This document includes the programming specifications for the following devices:

- PIC16F73
- PIC16F76
- PIC16F74
- PIC16F77

## 1.0 PROGRAMMING THE PIC16F7X

The PIC16F7X is programmed using a serial method. The Serial mode allows the PIC16F7X to be programmed while in the users’ system, allowing for increased design flexibility. This programming specification applies to PIC16F7X devices in all packages.

### 1.1 Hardware Requirements

The PIC16F7X requires two programmable power supplies, one for VDD (2.0V to 5.5V) and the other for VPP of 12.75V to 13.25V. Both supplies should have a minimum resolution of 0.25V.

### 1.2 Programming Mode

The Programming mode for the PIC16F7X allows programming of user program memory, special locations used for ID, and the configuration word.

**TABLE 1-1: PIN DESCRIPTIONS (DURING PROGRAMMING): PIC16F7X**

<table>
<thead>
<tr>
<th>Pin Name</th>
<th>During Programming</th>
<th>Pin Type</th>
<th>Pin Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RB6/PGC</td>
<td>CLOCK</td>
<td>I</td>
<td>Clock Input</td>
</tr>
<tr>
<td>RB7/PGD</td>
<td>DATA</td>
<td>I/O</td>
<td>Data Input/Output</td>
</tr>
<tr>
<td>MCLR/VPP</td>
<td>MODE CONTROL</td>
<td>P</td>
<td>Program Mode Select</td>
</tr>
<tr>
<td>VDD</td>
<td>VDD</td>
<td>P</td>
<td>Power Supply</td>
</tr>
<tr>
<td>Vss</td>
<td>Vss</td>
<td>P</td>
<td>Ground</td>
</tr>
</tbody>
</table>

Legend: I = Input, O = Output, P = Power
2.0 PROGRAM MODE ENTRY

2.1 User Program Memory Map

The user memory space extends from 0x0000 to 0x1FFF (8K), or 0x0000 to 0x0FFF (4K). Table 2-1 shows the actual implementation of program memory in the PIC16F7X family. Configuration memory begins at 0x2000, and continues to 0x3FFF. The PC will increment from 0x0000 to 0x1FFF and wrap to 0x0000, 0x2000 to 0x3FFF and wrap around to 0x2000 (not to 0x0000).

Once in configuration memory, the highest bit of the PC stays a ‘1’, thus always pointing to the configuration memory. The only way to point to program memory is to reset the part and re-enter Program/Verify mode, as described in Section 2.3.

Configuration memory is selected when the PC points to any address in the range of 0x2000-0x201F; however, only locations 0x2000 through 0x2007 are implemented. Addressing locations beyond 0x201F will access program memory (see Figure 2-1).

TABLE 2-1: IMPLEMENTATION OF PROGRAM MEMORY IN THE PIC16F7X FAMILY

<table>
<thead>
<tr>
<th>Device</th>
<th>Program Memory Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIC16F73</td>
<td>0x0000 – 0x0FFF (4K)</td>
</tr>
<tr>
<td>PIC16F74</td>
<td>0x0000 – 0x0FFF (4K)</td>
</tr>
<tr>
<td>PIC16F76</td>
<td>0x0000 – 0x1FFF (8K)</td>
</tr>
<tr>
<td>PIC16F77</td>
<td>0x0000 – 0x1FFF (8K)</td>
</tr>
</tbody>
</table>

2.2 ID Locations

A user may store identification information (ID) in four ID locations mapped to [0x2000:0x2003]. It is recommended that each ID location word is written as ‘11 1111 1000 bbbb’, where ‘bbbbb’ is ID information. The ID locations can be read after code protection is enabled.

To understand the program memory read mechanism after code protection is enabled, refer to Section 4.0. Table 4-1 shows specific calculations and behavior for each of the PIC16F7X devices.
FIGURE 2-1: PROGRAM MEMORY MAPPING

