14 Programming the Nonvolatile Memory

CHAPTER 14
PROGRAMMING THE NONVOLATILE MEMORY

The 8XC196KC/KD contains One-Time-Programmable Read-Only Memory (OTPROM), a version of EPROM. The 8XC196KC has 16 Kbytes at locations 2000H–5FFFH. The 8XC196KD has 32 Kbytes at locations 2000H–9FFFH. You may program the OTPROM yourself, or have the factory do it for you in the guise of a ROM product.

This chapter contains procedures and guidelines to help you program the device. You have several options. You may use Auto or Slave Programming mode or Run-Time Programming to help you program the OTPROM. Later, you may select ROM-Dump mode to verify the OTPROM contents. You may use the UPROM or PCCB Programming mode to establish part of the security scheme. This chapter describes the programming modes, how to select them, and how to use them.

14.1. SPECIFICATIONS

When you select a programming mode (except Run-Time) the device responds by executing an internal algorithm, which is located in the test ROM. In general, the internal algorithm performs the programming function by prompting the outside world for information.

Your programming-mode selection determines which algorithm the internal test-ROM code executes. To execute programs properly, the device must have these minimum hardware connections: XTAL1 driven, unused input pins strapped, and power and grounds applied. Follow the data sheet operating-condition specifications.

Later in this chapter, a section about each programming mode discusses specific requirements for that mode. The sections include figures illustrating the circuit diagrams we recommend.

14.2. PROGRAMMING MODES

The 8XC196KC/KD supports three methods of programming the OTPROM program memory: Auto Programming mode, Slave Programming mode, and Run-Time Programming.

- Auto Programming mode enables the 8XC196KC/KD to program itself from an external EPROM, without a specialized programmer. This mode allows you to construct a prototype application-programming circuit.
- Slave Programming mode supports programming with a specialized programmer. While using this programming mode, you can program and verify single or multiple words in the OTPROM.

- Run-Time Programming allows you to program individual OTPROM locations during normal program execution, under complete software control. You do not have to place the device into programming mode to perform this type of programming.

ROM-Dump mode and UPROM and PCCB Programming modes support other aspects of programming the OTPROM.

- ROM-Dump mode allows verification of the OTPROM contents by writing the entire OTPROM array to external memory.
- UPROM Programming mode enables programming of two unerasable bits that prevent the bus controller from fetching external instructions (DEI bit) or external data (DED bit).
- PCCB Programming mode enables programming of the Programming Chip Configuration Byte (PCCB). The PCCB is a non-memory-mapped location in the test ROM that establishes hardware security during programming modes.

14.2.1. Programming Mode Selection

Use the PMODE signal to select a programming mode. Table 14-1 describes the PMODE function and lists the PMODE values and programming modes.

Place the 8XC196KC/KD into programming mode by applying V_{PP} voltage (typically +12.5V) to EA# during the rising edge of RESET#. (The "Power-Up and Power-Down Sequences" section on page 14-6 contains more details.)

Table 14-1. PMODE Description and Values

PMODE Description	PMODE Value	Programming Mode
The PMODE value selects an 8XC196KC/KD programming mode (one of five, not including Run-Time Programming). Pins P0.4–P0.7 define the hex value of PMODE. The five programming modes are Slave, ROM-Dump, UPROM, Auto, and PCCB. PMODE is sampled on the rising edge of RESET#. You must reset the device to switch modes.	0H–4H	Reserved
	5H	Slave Programming
	6H	ROM Dump
	7H–8H	Reserved
	9H	UPROM Programming
	0AH–0BH	Reserved
	0CH	Auto Programming
	0DH	PCCB Programming
	0EH–0FH	Reserved

14.3. PROGRAMMING MODE PINS

To support the various programming modes, some device pins have new functions. Table 14-2 describes the new pin functions. Figure 14-1 shows the renamed pins.

Table 14-2. Programming-Mode Pin-Function Description

Function Name	Type	Programming Mode	Description
AINC# P2.4	I	Slave	Auto Increment. During Slave Programming mode, an active-low input enables the auto-increment mode. (Auto increment allows reading or writing of sequential OTPROM locations, without requiring address transactions across the PBUS for each read or write.)
CPVER P2.6	O	Slave	Cumulative Program Verification. During Slave Programming mode, a high signal indicates that all locations programmed correctly.
EA# Programming Mode Select	I	All (except Run-Time)	External Access. Controls program mode entry as follows. If EA# is at V_{PP} voltage on the rising edge of RESET#, the device enters programming mode. EA# is sampled and latched only on the rising edge of RESET#. Changing the level of EA# after reset has no effect.
PACT# P2.7	O	Auto	Programming Active. During Auto Programming mode, a low signal indicates programming is in progress and a high signal indicates programming is complete.
PALE# P2.1	I	Slave	Programming ALE Input. During Slave Programming mode, a falling edge causes the 8XC196KC/KD to read address/command information on Ports 3 and 4.
PBUS PORTS 3, 4	I/O	All (except Run-Time)	Address/Command/Data Bus. Used as a bidirectional port with open-drain outputs to pass commands, addresses, and data to or from the 8XC196KC/KD. Used as a regular system bus to access external memory during ROM-Dump and Auto Programming mode. Slave Programming mode requires pull-ups to V_{CC}.
PMODE P0.4-7	I	All (except Run-Time)	Programming Mode Select. Determines which OTPROM programming algorithm is performed. PMODE is sampled after a chip reset and must be static while the part is operating. (Table 14-1 lists the PMODE values and programming modes.)
PROG# P2.2	I	Slave	Programming. During Slave Programming mode, a falling edge latches data on PBUS and begins programming and a rising edge ends programming.
PVER P2.0	O	Slave Auto	Programming Verification. During Slave or Auto Programming mode, a high output signal indicates successful programming of a location and a low signal indicates a detected error.
V_{PP}	I	All	Programming Voltage. During programming, the V_{PP} pin is typically at +12.5V (V_{PP} voltage). Data sheets list the exact values for different devices. Exceeding the maximum V_{PP} voltage specification may result in device damage.

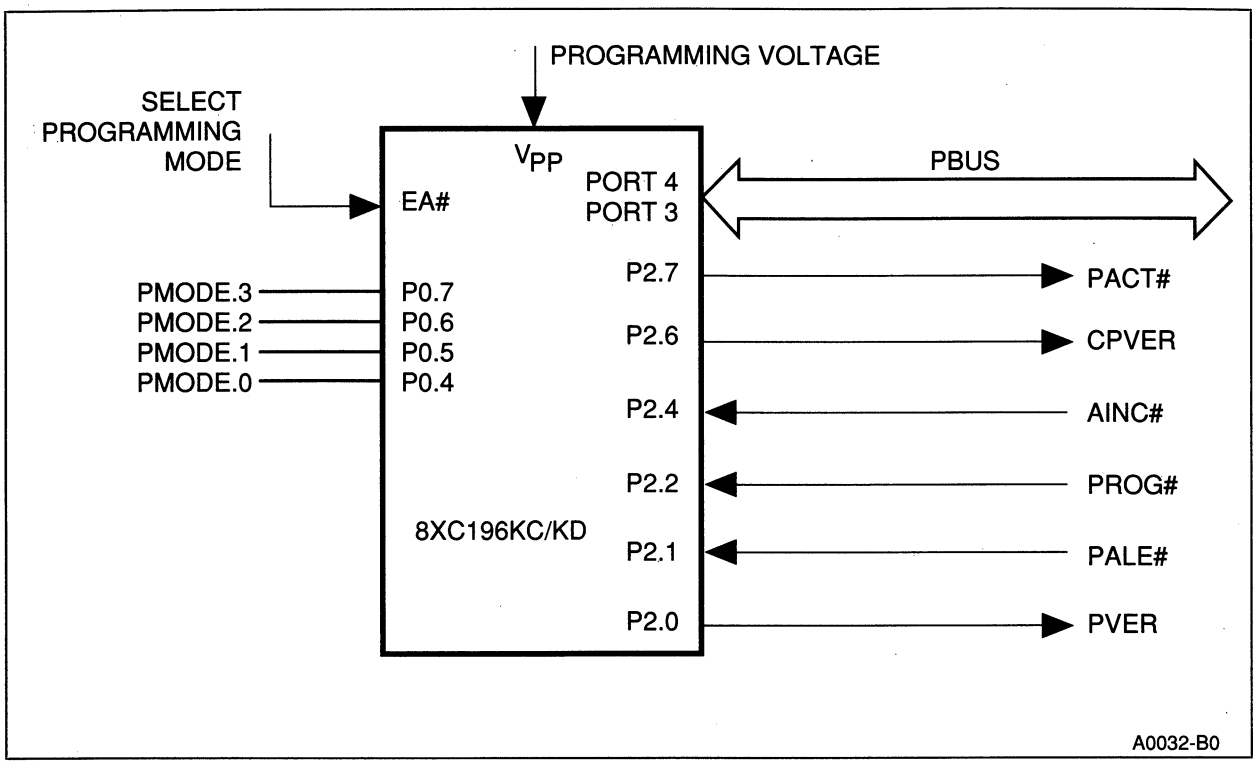


Figure 14-1. Programming-Mode Pin Functions

14.4. MEMORY MAP

The program memory, locations 2080H–5FFFH (8XC196KC) and 2080H–9FFFH (8XC196KD), and the special-purpose memory, locations 2000H–207FH, are user-specified. Figure 14-2 illustrates the special-purpose memory map. Table 14-3 compares the 8XC196KC and 8XC196KD special-purpose memory addresses and provides starting and ending addresses in both hexadecimal and decimal. (Chapter 4, "Memory Partitions," contains a complete set of 8XC196KC/KD memory maps and address tables).

