Xicor®

SerialFlash DATA BOOK

President's Message

Dear Customer:

Total customer satisfaction is Xicor's number one goal. Xicor provides an extensive product offering to satisfy your needs in field-programmable nonvolatile microperipherals and support memory chips such as SerialFlash Memory, E^2PROMs, NOVRAMs, E^2POTs, and others. These CMOS products are available in a wide variety of speeds, voltages, package types, and a variety of interface configurations. The majority of the products are offered with extended temperature ranges, and many comply with all of the requirements of MIL-STD-883 Revision C for Class B products.

Xicor has shipped to its customers more than 300 million units. New, innovative products continuously join them as a result of our extensive research and development activities. Xicor's worldwide sales, marketing and applications organizations are dedicated to supporting your requirements. We appreciate your business and look forward to supplying your present and future requirements.

Raphael Klein
President
September, 1996

September 1996
Printed in U.S.A.

Xicor offers the best applications support in the industry to our customers.

1. **Field Applications Engineers in each sales office**
2. **Factory Applications Support HOTLINE at (408) 432-8888**
3. **On line Applications Bulletin Board (800) 258-8864 —14.4K baud**
4. **Interfacing Software available on the Applications Bulletin Board**
5. **Latest Datasheets and Application Notes available by FaxBack (408) 954-1627 or (408) 954-1711**
6. **World Wide Web design resource at http://www.xicor.com**
7. **Xicor's e-mail address: info@smtpgate.xicor.com**

U.S. SALES OFFICES

Corporate Office
Xicor, Inc.
1511 Buckeye Drive
Milpitas, CA 95035
Phone: 408/432-8888
Fax: 408/432-0640

Northeast Region
Xicor, Inc.
1344 Main Street
Waltham, MA 02154
Phone: 617/899-5510
Fax: 617/899-6808
Email: xicor-ne@smtpgate.xicor.com

Southeast Region
Xicor, Inc.
100 East Sybelia Ave., Suite 355
Maitland, FL 32751
Phone: 407/740-8282
Fax: 407/740-8602
Email: xicor-se@smtpgate.xicor.com

Mid-Atlantic Region
Xicor, Inc.
50 North Street
Danbury, CT 06810
Phone: 203/743-1701
Fax: 203/794-9501
Email: xicor-ma@smtpgate.xicor.com

North Central Region
Xicor, Inc.
810 South Bartlett Road, Suite 103
Streamwood, IL 60107
Phone: 708/372-3200
Fax: 708/372-3210
Email: xicor-nc@smtpgate.xicor.com

South Central Region
Xicor, Inc.
11884 Greevville Ave., Suite 102
Dallas, TX 75243
Phone: 214/669-2022
Fax: 214/644-5835
Email: xicor-sc@smtpgate.xicor.com

Southwest Region
Xicor, Inc.
4100 Newport Place Drive, Suite 710
Newport Beach, CA 92660
Phone: 714/752-8700
Fax: 714/752-8634
Email: xicor-sw@smtpgate.xicor.com

Northwest Region
Xicor, Inc.
2700 Augustine Drive, Suite 219
Santa Clara, CA 95054
Phone: 408/292-2011
Fax: 408/980-9478
Email: xicor-nw@smtpgate.xicor.com

INTERNATIONAL SALES OFFICES

EUROPE

Northern Europe
Xicor, Ltd.
Grant Thornton House
Witan Way
Witney
Oxford 0X8 6FE
UK
Phone: (44) 19/937.00544
Fax: (44) 19/937.00533
Email: xicor-uk@smtpgate.xicor.com

Central Europe
Xicor GmbH
Technopark Neukeferloh
Bretonischer Ring 15
85630 Grasbrunn bei Muenchen
Germany
Phone: (49) 89/461.0080
Telex: (841) 5213883
Fax: (49) 89/460.5472
Email: xicor-gm@smtpgate.xicor.com

ASIA/PACIFIC

Japan
Xicor Japan K.K.
Suzuki Building, 4th Floor
1-6-8 Shinjuku, Shinjuku-ku
Tokyo 160
Japan
Phone: (81) 33/225.2004
Fax: (81) 33/225.2319
Email: xicor-jp@smtpgate.xicor.com

Mainland China
Taiwan/Hong Kong
Xicor, Inc.
4100 Newport Place Drive, Suite 710
Newport Beach, CA 92660
Phone: 714/752-8700
Fax: 714/752-8634
Email: xicor-5w@smtpgate.xicor.com

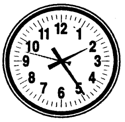
Singapore/Malaysia/India
Xicor, Inc.
2700 Augustine Drive, Suite 219
Santa Clara, CA 95054
Phone: 408/292-2011
Fax: 408/980-9478
Email: xicor-nw@smtpgate

Korea
Xicor Korea
27th Fl., Korea World Trade Ctr.
159, Samsung-dong
Kangnam Ku
Seoul 135-729
Korea
Phone: (82) 2551.2750
Fax: (82) 2551.2710
Email: xicor-ka@smtpgate.xicor.com

( ) = Country Code

Let our Applications Team help turn your ideas into reality!

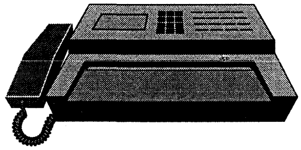
NEW—Xicor's FaxBack Service!



This year Xicor introduced an automated FaxBack response system, available twenty-four hours-a-day, seven days-a-week.

Up-to-date index provides Xicor's latest product offerings. Xicor FaxBack provides a selection index for product information. To receive this index, simply call (408) 954-1627 and request fax selection number, "2000."

Once you've received this index you can scan under the "part number" column to locate the particular data sheet of interest. By using the corresponding "FaxBack number" you can select the product data sheet you wish to receive. Within minutes your request will be delivered.



A touch-tone telephone and a fax machine/fax modem are all you need to instantly retrieve information about Xicor products. In addition, Xicor's FaxBack is also accessible from the Application Bulletin Board (BBS).

FREE—Applications Software!

Xicor offers an Applications Bulletin Board which contains applications information, product information, and interface software routines for a variety of Xicor products.

The BBS may be accessed at (800) 258-8864 or (408) 943-0655. Information is partitioned into the following file libraries based upon the topic or product. These libraries are currently available.

INTEL	Interface Routines for Intel Microcontrollers
MOTOROLA	Interface Routines for Motorola Microcontrollers
ZILOG	Interface Routines for Zilog Microcontrollers
NEC	Interface Routines for NEC Microcontrollers
PRGRMR	Latest Xicor Programmer Software
SLIC	The SLIC (Self Loading Integrated Code) Support Files
EEPOT	Interface Routines for Xicor's E^{2}POTs
SECURE	Interface Routines for PASS (Password Access Security Supervisor)
MPS	Interface Routines for Xicor MPS Devices
LapKit/51	Xicor PC Keyboard Controller and Power Management Chipset
HITACHI	Interface routines for H/8 Microcontroller
Misc.	Collection of some popular application notes and sample interface code
Public	FreeWare contributed by various sources

Xicor®, AUTOSTORE®, E^{2}POT®, SLIC®, and DirectWrite® are registered trademarks of Xicor, Inc. SerialFlash™ Memory, Block Lock™ Protection, Identi™ PROM, CONCURRENT READWRITE™, Classified Erase™, PushPot™, MPS™, PASS™ E^2 are trademarks of Xicor, Inc. NOVRAM is Xicor's Nonvolatile Static RAM Device. All other trademarks are the property of their respective owners.

LIMITED WARRANTY

Devices sold by Xicor, Inc. are covered by the warranty and patent indemnification provisions appearing in its Terms of Sale only. Xicor, Inc. makes no warranty, express, statutory, implied, or by description regarding the information set forth herein or regarding the freedom of the described devices from patent infringement. Xicor, Inc. makes no warranty of merchantability or fitness tor any purpose. Xicor, Inc. reserves the right to discontinue production and change specifications and prices at any time and without notice.

Xicor, Inc. assumes no responsibility for the use of any circuitry other than circuitry embodied in a Xicor, Inc. product. No other circuits, patents, licenses are implied.

U.S. PATENTS

Xicor products are covered by one or more of the following U.S. Patents: 4,263,664; 4,274,012; 4,300,212; 4,314,265; 4,326,134; 4,393,481; 4,404,475; 4,450,402; 4,486,769; 4,488,060; 4,520,461; 4,533,846; 4,599,706; 4,617,652; 4,668,932; 4,752,912; 4,829,482; 4,874,967; 4,883,976; 4,980,859; 5,012,132; 5,003,197; 5,023,694. Foreign patents and additional patents pending.

LIFE RELATED POLICY

In situations where semiconductor component failure may endanger life, system designers using this product should design the system with appropriate error detection and correction, redundancy and back-up features to prevent such an occurrence.

Xicor's products are not authorized for use as critical components in life support devices or systems.

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform, when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.
2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its satety or effectiveness.

Table of Contents

- Bold items included in this data book
- FaxBack numbers are listed under Alphanumeric Index.

SECTION 1—2-WIRE SERIALFLASH

SECTION 2—SPI SERIALFLASH

SECTION 3—SECURITY FLASH PRODUCTS

SECTION 4—MICRO PORT SAVER MPS™ SERIALFLASH

SECTION 5—DEVELOPMENT SYSTEMS

SECTION 6—APPLICATION NOTES

Table of Contents (Cont.)

SECTION 7—GENERAL INFORMATION

Alpha-Numeric Index

Application Note Index

SerialFlash Application Notes		Code[1] Printed	Avail	Faxback
AN54	*Programmer Vendors Supporting Xicor Devices*			*#3054*
AN75	*Interface Software for Xicor SPI Memories*	*X*	*X*	*#3075*
AN76	*Interfacing the X24F016/032/064 SerialFlash Memories to the Motorola 68HC11 Microcontroller*		*X*	*#3076*
AN77	*Interfacing the X25F008/016/032/064 SerialFlash Memories to the Motorola 68HC11 Microcontroller SPI Port*		*X*	*#3077*
AN78	*Interfacing the X24F016/032/064 SerialFlash Memories to the 8051 Microcontrollers*		*X*	*#3078*
AN79	*Interfacing the X25F008/016/032/064 SerialFlash Memories to 8051 Microcontrollers using the Serial Communication Port (UART)*			*#3079*
AN83	*Interfacing to the X76F041 PASS SecureFlash*	*X*	*X*	*#3083*
AN84	*Interfacing to the X24F128 2-Wire SerialFlash™ Memory*	*X*	*X*	*#3084*
AN87	*Block Lock™ and Programmable Hardware Write Protection for Xicor's 2-Wire Serial Devices*		*X*	*#3087*

Other related Application Notes

AN57	*What is MPS E²?*		*X*	*#3057*
AN61	*Programmable Write Protection for High Density Serial SecureFlash PROMs*			*#3061*
AN81	*Interfacing 2-Wire Serial Memories to Hitachi H8/3000*		*X*	*#3081*
AN82	*Interfacing 3-Wire Serial Memories to Hitachi H8/3000*		*X*	*#3082*

[1] Printed code is contained in this data book
Available code can be obtained from the World Wide Web or by contacting our BBS.

Precautions for the Handling of MOS Devices

Xicor products are designed with effective input protection to prevent damage to the devices under most conditions. However, any MOS circuit can be catastrophically damaged by excessive electrostatic discharge or transient voltages. The following procedures are recommended to avoid accidental circuit damage.

1. Testing MOS Circuits:

1. All units should be handled directly from the conductive or antistatic plastic tube in which they were shipped if possible. This action minimizes touching of individual leads.
2. If units are to be tested without using the tube carrier, the following precautions should be taken:
   a. Table surfaces which potentially will come in contact with the devices either directly or indirectly (such as through shipping tubes) must be metal or of another conductive material and should be electrically connected to the test equipment and to the test operator (a grounding bracelet is recommended).
   b. The units should be transported in bundled antistatic tubes or metal trays, both of which will assume a common potential when placed on a conductive table top.
   c. Do not band tubes together with adhesive tape or rubber bands without first wrapping them in a conductive layer.

II. Test Equipment (Including Environmental Equipment):

1. All equipment must be properly returned to the same reference potential (ground) as the devices, the operator, and the container for the devices.
2. Devices to be tested should be protected from high voltage surges developed by:
   a. Turning electrical equipment on or off.
   b. Relay switching.
   c. Transients from voltage sources (AC line or power supplies).

III. Assembling MOS Devices Onto PC Boards:

1. The MOS circuits should be mounted on the PC board last.
2. Similar precautions should be taken as in Item 1 above, at the assembly work station.
3. Soldering irons or solder baths should be at the same reference (ground) potential as the devices.
4. Plastic materials which are not antistatic treated should be kept away from devices as they develop and maintain high levels of static charge.

IV. Device Handling:

1. Handling of devices should be kept to a minimum. If handling is required, avoid touching the leads directly.

V. General:

1. The handler should take every precaution that the device will see the same reference potential when moved.
2. Anyone handling individual devices should develop a habit of first touching the container in which the units are stored before touching the units.
3. Before placing the units into a PC board, the handler should touch the PC board first.
4. Personnel should not wear clothing which will build up static charge. They should wear smocks and clothing made of 100% cotton rather than wool or synthetic fibers.
5. Be careful of electrostatic build up through the movement of air over plastic material. This is especially true of acid sinks.
6. Personnel or operators should always wear grounded wrist straps when working with MOS devices.
7. A 1MΩ resistance ground strap is recommended and will protect people up to 5000V AC RMS or DC by limiting current to 5mA.
8. Antistatic ionized air equipment is very effective and useful in preventing electrostatic damage.
9. Low humidity maximizes potential static problems. Maintaining humidity levels above 45% is one of the most effective ways to guard against static handling problems.

PRODUCT SELECTION GUIDE

SERIALFLASH MEMORY

	BIT		CLOCK	PACKAGE	(N0.PINS)			TEMP. RANGE			SUPPLY VOLTAGE 1.8V - 3.6V or 5V±10%	
DEVICE	DENSITY	ORG.	RATE	P(8)	S(8)	V(14)	V(20)	COM	EXT	IND		FEATURES
X24F016	16K	X8	100KHZ	◆	◆	◆		◆	◆	◆	◆	2-WIRE INTERFACE, BLOCKLOCK
X24F032	32K	X8	100KHZ	◆	◆	◆		◆	◆	◆	◆	2-WIRE INTERFACE, BLOCKLOCK
X24F064	64K	X8	100KHZ	◆	◆		◆	◆	◆	◆	◆	2-WIRE INTERFACE, BLOCKLOCK
X24F128	128K	X8	1MHZ	◆	S (16)			◆	◆	◆	◆	2-WIRE INTERFACE, BLOCKLOCK
X25F008	8K	X8	1MHZ	◆	◆	◆		◆	◆	◆	◆	SPI INTERFACE, BLOCKLOCK
X25F016	16K	X8	1MHZ	◆	◆	◆		◆	◆	◆	◆	SPI INTERFACE, BLOCKLOCK
X25F032	32K	X8	1MHZ	◆	◆	◆		◆	◆	◆	◆	SPI INTERFACE, BLOCKLOCK
X25F047	4K	X8	1MHZ	◆	◆			◆	◆	◆	◆	SPI INTERFACE, BLOCKLOCK
X25F064	64K	X8	1MHZ	◆	◆		◆	◆	◆	◆		SPI INTERFACE, BLOCKLOCK
X25F087	8K	X8	1MHZ	◆	◆			◆	◆	◆	◆	SPI INTERFACE, BLOCKLOCK
X25F128	128K	X8	1MHZ	◆	S (16)			◆	◆	◆	◆	SPI INTERFACE, BLOCKLOCK
X76F041	4K	X8	1MHZ	◆	◆			◆	◆	◆	3V to 3.6V	2-WIRE INTERFACE, BLOCKLOCK
X84F008	8K	X8	5MHZ	◆	◆	◆		◆	◆	◆	2.4V to 3.6V	MPS™ INTERFACE, BLOCKLOCK
X84F016	16K	X8	5MHZ	◆	◆	◆		◆	◆	◆	2.4V to 3.6V	MPS™ INTERFACE, BLOCKLOCK
X84F064	64K	X8	5MHZ	◆	S (14)		◆	◆	◆	◆	2.4V to 3.6V	MPS™ INTERFACE, BLOCKLOCK
X84F128	128K	X8	5MHZ	◆	S (16)			◆	◆	◆	2.4V to 3.6V	MPS™ INTERFACE, BLOCKLOCK

Xicor®

7018-4.0 8/26/96 T0/C0/D0 SH

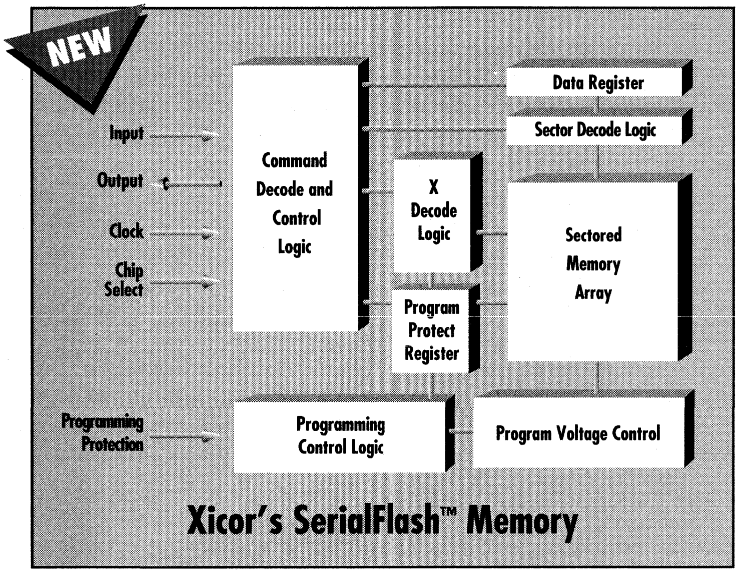
Design Engineering Bulletin

Xicor®

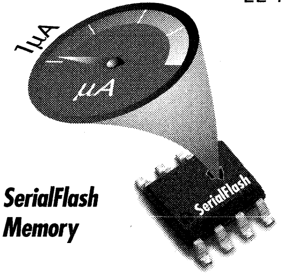
NUMBER 11

New Product and Applications Information for Design Engineers

SerialFlash™ Memory combines the performance features of Flash memory with the space efficiency associated with a serial interface.



"It was natural that sooner or later somebody would put two-and-two together."
— EE Times



SerialFlash Memory

SerialFlash Memory has been designed to meet the low power needs of Portable Battery Powered Products. When idle it consumes less than 5 microwatts, resulting in extended battery life.

If your design has a tight space requirement, SerialFlash Memory is an efficient solution. Unlike traditional Flash or parallel Nonvolatile memory, SerialFlash Memory is available in industry standard 150 mil wide JEDEC SOIC and TSSOP packages.

SerialFlash™ Features!

- ✓ Low Cost
- ✓ Small Package
- ✓ Low Voltage, 1.8 Volts
- ✓ Low Standby Current, 1µA
- ✓ 1 MHz Data Rate
- ✓ 32 Byte Sector Programming
- ✓ SPI Interface
- ✓ 2-Wire Interface
- ✓ Block Lock™ Data Protection

Features "Univolt" single voltage for program and read.

Choose the Optimum Interface and Density to Meet Your System Needs:

PART NO.	DENSITY	ORG.	PACKAGE	TEMP RANGE	INTERFACE
X25F008	8K BITS	1Kx8	P,S,V	COM, EXT, IND.	SPI
X25F016	16K BITS	2Kx8	P,S,V	COM, EXT, IND.	SPI
X25F032	32K BITS	4Kx8	P,S,V	COM, EXT, IND.	SPI
X25F064	64K BITS	8Kx8	P,S,V	COM, EXT, IND.	SPI
X25F128	128K BITS	16Kx8	P,S	COM, EXT, IND.	SPI
X24F016	16K BITS	2Kx8	P,S,V	COM, EXT, IND.	2-WIRE
X24F032	32K BITS	4Kx8	P,S,V	COM, EXT, IND.	2-WIRE
X24F064	64K BITS	8Kx8	P,S,V	COM, EXT, IND.	2-WIRE
X24F128	128K BITS	16Kx8	P,S	COM, EXT, IND.	2-WIRE

APPLICATION NOTE

AVAILABLE

AN84

128K X24F128 16K x 8 Bit

2-Wire SerialFlash with Block Lock™ Protection

FEATURES

- **1.8V to 3.6V or 5V "Univolt" Read and Program Power Supply Versions**
- **Low Power CMOS**
  - **—Active Read Current Less Than 1mA**
  - **—Active Program Current Less Than 3mA**
  - **—Standby Current Less Than 1µA**
- **100KHz 2-Wire Serial Interface**
- **Internally Organized 16K x 8**
- **Extended Temperature Range**
  - **— –20°C to +85°C**
- **32 Byte Sector Program Mode**
  - **—Minimizes Total Program Time Per Byte**
- **Programmable Block Lock Protection**
  - **—Block Lock (0, 1/4, 1/2, or all of SerialFlash)**
  - **—Software Program Protection**
  - **—Programmable Hardware Program Protect**
- **Bidirectional Data Transfer Protocol**
- **Self-Timed Program Cycle**
  - **—Typical Program Cycle Time of 5ms**
- **High Reliability**
  - **—Endurance: 100,000 Cycles**
  - **—Data Retention: 100 Years**
- **Available Packages**
  - **—8-Lead DIP**
  - **—16-Lead SOIC**

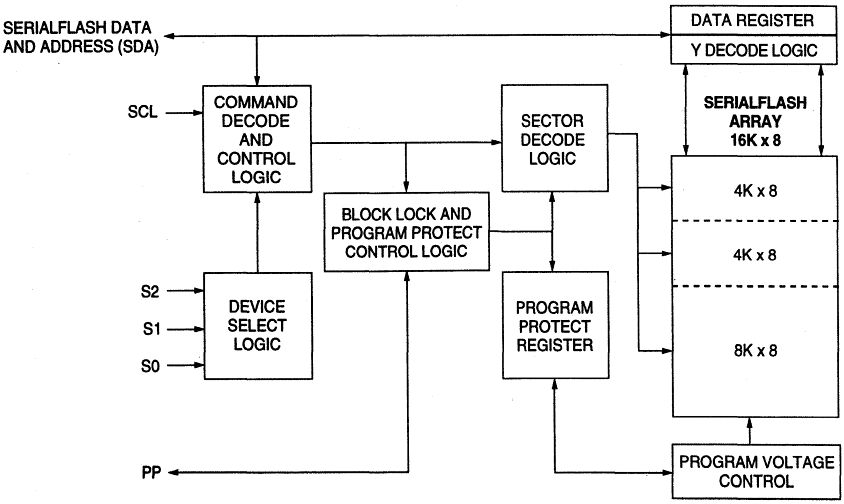
DESCRIPTION

The X24F128 is a CMOS SerialFlash Memory, internally organized 16K x 8. The device features a serial interface and software protocol allowing operation on a simple two wire bus.

Three device select inputs (S_0–S_2) allow up to eight devices to share a common two wire bus.

A Program Protect Register at the address location FFFFh provides three program protection features: Software Program Protect, Block Lock Protect, and Hardware Program Protect. The Software Program Protect feature prevents any nonvolatile writes to the device until the PEL bit in the Program Protect Register is set. The Block Lock Protection feature allows the user to individually block protect four blocks of the array by programming two bits in the Program Protect Register. The Programmable Hardware Program Protect feature allows the user to install the device with PP tied to V_{CC}, program the entire memory array in circuit, and then enable the hardware program protection by programming a PPEN bit in the Program Protect Register. After this, selected blocks of the array, including the Program Protect Register itself, are permanently protected from being erased.

FUNCTIONAL DIAGRAM



7012 ILL F01.4

Xicor SerialFlash Memories are designed and tested for applications requiring extended endurance. Inherent data retention is greater than 100 years.

PIN DESCRIPTIONS

Serial Clock (SCL)

The SCL input is used to clock all data into and out of the device.

Serial Data (SDA)

SDA is a bidirectional pin used to transfer data into and out of the device. It is an open drain output and may be wire-ORed with any number of open drain or open collector outputs.

An open drain output requires the use of a pull-up resistor. For selecting typical values, refer to the Pull-up resistor selection graph at the end of this data sheet.

Device Select (S_0, S_1, S_2)

The device select inputs (S_0, S_1, S_2) are used to set the first three bits of the 8-bit slave address. This allows up to eight devices to share a common bus. These inputs can be static or actively driven. If used statically they must be tied to V_{SS} or V_{CC} as appropriate. If actively driven, they must be driven with CMOS levels.

Program Protect (PP)

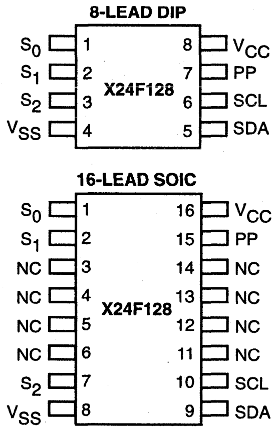
The Program Protect input controls the Hardware Program Protect feature. When held LOW, hardware program protection is disabled and the device can be programmed normally. When this input is held HIGH, and the PPEN bit in the Program Protect Register is set HIGH, program protection is enabled, and nonvolatile writes are disabled to the selected blocks as well as the Program Protect Register itself.

PIN NAMES

Symbol	Description
S_0, S_1, S_2	Device Select Inputs
SDA	Serial Data
SCL	Serial Clock
PP	Program Protect
V_{SS}	Ground
V_{CC}	Supply Voltage
NC	No Connect

7012 FRM T01

PIN CONFIGURATION



7012 ILL F02.1

DEVICE OPERATION

The device supports a bidirectional, bus oriented protocol. The protocol defines any device that sends data onto the bus as a transmitter, and the receiving device as the receiver. The device controlling the transfer is a master and the device being controlled is the slave. The master will always initiate data transfers, and provide the clock for both transmit and receive operations. Therefore, the X24F128 will be considered a slave in all applications.

Clock and Data Conventions

Data states on the SDA line can change only during SCL LOW. SDA state changes during SCL HIGH are reserved for indicating start and stop conditions. Refer to Figures 1 and 2.

Start Condition

All commands are preceded by the start condition, which is a HIGH to LOW transition of SDA when SCL is HIGH. The device continuously monitors the SDA and SCL lines for the start condition and will not respond to any command until this condition has been met.

Figure 1. Data Validity

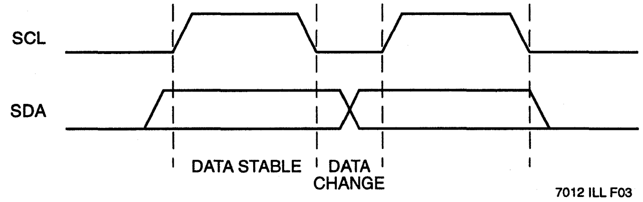
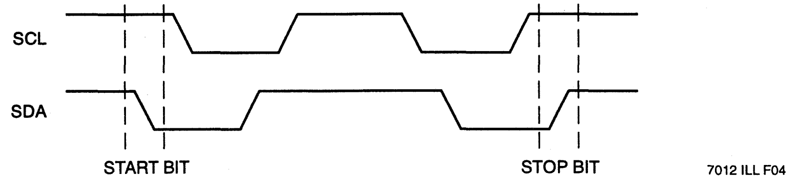


Figure 2. Definition of Start and Stop



Stop Condition

All communications must be terminated by a stop condition, which is a LOW to HIGH transition of SDA when SCL is HIGH. The stop condition is also used to place the device into the standby power mode after a read sequence. A stop condition can only be issued after the transmitting device has released the bus.

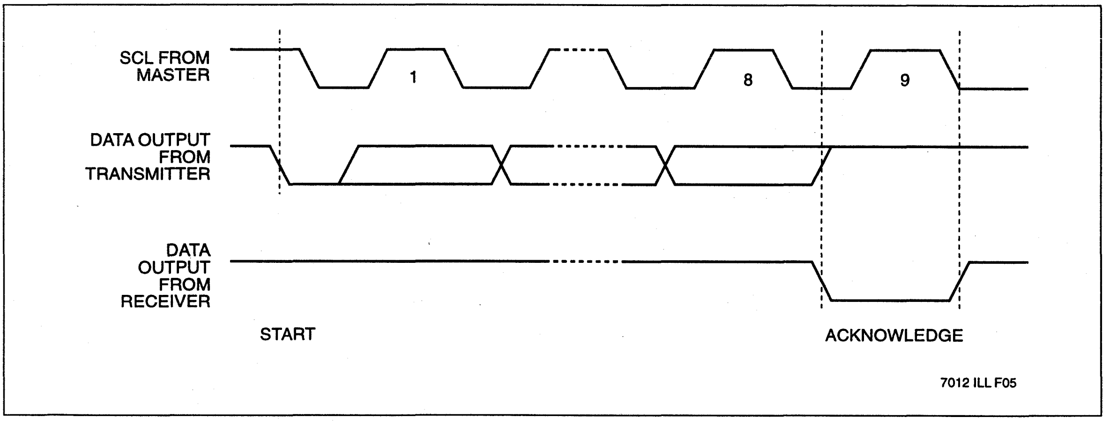
Acknowledge

Acknowledge is a software convention used to indicate successful data transfer. The transmitting device, either master or slave, will release the bus after transmitting eight bits. During the ninth clock cycle the receiver will pull the SDA line LOW to acknowledge that it received the eight bits of data. Refer to Figure 3.

The device will respond with an acknowledge after recognition of a start condition and its slave address. If both the device and a program operation have been selected, the device will respond with an acknowledge after the receipt of each subsequent byte.

In the read mode the device will transmit eight bits of data, release the SDA line and monitor the line for an acknowledge. If an acknowledge is detected and no stop condition is generated by the master, the device will continue to transmit data. If an acknowledge is not detected, the device will terminate further data transmissions. The master must then issue a stop condition to return the device to the standby power mode and place the device into a known state.

Figure 3. Acknowledge Response From Receiver



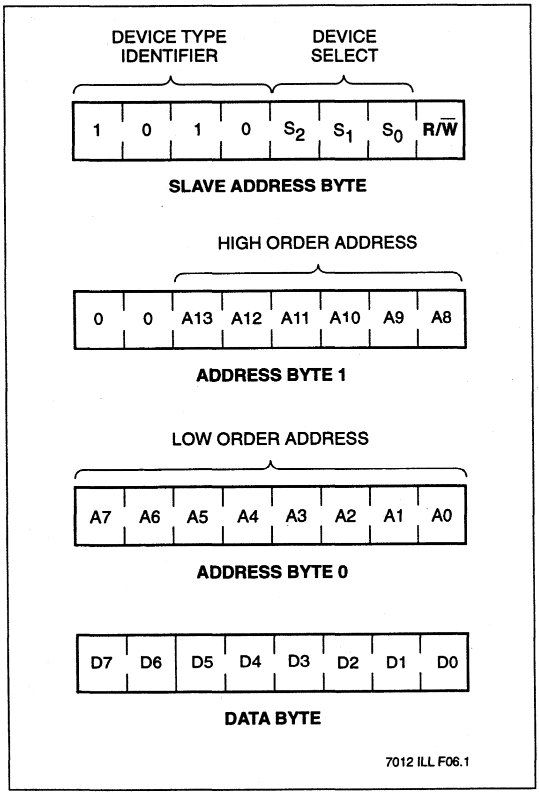
DEVICE ADDRESSING

Following a start condition, the master must output the address of the slave it is accessing. The first four bits of the Slave Address Byte are the device type identifier bits. These must equal "1010". The next 3 bits are the device select bits S_0, S_1, and S_2. This allows up to 8 devices to share a single bus. These bits are compared to the S_0, S_1, and S_2 device select input pins. The last bit of the Slave Address Byte defines the operation to be performed. When the R/$\overline{W}$ bit is a one, then a read operation is selected. When it is zero then a program operation is selected. Refer to figure 4. After loading the Slave Address Byte from the SDA bus, the device compares the device type bits with the value "1010" and the device select bits with the status of the device select input pins. If the compare is not successful, no acknowledge is output during the ninth clock cycle and the device returns to the standby mode.

The byte address is either supplied by the master or obtained from an internal counter, depending on the operation. When required, the master must supply the two Address Bytes as shown in figure 4.

The internal organization of the E^2 array is 512 sectors by 32 bytes per sector. The sector address is partially contained in the Address Byte 1 and partially in bits 7 through 5 of the Address Byte 0. The specific byte address is contained in bits 4 through 0 of the Address Byte 0. Refer to figure 4.

Figure 4. Device Addressing

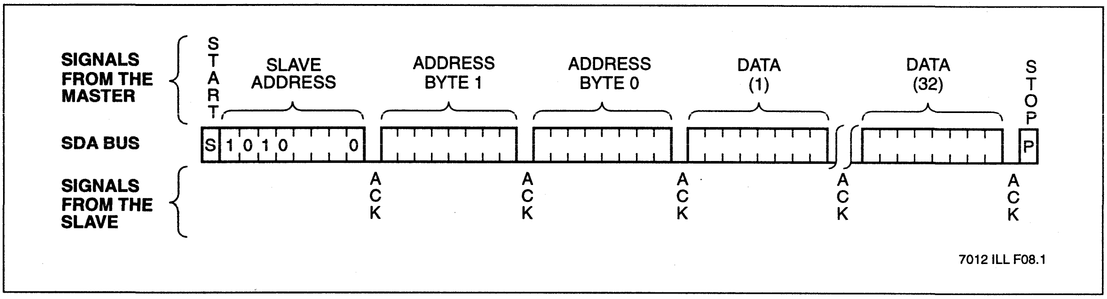


PROGRAMMING OPERATIONS

Sector Program Operation

The device executes a thirty-two byte sector program operation. For a sector program operation, the device requires the Slave Address Byte, Address Byte 1, and Address Byte 0. Address Byte 0 must contain the first byte of the sector to be programmed. Upon receipt of Address Byte 0, the device responds with an acknowledge, and waits for the first eight bits of data. After receiving the 8 bits of the first data byte, the device again responds with an acknowledge. The device will respond with an acknowledge after the receipt of each of 31 more bytes. Each time the byte address is internally incremented by one, while the sector address remains constant. When the counter reaches the end of the sector, the master terminates the data loading by issuing a stop condition, which causes the device to begin the nonvolatile write cycle. All inputs are disabled until completion of the nonvolatile write cycle. The SDA pin is at high impedance. Refer to figure 5 for the address, acknowledge, and data transfer sequence.

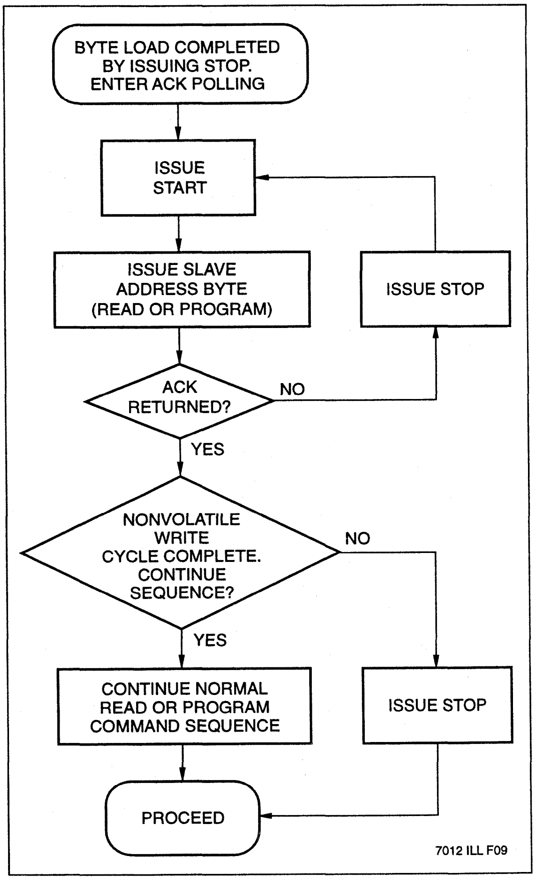
Figure 5. Sector Program Sequence



Acknowledge Polling

The maximum program cycle time can be significantly reduced using Acknowledge Polling. To initiate Acknowledge Polling, the master issues a start condition followed by the Slave Address Byte for a program or read operation. If the device is still busy with the nonvolatile write cycle, then no ACK will be returned. If the device has completed the nonvolatile write operation, an ACK will be returned and the host can then proceed with the read or program operation. Refer to figure 6.

Figure 6. Acknowledge Polling Sequence


BYTE LOAD COMPLETED BY ISSUING STOP. ENTER ACK POLLING
ISSUE START
ISSUE SLAVE ADDRESS BYTE (READ OR PROGRAM)
ISSUE STOP
ACK RETURNED?
NO
YES
NONVOLATILE WRITE CYCLE COMPLETE. CONTINUE SEQUENCE?
NO
YES
CONTINUE NORMAL READ OR PROGRAM COMMAND SEQUENCE
ISSUE STOP
PROCEED
7012 ILL F09

READ OPERATIONS

Read operations are initiated in the same manner as program operations with the exception that the R/W̄ bit of the Slave Address Byte is set to one. There are three basic read operations: Current Address Reads, Random Reads, and Sequential Reads.

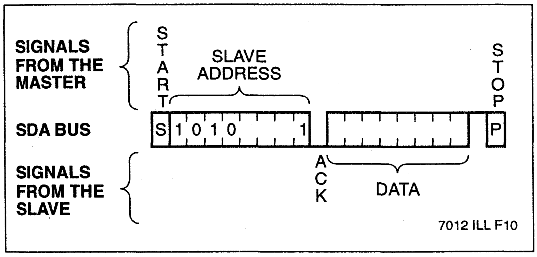
Current Address Read

Internally, the device contains an address counter that maintains the address of the last byte read or programmed, incremented by one. After a read operation from the last address in the array, the counter will "roll over" to the first address in the array. After a program operation to the last address in a given sector, the counter will "roll over" to the first address of the same sector.

Upon receipt of the Slave Address Byte with the R/W̄ bit set to one, the device issues an acknowledge and then transmits the byte at the current address. The master terminates the read operation when it does not respond with an acknowledge during the ninth clock and then issues a stop condition. Refer to figure 7 for the address, acknowledge, and data transfer sequence.

It should be noted that the ninth clock cycle of the read operation is not a "don't care." To terminate a read operation, the master must either issue a stop condition during the ninth cycle or hold SDA HIGH during the ninth clock cycle and then issue a stop condition.

Figure 7. Current Address Read Sequence



Random Read

Random read operation allows the master to access any memory location in the array. Prior to issuing the Slave Address Byte with the R/W̄ bit set to one, the master must first perform a "Dummy" program operation. The master issues the start condition and the Slave Address Byte with the R/W̄ bit low, receives an acknowledge, then issues Address Byte 1, receives another acknowledge, then issues Address Byte 0 containing the address of the byte to be read. After the device acknowledges receipt of Address Byte 0, the master issues another start condition and the Slave Address Byte with the R/W̄ bit set to one. This is followed by an acknowledge and then eight bits of data from the device. The master terminates the read operation by not responding with an acknowledge and then issuing a stop condition. Refer to figure 8 for the address, acknowledge, and data transfer sequence.

The device will perform a similar operation called "Set Current Address" if a stop is issued instead of the second start shown in figure 9. The device will go into standby mode after the stop and all bus activity will be ignored until a start is detected. The effect of this operation is that the new address is loaded into the address counter, but no data is output by the device.

The next Current Address Read operation will read from the newly loaded address.

Sequential Read

Sequential reads can be initiated as either a current address read or random read. The first byte is transmitted as with the other modes; however, the master now responds with an acknowledge, indicating it requires additional data. The device continues to output data for each acknowledge received. The master terminates the read operation by not responding with an acknowledge and then issuing a stop condition.

The data output is sequential, with the data from address n followed by the data from address n + 1. The address counter for read operations increments through all byte addresses, allowing the entire memory contents to be read during one operation. At the end of the address space the counter "rolls over" to address 0000h and the device continues to output data for each acknowledge received. Refer to figure 9 for the acknowledge and data transfer sequence.

Figure 8. Random Read Sequence

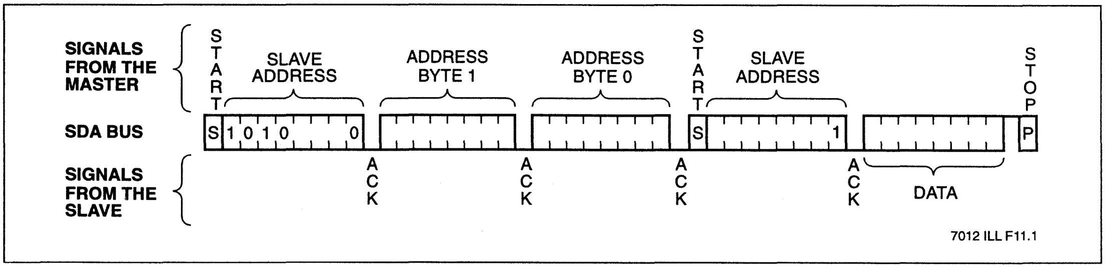
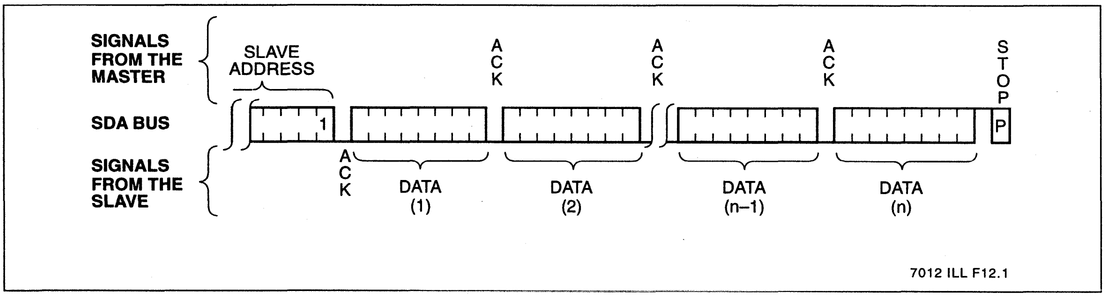


Figure 9. Sequential Read Sequence



PROGRAM PROTECT REGISTER (PPR)

Register Program Operation

The Program Protect Register can only be modified by programming one data byte directly to the address FFFFh as described below.

The data byte must contain zeroes where indicated in the procedural descriptions below; otherwise the operation will not be performed. Only one data byte is allowed for each Register Program Operation. The part will not acknowledge any data bytes after the first byte is entered. The user then has to issue a stop to initiate the nonvolatile write cycle that programs BL0, BL1, and PPEN to the nonvolatile bits. A stop must also be issued after volatile register program operations to put the device into Standby.

The state of the Program Protect Register can be read by performing a random read at FFFFh at any time. The part will reset itself after the first byte is read. The master should supply a stop condition to be consistent with the protocol. After the read, the address counter contains 0000h.

Program Protect Register: PPR (ADDR = FFFF$_h$)

7	6	5	4	3	2	1	0
PPEN	0	0	BL1	BL0	RPEL	PEL	0

PEL: Program Enable Latch (Volatile)

0 = PEL reset, programming disabled.

1 = PEL set, programming enabled.

RPEL: Register Program Enable Latch (Volatile)

0 = RPEL reset, programs to the Program Protect Register disabled.

1 = RPEL set, programs to the Program Protect Register enabled.

BL0, BL1: Block Lock Protect Bits (Nonvolatile)

The Block Lock Protect Bits, BL0 and BL1, determine which blocks of the array are protected. A program to a protected block of memory is ignored, but will receive an acknowledge. The master must issue a stop to put the part into standby, just as it would for a valid program; but the stop will not initiate an internal nonvolatile write cycle. See figure 10.

PPEN: Program Protect Enable Bit (Nonvolatile)

The Program Protect (PP) pin and the Program Protect Enable (PPEN) bit in the Program Protect Register control the Programmable Hardware Program Protection feature. Hardware Program Protection is enabled when the PP pin is HIGH and the PPEN bit is HIGH, and disabled when either the PP pin is LOW or the PPEN bit is LOW. When the chip is Hardware Program Protected, nonvolatile writes are disabled to the Program Protect Register, including the Block Lock Protect bits and the PPEN bit itself, as well as to the Block Lock protected sections in the memory array. Only the sections of the memory array that are not Block Lock protected, and the volatile bits PEL and RPEL, can be programmed. Note that since the PPEN bit is program protected, it cannot be changed back to a LOW state; so program protection is enabled as long as the PP pin is held HIGH. Figure 11 defines the program protect status for each combination of PPEN and PP.

Unused Bit Positions

Bits 0, 5 & 6 are not used. All programs to the PPR must have zeros in these bit positions. The data byte output during a PPR read will contain zeros in these bits.

Programming the PEL and RPEL bits

PEL and RPEL are volatile latches that power up in the LOW (disabled) state. While the PEL bit is LOW, program operations to any address other than FFFFh will be ignored (no acknowledge will be issued after the data byte). The PEL bit is set by programming 00000010 to address FFFFh. Once set, PEL remains HIGH until either it is reset to 0 (by programming 00000000 to FFFFh) or until power cycles. Programming PEL and RPEL does not cause a nonvolatile write cycle, so the device is ready for the next operation immediately after the stop condition.

The RPEL bit controls programming to the Block Lock Protect bits, BL0 and BL1, and the PPEN bit. If RPEL is 0 then no programming operations can be performed on BL0, BL1, or PPEN. RPEL is reset when power cycles or after any nonvolatile write, including those to the Block Lock Protect bits, the PPEN bit, or any sector in the memory array. RPEL must be reset before PEL can be reset. RPEL and PEL cannot be reset in one program operation. RPEL can also be reset by programming u00xy010 to FFFFh ONLY when the PPR is NOT protected. This is the same operation as in step 3 described below, and will result in programming BL0, BL1, and PPEN.

Programming to the BL and PPEN Bits

A 3 step sequence is required to change the nonvolatile Block Lock Protect or Program Protect Enable bits:

1) Set PEL=1, Program 00000010 to address FFFFh (Volatile Write Cycle.)

2) Set RPEL=1, Program 00000110 to address FFFFh (Volatile Write Cycle.)

3) Set BL1, BL0, and/or PPEN bits, Program u00xy010 to address FFFFh, where u=PPEN, x=BL1, and y=BL0. (Nonvolatile Write Cycle.)

The three step sequence was created to make it difficult to change the contents of the Program Protect Register accidentally. If PEL was set to one by a previous register program operation, the user may start at step 2. RPEL is reset to zero in step 3 so that user is required to perform steps 2 and 3 to make another change. RPEL must be 0 in step 3. If the RPEL bit in the data byte for step 3 is a one, then no changes are made to the Program Protect Register and the device remains at step 2.

The PP pin must be LOW or the PPEN bit must be LOW before a nonvolatile register program operation is initiated. Otherwise, the program operation will abort and the device will go into standby mode after the master issues the stop condition in step 3.

Step 3 is a nonvolatile write operation, requiring 10mS max to complete (acknowledge polling may be used to reduce this time requirement). It should be noted that step 3 MUST end with a stop condition. If a start condition is issued during or at the end of step 3 (instead of a stop condition) the device will abort the nonvolatile register program and remain at step 2. If the operation is aborted with a start condition, the master must issue a stop to put the device into standby mode.

Figure 10. Block Lock Protect Bits and Protected Addresses

BL1	BL0	Protected Addresses	Array Location
		X24F128	
0	0	None	No Protect
0	1	3000h - 3FFFh	Upper 1/4
1	0	2000h - 3FFFh	Upper 1/2
1	1	0000h - 3FFFh	Full Array

7012 FRM T02

Figure 11. PP Pin and PPEN Bit Functionality

PP	PPEN	Memory Array Not Lock Block Protected	Memory Array Block Lock Protected	Block Lock Bits	PPEN Bit
0	X	Programmable	Protected	Programmable	Programmable
X	0	Programmable	Protected	Programmable	Programmable
1	1	Programmable	Protected	Protected	Protected

7012 FRM T03

ABSOLUTE MAXIMUM RATINGS*

Temperature under Bias	
X24F128	–65°C to +135°C
Storage Temperature	–65°C to +150°C
Voltage on any Pin with	
Respect to V_{SS}	–1V to +7V
D.C. Output Current	5mA
Lead Temperature	
(Soldering, 10 seconds)	300°C

*COMMENT

Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and the functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

RECOMMENDED OPERATING CONDITIONS

Temperature	Min.	Max.
Commercial	0°C	+70°C
Extended	–20°C	+85°C

7012 FRM T04

Supply Voltage	Limits
X24F128	1.8V to 3.6V
X24F128–5	4.5V to 5.5V

7012 FRM T05

D.C. OPERATING CHARACTERISTICS

		Limits			
Symbol	Parameter	Min.	Max.	Units	Test Conditions
I_{CC1}	V_{CC} Supply Current (Read)		1	mA	SCL = V_{CC} X 0.1/V_{CC} X 0.9 Levels @ 100KHz, SDA = Open, All Other Inputs = V_{SS} or V_{CC} – 0.3V
I_{CC2}	V_{CC} Supply Current (Program)		3	mA	
I_{SB1}(1)	V_{CC} Standby Current		10	µA	SCL = SDA = V_{CC} – 0.3V, All Other Inputs = V_{SS} or V_{CC} – 0.3V, V_{CC} = 5V ± 10%
I_{SB2}(1)	V_{CC} Standby Current		1	µA	SCL = SDA = V_{CC} – 0.1V, All Other Inputs = V_{SS} or V_{CC} – 0.1V, V_{CC} = 1.8V
I_{LI}	Input Leakage Current		10	µA	V_{IN} = V_{SS} to V_{CC}
I_{LO}	Output Leakage Current		10	µA	V_{OUT} = V_{SS} to V_{CC}
V_{IL}(2)	Input LOW Voltage	–0.5	V_{CC} x 0.3	V	
V_{IH}(2)	Input HIGH Voltage	V_{CC} x 0.7	V_{CC} + 0.5	V	
V_{OL}	Output LOW Voltage		0.4	V	I_{OL} = 3mA
V_{hys}(3)	Hysteresis of Schmitt Trigger Inputs	V_{CC} x 0.05		V	

7012 FRM T06.1

CAPACITANCE T_A = +25°C, f = 1MHz, V_{CC} = 5V

Symbol	Parameter	Max.	Units	Test Conditions
$C_{I/O}$(3)	Input/Output Capacitance (SDA)	8	pF	$V_{I/O}$ = 0V
C_{IN}(3)	Input Capacitance (S_0, S_1, S_2, SCL, PP)	6	pF	V_{IN} = 0V

7012 FRM T07

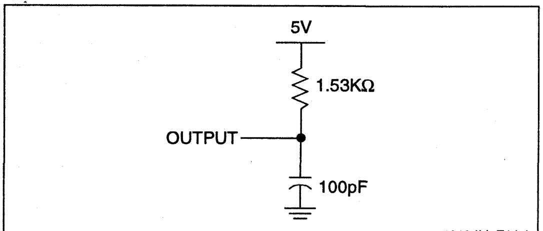
Notes: (1) Must perform a stop command prior to measurement.
(2) V_{IL} min. and V_{IH} max. are for reference only and are not 100% tested.
(3) This parameter is periodically sampled and not 100% tested.

A.C. CONDITIONS OF TEST

Input Pulse Levels	V_{CC} x 0.1 to V_{CC} x 0.9
Input Rise and Fall Times	10ns
Input and Output Timing Levels	V_{CC} X 0.5

7012 FRM T08

EQUIVALENT A.C. LOAD CIRCUIT



7012 ILL F14.1

A.C. OPERATING CHARACTERISTICS (Over the recommended operating conditions, unless otherwise specified.)

Read & Program Cycle Limits

Symbol	Parameter	Min.	Max.	Units
f_{SCL}	SCL Clock Frequency	0	100	KHz
T_I	Noise Suppression Time Constant at SCL, SDA Inputs	50	100	ns
t_{AA}	SCL LOW to SDA Data Out Valid	0.3	3.5	µs
t_{BUF}	Time the Bus Must Be Free Before a New Transmission Can Start	4.7		µs
$t_{HD:STA}$	Start Condition Hold Time	4		µs
t_{LOW}	Clock LOW Period	4.7		µs
t_{HIGH}	Clock HIGH Period	4		µs
$t_{SU:STA}$	Start Condition Setup Time (for a Repeated Start Condition)	4.7		µs
$t_{HD:DAT}$	Data In Hold Time	0		µs
$t_{SU:DAT}$	Data In Setup Time	250		ns
t_R	SDA and SCL Rise Time		1	µs
t_F	SDA and SCL Fall Time		300	ns
$t_{SU:STO}$	Stop Condition Setup Time	4.7		µs
t_{DH}	Data Out Hold Time	300		ns

7012 FRM T09

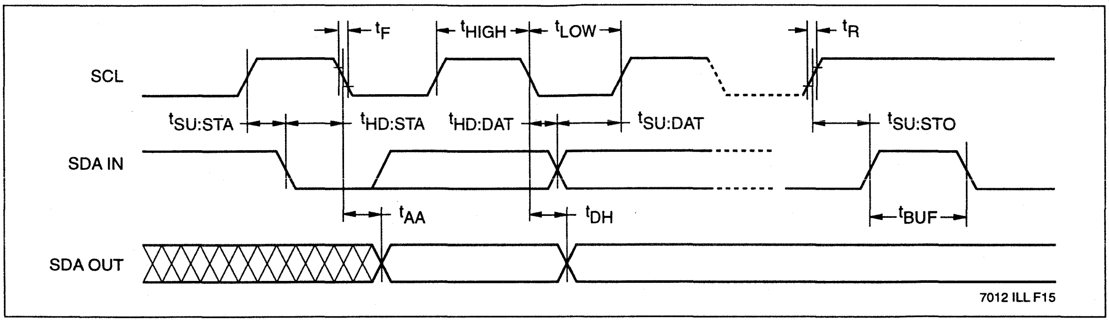
POWER-UP TIMING(4)

Symbol	Parameter	Max.	Units
t_{PUR}	Power-up to Read Operation	1	ms
t_{PUW}	Power-up to Write Operation	5	ms

7012 FRM T10

Notes: (4) t_{PUR} and t_{PUW} are the delays required from the time V_{CC} is stable until the specified operation can be initiated. These parameters are periodically sampled and not 100% tested.

Bus Timing



Program Cycle Limits

Symbol	Parameter	Min.	Typ.[5]	Max.	Units
T_{WR}[6]	Program Cycle Time		5	10	ms

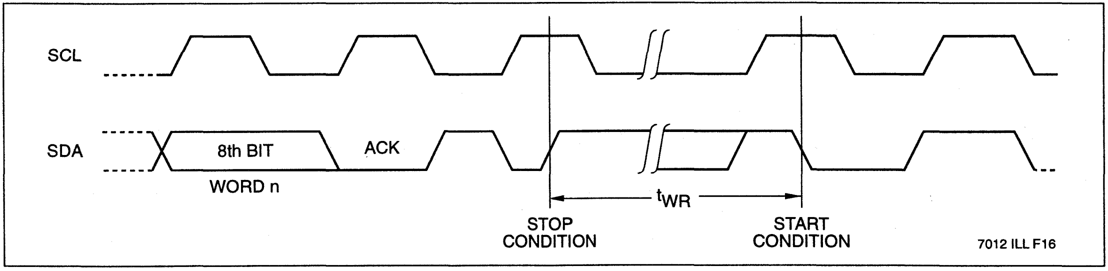
7012 FRM T11

Notes: (5) Typical values are for T_A = 25°C and nominal supply voltage (5V).

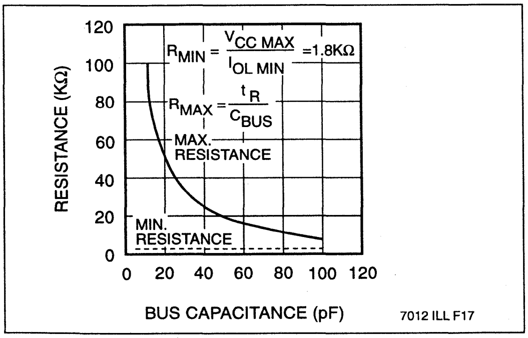
(6) t_{WR} is the minimum cycle time to be allowed from the system perspective unless polling techniques are used. It is the maximum time the device requires to automatically complete the nonvolatile write operation.

The program cycle time is the time from a valid stop condition of a program sequence to the end of the internal erase/program cycle. During the program cycle, the X24F128 bus interface circuits are disabled, SDA is allowed to remain HIGH, and the device does not respond to its slave address.

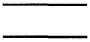
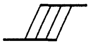
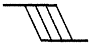
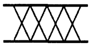
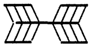
Bus Timing



Guidelines for Calculating Typical Values of Bus Pull-Up Resistors



SYMBOL TABLE

WAVEFORM	INPUTS	OUTPUTS
	Must be steady	Will be steady
	May change from Low to High	Will change from Low to High
	May change from High to Low	Will change from High to Low
	Don't Care: Changes Allowed	Changing: State Not Known
	N/A	Center Line is High Impedance

ORDERING INFORMATION

X24F128 X X -X

Device — X24F128

V_{CC} Range
Blank = 1.8V to 3.6V
5 = 4.5V to 5.5V

Temperature Range
Blank = 0°C to +70°C
E = –20°C to +85°C

Package

X24F128
P = 8-Lead Plastic DIP
S = 16-Lead SOIC

Part Mark Convention

X24F128

X — P = 8-Lead Plastic DIP
S = 16-Lead SOIC

X — Blank = 1.8V to 3.6V, 0°C to +70°C
E = 1.8V to 3.6V, –20°C to +85°C
5 = 4.5V to 5.5V, 0°C to +70°C
E5 = 4.5V to 5.5V, –20°C to +85°C

LIMITED WARRANTY

Devices sold by Xicor, Inc. are covered by the warranty and patent indemnification provisions appearing in its Terms of Sale only. Xicor, Inc. makes no warranty, express, statutory, implied, or by description regarding the information set forth herein or regarding the freedom of the described devices from patent infringement. Xicor, Inc. makes no warranty of merchantability or fitness for any purpose. Xicor, Inc. reserves the right to discontinue production and change specifications and prices at any time and without notice.

Xicor, Inc. assumes no responsibility for the use of any circuitry other than circuitry embodied in a Xicor, Inc. product. No other circuits, patents, licenses are implied.

U.S. PATENTS

Xicor products are covered by one or more of the following U.S. Patents: 4,263,664; 4,274,012; 4,300,212; 4,314,265; 4,326,134; 4,393,481; 4,404,475; 4,450,402; 4,486,769; 4,488,060; 4,520,461; 4,533,846; 4,599,706; 4,617,652; 4,668,932; 4,752,912; 4,829, 482; 4,874, 967; 4,883, 976. Foreign patents and additional patents pending.

LIFE RELATED POLICY

In situations where semiconductor component failure may endanger life, system designers using this product should design the system with appropriate error detection and correction, redundancy and back-up features to prevent such an occurence.

Xicor's products are not authorized for use in critical components in life support devices or systems.

