

Interfacing the MC68HC705J1A to 9356/9366 EEPROMs

By Mark Glenewinkel
CSIC Applications

INTRODUCTION

This application note describes the hardware and software interface used to communicate between the Freescale MC68HC705J1A MCU and 9356/9366 EEPROM chips. The 93XX series of EEPROMs are an industry standard used widely to store nonvolatile bits of information. The software listing in this application note will work with 9356 and 9366 EEPROMs. The EEPROM bits are arranged in 128 or 256 16-bit registers, respectively. With some modification, the software will work with other 93XX series EEPROMs.

Some of the applications in which EEPROMs can be utilized are listed below.

- ID number for remote addressing or security
- Storage of telecommunication information like phone number recall and speed dialing
- Power down information storage for consumer electronics like TVs and VCRs
- Reprogrammable calibration data for test/measurement equipment

The 93XX EEPROMs communicate with the outside world using a serial link. Since the MC68HC705J1A does not have the hardware on chip to communicate to the EEPROM, a software driver is used. This method bit programs an I/O port to properly transfer data to and from the EEPROM. A National NM93C56N was used for testing the software routines in this application note.

HARDWARE INTERFACE

The 9356 is a very simple 8-pin device. Appendix A shows a typical connection between the MC68HC705J1A and the 9356. The serial interface connection uses only four pins of the 9356. They are as follows:

- CS — Chip Select
- SK — Serial Clock
- DO — Serial Data Output
- DI — Serial Data Input

These signals must be clocked in a certain way in order to transfer the correct serial data to and from the MC68HC705J1A.



SOFTWARE INTERFACE

Communication between the MC68HC705J1A and the 9356 is done with a synchronous serial protocol. As mentioned earlier, the MC68HC705J1A bit programs its I/O pins to communicate with the 9356. A timing diagram of the serial link can be found in the 9356 data sheet if needed.

The 9356 will accept seven different commands. They are as follows:

- 1) READ — Read a 16-bit data word from an address in memory
- 2) WRITE — Write a 16-bit data word from an address in memory
- 3) WRALL — Write all addresses with the same 16-bit data word
- 4) ERASE — Erase a 16-bit data word from an address in memory
- 5) ERAL — Erase all addresses within the memory map
- 6) WEN — Erase/write enable the EEPROM memory
- 7) WDS — Erase/write disable the EEPROM memory

The 9356 transmission format is a frame of data bits containing an opcode, an address, and if needed, a word of data. The opcode is three bits long, the address is eight bits long, and the data word is 16 bits long. Table 1 illustrates the bit information each instruction needs.

Table 1.

Instruction	Opcode	Address	Data
READ	110	A7–A0	
WEN	100	11XXXXXX	
ERASE	111	A7–A0	
ERAL	100	10XXXXXX	
WRITE	101	A7–A0	D15–D0
WRALL	100	01XXXXXX	D15–D0
WDS	100	00XXXXXX	

IMPLEMENTATION AND TEST

Software was written to provide subroutines to perform each of the seven commands. A total of four bytes of RAM are needed to support the subroutines. These bytes are described below.

- 1) OP_CODE — Contains the opcode needed for the command
- 2) ADDR — Contains the address for the command
- 3) DATA_H — The high byte for the data word
- 4) DATA_L — The low byte for the data word

As needed, each EEPROM command subroutine will call other supporting subroutines to execute the transmission of data between the MC68HC705J1A and the 9356. Appendix B contains flowcharts for all of these subroutines.

Freescale Semiconductor, Inc.

Each EEPROM command subroutine has input data and output data. This data is inherent with some commands while others need the information passed to them before the subroutine is called. Table 2 lists the input data needed and output data generated for each of the seven commands.

Table 2:

Command	Subroutine Input	Subroutine Output
READ	ADDR	DATA_H/L
EWEN	—	—
ERASE	ADDR	—
ERAL	—	—
WRITE	ADDR & DATA_H/L	—
WRALL	DATA_H/L	—
EWDS	—	—

Code was written and tested with a level of quality equal to the Carnegie-Mellon Software Engineering Institute (SEI) Level 2. A test routine consisting of writing and reading the EEPROM is listed in Appendix C. Refer to Appendix A for the schematic used in the design and test of the software. An LED is used to verify that the test code works properly. The test routine executes the following:

- 1) Initializes the port on the HC705J1A for serial transmission. LED is turned off.
- 2) Writes EEPROM address \$00 with \$AA55.
- 3) Writes EEPROM address \$20 with \$1234.
- 4) Reads EEPROM address \$00 and stores it to RAM location TEST1 and TEST2.
- 5) Reads EEPROM address \$20 and stores it to RAM location TEST3 and TEST4.
- 6) Checks if TEST1 = \$AA, TEST2 = \$55, TEST3 = \$12, and TEST4 = \$34.
- 7) If check is good, then light the LED. If check is bad, do not light the LED.
- 8) Continue to run in an infinite loop until reset.