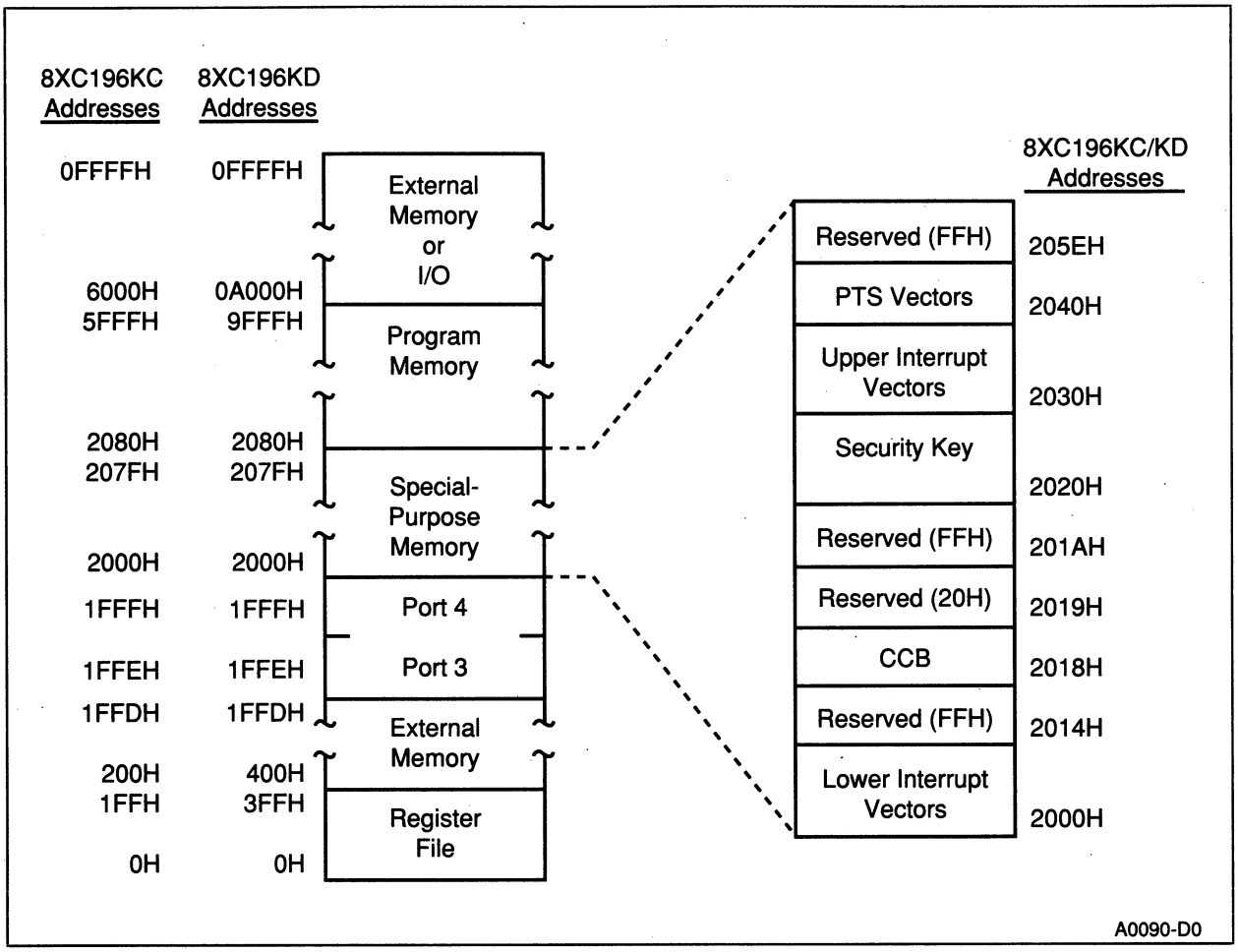


Figure 14-2. 8XC196KC/KD Special-Purpose Memory Map

Table 14-3. 8XC196KC/KD Special-Purpose Memory Addresses

Description	8XC196KC/KD Address Range	
	Hexadecimal	Decimal
Reserved (must contain 0FFH)	205EH–207FH	8286–8319
PTS Vectors	2040H–205DH	8256–8285
Upper Interrupt Vectors	2030H–203FH	8240–8255
Security Key	2020H–202FH	8224–8239
Reserved (must contain 0FFH)	201AH–201FH	8218–8223
Reserved (must contain 20H)	2019H	8217
CCB	2018H	8216
Reserved (must contain 0FFH)	2014H–2017H	8212–8215
Lower Interrupt Vectors	2000H–2013H	8192–8211

14.5. POWER-UP AND POWER-DOWN SEQUENCES

When you are ready to begin programming, follow these procedures to avoid device damage.

WARNINGS

Follow these rules or permanent device damage will result:

- Do not apply voltage to V_{PP} while V_{CC} is low.
- Do not exceed the maximum limit on V_{PP}.
- The power supplies to the V_{CC}, V_{PP}, EA# and RESET# pins must be well regulated and free of glitches and spikes.
- All V_{SS} pins must be well grounded.

14.5.1. Power-Up Sequence

1. Hold the RESET# pin low while V_{CC} stabilizes. Allow V_{PP} and EA# to float during this time.
2. After V_{CC} and the oscillator stabilize, while continuing to hold the device in reset, apply V_{PP} voltage to EA#. After EA# stabilizes, apply V_{PP} voltage to the V_{PP} pin. (Data sheets list the exact V_{PP} voltage for different devices.)
3. Set the PMODE value to select a programming algorithm. (Table 14-1 on page 14-2 identifies the PMODE value for each programming mode.)
4. Bring the RESET# pin high.
5. Complete the selected programming algorithm.

NOTE

The individual Programming Mode sections (later in this chapter) further describe the algorithms.

14.5.2. Power-Down Sequence

1. Assert the RESET# signal (RESET# = 0). RESET# must be held low throughout the power-down sequence.
2. Remove the V_{PP} voltage from the V_{PP} pin and allow the pin to float.
3. Remove the V_{PP} voltage from the EA# pin and allow the pin to float.
4. Turn off the V_{CC} supply and allow time for it to reach 0 volts.

14.6. MEMORY PROTECTION

NOTE

The developers have made a substantial effort to provide an adequate program protection scheme. However, Intel cannot and does not guarantee that these protection methods will always prevent unauthorized access.

Hardware and software protection are each part of the memory protection scheme. The hardware protection prevents OTPROM or external reads or writes in the following manner. Clearing the Chip Configuration Register security-lock bits, CCR.6 (LOC0) and CCR.7 (LOC1), disables OTPROM reads or writes. An attempt to write causes the bus controller to cycle through the write sequence; however, a write does not take place. An attempt to read does not access the internal memory. Setting the UPROM Special Function Register bits, USFR.2 (DED) and USFR.3 (DEI), disables the bus controller from external instruction or data access. If the bus controller attempts an external instruction or data fetch, a device reset occurs.

The software protection prevents unauthorized programming or verifying of the OTPROM. Software support of the memory protection scheme works in the following manner. A software security-key mechanism restricts access to internal memory. If you cleared either CCB.6 or CCB.7 and the device is in programming mode, the programming-mode algorithm requires a security key verification before it executes. If the verification fails, the device enters a program loop that continues until a reset occurs.

14.6.1. CCR Security-Lock Bits

The Chip Configuration Register (CCR) establishes the bus structure and other options of the device during programming. (Chapter 13, "Interfacing with External Memory," and Appendix C, "8XC196KC/KD Registers," contain information about the Chip Configuration Register.) Figure 14-3 illustrates the Chip Configuration Register, including the security-lock bits.

If the 8XC196KC/KD is in normal program execution mode, the reset sequence loads the CCR from location 2018H, the Chip Configuration Byte (CCB). This determines security during normal program execution. Clearing CCB.6 or CCB.7 affects Run-Time Programming and any attempts to access the internal OTPROM from outside the device.

If the device is in programming mode, the reset sequence loads the CCR from a non-memory-mapped test-ROM location (accessible only in programming mode), the Programming Chip Configuration Byte (PCCB). This establishes the hardware security during the programming modes. Clearing these bits affects OTPROM access in all the programming modes. The default value of the PCCB is 0FFH, which disables the hardware security in programming mode.

Figure 14-4 illustrates the CCB and PCCB relationship to the Chip Configuration Register.

