This document includes the programming specifications for the following devices:

- PIC12F629
- PIC16F630
- PIC12F675
- PIC16F676

1.0 PROGRAMMING THE PIC12F629/675/PIC16F630/676

The PIC12F629/675/PIC16F630/676 is programmed using a serial method. The Serial mode will allow the PIC12F629/675/PIC16F630/676 to be programmed while in the user’s system. This allows for increased design flexibility. This programming specification applies to PIC12F629/675/PIC16F630/676 devices in all packages.

FIGURE 1-1: 8-PIN DIAGRAMS FOR PIC12F629/675

1.1 Hardware Requirements

The PIC12F629/675/PIC16F630/676 requires one power supply for V DD (5.0 V) and one for V PP (12 V).

1.2 Programming Mode

The Programming mode for the PIC12F629/675/PIC16F630/676 allows programming of user program memory, data memory, special locations used for ID and the Configuration Word register.
**FIGURE 1-2:** 14-PIN DIAGRAMS FOR PIC16F630/676

**PDIP, SOIC, TSSOP**

<table>
<thead>
<tr>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
</tr>
</thead>
<tbody>
<tr>
<td>VDD</td>
<td>RA5/T1CKI/OSC1/CLKIN</td>
<td>RA4/T1G/OSC2/CLKOUT</td>
<td>RA3/MCLR/VPP</td>
<td>RC5</td>
<td>RC4</td>
<td>RC3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**PIC16F630**

<table>
<thead>
<tr>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>VSS</td>
<td>RA0/CIN+/ICSPDAT</td>
<td>RA1/CIN-/ICSPCLK</td>
<td>RA2/COUT/T0CKI/INT</td>
<td>RC0</td>
<td>RC1</td>
<td>RC2</td>
<td>RC3</td>
<td>RC4</td>
<td>RC5</td>
<td>RA3/MCLR/VPP</td>
<td>RA4/T1G/OSC2/CLKOUT</td>
<td>RA5/T1CKI/OSC1/CLKIN</td>
<td>VDD</td>
</tr>
</tbody>
</table>

**QFN**

<table>
<thead>
<tr>
<th>16</th>
<th>15</th>
<th>14</th>
<th>13</th>
<th>12</th>
<th>11</th>
<th>10</th>
<th>9</th>
<th>8</th>
<th>7</th>
<th>6</th>
<th>5</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>VDD</td>
<td>RC1/AN5</td>
<td>RC2/AN6</td>
<td>RC3/AN7</td>
<td>RC4</td>
<td>RC5</td>
<td>RA3/MCLR/VPP</td>
<td>RA4/T1G/OSC2/CLKOUT</td>
<td>RA5/T1CKI/OSC1/CLKIN</td>
<td>RA0/C1IN+/ICSPDAT</td>
<td>RA1/CIN-/VREF/ICSPCLK</td>
<td>RA2/COUT/T0CKI/INT</td>
<td>RA0/AN0/CIN+/ICSPDAT</td>
<td>RA1/AN1/CIN-/VREF/ICSPCLK</td>
<td>RA2/AN2/COUT/T0CKI/INT</td>
<td>VSS</td>
</tr>
</tbody>
</table>

© 2005 Microchip Technology Inc.
FIGURE 1-3: 20-PIN DIAGRAM FOR rfPIC12F675F/H/K

TABLE 1-1: PIN DESCRIPTIONS (DURING PROGRAMMING): PIC12F629/675/PIC16F630/676

<table>
<thead>
<tr>
<th>Pin Name</th>
<th>During Programming</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function</td>
<td>Pin Type</td>
</tr>
<tr>
<td>GP1</td>
<td>CLOCK</td>
</tr>
<tr>
<td>GP0</td>
<td>DATA</td>
</tr>
<tr>
<td>MCLR</td>
<td>Programming Mode</td>
</tr>
<tr>
<td>RA1</td>
<td>CLOCK</td>
</tr>
<tr>
<td>RA0</td>
<td>DATA</td>
</tr>
<tr>
<td>VDD</td>
<td>VDD</td>
</tr>
<tr>
<td>VSS</td>
<td>VSS</td>
</tr>
</tbody>
</table>