For increased reliability, the software watchdog on the MC68HC705J1A is used. Also, a low voltage inhibit circuit, the MC34064, is used to decrease susceptibility to brown out or short power failure conditions.

SUMMARY

This application note has described the interface needed to successfully communicate between the MC68HC705J1A and the 9356. For more information on the MC68HC705J1A, please consult the Technical Data Manual, MC68HC705J1A/D. Contact National Semiconductor or SGS Thompson for technical data on the 93XX series of EEPROM memories.

An electronic copy of the code listing in Appendix C and a listing of the test program to fully test all the EEPROM commands can be found on the Freescale MCU BBS. The BBS number is (512) 891-3733. The filename is j1a_9356.arc and is on the CSIC BBS under the APPNOTES directory.

Also, Freescale Application Note AN1221/D further details the software and hardware interfaces needed between the 93XX series and other HC05 MCUs.

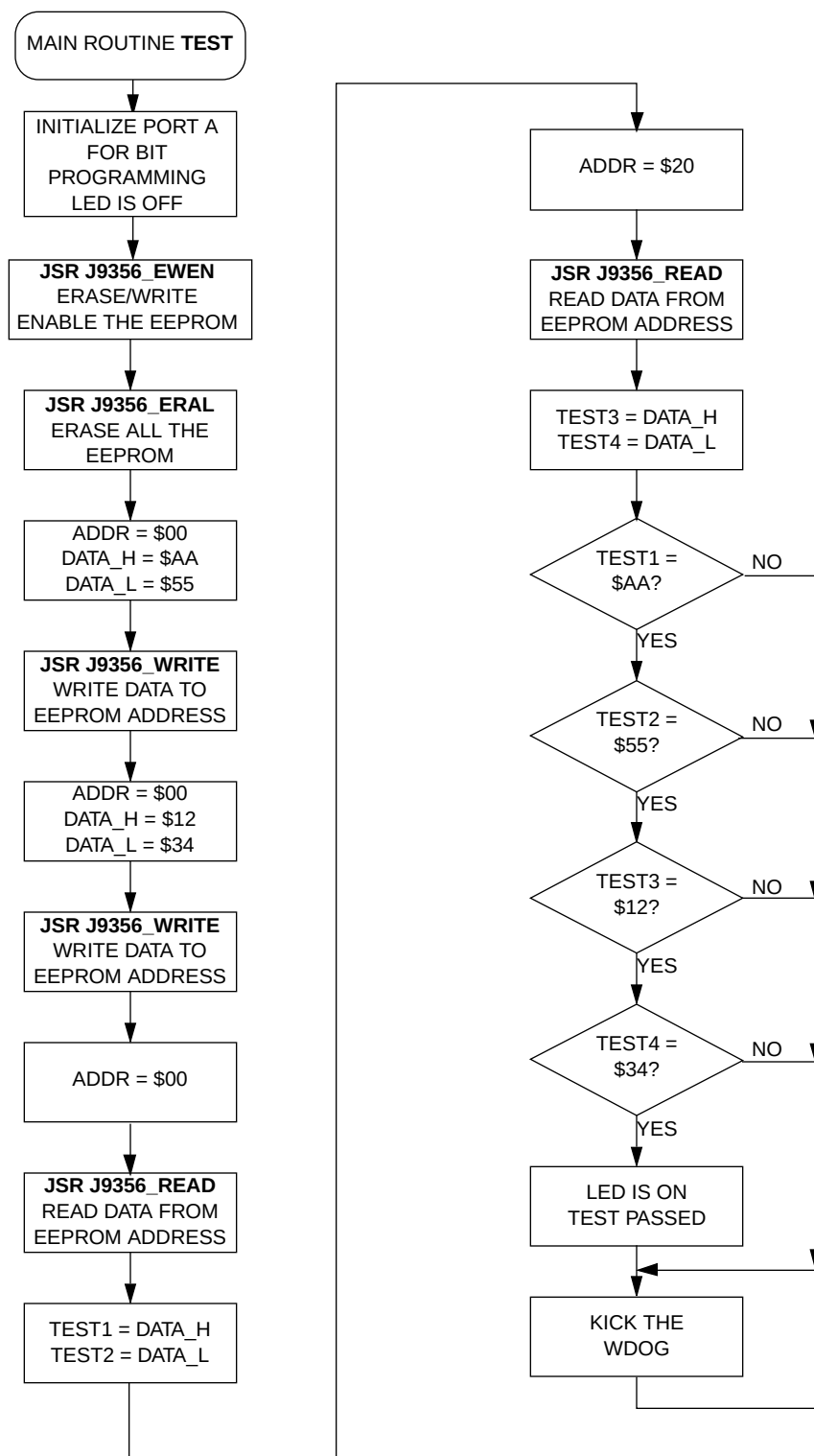
APPENDIX A

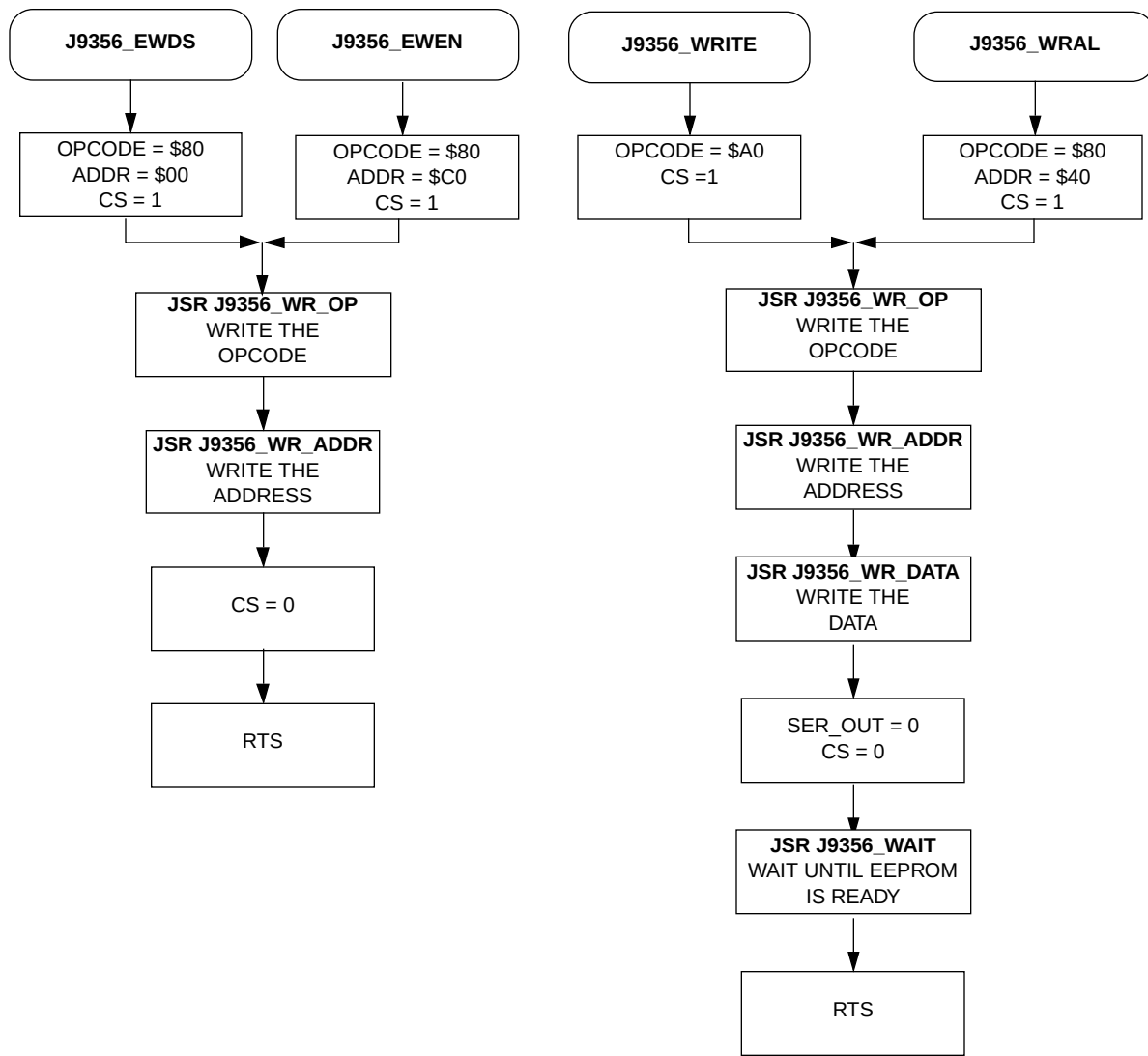
The circuit diagram shows the following components and connections:

