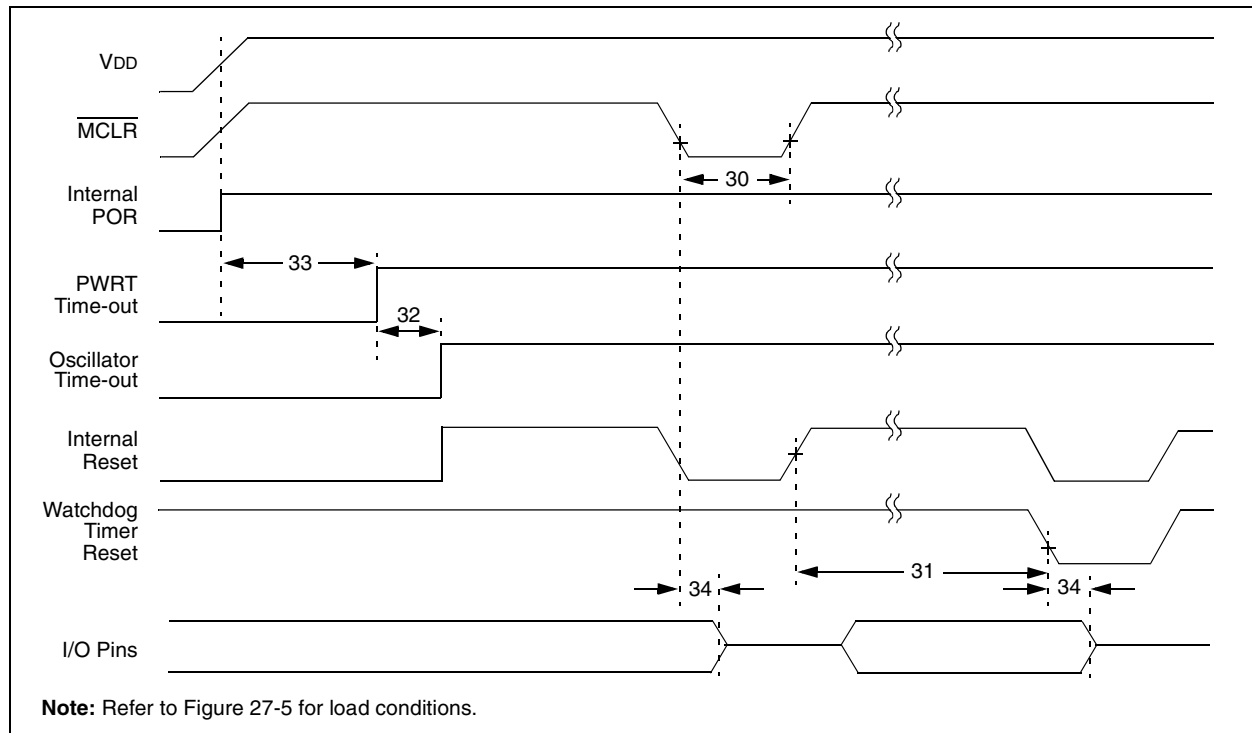


FIGURE 27-9: BROWN-OUT RESET AND LOW-VOLTAGE DETECT TIMING

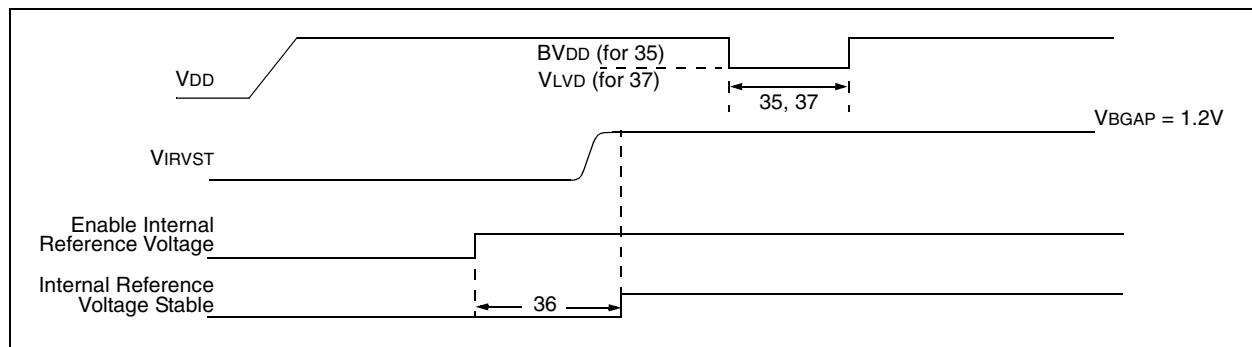


TABLE 27-9: RESET, WATCHDOG TIMER, OSCILLATOR START-UP TIMER, POWER-UP TIMER, BROWN-OUT RESET AND LOW-VOLTAGE DETECT REQUIREMENTS

