

EEPROM Memory Programming Specification

This document includes the programming specifications for the following devices:

- PIC16F870
- PIC16F874
- PIC16F871
- PIC16F876
- PIC16F872
- PIC16F877
- PIC16F873

1.0 PROGRAMMING THE PIC16F87X

The PIC16F87X is programmed using a serial method. The Serial mode will allow the PIC16F87X to be programmed while in the user's system. This allows for increased design flexibility. This programming specification applies to PIC16F87X devices in all packages.

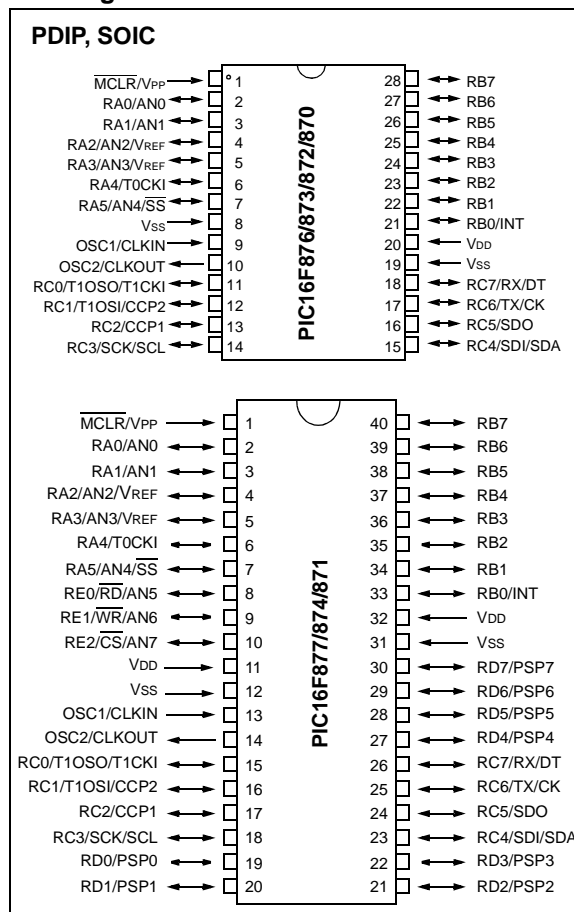
1.1 Programming Algorithm Requirements

The programming algorithm used depends on the operating voltage (VDD) of the PIC16F87X device. Algorithm 1 is designed for a VDD range of $2.2V \leq VDD < 5.5V$. Algorithm 2 is for a range of $4.5V \leq VDD \leq 5.5V$. Either algorithm can be used with the two available programming entry methods. The first method follows the normal Microchip Programming mode entry of applying a VPP voltage of $13V \pm .5V$. The second method, called Low Voltage ICSP™ or LVP for short, applies VDD to MCLR and uses the I/O pin RB3 to enter Programming mode. When RB3 is driven to VDD from ground, the PIC16F87X device enters Programming mode.

1.2 Programming Mode

The Programming mode for the PIC16F87X allows programming of user program memory, data memory, special locations used for ID, and the configuration word.

Pin Diagram



PIC16F87X

PIN DESCRIPTIONS (DURING PROGRAMMING): PIC16F87X

Pin Name	During Programming		
	Function	Pin Type	Pin Description
RB3	PGM	I	Low voltage ICSP programming input if LVP configuration bit equals 1
RB6	CLOCK	I	Clock input
RB7	DATA	I/O	Data input/output
<u>MCLR</u>	VTEST MODE	P*	Program Mode Select
VDD	VDD	P	Power Supply
VSS	VSS	P	Ground

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

* In the PIC16F87X, the programming high voltage is internally generated. To activate the Programming mode, high voltage needs to be applied to the MCLR input. Since the MCLR is used for a level source, this means that MCLR does not draw any significant current.

2.0 PROGRAM MODE ENTRY

2.1 User Program Memory Map

The user memory space extends from 0x0000 to 0x1FFF (8K). 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 0x0000, 0x2000 to 0x3FFF and wrap around to 0x2000 (not to 0x0000). Once in configuration memory, the highest bit of the PC stays a '1', thus always pointing to the configuration memory. The only way to point to user program memory is to reset the part and re-enter Program/Verify mode, as described in Section 2.4.

In the configuration memory space, 0x2000-0x200F are physically implemented. However, only locations 0x2000 through 0x2007 are available. Other locations are reserved. Locations beyond 0x200F will physically access user memory (see Figure 2-1).

2.2 Data EEPROM Memory

The EEPROM data memory space is a separate block of high endurance memory that the user accesses using a special sequence of instructions. The amount of data EEPROM memory depends on the device and is shown below in number of bytes.

Device	# of Bytes
PIC16F870	64
PIC16F871	64
PIC16F872	64
PIC16F873	128
PIC16F874	128
PIC16F876	256
PIC16F877	256

The contents of data EEPROM memory have the capability to be embedded into the HEX file.

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

The 256 data memory locations are logically mapped starting at address 0x2100. The format for data memory storage is one data byte per address location, LSB aligned.

2.3 ID Locations

A user may store identification information (ID) in four ID locations. The ID locations are mapped in [0x2000 : 0x2003]. It is recommended that the user use only the four Least Significant bits of each ID location. In some devices, the ID locations read out in an unscrambled fashion after code protection is enabled. For these devices, it is recommended that ID location is written as "11 1111 1000 bbbb" where 'bbbb' is ID information.

In other devices, the ID locations read out normally, even after code protection. To understand how the devices behave, refer to Table 5-1.

To understand the scrambling mechanism after code protection, refer to Section 4.0.

PIC16F87X

TABLE 2-1: PROGRAM MEMORY MAPPING

		2K words		4K words	8K words
2000h	ID Location	0h	Implemented	Implemented	Implemented
2001h	ID Location	1FFh	Implemented	Implemented	Implemented
2002h	ID Location	3FFh	Implemented	Implemented	Implemented
2003h	ID Location	400h	Implemented	Implemented	Implemented
2004h	Reserved	7FFh	Reserved	Implemented	Implemented
2005h	Reserved	800h	Reserved	Implemented	Implemented
2006h	Device ID	BFFh	Reserved	Implemented	Implemented
2007h	Configuration Word	C00h	Reserved	Implemented	Implemented
		FFFh	Reserved	Reserved	Implemented
		1000h	Reserved	Reserved	Implemented
		1FFFh	Reserved	Reserved	Reserved
		2008h	Reserved	Reserved	Reserved
		2100h	Reserved	Reserved	Reserved
		3FFFh	Reserved	Reserved	Reserved

2.4 Program/Verify Mode

The Program/Verify mode is entered by holding pins RB6 and RB7 low, while raising $\overline{\text{MCLR}}$ pin from V_{IL} to V_{IH} (high voltage). In this mode, the state of the RB3 pin does not effect programming. Low voltage ICSP Programming mode is entered by raising RB3 from V_{IL} to V_{DD} and then applying V_{DD} to $\overline{\text{MCLR}}$. 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 and RB7 are Schmitt Trigger Inputs in this mode.

