

## Programming Specifications for PIC16C6XX/7XX/9XX OTP MCUs

This document includes the programming specifications for the following devices:

- |             |              |              |
|-------------|--------------|--------------|
| • PIC16C61  | • PIC16C72A  | • PIC16CE623 |
| • PIC16C62  | • PIC16C73   | • PIC16CE624 |
| • PIC16C62A | • PIC16C73A  | • PIC16CE625 |
| • PIC16C62B | • PIC16C73B  | • PIC16C710  |
| • PIC16C63  | • PIC16C74   | • PIC16C711  |
| • PIC16C63A | • PIC16C74A  | • PIC16C712  |
| • PIC16C64  | • PIC16C74B  | • PIC16C716  |
| • PIC16C64A | • PIC16C76   | • PIC16C745  |
| • PIC16C65  | • PIC16C77   | • PIC16C765  |
| • PIC16C65A | • PIC16C620  | • PIC16C773  |
| • PIC16C65B | • PIC16C620A | • PIC16C774  |
| • PIC16C66  | • PIC16C621  | • PIC16C923  |
| • PIC16C67  | • PIC16C621A | • PIC16C924  |
| • PIC16C71  | • PIC16C622  |              |
| • PIC16C72  | • PIC16C622A |              |

### 1.0 PROGRAMMING THE PIC16C6XX/7XX/9XX

The PIC16C6XX/7XX/9XX can be programmed using a serial method. In serial mode the PIC16C6XX/7XX/9XX can be programmed while in the users system. This allows for increased design flexibility. This programming specification applies to PIC16C6XX/7XX/9XX devices in all packages.

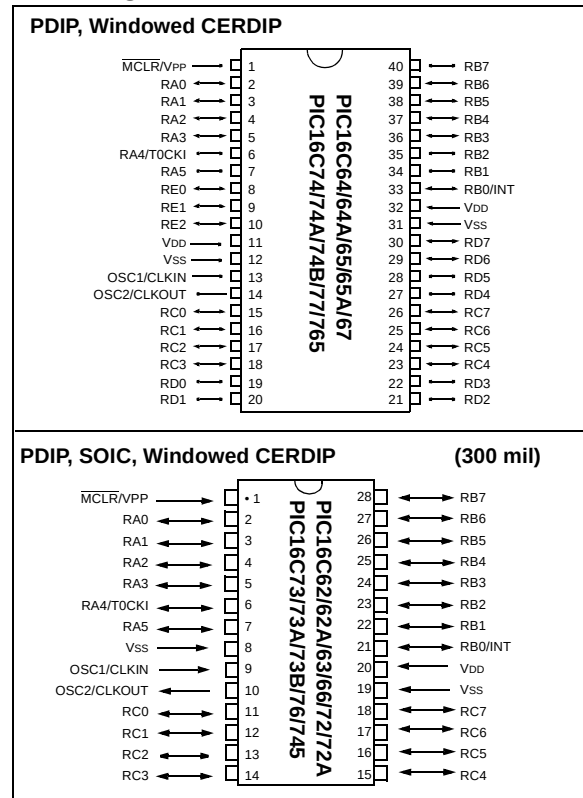
#### 1.1 Hardware Requirements

The PIC16C6XX/7XX/9XX requires two programmable power supplies, one for V<sub>DD</sub> (2.0V to 6.5V recommended) and one for V<sub>PP</sub> (12V to 14V). Both supplies should have a minimum resolution of 0.25V.

#### 1.2 Programming Mode

The programming mode for the PIC16C6XX/7XX/9XX allows programming of user program memory, special locations used for ID, and the configuration word for the PIC16C6XX/7XX/9XX.

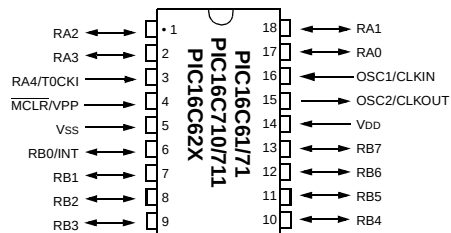
### Pin Diagrams



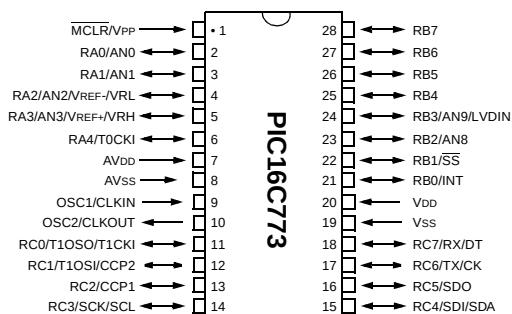
# PIC16C6XX/7XX/9XX

## Pin Diagrams (Con't)

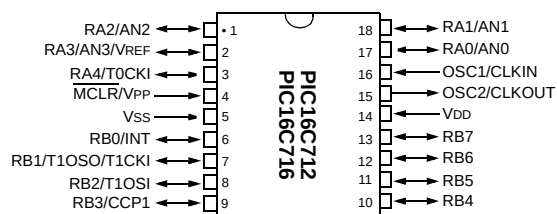
PDIP, SOIC, Windowed Cerdip



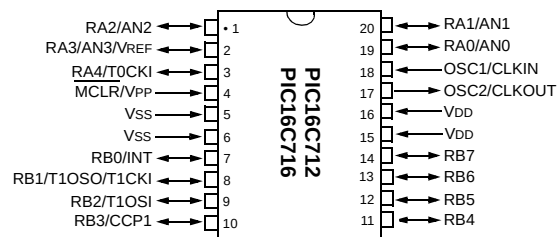
300 mil. SDIP, SOIC, Windowed Cerdip, SSOP



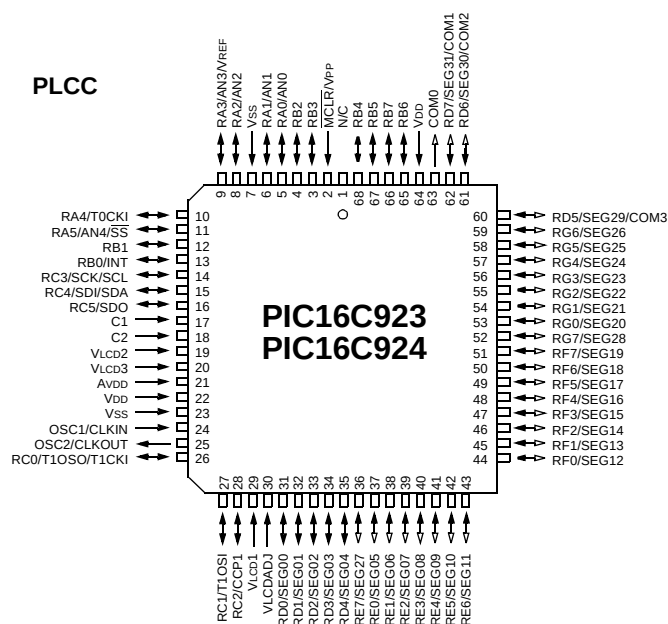
18 pin PDIP, SOIC, Windowed Cerdip



20 pin SSOP



PLCC



## 2.0 PROGRAM MODE ENTRY

### 2.1 User Program Memory Map

The user memory space extends from 0x0000 to 0x1FFF (8K). Table 2-1 shows actual implementation of program memory in the PIC16C6XX/7XX/9XX family.

**TABLE 2-1: IMPLEMENTATION OF PROGRAM MEMORY IN THE PIC16C6XX/7XX/9XX**

| Device           | Program Memory Size  |
|------------------|----------------------|
| PIC16C61         | 0x000 – 0x3FF (1K)   |
| PIC16C620/620A   | 0x000 – 0x1FF (0.5K) |
| PIC16C621/621A   | 0x000 – 0x3FF (1K)   |
| PIC16C622/622A   | 0x000 – 0x7FF (2K)   |
| PIC16C62/62A/62B | 0x000 – 0x7FF (2K)   |
| PIC16C63/63A     | 0x000 – 0xFFF (4K)   |
| PIC16C64/64A     | 0x000 – 0x7FF (2K)   |
| PIC16C65/65A/65B | 0x000 – 0xFFF (4K)   |
| PIC16CE623       | 0x000 – 0x1FF (0.5K) |
| PIC16CE624       | 0x000 – 0x3FF (1K)   |
| PIC16CE625       | 0x000 – 0x7FF (2K)   |
| PIC16C71         | 0x000 – 0x3FF (1K)   |
| PIC16C710        | 0x000 – 0x1FF (0.5K) |
| PIC16C711        | 0x000 – 0x3FF (1K)   |
| PIC16C712        | 0x000 – 0x3FF (1K)   |
| PIC16C716        | 0x000 – 0x7FF (2K)   |
| PIC16C72/72A     | 0x000 – 0x7FF (2K)   |
| PIC16C73/73A/73B | 0x000 – 0xFFF (4K)   |
| PIC16C74/74A/74B | 0x000 – 0xFFF (4K)   |
| PIC16C66         | 0x000 – 0x1FFF (8K)  |
| PIC16C67         | 0x000 – 0x1FFF (8K)  |
| PIC16C76         | 0x000 – 0x1FFF (8K)  |
| PIC16C77         | 0x000 – 0x1FFF (8K)  |
| PIC16C745        | 0x000 – 0x1FFF (8K)  |
| PIC16C765        | 0x000 – 0x1FFF (8K)  |
| PIC16C773        | 0x000 – 0xFFF (4K)   |
| PIC16C774        | 0x000 – 0xFFF (4K)   |
| PIC16C923/924    | 0x000 – 0xFFF (4K)   |

When the PC reaches the last location of the implemented program memory, it will wrap around and address a location within the physically implemented memory (see Figure 2-1).