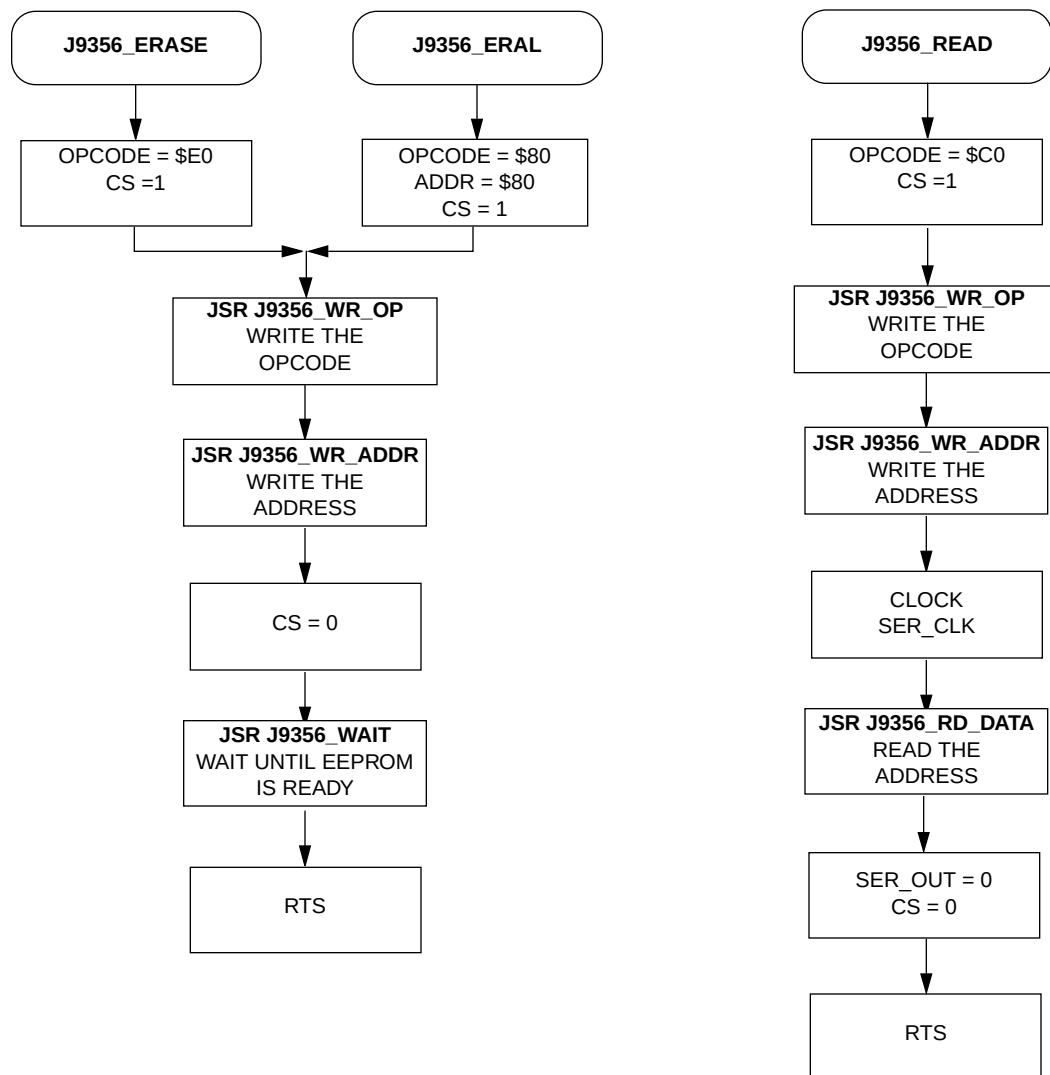
- MC68HC705J1AP (U1):**
 - Pin 1 (VDD):** Connected to VDD.
 - Pin 2 (VSS):** Connected to GND.
 - Pin 3 (RESET):** Connected to the output of U3.
 - Pin 4 (IRQ/VPP):** Connected to VDD.
 - Pin 5 (PB0):** Connected to VDD.
 - Pin 6 (PB1):** Connected to VDD.
 - Pin 7 (PB2):** Connected to VDD.
 - Pin 8 (PB3):** Connected to VDD.
 - Pin 9 (PB4):** Connected to VDD.
 - Pin 10 (PB5):** Connected to VDD.
 - Pin 11 (PA0):** Connected to U2 Pin 18.
 - Pin 12 (PA1):** Connected to U2 Pin 17.
 - Pin 13 (PA2):** Connected to U2 Pin 16.
 - Pin 14 (PA3):** Connected to U2 Pin 15.
 - Pin 15 (PA4):** Connected to U2 Pin 14.
 - Pin 16 (PA5):** Connected to U2 Pin 13.
 - Pin 17 (PA6):** Connected to U2 Pin 12.
 - Pin 18 (PA7):** Connected to U2 Pin 11.
 - Pin 19 (OSC1):** Connected to U2 Pin 2.
 - Pin 20 (OSC2):** Connected to U2 Pin 1.
- NM93C56N (U2):**
 - Pin 1 (VCC):** Connected to VDD.
 - Pin 2 (SK):** Connected to U1 Pin 19.
 - Pin 3 (DI):** Connected to U1 Pin 20.
 - Pin 4 (DO):** Connected to U1 Pin 3.
 - Pin 5 (NC):** Connected to GND.
 - Pin 6 (NC):** Connected to GND.
 - Pin 7 (NC):** Connected to GND.
 - Pin 8 (GND):** Connected to GND.
- MC34064 (U3):**
 - Pin 1 (INPUT):** Connected to VDD.
 - Pin 2 (RESET):** Connected to U1 Pin 3.
 - Pin 3 (GND):** Connected to GND.
- Capacitor C1 (0.1uF):** Connected between VDD and GND.
- LED D1:** Connected to the output of U2 (Pin 4, DO) through a resistor R1 (330Ω).

