PIC16C6XX/7XX/9XX

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

This document includes the programming specifications for the following devices:

- PIC16C61
- PIC16C62
- PIC16C62A
- PIC16C62B
- PIC16C63
- PIC16C63A
- PIC16C64
- PIC16C64A
- PIC16C65
- PIC16C65A
- PIC16C65B
- PIC16C66
- PIC16C67
- PIC16C71
- PIC16C72
- PIC16C72A
- PIC16C73
- PIC16C73A
- PIC16C73B
- PIC16C74
- PIC16C74A
- PIC16C74B
- PIC16C76
- PIC16C77
- PIC16C620
- PIC16C620A
- PIC16C621
- PIC16C621A
- PIC16C622
- PIC16C622A
- PIC16CE623
- PIC16CE624
- PIC16CE625
- PIC16C710
- PIC16C711
- PIC16C712
- PIC16C716
- PIC16C745
- PIC16C765
- PIC16C773
- PIC16C774
- PIC16C923
- PIC16C924
- PIC16C925
- PIC16C926

1.0 PROGRAMMING THE PIC16C6XX/7XX/9XX

The PIC16C6XX/7XX/9XX family 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

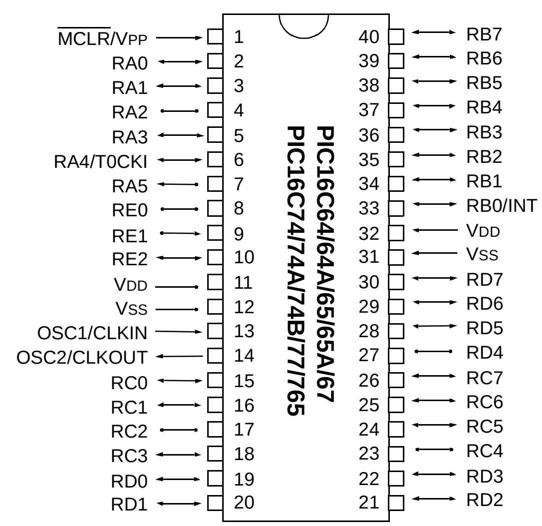
The PIC16C6XX/7XX/9XX requires two programmable power supplies, one for VDD (2.0V to 6.5V recommended) and one for VPP (12V to 14V). Both supplies should have a minimum resolution of 0.25V.

1.2 Programming Mode

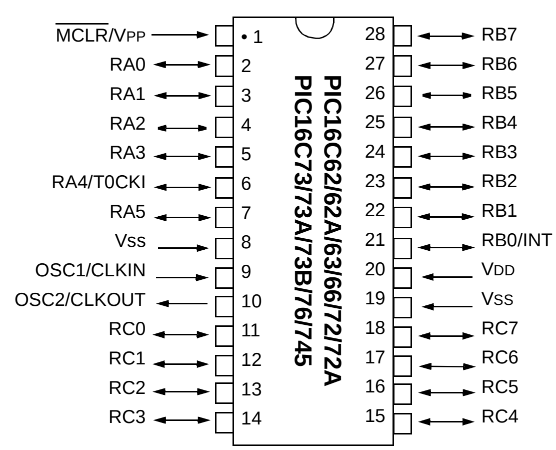
The Programming mode for the PIC16C6XX/7XX/9XX allows programming of user program memory, special locations used for ID, and the configuration word for the PIC16C6XX/7XX/9XX.

Pin Diagrams

PDIP, Windowed CERDIP

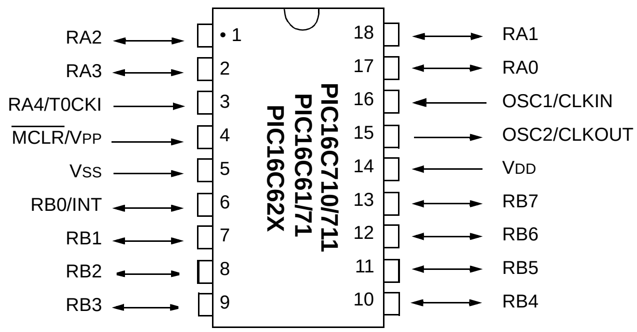


PDIP, SOIC, Windowed CERDIP (300 mil)

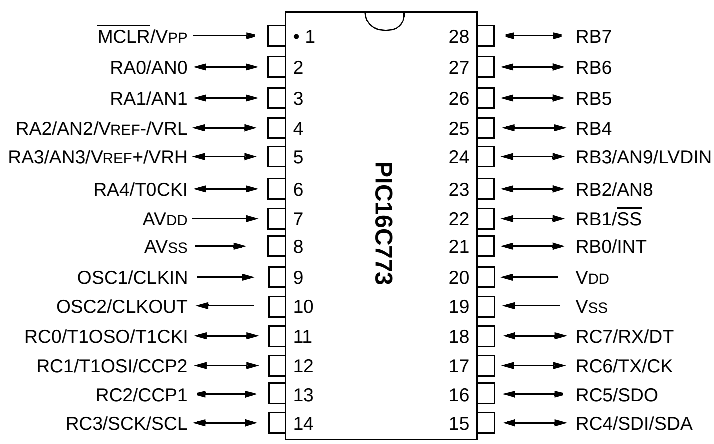


Pin Diagrams (Con't)

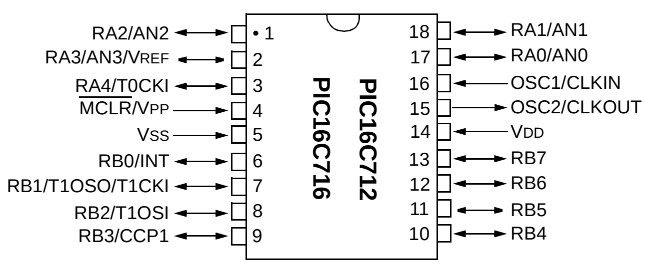
PDIP, SOIC, Windowed CERDIP



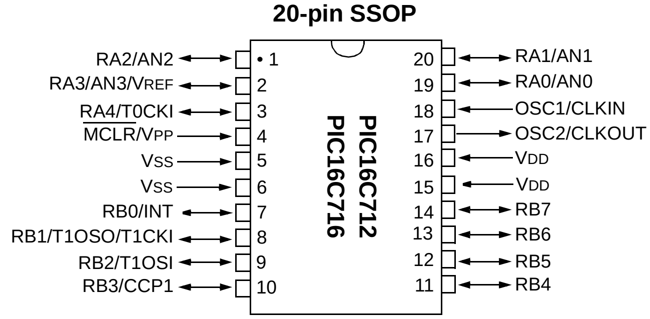
300 mil. SDIP, SOIC, Windowed CERDIP, SSOP