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform, when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.
2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

APPLICATION NOTE
AVAILABLE
AN76 • AN78 • AN81 • AN87

64K/32K/16K X24F064/032/016 8K/4K/2K x 8 Bit

SerialFlash™ Memory with Block Lock™ Protection

FEATURES

- **1.8V to 3.6V or 5V "Univolt" Read and Program Power Supply Versions**
- **Low Power CMOS**
  - **—Active Read Current Less Than 1mA**
  - **—Active Program Current Less Than 3mA**
  - **—Standby Current Less Than 1µA**
- **Internally Organized 8K/4K/2K x 8**
- **New Programmable Block Lock Protection**
  - **—Software Write Protection**
  - **—Programmable hardware Write Protect**
- **Block Lock (0, 1/4, 1/2, or all of the Flash Memory array)**
- **2 Wire Serial Interface**
- **Bidirectional Data Transfer Protocol**
- **32 Byte Sector Programming**
- **Self Timed Program Cycle**
  - **—Typical Programming Time of 5ms Per Sector**
- **High Reliability**
  - **—Endurance: 100,000 cycles per byte**
  - **—Data Retention: 100 Years**
- **Available Packages**
  - **—8-Lead PDIP**
  - **—8-Lead SOIC (JEDEC)**
  - **—14-Lead TSSOP (X24F032/016)**
  - **—20-Lead TSSOP (X24F064)**

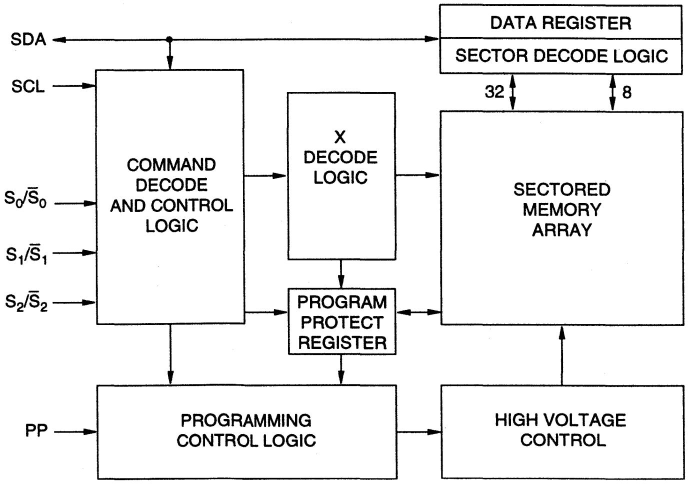
DESCRIPTION

The X24F064/032/016 is a CMOS SerialFlash Memory Family, internally organized 8K/4K/2K x 8. The family features a serial interface and software protocol allowing operation on a simple two wire bus.

Device select inputs (S_0, S_1, S_2) allow up to eight devices to share a common two wire bus.

A Program Protect Register accessed at the highest address location, provides three new programming protection features: Software Programming Protection, Block Lock Protection, and Hardware Programming Protection. The Software Programming Protection feature prevents any nonvolatile writes to the device until the WEL bit in the program protect register is set. The Block Lock™ Protection feature allows the user to individually protect four blocks of the array by programming two bits in the programming protect register. The Programmable Hardware Program Protect feature allows the user to install each device with PP tied to V_{CC}, program the entire memory array in place, and then enable the hardware programming protection by programming a PPEN bit in the program protect register. After this, selected blocks of the array, including the program protect register itself, are permanently protected from being programmed.

FUNCTIONAL DIAGRAM



6686 ILL F01.5

SerialFlash™ Memory and Block Lock™ Protection are trademarks of Xicor, Inc.

Characteristics subject to change without notice

Xicor SerialFlash Memories are designed and tested for applications requiring extended endurance. Inherent data retention is greater than 100 years.

PIN DESCRIPTIONS

Serial Clock (SCL)

The SCL input is used to clock all data into and out of the device.

Serial Data (SDA)

SDA is a bidirectional pin used to transfer data into and out of the device. It is an open drain output and may be wire-ORed with any number of open drain or open collector outputs.

An open drain output requires the use of a pull-up resistor. For selecting typical values, refer to the pull-up resistor selection graph at the end of this data sheet.

Device Select (S_0, $\overline{S}_0$, S_1, $\overline{S}_1$, S_2, $\overline{S}_2$)

The device select inputs are used to set the device select bits of the 8-bit slave address. This allows multiple devices to share a common bus. These inputs can be static or actively driven. If used statically they must be tied to V_{SS} or V_{CC} as appropriate. If actively driven, they must be driven with CMOS levels (driven to V_{CC} or V_{SS}).

Program Protect (PP)

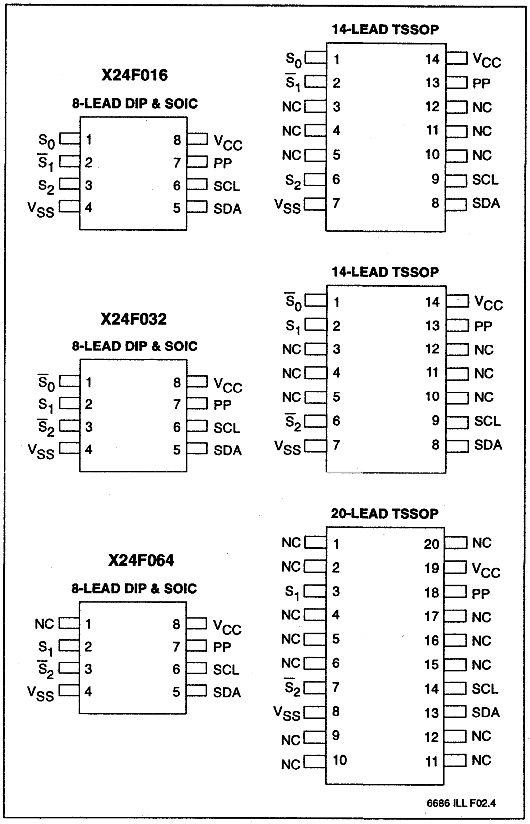
The program protect input controls the hardware program protect feature. When held LOW, hardware program protection is disabled and the X24F064/032/016 can be programmed normally. When this input is held HIGH, and the PPEN bit in the program protect register is set HIGH, program protection is enabled, and nonvolatile writes are disabled to the selected blocks as well as the program protect register itself.

PIN NAMES

Symbol	Description
S_0, $\overline{S}_0$, S_1, $\overline{S}_1$, S_2, $\overline{S}_2$	Device Select Inputs
SDA	Serial Data
SCL	Serial Clock
PP	Program Protect
V_{SS}	Ground
V_{CC}	Supply Voltage
NC	No Connect

6686 FRM T01.1

PIN CONFIGURATION



DEVICE OPERATION

The X24F064/032/016 supports a bidirectional bus oriented protocol. The protocol defines any device that sends data onto the bus as a transmitter, and the receiving device as the receiver. The device controlling the transfer is a master and the device being controlled is the slave. The master will always initiate data transfers, and provide the clock for both transmit and receive operations. Therefore, the X24F064/032/016 will be considered a slave in all applications.

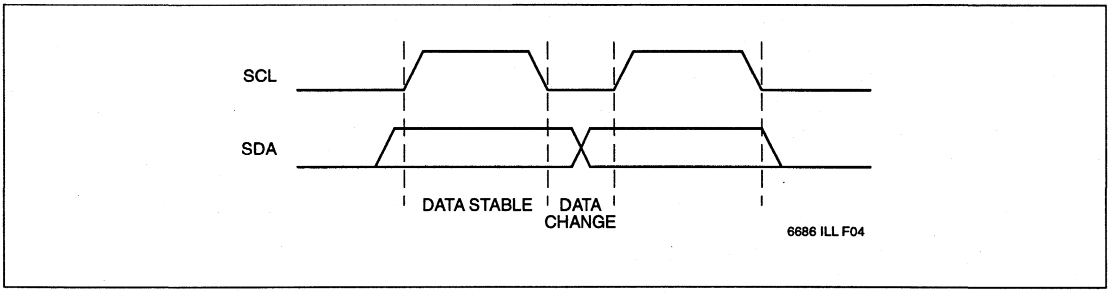
Clock and Data Conventions

Data states on the SDA line can change only during SCL LOW. SDA state changes during SCL HIGH are reserved for indicating start and stop conditions. Refer to Figures 1 and 2.

Start Condition

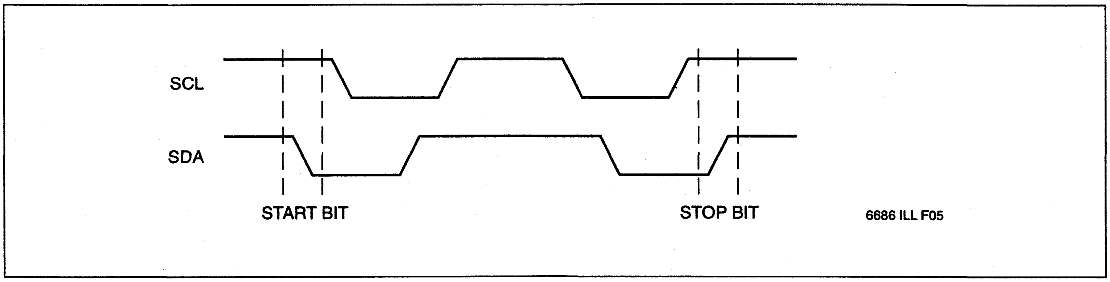
All commands are preceded by the start condition, which is a HIGH to LOW transition of SDA when SCL is HIGH. The X24F064/032/016 continuously monitors the SDA and SCL lines for the start condition and will not respond to any command until this condition has been met.

Figure 1. Data Validity



Notes: (5) Typical values are for T_A = 25°C and nominal supply voltage (2.7V)
(6) t_{PR} is the minimum cycle time from the system perspective when polling techniques are not used. It is the maximum time the device requires to perform the internal program operation.

Figure 2. Definition of Start and Stop



Stop Condition

All communications must be terminated by a stop condition, which is a LOW to HIGH transition of SDA when SCL is HIGH. The stop condition is also used to place the device into the standby power mode after a read sequence. A stop condition can only be issued after the transmitting device has released the bus.

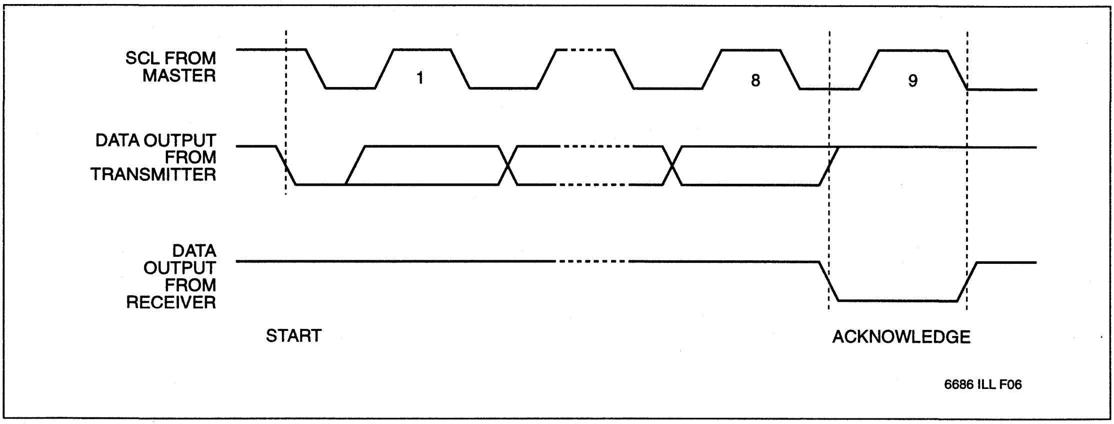
Acknowledge

Acknowledge is a software convention used to indicate successful data transfer. The transmitting device, either master or slave, will release the bus after transmitting eight bits. During the ninth clock cycle the receiver will pull the SDA line LOW to acknowledge that it received the eight bits of data. Refer to Figure 3.

The X24F064/032/016 will respond with an acknowledge after recognition of a start condition and its slave address. If both the device and a write operation have been selected, the X24F064/032/016 will respond with an acknowledge after the receipt of each subsequent eight-bit word.

In the read mode the X24F064/032/016 will transmit eight bits of data, release the SDA line and monitor the line for an acknowledge. If an acknowledge is detected and no stop condition is generated by the master, the X24F064/032/016 will continue to transmit data. If an acknowledge is not detected, the device will terminate further data transmissions. The master must then issue a stop condition to return the X24F064/032/016 to the standby power mode and place the device into a known state.

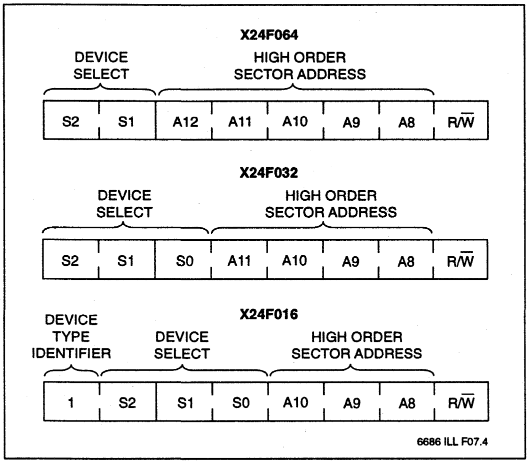
Figure 3. Acknowledge Response From Receiver



DEVICE ADDRESSING

Following a start condition the master must output the address of the slave it is accessing (see Figure 4). The next two bits are the device select bits. A system could have up to eight X24F032/016's on the bus or up to four 24F064's on the bus. The device addresses are defined by the state of the S_0, S_1, and S_2 inputs. Note some of the slave addresses must be the inverse of the corresponding input pin.

Figure 4. Slave Address



Also included in the slave address is an extension of the array's address which is concatenated with the eight bits of address in the sector address field, providing direct access to the entire SerialFlash Memory array.

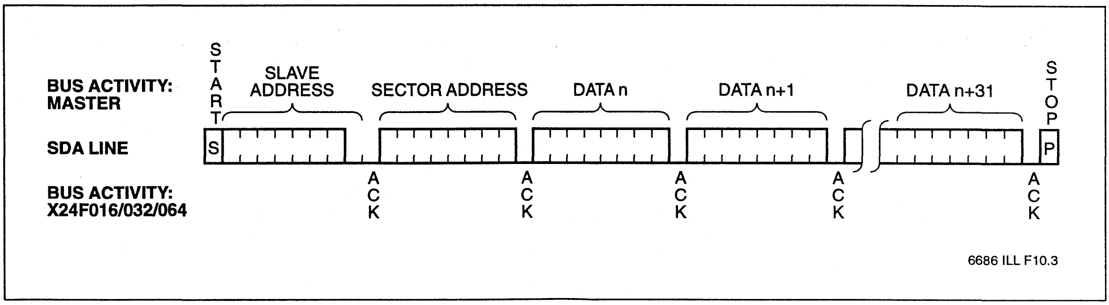
The last bit of the slave address defines the operation to be performed. When set HIGH a read operation is selected, when set LOW a program operation is selected.

Following the start condition, the X24F064/032/016 monitors the SDA bus comparing the slave address being transmitted with its slave address device type identifier. Upon a correct comparison of the device select inputs, the X24F064/032/016 outputs an acknowledge on the SDA line. Depending on the state of the R/W̄ bit, the X24F064/032/016 will execute a read or program operation.

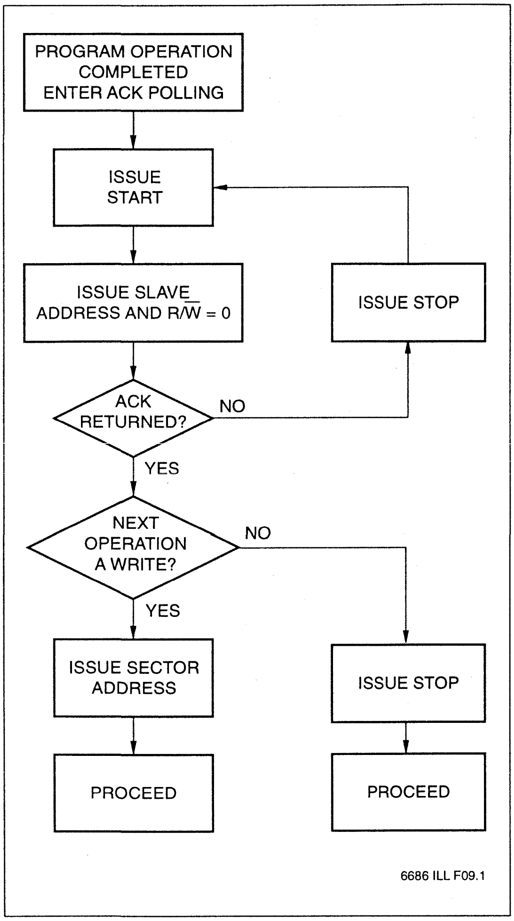
PROGRAMMING OPERATIONS

The X24F064/032/016 offers a 32-byte sector programming operation. For a program operation, the X24F064/032/016 requires a second address field. This field contains the address of the first byte in the sector. Upon receipt of the address, comprised of eight bits, the X24F064/032/016 responds with an acknowledge and awaits the next eight bits of data, again responding with an acknowledge. The master then transmits 31 more bytes. After the receipt of each byte, the X24F064/032/016 will respond with an acknowledge.

Figure 5. Sector Programming



Flow 1. ACK Polling Sequence



After the receipt of each byte, the five low order address bits are internally incremented by one. The high order bits of the sector address remain constant. If the master should transmit more or less than 32 bytes prior to generating the stop condition, the contents of the sector cannot be guaranteed. All inputs are disabled until completion of the internal program cycle. Refer to Figure 5 for the address, acknowledge and data transfer sequence.

Acknowledge Polling

The Max Write Cycle Time can be significantly reduced using Acknowledge Polling. To initiate Acknowledge Polling, the master issues a start condition followed by the Slave Address Byte for a write or read operation. If the device is still busy with the high voltage cycle, then no ACK will be returned. If the device has completed the write operation, an ACK will be returned and the host can then proceed with the read or write operation. Refer to Flow 1.

READ OPERATIONS

Read operations are initiated in the same manner as program operations with the exception that the R/$\overline{W}$ bit of the slave address is set HIGH. There are three basic read operations: current address read, random read and sequential read.

It should be noted that the ninth clock cycle of the read operation is not a “don’t care.” To terminate a read operation, the master must either issue a stop condition during the ninth cycle or hold SDA HIGH during the ninth clock cycle and then issue a stop condition.

Current Address Read

Internally, the X24F064/032/016 contains an address counter that maintains the address of the last byte read, incremented by one byte. Therefore, if the last read was from address n, the next read operation accesses data from address n + 1. Upon receipt of the slave address with the R/$\overline{W}$ set HIGH, the X24F064/032/016 issues an acknowledge and transmits the eight-bit word. The read operation is terminated by the master; by not responding with an acknowledge and by issuing a stop condition. Refer to Figure 6 for the sequence of address, acknowledge and data transfer.

Random Read

Random read operations allow the master to access any memory location in a random manner. Prior to issuing the slave address with the R/$\overline{W}$ bit set HIGH, the master must first perform a "dummy" write operation. The master issues the start condition, and the slave address with the R/$\overline{W}$ bit set LOW, followed by the byte address it is to read. After the byte address acknowledge, the master immediately reissues the start condition and the slave address with the R/$\overline{W}$ bit set HIGH. This will be followed by an acknowledge from the X24F064/032/016 and then by the eight-bit byte. The read operation is terminated by the master; by not responding with an acknowledge and by issuing a stop condition. Refer to Figure 7 for the address, acknowledge and data transfer sequence.

Figure 6. Current Address Read

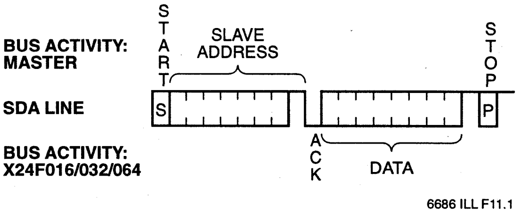
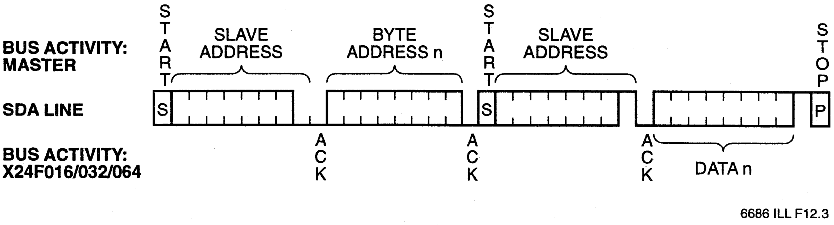


Figure 7. Random Read



Sequential Read

Sequential reads can be initiated as either a current address read or random access read. The first byte is transmitted as with the other modes, however, the master now responds with an acknowledge, indicating it requires additional data. The X24F064/032/016 continues to output data for each acknowledge received. The read operation is terminated by the master; by not responding with an acknowledge and then issuing a stop condition.

The data output is sequential, with the data from address n followed by the data from n + 1. The address counter for read operations increments all address bits, allowing the entire memory contents to be serially read during one operation. At the end of the address space, the counter "rolls over" to 0 and the X24F064/032/016 continues to output data for each acknowledge received. Refer to Figure 8 for the address, acknowledge and data transfer sequence.

Figure 8. Sequential Read

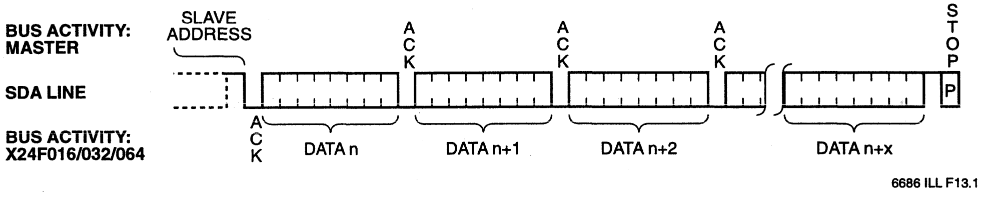
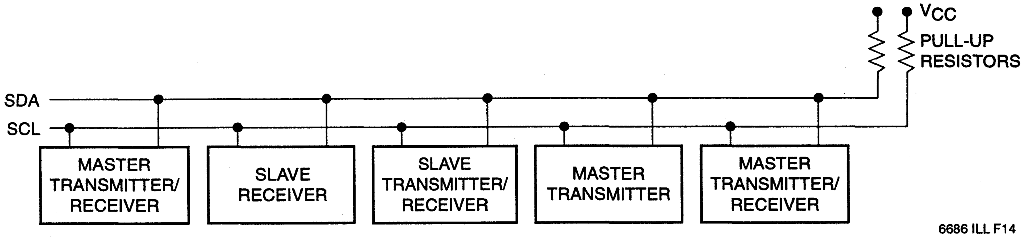


Figure 9. Typical System Configuration


VCC
PULL-UP RESISTORS
SDA
SCL
MASTER TRANSMITTER/ RECEIVER
SLAVE RECEIVER
SLAVE TRANSMITTER/ RECEIVER
MASTER TRANSMITTER
MASTER TRANSMITTER/ RECEIVER
6686 ILL F14

PROGRAM PROTECT REGISTER

The Program Protect Register (PPR) is accessed at the highest address of each device:

X24F064 = 1FFF
X24F032 = 0FFF
X24F016 = 07FF

Figure 10. Program Protect Register

7	6	5	4	3	2	1	0
PPEN	0	0	BL1	BL0	RWEL	WEL	0

6686 ILL F15

PPR.1 = WEL

- Write Enable Latch (Volatile)
  0 = Write enable latch reset, programming disabled
  1 = Write enable latch set, programming enabled

If WEL = 0 then "no ACK" after first byte of input data.

PPR.2 = RWEL
- Register Write Enable Latch (Volatile)
  0 = Register write enable latch reset, programming disabled
  1 = Register write enable latch set, programming enabled

PPR.3, PPR.4 = BL0, BL1
- Block Lock Bits (Nonvolatile)
(See Block Lock Bits section for definition)

PPR.7 = PPEN
- Programming Protect Enable Bit (Nonvolatile)
(See Programmable Hardware Program Protect section for definition)

Writing to the Program Protect Register

The Program Protect Register is written by performing a write of one byte directly to the highest address location. During normal Sector Programming, the byte in the array at the highest address will be written instead of the Program Protect Register (assuming programming is not disabled by the Block Lock register).

The state of the Program Protect Register can be read by performing a random read at the highest address location at any time. If a sequential read starting at any other address than the highest address location is performed, the contents of the byte in the array at the highest address location is read out instead of the Program Protect Register.

WEL and RWEL are volatile latches that power-up in the LOW (disabled) state. A write to any address other than the highest address location, where the Program Protect Register is located, will be ignored (no ACK) until the WEL bit is set HIGH. The WEL bit is set by writing 0000001x to the highest address location. Once set, WEL remains HIGH until either reset (by writing 00000000 to the highest address location) or until the part powers-up again. The RWEL bit controls writes to the Block Lock bits. RWEL is set by first setting WEL = 1 and then writing 0000011x to the highest address location. RWEL must be set in order to change the Block Lock bits (BL0 and BL1) or the PPEN bit. RWEL is reset when the Block Lock or PPEN bits are changed, or when the part powers-up again.

Programming the BL or PPEN Bits

A three step sequence is required to change the nonvolatile Block Lock or Program Protect Enable:

1) Set WEL = 1 (write 00000010 to the highest address location, volatile write cycle)

(Start)

2) Set RWEL = 1 (write 00000110 to the highest address location, volatile write cycle)

(Start)

3) Set BL1, BL0, and/or PPEN bits (Write w00yz010 to the highest address location)

w = PPEN, y = BL1, Z = BL0,

(Stop)

Step 3 is a nonvolatile program cycle, requiring 10ms to complete. RWEL is reset (0) by this program cycle, requiring another program cycle to set RWEL again before the Block Lock bits can be changed. RWEL must be 0 in step 3; if w00yz110 is written to the highest address location, RWEL is set but PPEN, BL1 and BL0 are not changed (the device remains at step 2).

Block Lock Bits

The Block Lock Bits BL0 and BL1 determine which blocks of the memory are write-protected:

Table 1. Block Lock Bits

BL1	BL0	Array Locked
0	0	None
0	1	Upper 1/4
1	0	Upper 1/2
1	1	Full Array (WPR not included)

6686 FRM T02

Programmable Hardware Program Protect

The Program Protect (PP) pin and the Program Protect Enable (PPEN) bit in the Program Protect Register control the programmable hardware program protect feature. Hardware program protection is enabled when the PP pin and the PPEN bit are both HIGH, and disabled when either the PP pin is LOW or the PPEN bit is LOW. When the chip is hardware program-protected, nonvolatile programming is disabled, including the Program Protect Register, the BL bits and the PPEN bit itself, as well as to Block Locked sections in the memory array. Only the sections of the memory array that are not Block Locked can be written. Note that since the PPEN bit is program-protected, it cannot be changed back to a LOW state, and program protection is disabled as long as the PP pin is held HIGH. Table 2 defines the program protection status for each state of PPEN and PP.

Table 2. Program Protect Status Table

PP	PPEN	Memory Array (Not Block Locked)	Memory Array (Block Locked)	BL Bits	PPEN Bit
0	X	Programmable	Locked	Programmable	Programmable
X	0	Programmable	Locked	Programmable	Programmable
1	1	Programmable	Locked	Locked	Locked

6686 FRM T03

ABSOLUTE MAXIMUM RATINGS*

Temperature Under Bias
X24F064/032/016 .... –65°C to +135°C
Storage Temperature........................ –65°C to +150°C
Voltage on any Pin with
Respect to V_{SS}.................................. –1V to +7V
D.C. Output Current.. 5mA
Lead Temperature (Soldering, 10 Seconds)...... 300°C

*COMMENT

Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and the functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

RECOMMENDED OPERATING CONDITIONS

Temperature	Min.	Max.
Commercial	0°C	+70°C
Extended	–20°C	+85°C
Industrial	–40°C	+85°C

6686 FRM T04.2

Supply Voltage	Limits
X24F064/032/016	1.8V to 3.6V
X24F064/032/016–5	4.5V to 5.5V

6686 FRM T05.2

D.C. OPERATING CHARACTERISTICS (Over the recommended operating conditions unless otherwise specified.)

Symbol	Parameter	Limits		Units	Test Conditions
		Min.	Max.		
I_{CC1}	V_{CC} Supply Current (Read)		1	mA	SCL = V_{CC} X 0.1/V_{CC} X 0.9 Levels @ 100KHz, SDA = Open, All Other Inputs = V_{SS} or V_{CC} – 0.3V
I_{CC2}	V_{CC} Supply Current (Write)		3	mA	
I_{SB1}(1)	V_{CC} Standby Current		1	µA	SCL = SDA = V_{CC}, All Other Inputs = V_{SS} or V_{CC} – 0.3V, V_{CC} = 3.6V
I_{SB2}(1)	V_{CC} Standby Current		10	µA	SCL = SDA = V_{CC}, All Other Inputs = V_{SS} or V_{CC} – 0.3V, V_{CC} = 5V ±10%
I_{LI}	Input Leakage Current		10	µA	V_{IN} = V_{SS} to V_{CC}
I_{LO}	Output Leakage Current		10	µA	V_{OUT} = V_{SS} to V_{CC}
V_{IL}(2)	Input LOW Voltage	–1	V_{CC} x 0.3	V	
V_{IH}(2)	Input HIGH Voltage	V_{CC} x 0.7	V_{CC} + 0.5	V	
V_{OL}	Output LOW Voltage		0.4	V	I_{OL} = 3mA

6686 FRM T06.4

CAPACITANCE T_A = +25°C, f = 1MHz, V_{CC} = 2.7V

Symbol	Parameter	Max.	Units	Test Conditions
$C_{I/O}$(3)	Input/Output Capacitance (SDA)	8	pF	$V_{I/O}$ = 0V
C_{IN}(3)	Input Capacitance (S_1, $\overline{S}_2$, SCL)	6	pF	V_{IN} = 0V

6686 FRM T07

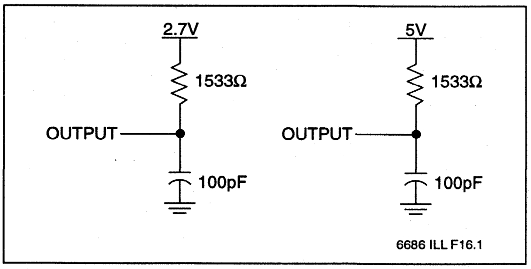
Notes: (1) Must perform a stop command prior to measurement.
(2) V_{IL} min. and V_{IH} max. are for reference only and are not 100% tested.
(3) This parameter is periodically sampled and not 100% tested.

A.C. CONDITIONS OF TEST

Input Pulse Levels	V_{CC} x 0.1 to V_{CC} x 0.9
Input Rise and Fall Times	10ns
Input and Output Timing Levels	V_{CC} X 0.5

6686 FRM T08

EQUIVALENT A.C. LOAD CIRCUIT



A.C. OPERATING CHARACTERISTICS (Over the recommended operating conditions, unless otherwise specified.)

Read & Write Cycle Limits

Symbol	Parameter	Min.	Max.	Units
f_{SCL}	SCL Clock Frequency	0	100	KHz
T_I	Noise Suppression Time Constant at SCL, SDA Inputs		100	ns
t_{AA}	SCL LOW to SDA Data Out Valid	0.3	3.5	µs
t_{BUF}	Time the Bus Must Be Free Before a New Transmission Can Start	4.7		µs
$t_{HD:STA}$	Start Condition Hold Time	4		µs
t_{LOW}	Clock LOW Period	4.7		µs
t_{HIGH}	Clock HIGH Period	4		µs
$t_{SU:STA}$	Start Condition Setup Time (for a Repeated Start Condition)	4.7		µs
$t_{HD:DAT}$	Data In Hold Time	0		µs
$t_{SU:DAT}$	Data In Setup Time	250		ns
t_R	SDA and SCL Rise Time		1	µs
t_F	SDA and SCL Fall Time		300	ns
$t_{SU:STO}$	Stop Condition Setup Time	4.7		µs
t_{DH}	Data Out Hold Time	300		ns

6686 FRM T09.1

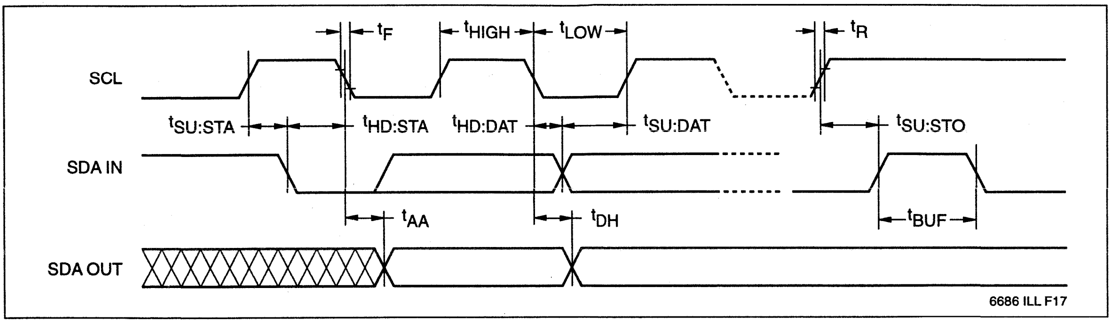
POWER-UP TIMING[4]

Symbol	Parameter	Max.	Units
t_{PUR}	Power-up to Read Operation	1	ms
t_{PUW}	Power-up to Write Operation	5	ms

6686 FRM T10

Notes: (4) t_{PUR} and t_{PUW} are the delays required from the time V_{CC} is stable until the specified operation can be initiated. These parameters are periodically sampled and not 100% tested.

Bus Timing



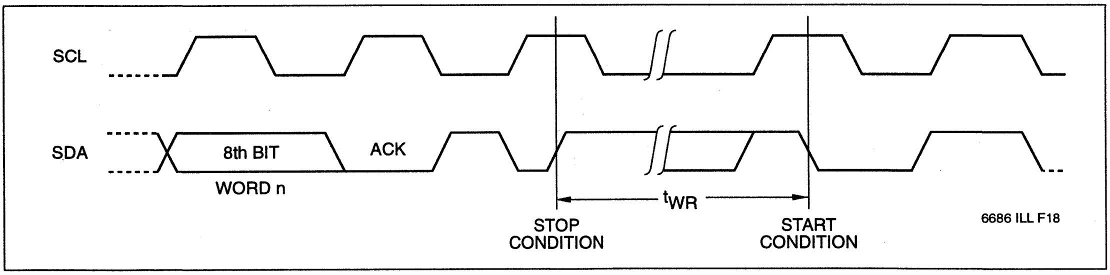
Program Cycle Limits

Symbol	Parameter	Min.	Typ.[5]	Max.	Units
t_{PR}[6]	Program Cycle Time		5	10	ms

6686 FRM T11.1

The program cycle time is the time from a valid stop condition of a write sequence to the end of the internal erase/program cycle. During the program cycle, the X24F064/032/016 bus interface circuits are disabled, SDA is allowed to remain HIGH, and the device does not respond to its slave address.

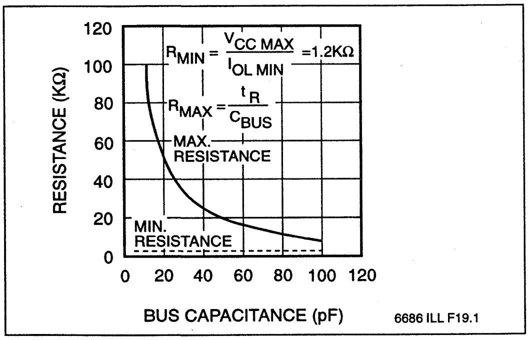
Bus Timing



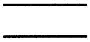
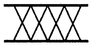
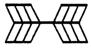
Notes: (5) Typical values are for T_A = 25°C and nominal supply voltage (2.7V).

(6) t_{WR} is the minimum cycle time to be allowed from the system perspective unless polling techniques are used. It is the maximum time the device requires to automatically complete the internal program operation.

Guidelines for Calculating Typical Values of Bus Pull-Up Resistors



SYMBOL TABLE

WAVEFORM	INPUTS	OUTPUTS
	Must be steady	Will be steady
	May change from LOW to HIGH	Will change from LOW to HIGH
	May change from HIGH to LOW	Will change from HIGH to LOW
	Don't Care: Changes Allowed	Changing: State Not Known
	N/A	Center Line is High Impedance

ORDERING INFORMATION

X24FXXX X X -X

V$_{CC}$ Range
Blank = 1.8V to 3.6V
5 = 4.5V to 5.5V

Device
X24F064
X24F032
X24F016

Temperature Range
Blank = Commercial = 0°C to +70°C
E = Extended = –20°C to +85°C
I = Industrial = –40°C to +85°C

Package

X24F064	**X24F032** **X24F016**
P = 8-Lead Plastic DIP	P = 8-Lead Plastic DIP
S = 8-Lead SOIC (JEDEC)	S = 8-Lead SOIC (JEDEC)
V = 20-Lead TSSOP	V = 14-Lead TSSOP

Part Mark Convention

X24FXXX X

X
P = 8-Lead Plastic DIP
Blank = 8-Lead SOIC (JEDEC)
V = 14/20-Lead TSSOP

X24F064
X24F032
X24F016

X
Blank = 1.8V to 3.6V, 0°C to +70°C
E = 1.8V to 3.6V, –20°C to +85°C
5 = 4.5V to 5.5V, 0°C to +70°C
I5 = 4.5V to 5.5V, –40°C to +85°C

LIMITED WARRANTY

Devices sold by Xicor, Inc. are covered by the warranty and patent indemnification provisions appearing in its Terms of Sale only. Xicor, Inc. makes no warranty, express, statutory, implied, or by description regarding the information set forth herein or regarding the freedom of the described devices from patent infringement. Xicor, Inc. makes no warranty of merchantability or fitness for any purpose. Xicor, Inc. reserves the right to discontinue production and change specifications and prices at any time and without notice.

Xicor, Inc. assumes no responsibility for the use of any circuitry other than circuitry embodied in a Xicor, Inc. product. No other circuits, patents, licenses are implied.

U.S. PATENTS

Xicor products are covered by one or more of the following U.S. Patents: 4,263,664; 4,274,012; 4,300,212; 4,314,265; 4,326,134; 4,393,481; 4,404,475; 4,450,402; 4,486,769; 4,488,060; 4,520,461; 4,533,846; 4,599,706; 4,617,652; 4,668,932; 4,752,912; 4,829, 482; 4,874, 967; 4,883, 976. Foreign patents and additional patents pending.

LIFE RELATED POLICY

In situations where semiconductor component failure may endanger life, system designers using this product should design the system with appropriate error detection and correction, redundancy and back-up features to prevent such an occurrence.

Xicor's products are not authorized for use in critical components in life support devices or systems.

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform, when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.
2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

Xicor®

Application Note
AVAILABLE
AN61 • AN75 • AN77 • AN79 • AN82

Xicor®

X25F128 — 16K x 8 Bits

SerialFlash™ Memory With Block Lock™ Protection

FEATURES

- **1MHz Clock Rate**
- **SPI Serial Interface**
- **16K X 8 Bits**
  - **—32 Byte Small Sector Program Mode**
- **Low Power CMOS**
  - **—<1µA Standby Current**
  - **—<5mA Active Current**
- **1.8V – 3.6V or 5V "Univolt" Read and Program Power Supply Versions**
- **Block Lock Protection**
  - **—Protect 1/4, 1/2 or all of E²PROM Array**
- **Built-in Inadvertent Program Protection**
  - **—Power-Up/Power-Down protection circuitry**
  - **—Program Enable Latch**
  - **—Program Protect Pin**
- **Self-Timed Program Cycle**
  - **—5ms Program Cycle Time (Typical)**
- **High Reliability**
  - **—Endurance: 100,000 cycles per byte**
  - **—Data Retention: 100 Years**
  - **—ESD protection: 2000V on all pins**
- **8-Lead PDIP Package**
- **16-Lead 150 mil SOIC Package**

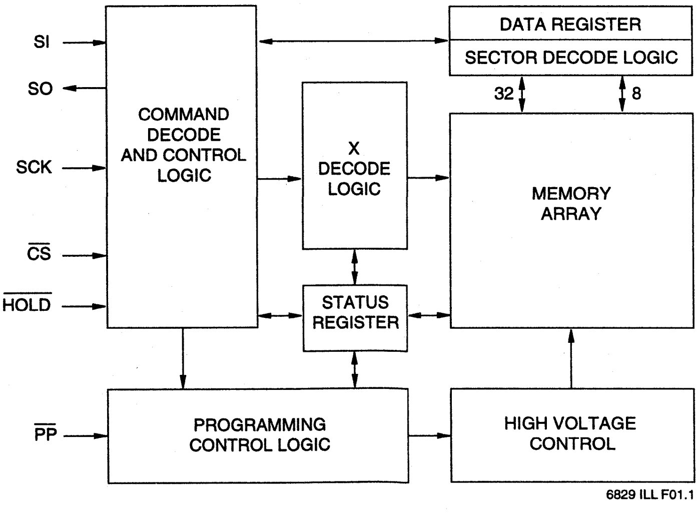
DESCRIPTION

The X25F128 is a 131,072-bit CMOS SerialFlash memory, internally organized 16K X 8. It features a "Univolt" Program and Read voltage, Serial Peripheral Interface (SPI), and software protocol allowing operation on a simple three-wire bus. The bus signals are a clock input (SCK), plus separate data in (SI) and data out (SO) lines. Access to the device is controlled through a chip select ($\overline{CS}$) input, allowing any number of devices to share the same bus.

The X25F128 also features two additional inputs that provide the end user with added flexibility. By asserting the $\overline{HOLD}$ input, the X25F128 will ignore transitions on its inputs, thus allowing the host to service higher priority interrupts. The $\overline{PP}$ input can be used as a hardwire input to the X25F128 disabling all program attempts to the status register, thus providing a mechanism for limiting end user capability of altering 0, 1/4, 1/2, or all of the memory.

The X25F128 utilizes Xicor's proprietary flash cell, providing a minimum endurance of 100,000 cycles and a minimum data retention of 100 years.

FUNCTIONAL DIAGRAM



SerialFlash™ and Block Lock™ Protection are trademarks of Xicor, Inc.

PIN DESCRIPTIONS

Serial Output (SO)

SO is a push-pull serial data output pin. During a read cycle, data is shifted out on this pin. Data is clocked out by the falling edge of the serial clock.

Serial Input (SI)

SI is the serial data input pin. All opcodes, byte addresses, and data to be written to the memory are input on this pin. Data is latched by the rising edge of the serial clock.

Serial Clock (SCK)

The Serial Clock controls the serial bus timing for data input and output. Opcodes, addresses, or data present on the SI pin are latched on the rising edge of the clock input, while data on the SO pin change after the falling edge of the clock input.

Chip Select ($\overline{\text{CS}}$)

When $\overline{\text{CS}}$ is HIGH, the X25F128 is deselected and the SO output pin is at high impedance and unless an internal program operation is underway the X25F128 will be in the standby power mode. $\overline{\text{CS}}$ LOW enables the X25F128, placing it in the active power mode. It should be noted that after power-up, a HIGH to LOW transition on $\overline{\text{CS}}$ is required prior to the start of any operation.

Program Protect ($\overline{\text{PP}}$)

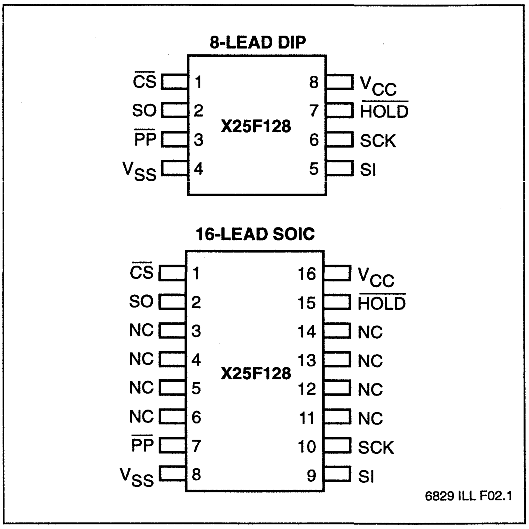
When $\overline{\text{PP}}$ is LOW and the nonvolatile bit PPEN is "1", nonvolatile programming of the X25F128 status register is disabled, but the part otherwise functions normally. When $\overline{\text{PP}}$ is held HIGH, all functions, including nonvolatile programming operate normally. $\overline{\text{PP}}$ going LOW while $\overline{\text{CS}}$ is still LOW will interrupt programming of the X25F128 status register. If the internal program cycle has already been initiated, $\overline{\text{PP}}$ going LOW will have no effect on programming.

The $\overline{\text{PP}}$ pin function is blocked when the PPEN bit in the status register is "0". This allows the user to install the X25F128 into a system with $\overline{\text{PP}}$ pin grounded and still be able to program the status register. The $\overline{\text{PP}}$ pin functions will be enabled when the PPEN bit is set "0".

Hold ($\overline{\text{HOLD}}$)

$\overline{\text{HOLD}}$ is used in conjunction with the $\overline{\text{CS}}$ pin to select the device. Once the part is selected and a serial sequence is underway, $\overline{\text{HOLD}}$ may be used to pause the serial communication with the controller without resetting the serial sequence. To pause, $\overline{\text{HOLD}}$ must be brought LOW while SCK is LOW. To resume communication, $\overline{\text{HOLD}}$ is brought HIGH, again while SCK is LOW. If the pause feature is not used, $\overline{\text{HOLD}}$ should be held HIGH at all times.

PIN CONFIGURATION



6829 ILL F02.1

PIN NAMES

SYMBOL	DESCRIPTION
$\overline{\text{CS}}$	Chip Select Input
SO	Serial Output
SI	Serial Input
SCK	Serial Clock Input
$\overline{\text{PP}}$	Program Protect Input
V_{SS}	Ground
V_{CC}	Supply Voltage
$\overline{\text{HOLD}}$	Hold Input
NC	No Connect

6829 PGM T01

PRINCIPLES OF OPERATION

The X25F128 is a SerialFlash Memory designed to interface directly with the synchronous serial peripheral interface (SPI) of many popular microcontroller families.

The X25F128 contains an 8-bit instruction register. It is accessed via the SI input, with data being clocked in on the rising SCK. $\overline{\text{CS}}$ must be LOW and the $\overline{\text{HOLD}}$ and $\overline{\text{PP}}$ inputs must be HIGH during the entire operation. The $\overline{\text{PP}}$ input is "Don't Care" if PPEN is set "0".

Table 1 contains a list of the instructions and their operation codes. All instructions, addresses and data are transferred MSB first.

Data input is sampled on the first rising edge of SCK after $\overline{\text{CS}}$ goes LOW. SCK is static, allowing the user to stop the clock and then resume operations. If the clock line is shared with other peripheral devices on the SPI bus, the user can assert the $\overline{\text{HOLD}}$ input to place the X25F128 into a "PAUSE" condition. After releasing $\overline{\text{HOLD}}$, the X25F128 will resume operation from the point when $\overline{\text{HOLD}}$ was first asserted.

Program Enable Latch

The X25F128 contains a program enable latch. This latch must be SET before a program operation will be completed internally. The PREN instruction will set the latch and the PRDI instruction will reset the latch. This latch is automatically reset on power-up and after the completion of a sector program or status register write cycle.

Status Register

The RDSR instruction provides access to the status register. The status register may be read at any time, even during a program cycle. The status register is formatted as follows:

7	6	5	4	3	2	1	0
PPEN	X	X	X	BL1	BL0	PEL	PIP

6829 PGM T02

PPEN, BL0, and BL1 are set by the PRSR instruction. PEL and PIP are "read-only" and automatically set by other operations.

The Programming-In-Process (PIP) bit indicates whether the X25F128 is busy with a program operation. When set to a "1" programming is in progress, when set to a "0" no programming is in progress. During programming, all other bits are set to "1".

The Program Enable Latch (PEL) bit indicates the status of the program enable latch. When set to a "1" the latch is set; when set to a "0" the latch is reset.

The Block Lock (BL0 and BL1) bits are nonvolatile and allow the user to select one of four levels of protection. The X25F128 array is divided into four equal segments. One, two, or all four of the segments may be locked. That is, the user may read the segments, but will be unable to alter (program) data within the selected segments. The partitioning is controlled as illustrated below.

Status Register Bits		Array Addresses
BL1	BL0	Locked
0	0	None
0	1	upper fourth
1	0	upper half
1	1	All

6829 PGM T03.1

Program-Protect Enable

The Program-Protect-Enable bit (PPEN) in the X25F128 status register acts as an enable bit for the $\overline{\text{PP}}$ pin.

Table 1. Instruction Set

Instruction Name	Instruction Format*	Operation
PREN	0000 0110	Set the Program Enable Latch (Enable Program Operations)
PRDI	0000 0100	Reset the Program Enable Latch (Disable Program Operations)
RDSR	0000 0101	Read Status Register
PRSR	0000 0001	Program Status Register
READ	0000 0011	Read from Memory Array beginning at Selected Address
PROGRAM	0000 0010	Program Memory Array beginning at Selected Address (32 Bytes)

6829 PGM T04.1

*Instructions are shown MSB in leftmost position. Instructions are transferred MSB first.

PPEN	$\overline{\text{PP}}$	PEL	Locked Blocks	Unlocked Blocks	Status Register
0	X	0	Locked	Locked	Locked
0	X	1	Locked	Programmable	Programmable
1	LOW	0	Locked	Locked	Locked
1	LOW	1	Locked	Programmable	Locked
X	HIGH	0	Locked	Locked	Locked
X	HIGH	1	Locked	Programmable	Programmable

6829 PGM T05

The Program Protect ($\overline{\text{PP}}$) pin and the nonvolatile Program Protect Enable (PPEN) bit in the Status Register control the programmable hardware write protect feature. Hardware program protection is enabled when $\overline{\text{PP}}$ pin is LOW, and the PPEN bit is "1". Hardware program protection is disabled when either the $\overline{\text{PP}}$ pin is HIGH or the PPEN bit is "0". When the chip is hardware program protected, nonvolatile programming of the Status Register in disabled, including the Block Lock bits and the PPEN bit itself, as well as the Block Lock sections in the memory array. Only the sections of the memory array that are not Block Locked can be programmed.

Note: Since the PPEN bit is program protected, it cannot be changed back to a "0", as long as the $\overline{\text{PP}}$ pin is held LOW.

Clock and Data Timing

Data input on the SI line is latched on the rising edge of SCK. Data is output on the SO line by the falling edge of SCK.

Read Sequence

When reading from the SerialFlash memory array, $\overline{\text{CS}}$ is first pulled LOW to select the device. The 8-bit READ instruction is transmitted to the X25F128, followed by the 16-bit address. After the read opcode and address are sent, the data stored in the memory at the selected address is shifted out on the SO line. The data stored in memory at the next address can be read sequentially by continuing to provide clock pulses. The address is automatically incremented to the next higher address after each byte of data is shifted out. When the highest address is reached the address counter rolls over to address $0000, allowing the read cycle to be continued indefinitely. The read operation is terminated by taking $\overline{\text{CS}}$ HIGH. Refer to the Read SerialFlash Memory Array Operation Sequence illustrated in Figure 1.

To read the status register, the $\overline{\text{CS}}$ line is first pulled LOW to select the device followed by the 8-bit instruction. After the read status register opcode is sent, the contents of the status register are shifted out on the SO line. The Read Status Register Sequence is illustrated in Figure 2.

Programming Sequence

Prior to any attempt to program the X25F128, the program enable latch must first be set by issuing the PREN instruction (See Figure 3). $\overline{\text{CS}}$ is first taken LOW, then the PREN instruction is clocked into the X25F128. After all eight bits of the instruction are transmitted, $\overline{\text{CS}}$ must then be taken HIGH. If the user continues the programming operation without taking $\overline{\text{CS}}$ HIGH after issuing the PREN instruction, the programming operation will be ignored.

To program the SerialFlash memory array, the user issues the PROGRAM instruction, followed by the address of the first location in the sector and then the data to be programmed. The data is programmed in a 256-clock operation. $\overline{\text{CS}}$ must go LOW and remain LOW for the duration of the operation. The 32 bytes must reside in the same sector and cannot cross sector boundaries. If the address counter reaches the end of the sector and the clock continues, or if fewer than 32 bytes are clocked in, the contents of the sector cannot be guaranteed.

For the program operation to be completed, $\overline{\text{CS}}$ can only be brought HIGH after bit 0 of data byte 32 is clocked in. If it is brought HIGH at any other time, the program operation will not be completed. Refer to Figure 4 below for a detailed illustration of the programming sequence and time frames in which $\overline{\text{CS}}$ going HIGH is valid.

To program the status register, the PRSR instruction is followed by the data to be programmed. Data bits 0, 1, 4, 5 and 6 must be "0". This sequence is shown in Figure 5.

While the program cycle is in progress, following a status register or memory write sequence, the status register may be read to check the PIP bit. During this time the PIP bit will be HIGH.

Hold Operation

The $\overline{\text{HOLD}}$ input should be HIGH (at V_{IH}) under normal operation. If a data transfer is to be interrupted $\overline{\text{HOLD}}$ can be pulled LOW to suspend the transfer until it can be resumed. The only restriction is that the SCK input must be LOW when $\overline{\text{HOLD}}$ is first pulled LOW and SCK must also be LOW when $\overline{\text{HOLD}}$ is released.

The $\overline{\text{HOLD}}$ input may be tied HIGH either directly to V_{CC} or tied to V_{CC} through a resistor.

Operational Notes

The device powers-up in the following state:

- The device is in the low power standby state.
- A HIGH to LOW transition on $\overline{CS}$ is required to enter an active state and receive an instruction.
- SO pin is high impedance.
- The program enable latch is reset.

Data Protection

The following circuitry has been included to prevent inadvertent programming:

- The program enable latch is reset upon power-up.
- A program enable instruction must be issued to set the program enable latch.
- $\overline{CS}$ must come HIGH at the proper clock count in order to start a program cycle.

Figure 1. Read SerialFlash Memory Array Operation Sequence

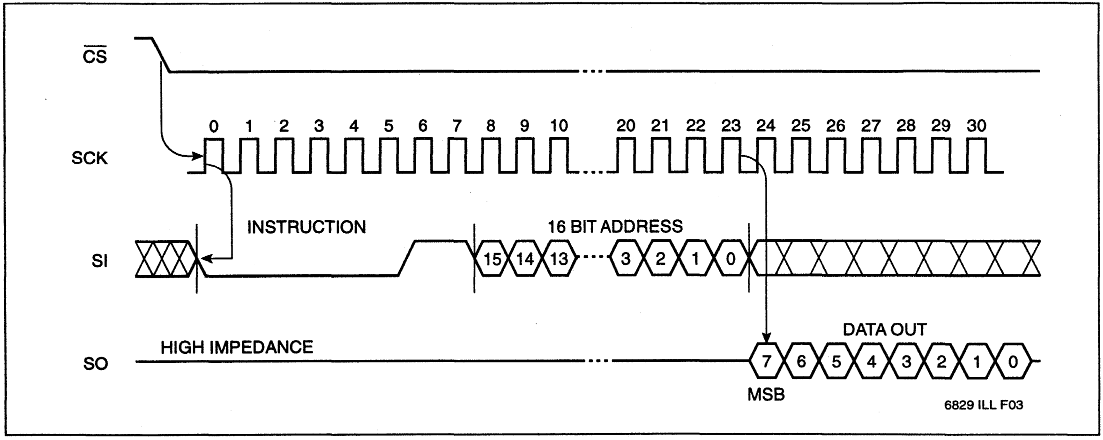


Figure 2. Read Status Register Operation Sequence

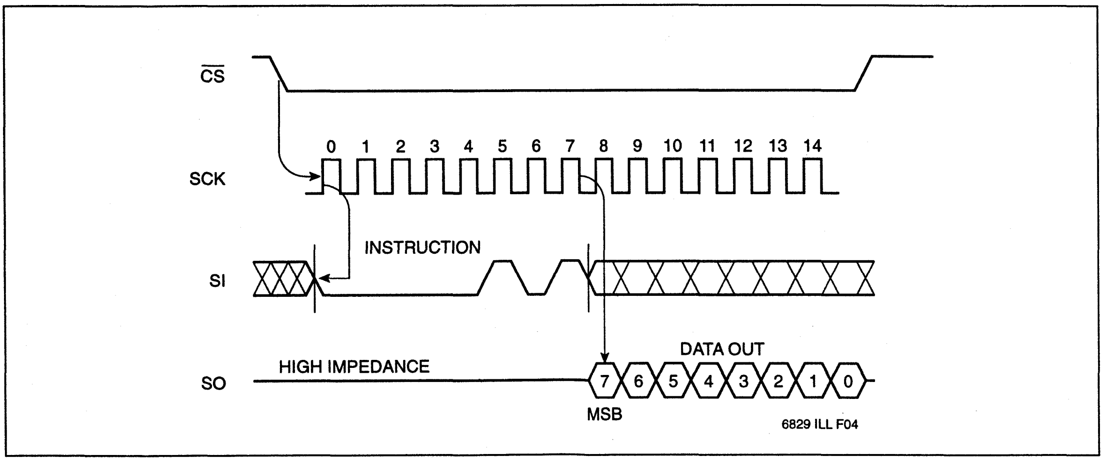


Figure 3. Program Enable Latch Sequence

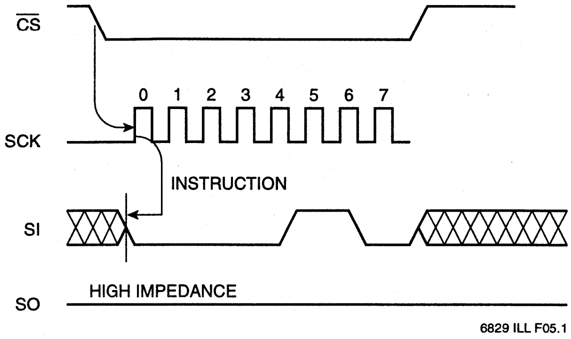


Figure 4. Programming Sequence

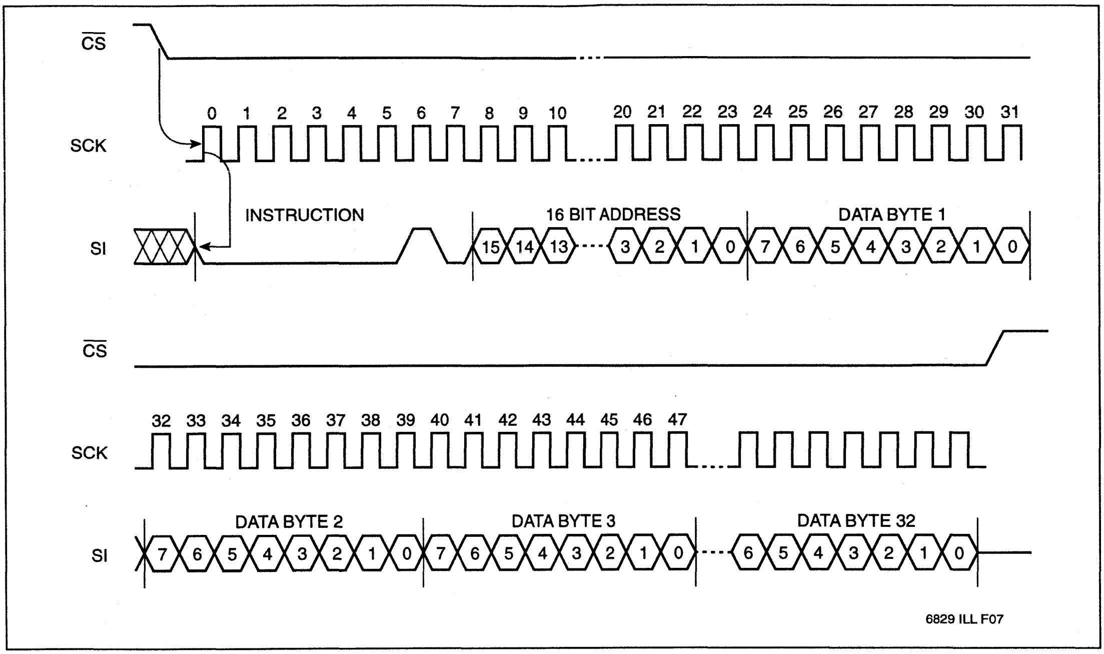
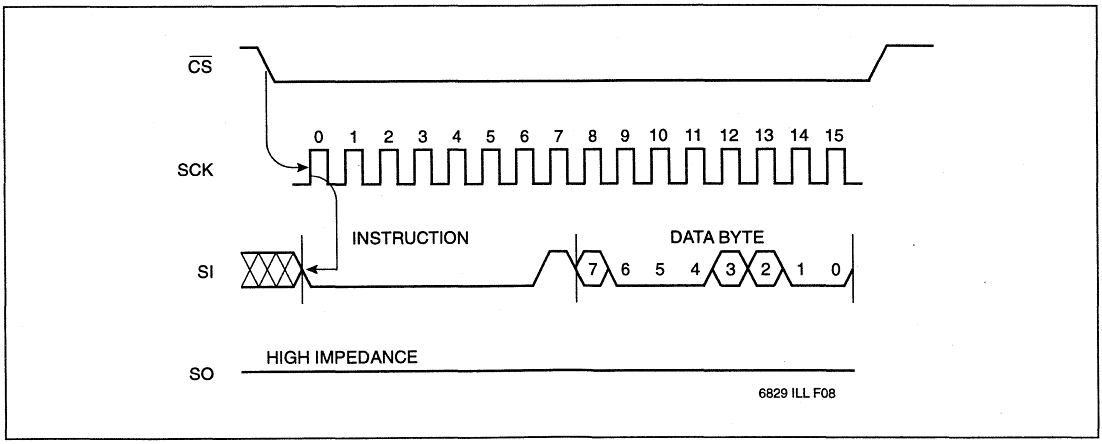


Figure 5. Program Status Register Operation Sequence



ABSOLUTE MAXIMUM RATINGS*

Temperature Under Bias .... –65°C to +135°C
Storage Temperature .... –65°C to +150°C
Voltage on any Pin with Respect to V_{SS} .... –1V to +7V
D.C. Output Current 5mA
Lead Temperature
(Soldering, 10 Seconds) .... 300°C

*COMMENT

Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and the functional operation of the device at these or any other conditions above those listed in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

RECOMMENDED OPERATING CONDITIONS

Temp	Min.	Max.
Commercial	0°C	+70°C
Extended	–20°C	+85°C
Industrial	–40°C	+85°C

6829 PGM T06.1

Supply Voltage	Limits
X25F128	1.8V to 3.6V
X25F128–5	4.5V to 5.5V

6829 PGM T07.1

D.C. OPERATING CHARACTERISTICS (Over the recommended operating conditions unless otherwise specified.)

