Chapter 8

Device Programming

8.1 Introduction

The SX device has a program memory consisting of 2,048 words of 12 bits per word, plus some additional 12-bit words that specify the device configuration. This memory is a non-volatile, electrically erasable (EEPROM) flash memory, rated for 10,000 rewrite cycles.

Before you can use the SX device, you must write the application code into the program memory. You do this by placing the device into a programming mode and following the protocol for accessing the program memory. You can write to the program memory only in the programming mode, not when the device is executing the application software.

8.1.1 Erasure and Reprogramming

When you erase the program memory, you automatically erase the entire memory, including the DEVICE word, FUSE word, and FUSEX word. An erased memory has all bits set to 1. When you program the device, you clear some of the bits to 0. If you want to reprogram a memory location and clear some more bits to 0, you can "overwrite" the memory location without erasing. However, if you want to program a bit to 1 that has already been cleared to 0, the only way to do so is to erase and reprogram the whole EEPROM memory.

8.1.2 Standard and Custom Programming Tools

The task of programming an SX device is most easily handled by using a third-party tool. One such tool is the SX-Key made by Parallax, Inc. The Parallax tool is simple and easy to use because is takes care of all of the programming details. You only need to write the source code and specify the device options. The tool compiles the code, erases the existing program in the device, and writes the new configuration settings and program code into the device.

Instead of using an existing tool such as the SX-Key, you might want to build your own programmer unit to accommodate the special requirements of your application. This chapter describes the device programming interfaces and protocols so that you can build your own programming unit.

8.1.3 In-System and Parallel Programming Modes

There are two basic device programming modes, called the "In-System Programming" (ISP) mode and the "parallel" mode. The In-System Programming mode uses just two device pins, OSC1, and OSC2, and writes the data to the device serially, one bit at a time. This mode lets you program devices that are

already installed in the target system. The parallel mode uses a larger set of pins and writes data 12 bits at a time, in parallel. This mode is a little faster but can only be used to program free-standing SX parts.

8.2 In-System Programming (ISP) Mode

The In-System Programming (ISP) mode lets you program or re-program an SX device that has been installed and soldered into the target system. Using the ISP mode has many advantages over traditional programming methods, in all stages of the product life: development, manufacturing, and customer service.

In the product development cycle, a separate "emulation" type device is not required. The controller device used for development is the same as the one used for final production, including the package type and pinout. The SX device can be soldered into the target system, and then programmed and reprogrammed any number of times, without removing and reinstalling it. No special socket or support circuitry is required, so the system can be debugged accurately, even in timing-sensitive and noise-sensitive applications.

For manufacturing, circuit boards can be pre-built with the controller installed and soldered on the board, even before the software has been finalized, to meet short time-to-market requirements. Additional information such as vendor numbers and serial numbers can be programmed into the device just prior to shipment. There is no risk of stocking out-dated, pre-programmed units because the software can be corrected or updated at any time.

Even after the product is received by the customer, it can be quickly and easily revised or patched by field service personnel. Customers can even reprogram their products themselves if they have the necessary programming equipment. This equipment is relatively inexpensive and easy to use.

8.2.1 Scenix In-System Programming Implementation

The Scenix ISP method is a proprietary system that uses just two device pins: the clock input pin, OSC1, and the clock output pin, OSC2 (VDD, and GND, and $\overline{\text{MCLR}}$ pins should be connected properly). This system eliminates the need for dedicated programming pins, thus reducing the total device pin count. There is no need for a JTAG tester, an expensive device required by some other programming systems.

OSC1 is used to supply the higher voltage necessary for programming the flash memory (12.5 V), while OSC2 is used to issue commands, to write data to the EEPROM, and to read data back from the EEPROM. The external programming device writes a data stream to OSC2 to specify the ISP programming operations, and to supply the data written into the program memory. When the specified operation is a request to read a program memory location, the SX returns the results as a data stream on the same pin, OSC2.

The OSC1 and OSC2 pins are usually connected to passive components such as resistors, capacitors, and crystals. In typical systems, these components do not interfere with the programming signals and are not harmed by the higher voltages used for programming. In these cases, they can be left connected to the SX device during programming. It is usually not necessary to install additional hardware to isolate the ISP circuit from the rest of the system.

There are three stages to the Scenix ISP protocol:

- Entering the ISP Mode
- Programming in ISP Mode
- Exiting the ISP Mode

8.2.2 Entering the ISP Mode

For normal operation of the SX device, the OSC2 pin is either left unconnected, connected to passive components, or used as a clock output pin, depending on the chosen clock configuration. To put the device into the ISP mode, you pull the OSC2 pin low for at least nine consecutive clock cycles on the OSC1 pin (or nine internal clock cycles in the internal clocking mode). This action is a signal to the SX to go into the programming mode.

The exact procedure for entering the ISP mode depends on whether you are using external or internal components for normal clocking of the device. If you are using an external crystal or resonator (including the XT, LP, or HS mode), an external RC oscillator, or an external clock signal for normal device operation, then you need to use the control signals and timing shown in Figure 8-1 to enter the programming mode. If you are using the internal RC oscillator for normal device operation, then you need to use the control signals and timing shown in Figure 8-2 to enter the programming mode.

