PIC16C6X/7X/9XX

In-Circuit Serial Programming (ICSP™) for PIC16C6X/7X/9XX OTP MCUs

This document includes the programming specifications for the following devices:

- PIC16C61
- PIC16C62
- PIC16C62A
- PIC16C63
- PIC16C64
- PIC16C64A
- PIC16C65
- PIC16C65A
- PIC16C71
- PIC16C710
- PIC16C711
- PIC16C72
- PIC16C73
- PIC16C73A
- PIC16C74
- PIC16C74A
- PIC16C66
- PIC16C67
- PIC16C76
- PIC16C77
- PIC16C620
- PIC16C621
- PIC16C622
- PIC16C923
- PIC16C924

1.0 PROGRAMMING THE PIC16C6X/7X/9XX

The PIC16C6X/7X/9XX can be programmed using a serial method. In serial mode the PIC16C6X/7X/9XX can be programmed while in the users system. This allows for increased design flexibility. This programming specification applies to PIC16C6X/7X/9XX devices in all packages.

1.1 Hardware Requirements

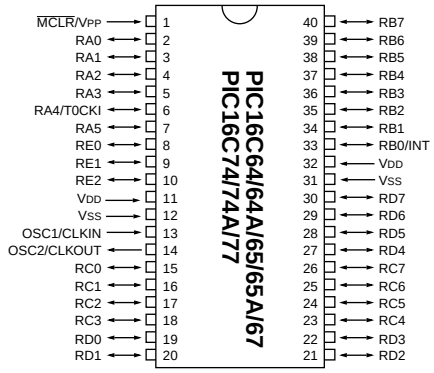
The PIC16C6X/7X/9XX requires two programmable power supplies, one for VDD (2.0V to 6.5V recommended) and one for VPP (12V to 14V). Both supplies should have a minimum resolution of 0.25V.

1.2 Programming Mode

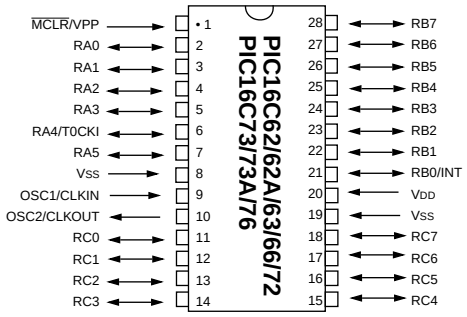
The programming mode for the PIC16C6X/7X/9XX allows programming of user program memory, special locations used for ID, and the configuration word for the PIC16C6X/7X/9XX.

Pin Diagrams

PDIP, Windowed CERDIP



PDIP, SOIC, Windowed CERDIP (300 mil)



Note: Peripheral pinout functions are not shown (see data sheets for full pinout information).

PIN DESCRIPTIONS (DURING PROGRAMMING):
PIC16C61/620/621/622/62/62A/63/64/64A/65/65A/66/67/71/73/73A/74/74A/76/77/710/711/923/924

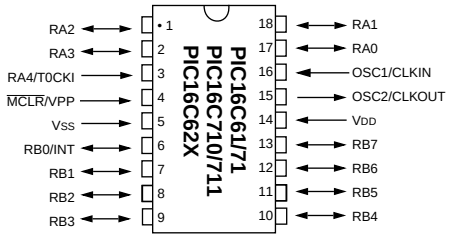
Pin Name	During Programming		
	Pin Name	Pin Type	Pin Description
RB6	CLOCK	I	Clock input
RB7	DATA	I/O	Data input/output
MCLR/VPP	VPP	P	Programming Power
VDD	VDD	P	Power Supply
VSS	VSS	P	Ground

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

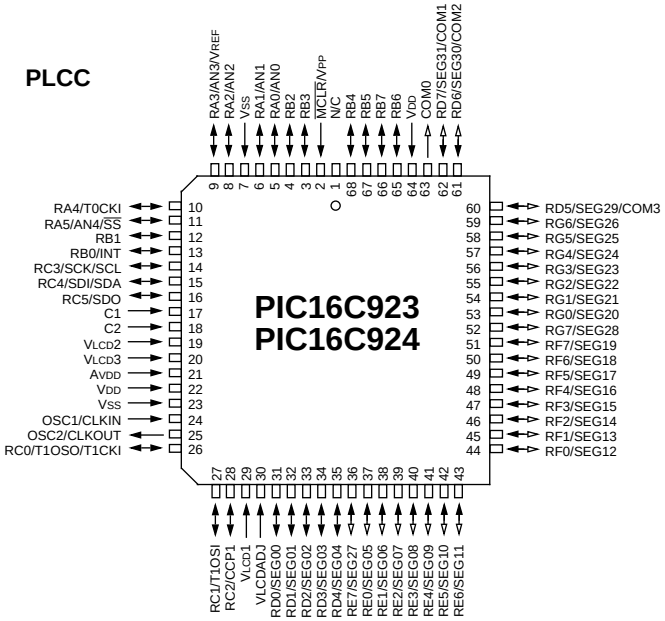
ICSP is a trademark of Microchip Technology Inc.

Pin Diagrams (Con't)

PDIP, SOIC, Windowed CERDIP



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 PIC16C6X/7X/9XX family.

TABLE 2-1: IMPLEMENTATION OF PROGRAM MEMORY IN THE PIC16C6X/7X/9XX

Device	Program Memory Size
PIC16C61	0x000-0x3FF (1K)
PIC16C620	0x000 -0x1FF (0.5K)
PIC16C621	0x000 - 0x3FF (1K)
PIC16C622	0x000 -0x7FF (2K)
PIC16C62/62A	0x000 - 0x7FF (2K)
PIC16C63	0x000 - 0xFFF (4K)
PIC16C64/64A	0x000 - 0x7FF (2K)
PIC16C65/65A	0x000 - 0xFFF (4K)
PIC16C71	0x000 - 0x3FF (1K)
PIC16C710	0x000 - 0x1FF (0.5K)
PIC16C711	0x000 - 0x3FF (1K)
PIC16C72	0x000 - 0x7FF (2K)
PIC16C73/73A	0x000 - 0xFFF (4K)
PIC16C74/74A	0x000 - 0xFFF (4K)
PIC16C66	0x000 - 0x1FFF (8K)
PIC16C67	0x000 - 0x1FFF (8K)
PIC16C76	0x000 - 0x1FFF (8K)
PIC16C77	0x000 - 0x1FFF (8K)
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).

In programming mode, the program memory space extends from 0x0000 to 0x3FFF, with the first half (0x0000-0x1FFF) being user program memory and the second half (0x2000-0x3FFF) being configuration memory. The PC will increment from 0x0000 to 0x1FFF and wrap to 0x000 or 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 user program memory is to reset the part and reenter program/verify mode, as described in Section 2.2.

In the configuration memory space, 0x2000-0x207F or 0x2000-0x20FF are utilized. When in a configuration memory, as in the user memory, the 0x2000-0x2XFF segment is repeatedly accessed as PC exceeds 0x2XFF (see Figure 2-1).