Legend:  I = Input, O = Output, P = Power

Note 1: In the PIC12F629/675/PIC16F630/676, the programming high voltage is internally generated. To activate the Programming mode, high voltage needs to be applied to the MCLR input. Since the MCLR is used for a level source, the MCLR does not draw any significant current.
2.0 PROGRAM MODE ENTRY

2.1 User Program Memory Map

The user memory space extends from 0x0000-0x1FFF. In Programming mode, the program memory space extends from 0x0000-0x3FFF, the first half (0x0000-0x1FFF) is user program memory and the second half (0x2000-0x3FFF) is configuration memory. The PC will increment from 0x0000-0x1FFF and wrap to 0x000, 0x2000-0x3FFF and wrap around to 0x2000 (not to 0x0000). Once in configuration memory, the highest bit of the PC remains a ‘1’, thus always pointing to the configuration memory. The only way to point to the user program memory is to reset the part and re-enter Program/Verify mode as described in Section 2.3 “Program/Verify Mode”.

In the configuration memory space, 0x2000-0x201F are physically implemented. However, only locations 0x2000-0x2003 and 0x2007 are available. Other locations are reserved.

2.2 ID Locations

A user may store identification information (ID) in four ID locations. The ID locations are mapped in [0x2000:0x2003]. It is recommended that the user use only the seven Least Significant bits (LSb) of each ID location. Locations read out normally, even after code protection. The ID locations read out in an unscrambled fashion after code protection is enabled. It is recommended that ID location is written as “xx xxxx xbbb bbbb” where ‘bbb bbbb’ is ID information.

The 14 bits may be programmed, but only the LSbs are displayed by MPLAB® IDE. xxxx’s are “don’t care” bits as they won’t be read by MPLAB® IDE.

FIGURE 2-1: PROGRAM MEMORY MAPPING
2.3 Program/Verify Mode

The Program/Verify mode is entered by holding pins clock and data low while raising MCLR pin from VIL to VIHH (high voltage). Apply VDD and data. Once in this mode, the user program memory, data memory and the configuration memory can be accessed and programmed in serial fashion. Clock is Schmitt Trigger and data is TTL input in this mode. GP4 (PIC12F629/675) or RA4 (PIC16F630/676) is tri-state, regardless of use setting.

The sequence that enters the device into the Programming/Verify mode places all other logic into the Reset state (the MCLR pin was initially at VIL). This means that all I/O’s are in the Reset state (high-impedance inputs).

A device Reset will clear the PC and set the address to ‘0’. 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 2-1.

2.3.1 SERIAL PROGRAM/VERIFY OPERATION

The clock pin is used as a clock input pin and the data pin is used for entering command bits and data input/output during serial operation. To input a command, the clock pin (CLOCK) is cycled six times. Each command bit is latched on the falling edge of the clock with the LSb of the command being input first. The data on pin DATA is required to have a minimum setup and hold time (see Table 5-1), with respect to the falling edge of the clock. Commands that have data associated with them (Read and Load) are specified to have a minimum delay of 1 μs between the command and the data. After this delay, the clock pin is cycled 16 times with the first cycle being a Start bit and the last cycle being a Stop bit. Data is also input and output LSb first.

Therefore, during a read operation, the LSB will be transmitted onto pin DATA on the rising edge of the second cycle. During a load operation, the LSb will be latched on the falling edge of the second cycle. A minimum 1 μs delay is also specified between consecutive commands.

All commands are transmitted LSb first. Data words are also transmitted LSb first. The data is transmitted on the rising edge and latched on the falling edge of the clock. To allow for decoding of commands and reversal of data pin configuration, a time separation of at least 1 μs is required between a command and a data word (or another command).

The commands that are available are described in Table 2-1.