Note: The OSC must not have 72 osc clocks while the device $\overline{\text{MCLR}}$ is between V_{IL} and V_{IH} .

The sequence that enters the device into the Programming/Verify mode places all other logic into the RESET state (the $\overline{\text{MCLR}}$ pin was initially at V_{IL}). This means that all I/O are in the RESET state (high impedance inputs).

The normal sequence for programming is to use the load data command to set a value to be written at the selected address. Issue the begin programming command followed by read data command to verify, and then increment the address.

A device RESET will clear the PC and set the address to 0. The "increment address" command will increment the PC. The "load configuration" command will set the PC to 0x2000. The available commands are shown in Table 2-2.

2.4.1 LOW VOLTAGE ICSP PROGRAMMING MODE

Low voltage ICSP Programming mode allows a PIC16F87X device to be programmed using V_{DD} only. However, when this mode is enabled by a configuration bit (LVP), the PIC16F87X device dedicates RB3 to control entry/exit into Programming mode.

When LVP bit is set to '1', the low voltage ICSP programming entry is enabled. Since the LVP configuration bit allows low voltage ICSP programming entry in its erased state, an erased device will have the LVP bit enabled at the factory. While LVP is '1', RB3 is dedicated to low voltage ICSP programming. Bring RB3 to V_{DD} and then $\overline{\text{MCLR}}$ to V_{DD} to enter programming mode. All other specifications for high voltage ICSP™ apply.

To disable low voltage ICSP mode, the LVP bit must be programmed to '0'. This must be done while entered with High Voltage Entry mode (LVP bit = 1). RB3 is now a general purpose I/O pin.

2.4.2 SERIAL 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 specifications), 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:

2.4.2.1 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 in a "data word," as described above, to be programmed into the configuration memory. A description of the memory mapping schemes of the program memory for normal operation and Configuration mode operation is shown in Figure 2-1. 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 $\overline{\text{MCLR}}$ low (V_{IL}).

2.4.2.2 Load Data for Program Memory

After receiving this command, the chip will load in a 14-bit "data word" when 16 cycles are applied, as described previously. A timing diagram for the load data command is shown in Figure 6-1.

PIC16F87X

2.4.2.3 Load Data for Data Memory

After receiving this command, the chip will load in a 14-bit “data word” when 16 cycles are applied. However, the data memory is only 8-bits wide, and thus, only the first 8-bits of data after the START bit will be programmed into the data memory. It is still necessary to cycle the clock the full 16 cycles in order to allow the internal circuitry to reset properly. The data memory contains up to 256 bytes. If the device is code protected, the data is read as all zeros.

2.4.2.4 Read Data from Program Memory

After receiving this command, the chip will transmit data bits out of the program memory (user or configuration) currently accessed, starting with the second rising edge of the clock input. The 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 6-2.

2.4.2.5 Read Data from Data Memory

After receiving this command, the chip will transmit data bits out of the data memory starting with the second rising edge of the clock input. The RB7 pin will go into Output mode on the second rising edge, and it will revert back to Input mode (hi-impedance) after the 16th rising edge. As previously stated, the data memory is 8-bits wide, and therefore, only the first 8-bits that are output are actual data.

2.4.2.6 Increment Address

The PC is incremented when this command is received. A timing diagram of this command is shown in Figure 6-3.

2.4.2.7 Begin Erase/Program Cycle

A load command must be given before every begin programming command. Programming of the appropriate memory (test program memory, user program memory or data memory) will begin after this command is received and decoded. An internal timing mechanism executes an erase before write. The user must allow for both erase and programming cycle times for programming to complete. No “end programming” command is required.

2.4.2.8 Begin Programming

Note: The Begin Program operation must take place at 4.5 to 5.5 VDD range.

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

This command is similar to the ERASE/PROGRAM CYCLE command, except that a word erase is not done. It is recommended that a bulk erase be performed before starting a series of programming only cycles.

TABLE 2-2: COMMAND MAPPING FOR PIC16F87X

Command	Mapping (MSB ... LSB)						Data	Voltage Range
Load Configuration	X	X	0	0	0	0	0, data (14), 0	2.2V - 5.5V
Load Data for Program Memory	X	X	0	0	1	0	0, data (14), 0	2.2V - 5.5V
Read Data from Program Memory	X	X	0	1	0	0	0, data (14), 0	2.2V - 5.5V
Increment Address	X	X	0	1	1	0		2.2V - 5.5V
Begin Erase Programming Cycle	0	0	1	0	0	0		2.2V - 5.5V
Begin Programming Only Cycle	0	1	1	0	0	0		4.5V - 5.5V
Load Data for Data Memory	X	X	0	0	1	1	0, data (14), 0	2.2V - 5.5V
Read Data from Data Memory	X	X	0	1	0	1	0, data (14), 0	2.2V - 5.5V
Bulk Erase Setup1	0	0	0	0	0	1		4.5V - 5.5V
Bulk Erase Setup2	0	0	0	1	1	1		4.5V - 5.5V

2.5 Erasing Program and Data Memory

Depending on the state of the code protection bits, program and data memory will be erased using different procedures. The first set of procedures is used when both program and data memories are not code protected. The second set of procedures must be used when either memory is code protected. A device programmer should determine the state of the code protection bits and then apply the proper procedure to erase the desired memory.

2.5.1 ERASING NON-CODE PROTECTED PROGRAM AND DATA MEMORY

When both program and data memories are not code protected, they must be individually erased using the following procedures. The only way that both memories are erased using a single procedure is if code protection is enabled for one of the memories. These procedures do not erase the configuration word or ID locations.