3

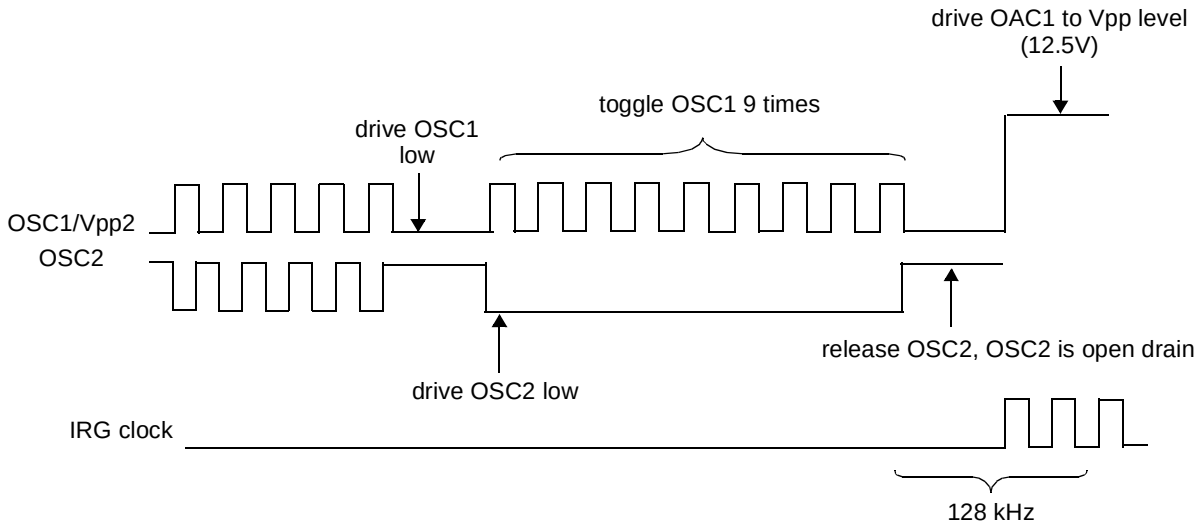

Figure 8-1 ISP Mode Entry with External Clocking

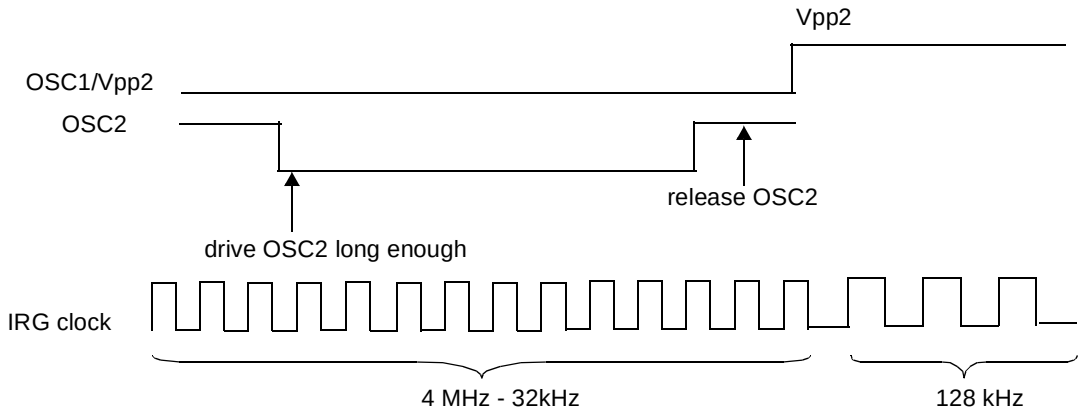

Figure 8-2 ISP Mode Entry with the Internal RC Oscillator

External Clocking

When the device is clocked by external components or an external clock signal, the programmer unit should use the following procedure to place the SX device in the ISP programming mode:

1. Drive the OSC1 pin low to stop the clock.
2. Drive the OSC2 pin low and toggle the OSC1 pin at least nine times. This is the signal to enter the ISP mode.
3. Release the OSC2 pin.
4. Apply the VPP programming voltage to the OSC1 pin. The SX internal RC oscillator starts operating at 128 kHz. This clock drives the SX device during ISP mode programming.

Internal RC Oscillator

When the device is clocked by the internal RC oscillator, the programmer unit should use the following procedure to place the SX device in the ISP programming mode:

1. Drive the OSC2 pin low for at least nine internal clock cycles. The internal clock frequency can be any one of eight values ranging from 31.25 kHz to 4 MHz, depending on the divide-by rate programmed into the FUSE word.
2. Release the OSC2 pin.
3. Apply the VPP programming voltage to the OSC1 pin. The SX internal RC oscillator starts operating at 128 kHz. This clock drives the SX device during ISP mode programming.

8.2.3 Programming in ISP Mode

Upon entry into the ISP mode, the SX device could be in the middle of executing a program, possibly with some I/O ports configured as outputs and driving other devices in the system. The first action of the ISP logic is to reset the SX device. This puts the device into a known logic state and configures the I/O ports to operate as inputs, thus preventing possible damage to other components in the system.