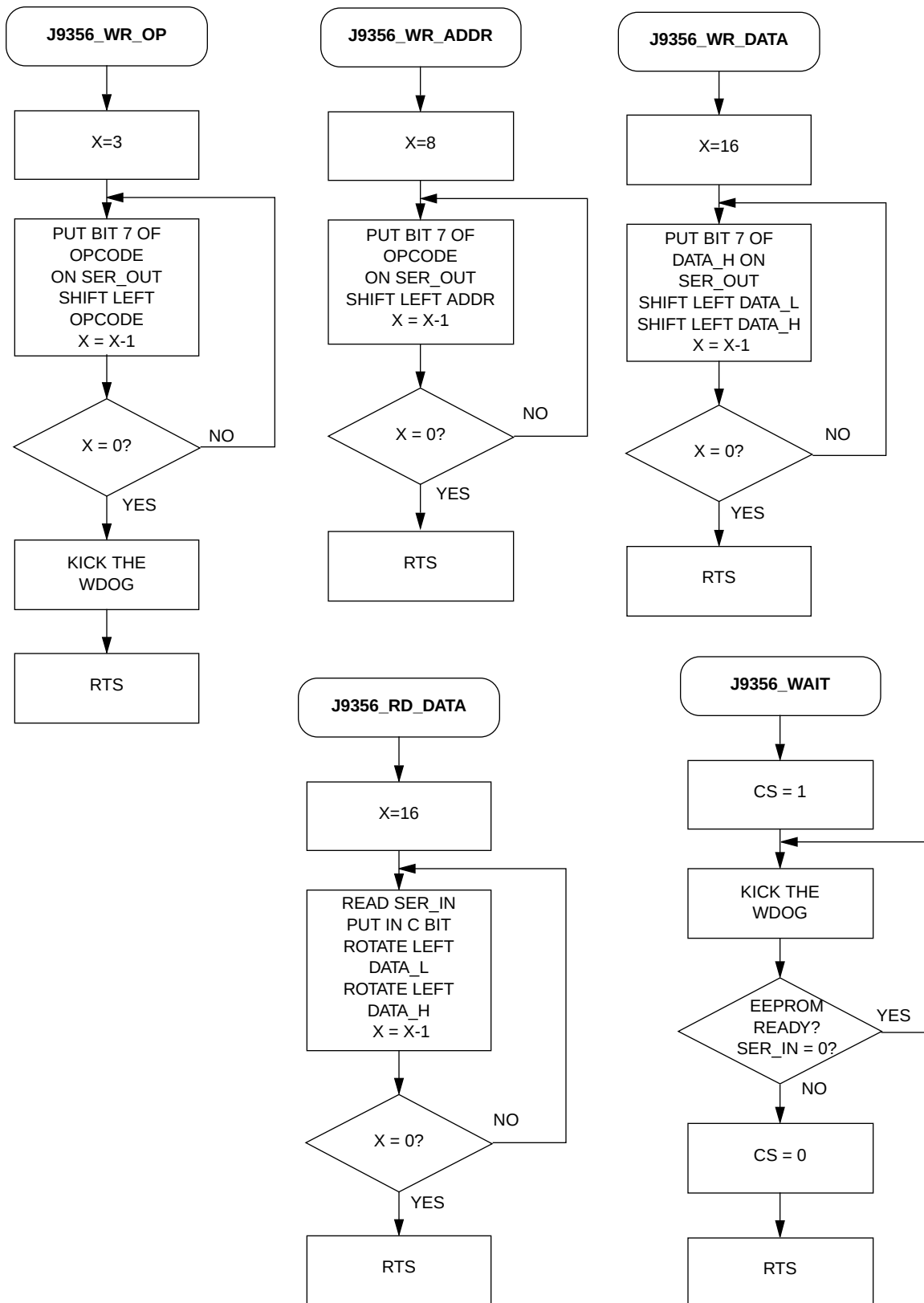
AN1241/D

APPENDIX B









APPENDIX C

```

*****
*****
*
*           Main Routine J1A_9356 - 705J1A to 9356 EEPROM
*
*****
*
* File Name: J1A_9356.RTN                      Copyright (c) Motorola 1995
*
* Full Functional Description Of Routine Design:
*   Program flow:
*     Reset:  Initializes ports for bit banging.
*             Calls EWEN sub to enable write to EEPROM.
*             Calls ERAL to erase all EEPROM
*             Writes $AA55 to EEPROM $00
*             Writes $1234 to EEPROM $20
*             Reads EEPROM $00 and $20
*             Check for correct data, light LED if correct
*             Execute endless loop
*
*****
*
*           Part Specific Framework Includes Section
*
*****

#nolist
#include 'H705J1A.FRK'                      ;Include the equates for the HC705J1A
                                           ;so that all labels can be used.
#list

*****
*
*           MOR Bytes Definitions for Main Routine
*
*****

                org      MOR
                db        $21                ;COP enabled, osc resistor enabled
                                           ;If used on a mask rom part,
                                           ; be sure to specify this option.

```

```
*****
*
*
* Equates and RAM Storage
*
*****
*
```

```
CS          equ    0          ;bit # for chip select
SER_CLK     equ    1          ;bit # for serial clock
SER_OUT     equ    2          ;bit # for serial data out
SER_IN      equ    3          ;bit # for serial data in
```

```
***      RAM storage variables      ***
```

```
          org      RAM          ;start of static RAM at $C0
OPCODE     rmb     1          ;command byte
ADDR       rmb     1          ;EEPROM address byte
DATA_H     rmb     1          ;MSByte of data
DATA_L     rmb     1          ;LSByte of data
TEST1      rmb     1          ;test byte #1
TEST2      rmb     1          ;test byte #2
TEST3      rmb     1          ;test byte #3
TEST4      rmb     1          ;test byte #4
```

```
*****
*
*
* Program Initialization
*
* This section sets up the port for bit banging.
*