Procedure to bulk erase program memory:

1. Execute a Load Data for Program Memory command (000010) with a '1' in all locations (0x3FFF)
2. Execute a Bulk Erase Setup1 command (000001)
3. Execute a Bulk Erase Setup2 command (000111)
4. Execute a Begin Erase/Programming command (001000)
5. Wait 8 ms
6. Execute a Bulk Erase Setup1 command (000001)
7. Execute a Bulk Erase Setup2 command (000111)

Procedure to bulk erase data memory:

1. Execute a Load Data for Data Memory command (000011) with a '1' in all locations (0x3FFF)
2. Execute a Bulk Erase Setup1 command (000001)
3. Execute a Bulk Erase Setup2 command (000111)
4. Execute a Begin Erase/Programming command (001000)
5. Wait 8 ms
6. Execute a Bulk Erase Setup1 command (000001)
7. Execute a Bulk Erase Setup2 command (000111)

2.5.2 ERASING CODE PROTECTED MEMORY

For the PIC16F87X devices, once code protection is enabled, all protected program and data memory locations read all '0's and further programming is disabled. The ID locations and configuration word read out unscrambled and can be reprogrammed normally. The only procedure to erase a PIC16F87X device that is code protected is shown in the following procedure. This method erases program memory, data memory, configuration bits and ID locations. **Since all data within the program and data memory will be erased when this procedure is executed, the security of the data or code is not compromised.**

1. Execute a Load Configuration command (000000) with a '1' in all locations (0x3FFF)
2. Execute Increment Address command (000110) to set address to configuration word location (0x2007)
3. Execute a Bulk Erase Setup1 command (000001)
4. Execute a Bulk Erase Setup2 command (000111)
5. Execute a Begin Erase/Programming command (001000)
6. Wait 8 ms
7. Execute a Bulk Erase Setup1 command (000001)
8. Execute a Bulk Erase Setup2 command (000111)

PIC16F87X

FIGURE 2-1: FLOW CHART - PIC16F87X PROGRAM MEMORY ($2.2V \leq V_{DD} < 5.5V$)

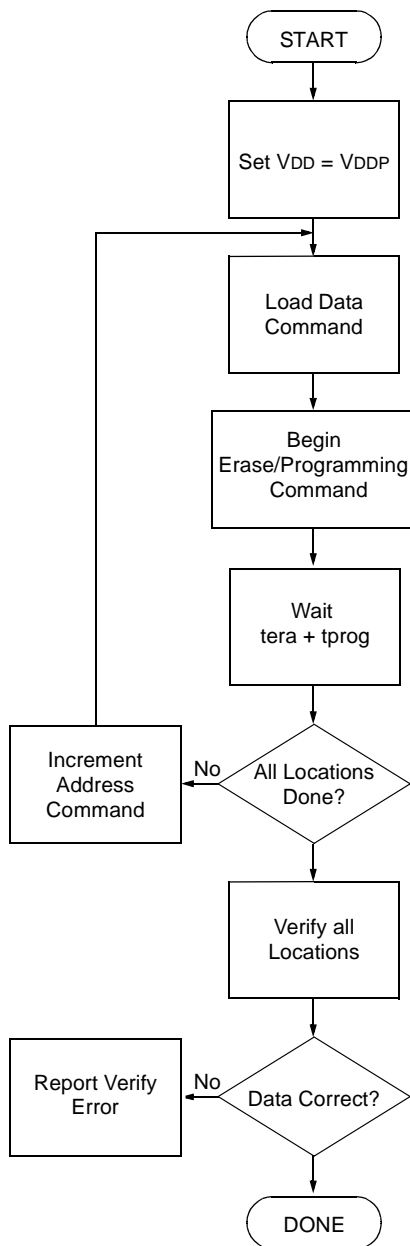


FIGURE 2-2: FLOW CHART – PIC16F87X PROGRAM MEMORY ($4.5V \leq V_{DD} \leq 5.5V$)

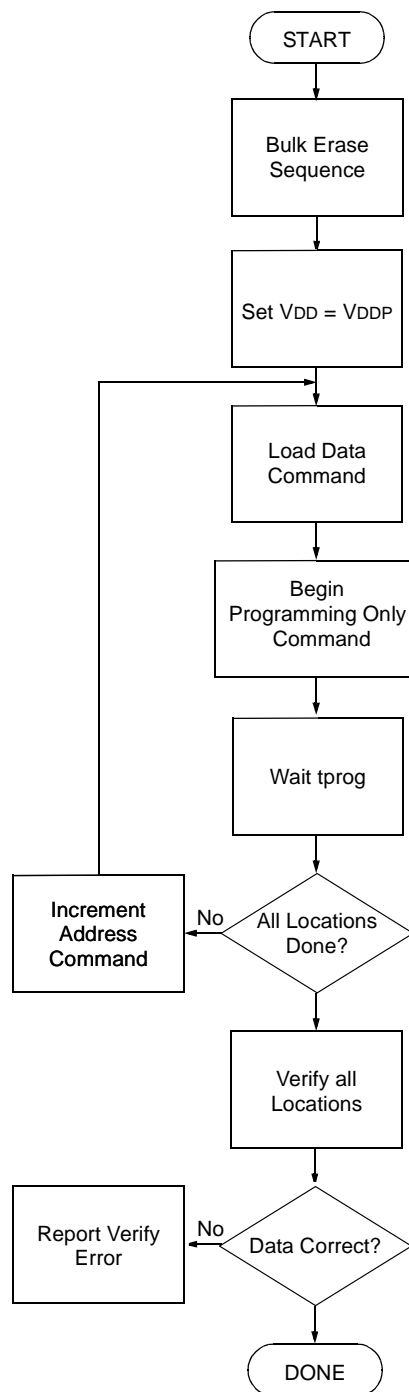


FIGURE 2-3: FLOW CHART – PIC16F87X CONFIGURATION MEMORY ($2.2V \leq V_{DD} < 5.5V$)