<table>
<thead>
<tr>
<th>Address</th>
<th>4K words</th>
<th>8K words</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0</td>
<td>Implemented</td>
<td>Implemented</td>
</tr>
<tr>
<td>0x1</td>
<td>Implemented</td>
<td>Implemented</td>
</tr>
<tr>
<td>0x2</td>
<td>Implemented</td>
<td>Implemented</td>
</tr>
<tr>
<td>0x3</td>
<td>Implemented</td>
<td>Implemented</td>
</tr>
<tr>
<td>0x4</td>
<td>Implemented</td>
<td>Implemented</td>
</tr>
<tr>
<td>0x5</td>
<td>Implemented</td>
<td>Implemented</td>
</tr>
<tr>
<td>0x6</td>
<td>Implemented</td>
<td>Implemented</td>
</tr>
<tr>
<td>0x7</td>
<td>Implemented</td>
<td>Implemented</td>
</tr>
<tr>
<td>0x00</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x01</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x02</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x03</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x04</td>
<td>Device ID</td>
<td>Accesses 0x0020 to 0xFF</td>
</tr>
<tr>
<td>0x05</td>
<td>Reserved</td>
<td>Accesses 0x0020 to 0x1FFF</td>
</tr>
<tr>
<td>0x06</td>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>0x07</td>
<td>Configuration Word</td>
<td>Accesses 0x0020 to 0x1FFF</td>
</tr>
<tr>
<td>0x0020</td>
<td>0x0020 to 0x0FFF</td>
<td>0x0020 to 0x1FFF</td>
</tr>
<tr>
<td>0x1FFF</td>
<td>0x0020 to 0x0FFF</td>
<td>0x0020 to 0x1FFF</td>
</tr>
<tr>
<td>0x3FFF</td>
<td>0x0020 to 0x0FFF</td>
<td>0x0020 to 0x1FFF</td>
</tr>
</tbody>
</table>

Device ID locations: 0x2000h to 0x2007h
Configuration Word locations: 0x2008h to 0x201Fh
2.3 Program/Verify Mode

The Program/Verify mode is entered by holding pins RB6 and RB7 low, while raising MCLR pin from VIL to VPP. Once in this mode, the user program memory and the configuration memory can be accessed and programmed in serial fashion. (RB6 and RB7 are Schmitt Trigger Inputs in this mode.)

The sequence that enters the device into the Program/Verify mode, places all other logic into the RESET state. All I/O pins are in the RESET state (high impedance inputs).

A device RESET will clear the PC and point to address 0x0000. The ‘Increment Address’ command will increment the PC. The ‘Load Configuration’ command will set the PC to 0x2000. The available commands are shown in Table 2-2.

The normal sequence for programming two program memory words at a time is as follows:
1. Issue the ‘Load Data’ command to load a word at the current (even) program memory address.
2. Issue an ‘Increment Address’ command.
3. Load a word at the current (odd) program memory address using the ‘Load Data’ command.
4. Issue a ‘Begin Programming’ command to begin programming.
5. Wait tprog (about 1 ms).
7. Increment to the next address.
8. Repeat this sequence as required to write program and configuration memory.

The alternative sequence for programming one program memory word at a time is as follows:
1. Set a word for the current memory location using the ‘Load Data’ command.
2. Issue a ‘Begin Programming’ command to begin programming.
3. Wait tprog.
4. Issue an ‘End Programming’ command.
5. Increment to the next address.
6. Repeat this alternative sequence as required to write program and configuration memory.

The address and program counter is reset to 0x0000 by resetting the device (taking MCLR below VIL) and re-entering Programming mode. Program and configuration memory may then be read or verified using the Read Data and Increment Address commands.

2.3.1 SERIAL PROGRAM/VERIFY OPERATION

RB6 is used as a clock input pin, and RB7 is used for entering command bits and data input/output. To enter a command, the clock pin (RB6) is pulsed six times. Each command bit is latched on the falling edge of the clock (RB6), with the Least Significant bit (LSb) of the command being input first. The data on pin RB7 needs a minimum setup (tset1) and hold time (thold1) with respect to the falling edge of the clock. The read and load commands are specified to have a minimum delay (tdly1) between the command and data. After this delay, the clock pin is cycled 16 times, with the first cycle being a START bit (0) and the last cycle being a STOP bit (0). Data is transferred LSB first (see Figure 5-1).

During a read operation, the LSB will be output on pin RB7 on the rising edge of the second clock pulse and during a load operation, the LSB will be latched on the falling edge of the second clock pulse. A minimum delay (tdly2) is required between consecutive commands (see Figure 5-2).

To allow for decoding of commands and reversal of data pin configuration, a time separation of at least (tdly1) is required between a command and a data word, or another command (see Figure 5-3).