<table>
<thead>
<tr>
<th>Command</th>
<th>Mapping (MSb ... LSb)</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load Configuration</td>
<td>X X 0 0 0 0</td>
<td>0, data (14), 0</td>
</tr>
<tr>
<td>Load Data for Program Memory</td>
<td>X X 0 0 1 0</td>
<td>0, data (14), 0</td>
</tr>
<tr>
<td>Load Data for Data Memory</td>
<td>X X 0 0 1 1</td>
<td>0, data (8), zero (6), 0</td>
</tr>
<tr>
<td>Read Data from Program Memory</td>
<td>X X 0 1 0 0</td>
<td>0, data (14), 0</td>
</tr>
<tr>
<td>Read Data from Data Memory</td>
<td>X X 0 1 0 1</td>
<td>0, data (8), zero (6), 0</td>
</tr>
<tr>
<td>Increment Address</td>
<td>X X 0 1 1 0</td>
<td></td>
</tr>
<tr>
<td>Begin Programming</td>
<td>0 0 1 0 0 0</td>
<td>Internally Timed</td>
</tr>
<tr>
<td>Begin Programming</td>
<td>0 1 1 0 0 0</td>
<td>Externally Timed</td>
</tr>
<tr>
<td>End Programming</td>
<td>0 0 1 0 0 0</td>
<td></td>
</tr>
<tr>
<td>Bulk Erase Program Memory</td>
<td>X X 1 0 0 1</td>
<td>Internally Timed</td>
</tr>
<tr>
<td>Bulk Erase Data Memory</td>
<td>X X 1 0 1 1</td>
<td>Internally Timed</td>
</tr>
</tbody>
</table>
2.3.1.1 Load Configuration

After receiving this command, the Program Counter (PC) will be set to 0x2000. Then, by applying 16 cycles to the clock pin, the chip will load 14 bits in a data word, as described above, which will be programmed into the configuration memory. A description of the memory mapping schemes of the program memory for normal operation and Configuration mode operation is shown in Figure 2-3. After the configuration memory is entered, the only way to get back to the user program memory is to exit the Program/Verify mode by taking MCLR low (VIL).

**FIGURE 2-3: LOAD CONFIGURATION COMMAND**

![Diagram of Load Configuration Command](image1)

**Note 1:** GP0 and GP1 apply to PIC12F629/675 only. For PIC16F630/676, use RA0 and RA1, respectively.

2.3.1.2 Load Data For Program Memory

After receiving this command, the chip will load in a 14-bit data word when 16 cycles are applied, as described previously. A timing diagram for the Load Data command is shown in Figure 2-4.

**FIGURE 2-4: LOAD DATA FOR PROGRAM MEMORY COMMAND**

![Diagram of Load Data for Program Memory Command](image2)

**Note 1:** GP0 and GP1 apply to PIC12F629/675 only. For PIC16F630/676, use RA0 and RA1, respectively.
2.3.1.3 Load Data For Data Memory

After receiving this command, the chip will load in a 14-bit data word when 16 cycles are applied. However, the data memory is only 8 bits wide and thus, only the first 8 bits of data after the Start bit will be programmed into the data memory. It is still necessary to cycle the clock the full 16 cycles in order to allow the internal circuitry to reset properly. The data memory contains 128 bytes. Only the lower 8 bits of the PC are decoded by the data memory and therefore, if the PC is greater than 0x7F, it will wrap around and address a location within the physically implemented memory.

FIGURE 2-5: LOAD DATA FOR DATA MEMORY COMMAND

2.3.1.4 Read Data From Program Memory

After receiving this command, the chip will transmit data bits out of the program memory (user or configuration) currently accessed, starting with the second rising edge of the clock input. The data pin will go into Output mode on the second rising clock edge and revert to Input mode (high-impedance) after the 16th rising edge. If the program memory is code-protected (CP = 0), the data is read as zeros.

FIGURE 2-6: READ DATA FROM PROGRAM MEMORY COMMAND

Note 1: GP0 and GP1 apply to PIC12F629/675 only. For PIC16F630/676, use RA0 and RA1, respectively.
2.3.1.5 Read Data From Data Memory

After receiving this command, the chip will transmit data bits out of the data memory starting with the second rising edge of the clock input. The data pin will go into Output mode on the second rising edge and revert to Input mode (high-impedance) after the 16th rising edge. As previously stated, the data memory is 8 bits wide and therefore, only the first 8 bits that are output are actual data. If the data memory is code-protected, the data is read as all zeros. A timing diagram of this command is shown in Figure 2-7.

**FIGURE 2-7: READ DATA FROM DATA MEMORY COMMAND**

![Timing Diagram](image)

*Note 1: GP0 and GP1 apply to PIC12F629/675 only. For PIC16F630/676, use RA0 and RA1, respectively.*

2.3.1.6 Increment Address

The PC is incremented when this command is received. A timing diagram of this command is shown in Figure 2-8.

It is not possible to decrement the address counter. To reset this counter, the user should exit and re-enter Programming mode.