After the device is reset, the ISP logic executes the ISP protocol. This is a "self-aligned" serial communication protocol that uses the OSC2 pin for both synchronization and for serial I/O. No separate clock pin is needed in this protocol. The OSC2 pin is implemented with an open drain and an internal pullup, allowing it to operate as an input or output.

Frames, Cycles, and Internal Clocks

Communication is carried out in packets called "frames." Each frame consists of 17 cycles, and each cycle consists of four internal clocks. The period of the internal clock is 7.81 microseconds, so each cycle is 31.3 microseconds and each frame is 531 microseconds.

Figure 8-3 shows the timing of an ISP frame. The frame consists of 17 cycles. The first cycle is the "sync" cycle, used to synchronize the programmer unit to the ISP frame. This is followed by four "command" cycles, designated C3 through C0. The programmer unit drives the OSC2 pin during these cycles to specify a programming operation such as "erase," "read," or "write." The command cycles are followed by 12 "data" cycles, designated D11 through D0. During these cycles, the programmer unit drives the OSC2 pin for a "write" operation, or the SX device drives the pin for a "read" operation.

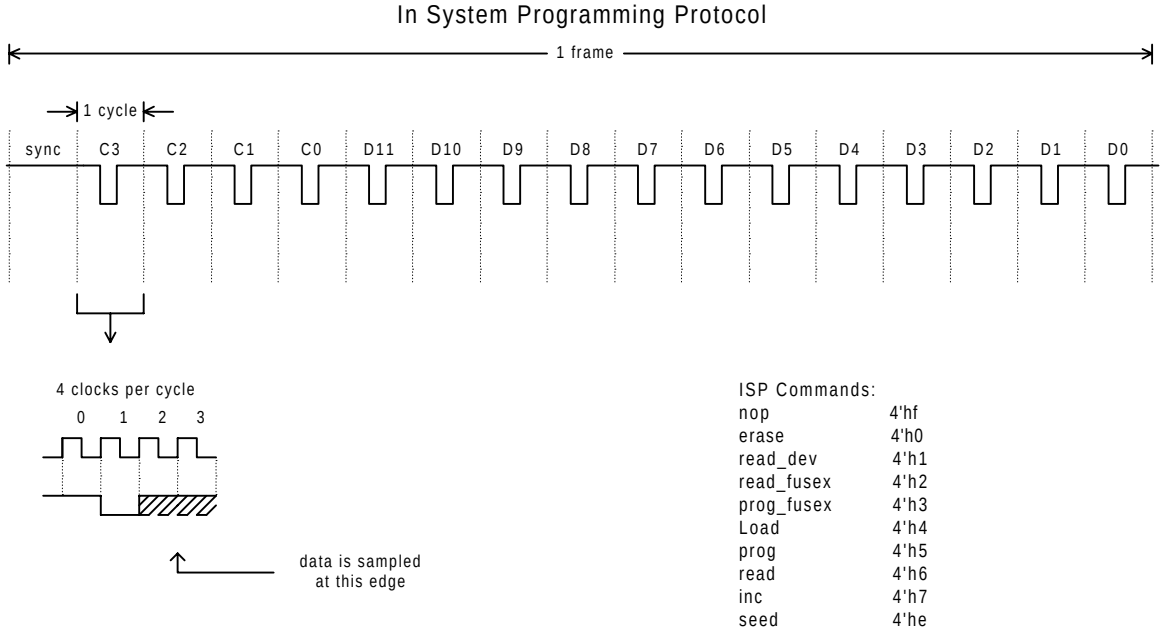

Figure 8-3 ISP Frame

Each of the 17 cycles consists of four internal clock periods.

In the first clock period, nothing drives the OSC2 pin, so the pin is pulled high by an internal pullup resistor.

In the second clock period, the SX device drives the OSC2 pin low. This is the synchronization pulse. The external programming unit uses the leading edge of this pulse to synchronize itself to the SX device. The pulse is omitted in the sync cycle (the first of 17 cycles in a frame) so that the programming unit can determine where the frame starts.

In the third and fourth clock periods, the programmer unit writes a data bit to the SX device or reads a data bit from the SX device, depending on the cycle and the type of command issued. The data bit is placed on the OSC2 pin during these two clock periods, either by the programmer unit or by the SX device, and then sampled by on the rising edge of the fourth clock period.

In the four command cycles (C3-C0), the programmer writes a four-bit command to the ISP logic, which tells the ISP logic what to do during the data cycles. In the 12 data cycles (D11-D0), for a "write" operation, the programmer writes the 12 bits that are to be written to a memory location. For a "read" operation, the programmer reads 12 bits supplied by the SX device from a memory location.

Internal Hardware

Figure 8-4 is a simplified block diagram of the chip-internal ISP hardware.