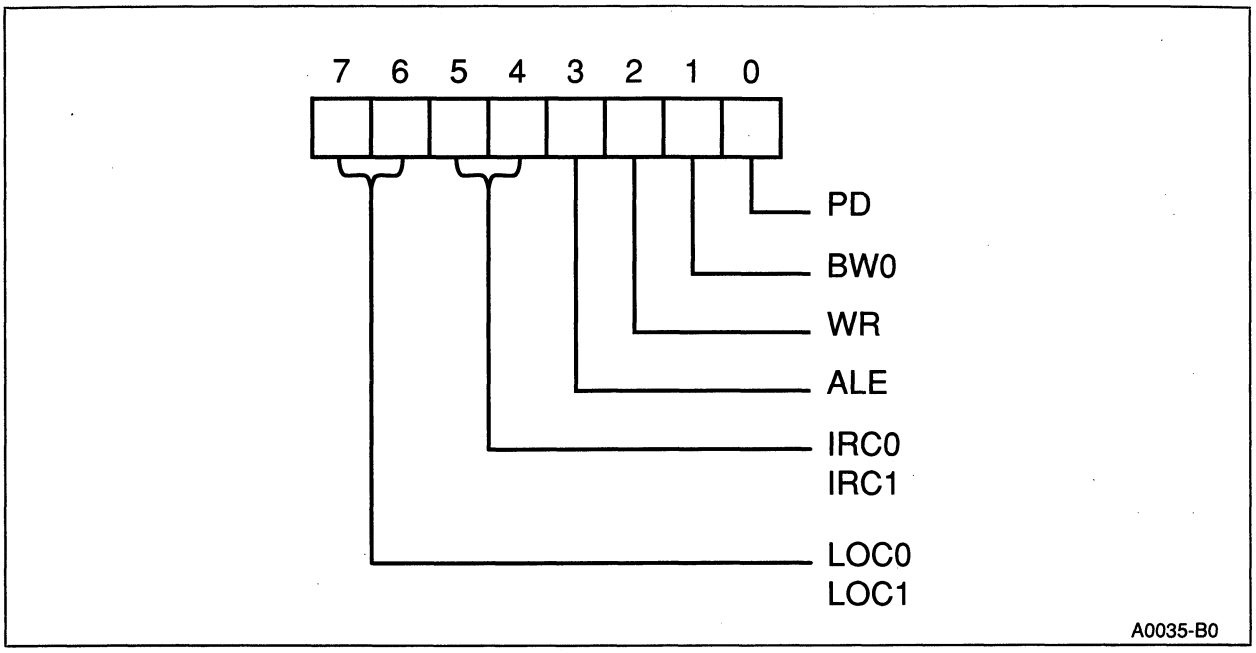


Figure 14-3. Chip Configuration Register

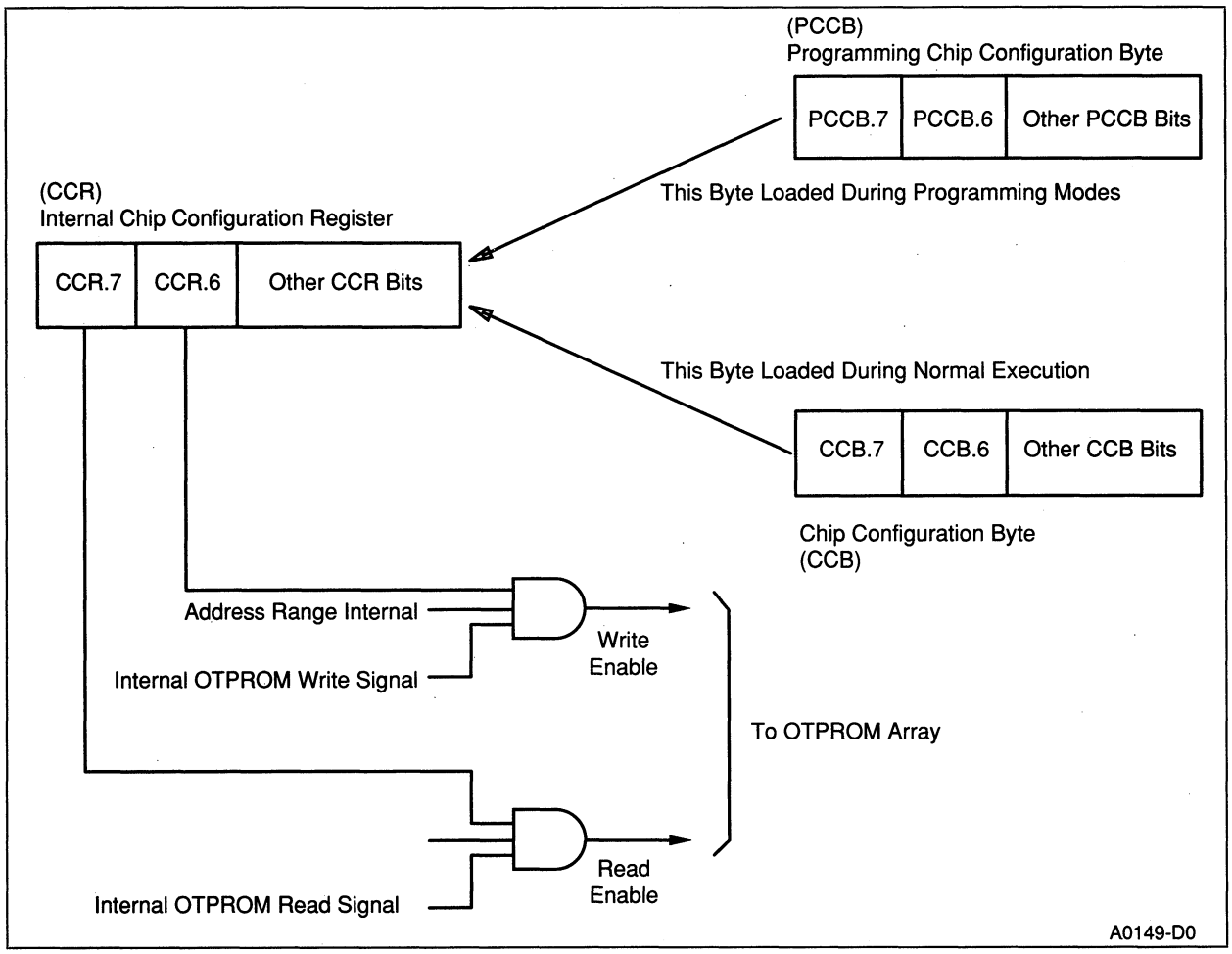


Figure 14-4. Chip Configuration Byte

Clear CCR.6 to disable writing and prevent further programming of the OTPROM array, 2000H–5FFFH (8XC196KC) or 2000H–9FFFH (8XC196KD). An attempt to write causes the bus controller to cycle through the write sequence; however, a write does not take place.

Clear CCR.7 to disable reading and prevent a program in external memory space from accessing the contents of the OTPROM. An attempt to read does not access the internal memory.

A data read can occur if the slave program counter (slave PC) is in the range of 2000–5FFFH (8XC196KC) or 2000–9FFFH (8XC196KD). The slave PC may be as much as 4 bytes ahead of the actual program execution. For this reason, an instruction located after 5FFAH (8XC196KC) or 9FFAH (8XC196KD) may not access protected memory. The "Memory Controller" section of Chapter 2 discusses the slave PC. The interrupt vectors and CCBs are not read protected because interrupts can occur even when executing out of external memory.

If you enable the security-lock bits, CCR.6 or CCR.7, some programming modes require security-key verification before executing and some modes will not execute. See Table 14-4 and each programming section for more information about the effects of enabling the lock bits. Table 14-4 summarizes the protection options.

Table 14-4. Protection Options

PCCB.6	CCB.6	Write Protection Options
1	1	No protection.
1	0	Run-Time Programming not allowed. Programming modes allowed after security-key verification.
0	1	Programming modes not allowed. Run-Time Programming allowed.
0	0	No programming allowed.
PCCB.7	**CCB.7**	**Read Protection Options**
1	1	No protection.
1	0	Programming modes allowed after security-key verification. Reads of internal OTPROM are not allowed if program execution is external.
0	1	Programming allowed.
0	0	Programming modes allowed after security-key verification. Reads of internal OTPROM are not allowed if program execution is external.

14.6.1.1. PCCB PROGRAMMING MODE

You can program the CCB in the same manner that you program any other OTPROM location, using Auto, Slave, or Run-Time Programming mode. You can program the PCCB during Slave Programming or in the PCCB Programming mode. If you need to program only the PCCB, you may use the PCCB Programming mode.

Drive PALE# low to begin PCCB Programming. The algorithm programs the Port 3 data into the PCCB. When programming ends, a verify occurs. The device drives PVER high if the bytes programmed correctly and low if they did not. You can pulse PALE# to repeat programming. Table 14-5 shows the function of the programming-mode signals during PCCB Programming mode.

Table 14-5. PCCB and UPROM Programming Mode Pin Functions

Function Name	Type	Description
PALE# (P2.1)	I	Programming ALE. Clear PALE# to begin reading from Port 3 into the CCB and PCCB.
PVER (P2.0)	O	Program Verification. Updated after each programming pulse is issued. A high signal indicates successful programming and a low signal indicates a detected error.

For PCCB Programming mode, we recommend the circuit illustrated in Figure 14-5.

Enter the PCCB Programming mode by using the standard power-up sequence (page 14-6), after setting the following values:

PMODE = 0DH;

Port 4 = 0FFH;

Port 3 = the data for PCCB.

The algorithm reads the value on Port 3/4 then programs the PCCB by sending 5 or 25 separate 100 µs programming cycles to the appropriate internal location.

Complete the procedure by following the standard power-down sequence (page 14-6).

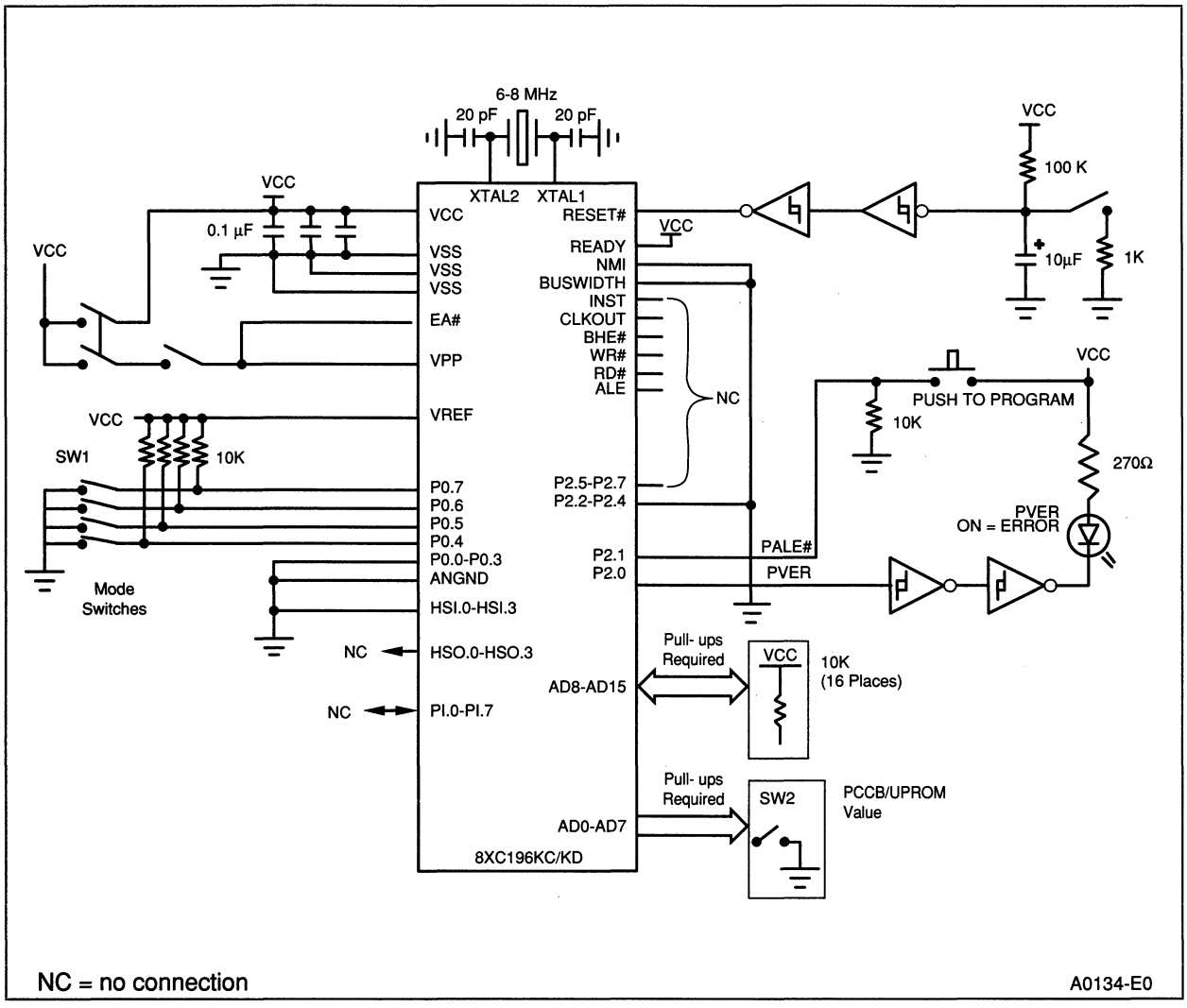


Figure 14-5. PCCB and UPROM Programming Mode Circuit

14.6.2. UPROM Security Bits

The 8XC196KC/KD includes two Unerasable-PROM (UPROM) bits implemented as an additional security feature. Figure 14-6, the UPROM Special Function Register (USFR), illustrates that USFR.2 and USFR.3, Disable External Data fetch and Disable External Instruction fetch (DED and DEI) are the UPROM security bits. Program these bits by writing a one to the appropriate location, using Slave Programming mode or UPROM Programming mode. Figure 14-5 illustrates the circuit we recommend for UPROM Programming mode.

WARNING

Programming these bits makes dynamic failure analysis impossible. For this reason, you cannot return a device to Intel for failure analysis if it has programmed UPROM bits.

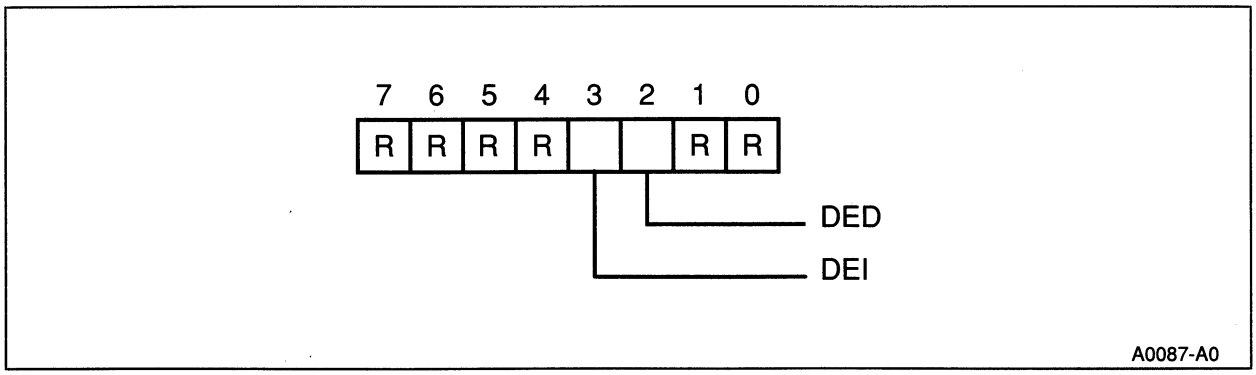


Figure 14-6. UPROM Special Function Register (USFR)

Table 14-6. UPROM Programming Memory Map

To set this bit	While in this Programming Mode	Write this value	To this location
DEI	Slave	08H	0718H
DEI	UPROM	08H	Port 3
DED	Slave	04H	0758H
DED	UPROM	04H	Port 3
DEI and DED	UPROM	0CH	Port 3

Setting USFR.2 (DEI) prevents the bus controller from executing external instruction fetches. If you enable this feature, the prefetch queue in the bus controller prevents code execution from the last four bytes of internal memory. Any attempt to load the slave program counter with an external address causes the device to reset itself. The automatic reset also gives extra protection against runaway code.