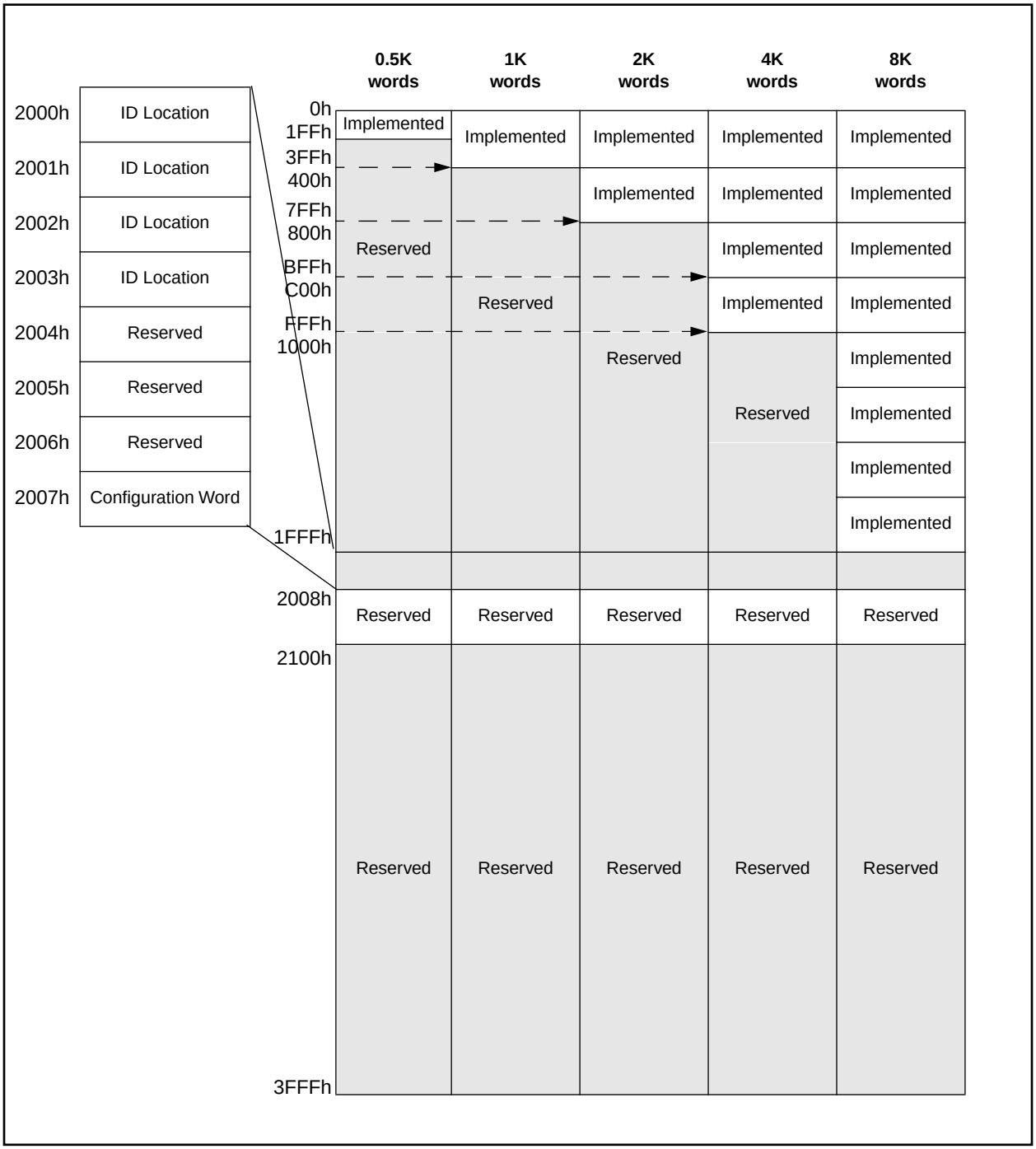
A user may store identification information (ID) in four ID locations. The ID locations are mapped in [0x2000 : 0x2003]. It is recommended that the user use only the four least significant bits of each ID location. In some devices, the ID locations read-out in a scrambled fashion after code protection 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 4.1.

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 $\overline{MCLR}$ pin from V_{IL} to V_{IHH} (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 $\overline{MCLR}$ pin was initially at V_{IL}). This means that all I/O are in the reset state (High impedance inputs).

Note:	The $\overline{MCLR}$ pin should be raised as quickly as possible from V_{IL} to V_{IHH}. this is to ensure that the device does not have the PC incremented while in valid operation range.

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 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 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 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 listed in Table 2-2.

2.2.1.1 LOAD CONFIGURATION

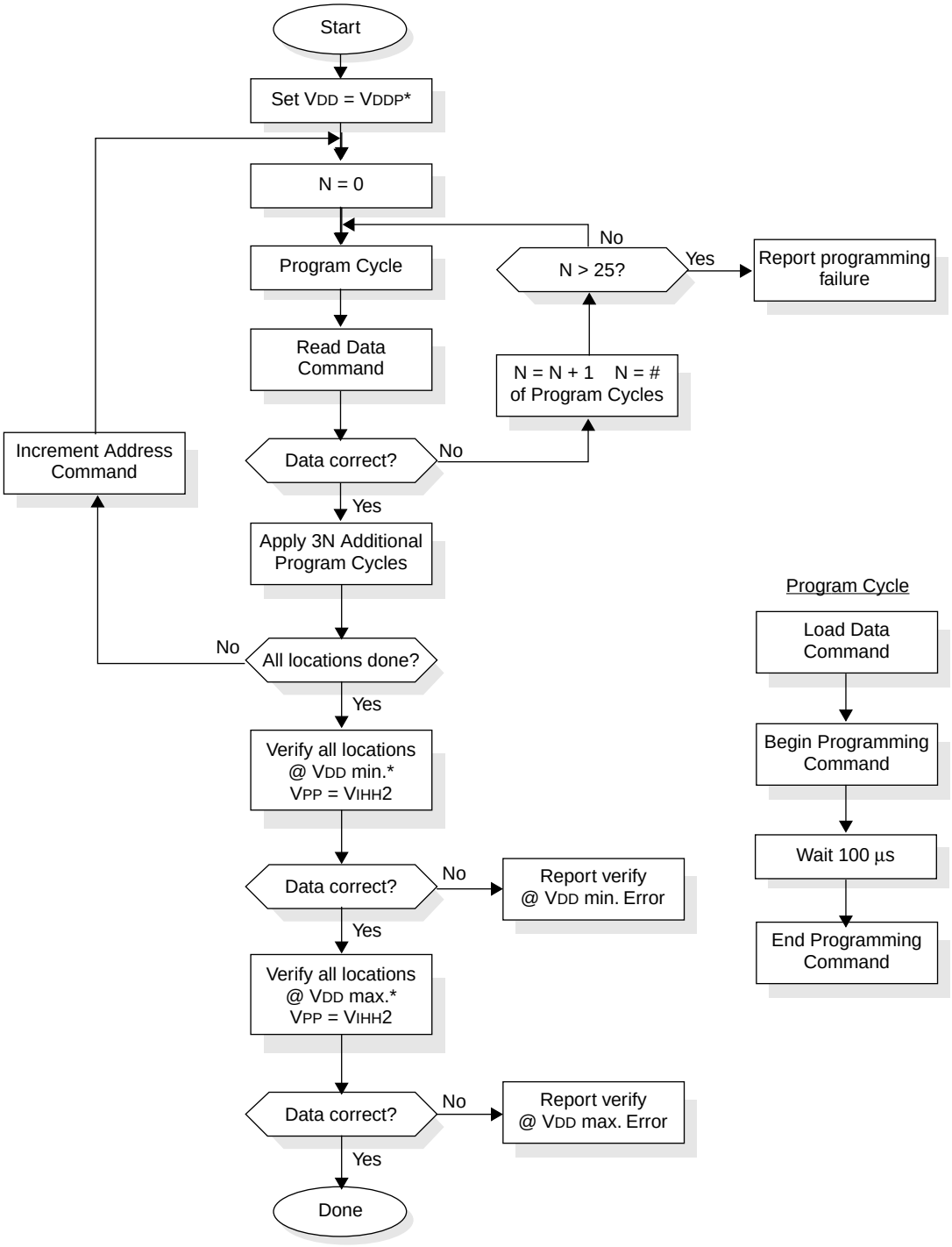
After receiving this command, the program counter (PC) will be set to 0x2000. By then applying 16 cycles to the clock pin, the chip will load 14-bits a “data word” as described above, to be programmed into the configuration memory. A description of the memory mapping schemes for normal operation and 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 $\overline{MCLR}$ low (V_{IL}).