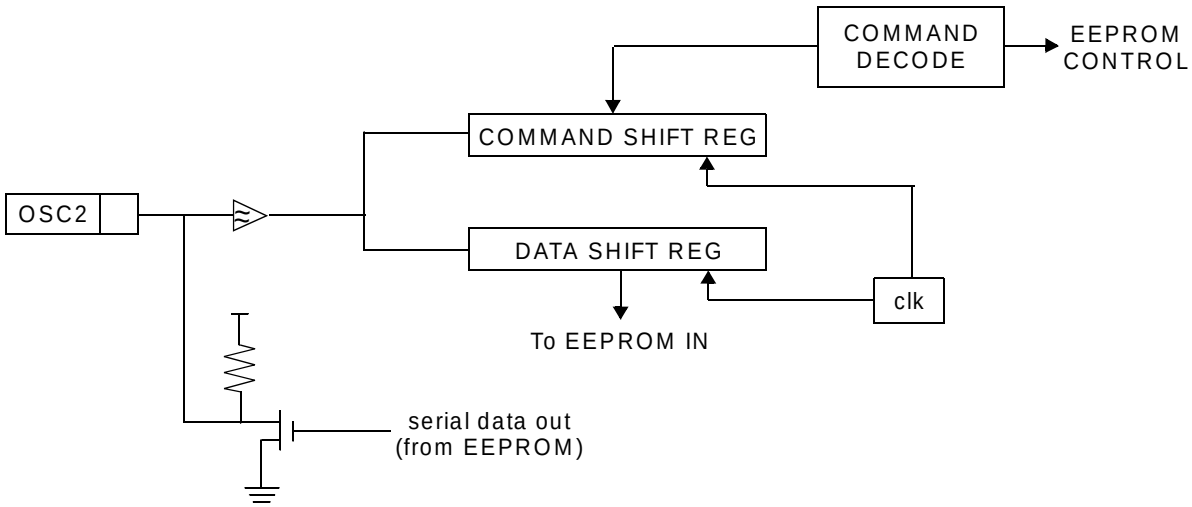

Figure 8-4 ISP Circuit Block Diagram

Serial data written to the OSC2 pin is shifted into the command shift register or data shift register, depending on whether command bits or data bits are being processed within a frame. Command bits are decoded and used to control the flash EEPROM block, while data bits are written to the flash EEPROM.

When the command is to read data from the program memory, the data bits are read from the EEPROM block and shifted out on the OSC2 pin during the data cycles. An open-drain transistor and a pullup

resistor pull the OSC2 pin low or high for each bit. This same transistor is used to pull the OSC2 pin low during the second clock within each cycle (except in the sync cycle).

Commands

The programmer unit writes a 4-bit command during the four command cycles at the beginning of each frame, just after the sync cycle. This 4-bit command tells the ISP logic what to do during the remaining 12 cycles of the frame. Table 8-1 lists and describes the programming commands. Codes not listed in the table are reserved for future expansion.

Table 8-1 ISP Commands

Name	Code	Description
Erase	0000	Erase all EEPROM locations.
Read DEVICE Word	0001	Read DEVICE word (memory size configuration).
Read FUSEX Word	0010	Read FUSEX word (configuration options)
Program FUSEX Word	0011	Program FUSEX word (configuration options)
Load Data	0100	Load data word to be programmed into memory.
Program Data	0101	Program previously loaded data word into memory.
Read Data	0110	Read data word from memory.
Increment Address	0111	Increment the program memory pointer by one.
NOP	1111	No operation.

The commands that erase or program the EEPROM registers must be repeated consecutively for a certain frames in order to work reliably. To determine the minimum required number of repetitions of a command, look in the Electrical Characterization appendix and find the minimum time requirement for the operation. Divide this value by the frame period, 0.53 milliseconds, and round up to the nearest whole number.

For example, if you find that the minimum time requirement for an "Erase" operation is 100 msec, divide 100 by 0.53 and round up, and the result is 189. This means that you must repeat the "Erase" command for at least 189 consecutive frames in order to complete the "Erase" operation reliably.

No repetition is necessary to read a register or to increment the memory address pointer. You can complete one of these operations in just a single frame.

NOP Command

The NOP (no-operation) command causes the ISP logic to do nothing and wait for the next command. The NOP command has a code of 1111 binary. Whenever the programmer unit is not driving to the OSC2 pin, the internal pullup resistor pulls the pin high, which produces 1111 as the command string and invokes the NOP command by default.

This is an important feature because the programmer unit needs some time to synchronize itself to the pulses generated by the ISP logic, and cannot begin driving the OSC2 until synchronization is achieved. In the meantime, the NOP command is executed by default, causing the ISP logic to wait for the first active command.

Reading the DEVICE Word

The DEVICE word is a hard-wired, read-only register containing device information such as the number of register banks and the size of the program memory and SX version number. To read the DEVICE register, the programmer unit issues the "Read DEVICE Word" command and reads the 12 bits of data in the data cycle portion of the frame.

Reading and Programming the FUSEX Word

The FUSEX word is a read/write register that controls device options such as carry flag operation and the brown-out reset function. The five highest-order bits of this register (bits 11:7) are factory-set to certain values that must not be changed. Therefore, the programmer must always read these bits before erasure and reprogram them to the same values after erasure. Bits 11, 9, and 8 are used to calibrate the internal clock. Bit 10 specifies the package size (1 for a 28-pin device or 0 for an 18-pin or 20-pin device). Bit 7 is reserved for future expansion.

To read the FUSEX register, the programmer unit issues the "Read FUSEX Word" command and reads the 12 bits of data in the data cycle portion of the frame.

