MOTOROLA
SEMICONDUCTOR APPLICATION NOTE

Order this document
as AN4003/D

AN4003/D

Testing and programming MCM2814 devices using an M68HC11EVBU Universal Evaluation Board

Andrew Birnie
Product Analysis Lab, East Kilbride, Scotland

Introduction

This application note details the use of an M68HC11EVBU Universal Evaluation Board and the PCbug11 software package for the MC68HC11, to provide a user-friendly system for testing a single MCM2814 serial EEPROM device and programming multiple devices.

Hardware Description

The hardware is based around the M68HC11EVBU, a low cost tool for the debugging and evaluation of MC68HC11 A series and E series MCUs. The case described here utilises the powerful MC68HC711E20 MCU [1].

The EVBU requires a user-supplied 5V DC power supply and an IBM compatible PC, with RS232C communication port (configuration described in Ref [2]). Communication between the MCU and the PC is established via the MCU's Serial Communication Interface (SCI), with the MCU in Bootstrap mode [3,4]. Communication between the MCU and the MCM2814 devices is established via the MCU's Serial Peripheral Interface (SPI). The M-bus mode operation of the MCM2814 is not exercised [5].

Another benefit of this tool is its size. Including the wire wrap development area provided on the EVBU, used here for the MCM2814 sockets, the total board outline is less than 10 x 20 cm. A socket for the testing of a single MCM2814 device is provided, along with 16 other sockets for programming/verifying multiple MCM2814 devices in parallel. The circuit diagram of the completed system is shown in Figure 1. (For a circuit diagram of the EVBU, see Ref [2].)

The M68HC11EVBU has an on-board low-voltage-inhibit (LVI) device (U2), an MC34064, which inhibits board operation below ~4V. However, the HC11 MCU, M68HC11EVBU and MCM2814 will all operate correctly below this voltage, to approximately 2.7V, so if low voltage operation is required, then this device should simply be removed from the M68HC11EVBU.

4/95

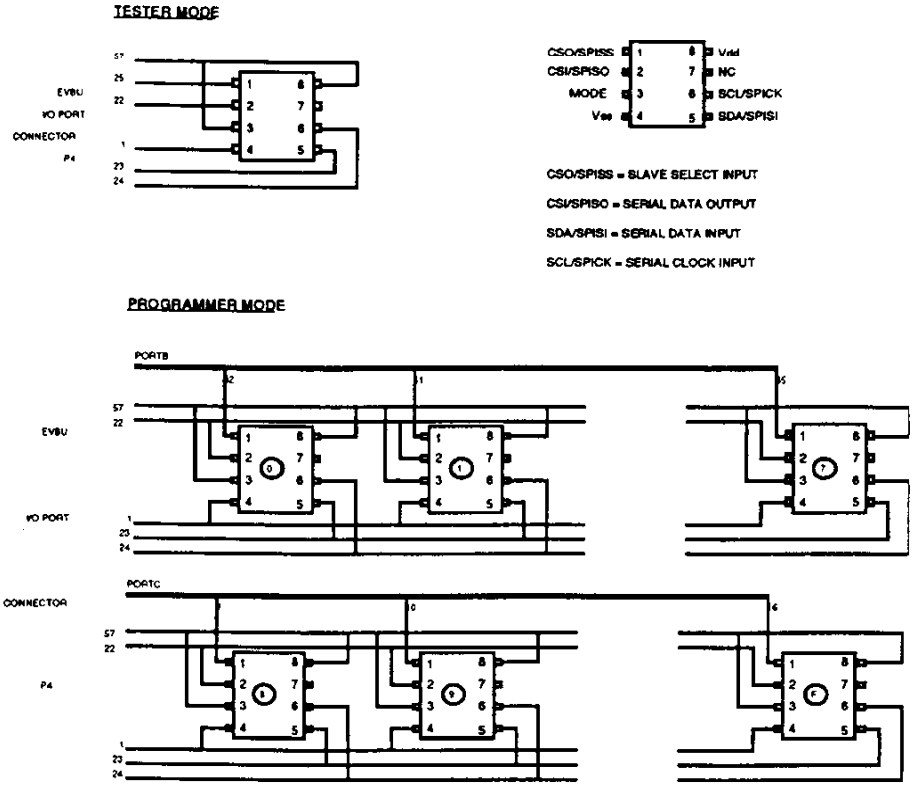


Figure 1. Hardware

Software Description

The system utilises PCbug11, a software package for easy access and manipulation of any MC68HC11 MCU. For detailed information about PCbug11, see Ref [4].

The computer screen for PCbug11 consists of four major areas, or windows (Figure 2):

1. Main window: the upper half of the screen, used for display of messages and memory contents, etc.
2. Register window: the centre of the screen, shows the last recorded contents of the MCU's registers
3. Status window: the centre right of the screen, shows MCU state, etc.
4. Command window: the bottom left of screen, used for command entry from the keyboard and error messages.