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 user program memory is to reset the part and reenter program/verify mode, as described in Section 2.2.

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 four least significant bits of each ID location. In some devices, the ID locations read-out in a scrambled fashion after code protection is enabled. For these devices, it is recommended that ID location is written as "11 1111 1bbb bbbb" where 'bbbb' is ID information.

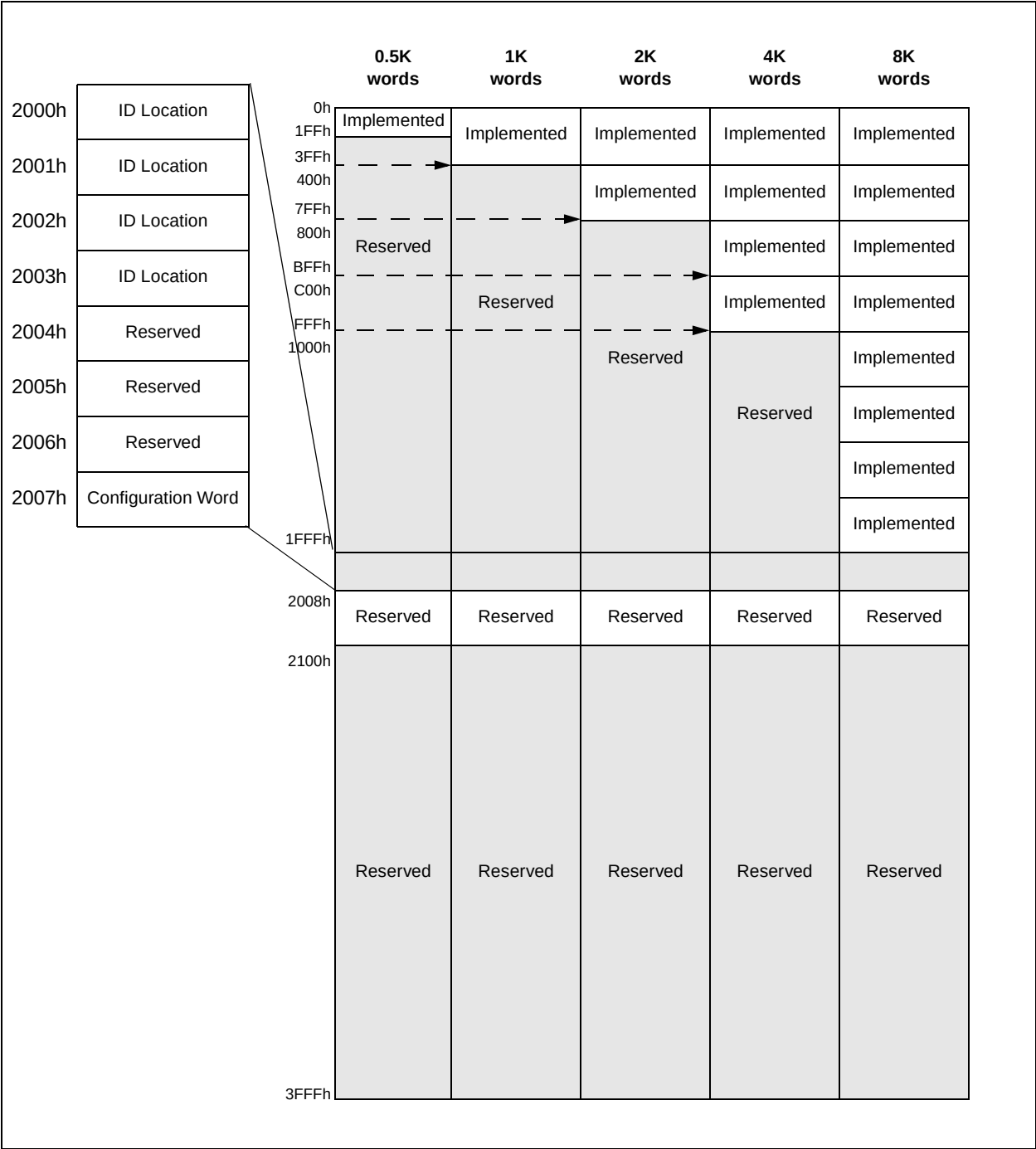
**Note:** All other locations are reserved and should not be programmed.

In other devices, the ID locations read out normally, even after code protection. To understand how the devices behave, refer to Table 4-1.

To understand the scrambling mechanism after code protection, refer to Section 3.1.

# PIC16C6XX/7XX/9XX

FIGURE 2-1: PROGRAM MEMORY MAPPING



## 2.2 Program/Verify Mode

The program/verify mode is entered by holding pins RB6 and RB7 low while raising MCLR pin from Vss to the appropriate VIH (high voltage). Once in this mode the user program memory and the configuration memory can be accessed and programmed in serial fashion. The mode of operation is serial, and the memory that is accessed is the user program memory. RB6 is a Schmitt Trigger input in this mode.

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 Vss). This means that all I/O are in the reset state (High impedance inputs).

**Note 1:** The MCLR pin should be raised as quickly as possible from VIL to VIH. This is to ensure that the device does not have the PC incremented while in valid operation range.

**2:** Do not power any pin before VDD is applied.

### 2.2.1 PROGRAM/VERIFY OPERATION

The RB6 pin is used as a clock input pin, and the RB7 pin is used for entering command bits and data input/output during serial operation. To input a command, the clock pin (RB6) is cycled six times. Each command bit is latched on the falling edge of the clock with the least significant bit (LSb) of the command being input first. The data on pin RB7 is required to have a minimum setup and hold time (see AC/DC specs) 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  $\mu$ s 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 RB7 on the rising edge of the second cycle, and during a load operation the LSb will be latched on the falling edge of the second cycle. A minimum 1  $\mu$ s 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  $\mu$ s is required between a command and a data word (or another command).

The commands that are available are listed in Table 2-2.

#### 2.2.1.1 LOAD CONFIGURATION

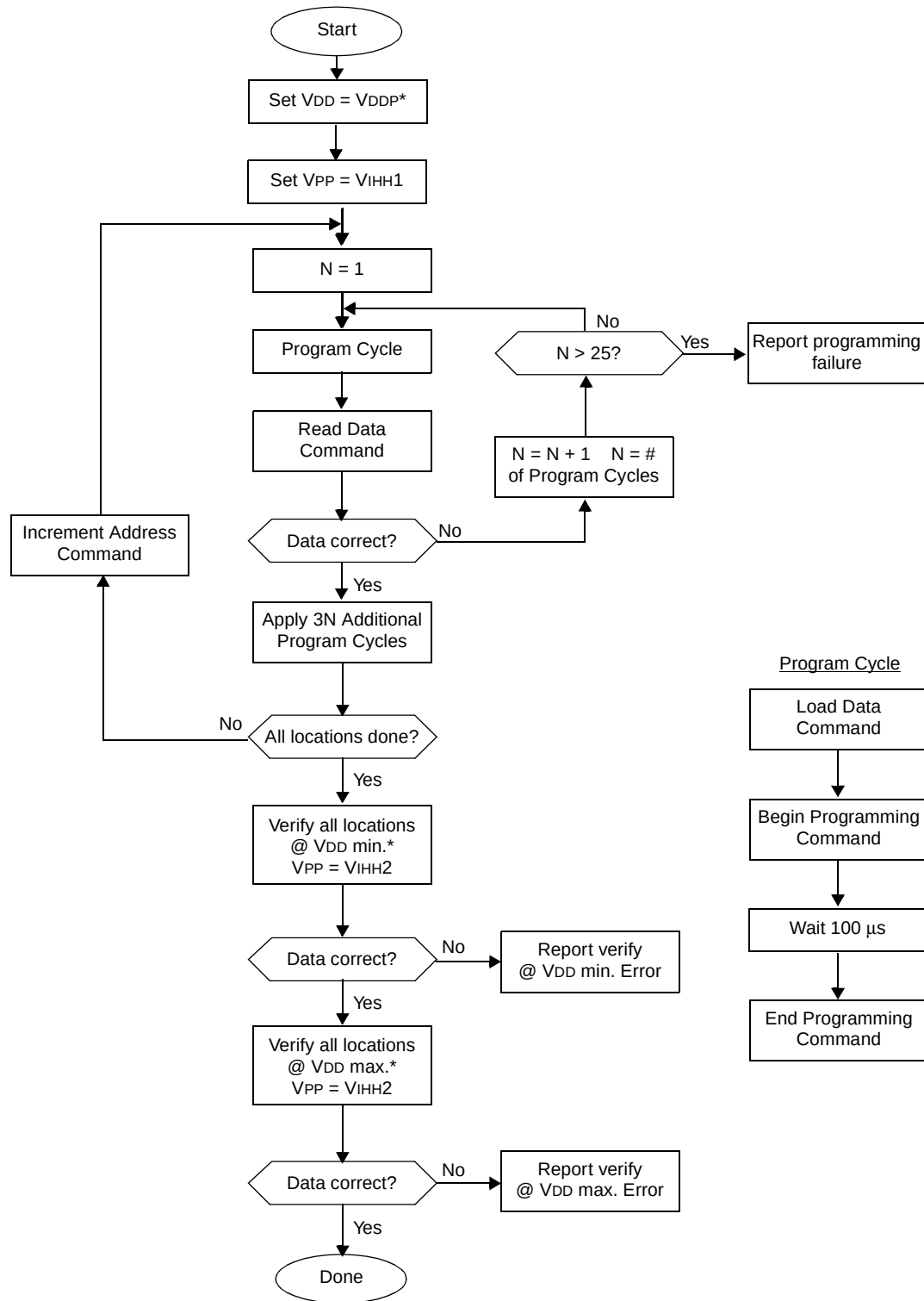
After receiving this command, the program counter (PC) will be set to 0x2000. By then applying 16 cycles to the clock pin, the chip will load 14-bits a "data word" as described above, to be programmed into the configuration memory. A description of the memory mapping schemes for normal operation and 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 (VIL).