Setting USFR.3 (DED) prevents the bus controller from executing external data reads and writes. Any attempt to access data through the bus controller causes the 8XC196KC/KD to reset itself.

You can verify a UPROM bit to make sure it programmed. You cannot erase UPROM bits. For this reason Intel cannot test the bits before shipment. Intel does test the features enabled by the UPROM bit so the only undetectable defects are those (highly unlikely) defects within the UPROM cells themselves.

14.6.2.1. UPROM PROGRAMMING MODE

You can program the UPROM during Slave Programming or in the UPROM Programming mode. You may use the UPROM Programming mode if you need to program only the UPROM.

Drive PALE# low to begin UPROM Programming. The algorithm programs the Port 3 data into the PCCB. When programming ends, a verify occurs. The device drives PVER high if the bytes programmed correctly and low if they did not. You can pulse PALE# to repeat programming. Table 14-5 shows the function of the programming-mode signals during UPROM Programming mode.

For UPROM Programming mode, we recommend the circuit illustrated in Figure 14-5.

Enter the UPROM Programming mode by using the standard power-up sequence (page 14-6), after setting the following values:

PMODE = 9H;

Port 4 = 0FFH;

Port 3 = the data for UPROM.

The algorithm reads the value on Port 3/4 then programs the UPROM by sending 5 or 25 separate 100 µs programming cycles to the appropriate internal location.

Complete the procedure by following the standard power-down sequence (page 14-6).

14.6.3. Security Key

The security key is a 128-bit number located in internal memory at locations 2020H–202FH. Program a password, your security key, into these locations. The Auto Programming, Slave Programming, and ROM-Dump mode sections contain information about how each mode verifies the key. In each instance, if you do not match the security key the device enters an endless loop and must be reset.

14.6.3.1. PROGRAMMING THE SECURITY KEY

If a glitch or reset occurs during programming of the security key, an unknown security key might accidentally be written, rendering the device inaccessible for further programming. To prevent this possibility during Slave Programming, program the rest of the OTPROM array before you program the CCB security-lock bits (CCB.6 and CCB.7).

To prevent the possibility of accidentally writing an unknown security key during Auto Programming, follow these sequences. The Auto programming algorithm skips all locations with a 0FFH value. For this reason, the first sequence programs all of the array except the CCB; the second sequence programs the CCB.

1. Follow these steps to program the entire OTPROM array, except the CCB. (These steps program, but do not enable, the security key.)

   a. Write 0FFH to the external CCB location (4018H = 0FFH).

   b. Place the Programming Pulse Width (PPW) in external location 2014H–2015H.

   c. Place the user code in the appropriate external EPROM locations (see Table 14-8).

   d. Initiate the Auto Programming algorithm, Figure 14-9. The algorithm skips the CCB and programs the rest of the OTPROM array.

2. Follow these steps to program only the CCB (location 2018H). (These steps enable the security key.)

   a. Place the appropriate CCB value in external location 4018H.

   b. Place the Programming Pulse Width (PPW) in external location 2014H–2015H.

   c. Write 0FFH to all other external EPROM locations.

   d. Initiate the Auto Programming algorithm, Figure 14-9. This time the algorithm programs the CCB and skips the rest of the OTPROM array.

NOTE

If you are absolutely sure that a glitch will not occur, you can program the lock bits (CCB.6 and CCB.7) at the same time that you program the security key.

14.7. MODIFIED QUICK-PULSE ALGORITHM

The code that implements the Modified Quick-Pulse Algorithm programs each OTPROM location by sending 5 or 25 separate 100 µs programming cycles to each location. After the 5th (25th) pulse, a verification routine compares the location's contents to the input data. Intel guarantees lifetime data retention for a device programmed with the Modified Quick-Pulse Algorithm.

The Auto, PCCB, and UPROM programming algorithms produce the appropriate number of programming pulses, either 5 (late 8XC196KC and all 8XC196KD devices) or 25 (early 8XC196KC devices).

During Slave Programming mode, the external programmer controls the number of programming pulses at each OTPROM location. Design the external programmer to perform the Modified Quick-Pulse Algorithm as described in Figure 14-7 on page 14-16.

14.8. SIGNATURE WORD

The 8XC196KC/KD contains a Signature Word at location 0070H. The Signature Word identifies device type. The word can be accessed in the Slave Programming mode by executing a Dump-Word command at memory location 0070H. The programming voltages are determined by reading locations 0072H and 0073H in Slave Programming mode. The Signature Word values and voltage levels are shown in Table 14-7. The voltages are calculated by using the following equation.

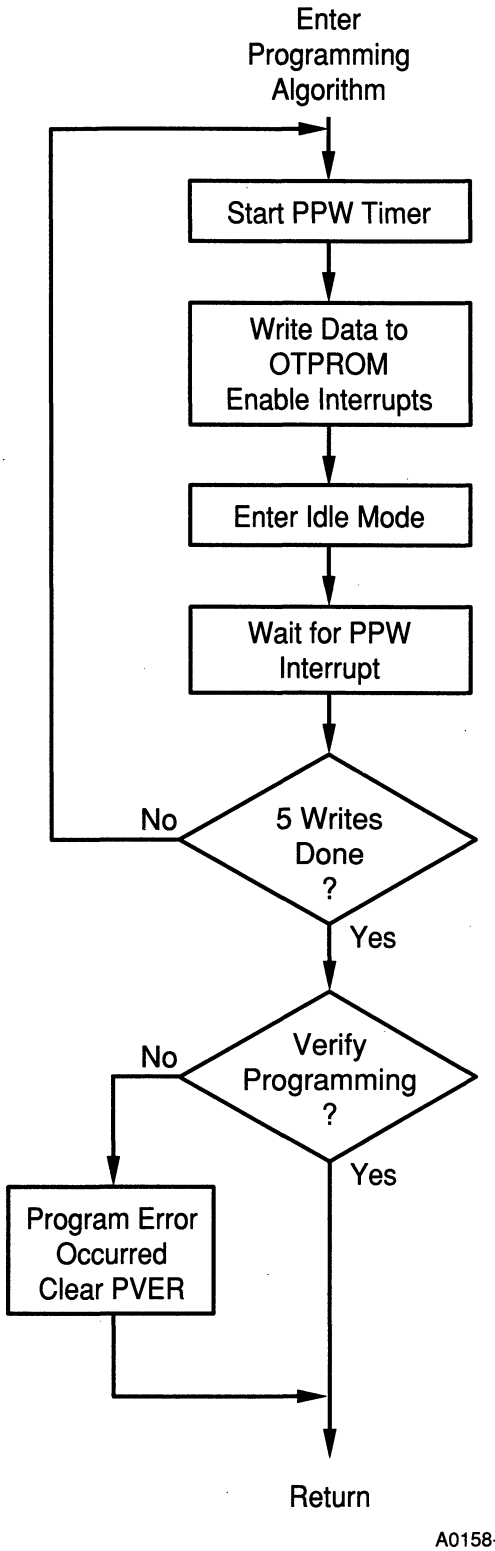
Voltage = 20 / 256 x (test ROM value)

Table 14-7. Signature Word and Voltage Levels

Description	Location*	Early 8XC196KC Value**	Late 8XC196KC Value	8XC196KD Value
Signature Word	0070H	879CH	879CH	879DH
Programming V_{CC}	0072H	0A0H	40H	40H
Programming V_{PP}	0073H	40H	0A0H	0A0H

* Location accessed with a Word Dump during Slave Programming mode.

** An error in the early 8XC196KC device reversed the V_{CC} and V_{PP} bytes.



1. Start the Programming Pulse Width timer.
2. Write the data word to the OTPROM. (Turns on internal programming voltage and begins programming the location.)
3. Enable interrupts and enter Idle mode.
4. Sleep until the PPW causes an interrupt to the CPU that terminates the programming pulse.
5. Read the loop counter. If ≠ 5 (or 25) reiterate. If = 5 (or 25) continue. (Slave Programming: loop counter = 1)
6. Verify programming. If programming verifies, return to the main program. If programming does not verify, clear PVER (P2.0 = low) then return to the main program.

NOTE

A verification error does not stop the programming process.

Figure 14-7. Programming Algorithm

14.9. AUTO PROGRAMMING MODE

The Auto Programming mode is a low-cost programming alternative. Using this programming mode, you can program the 8XC196KC/KD without using an EPROM programmer. The device programs itself with the data from an external EPROM, external locations 4000H–BFFFH (8XC196KD) or 4000H–7FFFH (8XC196KC). The Auto Programming algorithm uses the Programming Pulse Width (PPW) value (discussed on page 14-21) and the Modified Quick-Pulse Algorithm (discussed on page 14-15). It takes approximately 10 seconds to program 32 Kbytes of OTPROM (8XC196KD).

In the Auto Programming mode, the PCCB loads the Chip Configuration Register. The 8XC196KC/KD gets programming data through the external bus, so the PCCB must correctly correspond to the memory system in the programming setup, which is not necessarily the memory system of the application. The "Memory Protection" section beginning on page 14-7 discusses the PCCB.

If the CCB's security-lock bits (CCB.6 and CCB.7) are enabled, a security-key verification occurs at the beginning of the Auto Programming algorithm when you enter the Auto Programming mode. If the verification fails, the device enters an endless internal loop and you must reset. A security-key programming section begins on page 14-14.

14.9.1. Auto Programming Circuit

Figure 14-8 shows the circuit diagram we recommend for Auto Programming mode. (This circuit replaces previously recommended circuits, which did not use P1.0–P1.2 to generate the upper address bits.) This circuit functions with both the 8XC196KC and 8XC196KD devices. It remaps the external EPROM address as shown in Table 14-8.

Table 14-8. Auto Programming Memory Map

Device	External EPROM Address	Internal OTPROM Address	Description
8XC196KC/KD	2014H	N/A	PPW Least-Significant Bit
8XC196KC/KD	2015H	N/A	PPW Most-Significant Bit
8XC196KC	4000H–7FFFH	2000H–5FFFH	Reserved locations for code and data.
8XC196KD	4000H–BFFFH	2000H–9FFFH	Reserved locations for code and data.
8XC196KC/KD	E020H–E02FH	2020H–202FH	Security key, during verification.

The circuit in Figure 14-8 uses an 8-bit external bus (because BUSWIDTH is low). P1.0, P1.1, and P1.2 generate the three high-order bits of the external EPROM address. Hardwire P0.4–P0.7 to V_{SS} and V_{CC} to determine the programming mode. (In this case 1100B = 0CH = Auto Programming mode.) PACT# and PVER are status outputs, buffered by the 74HC14. They drive LEDs that indicate programming active (PACT#) and programming verification (PVER). All unused inputs are connected to ground (V_{SS}), and unused outputs are left floating. READY, NMI, and BUSWIDTH are active; connect them as indicated. Auto programming is specified for a crystal frequency of 6 to 8 MHz. At 8 MHz, a 27(C)512 EPROM with T_{ACC} = 250 ns and T_{OE} = 100 ns or faster specifications should be used. Figure 14-2 and Table 14-3 (see page 14-5) list and describe the device's memory map and addresses.