18-pin PDIP, SOIC, Windowed CERDIP

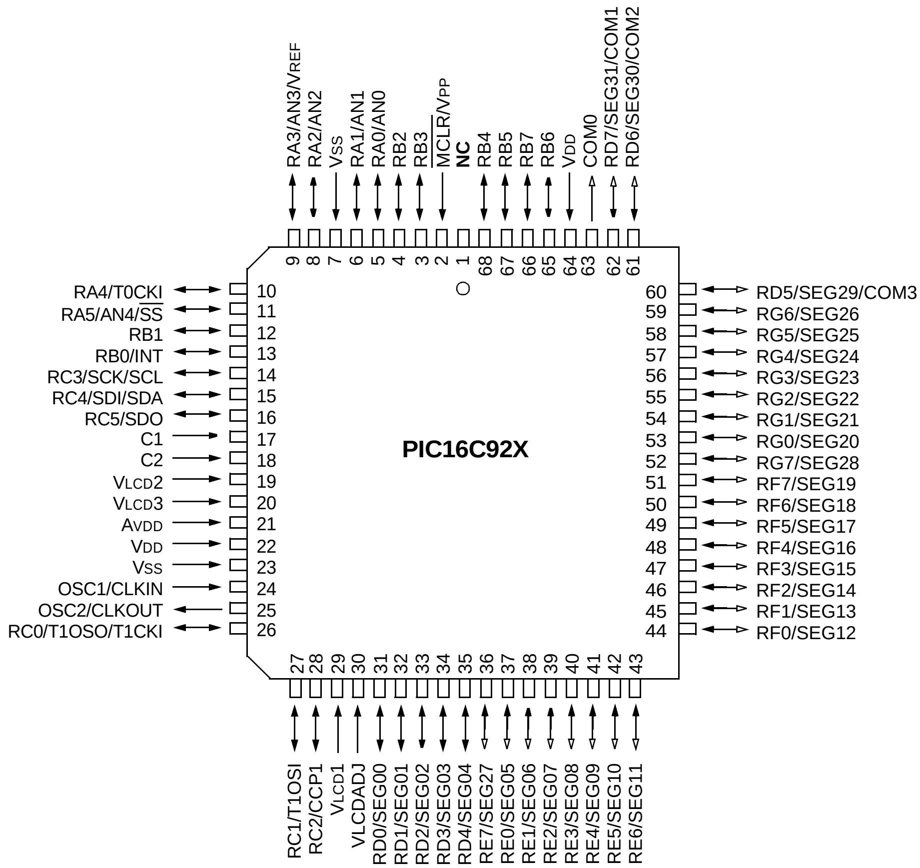


20-pin SSOP



Pin Diagrams (Con't)

PLCC, CLCC



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)
PIC16C923/924/925	0x000 – 0xFFF (4K)
PIC16C926	0x000 – 0x1FFF (8K)

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 re-enter Program/Verify mode, as described in Section 2.2.

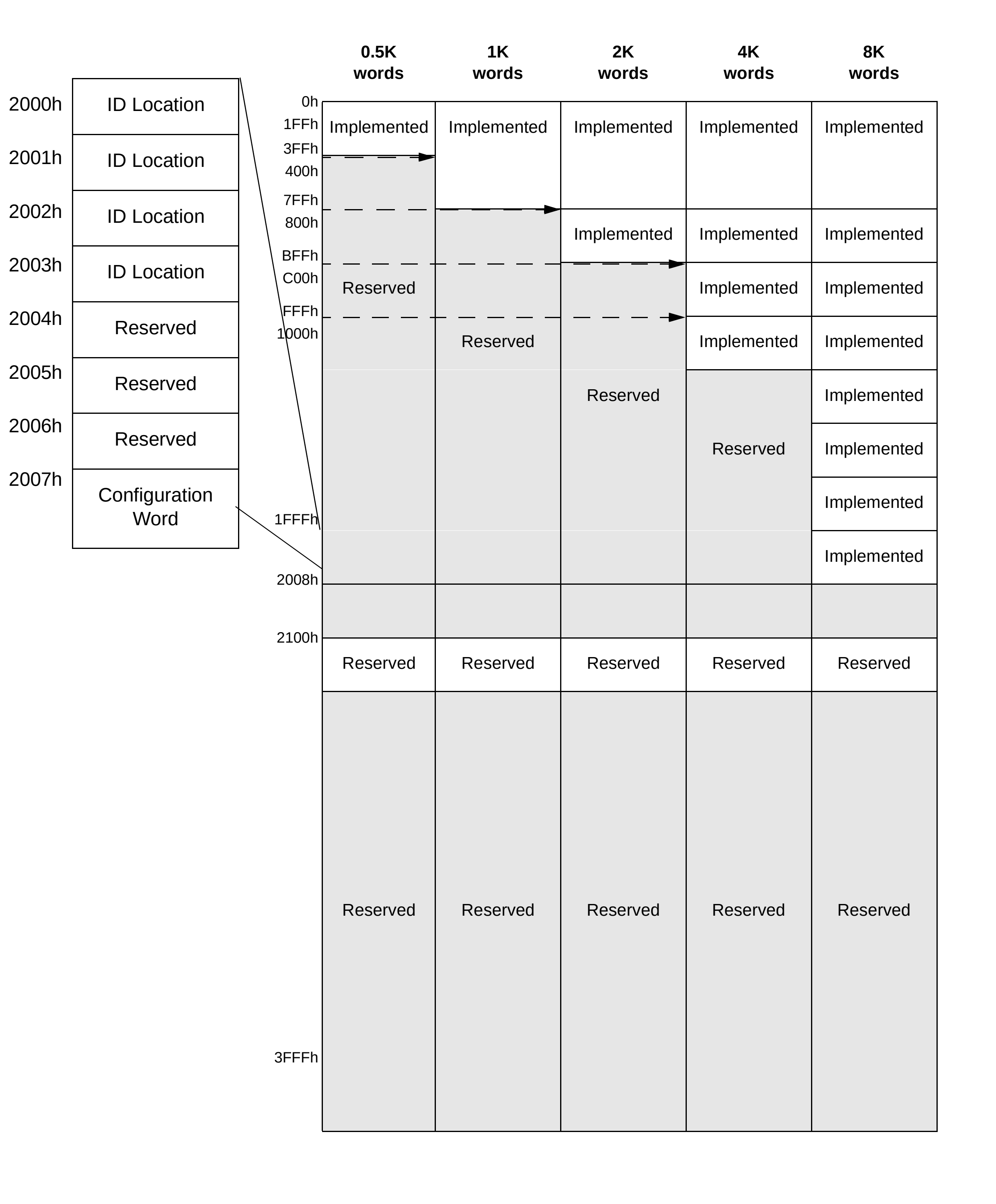
A user may store identification information (ID) in four ID locations. The ID locations are mapped in [0x2000: 0x2003]. It is recommended that the user use only the four Least Significant bits of each ID location. In some devices, the ID locations read-out in a scrambled fashion after code protection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

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 VSS to the appropriate VIHH (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 VSS). This means that all I/O are in the RESET state (high impedance inputs).

> **Note 1:** The $\overline{MCLR}$ pin should be raised as quickly as possible from VIL to VIHH. 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 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

