AN1761

Interfacing the MC68HC705C8A to the X76F041 PASS™ SecureFlash™

By Mark Glenewinkel
Field Applications Engineering
Consumer Systems Group
Austin, Texas

Introduction

Secure non-volatile memory can be a necessary requirement in today's embedded systems. With the increased frequency of code pirating and data tampering, securing access to system code and data is a must. The X76F041 password access security supervisor (PASS) SecureFlash from Xicor, Inc., gives the user the ability to password protect sensitive memory. Although no security system is infallible, the X76F041 adds an extra layer of difficulty to deter system misuse.

Some security applications of using the X76F041 are:

- User password protected system access
- Use as a node identification with password in networked systems
- Redefinable 64-bit non-volatile security passwords

™ PASS and SecureFlash are trademarks of Xicor, Inc.

freescale™ semiconductor

The non-volatile memory of the X76F041 also provides additional applications such as:

- Storing system calibration constants
- Power down information storage for consumer electronics like TVs, VCRs, and hand-held portable devices
- Identification number storage for remote addressing
- Storage of telecommunication information like phone number recall and speed dialing
- Ability to add code patches to established systems

This application note describes the interface between the MC68HC705C8A (C8A) and the X76F041. Circuitry and example code are included to demonstrate the interface between the two parts.

X76F041 Overview

Features

The X76F041 provides these features:

- Four 128-byte memory arrays
- 8-byte sector write
- 64-bit password security
- Three password modes: read, write, and configuration
- Programmable configuration of:
  - Read, write, and configuration passwords
  - Multiple array access/functionality
  - Retry counter
- 2-wire serial interface
- 3 mA of active current, 50 μA standby current
- Two operating voltage ranges:
  - 3 volts to 3.6 volts
  - 4.5 volts to 5.5 volts

- High endurance; 100,000 cycles
- High data retention; 100 years
- 8-pin DIP (dual in-line) or SOIC (small outline integrated circuit) package

Description

The X76F041 contains four 128 x 8 bit SecureFlash arrays.

Access is controlled by three different 64-bit passwords:

- One for reading data
- One for writing data
- One for device configuration

In addition, a non-volatile password retry counter can be used to count password attempts. After a preset number of attempts has occurred, the device is disabled.

Each array can be individually programmed for access according to four options:

- Read and write
- Read only
- Read only and program (possible to change a bit from 1 to 0 but not 0 to 1)
- No read or write

The use of any one of the 64-bit passwords may be required for any of these options. These passwords are pre-configured as write only and cannot be read. The configuration password acts as a master password and can access any operation.

Data is transceived via the 2-wire bus. The bus signals are the clock input (SCL) and a bidirectional data input and output (SDA). Access is controlled with the chip-select ($\overline{CS}$) signal.

X76F041 Hardware Interface

Pinout and Pin Descriptions

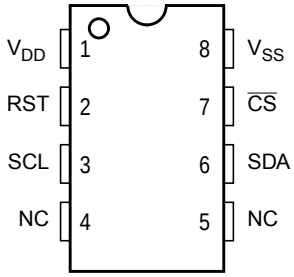


Figure 1. X76F041 Pinout

V_{DD} and V_{SS} — These pins serve as the power source for the device. Operating voltage is 5 volts or 3.3 volts.

SCL — The SCL pin is the clock input for the X76F041 2-wire serial interface.

SDA — The SDA pin is an I/O (input/output) pin used to transmit and receive data off the 2-wire serial interface. It is an open-drain pin that requires an external pullup resistor.

$\overline{CS}$ — When $\overline{CS}$ is HIGH, the X76F041 is deselected and the SDA pin turns into a high-impedance device. The X76F041 is in its standby mode. When $\overline{CS}$ goes LOW, the X76F041 is enabled and is operating in its active mode.

RST — The RST pin is used to reset the X76F041. When RST is pulsed HIGH while the $\overline{CS}$ pin is LOW (active mode), the device outputs 32 bits of fixed data which conforms to the ISO standard for “synchronous response to reset."

This feature is not used in this application note. For more information, consult the X76F041 data sheet from Xicor, Inc.

Block Diagram

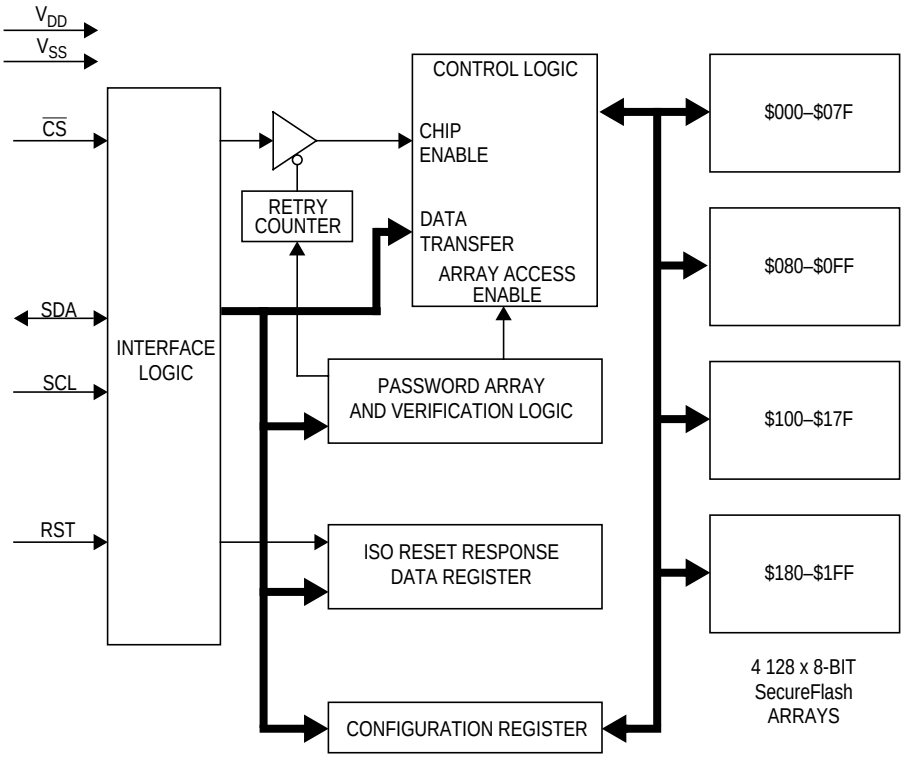


Figure 2. X76F041 Block Diagram

Serial Protocol

The X76F041 supports the bidirectional 2-wire protocol. The protocol has these characteristics:

- Any device sending out data is defined as a transmitter.
- Any device receiving data is defined as a receiver.
- The device controlling the transfer is called the master.
- The device being controlled is called the slave.
- The master initiates all transactions.
- The master always provides the clock for both transmit and receive operations.
- The X76F041 is always considered the slave.
- The clock signal is called SCL.
- The data signal is called SDA.
- All data is sent most significant bit (MSB) first.

Figure 3 shows the 2-wire bus interface between a master and slave.

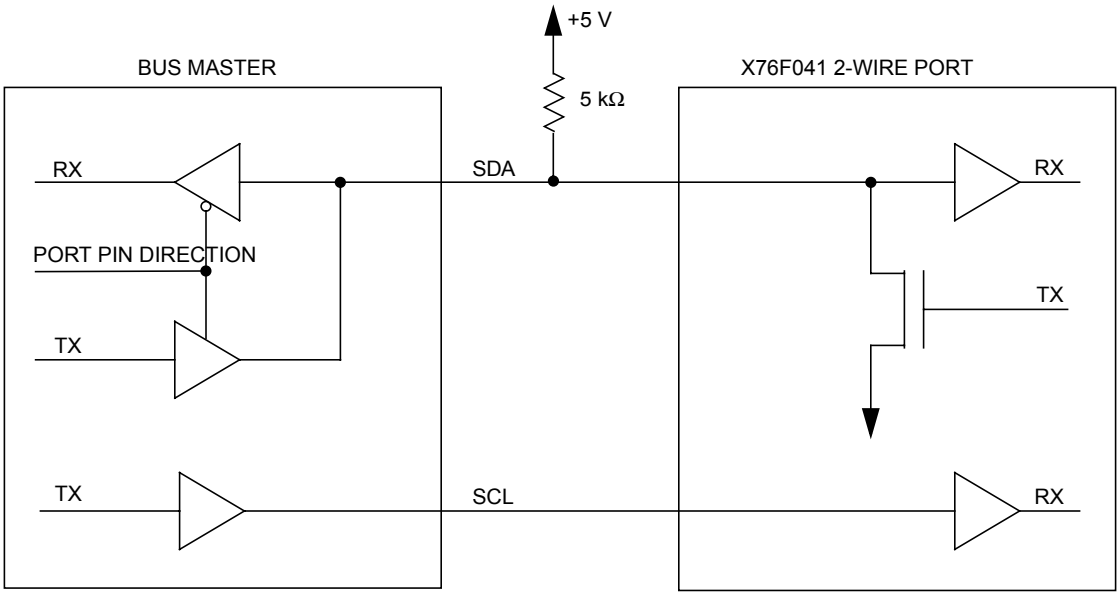


Figure 3. 2-Wire Serial Bus Interface

Bus Idle — In idle mode, both the SDA and SCL are held high.