		Limits			
Symbol	Parameter	Min.	Max.	Units	Test Conditions
I_{CC}	V_{CC} Supply Current (Active)		5	mA	SCK = V_{CC} x 0.1/V_{CC} x 0.9 @ 1MHz, SO = OPEN, $\overline{CS}$ = V_{SS}
I_{SB1}(2)	V_{CC} Supply Current (Standby)		1	µA	$\overline{CS}$ = V_{CC}, V_{IN} = V_{SS} or V_{CC}, V_{CC} = 3.6V
I_{SB2}	V_{CC} Supply Current (Standby)		10	µA	$\overline{CS}$ = V_{CC}, V_{IN} = V_{SS} or V_{CC}, V_{CC} = 5V
I_{LI}	Input Leakage Current		10	µA	V_{IN} = V_{SS} to V_{CC}
I_{LO}	Output Leakage Current		10	µA	V_{OUT} = V_{SS} to V_{CC}
V_{IL}(1)	Input LOW Voltage	–0.5	V_{CC} x 0.3	V	
V_{IH}(1)	Input HIGH Voltage	V_{CC} x 0.7	V_{CC} + 0.5	V	
V_{OL1}	Output LOW Voltage		0.4	V	I_{OL} = 1.5mA, V_{CC} = 2.7V
V_{OH1}	Output HIGH Voltage	V_{CC} – 0.3		V	I_{OH} = –0.4mA, V_{CC} = 2.7V
V_{OL2}	Output LOW Voltage		0.4	V	I_{OL} = 3mA, V_{CC} = 5V
V_{OH2}	Output HIGH Voltage	V_{CC} – 0.8		V	I_{OH} = –1.6mA, V_{CC} = 5V

6829 PGM T08.1

POWER-UP TIMING

Symbol	Parameter	Min.	Max.	Units
t_{PUR}(3)	Power-up to Read Operation		1	ms
t_{PUW}(3)	Power-up to Write Operation		5	ms

6829 PGM T09

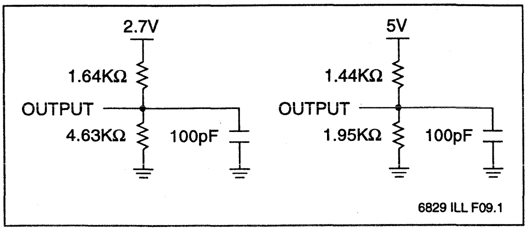
CAPACITANCE T_A = 25°C, f = 1MHz, V_{CC} = 5V.

Symbol	Test	Max.	Units	Conditions
C_{OUT}(2)	Output Capacitance (SO)	8	pF	V_{OUT} = 0V
C_{IN}(2)	Input Capacitance (SCK, SI, $\overline{CS}$, $\overline{PP}$, $\overline{HOLD}$)	6	pF	V_{IN} = 0V

6829 PGM T10

Notes: (1) V_{IL} min. and V_{IH} max. are for reference only and are not tested.
(2) This parameter is periodically sampled and not 100% tested.
(3) t_{PUR} and t_{PUW} are the delays required from the time V_{CC} is stable until the specified operation can be initiated. These parameters are periodically sampled and not 100% tested.

EQUIVALENT A.C. LOAD CIRCUIT



6829 ILL F09.1

A.C. TEST CONDITIONS

Input Pulse Levels	V_{CC} x 0.1 to V_{CC} x 0.9
Input Rise and Fall Times	10ns
Input and Output Timing Level	V_{CC} x 0.5

6829 PGM T11

A.C. CHARACTERISTICS (Over recommended operating conditions, unless otherwise specified)

Data Input Timing

Symbol	Parameter	Min.	Max.	Units
f_{SCK}	Clock Frequency	0	1	MHz
t_{CYC}	Cycle Time	1000		ns
t_{LEAD}	$\overline{CS}$ Lead Time	500		ns
t_{LAG}	$\overline{CS}$ Lag Time	500		ns
t_{WH}	Clock HIGH Time	400		ns
t_{WL}	Clock LOW Time	400		ns
t_{SU}	Data Setup Time	100		ns
t_{H}	Data Hold Time	100		ns
t_{RI}(4)	Data In Rise Time		2	µs
t_{FI}(4)	Data In Fall Time		2	µs
t_{HD}	$\overline{HOLD}$ Setup Time	200		ns
t_{CD}	$\overline{HOLD}$ Hold Time	200		ns
t_{CS}	$\overline{CS}$ Deselect Time	2		µs
t_{PC}(5)	Program Cycle Time		10	ms

6829 PGM T12.1

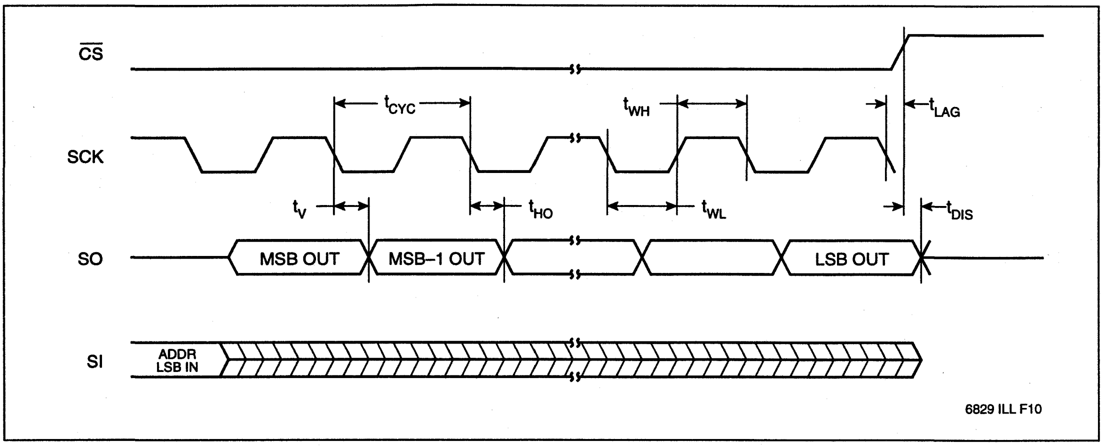
Data Output Timing

Symbol	Parameter	Min.	Max.	Units
f_{SCK}	Clock Frequency	0	1	MHz
t_{DIS}	Output Disable Time		500	ns
t_{V}	Output Valid from Clock LOW		400	ns
t_{HO}	Output Hold Time	0		ns
t_{RO}(4)	Output Rise Time		300	ns
t_{FO}(4)	Output Fall Time		300	ns
t_{LZ}(4)	$\overline{HOLD}$ HIGH to Output in Low Z	100		ns
t_{HZ}(4)	$\overline{HOLD}$ LOW to Output in High Z	100		ns

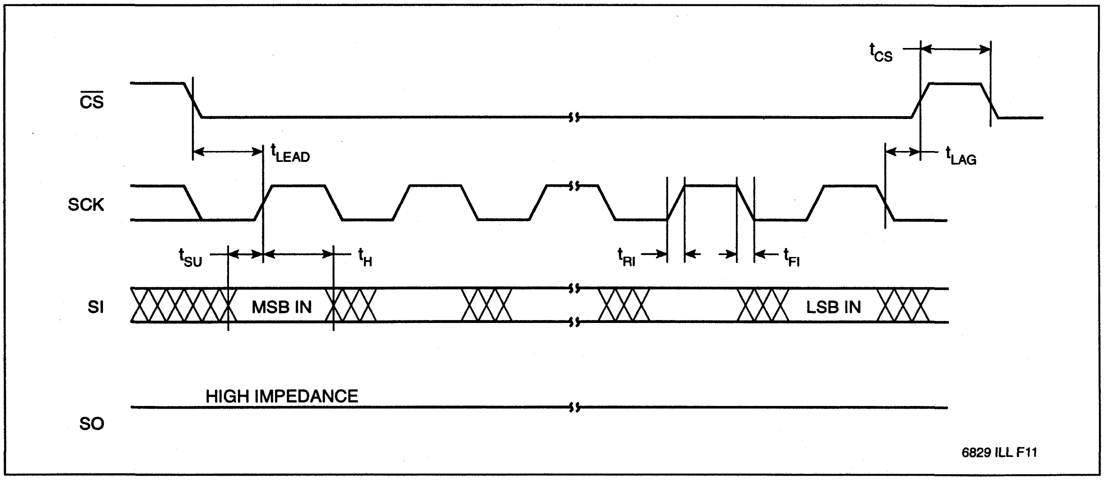
6829 PGM T13

Notes: (4) This parameter is periodically sampled and not 100% tested.
(5) t_{WC} is the time from the rising edge of $\overline{CS}$ after a valid write sequence has been sent to the end of the self-timed internal nonvolatile program cycle.

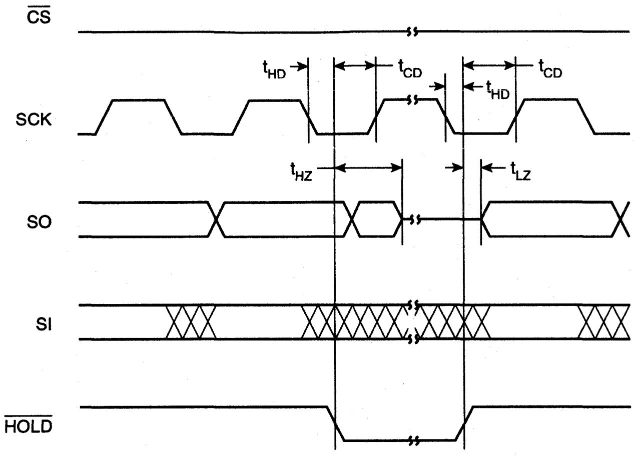
Serial Output Timing



Serial Input Timing



Hold Timing



6829 ILL F12

ORDERING INFORMATION

X25F128 P T -X

Device — X25F128

V_{CC} **Range**
Blank = 1.8V to 3.6V
5 = 4.5V to 5.5V

Temperature Range
Blank = Commercial = 0°C to +70°C
E = Extended = –20°C to +85°C
I = Industrial = –40°C to +85°C

Package
P = 8-Lead Plastic DIP
S = 16-Lead SOIC

Part Mark Convention

X25F128 — P = 8-Lead Plastic DIP
S = 16-Lead SOIC

X

Blank = 1.8V to 3.6V, 0°C to +70°C
5 = 4.5V to 5.5V, 0°C to +70°C
I5 = 4.5V to 5.5V, –40°C to +85°C

LIMITED WARRANTY

Devices sold by Xicor, Inc. are covered by the warranty and patent indemnification provisions appearing in its Terms of Sale only. Xicor, Inc. makes no warranty, express, statutory, implied, or by description regarding the information set forth herein or regarding the freedom of the described devices from patent infringement. Xicor, Inc. makes no warranty of merchantability or fitness for any purpose. Xicor, Inc. reserves the right to discontinue production and change specifications and prices at any time and without notice.

Xicor, Inc. assumes no responsibility for the use of any circuitry other than circuitry embodied in a Xicor, Inc. product. No other circuits, patents, licenses are implied.

U.S. PATENTS

Xicor products are covered by one or more of the following U.S. Patents: 4,263,664; 4,274,012; 4,300,212; 4,314,265; 4,326,134; 4,393,481; 4,404,475; 4,450,402; 4,486,769; 4,488,060; 4,520,461; 4,533,846; 4,599,706; 4,617,652; 4,668,932; 4,752,912; 4,829, 482; 4,874, 967; 4,883, 976. Foreign patents and additional patents pending.

LIFE RELATED POLICY

In situations where semiconductor component failure may endanger life, system designers using this product should design the system with appropriate error detection and correction, redundancy and backup features to prevent such an occurrence.

Xicor's products are not authorized for use in critical components in life support devices or systems.

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform, when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.
2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

APPLICATION NOTE
A V A I L A B L E
AN61 • AN75 • AN77 • AN79 • AN82

Xicor®

X25F064/032/016/008

SerialFlash™ Memory With Block Lock™ Protection

FEATURES

- 1MHz Clock Rate
- SPI Serial Interface
- 64K/32K/16K/8K Bits
  —32 Byte Small Sector Program Mode
- Low Power CMOS
  —<1µA Standby Current
  —<5mA Active Current
- 1.8V – 3.6V or 5V "Univolt" Read and Program Power Supply Versions
- Block Lock Protection
  —Protect 1/4, 1/2, or all of E²PROM Array
- Built-in Inadvertent Program Protection
  —Power-Up/Power-Down protection circuitry
  —Program Enable Latch
  —Program Protect Pin
- Self-Timed Program Cycle
  —5ms Program Cycle Time (Typical)
- High Reliability
  —Endurance: 100,000 cycles per byte
  —Data Retention: 100 Years
  —ESD protection: 2000V on all pins
- 8-Lead PDIP Package
- 8-Lead 150 mil SOIC Packages
- 32K, 16K, 8K available in 14-Lead TSSOP, 64K available in 20-Lead TSSOP

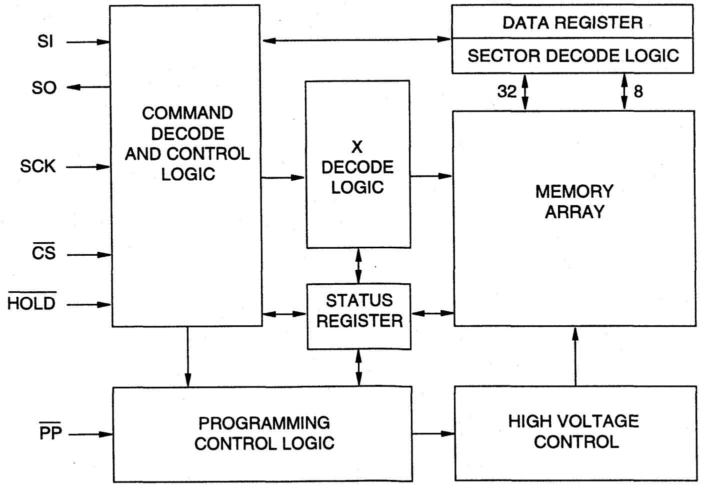
DESCRIPTION

The X25F064/032/016/008 family are 8/16/32/64K-bit CMOS SerialFlash memory, internally organized X 8. They feature a "Univolt" Program and Read voltage, Serial Peripheral Interface (SPI), and software protocol allowing operation on a simple three-wire bus. The bus signals are a clock input (SCK), plus separate data in (SI) and data out (SO) lines. Access to the device is controlled through a chip select ($\overline{CS}$) input, allowing any number of devices to share the same bus.

The X25F064/032/016/008 also features two additional inputs that provide the end user with added flexibility. By asserting the $\overline{HOLD}$ input, the X25F064/032/016/008 will ignore transitions on its inputs, thus allowing the host to service higher priority interrupts. The $\overline{PP}$ input can be used as a hardwire input to the X25F064/032/016/008 disabling all program attempts to the status register, thus providing a mechanism for limiting end user capability of altering 0, 1/4, 1/2, or all of the memory.

The X25F064/032/016/008 utilizes Xicor's proprietary flash cell, providing a minimum endurance of 100,000 cycles and a minimum data retention of 100 years.

FUNCTIONAL DIAGRAM



6685 ILL F01.4

SerialFlash™ and Block Lock™ Protection are trademarks of Xicor, Inc.

PIN DESCRIPTIONS

Serial Output (SO)

SO is a push-pull serial data output pin. During a read cycle, data is shifted out on this pin. Data is clocked out by the falling edge of the serial clock.

Serial Input (SI)

SI is the serial data input pin. All opcodes, byte addresses, and data to be written to the memory are input on this pin. Data is latched by the rising edge of the serial clock.

Serial Clock (SCK)

The Serial Clock controls the serial bus timing for data input and output. Opcodes, addresses, or data present on the SI pin are latched on the rising edge of the clock input, while data on the SO pin change after the falling edge of the clock input.

Chip Select ($\overline{\text{CS}}$)

When $\overline{\text{CS}}$ is HIGH, the X25F064/032/016/008 is deselected and the SO output pin is at high impedance and unless an internal program operation is underway the X25F064/032/016/008 will be in the standby power mode. $\overline{\text{CS}}$ LOW enables the X25F064/032/016/008, placing it in the active power mode. It should be noted that after power-up, a HIGH to LOW transition on $\overline{\text{CS}}$ is required prior to the start of any operation.

Program Protect ($\overline{\text{PP}}$)

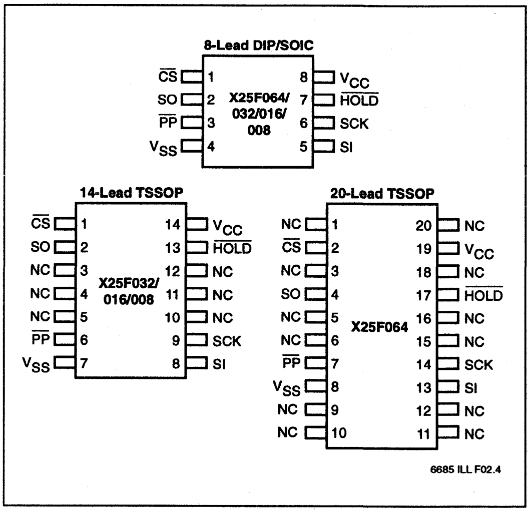
When $\overline{\text{PP}}$ is LOW and the nonvolatile bit PPEN is "1", nonvolatile programming of the X25F064/032/016/008 status register is disabled, but the part otherwise functions normally. When $\overline{\text{PP}}$ is held HIGH, all functions, including nonvolatile programming operate normally. $\overline{\text{PP}}$ going LOW while $\overline{\text{CS}}$ is still LOW will interrupt programming of the X25F064/032/016/008 status register. If the internal program cycle has already been initiated, $\overline{\text{PP}}$ going LOW will have no effect on programming.

The $\overline{\text{PP}}$ pin function is blocked when the PPEN bit in the status register is "0". This allows the user to install the X25F064/032/016/008 into a system with $\overline{\text{PP}}$ pin grounded and still be able to program the status register. The $\overline{\text{PP}}$ pin functions will be enabled when the PPEN bit is set "0".

Hold ($\overline{\text{HOLD}}$)

$\overline{\text{HOLD}}$ is used in conjunction with the $\overline{\text{CS}}$ pin to select the device. Once the part is selected and a serial sequence is underway, $\overline{\text{HOLD}}$ may be used to pause the serial communication with the controller without resetting the serial sequence. To pause, $\overline{\text{HOLD}}$ must be brought LOW while SCK is LOW. To resume communication, $\overline{\text{HOLD}}$ is brought HIGH, again while SCK is LOW. If the pause feature is not used, $\overline{\text{HOLD}}$ should be held HIGH at all times.

PIN CONFIGURATION



PIN NAMES

SYMBOL	DESCRIPTION
$\overline{\text{CS}}$	Chip Select Input
SO	Serial Output
SI	Serial Input
SCK	Serial Clock Input
$\overline{\text{PP}}$	Program Protect Input
V_{SS}	Ground
V_{CC}	Supply Voltage
$\overline{\text{HOLD}}$	Hold Input
NC	No Connect

6685 PGM T01.1

PRINCIPLES OF OPERATION

The X25F064/032/016/008 family are SerialFlash Memory designed to interface directly with the synchronous serial peripheral interface (SPI) of many popular microcontroller families.

The X25F064/032/016/008 family contains an 8-bit instruction register. It is accessed via the SI input, with data being clocked in on the rising SCK. $\overline{\text{CS}}$ must be LOW and the $\overline{\text{HOLD}}$ and $\overline{\text{PP}}$ inputs must be HIGH during the entire operation. The $\overline{\text{PP}}$ input is "Don't Care" if PPEN is set "0".

Table 1 contains a list of the instructions and their operation codes. All instructions, addresses and data are transferred MSB first.

Data input is sampled on the first rising edge of SCK after $\overline{\text{CS}}$ goes LOW. SCK is static, allowing the user to stop the clock and then resume operations. If the clock line is shared with other peripheral devices on the SPI bus, the user can assert the $\overline{\text{HOLD}}$ input to place the X25F064/032/016/008 into a "PAUSE" condition. After releasing $\overline{\text{HOLD}}$, the X25F064/032/016/008 device will resume operation from the point when $\overline{\text{HOLD}}$ was first asserted.

Program Enable Latch

The X25F064/032/016/008 device contains a program enable latch. This latch must be SET before a program operation will be completed internally. The PREN instruction will set the latch and the PRDI instruction will reset the latch. This latch is automatically reset on power-up and after the completion of a sector program or status register write cycle.

Status Register

The RDSR instruction provides access to the status register. The status register may be read at any time, even during a program cycle. The status register is formatted as follows:

7	6	5	4	3	2	1	0
PPEN	X	X	X	BL1	BL0	PEL	PIP

6685 PGM T02.2

PPEN, BL0, and BL1 are set by the PRSR instruction. PEL and PIP are "read-only" and automatically set by other operations.

The Programming-In-Process (PIP) bit indicates whether the X25F064/032/016/008 device is busy with a program operation. When set to a "1" programming is in progress, when set to a "0" no programming is in progress. During programming, all other bits are set to "1".

The Program Enable Latch (PEL) bit indicates the status of the program enable latch. When set to a "1" the latch is set; when set to a "0" the latch is reset.

The Block Lock (BL0 and BL1) bits are nonvolatile and allow the user to select one of four levels of protection. The X25F064/032/016/008 device array is divided into four equal segments. One, two, or all four of the segments may be locked. That is, the user may read the segments, but will be unable to alter (program) data within the selected segments. The partitioning is controlled as illustrated below.

Status Register Bits		Array Addresses
BL1	BL0	Locked
0	0	None
0	1	upper fourth
1	0	upper half
1	1	All

6685 PGM T03.1

Program-Protect Enable

The Program-Protect-Enable bit (PPEN) in the X25F064/032/016/008 status register acts as an enable bit for the $\overline{\text{PP}}$ pin.

Table 1. Instruction Set

Instruction Name	Instruction Format*	Operation
PREN	0000 0110	Set the Program Enable Latch (Enable Program Operations)
PRDI	0000 0100	Reset the Program Enable Latch (Disable Program Operations)
RDSR	0000 0101	Read Status Register
PRSR	0000 0001	Program Status Register
READ	0000 0011	Read from Memory Array beginning at Selected Address
PROGRAM	0000 0010	Program Memory Array beginning at Selected Address (32 Bytes)

6685 PGM T04.2

*Instructions are shown MSB in leftmost position. Instructions are transferred MSB first.

PPEN	$\overline{PP}$	PEL	Locked Blocks	Unlocked Blocks	Status Register
0	X	0	Locked	Locked	Locked
0	X	1	Locked	Programmable	Programmable
1	LOW	0	Locked	Locked	Locked
1	LOW	1	Locked	Programmable	Locked
X	HIGH	0	Locked	Locked	Locked
X	HIGH	1	Locked	Programmable	Programmable

6685 PGM T05.2

The Program Protect ($\overline{PP}$) pin and the nonvolatile Program Protect Enable (PPEN) bit in the Status Register control the programmable hardware write protect feature. Hardware program protection is enabled when $\overline{PP}$ pin is LOW, and the PPEN bit is "1". Hardware program protection is disabled when either the $\overline{PP}$ pin is HIGH or the PPEN bit is "0". When the chip is hardware program protected, nonvolatile programming of the Status Register in disabled, including the Block Lock bits and the PPEN bit itself, as well as the Block Lock sections in the memory array. Only the sections of the memory array that are not Block Locked can be programmed.

Note: Since the PPEN bit is program protected, it cannot be changed back to a "0", as long as the $\overline{PP}$ pin is held LOW.

Clock and Data Timing

Data input on the SI line is latched on the rising edge of SCK. Data is output on the SO line by the falling edge of SCK.

Read Sequence

When reading from the SerialFlash memory array, $\overline{CS}$ is first pulled LOW to select the device. The 8-bit READ instruction is transmitted to the X25F064/032/016/008 device, followed by the 16-bit address. After the read opcode and address are sent, the data stored in the memory at the selected address is shifted out on the SO line. The data stored in memory at the next address can be read sequentially by continuing to provide clock pulses. The address is automatically incremented to the next higher address after each byte of data is shifted out. When the highest address is reached the address counter rolls over to address $0000, allowing the read cycle to be continued indefinitely. The read operation is terminated by taking $\overline{CS}$ HIGH. Refer to the Read SerialFlash Memory Array Operation Sequence illustrated in Figure 1.

To read the status register, the $\overline{CS}$ line is first pulled LOW to select the device followed by the 8-bit instruction. After the read status register opcode is sent, the contents of the status register are shifted out on the SO line. The Read Status Register Sequence is illustrated in Figure 2.

Programming Sequence

Prior to any attempt to program the X25F064/032/016/008 device, the program enable latch must first be set by issuing the PREN instruction (See Figure 3). $\overline{CS}$ is first taken LOW, then the PREN instruction is clocked into the X25F064/032/016/008 device. After all eight bits of the instruction are transmitted, $\overline{CS}$ must then be taken HIGH. If the user continues the programming operation without taking $\overline{CS}$ HIGH after issuing the PREN instruction, the programming operation will be ignored.

To program the SerialFlash memory array, the user issues the PROGRAM instruction, followed by the address of the first location in the sector and then the data to be programmed. The data is programmed in a 256-clock operation. $\overline{CS}$ must go LOW and remain LOW for the duration of the operation. The 32 bytes must reside in the same sector and cannot cross sector boundaries. If the address counter reaches the end of the sector and the clock continues, or if fewer than 32 bytes are clocked in, the contents of the sector cannot be guaranteed.

For the program operation to be completed, $\overline{CS}$ can only be brought HIGH after bit 0 of data byte 32 is clocked in. If it is brought HIGH at any other time the program operation will not be completed. Refer to Figure 4 below for a detailed illustration of the programming sequence and time frames in which $\overline{CS}$ going HIGH is valid.

To program the status register, the PRSR instruction is followed by the data to be programmed. Data bits 0, 1, 4, 5 and 6 must be "0". This sequence is shown in Figure 5.

While the program cycle is in progress, following a status register or memory write sequence, the status register may be read to check the PIP bit. During this time the PIP bit will be HIGH.

Hold Operation

The $\overline{HOLD}$ input should be HIGH (at V_{IH}) under normal operation. If a data transfer is to be interrupted $\overline{HOLD}$ can be pulled LOW to suspend the transfer until it can be resumed. The only restriction is that the SCK input must be LOW when $\overline{HOLD}$ is first pulled LOW and SCK must also be LOW when $\overline{HOLD}$ is released.

The $\overline{HOLD}$ input may be tied HIGH either directly to V_{CC} or tied to V_{CC} through a resistor.

Operational Notes

The device powers-up in the following state:

- The device is in the low power standby state.
- A HIGH to LOW transition on $\overline{CS}$ is required to enter an active state and receive an instruction.
- SO pin is high impedance.
- The program enable latch is reset.

Data Protection

The following circuitry has been included to prevent inadvertent programming:

- The program enable latch is reset upon power-up.
- A program enable instruction must be issued to set the program enable latch.
- $\overline{CS}$ must come HIGH at the proper clock count in order to start a program cycle.

Figure 1. Read SerialFlash Memory Array Operation Sequence

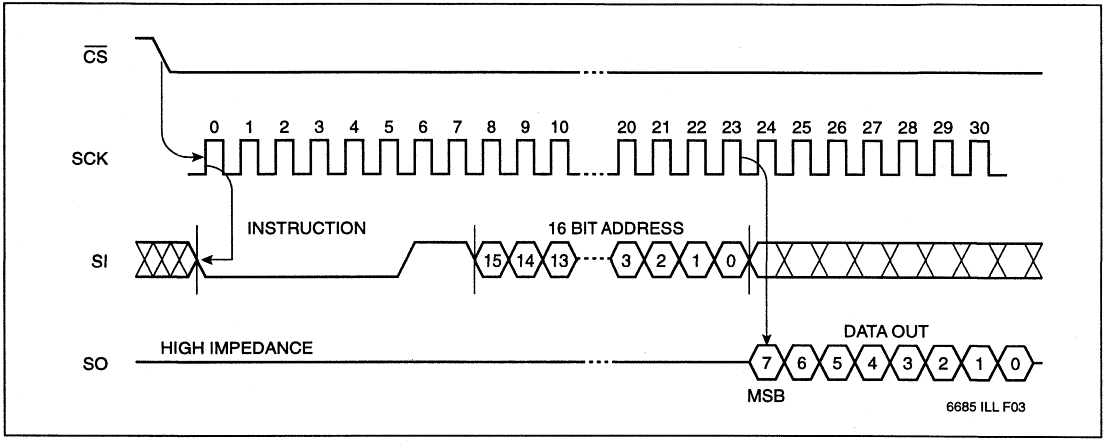


Figure 2. Read Status Register Operation Sequence

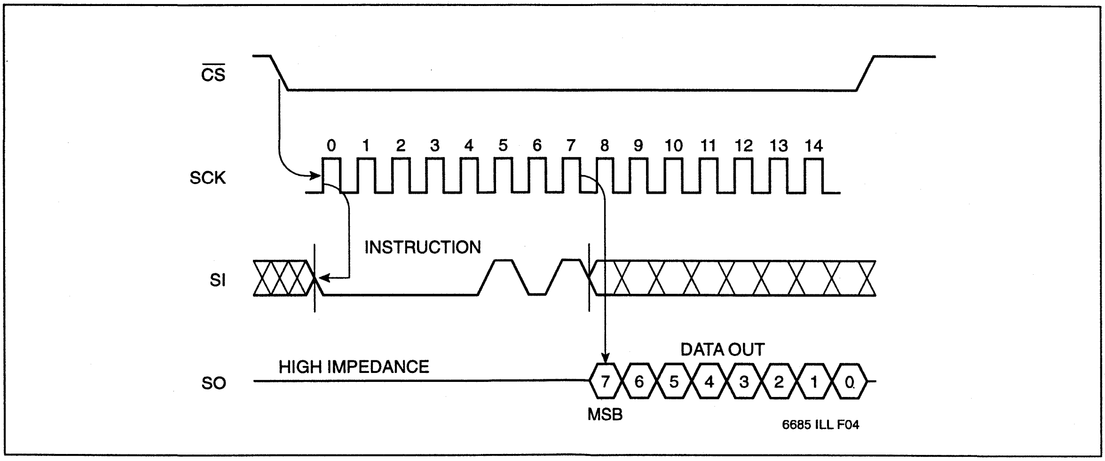
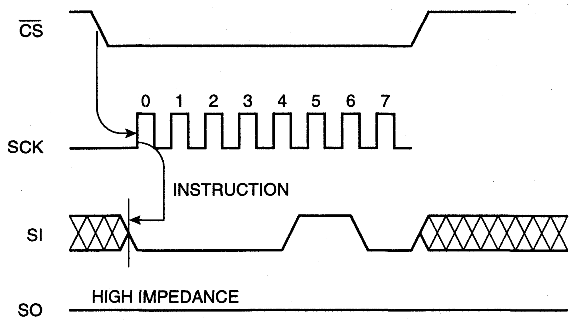


Figure 3. Program Enable Latch Sequence



6685 ILL F05.1

Figure 4. Programming Sequence

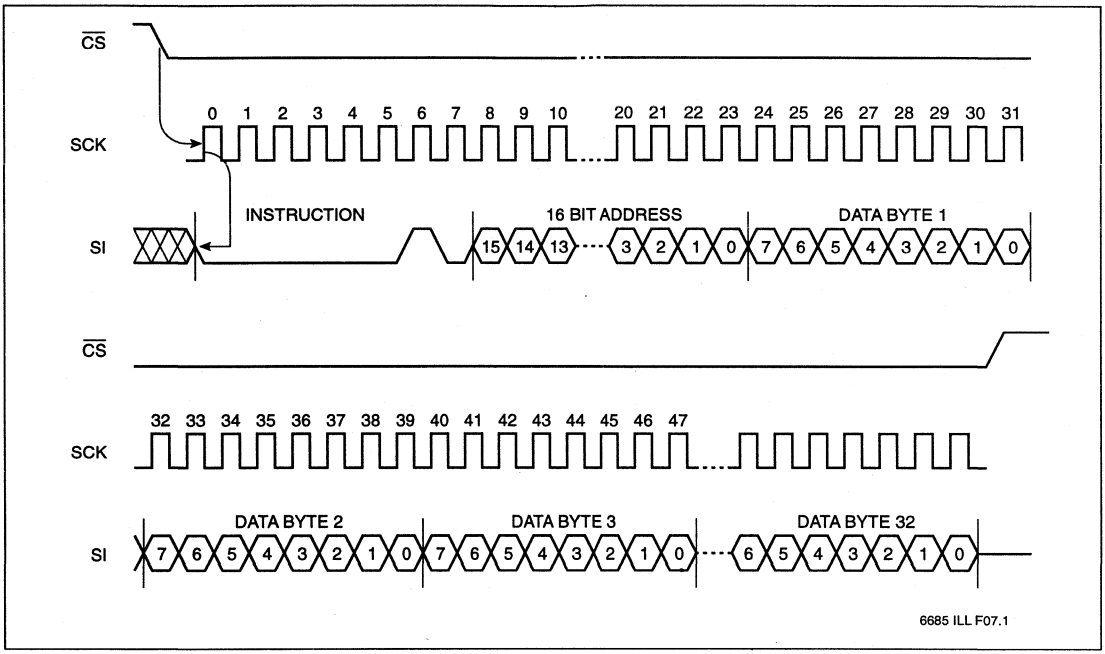
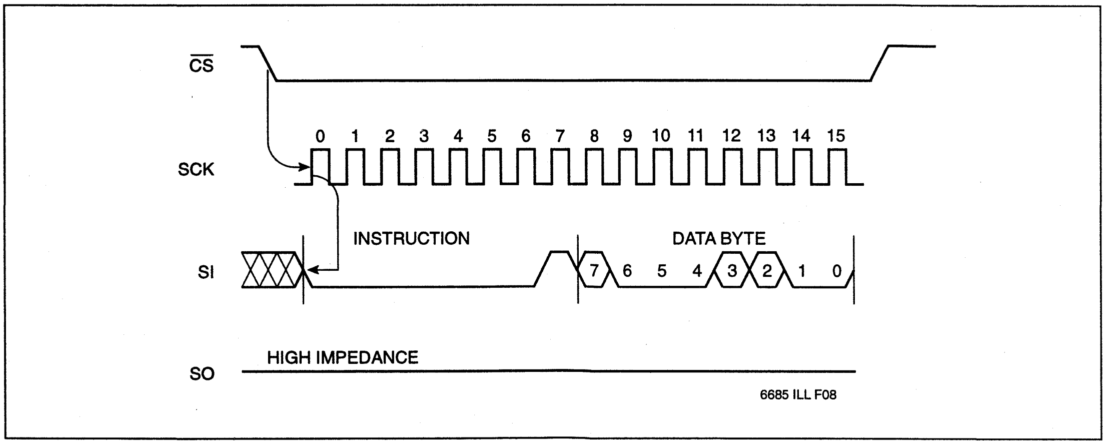


Figure 5. Program Status Register Operation Sequence



ABSOLUTE MAXIMUM RATINGS*

Temperature Under Bias	–65°C to +135°C
Storage Temperature	–65°C to +150°C
Voltage on any Pin with Respect to V_{SS}	–1V to +7V
D.C. Output Current	5mA
Lead Temperature (Soldering, 10 Seconds)	300°C

*COMMENT

Stresses above those listed under “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only and the functional operation of the device at these or any other conditions above those listed in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

RECOMMENDED OPERATING CONDITIONS

Temp	Min.	Max.
Commercial	0°C	+70°C
Extended	–20°C	+85°C
Industrial	–40°C	+85°C

6685 PGM T06.2

Supply Voltage	Limits
X25F064/032/016/008	1.8V to 3.6V
X25F064/032/016/008–5	4.5V to 5.5V

6685 PGM T07.3

D.C. OPERATING CHARACTERISTICS (Over the recommended operating conditions unless otherwise specified.)

		Limits			
Symbol	Parameter	Min.	Max.	Units	Test Conditions
I_{CC}	V_{CC} Supply Current (Active)		5	mA	SCK = V_{CC} x 0.1/V_{CC} x 0.9 @ 1MHz, SO = Open, $\overline{CS}$ = V_{SS}
I_{SB1}(2)	V_{CC} Supply Current (Standby)		1	µA	$\overline{CS}$ = V_{CC}, V_{IN} = V_{SS} or V_{CC}, V_{CC} = 3.6V
I_{SB2}	V_{CC} Supply Current (Standby)		10	µA	$\overline{CS}$ = V_{CC}, V_{IN} = V_{SS} or V_{CC}, V_{CC} = 5V
I_{LI}	Input Leakage Current		10	µA	V_{IN} = V_{SS} to V_{CC}
I_{LO}	Output Leakage Current	–1	10	µA	V_{OUT} = V_{SS} to V_{CC}
V_{IL}(1)	Input LOW Voltage	V_{CC} x 0.7	V_{CC} x 0.3	V	
V_{IH}(1)	Input HIGH Voltage		V_{CC} + 0.5	V	
V_{OL1}	Output LOW Voltage		0.4	V	I_{OL} = 1.5mA, V_{CC} = 2.7V
V_{OH1}	Output HIGH Voltage	V_{CC} – 0.3		V	I_{OH} = –0.4mA, V_{CC} = 2.7V
V_{OL2}	Output LOW Voltage		0.4	V	I_{OL} = 3mA, V_{CC} = 5V
V_{OH2}	Output HIGH Voltage	V_{CC} – 0.8		V	I_{OH} = –1.6mA, V_{CC} = 5V

6685 PGM T08.4

POWER-UP TIMING

Symbol	Parameter	Min.	Max.	Units
t_{PUR}(3)	Power-up to Read Operation		1	ms
t_{PUW}(3)	Power-up to Write Operation		5	ms

6685 PGM T09

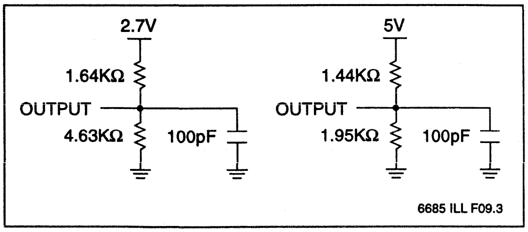
CAPACITANCE T_A = 25°C, f = 1MHz, V_{CC} = 5V.

Symbol	Test	Max.	Units	Conditions
C_{OUT}(2)	Output Capacitance (SO)	8	pF	V_{OUT} = 0V
C_{IN}(2)	Input Capacitance (SCK, SI, $\overline{CS}$, $\overline{WP}$, $\overline{HOLD}$)	6	pF	V_{IN} = 0V

6685 PGM T10.1

Notes: (1) V_{IL} min. and V_{IH} max. are for reference only and are not tested.
(2) This parameter is periodically sampled and not 100% tested.
(3) t_{PUR} and t_{PUW} are the delays required from the time V_{CC} is stable until the specified operation can be initiated. These parameters are periodically sampled and not 100% tested.

EQUIVALENT A.C. LOAD CIRCUIT



A.C. TEST CONDITIONS

Input Pulse Levels	V_{CC} x 0.1 to V_{CC} x 0.9
Input Rise and Fall Times	10ns
Input and Output Timing Level	V_{CC} x 0.5

6685 PGM T11

A.C. CHARACTERISTICS (Over recommended operating conditions, unless otherwise specified)

Data Input Timing

Symbol	Parameter	Min.	Max.	Units
f_{SCK}	Clock Frequency	0	1	MHz
t_{CYC}	Cycle Time	1000		ns
t_{LEAD}	$\overline{CS}$ Lead Time	500		ns
t_{LAG}	$\overline{CS}$ Lag Time	500		ns
t_{WH}	Clock HIGH Time	400		ns
t_{WL}	Clock LOW Time	400		ns
t_{SU}	Data Setup Time	100		ns
t_H	Data Hold Time	100		ns
t_{RI}(4)	Data In Rise Time		2	µs
t_{FI}(4)	Data In Fall Time		2	µs
t_{HD}	$\overline{HOLD}$ Setup Time	200		ns
t_{CD}	$\overline{HOLD}$ Hold Time	200		ns
t_{CS}	$\overline{CS}$ Deselect Time	2		µs
t_{PC}(5)	Program Cycle Time		10	ms

6685 PGM T12.3

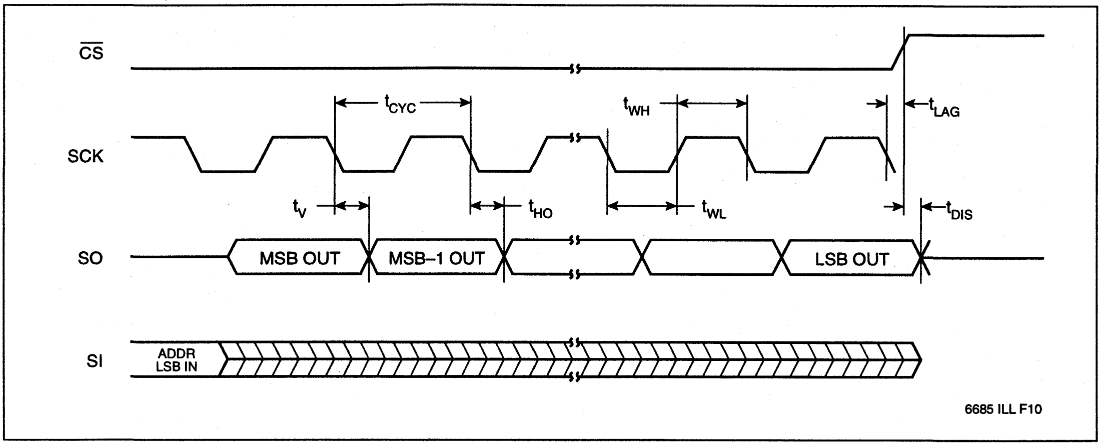
Data Output Timing

Symbol	Parameter	Min.	Max.	Units
f_{SCK}	Clock Frequency	0	1	MHz
t_{DIS}	Output Disable Time		500	ns
t_V	Output Valid from Clock LOW		400	ns
t_{HO}	Output Hold Time	0		ns
t_{RO}(4)	Output Rise Time		300	ns
t_{FO}(4)	Output Fall Time		300	ns
t_{LZ}(4)	$\overline{HOLD}$ HIGH to Output in Low Z	100		ns
t_{HZ}(4)	$\overline{HOLD}$ LOW to Output in High Z	100		ns

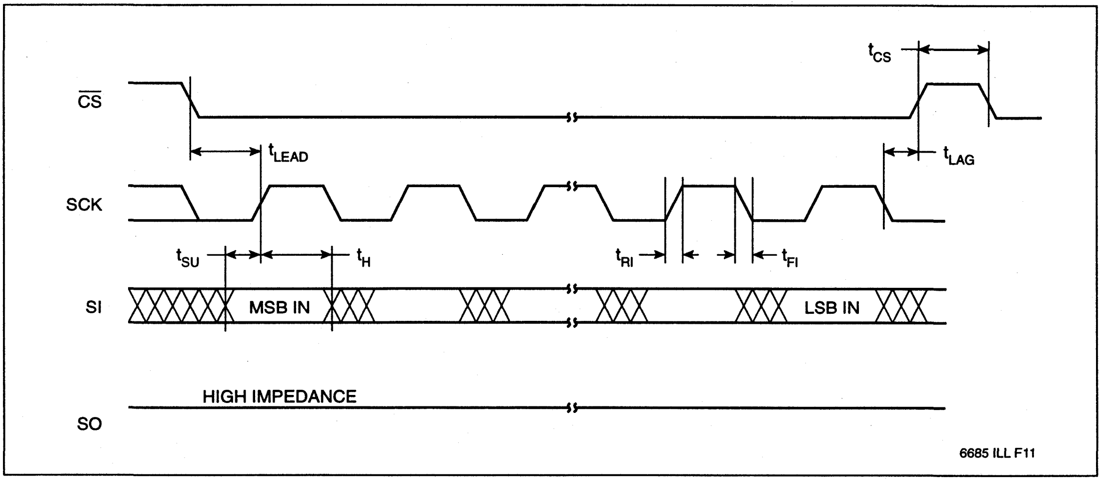
6685 PGM T13.2

Notes: (4) This parameter is periodically sampled and not 100% tested.
(5) t_{WC} is the time from the rising edge of $\overline{CS}$ after a valid write sequence has been sent to the end of the self-timed internal nonvolatile program cycle.

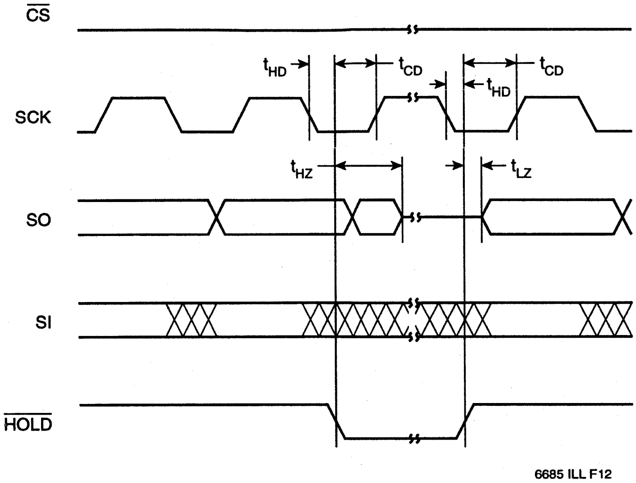
Serial Output Timing



Serial Input Timing



Hold Timing


$\overline{CS}$
SCK
SO
SI
$\overline{HOLD}$
t_{HD}
t_{CD}
t_{HD}
t_{CD}
t_{HZ}
t_{LZ}
6685 ILL F12

ORDERING INFORMATION

X25FXXX P T -X

Device — X25FXXX

V_{CC} Range
Blank = 1.8V to 3.6V
5 = 4.5V to 5.5V

Temperature Range
Blank = Commercial = 0°C to +70°C
E = Extended = –20°C to +85°C
I = Industrial = –40°C to +85°C

Package

X25F064	**X25F032**
P = 8-Lead Plastic DIP	**X25F016**
S = 8-Lead SOIC	**X25F008**
V = 20-Lead TSSOP	P = 8-Lead Plastic DIP
	S = 8-Lead SOIC
	V = 14-Lead TSSOP

Part Mark Convention

X25FXXX
P = 8-Lead Plastic DIP
Blank = 8-Lead SOIC
V = 14/20-Lead TSSOP

X25F064
X25F032
X25F016
X25F008

X
Blank = 1.8V to 3.6V, 0°C to +70°C
5 = 4.5V to 5.5V, 0°C to +70°C
I5 = 4.5V to 5.5V, –40°C to +85°C

LIMITED WARRANTY

Devices sold by Xicor, Inc. are covered by the warranty and patent indemnification provisions appearing in its Terms of Sale only. Xicor, Inc. makes no warranty, express, statutory, implied, or by description regarding the information set forth herein or regarding the freedom of the described devices from patent infringement. Xicor, Inc. makes no warranty of merchantability or fitness for any purpose. Xicor, Inc. reserves the right to discontinue production and change specifications and prices at any time and without notice.

Xicor, Inc. assumes no responsibility for the use of any circuitry other than circuitry embodied in a Xicor, Inc. product. No other circuits, patents, licenses are implied.

U.S. PATENTS

Xicor products are covered by one or more of the following U.S. Patents: 4,263,664; 4,274,012; 4,300,212; 4,314,265; 4,326,134; 4,393,481; 4,404,475; 4,450,402; 4,486,769; 4,488,060; 4,520,461; 4,533,846; 4,599,706; 4,617,652; 4,668,932; 4,752,912; 4,829, 482; 4,874, 967; 4,883, 976. Foreign patents and additional patents pending.

LIFE RELATED POLICY

In situations where semiconductor component failure may endanger life, system designers using this product should design the system with appropriate error detection and correction, redundancy and backup features to prevent such an occurrence.

Xicor's products are not authorized for use in critical components in life support devices or systems.

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform, when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.
2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

Application Note
AVAILABLE
AN75

Advance Information

8K **X25F087** **1024 x 8 Bit**

SPI SerialFlash with Block Lock^{TM} Protection

FEATURES

- 1MHz Clock Rate
- SPI Modes (0,0 & 1,1)
- 512 x 8 Bits
  - —16 Byte Small Sector Program Mode
- Low Power CMOS
  - —<1µA Standby Current
  - —<3mA Active Current during Program
  - —<400µA Active Current during Read
- 1.8V to 3.6V or 5V "Univolt" Read and Program Power Supply Versions
- Block Lock Protection
  - —Block Lock Protect 0, any 1/4, 1st 1/2, First or Last Sector of SerialFlash Array
- Built-in Inadvertent Program Protection
  - —Power-Up/Power-Down Protection Circuitry
  - —Program Enable Latch
  - —Program Protect Pin
- Self-Timed Program Cycle
  - —5ms Program Cycle Time (Typical)
- High Reliability
  - —Endurance: 100,000 Cycles/Byte
  - —Data Retention: 100 Years
  - —ESD: 2000V on all pins
- 8-Pin Mini-DIP Package
- 8-Lead SOIC Package

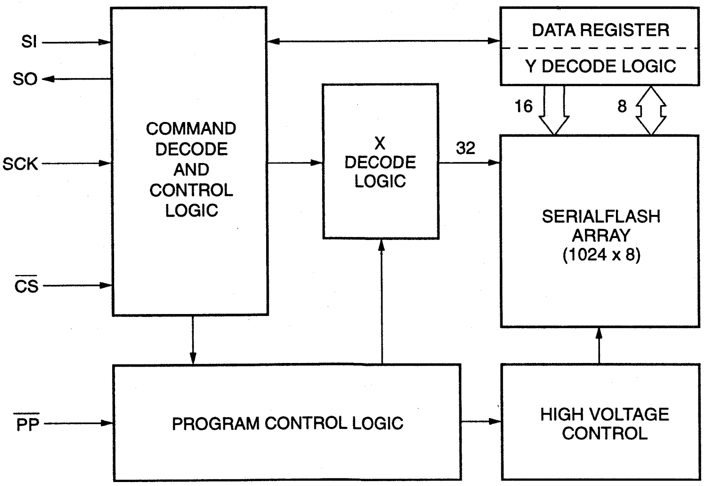
DESCRIPTION

The X25F087 is a CMOS 8k-bit SerialFlash, internally organized as 512 x 8. The X25F087 features a Serial Peripheral Interface (SPI) and software protocol allowing operation on a simple four-wire bus. The bus signals are a clock input (SCK) plus separate data in (SI) and data out (SO) lines. Access to the device is controlled through a chip select ($\overline{CS}$) input, allowing any number of devices to share the same bus.

There are eight options for programmable, nonvolatile, Block Lock Protection available to the end user. These options are implemented via special instructions programmed to the part. The X25F087 also features a $\overline{PP}$ pin that can be used for hardwire protection of the part, disabling all programming attempts, as well as a Program Enable Latch that must be set before a program operation can be initiated.

The X25F087 utilizes Xicor's proprietary Direct Write^{TM} cell, providing a minimum endurance of 100,000 cycles per sector and a minimum data retention of 100 years.

FUNCTIONAL DIAGRAM



7007 ILL F01

PIN DESCRIPTIONS

Serial Output (SO)

SO is a push/pull serial data output pin. During a read cycle, data is shifted out on this pin. Data is clocked out by the falling edge of the serial clock.

Serial Input (SI)

SI is a serial data input pin. All opcodes, byte addresses, and data to be programmed to the memory are input on this pin. Data is latched by the rising edge of the serial clock.

Serial Clock (SCK)

The Serial Clock controls the serial bus timing for data input and output. Opcodes, addresses, or data present on the SI pin are latched on the rising edge of the clock input, while data on the SO pin change after the falling edge of the clock input.

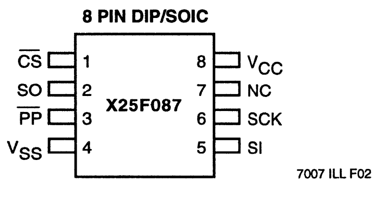
Chip Select ($\overline{CS}$)

When $\overline{CS}$ is HIGH, the X25F087 is deselected and the SO output pin is at high impedance and unless a nonvolatile write cycle is underway, the X25F087 will be in the standby power mode. $\overline{CS}$ LOW enables the X25F087, placing it in the active power mode. It should be noted that after power-up, a HIGH to LOW transition on $\overline{CS}$ is required prior to the start of any operation.

Program Protect ($\overline{PP}$)

When $\overline{PP}$ is LOW, nonvolatile writes to the X25F087 are disabled, but the part otherwise functions normally. When $\overline{PP}$ is held HIGH, all functions, including nonvolatile writes, operate normally. $\overline{PP}$ going LOW while $\overline{CS}$ is still LOW will interrupt a programming cycle to the X25F087. If the nonvolatile write cycle has already been initiated, $\overline{PP}$ going low will have no affect on this cycle.

PIN CONFIGURATION



PIN NAMES

Symbol	Description
$\overline{CS}$	Chip Select Input
SO	Serial Output
SI	Serial Input
SCK	Serial Clock Input
$\overline{PP}$	Program Protect Input
V_{SS}	Ground
V_{CC}	Supply Voltage
NC	No Connect

7007 FRM T01

PRINCIPLES OF OPERATION

The X25F087 is a 1024 x 8 SerialFlash designed to interface directly with the synchronous Serial Peripheral Interface (SPI) of many popular microcontroller families.

The X25F087 contains an 8-bit instruction register. It is accessed via the SI input, with data being clocked in on the rising edge of SCK. $\overline{CS}$ must be LOW and the $\overline{PP}$ input must be HIGH during the entire operation. Table 1 contains a list of the instructions and their opcodes. All instructions, addresses and data are transferred MSB first.

Data input is sampled on the first rising edge of SCK after $\overline{CS}$ goes LOW. SCK is static, allowing the user to stop the clock and then start it again to resume operations where left off.

Program Enable Latch

The X25F087 contains a "Program Enable" latch. This latch must be SET before a program operation is initiated. The PREN instruction will set the latch and the PRDI instruction will reset the latch (Figure 4). This latch is automatically reset upon a power-up condition and after the completion of a sector program cycle.

Block Lock Protection

There are eight Block Lock Protection options. The predefined blocks and associated address ranges are protected by programming the appropriate two byte Program Status instruction to the device (Table 1 and Figure 6). Once a Block Lock protect instruction has been completed, that Block Lock Protection setup is held in a nonvolatile Status Register (Figure 1) until the next Program Status instruction is issued. The sections of the memory array that are Block Lock protected can be read but not programmed until Block Lock Protection is removed or changed.

Figure 1. Status Register/Block Lock Protection Byte

7	6	5	4	3	2	1	0
0	0	0	0	0	BL2	BL1	BL0

Note: Bits [7:3] specified to be "0's"

7007 FRM T02

Read Sequence

When reading from the SerialFlash memory array, $\overline{\text{CS}}$ is first pulled LOW to select the device. The 8-bit READ instruction is transmitted to the X25F087, followed by the 16-bit address, of which the last 10 bits are used (bits [15:10] specified to be "0's"). After the READ opcode and address are sent, the data stored in the memory at the selected address is shifted out on the SO line. The data stored in memory at the next address can be read sequentially by continuing to provide clock pulses. The address is automatically incremented to the next higher address after each byte of data is shifted out. When the highest address is reached (03FFh), the address counter rolls over to address 0000h, allowing the read cycle to be continued indefinitely. The read operation is terminated by taking $\overline{\text{CS}}$ HIGH (Figure 2).

Sector Program Sequence

Prior to any attempt to program data into the X25F087, the "Program Enable" latch must first be set by issuing the PREN instruction (Table 1 and Figure 4). $\overline{\text{CS}}$ is first taken LOW. Then the PREN instruction is clocked into the X25F087. After all eight bits of the instruction are transmitted, $\overline{\text{CS}}$ must then be taken HIGH. If the user continues the program operation without taking $\overline{\text{CS}}$ HIGH after issuing the PREN instruction, the program operation will be ignored.

To program data to the SerialFlash memory array, the user then issues the PROGRAM instruction, followed by the 16 bit address of the first location in the sector and then the 16 bytes of data to be programmed. Only the last 9 bits of the address are used and bits [15:9] are specified to be "0's". The entire write operation takes 152 clocks. $\overline{\text{CS}}$ must go LOW and remain LOW for the duration of the operation. The host must program 16 bytes in each write with the restriction that these bytes reside on one sector. If the address counter reaches the end of the sector and the clock continues, or if fewer than 16 bytes are clocked in, the contents of the sector cannot be guaranteed.

For a sector program operation to be completed, $\overline{\text{CS}}$ can only be brought HIGH after bit 0 of the last data byte to be programmed is clocked in. If it is brought HIGH at any other time, the program operation will not be completed. (Figure 5)

Read Status Operation

If there is not a nonvolatile write in progress, the Read Status instruction returns the Block Lock Protection byte from the Status Register which contains the Block Lock Protection bits BL2-BL0 (Figure 1). The Block Lock Protection bits define the Block Lock Protection condition (Figure 1 and Table1). The other bits are reserved and will return "0's" when read (Figure 3).

If a nonvolatile write is in progress, the Read Status instruction returns the status of the internal write operation on SO. When the nonvolatile write cycle is completed, the status register data is again read out.