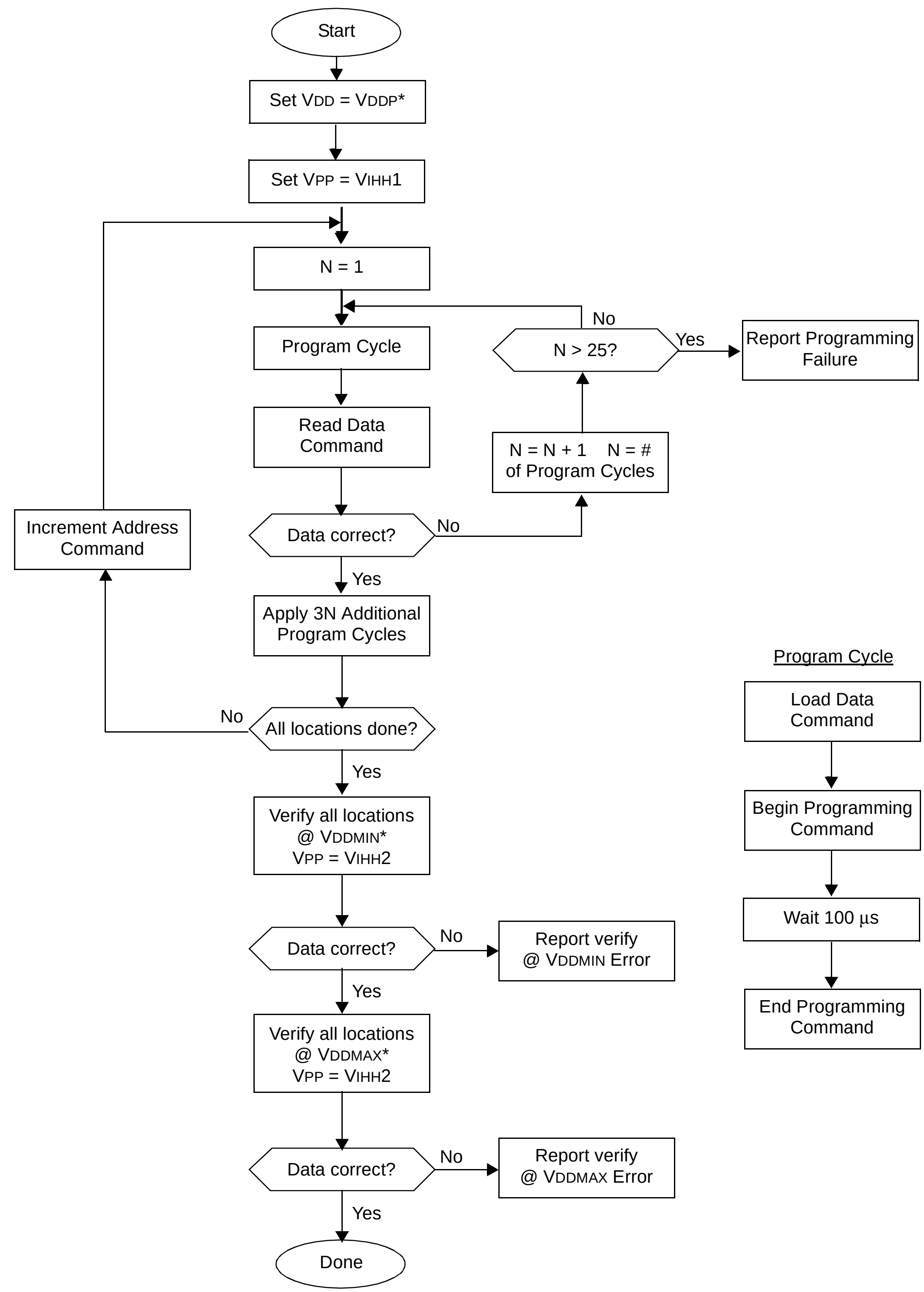
After receiving this command, the program counter (PC) will be set to 0x2000. By then applying 16 cycles to the clock pin, the chip will load 14-bits, a “data word” as described above, to be programmed into the configuration memory. A description of the memory mapping schemes for normal operation and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

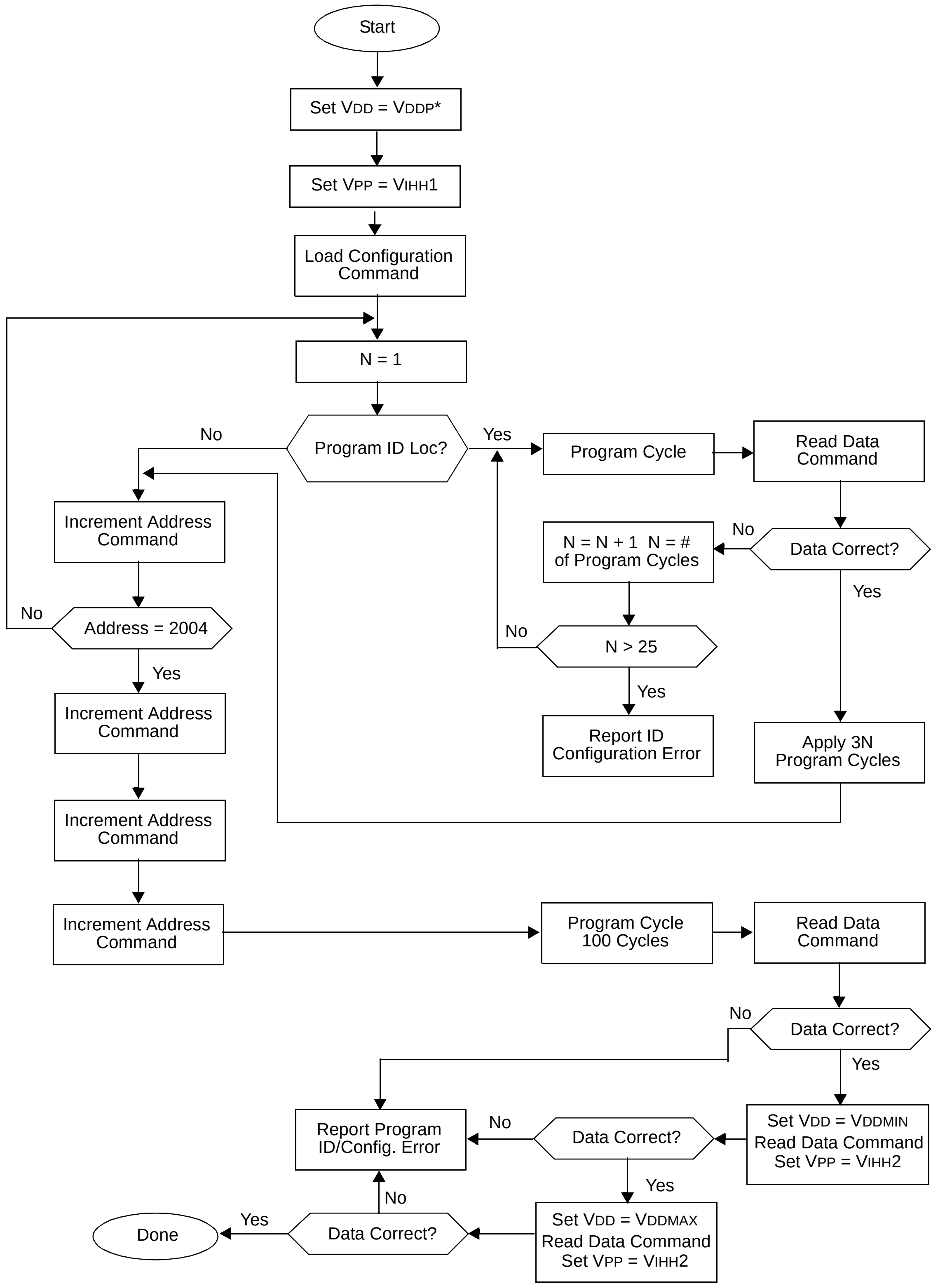
Note: The clock must be disabled during In-Circuit Serial Programming™.

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 AND ID LOCATIONS



*VDDP = VDD range for programming (typically 4.75V - 5.25V).
VDDMIN = Minimum VDD for device operation.
VDDMAX = Maximum VDD for device operation.

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 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 PIC16C6XX/7XX/9XX family uses an intelligent algorithm. The algorithm calls for program verification at V_{DDMIN} as well as V_{DDMAX}. Verification at V_{DDMIN} guarantees a good "erase margin". Verification at V_{DDMAX} guarantees a 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_{DDMIN} = minimum operating V_{DD} spec for the part.

V_{DDMAX} = maximum operating V_{DD} spec for the part

Programmers must verify the PIC16C6XX/7XX/9XX at its specified V_{DDMAX} and V_{DDMIN} levels. Since Microchip may introduce future versions of the PIC16C6XX/7XX/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 PIC16C6XX/7XX/9XX family members have several configuration bits. For all devices, these are part of the Configuration Word, located at address 2007h. These bits can be programmed (reads ’0’), or left unprogrammed (reads ’1’), to select various device configurations.

