AppKit 1-907/916

This AppkKit shows how to use the Xicor X25640 and X25128 EEPROMs with PIC microcontrollers and Parallax BASIC Stamps.
Both chips are used in an identiocal manner- the only difference being that the X25128 (16K) chip will accept addresses up to 16K rather than the 8K limit for the X25640 chip.

X25128 Chip
Has been discontinued
we have not found
a replacement
So this kit is
now only 1-907
with 8K chip in.

AppKit:
Using the Xicor X25640 8kB EEPROM

This AppKit shows how to use the Xicor X25640 EEPROM with PIC microcontrollers and the Parallax BASIC Stamp® single-board computer.

Description

The X25640 is an electrically erasable, programmable, read-only memory (EEPROM) device with 8192 bytes (8kB) of storage. Like all EEPROMs, it retains data with power off.

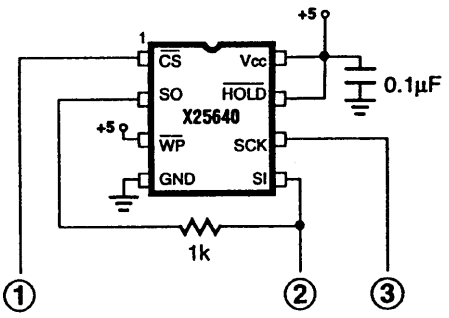
It is intended for applications in which data is read often and written infrequently, since write operations gradually wear it out. Xicor says that the X25640 will survive a minimum of 100,000 writes. Data stored in the EEPROM should remain intact without power for 100 years or more.

EEPROMs are typically used to store calibration tables, control settings, programs, maintenance logs, and small databases (such as lists of telephone numbers in a autodialer) that change fairly infrequently, but must be retained when the power is turned off. The X25640's relatively large storage capacity, simple interface, and low power consumption (100 µA inactive) make it well suited for data logging.

Hardware interface

The X25640's interface is compatible with Motorola's serial peripheral interface (SPI) bus. The basic connections are chip select (CS), which activates the device; serial clock (SCK), which shifts data into or out of it; and data, which consists of the serial in (SI) and serial out (SO) lines tied together.

There are two other control lines, hold (HOLD) and write-protect (WP), which are not used in these example applications. HOLD allows a busy, interrupt-equipped processor to pause a transaction with the X25640 while it tends to other business on the SPI bus. WP locks out any attempts to write to the device. Both of these lines are active low, so they are tied to Vcc to disable them.



	PIC	Stamp
①	ra.2	pin 0
②	ra.0	pin 2
③	ra.1	pin 1

The figure shows how to connect the X25640 to the PIC or Stamp for the demo programs. Do not omit the bypass capacitor—not even if you feel that your power supply is solid and well-filtered. Locate that cap as close as practical to the supply leads of the X25640. If you omit the bypass cap, you are almost certain to have intermittent communication with the X25640, especially during and after write operations.

Software interface

From a software standpoint, using the X25640 boils down to this:

(1) Activate CS by taking it low.
(2) Send an instruction (opcode) to the X25640 telling it what you want to do.
(3) If you are reading or writing a memory location, send the address.
(4) If you are reading data, shift it into the controller (PIC or Stamp).
(5) If you are writing data, shift it out to the X25640.
(6) Deactivate CS by making it high.

The program listings and Xicor data sheet show these processes in detail.

Tips for using the X25640

• When you first apply power to X25640, writes are disabled. After completion of a write cycle, the EEPROM also disables writes. As in the demo programs, you must send the write-enable instruction (WREN) before writing to the EEPROM. You can write-protect particular blocks of the EEPROM using the write-status-register (WRSR) instruction; see the Xicor data sheet, page 4.

• After you have written data to the X25640, it begins an internal programming cycle. It takes as much as 10 milliseconds (ms) to complete the write. PIC and BASIC Stamp II users should check the busy flag by reading the status register as shown in the program listings before attempting further reads or writes. The busy bit is least-significant bit (lsb) of the status register and reads high when busy. Users of the Stamp I will find that the write cycle is complete in the time before the Stamp can read the status register, so it's generally safe to ignore the busy bit or program a short time delay to ensure that the programming cycle is done.

• If you need to write to several sequential addresses of the X25640, take advantage of the page-write capability, described on pages 4 and 6 of the Xicor data sheets. You can write up to 32 bytes sequentially, provided that you don't cross a page boundary (located at 32n-1: 31, 63, 95, 127...). If you write beyond the edge of a 32-byte page boundary, you will overwrite the lowest addresses of the same page. For example, say you write 5 bytes starting at address 29. The bytes would be written to addresses 29, 30, 31, 0, 1. The advantage to a page write is that up to 32 bytes may be stored in a single 10-ms programming cycle. A page write starts exactly like the writes demonstrated in the programs, but instead of raising CS after one data byte, hold CS low and keep feeding additional bytes to the chip using the Shout subroutine. When you're finished sending bytes, raise CS.