During a nonvolatile write in progress, the SO pin will be set HIGH. At the end of the nonvolatile write cycle, SO is set to output the current bit from the status register. Clocking SCK is valid during a nonvolatile write in progress, but is not necessary. If the SCK line is clocked, the pointer to the status register is also clocked, even though the SO pin shows the status of the nonvolatile write operation (Figure 3). When the pointer reaches the end of the eight bit status register, it "rolls over" to the first bit of the register.

Program Status Operation

Prior to any attempt to perform a Program Status Operation, the PREN instruction must first be issued. This instruction sets the "Program Enable" latch and allows the part to respond to a Program Status sequence (Figure 6). The Program Status instruction follows and consists of one command byte followed by one Block Lock Protection byte (Figure 1). This byte contains the Block Lock Protection bits BL2-BL0. The rest of the bits [7:3] are unused and must be programmed as "0's". Bringing $\overline{\text{CS}}$ HIGH after the two byte Program Status instruction initiates a nonvolatile write to the Status Register. Programming more than one byte to the Status Register will overwrite the previously programmed Block Lock Protection byte (Table 1).

Data Protection

The following circuitry has been included to prevent inadvertant programming of data:

- The "Program Enable" latch is reset upon power-up.
- A PREN instruction must be issued to set the "Program Enable" latch.
- $\overline{CS}$ must come HIGH at the proper clock count in order to start a program cycle.

Operational Notes

The X25F087 powers up in the following state:

- The device is in the low power, standby state.
- A HIGH to LOW transition on $\overline{CS}$ is required to enter an active state and receive an instruction.
- SO pin is at high impedance.
- The "Program Enable" latch is reset.

Table 1. Instruction Set and Block Lock Protection Byte Definition

Instruction Format*	Instruction Name and Operation
0000 0110	PREN: Set the Program Enable Latch (Program Enable Operation)
0000 0100	PRDI: Reset the Program Enable Latch (Program Disable Operation)
0000 0001	**PROGRAM STATUS Instruction - followed by:** **Block Lock Protection Byte: (Figure 1)** 0000 0000 --->NO PROTECT: --------------------------------->None of the Array 0000 0001 --->PROTECT Q1: --- 0000h - 00FFh ---------->Lower Quadrant (Q1) 0000 0010 --->PROTECT Q2: --- 0100h - 01FFh----------->Q2 0000 0011 --->PROTECT Q3: --- 0200h - 02FFh----------->Q3 0000 0100 --->PROTECT Q4: --- 0300h - 03FFh----------->Upper Quadrant (Q4) 0000 0101 --->PROTECT H1: --- 0000h - 01FFh----------->Lower Half of the Array (H1) 0000 0110 --->PROTECT S0: --- 0000h - 000Fh----------->Lower Sector (S0) 0000 0111 --->PROTECT Sn: --- 03F0h - 03FFh----------->Upper Sector (Sn)
0000 0101	READ STATUS: Reads Block Lock Protection & nonvolatile write in progress status on SO Pin
0000 0010	PROGRAM: Program operation followed by address and data
0000 0011	READ: Read operation followed by address

7007 FRM T03

*Instructions are shown with MSB in leftmost position. Instructions are transferred MSB first.

Figure 2. Read Operation Sequence

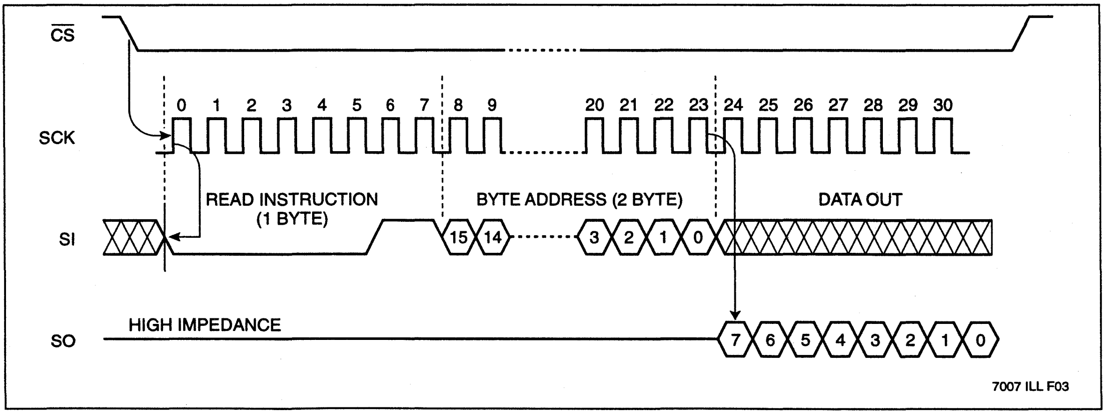
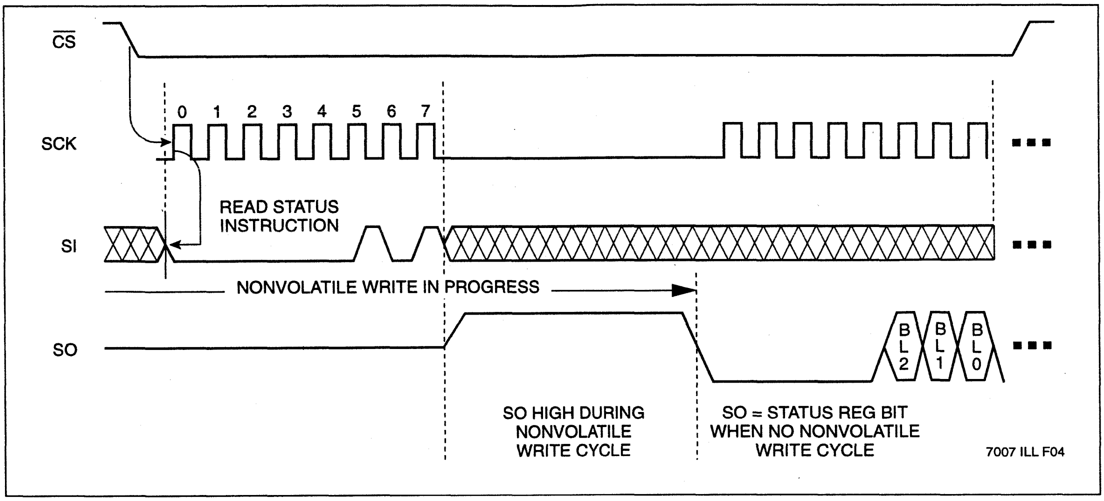
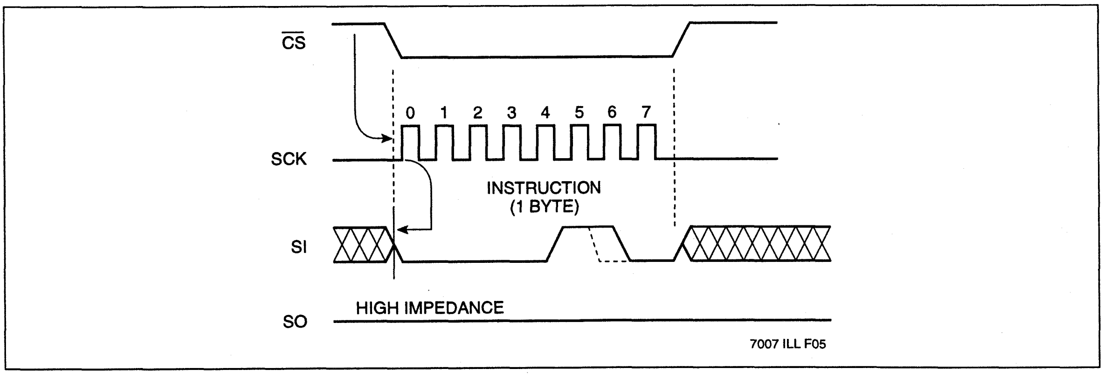


Figure 3. Read Status Operation Sequence



7007 ILL F04

Figure 4. Program Enable/Program Disable Sequence



7007 ILL F05

Figure 5. Sector Program Operation Sequence

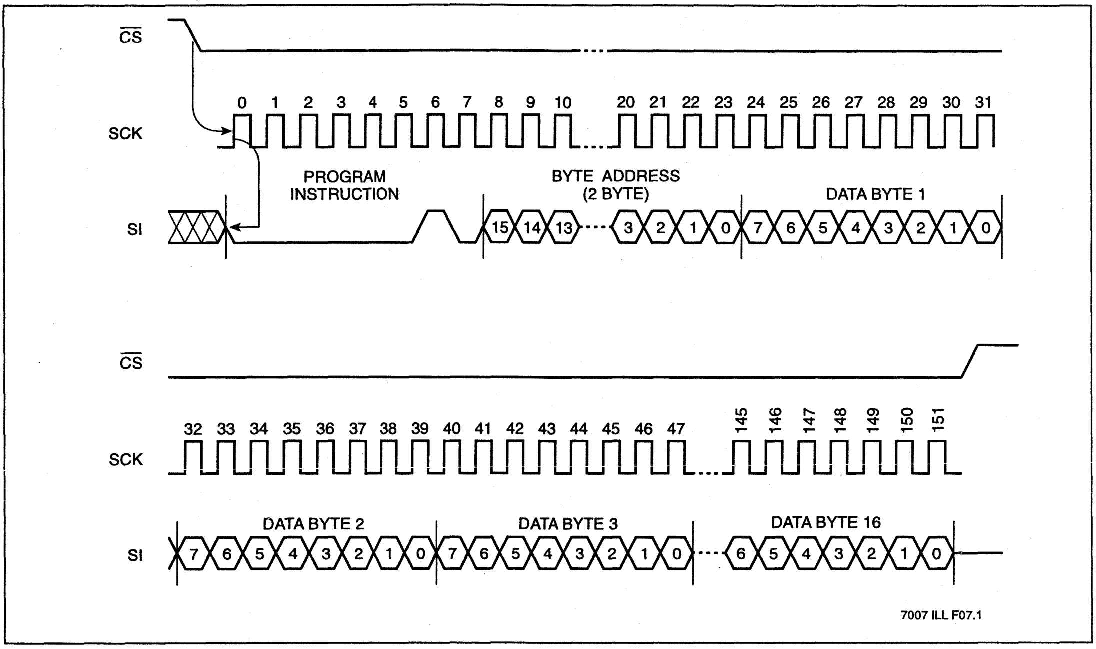
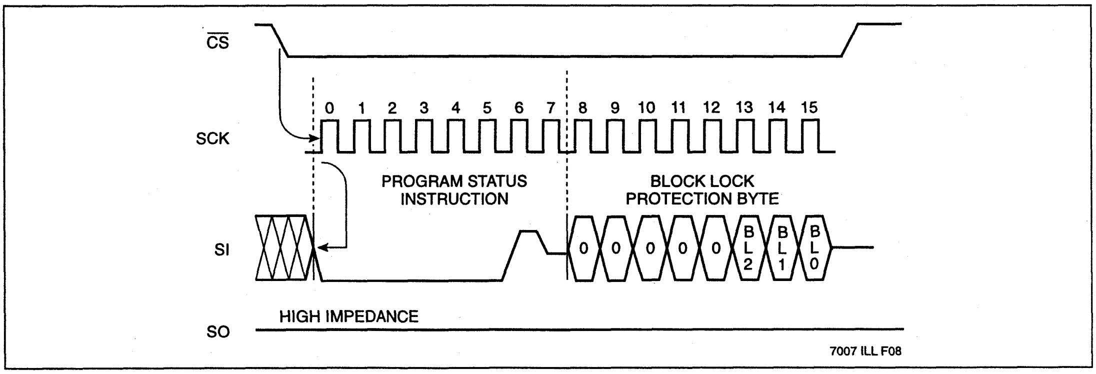


Figure 6. Program Status Operation Sequence



ABSOLUTE MAXIMUM RATINGS*

Temperature under Bias	–65°C to +135°C
Storage Temperature	–65°C to +150°C
Voltage on any Pin with Respect to V_{SS}	–1V to +7V
D.C. Output Current	5mA
Lead Temperature (Soldering, 10 seconds)	300°C

*COMMENT

Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and the functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

RECOMMENDED OPERATING CONDITIONS

Temperature	Min.	Max.
Commercial	0°C	+70°C
Industrial	–40°C	+85°C

7007 FRM T04

Supply Voltage	Limits
X25F087	1.8V to 3.6V
X25F087-5	4.5V to 5.5V

7007 FRM T05

D.C. OPERATING CHARACTERISTICS (Over the recommended operating conditions, unless otherwise specified.)

		Limits			
Symbol	Parameter	Min.	Max.	Units	Test Conditions
I_{CC1}	V_{CC} Read Current (Active)		1	mA	SCK = V_{CC} x 0.1/V_{CC} x 0.9 @ 1MHz, SO = Open, $\overline{CS}$ = V_{SS}
I_{CC2}	V_{CC} Write Current (Active)		3	mA	SCK = V_{CC} x 0.1/V_{CC} x 0.9 @ 1MHz, SO = Open, $\overline{CS}$ = V_{SS}
I_{SB}	V_{CC} Supply Current (Standby)		1	µA	$\overline{CS}$ = V_{CC} - 0.1, V_{IN} = V_{SS} or V_{CC}
I_{LI}	Input Leakage Current		10	µA	V_{IN} = V_{SS} to V_{CC}
I_{LO}	Output Leakage Current		10	µA	V_{OUT} = V_{SS} to V_{CC}
V_{IL}(1)	Input LOW Voltage	–0.5	V_{CC} x 0.3	V	
V_{IH}(1)	Input HIGH Voltage	V_{CC} x 0.7	V_{CC} + 0.5	V	
V_{OL1}	Output LOW Voltage		0.4	V	V_{CC} = 5.5V, I_{OL} = 2.1mA
V_{OH1}	Output HIGH Voltage	V_{CC} – 0.8		V	V_{CC} = 5.5V, I_{OH} = –1.0mA
V_{OL2}	Output LOW Voltage		0.4	V	V_{CC} = 3.6V, I_{OL} = 1.0mA
V_{OH2}	Output HIGH Voltage	V_{CC} – 0.4		V	V_{CC} = 3.6V, I_{OH} = –0.4mA
V_{OL3}	Output LOW Voltage		0.4	V	V_{CC} = 1.8V, I_{OL} = 0.5mA
V_{OH3}	Output HIGH Voltage	V_{CC} – 0.2		V	V_{CC} = 1.8V, I_{OH} = –0.25mA

7007 FRM T06

POWER-UP TIMING

Symbol	Parameter	Min.	Max.	Units
t_{PUR}(3)	Power-up to Read Operation		1	ms
t_{PUW}(3)	Power-up to Write Operation		5	ms

7007 FRM T07

CAPACITANCE $T_A = +25°C$, f = 1MHz, V_{CC} =5V.

Symbol	Parameter	Max.	Units	Conditions
C_{OUT}(2)	Output Capacitance (SO)	8	pF	$V_{OUT} = 0V$
C_{IN}(2)	Input Capacitance (SCK, SI, $\overline{CS}$, $\overline{PP}$)	6	pF	$V_{IN} = 0V$

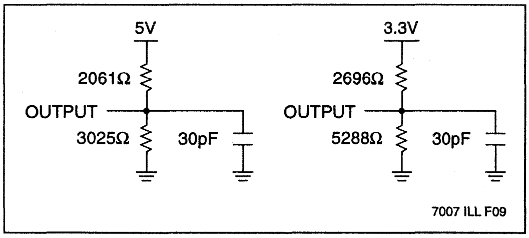
7007 FRM T08Not

Notes: (1) V_{IL} Min. and V_{IH} Max. are for reference only and are not 100% tested.

(2) This parameter is periodically sampled and not 100% tested.

(3) t_{PUR} and t_{PUW} are the delays required from the time V_{CC} is stable until the specified operation can be initiated. These parameters are periodically sampled and not 100% tested.

EQUIVALENT A.C. LOAD CIRCUIT



A.C. TEST CONDITIONS

Input Pulse Levels	V_{CC} x 0.1 to V_{CC} x 0.9
Input Rise and Fall Times	10ns
Input and Output Timing Level	V_{CC} X 0.5

7007 FRM T09

A.C. CHARACTERISTICS (Over the recommended operating conditions, unless otherwise specified.)

Data Input Timing

Symbol	Parameter	Min.	Max.	Units
f_{SCK}	Clock Frequency	0	1	MHz
t_{CYC}	Cycle Time	1000		ns
t_{LEAD}	$\overline{CS}$ Lead Time	500		ns
t_{LAG}	$\overline{CS}$ Lag Time	500		ns
t_{WH}	Clock HIGH Time	400		ns
t_{WL}	Clock LOW Time	400		ns
t_{SU}	Data Setup Time	100		ns
t_H	Data Hold Time	100		ns
t_{RI}(4)	Data In Rise Time		2	µs
t_{FI}(4)	Data In Fall Time		2	µs
t_{CS}	$\overline{CS}$ Deselect Time	2.0		µs
t_{WC}(5)	Write Cycle Time		10	ms

7007 FRM T10

Data Output Timing

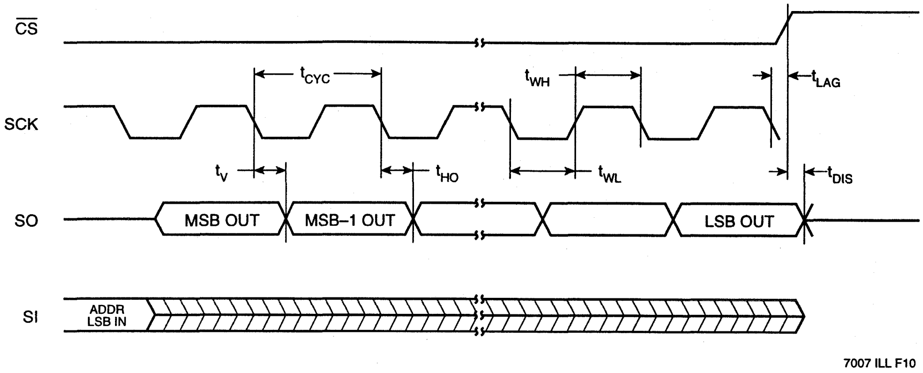
Symbol	Parameter	Min.	Max.	Units
f_{SCK}	Clock Frequency	0	1	MHz
t_{DIS}	Output Disable Time		500	ns
t_V	Output Valid from Clock LOW		400	ns
t_{HO}	Output Hold Time	0		ns
t_{RO}(4)	Output Rise Time		300	ns
t_{FO}(4)	Output Fall Time		300	ns

7007 FRM T11

Notes: (4) This parameter is periodically sampled and not 100% tested.

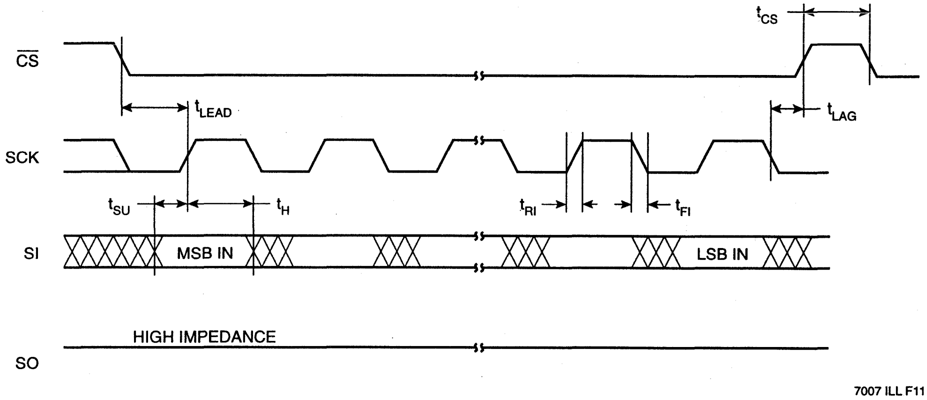
(5) t_{WC} is the time from the rising edge of $\overline{CS}$ after a valid program sequence has been sent to the end of the self-timed internal nonvolatile write cycle.

Figure 7. Serial Output Timing



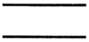
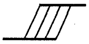
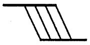
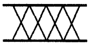
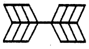
7007 ILL F10

Figure 8. Serial Input Timing



7007 ILL F11

SYMBOL TABLE

WAVEFORM	INPUTS	OUTPUTS
	Must be steady	Will be steady
	May change from LOW to HIGH	Will change from LOW to HIGH
	May change from HIGH to LOW	Will change from HIGH to LOW
	Don't Care: Changes Allowed	Changing: State Not Known
	N/A	Center Line is High Impedance

ORDERING INFORMATION

X25F087 P T -X

Device — **X25F087**

V_{CC} Range (-X)
Blank = 1.8V to 3.6V
5 = 4.5V to 5.5V

Temperature Range (T)
Blank = Commercial = 0°C to +70°C
I = Industrial = –40°C to +85°C

Package (P)
P = 8-Lead Plastic DIP
S = 8-Lead SOIC

Part Mark Convention

X25F087 X — P = 8-Lead Plastic DIP
Blank = 8-Lead SOIC

X — Blank = 1.8V to 3.6V, 0°C to +70°C
I = 1.8V to 3.6V, –40°C to +85°C
5 = 4.5V to 5.5V, 0°C to +70°C
I5 = 4.5V to 5.5V, –40°C to +85°C

LIMITED WARRANTY

Devices sold by Xicor, Inc. are covered by the warranty and patent indemnification provisions appearing in its Terms of Sale only. Xicor, Inc. makes no warranty, express, statutory, implied, or by description regarding the information set forth herein or regarding the freedom of the described devices from patent infringement. Xicor, Inc. makes no warranty of merchantability or fitness for any purpose. Xicor, Inc. reserves the right to discontinue production and change specifications and prices at any time and without notice.

Xicor, Inc. assumes no responsibility for the use of any circuitry other than circuitry embodied in a Xicor, Inc. product. No other circuits, patents, licenses are implied.

U.S. PATENTS

Xicor products are covered by one or more of the following U.S. Patents: 4,263,664; 4,274,012; 4,300,212; 4,314,265; 4,326,134; 4,393,481; 4,404,475; 4,450,402; 4,486,769; 4,488,060; 4,520,461; 4,533,846; 4,599,706; 4,617,652; 4,668,932; 4,752,912; 4,829, 482; 4,874, 967; 4,883, 976. Foreign patents and additional patents pending.

LIFE RELATED POLICY

In situations where semiconductor component failure may endanger life, system designers using this product should design the system with appropriate error detection and correction, redundancy and back-up features to prevent such an occurence.

Xicor's products are not authorized for use in critical components in life support devices or systems.

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform, when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.
2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

APPLICATION NOTE
AVAILABLE
AN75

Xicor®

Advance Information

4K X25F047 512 x 8 Bit

SPI SerialFlash with Block Lock™ Protection

FEATURES

- **1MHz Clock Rate**
- **SPI Modes (0,0 & 1,1)**
- **512 x 8 Bits**
  - **—16 Byte Small Sector Program Mode**
- **Low Power CMOS**
  - **—<1µA Standby Current**
  - **—<3mA Active Current during Program**
  - **—<400µA Active Current during Read**
- **1.8V to 3.6V or 5V "Univolt" Read and Program Power Supply Versions**
- **Block Lock Protection**
  - **—Block Lock Protect 0, any 1/4, 1st 1/2, First or Last Sector of SerialFlash Array**
- **Built-in Inadvertent Program Protection**
  - **—Power-Up/Power-Down Protection Circuitry**
  - **—Program Enable Latch**
  - **—Program Protect Pin**
- **Self-Timed Program Cycle**
  - **—5ms Program Cycle Time (Typical)**
- **High Reliability**
  - **—Endurance: 100,000 Cycles/Byte**
  - **—Data Retention: 100 Years**
  - **—ESD: 2000V on all pins**
- **8-Pin Mini-DIP Package**
- **8-Lead SOIC Package**

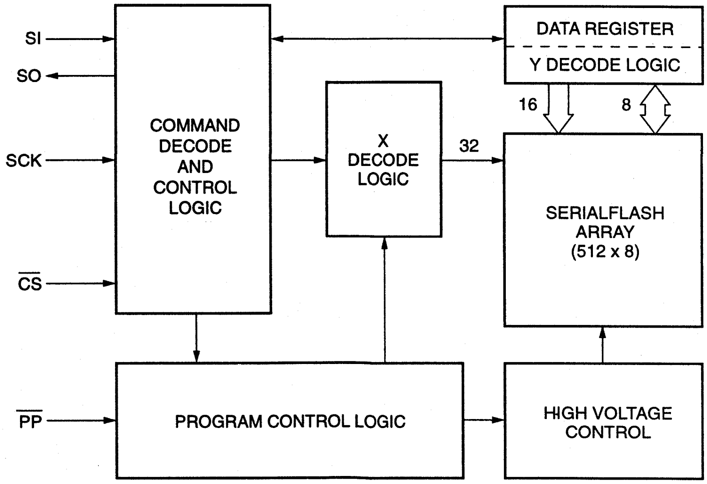
DESCRIPTION

The X25F047 is a CMOS 4K-bit SerialFlash, internally organized as 512 x 8. The X25F047 features a Serial Peripheral Interface (SPI) and software protocol allowing operation on a simple four-wire bus. The bus signals are a clock input (SCK) plus separate data in (SI) and data out (SO) lines. Access to the device is controlled through a chip select ($\overline{CS}$) input, allowing any number of devices to share the same bus.

There are eight options for programmable, nonvolatile, Block Lock Protection available to the end user. These options are implemented via special instructions programmed to the part. The X25F047 also features a $\overline{PP}$ pin that can be used for hardwire protection of the part, disabling all programming attempts, as well as a Program Enable Latch that must be set before a program operation can be initiated.

The X25F047 utilizes Xicor's proprietary Direct Write™ cell, providing a minimum endurance of 100,000 cycles per sector and a minimum data retention of 100 years.

FUNCTIONAL DIAGRAM



7005 ILL F01.3

Characteristics subject to change without notice

PIN DESCRIPTIONS

Serial Output (SO)

SO is a push/pull serial data output pin. During a read cycle, data is shifted out on this pin. Data is clocked out by the falling edge of the serial clock.

Serial Input (SI)

SI is a serial data input pin. All opcodes, byte addresses, and data to be programmed to the memory are input on this pin. Data is latched by the rising edge of the serial clock.

Serial Clock (SCK)

The Serial Clock controls the serial bus timing for data input and output. Opcodes, addresses, or data present on the SI pin are latched on the rising edge of the clock input, while data on the SO pin change after the falling edge of the clock input.

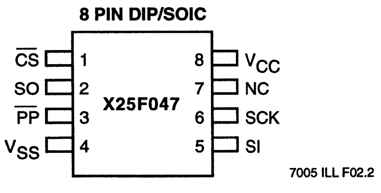
Chip Select ($\overline{\text{CS}}$)

When $\overline{\text{CS}}$ is HIGH, the X25F047 is deselected and the SO output pin is at high impedance and unless a nonvolatile write cycle is underway, the X25F047 will be in the standby power mode. $\overline{\text{CS}}$ LOW enables the X25F047, placing it in the active power mode. It should be noted that after power-up, a HIGH to LOW transition on $\overline{\text{CS}}$ is required prior to the start of any operation.

Program Protect ($\overline{\text{PP}}$)

When $\overline{\text{PP}}$ is LOW, nonvolatile writes to the X25F047 are disabled, but the part otherwise functions normally. When $\overline{\text{PP}}$ is held HIGH, all functions, including nonvolatile writes, operate normally. $\overline{\text{PP}}$ going LOW while $\overline{\text{CS}}$ is still LOW will interrupt a programming cycle to the X25F047. If the nonvolatile write cycle has already been initiated, $\overline{\text{PP}}$ going low will have no affect on this cycle.

PIN CONFIGURATION



PIN NAMES

Symbol	Description
$\overline{\text{CS}}$	Chip Select Input
SO	Serial Output
SI	Serial Input
SCK	Serial Clock Input
$\overline{\text{PP}}$	Program Protect Input
V_{SS}	Ground
V_{CC}	Supply Voltage
NC	No Connect

7005 FRM T01

PRINCIPLES OF OPERATION

The X25F047 is a 512 x 8 SerialFlash designed to interface directly with the synchronous Serial Peripheral Interface (SPI) of many popular microcontroller families.

The X25F047 contains an 8-bit instruction register. It is accessed via the SI input, with data being clocked in on the rising edge of SCK. $\overline{\text{CS}}$ must be LOW and the $\overline{\text{PP}}$ input must be HIGH during the entire operation. Table 1 contains a list of the instructions and their opcodes. All instructions, addresses and data are transferred MSB first.

Data input is sampled on the first rising edge of SCK after $\overline{\text{CS}}$ goes LOW. SCK is static, allowing the user to stop the clock and then start it again to resume operations where left off.

Program Enable Latch

The X25F047 contains a “Program Enable” latch. This latch must be SET before a program operation is initiated. The PREN instruction will set the latch and the PRDI instruction will reset the latch (Figure 4). This latch is automatically reset upon a power-up condition and after the completion of a sector program cycle.

Block Lock Protection

There are eight Block Lock Protection options. The predefined blocks and associated address ranges are protected by programming the appropriate two byte Program Status instruction to the device (Table 1 and Figure 6). Once a Block Lock protect instruction has been completed, that Block Lock Protection setup is held in a nonvolatile Status Register (Figure 1) until the next Program Status instruction is issued. The sections of the memory array that are Block Lock protected can be read but not programmed until Block Lock Protection is removed or changed.

Figure 1. Status Register/Block Lock Protection Byte

7	6	5	4	3	2	1	0
0	0	0	0	0	BL2	BL1	BL0

Note: Bits [7:3] specified to be "0's"

7005 FRM T02.1

Read Sequence

When reading from the SerialFlash memory array, $\overline{CS}$ is first pulled LOW to select the device. The 8-bit READ instruction is transmitted to the X25F047, followed by the 16-bit address, of which the last 9 bits are used (bits [15:9] specified to be "0's"). After the READ opcode and address are sent, the data stored in the memory at the selected address is shifted out on the SO line. The data stored in memory at the next address can be read sequentially by continuing to provide clock pulses. The address is automatically incremented to the next higher address after each byte of data is shifted out. When the highest address is reached (01FFh), the address counter rolls over to address 0000h, allowing the read cycle to be continued indefinitely. The read operation is terminated by taking $\overline{CS}$ HIGH (Figure 2).

Sector Program Sequence

Prior to any attempt to program data into the X25F047, the "Program Enable" latch must first be set by issuing the PREN instruction (Table 1 and Figure 4). $\overline{CS}$ is first taken LOW. Then the PREN instruction is clocked into the X25F047. After all eight bits of the instruction are transmitted, $\overline{CS}$ must then be taken HIGH. If the user continues the program operation without taking $\overline{CS}$ HIGH after issuing the PREN instruction, the program operation will be ignored.

To program data to the SerialFlash memory array, the user then issues the PROGRAM instruction, followed by the 16 bit address of the first location in the sector and then the 16 bytes of data to be programmed. Only the last 9 bits of the address are used and bits [15:9] are specified to be "0's". The entire write operation takes 152 clocks. $\overline{CS}$ must go LOW and remain LOW for the duration of the operation. The host must program 16 bytes in each write with the restriction that these bytes reside on one sector. If the address counter reaches the end of the sector and the clock continues, or if fewer than 16 bytes are clocked in, the contents of the sector cannot be guaranteed.

For a sector program operation to be completed, $\overline{CS}$ can only be brought HIGH after bit 0 of the last data byte to be programmed is clocked in. If it is brought HIGH at any other time, the program operation will not be completed. (Figure 5)

Read Status Operation

If there is not a nonvolatile write in progress, the Read Status instruction returns the Block Lock Protection byte from the Status Register which contains the Block Lock Protection bits BL2-BL0 (Figure 1). The Block Lock Protection bits define the Block Lock Protection condition (Figure 1 and Table1). The other bits are reserved and will return "0's" when read (Figure 3).

If a nonvolatile write is in progress, the Read Status instruction returns the status of the internal write operation on SO. When the nonvolatile write cycle is completed, the status register data is again read out.

During a nonvolatile write in progress, the SO pin will be set HIGH. At the end of the nonvolatile write cycle, SO is set to output the current bit from the status register. Clocking SCK is valid during a nonvolatile write in progress, but is not necessary. If the SCK line is clocked, the pointer to the status register is also clocked, even though the SO pin shows the status of the nonvolatile write operation (Figure 3). When the pointer reaches the end of the eight bit status register, it "rolls over" to the first bit of the register.

Program Status Operation

Prior to any attempt to perform a Program Status Operation, the PREN instruction must first be issued. This instruction sets the "Program Enable" latch and allows the part to respond to a Program Status sequence (Figure 6). The Program Status instruction follows and consists of one command byte followed by one Block Lock Protection byte (Figure 1). This byte contains the Block Lock Protection bits BL2-BL0. The rest of the bits [7:3] are unused and must be programmed as "0's". Bringing $\overline{CS}$ HIGH after the two byte Program Status instruction initiates a nonvolatile write to the Status Register. Programming more than one byte to the Status Register will overwrite the previously programmed Block Lock Protection byte (Table 1).

Data Protection

The following circuitry has been included to prevent inadvertant programming of data:

- The "Program Enable" latch is reset upon power-up.
- A PREN instruction must be issued to set the "Program Enable" latch.
- $\overline{CS}$ must come HIGH at the proper clock count in order to start a program cycle.

Operational Notes

The X25F047 powers up in the following state:

- The device is in the low power, standby state.
- A HIGH to LOW transition on $\overline{CS}$ is required to enter an active state and receive an instruction.
- SO pin is at high impedance.
- The "Program Enable" latch is reset.

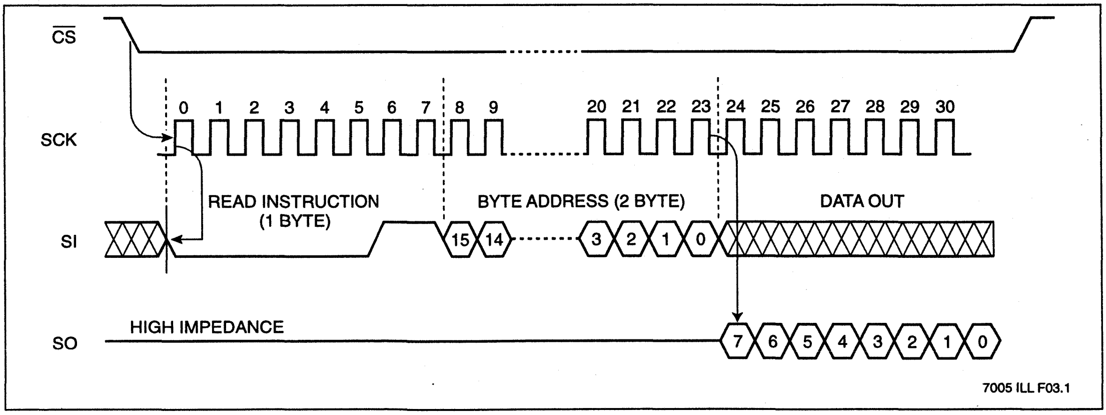
Table 1. Instruction Set and Block Lock Protection Byte Definition

Instruction Format*	Instruction Name and Operation
0000 0110	PREN: Set the Program Enable Latch (Program Enable Operation)
0000 0100	PRDI: Reset the Program Enable Latch (Program Disable Operation)
0000 0001	**PROGRAM STATUS Instruction - followed by:** **Block Lock Protection Byte: (Figure 1)** 0000 0000 --->NO PROTECT: ---------------------------------->None of the Array 0000 0001 --->PROTECT Q1: --- 0000h - 007Fh ---------->Lower Quadrant (Q1) 0000 0010 --->PROTECT Q2: --- 0080h - 00FFh----------->Q2 0000 0011 --->PROTECT Q3: --- 0100h - 017Fh----------->Q3 0000 0100 --->PROTECT Q4: --- 0180h - 01FFh----------->Upper Quadrant (Q4) 0000 0101 --->PROTECT H1: --- 0000h - 00FFh----------->Lower Half of the Array (H1) 0000 0110 --->PROTECT S0: --- 0000h - 000Fh----------->Lower Sector (S0) 0000 0111 --->PROTECT Sn: --- 01F0h - 01FFh----------->Upper Sector (Sn)
0000 0101	READ STATUS: Reads Block Lock Protection & nonvolatile write in progress status on SO Pin
0000 0010	PROGRAM: Program operation followed by address and data
0000 0011	READ: Read operation followed by address

7005 FRM T03.1

*Instructions are shown with MSB in leftmost position. Instructions are transferred MSB first.

Figure 2. Read Operation Sequence



7005 ILL F03.1

Figure 3. Read Status Operation Sequence

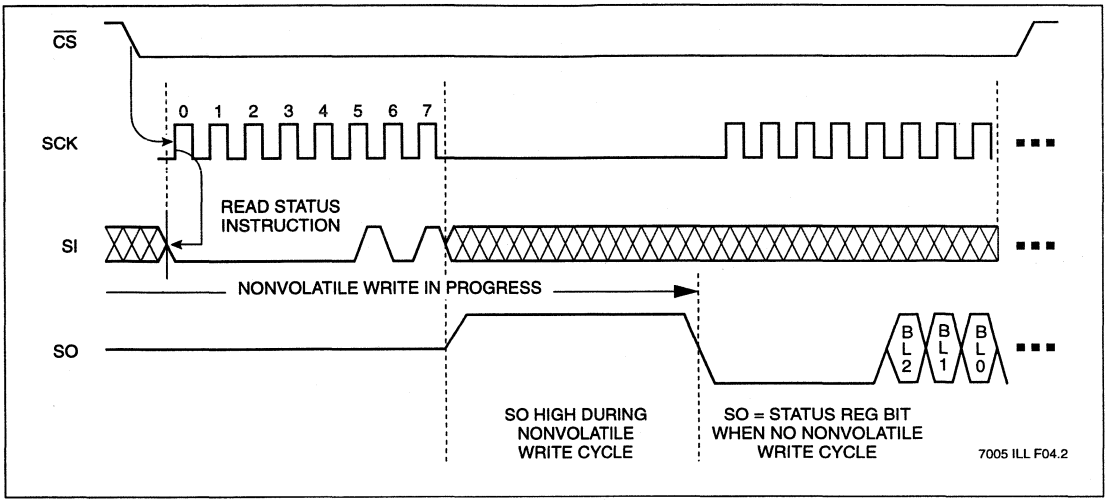


Figure 4. Program Enable/Program Disable Sequence

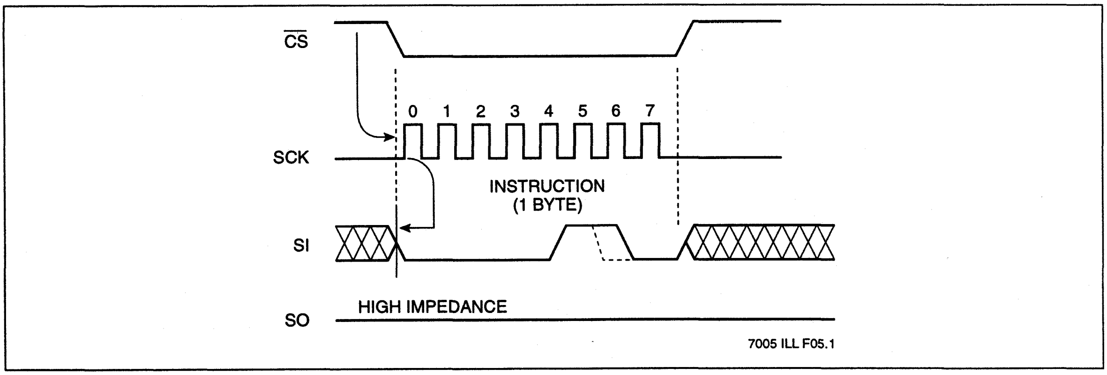


Figure 5. Sector Program Operation Sequence

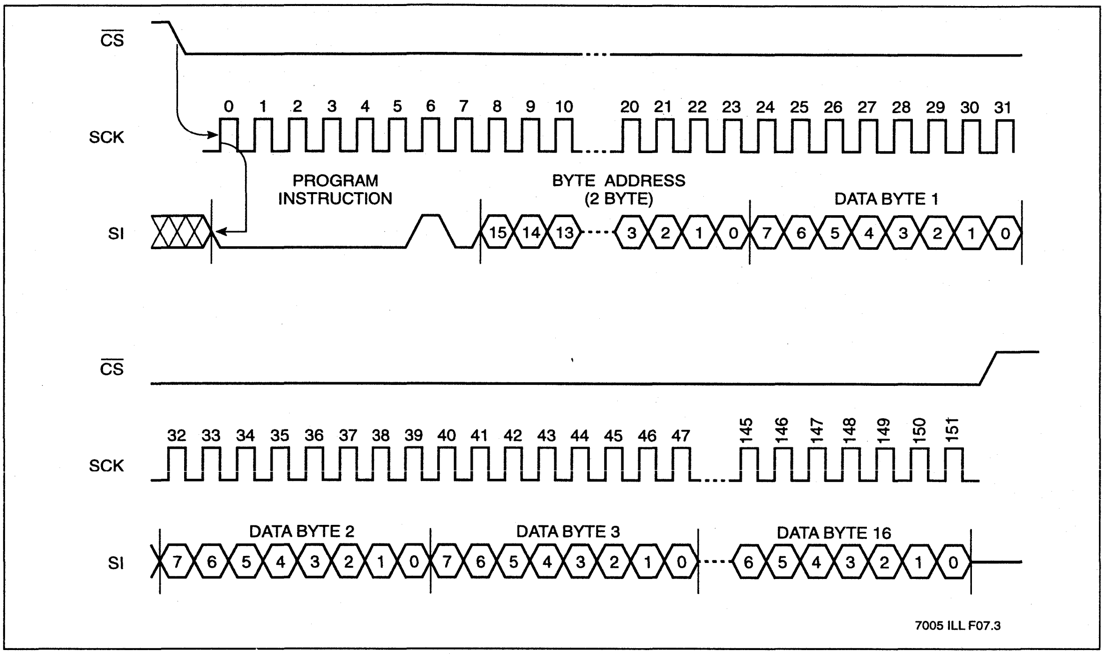
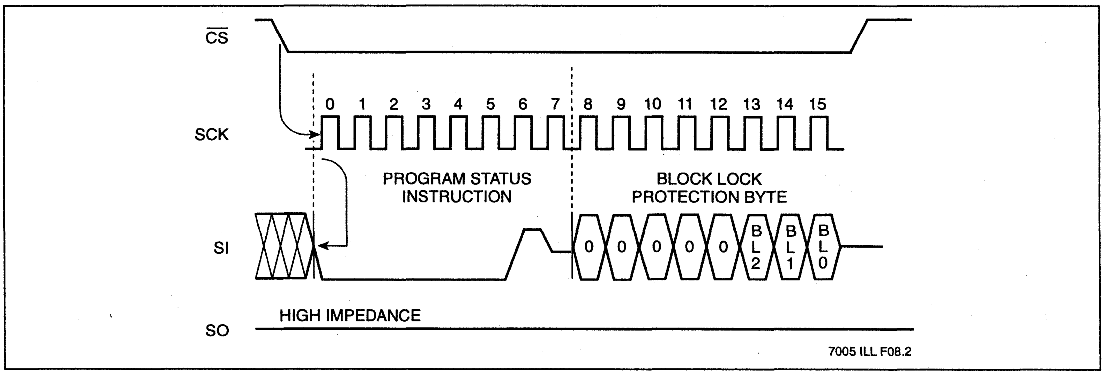


Figure 6. Program Status Operation Sequence



ABSOLUTE MAXIMUM RATINGS*

Temperature under Bias	–65°C to +135°C
Storage Temperature	–65°C to +150°C
Voltage on any Pin with Respect to V_{SS}	–1V to +7V
D.C. Output Current	5mA
Lead Temperature (Soldering, 10 seconds)	300°C

*COMMENT

Stresses above those listed under “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only and the functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

RECOMMENDED OPERATING CONDITIONS

Temperature	Min.	Max.
Commercial	0°C	+70°C
Industrial	–40°C	+85°C

7005 FRM T04

Supply Voltage	Limits
X25F047	1.8V to 3.6V
X25F047-5	4.5V to 5.5V

7005 FRM T05

D.C. OPERATING CHARACTERISTICS (Over the recommended operating conditions, unless otherwise specified.)

		Limits			
Symbol	Parameter	Min.	Max.	Units	Test Conditions
I_{CC1}	V_{CC} Read Current (Active)		1	mA	SCK = V_{CC} x 0.1/V_{CC} x 0.9 @ 1MHz, SO = Open, $\overline{CS}$ = V_{SS}
I_{CC2}	V_{CC} Write Current (Active)		3	mA	SCK = V_{CC} x 0.1/V_{CC} x 0.9 @ 1MHz, SO = Open, $\overline{CS}$ = V_{SS}
I_{SB}	V_{CC} Supply Current (Standby)		1	µA	$\overline{CS}$ = V_{CC} - 0.1, V_{IN} = V_{SS} or V_{CC}
I_{LI}	Input Leakage Current		10	µA	V_{IN} = V_{SS} to V_{CC}
I_{LO}	Output Leakage Current		10	µA	V_{OUT} = V_{SS} to V_{CC}
V_{IL}[(1)]	Input LOW Voltage	–0.5	V_{CC} x 0.3	V	
V_{IH}[(1)]	Input HIGH Voltage	V_{CC} x 0.7	V_{CC} + 0.5	V	
V_{OL1}	Output LOW Voltage		0.4	V	V_{CC} = 5.5V, I_{OL} = 2.1mA
V_{OH1}	Output HIGH Voltage	V_{CC} – 0.8		V	V_{CC} = 5.5V, I_{OH} = –1.0mA
V_{OL2}	Output LOW Voltage		0.4	V	V_{CC} = 3.6V, I_{OL} = 1.0mA
V_{OH2}	Output HIGH Voltage	V_{CC} – 0.4		V	V_{CC} = 3.6V, I_{OH} = –0.4mA
V_{OL3}	Output LOW Voltage		0.4	V	V_{CC} = 1.8V, I_{OL} = 0.5mA
V_{OH3}	Output HIGH Voltage	V_{CC} – 0.2		V	V_{CC} = 1.8V, I_{OH} = –0.25mA

7005 FRM T06

POWER-UP TIMING

Symbol	Parameter	Min.	Max.	Units
t_{PUR}(3)	Power-up to Read Operation		1	ms
t_{PUW}(3)	Power-up to Write Operation		5	ms

7005 FRM T07.1

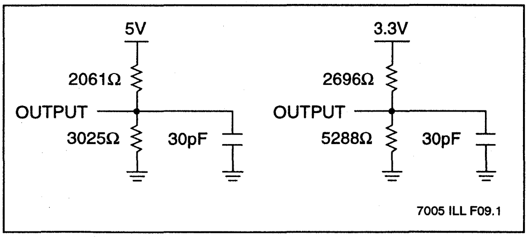
CAPACITANCE T_A = +25°C, f = 1MHz, V_{CC} =5V.

Symbol	Parameter	Max.	Units	Conditions
C_{OUT}(2)	Output Capacitance (SO)	8	pF	V_{OUT} = 0V
C_{IN}(2)	Input Capacitance (SCK, SI, $\overline{CS}$, $\overline{PP}$)	6	pF	V_{IN} = 0V

7005 FRM T08

Notes: (1) V_{IL} Min. and V_{IH} Max. are for reference only and are not 100% tested.
(2) This parameter is periodically sampled and not 100% tested.
(3) t_{PUR} and t_{PUW} are the delays required from the time V_{CC} is stable until the specified operation can be initiated. These parameters are periodically sampled and not 100% tested.

EQUIVALENT A.C. LOAD CIRCUIT



7005 ILL F09.1

A.C. TEST CONDITIONS

Input Pulse Levels	V_{CC} x 0.1 to V_{CC} x 0.9
Input Rise and Fall Times	10ns
Input and Output Timing Level	V_{CC} X 0.5

7005 FRM T09

A.C. CHARACTERISTICS (Over the recommended operating conditions, unless otherwise specified.)

Data Input Timing

Symbol	Parameter	Min.	Max.	Units
f_{SCK}	Clock Frequency	0	1	MHz
t_{CYC}	Cycle Time	1000		ns
t_{LEAD}	$\overline{CS}$ Lead Time	500		ns
t_{LAG}	$\overline{CS}$ Lag Time	500		ns
t_{WH}	Clock HIGH Time	400		ns
t_{WL}	Clock LOW Time	400		ns
t_{SU}	Data Setup Time	100		ns
t_H	Data Hold Time	100		ns
t_{RI}(4)	Data In Rise Time		2	µs
t_{FI}(4)	Data In Fall Time		2	µs
t_{CS}	$\overline{CS}$ Deselect Time	2.0		µs
t_{WC}(5)	Write Cycle Time		10	ms

7005 FRM T10

Data Output Timing

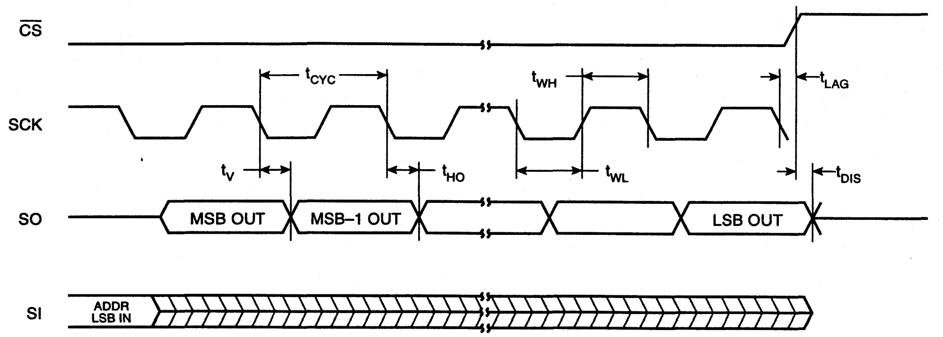
Symbol	Parameter	Min.	Max.	Units
f_{SCK}	Clock Frequency	0	1	MHz
t_{DIS}	Output Disable Time		500	ns
t_V	Output Valid from Clock LOW		400	ns
t_{HO}	Output Hold Time	0		ns
t_{RO}(4)	Output Rise Time		300	ns
t_{FO}(4)	Output Fall Time		300	ns

7005 FRM T11

Notes: (4) This parameter is periodically sampled and not 100% tested.

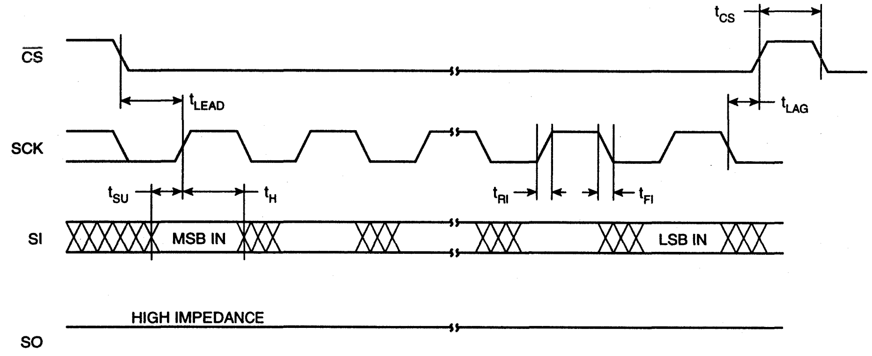
(5) t_{WC} is the time from the rising edge of $\overline{CS}$ after a valid program sequence has been sent to the end of the self-timed internal nonvolatile write cycle.

Figure 7. Serial Output Timing



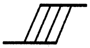
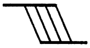
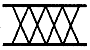
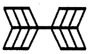
7005 ILL F10

Figure 8. Serial Input Timing



7005 ILL F11

SYMBOL TABLE

WAVEFORM	INPUTS	OUTPUTS
	Must be steady	Will be steady
	May change from LOW to HIGH	Will change from LOW to HIGH
	May change from HIGH to LOW	Will change from HIGH to LOW
	Don't Care: Changes Allowed	Changing: State Not Known
	N/A	Center Line is High Impedance

ORDERING INFORMATION

X25F047 P T -X

Device — X25F047

-X — V_{CC} **Range**
Blank = 1.8V to 3.6V
5 = 4.5V to 5.5V

T — **Temperature Range**
Blank = Commercial = 0°C to +70°C
I = Industrial = –40°C to +85°C

P — **Package**
P = 8-Lead Plastic DIP
S = 8-Lead SOIC

Part Mark Convention

X25F047 X — P = 8-Lead Plastic DIP
Blank = 8-Lead SOIC

X — Blank = 1.8V to 3.6V, 0°C to +70°C
I = 1.8V to 3.6V, –40°C to +85°C
5 = 4.5V to 5.5V, 0°C to +70°C
I5 = 4.5V to 5.5V, –40°C to +85°C

LIMITED WARRANTY

Devices sold by Xicor, Inc. are covered by the warranty and patent indemnification provisions appearing in its Terms of Sale only. Xicor, Inc. makes no warranty, express, statutory, implied, or by description regarding the information set forth herein or regarding the freedom of the described devices from patent infringement. Xicor, Inc. makes no warranty of merchantability or fitness for any purpose. Xicor, Inc. reserves the right to discontinue production and change specifications and prices at any time and without notice.

Xicor, Inc. assumes no responsibility for the use of any circuitry other than circuitry embodied in a Xicor, Inc. product. No other circuits, patents, licenses are implied.

U.S. PATENTS

Xicor products are covered by one or more of the following U.S. Patents: 4,263,664; 4,274,012; 4,300,212; 4,314,265; 4,326,134; 4,393,481; 4,404,475; 4,450,402; 4,486,769; 4,488,060; 4,520,461; 4,533,846; 4,599,706; 4,617,652; 4,668,932; 4,752,912; 4,829, 482; 4,874, 967; 4,883, 976. Foreign patents and additional patents pending.

LIFE RELATED POLICY

In situations where semiconductor component failure may endanger life, system designers using this product should design the system with appropriate error detection and correction, redundancy and back-up features to prevent such an occurence.

Xicor's products are not authorized for use in critical components in life support devices or systems.

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform, when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.
2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

Xicor®

APPLICATION NOTE
AVAILABLE
AN83 • Development Tools XK76C

Password Access Security Supervisor

Xicor®

4K X76F041 4 x 128 x 8 Bit

PASS™ SecureFlash

FEATURES

- 64-Bit Password Security
- Three Password Modes
  - —Secure Read Access
  - —Secure Write Access
  - —Secure Configuration Access
- Programmable Configuration
  - —Read, Write and Configuration Access Passwords
  - —Multiple Array Access/Functionality
  - —Retry Register/Counter
- 8 Byte Sector Write
- (4) 1K Memory Arrays
- ISO Response to Reset
- Low Power CMOS
  - —50µA Standby Current
  - —3mA Active Current
- 1.8V to 3.6V or 5V "Univolt" Read and Program Power Supply Versions
- High Reliability
  - —Endurance: 100,000 Cycles
  - —Data Retention: 100 Years
  - —ESD Protection: 2000V on All Pins

DESCRIPTION

The X76F041 is a password access security supervisor device, containing four 128 x 8 bit SecureFlash arrays. Access can be controlled by three 64-bit programmable passwords, one for read operations, one for write operations and one for device configuration.

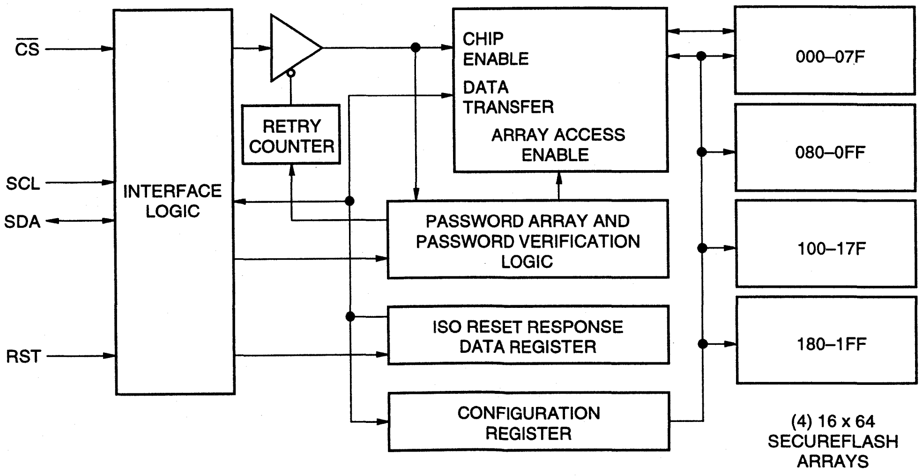
The X76F041 features a serial interface and software protocol allowing operation on a simple two wire bus. The bus signals are a clock input (SCL) and a bidirectional data input and output (SDA). Access to the device is controlled through a chip select input ($\overline{CS}$), allowing any number of devices to share the same bus.

The X76F041 also features a synchronous response to reset; providing an automatic output of a pre-configured 32-bit data stream conforming to the ISO standard for memory cards.

The X76F041 utilizes Xicor's proprietary Direct Write™ cell, providing a minimum endurance of 100,000 cycles per sector and a minimum data retention of 100 years.

3

FUNCTIONAL DIAGRAM



7002 ILL F01

PIN DESCRIPTION

Serial Data Input/Output (SDA)

SDA is a true three state serial data input/output pin. During a read cycle, data is shifted out on this pin. During a write cycle, data is shifted in on this pin. In all other cases this pin is in a high impedance state.

Serial Clock (SCL)

The Serial Clock controls the serial bus timing for data input and output.

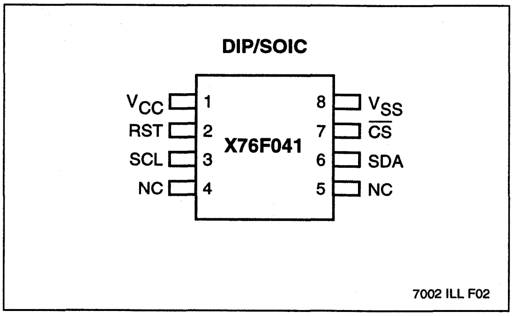
Chip Select ($\overline{CS}$)

When $\overline{CS}$ is HIGH, the X76F041 is deselected and the SDA pin is at high impedance and unless an internal write operation is underway the X76F041 will be in the standby power mode. $\overline{CS}$ LOW enables the X76F041, placing it in the active power mode.

Reset (RST)

RST is a device reset pin. When RST is pulsed HIGH while $\overline{CS}$ is LOW the X76F041 will output 32 bits of fixed data which conforms to the ISO standard for "synchronous response to reset". $\overline{CS}$ must remain LOW and the part must not be in a write cycle for the response to reset to occur. If at any time during the response to reset $\overline{CS}$ goes HIGH, the response to reset will be aborted and the part will return to the standby mode.

PIN CONFIGURATION



7002 ILL F02

Symbol	Description
$\overline{CS}$	Chip Select Input
SDA	Serial Data Input/Output
RST	Reset Input
SCL	Serial Clock Input
V_{SS}	Ground
V_{CC}	Supply Voltage
NC	No Connect

7002 FRM T01

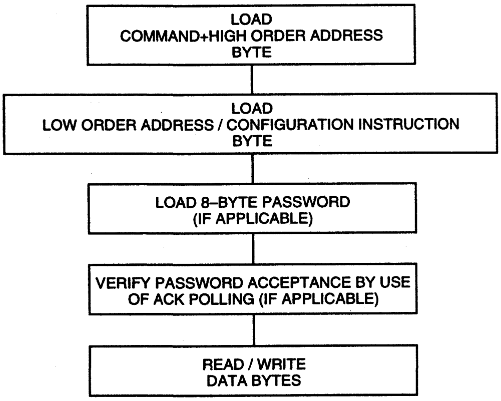
DEVICE OPERATION

There are three primary modes of operation for the X76F041; READ, WRITE and CONFIGURATION. The READ and WRITE modes may be performed with or without an 8-byte password. The CONFIGURATION mode always requires an 8-byte password.