Because the PIC16C6XX/7XX/9XX family spans so many devices, there are a number of different bit configurations possible for the Configuration Word. Registers 3-1 through 3-7 provide details for each of the seven distinct groups. Table 3-1 provides a cross-index of a particular device name to its appropriate Configuration Word listing.

Note: Throughout the PIC16C6XX/7XX/9XX family, two different implementations of the Power-up Timer Enable bit are used. PWRTEN (timer enabled when bit is set to ‘1’) is used on some earlier PIC16C6X and PIC16C7X devices. $\overline{\text{PWRTEN}}$ (timer enabled when bit is set to ‘0’) is used for all other devices. Please carefully note the distinction between these two versions.

TABLE 3-1: PIC16C6XX/7XX/9XX DEVICES AND THEIR CONFIGURATION WORD REGISTERS

Device	Register	Page	Device	Register	Page	Device	Register	Page
PIC16C61	3-1	11	PIC16C72A	3-3	12	PIC16CE623	3-3	12
PIC16C62	3-2	11	PIC16C73	3-2	11	PIC16CE624	3-3	12
PIC16C62A	3-3	12	PIC16C73A	3-3	12	PIC16CE625	3-3	12
PIC16C62B	3-3	12	PIC16C73B	3-3	12	PIC16C710	3-4	13
PIC16C63	3-3	12	PIC16C74	3-2	11	PIC16C711	3-4	13
PIC16C63A	3-3	12	PIC16C74A	3-3	12	PIC16C712	3-3	12
PIC16C64	3-2	11	PIC16C74B	3-3	12	PIC16C716	3-3	12
PIC16C64A	3-3	12	PIC16C76	3-3	12	PIC16C745	3-6	15
PIC16C65	3-2	11	PIC16C77	3-3	12	PIC16C765	3-6	15
PIC16C65A	3-3	12	PIC16C620	3-3	12	PIC16C773	3-5	14
PIC16C65B	3-3	12	PIC16C620A	3-3	12	PIC16C774	3-5	14
PIC16C66	3-3	12	PIC16C621	3-3	12	PIC16C923	3-6	15
PIC16C67	3-3	12	PIC16C621A	3-3	12	PIC16C924	3-6	15
PIC16C71	3-1	11	PIC16C622	3-3	12	PIC16C925	3-7	16
PIC16C72	3-3	12	PIC16C622A	3-3	12	PIC16C926	3-7	16

REGISTER 3-1: CONFIGURATION WORD FOR PIC16C61/71 (ADDRESS 2007h)

—	—	—	—	—	—	—	—	—	CP0	PWTREN	WDTEN	F0SC1	F0SC0
bit13													bit0

bit 13-5 **Unimplemented:** Read as ‘1’

bit 4 **CP0**: Code Protection bit
`1` = Code protection off
`0` = All memory code protected

bit 3 **PWTREN**: Power-up Timer Enable bit
`1` = PWRT enabled
`0` = PWRT disabled

bit 2 **WDTEN**: Watchdog Timer Enable bit
`1` = WDT enabled
`0` = WDT disabled

bit 1-0 **FOSC1:FOSC0**: Oscillator Selection bits
`11` = RC oscillator
`10` = HS oscillator
`01` = XT oscillator
`00` = LP oscillator

REGISTER 3-2: CONFIGURATION WORD FOR PIC16C62/64/65/73/74 (ADDRESS 2007h)

—	—	—	—	—	—	—	—	CP1	CP0	PWTREN	WDTEN	F0SC1	F0SC0
bit13													bit0

bit 13-6 **Unimplemented:** Read as ‘1’

bit 5-4 **CP<1:0>:** Code Protection bits
`11` = Code protection off
`10` = Upper 1/2 memory code protected
`01` = Upper 3/4 memory code protected
`00` = All memory is protected

bit 3 **PWTREN**: Power-up Timer Enable bit[(2)]
`1` = PWRT enabled
`0` = PWRT disabled

bit 2 **WDTEN**: Watchdog Timer Enable bit
`1` = WDT enabled
`0` = WDT disabled

bit 1-0 **FOSC1:FOSC0**: Oscillator Selection bits
`11` = RC oscillator
`10` = HS oscillator
`01` = XT oscillator
`00` = LP oscillator

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

REGISTER 3-3: CONFIGURATION WORD FOR: PIC16C62A/62B/62C/63/63A/64A/65A/65B/66/67 PIC16C72/72A/73A/73B/74A/74B/76/77 PIC16C620/620A/621/621A/622/622A/712/716 PIC16CE623/624/625 (ADDRESS 2007h)

CP1	CP0	CP1	CP0	CP1	CP0	—	BOREN	CP1	CP0	$\overline{\text{PWTREN}}$	WDTEN	F0SC1	F0SC0
bit13													bit0

bit 13-8 **CP<1:0>:** Code Protection bits(1)
bit 5-4 For all devices EXCEPT PIC16C620, PIC16C621, PIC16CE623 and PIC16CE624:
`11` = Code protection off
`10` = Upper 1/2 of program memory code protected
`01` = Upper 3/4 of program memory code protected
`00` = All memory is protected

For the PIC16C621 and PIC16CE624:
`1x` = Code protection off
`01` = Upper 1/2 of program memory code protected
`00` = All program memory is code protected

For the PIC16C620 and PIC16CE623:
`1x,01` = Code protection off
`00` = All program memory is code protected

bit 7 **Unimplemented:** Read as ‘1’

bit 6 **BOREN**: Brown-out Reset Enable bit(2)
`1` = BOR enabled
`0` = BOR disabled

bit 3 **$\overline{\text{PWTREN}}$**: Power-up Timer Enable bit(2)
`1` = PWRT disabled
`0` = PWRT enabled

bit 2 **WDTEN**: Watchdog Timer Enable bit
`1` = WDT enabled
`0` = WDT disabled

bit 1-0 **FOSC1:FOSC0**: Oscillator Selection bits
`11` = RC oscillator
`10` = HS oscillator
`01` = XT oscillator
`00` = LP oscillator

Note 1: All of the CP<1:0> bit pairs have to be given the same value to enable the code protection scheme listed.

2: Enabling Brown-out Reset automatically enables Power-up Timer (PWRT), regardless of the value of bit $\overline{\text{PWTREN}}$. Ensure the Power-up Timer is enabled any time Brown-out Reset is enabled.

REGISTER 3-4: CONFIGURATION WORD, PIC16C710/711 (ADDRESS 2007h)

CP0	CP0	CP0	CP0	CP0	CP0	CP0	BOREN	CP0	CP0	$\overline{\text{PWTREN}}$	WDTEN	F0SC1	F0SC0

bit13 bit0

bit 13-7
bit 5-4 **CP0:** Code Protection bits[1]
`1` = Code protection off
`0` = All program memory is code protected, but 00h - 3Fh is writable

bit 6 **BOREN**: Brown-out Reset Enable bit[2]
`1` = BOR enabled
`0` = BOR disabled

bit 3 **$\overline{\text{PWTREN}}$**: Power-up Timer Enable bit[2]
`1` = PWRT disabled
`0` = PWRT enabled

bit 2 **WDTEN**: Watchdog Timer Enable bit
`1` = WDT enabled
`0` = WDT disabled

bit 1-0 **FOSC1:FOSC0**: Oscillator Selection bits
`11` = RC oscillator
`10` = HS oscillator
`01` = XT oscillator
`00` = LP oscillator

Note 1: All of the CP0 bits have to be given the same value to enable the code protection scheme listed.