PIC Program Listing

```
; Program: XICOR8K.SRC
; Routines to read and write the XICOR X25640 8k serial EEPROM
; via a three-wire serial interface.

; Connect the PIC pins below to the like-named pins of the X25640.
; If you change these connections, be sure to change these equates
; <<AND>> the values DataIn and DataOut so that the DATA line is
; an input during reads and an output during writes. If the EEPROM
; shares a port with other I/O, make sure that the values you give
; to DataIn and DataOut correctly set the data direction for the
; other pins as well. For the purposes of this program, DataIn just
; consists of a 1 in the bit position corresponding to the DATA pin
; assignment. Since the DATA pin is in position 0 (ra.0), DataIn
; is 0001.

DATA        =       ra.0
SCK         =       ra.1
CS          =       ra.2
DataIn      =       1
DataOut     =       0

; Opcodes for XICOR 25640:
WREN        =       6           ; Write-enable.
WRDI        =       4           ; Write-disable.
RDSR        =       5           ; Read status register.
WRSR        =       1           ; Write status register (not used here).
READ        =       3           ; Read the EEPROM.
WRITE       =       2           ; Write to the EEPROM.

            org     8
shifts      ds      1           ; Shift counter.
addrH       ds      1
addrL       ds      1
EEdata      ds      1
temp        ds      1

; Device data and reset vector
            device  pic16c55,xt_osc,wdt_off,protect_off
            reset   start
            org     0

; This program write-enables the EEPROM, writes a byte to one of its
; memory locations, waits for the write-in-progress bit of the status
; register to reset (indicating that the write is complete), then
; reads back the previously written data.

start       mov     ra,#4       ; Initialize ra with CS high (EEPROM off).
            mov     !ra,#DataOut ; Set data direction to output.
            mov     !rc,#0      ; Make all of rc output for LEDs.
            clr     rc
```

PIC Program Listing (cont)

```
                call    Write_enable    ; Send write-enable opcode to EEPROM.
                mov     addrH,#01fh     ; Now set up the EEPROM address (0-1FFFh).
                mov     addrL,#0ffh
                mov     EEdata,#99      ; And data to be written to that address.
                call    Write_byte      ; Write the data.
                call    Write_disable

:loop           call    Read_Status     ; Get status byte and check its 0th bit.
                jb      EEdata.0,:loop  ; The bit will stay high until write is done.

; For the sake of this demo, we move the address into the high and low
; address bytes (addrH and addrL), but it's not really necessary. They
; weren't altered by Write_byte, so we could really skip this step.

:read           mov     addrH,#01fh     ; Rewrite the EEPROM address.
                mov     addrL,#0ffh
                call    Read_byte       ; Read the byte.
                mov     rc,EEdata       ; Show the recovered data on LEDs
                jmp     $               ; Stop the program here.

; Subroutine shifts the eight bits of the byte pointed to
; by the fsr out to the SI pin of the X25640.

Shout           mov     shifts,#8
:loop           clrb    SCK
                rl      indirect
                movb    DATA,c
                setb    SCK
                djnz    shifts,:loop
                rl      indirect
                ret

; Subroutine shifts one byte of serial data from the XDATA
; pin into the byte pointed to by the fsr. The nop may be required
; at PIC clock speeds above 4 MHz, especially if the resistor
; from SI to SO is larger than 1k. If the EEPROM incorrectly
; sets the msbs of bytes stored at odd-numbered addresses, you
; need a smaller resistor, the nop, or both. (The explanation for
; the odd symptom is that the lsb of the address output shows up
; in the msb of the data input. XICOR says that the X25640 may
; require up to 400 ns from a falling clock edge to valid data
; at SO. Combine this with whatever time SO needs to go from
; hi-Z to output, and you've got this telltale error.)

Shin
                mov     !ra,#DataIn
                mov     shifts,#8
:loop           clrb    SCK
                nop                             ; Nop may be required at high speeds.
                movb    c,DATA
```

PIC Program Listing (cont)

```
                rl          indirect
                setb        SCK
                djnz        shifts,:loop
                mov         !ra,#DataOut
                ret

; Returns the contents of the EEPROM's status register in
; EEdata. Uses EEdata to shift out the op code.

Read_status
                mov         fsr,#EEdata
                mov         indirect,#RDSR
                clrb        CS
                call        Shout
                call        Shin
                setb        CS
                ret

; Reads the EEPROM data at the address specified by addrH and addrL
; and writes it to the location EEdata.

Read_byte
                mov         fsr,#temp
                mov         indirect,#READ
                clrb        CS
                call        Shout
                mov         fsr,#addrH
                call        Shout
                mov         fsr,#addrL
                call        Shout
                mov         fsr,#EEdata
                call        Shin
                setb        CS
                ret

; Turns off the EEPROM's write protection. Must be called
; after power up in order to write to the EEPROM.

Write_enable
                mov         fsr,#EEdata
                mov         indirect,#WREN
                clrb        CS
                call        Shout
                setb        CS
                ret
```