Table 14-9 shows the function of the programming-mode signals during Auto Programming mode.

Table 14-9. Auto Programming Mode Pin Function

Function Name	Type	Description
PACT# (P2.7)	O	Programming Active. A zero indicates that programming activity is occurring.
PBUS (Ports 3 and 4)	I/O	Address/Command/Data Bus. Used as a regular system bus to access external memory.
PVER (P2.0)	O	Program Verification. Updated after each programming pulse is issued. A high signal indicates successful programming; a low signal indicates a detected error.
P1.0–P1.2	0	Block select lines. Forms the upper address for the external EPROM.

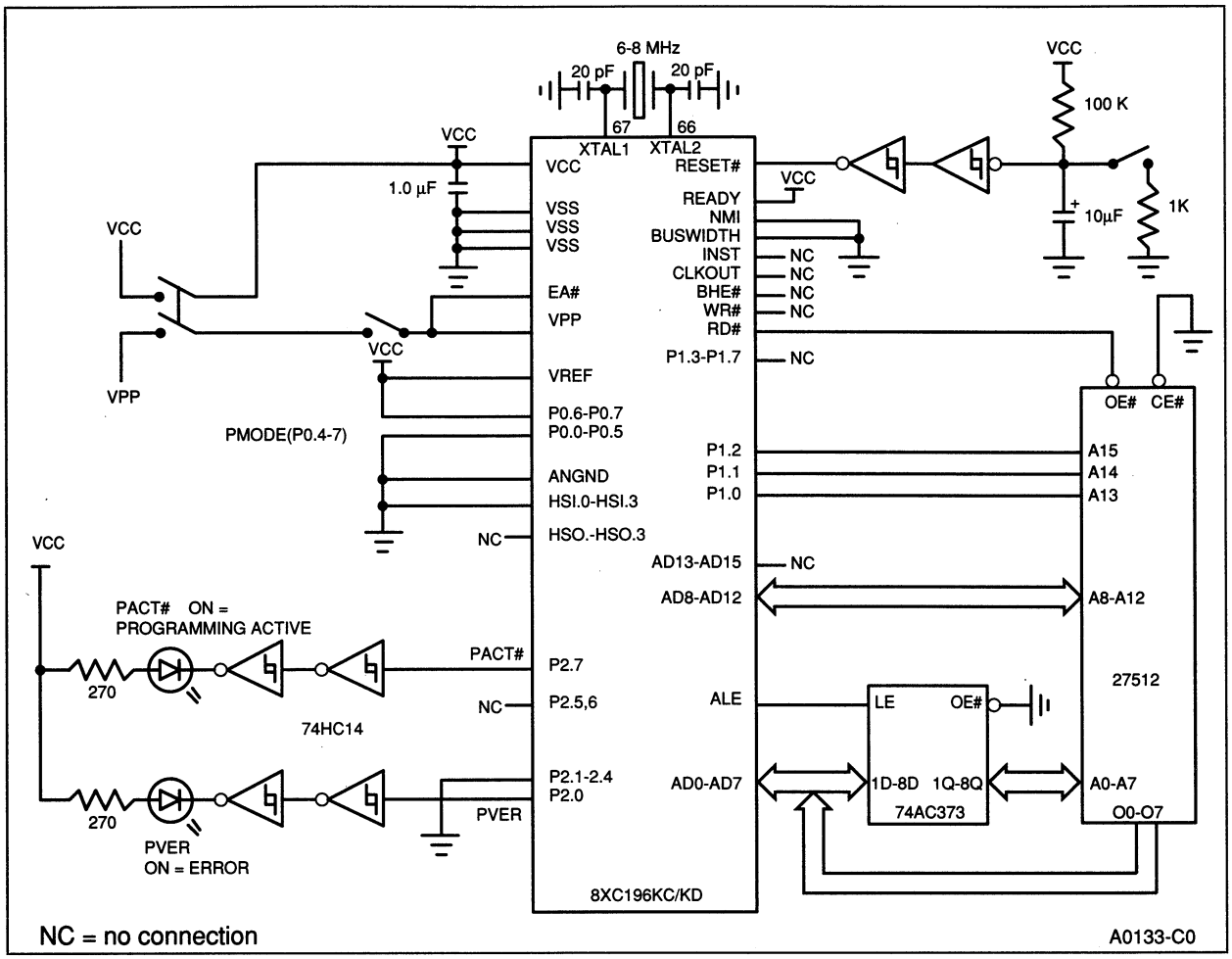


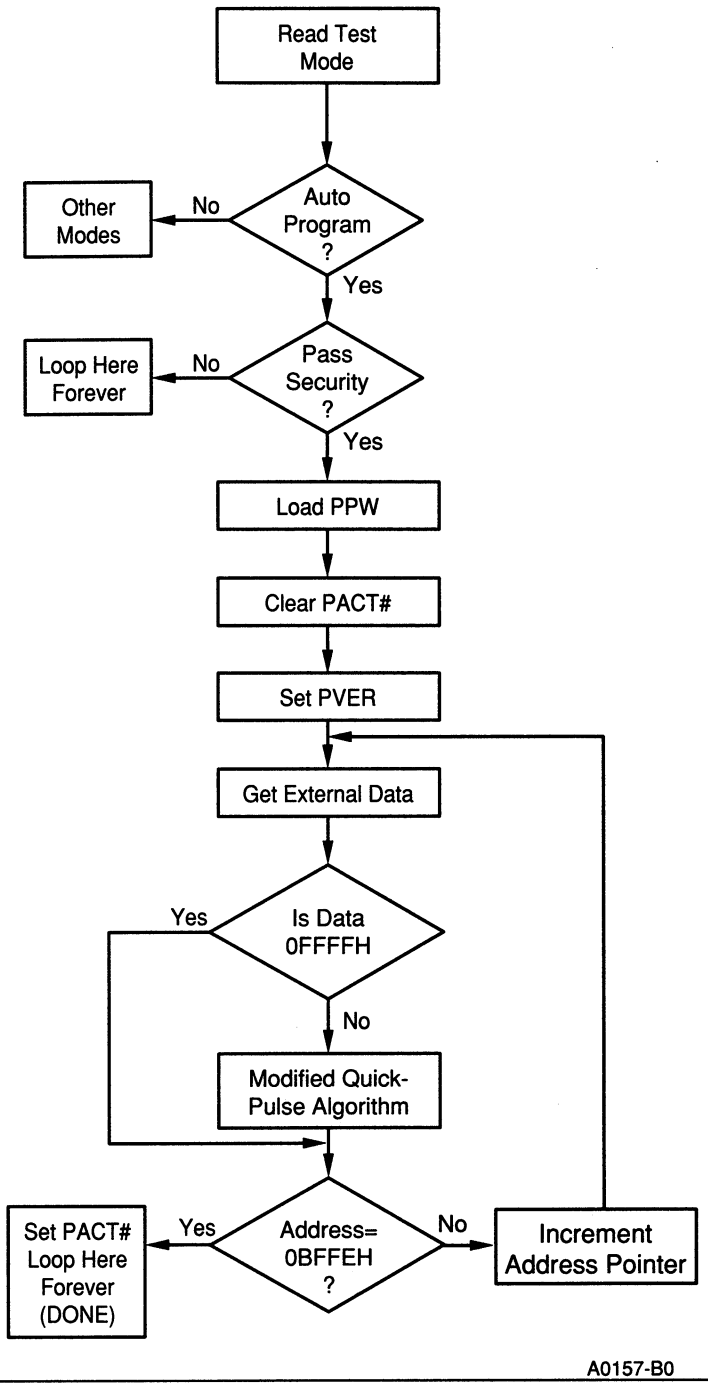
Figure 14-8. Auto Programming Mode Circuit

14.9.2. Verify the Security Key in Auto Programming Mode

If you program the device and clear either CCB.6 or CCB.7 to enable the security feature, the Auto Programming mode algorithm verifies the security key before Auto programming begins. The security-key verification compares external locations E020H–E02FH to internal locations 2020H–202FH. The internal and external locations must match or the device enters an endless loop, preventing programming.

14.9.3. Auto Programming Mode Algorithm

To enter programming mode and initiate the following sequence of events, set PMODE at 0CH and follow the "Power-Up Sequence" described on page 14-6. Figure 14-9 describes the test-ROM program.



1. Determine the programming mode. (Read PMODE.)
2. If Auto Programming mode is selected, continue with this algorithm. (PMODE = 0CH.)
3. If 2020H–202FH is programmed, compare the security key to external locations E020H–E02FH.
4. Load the PPW. (External locations 2014H and 2015H.)
5. Activate PACT# (P2.7 = low).
6. Initialize PVER (P2.0 = high).
7. Read external data, beginning at 4000H.
8. If a word ≠ 0FFFFH, call the Modified Quick-Pulse subroutine.
9. See Figure 14-7.
10. If the address = 0BFFEH, deactivate PACT# (P2.7 = high) and enter an infinite loop. If the address ≠ 0BFFEH, increment address pointer and reiterate programming.

Figure 14-9. Auto Programming Mode Algorithm

14.9.4. Calculating the Programming Pulse Width

The Programming Pulse Width (PPW) register is accessible only during Auto Programming mode; load it from external address 2014H–2015H.

The PPW must equal at least 100 µs for the programmer to function correctly. Use the following formula to calculate the PPW_VALUE. Round the PPW_VALUE to the next higher integer value.

$$PPW\_VALUE = (0.6944 \times F_{OSC}) - 1$$

where:

PPW_VALUE is a 15-bit word

F_{OSC} is the XTAL1 frequency, in MHz

For example, assume XTAL1 is 8 MHz:

$$\begin{aligned} PPW\_VALUE &= (0.6944 \times 8) - 1 \\ &= 5.5552 - 1 \\ &= 4.5552 \\ &\equiv 5 \end{aligned}$$

External memory location 2014H is loaded with the low byte and location 2015H is loaded with the high byte of PPW. Location 2015H usually equals 80H, although it can hold a different value. In every instance, the most-significant bit must equal 1.

The register in Figure 14-10 illustrates the example: most-significant bit (must) = 1, location 2015H = 80H, and location 2014H = 05H.