Start Transfer — All transfers start with the start transfer condition. This is done by bringing the SDA pin from HIGH to LOW while the SCL pin is HIGH. The X76F041 is monitoring the bus for this signal and will not start any transactions until this condition is met. See Figure 4.

Stop Transfer — All transfers must be terminated with the stop transfer condition. This is done by taking the SDA pin from LOW to HIGH while the SCL pin is HIGH. A stop transfer can be used only after the transmitting device releases the bus. See Figure 4.

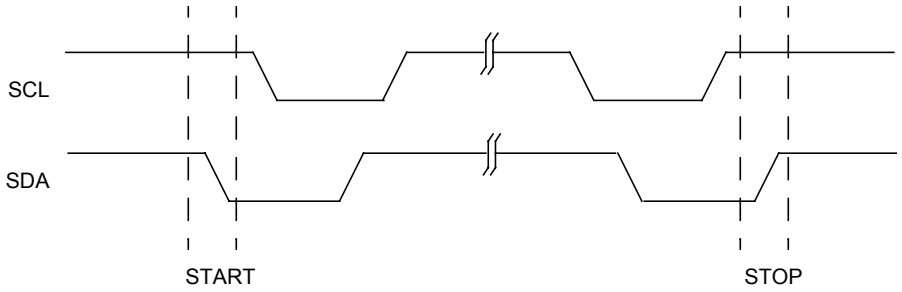


Figure 4. Start and Stop Transfer Timing

Data Transfer

Data is transmitted on the rising edge of SCL. Data can be changed only while SCL is LOW. The receiving device samples the bus after SCL goes HIGH. There is one clock pulse per bit of data transmitted. See **Figure 5**.

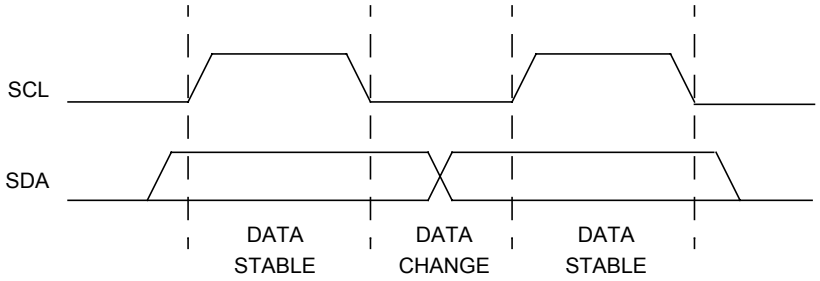


Figure 5. Data Transfer Timing

Acknowledge Transfer

The acknowledge transfer is a type of handshaking convention used to signify that a successful transfer of data has taken place. After the transmitting device sends out the eighth bit of a byte of data, it releases the bus. The master sends out a ninth clock signal and the receiver acknowledges the transfer by pulling SDA LOW. Once the transmitter reads the LOW condition of SDA, it proceeds by taking over the bus and sending out the next byte of data.

If the X76F041 is transmitting data and the master wants to end further transmissions, the master sends a NO ACK signal (HIGH) back to the X76F041. This tells the X76F041 that no more transfers are needed and the stop transfer condition will be initiated soon. See **Figure 6** for these different timing patterns.

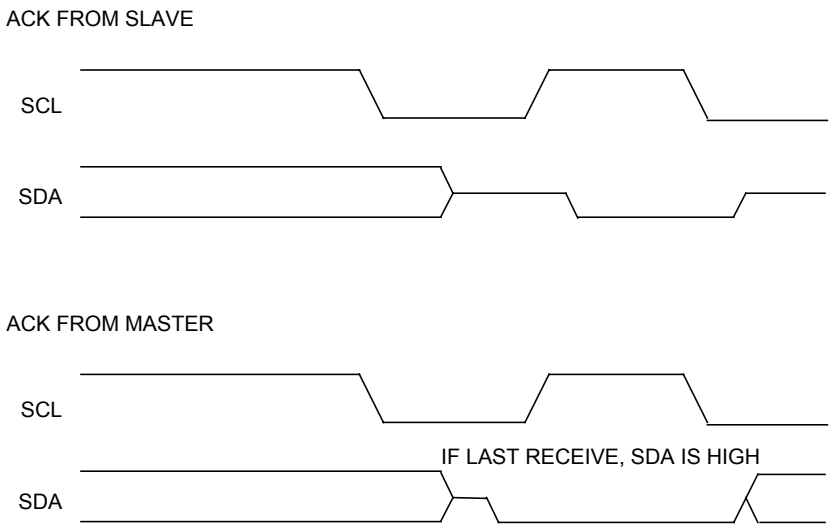


Figure 6. Acknowledge Timing

2-Wire Protocol Example

Here is an example of the protocol needed to write an 8-byte buffer to address $120 of the X76F041 with no password required:

1. The master transmits a start transfer.
2. The master transmits the X76F041 3-bit command code, %000, concatenated with the high bits of the address, %00001. This results in the value %00000001 transmitted to the X76F041.
3. Since a byte has just been transmitted, the receiver (X76F041) will now send out a LOW to acknowledge the transfer.
4. The master reads the SDA pin for a LOW.
5. The master sends out the low byte address of $20 to the X76F041 and receives back an acknowledge.
6. The master sends out the eight bytes of data. After each byte is transmitted, an ACK signal from the X76F041 is received.
7. Finally, a stop transfer is sent to the X76F041 to complete the transaction.
8. Since this write command requires programming of the FLASH, wait for at least 10 ms before executing any other commands to the X76F041.

X76F041 Software Interface

Memory Map

The X76F041 has a total of 4096 bits of SecureFlash. It is arranged in four sectors of 1024 bits. Most commands address each sector in 8-byte sections. The X76F041's memory map is shown in **Figure 7** with addresses shown for 8-byte segments of memory. This results in 16 sets of 8-byte segments per sector.

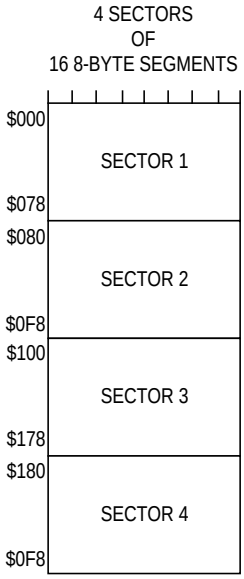


Figure 7. X76F041 Memory Map

Password Registers

The X76F041 passwords consist of three 64-bit write-only registers.

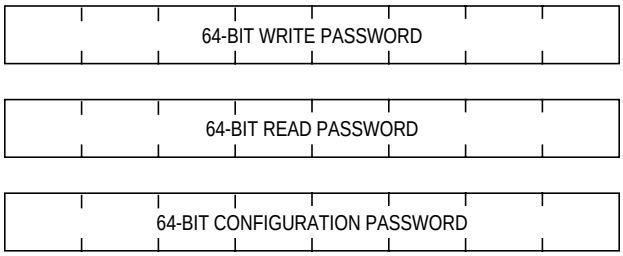


Figure 8. Password Registers

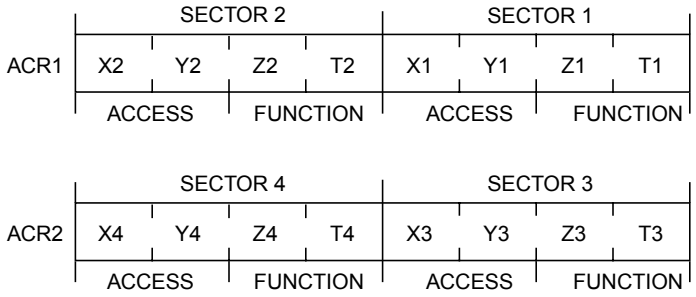
Configuration Registers

Five 8-bit configuration registers are used to control the X76F041. They are written to and read from in this sequence:

- ACR1 — Array control register 1
- ACR2 — Array control register 2
- CR — Configuration register
- RR — Password retry register
- RC — Password retry counter

Array Control Registers 1 and 2

Each memory sector can be programmed with different levels of access and functionality. **Figure 9** details ACR1 and ACR2 register options.



FUNCTION BITS

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

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

Figure 9. ACR1 and ACR2 Registers

Configuration Register

The configuration register contains four bits to control unauthorized access and to enable the retry counter. See **Figure 10**. This register gives added system protection by allowing the user to set a number of password attempts that are allowed before access is totally cut off.

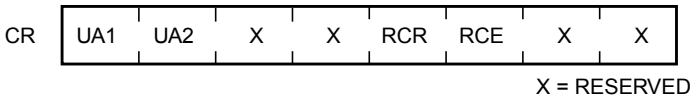


Figure 10. Configuration Register

UA1 and UA2 — Unauthorized Access Bits

10 — Access is forbidden if retry register equals the retry counter, no further access of any kind will be permitted

00, 01, 11 — Only configuration operations are allowed if the retry register equals the retry counter

RCR — Retry Counter Reset Bits

1 = Retry counter will be reset following the correct password.
0 = Retry counter will not be reset following the correct password.