PIC Program Listing (cont)

```
; Turns on the EEPROM's write protection feature to prevent accidental
; overwriting of memory in the event that a program goes out of control.

Write_disable
        mov     indirect,#WRDI
        clrb    CS
        call    Shout
        setb    CS
        ret

; Writes the byte in EEdata to the EEPROM address specified by addrH
; and addrL. The contents of EEdata, addrH and addrL, are not
; changed, although the bits may be a little dizzy from being rapdily
; rotated by the trip through Shout.

Write_byte
        mov     fsr,#temp
        mov     indirect,#WRITE
        clrb    CS
        call    Shout
        mov     fsr,#addrH
        call    Shout
        mov     fsr,#addrL
        call    Shout
        mov     fsr,#EEdata
        call    Shout
        setb    CS
        ret
```

Stamp I (BS1) Program Listing

```
' Program: XICOR8K.BS1
' This program demonstrates subroutines for storing data in a
' XICOR X25640 serial EEPROM using the original BASIC Stamp (BS1).

Symbol   CS       = 0        ' Chip-select line to pin 0.
Symbol   CLK      = 1        ' Clock line to pin 1.
Symbol   DATA     = pin2     ' Destination of Shout; input to Shin
Symbol   DATA_N   = 2        ' Pin # of DATA for "input" & "output"
Symbol   ReadEE   = 3        ' EEPROM opcode for read.
Symbol   WREN     = 6        ' EEPROM opcode to enable writes.
Symbol   WRDI     = 4        ' EEPROM opcode to disable writes.
Symbol   RDSR     = 5        ' EEPROM opcode to read the status register.
Symbol   WriteEE  = 2        ' EEPROM opcode for write.
Symbol   i        = b6       ' Counter for shift loops
Symbol   EEdata   = b4       ' Data byte to/from EEPROM
Symbol   EEaddr   = w1       ' 16-bit EEPROM address.
Symbol   Shreg    = b1       ' Shift register used by Shin/Shout.

' Beginning of demonstration program. We're just going to write-enable
' the EEPROM, write a value to one of its addresses, then retrieve
' that value.

Demo:
    output DATA_N            ' Output to EEPROM data lines.
    high CS                  ' Deactivate EEPROM.

' This chunk of code sends the write-enable opcode to the EEPROM.
' To disable writes, use this same code, but substitute WRDI for WREN.
    let Shreg = WREN         ' Write-enable opcode.
    low CS                   ' Activate the EEPROM.
    gosub Shout              ' Send the opcode.
    high CS                  ' Deactivate the EEPROM.
' ================================================================
' Now write some data to the EEPROM. The legal range of addresses
' is 0 to 8191.
    let EEdata = 195         ' Sample data.
    let EEaddr = 800         ' Set an address.
    gosub EEwrite            ' Write the data.
    pause 10                 ' Wait for write cycle to finish.
    gosub EEread             ' Read back the data.
    debug EEdata             ' Show us.
here:
    goto here                ' Endless loop to terminate program.
```

BS1 Program Listing (cont)

```
' ================================================================
' The subroutines for reading and writing the EEPROM.
' ================================================================

' Shift data into Shreg.
Shin:
    input DATA_N            ' Change the data line to input.
    let Shreg=0             ' Clear data byte.
    for i = 1 to 8          ' Prepare to get 8 bits.
    low CLK                 ' Data valid on falling edge.
    let Shreg=Shreg*2       ' Shift Shreg to the left.
    let bit8=DATA           ' Move data to lsb of variable.
    high CLK                ' End of clock pulse.
    next i                  ' Get another bit.
    output DATA_N           ' Restore data line to output.
    return

' Shift data out of Shreg.
Shout:
    for i = 1 to 8          ' Shift out 8 bits.
    low CLK                 ' Clear clock bit.
    let DATA=bit15          ' Get bit 7 of Shreg.
    let Shreg=Shreg*2       ' Shift register to the left.
    high CLK                ' Data valid on rising edge of clock.
    next i                  ' Send another bit.
    return

' Write EEdata to EEaddr.
EEwrite:
    let Shreg = WriteEE     ' Set up the write opcode.
    gosub Send_addr         ' Send it and the address.
    let Shreg = EEdata      ' Now send the data.
    gosub Shout
    high CS                 ' Deactivate the EEPROM.
    return

' Read data at EEaddr and put it into EEdata.
EEread:
    let Shreg = ReadEE      ' Set up the read opcode.
    gosub Send_addr         ' Send it and the address.
    gosub Shin              ' Now get the data.
    let EEdata = Shreg      ' Transfer from Shreg to EEdata.
    high CS                 ' Deactivate the EEPROM.
    return
```

BS1 Program Listing (cont)

```
' Send the opcode and address for write or read.
Send_addr:
    low CS                  ' Activate the EEPROM.
    gosub Shout             ' Send opcode in Shreg.
    let Shreg = b3          ' Send upper byte of address.
    gosub Shout
    let Shreg = b2          ' Send lower byte of address.
    gosub Shout
    return
```