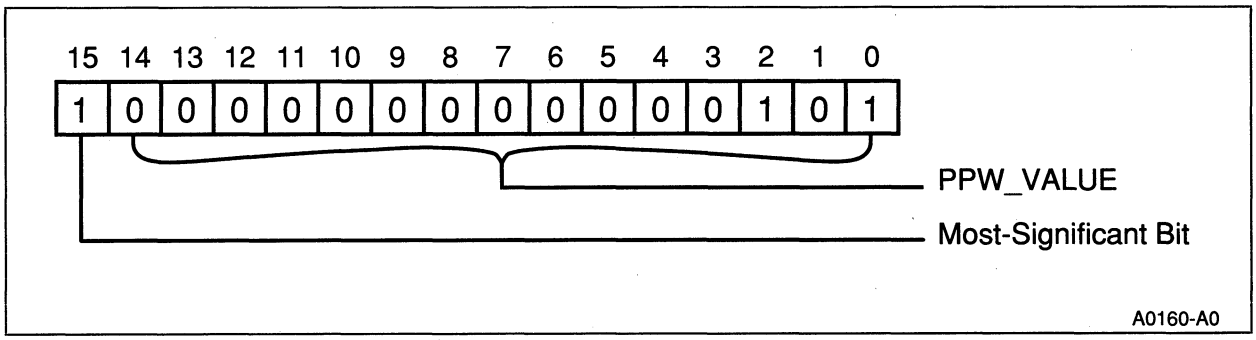


Figure 14-10. Programming Pulse Width Register

Table 14-10. Slave Programming Mode Pin Function

Function Name	Type	Description
AINC# (P2.4)	I	Auto Increment. An active-low signal enables the auto increment mode. Auto increment allows reading from or writing to sequential OTPROM locations without requiring address transactions across the programming bus for each read or write. During programming, AINC# is sampled after each location is programmed. If AINC# is asserted, the address is incremented and the next data word is input. During word dump, AINC# is sampled after each word has been written to Port 3/4. If AINC# is asserted, the address is incremented and the device is ready to output the next data word.
CPVER (P2.6)	O	Cumulative Program Verify. When programming is finished, a high signal indicates that all locations programmed successfully, and a low signal indicates that an error occurred during one of the programming operations.
PALE# (P2.1)	I	Programming Address Latch Enable. Used as the handshake signal for reading address and commands into the device. PALE# is normally held high by the user. When asserted, PALE# causes the device to read an address/command from Ports 3 and 4.
PBUS (Ports 3/4)	I/O	Address/Command/Data Bus. Used to read and write commands, addresses, and data. Ports 3 and 4 are used in their open-drain I/O port modes (not as a system bus). Add external pull-up resistors so you can read data from the device during the Dump Word routine.
PROG# (P2.2)	I	Programming Start. During the Program Word routine, PROG# is used as the handshaking signal for reading programming data into the device. PROG# is normally held high by the user. When asserted, PROG# causes the device to read data from Port 3/4 and program the internal OTPROM. If PROG# remains asserted, that location is programmed with the same data for five 100 µs pulses. For this reason, the data on Port 3/4 must remain stable while PROG# is asserted. When PROG# is deasserted, program flow continues. During Dump Word, PROG# is used as handshaking for outputting data from Port 3/4. PROG# is normally held high by the user. When asserted, PROG# causes the contents of an OTPROM location to output on Port 3/4. When PROG# is deasserted, program flow continues.
PVER (P2.0)	O	Program Verification. Updated after each programming pulse is issued. A high signal indicates successful programming; a low signal indicates a detected error.

The diagram in Figure 14-5 shows the connections we recommend for Slave Programming mode.

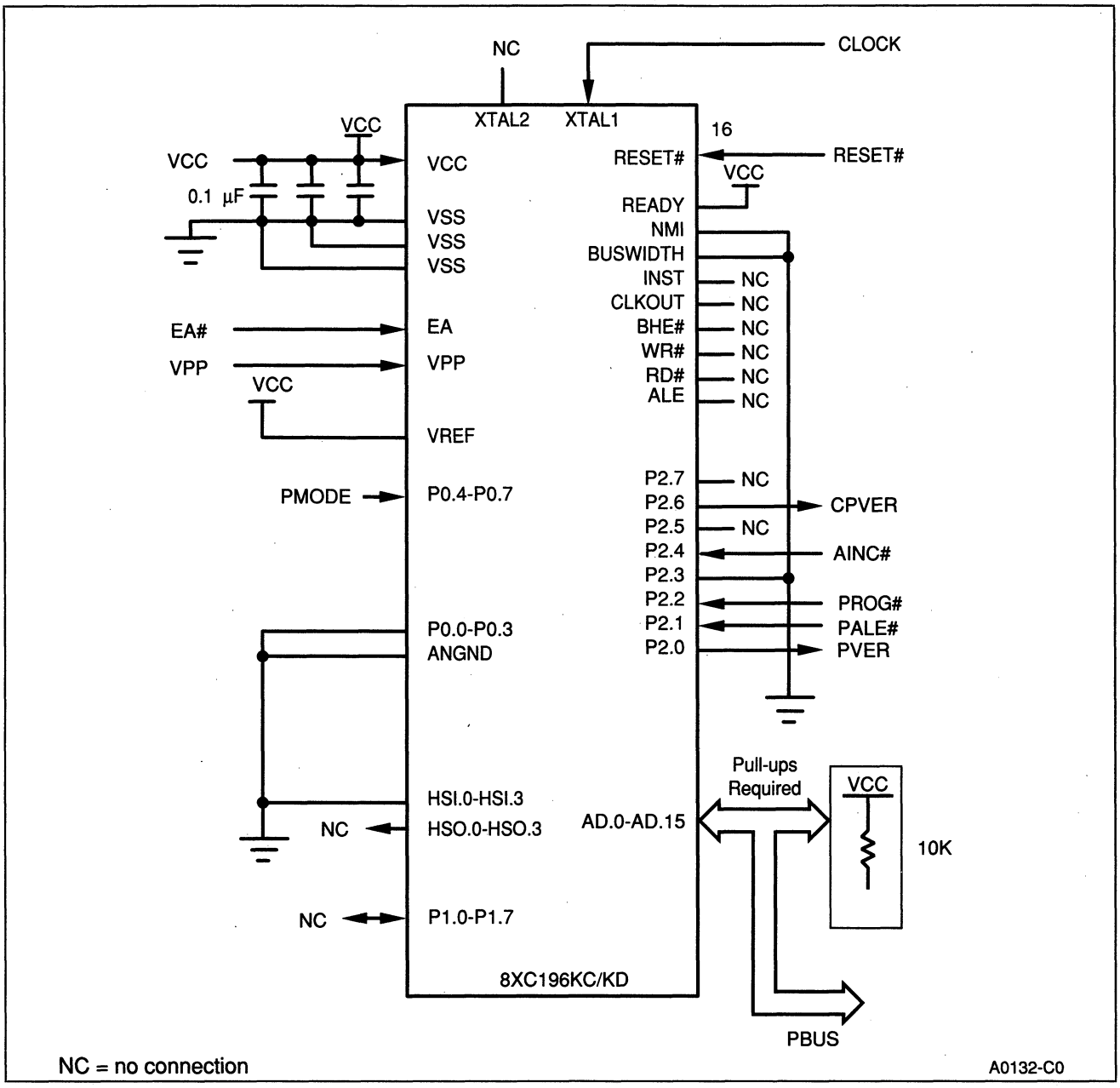


Figure 14-11. Slave Programming Circuit

14.10.2. Verify the Security Key in Slave Programming Mode

The Slave Programming algorithm checks the security-lock bits in the CCB (OTPROM location 2018H) before running the Program Word or Dump Word routine. If the algorithm finds that you cleared either CCB.6 or CCB.7 (write or read protection), programming the security key locations is the next step. Program eight consecutive words, starting at location 2020H and continuing to 202FH. (No programming actually takes place, but the handshaking is identical to that of the Program Word routine.) The algorithm compares the programmed values with internal OTPROM locations 2020H–202FH. If security is passed, access to the device is granted; otherwise, the device hangs up and must be reset. If the verification fails, the device prevents programming or dumping of any location.