**FIGURE 2-8: INCREMENT ADDRESS COMMAND (PROGRAM/VERIFY)**

![Timing Diagram](image)

*Note 1: GP0 and GP1 apply to PIC12F629/675 only. For PIC16F630/676, use RA0 and RA1, respectively.*
2.3.1.7 Begin Programming (Internally Timed)

A Load command must be given before every Begin Programming command. Programming of the appropriate memory (user program memory or data memory) will begin after this command is received and decoded. An internal timing mechanism executes a write. The user must allow for program cycle time for programming to complete. No End Programming command is required.

When programming data memory, the byte being addressed is erased before being programmed.

FIGURE 2-9: BEGIN PROGRAMMING COMMAND (INTERNALLY TIMED)

Note 1: GP0 and GP1 apply to PIC12F629/675 only. For PIC16F630/676, use RA0 and RA1, respectively.
2.3.1.8 Begin Programming (Externally Timed)

A Load command must be given before every Begin Programming command. Programming of the appropriate memory (user program memory or data memory) will begin after this command is received and decoded. Programming requires (TPROG2) time and is terminated using an End Programming command (see Figure 2-11). This command programs the current location, no erase is performed.

**FIGURE 2-10: BEGIN PROGRAMMING (EXTERNALLY TIMED)**

**FIGURE 2-11: END PROGRAMMING (SERIAL PROGRAM/VERIFY)**
2.3.1.9 Bulk Erase Program Memory

After this command is performed and Calibration bits are erased, the entire program memory is erased. If data is code-protected, data memory will also be erased.

**Note 1:** The OSCCAL word and BG bits must be read prior to erasing the device and restored during the programming operation. OSCCAL is at location 0x3FF and the BG bits are bits 12 and 13 of the Configuration Word (0x2007).

**Note 2:** The OSCCAL location must contain the `RETLW` instruction within its data in order to be verified properly. The data in the OSCCAL location should be `11 01xx xxxx xxxx`, where the x's are "don't care" bits and are ignored by the programmer.

To perform a bulk erase of the program memory, the following sequence must be performed.
1. Read OSCCAL 0x3FF.
2. Verify `RETLW` instruction for OSCCAL location.
3. Read Configuration Word.
4. Do a Bulk Erase Program Memory command.
5. Wait TERA to complete bulk erase.

If the address is pointing to the ID/configuration program memory (0x2000-0x201F), then both the user memory and the ID locations will be erased.

**FIGURE 2-12: BULK ERASE PROGRAM MEMORY COMMAND**

Note 1: GP0 and GP1 apply to PIC12F629/675 only. For PIC16F630/676, use RA0 and RA1, respectively.
2.3.1.10 Bulk Erase Data Memory

To perform a bulk erase of the data memory, the following sequence must be performed.

1. Do a Bulk Erase Data Memory command.
2. Wait TER to complete bulk erase.

Data memory won’t erase if code-protected (CPD = 0).

Note: All bulk erase operations must take place at 4.5V to 5.5V VDD range for PIC12F629/675/PIC16F630/676 devices and 2.0V to 5.5V VDD for PIC16F630-ICD device.

FIGURE 2-13: BULK ERASE DATA MEMORY COMMAND

Note 1: GP0 and GP1 apply to PIC12F629/675 only. For PIC16F630/676, use RA0 and RA1, respectively.
FIGURE 2-14: PROGRAM FLOWCHART – PIC12F629/675/PIC16F630/676 PROGRAM MEMORY

Start

Read and Save OSCCAL value

RETlw Instruction Correct?

Yes

No

Read and Save Band Gap Cal. Value

Bulk Erase Device

Program Cycle

Program cycle

Read Data from Program Memory

Data Correct?

Yes

No

All Locations Done?

Yes

No

Program OSCCAL

Program Data Memory (if required)

Verify all Locations

Data Correct?

Yes

No

Report Verify Error

Program Band Gap Cal. and Config. bits

Done

Program Cycle

Load Data for Program Memory

Begin Programming Command (Internally timed)

Wait TPROG1

Begin Programming Command (Externally timed)

Wait TPROG2

End Programming
FIGURE 2-15: PROGRAM FLOWCHART – PIC12F629/675/PIC16F630/676 CONFIGURATION MEMORY

Start

Load Configuration Data

Program Cycle

Read Data Command

Data Correct?