Stamp II (BS2) Program Listing

```
' Program: XICOR8K.BS2
' This program demonstrates subroutines for storing data in a
' XICOR X25640 serial EEPROM using the BASIC Stamp II (BS2).

CS        con     0             ' Chip-select line to pin 0.
CLK       con     1             ' Clock line to pin 1.
DATA_N    con     2             ' Pin # (2) of DATA line.
ReadEE    con     3             ' EEPROM opcode for read.
WREN      con     6             ' EEPROM opcode to enable writes.
WRDI      con     4             ' EEPROM opcode to disable writes.
RDSR      con     5             ' EEPROM opcode to read status register.
WriteEE   con     2             ' EEPROM opcode for write.

EEaddr    var     word          ' EEPROM address for reads or writes.
EEdata    var     byte          ' Data written/read to/from EEPROM.
EEstats   var     byte          ' Copy of the EEPROM status register.
EEbusy    var     EEstats.bit0  ' EEPROM busy bit.

' Demonstration program. We're just going to write-enable
' the EEPROM, write a value to one of its addresses, then retrieve
' that value.

Demo:
  high CS                       ' Deactivate EEPROM.
  pause 100                     ' Wait for settling.
  EEdata = 113                  ' Write the value 113 to address 8191.
  EEaddr = 8191
  gosub EEnable                 ' Enable the EEPROM for writes.
  gosub EEwrite                 ' Write the data.
is_busy:                        ' Stay in a loop until EEPROM finishes
write.
  gosub Read_stats              ' Read status register.
  if EEbusy = 1 then is_busy    ' Wait until busy bit says 'done.'
  gosub EEread                  ' Now read back the data and..
  debug ? EEdata                ' ..display it on the PC screen.

STOP                            ' End program.

' ================================================================
' The subroutines for reading and writing the EEPROM.
' ================================================================

' Write-enable the EEPROM.
EEnable:
  low CS                                ' Activate the EEPROM.
  shiftout DATA_N,CLK,msbfirst,[WREN]   ' Send the enable opcode.
  high CS                               ' Deactivate the EEPROM.
return                                  ' Return to program.
```

BS1 Program Listing (cont)

```
' Write-protect (disable) the EEPROM.
EEdisable:
  low CS                                    ' Activate the EEPROM.
  shiftout DATA_N,CLK,msbfirst,[WRDI]       ' Send the disable opcode.
  high CS                                   ' Deactivate the EEPROM.
return                                      '

' Write data to the EEPROM. Use the address in EEaddr
' and data in EEdata.
EEwrite:
  low CS          ' Activate the EEPROM.
  shiftout DATA_N,CLK,msbfirst,[WriteEE]    ' Send write opcode.
  gosub Send_addr                           ' Send the address.
  shiftout DATA_N,CLK,msbfirst,[EEdata]     ' Send the data.
  high CS                                   ' Deactivate the EEPROM.
return                                      ' Return to program.

' Read data from the EEPROM. Use the address in EEaddr;
' put data into EEdata.
EEread:
  low CS          ' Activate the EEPROM.
  shiftout DATA_N,CLK,msbfirst,[ReadEE]     ' Send the read opcode.
  gosub Send_addr                           ' Send the address.
  shiftin DATA_N,CLK,msbpre,[EEdata]        ' Send the data.
  high CS                                   ' Deactivate the EEPROM.
return                                      ' Return to program.

' Send the address for a read or write.
Send_addr:
  shiftout DATA_N,CLK,msbfirst,[EEaddr\16]  ' Shift the address out.
return                                       ' Return to program.

' Read the EEPROM status register.
Read_stats:
  low CS                                    ' Activate the EEPROM.
  shiftout DATA_N,CLK,msbfirst,[RDSR]       ' Send read-status opcode.
  shiftin DATA_N,CLK,msbpre,[EEstats]       ' Get the status register.
  high CS                                   ' Deactivate the EEPROM.
return                                      ' Return to program.
```

Advance Information

64K X25640 8K x 8 Bit

SPI Serial E²PROM

FEATURES

- 1MHz Clock Rate
- SPI Modes 0 & 3
- 8K X 8 Bits
  - —32 Byte Page Mode
- Low Power CMOS
  - —100µA Standby Current
  - —5mA Active Current
- 2.7V To 5.5V Power Supply
- Block Write Protection
  - —Protect 1/4, 1/2 or all of E²PROM Array
- Built-In Inadvertent Write Protection
  - —Power-Up/Power-Down protection circuitry
  - —Write Latch
  - —Write Protect Pin
- Self-Timed Write Cycle
  - —5mS Write Cycle Time (Typical)
- High Reliability
  - —Endurance: 100,000 cycles per byte
  - —Data Retention: 100 Years
  - —ESD protection: 2000V on all pins
- 8-Pin Mini-DIP Package
- 14-Pin SOIC Package