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	

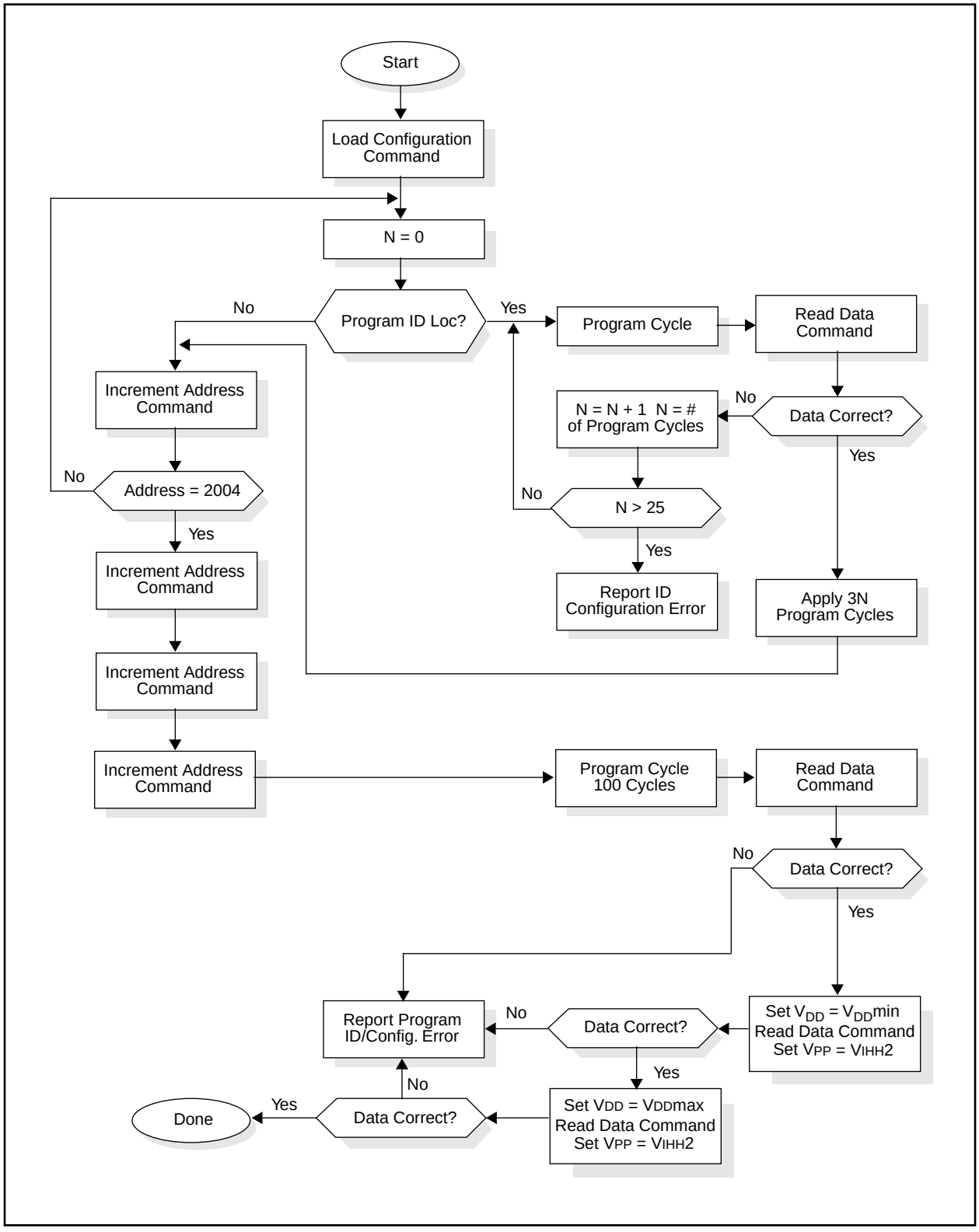
Note: The clock must be disabled during In-Circuit Serial Programming.

FIGURE 2-2: PROGRAM FLOW CHART - PIC16C6X/7X/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 - PIC16C6X/7X/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 5-1.

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 5-2.

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 5-3.

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 100μs 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 V_{DD}

The PIC16C6X/7X/9XX uses an intelligent algorithm. The algorithm calls for program verification at V_{DD}min as well as V_{DD}max. Verification at V_{DD}min guarantees good “erase margin”. Verification at V_{DD}max guarantees good “program margin”.

The actual programming must be done with V_{DD} in the V_{DDP} range (4.75 - 5.25V).

V_{DDP} = V_{CC} range required during programming.

V_{DD} min. = minimum operating V_{DD} spec for the part.

V_{DD}max = maximum operating V_{DD} spec for the part.

Programmers must verify the PIC16C6X/7X/9XX at its specified V_{DD}max and V_{DD}min levels. Since Microchip may introduce future versions of the PIC16C6X/7X/9XX with a broader V_{DD} range, it is best that these levels are user selectable (defaults are ok).

Note:	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 PIC16C6X/7X/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 provides an overview of configuration bits.

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	PWRTE	WDTE	FOSC1	FOSC0
PIC16C62/64/65/73/74	—	—	—	—	—	—	—	0	CP1	CP0	PWRTE	WDTE	FOSC1	FOSC0
PIC16C710/711	CP0	CP0	CP0	CP0	CP0	CP0	CP0	BODEN	CP0	CP0	$\overline{\text{PWRTE}}$	WDTE	FOSC1	FOSC0
PIC16C62A/CR62/63/64A/CR64/65A/66/67/72/73A/74A/76/77/620/621/622	CP1	CP0	CP1	CP0	CP1	CP0	—	BODEN	CP1	CP0	$\overline{\text{PWRTE}}$	WDTE	FOSC1	FOSC0
PIC16C9XX	CP1	CP0	CP1	CP0	CP1	CP0	—	—	CP1	CP0	$\overline{\text{PWRTE}}$	WDTE	FOSC1	FOSC0

bit 6: **Reserved**, '–' write as '1' for PIC16C6X/7X/9XX

bit 5-4: **CP1:CP0**, Code Protect

Device	CP1	CP0	Code Protection
PIC16C622 PIC16C62/62A/63 PIC16C64/64A PIC16C65/65A/66/67 PIC16C72 PIC16C73/73A PIC16C74/74A/76/77 PIC16C9XX	0	0	All memory protected
	0	1	Upper 3/4 memory protected
	1	0	Upper 1/2 memory protected
	1	1	Code protection off
PIC16C61/71 PIC16C710/711	—	0	On
	—	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
	0	1	Upper 1/2 memory protected
	1	0	Do not use
	1	1	Code protection off

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

bit 4: **$\overline{\text{PWRTE}}$/PWRTE**, Power Up Timer Enable Bit
PIC16C61/62/64/65/71/73/74:
1 = Power up timer enabled
0 = Power up timer disabled
PIC16C620/621/622/62A/63/65A/66/67/72/73A/74A/76/77:
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

4.0 CODE PROTECTION

The program code written into the EPROM can be protected by writing to the CP0 & CP1 bits of the configuration word.

In PIC16C61/71 it is still possible to program locations 0x000 through 0x03F, after code protection. For all other devices, writing to all protected memory is disabled.

4.1 Programming Locations 0x0000 to 0x03F after Code Protection

For PIC16C61/71 devices, once code protection is enabled, all program memory locations read out in a scrambled fashion. The ID locations and the configuration word also read out in a scrambled fashion. Further programming is disabled for locations 0x040 and above. It is possible to program the ID locations and the configuration word.

For PIC16C61/71 devices, program memory locations 0x000 through 0x03F are essentially unprotected, i.e., these locations can be further programmed after code protection is enabled. However, since the data reads out in a scrambled fashion, to correctly overprogram these locations, the programmer must program seven bits at a time. For example, to program 0x3AD2 ("11 1010 1101 0010") in a blank location, first program the location with "11 1111 1101 0010" and verify scrambled output to be "xx xxxx x101 0010". Next, program the location with "11 1010 1101 0010" and verify scrambled output to be "xx xxxx x101 1000".

For all other PIC16C6X/7X/9XX devices, once code protection is enabled, all protected segments read '0's (or "garbage values") and are prevented from further programming. All unprotected segments, including ID locations and configuration word, read normally. These locations can be programmed.

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.