**TABLE 2-2: COMMAND MAPPING**

| Command            | Mapping (MSb ... LSb) |   |   |   |   |   | Data           |
|--------------------|-----------------------|---|---|---|---|---|----------------|
| Load Configuration | 0                     | 0 | 0 | 0 | 0 | 0 | 0, data(14), 0 |
| Load Data          | 0                     | 0 | 0 | 0 | 1 | 0 | 0, data(14), 0 |
| Read Data          | 0                     | 0 | 0 | 1 | 0 | 0 | 0, data(14), 0 |
| Increment Address  | 0                     | 0 | 0 | 1 | 1 | 0 |                |
| Begin programming  | 0                     | 0 | 1 | 0 | 0 | 0 |                |
| End Programming    | 0                     | 0 | 1 | 1 | 1 | 0 |                |

**Note:** The clock must be disabled during In-Circuit Serial Programming.

FIGURE 2-2: PROGRAM FLOW CHART - PIC16C6XX/7XX/9XX PROGRAM MEMORY

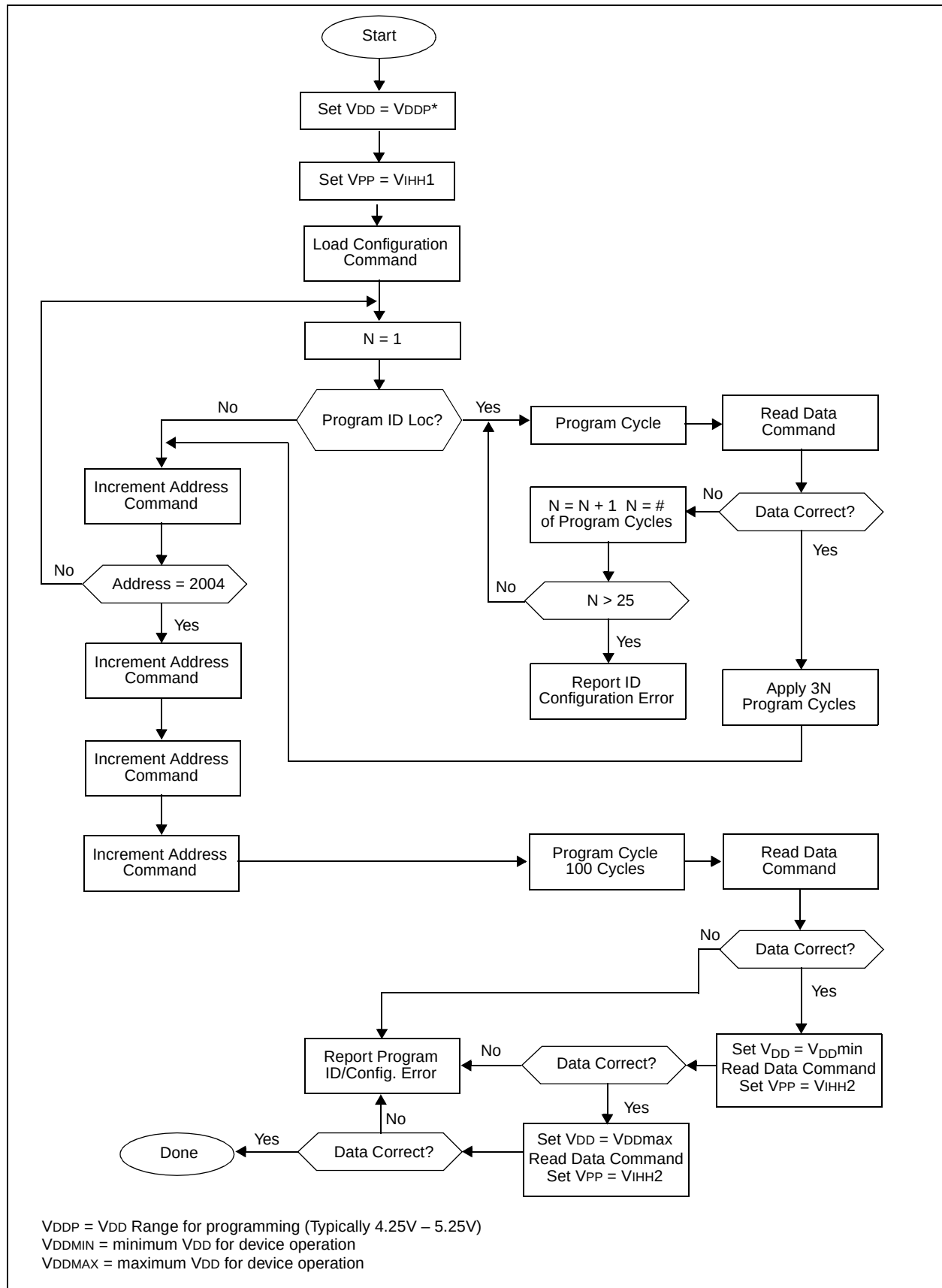


\* VDDP = VDD range for programming (typically 4.75V - 5.25V).

VDDmin = Minimum VDD for device operation.

VDDmax = Maximum VDD for device operation.

**FIGURE 2-3: PROGRAM FLOW CHART - PIC16C6XX/7XX/9XX CONFIGURATION WORD & ID LOCATIONS**



## 2.2.1.2 LOAD DATA

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 4-1.

## 2.2.1.3 READ DATA

After receiving this command, the chip will transmit data bits out of the memory currently accessed starting with the second rising edge of the clock input. The RB7 pin will go into output mode on the second rising clock edge, and it will revert back to input mode (hi-impedance) after the 16th rising edge. A timing diagram of this command is shown in Figure 4-2.

## 2.2.1.4 INCREMENT ADDRESS

The PC is incremented when this command is received. A timing diagram of this command is shown in Figure 4-3.

## 2.2.1.5 BEGIN PROGRAMMING

**A load command (load configuration or load data) must be given before every begin programming command.** Programming of the appropriate memory (test program memory or user program memory) will begin after this command is received and decoded. Programming should be performed with a series of 100 $\mu$ s programming pulses. A programming pulse is defined as the time between the begin programming command and the end programming command.

## 2.2.1.6 END PROGRAMMING

After receiving this command, the chip stops programming the memory (configuration program memory or user program memory) that it was programming at the time.

## 2.3 Programming Algorithm Requires Variable VDD

The PIC16C6XX/7XX/9XX uses an intelligent algorithm. The algorithm calls for program verification at VDDmin as well as VDDmax. Verification at VDDmin guarantees good “erase margin”. Verification at VDDmax guarantees good “program margin”.

The actual programming must be done with VDD in the VDDP range (4.75 - 5.25V).

VDDP = VCC range required during programming.

VDD min. = minimum operating VDD spec for the part.

VDDmax = maximum operating VDD spec for the part.

Programmers must verify the PIC16C6XX/7XX/9XX at its specified VDDmax and VDDmin levels. Since Microchip may introduce future versions of the PIC16C6XX/7XX/9XX with a broader VDD range, it is best that these levels are user selectable (defaults are ok).

|                                                                                                                                                                         |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>Note:</b> Any programmer not meeting these requirements may only be classified as “prototype” or “development” programmer but not a “production” quality programmer. |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|



## 3.0 CONFIGURATION WORD

The PIC16C6XX/7XX/9XX family members have several configuration bits. These bits can be programmed (reads '0') or left unprogrammed (reads '1') to select various device configurations. Figure 3-1 and Figure 3-2 provides an overview of configuration bits.

# PIC16C6XX/7XX/9XX

**FIGURE 3-1: CONFIGURATION WORD BIT MAP**

| Bit Number:                                                                                                      | 13  | 12  | 11  | 10  | 9   | 8   | 7 | 6     | 5   | 4   | 3                          | 2    | 1     | 0     |
|------------------------------------------------------------------------------------------------------------------|-----|-----|-----|-----|-----|-----|---|-------|-----|-----|----------------------------|------|-------|-------|
| PIC16C61/71                                                                                                      | —   | —   | —   | —   | —   | —   | — | —     | —   | CP0 | PWRT $\overline{\text{E}}$ | WDTE | FOSC1 | FOSC0 |
| PIC16C62/64/65/73/74                                                                                             | —   | —   | —   | —   | —   | —   | — | 0     | CP1 | CP0 | PWRT $\overline{\text{E}}$ | WDTE | FOSC1 | FOSC0 |
| PIC16C62A/62B/63A/CR62/63/64A/CR64/65A/65B/66/67/72/72A/73A/73B/74A/74B/76/77/620/620A/621/621A/622/622A/712/716 | CP1 | CP0 | CP1 | CP0 | CP1 | CP0 | — | BODEN | CP1 | CP0 | PWRT $\overline{\text{E}}$ | WDTE | FOSC1 | FOSC0 |
| PIC16C9XX/745/765                                                                                                | CP1 | CP0 | CP1 | CP0 | CP1 | CP0 | — | —     | CP1 | CP0 | PWRT $\overline{\text{E}}$ | WDTE | FOSC1 | FOSC0 |