The basic method of communication is established by first enabling the device ($\overline{CS}$ LOW), generating a start condition and then transmitting a command and address field followed by the correct password (if configured to require a password). All parts will be shipped from the factory in the non-password mode. The user must perform an ACK Polling routine to determine the validity of the password and start the data transfer (see Acknowledge Polling). Only after the correct password is accepted and an ACK Polling has been performed can the data transfer occur.

To ensure correct communication, RST must remain LOW under all conditions except when initiating a "Response to Reset sequence".

Figure 1. X76F041 Device Operation


LOAD
COMMAND+HIGH ORDER ADDRESS
BYTE
LOAD
LOW ORDER ADDRESS / CONFIGURATION INSTRUCTION
BYTE
LOAD 8–BYTE PASSWORD
(IF APPLICABLE)
VERIFY PASSWORD ACCEPTANCE BY USE
OF ACK POLLING (IF APPLICABLE)
READ / WRITE
DATA BYTES

7002 ILL F03

Data is transferred in 8-bit segments, with each transfer being followed by an ACK, generated by the receiving device.

If the X76F041 is in a nonvolatile write cycle a "no ACK" (SDA HIGH) response will be issued in response to loading of the command + high order address byte. If a stop condition is issued prior to the nonvolatile write cycle the write operation will be terminated and the part will reset and enter into a standby mode.

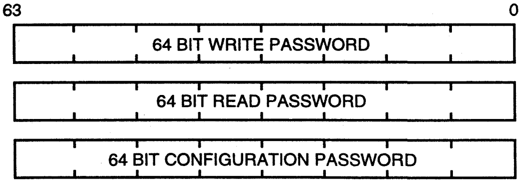
The basic sequence is illustrated in Figure 1.

After each transaction is completed, the X76F041 will reset and enter into a standby mode. This will also be the response if an attempt is made to access any limited array.

Password Registers

The three passwords, Read, Write and Configuration are stored in three 64 bit Write Only registers as illustrated in figure 2.

Figure 2. Password Registers

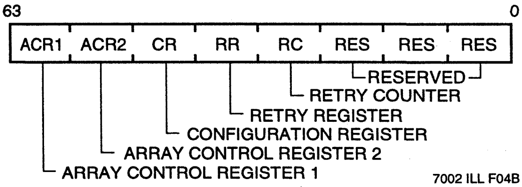


7002 ILL F04

Device Configuration

Five 8-Bit configuration registers are used to configure the X76F041. These are shown in figure 3.

Figure 3. Configuration Registers



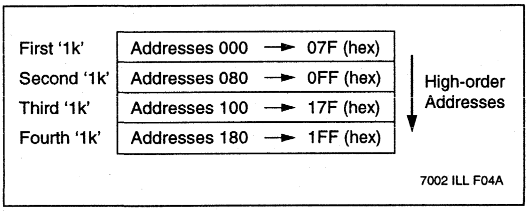
7002 ILL F04B

Array Control

The four 1K arrays, are each programmable to different levels of access and functionality. Each array can be programmed to require or not require the read/write passwords. The functional options are:

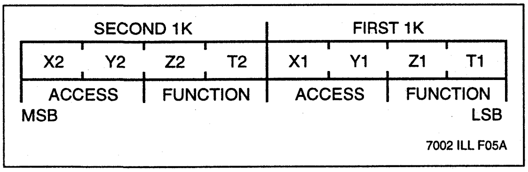
- Read and Write Access.
- Read access with all write operations locked out.
- Read access and program only (writing a "1" to a "0"). If an attempt to change a "0" to a "1" occurs the X76F041 will reset, issue a "no ACK" and enter the standby power mode.
- No read or write access to the memory. Access only through use of the configuration password.

Array Map


First '1k'
Addresses 000 → 07F (hex)
Second '1k'
Addresses 080 → 0FF (hex)
Third '1k'
Addresses 100 → 17F (hex)
Fourth '1k'
Addresses 180 → 1FF (hex)
High-order Addresses

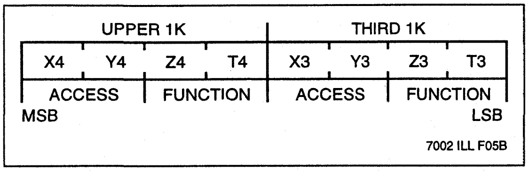
7002 ILL F04A

8 Bit Array Control Register 1



7002 ILL F05A

8 Bit Array Control Register 2



7002 ILL F05B

Functional Bits

Z	T	FUNCTIONALITY
0	0	READ AND WRITE UNLIMITED
1	0	READ ONLY, WRITE LIMITED
0	1	PROGRAM & READ ONLY, ERASE LIMITED
1	1	NO READ OR WRITE, FULLY LIMITED

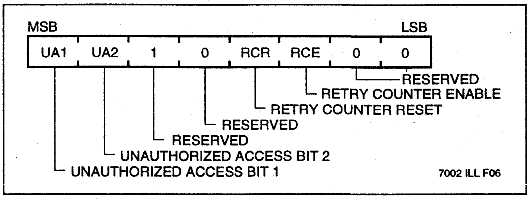
7002 FRM T02

Access Bits

X	Y	READ PASSWORD	WRITE PASSWORD
0	0	NOT REQUIRED	NOT REQUIRED
1	0	NOT REQUIRED	REQUIRED
0	1	REQUIRED	NOT REQUIRED
1	1	REQUIRED	REQUIRED

7002 FRM T03

8-Bit Configuration Register



7002 ILL F06

Unauthorized Access Bits (UA1, UA2):

1 0

Access is forbidden if retry register equals the retry counter (provided that the retry counter is enabled) and no further access of any kind will be allowed.

0 1, 0 0, 1 1

Only configuration operations are allowed if the retry register equals the retry counter (provided that the retry counter is enabled).

Retry Counter Reset Bit (RCR):

If the retry counter reset bit is a "1" then the retry counter will be reset following a correct password, provided the retry counter is enabled.

If the retry counter reset bit is a "0" then the retry counter will not be reset following a correct password, provided the retry counter is enabled.

Retry Counter Enable Bit (RCE):

If the Retry counter enable bit is a "1", then the retry counter is enabled. An initial comparison between the retry register and retry counter determines whether the number of allowed incorrect password attempts has been reached. If not, the protocol continues and in case of a wrong password, the retry counter is incremented by one. If the password is correct then the retry counter will either be reset or unchanged, depending on the reset bit.

The retry register must have a higher value than the retry counter for correct device operation. If the retry counter value is larger than the retry register and the retry counter is enabled, the device will wrap around allowing up to an additional 255 incorrect access attempts.

If the Retry counter enable bit is a "0", then the retry counter is disabled.

Retry Register/Counter

Both the retry register and retry counter are accessible in the configuration mode and may be programmed with a value of 0 to 255.

The difference between the retry register and the retry counter is the number of access attempts allowed, therefore the retry counter must be programmed to a smaller value than the retry register to prevent wrap around.

DEVICE PROTOCOL

The X76F041 supports a bidirectional bus oriented protocol. The protocol defines any device that sends data onto the bus as a transmitter, and the receiving device as the receiver. The device controlling the transfer is a master and the device being controlled is the slave. The master will always initiate data transfers, and provide the clock for both transmit and receive operations. Therefore, the X76F041 will be considered a slave in all applications.

Start Condition

All commands except for response to reset are preceded by the start condition, which is a HIGH to LOW transition of SDA when SCL is HIGH. The X76F041 continuously monitors the SDA and SCL lines for the start condition and will not respond to any command until this condition has been met.

Figure 4. Data Validity During Write

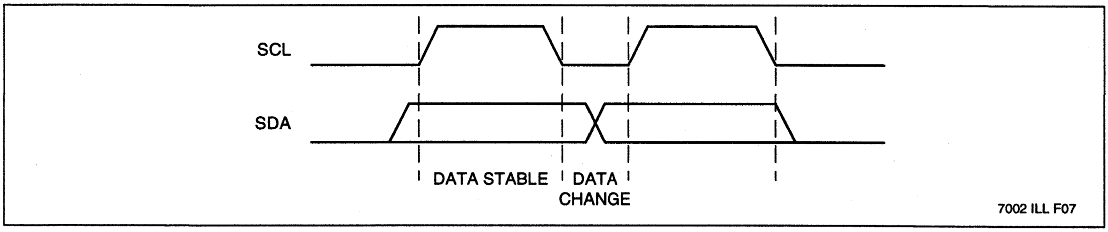
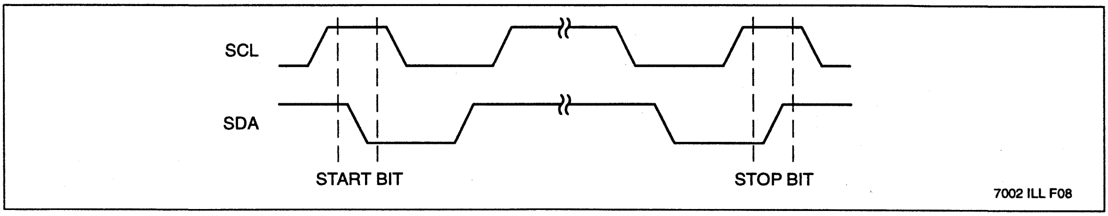


Figure 5. Definition of Start and Stop



NOTE: The part requires the SCL input to be LOW during non-active periods of operation. In other words, the SCL will need to be LOW prior to any START condition and LOW after a STOP condition. This is also reflected in the timing diagram.

Stop Condition

All communications must be terminated by a stop condition, which is a LOW to HIGH transition of SDA when SCL is HIGH. A stop condition can only be issued after the transmitting device has released the bus.

Acknowledge

Acknowledge is a software convention used to indicate successful data transfer. The transmitting device, either master or slave, will release the bus after transmitting eight bits. During the ninth clock cycle the receiver will pull the SDA line LOW to acknowledge that it received the eight bits of data.

OPERATIONAL MODES

THE FIRST BYTE IN THE PROTOCOL	THE SECOND BYTE IN THE PROTOCOL	COMMAND DESCRIPTION	PASSWORD USED:
0 0 0XXXXA	Write address	Write (Sector)	Write
0 0 1XXXXA	Read address	Read (Random / Sequential)	Read
0 1 0XXXXA	Write address	Write (Sector)	Configuration
0 1 1XXXXA	Read address	Read (Random / Sequential)	Configuration
1 0 0XXXXX	0 0 0 0 0 0 0 0	Program write-password	Write
1 0 0XXXXX	0 0 0 1 0 0 0 0	Program read-password	Read
1 0 0XXXXX	0 0 1 0 0 0 0 0	Program configuration-password	Configuration
1 0 0XXXXX	0 0 1 1 0 0 0 0	Reset write password (all 0's)	Configuration
1 0 0XXXXX	0 1 0 0 0 0 0 0	Reset read password (all 0's)	Configuration
1 0 0XXXXX	0 1 0 1 0 0 0 0	Program configuration registers	Configuration
1 0 0XXXXX	0 1 1 0 0 0 0 0	Read configuration registers	Configuration
1 0 0XXXXX	0 1 1 1 0 0 0 0	Mass program	Configuration
1 0 0XXXXX	1 0 0 0 0 0 0 0	Mass erase	Configuration
All the rest		Reserved	

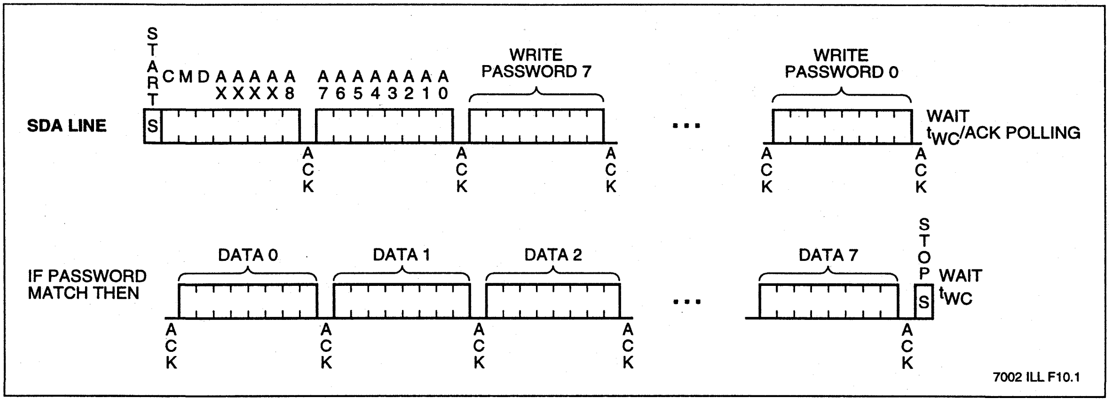
7002 FRM T04

WRITE OPERATION

Sector Write

The Sector Write mode requires issuing the 3-bit write command followed by the address, password if required and then the data bytes transferred as illustrated in Figure 6. Eight bytes must be transferred. After the last byte to be transferred is acknowledged, a stop condition is issued, which starts the nonvolatile write cycle. If more than 8 bytes are transferred the data will wrap around and previous data will be overwritten. All data will be written to the same sector as defined by A_8–A_3.

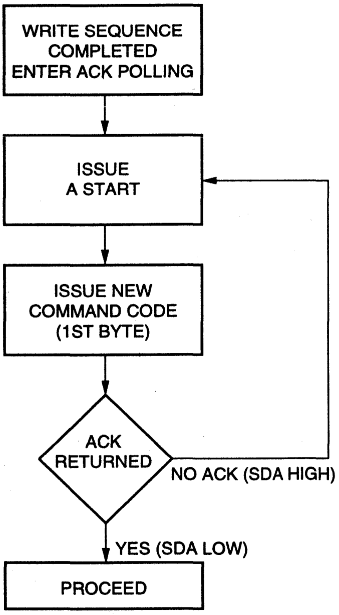
Figure 6. Sector Write



ACK Polling

Once a stop condition is issued to indicate the end of the host's write sequence, the X76F041 initiates the internal nonvolatile write cycle. In order to take advantage of the typical 5ms write cycle, ACK polling can be initiated immediately. This involves issuing the Start condition followed by the new command code of eight bits (1st byte of the protocol). If the X76F041 is still busy with the nonvolatile write operation, it will issue a "no ACK" in response. If the nonvolatile write operation has completed, an "ACK" will be returned and the host can then proceed with the rest of the protocol. Refer to the following flow:

ACK Polling Sequence


WRITE SEQUENCE COMPLETED ENTER ACK POLLING
ISSUE A START
ISSUE NEW COMMAND CODE (1ST BYTE)
ACK RETURNED
NO ACK (SDA HIGH)
YES (SDA LOW)
PROCEED

7002 ILL F12A

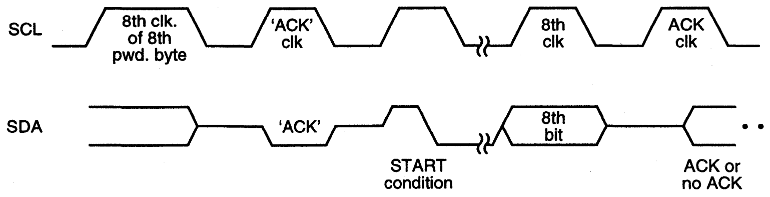
After a password sequence, there is always a nonvolatile write cycle. In order to continue the transaction, the X76F041 requires the master to perform an ACK polling with the specific code of C0h. As with regular acknowledge polling the user can either time out for 10ms, and then issue the ACK polling once, or continuously loop as described in the flow.

As with regular acknowledge polling, if the user chooses to loop, then as long as the nonvolatile write cycle is active, a no ACK will be issued in response to each polling cycle.

If the password that was inserted was correct, then an "ACK" will be returned once the nonvolatile write cycle is over, in response to the ACK polling cycle immediately following it.

If the password that was inserted was incorrect, then a "no ACK" will be returned even if the nonvolatile write cycle is over. Therefore, the user cannot be certain that the password is incorrect until the 10ms write cycle time has elapsed.

Figure 7. Acknowledge Polling



7002 ILL F11

READ OPERATION

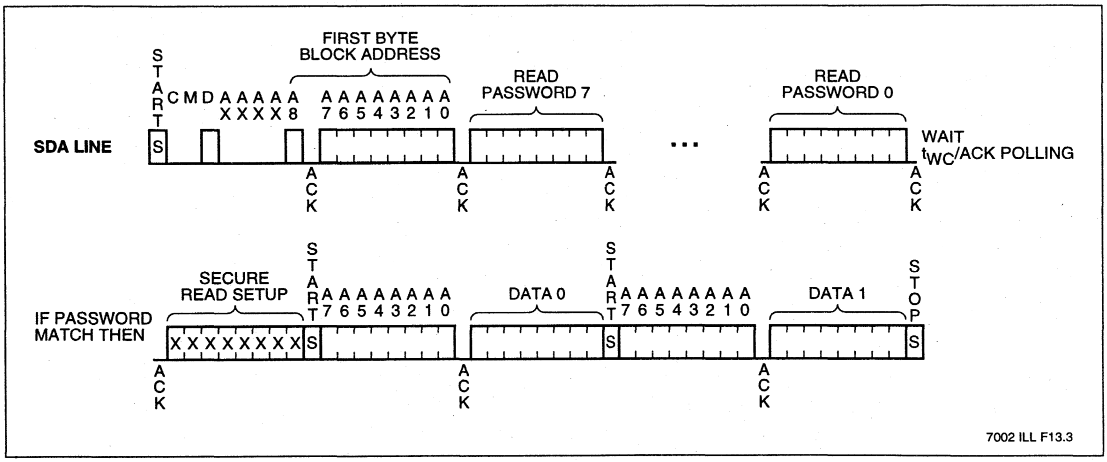
Random Read with Password

Random read with password operations are initiated with a START command followed by the read command and the address of the first byte of the block in which data is to be read:

Block 0 = 000h
Block 1 = 080h
Block 2 = 100h
Block 3 = 180h

This is followed by the eight byte read password sequence which includes the 10ms wait time and the password acknowledge polling sequence. If the password is accepted an "ACK" will be returned followed by eight bits of "secure read setup" which is to be ignored. At this point a START is issued followed by the address and data to be read within the original 1K block. See figure 8. Once the first byte has been read, another start can be issued followed by a new 8-bit address. Random reads are allowed only within the original 1K-bit block. To access another 1K-bit block, a stop must be issued followed by a new command/block address/password sequence.

Figure 8. Random Read with Password



Random Read without Password

Random read operations without a password do not require the first byte block initiation address. To perform a random read without password, a START is followed by the read command plus address location of the byte to be read. This is followed by an "ACK" and the eight bits of data to be read. Other bytes within the same 1K-bit block may be read by issuing another START followed by a new 8-bit address as shown in figure 9.

Sequential Read

Once past the password acceptance sequence (when required) and "secure read setup", the host can read sequentially within the originally addressed 1K-bit array. The data output is sequential, with the data from address n followed by the data from address n+1. The address counter for read operations increments the address, allowing the 1K memory contents to be serially read during one operation. At the end of the address space (address 127), the counter "rolls over" to address space 0 within the 1K Block and the X76F041 continues to output data for each acknowledge received. Refer to figure 10 for the address, acknowledge and data transfer sequence. An acknowledge must follow each 8-bit data transfer. After the last bit has been read, a stop condition is generated without a preceding acknowledge.

Figure 9. Random Read without Password

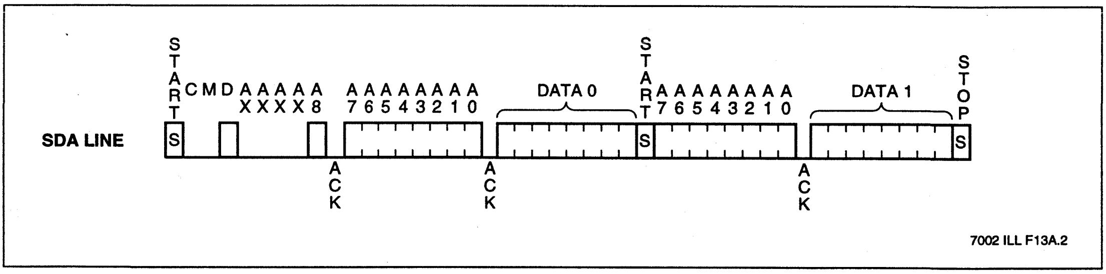
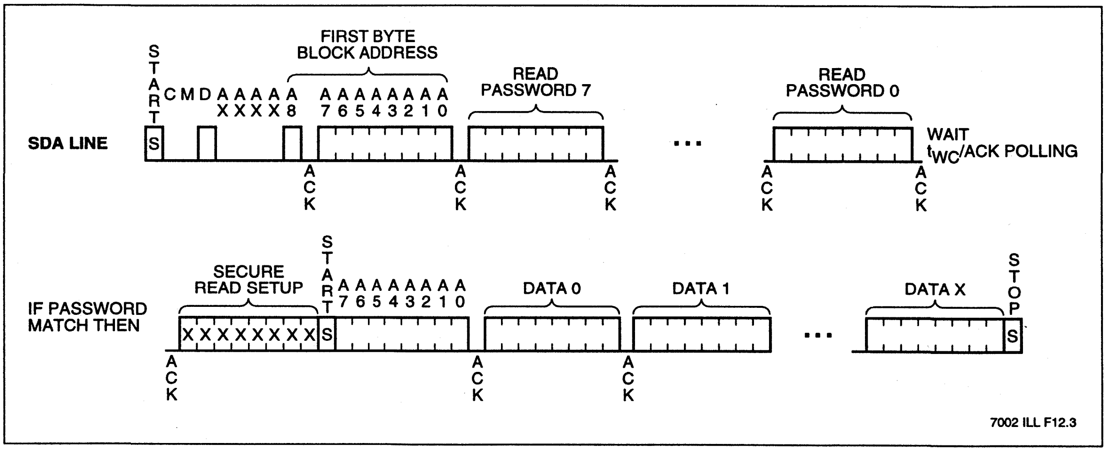


Figure 10. Sequential Read with Password



CONFIGURATION OPERATIONS

Configuration commands generally require the configuration password. The exception is that programming a new read/write password requires the old read/write password and not the configuration password. In most cases these operations will be performed by the equipment manufacturer or end distributor of the equipment or card.

Configuration Read/Write

Configuration read/write allows access to all of the nonvolatile memory arrays regardless of the contents of the configuration registers. Access includes sector writes, random and sequential reads using the same format as normal reads and writes.

In general, the configuration read/write operation enables access to any memory location that may otherwise be limited. The configuration password, in this sense, is like a master key that can override the limits caused by the control partitioning of the arrays.

Figure 11. Configuration Write

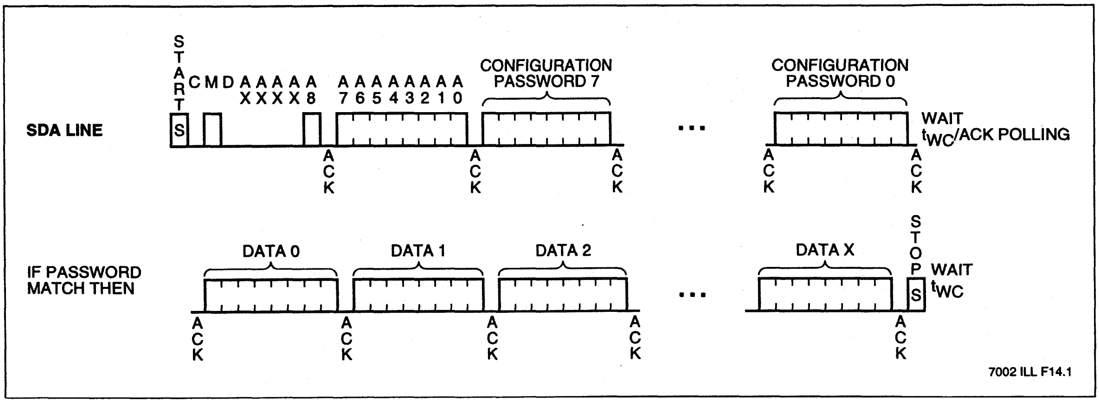
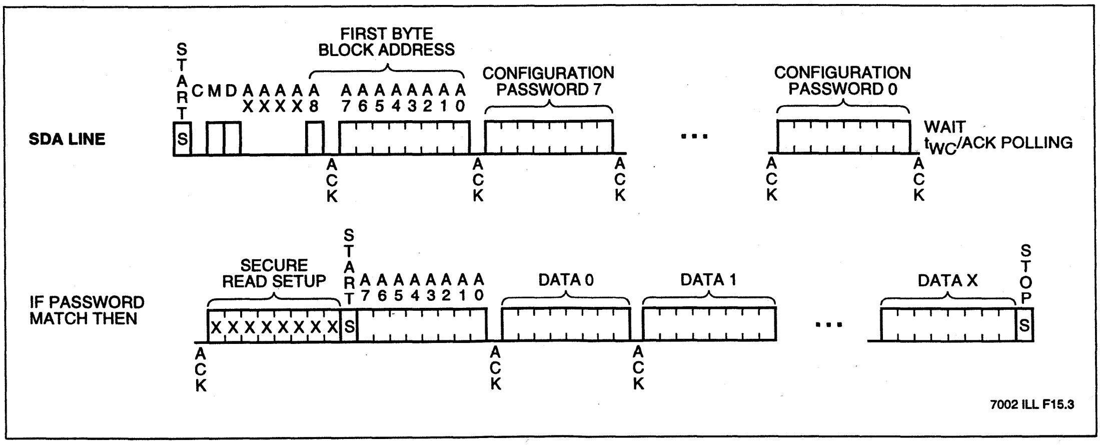


Figure 12. Configuration Sequential Read



Configuration of Passwords

The sequence in figure 14 will change (program) the write, read and configuration passwords. The programming of passwords is done twice prior to the nonvolatile write cycle in order to verify that the new password is consistent. After the eight bytes are entered in the second pass, a comparison takes place. A mismatch will cause the part to reset and enter into the standby mode and a "no ACK" will be issued.

There is no way to read the Read/Write/Configuration passwords.

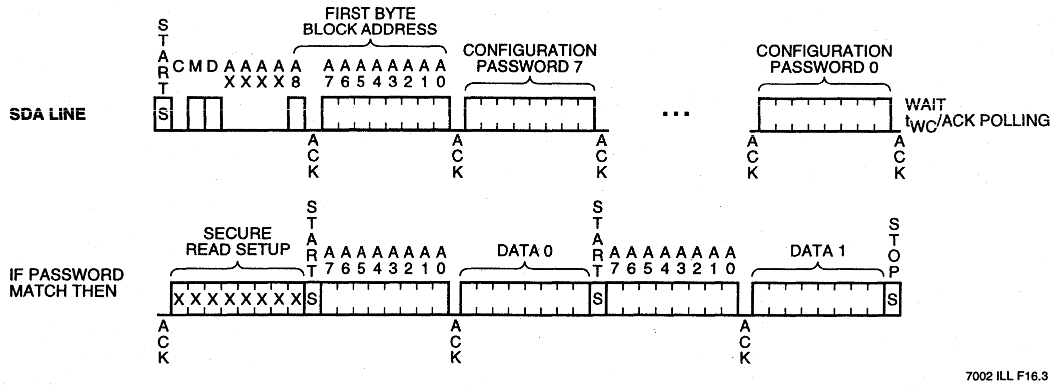
Program Configuration Registers

This mode allows programming of the five configuration/control registers using the configuration password. The retry counter must be programmed with a value less than the retry register. If it is programmed with a value larger than the retry register there will be a wrap around.

Read Configuration Registers

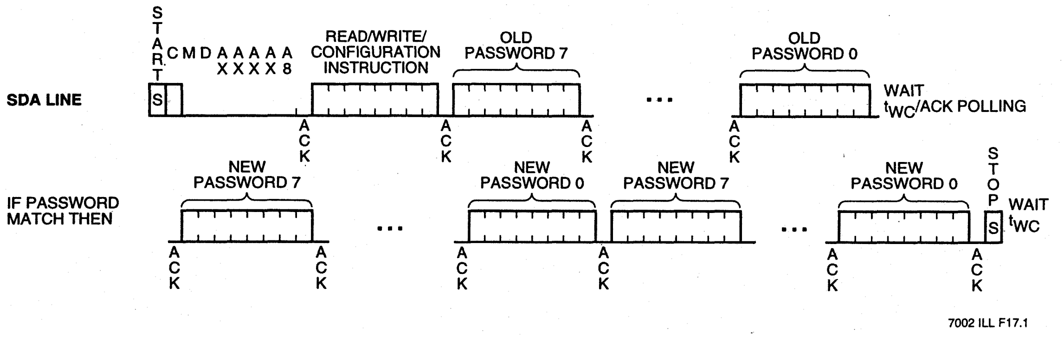
This mode allows reading of the 5 configuration/control registers with the configuration password. It may be useful for monitoring purposes.

Figure 13. Configuration Random Read



7002 ILL F16.3

Figure 14. Program Passwords



7002 ILL F17.1

Read Password Reset

This mode allows resetting of the READ password to all "0"s in case re-programming is needed and the old password is not known.

Write Password Reset

This mode allows resetting of the WRITE password to all "0"s in case re-programming is needed and the old password is not known.

Mass Program

This mode allows mass programming of the array, configuration registers and password to all "0"s using a special configuration command. All parts are shipped mass programmed.

Mass Erase

This mode allows mass erase of the array, configuration register and password to all "1"s using a special configuration command.

Figure 15. Program Configuration Registers

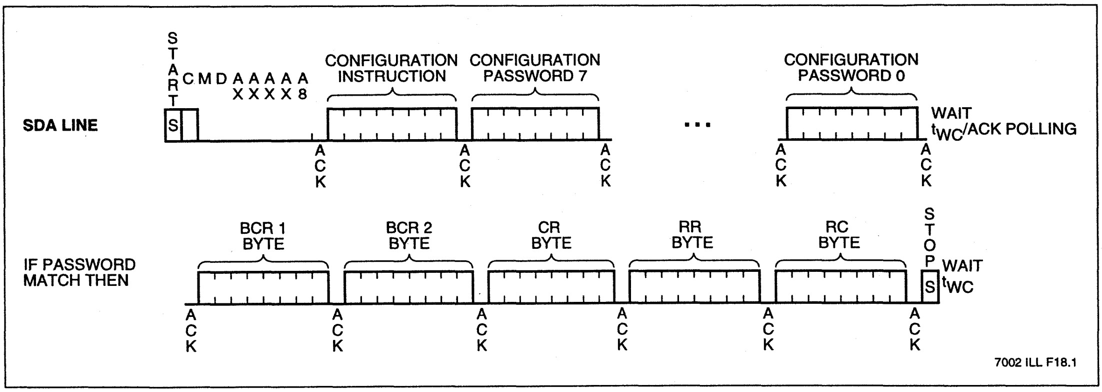


Figure 16. Read Configuration Registers

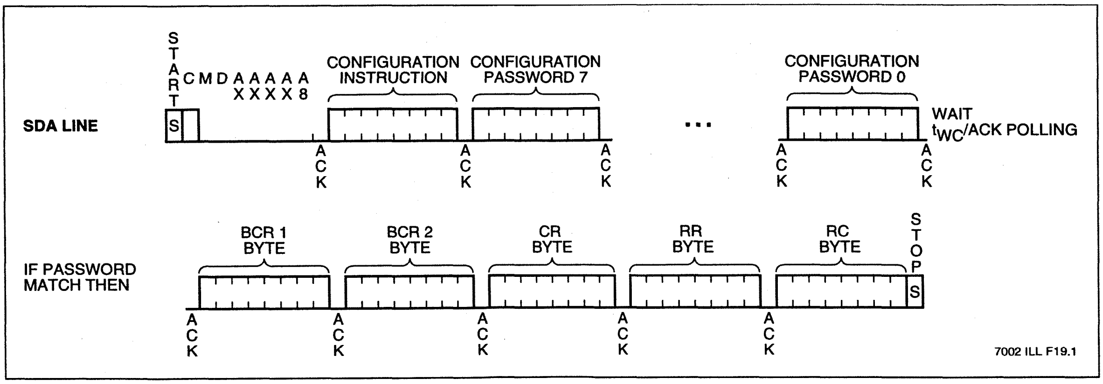


Figure 17. Read/Write Password Reset

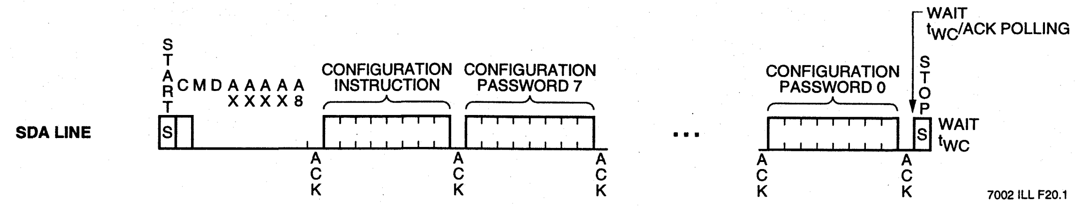
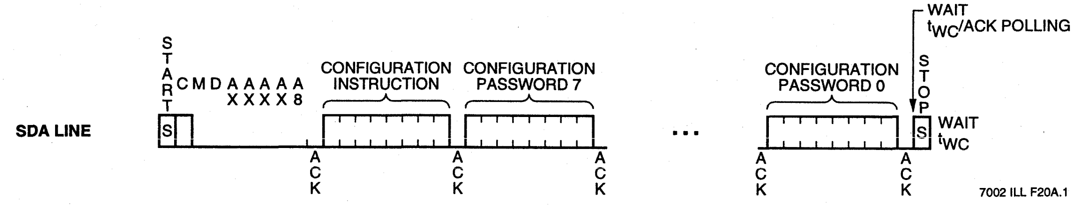
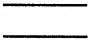
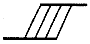
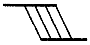
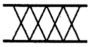
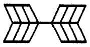


Figure 18. Mass Program/Erase



SYMBOL TABLE

WAVEFORM	INPUTS	OUTPUTS
	Must be steady	Will be steady
	May change from LOW to HIGH	Will change from LOW to HIGH
	May change from HIGH to LOW	Will change from HIGH to LOW
	Don't Care: Changes Allowed	Changing: State Not Known
	N/A	Center Line is High Impedance

ABSOLUTE MAXIMUM RATINGS*

Temperature under Bias	–65°C to +135°C
Storage Temperature	–65°C to +150°C
Voltage on any Pin with Respect to V_{SS}	–1V to +7V
D.C. Output Current	5mA
Lead Temperature (Soldering, 10 seconds)	300°C

*COMMENT

Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and the functional operation of the device at these or any other conditions above those listed in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

RECOMMENDED OPERATING CONDITIONS

Temp	Min.	Max.
Commercial	0°C	+70°C
Extended	–20°C	+85°C

7002 FRM T05

Supply Voltage	Limits
X76F041	4.5V to 5.5V
X76F041 – 3	3V to 3.6V

7002 FRM T06.1

D.C. OPERATING CHARACTERISTICS (Over the recommended operating conditions unless otherwise specified.)

		Limits			
Symbol	Parameter	Min.	Max.	Units	Test Conditions
I_{CC1}	V_{CC} Supply Current (Read)		2	mA	$f_{SCL} = V_{CC} \times 0.1/V_{CC} \times 0.9$ Levels @ 1MHz, SDA = Open RST = $\overline{CS}$ = V_{SS}
I_{CC2}(3)	V_{CC} Supply Current (Write)		3	mA	$f_{SCL} = V_{CC} \times 0.1/V_{CC} \times 0.9$ Levels @ 1MHz, SDA = Open RST = $\overline{CS}$ = V_{SS}
I_{SB1}(1)	V_{CC} Supply Current (Standby)		100	µA	SCL = V_{SS}, $\overline{CS}$ = V_{CC} – 0.3V SDA = Open, RST = V_{CC} = 5.5V
I_{SB2}(1)	V_{CC} Supply Current (Standby)		50	µA	SCL = V_{SS}, $\overline{CS}$ = V_{CC} – 0.3V SDA = Open, RST = V_{SS}, V_{CC} = 3V
I_{LI}	Input Leakage Current		10	µA	V_{IN} = V_{SS} to V_{CC}
I_{LO}	Output Leakage Current		10	µA	V_{OUT} = V_{SS} to V_{CC}
V_{IL1}(2)	Input LOW Voltage	–0.5	V_{CC} x 0.3	V	V_{CC} = 5.5V
V_{IH1}(2)	Input HIGH Voltage	V_{CC} x 0.7	V_{CC} + 0.5	V	V_{CC} = 5.5V
V_{IL2}(2)	Input LOW Voltage	–0.5	V_{CC} x 0.1	V	V_{CC} = 3.0V
V_{IH2}(2)	Input HIGH Voltage	V_{CC} x 0.9	V_{CC} + 0.5	V	V_{CC} = 3.0V
V_{OL}	Output LOW Voltage		0.4	V	I_{OL} = 2mA
V_{OH}	Output HIGH Voltage	V_{CC} – 0.8		V	I_{OH} = –1mA

7002 FRM T07.1

CAPACITANCE T_A = +25°C, f = 1MHz, V_{CC} = 5V

Symbol	Test	Max.	Units	Conditions
C_{OUT}(3)	Output Capacitance (SDA)	10	pF	$V_{I/O}$ = 0V
C_{IN}(3)	Input Capacitance (RST, SCL, $\overline{CS}$)	10	pF	V_{IN} = 0V

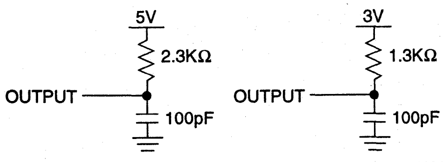
7002 FRM T08

NOTES: (1) Must perform a stop command after a read command prior to measurement

(2) V_{IL} min. and V_{IH} max. are for reference only and are not tested.

(3) This parameter is periodically sampled and not 100% tested.

EQUIVALENT A.C. LOAD CIRCUIT



7002 ILL F21.1

A.C. TEST CONDITIONS

Input Pulse Levels	V_{CC} x 0.1 to V_{CC} x 0.9
Input Rise and Fall Times	10ns
Input and Output Timing Level	V_{CC} x 0.5
Output Load	100pF

7002 FRM T09

A.C. CHARACTERISTICS (Over recommended operating conditions, unless otherwise specified)

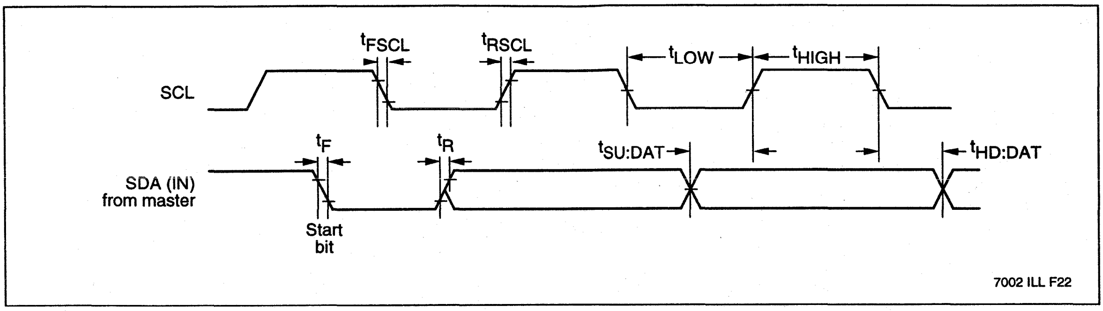
Read & Write Cycle Limits

Symbol	Parameter	Min.	Max.	Units
f_{SCL}	SCL Clock Frequency		1	MHz
TI	Noise Suppression Time Constant at SCL & SDA Inputs		20	ns
t_{DV}	SCL HIGH to SDA Data Valid		450	ns
t_{LOW}	Clock LOW Period	500		ns
t_{HIGH}	Clock HIGH Period	500		ns
t_{STAS1}	Start Condition Setup Time to Rising Edge of SCL	150		ns
t_{STAS2}	Start Condition Setup Time to Falling Edge of SCL	150		ns
t_{STAH1}	Start Condition Hold Time to Rising Edge of SCL	50		ns
t_{STAH2}	Start Condition Hold Time to Falling Edge of SCL	50		ns
t_{STPS1}	Stop Condition Setup Time to Rising Edge of SCL	150		ns
t_{STPS2}	Stop Condition Setup Time to Falling Edge of SCL	150		ns
t_{STPH1}	Stop Condition Hold Time to Rising Edge of SCL	50		ns
t_{STPH2}	Stop Condition Hold Time to Falling Edge of SCL	50		ns
$t_{HD:DAT}$	Data in Hold Time	10		ns
$t_{SU:DAT}$	Data in Setup Time	150		ns
t_{RSCL}(4)	SCL Rise Time		90	ns
t_{FSCL}(4)	SCL Fall Time		90	ns
t_R(4)	SDA, $\overline{CS}$, RST Rise Time		90	ns
t_F(4)	SDA, $\overline{CS}$, RST Fall Time		90	ns
t_{DH}	Data Out Hold Time	0		ns
t_{HZ1}	SCL LOW to High Impedance		150	ns
t_{LZ}	SCL HIGH to Output Active	0		ns
t_{VCCS}	V_{CC} to $\overline{CS}$ Setup Time	5		ms
$t_{SU:CS}$	$\overline{CS}$ Setup Time	200		ns
$t_{HD:CS}$	$\overline{CS}$ Hold Time	100		ns
t_{HZ2}	$\overline{CS}$ Deselect Time		150	ns
$t_{SU:SCL}$	SCL Setup Time to $\overline{CS}$ LOW after Power Up	200		ns
t_{RST}	RST HIGH Time	1500		ns
$t_{SU:RST}$	RST Setup Time	500		ns
$f_{SCL:RST}$	SCL Frequency During Response to Reset		1	MHz
$t_{LOW:RST}$	SCL LOW Time During Response to Reset	500		ns
$t_{HIGH:RST}$	SCL HIGH Time During Response to Reset	500		ns
t_{PD}	SCL LOW to SDA Valid During Response to Reset		450	ns
t_{NOL}	RST to SCL Non-Overlap	500		ns
t_{WC}	Nonvolatile Write Cycle		10	ms

7002 FRM T10

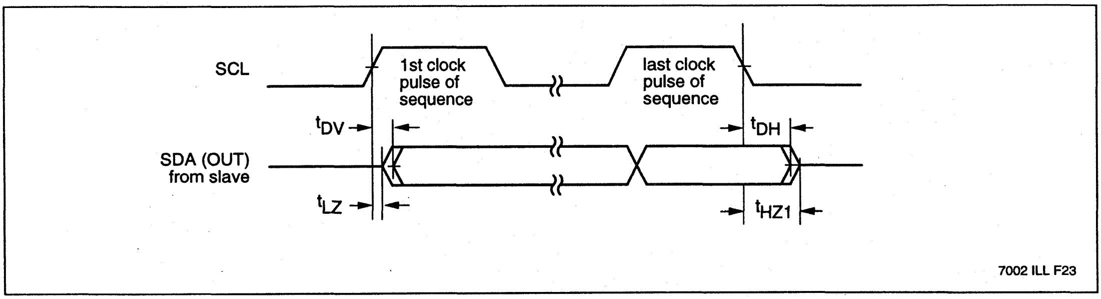
NOTES: (4) This parameter is periodically sampled and not 100% tested.

Bus Timing(1) — SDA Driven by the Bus Master



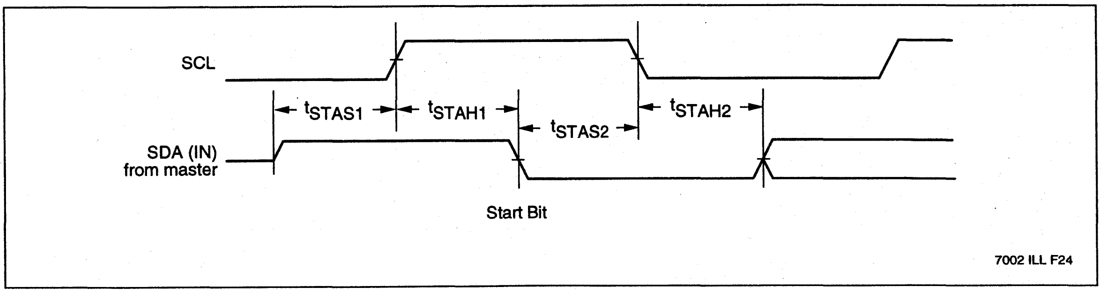
7002 ILL F22

Bus Timing(2) — SDA Driven by the Slave



7002 ILL F23

START Condition Timing

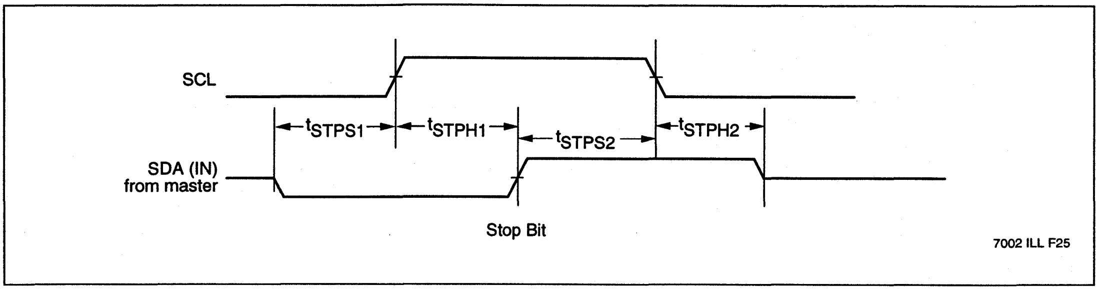


7002 ILL F24

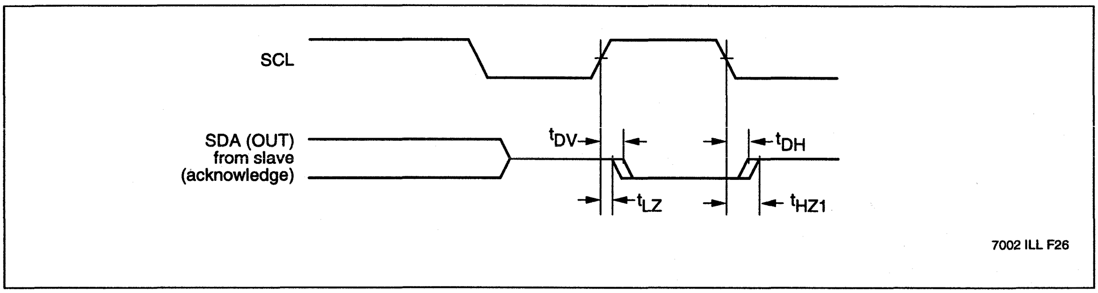
NOTES: (1) The master may issue a STOP condition at any given time in which it is driving the SDA line. In other words, when the part is sending ACK or data the master may NOT issue a STOP condition. The part will not respond to any such attempt which also causes bus contention. At any other time, a STOP condition will cause the part to reset and stop (enter a stand-by mode). Write operations will terminate prior to entering the stand-by mode.

(2) When the part drives the SDA line, it will tri-state the bus only after the last bit of the sequence. In other words, after the 8th bit of a byte that is read or after ACK between incoming bytes. In all other cases when the part drives the bus (between successive bits) it will continue to drive the bus also during the clock LOW periods.

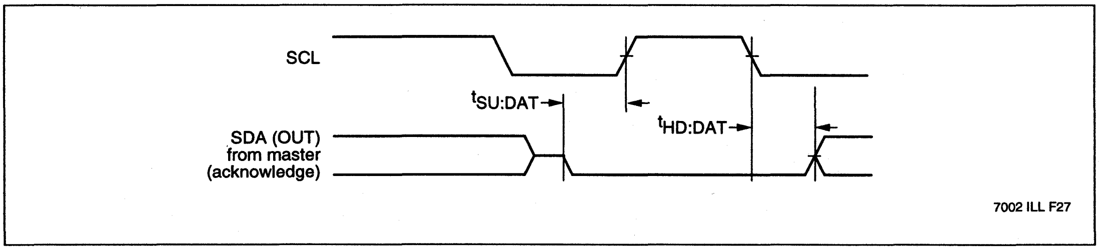
STOP Condition Timing



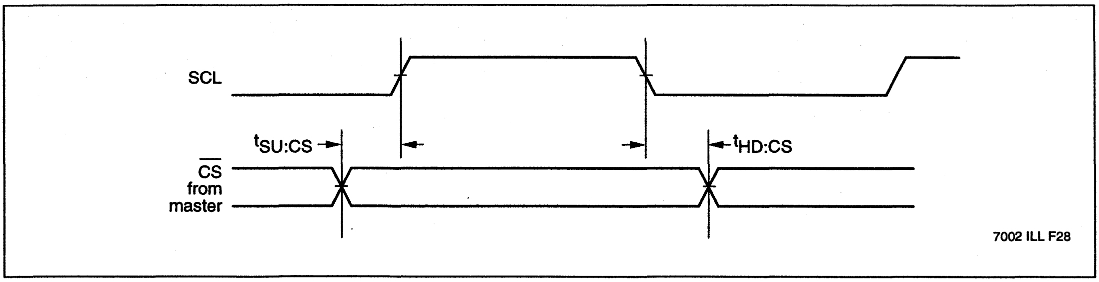
Acknowledge Response from Slave (Same Timing as Data Out)



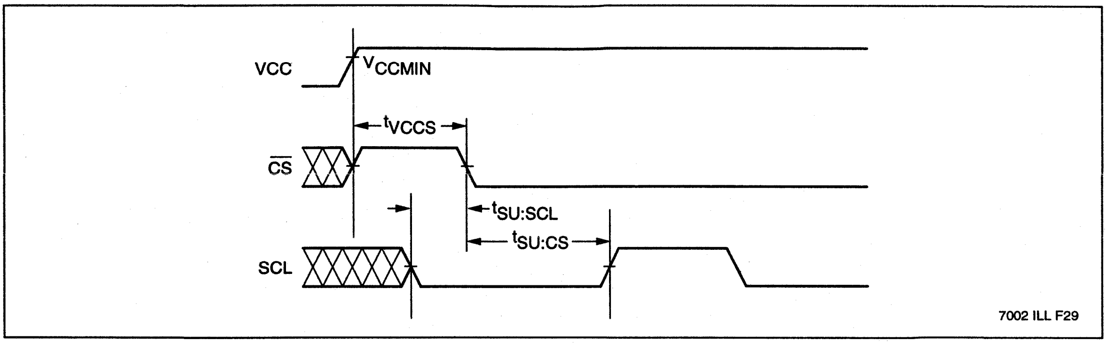
Acknowledge Response from Master



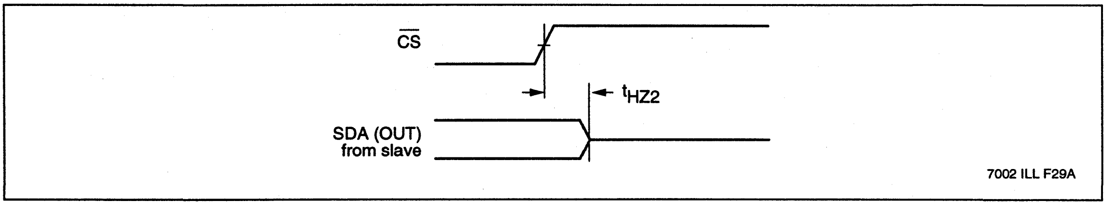
$\overline{CS}$ Timing Diagram (Selecting/Deselecting the Part)



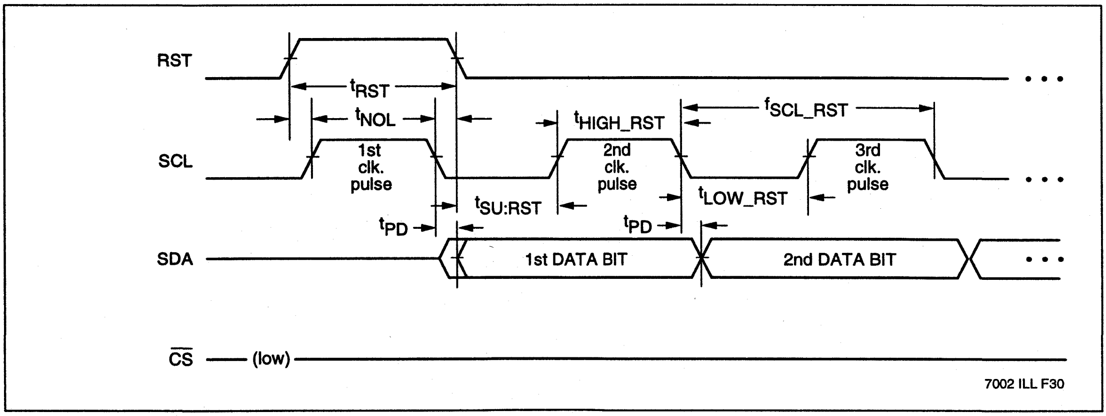
V_{CC} to $\overline{CS}$ Setup Timing Diagram



$\overline{CS}$ Deselect



RST Timing Diagram — Response to a Synchronous Reset (ISO)



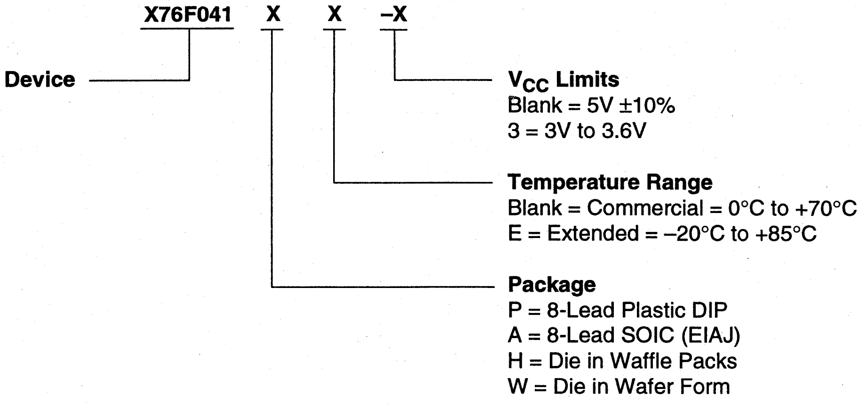
NOTES: (1) The reset operation results in an answer from the part containing a header transmitted from the part to the master. The header has a fixed length of 32 bits and begins with two mandatory fields of eight bits : H1 and H2.

(2) The chronological order of transmission of the information bits shall correspond to bit identification b1 to b32 with the LEAST significant bit transmitted first.

(3) The current values are:

H1 : 19 h
H2 : 55 h
H3 : AA h
H4 : 55 h

ORDERING INFORMATION



X76F041 X X –X

Device

V_{CC} Limits
Blank = 5V ±10%
3 = 3V to 3.6V

Temperature Range
Blank = Commercial = 0°C to +70°C
E = Extended = –20°C to +85°C

Package
P = 8-Lead Plastic DIP
A = 8-Lead SOIC (EIAJ)
H = Die in Waffle Packs
W = Die in Wafer Form

LIMITED WARRANTY

Devices sold by Xicor, Inc. are covered by the warranty and patent indemnification provisions appearing in its Terms of Sale only. Xicor, Inc. makes no warranty, express, statutory, implied, or by description regarding the information set forth herein or regarding the freedom of the described devices from patent infringement. Xicor, Inc. makes no warranty of merchantability or fitness for any purpose. Xicor, Inc. reserves the right to discontinue production and change specifications and prices at any time and without notice.

Xicor, Inc. assumes no responsibility for the use of any circuitry other than circuitry embodied in a Xicor, Inc. product. No other circuits, patents, licenses are implied.

U.S. PATENTS

Xicor products are covered by one or more of the following U.S. Patents: 4,263,664; 4,274,012; 4,300,212; 4,314,265; 4,326,134; 4,393,481; 4,404,475; 4,450,402; 4,486,769; 4,488,060; 4,520,461; 4,533,846; 4,599,706; 4,617,652; 4,668,932; 4,752,912; 4,829, 482; 4,874, 967; 4,883, 976. Foreign patents and additional patents pending.

LIFE RELATED POLICY

In situations where semiconductor component failure may endanger life, system designers using this product should design the system with appropriate error detection and correction, redundancy and back-up features to prevent such an occurence.

Xicor's products are not authorized for use in critical components in life support devices or systems.

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform, when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.
2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

Xicor®

Advance Information

128K/64K **X84F128/064** **MPS™ Serial Flash**

Micro Port Saver SerialFlash with Block Lock™ Protection

FEATURES

- **Direct Interface to Micros**
  - **—Eliminates I/O port requirements**
  - **—No interface glue logic required**
  - **—Eliminates need for parallel to serial converters**
- **New Programmable Block Lock™ Protection**
  - **—Software Program Protection**
  - **—Programmable Hardware Program Protection**
- **Block Lock (0, 1/4, 1/2, or all of the array)**
- **Up to 5Mbps data transfer rate**
- **Low Power CMOS**
  - **—3V or 5V "Univolt" Read and Program Power Supply Versions**
  - **—Standby Current Less than 1µA**
  - **—Active Current Less than 1mA**
- **Minimum 45ns Read Access Time**
- **256-Bit Sector Program Mode**
- **Typical Nonvolatile Program Cycle Time: 5ms**
- **High Reliability**
  - **—100,000 Endurance Cycles**
  - **—Data Retention: 100 Years**

DESCRIPTION

The X84F128/064 Micro Port SerialFlash devices are 16K/8K-bit CMOS memories designed for direct interface to port limited microcontroller or I/O limited ASIC and microprocessor designs. The MPS interface eliminates the need for parallel to serial conversion hardware simplifying system design.

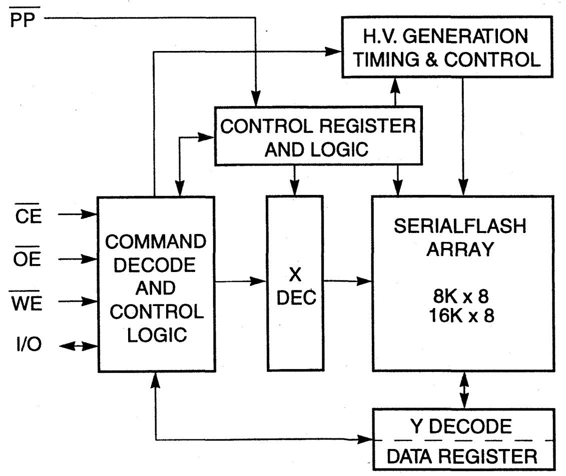
The X84F128/064 provide all of the benefits of serial memories, such as low cost, low power, low voltage operation, and small package size, while featuring higher data transfer rates and reduced interface code requirements—without the need for a dedicated serial bus. All of the products are organized into 256 bit sectors and are suitable in 8-bit, 16-bit, 32-bit, or 64-bit environments, due to the bit serial nature of the interface.

The X84F128/064 devices directly connect to the system bus and communicate over a single data line using a sequence of standard bus read and write operations. This eliminates the need for dedicated port pins, parallel to serial converters, complicated ASIC implementations, or other glue logic, lowering system cost.

The X84F128/064 provides additional data security features through Block Lock and programmable Hardware Program Protection. These allow some or all of the array to be program protected by software command or by hardware. System Configuration, Company ID, calibration information, or other critical data can be secured a gainst unexpected or inadvertent program operations, leaving the remainder of the memory available for the system or user access.

4

BLOCK DIAGRAM



7022 FRM F01

Xicor E^{2}PROMs are designed and tested for applications requiring extended endurance. Inherent data retention is greater than 100 years.

PIN DESCRIPTIONS

Chip Enable ($\overline{CE}$)

The Chip Enable input must be LOW to enable all read/write operations. When $\overline{CE}$ is HIGH, the chip is deselected, the I/O pin is in the high impedance state, and unless a nonvolatile program operation is underway, the device is in the standby power mode.