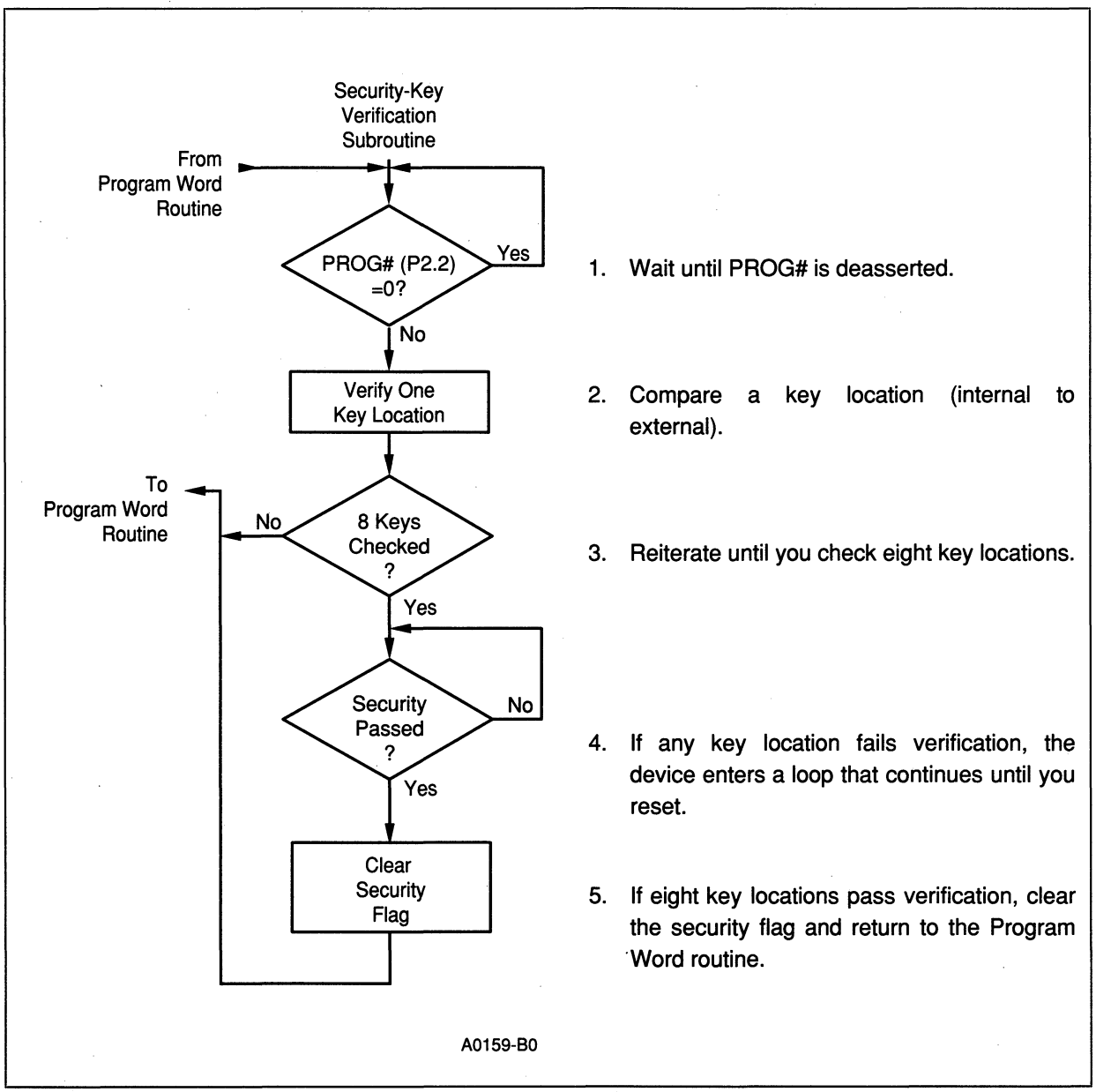


Figure 14-12. Security-Key Verification Subroutine

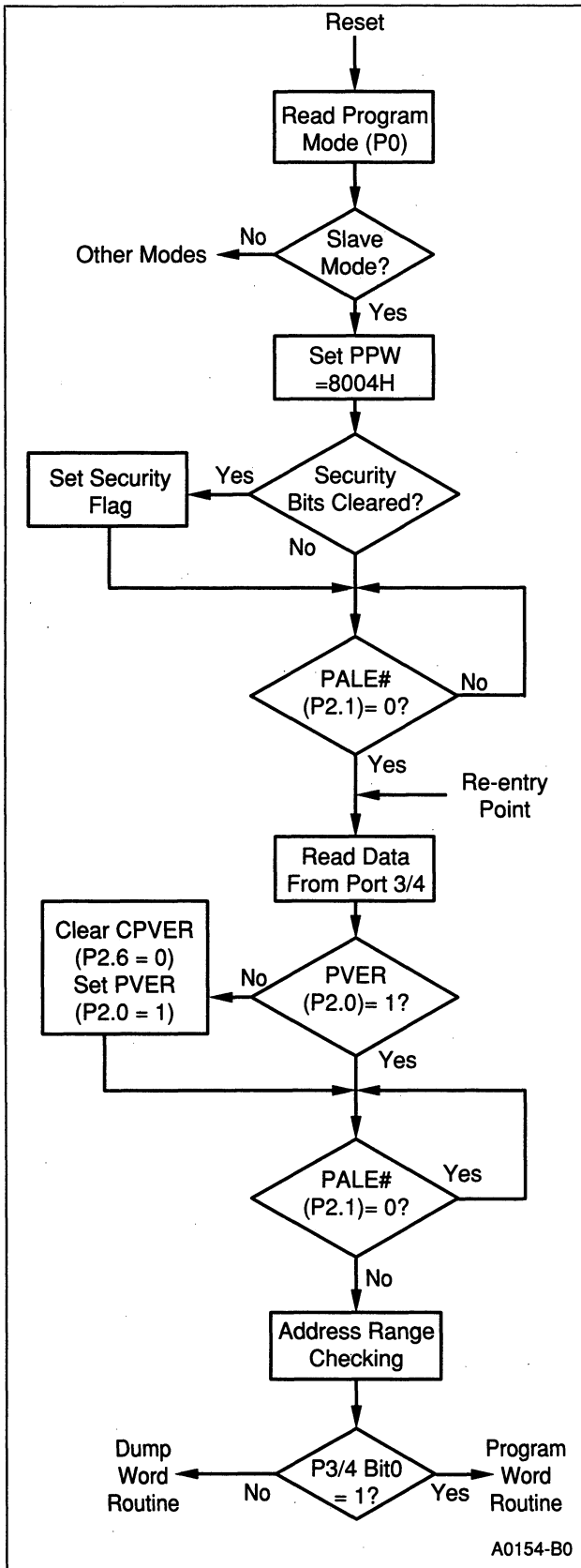
14.10.3. Slave Programming Mode Algorithm

The Power-Up Sequence (page 14-6) and the PMODE Value table (Table 14-1 on page 14-2) provide the information you need to enter Slave Programming mode. The Slave Programming mode algorithm comprises three functional blocks: the Address/Command Decoder, the Program Word routine, and the Dump Word routine.

Figure 14-13 illustrates the Address/Command Decoder routine. In the address/command decoder and initialization block, the algorithm reads PMODE, initializes the PPW register, checks security, and performs handshaking with the PALE# input.

Figure 14-14 illustrates the Program Word routine. This routine programs individual or sequential OTPROM locations. It includes the security-key verification subroutine.

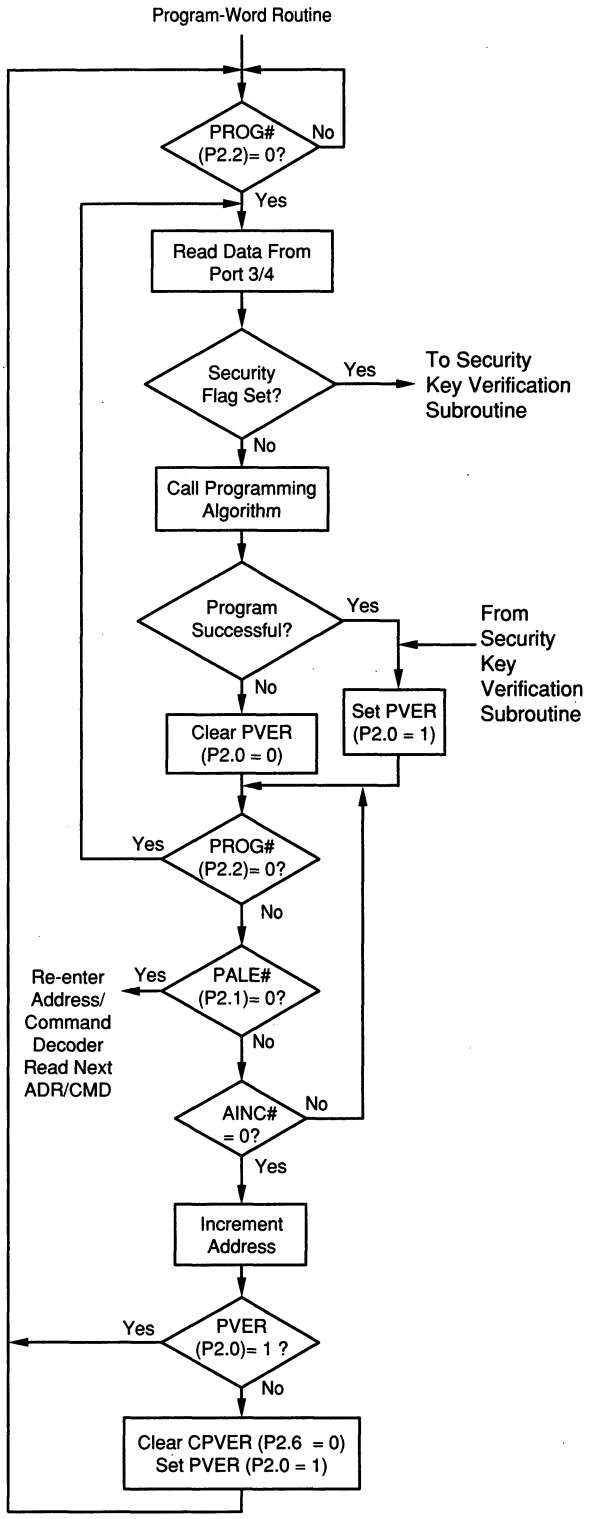
Figure 14-15 illustrates the Dump Word routine. This routine outputs individual or sequential OTPROM locations to Port 3/4.



1. Determine the programming mode. (Read PMODE.)
2. If Slave Programming mode is selected (PMODE = 5H), continue with this algorithm.
3. Initialize the PPW to 8004H.
4. If the CCB.6 or CCB.7 is cleared, set the security flag before going to step 5.
5. If PALE# is asserted, go to the next step; otherwise, reiterate.

   (Re-entry point for the Dump Word or Program Word routine.)
6. Read data from Port 3/4.
7. If PVER is set (no error) go to step 8. If PVER is not set, clear CPVER, set PVER, then go to step 8.
8. Wait until PALE# is deasserted.
9. Check the Port 3/4 address range.
10. If P3.0 = 1, go to the Program Word routine (command mode). If P3.0 = 0, go to the Dump Word routine (address mode). (Use the Port 3/4 word as an address to program to or read from.)

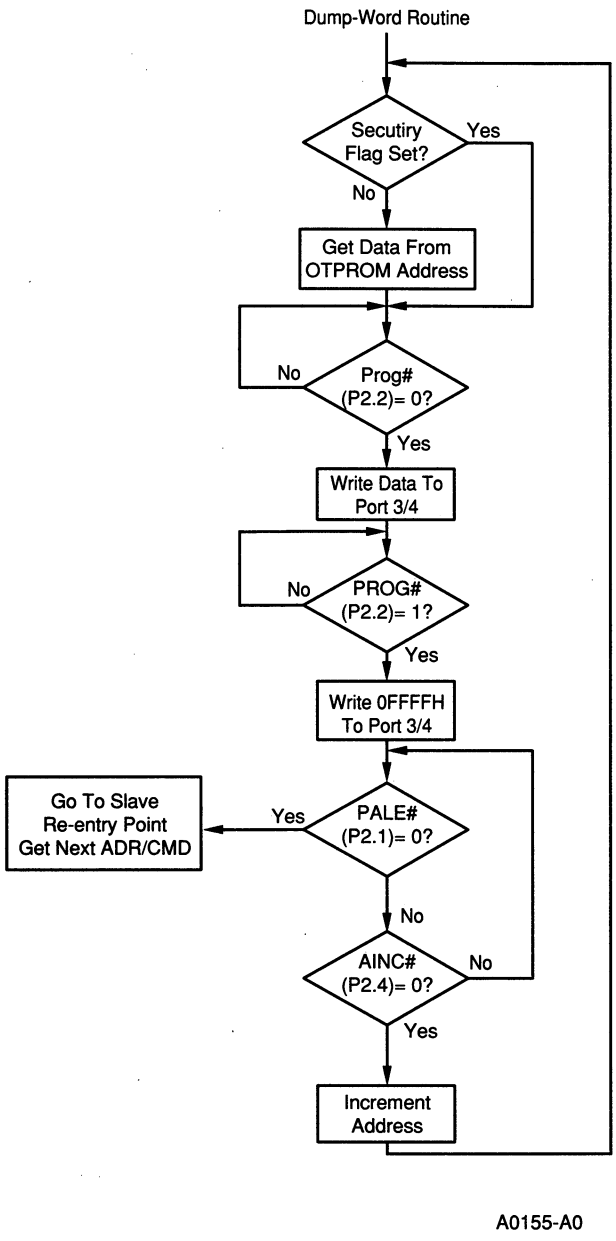
Figure 14-13. Address/Command Decoder Flow



A0156-C0

1. Wait until PROG# is asserted.
2. As soon as PROG# is asserted, read the data from Port 3/4.
3. If the security flag is set (see Figure 14-13), go to the Security-Key Verification subroutine. If the security flag is clear, go to step 4.
4. Call the programming algorithm. (See page 14-16.)
5. If programming passes verification, set PVER; if it fails, clear PVER. (The algorithm completes this step for each of the five 100 µs pulses that programs the word.)
6. Hold PROG# asserted until the program loops five times, then deassert PROG#. (An alternative is to pulse PROG# five times while PALE# = 1 and AINC# = 1.)
7. If PALE# is asserted, re-enter the Address/Command Decoder routine and read the next address/command.
8. If PALE# is deasserted, test AINC#.
9. If AINC# is asserted, increment the address by two, then verify.
10. If PVER is asserted (no error occurred) re-enter the Program Word routine.
11. If PVER is not asserted (verification failed), clear CPVER, set PVER, then re-enter the Program Word routine.

Figure 14-14. Program Word Routine



1. If the security flag is set (see Figure 14-13), skip step 2. If the security flag is not set, go to step 2.
2. Read data from the OTPROM.
3. Wait until PROG# is asserted.
4. As soon as PROG# is asserted, output the data to Port 3/4.
5. Wait until PROG# is deasserted.
6. Write all ones to Port 3/4. (This action allows the open-drain port's pins to float, avoiding conflict with the Program Word and Address/Command Decoder routines, which use Port 3/4 as an input.)
7. If PALE# is asserted, return to the Address/Command-Decoder routine. If PALE# is deasserted, read AINC#.
8. If AINC# is not asserted, return to step 7.
9. If AINC# is asserted, increment the address by two and re-enter this routine.

Figure 14-15. Dump Word Routine

14.10.4. Program Word and Dump Word Commands

Table 14-11 defines the timing mnemonics used in the Program Word and Dump Word waveforms. The 8XC196KC/KD data sheets include timing specifications for these signals.