Reserved, '—' write as '1' for PIC16C6XX/7XX/9XX

## CP <1:0>, Code Protect

| Device                 | CP1 | CP0 | Code Protection            |
|------------------------|-----|-----|----------------------------|
| PIC16C622/622A         | 0   | 0   | All memory protected       |
| PIC16C62/62A/62B       | 0   | 0   | All memory protected       |
| PIC16C63/63A           | 0   | 1   | Upper 3/4 memory protected |
| PIC16C64/64A/712/716   | 0   | 1   | Upper 3/4 memory protected |
| PIC16C65/65A/65B       | 1   | 0   | Upper 1/2 memory protected |
| PIC16C66/67/72/72A     | 1   | 0   | Upper 1/2 memory protected |
| PIC16C73/73A/73B       | 1   | 1   | Code protection off        |
| PIC16C74/74A/74B/76/77 | 1   | 1   | Code protection off        |
| PIC16C745/765          | 1   | 1   | Code protection off        |
| PIC16C9XX              | 1   | 1   | Code protection off        |
| PIC16C61/71            | —   | 0   | All memory protected       |
| PIC16C710/711          | —   | 1   | Off                        |
| PIC16C620              | 0   | 0   | All memory protected       |
|                        | 0   | 1   | Do not use                 |
|                        | 1   | 0   | Do not use                 |
|                        | 1   | 1   | Code protection off        |
| PIC16C621              | 0   | 0   | All memory protected       |
|                        | 1   | 0   | Upper 1/2 memory protected |
|                        | 1   | 1   | Code protection off        |

bit 6: **BODEN**, Brown Out Enable Bit

1 = Enabled

0 = Disable

bit 4: **PWRT $\overline{\text{E}}$ /PWRT $\overline{\text{E}}$** , Power Up Timer Enable Bit

PIC16C61/62/64/65/71/73/74:

1 = Power up timer enabled

0 = Power up timer disabled

PIC16C620/620A/621/621A/622/622A/62A/63/63A/65A/65B/66/67/72/72A/73A/73B/74A/74B/76/77/710/

711/923/924/745/765:

0 = Power up timer enabled

1 = Power up timer disabled

bit 3-2: **WDTE**, WDT Enable Bit

1 = WDT enabled

0 = WDT disabled

bit 1-0: **FOSC<1:0>**, Oscillator Selection Bit

11: RC oscillator

10: HS oscillator

01: XT oscillator

00: LP oscillator

bit 1-0: **FOSC<1:0>**, PIC16C745/765

11: E external clock with 4k PLL

10: H HS oscillator with 4k PL enabled

01: EC external clock, clkout on osc2

00: HS

Note 1: Enabling Brown-out Reset automatically enables the Power-up Timer (PWRT) regardless of the value of bit PWRT $\overline{\text{E}}$ . Ensure the Power-up Timer is enabled anytime Brown-out Reset is enabled.

**FIGURE 3-2: CONFIGURATION WORD FOR PIC16C773/774 DEVICE**

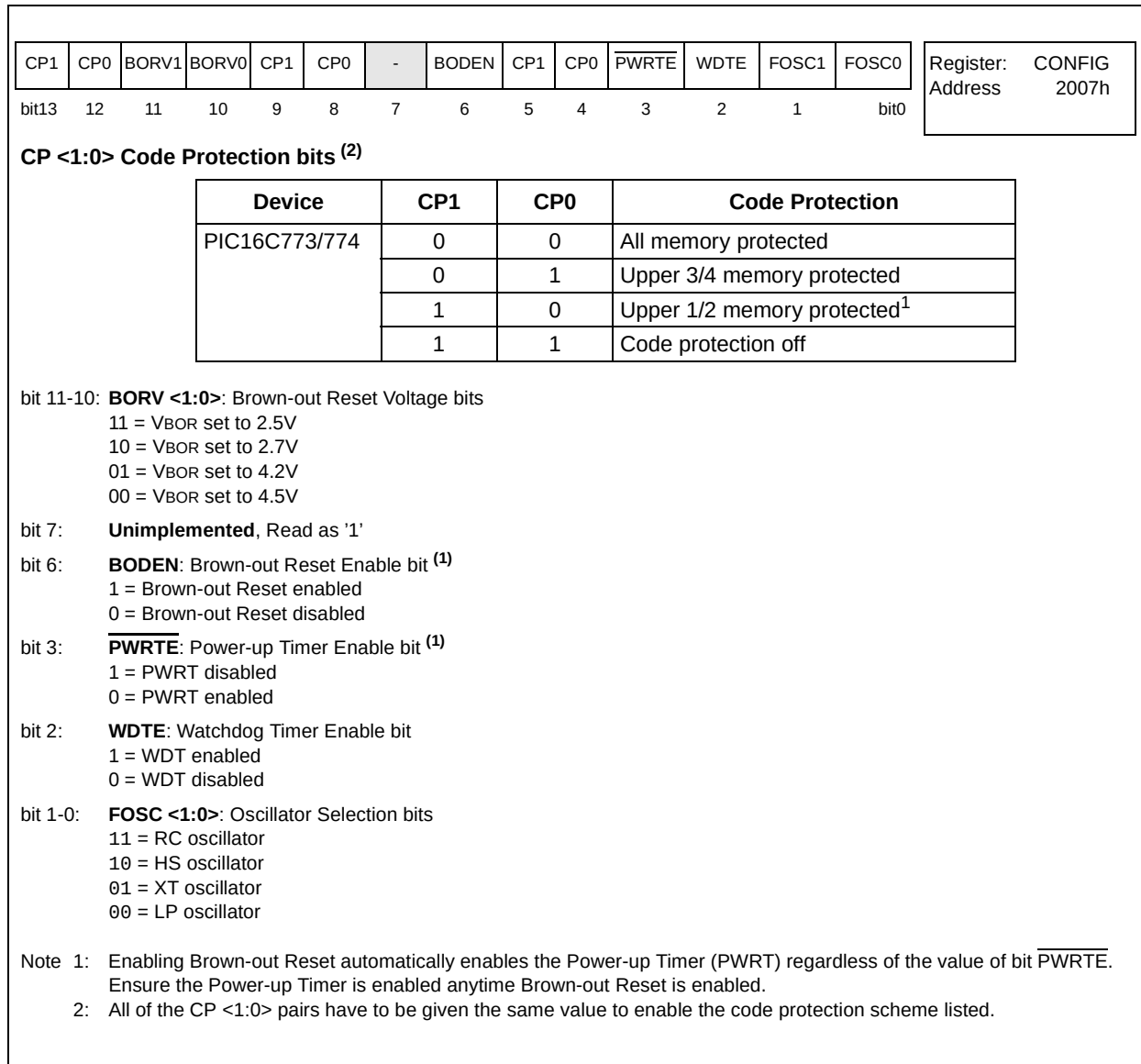
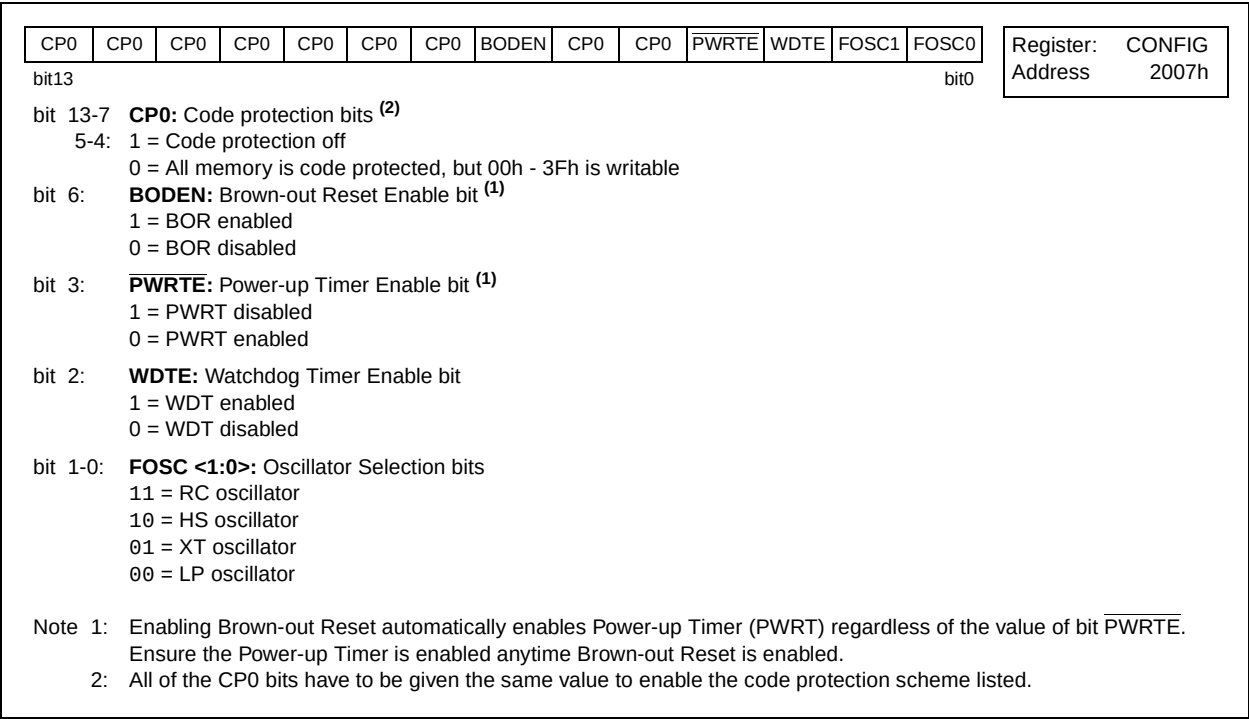


FIGURE 3-3: CONFIGURATION WORD, PIC16C710/711



## 3.1 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 beneficial to the end customer.

# PIC16C6XX/7XX/9XX

## 3.2 Checksum

### 3.2.1 CHECKSUM CALCULATIONS

Checksum is calculated by reading the contents of the PIC16C6XX/7XX/9XX memory locations and adding up the opcodes up to the maximum user addressable location, e.g., 0x1FF for the PIC16C74. 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 PIC16C6XX/7XX/9XX devices is shown in Table 3-1.