RCE — Retry Counter Enable Bits

1 = Retry counter is enabled.
0 = Retry counter is disabled.

Device Operation

Once the $\overline{CS}$ pin is LOW, the X76F041 will accept commands off the serial bus. Communication is started by issuing a start condition followed by a command and address field. If any passwords are needed, they are sent next. After the password is verified, data is then read or written to the device. All bytes of data must be followed by an ACK condition. Refer to **Figure 11**.

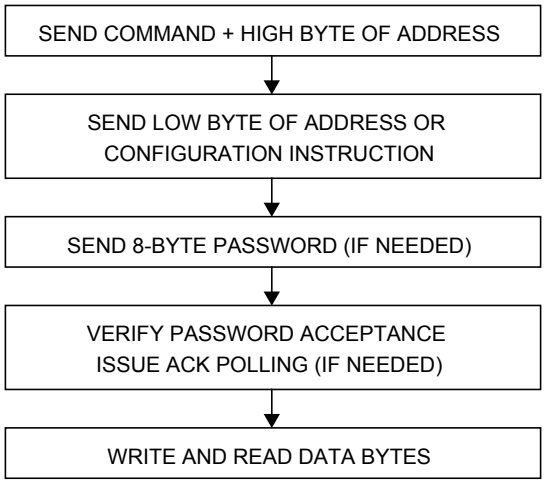


Figure 11. Device Operation Sequence

All devices are shipped from the factory in the non-password mode (all 0s). After a password is sent to the X76F041, a waiting period of at least

10 ms must be observed. To continue the transaction, the master must then initiate an ACK polling sequence. The code $C0 is continually transmitted and checks for an ACK back from the X76F041 if the password is correct.

Commands

The commands or modes of operations for the X76F041 are listed in **Table 1**. For brevity, the write with no password and read with no password (sequential) protocols are described in **Figure 12** and **Figure 13**. These serve as examples of the protocols needed to create the commands listed in **Table 1**. For a detailed protocol listing of each command, consult the X76F041 data sheet.

Table 1. X76F041 Command Set

Command Description	First Byte	Second Byte	Password Used
Write to a sector	000XXXXA	Write address	Write
Read (random or sequential)	001XXXXA	Read address	Read
Write to a sector	010XXXXA	Write address	Con gur ation
Read (random or sequential)	011XXXXA	Read address	Con gur ation
Program write password	100XXXXX	00000000	Write
Program read password	100XXXXX	00010000	Read
Program con gur ation password	100XXXXX	00100000	Con gur ation
Reset the write password to $00	100XXXXX	00110000	Con gur ation
Reset the read password to $00	100XXXXX	01000000	Con gur ation
Program con gur ation registers	100XXXXX	01010000	Con gur ation
Read con gur ation registers	100XXXXX	01100000	Con gur ation
Mass program (all 0s)	100XXXXX	01110000	Con gur ation
Mass erase (all 1s)	100XXXXX	10000000	Con gur ation

Write Sector with No Password

The first byte transmitted in a write to the X76F041 is its 3-bit command code concatenated with the high byte of the address being written. The X76F041 then transmits the low byte of the address. A buffer of eight bytes is then sent to the X76F041. The transmission is stopped by issuing a stop condition on the bus.

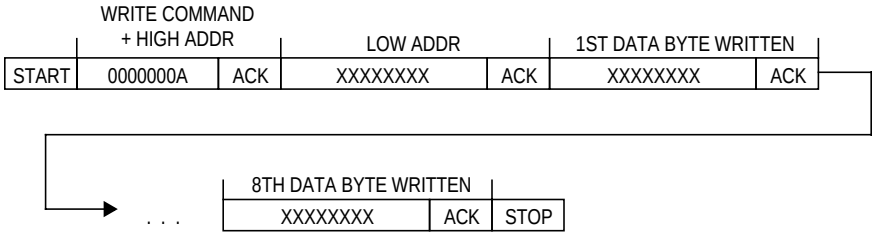


Figure 12. Write with No Password Sequence

Read Sector with No Password (Sequential)

The first byte transmitted in a read from the X76F041 is its 3-bit command code concatenated with the high byte of the address being written. The X76F041 then transmits the low byte of the address. When executing a sequential read, an unlimited amount of bytes can be read from the sector. Once address 127 of the sector is reached, the address pointer wraps around to address 0 of the sector. The sequence shown in **Figure 13** shows a process of reading out eight bytes only. Each read is followed by an ACK from the master except for the last byte read. The transmission is then stopped by issuing a stop condition on the bus.

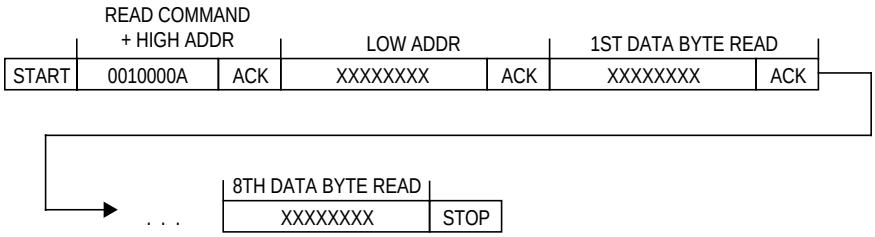


Figure 13. Read with No Password Sequence

MC68HC705C8A Hardware Interface

The MC68HC705C8A (C8A) is one of the most popular members of the HC05 Family. It has a total of 7744 bytes of erasable programmable read-only memory (EPROM) and 176 bytes of RAM. The part includes a total of 24 I/O pins and seven input-only pins. Peripherals include the serial peripheral bus (SPI), serial communications interface (SCI), and a 16-bit capture/compare timer.

The schematic used for testing the C8A to X76F041 interface on the MMEVS development system is shown in **Figure 14**. The pins used to drive the X76F041 on the C8A are:

- Port A, bit 0 — This I/O pin (SCL) is configured as an output to drive the serial clock pin, SCL, of the X76F041.
- Port A, bit 1 — This I/O pin (SDA) is used to transmit and receive data on the SDA pin of the X76F041.

For further information on the C8A, consult *MC68HC705C8A Technical Data*, Freescale document order number MC68HC705C8A/D.

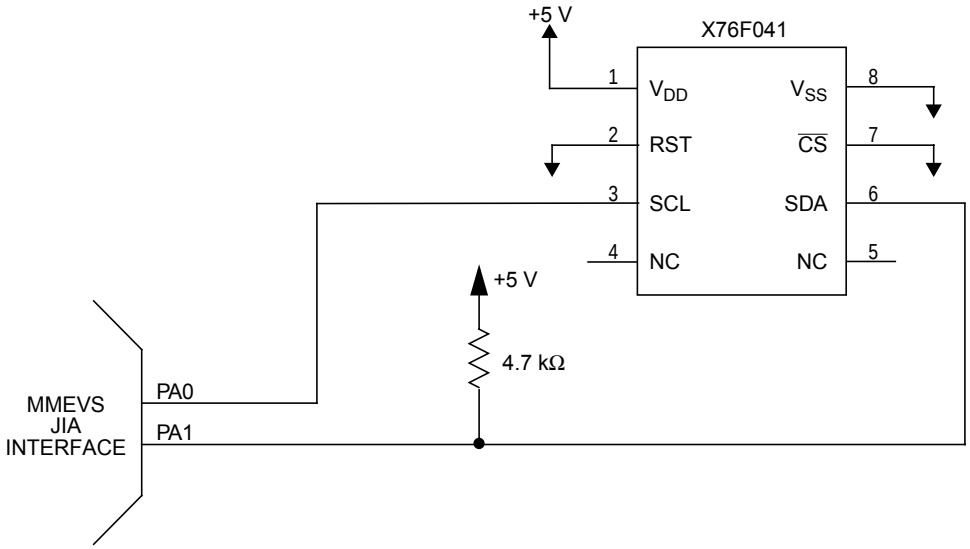


Figure 14. MC68HC705C8A to X76F041 Interface Test Circuit

MC68HC705C8A Software Interface

Serial interface Routines

I/O driving or manipulation is the process of toggling I/O pins with software instructions to create a certain hardware peripheral. The HC05 CPU provides special instructions to specifically manipulate single I/O pins. Five subroutines were created to provide an easy application programming interface (API) for the 2-wire serial interface.

These routines are:

- START_SER — Sends out a start condition on the bus
- STOP_SER — Sends out a stop condition on the bus
- ACK_POLL — Sends out the code $C0 and accepts an acknowledge back from the X76F041
- TXD — The master takes the contents of AccA and transmits it MSB (most significant bit) first to the X76F041. The master also checks for acknowledgement from the X76F041.
- RXD — After the master addresses the X76F041 with a command code and address or protocol byte, the X76F041 transmits a byte of data back to the master. This routine reads that byte and puts it into AccA. The master then generates an acknowledgment back to the X76F041.
- RXD_LAST — This routine is just like RXD but it is used for the last byte read from the X76F041. It does not generate an acknowledgment back to the X76F041.
- NV_WAIT — This routine is a 10-ms loop that waits for the non-volatile programming time for the X76F041 to expire.