2: Enabling Brown-out Reset automatically enables Power-up Timer (PWRT), regardless of the value of bit $\overline{\text{PWTREN}}$. Ensure the Power-up Timer is enabled any time Brown-out Reset is enabled.

REGISTER 3-5: CONFIGURATION WORD, PIC16C773/774 (ADDRESS 2007h)

CP1	CP0	BORV1	BORV0	CP1	CP0	—	BOREN	CP1	CP0	$\overline{\text{PWTREN}}$	WDTEN	F0SC1	F0SC0

bit13 bit0

bit 13-7
bit 9-8
bit 5-4
CP<1:0>: Code Protection bits[1]
11 = Code protection off
10 = Upper 1/2 of program memory code protected
01 = Upper 3/4 of program memory code protected
00 = All program memory is code protected

bit 11-10 **BORV <1:0>:** Brown-out Reset Voltage bits
11 = V_{BOR} set to 2.5V
10 = V_{BOR} set to 2.7V
01 = V_{BOR} set to 4.2V
00 = V_{BOR} set to 4.5V

bit 6 **BOREN:** Brown-out Reset Enable bit[2]
1 = BOR enabled
0 = BOR disabled

bit 3 **$\overline{\text{PWTREN}}$:** Power-up Timer Enable bit[2]
1 = PWRT disabled
0 = PWRT enabled

bit 2 **WDTEN:** Watchdog Timer Enable bit
1 = WDT enabled
0 = WDT disabled

bit 1-0 **FOSC1:FOSC0:** Oscillator Selection bits
11 = RC oscillator
10 = HS oscillator
01 = XT oscillator
00 = LP oscillator

Note 1: All of the CP<1:0> bits pairs have to be given the same value to enable the code protection scheme listed.

2: Enabling Brown-out Reset automatically enables Power-up Timer (PWRT), regardless of the value of bit $\overline{\text{PWTREN}}$. Ensure the Power-up Timer is enabled any time Brown-out Reset is enabled.

REGISTER 3-6: CONFIGURATION WORD FOR: PIC16C745/765/923/924 (ADDRESS 2007h)

CP1	CP0	CP1	CP0	CP1	CP0	—	—	CP1	CP0	$\overline{\text{PWTREN}}$	WDTEN	F0SC1	F0SC0
bit13													bit0

bit 13-8
bit 5-4
CP<1:0>: Code Protection bits(1)
`11` = Code protection off
`10` = Upper 1/2 of program memory code protected
`01` = Upper 3/4 of program memory code protected
`00` = All program memory is code protected

bit 7-6 **Unimplemented:** Read as ‘1’

bit 3 **$\overline{\text{PWTREN}}$:** Power-up Timer Enable bit(2)
`1` = PWRT disabled
`0` = PWRT enabled

bit 2 **WDTEN:** Watchdog Timer Enable bit
`1` = WDT enabled
`0` = WDT disabled

bit 1-0 **FOSC1:FOSC0:** Oscillator Selection bits
For PIC16745/765:
`11` = E external clock with 4K PLL
`10` = H HS oscillator with 4K PL enabled
`01` = EC external clock with CLKOUT on OSC2
`00` = HS oscillator
For PIC16923/924:
`11` = RC oscillator
`10` = HS oscillator
`01` = XT oscillator
`00` = LP oscillator

Note 1: All of the CP<1:0> bits pairs have to be given the same value to enable the code protection scheme listed.

REGISTER 3-7: CONFIGURATION WORD FOR PIC16C925/926 (ADDRESS 2007h)

—	—	—	—	—	—	—	BOREN	CP1	CP0	$\overline{\text{PWTREN}}$	WDTEN	F0SC1	F0SC0
bit13													bit0

bit 13-7 **Unimplemented:** Read as '1'

bit 6 **BOREN:** Brown-out Reset Enable bit[(1)]
1 = BOR enabled
0 = BOR disabled

bit 5-4 **CP<1:0>:** Program Memory Code Protection bits
For PIC16C926:
11 = Code protection off
10 = Lower 1/2 of program memory code protected (0000h-0FFFh)
01 = All but last 256 bytes of program memory code protected (0000h-1EFFh)
00 = All memory is protected
For PIC16C925:
11 = Code protection off
10 = Lower 1/2 of program memory code protected (0000h-07FFh)
01 = All but last 256 bytes of program memory code protected (0000h-0EFFh)
00 = All program memory is protected

Note: For PIC16C925, address values of 1000h to 1FFFh wrap around to 0000h to 0FFFh.

bit 3 **$\overline{\text{PWTREN}}$:** Power-up Timer Enable bit[(1)]
1 = PWRT disabled
0 = PWRT enabled

bit 2 **WDTEN:** Watchdog Timer Enable bit
1 = WDT enabled
0 = WDT disabled

bit 1-0 **FOSC1:FOSC0:** Oscillator Selection bits
11 = RC oscillator
10 = HS oscillator
01 = XT oscillator
00 = LP oscillator

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

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.

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 3-2.

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 3-2: CHECKSUM COMPUTATION

Device	Code Protect	Checksum*	Blank Value	0x25E6 at 0 and Max Address
PIC16C61	OFF ON	SUM[0x000:0x3FF] + CFGW & 0x001F + 0x3FE0 SUM_XNOR7[0x000:0x3FF] + (CFGW & 0x001F | 0x0060)	0x3BFF 0xFC6F	0x07CD 0xFC15
PIC16C620	OFF ON	SUM[0x000:0x1FF] + CFGW & 0x3F7F SUM_ID + CFGW & 0x3F7F	0x3D7F 0x3DCE	0x094D 0x099C
PIC16C620A	OFF ON	SUM[0x000:0x1FF] + CFGW & 0x3F7F SUM_ID + CFGW & 0x3F7F	0x3D7F 0x3DCE	0x094D 0x099C
PIC16C621	OFF 1/2 ALL	SUM[0x000:0x3FF] + CFGW & 0x3F7F SUM[0x000:0x1FF] + CFGW & 0x3F7F + SUM_ID CFGW & 0x3F7F + SUM_ID	0x3B7F 0x4EDE 0x3BCE	0x074D 0x0093 0x079C
PIC16C621A	OFF 1/2 ALL	SUM[0x000:0x3FF] + CFGW & 0x3F7F SUM[0x000:0x1FF] + CFGW & 0x3F7F + SUM_ID CFGW & 0x3F7F + SUM_ID	0x3B7F 0x4EDE 0x3BCE	0x074D 0x0093 0x079C

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 3-2: CHECKSUM COMPUTATION (CONTINUED)