TABLE 4-1: CONFIGURATION WORD

PIC16C61

To code protect:

- Protect all memory `1111111110xxxx`

Program Memory Segment	R/W in Protected Mode	R/W in Unprotected Mode
Configuration Word (0x2007)	Read Scrambled, Write Enabled	Read Unscrambled, Write Enabled
Unprotected memory segment	Read Scrambled, Write Enabled	Read Unscrambled, Write Enabled
Protected memory segment	Read Scrambled, Write Disabled	Read Unscrambled, Write Enabled
ID Locations (0x2000 : 0x2003)	Read Scrambled, Write Enabled	Read Unscrambled, Write Enabled

PIC16C620

To code protect:

- Protect all memory `0000001x00xxxx`
- No code protection `1111111x11xxxx`

Program Memory Segment	R/W in Protected Mode	R/W in Unprotected Mode
Configuration Word (0x2007)	Read Unscrambled, Write Enabled	Read Unscrambled, Write Enabled
Unprotected memory segment	Read Unscrambled, Write Enabled	Read Unscrambled, Write Enabled
Protected memory segment	Read All 0's, Write Disabled	Read Unscrambled, Write Enabled
ID Locations (0x2000 : 0x2003)	Read Unscrambled, Write Enabled	Read Unscrambled, Write Enabled

PIC16C621

To code protect:

- Protect all memory `0000001X00XXXX`
- Protect upper 1/2 memory `0101011X01XXXX`
- No code protection `1111111X11XXXX`

Program Memory Segment	R/W in Protected Mode	R/W in Unprotected Mode
Configuration Word (0x2007)	Read Unscrambled, Write Enabled	Read Unscrambled, Write Enabled
Unprotected memory segment	Read Unscrambled, Write Enabled	Read Unscrambled, Write Enabled
Protected memory segment	Read All 0's, Write Disabled	Read Unscrambled, Write Enabled
ID Locations (0x2000 : 0x2003)	Read Unscrambled, Write Enabled	Read Unscrambled, Write Enabled

PIC16C622

To code protect:

- Protect all memory `0000001X00XXXX`
- Protect upper 3/4 memory `0101011X01XXXX`
- Protect upper 1/2 memory `1010101X10XXXX`
- No code protection `1111111X11XXXX`

Program Memory Segment	R/W in Protected Mode	R/W in Unprotected Mode
Configuration Word (0x2007)	Read Unscrambled, Write Enabled	Read Unscrambled, Write Enabled
Unprotected memory segment	Read Unscrambled, Write Enabled	Read Unscrambled, Write Enabled
Protected memory segment	Read All 0's, Write Disabled	Read Unscrambled, Write Enabled
ID Locations (0x2000 : 0x2003)	Read Unscrambled, Write Enabled	Read Unscrambled, Write Enabled

PIC16C62

To code protect:

- Protect all memory `1111111000XXXX`
- Protect upper 3/4 memory `1111111001XXXX`
- Protect upper 1/2 memory `1111111010XXXX`
- No code protection `1111111011XXXX`

Program Memory Segment	R/W in Protected Mode	R/W in Unprotected Mode
Configuration Word (0x2007)	Read Unscrambled, Write Enabled	Read Unscrambled, Write Enabled
Unprotected memory segment	Read Unscrambled, Write Enabled	Read Unscrambled, Write Enabled
Protected memory segment	Read Scrambled, Write Disabled	Read Unscrambled, Write Enabled
ID Locations (0x2000 : 0x2003)	Read Unscrambled, Write Enabled	Read Unscrambled, Write Enabled

PIC16C62A

To code protect:

- Protect all memory `0000001X00XXXX`
- Protect upper 3/4 memory `0101011X01XXXX`
- Protect upper 1/2 memory `1010101X10XXXX`
- No code protection `1111111X11XXXX`

Program Memory Segment	R/W in Protected Mode	R/W in Unprotected Mode
Configuration Word (0x2007)	Read Unscrambled, Write Enabled	Read Unscrambled, Write Enabled
Unprotected memory segment	Read Unscrambled, Write Enabled	Read Unscrambled, Write Enabled
Protected memory segment	Read All 0's, Write Disabled	Read Unscrambled, Write Enabled
ID Locations (0x2000 : 0x2003)	Read Unscrambled, Write Enabled	Read Unscrambled, Write Enabled

PIC16C63

To code protect:

- Protect all memory `0000001x00xxxx`
- Protect upper 3/4 memory `0101011x01xxxx`
- Protect upper 1/2 memory `1010101x10xxxx`
- No code protection `1111111x11xxxx`

Program Memory Segment	R/W in Protected Mode	R/W in Unprotected Mode
Configuration Word (0x2007)	Read Unscrambled, Write Enabled	Read Unscrambled, Write Enabled
Unprotected memory segment	Read Unscrambled, Write Enabled	Read Unscrambled, Write Enabled
Protected memory segment	Read All 0's, Write Disabled	Read Unscrambled, Write Enabled
ID Locations (0x2000 : 0x2003)	Read Unscrambled, Write Enabled	Read Unscrambled, Write Enabled

PIC16C64

To code protect:

- Protect all memory `1111111000xxxx`
- Protect upper 3/4 memory `1111111001xxxx`
- Protect upper 1/2 memory `1111111010xxxx`
- No code protection `1111111011xxxx`

Program Memory Segment	R/W in Protected Mode	R/W in Unprotected Mode
Configuration Word (0x2007)	Read Unscrambled, Write Enabled	Read Unscrambled, Write Enabled
Unprotected memory segment	Read Unscrambled, Write Enabled	Read Unscrambled, Write Enabled
Protected memory segment	Read Scrambled, Write Disabled	Read Unscrambled, Write Enabled
ID Locations (0x2000 : 0x2003)	Read Unscrambled, Write Enabled	Read Unscrambled, Write Enabled

PIC16C64A

To code protect:

- Protect all memory `0000001x00xxxx`
- Protect upper 3/4 memory `0101011x01xxxx`
- Protect upper 1/2 memory `1010101x10xxxx`
- No code protection `1111111x11xxxx`

Program Memory Segment	R/W in Protected Mode	R/W in Unprotected Mode
Configuration Word (0x2007)	Read Unscrambled, Write Enabled	Read Unscrambled, Write Enabled
Unprotected memory segment	Read Unscrambled, Write Enabled	Read Unscrambled, Write Enabled
Protected memory segment	Read All 0's, Write Disabled	Read Unscrambled, Write Enabled
ID Locations (0x2000 : 0x2003)	Read Unscrambled, Write Enabled	Read Unscrambled, Write Enabled

PIC16C65

To code protect:

- Protect all memory `1111111000xxxx`
- Protect upper 3/4 memory `1111111001xxxx`
- Protect upper 1/2 memory `1111111010xxxx`
- No code protection `1111111011xxxx`

Program Memory Segment	R/W in Protected Mode	R/W in Unprotected Mode
Configuration Word (0x2007)	Read Unscrambled, Write Enabled	Read Unscrambled, Write Enabled
Unprotected memory segment	Read Unscrambled, Write Enabled	Read Unscrambled, Write Enabled
Protected memory segment	Read Scrambled, Write Disabled	Read Unscrambled, Write Enabled
ID Locations (0x2000 : 0x2003)	Read Unscrambled, Write Enabled	Read Unscrambled, Write Enabled

PIC16C65A

To code protect:

- Protect all memory `0000001x00xxxx`
- Protect upper 3/4 memory `0101011x01xxxx`
- Protect upper 1/2 memory `1010101x10xxxx`
- No code protection `1111111x11xxxx`

Program Memory Segment	R/W in Protected Mode	R/W in Unprotected Mode
Configuration Word (0x2007)	Read Unscrambled, Write Enabled	Read Unscrambled, Write Enabled
Unprotected memory segment	Read Unscrambled, Write Enabled	Read Unscrambled, Write Enabled
Protected memory segment	Read All 0's, Write Disabled	Read Unscrambled, Write Enabled
ID Locations (0x2000 : 0x2003)	Read Unscrambled, Write Enabled	Read Unscrambled, Write Enabled

PIC16C66

To code protect:

- Protect all memory `0000001x00xxxx`
- Protect upper 3/4 memory `0101011x01xxxx`
- Protect upper 1/2 memory `1010101x10xxxx`
- No code protection `1111111x11xxxx`

Program Memory Segment	R/W in Protected Mode	R/W in Unprotected Mode
Configuration Word (0x2007)	Read Unscrambled, Write Enabled	Read Unscrambled, Write Enabled
Unprotected memory segment	Read Unscrambled, Write Enabled	Read Unscrambled, Write Enabled
Protected memory segment	Read All 0's, Write Disabled	Read Unscrambled, Write Enabled
ID Locations (0x2000 : 0x2003)	Read Unscrambled, Write Enabled	Read Unscrambled, Write Enabled

PIC16C67

To code protect:

- Protect all memory `0000001x00xxxx`
- Protect upper 3/4 memory `0101011x01xxxx`
- Protect upper 1/2 memory `1010101x10xxxx`
- No code protection `1111111x11xxxx`

Program Memory Segment	R/W in Protected Mode	R/W in Unprotected Mode
Configuration Word (0x2007)	Read Unscrambled, Write Enabled	Read Unscrambled, Write Enabled
Unprotected memory segment	Read Unscrambled, Write Enabled	Read Unscrambled, Write Enabled
Protected memory segment	Read All 0's, Write Disabled	Read Unscrambled, Write Enabled
ID Locations (0x2000 : 0x2003)	Read Unscrambled, Write Enabled	Read Unscrambled, Write Enabled

PIC16C710

To code protect:

- Protect all memory `0000000x00xxxx`

Program Memory Segment	R/W in Protected Mode	R/W in Unprotected Mode
Configuration Word (0x2007)	Read Unscrambled, Write Enabled	Read Unscrambled, Write Enabled
Unprotected memory segment	Read Unscrambled, Write Enabled	Read Unscrambled, Write Enabled
Protected memory segment	Read All 0's, Write Disabled	Read Unscrambled, Write Enabled
ID Locations (0x2000 : 0x2003)	Read Unscrambled, Write Enabled	Read Unscrambled, Write Enabled

PIC16C71

To code protect:

- Protect all memory `1111111110xxxx`

Program Memory Segment	R/W in Protected Mode	R/W in Unprotected Mode
Configuration Word (0x2007)	Read Scrambled, Write Enabled	Read Unscrambled, Write Enabled
Unprotected memory segment	Read Scrambled, Write Enabled	Read Unscrambled, Write Enabled
Protected memory segment	Read Scrambled, Write Disabled	Read Unscrambled, Write Enabled
ID Locations (0x2000 : 0x2003)	Read Scrambled, Write Enabled	Read Unscrambled, Write Enabled

PIC16C711

To code protect:

- Protect all memory `0000000x00xxxx`

Program Memory Segment	R/W in Protected Mode	R/W in Unprotected Mode
Configuration Word (0x2007)	Read Unscrambled, Write Enabled	Read Unscrambled, Write Enabled
Unprotected memory segment	Read Unscrambled, Write Enabled	Read Unscrambled, Write Enabled
Protected memory segment	Read All 0's, Write Disabled	Read Unscrambled, Write Enabled
ID Locations (0x2000 : 0x2003)	Read Unscrambled, Write Enabled	Read Unscrambled, Write Enabled

PIC16C72

To code protect:

- Protect all memory `0000001x00xxxx`
- Protect upper 3/4 memory `0101011x01xxxx`
- Protect upper 1/2 memory `1010101x10xxxx`
- No code protection `1111111x11xxxx`

Program Memory Segment	R/W in Protected Mode	R/W in Unprotected Mode
Configuration Word (0x2007)	Read Unscrambled, Write Enabled	Read Unscrambled, Write Enabled
Unprotected memory segment	Read Unscrambled, Write Enabled	Read Unscrambled, Write Enabled
Protected memory segment	Read All 0's, Write Disabled	Read Unscrambled, Write Enabled
ID Locations (0x2000 : 0x2003)	Read Unscrambled, Write Enabled	Read Unscrambled, Write Enabled

PIC16C73

To code protect:

- Protect all memory `1111111000xxxx`
- Protect upper 3/4 memory `1111111001xxxx`
- Protect upper 1/2 memory `1111111010xxxx`
- No code protection `1111111011xxxx`

Program Memory Segment	R/W in Protected Mode	R/W in Unprotected Mode
Configuration Word (0x2007)	Read Unscrambled, Write Enabled	Read Unscrambled, Write Enabled
Unprotected memory segment	Read Unscrambled, Write Enabled	Read Unscrambled, Write Enabled
Protected memory segment	Read Scrambled, Write Disabled	Read Unscrambled, Write Enabled
ID Locations (0x2000 : 0x2003)	Read Unscrambled, Write Enabled	Read Unscrambled, Write Enabled

PIC16C73A

To code protect:

- Protect all memory `0000001x00xxxx`
- Protect upper 3/4 memory `0101011x01xxxx`
- Protect upper 1/2 memory `1010101x10xxxx`
- No code protection `1111111x11xxxx`

Program Memory Segment	R/W in Protected Mode	R/W in Unprotected Mode
Configuration Word (0x2007)	Read Unscrambled, Write Enabled	Read Unscrambled, Write Enabled
Unprotected memory segment	Read Unscrambled, Write Enabled	Read Unscrambled, Write Enabled
Protected memory segment	Read All 0's, Write Disabled	Read Unscrambled, Write Enabled
ID Locations (0x2000 : 0x2003)	Read Unscrambled, Write Enabled	Read Unscrambled, Write Enabled

PIC16C74

To code protect:

- Protect all memory `1111111000xxxx`
- Protect upper 3/4 memory `1111111001xxxx`
- Protect upper 1/2 memory `1111111010xxxx`
- No code protection `1111111011xxxx`

Program Memory Segment	R/W in Protected Mode	R/W in Unprotected Mode
Configuration Word (0x2007)	Read Unscrambled, Write Enabled	Read Unscrambled, Write Enabled
Unprotected memory segment	Read Unscrambled, Write Enabled	Read Unscrambled, Write Enabled
Protected memory segment	Read Scrambled, Write Disabled	Read Unscrambled, Write Enabled
ID Locations (0x2000 : 0x2003)	Read Unscrambled, Write Enabled	Read Unscrambled, Write Enabled

PIC16C74A

To code protect:

- Protect all memory `0000001x00xxxx`
- Protect upper 3/4 memory `0101011x01xxxx`
- Protect upper 1/2 memory `1010101x10xxxx`
- No code protection `1111111x11xxxx`

Program Memory Segment	R/W in Protected Mode	R/W in Unprotected Mode
Configuration Word (0x2007)	Read Unscrambled, Write Enabled	Read Unscrambled, Write Enabled
Unprotected memory segment	Read Unscrambled, Write Enabled	Read Unscrambled, Write Enabled
Protected memory segment	Read All 0's, Write Disabled	Read Unscrambled, Write Enabled
ID Locations (0x2000 : 0x2003)	Read Unscrambled, Write Enabled	Read Unscrambled, Write Enabled

PIC16C76

To code protect:

- Protect all memory `0000001x00xxxx`
- Protect upper 3/4 memory `0101011x01xxxx`
- Protect upper 1/2 memory `1010101x10xxxx`
- No code protection `1111111x11xxxx`

Program Memory Segment	R/W in Protected Mode	R/W in Unprotected Mode
Configuration Word (0x2007)	Read Unscrambled, Write Enabled	Read Unscrambled, Write Enabled
Unprotected memory segment	Read Unscrambled, Write Enabled	Read Unscrambled, Write Enabled
Protected memory segment	Read All 0's, Write Disabled	Read Unscrambled, Write Enabled
ID Locations (0x2000 : 0x2003)	Read Unscrambled, Write Enabled	Read Unscrambled, Write Enabled

PIC16C77

To code protect:

- Protect all memory `0000001x00xxxx`
- Protect upper 3/4 memory `0101011x01xxxx`
- Protect upper 1/2 memory `1010101x10xxxx`
- No code protection `1111111x11xxxx`