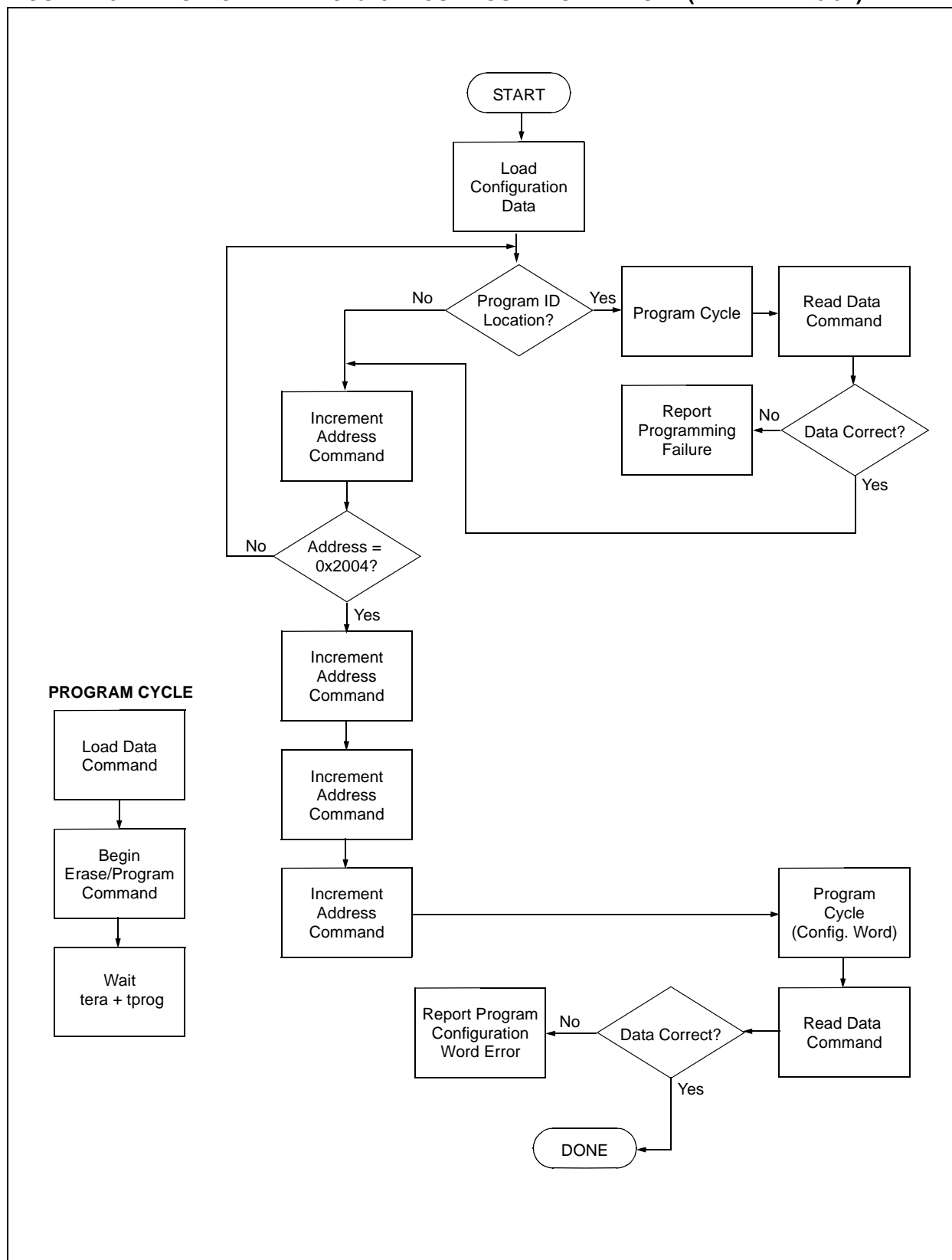
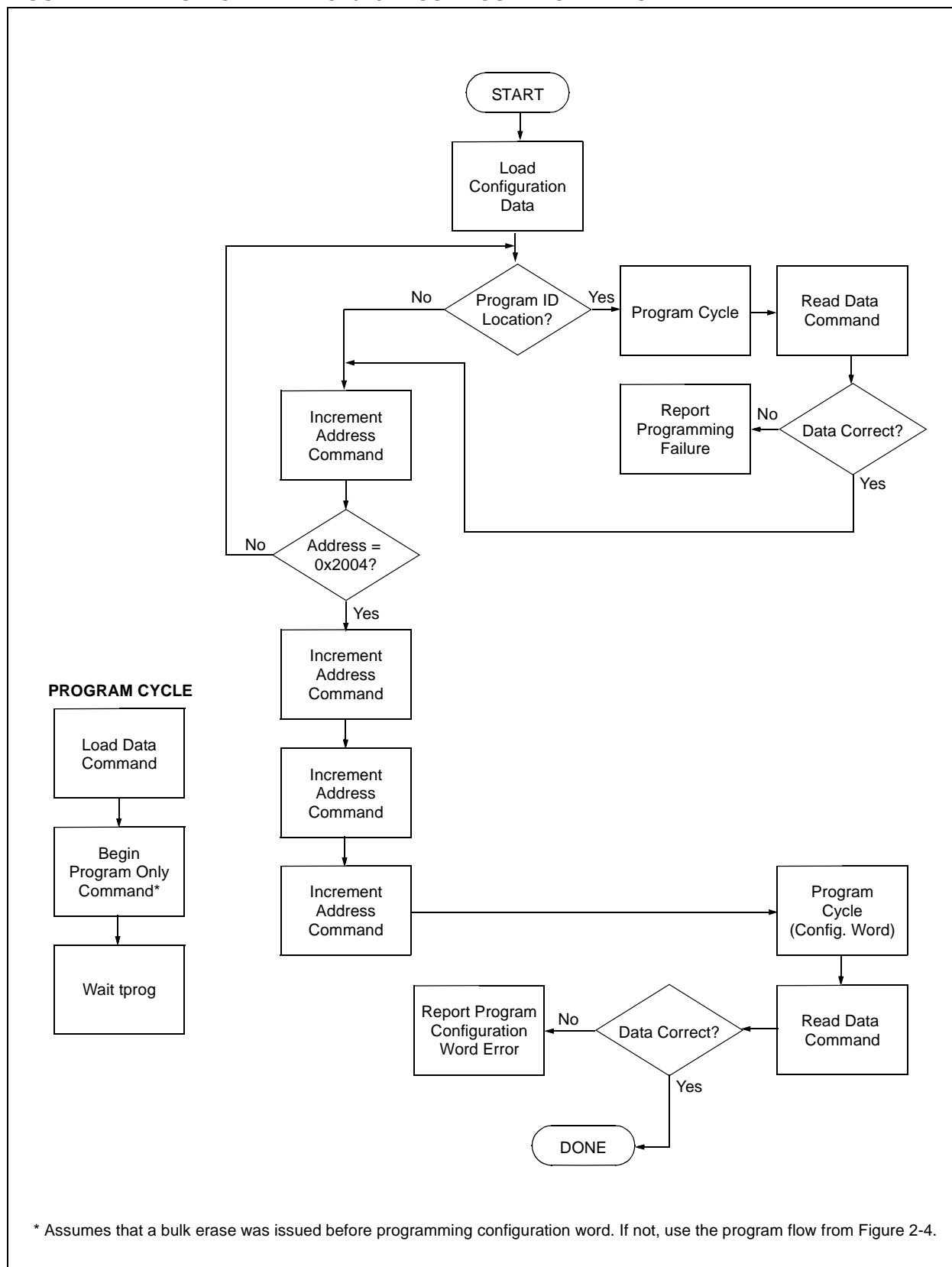


FIGURE 2-4: FLOW CHART - PIC16F87X CONFIGURATION MEMORY



PIC16F87X

3.0 CONFIGURATION WORD

The PIC16F87X has several configuration bits. These bits can be set (reads '0'), or left unchanged (reads '1'), to select various device configurations.