To program the FUSEX register, first issue the Load command,, the programmer unit issues the "Program FUSEX Word" command and writes the 12 bits of data in the data cycle portion of the frame. This command must be repeated consecutively for a certain number of frames in order to program the register reliably, as explained earlier.

Erasing the Memory

The "Erase" command erases all of the EEPROM memory, including the DEVICE word and FUSEX word. The command must be repeated consecutively for a certain number of frames in order to complete the operation reliably, as described earlier.

The programmer unit should always read the FUSEX word before erasure and restore the five highest-order bits of that register after erasure. These bits have factory-set values that must be maintained.

Reading the Memory

To read the EEPROM program memory, you use two commands: "Read Data" to read the current memory location and "Increment Address" to change an internal memory address pointer from one location to the next.

Upon entry into the ISP mode, the ISP logic is set to access address FFFh, which is the address of the FUSE word. The FUSE word controls many of the device configuration options such as the clocking, stack size, and Watchdog options. To read this initial memory location, the programmer unit issues the "Read Data" command and reads the 12 bits of data in the data cycle portion of the frame.

To read the word at the next address, the programmer unit issues the "Increment Address" command. This increments an internal pointer to the program memory, allowing access to address 000h. It does not matter what the programmer unit does during the 12 data cycles of the "Increment Address" frame. Following this frame, the programmer issues another "Read Data" command and reads the 12 bits of data in the data cycle portion of the frame.

This sequence is repeated to read consecutive memory locations. The first memory location is FFFh (the FUSE word register), followed by 000h, 001h, 002h, and so on up to the top memory address, 7FFh. The programmer can skip over any number of memory locations by repeating the "Increment Address" command consecutively, without using the "Read Data" command. The "Increment Address" command must be used 2,048 times to traverse the whole program memory.

Programming the Memory

To program the EEPROM program memory, you use three commands: "Load Data" to load the a word to be written to a memory location, "Program Data" to write the word into memory, and "Increment Address" to change the memory address pointer from one location to the next.

Upon entry into the ISP mode, the ISP logic is initially set to access the FUSE word. To program this memory location, the programmer unit issues the "Load Data" command and writes the 12 bits of data in the data cycle portion of the frame. Then it issues a "Program Data" command to write the loaded word. It does not matter what the programmer unit does during the 12 data cycles of the "Program Data" frame. This command must be repeated a certain number of times in order to program the register reliably, as explained earlier.

To program the word at the next address, the programmer unit issues the "Increment Address" command, which increments the internal pointer to access the words at address 000h. Following the "Increment Address" frame, the programmer issues another "Load Data" command and writes the 12 bits of data in the data cycle portion of the frame. Then it issues the "Program Data" command consecutively for a certain number of frames.

This sequence is repeated to program consecutive memory locations. The first memory location is FFFh (the FUSE word register), followed by 000h, 001h, 002h, and so on up to the top memory address, 7FFh. The programmer can skip over any number of memory locations by repeating the "Increment Address" command consecutively.

8.2.4 Exiting the ISP Mode

Exiting from the ISP mode must be done according to the following protocol to prevent possible damage to system components:

1. The programmer drops the voltage on the OSC1 pin from the programming voltage (VPP = 12.5 V) to logic zero. This is a signal to exit from the ISP mode.
2. On the next rising clock edge after the sync cycle, the SX device exits from the ISP mode and generates an internal reset signal that resets the device (programmer has to observe protocal until stage 2).
3. The programmer releases the OSC1 pin, allowing the SX device to begin normal operation.

8.3 Parallel Programming Mode

The parallel programming mode is faster than the ISP mode because you read and write data in parallel, 12 bits at a time, rather than serially. However, the parallel mode uses a larger number of pins, which means that you can use it only to program free-standing devices, or devices whose programming pins can be isolated from the rest of the system.

The parallel programming modes uses the following device pins:

- Vss (ground)
- Vdd to supply the normal operating voltage
- $\overline{\text{MCLR}}$/Vpp to supply the programming voltage (12.5 V)
- RA0-RA3 (Port A pins) to read and write the four low-order data bits
- RB0-RB7 (Port B pins) to read and write the eight high-order data bits
- RTCC to control programming operations
- OSC1 to increment the address pointer

8.3.1 Parallel Programming Operations

To use the parallel programming mode, you first power up the device in the normal operating mode, keeping the device in the reset state by holding the $\overline{\text{MCLR}}$ input low. You apply a 12-bit command to the Port A and Port B pins, and then apply the programming voltage (Vpp = 12.5 V) to the $\overline{\text{MCLR}}$ pin. This puts the device into the parallel programming mode and latches the command into the programming logic. The rise time of the signal on the $\overline{\text{MCLR}}$ pin should be at least 1.0 microsecond.

After the device has been put in the parallel programming mode, you use the port pins to read and write data, the RTCC pin to control the timing of programming operations, and the OSC1 pin to increment the address pointer.

To exit from the parallel programming mode, you can bring the $\overline{\text{MCLR}}$ back down to zero volts, which puts the device back into the reset state; or bring the Vdd pin down to zero volts, which shuts off power to the device.