Device	Code Protect	Checksum*	Blank Value	0x25E6 at 0 and Max Address
PIC16C622	OFF 1/2 3/4 ALL	SUM[0x000:0x7FF] + CFGW & 0x3F7F SUM[0x000:0x3FF] + CFGW & 0x3F7F + SUM_ID SUM[0x000:0x1FF] + CFGW & 0x3F7F + SUM_ID CFGW & 0x3F7F + SUM_ID	0x377F 0x5DEE 0x4ADE 0x37CE	0x034D 0x0FA3 0xFC93 0x039C
PIC16C622A	OFF 1/2 3/4 ALL	SUM[0x000:0x7FF] + CFGW & 0x3F7F SUM[0x000:0x3FF] + CFGW & 0x3F7F + SUM_ID SUM[0x000:0x1FF] + CFGW & 0x3F7F + SUM_ID CFGW & 0x3F7F + SUM_ID	0x377F 0x5DEE 0x4ADE 0x37CE	0x034D 0x0FA3 0xFC93 0x039C
PIC16CE623	OFF ON	SUM[0x000:0x1FF] + CFGW & 0x3F7F SUM_ID + CFGW & 0x3F7F	0x3D7F 0x3DCE	0x094D 0x099C
PIC16CE624	OFF 1/2 ALL	SUM[0x000:0x3FF] + CFGW & 0x3F7F SUM[0x000:0x1FF] + CFGW & 0x3F7F + SUM_ID CFGW & 0x3F7F + SUM_ID	0x3B7F 0x4EDE 0x3BCE	0x074D 0x0093 0x079C
PIC16CE625	OFF 1/2 3/4 ALL	SUM[0x000:0x7FF] + CFGW & 0x3F7F SUM[0x000:0x3FF] + CFGW & 0x3F7F + SUM_ID SUM[0x000:0x1FF] + CFGW & 0x3F7F + SUM_ID CFGW & 0x3F7F + SUM_ID	0x377F 0x5DEE 0x4ADE 0x37CE	0x034D 0x0FA3 0xFC93 0x039C
PIC16C62	OFF 1/2 3/4 ALL	SUM[0x000:0x7FF] + CFGW & 0x003F + 0x3F80 SUM[0x000:0x3FF] + SUM_XNOR7[0x400:0x7FF] + CFGW & 0x003F + 0x3F80 SUM[0x000:0x1FF] + SUM_XNOR7[0x200:0x7FF] + CFGW & 0x003F + 0x3F80 SUM_XNOR7[0x000:0x7FF] + CFGW & 0x003F + 0x3F80	0x37BF 0x37AF 0x379F 0x378F	0x038D 0x1D69 0x1D59 0x3735
PIC16C62A	OFF 1/2 3/4 ALL	SUM[0x000:0x7FF] + CFGW & 0x3F7F SUM[0x000:0x3FF] + CFGW & 0x3F7F + SUM_ID SUM[0x000:0x1FF] + CFGW & 0x3F7F + SUM_ID CFGW & 0x3F7F + SUM_ID	0x377F 0x5DEE 0x4ADE 0x37CE	0x034D 0x0FA3 0xFC93 0x039C
PIC16C62B	OFF 1/2 3/4 ALL	SUM[0x000:0x7FF] + CFGW & 0x3F7F SUM[0x000:0x3FF] + CFGW & 0x3F7F + SUM_ID SUM[0x000:0x1FF] + CFGW & 0x3F7F + SUM_ID CFGW & 0x3F7F + SUM_ID	0x377F 0x5DEE 0x4ADE 0x37CE	0x034D 0x0FA3 0xFC93 0x039C
PIC16C63	OFF 1/2 3/4 ALL	SUM[0x000:0xFFF] + CFGW & 0x3F7F SUM[0x000:0x7FF] + CFGW & 0x3F7F + SUM_ID SUM[0x000:0x3FF] + CFGW & 0x3F7F + SUM_ID CFGW & 0x3F7F + SUM_ID	0x2F7F 0x51EE 0x40DE 0x2FCE	0xFB4D 0x03A3 0xF293 0xFB9C
PIC16C63A	OFF 1/2 3/4 ALL	SUM[0x000:0xFFF] + CFGW & 0x3F7F SUM[0x000:0x7FF] + CFGW & 0x3F7F + SUM_ID SUM[0x000:0x3FF] + CFGW & 0x3F7F + SUM_ID CFGW & 0x3F7F + SUM_ID	0x2F7F 0x51EE 0x40DE 0x2FCE	0xFB4D 0x03A3 0xF293 0xFB9C
PIC16C64	OFF 1/2 3/4 ALL	SUM[0x000:0x7FF] + CFGW & 0x003F + 0x3F80 SUM[0x000:0x3FF] + SUM_XNOR7[0x400:0x7FF] + CFGW & 0x003F + 0x3F80 SUM[0x000:0x1FF] + SUM_XNOR7[0x200:0x7FF] + CFGW & 0x003F + 0x3F80 SUM_XNOR7[0x000:0x7FF] + CFGW & 0x003F + 0x3F80	0x37BF 0x37AF 0x379F 0x378F	0x038D 0x1D69 0x1D59 0x3735

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 3-2: CHECKSUM COMPUTATION (CONTINUED)

Device	Code Protect	Checksum*	Blank Value	0x25E6 at 0 and Max Address
PIC16C64A	OFF 1/2 3/4 ALL	SUM[0x000:0x7FF] + CFGW & 0x3F7F SUM[0x000:0x3FF] + CFGW & 0x3F7F + SUM_ID SUM[0x000:0x1FF] + CFGW & 0x3F7F + SUM_ID CFGW & 0x3F7F + SUM_ID	0x377F 0x5DEE 0x4ADE 0x37CE	0x034D 0x0FA3 0xFC93 0x039C
PIC16C65	OFF 1/2 3/4 ALL	SUM[0x000:0xFFF] + CFGW & 0x003F + 0x3F80 SUM[0x000:0x7FF] + SUM_XNOR7[0x800:FFF] + CFGW & 0x003F + 0x3F80 SUM[0x000:0x3FF] + SUM_XNOR7[0x400:FFF] + CFGW & 0x003F + 0x3F80 SUM_XNOR7[0x000:0xFFF] + CFGW & 0x003F + 0x3F80	0x2FBF 0x2FAF 0x2F9F 0x2F8F	0xFB8D 0x1569 0x1559 0x2F35
PIC16C65A	OFF 1/2 3/4 ALL	SUM[0x000:0xFFF] + CFGW & 0x3F7F SUM[0x000:0x7FF] + CFGW & 0x3F7F + SUM_ID SUM[0x000:0x3FF] + CFGW & 0x3F7F + SUM_ID CFGW & 0x3F7F + SUM_ID	0x2F7F 0x51EE 0x40DE 0x2FCE	0xFB4D 0x03A3 0xF293 0xFB9C
PIC16C65B	OFF 1/2 3/4 ALL	SUM[0x000:0xFFF] + CFGW & 0x3F7F SUM[0x000:0x7FF] + CFGW & 0x3F7F + SUM_ID SUM[0x000:0x3FF] + CFGW & 0x3F7F + SUM_ID CFGW & 0x3F7F + SUM_ID	0x2F7F 0x51EE 0x40DE 0x2FCE	0xFB4D 0x03A3 0xF293 0xFB9C
PIC16C66	OFF 1/2 3/4 ALL	SUM[0x000:0x1FFF] + CFGW & 0x3F7F SUM[0x000:0xFFF] + CFGW & 0x3F7F + SUM_ID SUM[0x000:0x7FF] + CFGW & 0x3F7F + SUM_ID CFGW & 0x3F7F + SUM_ID	0x1F7F 0x39EE 0x2CDE 0x1FCE	0xEB4D 0xEBA3 0xDE93 0xEB9C
PIC16C67	OFF 1/2 3/4 ALL	SUM[0x000:0x1FFF] + CFGW & 0x3F7F SUM[0x000:0xFFF] + CFGW & 0x3F7F + SUM_ID SUM[0x000:0x7FF] + CFGW & 0x3F7F + SUM_ID CFGW & 0x3F7F + SUM_ID	0x1F7F 0x39EE 0x2CDE 0x1FCE	0xEB4D 0xEBA3 0xDE93 0xEB9C
PIC16C710	OFF ON	SUM[0x000:0x1FF] + CFGW & 0x3FFF SUM[0x00:0x3F] + CFGW & 0x3FFF + SUM_ID	0x3DFF 0x3E0E	0x09CD 0xEFC3
PIC16C71	OFF ON	SUM[0x000:0x3FF] + CFGW & 0x001F + 0x3FE0 SUM_XNOR7[0x000:0x3FF] + (CFGW & 0x001F | 0x0060)	0x3BFF 0xFC6F	0x07CD 0xFC15
PIC16C711	OFF ON	SUM[0x000:0x03FF] + CFGW & 0x3FFF SUM[0x00:0x3FF] + CFGW & 0x3FFF + SUM_ID	0x3BFF 0x3C0E	0x07CD 0xEDC3
PIC16C712	OFF 1/2 ALL	SUM[0x000:0x07FF] + CFGW & 0x3F7F SUM[0x000:0x03FF] + CFGW & 3F7F + SUM_ID CFGW & 0x3F7F + SUM_ID	0x377F 0x5DEE 0x37CE	0x034D 0xF58A 0x039C
PIC16C716	OFF 1/2 3/4 ALL	SUM[0x000:0x07FF] + CFGW & 0x3F7F SUM[0x000:0x03FF] + CFGW & 0x3F7F + SUM_ID SUM]0x000:0x01FF] + CFGW & 0x3F7F + SUM_ID CFGW & 0x3F7F + SUM_ID	0x377F 0x5DEE 0x4ADE 0x37CE	0x034D 0x0FA3 0xFC93 0x039C
PIC16C72	OFF 1/2 3/4 ALL	SUM[0x000:0x7FF] + CFGW & 0x3F7F SUM[0x000:0x3FF] + CFGW & 0x3F7F + SUM_ID SUM[0x000:0x1FF] + CFGW & 0x3F7F + SUM_ID CFGW & 0x3F7F + SUM_ID	0x377F 0x5DEE 0x4ADE 0x37CE	0x034D 0x0FA3 0xFC93 0x039C