3.1 Device ID Word

The device ID word for the PIC16F87X is located at 2006h.

TABLE 3-1: DEVICE ID VALUE

Device	Device ID Value	
	Dev	Rev
PIC16F870	00 1101 000	x xxxx
PIC16F871	00 1101 001	x xxxx
PIC16F872	00 1000 111	x xxxx
PIC16F873	00 1001 011	x xxxx
PIC16F874	00 1001 001	x xxxx
PIC16F876	00 1001 111	x xxxx
PIC16F877	00 1001 101	x xxxx

REGISTER 3-1: CONFIG: CONFIGURATION WORD FOR PIC16F873/874/876/877 (ADDRESS 2007h)

U-0	U-0	U-0	U-0	U-0	U-0	U-0	R/P-1	U-0	R/P-1	R/P-1	R/P-1	R/P-1	R/P-1
CP1	CP0	RESV	—	WRT	CPD	LVP	BODEN	CP1	CP0	PWRT $\overline{\text{E}}$	WDTE	FOSC1	FOSC0
bit 13							bit 0						

bit 13-12 **CP1:CP0:** FLASH Program Memory Code Protection bits⁽²⁾

bit 5-4

4 K Devices:

- 11 = Code protection off
- 10 = 0F00h to 0FFFh code protected
- 01 = 0800h to 0FFFh code protected
- 00 = 0000h to 0FFFh code protected

8 K Devices:

- 11 = Code protection off
- 10 = 1F00h to 1FFFh code protected
- 01 = 1000h to 1FFFh code protected
- 00 = 0000h to 1FFFh code protected

bit 11 **Reserved:** Set to '1' for normal operation

bit 10 **Unimplemented:** Read as '1'

bit 9 **WRT:** FLASH Program Memory Write Enable bit

- 1 = Unprotected program memory may be written to by EECON control
- 0 = Unprotected program memory may not be written to by EECON control

bit 8 **CPD:** Data EE Memory Code Protection bit

- 1 = Code protection off
- 0 = Data EE memory code protected

bit 7 **LVP:** Low Voltage ICSP Programming Enable bit

- 1 = RB3/PGM pin has PGM function, low voltage programming enabled
- 0 = RB3 is digital I/O, HV on MCLR must be used for programming

bit 6 **BODEN:** Brown-out Reset Enable bit⁽²⁾

- 1 = BOR enabled
- 0 = BOR disabled

bit 3 **PWRT $\overline{\text{E}}$:** Power-up Timer Enable bit

- 1 = PWRT disabled
- 0 = PWRT enabled

bit 2 **WDTE:** 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 PWRT $\overline{\text{E}}$. Ensure the Power-up Timer is enabled any time Brown-out Reset is enabled.

2: All of the CP1:CP0 pairs have to be given the same value to enable the code protection scheme listed.

Legend:

R = Readable bit

P = Programmable bit

U = Unimplemented bit, read as '0'

- n = Value when device is unprogrammed

u = Unchanged from programmed state

PIC16F87X

REGISTER 3-2: CONFIG: CONFIGURATION WORD FOR PIC16F870/871/872 (ADDRESS 2007h)

U-0	U-0	U-0	U-0	U-0	U-0	U-0	R/P-1	U-0	R/P-1	R/P-1	R/P-1	R/P-1	R/P-1
CP1	CP0	RESV	—	WRT	CPD	LVP	BODEN	CP1	CP0	PWRT $\overline{\text{E}}$	WDTE	F0SC1	F0SC0
bit 13										bit 0			

bit 13-12	CP1:CP0: FLASH Program Memory Code Protection bits ⁽²⁾ 11 = Code protection off 10 = Not supported 01 = Not supported 00 = 0000h to 07FFh code protected
bit 5-4	
bit 11	
bit 10	
bit 9	
	WRT: FLASH Program Memory Write Enable bit 1 = Unprotected program memory may be written to by EECON control 0 = Unprotected program memory may not be written to by EECON control
bit 8	CPD: Data EE Memory Code Protection bit 1 = Code protection off 0 = Data EE memory code protected
bit 7	LVP: Low Voltage ICSP Programming Enable bit 1 = RB3/PGM pin has PGM function, low voltage programming enabled 0 = RB3 is digital I/O, HV on MCLR must be used for programming
bit 6	BODEN: Brown-out Reset Enable bit ⁽²⁾ 1 = BOR enabled 0 = BOR disabled
bit 3	PWRT$\overline{\text{E}}$: Power-up Timer Enable bit 1 = PWRT disabled 0 = PWRT enabled
bit 2	WDTE: Watchdog Timer Enable bit 1 = WDT enabled 0 = WDT disabled
bit 1-0	F0SC1:F0SC0: 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 PWRT $\overline{\text{E}}$. Ensure the Power-up Timer is enabled any time Brown-out Reset is enabled.

2: All of the CP1:CP0 pairs have to be given the same value to enable the code protection scheme listed.

Legend:

R = Readable bit

P = Programmable bit

U = Unimplemented bit, read as '0'

- n = Value when device is unprogrammed

u = Unchanged from programmed state

4.0 EMBEDDING THE CONFIGURATION WORD AND ID INFORMATION IN THE HEX FILE

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

Specifically for the PIC16F87X, the EEPROM data memory should also be embedded in the HEX file (see Section 2.2).

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

5.0 CHECKSUM COMPUTATION

Checksum is calculated by reading the contents of the PIC16F87X memory locations and adding up the opcodes, up to the maximum user addressable location, e.g., 0x1FF for the PIC16F87X. 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 PIC16F87X devices is shown in Table 5-1.