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 is the checksum.

The following table describes how to calculate the checksum for each device. Note that the checksum calculation differs depending on the code protect setting. Since the program memory locations read out differently depending on the code protect 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 by reading a device, the entire program memory can simply be read and summed. The configuration word and ID locations can always be read.

Note that some older devices have an additional value added in the checksum. This is to maintain compatibility with older device programmer checksums.

TABLE 3-1: CHECKSUM COMPUTATION

| Device     | Code Protect             | Checksum*                                                                                                                                            | Blank Value                          | 0x25E6 at 0 and max address          |
|------------|--------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------|--------------------------------------|
| PIC16C61   | OFF<br>ON                | SUM[0x000:0x3FF] + CFGW & 0x001F + 0x3FE0<br>SUM_XNOR7[0x000:0x3FF] + (CFGW & 0x001F   0x0060)                                                       | 0x3BFF<br>0xFC6F                     | 0x07CD<br>0xFC15                     |
| PIC16C620  | OFF<br>ON                | SUM[0x000:0x1FF] + CFGW & 0x3F7F<br>SUM_ID + CFGW & 0x3F7F                                                                                           | 0x3D7F<br>0x3DCE                     | 0x094D<br>0x099C                     |
| PIC16C620A | OFF<br>ON                | SUM[0x000:0x1FF] + CFGW & 0x3F7F<br>SUM_ID + CFGW & 0x3F7F                                                                                           | 0x3D7F<br>0x3DCE                     | 0x094D<br>0x099C                     |
| PIC16C621  | OFF<br>1/2<br>ALL        | SUM[0x000:0x3FF] + CFGW & 0x3F7F<br>SUM[0x000:0x1FF] + CFGW & 0x3F7F + SUM_ID<br>CFGW & 0x3F7F + SUM_ID                                              | 0x3B7F<br>0x4EDE<br>0x3BCE           | 0x074D<br>0x0093<br>0x079C           |
| PIC16C621A | OFF<br>1/2<br>ALL        | SUM[0x000:0x3FF] + CFGW & 0x3F7F<br>SUM[0x000:0x1FF] + CFGW & 0x3F7F + SUM_ID<br>CFGW & 0x3F7F + SUM_ID                                              | 0x3B7F<br>0x4EDE<br>0x3BCE           | 0x074D<br>0x0093<br>0x079C           |
| PIC16C622  | OFF<br>1/2<br>3/4<br>ALL | SUM[0x000:0x7FF] + CFGW & 0x3F7F<br>SUM[0x000:0x3FF] + CFGW & 0x3F7F + SUM_ID<br>SUM[0x000:0x1FF] + CFGW & 0x3F7F + SUM_ID<br>CFGW & 0x3F7F + SUM_ID | 0x377F<br>0x5DEE<br>0x4ADE<br>0x37CE | 0x034D<br>0x0FA3<br>0xFC93<br>0x039C |
| PIC16C622A | OFF<br>1/2<br>3/4<br>ALL | SUM[0x000:0x7FF] + CFGW & 0x3F7F<br>SUM[0x000:0x3FF] + CFGW & 0x3F7F + SUM_ID<br>SUM[0x000:0x1FF] + CFGW & 0x3F7F + SUM_ID<br>CFGW & 0x3F7F + SUM_ID | 0x377F<br>0x5DEE<br>0x4ADE<br>0x37CE | 0x034D<br>0x0FA3<br>0xFC93<br>0x039C |
| PIC16CE623 | OFF<br>ON                | SUM[0x000:0x1FF] + CFGW & 0x3F7F<br>SUM_ID + CFGW & 0x3F7F                                                                                           | 0x3D7F<br>0x3DCE                     | 0x094D<br>0x099C                     |

Legend: CFGW = Configuration Word

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

SUM\_XNOR7[a:b] = XNOR of the seven high order bits of memory location with the seven low order bits summed over locations a through b inclusive. For example, XNOR(0x3C31)=0x78 XNOR 0c31 = 0x0036.

SUM\_ID = ID locations masked by 0xF then made into a 16-bit value with ID0 as the most significant nibble. For example, ID0 = 0x12, ID1 = 0x37, ID2 = 0x4, ID3 = 0x26, then SUM\_ID = 0x2746.

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

+ = Addition

& = Bitwise AND

| = Bitwise OR

**TABLE 3-1: CHECKSUM COMPUTATION (CONTINUED)**

| Device     | Code Protect             | Checksum*                                                                                                                                                                                                                                | Blank Value                          | 0x25E6 at 0 and max address          |
|------------|--------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------|--------------------------------------|
| PIC16CE624 | OFF<br>1/2<br>ALL        | SUM[0x000:0x3FF] + CFGW & 0x3F7F<br>SUM[0x000:0x1FF] + CFGW & 0x3F7F + SUM_ID<br>CFGW & 0x3F7F + SUM_ID                                                                                                                                  | 0x3B7F<br>0x4EDE<br>0x3BCE           | 0x074D<br>0x0093<br>0x079C           |
| PIC16CE625 | OFF<br>1/2<br>3/4<br>ALL | SUM[0x000:0x7FF] + CFGW & 0x3F7F<br>SUM[0x000:0x3FF] + CFGW & 0x3F7F + SUM_ID<br>SUM[0x000:0x1FF] + CFGW & 0x3F7F + SUM_ID<br>CFGW & 0x3F7F + SUM_ID                                                                                     | 0x377F<br>0x5DEE<br>0x4ADE<br>0x37CE | 0x034D<br>0x0FA3<br>0xFC93<br>0x039C |
| PIC16C62   | OFF<br>1/2<br>3/4<br>ALL | SUM[0x000:0x7FF] + CFGW & 0x003F + 0x3F80<br>SUM[0x000:0x3FF] + SUM_XNOR7[0x400:0x7FF] + CFGW & 0x003F + 0x3F80<br>SUM[0x000:0x1FF] + SUM_XNOR7[0x200:0x7FF] + CFGW & 0x003F + 0x3F80<br>SUM_XNOR7[0x000:0x7FF] + CFGW & 0x003F + 0x3F80 | 0x37BF<br>0x37AF<br>0x379F<br>0x378F | 0x038D<br>0x1D69<br>0x1D59<br>0x3735 |
| PIC16C62A  | OFF<br>1/2<br>3/4<br>ALL | SUM[0x000:0x7FF] + CFGW & 0x3F7F<br>SUM[0x000:0x3FF] + CFGW & 0x3F7F + SUM_ID<br>SUM[0x000:0x1FF] + CFGW & 0x3F7F + SUM_ID<br>CFGW & 0x3F7F + SUM_ID                                                                                     | 0x377F<br>0x5DEE<br>0x4ADE<br>0x37CE | 0x034D<br>0x0FA3<br>0xFC93<br>0x039C |
| PIC16C62B  | OFF<br>1/2<br>3/4<br>ALL | SUM[0x000:0x7FF] + CFGW & 0x3F7F<br>SUM[0x000:0x3FF] + CFGW & 0x3F7F + SUM_ID<br>SUM[0x000:0x1FF] + CFGW & 0x3F7F + SUM_ID<br>CFGW & 0x3F7F + SUM_ID                                                                                     | 0x377F<br>0x5DEE<br>0x4ADE<br>0x37CE | 0x034D<br>0x0FA3<br>0xFC93<br>0x039C |
| PIC16C63   | OFF<br>1/2<br>3/4<br>ALL | SUM[0x000:0xFFF] + CFGW & 0x3F7F<br>SUM[0x000:0x7FF] + CFGW & 0x3F7F + SUM_ID<br>SUM[0x000:0x3FF] + CFGW & 0x3F7F + SUM_ID<br>CFGW & 0x3F7F + SUM_ID                                                                                     | 0x2F7F<br>0x51EE<br>0x40DE<br>0x2FCE | 0xFB4D<br>0x03A3<br>0xF293<br>0xFB9C |
| PIC16C63A  | OFF<br>1/2<br>3/4<br>ALL | SUM[0x000:0xFFF] + CFGW & 0x3F7F<br>SUM[0x000:0x7FF] + CFGW & 0x3F7F + SUM_ID<br>SUM[0x000:0x3FF] + CFGW & 0x3F7F + SUM_ID<br>CFGW & 0x3F7F + SUM_ID                                                                                     | 0x2F7F<br>0x51EE<br>0x40DE<br>0x2FCE | 0xFB4D<br>0x03A3<br>0xF293<br>0xFB9C |
| PIC16C64   | OFF<br>1/2<br>3/4<br>ALL | SUM[0x000:0x7FF] + CFGW & 0x003F + 0x3F80<br>SUM[0x000:0x3FF] + SUM_XNOR7[0x400:0x7FF] + CFGW & 0x003F + 0x3F80<br>SUM[0x000:0x1FF] + SUM_XNOR7[0x200:0x7FF] + CFGW & 0x003F + 0x3F80<br>SUM_XNOR7[0x000:0x7FF] + CFGW & 0x003F + 0x3F80 | 0x37BF<br>0x37AF<br>0x379F<br>0x378F | 0x038D<br>0x1D69<br>0x1D59<br>0x3735 |
| PIC16C64A  | OFF<br>1/2<br>3/4<br>ALL | SUM[0x000:0x7FF] + CFGW & 0x3F7F<br>SUM[0x000:0x3FF] + CFGW & 0x3F7F + SUM_ID<br>SUM[0x000:0x1FF] + CFGW & 0x3F7F + SUM_ID<br>CFGW & 0x3F7F + SUM_ID                                                                                     | 0x377F<br>0x5DEE<br>0x4ADE<br>0x37CE | 0x034D<br>0x0FA3<br>0xFC93<br>0x039C |
| PIC16C65   | OFF<br>1/2<br>3/4<br>ALL | SUM[0x000:0xFFF] + CFGW & 0x003F + 0x3F80<br>SUM[0x000:0x7FF] + SUM_XNOR7[0x800:FFF] + CFGW & 0x003F + 0x3F80<br>SUM[0x000:0x3FF] + SUM_XNOR7[0x400:FFF] + CFGW & 0x003F + 0x3F80<br>SUM_XNOR7[0x000:0xFFF] + CFGW & 0x003F + 0x3F80     | 0x2FBF<br>0x2FAF<br>0x2F9F<br>0x2F8F | 0xFB8D<br>0x1569<br>0x1559<br>0x2F35 |