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 3-2: CHECKSUM COMPUTATION (CONTINUED)

Device	Code Protect	Checksum*	Blank Value	0x25E6 at 0 and Max Address
PIC16C72A	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
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
PIC16C73B	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
PIC16C74B	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
PIC16C773	OFF	SUM[0x000:0x0FFF] + CFGW & 0x3F7F	0x2F7F	0xFB4D
	1/2	SUM[0x000:07FF] + CFGW & 0x3F7F + SUM_ID	0x55EE	0x07A3
	3/4	SUM[0x000:03FF] + CFGW & 0x3F7F + SUM_ID	0x48DE	0xFA93
	ALL	CFGW & 0x3F7F + SUM_ID	0x3BCE	0x079C
PIC16C774	OFF	SU:M[0x000:0FFF] + CFGW & 0x3F7F	0x2F7F	0xFB4D
	1/2	SUM[0x000:07FF] + CFGW & 0x3F7F + SUM_ID	0X55EE	0x07A3
	3/4	SUM[0x000:03FF] + CFGW & 0x3F7F + SUM_ID	0X48DE	0xFA93
	ALL	CFGW & 0x3F7F + SUM_ID	0x3BCE	0X079C

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 3-2: CHECKSUM COMPUTATION (CONTINUED)

Device	Code Protect	Checksum*	Blank Value	0x25E6 at 0 and Max Address
PIC16C923 PIC16C925	OFF 1/2 3/4 ALL	SUM[0x000:0xFFF] + CFGW & 0x3F3F SUM[0x000:0x7FF] + CFGW & 0x3F3F + SUM_ID SUM[0x000:0x3FF] + CFGW & 0x3F3F + SUM_ID CFGW & 0x3F3F + SUM_ID	0x2F3F 0x516E 0x405E 0x2F4E	0xFB0D 0x0323 0xF213 0xFB1C
PIC16C924 PIC16C926	OFF 1/2 3/4 ALL	SUM[0x000:0xFFF] + CFGW & 0x3F3F SUM[0x000:0x7FF] + CFGW & 0x3F3F + SUM_ID SUM[0x000:0x3FF] + CFGW & 0x3F3F + SUM_ID CFGW & 0x3F3F + SUM_ID	0x2F3F 0x516E 0x405E 0x2F4E	0xFB0D 0x0323 0xF213 0xFB1C
PIC16C745	OFF 1000:1FFF 800:1FFF ALL	SUM(0000:1FFF) + CFGW & 0x3F3F SUM(0000:0FFF) + CFGW & 0x3F3F+SUM_ID SUM(0000:07FF) + CFGW & 0x3F3F + SUM_ID CFGW * 0x3F3F + SUM_ID	0x1F3F 0x396E 0x2C5E 0x1F4E	0xEB0D 0xEB23 0xDE13 0xEB1C
PIC16C765	OFF 1000:1FFF 800:1FFF ALL	SUM(0000:1FFF) + CFGW & 0x3F3F SUM(0000:0FFF) + CFGW & 0x3F3F+SUM_ID SUM(0000:07FF) + CFGW & 0x3F3F + SUM_ID CFGW * 0x3F3F + SUM_ID	0x1F3F 0x396E 0x2C5E 0x1F4E	0xEB0D 0xEB23 0xDE13 0xEB1C

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°C ≤ T_A ≤ +40°C, unless otherwise stated (20°C recommended)
Operating Voltage: 4.5V ≤ V_{DD} ≤ 5.5V, unless otherwise stated

Parameter No.	Sym.	Characteristic	Min.	Typ.	Max.	Units	Conditions
General							
PD1	V_{DDP}	Supply voltage during programming	4.75	5.0	5.25	V	
PD2	I_{DDP}	Supply current (from V_{DD}) during programming	–	–	20	mA	
PD3	V_{DDV}	Supply voltage during verify	V_{DDMIN}	–	V_{DDMAX}	V	**(Note 1)**
PD4	V_{IHH1}	Voltage on $\overline{MCLR}/V_{PP}$ during programming	12.75	–	13.25	V	**(Note 2)**
PD5	V_{IHH2}	Voltage on $\overline{MCLR}/V_{PP}$ during verify	V_{DD} + 4.5	–	13.25	–	
PD6	I_{PP}	Programming supply current (from V_{PP})	–	–	50	mA	
PD9	V_{IH}	(RB6, RB7) input high level	0.8 V_{DD}	–	–	V	Schmitt Trigger input
PD8	V_{IL}	(RB6, RB7) input low level	0.2 V_{DD}	–	–	V	Schmitt Trigger input
Serial Program Verify							
P1	T_R	$\overline{MCLR}/V_{PP}$ rise time (V_{SS} to V_{HH}) 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 V_{DD} limits for the part.
2: V_{IHH} must be greater than V_{DD} + 4.5V to stay in Programming/Verify mode.