The flowcharts for the X76F041 serial I/O drivers are shown in **Figure 15**, **Figure 16**, **Figure 17**, and **Figure 18**. These routines were written especially for the X76F041 and may not be able to properly drive other MCU peripherals with 2-wire serial buses.

Command Routines

The command routines were generated by using the 2-wire serial subroutines as building blocks. The command subroutines are:

- WRITE_NO_PASS — Sends out the 8-byte buffer WRITE_BUFFER to the address in X76_ADDR
- READ_NO_PASS — Starting at X76_ADDR, the routine reads eight bytes and puts them in READ_BUFFER
- MASS_PROGRAM — Programs entire array to 0s
- PROG_CONFIG — Programs the configuration register with five bytes from the WRITE_BUFFER

- READ CONFIG — Reads out the five bytes from the configuration register and stores them in READ_BUFFER
- PROG_READ_PW — Programs the read password with eight bytes from the WRITE_BUFFER
- READ_R_PASS — Starting at X76_ADDR, the routine reads eight bytes and puts them in READ_BUFFER. Requires the read password.

Refer to the X76F041 data sheet to create other command subroutines.

Main Test Routine

The main test routine was written to verify the bus interface between the X76F041 and the C8A. It writes and reads data from the X76F041 with and without passwords. The eight bytes of data that are to be sent out are held in a buffer called WRITE_BUFFER. The eight bytes of data that are read are put into a buffer called READ_BUFFER. When the emulator is stopped, read the contents of the READ_BUFFER to verify the transmission process.

Figure 19 shows the flowchart for the main test routine.

The sequence of tests is:

1. Mass program the device — This command writes 0s throughout the FLASH array. The 8-byte passwords for read and config are also set to $00.
2. Write data to FLASH sector 3 address $100 with no password — Eight bytes of $77 are written to sector 3 starting at location $100.
3. Read data from FLASH sector 3 address $100 with no password — Stop the emulator and verify that the contents of READ_BUFFER are all $77.
4. Mass program the device — This command writes 0s throughout the FLASH array. The 8-byte passwords for read and config are also set to $00.
5. Read data from FLASH sector 3 address $100 with no password — Stop the emulator and verify that the contents of READ_BUFFER are all $00 and the mass program command works.

6. Program the config registers — The config registers are programmed as follows: ACR1 = $00, ACR2 = $04, CR = $00, RR = $00, and RC = $00. This configuration configures sector 3 to require the read password when reading from this section of memory.
7. Read the config registers — Stop the emulator and verify that the first five bytes of READ_BUFFER equal what was written to them.
8. Program the read password with eight bytes of $55.
9. Write data to FLASH sector 3 address $100 with no password — Eight bytes of $AA are written to sector 3 starting at location $100.
10. Read data from FLASH sector 3 address $100 with password — Stop the emulator and verify that the contents of READ_BUFFER are all $AA.

Only some of the X76F041 commands were used in this example. Other commands can be created by using the API building blocks of the serial interface subroutines. Refer to the X76F041 data sheet for a detailed explanation of its command set.

The main test routine demonstrates the interface software needed to communicate with the X76F041. Although the C8A was used, any HC05 device could utilize this interface code. Minor adjustments of port pins and memory maps might be necessary.

The assembly code for the test routine is provided in **Code Listing**.

Development Tools

The interface was created and tested using these development tools:

- M68MMPFB0508 — Freescale MMEVS platform board
- X68EM05C9A — Freescale C Series emulation module
- Win IDE Version 1.02 — Editor, assembler, and debugger by P&E Microcomputer Systems