Output Enable ($\overline{OE}$)

The Output Enable input must be LOW to enable the output buffer and to read data from the device on the I/O line.

Write Enable ($\overline{WE}$)

The Write Enable input must be LOW to write either data or command sequences to the device.

Data In/Data Out (I/O)

Data and command sequences are serially written to or serially read from the device through the I/O pin.

Program Protect ($\overline{PP}$)

The Program Protect input controls the Hardware Program Protect feature. When $\overline{PP}$ is LOW and the nonvolatile bit PPEN is "1", nonvolatile programming of the X84F128/064 control register is disabled, but the part otherwise functions normally. When $\overline{PP}$ is held HIGH, all functions, including nonvolatile programming operate normally. $\overline{PP}$ going LOW while $\overline{CS}$ is still LOW will interrupt a program to the X84F128/064 control register. If the internal Program cycle has already been initiated, $\overline{PP}$ going LOW will have no effect on programming.

The $\overline{PP}$ pin function is blocked when the PPEN bit in the control register is "0". This allows the user to install the X84F128/064 in a system with $\overline{PP}$ pin grounded and still be able to program to the control register. The $\overline{PP}$ pin functions will be enabled when the PPEN bit is set "1".

PIN NAMES

I/O	Data Input/Output
$\overline{CE}$	Chip Enable Input
$\overline{OE}$	Output Enable Input
$\overline{WE}$	Write Enable Input
$\overline{PP}$	Program Protect Input
V_{CC}	Supply Voltage
V_{SS}	Ground
NC	No Connect

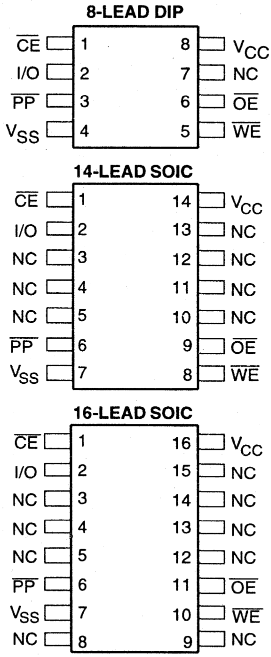
7022 FRM T01

PINOUT/PACKAGE SELECTION GUIDE

X84F064	8-Lead Dip (P) 14-Lead SOIC (S)
X84F128	8-Lead Dip (P) 16-Lead SOIC (S)

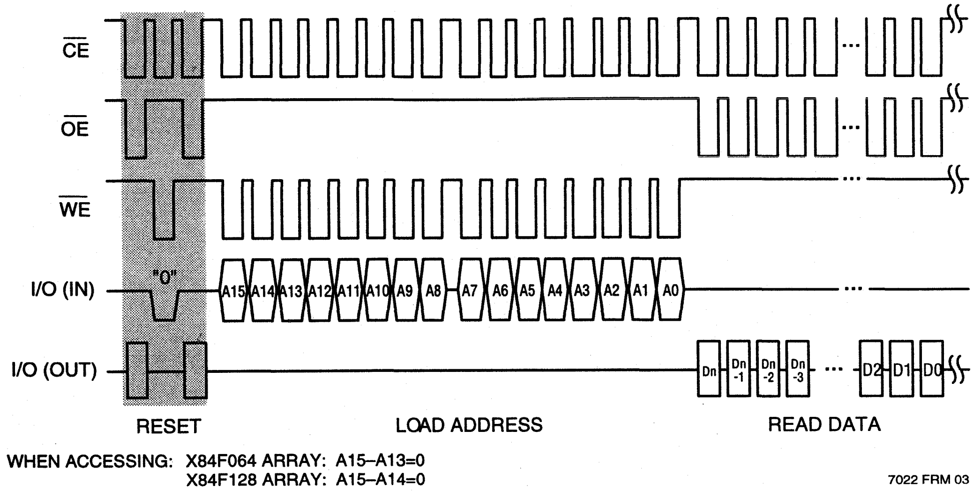
7015 FRM T0A

PIN CONFIGURATIONS



7015 FRM 02

Figure 1. Read Sequence



WHEN ACCESSING: X84F064 ARRAY: A15–A13=0
X84F128 ARRAY: A15–A14=0

7022 FRM 03

DEVICE OPERATION

The X84F128/064 are SerialFlash devices designed to interface directly with most microprocessor buses. Standard $\overline{CE}$, $\overline{OE}$, and $\overline{WE}$ signals control the read and program operations, and a single I/O line is used to send and receive data and commands serially.

Data Timing

Data input on the I/O line is latched on the rising edge of either $\overline{WE}$ or $\overline{CE}$, whichever occurs first. Data output on the I/O line is active whenever both $\overline{OE}$ and $\overline{CE}$ are LOW. Care should be taken to ensure that $\overline{WE}$ and $\overline{OE}$ are never both LOW while $\overline{CE}$ is LOW.

Read Sequence

A read sequence consists of sending a 16-bit address followed by the reading of data serially. The address is written by issuing 16 separate write cycles ($\overline{WE}$ and $\overline{CE}$ LOW, $\overline{OE}$ HIGH) to the part without a read cycle between the write cycles. The address is sent serially, most significant bit first, over the I/O line. Note that this sequence is fully static, with no special timing restrictions, and the processor is free to perform other tasks on the bus whenever the device $\overline{CE}$ pin is HIGH. Once the 16 address bits are sent, a byte of data can be read on the I/O line by issuing 8 separate read cycles ($\overline{OE}$ and $\overline{CE}$ LOW, $\overline{WE}$ HIGH). At this point, writing a '1' will terminate the read sequence and enter the low power standby state, otherwise the device will await further reads in the sequential read mode.

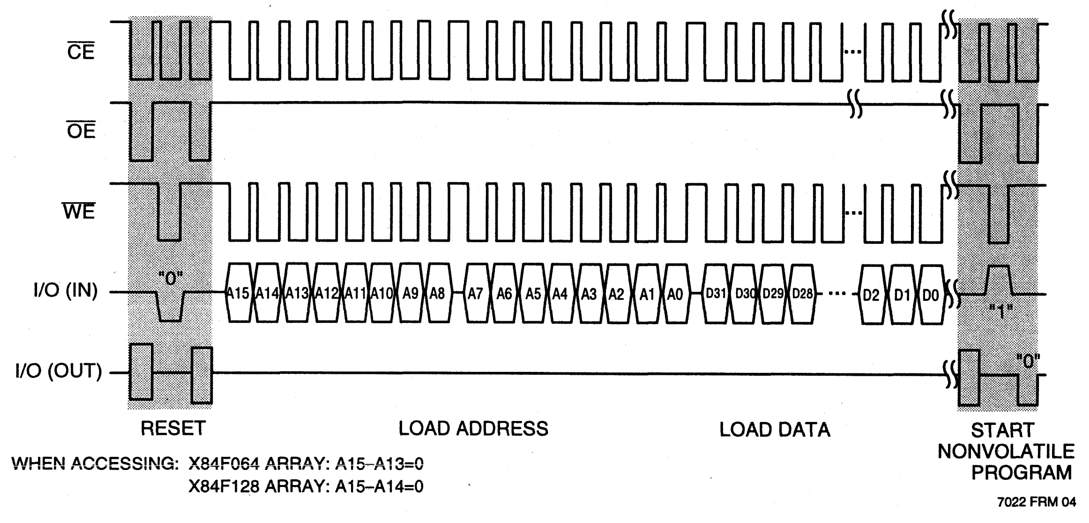
Sequential Read

The bit address is automatically incremented to the next higher address after each bit of data is read. The data stored in the memory at the next address can be read sequentially by continuing to issue read cycles. When the highest address in the array is reached, the address counter rolls over to address $000 and reading may be continued indefinitely.

Reset Sequence

The reset sequence resets the device and sets an internal Program enable latch. A reset sequence can be sent at any time by performing a read/write "0"/read operation (see Figs. 1 and 2). This breaks the multiple read or Program cycle sequences that are normally used to read from or program the part. The reset sequence can be used at any time to interrupt or end a sequential read or page load. As soon as the write "0" cycle is complete, the part is reset (unless a nonvolatile program cycle is in progress). The second read cycle in this sequence, and any further read cycles, will read a HIGH on the I/O pin until a valid read sequence (which includes the address) is issued. The reset sequence must be issued at the beginning of both read and program sequences to be sure the device initiates these operations properly.

Figure 2: Program Sequence



WHEN ACCESSING: X84F064 ARRAY: A15–A13=0
X84F128 ARRAY: A15–A14=0

7022 FRM 04

Program Sequence

A nonvolatile program sequence consists of sending a reset sequence, the 16-bit address of the first location in a sector, 256-bits bytes of data, and then a special “start nonvolatile program cycle” command sequence. The reset sequence is issued first (as described in the Reset Sequence section) to set an internal program enable latch. The address is written serially by issuing 16 separate write cycles ($\overline{\text{WE}}$ and $\overline{\text{CE}}$ LOW, $\overline{\text{OE}}$ HIGH) to the part without any read cycles between the writes.The address is sent serially, most significant bit first,on the I/O pin. The 256-bits of data are programmed by issuing 256 separate write cycles. Again, no read cycles are allowed between writes and a full 256-bit sector must be programmed.

The nonvolatile program cycle is initiated by issuing a special read/write “1”/read sequence. The first read cycle ends the sector load, then the write “1” followed by a read starts the nonvolatile program cycle. The device recognizes 256-bit sectors beginning at addresses XXXX00000. When sending data to the part, attempts to exceed the upper address of the page will result in undefined data being programmed in the array.

A nonvolatile program cycle will not start if a partial or incomplete program sequence is issued. The internal program enable latch is reset when the nonvolatile program cycle is completed to prevent inadvertent programming.

Nonvolatile Program Status

The status of a nonvolatile program cycle can be determined at any time by simply reading the state of the I/O pin on the device. This pin is read when $\overline{\text{OE}}$ and $\overline{\text{CE}}$ are LOW and $\overline{\text{WE}}$ is HIGH. During a nonvolatile program cycle the I/O pin is LOW. When the nonvolatile program cycle is complete, the I/O pin goes HIGH. A reset sequence can also be issued during a nonvolatile program cycle with the same result: I/O is LOW as long as a nonvolatile program cycle is in progress, and I/O is HIGH when the nonvolatile program cycle is done.

CONTROL REGISTER

The X84F128/064 has one register that contains control bits for the devices. The control bits, PPEN, BP1, and BP0, are shown in Table 1. To read or change the contents of this register requires a one byte operation to address FFFFh.

A read from FFFFh returns the one byte contents of the control register unused bits return 0. Continued reads return undefined data. Programming address FFFFh changes the value of the bits. Unused bits are programmed as "0". Writing more than one byte to the control register is a violation and the operation will be aborted. After sending one byte to the control register, a start nonvolatile program cycle will latch in the new state.

Table 1

7	6	5	4	3	2	1	0
PPEN	0	0	0	BP1	BP0	0	0

7022 FRM T02

PPEN: Program Protect Enable Bit

The Program-Protect-Enable (PPEN) bit is an enable bit for the $\overline{\text{PP}}$ pin.

Table Table 2

PPEN	$\overline{\text{PP}}$	Protected Blocks	Unprotected Blocks	Control Register
0	X	Protected	Programmable	Programmable
1	LOW	Protected	Programmable	Protected
X	HIGH	Protected	Programmable	Programmable

7022 FRM T03.

The Program Protect ($\overline{\text{PP}}$) pin and the nonvolatile Program Protect Enable (PPEN) bit in the Control Register control the programmable hardware program protect feature. Hardware program protection is enabled when $\overline{\text{PP}}$ pin is LOW, and the PPEN bit is "1". Hardware program protection is disabled when either the $\overline{\text{PP}}$ pin is HIGH or the PPEN bit is "0". When the chip is hardware program protected, nonvolatile programming is disabled to the Control Register, including the Block Protect bits and the PPEN bit itself, as well as the block-protected sections in the memory array. Only the sections of the memory array that are not block-protected can be programmed.

Note: When the $\overline{\text{PP}}$ pin is tied to V_{SS} and the PPEN bit is HIGH, the PPEN bit is Program protected. It cannot be changed back to a "0", as long as the $\overline{\text{PP}}$ pin is held LOW.

BP1, BP0: Block Protect Bits

The Block Protect (BP0 and BP1) bits are nonvolatile and allow the user to select one of four levels of protection. The X84F128/064 is divided into four segments. One, two, or all four of the segments may be protected. That is, the user may read the segments but will be unable to alter (program) data within the selected segments. The partitioning is controlled as illustrated in table 3 below.

Table 3. Block Lock Protection

Control Register Bits		Array Address Protected		
BP1	BP0	X84F064	X84F128	
0	0	None	None	
0	1	1800h–1FFFh	3000h–3FFFh	upper 1/4
1	0	1000h–1FFFh	2000h–3FFFh	upper 1/2
1	1	0000–1FFFh	0000–3FFFh	Full Array (Not including the control register.)

7015 FRM T04

Low Power Operation

The device enters an idle state, which draws minimal current when:

—an illegal sequence is entered. The following are the more common illegal sequences:

- Read/Write/Write—any time
- Read/Write '1'—When writing the address or writing data.
- Write '1'—when reading data
- Read/Read/Write '1'—after data is written to device, but before entering the NV program sequence.

—the device powers-up;

—a nonvolatile program operation completes.

While a sequential read is in progress, the device remains in an active state. This state draws more current than the idle state, but not as much as during a read itself. To go back to the lowest power condition, an invalid condition is created by writing a '1' after the last bit of a read operation.

Program Protection

The following circuitry has been included to prevent inadvertent nonvolatile programming:

—The internal program enable latch is reset upon power-up.

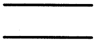
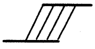
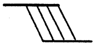
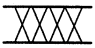
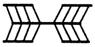
—A reset sequence must be issued to set the internal program enable latch before starting a program sequence.

—A special "start nonvolatile program" command sequence is required to start a nonvolatile program cycle.

—The internal program enable latch is reset automatically at the end of a nonvolatile program cycle.

—The internal program Enable latch is reset and remains reset as long as the $\overline{PP}$ pin is LOW, which blocks all nonvolatile program cycles.

SYMBOL TABLE

WAVEFORM	INPUTS	OUTPUTS
	Must be steady	Will be steady
	May change from LOW to HIGH	Will change from LOW to HIGH
	May change from HIGH to LOW	Will change from HIGH to LOW
	Don't Care: Changes Allowed	Changing: State Not Known
	N/A	Center Line is High Impedance

7022 FRM 05

ABSOLUTE MAXIMUM RATINGS*

Temperature under Bias	–65°C to +135°C
Storage Temperature	–65°C to +150°C
Terminal Voltage with Respect to V_{SS}	–1V to +7V
DC Output Current	5mA
Lead Temperature (Soldering, 10 seconds)	300°C

RECOMMENDED OPERATING CONDITIONS

Temperature	Min.	Max.
Commercial	0°C	+70°C
Extended	–20°C	+85°C
Industrial	-40°C	+85°C

7022 FRM T05

*COMMENT

Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and the functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Supply Voltage	Limits
X84F128/064	5V ±10%
X84F128/064	3V ± 20%

7022 FRM T06

D.C. OPERATING CHARACTERISTICS: (VCC = 5V ±10%)

(Over the recommended operating conditions, unless otherwise specified.)

		Limits			
Symbol	Parameter	Min.	Max.	Units	Test Conditions
I_{CC1}	V_{CC} Supply Current (Read)		1	mA	$\overline{OE} = V_{IL}$, $\overline{WE} = V_{IH}$, I/O = Open, $\overline{CE}$ clocking @ 5MHz
I_{CC2}	V_{CC} Supply Current (Program)		3	mA	I_{CC} During Nonvolatile Program Cycle All Inputs at CMOS Levels
I_{SB1}(2)	V_{CC} Standby Current		1	µA	$\overline{CE} = V_{CC}$, Other Inputs = V_{CC} or V_{SS}
I_{LI}	Input Leakage Current		10	µA	$V_{IN} = V_{SS}$ to V_{CC}
I_{LO}	Output Leakage Current		10	µA	$V_{OUT} = V_{SS}$ to V_{CC}
V_{IL} (1)	Input LOW Voltage	–1	V_{CC} x 0.3	V	
V_{IH} (1)	Input HIGH Voltage	V_{CC} x 0.7	V_{CC} + 0.5	V	
V_{OL}	Output LOW Voltage		0.4	V	I_{OL} = 2.1mA
V_{OH}	Output HIGH Voltage	V_{CC} – 0.8		V	I_{OH} = –1mA

7022 FRM T07

Notes: (1) V_{IL} Min. and V_{IH} Max. are for reference only and are not tested.

(2) This parameter is periodically sampled and not 100% tested.

D.C. OPERATING CHARACTERISTICS (V_{CC} = 3V ± 20%)

(Over the recommended operating conditions, unless otherwise specified.)

Symbol	Parameter	Limits		Units	Test Conditions
		Min.	Max.		
I_{CC1}	V_{CC} Supply Current (Read)		500	µA	$\overline{OE} = V_{IL}$, $\overline{WE} = V_{IH}$, I/O = Open, $\overline{CE}$ clocking @ 3MHz
I_{CC2}	V_{CC} Supply Current (Program)		2	mA	I_{CC} During Nonvolatile Program Cycle All Inputs at CMOS Levels
I_{SB1}[2]	V_{CC} Standby Current		1	µA	$\overline{CE} = V_{CC}$, Other Inputs = V_{CC} or V_{SS}
I_{LI}	Input Leakage Current		10	µA	$V_{IN} = V_{SS}$ to V_{CC}
I_{LO}	Output Leakage Current		10	µA	$V_{OUT} = V_{SS}$ to V_{CC}
V_{IL}[1]	Input LOW Voltage	–1	V_{CC} x 0.3	V	
V_{IH}[1]	Input HIGH Voltage	V_{CC} x 0.7	V_{CC} + 0.5	V	
V_{OL}	Output LOW Voltage		0.4	V	I_{OL} = 1mA
V_{OH}	Output HIGH Voltage	V_{CC} – 0.4		V	I_{OH} = –400µA

7022 FRM T08

CAPACITANCE T_A = +25°C, f = 1MHz, V_{CC} = 5V

Symbol	Parameter	Max.	Units	Test Conditions
$C_{I/O}$[2]	Input/Output Capacitance	8	pF	$V_{I/O}$ = 0V
C_{IN}[2]	Input Capacitance	6	pF	V_{IN} = 0V

7022 FRM T09

POWER-UP TIMING

Symbol	Parameter	Max.	Units
t_{PUR}[3]	Power-up to Read Operation	1	ms
t_{PUW}[3]	Power-up to Program Operation	1	ms

7022 FRM T10

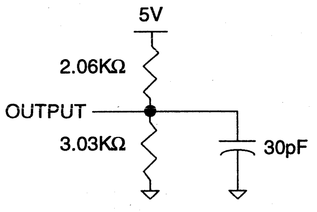
A.C. CONDITIONS OF TEST

Input Pulse Levels	V_{CC} x 0.1 to V_{CC} x 0.9
Input Rise and Fall Times	10ns
Input and Output Timing Levels	V_{CC} x 0.5

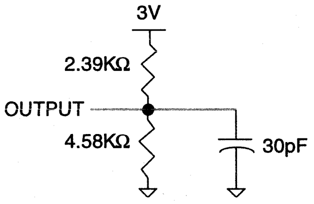
7022 FRM T11

Notes: (3) Time delays required from the time the V_{CC} is stable until the specific operation can be initiated. Periodically sampled, and not 100% tested.

EQUIVALENT A.C. LOAD CIRCUITS



7022 ILL F06



7022 ILL F07

A.C. CHARACTERISTICS

(Over the recommended operating conditions, unless otherwise specified.)

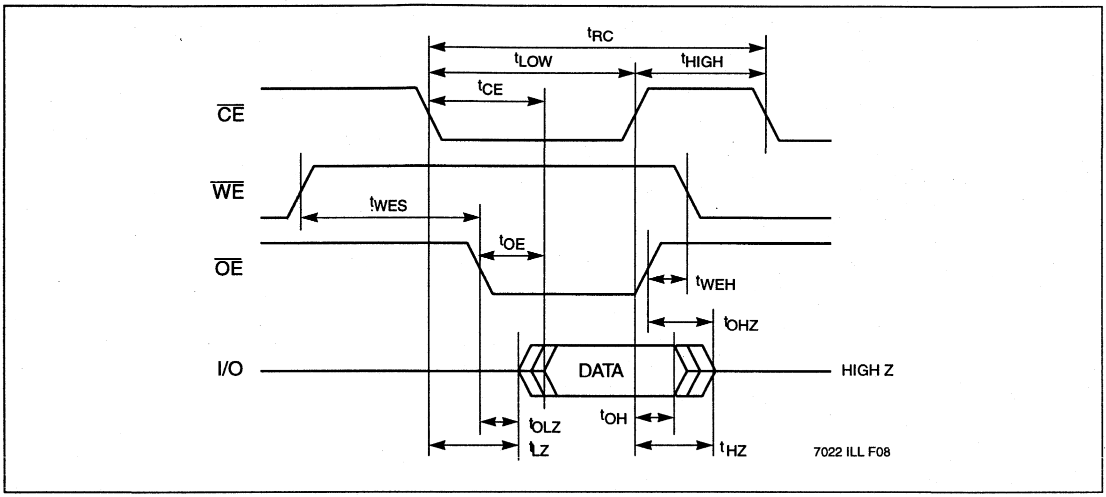
Read Cycle Limits – X84F128/064

Symbol	Parameter	V_{CC} = 5V ±10%		V_{CC} = 3V ± 20%		Units
		Min.	Max.	Min.	Max.	
t_{RC}	Read Cycle Time	200		330		ns
t_{CE}	$\overline{CE}$ Access Time		45		120	ns
t_{OE}	$\overline{OE}$ Access Time		45		120	ns
t_{LOW}	$\overline{CE}$ LOW Time	70		150		ns
t_{HIGH}	$\overline{CE}$ HIGH Time	70		150		ns
t_{LZ}(4)	$\overline{CE}$ LOW to Output In Low Z	0		0		ns
t_{HZ}(4)	$\overline{CE}$ HIGH to Output In High Z	0	30	0	45	ns
t_{OLZ}(4)	$\overline{OE}$ LOW to Output In Low Z	0		0		ns
t_{OHZ}(4)	$\overline{OE}$ HIGH to Output In High Z	0	30	0	45	ns
t_{OH}	Output Hold from $\overline{CE}$ or $\overline{OE}$ HIGH	0		0		ns
t_{WES}	$\overline{WE}$ HIGH Setup Time	25		25		ns
t_{WEH}	$\overline{WE}$ HIGH Hold Time	25		25		ns

7022 FRM T12

Notes: (4) Periodically sampled, but not 100% tested. t_{HZ} and t_{OHZ} are measured from the point where $\overline{CE}$ or $\overline{OE}$ goes HIGH (whichever occurs first) to the time when I/O is no longer being driven into a 5pF load.

Read Cycle



Program Cycle Limits – X84F128/064

Symbol	Parameter	V_{CC} = 5V ±10%		V_{CC} = 3V ± 20%		Units
		Min.	Max.	Min.	Max.	
t_{NVPC}(5)	NonVolatile Program Cycle Time		5		5	ms
t_{CYC}	Cycle Time	200	10,000	330	10,000	ns
t_{WP}	$\overline{\text{WE}}$ Pulse Width	30		45		ns
t_{WPH}	$\overline{\text{WE}}$ HIGH Recovery Time	170		255		ns
t_{CS}	Program Setup Time	0		0		ns
t_{CH}	Program Hold Time	0		0		ns
t_{CP}	$\overline{\text{CE}}$ Pulse Width	30		45		ns
t_{CPH}	$\overline{\text{CE}}$ HIGH Recovery Time	170		255		ns
t_{OES}	$\overline{\text{OE}}$ HIGH Setup Time	25		37		ns
t_{OEH}	$\overline{\text{OE}}$ HIGH Hold Time	25		37		ns
t_{DS}(6)	Data Setup Time	30		45		ns
t_{DH}(6)	Data Hold Time	5		5		ns
t_{PPCS}(7)	$\overline{\text{PP}}$ HIGH Before $\overline{\text{CE}}$	100		150		ns
t_{PPCH}(7)	$\overline{\text{PP}}$ HIGH After $\overline{\text{CE}}$	100		150		ns

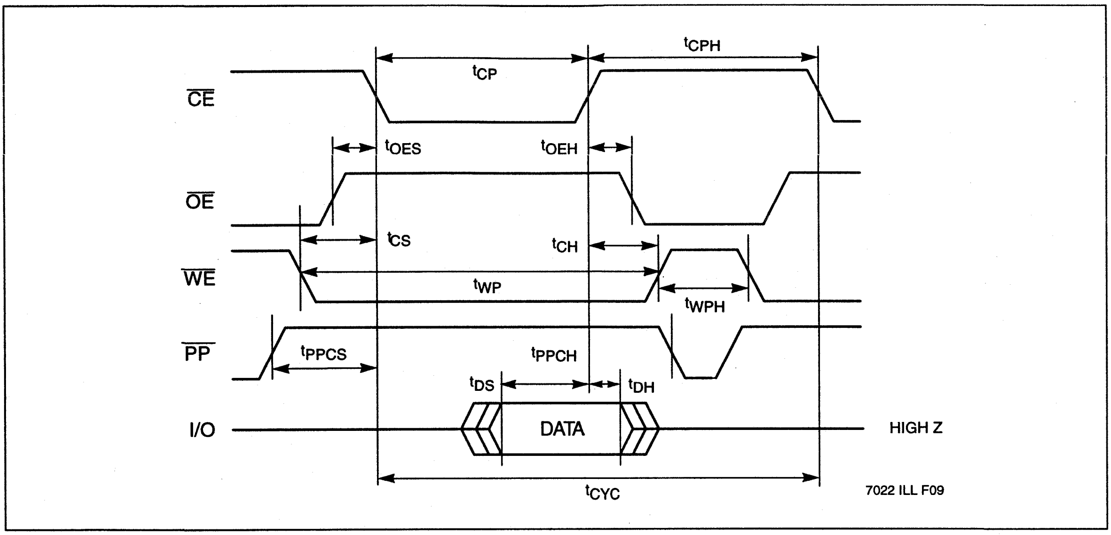
7022 FRM T13

Notes: (5) t_{NVPC} is the time from the falling edge of $\overline{\text{OE}}$ or $\overline{\text{CE}}$ (whichever occurs last) of the second read cycle in the "start nonvolatile program cycle" sequence until the self-timed, internal nonvolatile program cycle is completed.

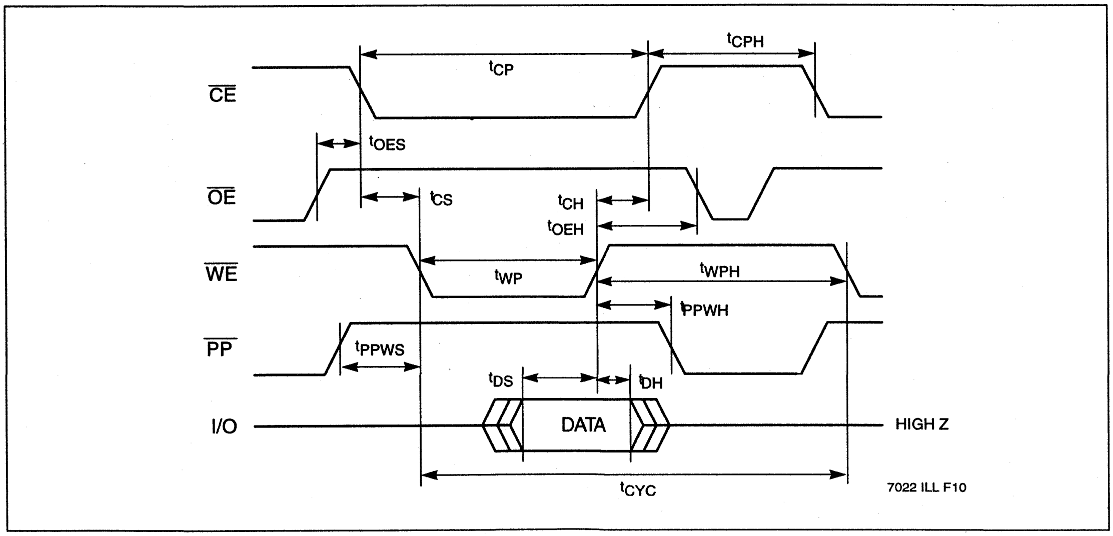
(6) Data is latched into the X84F128/064 on the rising edge of $\overline{\text{CE}}$ or $\overline{\text{WE}}$, whichever occurs first.

(7) Periodically sampled, but not 100% tested.

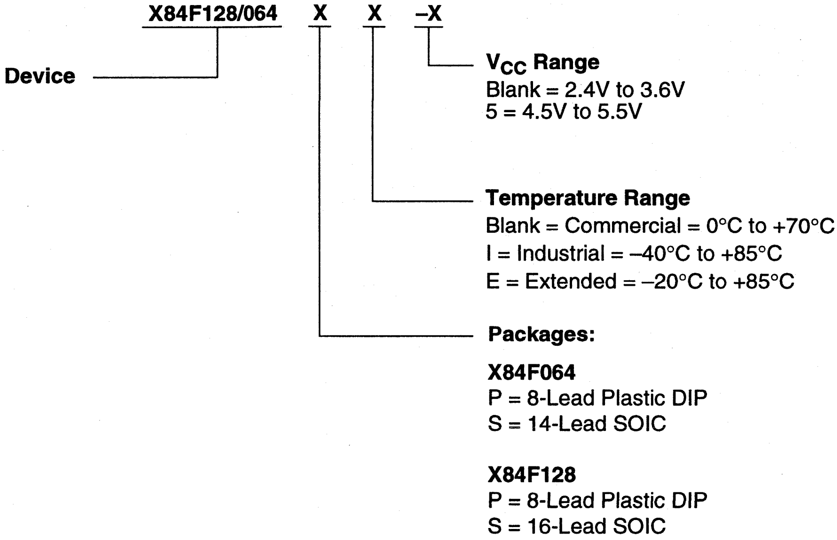
$\overline{\text{CE}}$ Controlled Program Cycle



$\overline{\text{WE}}$ Controlled Program Cycle



ORDERING INFORMATION



X84F128/064 X X –X

Device

V$_{CC}$ Range
Blank = 2.4V to 3.6V
5 = 4.5V to 5.5V

Temperature Range
Blank = Commercial = 0°C to +70°C
I = Industrial = –40°C to +85°C
E = Extended = –20°C to +85°C

Packages:

X84F064
P = 8-Lead Plastic DIP
S = 14-Lead SOIC

X84F128
P = 8-Lead Plastic DIP
S = 16-Lead SOIC

LIMITED WARRANTY

Devices sold by Xicor, Inc. are covered by the warranty and patent indemnification provisions appearing in its Terms of Sale only. Xicor, Inc. makes no warranty, express, statutory, implied, or by description regarding the information set forth herein or regarding the freedom of the described devices from patent infringement. Xicor, Inc. makes no warranty of merchantability or fitness for any purpose. Xicor, Inc. reserves the right to discontinue production and change specifications and prices at any time and without notice.

Xicor, Inc. assumes no responsibility for the use of any circuitry other than circuitry embodied in a Xicor, Inc. product. No other circuits, patents, licenses are implied.

U.S. PATENTS

Xicor products are covered by one or more of the following U.S. Patents: 4,263,664; 4,274,012; 4,300,212; 4,314,265; 4,326,134; 4,393,481; 4,404,475; 4,450,402; 4,486,769; 4,488,060; 4,520,461; 4,533,846; 4,599,706; 4,617,652; 4,668,932; 4,752,912; 4,829, 482; 4,874, 967; 4,883, 976. Foreign patents and additional patents pending.

LIFE RELATED POLICY

In situations where semiconductor component failure may endanger life, system designers using this product should design the system with appropriate error detection and correction, redundancy and back-up features to prevent such an occurence.

Xicor's products are not authorized for use in critical components in life support devices or systems.

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform, when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.
2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

16K/8K X84F016/008 MPS™ Serial Flash

Micro Port Saver SerialFlash with Block Lock™ Protection

FEATURES

- Direct Interface to Micros
  - —Eliminates I/O port requirements
  - —No interface glue logic required
  - —Eliminates need for parallel to serial converters
- New Programmable Block Lock™ Protection
  - —Software Program Protection
  - —Programmable Hardware Program Protection
- Block Lock (0, 1/4, 1/2, or all of the array)
- Up to 5Mbps data transfer rate
- Low Power CMOS
  - —3V or 5V "Univolt" Read and Program Power Supply Versions
  - —Standby Current Less than 1µA
  - —Active Current Less than 1mA
- Minimum 45ns Read Access Time
- 256-Bit Sector Program Mode
- Typical Nonvolatile Program Cycle Time: 5ms
- High Reliability
  - —100,000 Endurance Cycles
  - —Data Retention: 100 Years

DESCRIPTION

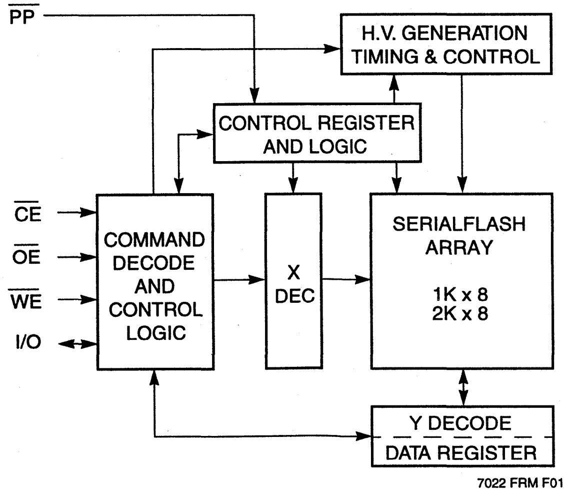
The X84F016/008 Micro Port SerialFlash devices are 16K/8K-bit CMOS memories designed for direct interface to port limited microcontroller or I/O limited ASIC and microprocessor designs. The MPS interface eliminates the need for parallel to serial conversion hardware simplifying system design.

The X84F016/008 provide all of the benefits of serial memories, such as low cost, low power, low voltage operation, and small package size, while featuring higher data transfer rates and reduced interface code requirements—without the need for a dedicated serial bus. All of the products are organized into 256 bit sectors and are suitable in 8-bit, 16-bit, 32-bit, or 64-bit environments, due to the bit serial nature of the interface.

The X84F016/008 devices directly connect to the system bus and communicate over a single data line using a sequence of standard bus read and write operations. This eliminates the need for dedicated port pins, parallel to serial converters, complicated ASIC implementations, or other glue logic, lowering system cost.

The X84F016/008 provides additional data security features through Block Lock and programmable Hardware Program Protection. These allow some or all of the array to be program protected by software command or by hardware. System Configuration, Company ID, calibration information, or other critical data can be secured a gainst unexpected or inadvertent program operations, leaving the remainder of the memory available for the system or user access.

BLOCK DIAGRAM



7022 FRM F01

Xicor E^{2}PROMs are designed and tested for applications requiring extended endurance. Inherent data retention is greater than 100 years.

PIN DESCRIPTIONS

Chip Enable ($\overline{CE}$)

The Chip Enable input must be LOW to enable all read/write operations. When $\overline{CE}$ is HIGH, the chip is deselected, the I/O pin is in the high impedance state, and unless a nonvolatile program operation is underway, the device is in the standby power mode.

Output Enable ($\overline{OE}$)

The Output Enable input must be LOW to enable the output buffer and to read data from the device on the I/O line.

Write Enable ($\overline{WE}$)

The Write Enable input must be LOW to write either data or command sequences to the device.

Data In/Data Out (I/O)

Data and command sequences are serially written to or serially read from the device through the I/O pin.

Program Protect ($\overline{PP}$)

The Program Protect input controls the Hardware Program Protect feature. When $\overline{PP}$ is LOW and the nonvolatile bit PPEN is "1", nonvolatile programming of the X84F016/008 control register is disabled, but the part otherwise functions normally. When $\overline{PP}$ is held HIGH, all functions, including nonvolatile programming operate normally. $\overline{PP}$ going LOW while $\overline{CS}$ is still LOW will interrupt a program to the X84F016/008 control register. If the internal Program cycle has already been initiated, $\overline{PP}$ going LOW will have no effect on programming.

The $\overline{PP}$ pin function is blocked when the PPEN bit in the control register is "0". This allows the user to install the X84F016/008 in a system with $\overline{PP}$ pin grounded and still be able to program to the control register. The $\overline{PP}$ pin functions will be enabled when the PPEN bit is set "1".

PIN NAMES

I/O	Data Input/Output
$\overline{CE}$	Chip Enable Input
$\overline{OE}$	Output Enable Input
$\overline{WE}$	Write Enable Input
$\overline{PP}$	Program Protect Input
V_{CC}	Supply Voltage
V_{SS}	Ground
NC	No Connect

7022 FRM T01

PINOUT/PACKAGE SELECTION GUIDE

X84F008	8-Lead Dip (P) 8-Lead SOIC (S) 14-Lead TSSOP (V)
X84F016	8-Lead Dip (P) 8-Lead SOIC (S) 14-Lead TSSOP (V)

7022 FRM T0A

PIN CONFIGURATIONS

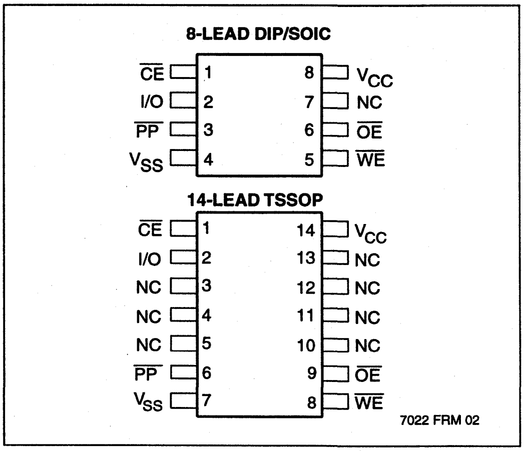
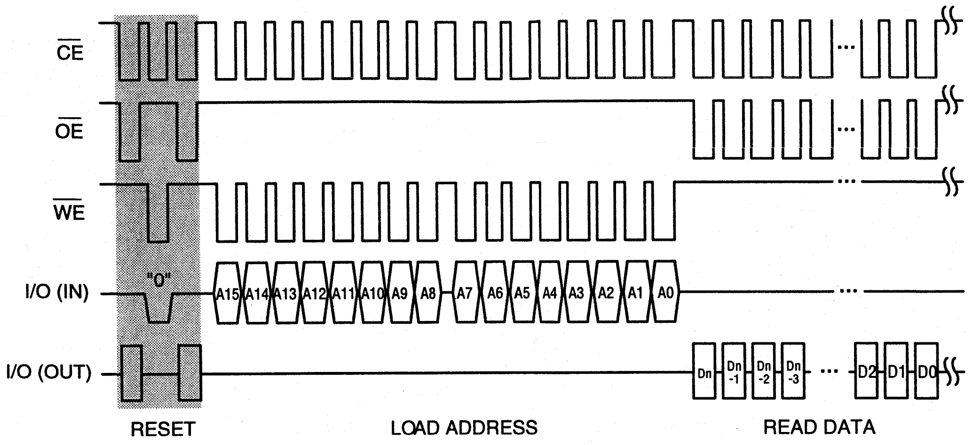


Figure 1. Read Sequence



WHEN ACCESSING: X84F008 ARRAY: A15–A10=0
X84F016 ARRAY: A15–A11=0

7022 FRM 03

DEVICE OPERATION

The X84F016/008 are SerialFlash devices designed to interface directly with most microprocessor buses. Standard $\overline{CE}$, $\overline{OE}$, and $\overline{WE}$ signals control the read and program operations, and a single I/O line is used to send and receive data and commands serially.

Data Timing

Data input on the I/O line is latched on the rising edge of either $\overline{WE}$ or $\overline{CE}$, whichever occurs first. Data output on the I/O line is active whenever both $\overline{OE}$ and $\overline{CE}$ are LOW. Care should be taken to ensure that $\overline{WE}$ and $\overline{OE}$ are never both LOW while $\overline{CE}$ is LOW.

Read Sequence

A read sequence consists of sending a 16-bit address followed by the reading of data serially. The address is written by issuing 16 separate write cycles ($\overline{WE}$ and $\overline{CE}$ LOW, $\overline{OE}$ HIGH) to the part without a read cycle between the write cycles. The address is sent serially, most significant bit first, over the I/O line. Note that this sequence is fully static, with no special timing restrictions, and the processor is free to perform other tasks on the bus whenever the device $\overline{CE}$ pin is HIGH. Once the 16 address bits are sent, a byte of data can be read on the I/O line by issuing 8 separate read cycles ($\overline{OE}$ and $\overline{CE}$ LOW, $\overline{WE}$ HIGH). At this point, writing a '1' will terminate the read sequence and enter the low power standby state, otherwise the device will await further reads in the sequential read mode.

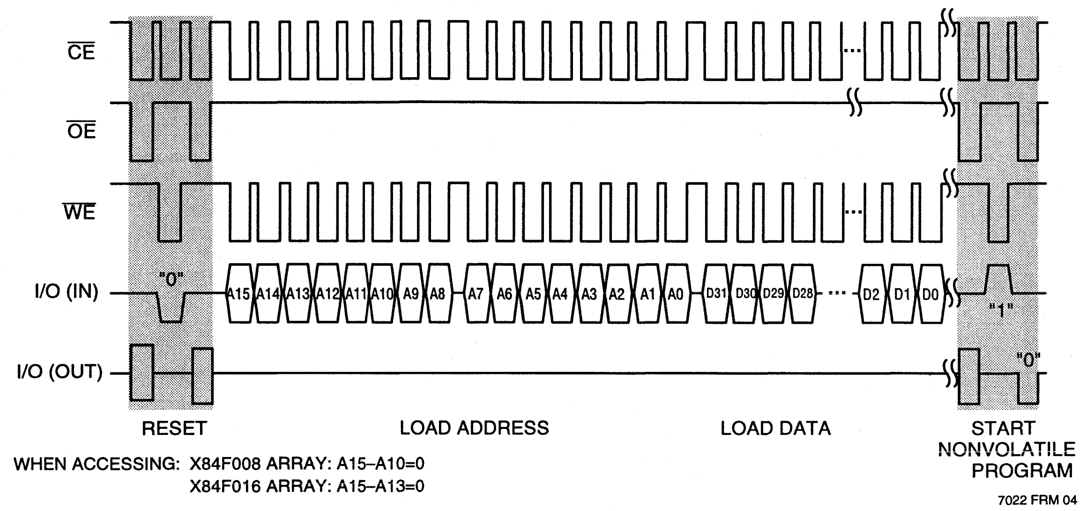
Sequential Read

The bit address is automatically incremented to the next higher address after each bit of data is read. The data stored in the memory at the next address can be read sequentially by continuing to issue read cycles. When the highest address in the array is reached, the address counter rolls over to address $000 and reading may be continued indefinitely.

Reset Sequence

The reset sequence resets the device and sets an internal Program enable latch. A reset sequence can be sent at any time by performing a read/write "0"/read operation (see Figs. 1 and 2). This breaks the multiple read or Program cycle sequences that are normally used to read from or program the part. The reset sequence can be used at any time to interrupt or end a sequential read or page load. As soon as the write "0" cycle is complete, the part is reset (unless a nonvolatile program cycle is in progress). The second read cycle in this sequence, and any further read cycles, will read a HIGH on the I/O pin until a valid read sequence (which includes the address) is issued. The reset sequence must be issued at the beginning of both read and program sequences to be sure the device initiates these operations properly.

Figure 2: Program Sequence


CE
OE
WE
I/O (IN)
I/O (OUT)
"0"
A15 A14 A13 A12 A11 A10 A9 A8 A7 A6 A5 A4 A3 A2 A1 A0
D31 D30 D29 D28 ··· D2 D1 D0
"1"
"0"
RESET
LOAD ADDRESS
LOAD DATA
START NONVOLATILE PROGRAM
WHEN ACCESSING: X84F008 ARRAY: A15–A10=0
X84F016 ARRAY: A15–A13=0
7022 FRM 04

Program Sequence

A nonvolatile program sequence consists of sending a reset sequence, the 16-bit address of the first location in a sector, 256-bits bytes of data, and then a special “start nonvolatile program cycle” command sequence. The reset sequence is issued first (as described in the Reset Sequence section) to set an internal program enable latch. The address is written serially by issuing 16 separate write cycles ($\overline{WE}$ and $\overline{CE}$ LOW, $\overline{OE}$ HIGH) to the part without any read cycles between the writes.The address is sent serially, most significant bit first,on the I/O pin. The 256-bits of data are programmed by issuing 256 separate write cycles. Again, no read cycles are allowed between writes and a full 256-bit sector must be programmed.

The nonvolatile program cycle is initiated by issuing a special read/write “1”/read sequence. The first read cycle ends the sector load, then the write “1” followed by a read starts the nonvolatile program cycle. The device recognizes 256-bit sectors beginning at addresses XXXX00000. When sending data to the part, attempts to exceed the upper address of the page will result in undefined data being programmed in the array.

A nonvolatile program cycle will not start if a partial or incomplete program sequence is issued. The internal program enable latch is reset when the nonvolatile program cycle is completed to prevent inadvertent programming.

Nonvolatile Program Status

The status of a nonvolatile program cycle can be determined at any time by simply reading the state of the I/O pin on the device. This pin is read when $\overline{OE}$ and $\overline{CE}$ are LOW and $\overline{WE}$ is HIGH. During a nonvolatile program cycle the I/O pin is LOW. When the nonvolatile program cycle is complete, the I/O pin goes HIGH. A reset sequence can also be issued during a nonvolatile program cycle with the same result: I/O is LOW as long as a nonvolatile program cycle is in progress, and I/O is HIGH when the nonvolatile program cycle is done.

CONTROL REGISTER

The X84F016/008 has one register that contains control bits for the devices. The control bits, PPEN, BP1, and BP0, are shown in Table 1. To read or change the contents of this register requires a one byte operation to address FFFFh.

A read from FFFFh returns the one byte contents of the control register unused bits return 0. Continued reads return undefined data. Programming address FFFFh changes the value of the bits. Unused bits are programmed as "0". Writing more than one byte to the control register is a violation and the operation will be aborted. After sending one byte to the control register, a start nonvolatile program cycle will latch in the new state.

Table 1

7	6	5	4	3	2	1	0
PPEN	0	0	0	BP1	BP0	0	0

7022 FRM T02

PPEN: Program Protect Enable Bit

The Program-Protect-Enable (PPEN) bit is an enable bit for the $\overline{PP}$ pin.

Table Table 2

PPEN	$\overline{PP}$	Protected Blocks	Unprotected Blocks	Control Register
0	X	Protected	Programmable	Programmable
1	LOW	Protected	Programmable	Protected
X	HIGH	Protected	Programmable	Programmable

7022 FRM T03.

The Program Protect ($\overline{PP}$) pin and the nonvolatile Program Protect Enable (PPEN) bit in the Control Register control the programmable hardware program protect feature. Hardware program protection is enabled when $\overline{PP}$ pin is LOW, and the PPEN bit is "1". Hardware program protection is disabled when either the $\overline{PP}$ pin is HIGH or the PPEN bit is "0". When the chip is hardware program protected, nonvolatile programming is disabled to the Control Register, including the Block Protect bits and the PPEN bit itself, as well as the block-protected sections in the memory array. Only the sections of the memory array that are not block-protected can be programmed.

Note: When the $\overline{PP}$ pin is tied to V_{SS} and the PPEN bit is HIGH, the PPEN bit is Program protected. It cannot be changed back to a "0", as long as the $\overline{PP}$ pin is held LOW.

BP1, BP0: Block Protect Bits

The Block Protect (BP0 and BP1) bits are nonvolatile and allow the user to select one of four levels of protection. The X84F016/008 is divided into four segments. One, two, or all four of the segments may be protected. That is, the user may read the segments but will be unable to alter (program) data within the selected segments. The partitioning is controlled as illustrated in table 3 below.

Table 3. Block Lock Protection

Control Register Bits		Array Address Protected		
BP1	BP0	X84F008	X84F016	
0	0	None	None	
0	1	0300h–03FFh	0600h–07FFh	upper 1/4
1	0	0200h–03FFh	0400h–07FFh	upper 1/2
1	1	0000–03FFh	0000–07FFh	Full Array (Not including the control register.)

7022 FRM T04

Low Power Operation

The device enters an idle state, which draws minimal current when:

—an illegal sequence is entered. The following are the more common illegal sequences:

- Read/Write/Write—any time
- Read/Write '1'—When writing the address or writing data.
- Write '1'—when reading data
- Read/Read/Write '1'—after data is written to device, but before entering the NV program sequence.

—the device powers-up;

—a nonvolatile program operation completes.

While a sequential read is in progress, the device remains in an active state. This state draws more current than the idle state, but not as much as during a read itself. To go back to the lowest power condition, an invalid condition is created by writing a '1' after the last bit of a read operation.

Program Protection

The following circuitry has been included to prevent inadvertent nonvolatile programming:

—The internal program enable latch is reset upon power-up.

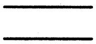
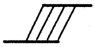
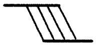
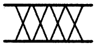
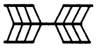
—A reset sequence must be issued to set the internal program enable latch before starting a program sequence.

—A special "start nonvolatile program" command sequence is required to start a nonvolatile program cycle.

—The internal program enable latch is reset automatically at the end of a nonvolatile program cycle.

—The internal program Enable latch is reset and remains reset as long as the $\overline{PP}$ pin is LOW, which blocks all nonvolatile program cycles.

SYMBOL TABLE

WAVEFORM	INPUTS	OUTPUTS
	Must be steady	Will be steady
	May change from LOW to HIGH	Will change from LOW to HIGH
	May change from HIGH to LOW	Will change from HIGH to LOW
	Don't Care: Changes Allowed	Changing: State Not Known
	N/A	Center Line is High Impedance

7022 FRM 05

ABSOLUTE MAXIMUM RATINGS*

Temperature under Bias	–65°C to +135°C
Storage Temperature	–65°C to +150°C
Terminal Voltage with Respect to V_{SS}	–1V to +7V
DC Output Current	5mA
Lead Temperature (Soldering, 10 seconds)	300°C

*COMMENT

Stresses above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and the functional operation of the device at these or any other conditions above those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

RECOMMENDED OPERATING CONDITIONS

Temperature	Min.	Max.
Commercial	0°C	+70°C
Extended	–20°C	+85°C
Industrial	-40°C	+85°C

7022 FRM T05

Supply Voltage	Limits
X84F016/008	5V ±10%
X84F016/008	3V to ± 20%

7022 FRM T06

D.C. OPERATING CHARACTERISTICS: (VCC = 5V ±10%)

(Over the recommended operating conditions, unless otherwise specified.)

Symbol	Parameter	Limits		Units	Test Conditions
		Min.	Max.		
I_{CC1}	V_{CC} Supply Current (Read)		1	mA	$\overline{OE} = V_{IL}$, $\overline{WE} = V_{IH}$, I/O = Open, $\overline{CE}$ clocking @ 5MHz
I_{CC2}	V_{CC} Supply Current (Program)		3	mA	I_{CC} During Nonvolatile Program Cycle All Inputs at CMOS Levels
I_{SB1}(2)	V_{CC} Standby Current		1	µA	$\overline{CE} = V_{CC}$, Other Inputs = V_{CC} or V_{SS}
I_{LI}	Input Leakage Current		10	µA	$V_{IN} = V_{SS}$ to V_{CC}
I_{LO}	Output Leakage Current		10	µA	$V_{OUT} = V_{SS}$ to V_{CC}
V_{IL} (1)	Input LOW Voltage	–1	V_{CC} x 0.3	V	
V_{IH} (1)	Input HIGH Voltage	V_{CC} x 0.7	V_{CC} + 0.5	V	
V_{OL}	Output LOW Voltage		0.4	V	I_{OL} = 2.1mA
V_{OH}	Output HIGH Voltage	V_{CC} – 0.8		V	I_{OH} = –1mA

7022 FRM T07

Notes: (1) V_{IL} Min. and V_{IH} Max. are for reference only and are not tested.

(2) This parameter is periodically sampled and not 100% tested.

D.C. OPERATING CHARACTERISTICS (V_{CC} = 3V ± 20%)

(Over the recommended operating conditions, unless otherwise specified.)

Symbol	Parameter	Limits		Units	Test Conditions
		Min.	Max.		
I_{CC1}	V_{CC} Supply Current (Read)		500	µA	$\overline{OE}$ = V_{IL}, $\overline{WE}$ = V_{IH}, I/O = Open, $\overline{CE}$ clocking @ 3MHz
I_{CC2}	V_{CC} Supply Current (Program)		2	mA	I_{CC} During Nonvolatile Program Cycle All Inputs at CMOS Levels
I_{SB1}(2)	V_{CC} Standby Current		1	µA	$\overline{CE}$ = V_{CC}, Other Inputs = V_{CC} or V_{SS}
I_{LI}	Input Leakage Current		10	µA	V_{IN} = V_{SS} to V_{CC}
I_{LO}	Output Leakage Current		10	µA	V_{OUT} = V_{SS} to V_{CC}
V_{IL}(1)	Input LOW Voltage	–1	V_{CC} x 0.3	V	
V_{IH}(1)	Input HIGH Voltage	V_{CC} x 0.7	V_{CC} + 0.5	V	
V_{OL}	Output LOW Voltage		0.4	V	I_{OL} = 1mA
V_{OH}	Output HIGH Voltage	V_{CC} – 0.4		V	I_{OH} = –400µA

7022 FRM T08

CAPACITANCE T_A = +25°C, f = 1MHz, V_{CC} = 5V

Symbol	Parameter	Max.	Units	Test Conditions
$C_{I/O}$(2)	Input/Output Capacitance	8	pF	$V_{I/O}$ = 0V
C_{IN}(2)	Input Capacitance	6	pF	V_{IN} = 0V

7022 FRM T09

POWER-UP TIMING

Symbol	Parameter	Max.	Units
t_{PUR}(3)	Power-up to Read Operation	1	ms
t_{PUW}(3)	Power-up to Program Operation	1	ms

7022 FRM T10

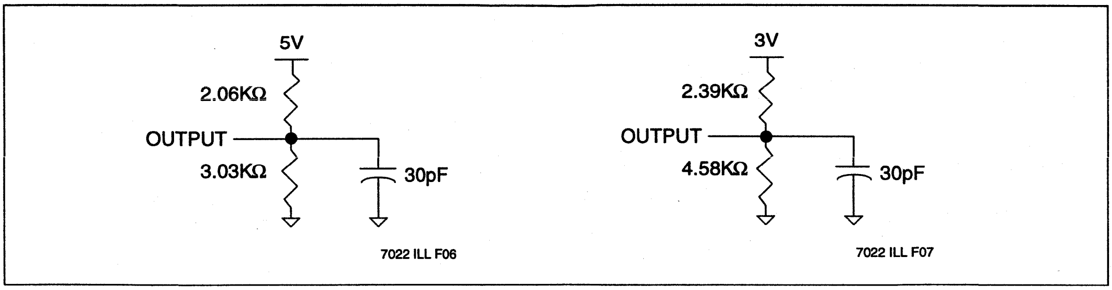
A.C. CONDITIONS OF TEST

Input Pulse Levels	V_{CC} x 0.1 to V_{CC} x 0.9
Input Rise and Fall Times	10ns
Input and Output Timing Levels	V_{CC} x 0.5

7022 FRM T11

Notes: (3) Time delays required from the time the V_{CC} is stable until the specific operation can be initiated. Periodically sampled, and not 100% tested.

EQUIVALENT A.C. LOAD CIRCUITS



A.C. CHARACTERISTICS

(Over the recommended operating conditions, unless otherwise specified.)

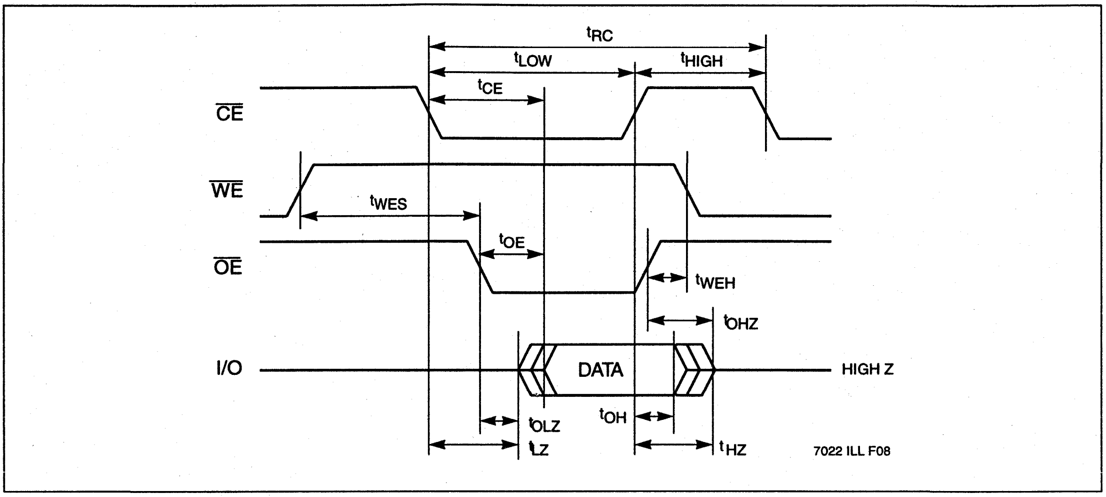
Read Cycle Limits – X84F016/008

Symbol	Parameter	V_{CC} = 5V ±10%		V_{CC} = 3V ± 20%		Units
		Min.	**Max.**	**Min.**	**Max.**	
t_{RC}	Read Cycle Time	200		330		ns
t_{CE}	$\overline{CE}$ Access Time		45		120	ns
t_{OE}	$\overline{OE}$ Access Time		45		120	ns
t_{LOW}	$\overline{CE}$ LOW Time	70		150		ns
t_{HIGH}	$\overline{CE}$ HIGH Time	70		150		ns
t_{LZ}(4)	$\overline{CE}$ LOW to Output In Low Z	0		0		ns
t_{HZ}(4)	$\overline{CE}$ HIGH to Output In High Z	0	30	0	45	ns
t_{OLZ}(4)	$\overline{OE}$ LOW to Output In Low Z	0		0		ns
t_{OHZ}(4)	$\overline{OE}$ HIGH to Output In High Z	0	30	0	45	ns
t_{OH}	Output Hold from $\overline{CE}$ or $\overline{OE}$ HIGH	0		0		ns
t_{WES}	$\overline{WE}$ HIGH Setup Time	25		25		ns
t_{WEH}	$\overline{WE}$ HIGH Hold Time	25		25		ns

7022 FRM T12

Notes: (4) Periodically sampled, but not 100% tested. t_{HZ} and t_{OHZ} are measured from the point where $\overline{CE}$ or $\overline{OE}$ goes HIGH (whichever occurs first) to the time when I/O is no longer being driven into a 5pF load.