It is not necessary to shut off the power between successive programming operations. For example, you can erase the memory and then proceed directly to programming the memory while leaving the power supplied to the Vdd pin. You only need to use the $\overline{\text{MCLR}}$ pin to latch the next command.

To protect the internal circuitry of the device, use a 100 (resistor between the $\overline{\text{MCLR}}$ pin and the Vpp power supply.

8.3.2 Commands

The programmer unit writes a 12-bit command to tell the SX programming logic what to do. Table 8-2 lists and describes the programming commands. Command codes are shown in hexadecimal format. Codes not listed in the table are reserved for future expansion.

Table 8-2 ISP Commands

Name	Code	Description
Erase	010h	Erase all EEPROM locations.
Read DEVICE Word	001h	Read DEVICE word (memory size configuration).
Read FUSEX Word	002h	Read FUSEX word (configuration options)
Program FUSEX Word in SX28 device	003h	Program FUSEX word (configuration options) in a 28-pin SX device
Program FUSEX Word in SX18 device	020h	Program FUSEX word (configuration options) in an 18-pin SX device
Read Data	004h	Read data words from memory.
Program Data	FFEh	Program data words into memory.

The RTCC pin is used to control the timing of programming operations. For each operation, the RTCC signal must be asserted for at least a specific period of time in order to work reliably. To determine the minimum required time, look in the Electrical Characterization appendix and find the time requirement for that particular operation.

8.3.3 Erasing the Memory

The "Erase" command erases all of the EEPROM memory, including the DEVICE word, FUSE word, and FUSEX word.

Before you perform an erase operation, you should read and save the FUSEX register so that you can restore the five high-order bits when you reprogram the device. These bits have factory-set values that must be maintained by the programming unit.

Figure 8-5 shows the signals used and the timing requirements for an "Erase" operation. The "Erase" signal at the bottom of the diagram is a chip-internal signal that is asserted during the erase operation. It becomes active as soon as the "Erase" command is decoded and is deactivated by the falling edge of the RTCC signal. The duration of this signal must be at least the value specified for this operation in the Electrical Characterization appendix.

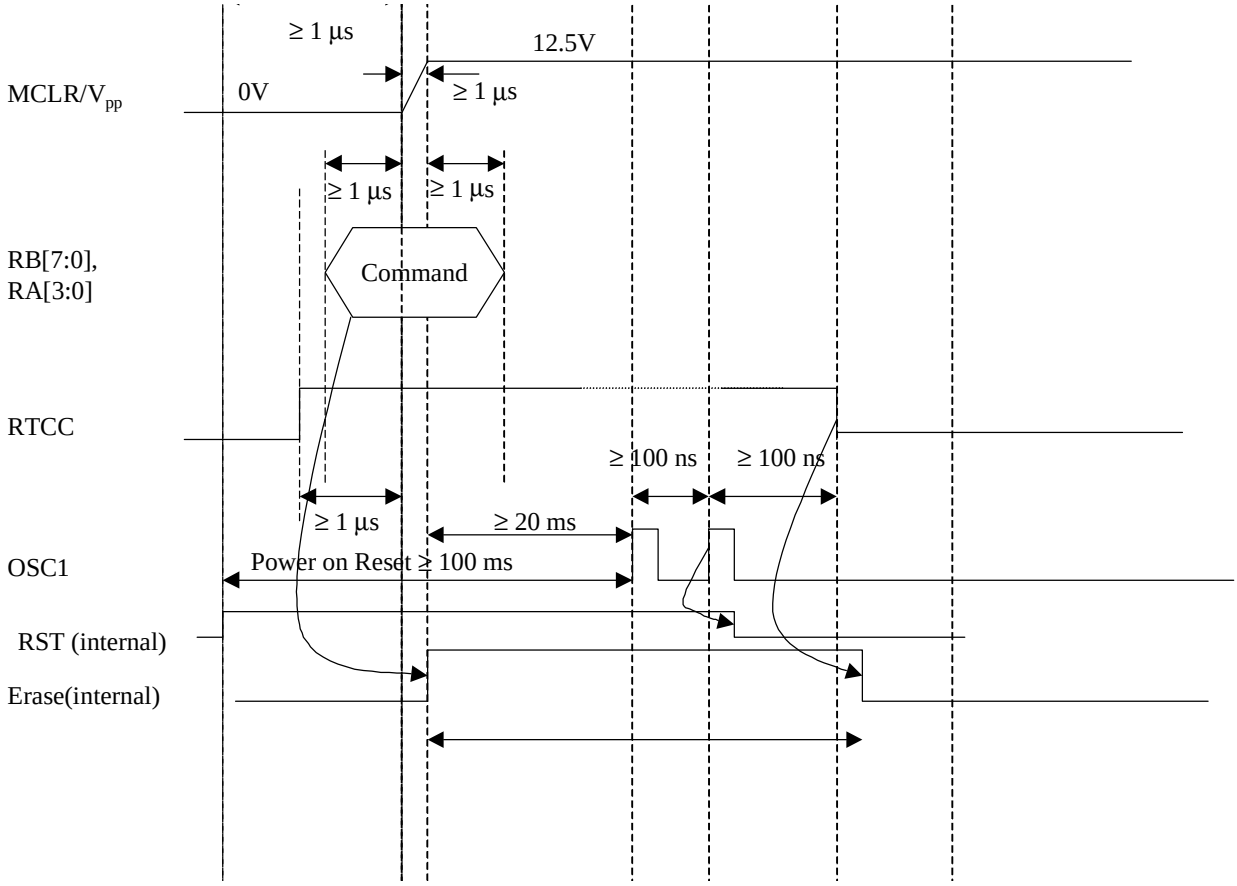

Figure 8-5 Erase Timing in Parallel Mode

This is the procedure shown in Figure 8-5:

1. If the power is not already on, apply normal power to the Vdd pin while holding the $\overline{\text{MCLR}}$ pin low.
2. Apply a logic high signal to the RTCC pin.