The available commands are listed below:
- Load Configuration
- Load Data for Memory
- Read Data from Memory
- Increment Address
- Begin Programming
- Bulk Erase Program Memory
- End Programming

<table>
<thead>
<tr>
<th>Command</th>
<th>Mapping (LSb … MSb)</th>
<th>Data (LSb first)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load Configuration (Set PC = 2000h)</td>
<td>0 0 0 0 X X</td>
<td>0, data (14), 0</td>
</tr>
<tr>
<td>Load Data for Memory</td>
<td>0 1 0 0 X X</td>
<td>0, data (14), 0</td>
</tr>
<tr>
<td>Read Data from Memory</td>
<td>0 0 1 0 X X</td>
<td>0, data (14), 0</td>
</tr>
<tr>
<td>Increment Address</td>
<td>0 1 1 0 X X</td>
<td></td>
</tr>
<tr>
<td>Begin Programming</td>
<td>0 0 0 1 X X</td>
<td></td>
</tr>
<tr>
<td>Bulk Erase Program Memory (Chip Erase)</td>
<td>1 0 0 1 X X</td>
<td></td>
</tr>
<tr>
<td>End Programming</td>
<td>0 1 1 1 X X</td>
<td></td>
</tr>
</tbody>
</table>

TABLE 2-2: COMMAND MAPPING FOR PIC16F7X
2.3.1.1 Load Configuration
After receiving the Load Configuration command, the PC will be set to 0x2000 and the data sent with the command is discarded. The four ID locations and the configuration word can then be programmed using the normal programming sequence, as described in Section 2.3. A description of the memory mapping schemes of the program memory for normal operation and Configuration mode operation is shown in Figure 2-1. After the configuration memory is entered, the only way to get back to the user program memory is to exit the Program/Verify Test mode by taking MCLR low.

2.3.1.2 Load Data for Memory
The device will load in a 14-bit “data word” (LSb first) when 16 cycles are applied, as described previously. A timing diagram for the load data command is shown in Figure 5-1.

2.3.1.3 Read Data from Memory
The device will transmit data bits out of the memory (program or configuration) currently addressed by the PC, starting with the second rising edge of the clock input. RB7 will go into Output mode on the second rising clock edge and will revert back to Input mode (high impedance) after the 16th rising edge. Data is sent out LSB first. A timing diagram for this command is shown in Figure 5-2.

If the device is code protected, user program memory will read all ‘0’s. Configuration memory can still be read.

2.3.1.4 Increment Address
The PC is incremented by one. A timing diagram for this command is shown in Figure 5-3.

2.3.1.5 Begin Programming
A ‘Load Data’ command must be issued before every ‘Begin Programming’ command. Programming of memory (configuration or program) will begin after this command is received and decoded. Programming requires (tprog) time and is terminated using an ‘End Programming’ command.

2.3.1.6 Chip Erase (Program Memory)
Erasure of configuration and program memory begins after this command is received and decoded. The erase sequence is self-timed and it is not necessary to issue an ‘End Programming’ command, only to wait for the appropriate time interval (tera) for the entire erase sequence, before issuing another command.

This procedure will disable code protection (code protect bit = 1); however, all data within the program memory will be erased when this command is executed and thus, the security of the data or code is not compromised.

**Note:** All CHIP ERASE operations must take place with VDD between 4.75V and 5.25V (i.e., VDDP).

2.4 Programming Algorithm Requires Variable VDD
The PIC16F7X uses an intelligent algorithm. The algorithm calls for program verification at VDDAPP.

The actual chip erase and programming must be done with VDD in the VDDP range (See Table 5-1).

VDDP = VDD range required during programming
VDDAPP = VDD in the target application

Programmers must verify the PIC16F7X at VDDAPP. Since Microchip may introduce future versions of the PIC16F7X with a broader VDD range, it is best that these levels are user selectable (defaults are OK).

**Note:** Any programmer not meeting this requirement may only be classified as a “prototype” or “development” programmer, but not a “production quality” programmer.
FIGURE 2-2: PROGRAMMING METHOD FLOW CHART (SHEET 1 OF 3)

```
START

CHIP ERASE

BLANK CHECK
AT VDD = VDDMIN

PASS?
YES

NO

REPORT POSSIBLE
ERASE FAILURE.
CONTINUE
PROGRAMMING
AT USER’S OPTION

PROGRAM TWO
LOCATIONS
VPP = 12.75 TO 13.25V
VDD = VDDP

ALL
LOCATIONS
DONE?
YES

NO

VERIFY ALL
PROGRAM MEMORY
LOCATIONS AT
VDD = VDDAPP

PASS?
YES

NO

REPORT VERIFY
FAILURE
AT VDDAPP

LOAD
CONFIGURATION
(PC = 2000h)

PROGRAM TWO
ID LOCATIONS
VPP = 12.75 TO 13.25V
VDD = VDDP

ALL
ID LOCATIONS
DONE?
YES

NO

A
```
FIGURE 2-3: PROGRAMMING METHOD FLOW CHART (SHEET 2 OF 3)