| Param No. | Symbol | Characteristic | Min | Typ | Max | Units | Conditions |
|-----------|--------|--|-----------|-----|-----------|-------|---------------------------|
| 30 | TmcL | 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 | 1024 TOSC | — | 1024 TOSC | — | TOSC = OSC1 period |
| 33 | TPWRT | Power-up Timer Period | 28 | 72 | 132 | ms | |
| 34 | TIOZ | I/O High-Impedance from MCLR Low or Watchdog Timer Reset | — | 2 | — | μs | |
| 35 | TBOR | Brown-out Reset Pulse Width | 200 | — | — | μs | For VDD ≤ BVDD (see D005) |
| 36 | TIRVST | Time for Internal Reference Voltage to become stable | — | 20 | 50 | μs | |
| 37 | TLVD | Low-Voltage Detect Pulse Width | 200 | — | — | μs | For VDD ≤ VLVD (see D420) |

FIGURE 27-21: USART SYNCHRONOUS TRANSMISSION (MASTER/SLAVE) TIMING

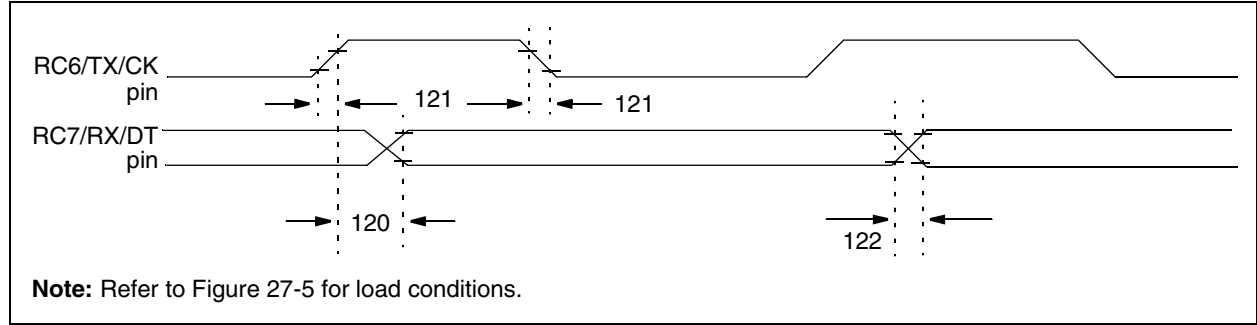


TABLE 27-21: USART SYNCHRONOUS TRANSMISSION REQUIREMENTS

| Param No. | Symbol | Characteristic | Min | Max | Units | Conditions |
|-----------|----------|--|------------|-----|-------|------------|
| 120 | TckH2dtV | SYNC XMIT (Master & Slave) Clock High to Data-Out Valid | PIC18FXX8 | — | 50 | ns |
| | | | PIC18LFXX8 | — | 150 | ns |
| 121 | Tckrf | Clock Out Rise Time and Fall Time (Master mode) | PIC18FXX8 | — | 25 | ns |
| | | | PIC18LFXX8 | — | 60 | ns |
| 122 | Tdtrf | Data-Out Rise Time and Fall Time | PIC18FXX8 | — | 25 | ns |
| | | | PIC18LFXX8 | — | 60 | ns |