3. Apply the "Erase" command to the Port A and Port B pins.
4. Apply the programming voltage to the $\overline{\text{MCLR}}$ pin. This latches the "Erase" command and starts the erase operation.
5. After the erase time has elapsed, apply a logic low signal to the RTCC pin.
6. Bring the $\overline{\text{MCLR}}$ pin back down to zero.

Reading the Memory

To read the EEPROM program memory, you use the "Read Data" command in conjunction with the OSC1 pin, which operates as a device input. You use the OSC1 pin to increment the internal pointer for each successive memory address.

Figure 8-6 shows the signals used and the timing requirements for a "Read Data" operation. The "RST" and "Address" signals at the bottom of the diagram are chip-internal signals. The "RST" signal is an internal reset signal that is deactivated by the second pulse on the OSC1 pin. The "Address" waveform represents the internal address bus used to access the program memory. The address is incremented by each pulse on the OSC1 pin, starting with the second pulse.

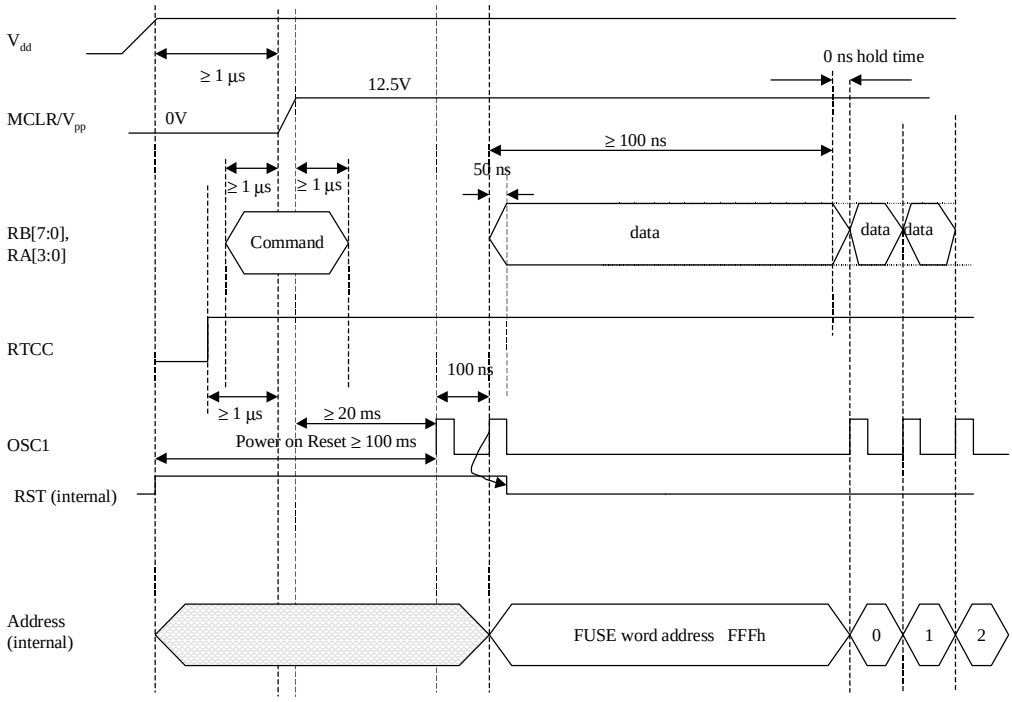

Figure 8-6 Read Timing in Parallel Mode

This is the procedure shown in Figure 8-6:

1. If the power is not already on, apply normal power to the Vdd pin while holding the $\overline{\text{MCLR}}$ pin low.
2. Apply a logic high signal to the RTCC pin.

3. Apply the "Read Data" command to the Port A and Port B pins.
4. Apply the programming voltage to the $\overline{MCLR}$ pin. This latches the "Read Data" command.
5. After the required time has elapsed (100 msec from power-up or 20 msec from latching the command), generate two pulses on the OSC1 pin. The second pulse takes the device out of the reset mode and generates the first device address, FFFh (the address of the FUSE word). The device reads the data from that address and places the 12-bit result on the Port A and Port B pins, allowing the programmer unit to read the data.
6. Generate a single pulse on the OSC1 pin. This advances the address to the next memory location (000h comes after FFFh) and reads the data from that memory location.
7. Repeat step 6 to read successive memory locations. Do this step 2,048 times to read all memory-mapped program locations from 000h through 7FFh.
8. Bring the $\overline{MCLR}$ pin back down to zero.