A

INCREMENT ADDRESS TO CONFIGURATION WORD

LOAD DATA FOR MEMORY

BEGIN PROGRAMMING

WAIT tprog

END PROGRAMMING

VERIFY ALL CONFIGURATION MEMORY LOCATIONS AT VDD = VDDAPP

PASS?

REPORT VERIFY ERROR

YES

DONE

NO
FIGURE 2-4: PROGRAMMING METHOD FLOW CHART (SHEET 3 OF 3)

PROGRAM TWO LOCATIONS

START

LOAD DATA FOR MEMORY (EVEN ADDRESS)

INCREMENT ADDRESS

LOAD DATA FOR MEMORY (ODD ADDRESS)

BEGIN PROGRAMMING

WAIT tprog

END PROGRAMMING

INCREMENT ADDRESS

RETURN

VERIFY ALL LOCATIONS

START

RESET DEVICE, RETURN TO PROGRAMMING MODE (VDD = VDDAPP)

HAS PROGRAM MEMORY BEEN VERIFIED?

YES

LOAD CONFIGURATION

NO

READ DATA FROM MEMORY

COMPARE DATA TO EXPECTED DATA

DOES DATA MATCH?

NO

FAIL

YES

INCREMENT ADDRESS

ALL LOCATIONS VERIFIED?

NO

RETURN

YES

PASS

INCREMENT ADDRESS

RETURN
3.0 CONFIGURATION WORD

The PIC16F7X has configuration bits in a configuration word located at 0x2007. These bits can be cleared (reads ‘0’), or left unchanged (reads ‘1’), to select various device configurations.

3.1 Device ID Word

The device ID word for the PIC16F7X is located at 2006h. The nine Most Significant bits are the device ID number, while the five Least Significant bits are the device revision number.

REGISTER 3-1: CONFIGURATION WORD FOR PIC16F7X

<table>
<thead>
<tr>
<th>Device</th>
<th>Device ID Word (0x2006)</th>
<th>Dev</th>
<th>Rev</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIC16F73</td>
<td>00 0110 000</td>
<td>n</td>
<td>nnnn</td>
</tr>
<tr>
<td>PIC16F74</td>
<td>00 0110 001</td>
<td>n</td>
<td>nnnn</td>
</tr>
<tr>
<td>PIC16F76</td>
<td>00 0110 010</td>
<td>n</td>
<td>nnnn</td>
</tr>
<tr>
<td>PIC16F77</td>
<td>00 0110 011</td>
<td>n</td>
<td>nnnn</td>
</tr>
</tbody>
</table>

Note 1: Enabling Brown-out Reset automatically enables Power-up Timer (PWRT), regardless of the value of bit PWRTE. Ensure the Power-up Timer is enabled any time Brown-out Reset is enabled.

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
4.0 CODE PROTECTION

Once code protection is enabled, all program memory locations read all ‘0’s; further programming of program memory is disabled. ID locations and the configuration word may still be read and programmed (1’s to 0’s only).

4.1 Disabling Code Protection

The following procedure should be performed before any other programming is attempted. This procedure also turns off code protection (code protect bit = 1); however, all program memory will be erased when this procedure is executed and thus, the security of the code is not compromised.

Procedure to disable code protection:

a) Issue the ‘Chip Erase’ command.

b) Wait for the erase cycle time (tera) to pass. The program memory is erased, then the configuration memory is erased.

4.2 Embedding Configuration Word and ID Information in the HEX File

To allow portability of code, the programmer is required to read the configuration word and ID locations from the HEX file, when loading the HEX file. If configuration word information was not present in the HEX file, then a simple warning message may be issued. Similarly, while saving a HEX file, configuration word and ID information must be included. An option to not include this information may be provided.