FIGURE 27-22: USART SYNCHRONOUS RECEIVE (MASTER/SLAVE) TIMING

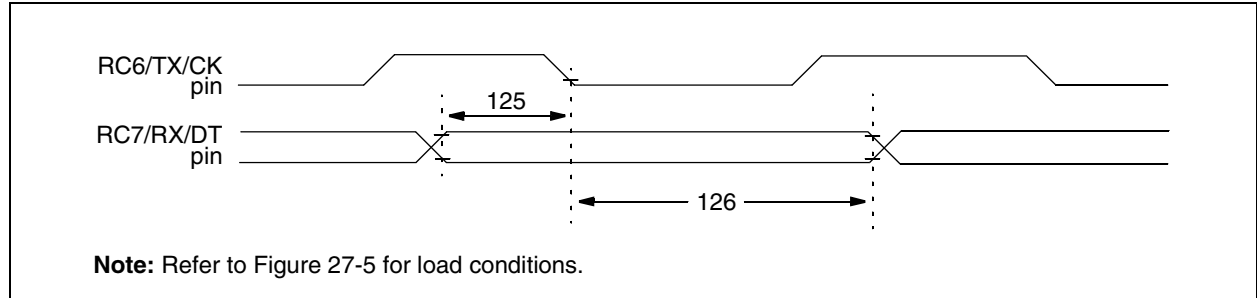


TABLE 27-22: USART SYNCHRONOUS RECEIVE REQUIREMENTS

| Param No. | Symbol | Characteristic | Min | Max | Units | Conditions |
|-----------|----------|---|-----|-----|-------|------------|
| 125 | TdtV2ckl | SYNC RCV (Master & Slave) Data-Hold before CK ↓ (DT hold time) | 10 | — | ns | |
| 126 | TckL2dtl | Data-Hold after CK ↓ (DT hold time) | 15 | — | ns | |

PIC18FXX8

FIGURE 28-11: TYPICAL AND MAXIMUM I_{DD} vs. V_{DD}
(TIMER1 AS MAIN OSCILLATOR 32.768 kHz, C1 AND C2 = 47 pF)