The checksum is calculated by summing the following:

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

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

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 5-1: CHECKSUM COMPUTATION

Device	Code Protect	Checksum*	Blank Value	0x25E6 at 0 and max address
PIC16F870	OFF ALL	SUM[0x0000:0x07FFF] + CFGW & 0x3BFF CFGW & 0x3BFF + SUM_ID	0x33FF 0x3FCE	0xFFCD 0x0B9C
PIC16F871	OFF ALL	SUM[0x0000:0x07FFF] + CFGW & 0x3BFF CFGW & 0x3BFF + SUM_ID	0x33FF 0x3FCE	0xFFCD 0x0B9C
PIC16F872	OFF ALL	SUM[0x0000:0x07FFF] + CFGW & 0x3BFF CFGW & 0x3BFF + SUM_ID	0x33FF 0x3FCE	0xFFCD 0x0B9C
PIC16F873	OFF 0x0F00 : 0xFFFF 0x0800 : 0xFFFF ALL	SUM[0x0000:0x0FFF] + CFGW & 0x3BFF SUM[0x0000:0x0EFF] + CFGW & 0x3BFF +SUM_ID SUM[0x0000:0x07FF] + CFGW & 0x3BFF + SUM_ID CFGW & 0x3BFF + SUM_ID	0x2BFF 0x48EE 0x3FDE 0x37CE	0xF7CD 0xFAA3 0xF193 0x039C
PIC16F874	OFF 0x0F00 : 0xFFFF 0x0800 : 0xFFFF ALL	SUM[0x0000:0x0FFF] + CFGW & 0x3BFF SUM[0x0000:0x0EFF] + CFGW & 0x3BFF +SUM_ID SUM[0x0000:0x07FF] + CFGW & 0x3BFF + SUM_ID CFGW & 0x3BFF + SUM_ID	0x2BFF 0x48EE 0x3FDE 0x37CE	0xF7CD 0xFAA3 0xF193 0x039C
PIC16F876	OFF 0x1F00 : 0x1FFF 0x1000 : 0x1FFF ALL	SUM[0x0000:0x1FFF] + CFGW & 0x3BFF SUM[0x0000:0x1EFF] + CFGW & 0x3BFF +SUM_ID SUM[0x0000:0x0FFF] + CFGW & 0x3BFF + SUM_ID CFGW & 0x3BFF + SUM_ID	0x1BFF 0x28EE 0x27DE 0x27CE	0xE7CD 0xDAA3 0xD993 0xF39C
PIC16F877	OFF 0x1F00 : 0x1FFF 0x1000 : 0x1FFF ALL	SUM[0x0000:0x1FFF] + CFGW & 0x3BFF SUM[0x0000:0x1EFF] + CFGW & 0x3BFF +SUM_ID SUM[0x0000:0x0FFF] + CFGW & 0x3BFF + SUM_ID CFGW & 0x3BFF + SUM_ID	0x1BFF 0x28EE 0x27DE 0x27CE	0xE7CD 0xDAA3 0xD993 0xF39C

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

PIC16F87X

6.0 PROGRAM/VERIFY MODE ELECTRICAL CHARACTERISTICS

TABLE 6-1: TIMING REQUIREMENTS FOR PROGRAM/VERIFY MODE

AC/DC CHARACTERISTICS	Standard Operating Conditions (unless otherwise stated)					
	Operating Temperature: $0^{\circ}\text{C} \leq T_A \leq +70^{\circ}\text{C}$ Operating Voltage: $2.2\text{V} \leq V_{DD} \leq 5.5\text{V}$					
Characteristics	Sym	Min	Typ	Max	Units	Conditions/Comments
General						
VDD level for Algorithm 1	VDD	2.2		5.5	V	Limited command set (See Table 2-2)
VDD level for Algorithm 2	VDD	4.5		5.5	V	All commands available
High voltage on MCLR for high voltage programming entry	VIHH	$V_{DD} + 3.5$		13.5	V	
Voltage on MCLR for low voltage ICSP programming entry	VIH	2.2		5.5	V	
MCLR rise time (V_{SS} to V_{HH}) for Test mode entry	tVHHR			1.0	μs	
(RB6, RB7) input high level	VIH1	$0.8 V_{DD}$			V	Schmitt Trigger input
(RB6, RB7) input low level	VIL1	$0.2 V_{DD}$			V	Schmitt Trigger input
RB<7:6> setup time before MCLR \uparrow	tset0	100			ns	
RB<7:6> hold time after MCLR \uparrow	thld0	5			μs	
RB3 setup time before MCLR \uparrow	tset2	100			ns	
Serial Program/Verify						
Data in setup time before clock \downarrow	tset1	100			ns	
Data in hold time after clock \downarrow	thld1	100			ns	
Data input not driven to next clock input (delay required between command/data or command/command)	tdly1	1.0			μs	
Delay between clock \downarrow to clock \uparrow of next command or data	tdly2	1.0			μs	
Clock \uparrow to data out valid (during read data)	tdly3	80			ns	
Erase cycle time	tera		2	4	ms	
Programming cycle time	tprog		2	4	ms	

FIGURE 6-1: LOAD DATA COMMAND $\overline{\text{MCLR}} = V_{\text{IH}} \text{ (PROGRAM/VERIFY)}$

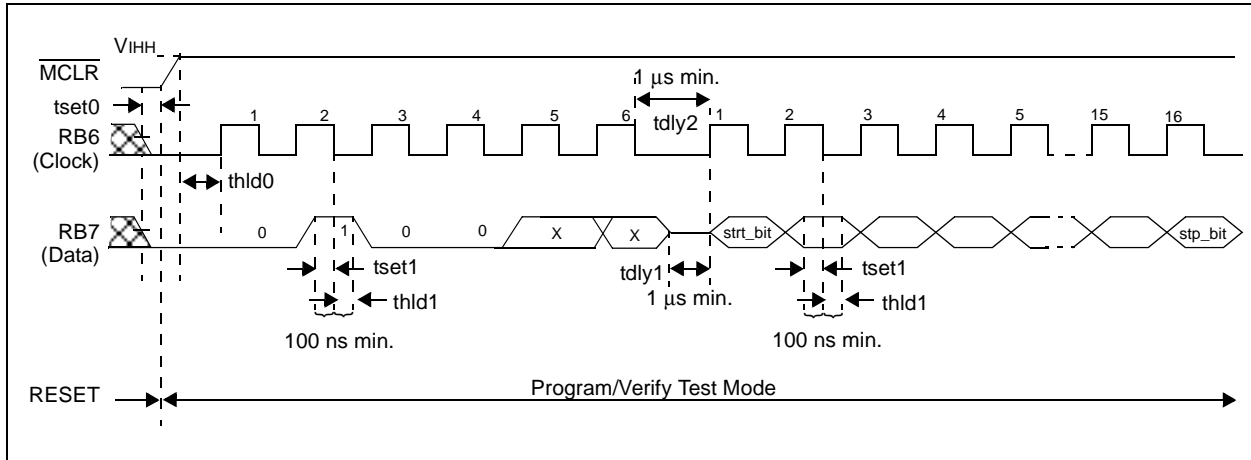


FIGURE 6-2: READ DATA COMMAND $\overline{\text{MCLR}} = V_{\text{IH}} \text{ (PROGRAM/VERIFY)}$