No

Report Programming Failure

Yes

Increment Address Command

Address = 0x2004?

No

Increment Address Command

Yes

Increment Address Command

Increment Address Command

Set Bits 12 and 13 to Saved Band Gap Bits

Program Cycle (Config. Word)

Read Data Command

Data Correct?

No

Report Programming Failure

Yes

Done
FIGURE 2-16: PROGRAM FLOWCHART – PIC12F629/675/PIC16F630/676 DATA MEMORY

Start

Program Cycle

Read Data from Data Memory

Data Correct?

Yes

Report Programming Failure

No

Increment Address Command

All Locations Done?

Yes

Done

No

Program Cycle

Load Data for Program Memory

Begin Programming Command (Internally timed)

Wait TPROG1

Begin Programming Command (Externally timed)

Wait TPROG2

End Programming
FIGURE 2-17: PROGRAM FLOWCHART – PIC12F629/675/PIC16F630/676 ERASE FLASH MEMORY

```
Start

Read and Save OSCCAL Value

RETLW Instruction Correct? No

Report OSCCAL Instruction Error

Yes

Read and Save Band Gap Cal. Value

Bulk Erase Device

Program OSCCAL

Program Band Gap Cal. Bits

Done
```
### 3.0 CONFIGURATION WORD

The PIC12F629/675/PIC16F630/676 has several Configuration bits. These bits can be programmed (reads ‘0’) or left unchanged (reads ‘1’) to select various device configurations.

#### REGISTER 3-1: CONFIGURATION WORD FOR PIC12F629/675/PIC16F630/676

<table>
<thead>
<tr>
<th>Bit</th>
<th>Description</th>
<th>Value</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>bit 13-12</td>
<td><strong>BG&lt;1:0&gt;</strong></td>
<td>Band Gap Calibration bits(2)</td>
<td>00 = Lowest band gap voltage, 11 = Highest band gap voltage</td>
</tr>
<tr>
<td>bit 11-9</td>
<td><strong>Unimplemented</strong></td>
<td>Read as ‘0’</td>
<td></td>
</tr>
<tr>
<td>bit 8</td>
<td><strong>CPD</strong></td>
<td>Code Protection Data bit</td>
<td>1 = Data memory is not protected, 0 = Data memory is external read protected</td>
</tr>
<tr>
<td>bit 7</td>
<td><strong>CP</strong></td>
<td>Code Protection bit</td>
<td>1 = Program memory is not code-protected, 0 = Program memory is code-protected</td>
</tr>
<tr>
<td>bit 6</td>
<td><strong>BODEN</strong></td>
<td>Brown-out Detect Enable bit(1)</td>
<td>1 = BOD enabled, 0 = BOD disabled</td>
</tr>
<tr>
<td>bit 5</td>
<td><strong>MCLRE</strong></td>
<td>MCLR Pin Function Select bit</td>
<td>1 = MCLR pin is MCLR function, 0 = MCLR pin is alternate function, MCLR function is internally disabled</td>
</tr>
<tr>
<td>bit 4</td>
<td><strong>PWRTE</strong></td>
<td>Power-up Timer Enable bit(1)</td>
<td>1 = PWRT disabled, 0 = PWRT enabled</td>
</tr>
<tr>
<td>bit 3</td>
<td><strong>WDTE</strong></td>
<td>Watchdog Timer Enable bit</td>
<td>1 = WDT enabled, 0 = WDT disabled</td>
</tr>
<tr>
<td>bit 2-0</td>
<td><strong>FOSC&lt;2:0&gt;</strong></td>
<td>Oscillator Selection bits(3)</td>
<td>000 = LP oscillator, 001 = XT oscillator, 010 = HS oscillator, 011 = EC oscillator, 100 = INTOSC oscillator, 101 = INTOSC oscillator, 110 = RC oscillator, 111 = RC oscillator</td>
</tr>
</tbody>
</table>

**Notes:**
1. Enabling Brown-out Detect Reset Enable does not automatically enable the Power-up Timer Enable (PWRTE).
2. The Band Gap Calibration bits must be read and preserved, then replaced by the user during any bulk erase operation.
3. GP4 and GP5 apply to PIC12F629/675 only. For PIC16F630/676, use RA4 and RA5, respectively.

**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
### 3.1 Device ID Word

The device ID word for each device is located at 2006h.