* To prevent floating inputs and associated high current draw,
* the HC705J1A has pulldown devices on all I/O pins. This
* initialization should enable these pulldowns on unused I/O
* pins. RESET_ enables the pulldowns, so no code is required.
*
*****
*
```

```
J9356_START  org      EPROM
             lda      #$80          ;init portA
             sta      PORTA         ;
             sta      COPR          ;kick the wdog
             lda      #$87          ;init i/o of port A
             sta      DDRA
```

```

*****
*
*      J1A_9356 Main Program Loop
*
* It then runs through the test routine to check for
* proper serial transmission. The LED is lit if the test passes.
*
*****

***      Enable erase/write mode of EEPROM
          jsr      J9356_EWEN      ;call ewen routine

***      Erase all EEPROM memory map
          jsr      J9356_ERAL      ;call eral routine

***      Write $AA55 to $00
          lda      #$00            ;load address
          sta      ADDR
          lda      #$AA            ;load data byte high
          sta      DATA_H
          lda      #$55            ;load data byte low
          sta      DATA_L
          jsr      J9356_WRITE     ;call write routine

***      Write $1234 to $20
          lda      #$20            ;load address
          sta      ADDR
          lda      #$12            ;load data byte high
          sta      DATA_H
          lda      #$34            ;load data byte low
          sta      DATA_L
          jsr      J9356_WRITE     ;call write routine

***      Read $00
          lda      #$00            ;load address
          sta      ADDR
          jsr      J9356_READ      ;call read routine
          lda      DATA_H
          sta      TEST1           ;store away data_h to test1
          lda      DATA_L
          sta      TEST2           ;store away data_l to test2

***      Read $20
          lda      #$20            ;load address
          sta      ADDR
          jsr      J9356_READ      ;call read routine
          lda      DATA_H
          sta      TEST3           ;store away data_h to test3
          lda      DATA_L
          sta      TEST4           ;store away data_l to test4