Microchip Technology Inc. feels strongly that this feature is important for the benefit of the end customer.
4.3 Checksum Computation

The checksum is calculated by reading the contents of the PIC16F7X memory locations and adding up the opcodes, up to the maximum user addressable location (e.g., 0x1FFF for the PIC16F7X). Any carry bits exceeding 16 bits are neglected. Finally, the configuration word (appropriately masked) is added to the checksum. Checksum computation for each member of the PIC16F7X devices is shown in Table 4-1.

The checksum is calculated by summing the following:

- The contents of all program memory locations
- The configuration word, appropriately masked
- Masked ID locations (when applicable)

The Least Significant 16 bits of this sum are the checksum.

Table 4-1 describes how to calculate the checksum for each device. Note that the checksum calculation differs depending on the code protection setting. Since the program memory locations read out differently depending on the code protection setting, the table describes how to manipulate the actual program memory values to simulate the values that would be read from a protected device. When calculating a checksum of a non-protected device, the entire program memory can simply be read and summed. The configuration word and ID locations can always be read.

### TABLE 4-1: CHECKSUM COMPUTATION

<table>
<thead>
<tr>
<th>Device</th>
<th>Code Protect</th>
<th>Checksum</th>
<th>Blank Value</th>
<th>0x05E6 at 0x0000 and max address</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIC16F73</td>
<td>OFF</td>
<td>SUM[0x000:0x0FFF] + CFWD &amp; 0x005F</td>
<td>0xF05F</td>
<td>0x07C2D</td>
</tr>
<tr>
<td></td>
<td>ALL</td>
<td>CFWD &amp; 0x005F + SUM_ID</td>
<td>0x005E</td>
<td>0x005E</td>
</tr>
<tr>
<td>PIC16F74</td>
<td>OFF</td>
<td>SUM[0x000:0x0FFF] + CFWD &amp; 0x005F</td>
<td>0xF05F</td>
<td>0x07C2D</td>
</tr>
<tr>
<td></td>
<td>ALL</td>
<td>CFWD &amp; 0x005F + SUM_ID</td>
<td>0x005E</td>
<td>0x005E</td>
</tr>
<tr>
<td>PIC16F76</td>
<td>OFF</td>
<td>SUM[0x000:0x1FFF] + CFWD &amp; 0x005F</td>
<td>0xE05F</td>
<td>0x08C2D</td>
</tr>
<tr>
<td></td>
<td>ALL</td>
<td>CFWD &amp; 0x005F + SUM_ID</td>
<td>0x005E</td>
<td>0x005E</td>
</tr>
<tr>
<td>PIC16F77</td>
<td>OFF</td>
<td>SUM[0x000:0x1FFF] + CFWD &amp; 0x005F</td>
<td>0xE05F</td>
<td>0x08C2D</td>
</tr>
<tr>
<td></td>
<td>ALL</td>
<td>CFWD &amp; 0x005F + SUM_ID</td>
<td>0x005E</td>
<td>0x005E</td>
</tr>
</tbody>
</table>

Legend: CFWD = Configuration Word

SUM[a:b] = [Sum of locations a to b inclusive]

SUM_ID = ID locations masked by 0x0F, then concatenated into a 16-bit value with ID0 as the most significant nibble.

For example, ID0 = 0x01, ID2 = 0x02, ID3 = 0x03, ID4 = 0x04, then SUM_ID = 0x1234

Checksum = [Sum of all the individual expressions] **MODULO** [0xFFFF]