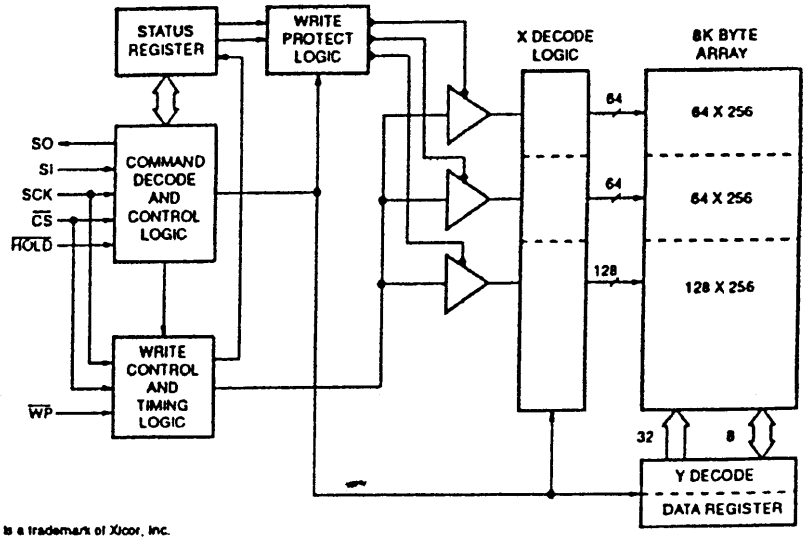
DESCRIPTION

The X25640 is a CMOS 65536 bit serial E²PROM, internally organized as 8K x 8. The X25640 features a 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 X25640 also features two additional inputs that provide the end user with added flexibility. By asserting the $\overline{HOLD}$ input, the X25640 will ignore transitions on its inputs, thus allowing the host to service higher priority interrupts. The $\overline{WP}$ input can be used as a hardwire input to the X25640 disabling all write 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 X25640 utilizes Xicor's proprietary Direct Write™ cell, providing a minimum endurance of 100,000 cycles per byte and a minimum data retention of 100 years.

FUNCTIONAL DIAGRAM



Direct Write™ is a trademark of Xicor, Inc.

Characteristics subject to change without notice

2255

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 ($\overline{CS}$)

When $\overline{CS}$ is high, the X25640 is deselected and the SO output pin is at high impedance and unless an internal write operation is underway the X25640 will be in the standby power mode. $\overline{CS}$ low enables the X25640, placing it in the active power mode. It should be noted that after power-on, a high to low transition on $\overline{CS}$ is required prior to the start of any operation.

Write Protect ($\overline{WP}$)

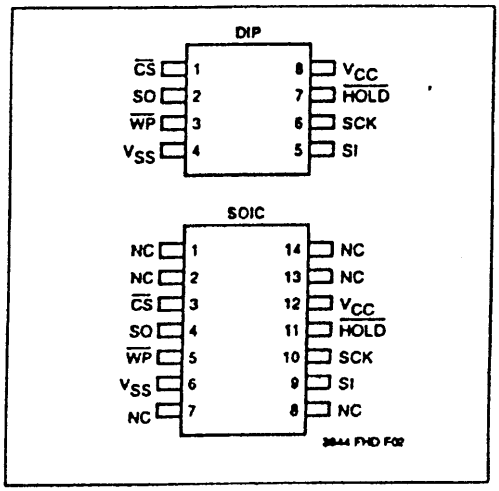
When $\overline{WP}$ is low and the nonvolatile bit WPEN is high, nonvolatile writes to the X25640 status register are disabled, but the part otherwise functions normally. When $\overline{WP}$ is held high, all functions, including nonvolatile writes operate normally. $\overline{WP}$ going low while $\overline{CS}$ is still low will interrupt a write to the X25640 status register. If the internal write cycle has already been initiated, $\overline{WP}$ going low will have no affect on write.

The $\overline{WP}$ pin function is blocked when the WPEN bit in the status register is low. This allows the user to install the X25640 in a system with $\overline{WP}$ pin grounded and still be able to write to the status register. The WP pin functions will be enabled when the WPEN bit is set high.

Hold ($\overline{HOLD}$)

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

PIN CONFIGURATION



PIN NAMES

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

PRINCIPLES OF OPERATION

The X25640 is a 8K x 8 E^2PROM designed to interface directly with the synchronous serial peripheral interface (SPI) of the popular 6805 and 68HC11 microcontroller families.

The X25640 contains an 8-bit instruction register. It is accessed via the SI input, with data being clocked in on the rising SCK. $\overline{CS}$ must be low and the $\overline{HOLD}$ and $\overline{WP}$ inputs must be high during the entire operation.

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 $\overline{CS}$ 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 $\overline{HOLD}$ input to place the X25640 into a "PAUSE" condition. After releasing $\overline{HOLD}$, the X25640 will resume operation from the point when $\overline{HOLD}$ was first asserted.

Write Enable (WREN) and Write Disable (WRDI)

The X25640 contains a write enable latch. This latch must be SET before a write operation will be completed internally. The WREN instruction will set the latch and the WRDI instruction will reset the latch. This latch is automatically reset upon a power-on condition and after the completion of a byte, page, or status register write cycle.

Read Status Register (RDSR)

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

7	6	5	4	3	2	1	0
WPEN	X	X	X	BP1	BP0	WEL	WIP

The Write-In-Process (WIP) bit indicates whether the X25640 is busy with a write operation. When set to a "1" a write is in progress, when set to a "0" no write is in progress. During a write all other bits are set to "1".

The Write Enable Latch (WEL) bit indicates the status of the write enable latch. When set to a "1" the latch is set, when set to a "0" the latch is reset.