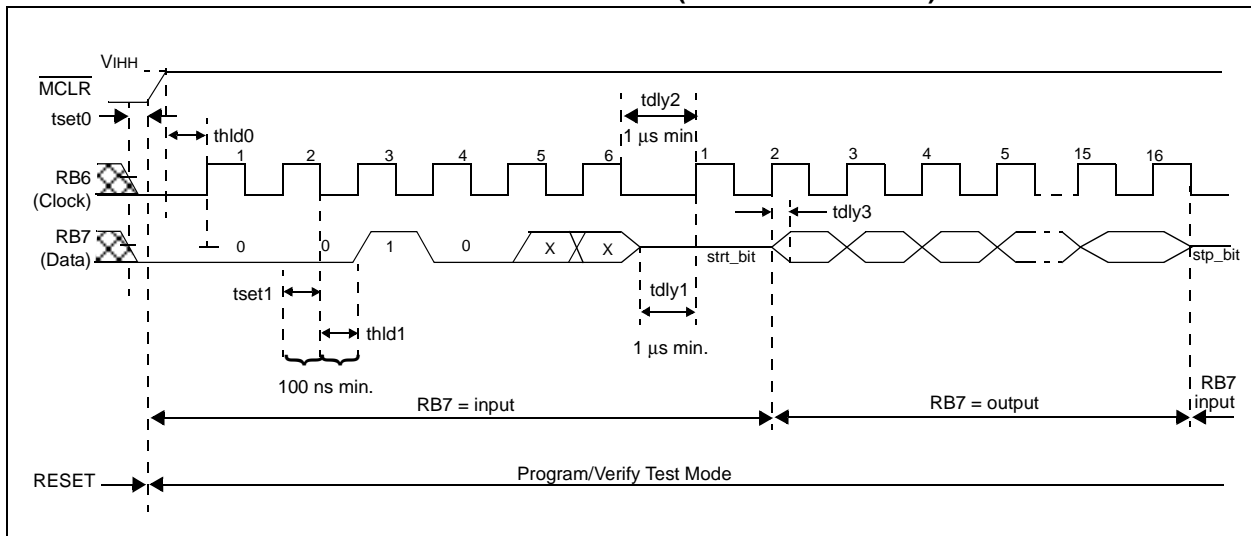
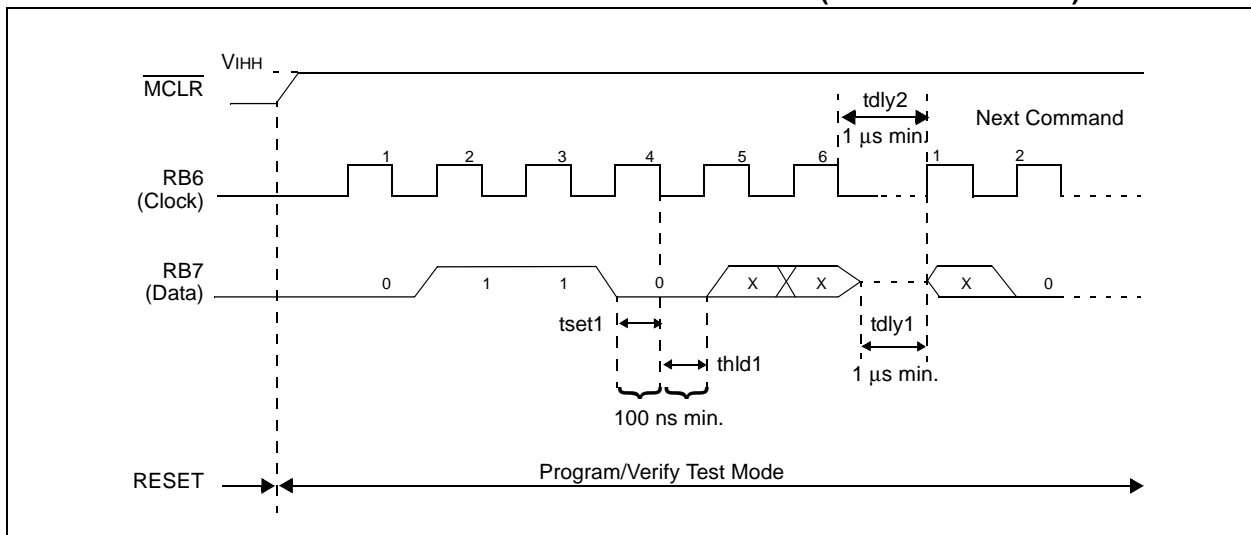


FIGURE 6-3: INCREMENT ADDRESS COMMAND $\overline{\text{MCLR}} = V_{\text{IH}} \text{ (PROGRAM/VERIFY)}$



PIC16F87X

FIGURE 6-4: LOAD DATA COMMAND $\overline{\text{MCLR}} = V_{DD}$ (PROGRAM/VERIFY)

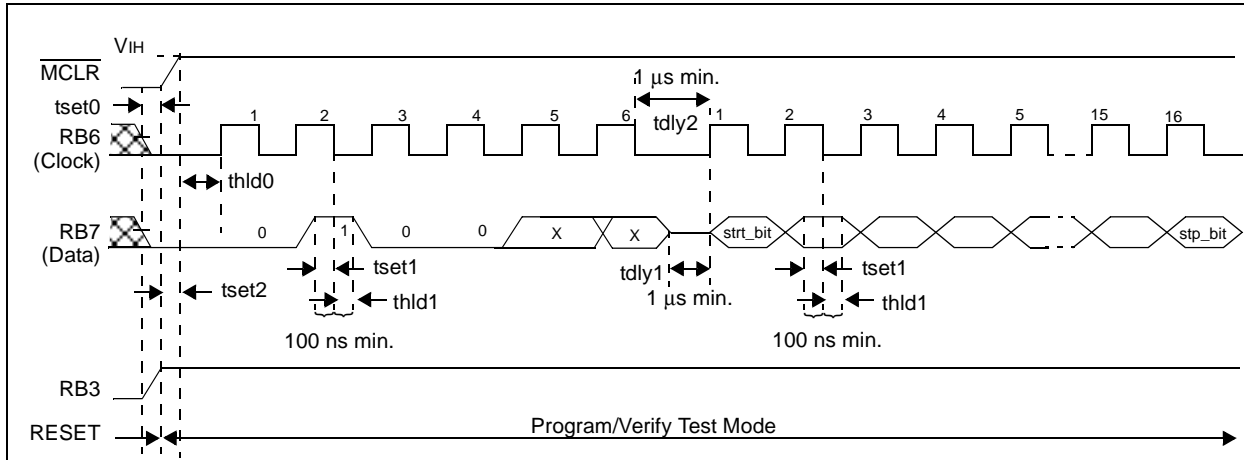


FIGURE 6-5: READ DATA COMMAND $\overline{\text{MCLR}} = V_{DD}$ (PROGRAM/VERIFY)