Program Memory Segment	R/W in Protected Mode	R/W in Unprotected Mode
Configuration Word (0x2007)	Read Unscrambled, Write Enabled	Read Unscrambled, Write Enabled
Unprotected memory segment	Read Unscrambled, Write Enabled	Read Unscrambled, Write Enabled
Protected memory segment	Read All 0's, Write Disabled	Read Unscrambled, Write Enabled
ID Locations (0x2000 : 0x2003)	Read Unscrambled, Write Enabled	Read Unscrambled, Write Enabled

PIC16C923

To code protect:

- Protect all memory `0000001x00xxxx`
- Protect upper 3/4 memory `0101011x01xxxx`
- Protect upper 1/2 memory `1010101x10xxxx`
- No code protection `1111111x11xxxx`

Program Memory Segment	R/W in Protected Mode	R/W in Unprotected Mode
Configuration Word (0x2007)	Read Unscrambled, Write Enabled	Read Unscrambled, Write Enabled
Unprotected memory segment	Read Unscrambled, Write Enabled	Read Unscrambled, Write Enabled
Protected memory segment	Read All 0's, Write Disabled	Read Unscrambled, Write Enabled
ID Locations (0x2000 : 0x2003)	Read Unscrambled, Write Enabled	Read Unscrambled, Write Enabled

PIC16C924

To code protect:

- Protect all memory `0000001x00xxxx`
- Protect upper 3/4 memory `0101011x01xxxx`
- Protect upper 1/2 memory `1010101x10xxxx`
- No code protection `1111111x11xxxx`

Program Memory Segment	R/W in Protected Mode	R/W in Unprotected Mode
Configuration Word (0x2007)	Read Unscrambled, Write Enabled	Read Unscrambled, Write Enabled
Unprotected memory segment	Read Unscrambled, Write Enabled	Read Unscrambled, Write Enabled
Protected memory segment	Read All 0's, Write Disabled	Read Unscrambled, Write Enabled
ID Locations (0x2000 : 0x2003)	Read Unscrambled, Write Enabled	Read Unscrambled, Write Enabled

Legend: X = Don't care

4.3 Checksum

4.3.1 CHECKSUM CALCULATIONS

Checksum is calculated by reading the contents of the PIC16C6X/7X/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 PIC16C8X devices is shown in Table 4-2.

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 4-2: CHECKSUM COMPUTATION

Device	Code Protect	Checksum*	Blank Value	0x25E6 at 0 and max address
PIC16C61	OFF	SUM[0x000:0x3FF] + CFGW & 0x001F + 0x3FE0	0x3BFF	0x07CD
	ON	SUM_XNOR7[0x000:0x3FF] + (CFGW & 0x001F | 0x0060)	0xFC6F	0xFC15
PIC16C620	OFF	SUM[0x000:0x1FF] + CFGW & 0x3F7F	0x3D7F	0x094D
	ON	SUM_ID + CFGW & 0x3F7F	0x3DCE	0x099C
PIC16C621	OFF	SUM[0x000:0x3FF] + CFGW & 0x3F7F	0x3B7F	0x074D
	1/2	SUM[0x000:0x1FF] + CFGW & 0x3F7F + SUM_ID	0x4EDE	0x0093
	ALL	CFGW & 0x3F7F + SUM_ID	0x3BCE	0x079C
PIC16C622	OFF	SUM[0x000:0x7FF] + CFGW & 0x3F7F	0x377F	0x034D
	1/2	SUM[0x000:0x3FF] + CFGW & 0x3F7F + SUM_ID	0x5DEE	0x0FA3
	3/4	SUM[0x000:0x1FF] + CFGW & 0x3F7F + SUM_ID	0x4ADE	0xFC93
	ALL	CFGW & 0x3F7F + SUM_ID	0x37CE	0x039C
PIC16C62	OFF	SUM[0x000:0x7FF] + CFGW & 0x003F + 0x3F80	0x37BF	0x038D
	1/2	SUM[0x000:0x3FF] + SUM_XNOR7[0x400:0x7FF] + CFGW & 0x003F + 0x3F80	0x37AF	0x1D69
	3/4	SUM[0x000:0x1FF] + SUM_XNOR7[0x200:0x7FF] + CFGW & 0x003F + 0x3F80	0x379F	0x1D59
	ALL	SUM_XNOR7[0x000:0x7FF] + CFGW & 0x003F + 0x3F80	0x378F	0x3735
PIC16C62A	OFF	SUM[0x000:0x7FF] + CFGW & 0x3F7F	0x377F	0x034D
	1/2	SUM[0x000:0x3FF] + CFGW & 0x3F7F + SUM_ID	0x5DEE	0x0FA3
	3/4	SUM[0x000:0x1FF] + CFGW & 0x3F7F + SUM_ID	0x4ADE	0xFC93
	ALL	CFGW & 0x3F7F + SUM_ID	0x37CE	0x039C
PIC16C63	OFF	SUM[0x000:0xFFF] + CFGW & 0x3F7F	0x2F7F	0xFB4D
	1/2	SUM[0x000:0x7FF] + CFGW & 0x3F7F + SUM_ID	0x51EE	0x03A3
	3/4	SUM[0x000:0x3FF] + CFGW & 0x3F7F + SUM_ID	0x40DE	0xF293
	ALL	CFGW & 0x3F7F + SUM_ID	0x2FCE	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, location_a = 0x123 and location_b = 0x456, then SUM_XNOR7 [location_a : location_b] = 0x001F.
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

TABLE 4-2: CHECKSUM COMPUTATION (CONTINUED)