#### TABLE 3-1: DEVICE ID VALUES

<table>
<thead>
<tr>
<th>Device</th>
<th>Device ID Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIC12F629</td>
<td>00 1111 100</td>
</tr>
<tr>
<td>PIC12F675</td>
<td>00 1111 110</td>
</tr>
<tr>
<td>PIC16F630</td>
<td>01 0000 110</td>
</tr>
<tr>
<td>PIC16F676</td>
<td>01 0000 111</td>
</tr>
</tbody>
</table>
4.0 CODE PROTECTION

For PIC12F629/675/PIC16F630/676 devices, once code protection is enabled, all program memory locations, except 0X3FF, reads all ‘0’s. The ID locations and the Configuration Word read out in an unprotected fashion. Further programming is disabled for the entire program memory. Data memory is protected with its own Code Protection Data bit (CPD). It is possible to program the ID locations and the Configuration Word.

4.1 Disabling Code Protection

It is recommended that the following procedure be performed before any other programming is attempted. It is also possible to turn code protection off (CPD = 1) using this procedure. However, all data within the program memory and the data memory will be erased when this procedure is executed and thus, the security of the data or code is not compromised.

<table>
<thead>
<tr>
<th>To disable code-protect:</th>
</tr>
</thead>
<tbody>
<tr>
<td>a) Read and store OSCCAL and BG bits.</td>
</tr>
<tr>
<td>b) Execute Load Configuration (000000).</td>
</tr>
<tr>
<td>c) Execute Bulk Erase Program Memory (001001).</td>
</tr>
<tr>
<td>d) Wait TERA.</td>
</tr>
<tr>
<td>e) Execute Bulk Erase Data Memory (001011).</td>
</tr>
<tr>
<td>f) Wait TERA.</td>
</tr>
<tr>
<td>g) Reset device to reset address counter before reprogramming device.</td>
</tr>
<tr>
<td>h) Restore OSCCAL and BG bits.</td>
</tr>
</tbody>
</table>

| Note: To ensure system security, if CPD bit = 0, step c) will also erase data memory. |

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.

Specifically for the PIC12F629/675/PIC16F630/676, the EEPROM data memory should also be embedded in the hex file (see Section 4.3.2 “Embedding Data EEPROM Contents In Hex File”). Microchip Technology Incorporated feels strongly that this feature is important for the benefit of the end customer.
4.3 Checksum Computation

4.3.1 CHECKSUM

Checksum is calculated by reading the contents of the PIC12F629/675/PIC16F630/676 memory locations and adding up the opcodes to the maximum user addressable location (e.g., 0x3FE for the PIC12F629/675/PIC16F630/676). Any carry bits exceeding 16 bits are neglected. Finally, the Configuration Word (appropriately masked) is added to the checksum. Checksum computation for the 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 following table describes how to calculate the checksum for each device.

**TABLE 4-1: CHECKSUM COMPUTATION**

<table>
<thead>
<tr>
<th>Device</th>
<th>Code-Protect</th>
<th>Checksum*</th>
<th>Blank Value</th>
<th>0x25E6 at 0 and Max. Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>PIC12F629/675</td>
<td>OFF</td>
<td>SUM[0x0000:0x3FE] + CFGW &amp; 01FF</td>
<td>BE00</td>
<td>89CE</td>
</tr>
<tr>
<td>PIC16F630/676</td>
<td>ALL</td>
<td>CFGW &amp; 0x01FF + SUM_ID</td>
<td>BF7F</td>
<td>8B4D</td>
</tr>
</tbody>
</table>

Legend:
- CFGW = Configuration Word
- SUM[a:b] = [Sum of locations a to b inclusive]
- SUM_ID = ID locations masked by 0xF then made into a 16-bit value with ID0 as the Most Significant nibble.
- For example: ID0 = 0x1, ID1 = 0x2, ID3 = 0x3, ID4 = 0x4, then SUM_ID = 0x1234
- *Checksum = [Sum of all the individual expressions] MODULO [0xFFFF]

4.3.2 EMBEDDING DATA EEPROM CONTENTS IN HEX FILE

The programmer should be able to read data EEPROM information from a hex file and conversely (as an option), write data EEPROM contents to a hex file, along with program memory information and fuse information.

The 128 data memory locations are logically mapped starting at address 0x2100. The format for data memory storage is one data byte per address location, LSb aligned.