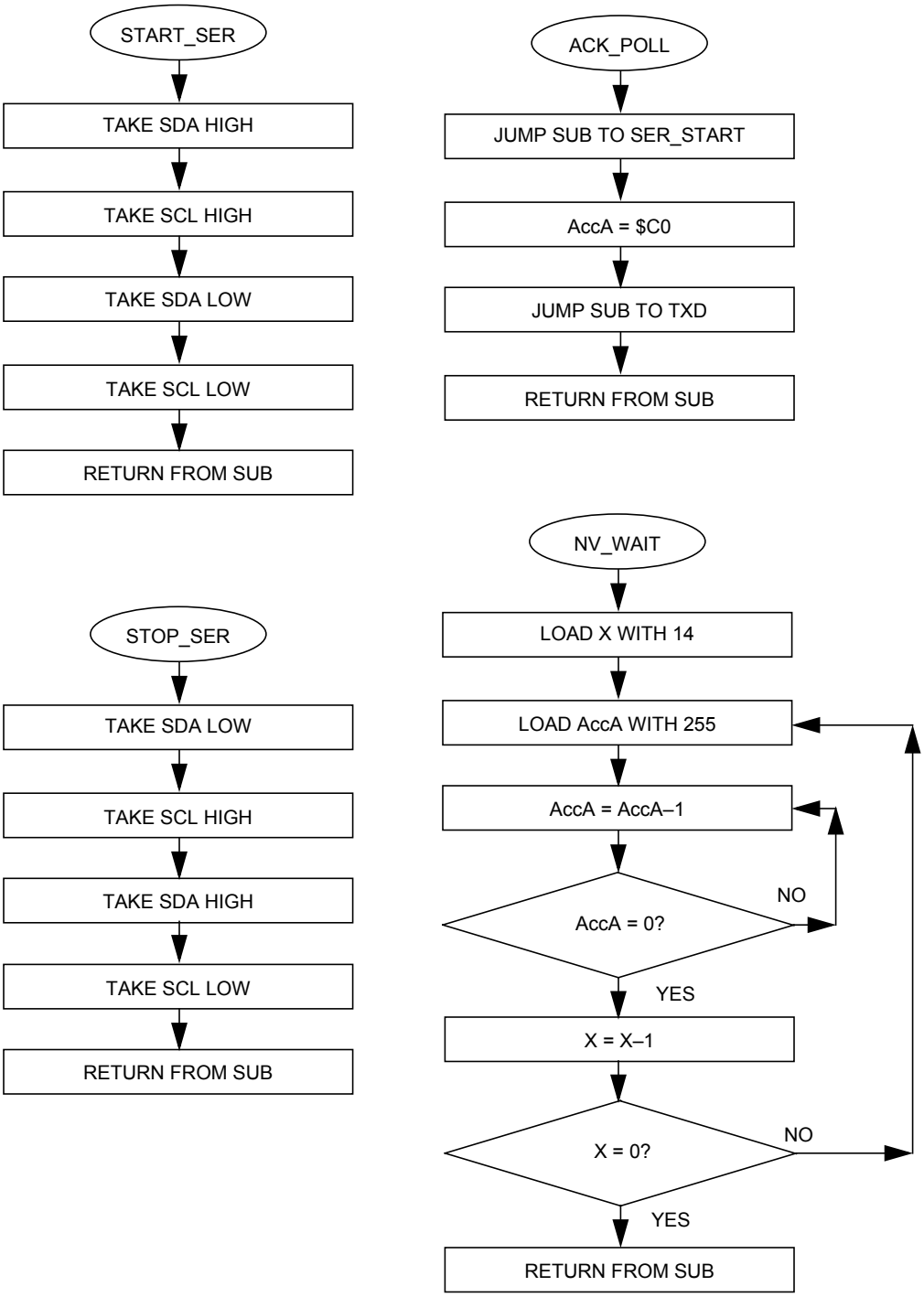


Figure 15. START_SER, ACK_POLL, STOP_SER, and NV_WAIT Subroutine Flowcharts

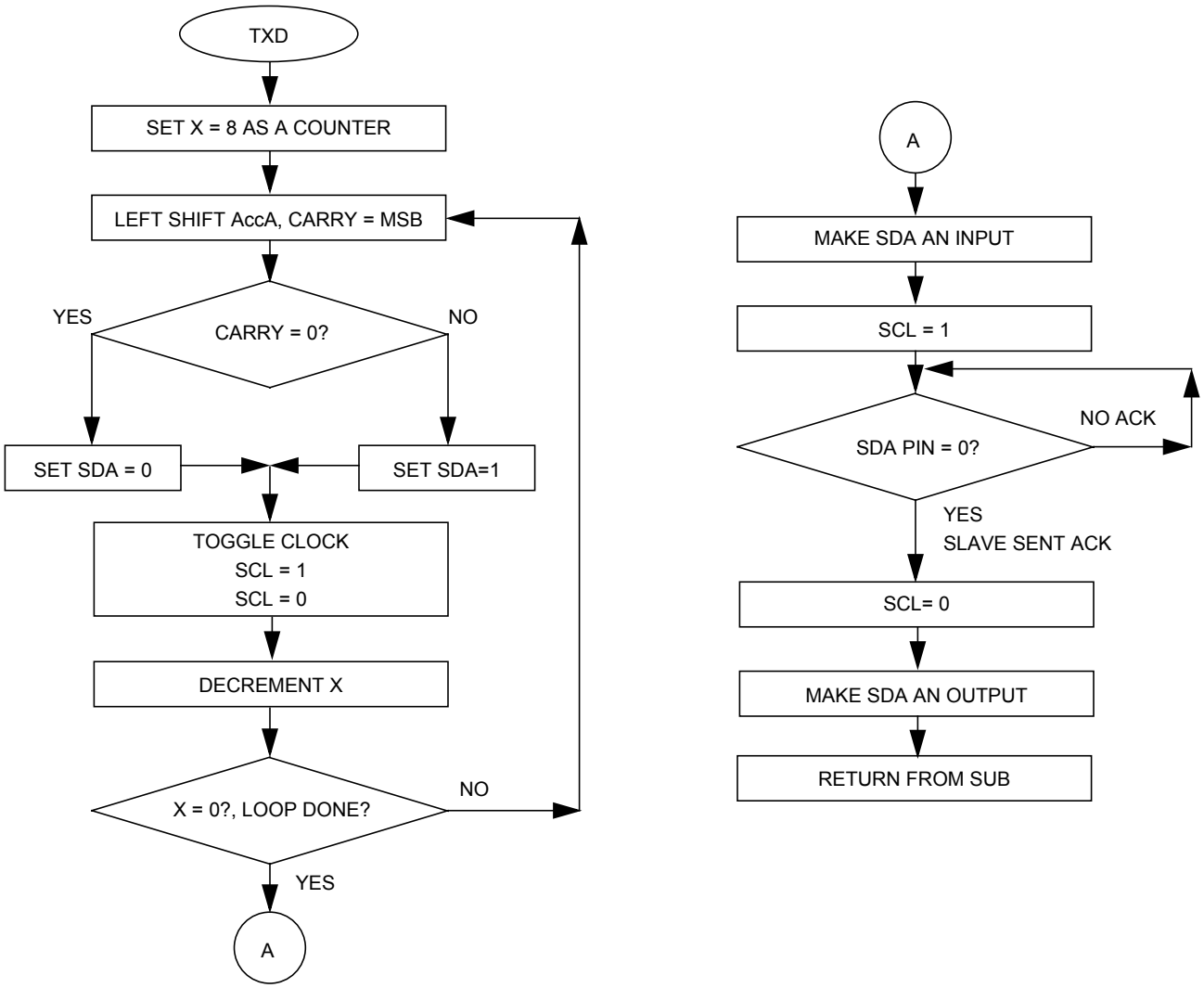


Figure 16. TXD Subroutine Flowchart

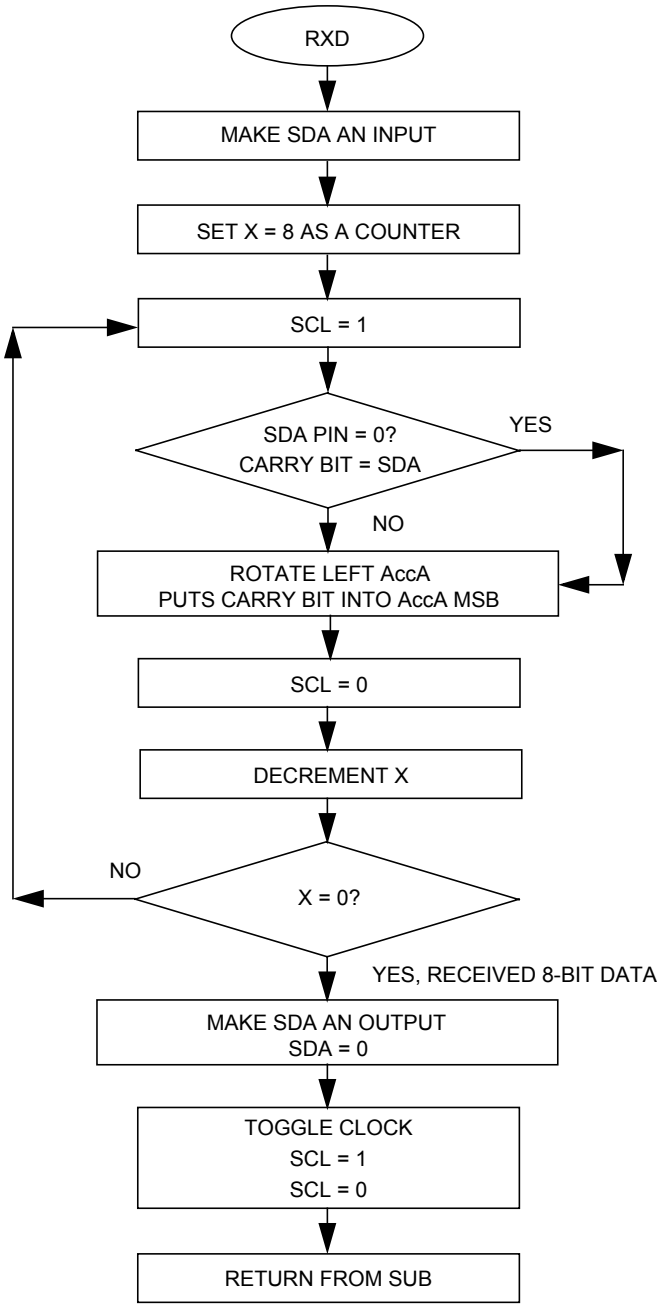


Figure 17. RXD Subroutine Flowchart