FIGURE 4-1: LOAD DATA COMMAND (PROGRAM/VERIFY)

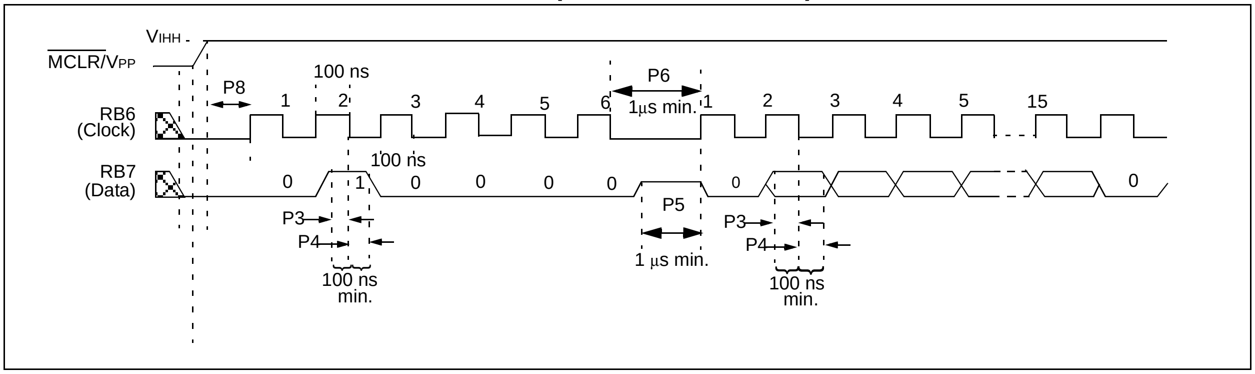


FIGURE 4-2: READ DATA COMMAND (PROGRAM/VERIFY)

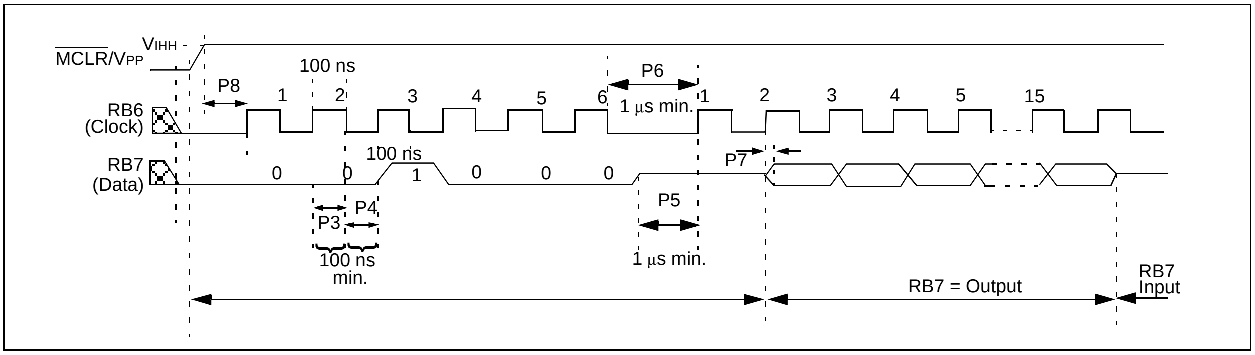
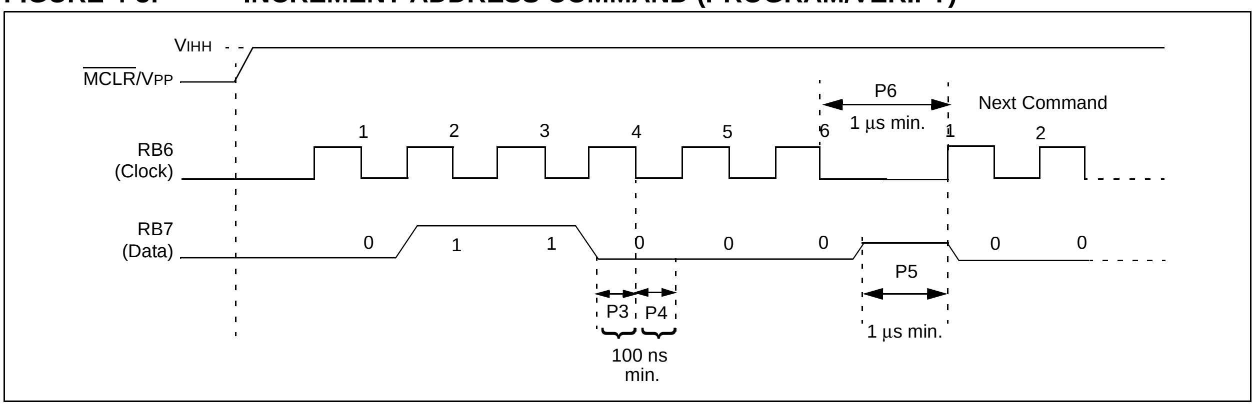


FIGURE 4-3: INCREMENT ADDRESS COMMAND (PROGRAM/VERIFY)



NOTES:

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, KEELOQ® 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.

WORLDWIDE SALES AND SERVICE

AMERICAS

Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7200 Fax: 480-792-7277
Technical Support: 480-792-7627
Web Address: http://www.microchip.com

Rocky Mountain
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 480-792-7966 Fax: 480-792-7456

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

Austin
Analog Product Sales
8303 MoPac Expressway North
Suite A-201
Austin, TX 78759
Tel: 512-345-2030 Fax: 512-345-6085

Boston
2 Lan Drive, Suite 120
Westford, MA 01886
Tel: 978-692-3848 Fax: 978-692-3821

Boston
Analog Product Sales
Unit A-8-1 Millbrook Tarry Condominium
97 Lowell Road
Concord, MA 01742
Tel: 978-371-6400 Fax: 978-371-0050

Chicago
333 Pierce Road, Suite 180
Itasca, IL 60143
Tel: 630-285-0071 Fax: 630-285-0075

Dallas
4570 Westgrove Drive, Suite 160
Addison, TX 75001
Tel: 972-818-7423 Fax: 972-818-2924

Dayton
Two Prestige Place, Suite 130
Miamisburg, OH 45342
Tel: 937-291-1654 Fax: 937-291-9175

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

Los Angeles
18201 Von Karman, Suite 1090
Irvine, CA 92612
Tel: 949-263-1888 Fax: 949-263-1338

Mountain View
Analog Product Sales
1300 Terra Bella Avenue
Mountain View, CA 94043-1836
Tel: 650-968-9241 Fax: 650-967-1590

New York
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

Toronto
6285 Northam Drive, Suite 108
Mississauga, Ontario L4V 1X5, Canada
Tel: 905-673-0699 Fax: 905-673-6509

ASIA/PACIFIC

Australia
Microchip Technology Australia Pty Ltd
Suite 22, 41 Rawson Street
Epping 2121, NSW
Australia
Tel: 61-2-9868-6733 Fax: 61-2-9868-6755

China - Beijing
Microchip Technology Beijing Office
Unit 915
New China Hong Kong Manhattan Bldg.
No. 6 Chaoyangmen Beidajie
Beijing, 100027, No. China
Tel: 86-10-85282100 Fax: 86-10-85282104

China - Shanghai
Microchip Technology Shanghai Office
Room 701, Bldg. B
Far East International Plaza
No. 317 Xian Xia Road
Shanghai, 200051
Tel: 86-21-6275-5700 Fax: 86-21-6275-5060

Hong Kong
Microchip Asia Pacific
RM 2101, Tower 2, Metroplaza
223 Hing Fong Road
Kwai Fong, N.T., Hong Kong
Tel: 852-2401-1200 Fax: 852-2401-3431

India
Microchip Technology Inc.
India Liaison Office
Divyasree Chambers
1 Floor, Wing A (A3/A4)
No. 11, O'Shaugnessey Road
Bangalore, 560 025, India
Tel: 91-80-2290061 Fax: 91-80-2290062

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

ASIA/PACIFIC (continued)

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

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

Taiwan
Microchip Technology Taiwan
11F-3, No. 207
Tung Hua North Road
Taipei, 105, Taiwan
Tel: 886-2-2717-7175 Fax: 886-2-2545-0139

EUROPE

Denmark
Microchip Technology Denmark ApS
Regus Business Centre
Lautrup hoj 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 - ler 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 Munich, Germany
Tel: 49-89-627-144 0 Fax: 49-89-627-144-44

Germany
Analog Product Sales
Lochhamer Strasse 13
D-82152 Martinsried, Germany
Tel: 49-89-895650-0 Fax: 49-89-895650-22

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 5869 Fax: 44-118 921-5820

01/30/01