Read Cycle



Program Cycle Limits – X84F016/008

Symbol	Parameter	V_{CC} = 5V ±10%		V_{CC} = 3V ± 20%		Units
		Min.	Max.	Min.	Max.	
t_{NVPC}[5]	NonVolatile Program Cycle Time		5		5	ms
t_{CYC}	Cycle Time	200	10,000	330	10,000	ns
t_{WP}	$\overline{\text{WE}}$ Pulse Width	30		45		ns
t_{WPH}	$\overline{\text{WE}}$ HIGH Recovery Time	170		255		ns
t_{CS}	Program Setup Time	0		0		ns
t_{CH}	Program Hold Time	0		0		ns
t_{CP}	$\overline{\text{CE}}$ Pulse Width	30		45		ns
t_{CPH}	$\overline{\text{CE}}$ HIGH Recovery Time	170		255		ns
t_{OES}	$\overline{\text{OE}}$ HIGH Setup Time	25		37		ns
t_{OEH}	$\overline{\text{OE}}$ HIGH Hold Time	25		37		ns
t_{DS}[6]	Data Setup Time	30		45		ns
t_{DH}[6]	Data Hold Time	5		5		ns
t_{PPCS}[7]	$\overline{\text{PP}}$ HIGH Before $\overline{\text{CE}}$	100		150		ns
t_{PPCH}[7]	$\overline{\text{PP}}$ HIGH After $\overline{\text{CE}}$	100		150		ns

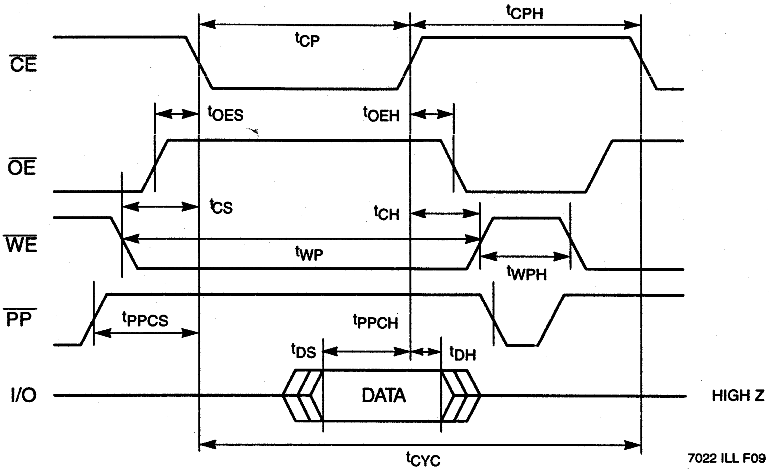
7022 FRM T13

Notes: (5) t_{NVPC} is the time from the falling edge of $\overline{\text{OE}}$ or $\overline{\text{CE}}$ (whichever occurs last) of the second read cycle in the "start nonvolatile program cycle" sequence until the self-timed, internal nonvolatile program cycle is completed.

(6) Data is latched into the X84F016/008 on the rising edge of $\overline{\text{CE}}$ or $\overline{\text{WE}}$, whichever occurs first.

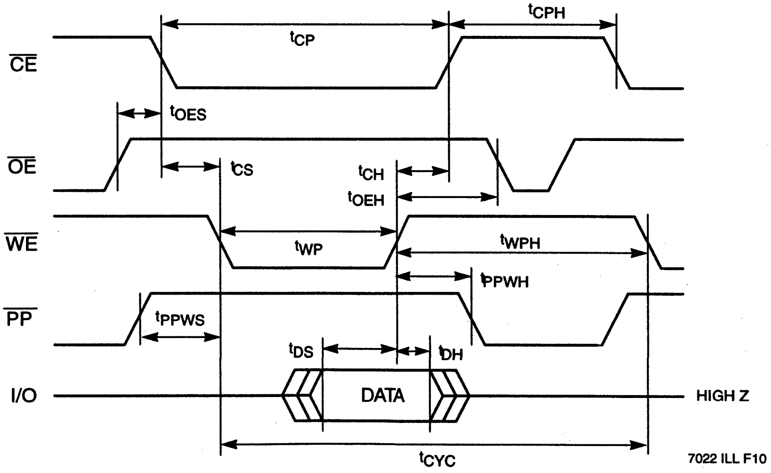
(7) Periodically sampled, but not 100% tested.

$\overline{\text{CE}}$ Controlled Program Cycle



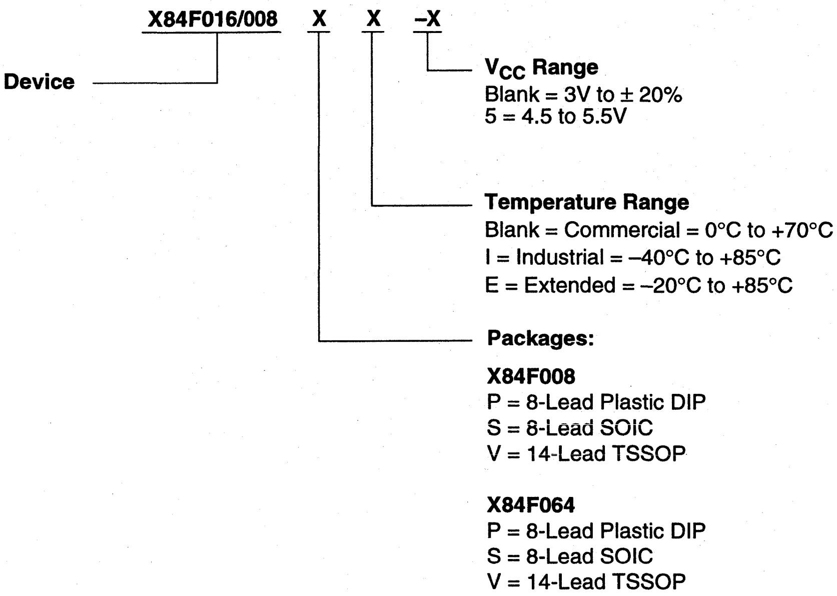
7022 ILL F09

$\overline{\text{WE}}$ Controlled Program Cycle



7022 ILL F10

ORDERING INFORMATION



X84F016/008 X X -X

Device

V_{CC} Range
Blank = 3V to ± 20%
5 = 4.5 to 5.5V

Temperature Range
Blank = Commercial = 0°C to +70°C
I = Industrial = –40°C to +85°C
E = Extended = –20°C to +85°C

Packages:

X84F008
P = 8-Lead Plastic DIP
S = 8-Lead SOIC
V = 14-Lead TSSOP

X84F064
P = 8-Lead Plastic DIP
S = 8-Lead SOIC
V = 14-Lead TSSOP

LIMITED WARRANTY

Devices sold by Xicor, Inc. are covered by the warranty and patent indemnification provisions appearing in its Terms of Sale only. Xicor, Inc. makes no warranty, express, statutory, implied, or by description regarding the information set forth herein or regarding the freedom of the described devices from patent infringement. Xicor, Inc. makes no warranty of merchantability or fitness for any purpose. Xicor, Inc. reserves the right to discontinue production and change specifications and prices at any time and without notice.

Xicor, Inc. assumes no responsibility for the use of any circuitry other than circuitry embodied in a Xicor, Inc. product. No other circuits, patents, licenses are implied.

U.S. PATENTS

Xicor products are covered by one or more of the following U.S. Patents: 4,263,664; 4,274,012; 4,300,212; 4,314,265; 4,326,134; 4,393,481; 4,404,475; 4,450,402; 4,486,769; 4,488,060; 4,520,461; 4,533,846; 4,599,706; 4,617,652; 4,668,932; 4,752,912; 4,829, 482; 4,874, 967; 4,883, 976. Foreign patents and additional patents pending.

LIFE RELATED POLICY

In situations where semiconductor component failure may endanger life, system designers using this product should design the system with appropriate error detection and correction, redundancy and back-up features to prevent such an occurence.

Xicor's products are not authorized for use in critical components in life support devices or systems.

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform, when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.
2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

Xicor®

Xicor®

XK76C

X76F041 Development System (Card Reader Version)

CONTENTS

- Card Reader with Interface Cable
- 3.5" Diskette, PC Format
  - —Executable Development System Software
  - —Source C Interface Routines
  - —Card Reader Schematic
- User Guide
- Two X76F041 Smartcards

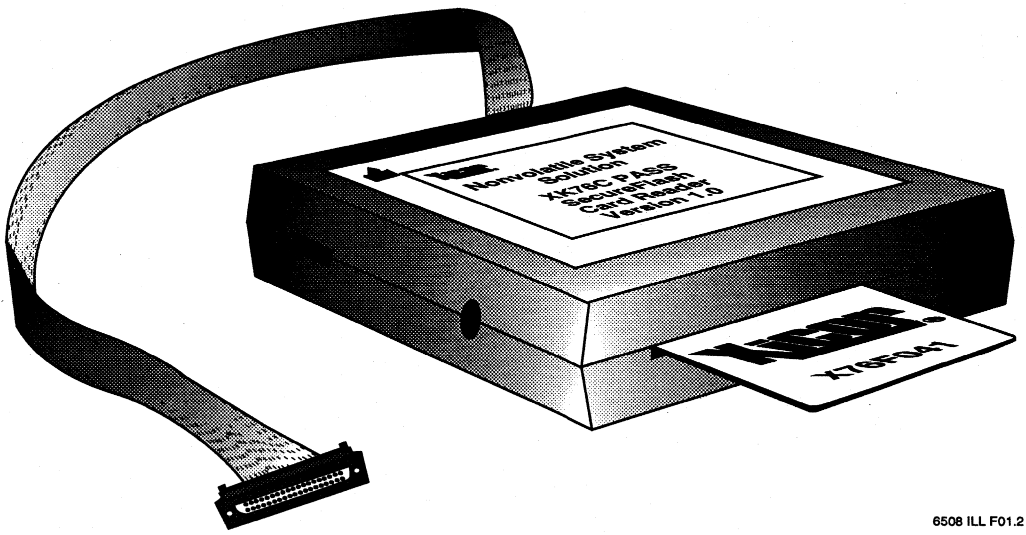
FEATURES

- Simple PC Based Platform
- Menu Driven User Interface
- Programming Editor for Memory Arrays
- Provides Complete Control of PASS™ SecureFlash Device Operation
  - —Password Entry and Programming
  - —Configuration Registers Programming
- Simplifies Design and Debug Process
- Supports Three Stages of Development

Evaluation—the System provides a comprehensive platform for evaluating the suitability of the X76F041 for a smartcard application. It allows for a complete understanding of the device in minutes not hours, by utilizing the card reader and graphical user interface software.

Appraisal—the Development System contains the hardware and generic software routines required to appraise the “smartcard solution”. It saves the associated costs (time and money) of developing small numbers of hardware units and writing device interface software for the application.

Prototype—the XK76C System simplifies design by supplying card reader schematic and application code. The Development System is invaluable during debug for presetting and verifying the X76F041 configuration and data.


Nonvolatile System Solution
XK76C PASS SecureFlash Card Reader Version 1.0
Xicor
X76F041

6508 ILL F01.2

PASS is a trademark of Xicor, Inc.

Characteristics subject to change without notice

DESCRIPTION

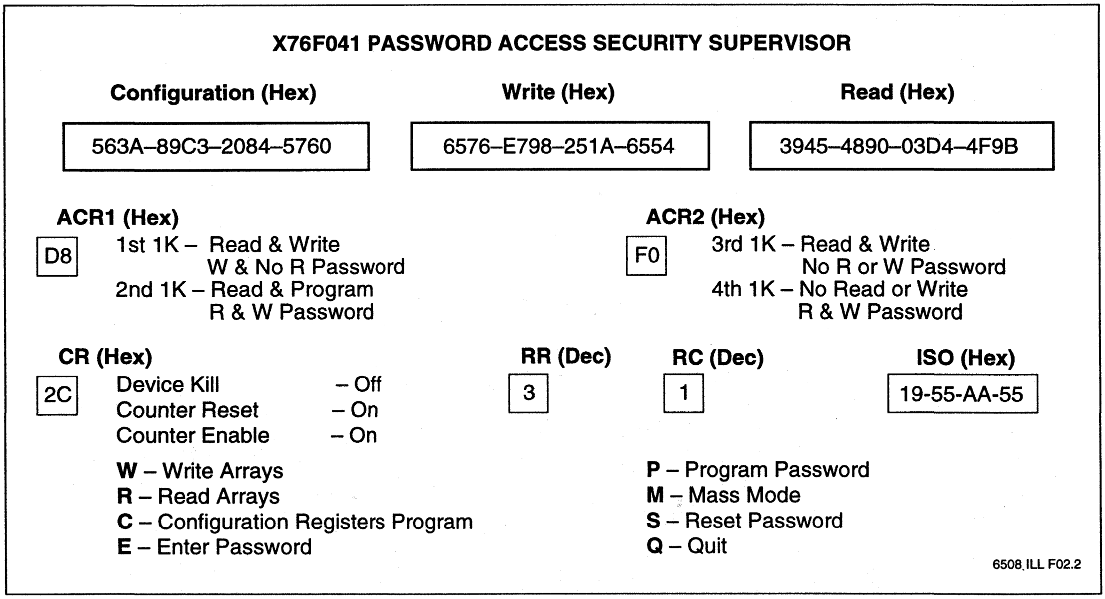
The XK76C Development System significantly reduces the time involved in understanding, evaluating and designing in the Xicor X76F041 smartcard. It provides comprehensive hardware and software support that includes a card reader and development system software.

The System is designed to enable users to quickly and easily understand the capabilities of the X76F041. It is an ideal platform for evaluating the suitability of the device for a particular application. The development system software, used to program and configure the device, enables various levels of security to be set and tested by attempting to access memory arrays using one or none of the passwords. The XK76C provides the hardware and application alterable software required to appraise the X76F041 smartcard solution, saving the designer from building an evaluation card reader and writing code from scratch. It also serves as an effective debug tool during prototype by providing a programming and verifying capability.

The status and values of the X76F041's configuration registers and four memory arrays are represented by the development software using two graphical interfaces. The MAIN interface is used to display the device's ISO identification number; enter, display, reset and program passwords; display and program configuration registers; and mass program/erase memory. The ARRAY interface is used to read the four memory arrays and to enter the array programming editor. The various options in each interface are displayed using menus and selected using a single key input.

The System consists of a card reader, two X76F041 smartcards, and a 3.5" software diskette. It is easy to use and simple to set up, operating with any IBM

Development System's MAIN interface screen format



X76F041 PASSWORD ACCESS SECURITY SUPERVISOR

Configuration (Hex)	Write (Hex)	Read (Hex)
563A–89C3–2084–5760	6576–E798–251A–6554	3945–4890–03D4–4F9B

ACR1 (Hex)

D8 1st 1K – Read & Write
W & No R Password
2nd 1K – Read & Program
R & W Password

ACR2 (Hex)

F0 3rd 1K – Read & Write
No R or W Password
4th 1K – No Read or Write
R & W Password

CR (Hex)

2C Device Kill – Off
Counter Reset – On
Counter Enable – On

RR (Dec)

3

RC (Dec)

1

ISO (Hex)

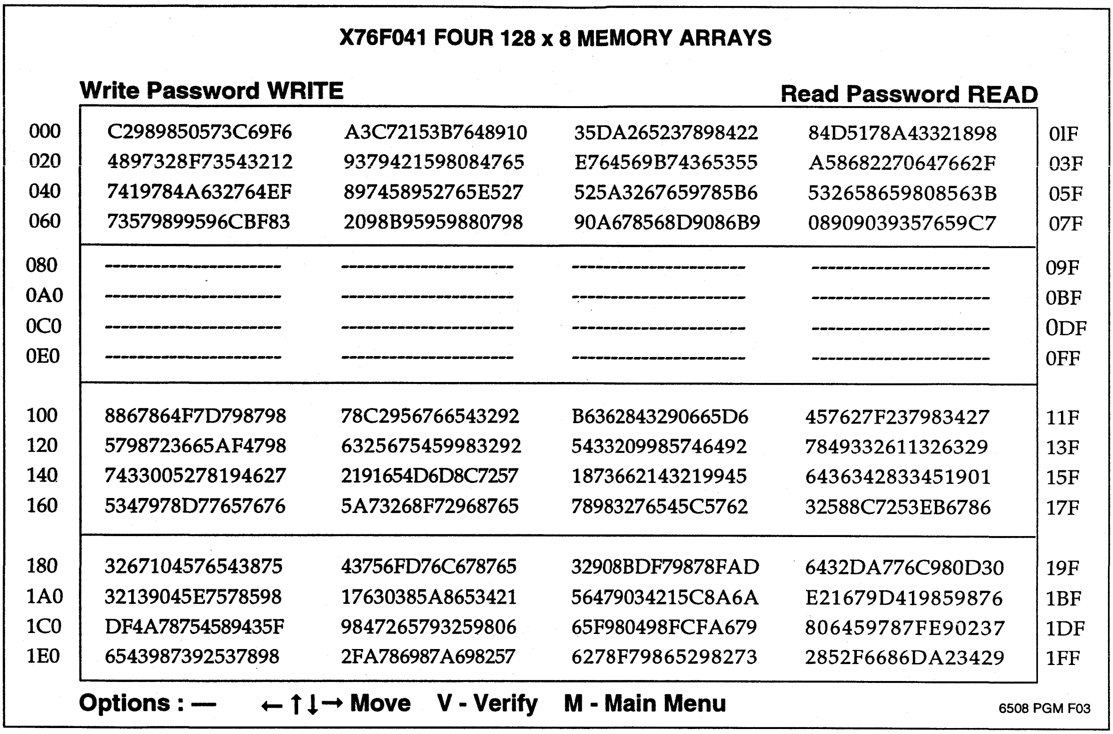
19-55-AA-55

W – Write Arrays
R – Read Arrays
C – Configuration Registers Program
E – Enter Password

P – Program Password
M – Mass Mode
S – Reset Password
Q – Quit

6508 ILL F02.2

Development System's ARRAY user interface screen format



```
                        X76F041 FOUR 128 x 8 MEMORY ARRAYS

      Write Password WRITE                                            Read Password READ
000   C2989850573C69F6    A3C72153B7648910    35DA265237898422    84D5178A43321898    0lF
020   4897328F73543212    9379421598084765    E764569B74365355    A58682270647662F    03F
040   7419784A632764EF    897458952765E527    525A3267659785B6    532658659808563B    05F
060   73579899596CBF83    2098B95959880798    90A678568D9086B9    08909039357659C7    07F

080   ----------------    ----------------    ----------------    ----------------    09F
0A0   ----------------    ----------------    ----------------    ----------------    0BF
0C0   ----------------    ----------------    ----------------    ----------------    0DF
0E0   ----------------    ----------------    ----------------    ----------------    0FF

100   8867864F7D798798    78C2956766543292    B6362843290665D6    457627F237983427    11F
120   5798723665AF4798    6325675459983292    5433209985746492    7849332611326329    13F
140   7433005278194627    2191654D6D8C7257    1873662143219945    6436342833451901    15F
160   5347978D77657676    5A73268F72968765    78983276545C5762    32588C7253EB6786    17F

180   3267104576543875    43756FD76C678765    32908BDF79878FAD    6432DA776C980D30    19F
1A0   32139045E7578598    17630385A8653421    56479034215C8A6A    E21679D419859876    1BF
1C0   DF4A78754589435F    9847265793259806    65F980498FCFA679    806459787FE90237    1DF
1E0   6543987392537898    2FA786987A698257    6278F79865298273    2852F6686DA23429    1FF

      Options : —     ← ↑ ↓ → Move    V - Verify    M - Main Menu
```

6508 PGM F03

compatible PC that is equipped with a parallel printer port. The hardware is powered using either a 9 volt battery or standard DC power plug. The software diskette contains three files: the development system executable file, the source C interface routines, and the OrCAD schematic of the card reader. The executable file can run with any PC supporting DOS 3.0 or higher.

NOTES

Xicor®

Programmer Vendors Supporting Xicor Devices

by Jordon Inkeles, July 1996

The following listing summarizes the names and phone numbers of programmer vendors who support Xicor devices. The complete Application Note AN54 provides a table cross-referencing Xicor parts with specific programmers. To obtain this version of Application Note AN54 please refer to our website at (www.xicor.com) or our FAX back system at (408) 954-1627. For information pertaining to a particular programmer, such as adapter or software revision, the programmer vendor should be contacted directly.

Advin Systems Inc.
1050 - L East Duane Ave.
Sunnyvale, CA 94086
(408) 243-7000

American Reliance Inc.
11801 Goldring Rd.
Arcadia, CA 91006
(818) 303-6688
http://www.amrel.com

BP Microsystems
1000 N. Post Oak Rd., Suite 225
Houston, TX 77055-7237
(713) 688-4600
http://www.bpmicro.com

Bytek Corporation
543 N.W. 77th St.
Boca Raton, FL 33487-1323
(407) 994-3520

Data I/O Corporation
10525 Willows Rd. N.E.
P.O. Box 97046
Redmond, WA 98073-9746
(800) 247-5700
http://www.data-io.com

EE Tools
544 Weddell Dr., Suite 6
Sunnyvale, CA 94089
(408) 734-8184

Link Instruments, Inc.
369 Passaic Ave, Suite 100
Fairfield, NJ 07004
(201) 808-8990
http://www.linkinstruments.com

Logical Devices, Inc.
130 Capital Drive, Ste. A & B
Golden, CO 80401
(303) 279-6868
http://www.logicaldevices.com

Needham's Electronics, Inc.
4630 Beloit Dr., Suite #20
Sacramento, CA 95838
(916) 924-8037
http://www.needhams.com

Stag Programmers Ltd.
Silver Court
Watchmead
Welwyn Garden City
Hert AL7 1LT, UK
Tel +44 1707 332148

System General Corporation
1603A S. Main St.
Milpitas, CA 95035
(408) 263-6667

Tribal Microsystems Inc.
aka **Hi-Lo Systems Research Co., Ltd.**
44388 S. Grimmer Blvd.
Fremont, CA 94538
(510) 623-8859

Xeltek
3563 Ryder Street
Santa Clara, CA 95051-0707
(408) 524-1929
http://www.xeltek.com

This page left intentionally blank.

Xicor® Application Note AN75

Interface Software for Xicor SPI Serial Memories

by Gray Creager, August 1995

The following code is intended to provide generic C routines that can be adapted for use with a number of Xicor SPI serial E^{2}PROMs and SPI SerialFlash memory devices. This code was written for and can be used "as-is" with the X25080, X25160, X25320, X25640, X25642, or X25128 serial SPI E^{2}PROMs. With a slight modification, these routines are compatible with the X25F008, X25F016, X25F032, X25F064 and X25F128 SPI SerialFlash memories as well. Figure 1 shows the circuit used to test and debug this code from the Centronics parallel printer port on a generic PC.

When using this code with SPI serial E^{2}PROMs, a single instruction sequence allows for any number of bytes to be read or up to 32 bytes to be written on a page before wrap-around on that page. With SPI SerialFlash memories, any number of bytes can be read in a single sequence as well, however, entire sectors of 32 bytes must be written when writing to these devices. This requires that all routine calls for WRITE_SPI be made in the following manner:

WRITE_SPI(32,addr,&array)

Otherwise, the code is entirely compatible, except for some changes in nomenclature (e.g. the WEL bit becomes the PEL bit, etc.). The circuit shown in Figure 1 could also be used with SerialFlash memories, providing that the limits for V_{CC}, V_{IH}, and V_{OL} are observed. This code can be obtained from Xicor's BBS, which can be reached toll free at 1-800-258-8864, or in the (408) area code and internationally at 1-408-943-0655. Xicor's BBS will support up to a 19.2K baud rate modem (no parity, 8 bit words, 1 stop bit, and no local echo). These listings can be found in the Motorola SIG (Special Interest Group). Xicor application notes are also available through Xicor's FaxBack system at (408) 954-1627 and on the World Wide Web (at the URL, http://www.xicor.com).

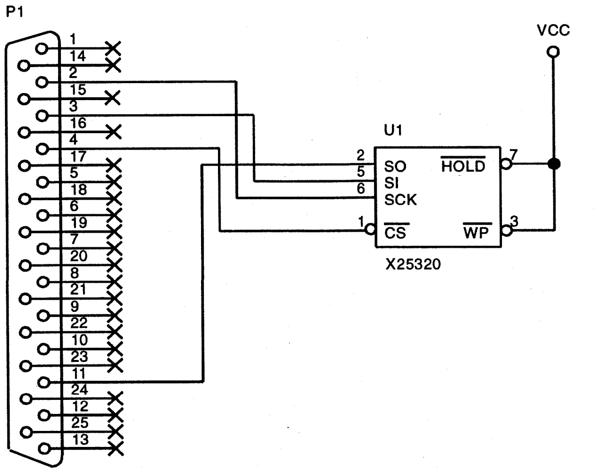


Figure 1 - Hardware connection used to verify this code with an X25320 serial E^{2}PROM.

```
/*********************************************************************/
/*                                                                   */
/* Turbo C code for interfacing Xicor second generation SPI serial   */
/* EEPROMs to the Centronics parallel printer port. These devices    */
/* include the X25080, X25160, X25320, X25640, X25642, and X25128.   */
/* SCK is connected to D0, SI to D1, SO to BUSY, /CS to D2, and both */
/* /HOLD and /WP are assumed HIGH.                                   */
/*                                                                   */
/* With slight modifications, this code can also be used with the    */
/* X25F008, X25F016, X25F032, X25F064, and X25F128 SerialFlash ICs.  */
/*                                                                   */
/*********************************************************************/

#include <stdio.h>
#include <stdlib.h>

int data_port = 0x0378;       /* typical output printer port */
int status_port = 0x0379;     /* typical input printer port */

unsigned char mask = 0xFC;    /* default bits X X X X X CS SI SCK */

void SCK_HIGH() {
      mask = mask | 0x01;           /* forces SCK pin HIGH */
      outportb(data_port,mask);
}

void SCK_LOW() {
      mask = mask & 0xFE;           /* forces SCK pin LOW */
      outportb(data_port,mask);
}

void SI_HIGH() {
      mask = mask | 0x02;           /* forces SI pin HIGH */
      outportb(data_port,mask);
}

void SI_LOW() {
      mask = mask & 0xFD;           /* forces SI pin LOW */
      outportb(data_port,mask);
}

unsigned char SO_SAMPLE() {
unsigned char SO_value;
      SO_value = inportb(status_port) & 0x80;  /* get value on SO pin */
                                               /* and isolate */
      SO_value = SO_value >> 7;                /* shift to LSB */
      SCK_HIGH();                              /* provide clock */
      SCK_LOW();
      return(SO_value);
}
```

```
void CS_HIGH() {
      mask = mask | 0x04;           /* forces CS pin HIGH */
      outportb(data_port,mask);
}

void CS_LOW() {
      mask = mask & 0xFB;           /* forces CS pin LOW */
      outportb(data_port,mask);
}

/******************************************************************/
/*                                                                */
/* Routine transmits a databyte to the SPI memory. The databyte   */
/* is passed to this routine directly when called.                */
/*                                                                */
/******************************************************************/
void SEND_BYTE(unsigned char byte) {
char count;

      for (count = 0; count <= 7; count++) {  /* loop to pass each bit */
            if ((byte & 0x80) == 0)           /* is the bit LOW? */
                  SI_LOW();
            else
                  SI_HIGH();
            byte = byte << 1;                 /* rotate to get next bit */
            SCK_HIGH();                       /* provide clock */
            SCK_LOW();
      }
}

/******************************************************************/
/*                                                                */
/* Routine receives a databyte from the SPI memory and passes it  */
/* back to the calling routine as an unsigned char.               */
/*                                                                */
/******************************************************************/
unsigned char GET_BYTE() {
int count;
unsigned char byte,temp;

      byte = 0;                                 /* reset byte holder */
      for (count = 0; count <= 7; count++) {    /* loop to get each bit */
            byte = byte << 1;                   /* rotate for next bit */
            temp = SO_SAMPLE();                 /* read SO pin */
            if (temp == 0)
                  byte = byte | 0x01;           /* reconstruct current bit */
      }
      return(byte);
}
```

```
/******************************************************************/
/*                                                                */
/* Set the WEL bit in the status register.                        */
/*                                                                */
/******************************************************************/
void WREN() {
      CS_LOW();
      SEND_BYTE(0x06);        /* WREN instruction */
      CS_HIGH();
}

/******************************************************************/
/*                                                                */
/* Reset the WEL bit in the status register.                      */
/*                                                                */
/******************************************************************/
void WRDI(){
      CS_LOW();
      SEND_BYTE(0x04);        /* WRDI instruction */
      CS_HIGH();
}

/******************************************************************/
/*                                                                */
/* Read the status register.                                      */
/*                                                                */
/******************************************************************/
unsigned char RDSR() {
unsigned char byte;
      CS_LOW();
      SEND_BYTE(0x05);        /* RDSR instruction */
      byte = GET_BYTE();      /* retrieve SR databyte */
      CS_HIGH();
      return(byte);
}

/******************************************************************/
/*                                                                */
/* Poll the status of the WIP bit to determine the early          */
/* completion of a nonvolatile write cycle.                       */
/*                                                                */
/******************************************************************/
void WIP_POLL() {
unsigned char byte;
      byte = 0;                     /* reset byte holder */
      do {
            byte = RDSR();          /* read the SR */
            byte = byte & 0x01;     /* isolate WIP bit */
      } while (byte != 0);          /* repeat until WIP bit is LOW */
}
```

```
/******************************************************************/
/*                                                                */
/* Write the status register with the value passed to the routine */
/* in the following format: WPEN|x|x|x|BP1|BP0|x|x               */
/*                                                                */
/******************************************************************/
void WRSR(unsigned char byte) {
      WREN();                       /* set WEL bit */
      byte = byte & 0x8C;           /* force other bits to 0 */
      CS_LOW();
      SEND_BYTE(0x01);              /* WRSR instruction */
      SEND_BYTE(byte);              /* send SR data byte */
      CS_HIGH();
      WIP_POLL();                   /* poll for completion of write cycle */
}

/*******************************************************************/
/*                                                                 */
/* Routine to read multiple consecutive bytes from the SPI memory. */
/* The specified number of bytes (no_bytes) are read starting from */
/* location (addr) and are stored to the array pointed to by       */
/* (*bytes).                                                       */
/*                                                                 */
/*******************************************************************/
void READ_SPI(int no_bytes,int addr,unsigned char *bytes) {
unsigned char addrhi,addrlo;
int n;
      CS_LOW();
      SEND_BYTE(0x03);                    /* READ instruction */
      addrhi = ((addr & 0x0FF00) >> 8);   /* decompose addr into 2 bytes */
      addrlo = addr & 0x0FF;
      SEND_BYTE(addrhi);                  /* send 2 address bytes */
      SEND_BYTE(addrlo);
      for (n = 0; n < no_bytes; n++) {    /* loop to read bytes */
            bytes[n]=GET_BYTE();          /* read next byte into the array */
      }
      CS_HIGH();
}

/*******************************************************************/
/*                                                                 */
/* Routine to write multiple consecutive bytes to the SPI memory.  */
/* The specified number of bytes (no_bytes) are written starting   */
/* at location (addr) and are taken from the array pointed to by   */
/* (*bytes).                                                       */
/*                                                                 */
/*******************************************************************/
void WRITE_SPI(int no_bytes,int addr,unsigned char *bytes) {
unsigned char addrhi,addrlo,byte;
int n;
      WREN();                             /* set WEL bit */
      CS_LOW();
      SEND_BYTE(0x02);                   /* WRITE instruction */
      addrhi = ((addr & 0x0FF00) >> 8);  /* decompose addr into 2 bytes */
      addrlo = addr & 0x0FF;
      SEND_BYTE(addrhi);                 /* send 2 address bytes */
      SEND_BYTE(addrlo);
      for (n = 0; n < no_bytes; n++) {   /* loop to write bytes */
            byte = bytes[n];             /* retrieve next byte to write */
            SEND_BYTE(byte);             /* write it to the SPI memory */
      }
      CS_HIGH();
      WIP_POLL();            /* poll for completion of write cycle */
}

/*****************************************************************/
/*                                                               */
/* Sample MAIN program to demonstrate the use of these routines  */
/*                                                               */
/*****************************************************************/
main() {
unsigned char data1[]={10,20,30,40,50,60,70,80,90,100};
unsigned char data2[]={15,25,35,45,55,65,75,85,95,105};
unsigned char data3[256];
unsigned char data4[128];

      outportb(data_port,mask);    /* initialize SI, CS, and SCK on power-up */
      WRITE_SPI(5,513,&data1);     /* write 5 bytes from 513 */
      READ_SPI(1,45,&data4);       /* read 1 byte from 45 */
      WRITE_SPI(1,45,&data1);      /* write 1 byte at 45 */
      WRITE_SPI(8,524,&data2);     /* write 8 bytes at 524 */
      READ_SPI(71,500,&data3);     /* read 71 bytes from 500 */
}
```

Xicor® Application Note AN76

Interfacing the X24F016/032/064 SerialFlash Memories to the Motorola 68HC11 Microcontroller

by Ray Kahidi, October 1995

This application note demonstrates how the Xicor X24Fxxx family of SerialFlash memories can be interfaced to the 68HC11 microcontroller family when connected as shown in Fig. 1. The interface uses two general purpose port D pins to interface to the SerialFlash memories. The 68HC11 assembly code listing for this application note can be obtained from the Xicor BBS, FaxBack system, or Xicor's website. The Xicor BBS can be reached at 1-800-258-8864, or in the (408) area code and internationally at 1-408-943-0655. Xicor's BBS will support up to a 14.4K baud rate modem (8 bits, no parity, 1 stop bit, and no local echo). The listing can be found in the MOTOROLA file library. Xicor application notes are also available through Xicor's FaxBack system at (408) 954-1627 and from the WWW using the URL: http://www.xicor.com.

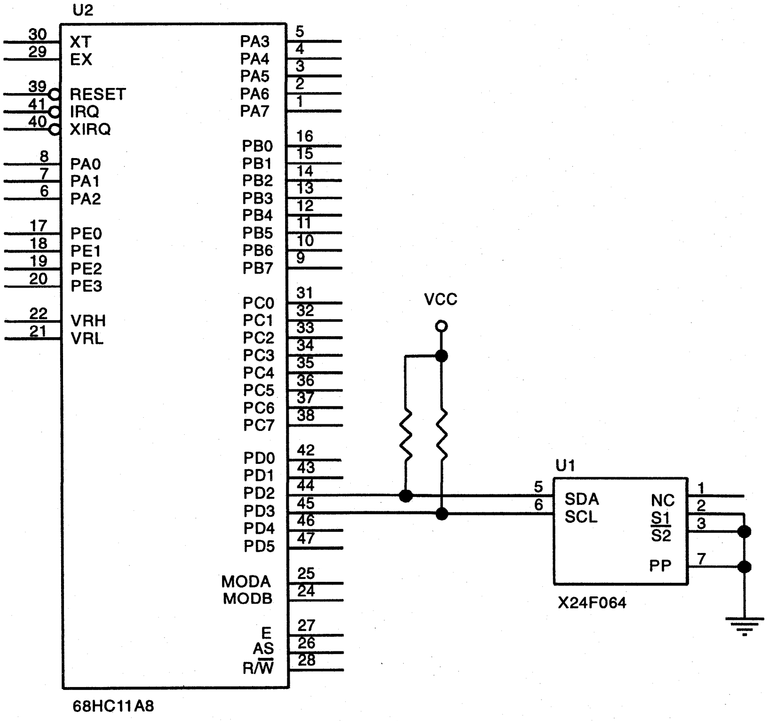


Figure 1 - Typical hardware connection for interfacing an X24F064 to the 68HC11 microcontroller

This page left intentionally blank.

Application Note AN77

Interfacing the X25F008/016/032/064 SerialFlash Memories to the Motorola 68HC11 Microcontroller SPI Port

by Ray Kahidi, October 1995

This application note demonstrates how the Xicor X25Fxxx family of SerialFlash memories can be interfaced to the 68HC11 microcontroller family when connected as shown in Fig. 1. The interface uses the SPI port pins to interface to the SerialFlash memories. The 68HC11 assembly code listing for this application note can be obtained from the Xicor BBS, FaxBack system, or Xicor's website. The Xicor BBS can be reached at 1-800-258-8864, or in the (408) area code and internationally at 1-408-943-0655. Xicor's BBS will support up to a 14.4K baud rate modem (8 bits, no parity, 1 stop bit, and no local echo). The listing can be found in the MOTOROLA files library. Xicor application notes are also available through Xicor's FaxBack system at (408) 954-1627 and from the WWW using the URL: http://www.xicor.com.

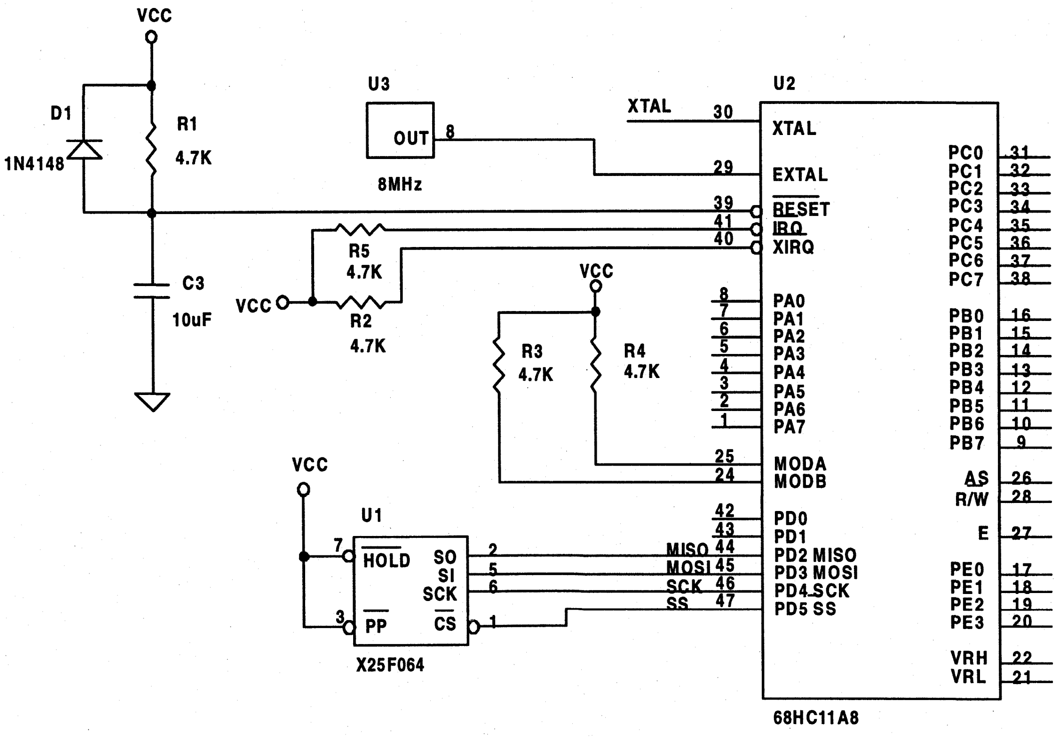


Figure 1 - Typical hardware connection for interfacing an X25F064 to the 68HC11 microcontroller

This page left intentionally blank.

Xicor® Application Note AN78

Interfacing the X24F016/032/064 SerialFlash Memories to 8051 Microcontrollers

by Ray Kahidi, October 1995

This application note demonstrates how the Xicor X24Fxxx family of SerialFlash memories can be interfaced to the 8051 microcontroller family when connected as shown in Fig. 1. The interface uses the port 1 pins to interface to the SerialFlash memories. The 8051 assembly code listing for this application note can be obtained from the Xicor BBS, FaxBack system, or Xicor's website. The Xicor BBS can be reached at 1-800-258-8864, or in the (408) area code and internationally at 1-408-943-0655. Xicor's BBS will support up to a 14.4K baud rate modem (8 bits, no parity, 1 stop bit, and no local echo). The listing can be found in the INTEL files library. Xicor application notes are also available through Xicor's FaxBack system at (408) 954-1627 and from the WWW using the URL: http://www.xicor.com.

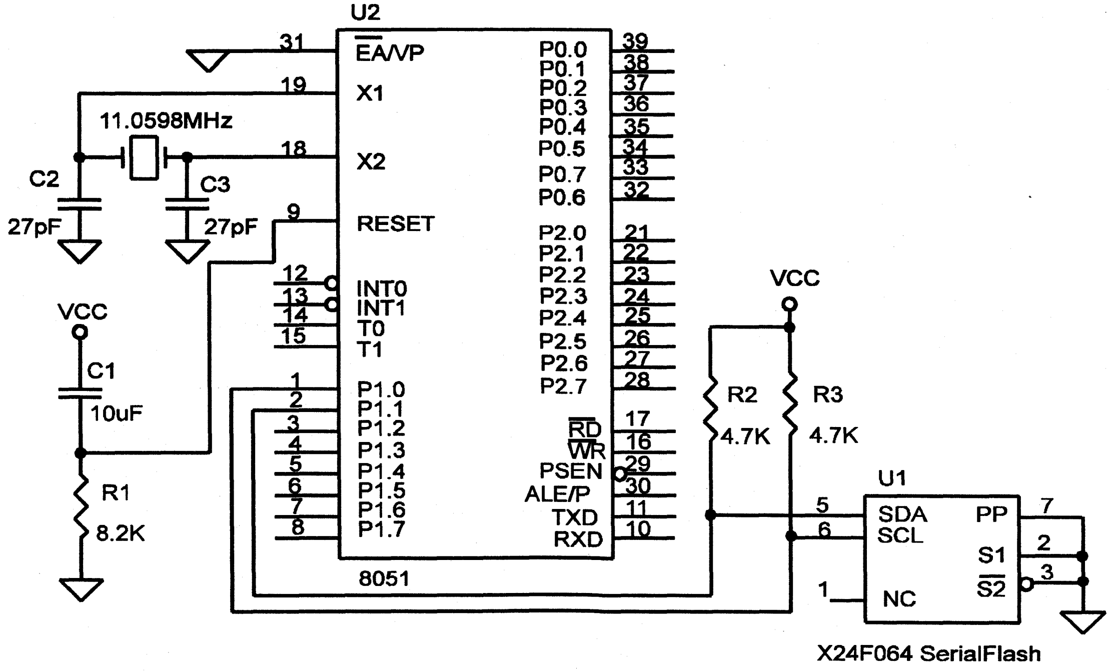


Figure 1 - Typical hardware connection for interfacing an X24F064 to the 8051 microcontroller

This page left intentionally blank.

Xicor® Application Note AN79

Interfacing the X25F008/016/032/064 SerialFlash Memories to 8051 Microcontrollers using the Serial Communication Port (UART)

by Ray Kahidi, October 1995

This application note demonstrates how the Xicor X25Fxxx family of SerialFlash memories can be interfaced to the 8051 microcontroller family when connected as shown in Fig. 1. The interface uses the serial communication port pins to interface to the SerialFlash memories. The 8051 assembly code listing for this application note can be obtained from the Xicor BBS, FaxBack system, or Xicor's website. The Xicor BBS can be reached at 1-800-258-8864, or in the (408) area code and internationally at 1-408-943-0655. Xicor's BBS will support up to a 14.4K baud rate modem (8 bits, no parity, 1 stop bit, and no local echo). The listing can be found in the INTEL files library. Xicor application notes are also available through Xicor's FaxBack system at (408) 954-1627 and from the WWW using the URL: http://www.xicor.com.

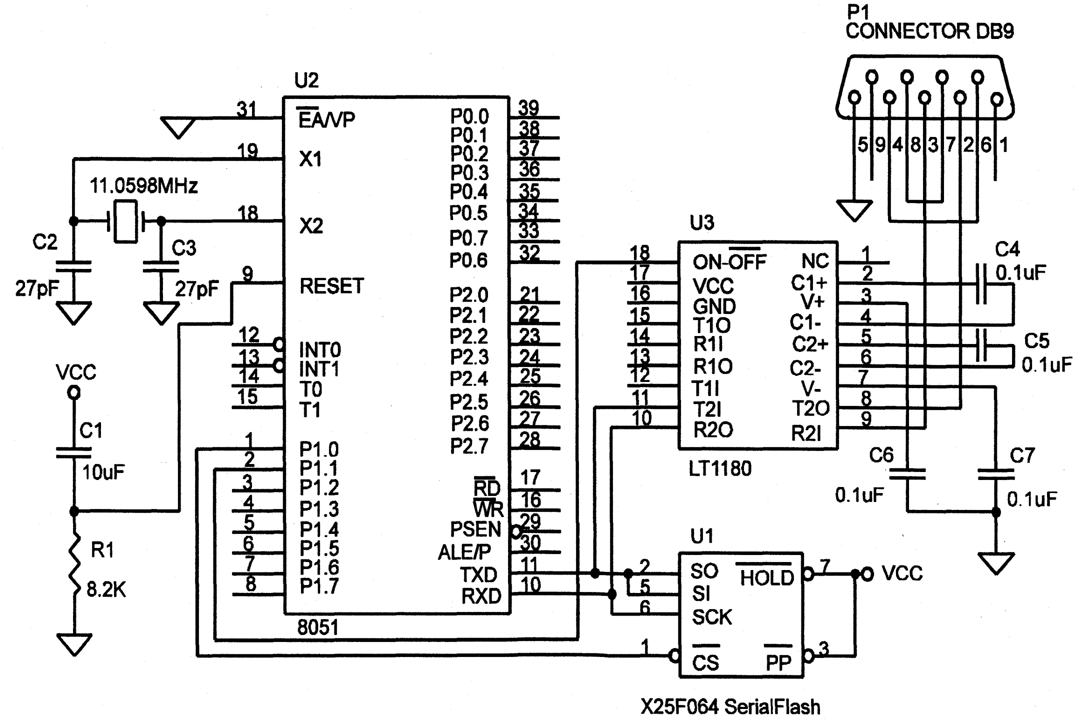


Figure 1 - Typical hardware connection for interfacing an X25F064 to the 8051's UART

This page left intentionally blank.

Interfacing to the X76F041 PASS SecureFlash

by Peter Chan, January 1996

Introduction

This application note discusses how to program the X76F041 PASS SecureFlash. The X76F041 is a password protected device, containing four 128 x 8 bit SecureFlash arrays. The device utilized three separate 64-bit passwords for maximum security protection. The Read and Write Passwords allow limited access to the protected SerialFlash array, while the Configuration Password "master key" provides unlimited access to all SerialFlash arrays.

Sector Write Operation

All commands except for response to reset are preceded by the start condition, which is a HIGH to LOW transition of SDA when SCL is HIGH. The X76F041 continuously monitors the SDA and SCL lines for the start condition and will not respond to any command until this condition has been met. The Sector Write mode begins by issuing the 3-bit write command followed by the sector address. Enter the password if required and then transfer 8 bytes of data. After the last byte to be transferred is acknowledged, a stop condition is issued, which starts the nonvolatile write cycle. If more than 8 bytes are transferred, the data will wrap around and previous data will be overwritten.

Random Read with Password Operation

Issue a start condition followed by the 3-bit read command and the first byte of the block address in which data is to be read:

Block 0 = 000h

Block 1 = 080h

Block 2 = 100h

Block 3 = 180h

This is followed by the 8 bytes read password sequence which includes the 10ms wait time and the password acknowledge polling sequence. If the password is accepted an "ACK" will be returned followed by 8 bits of "secure read setup" which is to be ignored. At this point a START is issued followed by the read address and data to be read within the original 1K block. Once the first byte has been read, another start command can be issued followed by a new 8 bit address within the same block. To access a read address in a different block, a stop command must be issued followed by a new read command, block address, and password sequence.

Sequential Read with Password

Once past the password acceptance and "secure read setup" sequence, now the host can read sequentially within the destined 1K-bit array. After the last bit has been read, a stop condition is generated without a preceding acknowledge.

Code Listing

The entire code listing for this X76F041 PASS SecureFlash firmware is available on the Xicor BBS (or through the Xicor FaxBack system). The Xicor BBS can be reached toll free at 1-800-258-8864, or in the (408) area code and internationally at 1-408-943-0655. Xicor's BBS will support up to a 19.2K baud rate modem (no parity, 8 bit words, 1 stop bit, and no local echo). These listings can be found in the SECURE SIG (Special Interest Group). Xicor application notes are also available through Xicor's FaxBack system at (408) 954-1627.