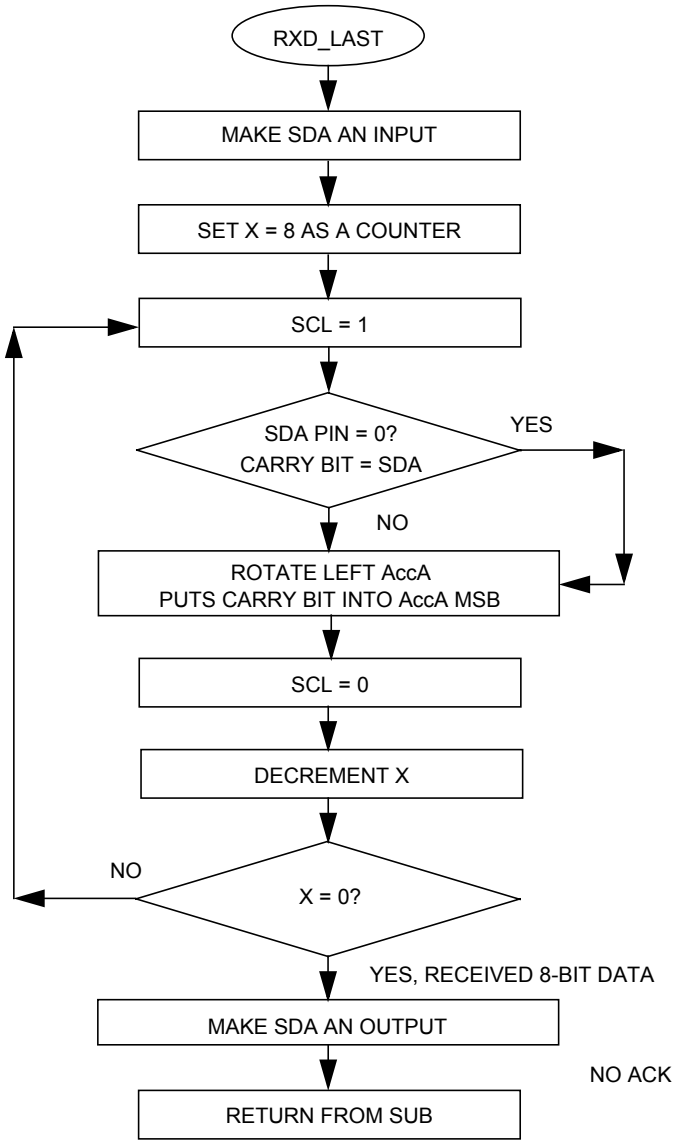


Figure 18. RXD_LAST Subroutine Flowchart

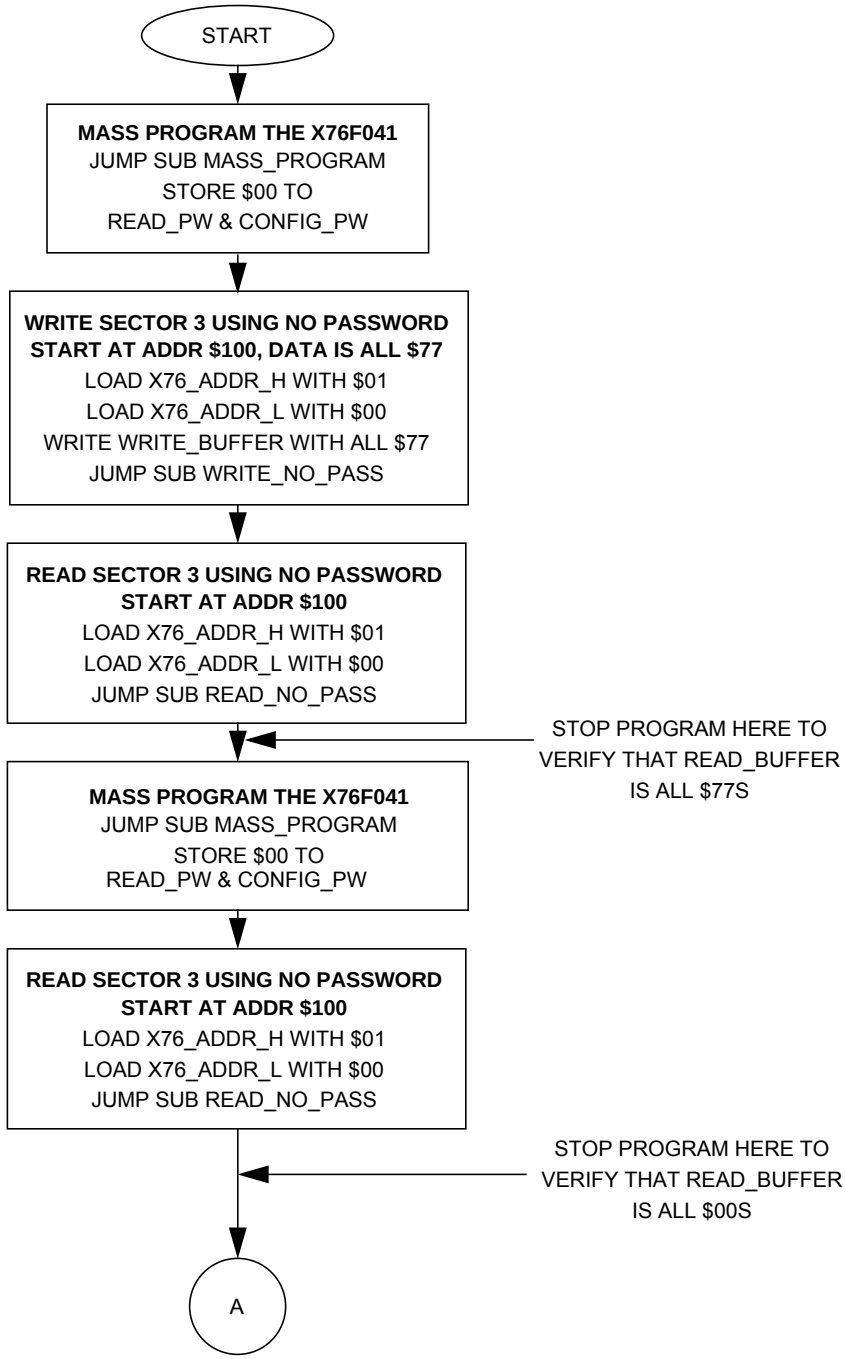


Figure 19. Test Routine Flowchart (Sheet 1 of 2)

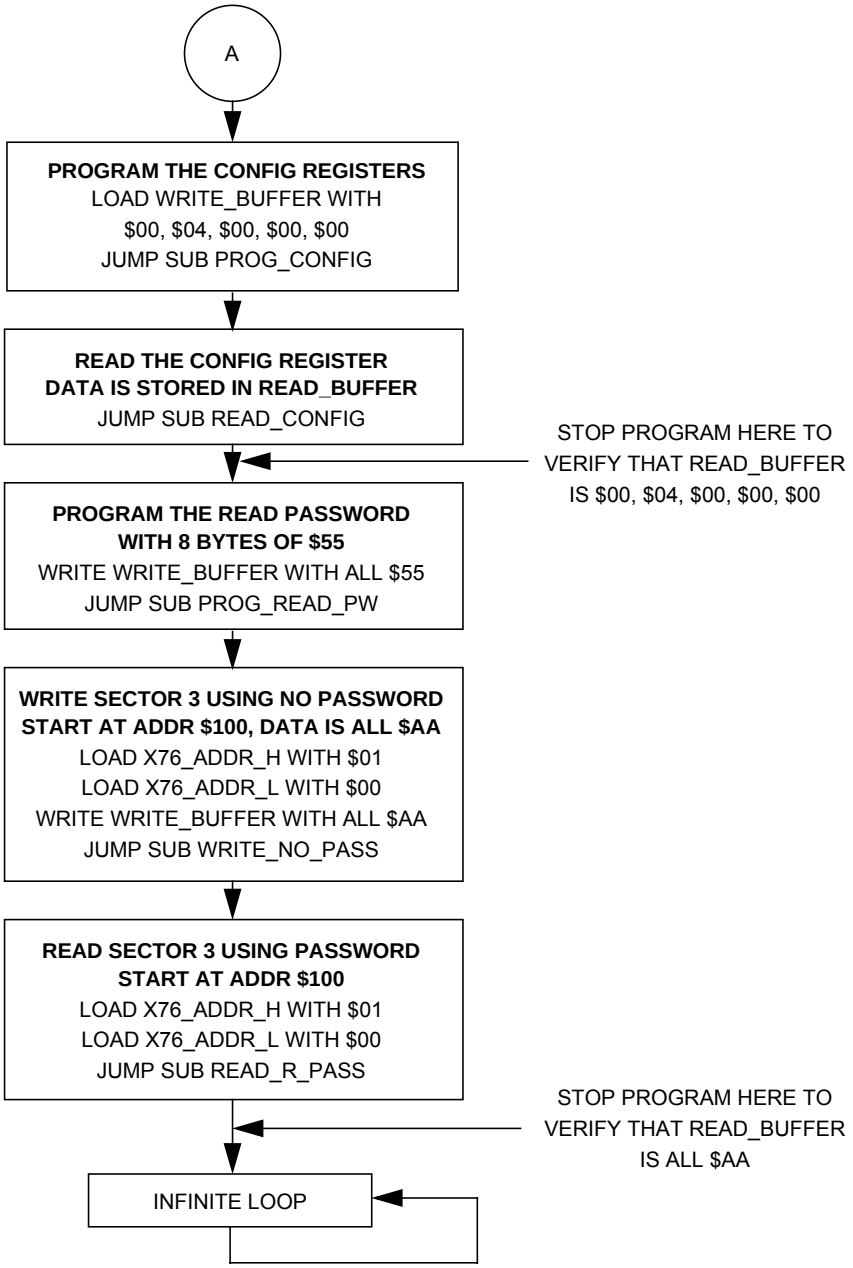


Figure 19. Test Routine Flowchart (Sheet 2 of 2)