Table 14-11. Timing Mnemonics

Mnemonic	Description
T_{SHLL}	Reset High to First PALE# Low.
T_{LLLH}	PALE# Pulse Width.
T_{AVLL}	Address Setup Time.
T_{LLAX}	Address Hold Time.
T_{PLDV}	PROG# Low to Word Dump Valid.
T_{PHDX}	Word Dump Data Hold.
T_{DVPL}	Data Setup Time.
T_{PLDX}	Data Hold Time.
T_{PLPH}	PROG# Pulse Width.
T_{PHLL}	PROG# High to Next PALE# Low.
T_{LHPL}	PALE# High to PROG# Low.
T_{PHPL}	PROG# High to Next PROG# Low.
T_{PHIL}	PROG# High to AINC# Low.
T_{ILIH}	AINC# Pulse Width.
T_{ILVH}	PVER Hold After AINC# Low.
T_{ILPL}	AINC# Low to PROG# Low.
T_{PHVL}	PROG# High to PVER Valid.

Figure 14-16 shows the timing of the Program Word command with a repeated programming pulse and auto increment. A one on P3.0 selects the Program Word command. An address of 3501H programs the word location at 3500H. The device receives an input signal, PALE#, to indicate that a valid command is present; asserting PALE# latches the command and address on Ports 3 and 4 (PBUS). Asserting PROG# latches the data on Ports 3 and 4 to start the programming sequence; the device reads in or outputs a word. The PROG# pulse width determines a programming pulse width. (Slave Programming mode does not use the PPW.) After the rising edge of PROG#, the slaves automatically verify the contents of the location just programmed. An asserted PVER indicates successful programming. AINC# is optional and can automatically increment the address for the next location.

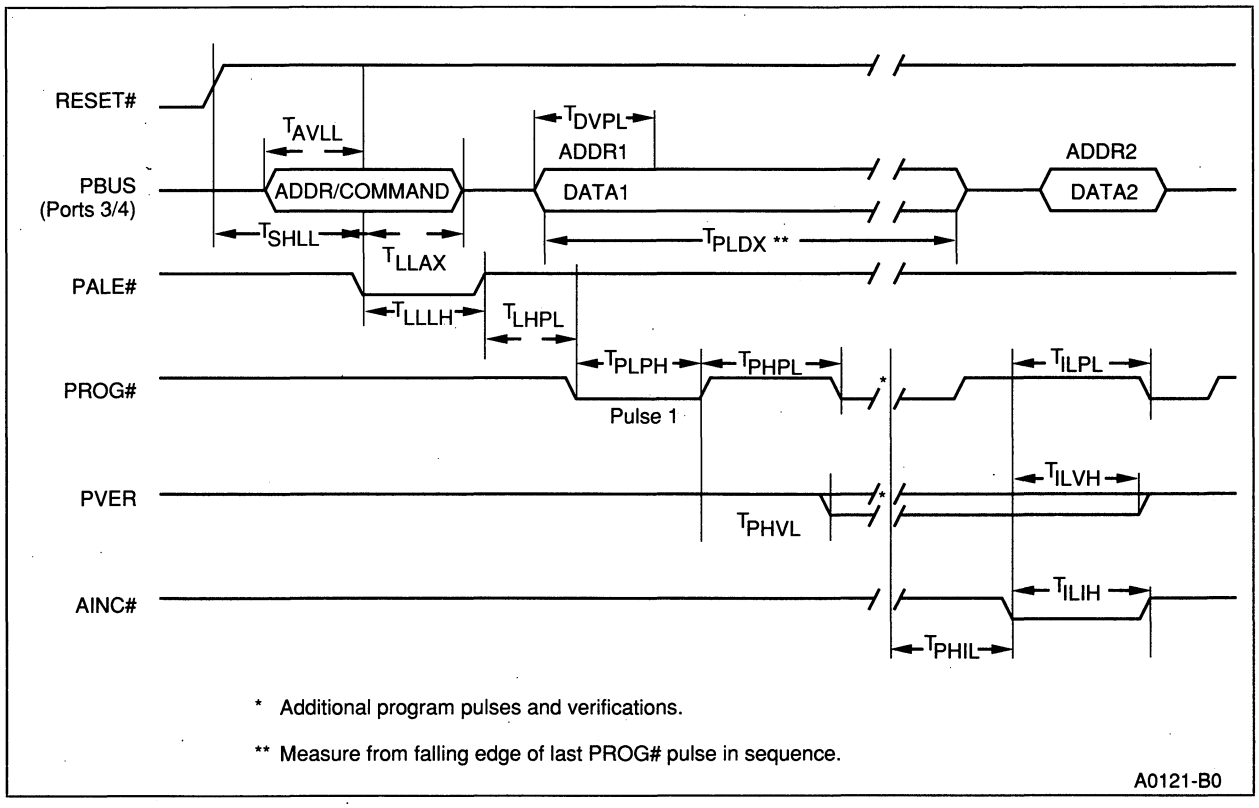


Figure 14-16. Program Word Waveform

Figure 14-17 illustrates the Dump Word command. A zero on P3.0 selects the Program Word command. Sending the command "2100H" results in the slave placing the word at internal address 2100H on Ports 3 and 4. PROG# governs when the device drives the bus. The timings before the Dump Word command are the same as those shown in Figure 14-16. In the Dump Word mode, the AINC# pin can remain active and toggling. The PROG# pin automatically increments the address.

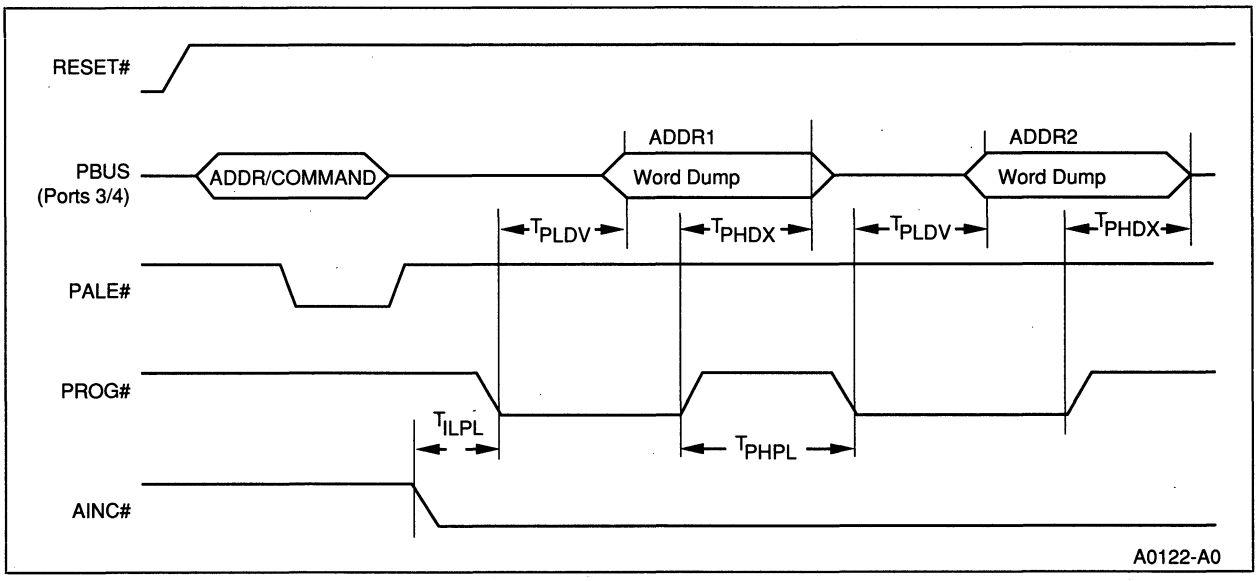


Figure 14-17. Dump Word Waveform

14.11. ROM-DUMP MODE

The ROM-Dump mode is an easy way to verify the contents of the OTPROM array. The Power-Up Sequence (page 14-6) and the PMODE Value table (Table 14-1 on page 14-2) will help you enter ROM-Dump mode.

After verifying the security key, ROM-Dump mode writes the entire OTPROM array to external memory. The security-key verification must find the internal security key duplicated at the corresponding external addresses. If the security key doesn't match, the device enters an endless loop of internal execution, which can be exited only by a chip reset. If USFR.2 (the Disable External Data fetches (DED) bit) has been programmed, ROM-Dump mode is disabled entirely. (The "Memory Protection Options" section provides more information about the security key and UPROM programming.)

Table 14-12. ROM-Dump Mode Memory Map

Device	Internal OTPROM Address	External Memory Address
8XC196KC	2000H–5FFFH	2000H–5FFFH
8XC196KD*	2000H–9FFFH	4000H–BFFFH
8XC196KC*	2000H–5FFFH	4000H–7FFFH
8XC196KC Security Key	2020H–202FH	2020H–202FH
8XC196KD Security Key*	2020H–202FH	E020H–E02FH
8XC196KC Security Key*	2020H–202FH	E020H–E02FH

* Must use bank Switching; P1.0–P1.2 replaces A13–A15.

For devices with 16 Kbytes or less of on-chip OTPROM, the array is dumped to the external memory locations indicated in the table. A bank switching mechanism is provided for devices with more than 16 Kbytes of on-chip OTPROM. The bank switching mechanism must be implemented externally to allow dumps of larger OTPROM arrays, such as an 8XC196KD (32 Kbytes OTPROM). The bank pins (P1.0–P1.2) replace the four high address pins. For instance, if 32 Kbytes are dumped (8XC196KD), they are dumped to external addresses generated by using the bank address pins. However, the real address pins will indicate two consecutive 16 Kbyte dumps. Table 14-13 shows the substitutions for the address bus that must take place before bank switching can operate.

Table 14-13. Bank Switching Mechanism

Bank Pin		High Address Pin
P1.2	replaces	A15
P1.1	replaces	A14
P1.0	replaces	A13

14.12. RUN-TIME PROGRAMMING MODE

You can program an OTPROM location during normal code execution. To make the OTPROM array accessible, hold EA# high while you reset the device. Then simply write to the location to be programmed. Apply V_{PP} voltage to the V_{PP} pin during the entire programming process.

Immediately after writing to the OTPROM, the device must either enter the Idle mode or execute code from external memory; an access to OTPROM would abort the current programming cycle. Each programming cycle begins when the word is written to the OTPROM and ends when the next OTPROM access occurs. Each word requires a total of five programming cycles. The length of each programming cycle must be approximately 100 µs.

Figure 14-18 is a Run-Time Programming code example.

```
Program:      POP ADDRESS_TEMP                          ;load program data
              POP DATA_TEMP                             ;and address
              PUSHF
              LD COUNT, #5t                             ;program using
                                                        ;Modified Quick-
                                                        ;Pulse
LOOP:         LDB INT_MASK, #ENABLE_SWT                 ;program SWT for
              LDB HSO_COMMAND, #SWT0_OVF                ;program pulse width
              ADD HSO_TIME, TIMER1, #PROGRAM_PULSE
              EI
              ST DATA_TEMP, [ADDR_TEMP]                 ;enter idle mode until
              IDLPD 1                                   ;SWT expires
              DJNZ COUNT, LOOP                          ;loop 5 times
              POPF
              RET

SWT_EXPIRED:                                            ;service SWT
              RET                                       ;and return
```

Figure 14-18. Run-Time Programming Code Example