```

```

***      Check results of write and read, light LED if good
J9356_CKSUM    lda      TEST1          ;check test1
               cmpa     #$AA
               bne      J9356_BRANCH   ;branch if no good, no LED

               lda      TEST2          ;check test2
               cmpa     #$55
               bne      J9356_BRANCH   ;branch if no good, no LED

               lda      TEST3          ;check test3
               cmpa     #$12
               bne      J9356_BRANCH   ;branch if no good, no LED

               lda      TEST4          ;check test4
               cmpa     #$34
               bne      J9356_BRANCH   ;branch if no good, no LED

               bclr     7,PORTA         ;EEPROM write and read is good
                                   ; light LED

J9356_BRANCH   clra
               sta      COPR           ;kick the wdog
               bra      J9356_BRANCH
    
```

```
*****
*
*           EEPROM Command SubRoutines
*
* These 7 subroutines execute each of the 7 commands
* that the EEPROM will respond to
*
*****
```

*** EWEN - subroutine to enable write/erase *****

```
J9356_EWEN   lda    #$80          ;load opcode
              sta    OPCODE
              lda    #$C0          ;load address
              sta    ADDR
              bset   CS,PORTA      ;CS line is high
              jsr    J9356_WR_OP   ;write opcode
              jsr    J9356_WR_ADDR ;write address
              bclr   CS,PORTA      ;CS line is low
              rts                  ;return
```

*** EWDS - subroutine to disable write/erase *****

```
J9356_EWDS   lda    #$80          ;load opcode
              sta    OPCODE
              clr    ADDR          ;load addr
              bset   CS,PORTA      ;CS line is high
              jsr    J9356_WR_OP   ;write opcode
              jsr    J9356_WR_ADDR ;write address
              bclr   CS,PORTA      ;CS line is low
              rts                  ;return
```

*** WRITE - subroutine to write EEPROM *****

```
J9356_WRITE   lda    #$A0          ;load opcode
              sta    OPCODE
              bset   CS,PORTA      ;CS line is high
              jsr    J9356_WR_OP   ;write opcode
              jsr    J9356_WR_ADDR ;write address
              jsr    J9356_WR_DATA ;write data
              bclr   SER_OUT,PORTA
              bclr   CS,PORTA      ;CS line is low
              jsr    J9356_WAIT     ;wait until EEPROM is ready
              rts                  ;return
```

```

***      WRAL - subroutine to write all EEPROM *****
J9356_WRAL    lda    #$80            ;load opcode
               sta    OPCODE
               lda    #$40            ;load addr
               sta    ADDR
               bset   CS,PORTA        ;CS line is high
               jsr    J9356_WR_OP     ;write opcode
               jsr    J9356_WR_ADDR   ;write address
               jsr    J9356_WR_DATA   ;write data
               bclr   SER_OUT,PORTA
               bclr   CS,PORTA        ;CS line is low
               jsr    J9356_WAIT      ;wait until EEPROM is ready
               rts                    ;return

```

```

***      READ - subroutine to read EEPROM *****
J9356_READ    lda    #$C0            ;load opcode
               sta    OPCODE
               bset   CS,PORTA        ;CS line is high
               jsr    J9356_WR_OP     ;write opcode
               jsr    J9356_WR_ADDR   ;write address
               bset   SER_CLK,PORTA   ;clock the EE
               bclr   SER_CLK,PORTA
               jsr    J9356_RD_DATA   ;read data
               bclr   SER_OUT,PORTA
               bclr   CS,PORTA        ;CS line is low
               rts                    ;return

```

```

***      ERASE - subroutine to erase EEPROM *****
J9356_ERASE    lda    #$E0            ;load opcode
               sta    OPCODE
               bset   CS,PORTA        ;CS line is high
               jsr    J9356_WR_OP     ;write opcode
               jsr    J9356_WR_ADDR   ;write address
               bclr   CS,PORTA        ;CS line is low
               jsr    J9356_WAIT      ;wait until EEPROM is ready
               rts

```

```

***      ERAL - subroutine to erase all EEPROM *****
J9356_ERAL    lda    #$80            ;load opcode and addr
               sta    OPCODE
               sta    ADDR
               bset   CS,PORTA        ;CS line is high
               jsr    J9356_WR_OP     ;write opcode
               jsr    J9356_WR_ADDR   ;write address
               bclr   CS,PORTA        ;CS line is low
               jsr    J9356_WAIT      ;wait until EEPROM is ready
               rts