Legend: CFGW = Configuration Word

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

SUM\_XNOR7[a:b] = XNOR of the seven high order bits of memory location with the seven low order bits summed over locations a through b inclusive. For example, XNOR(0x3C31)=0x78 XNOR 0c31 = 0x0036.

SUM\_ID = ID locations masked by 0xF then made into a 16-bit value with ID0 as the most significant nibble. For example, ID0 = 0x12, ID1 = 0x37, ID2 = 0x4, ID3 = 0x26, then SUM\_ID = 0x2746.

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

+ = Addition

& = Bitwise AND

| = Bitwise OR

# PIC16C6XX/7XX/9XX

**TABLE 3-1: CHECKSUM COMPUTATION (CONTINUED)**

| Device    | Code Protect             | Checksum*                                                                                                                                                                                                                              | Blank Value                          | 0x25E6 at 0 and max address          |
|-----------|--------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------|--------------------------------------|
| PIC16C65A | OFF<br>1/2<br>3/4<br>ALL | SUM[0x000:0xFFFF] + CFGW & 0x3F7F<br>SUM[0x000:0x7FF] + CFGW & 0x3F7F + SUM_ID<br>SUM[0x000:0x3FF] + CFGW & 0x3F7F + SUM_ID<br>CFGW & 0x3F7F + SUM_ID                                                                                  | 0x2F7F<br>0x51EE<br>0x40DE<br>0x2FCE | 0xFB4D<br>0x03A3<br>0xF293<br>0xFB9C |
| PIC16C65B | OFF<br>1/2<br>3/4<br>ALL | SUM[0x000:0xFFFF] + CFGW & 0x3F7F<br>SUM[0x000:0x7FF] + CFGW & 0x3F7F + SUM_ID<br>SUM[0x000:0x3FF] + CFGW & 0x3F7F + SUM_ID<br>CFGW & 0x3F7F + SUM_ID                                                                                  | 0x2F7F<br>0x51EE<br>0x40DE<br>0x2FCE | 0xFB4D<br>0x03A3<br>0xF293<br>0xFB9C |
| PIC16C66  | OFF<br>1/2<br>3/4<br>ALL | SUM[0x000:0x1FFF] + CFGW & 0x3F7F<br>SUM[0x000:0xFFFF] + CFGW & 0x3F7F + SUM_ID<br>SUM[0x000:0x7FF] + CFGW & 0x3F7F + SUM_ID<br>CFGW & 0x3F7F + SUM_ID                                                                                 | 0x1F7F<br>0x39EE<br>0x2CDE<br>0x1FCE | 0xEB4D<br>0xEBA3<br>0xDE93<br>0xEB9C |
| PIC16C67  | OFF<br>1/2<br>3/4<br>ALL | SUM[0x000:0x1FFF] + CFGW & 0x3F7F<br>SUM[0x000:0xFFFF] + CFGW & 0x3F7F + SUM_ID<br>SUM[0x000:0x7FF] + CFGW & 0x3F7F + SUM_ID<br>CFGW & 0x3F7F + SUM_ID                                                                                 | 0x1F7F<br>0x39EE<br>0x2CDE<br>0x1FCE | 0xEB4D<br>0xEBA3<br>0xDE93<br>0xEB9C |
| PIC16C710 | OFF<br>ON                | SUM[0x000:0x1FF] + CFGW & 0x3FFF<br>SUM[0x00:0x3F] + CFGW & 0x3FFF + SUM_ID                                                                                                                                                            | 0x3DFF<br>0x3E0E                     | 0x09CD<br>0xEFC3                     |
| PIC16C71  | OFF<br>ON                | SUM[0x000:0x3FF] + CFGW & 0x001F + 0x3FE0<br>SUM_XNOR7[0x000:0x3FF] + (CFGW & 0x001F   0x0060)                                                                                                                                         | 0x3BFF<br>0xFC6F                     | 0x07CD<br>0xFC15                     |
| PIC16C711 | OFF<br>ON                | SUM[0x000:0x03FF] + CFGW & 0x3FFF<br>SUM[0x00:0x3FF] + CFGW & 0x3FFF + SUM_ID                                                                                                                                                          | 0x3BFF<br>0x3C0E                     | 0x07CD<br>0xEDC3                     |
| PIC16C712 | OFF<br>1/2<br>ALL        | SUM[0x000:0x07FF] + CFGW & 0x3F7F<br>SUM[0x000:0x03FF] + CFGW & 3F7F + SUM_ID<br>CFGW & 0x3F7F + SUM_ID                                                                                                                                | 0x377F<br>0x5DEE<br>0x37CE           | 0x034D<br>0xF58A<br>0x039C           |
| PIC16C716 | OFF<br>1/2<br>3/4<br>ALL | SUM[0x000:0x07FF] + CFGW & 0x3F7F<br>SUM[0x000:0x03FF] + CFGW & 0x3F7F + SUM_ID<br>SUM[0x000:0x01FF] + CFGW & 0x3F7F + SUM_ID<br>CFGW & 0x3F7F + SUM_ID                                                                                | 0x377F<br>0x5DEE<br>0x4ADE<br>0x37CE | 0x034D<br>0x0FA3<br>0xFC93<br>0x039C |
| PIC16C72  | OFF<br>1/2<br>3/4<br>ALL | SUM[0x000:0x7FF] + CFGW & 0x3F7F<br>SUM[0x000:0x3FF] + CFGW & 0x3F7F + SUM_ID<br>SUM[0x000:0x1FF] + CFGW & 0x3F7F + SUM_ID<br>CFGW & 0x3F7F + SUM_ID                                                                                   | 0x377F<br>0x5DEE<br>0x4ADE<br>0x37CE | 0x034D<br>0x0FA3<br>0xFC93<br>0x039C |
| PIC16C72A | OFF<br>1/2<br>3/4<br>ALL | SUM[0x000:0x7FF] + CFGW & 0x3F7F<br>SUM[0x000:0x3FF] + CFGW & 0x3F7F + SUM_ID<br>SUM[0x000:0x1FF] + CFGW & 0x3F7F + SUM_ID<br>CFGW & 0x3F7F + SUM_ID                                                                                   | 0x377F<br>0x5DEE<br>0x4ADE<br>0x37CE | 0x034D<br>0x0FA3<br>0xFC93<br>0x039C |
| PIC16C73  | OFF<br>1/2<br>3/4<br>ALL | SUM[0x000:0xFFFF] + CFGW & 0x003F + 0x3F80<br>SUM[0x000:0x7FF] + SUM_XNOR7[0x800:FFF] + CFGW & 0x003F + 0x3F80<br>SUM[0x000:0x3FF] + SUM_XNOR7[0x400:FFF] + CFGW & 0x003F + 0x3F80<br>SUM_XNOR7[0x000:0xFFFF] + CFGW & 0x003F + 0x3F80 | 0x2FBF<br>0x2FAF<br>0x2F9F<br>0x2F8F | 0xFB8D<br>0x1569<br>0x1559<br>0x2F35 |
| PIC16C73A | OFF<br>1/2<br>3/4<br>ALL | SUM[0x000:0xFFFF] + CFGW & 0x3F7F<br>SUM[0x000:0x7FF] + CFGW & 0x3F7F + SUM_ID<br>SUM[0x000:0x3FF] + CFGW & 0x3F7F + SUM_ID<br>CFGW & 0x3F7F + SUM_ID                                                                                  | 0x2F7F<br>0x51EE<br>0x40DE<br>0x2FCE | 0xFB4D<br>0x03A3<br>0xF293<br>0xFB9C |

Legend: CFGW = Configuration Word

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

SUM\_XNOR7[a:b] = XNOR of the seven high order bits of memory location with the seven low order bits summed over locations a through b inclusive. For example, XNOR(0x3C31)=0x78 XNOR 0c31 = 0x0036.

SUM\_ID = ID locations masked by 0xF then made into a 16-bit value with ID0 as the most significant nibble. For example, ID0 = 0x12, ID1 = 0x37, ID2 = 0x4, ID3 = 0x26, then SUM\_ID = 0x2746.