The Block Protect (BP0 and BP1) bits indicate the extent of protection employed. These bits are set by the user issuing the WRSR instruction.

Write-Protect Enable

The Write-Protect-Enable (WPEN) is available for the X25640 as an enable bit for the $\overline{WP}$ pin.

WPEN	$\overline{WP}$	WEL	Protected Blocks	Unprotected Blocks	Status Register
0	X	0	Protected	Protected	Protected
0	X	1	Protected	Writable	Writable
1	0	0	Protected	Protected	Protected
1	0	1	Protected	Writable	Protected
X	1	0	Protected	Protected	Protected
X	1	1	Protected	Writable	Writable

The Write Protect ($\overline{WP}$) pin and the Write Protect Enable (WPEN) bit in the Status Register control the programmable hardware write protect feature. Hardware write protection is enabled when $\overline{WP}$ pin is low, and the WPEN bit is high. Harware write protection is disabled when either the WP pin is high or the WPEN bit is low. When the chip is hardware write protected, nonvolatile writes are disabled to the Status Register, including the Block Protect bits and the WPEN bit itself, as well as the

Table 1. Instruction Set

Instruction Name	Instruction Format*	Operation
WREN	0000 0110	Set the Write Enable Latch (Enable Write Operations)
WRDI	0000 0100	Reset the Write Enable Latch (Disable Write Operations)
RDSR	0000 0101	Read Status Register
WRSR	0000 0001	Write Status Register
READ	0000 0011	Read Data from Memory Array beginning at selected address
WRITE	0000 0010	Write Data to Memory Array beginning at Selected Address (1 to 32 Bytes)

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

block-portected sections in the memory array. Only the sections of the memory arrary that are not block-protected can be written.

Note: Since the WPEN bit is write protected, it cannot be changed back to a low state, write protection disabled, as long as the $\overline{WP}$ pin is held low.

Write Status Register (WRSR)

The write status register instruction allows the user to select one of four levels of protection. The X25640 is divided into four 16384-bit 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 (write) data within the selected segments. The partitioning is controlled as illustrated below.

Status Register Bits		Array Addresses Protected
BP1	BP0	
0	0	None
0	1	$1800–$1FFF
1	0	$1000–$1FFF
1	1	$0000–$1FFF

DEVICE OPERATION

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

The $\overline{CS}$ line is first pulled low to select the device. The 8 bit read instruction is transmitted to the X25640, followed by the 16 bit address which the last 13 are used. 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 ($1FFF) 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 operation sequence illustrated in Figure 1.

Write Sequence

Prior to any attempt to write data into the X25640 the write enable latch must first be set by issuing the WREN instruction (See Fig. 2). $\overline{CS}$ is first taken low, then the instruction is clocked into the X25640. After all eight bits of the instruction are transmitted, $\overline{CS}$ must then be taken high. If the user continues the write operation without taking $\overline{CS}$ high after issuing the WREN instruction the write operation will be ignored.

Once the write enable latch is set, the user may proceed by issuing the write instruction, followed by the address and then the data to be written. This is minimally a thirty-two clock operation. $\overline{CS}$ must go low and remain low for the duration of the operation. The host may continue to write up to 32 bytes of data to the X25640. The only restriction is the 32 bytes must reside on the same page. If the address counter reaches the end of the page and the clock continues the counter will roll back to the first address of the page and overwrite any data that may have been written.

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

While the write is in progress the status register may be read to check the WIP bit. During this time the WIP 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 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 X25640 powers-on 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 write enable latch is reset.

Data Protection

The following circuitry has been included to prevent inadvertent writes:

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

Figure 1. Read Operation Sequence.