```

```

*****
*
*                               EEPROM Supporting SubRoutines
*
* These subroutines support the functions called from the Command subs
*
*****

***      Subroutine to write 3 bit opcode      *****
J9356_WR_OP      ldx      #3T                      ;init counter for LOOP1

*          Write to the serial output pin
J9356_LOOP1      brclr    7,OPCODE,J9356_L1_2      ;if opcode bit7 = 0, goto L1_2
                  bset     SER_OUT,PORTA           ;ser_out = 1
                  bra      J9356_L1_3              ;goto L1_3
J9356_L1_2       bclr     SER_OUT,PORTA           ;ser_out = 0

*          Clock the serial clock pin
J9356_L1_3       bset     SER_CLK,PORTA           ;ser_clk = 1
                  bclr     SER_CLK,PORTA           ;ser_clk = 0
                  asl       OPCODE                  ;rotate the opcode
                  decx      ;decrease counter loop
                  bne       J9356_LOOP1             ;is LOOP1 finished?
                  clra
                  sta       COPR                    ;kick the wdog
                  rts
                  ;return

*** Subroutine to write 8 bit address *****
J9356_WR_ADDR    ldx      #8T                      ;init counter for LOOP2

*          Write to the serial output pin
J9356_LOOP2      brclr    7,ADDR,J9356_L2_2        ;if addr bit7 = 0, goto L2_2
                  bset     SER_OUT,PORTA           ;ser_out = 1
                  bra      J9356_L2_3              ;goto L2_3
J9356_L2_2       bclr     SER_OUT,PORTA           ;ser_out = 0

*          Clock the serial clock pin
J9356_L2_3       bset     SER_CLK,PORTA           ;ser_clk = 1
                  bclr     SER_CLK,PORTA           ;ser_clk = 0
                  asl       ADDR                    ;rotate the addr
                  decx      ;decrease counter loop
                  bne       J9356_LOOP2             ;is LOOP2 finished?
                  rts
                  ;return

```

```

*** Subroutine to write 16 bit data *****
J9356_WR_DATA    ldx      #16T                ;init counter for LOOP4

*      Write the serial output pin with data
J9356_LOOP4      brclr    7,DATA_H,J9356_L4_2  ;if addr bit7 = 0, goto L4_2
                  bset     SER_OUT,PORTA      ;ser_out = 1
                  bra      J9356_L4_3        ;goto L4_3
J9356_L4_2       bclr     SER_OUT,PORTA      ;ser_out = 0

*      Clock the serial clock pin
J9356_L4_3       bset     SER_CLK,PORTA      ;ser_clk = 1
                  bclr     SER_CLK,PORTA      ;ser_clk = 0
                  asl       DATA_L           ;rotate the DATA_L
                  rol       DATA_H           ;rotate the DATA_H
                  decx      ;decrease counter loop
                  bne       J9356_LOOP4       ;is LOOP4 finished?
                  rts                    ;return

***      Subroutine to read 16 bit data *****
J9356_RD_DATA    ldx      #16T                ;init counter for LOOP3

*      Read the serial input pin
J9356_LOOP3      brclr    SER_IN,PORTA,J9356_L3 ;carry bit = serial in
J9356_L3         rol       DATA_L           ;rotate left result
                  rol       DATA_H

*      Clock the serial clock pin
                  bset     SER_CLK,PORTA      ;ser_clk = 1
                  bclr     SER_CLK,PORTA      ;ser_clk = 0
                  decx      ;decrease counter loop
                  bne       J9356_LOOP3       ;is LOOP3 finished?
                  rts                    ;return

*      Wait until write cycle is over
J9356_WAIT       bset     CS,PORTA           ;CS line is high
J9356_W2         clra
                  sta       COPR              ;kick the wdog
                  brclr    SER_IN,PORTA,J9356_W2
                  bclr     CS,PORTA           ;CS line is low
                  rts                    ;return

```

```
*****
*
*      Interrupt and Reset vectors for Main Routine
*
*****
```

```
org      RESET
fdb      J9356_START
```

NOTES

NOTES

Freescale Semiconductor, Inc.

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

Information in this document is provided solely to enable system and software implementers to use Freescale Semiconductor products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits or integrated circuits based on the information in this document.

Freescale Semiconductor reserves the right to make changes without further notice to any products herein. Freescale Semiconductor makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Freescale Semiconductor assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters which may be provided in Freescale Semiconductor data sheets and/or specifications can and do vary in different applications and actual performance may vary over time. All operating parameters, including "Typicals" must be validated for each customer application by customer's technical experts. Freescale Semiconductor does not convey any license under its patent rights nor the rights of others. Freescale Semiconductor products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Freescale Semiconductor product could create a situation where personal injury or death may occur. Should Buyer purchase or use Freescale Semiconductor products for any such unintended or unauthorized application, Buyer shall indemnify and hold Freescale Semiconductor and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Freescale Semiconductor was negligent regarding the design or manufacture of the part.