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

+ = Addition

& = Bitwise AND

| = Bitwise OR



**TABLE 3-1: CHECKSUM COMPUTATION (CONTINUED)**

| Device    | Code Protect                        | Checksum*                                                                                                                                                                                                                              | Blank Value                          | 0x25E6 at 0 and max address          |
|-----------|-------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------|--------------------------------------|
| PIC16C73B | OFF<br>1/2<br>3/4<br>ALL            | SUM[0x000:0xFFFF] + CFGW & 0x3F7F<br>SUM[0x000:0x7FF] + CFGW & 0x3F7F + SUM_ID<br>SUM[0x000:0x3FF] + CFGW & 0x3F7F + SUM_ID<br>CFGW & 0x3F7F + SUM_ID                                                                                  | 0x2F7F<br>0x51EE<br>0x40DE<br>0x2FCE | 0xFB4D<br>0x03A3<br>0xF293<br>0xFB9C |
| PIC16C74  | OFF<br>1/2<br>3/4<br>ALL            | SUM[0x000:0xFFFF] + CFGW & 0x003F + 0x3F80<br>SUM[0x000:0x7FF] + SUM_XNOR7[0x800:FFF] + CFGW & 0x003F + 0x3F80<br>SUM[0x000:0x3FF] + SUM_XNOR7[0x400:FFF] + CFGW & 0x003F + 0x3F80<br>SUM_XNOR7[0x000:0xFFFF] + CFGW & 0x003F + 0x3F80 | 0x2FBF<br>0x2FAF<br>0x2F9F<br>0x2F8F | 0xFB8D<br>0x1569<br>0x1559<br>0x2F35 |
| PIC16C74A | OFF<br>1/2<br>3/4<br>ALL            | SUM[0x000:0xFFFF] + CFGW & 0x3F7F<br>SUM[0x000:0x7FF] + CFGW & 0x3F7F + SUM_ID<br>SUM[0x000:0x3FF] + CFGW & 0x3F7F + SUM_ID<br>CFGW & 0x3F7F + SUM_ID                                                                                  | 0x2F7F<br>0x51EE<br>0x40DE<br>0x2FCE | 0xFB4D<br>0x03A3<br>0xF293<br>0xFB9C |
| PIC16C74B | OFF<br>1/2<br>3/4<br>ALL            | SUM[0x000:0xFFFF] + CFGW & 0x3F7F<br>SUM[0x000:0x7FF] + CFGW & 0x3F7F + SUM_ID<br>SUM[0x000:0x3FF] + CFGW & 0x3F7F + SUM_ID<br>CFGW & 0x3F7F + SUM_ID                                                                                  | 0x2F7F<br>0x51EE<br>0x40DE<br>0x2FCE | 0xFB4D<br>0x03A3<br>0xF293<br>0xFB9C |
| PIC16C76  | OFF<br>1/2<br>3/4<br>ALL            | SUM[0x000:0x1FFF] + CFGW & 0x3F7F<br>SUM[0x000:0xFFFF] + CFGW & 0x3F7F + SUM_ID<br>SUM[0x000:0x7FF] + CFGW & 0x3F7F + SUM_ID<br>CFGW & 0x3F7F + SUM_ID                                                                                 | 0x1F7F<br>0x39EE<br>0x2CDE<br>0x1FCE | 0xEB4D<br>0xEBA3<br>0xDE93<br>0xEB9C |
| PIC16C77  | OFF<br>1/2<br>3/4<br>ALL            | SUM[0x000:0x1FFF] + CFGW & 0x3F7F<br>SUM[0x000:0xFFFF] + CFGW & 0x3F7F + SUM_ID<br>SUM[0x000:0x7FF] + CFGW & 0x3F7F + SUM_ID<br>CFGW & 0x3F7F + SUM_ID                                                                                 | 0x1F7F<br>0x39EE<br>0x2CDE<br>0x1FCE | 0xEB4D<br>0xEBA3<br>0xDE93<br>0xEB9C |
| PIC16C773 | OFF<br>1/2<br>3/4<br>ALL            | SUM[0x000:0x0FFF] + CFGW & 0x3F7F<br>SUM[0x000:07FF] + CFGW & 0x3F7F + SUM_ID<br>SUM[0x000:03FF] + CFGW & 0x3F7F + SUM_ID<br>CFGW & 0x3F7F + SUM_ID                                                                                    | 0x2F7F<br>0x55EE<br>0x48DE<br>0x3BCE | 0xFB4D<br>0x07A3<br>0xFA93<br>0x079C |
| PIC16C774 | OFF<br>1/2<br>3/4<br>ALL            | SUM[0x000:0FFF] + CFGW & 0x3F7F<br>SUM[0x000:07FF] + CFGW & 0x3F7F + SUM_ID<br>SUM[0x000:03FF] + CFGW & 0x3F7F + SUM_ID<br>CFGW & 0x3F7F + SUM_ID                                                                                      | 0x2F7F<br>0x55EE<br>0x48DE<br>0x3BCE | 0xFB4D<br>0x07A3<br>0xFA93<br>0x079C |
| PIC16C923 | OFF<br>1/2<br>3/4<br>ALL            | SUM[0x000:0xFFFF] + CFGW & 0x3F3F<br>SUM[0x000:0x7FF] + CFGW & 0x3F3F + SUM_ID<br>SUM[0x000:0x3FF] + CFGW & 0x3F3F + SUM_ID<br>CFGW & 0x3F3F + SUM_ID                                                                                  | 0x2F3F<br>0x516E<br>0x405E<br>0x2F4E | 0xFB0D<br>0x0323<br>0xF213<br>0xFB1C |
| PIC16C924 | OFF<br>1/2<br>3/4<br>ALL            | SUM[0x000:0xFFFF] + CFGW & 0x3F3F<br>SUM[0x000:0x7FF] + CFGW & 0x3F3F + SUM_ID<br>SUM[0x000:0x3FF] + CFGW & 0x3F3F + SUM_ID<br>CFGW & 0x3F3F + SUM_ID                                                                                  | 0x2F3F<br>0x516E<br>0x405E<br>0x2F4E | 0xFB0D<br>0x0323<br>0xF213<br>0xFB1C |
| PIC16C745 | OFF<br>1000:1FFF<br>800:1FFF<br>ALL | SUM(0000:1FFF) + CFGW & 0x3F3F<br>SUM(0000:0FFF) + CFGW & 0x3F3F + SUM_ID<br>SUM(0000:07FF) + CFGW & 0x3F3F + SUM_ID<br>CFGW * 0x3F3F + SUM_ID                                                                                         | 1F3F<br>396E<br>2C5E<br>1F4E         | EB0D<br>EB23<br>DE13<br>EB1C         |

Legend: CFGW = Configuration Word

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

SUM\_XNOR7[a:b] = XNOR of the seven high order bits of memory location with the seven low order bits summed over locations a through b inclusive. For example, XNOR(0x3C31)=0x78 XNOR 0c31 = 0x0036.

SUM\_ID = ID locations masked by 0xF then made into a 16-bit value with ID0 as the most significant nibble. For example, ID0 = 0x12, ID1 = 0x37, ID2 = 0x4, ID3 = 0x26, then SUM\_ID = 0x2746.

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

+ = Addition

& = Bitwise AND

| = Bitwise OR

# PIC16C6XX/7XX/9XX

TABLE 3-1: CHECKSUM COMPUTATION (CONTINUED)

| Device    | Code Protect | Checksum*                               | Blank Value | 0x25E6 at 0 and max address |
|-----------|--------------|-----------------------------------------|-------------|-----------------------------|
| PIC16c765 | OFF          | SUM(0000:1FFF) + CFGW & 0x3F3F          | 1F3F        | EB0D                        |
|           | 1000:1FFF    | SUM(0000:0FFF) + CFGW & 0x3F3F+SUM_ID   | 396E        | EB23                        |
|           | 800:1FFF     | SUM(0000:07FF) + CFGW & 0x3F3F + SUM_ID | 2C5E        | DE13                        |
|           | ALL          | CFGW * 0x3F3F + SUM_ID                  | 1F4E        | EB1C                        |

Legend: CFGW = Configuration Word

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

SUM\_XNOR7[a:b] = XNOR of the seven high order bits of memory location with the seven low order bits summed over locations a through b inclusive. For example, XNOR(0x3C31)=0x78 XNOR 0c31 = 0x0036.

SUM\_ID = ID locations masked by 0xF then made into a 16-bit value with ID0 as the most significant nibble. For example, ID0 = 0x12, ID1 = 0x37, ID2 = 0x4, ID3 = 0x26, then SUM\_ID = 0x2746.