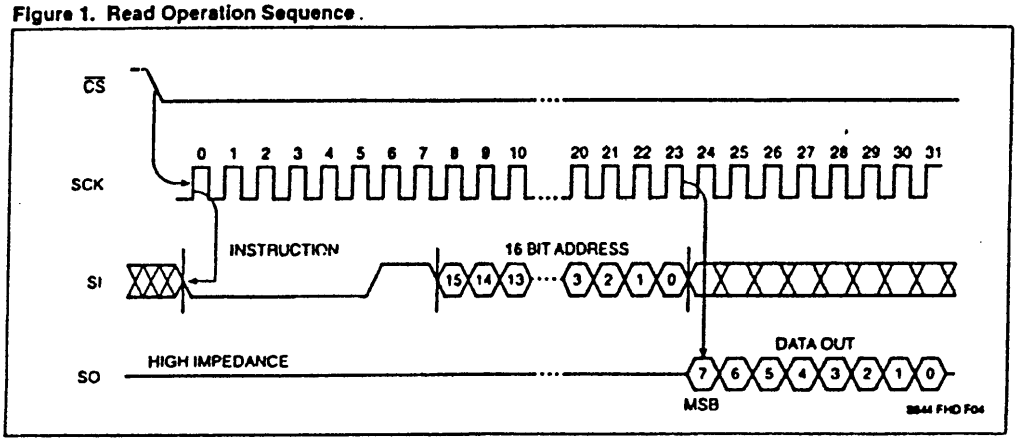


Figure 2. Write Enable Latch

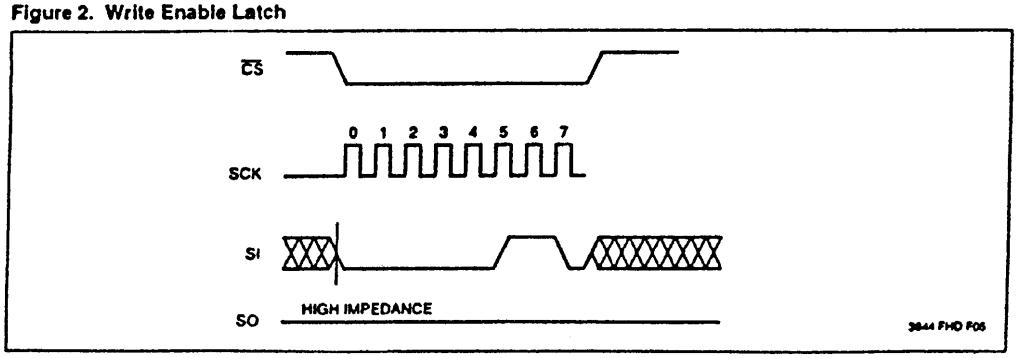


Figure 3. Write Operation Sequence

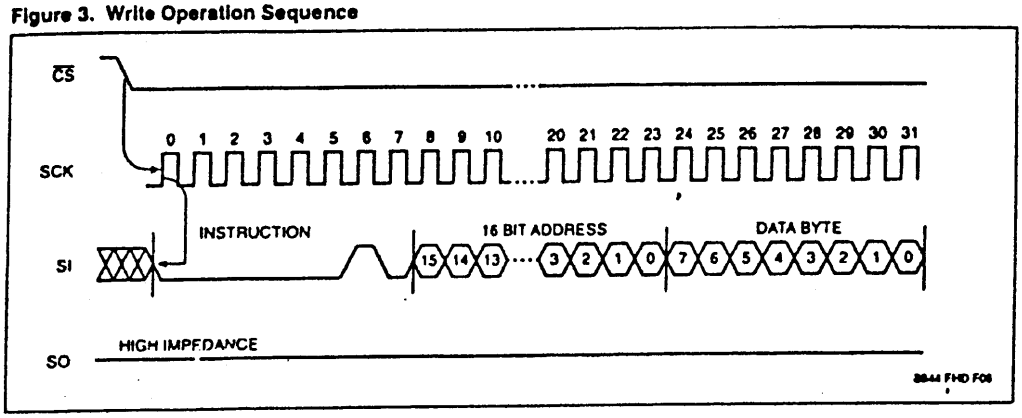
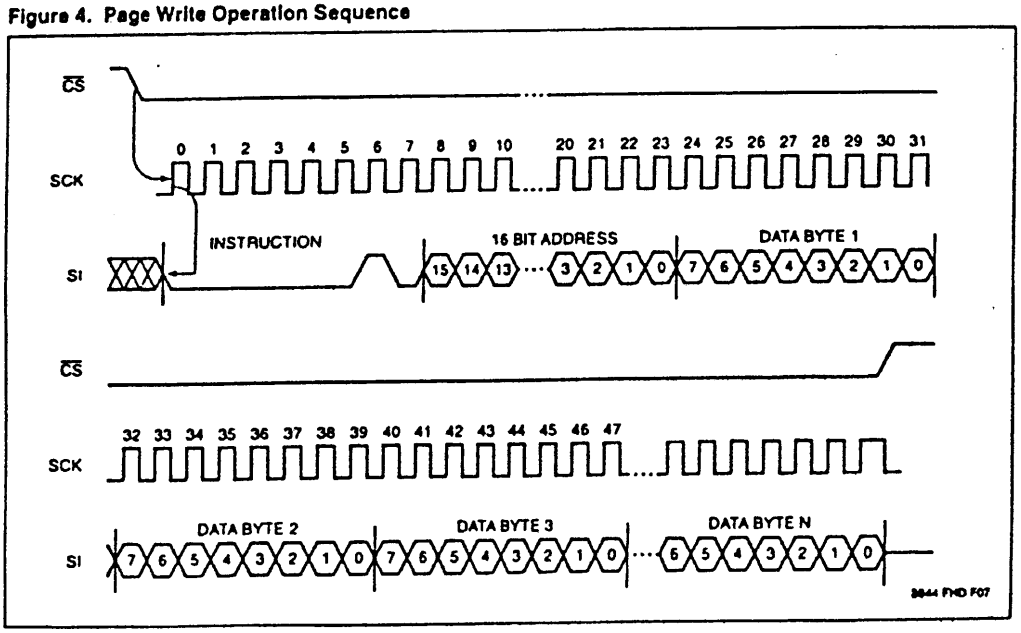


Figure 4. Page Write Operation Sequence



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 Ground –1.0V 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
Industrial	–40°C	+85°C
Military	–55°C	+125°C