+ = Addition

& = Bitwise AND
5.0 PROGRAM/VERIFY MODE ELECTRICAL CHARACTERISTICS

5.1 AC/DC Characteristics

### TABLE 5-1: TIMING REQUIREMENTS FOR PROGRAM/VERIFY MODE

<table>
<thead>
<tr>
<th>Characteristics</th>
<th>Sym</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Units</th>
<th>Conditions/Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>General</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VDD level for read and verification</td>
<td>VDD</td>
<td>2.0</td>
<td>5.5</td>
<td>V</td>
<td>VDD</td>
<td></td>
</tr>
<tr>
<td>VDD level for programming and erasing</td>
<td>VDDP</td>
<td>4.75</td>
<td>5.25</td>
<td>V</td>
<td>VDDP</td>
<td></td>
</tr>
<tr>
<td>High voltage on MCLR for chip erase and program write operations</td>
<td>VPP</td>
<td>12.75</td>
<td>13.25</td>
<td>V</td>
<td>VPP</td>
<td></td>
</tr>
<tr>
<td>MCLR rise time (VSS to VPP) for Test mode entry</td>
<td>tVHHR</td>
<td>1.0</td>
<td></td>
<td>µs</td>
<td>tVHHR</td>
<td></td>
</tr>
<tr>
<td>(RB6, RB7) input high level</td>
<td>ViH1</td>
<td>0.8 VDD</td>
<td></td>
<td>V</td>
<td>Schmitt Trigger input</td>
<td></td>
</tr>
<tr>
<td>(RB6, RB7) input low level</td>
<td>Vil1</td>
<td>0.2 VDD</td>
<td></td>
<td>V</td>
<td>Schmitt Trigger input</td>
<td></td>
</tr>
<tr>
<td><strong>Serial Program/Verify</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data in setup time before clock↓</td>
<td>tsetup</td>
<td>100</td>
<td></td>
<td>ns</td>
<td>tsetup</td>
<td></td>
</tr>
<tr>
<td>Data in hold time after clock↓</td>
<td>thold</td>
<td>100</td>
<td></td>
<td>ns</td>
<td>thold</td>
<td></td>
</tr>
<tr>
<td>Data input not driven to next clock input (delay required between command/data or command/command)</td>
<td>tdy1</td>
<td>1.0</td>
<td></td>
<td>µs</td>
<td>tdy1</td>
<td></td>
</tr>
<tr>
<td>Delay between clock↓ to clock↑ of next command or data</td>
<td>tdy2</td>
<td>1.0</td>
<td></td>
<td>µs</td>
<td>tdy2</td>
<td></td>
</tr>
<tr>
<td>Clock↑ to data out valid (during read data)</td>
<td>tdy3</td>
<td>200</td>
<td></td>
<td>ns</td>
<td>tdy3</td>
<td></td>
</tr>
<tr>
<td>Erase cycle time</td>
<td>tera</td>
<td>30</td>
<td></td>
<td>ms</td>
<td>tera</td>
<td>(Note 3)</td>
</tr>
<tr>
<td>Programming cycle time</td>
<td>tprog</td>
<td>1</td>
<td></td>
<td>1 ms</td>
<td>tprog</td>
<td></td>
</tr>
</tbody>
</table>

**Note 1:** VPP should be current limited to about 100 mA.

**Note 2:** VPP must remain above VDDP + 4.0V to remain in Programming mode, while not actually erasing or programming.

**Note 3:** The chip erase is self-timed.
FIGURE 5-1: LOAD DATA COMMAND MODE (PROGRAM/VERIFY)

FIGURE 5-2: READ DATA COMMAND MODE (PROGRAM/VERIFY)

FIGURE 5-3: INCREMENT ADDRESS COMMAND MODE (PROGRAM/VERIFY)
Note the following details of the code protection feature on PICmicro® MCUs.

- The PICmicro family meets the specifications contained in the Microchip Data Sheet.
- Microchip believes that its family of PICmicro microcontrollers is one of the most secure products 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 PICmicro microcontroller in a manner outside the operating specifications contained in the data sheet. The person doing so may be 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 product.

If you have any further questions about this matter, please contact the local sales office nearest to you.

Information contained in this publication regarding device applications and the like is intended through suggestion only and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. No representation or warranty is given and no liability is assumed by Microchip Technology Incorporated with respect to the accuracy or use of such information, or infringement of patents or other intellectual property rights arising from such use or otherwise. Use of Microchip’s products as critical components in life support systems is not authorized except with express written approval by Microchip. No licenses are conveyed, implicitly or otherwise, under any intellectual property rights.

Trademark

The Microchip name and logo, the Microchip logo, FilterLab, KEELoQ, MPLAB, PIC, PICmicro, PICMASTER, PICSTART, PRO MATE, SEEVAL and The Embedded Control Solutions Company are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.

dsPIC, ECONOMONITOR, FanSense, FlexROM, fuzzyLAB, In-Circuit Serial Programming, ICSP, ICEPIC, microID, microPort, Migratable Memory, MPASM, MPLIB, MPLINK, MPSIM, MXDEV, PICC, PICDEM, PICDEM.net, rIPIC, Select Mode and Total Endurance are trademarks of Microchip Technology Incorporated in the U.S.A.

Serialized Quick Term Programming (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.

© 2002, Microchip Technology Incorporated, Printed in the U.S.A., All Rights Reserved.

Printed on recycled paper.