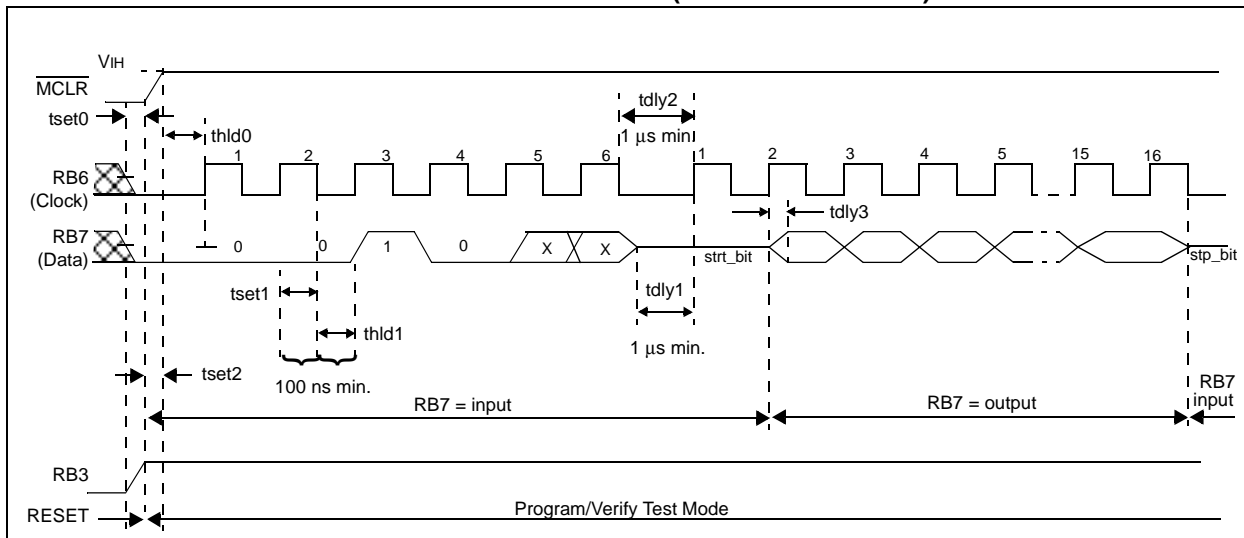
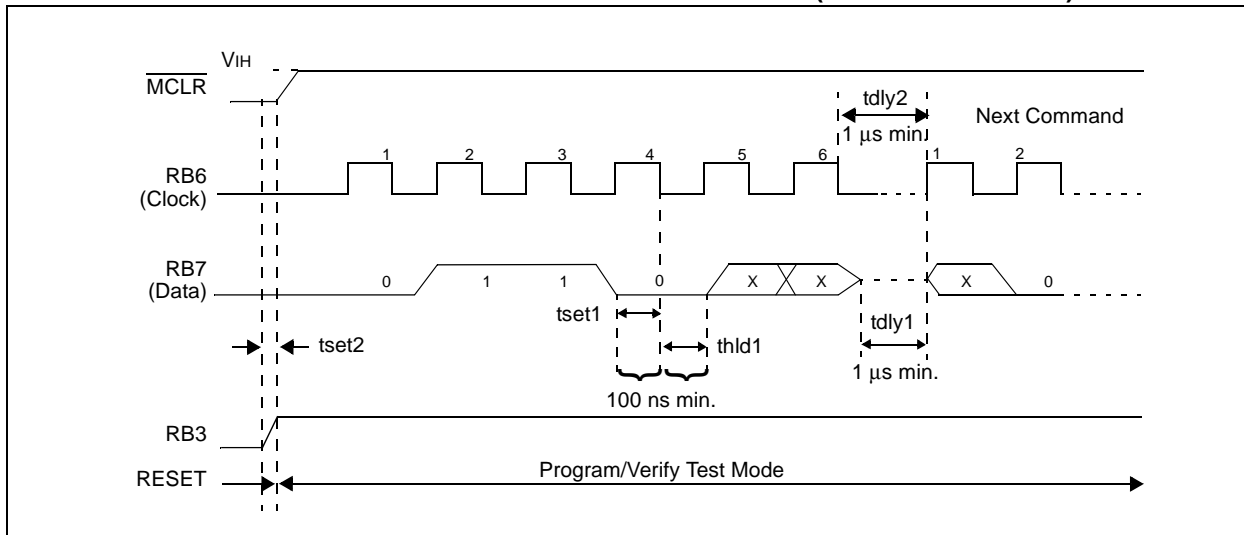


FIGURE 6-6: INCREMENT ADDRESS COMMAND $\overline{\text{MCLR}} = V_{DD}$ (PROGRAM/VERIFY)



Note the following details of the code protection feature on PICmicro® MCUs.

- The PICmicro family meets the specifications contained in the Microchip Data Sheet.
- Microchip believes that its family of PICmicro microcontrollers is one of the most secure products of its kind on the market today, when used in the intended manner and under normal conditions.
- There are dishonest and possibly illegal methods used to breach the code protection feature. All of these methods, to our knowledge, require using the PICmicro microcontroller in a manner outside the operating specifications contained in the data sheet. The person doing so may be engaged in theft of intellectual property.
- Microchip is willing to work with the customer who is concerned about the integrity of their code.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of their code. Code protection does not mean that we are guaranteeing the product as “unbreakable”.
- Code protection is constantly evolving. We at Microchip are committed to continuously improving the code protection features of our product.

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

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

Trademarks


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

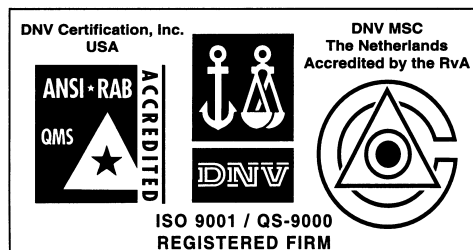
dsPIC, ECONOMONITOR, FanSense, FlexROM, fuzzyLAB, In-Circuit Serial Programming, ICSP, ICEPIC, microPort, Migratable Memory, MPASM, MPLIB, MPLINK, MPSIM, MXDEV, PICC, PICDEM, PICDEM.net, rPIC, Select Mode and Total Endurance are trademarks of Microchip Technology Incorporated in the U.S.A.

Serialized Quick Term Programming (SQTP) is a service mark of Microchip Technology Incorporated in the U.S.A.

All other trademarks mentioned herein are property of their respective companies.

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

 Printed on recycled paper.



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.