Code Listing

```
*****************************************************************************
*
* File name: X76F041.ASM
* Example Code for the MC68HC705C8A Interface to the Xicor X76F041 SecureFlash
* Ver: 1.0
* Date: June 22, 1998
* Author: Mark Glenewinkel
*         Freescale Field Applications
*         Consumer Systems Group
* Assembler: P&E IDE ver 1.02
*
* For code explanation and flow charts,
* please consult Freescale Application Note
*    “Interfacing the MC68HC705C8A to the X76F041 SecureFlash”
*    Literature # AN1761/D
*
* NOTE: All timing routines are based on a
*       2MHz internal bus frequency
*
*****************************************************************************

*** SYSTEM DEFINITIONS AND EQUATES ************************************************
*** Internal Register Definitions
PORTA          EQU      $00                          ;PortA
DDRA           EQU      $04                          ;data direction for PortA

*** Application Specific Definitions
SER_PORT       EQU      $00                          ;PortA is SER_PORT
SCL            EQU      0T                           ;PortA, bit 0, clock signal
SDA            EQU      1T                           ;PortA, bit 1, data signal

*** Memory Definitions
EPROM          EQU      $160                         ;start of EPROM mem
RAM            EQU      $50                          ;start of RAM mem
RESET          EQU      $1FFE                        ;vector for reset

*** RAM VARIABLES *****************************************************************
* Buffers for writing and reading data from the X76F041
               ORG      RAM
READ_PW        DB       $00,$00,$00,$00,$00,$00,$00,$00;read password
CONFIG_PW      DB       $00,$00,$00,$00,$00,$00,$00,$00;config password
WRITE_BUFFER   DS       8                            ;8byte buffer for storing to FLASH
READ_BUFFER    DS       8                            ;storage for read from FLASH
X76_ADDR_H     DB       $00                          ;starting address, high byte
X76_ADDR_L     DB       $00                          ;starting address, low byte
COUNTER        DB       $00                          ;temp counter
```

```
*** MAIN ROUTINE ****************************************************************
              ORG     EPROM                           ;start at beginning of EPROM
*** Initialize Ports
START         lda     #%00000010                      ;init SER_PORT
              sta     SER_PORT
              lda     #$03                            ;make SER_PORT pins outputs
              sta     DDRA

*** Mass Program X76F041 to all zeros
* Make sure the Read and Config registers in RAM are = $00
              jsr     MASS_PROGRAM                    ;array cleared
              clra
              clrx
C1            st      READ_PW,X                       ;store $00 to passwords
              incx
              cpx     #16T                            ;loop for 16 times
              bne     C1

*** Write Sector 3 using no Password, starting address = $100
*** Data is 8 bytes of $77
* load address
              lda     #$01
              sta     X76_ADDR_H                      ;store high address
              lda     #$00
              sta     X76_ADDR_L                      ;store low address
* load 8 byte buffer
              clrx
              lda     #$77
W10           sta     WRITE_BUFFER,X                  ;load buffer with $77
              incx
              cpx     #$08                            ;loop for 8 bytes
              bne     W10

              jsr     WRITE_NO_PASS                   ;write sector, no pass

*** Read Sector 3 using no Password, starting address = $100
*** Store to READ_BUFFER
* load address
              lda     #$01
              st      X76_ADDR_H                      ;store high address
              lda     #$00
              sta     X76_ADDR_L                      ;store low address

              jsr     READ_NO_PASS                    ;read sector, no pass

*** Mass Program X76F041 to all zeros
* Make sure the Read and Config registers in RAM are = $00
              jsr     MASS_PROGRAM                    ;array cleared
              clra
              clrx
C2            sta     READ_PW,X                       ;store $00 to passwords
```

```
                incx
                cpx     #16T                    ;loop for 16 times
                bne     C2


*** Read Sector 3 using no Password, starting address = $100
*** Store to READ_BUFFER, should be all zeros after mass program
* load address
                lda     #$01
                sta     X76_ADDR_H              ;store high address
                lda     #$00
                sta     X76_ADDR_L              ;store low address

                jsr     READ_NO_PASS            ;read sector, no pass


*** Program the Config Registers
*** Make Third Sector need Read Password
                lda     #$00
                sta     WRITE_BUFFER            ;ACR1=$00
                lda     #$04
                sta     WRITE_BUFFER+1          ;ACR2=$04
                lda     #$00
                sta     WRITE_BUFFER+2          ;CR=$00
                lda     #$00
                sta     WRITE_BUFFER+3          ;RR=$00
                lda     #$00
                sta     WRITE_BUFFER+4          ;RC=$00

                jsr     PROG_CONFIG             ;program the config reg


*** Read Config registers of X76F041
*** Store to READ_BUFFER
                jsr     READ_CONFIG             ;read config


*** Program the Read Password to all $55
                crux
                lda     #$55
R_PW1           sta     WRITE_BUFFER,X          ;store $55 to buffer
                incx
                cpx     #$08                    ;loop for 8 bytes
                bne     R_PW1

                jsr     PROG_READ_PW            ;program read pass
```

```
*** Write Sector 3 using no Password, starting address = $100
*** Data is 8 bytes of $AA
* load address
              lda     #$01
              sta     X76_ADDR_H                ;store high address
              lda     #$00
              sta     X76_ADDR_L                ;store low address
* load 8 byte buffer
              clrx
              lda     #$AA
W20           sta     WRITE_BUFFER,X            ;store $AA to buffer
              incx
              cpx     #$08                      ;loop for 8 bytes
              bne     W20

              jsr     WRITE_NO_PASS             ;write sector, no pass


*** Read Sector 3 using Read Password, starting address = $100
*** Store to READ_BUFFER, verify for $AA
              lda     #$01
              sta     X76_ADDR_H                ;store high address
              lda     #$00
              sta     X76_ADDR_L                ;store low address

              jsr     READ_R_PASS               ;read sector w/ pass


DUMMY         bra     DUMMY                     ;test sequence is over


*** SUBROUTINES - SERIAL BUS ******************************************************
*** Sends out Start command on bus
START_SER     bset    SDA,SER_PORT              ;SDA=1
              bset    SCL,SER_PORT              ;SCL=1
              bclr    SDA,SER_PORT              ;SDA=0
              bclr    SCL,SER_PORT              ;SCL=0
              rts

*** Sends out Stop command on bus
STOP_SER      bclr    SDA,SER_PORT              ;SDA=0
              bset    SCL,SER_PORT              ;SCL=1
              bset    SDA,SER_PORT              ;SDA=1
              bclr    SCL,SER_PORT              ;SCL=0
              rts

*** Sends out ACK polling
ACK_POLL      jsr     START_SER
              lda     #$C0
              jsr     TXD                       ;send out $C0, ACK?
              rts
```

```
*** Routine takes contents of AccA and transmits it serially to
*** the X76F041, MSB first
*** Looks for ACK, infinite loop if no ACK
TXD             ldx     #8T                         ;set counter

WRITE           asla                                ;Carry bit = MSB
                bcc     J1
                bset    SDA,SER_PORT                ;SDA=1
                bra     CLOCK_IT                    ;branch to clock_it
J1              bclr    SDA,SER_PORT                ;SDA=0
                brn     J1                          ;evens it out

CLOCK_IT        bset    SCL,SER_PORT                ;SCL=1
                bclr    SCL,SER_PORT                ;SCL=0
                decx                                ;decrement counter
                bne     WRITE
* Check for ACK
                bclr    SDA,DDRA                    ;SDA is input
                bset    SCL,SER_PORT                ;SCL=1
J2              brset   SDA,SER_PORT,J2             ;if SDA=0, slave ACK
                                                    ;no slave ack, stay in loop

                bclr    SCL,SER_PORT                ;SCL=0
                bset    SDA,DDRA                    ;SDA is output
                rts                                 ;return from sub


*** Routine clocks the X76F041 to read data from SDA, MSB first
*** 8 bit contents are put in AccA
*** Generates ACK back to slave except for last read
RXD             bclr    SDA,DDRA                    ;make the SDA pin on J1A input
                ldx     #8T                         ;set counter

READ            bset    SCL,SER_PORT                ;SCL=1
                brclr   SDA,SER_PORT,J3             ;carry bit = SDA
J3              rola                                ;put carry bit into AccA MSB
                bclr    SCL,SER_PORT                ;SCL=0

                decx                                ;decrement counter
                bne     READ
* ACK back to slave
                bset    SDA,DDRA                    ;make the SDA pin on J1A output
                bclr    SDA,SER_PORT                ;SDA=0
                bset    SCL,SER_PORT                ;SCL=1
                bclr   SCL,SER_PORT                 ;SCL=0
READ_DONE       rts                                 ;return from sub
```

```
*** Routine clocks the X76F041 to read data from SDA, MSB first
*** 8 bit contents are put in AccA
*** Generates NO ACK back to slave, signals last read to DS1307
RXD_LAST      bclr    SDA,DDRA                    ;make the SDA pin on J1A input
              ldx     #8T                         ;set counter

READ_LAST     bset    SCL,SER_PORT                ;SCL=1
              brclr   SDA,SER_PORT,J4             ;carry bit = SDA
J4            rola                                ;put carry bit into AccA MSB
              bclr    SCL,SER_PORT                ;SCL=0

              decx                                ;decrement counter
              bne     READ_LAST
* NO ACK back to slave
              bset    SDA,DDRA                    ;make the SDA pin on J1A output
              rts                                 ;return from sub

*** Routine creates a ~10ms routine with a 2MHz MCU internal bus for
*** NV memory to be set correctly
NV_WAIT       ldx     #14T
J12           lda     #255T
J13           deca                                ;3
              bne     J13                         ;3
              decx
              bne     J12
              rts


*** SUBROUTINES - COMMANDS ***********************************************************
*** Routine writes to X76F041 with address and data buffer
WRITE_NO_PASS jsr     START_SER                   ;start serial transmission

              lda     X76_ADDR_H                  ;send out command + high addr
              ora     #%00000000
              jsr     TXD
              lda     X76_ADDR_L                  ;send out low addr
              jsr     TXD

* transmit WRITE_BUFFER to X76F041
              clrx                                ;X=0
W1            lda     WRITE_BUFFER,X              ;AccA=buffer
              stx     COUNTER                     ;COUNTER=X
              jsr     TXD                         ;transmit byte
              ldx     COUNTER                     ;X=COUNTER
              incx                                ;X=X+1
              cpx     #$08                        ;loop 8 times
              bne     W1                          ;done?

              jsr     STOP_SER                    ;stop serial transmission
              jsr     NV_WAIT                     ;wait for 10ms
              rts                                 ;return from sub
```

```
*** Routine reads X76F041 with address and stores to READ_BUFFER
READ_NO_PASS   jsr     START_SER                  ;start serial transmission

               lda     X76_ADDR_H                 ;send out command + high addr
               ora     #%00100000
               jsr     TXD
               lda     X76_ADDR_L                 ;send out low addr
               jsr     TXD

* Read from X76F041, store to READ_BUFFER
               clrx                               ;X=0
R1             stx     COUNTER                    ;COUNTER=X
               jsr     RXD                        ;read a byte
               ldx     COUNTER                    ;X=COUNTER
               sta     READ_BUFFER,X              ;store byte to read buffer
               incx                               ;X=X+1
               cpx     #$07
               bne     R1                         ;7 bytes done?
               jsr     RXD_LAST                   ;send out with no ack
               ldx     #$07                       ;X=$07
               sta     READ_BUFFER,X              ;store last byte

               jsr     STOP_SER                   ;stop serial transmission
               rts

*** Routine mass programs X76F041 to all zeros
MASS_PROGRAM   jsr     START_SER                  ;start serial transmission

               lda     #$80                       ;send out command + high addr
               jsr     TXD
               lda     #$70                       ;send out low addr
               jsr     TXD
* send out config password
               clrx                               ;X=0
M1             lda     CONFIG_PW,X                ;AccA=Config[x]
               stx     COUNTER                    ;COUNTER=X
               jsr     TXD                        ;transmit AccA
               ldx     COUNTER                    ;X=COUNTER
               incx                               ;X=X+1
               cpx     #$08                       ;COUNTER=8?
               bne     M1                         ;if not, loop again

               jsr     NV_WAIT                    ;wait for 10ms
               jsr     ACK_POLL                   ;ack polling
               jsr     STOP_SER                   ;stop serial transmission
               jsr     NV_WAIT                    ;wait for 10ms
               rts
```

```
*** Routine programs the config registers with data from WRITE_BUFFER
PROG_CONFIG    jsr     START_SER                   ;start serial transmission

               lda     #$80                        ;send out command + high addr
               jsr     TXD
               lda     #$50                        ;send out low addr
               jsr     TXD
* Send out config password
               clrx                                ;X=0
PC1            lda     CONFIG_PW,X                 ;AccA=CONFIG_PW[x]
               stx     COUNTER                     ;COUNTER=X
               jsr     TXD                         ;transmit AccA
               ldx     COUNTER                     ;X=COUNTER
               incx                                ;X=X+1
               cpx     #$08                        ;COUNTER=8?
               bne     PC1                         ;if not, loop again

               jsr     NV_WAIT                     ;wait for 10ms
               jsr     ACK_POLL                    ;ack polling

               clrx                                ;X=0
PC2            lda     WRITE_BUFFER,X              ;AccA=buffer byte
               stx     COUNTER                     ;COUNTER=X
               jsr     TXD                         ;transmit byte
               ldx     COUNTER                     ;X=COUNTER
               incx                                ;X=X+1
               cpx     #$05
               bne     PC2                         ;5 bytes done?

               jsr     STOP_SER                    ;stop serial transmission
               jsr     NV_WAIT                     ;wait for 10ms
               rts

*** Routine programs reads the config registers, stores to READ_BUFFER
READ_CONFIG    jsr     START_SER                   ;start serial transmission

               lda     #$80                        ;send out command + high addr
               jsr     TXD
               lda     #$60                        ;send out low addr
               js      TXD
* Send out Config Password
               clrx                                ;X=0
R3             lda     CONFIG_PW,X                 ;AccA=Config[x]
               stx     COUNTER                     ;COUNTER=X
               jsr     TXD                         ;transmit AccA
               ldx     COUNTER                     ;X=COUNTER
               incx                                ;X=X+1
               cpx     #$08                        ;COUNTER=8?
               bne     R3                          ;if not, loop again

               jsr     NV_WAIT                     ;wait for 10ms
```

```
                jsr     ACK_POLL                    ;poll ack

* Read and store to READ_BUFFER
                clrx                                ;X=0
R4              stx     COUNTER                     ;COUNTER=X
                jsr     RXD                         ;receive byte
                ldx     COUNTER                     ;COUNTER=X
                sta     READ_BUFFER,X               ;store away byte
                incx                                ;X=X+1
                cpx     #$04
                bne     R4                          ;4 bytes received

                jsr     RXD_LAST                    ;read last byte
                ldx     #$04
                sta     READ_BUFFER,X               ;store lst byte
                jsr     STOP_SER                    ;stop serial transmission
                jsr     NV_WAIT                     ;wait for 10ms
                rts

*** Routine programs the Read Password with data from WRITE_BUFFER
PROG_READ_PW    jsr     START_SER                   ;start serial transmission

                lda     #$80                        ;send out command + high addr
                jsr     TXD
                lda     #$10                        ;send out low addr
                jsr     TXD
* send out old read password to change
                clrx                                ;X=0
R_PW0           lda     READ_PW,X                   ;AccA=0
                stx     COUNTER                     ;COUNTER=X
                jsr     TXD                         ;transmit AccA
                ldx     COUNTER                     ;X=COUNTER
                incx                                ;X=X+1
                cpx     #$08                        ;COUNTER=8?
                bne     R_PW0                       ;if not, loop again

                jsr     NV_WAIT                     ;wait for 10ms
                jsr     ACK_POLL                    ;ack polling

                clrx                                ;X=0
R_PW2           lda     WRITE_BUFFER,X              ;load byte from buffer
                stx     COUNTER                     ;COUNTER=X
                jsr     TXD                         ;transmit byte
                ldx     COUNTER                     ;X=COUNTER
                incx                                ;X=X+1
                cpx     #$08
                bne     R_PW2                       ;8 bytes sent?

                clrx                                ;X=0
R_PW3           lda     WRITE_BUFFER,X              ;load byte from buffer
                stx     COUNTER                     ;COUNTER=X
```

```
                jsr     TXD                     ;transmit
                ldx     COUNTER                 ;X=COUNTER
                incx                            ;X=X+1
                cpx     #$08
                bne     R_PW3                   ;8 bytes sent?

                jsr     STOP_SER                ;stop serial transmission
                jsr     NV_WAIT                 ;wait for 10ms

* Store new password
                clrx                            ;X=0
R_PW4           lda     WRITE_BUFFER,X          ;load byte from buffer
                sta     READ_PW,X               ;store away
                incx                            ;X=X+1
                cpx     #$08
                bne     R_PW4                   ;8 bytes written?
                rts

*** Routine reads X76F041 with address and stores to READ_BUFFER
*** Needs 8 byte Read Password
READ_R_PASS     jsr     START_SER               ;start serial transmission

                lda     X76_ADDR_H              ;send out command + high addr
                ora     #%00100000
                jsr     TXD
                lda     X76_ADDR_L              ;send out low addr
                jsr     TXD

                clrx                            ;X=0
RF1             lda     READ_PW,X               ;AccA=READ_PW[x]
                stx     COUNTER                 ;COUNTER=X
                jsr     TXD                     ;transmit AccA
                ldx     COUNTER                 ;X=COUNTER
                incx                            ;X=X+1
                cpx     #$08                    ;COUNTER=8?
                bne     RF1                     ;if not, loop again

                jsr     NV_WAIT                 ;wait for 10ms
                jsr     ACK_POLL                ;ack polling
                jsr     RXD_LAST                ;dummy read for X76F041
                jsr     START_SER               ;start condition
                lda     X76_ADDR_L              ;send out low addr
                jsr     TXD

                clrx                            ;X=0
RF2             stx     COUNTER                 ;COUNTER=X
                jsr     RXD                     ;read byte
                ldx     COUNTER                 ;X=COUNTER
                sta     READ_BUFFER,X           ;store away
                incx                            ;X=X+1
                cpx     #$07                    ;7 bytes read?
```

```
                bne     RF2
                jsr     RXD_LAST                ;read last byte
                ldx     #$07
                sta     READ_BUFFER,X           ;store last byte

                jsr     STOP_SER                ;stop serial transmission
                rts


*** VECTOR TABLE ************************************************************
                ORG     RESET
                DW      START
```