Device	Code Protect	Checksum*	Blank Value	0x25E6 at 0 and max address
PIC16C64	OFF	SUM[0x000:0x7FF] + CFGW & 0x003F + 0x3F80	0x37BF	0x038D
	1/2	SUM[0x000:0x3FF] + SUM_XNOR7[0x400:0x7FF] + CFGW & 0x003F + 0x3F80	0x37AF	0x1D69
	3/4	SUM[0x000:0x1FF] + SUM_XNOR7[0x200:0x7FF] + CFGW & 0x003F + 0x3F80	0x379F	0x1D59
	ALL	SUM_XNOR7[0x000:0x7FF] + CFGW & 0x003F + 0x3F80	0x378F	0x3735
PIC16C64A	OFF	SUM[0x000:0x7FF] + CFGW & 0x3F7F	0x377F	0x034D
	1/2	SUM[0x000:0x3FF] + CFGW & 0x3F7F + SUM_ID	0x5DEE	0x0FA3
	3/4	SUM[0x000:0x1FF] + CFGW & 0x3F7F + SUM_ID	0x4ADE	0xFC93
	ALL	CFGW & 0x3F7F + SUM_ID	0x37CE	0x039C
PIC16C65	OFF	SUM[0x000:0xFFF] + CFGW & 0x003F + 0x3F80	0x2FBF	0xFB8D
	1/2	SUM[0x000:0x7FF] + SUM_XNOR7[0x800:FFF] + CFGW & 0x003F + 0x3F80	0x2FAF	0x1569
	3/4	SUM[0x000:0x3FF] + SUM_XNOR7[0x400:FFF] + CFGW & 0x003F + 0x3F80	0x2F9F	0x1559
	ALL	SUM_XNOR7[0x000:0xFFF] + CFGW & 0x003F + 0x3F80	0x2F8F	0x2F35
PIC16C65A	OFF	SUM[0x000:0xFFF] + CFGW & 0x3F7F	0x2F7F	0xFB4D
	1/2	SUM[0x000:0x7FF] + CFGW & 0x3F7F + SUM_ID	0x51EE	0x03A3
	3/4	SUM[0x000:0x3FF] + CFGW & 0x3F7F + SUM_ID	0x40DE	0xF293
	ALL	CFGW & 0x3F7F + SUM_ID	0x2FCE	0xFB9C
PIC16C66	OFF	SUM[0x000:0x1FFF] + CFGW & 0x3F7F	0x1F7F	0xEB4D
	1/2	SUM[0x000:0xFFF] + CFGW & 0x3F7F + SUM_ID	0x39EE	0xEBA3
	3/4	SUM[0x000:0x7FF] + CFGW & 0x3F7F + SUM_ID	0x2CDE	0xDE93
	ALL	CFGW & 0x3F7F + SUM_ID	0x1FCE	0xEB9C
PIC16C67	OFF	SUM[0x000:0x1FFF] + CFGW & 0x3F7F	0x1F7F	0xEB4D
	1/2	SUM[0x000:0xFFF] + CFGW & 0x3F7F + SUM_ID	0x39EE	0xEBA3
	3/4	SUM[0x000:0x7FF] + CFGW & 0x3F7F + SUM_ID	0x2CDE	0xDE93
	ALL	CFGW & 0x3F7F + SUM_ID	0x1FCE	0xEB9C
PIC16C710	OFF	SUM[0x000:0x1FF] + CFGW & 0x3FFF	0x3DFF	0x09CD
	ON	SUM[0x00:0x3F] + CFGW & 0x3FFF + SUM_ID	0x3E0E	0xEFC3
PIC16C71	OFF	SUM[0x000:0x3FF] + CFGW & 0x001F + 0x3FE0	0x3BFF	0x07CD
	ON	SUM_XNOR7[0x000:0x3FF] + (CFGW & 0x001F | 0x0060)	0xFC6F	0xFC15
PIC16C711	OFF	SUM[0x000:0x3FF] + CFGW & 0x3FFF	0x3BFF	0x07CD
	ON	SUM[0x00:0x3F] + CFGW & 0x3FFF + SUM_ID	0x3C0E	0xEDC3
PIC16C72	OFF	SUM[0x000:0x7FF] + CFGW & 0x3F7F	0x377F	0x034D
	1/2	SUM[0x000:0x3FF] + CFGW & 0x3F7F + SUM_ID	0x5DEE	0x0FA3
	3/4	SUM[0x000:0x1FF] + CFGW & 0x3F7F + SUM_ID	0x4ADE	0xFC93
	ALL	CFGW & 0x3F7F + SUM_ID	0x37CE	0x039C
PIC16C73	OFF	SUM[0x000:0xFFF] + CFGW & 0x003F + 0x3F80	0x2FBF	0xFB8D
	1/2	SUM[0x000:0x7FF] + SUM_XNOR7[0x800:FFF] + CFGW & 0x003F + 0x3F80	0x2FAF	0x1569
	3/4	SUM[0x000:0x3FF] + SUM_XNOR7[0x400:FFF] + CFGW & 0x003F + 0x3F80	0x2F9F	0x1559
	ALL	SUM_XNOR7[0x000:0xFFF] + CFGW & 0x003F + 0x3F80	0x2F8F	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, location_a = 0x123 and location_b = 0x456, then SUM_XNOR7 [location_a : location_b] = 0x001F.

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

TABLE 4-2: CHECKSUM COMPUTATION (CONTINUED)

Device	Code Protect	Checksum*	Blank Value	0x25E6 at 0 and max address
PIC16C73A	OFF	SUM[0x000:0xFFF] + CFGW & 0x3F7F	0x2F7F	0xFB4D
	1/2	SUM[0x000:0x7FF] + CFGW & 0x3F7F + SUM_ID	0x51EE	0x03A3
	3/4	SUM[0x000:0x3FF] + CFGW & 0x3F7F + SUM_ID	0x40DE	0xF293
	ALL	CFGW & 0x3F7F + SUM_ID	0x2FCE	0xFB9C
PIC16C74	OFF	SUM[0x000:0xFFF] + CFGW & 0x003F + 0x3F80	0x2FBF	0xFB8D
	1/2	SUM[0x000:0x7FF] + SUM_XNOR7[0x800:FFF] + CFGW & 0x003F + 0x3F80	0x2FAF	0x1569
	3/4	SUM[0x000:0x3FF] + SUM_XNOR7[0x400:FFF] + CFGW & 0x003F + 0x3F80	0x2F9F	0x1559
	ALL	SUM_XNOR7[0x000:0xFFF] + CFGW & 0x003F + 0x3F80	0x2F8F	0x2F35
PIC16C74A	OFF	SUM[0x000:0xFFF] + CFGW & 0x3F7F	0x2F7F	0xFB4D
	1/2	SUM[0x000:0x7FF] + CFGW & 0x3F7F + SUM_ID	0x51EE	0x03A3
	3/4	SUM[0x000:0x3FF] + CFGW & 0x3F7F + SUM_ID	0x40DE	0xF293
	ALL	CFGW & 0x3F7F + SUM_ID	0x2FCE	0xFB9C
PIC16C76	OFF	SUM[0x000:0x1FFF] + CFGW & 0x3F7F	0x1F7F	0xEB4D
	1/2	SUM[0x000:0xFFF] + CFGW & 0x3F7F + SUM_ID	0x39EE	0xEBA3
	3/4	SUM[0x000:0x7FF] + CFGW & 0x3F7F + SUM_ID	0x2CDE	0xDE93
	ALL	CFGW & 0x3F7F + SUM_ID	0x1FCE	0xEB9C
PIC16C77	OFF	SUM[0x000:0x1FFF] + CFGW & 0x3F7F	0x1F7F	0xEB4D
	1/2	SUM[0x000:0xFFF] + CFGW & 0x3F7F + SUM_ID	0x39EE	0xEBA3
	3/4	SUM[0x000:0x7FF] + CFGW & 0x3F7F + SUM_ID	0x2CDE	0xDE93
	ALL	CFGW & 0x3F7F + SUM_ID	0x1FCE	0xEB9C
PIC16C923	OFF	SUM[0x000:0xFFF] + CFGW & 0x3F3F	0x2F3F	0xFB0D
	1/2	SUM[0x000:0x7FF] + CFGW & 0x3F3F + SUM_ID	0x516E	0x0323
	3/4	SUM[0x000:0x3FF] + CFGW & 0x3F3F + SUM_ID	0x405E	0xF213
	ALL	CFGW & 0x3F3F + SUM_ID	0x2F4E	0xFB1C
PIC16C924	OFF	SUM[0x000:0xFFF] + CFGW & 0x3F3F	0x2F3F	0xFB0D
	1/2	SUM[0x000:0x7FF] + CFGW & 0x3F3F + SUM_ID	0x516E	0x0323
	3/4	SUM[0x000:0x3FF] + CFGW & 0x3F3F + SUM_ID	0x405E	0xF213
	ALL	CFGW & 0x3F3F + SUM_ID	0x2F4E	0xFB1C

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, location_a = 0x123 and location_b = 0x456, then SUM_XNOR7 [location_a : location_b] = 0x001F.

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

5.0 PROGRAM/VERIFY MODE ELECTRICAL CHARACTERISTICS

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

Standard Operating Conditions Operating Temperature: +10°C ≤ TA ≤ +40°C, unless otherwise stated, (20°C recommended) Operating Voltage: 4.5V ≤ VDD ≤ 5.5V, unless otherwise stated.							
Parameter No.	**Sym.**	**Characteristic**	**Min.**	**Typ.**	**Max.**	**Units**	**Conditions**
General							
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 $\overline{MCLR}$/VPP during programming	12.75	–	13.25	V	Note 2
PD5	VIHH2	Voltage on $\overline{MCLR}$/VPP during verify	VDD + 4.0	–	13.5	–	
PD6	IPP	Programming supply current (from VPP)	–	–	50	mA	
PD9	VIH1	(RB6, RB7) input high level	0.8 VDD	–	–	V	Schmitt Trigger input
PD8	VIL1	(RB6, RB7) input low level	0.2 VDD	–	–	V	Schmitt Trigger input
Serial Program Verify							
P1	TR	$\overline{MCLR}$/VPP rise time (VSS to VHH) for test mode entry	–	–	8.0	µs	
P2	Tf	$\overline{MCLR}$ 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 date out valid (during read data)	200	–	–	ns	
P8	Thld0	Hold time after $\overline{MCLR}$ ↑	2	–	–	µs	

Note 1: Program must be verified at the minimum and maximum VDD limits for the part.
Note 2: VIHH must be greater than VDD + 4.5V to stay in programming/verify mode.