Programming the Memory

To write to the EEPROM program memory, you use the "Program Data" command in conjunction with the RTCC and OSC1 pins, which operate as device inputs. You use the RTCC pin to tell the device when to write the data and read back the data, and the OSC1 pin to increment the internal memory address pointer.

Figure 8-7 shows the signals used and the timing requirements for a "Program Data" operation. The "RST" and "Address" signals at the bottom of the diagram are chip-internal signals.

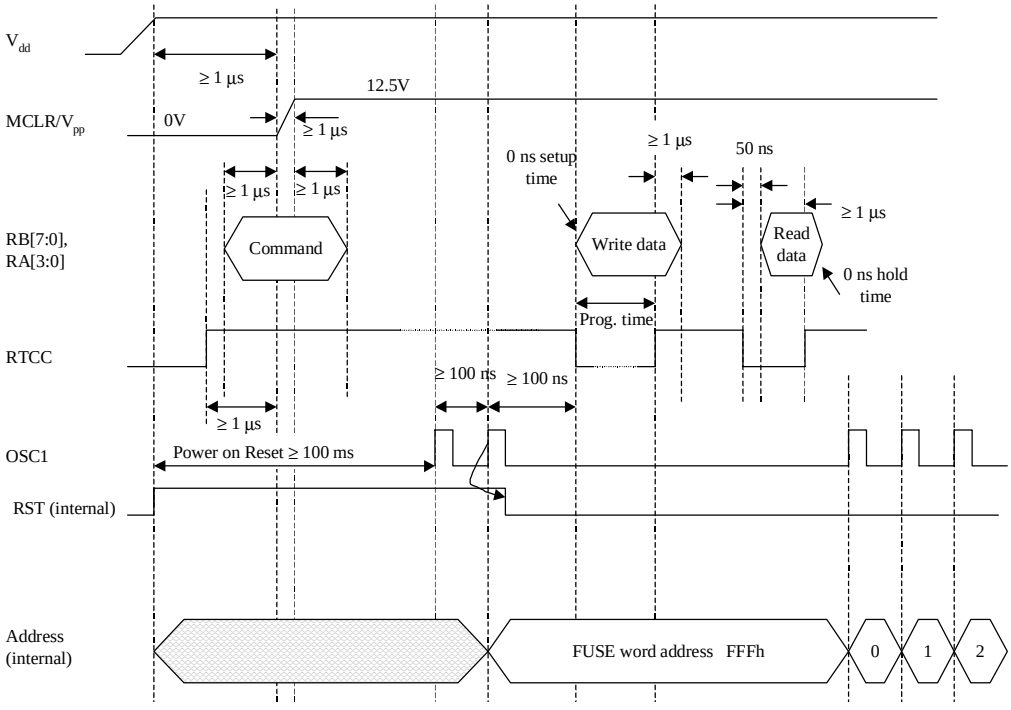

Figure 8-7 Program Timing in Parallel Mode

This is the procedure shown in Figure 8-7:

1. If the power is not already on, apply normal power to the Vdd pin while holding the $\overline{\text{MCLR}}$ pin low.
2. Apply a logic high signal to the RTCC pin.
3. Apply the "Program Data" command to the Port A and Port B pins.
4. Apply the programming voltage to the $\overline{\text{MCLR}}$ pin. This latches the "Program Data" command.
5. After the required time has elapsed, generate two pulses on the OSC1 pin. The second pulse takes the device out of the reset mode and generates the first device address, FFFh (the address of the FUSE word).
6. Apply the 12-bit data word to the Port A and Port B pins
7. Pull the RTCC pin low. This causes the device to read the data on the port pins and write that data to the program memory. After the required time has elapsed, pull the RTCC pin high.
8. Pull the RTCC pin low. This puts the device through a read cycle (as described in the previous section), allowing the programmer unit to verify that the data word has been written correctly to the program memory location. After reading is done, drive the RTCC pin high again.
9. Generate a single pulse on the OSC1 pin. This advances the address to the next memory location (000h comes after FFFh). Repeat to skip over multiple memory locations.
10. Repeat steps 6 through 9 to program successive memory locations. Do these steps 2,048 times to program all memory-mapped program locations, from 000h to 7FFh.
11. Bring the $\overline{\text{MCLR}}$ pin back down to zero.

At each new address, the first active-low pulse on the RTCC pin writes a 12-bit data word, and the second such pulse on the RTCC pin reads back the written data word. You can simply write and read the data and then immediately proceed to the next / skip n memory location by generating another pulse/pulses on the OSC1 pin.

Reading and Writing the FUSEX and DEVICE Words

The DEVICE word and FUSEX word are not memory-mapped in the program address space. They can be accessed by using the following special-purpose programming commands:

- Read DEVICE Word, code 001h
- Read FUSEX Word, code 002h
- Program FUSEX Word in SX28 device, code 003h
- Program FUSEX Word in SX18 device, code 020h

The procedures for reading and writing these registers is the same as for reading or writing the main program memory, except for the command code and OSC1 signal. Because you access only one register with each command, you just leave the OSC1 pin low during the read or write procedure.