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

+ = Addition

& = Bitwise AND

| = Bitwise OR

## 4.0 PROGRAM/VERIFY MODE

**TABLE 4-1: AC/DC CHARACTERISTICS**  
**TIMING REQUIREMENTS FOR PROGRAM/VERIFY TEST MODE**

| Standard Operating Conditions                                                                                                |                 |                                              |           |      |        |       |                       |
|------------------------------------------------------------------------------------------------------------------------------|-----------------|----------------------------------------------|-----------|------|--------|-------|-----------------------|
| Operating Temperature: $+10^{\circ}\text{C} \leq T_A \leq +40^{\circ}\text{C}$ , unless otherwise stated, (20°C recommended) |                 |                                              |           |      |        |       |                       |
| Operating Voltage: $4.5\text{V} \leq V_{DD} \leq 5.5\text{V}$ , unless otherwise stated.                                     |                 |                                              |           |      |        |       |                       |
| Parameter No.                                                                                                                | Sym.            | Characteristic                               | Min.      | Typ. | Max.   | Units | Conditions            |
| <b>General</b>                                                                                                               |                 |                                              |           |      |        |       |                       |
| PD1                                                                                                                          | VDDP            | Supply voltage during programming            | 4.75      | 5.0  | 5.25   | V     |                       |
| PD2                                                                                                                          | IDDP            | Supply current (from VDD) during programming | –         | –    | 20     | mA    |                       |
| PD3                                                                                                                          | VDDV            | Supply voltage during verify                 | VDDmin    | –    | VDDmax | V     | Note 1                |
| PD4                                                                                                                          | VIHH1           | Voltage on MCLR/VPP during programming       | 12.75     | –    | 13.25  | V     | Note 2                |
| PD5                                                                                                                          | VIHH2           | Voltage on MCLR/VPP during verify            | VDD + 4.5 | –    | 13.25  | –     |                       |
| PD6                                                                                                                          | I <sub>PP</sub> | Programming supply current (from VPP)        | –         | –    | 50     | mA    |                       |
| PD9                                                                                                                          | VIH             | (RB6, RB7) input high level                  | 0.8 VDD   | –    | –      | V     | Schmitt Trigger input |
| PD8                                                                                                                          | VIL             | (RB6, RB7) input low level                   | 0.2 VDD   | –    | –      | V     | Schmitt Trigger input |

| <b>Serial Program Verify</b> |       |                                                                                                    |     |   |     |    |  |
|------------------------------|-------|----------------------------------------------------------------------------------------------------|-----|---|-----|----|--|
| P1                           | Tr    | MCLR/VPP rise time (VSS to VHH) for test mode entry                                                | –   | – | 8.0 | μs |  |
| P2                           | Tf    | MCLR Fall time                                                                                     | –   | – | 8.0 | μs |  |
| P3                           | Tset1 | Data in setup time before clock ↓                                                                  | 100 | – | –   | ns |  |
| P4                           | Thld1 | Data in hold time after clock ↓                                                                    | 100 | – | –   | ns |  |
| P5                           | Tdly1 | Data input not driven to next clock input (delay required between command/data or command/command) | 1.0 | – | –   | μs |  |
| P6                           | Tdly2 | Delay between clock ↓ to clock ↑ of next command or data                                           | 1.0 | – | –   | μs |  |
| P7                           | Tdly3 | Clock ↑ to data out valid (during read data)                                                       | 200 | – | –   | ns |  |
| P8                           | Thld0 | Hold time after MCLR ↑                                                                             | 2   | – | –   | μs |  |

**Note 1:** Program must be verified at the minimum and maximum VDD limits for the part.

**2:** VIHH must be greater than VDD + 4.5V to stay in programming/verify mode.



NOTES:

NOTES:

NOTES:



## WORLDWIDE SALES AND SERVICE

### AMERICAS

#### Corporate Office

Microchip Technology Inc.  
2355 West Chandler Blvd.  
Chandler, AZ 85224-6199  
Tel: 480-786-7200 Fax: 480-786-7277  
Technical Support: 480-786-7627  
Web Address: <http://www.microchip.com>

#### Atlanta

Microchip Technology Inc.  
500 Sugar Mill Road, Suite 200B  
Atlanta, GA 30350  
Tel: 770-640-0034 Fax: 770-640-0307

#### Boston

Microchip Technology Inc.  
5 Mount Royal Avenue  
Marlborough, MA 01752  
Tel: 508-480-9990 Fax: 508-480-8575

#### Chicago

Microchip Technology Inc.  
333 Pierce Road, Suite 180  
Itasca, IL 60143  
Tel: 630-285-0071 Fax: 630-285-0075

#### Dallas

Microchip Technology Inc.  
4570 Westgrove Drive, Suite 160  
Addison, TX 75248  
Tel: 972-818-7423 Fax: 972-818-2924

#### Dayton

Microchip Technology Inc.  
Two Prestige Place, Suite 150  
Miamisburg, OH 45342  
Tel: 937-291-1654 Fax: 937-291-9175

#### Detroit

Microchip Technology Inc.  
Tri-Atria Office Building  
32255 Northwestern Highway, Suite 190  
Farmington Hills, MI 48334  
Tel: 248-538-2250 Fax: 248-538-2260

#### Los Angeles

Microchip Technology Inc.  
18201 Von Karman, Suite 1090  
Irvine, CA 92612  
Tel: 949-263-1888 Fax: 949-263-1338

#### New York

Microchip Technology Inc.  
150 Motor Parkway, Suite 202  
Hauppauge, NY 11788  
Tel: 631-273-5305 Fax: 631-273-5335

#### San Jose

Microchip Technology Inc.  
2107 North First Street, Suite 590  
San Jose, CA 95131  
Tel: 408-436-7950 Fax: 408-436-7955

### AMERICAS (continued)

#### Toronto

Microchip Technology Inc.  
5925 Airport Road, Suite 200  
Mississauga, Ontario L4V 1W1, Canada  
Tel: 905-405-6279 Fax: 905-405-6253

### ASIA/PACIFIC

#### Beijing

Microchip Technology, Beijing  
Unit 915, 6 Chaoyangmen Bei Dajie  
Dong Erhuan Road, Dongcheng District  
New China Hong Kong Manhattan Building  
Beijing 100027 PRC  
Tel: 86-10-85282100 Fax: 86-10-85282104

#### Hong Kong

Microchip Asia Pacific  
Unit 2101, Tower 2  
Metroplaza  
223 Hing Fong Road  
Kwai Fong, N.T., Hong Kong  
Tel: 852-2-401-1200 Fax: 852-2-401-3431

#### India

Microchip Technology Inc.  
India Liaison Office  
No. 6, Legacy, Convent Road  
Bangalore 560 025, India  
Tel: 91-80-229-0061 Fax: 91-80-229-0062

#### Japan

Microchip Technology Intl. Inc.  
Benex S-1 6F  
3-18-20, Shinyokohama  
Kohoku-Ku, Yokohama-shi  
Kanagawa 222-0033 Japan  
Tel: 81-45-471- 6166 Fax: 81-45-471-6122

#### Korea

Microchip Technology Korea  
168-1, Youngbo Bldg. 3 Floor  
Samsung-Dong, Kangnam-Ku  
Seoul, Korea  
Tel: 82-2-554-7200 Fax: 82-2-558-5934

#### Shanghai

Microchip Technology  
Unit B701, Far East International Plaza,  
No. 317, Xianxia Road  
Shanghai, 200051 P.R.C  
Tel: 86-21-6275-5700 Fax: 86-21-6275-5060

### ASIA/PACIFIC (continued)

#### Singapore

Microchip Technology Singapore Pte Ltd.  
200 Middle Road  
#07-02 Prime Centre  
Singapore 188980  
Tel: 65-334-8870 Fax: 65-334-8850

#### Taiwan, R.O.C

Microchip Technology Taiwan  
10F-1C 207  
Tung Hua North Road  
Taipei, Taiwan, ROC  
Tel: 886-2-2717-7175 Fax: 886-2-2545-0139

### EUROPE

#### Denmark

Microchip Technology Denmark ApS  
Regus Business Centre  
Lautrup høj 1-3  
Ballerup DK-2750 Denmark  
Tel: 45 4420 9895 Fax: 45 4420 9910

#### France

Arizona Microchip Technology SARL  
Parc d'Activite du Moulin de Massy  
43 Rue du Saule Trapu  
Batiment A - 1er Etage  
91300 Massy, France  
Tel: 33-1-69-53-63-20 Fax: 33-1-69-30-90-79

#### Germany

Arizona Microchip Technology GmbH  
Gustav-Heinemann-Ring 125  
D-81739 München, Germany  
Tel: 49-89-627-144 0 Fax: 49-89-627-144-44

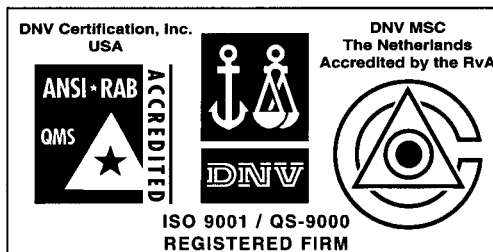
#### Italy

Arizona Microchip Technology SRL  
Centro Direzionale Colleoni  
Palazzo Taurus 1 V. Le Colleoni 1  
20041 Agrate Brianza  
Milan, Italy  
Tel: 39-039-65791-1 Fax: 39-039-6899883

#### United Kingdom

Arizona Microchip Technology Ltd.  
505 Eskdale Road  
Winnersh Triangle  
Wokingham  
Berkshire, England RG41 5TU  
Tel: 44 118 921 5858 Fax: 44-118 921-5835

01/21/00



Microchip received QS-9000 quality system certification for its worldwide headquarters, design and wafer fabrication facilities in Chandler and Tempe, Arizona in July 1999. The Company's quality system processes and procedures are QS-9000 compliant for its PICmicro® 8-bit MCUs, KEELoC® code hopping devices, Serial EEPROMs and microperipheral products. In addition, Microchip's quality system for the design and manufacture of development systems is ISO 9001 certified.

All rights reserved. © 2/1/00 Microchip Technology Incorporated. Printed in the USA. Tuesday, February 01, 2000 Printed on recycled paper.

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, except as maybe explicitly expressed herein, under any intellectual property rights. The Microchip logo and name are registered trademarks of Microchip Technology Inc. in the U.S.A. and other countries. All rights reserved. All other trademarks mentioned herein are the property of their respective companies.