FIGURE 5-1: LOAD DATA COMMAND (PROGRAM/VERIFY)

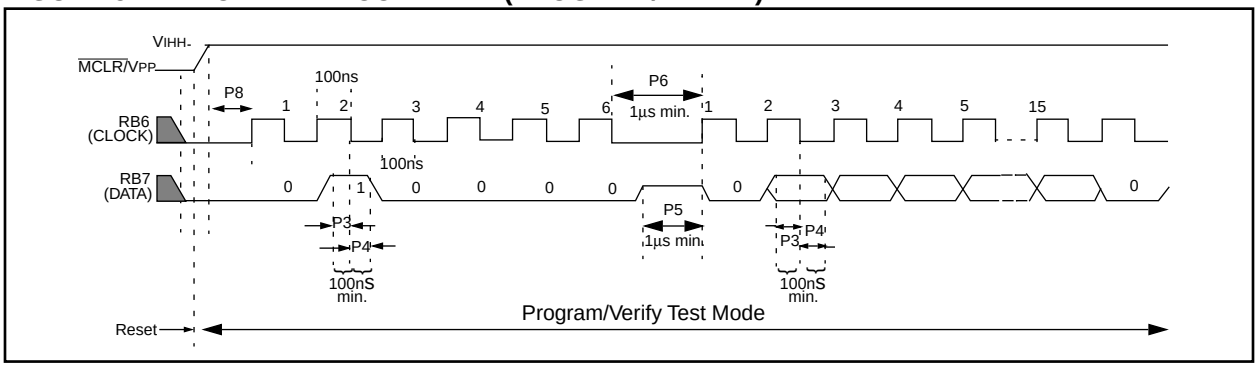


FIGURE 5-2: READ DATA COMMAND (PROGRAM/VERIFY)

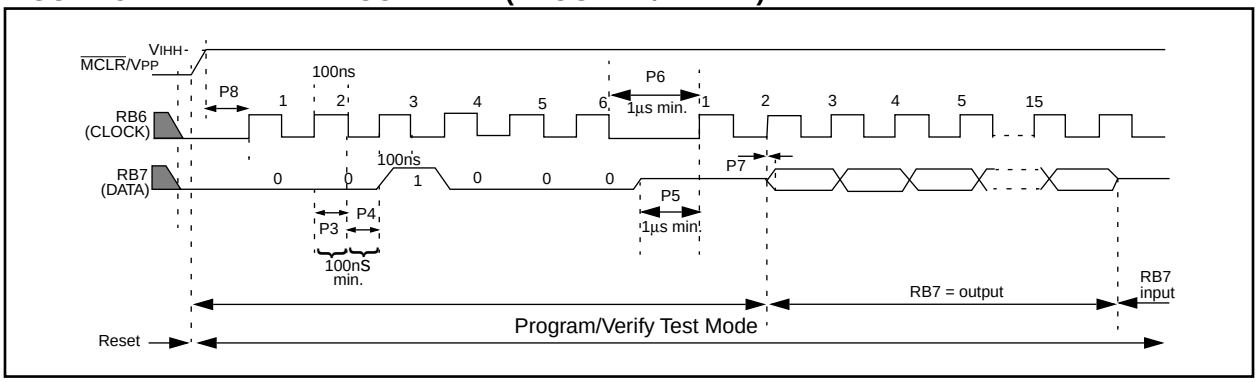
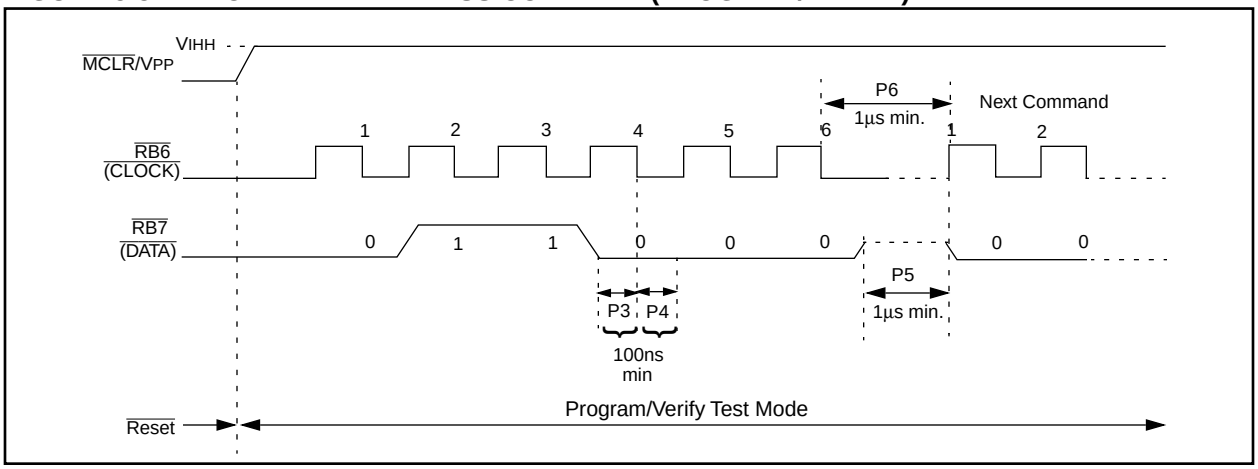


FIGURE 5-3: INCREMENT ADDRESS COMMAND (PROGRAM/VERIFY)



WORLDWIDE SALES & SERVICE

AMERICAS

Corporate Office
Microchip Technology Inc.
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 602-786-7200 Fax: 602-786-7277
Technical Support: 602 786-7627
Web: 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.
14651 Dallas Parkway, Suite 816
Dallas, TX 75240-8809
Tel: 972-991-7177 Fax: 972-991-8588

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

Los Angeles
Microchip Technology Inc.
18201 Von Karman, Suite 1090
Irvine, CA 92612
Tel: 714-263-1888 Fax: 714-263-1338

New York
Microchip Technology Inc.
150 Motor Parkway, Suite 416
Hauppauge, NY 11788
Tel: 516-273-5305 Fax: 516-273-5335

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

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

ASIA/PACIFIC

Hong Kong
Microchip Asia Pacific
RM 3801B, Tower Two
Metroplaza
223 Hing Fong Road
Kwai Fong, N.T., Hong Kong
Tel: 852-2-401-1200 Fax: 852-2-401-3431

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

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
RM 406 Shanghai Golden Bridge Bldg.
2077 Yan'an Road West, Hongiao District
Shanghai, PRC 200335
Tel: 86-21-6275-5700
Fax: 86 21-6275-5060

Singapore
Microchip Technology Taiwan
Singapore Branch
200 Middle Road
#10-03 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 2-717-7175 Fax: 886-2-545-0139

EUROPE

United Kingdom
Arizona Microchip Technology Ltd.
Unit 6, The Courtyard
Meadow Bank, Furlong Road
Bourne End, Buckinghamshire SL8 5AJ
Tel: 44-1628-851077 Fax: 44-1628-850259

France
Arizona Microchip Technology SARL
Zone Industrielle de la Bonde
2 Rue du Buisson aux Fraises
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 Müchen, Germany
Tel: 49-89-627-144 0 Fax: 49-89-627-144-44

Italy
Arizona Microchip Technology SRL
Centro Direzionale Colleone
Palazzo Taurus 1 V. Le Colleoni 1
20041 Agrate Brianza
Milan, Italy
Tel: 39-39-6899939 Fax: 39-39-6899883

JAPAN

Microchip Technology Intl. Inc.
Benex S-1 6F
3-18-20, Shin Yokohama
Kohoku-Ku, Yokohama
Kanagawa 222 Japan
Tel: 81-4-5471- 6166 Fax: 81-4-5471-6122

06/16/97

DNV Certification, Inc. USA
ACCREDITED REGISTRAR ANSI-RAB
DNV
DNV MSC The Netherlands Accredited by the RvA
ISO 9001 REGISTERED FIRM