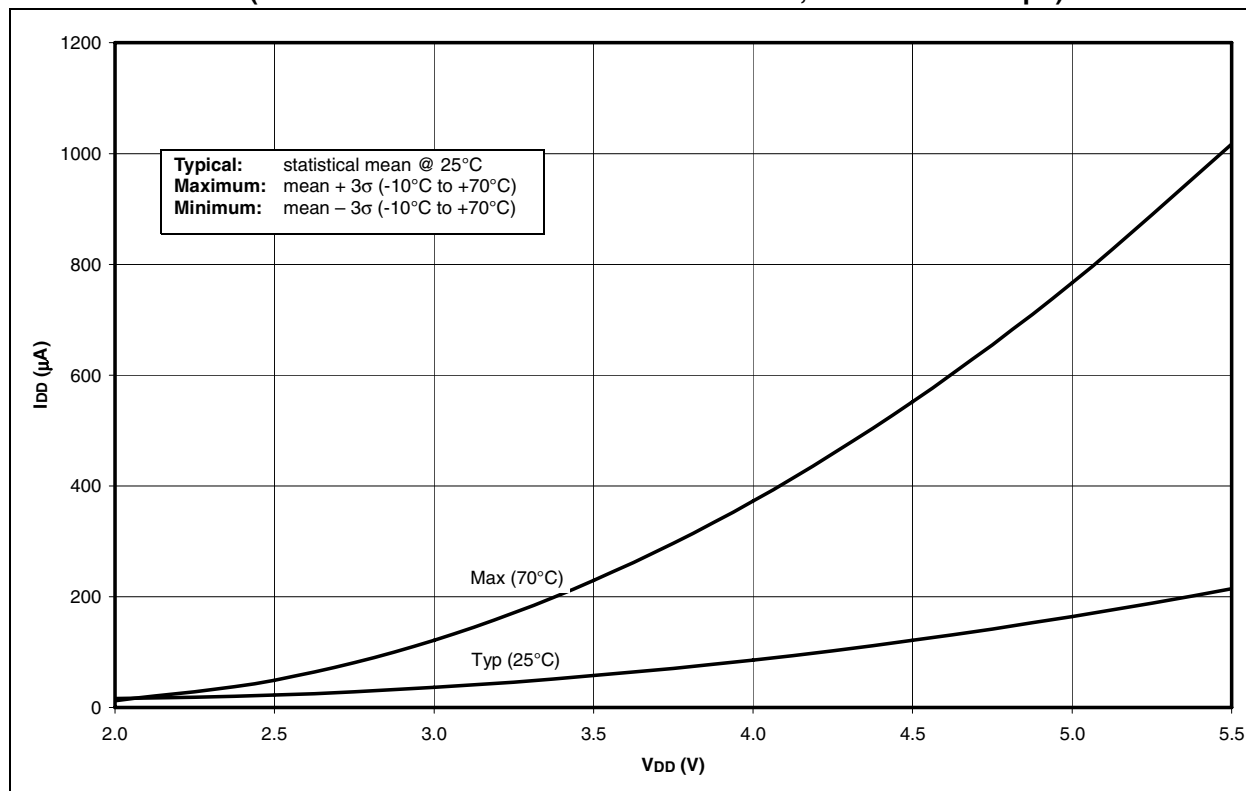
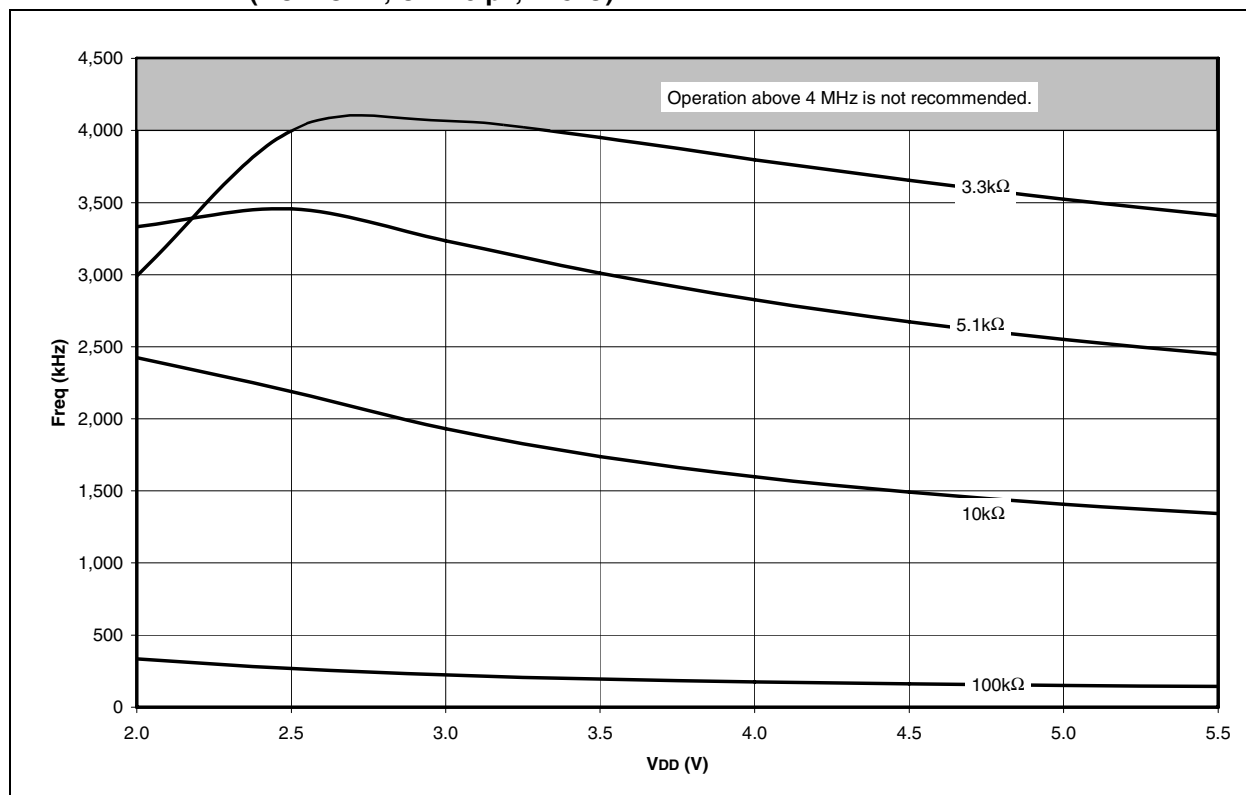


FIGURE 28-12: AVERAGE F_{osc} vs. V_{DD} FOR VARIOUS VALUES OF R
(RC MODE, C = 20 pF, +25°C)



APPENDIX C: DEVICE MIGRATIONS

This section is intended to describe the functional and electrical specification differences when migrating between functionally similar devices (such as from a PIC16C74A to a PIC16C74B).

Not Applicable

APPENDIX D: MIGRATING FROM OTHER PICmicro® DEVICES

This discusses some of the issues in migrating from other PICmicro devices to the PIC18FXX8 family of devices.

D.1 PIC16CXXX to PIC18FXX8

See Application Note AN716 “*Migrating Designs from PIC16C74A/74B to PIC18C442*” (DS00716).

D.2 PIC17CXXX to PIC18FXX8

See Application Note AN726 “*PIC17CXXX to PIC18CXXX Migration*” (DS00726).