Supply Voltage	Limits
X25640	5V ± 10%
X25640-3	3V to 5.5V

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

Symbol	Parameter	Limits		Units	Test Conditions
		Min.	Max.		
I_{CC}	V_{CC} Supply Current (Active)		5	mA	SCK = V_{CC} x 0.1/V_{CC} x 0.9 @ 1MHz, SO = OPEN, $\overline{CS}$ = Gnd
I_{SB1}	V_{CC} Supply Current (Standby)		100	µA	$\overline{CS}$ = V_{CC}, V_{IN} = Gnd or V_{CC} – 0.3V V_{CC} = 5.5V
I_{SB2}	V_{CC} Supply Current (Standby)		50	µA	$\overline{CS}$ = V_{CC}, V_{IN} = Gnd or V_{CC}, V_{CC} = 3V
V_{OH2}	Output High Voltage	V_{CC} – 0.4		V	I_{OH} = –0.4mA
I_{LI}	Input Leakage Current		10	µA	V_{IN} = GND to V_{CC}
I_{LO}	Output Leakage Current		10	µA	V_{OUT} = GND to V_{CC}
V_{IL}(1)	Input Low Voltage	–1.0	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} = 2mA
V_{OH1}	Output High Voltage	V_{CC}–0.8		V	I_{OH} = –1.0mA

POWER-UP TIMING

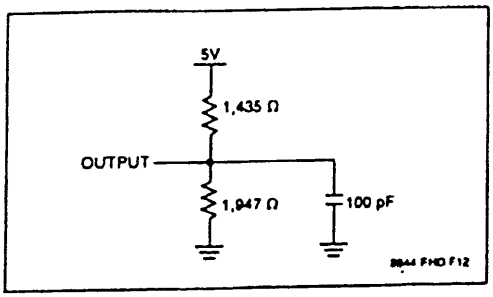
Symbol	Parameter	Min.	Max.	Units
t_{PUR}(1)	Power-up to Read Operation		1	ms
t_{PUW}(1)	Power-up to Write Operation		5	ms

CAPACITANCE T_A = 25°C, f = 1.0MHz, 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, CS, WP, HOLD)	6	pF	V_{IN} = 0V

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.

EQUIVALENT A.C. LOAD CIRCUIT, V_{CC} = 5V



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

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

Data Input Timing

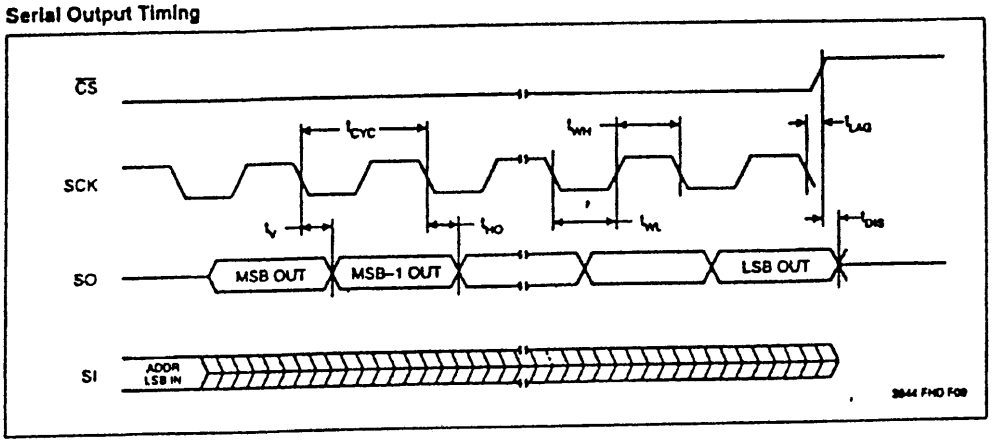
Symbol	Parameter	Min.	Max.	Units
f_{SCK}	Clock Frequency	0	1	MHz
t_{CYC}	Cycle Time	1000		ns
t_{LEAD}	$\overline{CS}$ Lead Time	500		ns
t_{LAG}	$\overline{CS}$ Lag Time	500		ns
t_{WH}	Clock High Time	400		ns
t_{WL}	Clock Low Time	400		ns
t_{SU}	Data Setup Time	100		ns
t_H	Data Hold Time	100		ns
t_{RI}(3)	Data In Rise Time		2.0	µs
t_{FI}(3)	Data In Fall Time		2.0	µs
t_{HD}	$\overline{HOLD}$ Setup Time	200		ns
t_{CD}	$\overline{HOLD}$ Hold Time	200		ns
t_{CS}	$\overline{CS}$ Deselect Time	500		ns
t_{WC}(4)	Write Cycle Time		10	ms

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}(3)	Output Rise Time		300	ns
t_{FO}(3)	Output Fall Time		300	ns
t_{LZ}(3)	$\overline{HOLD}$ High to Output in Low Z	100		ns
t_{HZ}(3)	$\overline{HOLD}$ Low to Output in High Z	100		ns

Notes: (3) This parameter is periodically sampled and not 100% tested.
(4) 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 write cycle.

Serial Output Timing



Serial Input Timing