---

**Note 1:** The checksum calculation differs depending on the code-protect setting. Since the program memory locations read out differently depending on the code-protect setting, Table 4-1 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 by reading a device, the entire program memory can simply be read and summed. The Configuration Word and ID locations can always be read.

**2:** Some older devices have an additional value added in the checksum. This is to maintain compatibility with older device programmer checksums.
## 5.0 PROGRAM/VERIFY MODE ELECTRICAL CHARACTERISTICS

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

<table>
<thead>
<tr>
<th>AC/DC CHARACTERISTICS</th>
<th>Standard Operating Conditions (unless otherwise stated)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Operating Temperature: $-40^\circ\text{C} \leq T_A \leq +85^\circ\text{C}$</td>
</tr>
<tr>
<td></td>
<td>Operating Voltage: $4.5\text{V} \leq V_{DD} \leq 5.5\text{V}$</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Sym.</strong></td>
<td><strong>Characteristics</strong></td>
</tr>
<tr>
<td>General</td>
<td></td>
</tr>
<tr>
<td>VDD</td>
<td>VDD level for word operations, program memory</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>VDD</td>
<td>VDD level for word operations, data memory</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>VDD</td>
<td>VDD level for bulk erase/write operations, program and data memory</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>VIHH</td>
<td>High voltage on MCLR for Programming mode entry</td>
</tr>
<tr>
<td>TVHHR</td>
<td>MCLR rise time (VSS to VHH) for Programming mode entry</td>
</tr>
<tr>
<td>TPPDP</td>
<td>Hold time after VPP ↑</td>
</tr>
<tr>
<td>Vih1</td>
<td>(CLOCK, DATA) input high level</td>
</tr>
<tr>
<td>Vil1</td>
<td>(CLOCK, DATA) input low level</td>
</tr>
<tr>
<td>TSET0</td>
<td>CLOCK, DATA setup time before MCLR ↑ (Programming mode selection pattern setup time)</td>
</tr>
<tr>
<td>THLD0</td>
<td>CLOCK, DATA hold time after MCLR ↑ (Programming mode selection pattern setup time)</td>
</tr>
<tr>
<td>Serial Program/Verify</td>
<td></td>
</tr>
<tr>
<td>TSET1</td>
<td>Data in setup time before clock ↓</td>
</tr>
<tr>
<td>THLD1</td>
<td>Data in hold time after clock ↓</td>
</tr>
<tr>
<td>TDLY1</td>
<td>Data input not driven to next clock input (delay required between command/data or command/command)</td>
</tr>
<tr>
<td>TDLY2</td>
<td>Delay between clock ↓ to clock ↑ of next command or data</td>
</tr>
<tr>
<td>TDLY3</td>
<td>Clock ↑ to data out valid (during read data)</td>
</tr>
<tr>
<td>TERA</td>
<td>Erase cycle time</td>
</tr>
<tr>
<td>TPROG1</td>
<td>Programming cycle time (internally timed)</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>TPROG2</td>
<td>Programming cycle time (externally timed)</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>TDIS</td>
<td>Time delay from program to compare (HV discharge time)</td>
</tr>
</tbody>
</table>
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’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 Microchip intellectual property rights.

Trademarks
The Microchip name and logo, the Microchip logo, Accuron, dsPIC, KEELOQ, microID, MPLAB, PIC, PICmicro, PICSTART, PRO MATE, PowerSmart, rPIC, and SmartShunt are registered trademarks of Microchip Technology Incorporated in the U.S.A. and other countries.
AmpLab, FilterLab, Migratable Memory, MXDEV, MXLAB, PICMASTER, 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, dsPICDEM, dsPICDEM.net, dsPicworks, ECAN, ECONOMONITOR, FanSense, FlexROM, fuzzyLAB, In-Circuit Serial Programming, ICSP, ICEPIC, Linear Active Thermistor, MPASM, MPLIB, MPLINK, MPSIM, PICkit, PICDEM, PICDEM.net, PICLAB, PICtail, PowerCal, PowerInfo, PowerMate, PowerTool, rFLAB, rPICDEM, Select Mode, Smart Serial, SmartTel, Total Endurance and WiperLock 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.
© 2005, 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 quality system certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona and Mountain View, California in October 2003. 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.