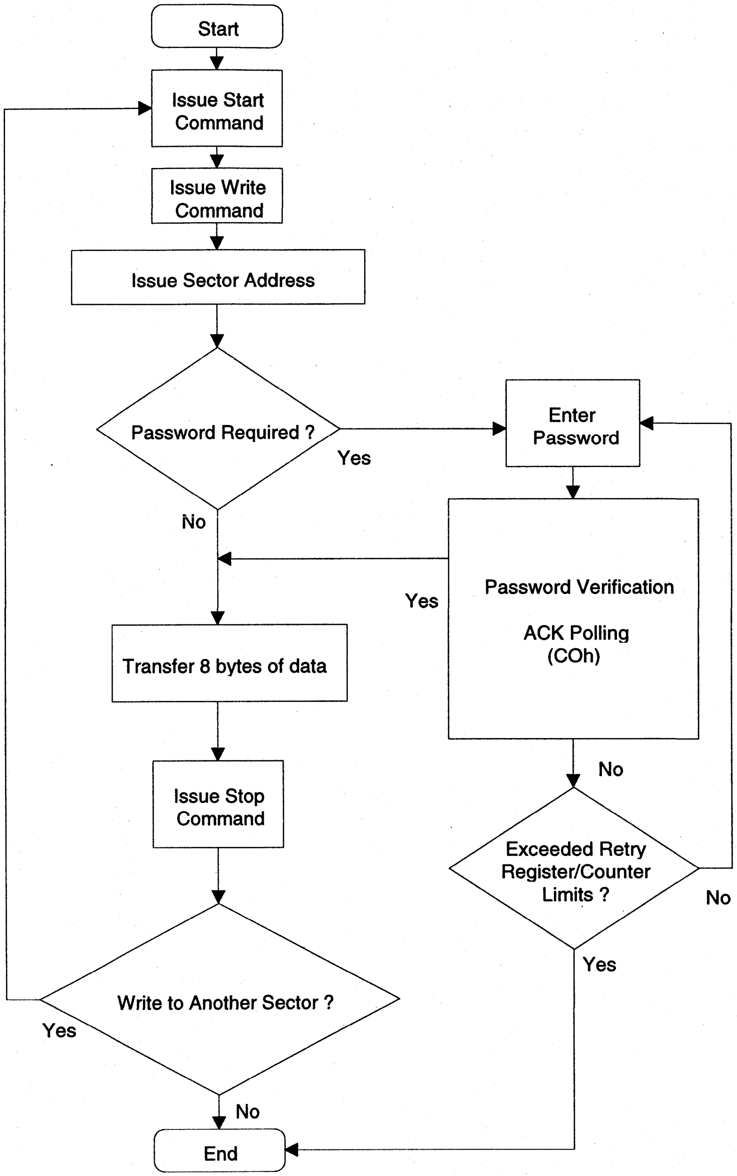


Figure 1 - Sector Write Command Flow

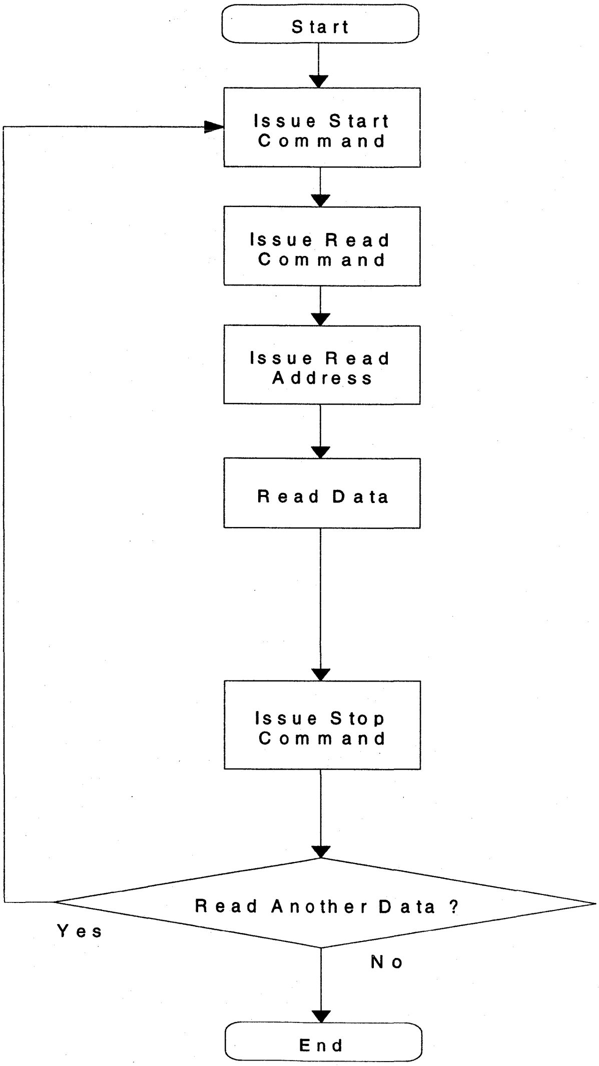


Figure 2 - Random Read without Password Command Flow

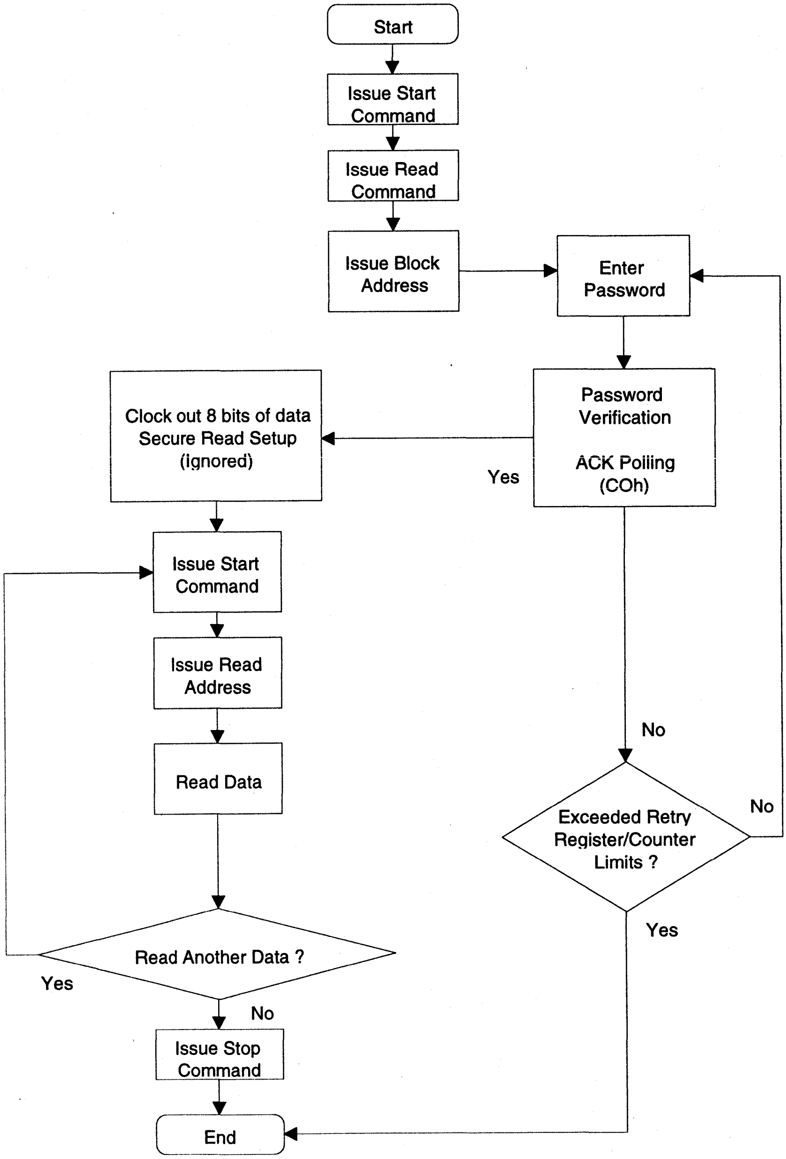


Figure 3- Random Read with Password Command Flow

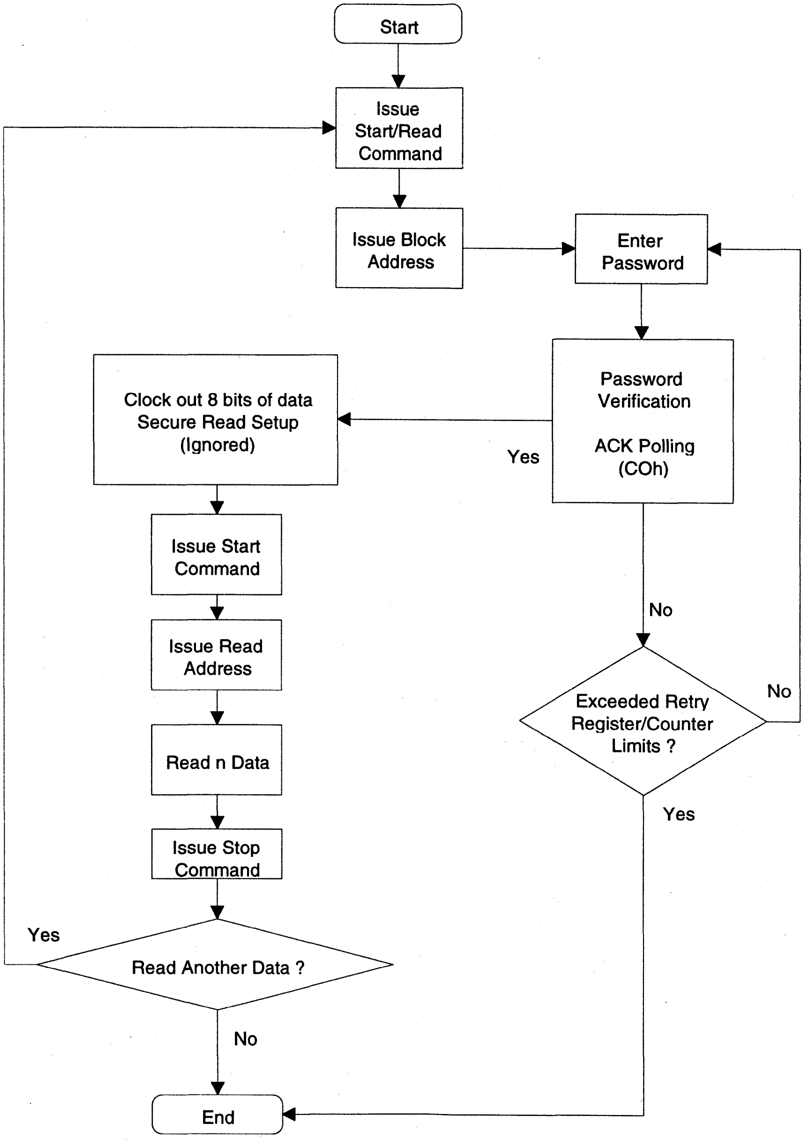


Figure 4- Sequential Read with Password Command Flow

```
/***************************************************************************************/
/*                    Xicor X76F041 PASS SecureFlash                                   */
/*                                                                                     */
/* The following C routines are provided to interface the X76F041 to the              */
/* Centronics parallel port on an IBM PC or compatible environment.                   */
/* Data is sent out to the X76F041 using D0 for the SCL line, D1 for the              */
/* SDA line, D6 for the RST line and D7 for the /CS line. Data is read                */
/* from the X76F041 using the SELECT line.                                             */
/*                                                                                     */
/* The command routines provided demonstrate the protocol for all X76F041             */
/* operation modes. These high level routines call interface functions to             */
/* perform low level tasks. The command routines are :-                               */
/*                                                                                     */
/*      1.  Write (Sector) using No Password                                           */
/*      2.  Write (Sector) using Write Password                                        */
/*      3.  Write (Sector) using Configuration Password                                */
/*      4.  Read (Random/Sequential) using No Password                                 */
/*      5.  Read (Random/Sequential) using Read Password                               */
/*      6.  Read (Random/Sequential) using Configuration Password                      */
/*      7.  Program Write Password                                                     */
/*      8.  Program Read Password                                                      */
/*      9.  Program Configuration Password                                             */
/*     10.  Reset Write Password                                                       */
/*     11.  Reset Read Password                                                        */
/*     12.  Read Configuration Registers                                               */
/*     13.  Program Configuration Registers                                            */
/*     14.  Mass Program                                                               */
/*     15.  Mass Erase                                                                 */
/*                                                                                     */
/*  An initial Hex configuration password = 0101010101010101                           */
/*                  write        password = 0202020202020202                           */
/*                  read         password = 0303030303030303                           */
/*                                                                                     */
/*  The main section of code reads the X76F041 ISO identifier; programs               */
/*  and read the configuration registers; performs a sector write; executes           */
/*  a random and sequential read; changes a password; resets a password               */
/*  and performs a mass erase.                                                         */
/*                                                                                     */
/*                                                                                PSC  */
/***************************************************************************************/

#include <stdio.h>
#include <dos.h>
#include <conio.h>

int data_port = 0x0378;                 /* parallel port data address */
int status_port = 0x0379;               /* parallel port status address */
unsigned char control = 0x00;           /* port control variable */
int address = 0x180;                    /* initial address byte */
int baddress = 0x180;                   /* initial block address (block 3) */
unsigned char data = 0x00;              /* data byte variable */
unsigned char ACR1 = 0xCC;              /* array control register 1 */
                                        /* Read & Write, R & W Password */
unsigned char ACR2 = 0xCC;              /* array control register 2 */
                                        /* Read & Write, R & W Password */
unsigned char CR = 0x20;                /* configuration register */
unsigned char RR = 0x10;                /* set retry register to 16 */
unsigned char RC = 0x00;                    /* set retry counter to 0 */
unsigned char ISO[4];                       /* ISO Response to Reset bytes */

/* byte array */
unsigned char byte_array[8] = {0x11,0x22,0x33,0x44,0x55,0x66,0x77,0x88};

/* configuration password bytes */
unsigned char conf_pass[8] = {0x01,0x01,0x01,0x01,0x01,0x01,0x01,0x01};

/* write password bytes */
unsigned char write_pass[8] = {0x02,0x02,0x02,0x02,0x02,0x02,0x02,0x02};

/* read password bytes */
unsigned char read_pass[8] = {0x03,0x03,0x03,0x03,0x03,0x03,0x03,0x03};

/* new password bytes */
unsigned char new_pass[8] = {0x40,0x40,0x40,0x40,0x40,0x40,0x40,0x40};


/**********************************************************************************/
/*                        CONTROL LINE STATE SETTINGS                             */
/**********************************************************************************/
void CE_high()  {
    control = control | 0x80;               /* forces CE pin high */
    outportb(data_port, control);
}

void CE_low()  {
    control = control & 0x7F;               /* forces CE pin low */
    outportb(data_port, control);
}

void RST_high()  {
    control = control | 0x40;               /* forces RST pin high */
    outportb(data_port, control);
}

void RST_low()  {
    control = control & 0xBF;               /* forces RST pin low */
    outportb(data_port, control);
}

void SCL_high()  {
   control = control | 0x01;                /* forces SCL pin high */
   outportb(data_port, control);
}

void SCL_low()  {
   control = control & 0xFE;                /* forces SCL pin low */
   outportb(data_port,control);
}

void SDA_high()  {
   control = control | 0x02;                /* forces SDA pin high */
   outportb(data_port, control);
}
```

```
void SDA_low()    {
   control = control & 0xFD;                /* forces SDA pin low */
   outportb(data_port, control);
}


/*******************************************************************************/
/*                             INTERFACE ROUTINES                              */
/*******************************************************************************/
/* 1. Issue a start condition */
/******************************/
void start()   {
   SDA_high();
   SCL_high();
   SDA_low();
   SCL_low();
}

/*****************************/
/* 2. Issue a stop condition */
/*****************************/
void stop()    {
    SDA_low();
    SCL_high();
    SDA_high();
    SCL_low();
}

/********************************************/
/* 3. Issue a clock pulse and read SDA line */
/********************************************/
unsigned char clock()    {
unsigned char SDA_value;

   SCL_high();                              /* send SCL high */
   SDA_value = inportb(status_port);        /* read parallel data port */
   SDA_value = SDA_value & 0x10;            /* determine SDA status */
   SCL_low();                               /* send SCL low */
   return (SDA_value);                      /* return SDA status */
}

/******************************************************************************/
/* 4. Master issues an acknowledge by sending SDA low to signal data received */
/******************************************************************************/
void ack()   {
    SDA_low();
    clock();
}

/*******************************************************************************/
/* 5. X76F041 issues an acknowledge by pulling SDA low to signal data received */
/*******************************************************************************/
void nack()   {
    SDA_high();
    clock();
}
```

```
/****************************************/
/* 6. Sends 8 bit 'byte' to the X76F041 */
/****************************************/
void out_byte(unsigned char byte) {
char count;                                 /* counter variable */

    for (count = 0; count <= 7; count++) {  /* loop 8 times */
       if ((byte & 0x80) == 0x00)
            SDA_low();                      /* SDA sent low if 'byte' MSB = 0 */
       else
            SDA_high();                     /* SDA sent high if 'byte' MSB =1 */
       clock();                             /* clock out SDA status */
       byte = byte << 1;                    /* shift 'byte' one bit left */
       }                                    /* end of loop */
}

/*****************************************************************************/
/* 7. Reads 8 bits from the X76F041 & stores value in global variable 'data' */
/*****************************************************************************/
char read_data() {
char count;                                 /* counter variable */

    for (count = 0; count <= 7; count++) {  /* loop 8 times */
       data = data << 1;                    /* shift 'data' one bit left */
       SDA_high();                          /* place SDA high */
       if ((clock()) == 0x10)               /* clock in and read SDA line */
            data = data | 0x01;             /* if SDA high set 'data' LSB to '1' */
       }                                    /* end of loop */
}

/*******************************************************/
/* 8. Performs a single Acknowledge Polling sequence */
/*******************************************************/

char ack_polling()  {
char pass;                                  /* acknowledge received (y/n) variable */

     start();                               /* issue start condition */
     out_byte(0xC0);                        /* send Hex byte C0 */
     SDA_high();                            /* place SDA high */
     if ((clock()) == 0x10)                 /* examine SDA line for an acknowledge */
       pass = 'n';                          /* acknowledge received */
     else
       pass = 'y';                          /* no acknowledge received */
     return(pass);                          /* return acknowledge status */
}

/**************************************************************/
/* 9. Concatenates high order address bit with command bits */
/**************************************************************/
unsigned char comm_byte_get(unsigned char comm_byte) {

    if ((address & 0x100) == 0x100)         /* determine status of 9th address bit */

       comm_byte = comm_byte | 0x01;        /* concatenate 9th address bit with */
                                            /* command bits */
    return(comm_byte);                            /* return new command byte */
}

/***************************************************/
/* 10. Reads 32 bit ISO standard response to reset */
/***************************************************/
void ISO_no_read() {
int count1, count2;                               /* counter variables */

    RST_high();                                   /* send Response to Reset pin high  */
    clock();                                      /* send clock pulse */
    RST_low();                                    /* return Response to Reset pin low */
    for (count1 = 0; count1 <= 3; count1++) {     /* loop 4 times */
       for (count2 = 0; count2 <= 7; count2++) {  /* loop 8 times */
            SDA_high();                           /* place SDA high */
            data = data >> 1;                     /* shift 'data' one bit left */
            if ((clock()) == 0x10)                /* clock in and read SDA line */
               data = data | 0x80;                /* if SDA high set 'data' */
                                                  /* MSB to '1' */
            }                                     /* end of 8 loops */
       ISO[count1] = data;                        /* store ISO Response */
                                                  /* to Reset byte */
       }                                          /* end of four loops */
}


/*****************************************************************************/
/*                          COMMAND ROUTINES                               */
/*****************************************************************************/
/*  1. WRITE (SECTOR) USING NO PASSWORD  */
/*****************************************/

void write_using_no_pass(int no_bytes) {
int count;                                        /* counter variable */
unsigned char comm_byte;                          /* command byte variable */

     comm_byte = comm_byte_get(0x00);             /* concatenate command bits with */
                                                  /* high order address bit */
     start();                                     /* issue start condition */
     out_byte(comm_byte);                         /* send command byte */
     nack();                                      /* receive an acknowledge */
     out_byte(address);                           /* send sector address byte */
     nack();                                      /* receive an acknowledge */
     for (count = 0; count <= (no_bytes - 1); count ++) {    /* loop for each byte */
                                                             /* to be sent */
       out_byte(byte_array[count]);               /* send data byte */
       nack();                                    /* receive an acknowledge */
       }                                          /* end of loop */
     stop();                                      /* issue stop condition */
     delay(10);                                   /* wait 10ms */
}

/********************************************/
/*  2. WRITE (SECTOR) USING WRITE PASSWORD  */
/********************************************/

void write_using_write_pass(int no_bytes) {
```

```
int count;                                    /* counter variable */
unsigned char comm_byte;                      /* command byte variable */

     comm_byte = comm_byte_get(0x00);         /* concatenate command bits with */
                                              /* high order address bit */
     start();                                 /* issue start condition */
     out_byte(comm_byte);                     /* send command byte */
     nack();                                  /* receive an acknowledge */
     out_byte(address);                       /* send sector address byte */
     nack();                                  /* receive an acknowledge */
     for (count = 0; count <= 7; count++) {   /* loop 8 times */
       out_byte(write_pass[count]);           /* send write password byte */
       nack();                                /* receive an acknowledge */
     }                                        /* end of 8 loops */
     delay(10);                               /* wait 10ms */
     ack_polling();                           /* acknowledge polling routine */
     for (count = 0; count <= (no_bytes - 1); count ++) {    /* loop for each byte */
                                                             /* to be sent */
       out_byte(byte_array[count]);           /* send data byte */
       nack();                                /* receive an acknowledge */
       }                                      /* end of loop */
     stop();                                  /* issue stop condition */
     delay(10);                               /* wait 10ms */
}

/****************************************************/
/*  3. WRITE (SECTOR) USING CONFIGURATION PASSWORD  */
/****************************************************/

void write_using_conf_pass(int no_bytes) {
int count;                                    /* counter variable */
unsigned char comm_byte;                      /* command byte variable */

     comm_byte = comm_byte_get(0x40);         /* concatenate command bits with */
                                              /* high order address bit */
     start();                                 /* issue start condition */
     out_byte(comm_byte);                     /* send command byte */
     nack();                                  /* receive an acknowledge */
     out_byte(address);                       /* send sector address byte */
     nack();                                  /* receive an acknowledge */
     for (count = 0; count <= 7; count++) {   /* loop 8 times */
       out_byte(conf_pass[count]);            /* send configuration password byte */
       nack();                                /* receive an acknowledge */
     }                                        /* end of loop */
     delay(10);                               /* wait 10ms */
     ack_polling();                           /* acknowledge polling routine */
     for (count = 0; count <= (no_bytes - 1); count++) {     /* loop for each byte */
                                                             /* to be sent */
       out_byte(byte_array[count]);           /* send data byte */
       nack();                                /* receive an acknowledge */
       }                                      /* end of loop */
     stop();                                  /* issue stop condition */
     delay(10);                               /* wait 10ms */
}
```

```
/****************************************************/
/*  4. READ (RANDOM/SEQUENTIAL) USING NO PASSWORD  */
/****************************************************/

void read_using_no_pass(int no_bytes) {
int count;                                    /* counter variable */
unsigned char comm_byte;                      /* command byte variable */

    comm_byte = comm_byte_get(0x20);          /* concatenate command bits with */
                                              /* high order address bit */
    start();                                  /* issue start condition */
    out_byte(comm_byte);                      /* send command byte */
    nack();                                   /* receive an acknowledge */
    out_byte(address);                        /* send address byte */
    nack();                                   /* receive an acknowledge */
    for (count = 0; count <= (no_bytes - 1); count++) {   /* loop for each byte */
                                                          /* to be read */
       read_data();                           /* read data byte */
       byte_array[count] = data;              /* store data byte in 'byte_array' */
       if (count != (no_bytes -1)) ack();     /* send an acknowledge if another */
                                              /* byte to be read */
       }                                      /* end of loop */
     stop();                                  /* issue stop condition */
}

/****************************************************/
/*  5. READ (RANDOM/SEQUENTIAL) USING READ PASSWORD  */
/****************************************************/

void read_using_read_pass(int no_bytes) {
int count;                                    /* counter variable */
unsigned char comm_byte;                      /* command byte variable */

    comm_byte = comm_byte_get(0x20);          /* concatenate command bits with */
                                              /* high order address bit */
    start();                                  /* issue start condition */
    out_byte(comm_byte);                      /* send command byte */
    nack();                                   /* receive an acknowledge */
    out_byte(baddress);                       /* send block address byte */
    nack();                                   /* receive an acknowledge */
    for (count = 0; count <= 7; count++) {    /* loop 8 times */
       out_byte(read_pass[count]);            /* send read password byte */
       nack();                                /* receive an acknowledge */
     }                                        /* end of loop */
     delay(10);                               /* wait 10ms */
     ack_polling();                           /* acknowledge polling routine */
     read_data();                             /* secure read setup */
     start();                                 /* issue start condition */
     out_byte(address);                       /* send read address */
     nack();                                  /* receive an acknowledge */
     for (count = 0; count <= (no_bytes - 1); count++) {   /* loop for each byte */
                                                           /* to be read */
       read_data();                           /* read data byte */
       byte_array[count] = data;              /* store data byte in 'byte_array' */
       if (count != 7) ack();                 /* send an acknowledge if another */
                                              /* byte to be read */
       }                                      /* end of loop */
     stop();                                 /* issue stop condition */
}

/***************************************************************/
/*  6. READ (RANDOM/SEQUENTIAL) USING CONFIGURATION PASSWORD  */
/***************************************************************/

void read_using_conf_pass(int no_bytes) {
int count;                                   /* counter variable */
unsigned char comm_byte;                     /* command byte variable */

    comm_byte = comm_byte_get(0x60);         /* concatenate command bits with */
                                             /* high order address bit */
    start();                                 /* issue start condition */
    out_byte(comm_byte);                     /* send command byte */
    nack();                                  /* receive an acknowledge */
    out_byte(baddress);                      /* send block address byte */
    nack();                                  /* receive an acknowledge */
    for (count = 0; count <= 7; count++) {   /* loop 8 times */
       out_byte(conf_pass[count]);           /* send configuration password byte */
       nack();                               /* receive an acknowledge */
    }                                        /* end of loop */
    delay(10);                               /* wait 10ms */
    ack_polling();                           /* acknowledge polling routine */
    read_data();                             /* secure read setup */
    start();                                 /* issue start condition */
    out_byte(address);
    nack();                                  /* receive an acknowledge */
    for (count = 0; count <= (no_bytes - 1); count++) {   /* loop for each byte */
                                                          /* to be read */
      read_data();                           /* read data byte */
      byte_array[count] = data;              /* store data byte in 'byte_array' */
      if (count != (no_bytes - 1)) ack();    /* send an acknowledge if another */
                                             /* byte to be read */
      }                                      /* end of loop */
    stop();                                  /* issue stop condition */
}

/*****************************************************/
/*  7. PROGRAM WRITE PASSWORD using write password  */
/*****************************************************/

void write_pass_prog() {
char count;                                  /* counter variable */

     start();                                /* issue start condition */
     out_byte(0x80);                         /* send command byte Hex 80 */
     nack();                                 /* receive an acknowledge */
     out_byte(0x00);                         /* send command byte Hex 00 */
     nack();                                 /* receive an acknowledge */
     for (count = 0; count <= 7; count++) {  /* loop 8 times */
       out_byte(write_pass[count]);          /* send write password byte */
       nack();                               /* receive an acknowledge */
     }                                       /* end of loop */
     delay(10);                              /* wait 10ms */
     ack_polling();                          /* acknowledge polling routine */
     for (count = 0; count <= 7; count++) {  /* loop 8 times */
```

```
       out_byte(new_pass[count]);             /* send new write password byte */
       nack();                                /* receive an acknowledge */
     }                                        /* end of loop */
     for (count = 0; count <= 7; count++) {   /* loop 8 times */
       out_byte(new_pass[count]);             /* resend new write password byte */
       nack();                                /* receive an acknowledge */
     }                                        /* end of loop */
     stop();                                  /* issue stop condition */
     delay(10);                               /* wait 10ms */
}
/*************************************************/
/*  8. PROGRAM READ PASSWORD using read password */
/*************************************************/

void read_pass_prog() {
char count;                                   /* counter variable */

     start();                                 /* issue start condition */
     out_byte(0x80);                          /* send command byte Hex 80 */
     nack();                                  /* receive an acknowledge */
     out_byte(0x10);                          /* send command byte Hex 10 */
     nack();                                  /* receive an acknowledge */
     for (count = 0; count <= 7; count++) {   /* loop 8 times */
       out_byte(read_pass[count]);            /* send read password byte */
       nack();                                /* receive an acknowledge */
     }                                        /* end of loop */
     delay(10);                               /* wait 10ms */
     ack_polling();                           /* acknowledge polling routine */
     for (count = 0; count <= 7; count++) {   /* loop 8 times */
       out_byte(new_pass[count]);             /* send new read password byte */
       nack();                                /* receive an acknowledge */
     }                                        /* end of loop */
     for (count = 0; count <= 7; count++) {   /* loop 8 times */
       out_byte(new_pass[count]);             /* resend new read password byte */
       nack();                                /* receive an acknowledge */
     }                                        /* end of loop */
     stop();                                  /* issue stop condition */
     delay(10);                               /* wait 10ms */
}

/*******************************************************************/
/*  9. PROGRAM CONFIGURATION PASSWORD using configuration password */
/*******************************************************************/

void conf_pass_prog() {
char count;                                   /* counter variable */

     start();                                 /* issue start condition */
     out_byte(0x80);                          /* send command byte Hex 80 */
     nack();                                  /* receive an acknowledge */
     out_byte(0x20);                          /* send command byte Hex 20 */
     nack();                                  /* receive an acknowledge */
     for (count = 0; count <= 7; count++) {   /* loop 8 times */
       out_byte(conf_pass[count]);            /* send configuration password byte */
       nack();                                /* receive an acknowledge */
     }                                        /* end of loop */
     delay(10);                               /* wait 10ms */
     ack_polling();                           /* acknowledge polling routine */
     for (count = 0; count <= 7; count++) {   /* loop 8 times */
       out_byte(new_pass[count]);             /* send new configuration pass byte */
       nack();                                /* receive a acknowledge */
     }                                        /* end of loop */
     for (count = 0; count <= 7; count++) {   /* loop 8 times */
       out_byte(new_pass[count]);             /* resend new configuration pass byte*/
       nack();                                /* receive an acknowledge */
     }                                        /* end of loop */
     stop();                                  /* issue stop condition */
     delay(10);                               /* wait 10ms */
}

/****************************************************************************/
/*  10. RESET WRITE PASSWORD to all zero bits using configuration password  */
/****************************************************************************/

void reset_write_pass() {
char count;                                   /* counter variable */

     start();                                 /* issue start condition */
     out_byte(0x80);                          /* send command byte Hex 80 */
     nack();                                  /* receive an acknowledge */
     out_byte(0x30);                          /* send command byte Hex 30 */
     nack();                                  /* receive an acknowledge */
     for (count = 0; count <= 7; count++) {   /* loop 8 times */
       out_byte(conf_pass[count]);            /* send configuration password byte */
       nack();                                /* receive an acknowledge */
     }                                        /* end of loop */
     delay(10);                               /* wait 10ms */
     ack_polling();                           /* acknowledge polling routine */
     stop();                                  /* issue stop condition */
     delay(10);                               /* wait 10ms */
}

/***************************************************************************/
/*  11. RESET READ PASSWORD to all zero bits using configuration password  */
/***************************************************************************/

void reset_read_pass() {
char count;                                   /* counter variable */

     start();                                 /* issue start condition */
     out_byte(0x80);                          /* send command byte Hex 80 */
     nack();                                  /* receive an acknowledge */
     out_byte(0x40);                          /* send command byte Hex 40 */
     nack();                                  /* receive an acknowledge */
     for (count = 0; count <= 7; count++) {   /* loop 8 times */
       out_byte(conf_pass[count]);            /* send configuration password byte */
       nack();                                /* receive an acknowledge */
     }                                        /* end of loop */
     delay(10);                               /* wait 10ms */
     ack_polling();                           /* acknowledge polling routine */
     stop();                                  /* issue stop condition */
     delay(10);                               /* wait 10ms */
}
```

```
/***************************************************************************/
/*  12. PROGRAM FIVE CONFIGURATION REGISTERS using configuration password  */
/***************************************************************************/

void conf_regs_program() {
char count;                                  /* counter variable */

     start();                                /* issue start condition */
     out_byte(0x80);                         /* send command byte Hex 80 */
     nack();                                 /* receive an acknowledge */
     out_byte(0x50);                         /* send command byte Hex 50 */
     nack();                                 /* receive an acknowledge */
     for (count = 0; count <= 7; count++) {  /* loop 8 times */
       out_byte(conf_pass[count]);           /* send configuration password byte */
       nack();                               /* receive an acknowledge */
     }                                       /* end of loop */
     delay(10);                              /* wait 10ms */
     ack_polling();                          /* acknowledge polling routine */
     out_byte(ACR1);                         /* send array control register 1 byte */
     nack();                                 /* receive an acknowledge */
     out_byte(ACR2);                         /* send array control register 2 byte */
     nack();                                 /* receive an acknowledge */
     out_byte(CR);                           /* send configuration register byte */
     nack();                                 /* receive an acknowledge */
     out_byte(RR);                           /* send retry register byte */
     nack();                                 /* receive an acknowledge */
     out_byte(RC);                           /* send retry counter register byte */
     nack();                                 /* receive an acknowledge */
     stop();                                 /* issue stop condition */
     delay(10);                              /* wait 10ms */
}

/************************************************************************/
/*  13. READ FIVE CONFIGURATION REGISTERS using configuration password  */
/************************************************************************/

void conf_regs_read() {
char count;                                  /* counter variable */

     start();                                /* issue start condition */
     out_byte(0x80);                         /* send command byte Hex 80 */
     nack();                                 /* receive an acknowledge */
     out_byte(0x60);                         /* send command byte Hex 60 */
     nack();                                 /* receive an acknowledge */
     for (count = 0; count <= 7; count++) {  /* loop 8 times */
       out_byte(conf_pass[count]);           /* send configuration password byte */
       nack();                               /* receive an acknowledge */
     }                                       /* end of loop */
     delay(10);                              /* wait 10ms */
     ack_polling();                          /* acknowledge polling routine */
     read_data();                            /* read array control register 1 */
     ack();                                  /* send an acknowledge */
     ACR1 = data;                            /* store array control register 1 */
     read_data();                            /* read array control register 2 */
     ack();                                  /* send an acknowledge */
     ACR2 = data;                            /* store array control register 2 */
     read_data();                            /* read configuration register */
     ack();                                  /* send an acknowledge */
     CR = data;                              /* store configuration register */
     read_data();                            /* read retry register */
     ack();                                  /* send an acknowledge */
     RR = data;                              /* store retry register */
     read_data();                            /* read retry counter register */
     ack();                                  /* send an acknowledge */
     RC = data;                              /* store retry counter register */
     stop();                                 /* issue stop condition */
}

/***************************************************************************/
/*  14. MASS PROGRAM changes all password, memory array and configuration  */
/*  register bits to zeros using the configuration password.               */
/***************************************************************************/

void mass_prog() {
char count;                                  /* counter variable */

     start();                                /* issue start condition */
     out_byte(0x80);                         /* send command byte Hex 80 */
     nack();                                 /* receive an acknowledge */
     out_byte(0x70);                         /* send command byte Hex 70 */
     nack();                                 /* receive an acknowledge */
     for (count = 0; count <= 7; count++) {  /* loop 8 times */
       out_byte(conf_pass[count]);           /* send configuration password byte */
       nack();                               /* receive an acknowledge */
     }                                       /* end of loop */
     delay(10);                              /* wait 10ms */
     ack_polling();                          /* acknowledge polling routine */
     stop();                                 /* issue stop condition */
     delay(10);                              /* wait 10ms */
}

/*************************************************************************/
/*  15. MASS ERASE changes all password, memory array and configuration  */
/*  register (except retry counter) bits to ones using the configuration */
/*  password. The retry counter register bits change to zeros.           */
/*************************************************************************/

void mass_erase() {
char count;                                  /* counter variable */

     start();                                /* issue start condition */
     out_byte(0x80);                         /* send command byte Hex 80 */
     nack();                                 /* receive an acknowledge */
     out_byte(0x80);                         /* send command byte Hex 80 */
     nack();                                 /* receive an acknowledge */
     for (count = 0; count <= 7; count++) {  /* loop 8 times */
       out_byte(conf_pass[count]);           /* send configuration password byte */
       nack();                               /* receive an acknowledge */
     }                                       /* end of loop */
     delay(10);                              /* wait 10ms */
     ack_polling();                          /* acknowledge polling routine */
     stop();                                 /* issue stop condition */
     delay(10);                              /* wait 10ms */
}
```

```
/************************************************************************************/
/*                             MAIN SECTION OF CODE                                 */
/************************************************************************************/

main() {
int count;

    delay(0);                               /* calibrate clock */
    CE_low();                               /* send /CE low */
    ISO_no_read();                          /* read 32 bit ISO response to reset */
    for (count = 0; count <= 3; count++)    /* loop 4 times */
       printf("%x",ISO[count]);             /* output read 32 bit ISO identifier */
    conf_regs_program();                    /* program configuration registers */
    conf_regs_read();                       /* read configuration registers */
    printf("%x", ACR1);                     /* output read array control register 1 */
    printf("%x", ACR2);                     /* output read array control register 2 */
    printf("%x", CR);                       /* output read configuration register */
    printf("%x", RR);                       /* output read retry register */
    printf("%x", RC);                       /* output read retry counter */
    read_using_conf_pass(1);                /* random read using conf password */
    printf("%x",byte_array[0]);             /* output read data byte */
    write_using_write_pass(8);              /* sector write using write password */
    read_using_conf_pass(8);                /* sequential read using conf password */
    for (count = 0; count <= 7; count++)    /* loop 8 times */
       printf("%x",byte_array[count]);      /* output read data bytes */
    write_pass_prog();                      /* change write password */
    reset_read_pass();                      /* reset all read password bits to 0 */
    mass_erase();                           /* mass erase part */
    CE_high();                              /* send /CE high */
}
```

Interfacing to the X24F128 2-Wire SerialFlash™ Memory

by Gray Creager, May 1996

The following code is intended to provide generic C routines that can be used with Xicor's 3-byte protocol I^{2}C-compatible memory devices. This code was written for (and tested with) the X24F128 SerialFlash memory, but will also be compatible with future 3-byte protocol SerialFlash memories. Figure 1 shows the circuit used to test and debug this code from the Centronics parallel printer port on a generic PC.

The code implements a complete set of operations for the device, including the ability to independently access the PEL, RPEL, WPEN, BL1, and BL0 bits in the Program Protect Register (PPR). To allow for compatibility with future devices, the write routine is implemented assuming a variable sector size. This allows the same routine to handle sector writes of 32 bytes each to the memory array, simply by passing the sector size as a variable during the routine call. The correct use of each of these routines is explained and demonstrated within the attached code.

Downloadable code from this application notes can be obtained from Xicor's BBS, which can be reached toll free at 1-800-258-8864, or in the (408) area code and internationally at 1-408-943-0655. Xicor's BBS will support up to a 19.2K baud rate modem (no parity, 8 bit words, 1 stop bit, and no local echo). These listings can be found in the MISC SIG (Special Interest Group). Xicor application notes are also available through Xicor's FaxBack system at (408) 954-1627 and on the World Wide Web (at the URL, http://www.xicor.com).

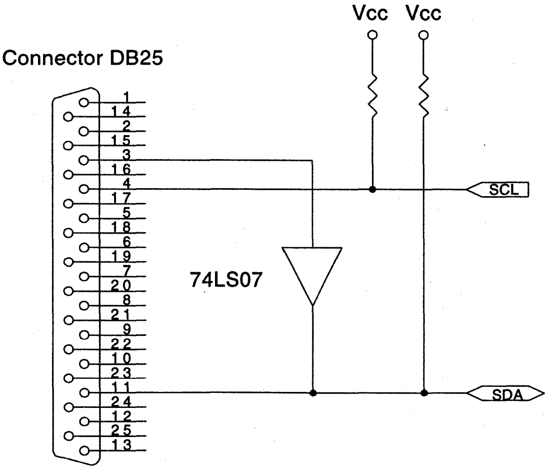


Figure 1 - Simple interface between 2-wire serial memories and a parallel printer port on a PC.

SerialFlash™ is a trademark of Xicor, Inc.

```
/************************************************************************/
/*                                                                      */
/* Software for Xicor's X24F128 Two-Wire SerialFlash memory             */
/*                                                                      */
/* This code was written and tested with the PC port diagram given      */
/* in this application note. Because of capacitive port delays          */
/* on the test computer, 1ms delays have been added to guarantee        */
/* that SCL and SDA setup times are met. These may be deleted           */
/* for systems without this problem. This code contains all of the      */
/* routines necessary to implement this interface.                      */
/*                                                                      */
/*                                                           GHC IV     */
/************************************************************************/

#include <stdio.h>
#include <stdlib.h>

int data_port = 0x3bc;              /* test computer printer port output address */
int status_port = 0x3bd;            /* test computer printer port input address */
unsigned char control = 0xff;
unsigned char slave_w = 0xa0;       /* slave address for writes (S2,S1,S0=0) */
unsigned char slave_r = 0xa1;       /* slave address for reads (S2,S1,S0=0) */

void SCL_high(){
      control = control | 0x04;     /* set SCL bit at port */
      outportb(data_port, control);
      delay(1);                     /* compensate for high bus capacitance */
}

void SCL_low(){
      control = control & 0xfb;     /* reset SCL bit at port */
      outportb(data_port, control);
      delay(1);                     /* compensate for high bus capacitance */
}

void SDA_high(){
      control = control | 0x02;     /* set SDA bit at port */
      outportb(data_port,control);
      delay(1);                     /* compensate for high bus capacitance */
}

void SDA_low(){
      control = control & 0xfd;     /* reset SDA bit at port */
      outportb(data_port,control);
      delay(1);                     /* compensate for high bus capacitance */
}

void start(){                       /* generate I2C start condition */
      SDA_high();
      SCL_high();
      SDA_low();
      SCL_low();
}
```

```
void stop(){                        /* generate I2C stop condition */
      SDA_low();
      SCL_high();
      SDA_high();
}

unsigned char clock(){              /* generate clock pulse and read SDA */
unsigned char SDA_value;

      SCL_high();
      SDA_value = inportb(status_port);   /* read parallel port */
      SDA_value = SDA_value & 0x80;       /* isolate SDA */
      SCL_low();
      SDA_value = SDA_value >> 7;         /* shift to LSB */
      return(SDA_value);                  /* and return data */
}

void ack(){                               /* master sends acknowledge */
      SDA_low();
      clock();
}

void nack(){                        /* master receives acknowledge */
      SDA_high();
      clock();
}

void out_byte(unsigned char byte){  /* master sends a byte */
char count;

      for (count = 0; count <= 7; count++){     /* send data byte to device */
            if ((byte & 0x80) == 0)             /* one bit at a time */
                  SDA_low();
            else
                  SDA_high();
            byte = byte << 1;       /* shift for next bit */
            clock();
      }
}

unsigned char get_byte(){           /* master receives a byte */
int count;
unsigned char byte,temp;

      byte = 0;
      for (count = 0; count <= 7; count++){     /* read data byte from device */
            byte = byte << 1;                   /* one bit at a time */
            SDA_high();
            temp = clock();               /* input bit from port */
            if (temp == 0)
                  byte = byte | 0x01;
      }
      return(byte);                       /* return data byte */
}
```

```
void ack_poll(){                    /* check for acknowledge from slave */
unsigned char poll;

      do {                          /* ack polling loop */
            start();
            out_byte(slave_w);      /* slave address */
            SDA_high();
            poll = clock();         /* check SDA level */
      } while (poll ==  0);         /* repeat while SDA is HIGH */
      stop();                       /* SDA is LOW so stop */
}

void dummy_write(int addr){
int temp;

            start();
            out_byte(slave_w);      /* slave address for writes */
            nack();
            if (addr != 0xffff) {   /* accessing the program protect register? */

                  temp = ((addr & 0x3f00) >> 16);    /* no, construct */
                                                     /* first address byte */
                  out_byte(temp);
                  nack();
                  temp = addr & 0xff;          /* construct second address byte */
                  out_byte(temp);
            }
            else {                  /* yes */
                  out_byte(0xff);   /* send program protect register address */
                  nack();
                  out_byte(0xff);
            }
            nack();
}

/**********************************************************************/
/*                                                                    */
/* Sequential read routine that handles all read operations.  For    */
/* current address reads, set current != 0 and addr as a don't care   */
/* when calling the routine, otherwise addr = starting address and    */
/* *bytes points to buffer where data is to be stored for later       */
/* use. no_bytes is the number of bytes to be sequentially read       */
/* from the SerialFlash.                                              */
/*                                                                    */
/**********************************************************************/

void seq_read(int current,int no_bytes,int addr,unsigned char *bytes){
int n, temp;

      if (current == 0)             /* random read? */
            dummy_write(addr);      /* yes, send address */

      start();
      out_byte(slave_r);                  /* slave address for read */
      nack();
      for (n = 0;n < no_bytes - 1; n++){  /* sequentially read data */
            bytes[n] = get_byte();        /* in loop and send acknowledges */
            ack();
      }
      bytes[no_bytes - 1] = get_byte();   /* for last databyte, don't */
      stop();                             /* send acknowledge */
}

/************************************************************************/
/*                                                                      */
/* Sequential write routine handles all write operations. no_bytes      */
/* is number of bytes to write, starting at address (addr), and         */
/* takes the data from the buffer pointed to by *bytes. This routine is */
/* written to allow for variable sector sizes to ensure compatibility   */
/* with future devices with a minimum of changes. When writing to the   */
/* X24F128 SerialFlash memory array, no_bytes should always be 32.      */
/*                                                                      */
/************************************************************************/

void seq_write(int no_bytes,int addr,unsigned char *bytes){
unsigned char temp;
int n;

      dummy_write(addr);                  /* send address */
      for (n = 0;n < no_bytes; n++){
            out_byte(bytes[n]);           /* loop and send data bytes */
            nack();
      }
      stop();                             /* begin nonvolatile write cycle */
      ack_poll();                         /* poll for cycle completion */
}

/************************************************************************/
/*                                                                      */
/* PEL write routine allows for the PEL bit to be set or reset.         */
/*                                                                      */
/* To set this bit, call this routine as: write_PEL(1)                  */
/* To reset this bit call this routine as: write_PEL(0)                 */
/*                                                                      */
/************************************************************************/

void write_PEL(int state) {
      dummy_write(0xffff);          /* write to program protect register */
      if (state == 1)               /* if state = 1, then we set PEL bit */
            out_byte(0x02);         /* set PEL bit */
      else
            out_byte(0x00);         /* reset PEL bit */
      nack();
      stop();
}
```

```
/************************************************************************/
/*                                                                      */
/* RPEL write routine allows for the RPEL bit to be set or reset.       */
/*                                                                      */
/* To set this bit, call this routine as: write_RPEL(1);                */
/* To reset this bit call this routine as: write_RPEL(0);               */
/*                                                                      */
/* It is unlikely that the need to reset RPEL will arise, however the   */
/* possibility is provided for in the routine.                          */
/*                                                                      */
/************************************************************************/

void write_RPEL(int state) {
      dummy_write(0xffff);          /* write to program protect register */
      if (state == 1)               /* if state = 1, then we set RPEL bit */
            out_byte(0x06);         /* set RPEL bit */
      else
            out_byte(0x02);         /* reset RPEL bit */
                                    /* note that if PEL = 0 and you call */
                                    /* this routine with state = 0, then */
                                    /* you'll end up setting PEL! */

      nack();
      stop();
}

/************************************************************************/
/*                                                                      */
/* program PPR routine allows for the PPEN, BL0, and BL1 bits to be set */
/* in the program protect register. This routine will have no effect    */
/* unless the PEL and RPEL bits are set when it is called. If PPEN = 1  */
/* and the WP pin is HIGH, this routine will have no effect because the */
/* status register will be locked.                                      */
/*                                                                      */
/* program_PPR(PPEN,BL1,BL0);                                           */
/*                                                                      */
/* To set a bit, use 1 in its' position                                 */
/* To reset a bit, use 0 in its' position                               */
/*                                                                      */
/* to call this routine, substitute the desired bit values for each     */
/* register bit per this example:                                       */
/*                                                                      */
/* program_PPR(0,1,1); <-- resets PPEN and sets both BL1 and BL0        */
/*                                                                      */
/************************************************************************/

void program_PPR(int PPEN, int BL1, int BL0){
int temp;

      dummy_write(0xffff);          /* write to program protect register */
      temp = 0;
      if (PPEN == 1)                /* set the necessary bits */
            temp = temp | 0x80;
      if (BL1 == 1)
            temp = temp | 0x10;
      if (BL0 == 1)
            temp = temp | 0x08;
      out_byte(temp | 0x02);        /* write PPEN, BL1, and BL0 bits, */
                                    /* reset RPEL, leave PEL bit set */
      nack();
      stop();
      ack_poll();                   /* poll for cycle completion */
}

/******************************************************/
/*                                                    */
/* Simple program to demonstrate these routines.     */
/*                                                    */
/******************************************************/

main(){
unsigned char data1[] ={0x20,0x21,0x22,0x23,0x24,0x25,0x26,0x27,
                        0x28,0x29,0x2a,0x2b,0x2c,0x2d,0x2e,0x2f,
                        0x30,0x31,0x32,0x33,0x34,0x35,0x36,0x37,
                        0x38,0x39,0x3a,0x3b,0x3c,0x3d,0x3e,0x3f};

unsigned char data2[] ={0x40,0x41,0x42,0x43,0x44,0x45,0x46,0x47,
                        0x48,0x49,0x4a,0x4b,0x4c,0x4d,0x4e,0x4f,
                        0x50,0x51,0x52,0x53,0x54,0x55,0x56,0x57,
                        0x58,0x59,0x5a,0x5b,0x5c,0x5d,0x5e,0x5f};

unsigned char data3[512];           /* buffer to store bytes from SerialFlash */
unsigned char data4[1];

      write_PEL(1);                       /* set PEL bit */
      write_RPEL(1);                      /* set RPEL bit */
      program_PPR(0,0,0);                 /* write to PPR (PPEN,BL1,BL0)*/
      seq_read(0,1,0xffff,&data4);        /* read PPR */
      seq_write(32,0,&data1);             /* write first sector */
      seq_write(32,32,&data2);            /* write second sector */
      seq_read(0,64,0,&data3);            /* random read of 2 sectors */
                                          /* for verification of writes */
      seq_write(32,0,&data2);             /* sector write */
      seq_read(0,32,0,&data3);            /* verify sector */
      write_RPEL(1);                      /* set RPEL bit */
      program_PPR(0,1,1);                 /* write PPR and set BlockLock for */
                                          /* the entire array */
      seq_read(0,1,0xffff,&data4);        /* read and verify PPR */
      seq_write(32,32,&data1);            /* attempt a sector write */
      seq_read(0,32,32,&data3);           /* verify sector was not changed */
      write_RPEL(1);                      /* set RPEL bit */
      program_PPR(0,0,0);                 /* write PPR and turn off BlockLock */
      seq_read(0,1,0xffff,&data4);        /* read PPR */
      seq_read(0,32,0,&data3);            /* random read of sector */
      seq_read(1,32,0,&data3);            /* current address read of sector */
      seq_write(32,0,&data1);             /* sector write */
      seq_read(0,32,0,&data3);            /* random read of sector */
}
```

This page left intentionally blank.

Block Lock^{TM} and Programmable Hardware Write Protection for Xicor's 2-Wire Serial Devices

by Carlos Martinez, May 1996

One of the major drawbacks with memory devices using the 2-wire serial interface is the loss or corruption of data, either due to noise on the bus or from inadvertent write operations. Xicor has addressed this need with the Block Lock^{TM} and Programmable Hardware Write protection features on the X24165/325/645 product family of E^{2}PROMs, the X24F016/032/064 family of SerialFlash^{TM} and the X24F128 SerialFlash.

Block Lock provides the user the ability to software protect 1/4, 1/2 or all of the memory array. This means that by setting two bits in a Write Protect Register, the user can prevent a write operation from changing data in that block protected region. This is especially important when there are configuration parameters, critical data or manufacturing information on the same memory device as other data that is being changed more often or is of less significance.

The value of Block Lock is also demonstrated when the processor fails or the bus becomes noisy. In these cases, the protected memory must be unprotected before a write operation becomes valid. This two step process reduces the likelihood of corrupted data. Since these protection bits are nonvolatile, the protection scheme is in effect even after cycling the power.

The second feature, Programmable Hardware Write Protection, provides a higher level of data security that allows the Block Lock protected areas of memory to be permanently protected via a hardware write protect pin that is enabled through software. The advantage of this scheme is that data can be downloaded to the device, while in the system, and then be secured through software. Once secured, a hardware change is required before protected areas of memory can be changed.

Traditionally, nonvolatile memories with hardware protection had to be pre-programmed before being mounted on the circuit board, adding several manufacturing steps. Now, the Programmable Hardware Write Protection Feature from Xicor allows a blank device to be placed on the board and to be programmed in-situ. This reduces special handling of the nonvolatile memory device and simplifies the manufacturing flow. This is particularly useful for systems utilizing surface mount devices that are cumbersome to pre-program or in systems where programming may be required just prior to shipment.

Programmable Hardware Write Protection is implemented through a combination of a write protect pin (WP) and a write protect enable (WPEN) bit in the Write Protect Register. SerialFLASH devices have a program protect pin (PP) and a Program Protect Enable bit (PPEN). However, other than nomenclature, their operation is identical to the WP pin and WPEN bit.

When the WP pin is tied to V_{CC} and the WPEN bit is set to '1', block protected areas cannot be written and the WPEN bit cannot be changed. In this way, the WP pin must be unsoldered from the board before block protection can be changed. The various combinations of WPEN and WP states are shown in Table 1, on page 2.

In practice, a likely sequence of events during system manufacturing would be as follows:

1. Along with the other devices on the board, solder Xicor's blank nonvolatile memory to the board with the WP pin tied to V_{CC}. Since the WPEN bit is '0', the device can still be programmed.
2. Run a test program on the finished board that initializes a block protected configuration area of the nonvolatile memory. This configuration memory can contain the board serial number, manufacturer data, results of the final test along with the date and location of the test, the model number of the system in which the board is installed, and much more. With densities up to 64K bits, there is a lot of room for both configuration and user information.

3. If the board passes all tests, the WPEN bit is set which "locks" the block protected area. If the board fails before being locked, it can be run through rework with the configuration area open to new data, but still protected on a software level.

In the Xicor 2-wire devices the Write Protect Register is located at the same address as the highest location in the array (except for the X24F128, where the Write Protect Register is located at FFFFh and does not overlap the array). For the 2-wire devices with this register/memory overlap, there are some programming considerations. A one byte read or write to the address of the highest location in the array accesses the Write Protect Register. A page/sector operation, beginning at some lower address, is required for access to the highest location of the memory array.

To protect against an inadvertent write to the Write Protect Register, a three step sequence is required to change the block protect or WPEN bits. This three step sequence reduces the possibility of overwriting the register when attempting to write to the array.

In order to change the Block Lock bits or the WPEN bit, the following steps are required:

1. Write a 02H to the Write Protect Register to set the Write Enable Latch (WEL). This is a volatile operation, so no delay is required after the write.
2. Write a 06H to the Write Protect Register to set both the Register Write Enable Latch (RWEL) and the WEL bit. This is also a nonvolatile cycle.
3. Write a value to the Write Protect Register that includes the BP1, BP0 and WPEN bits set to the desired state, with the WEL bit set to '1' and the RWEL bit set to '0'. This can be represented as w00yz010 in binary, where w is the WPEN bit, y is the BP1 bit and z is the BP0 bit. Since this is a nonvolatile write cycle it will take up to 10ms to complete. The RWEL bit is reset by this cycle and the sequence must be repeated to change the bits again. If bit 2 is set to '1' in this third step (w00yz110) then the RWEL bit is set, but the WPEN , BP1 and BP0 bits remain unchanged.

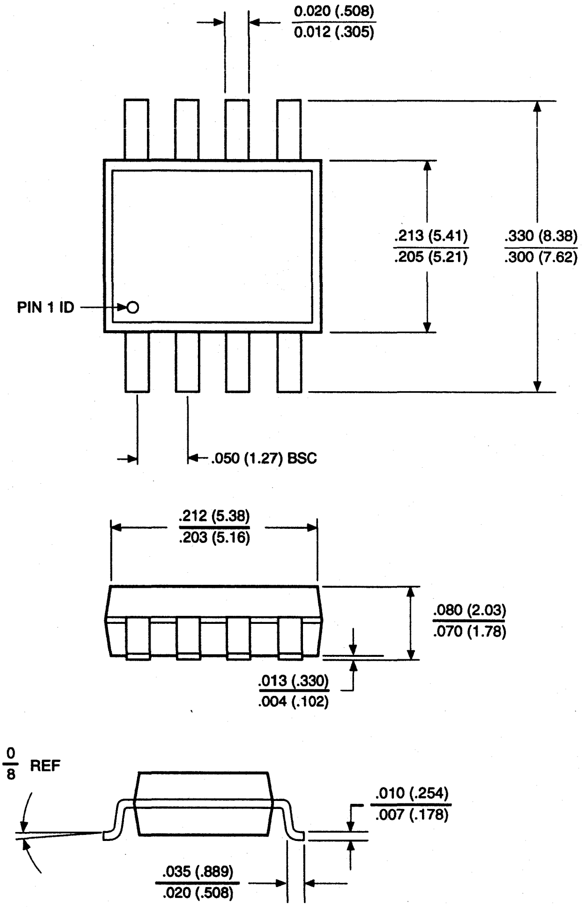
Assembly code listings for various processors showing how the Block Lock and Hardware Write Protection features are implemented, can be obtained from the Xicor BBS, FaxBack system, or Xicor's website. The Xicor BBS can be reached at 1-800-258-8864, or in the (408) area code and internationally at 1-408-943-0655. Xicor's BBS will support up to a 14.4K baud rate modem (8 bits, no parity, 1 stop bit, and no local echo). Xicor application notes are also available through Xicor's FaxBack system at (408) 954-1627 and from the WWW using the URL: http://www.xicor.com.

Table 1: Write Protect Status Table

WP	*WPEN*	*Memory Array (not Block Protected)*	*Memory Array (Block Protected)*	*BP bits*	*WPEN bit*
0	X	Can be written	Cannot be written	Can be written	Can be written
X	0	Can be written	Cannot be written	Can be written	Can be written
1	1	Can be written	Cannot be written	Cannot be written	Cannot be written

Xicor®

8-LEAD PLASTIC, 0.200" WIDE SMALL OUTLINE GULLWING PACKAGE TYPE "A" (EIAJ SOIC)

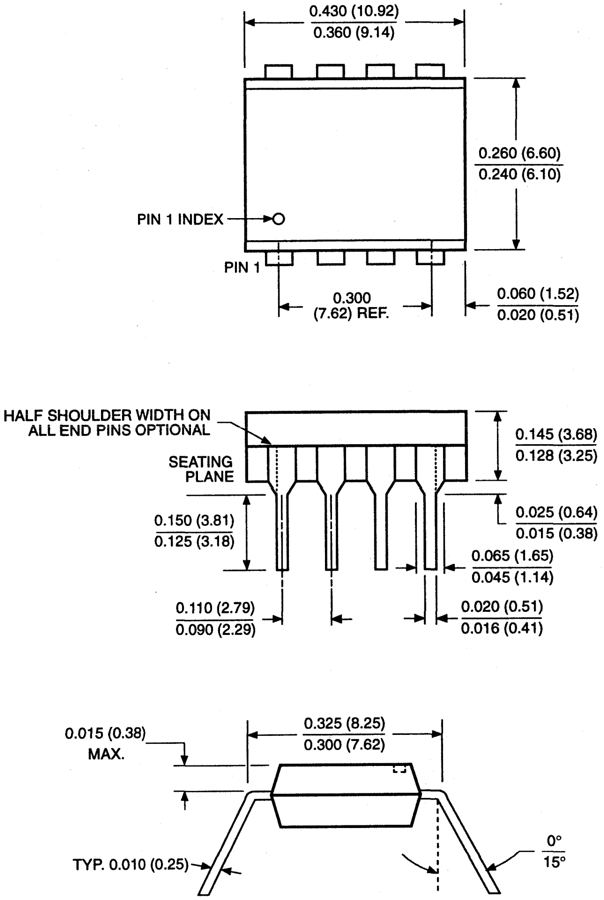


NOTE:

1. **ALL DIMENSIONS IN INCHES (IN PARENTHESES IN MILLIMETERS)**
2. **PACKAGE DIMENSIONS EXCLUDE MOLDING FLASH**

3926 ILL F33.1

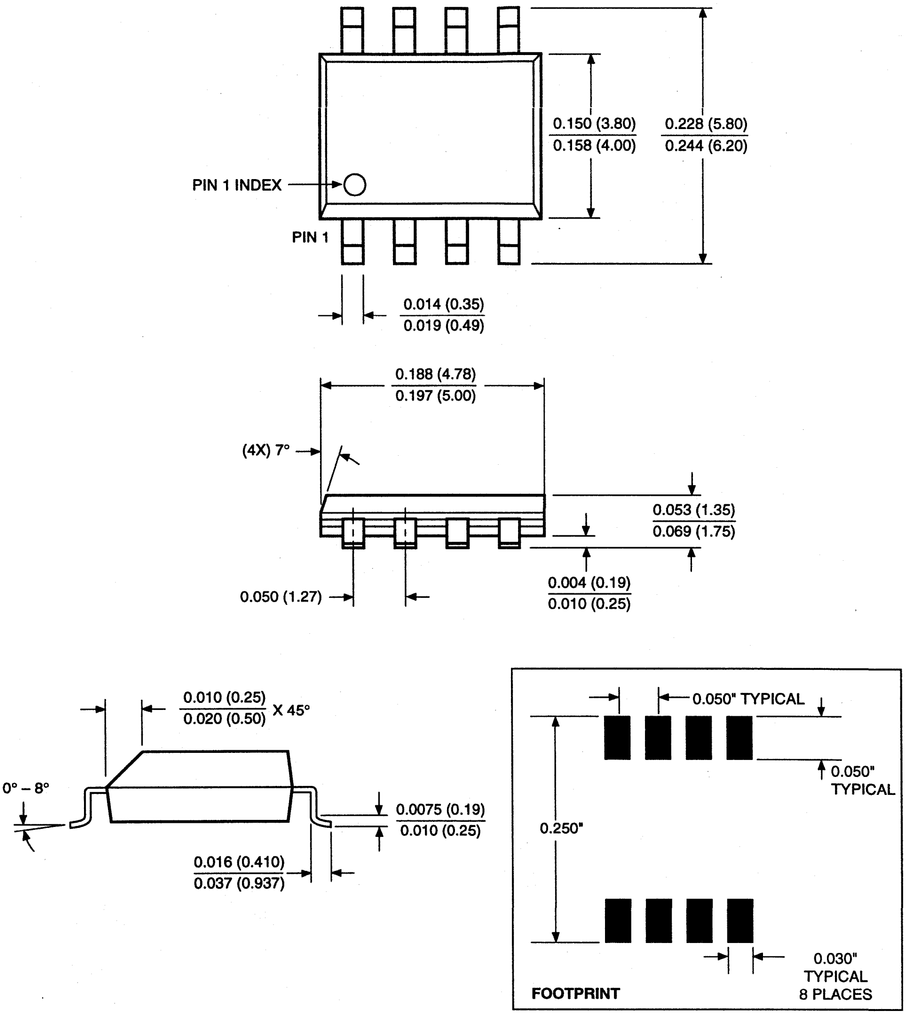
8-LEAD PLASTIC DUAL IN-LINE PACKAGE TYPE P



NOTE:
1. **ALL DIMENSIONS IN INCHES (IN PARENTHESES IN MILLIMETERS)**
2. **PACKAGE DIMENSIONS EXCLUDE MOLDING FLASH**

3926 FHD F01

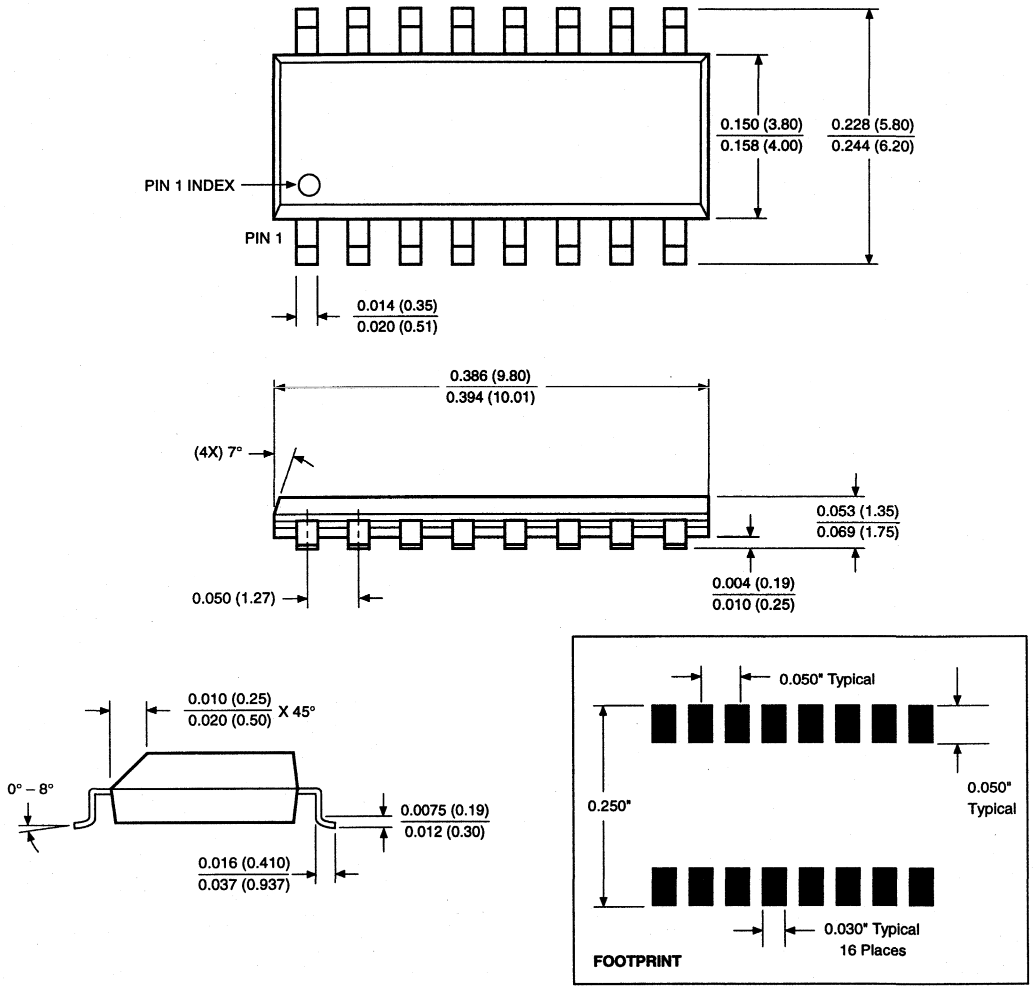
8-LEAD PLASTIC SMALL OUTLINE GULL WING PACKAGE TYPE S



NOTE: ALL DIMENSIONS IN INCHES (IN PARENTHESES IN MILLIMETERS)

3926 FHD F22.1

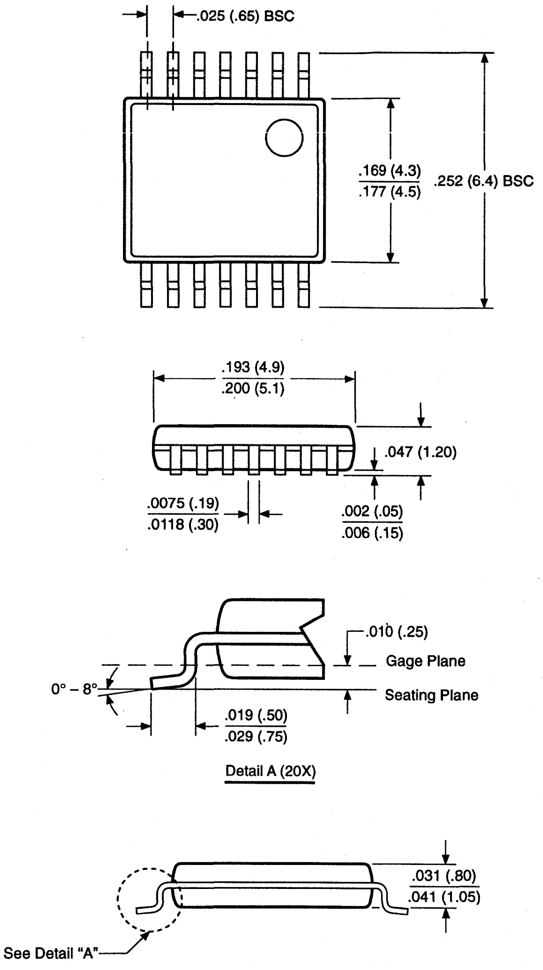
16-LEAD PLASTIC SMALL OUTLINE GULL WING PACKAGE TYPE S



NOTE: ALL DIMENSIONS IN INCHES (IN PARENTHESES IN MILLIMETERS)

3926 FHD F26

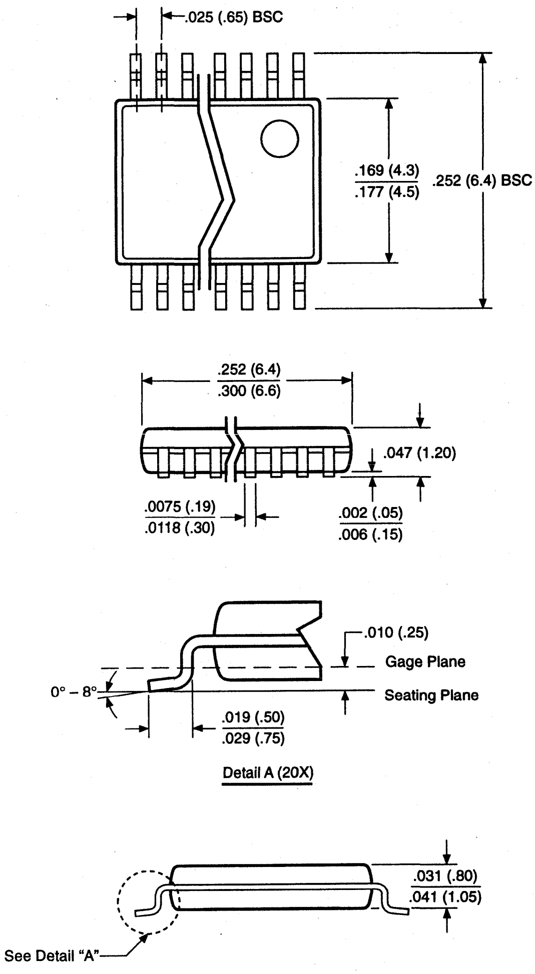
14-LEAD PLASTIC, TSSOP PACKAGE TYPE V



NOTE: ALL DIMENSIONS IN INCHES (IN PARENTHESES IN MILLIMETERS)

3926 FHD F32

20-LEAD PLASTIC, TSSOP PACKAGE TYPE V



Detail A (20X)

NOTE: ALL DIMENSIONS IN INCHES (IN PARENTHESES IN MILLIMETERS)

3926 FHD F45