References

MC68HC705C8A Technical Data, Freescale document order number MC68HC705C8A/D, 1996.

M68HC05 Applications Guide, Freescale document order number M68HC05AG/AD, 1996.

X76F041 Data Sheet, Xicor, Inc., 1997.

How to Reach Us:

Home Page:
www.freescale.com

E-mail:
support@freescale.com

USA/Europe or Locations Not Listed:
Freescale Semiconductor
Technical Information Center, CH370
1300 N. Alma School Road
Chandler, Arizona 85224
+1-800-521-6274 or +1-480-768-2130
support@freescale.com

Europe, Middle East, and Africa:
Freescale Halbleiter Deutschland GmbH
Technical Information Center
Schatzbogen 7
81829 Muenchen, Germany
+44 1296 380 456 (English)
+46 8 52200080 (English)
+49 89 92103 559 (German)
+33 1 69 35 48 48 (French)
support@freescale.com

Japan:
Freescale Semiconductor Japan Ltd.
Headquarters
ARCO Tower 15F
1-8-1, Shimo-Meguro, Meguro-ku,
Tokyo 153-0064
Japan
0120 191014 or +81 3 5437 9125
support.japan@freescale.com

Asia/Pacific:
Freescale Semiconductor Hong Kong Ltd.
Technical Information Center
2 Dai King Street
Tai Po Industrial Estate
Tai Po, N.T., Hong Kong
+800 2666 8080
support.asia@freescale.com

For Literature Requests Only:
Freescale Semiconductor Literature Distribution Center
P.O. Box 5405
Denver, Colorado 80217
1-800-441-2447 or 303-675-2140
Fax: 303-675-2150
LDCForFreescaleSemiconductor@hibbertgroup.com

freescale™
semiconductor