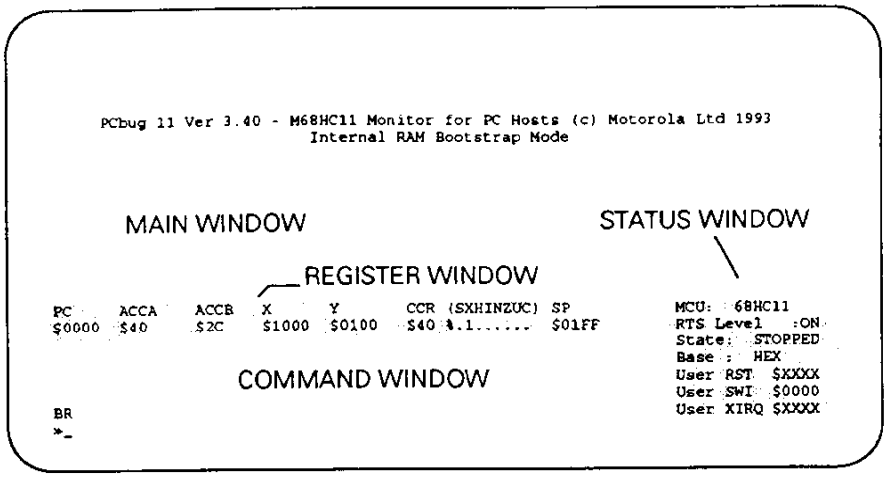


Figure 2. PCbug11 screen

The entire MCM2814 tester/programmer system requires the following software:

MCM2814.bat	batch file to ease start-up
pcbug11.exe	pcbug11 software
pcbug11.hlp	pcbug11 help file
2814welc.mcr	pcbug11 macro functions for MCM2814 'welcome'
2814test.mcr	pcbug11 macro functions for MCM2814 tester commands
2814pgmr.mcr	pcbug11 macro functions for MCM2814 programmer commands
mcm2814.p11	pcbug11 map file, customising pcbug11/EVBU for this application
2814app.s19	HC11 code for MCM2814 manipulation

(Listings of the 68HC11 code are in the appendices.)

Start-up procedure:

power-up EVBU
press EVBU RESET button
type *MCM2814* (from the directory where all the software resides)

The PC screen, after an initial down-loading message, will appear with the PCbug11 screen described above, but with a welcome message and prompt for mode selection (Figure 3).

The HC11 code is designed to be resident in the MCU EPROM. This means that it need only be downloaded prior to the first time of operation. This is done by the following procedure:

connect EPROM programming voltage (15V) to MCU Vpp pin
start-up as described above
type *EPROM $D000 $DFFF* (define EPROM to be used)
type *LOADS 2814APP.S19* (or directory string,
loads into EPROM from address $D000)
type *VERF 2814APP.S19* (check it loaded OK)
if no error message, disconnect Vpp
type *AUTOSTART*

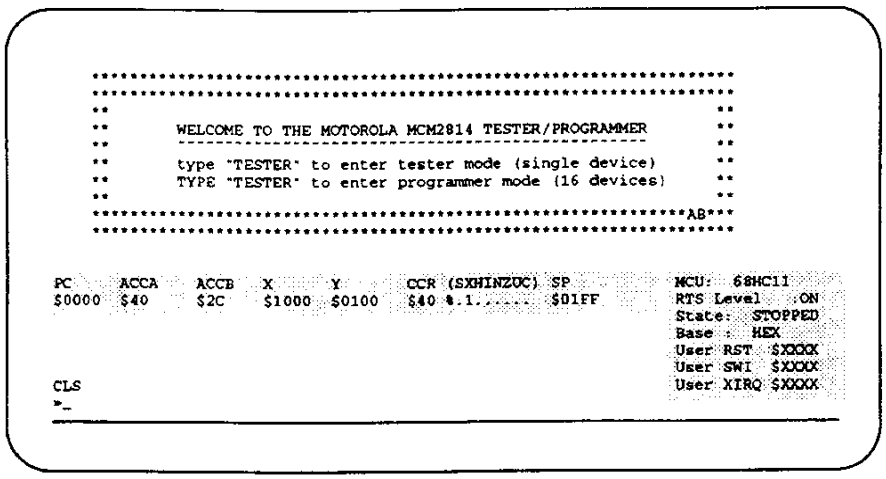


Figure 3. MCM2814 Tester/Programmer welcome message and option selection prompt

If communication problems are experienced, then please refer to the PCbug11 User's Manual - Ref [4].

When finished, type *QUIT* to get out of the PCbug11 environment.

The PCbug11 environment is merely a convenient and user-friendly way to interface to the MCU. It is the MCU on the EVBU which is communicating with the MCM2814 device(s). This communication is via the MCU's Serial Peripheral Interface (SPI). Two bytes of code are sent serially to the MCM2814 device to perform any function - the first byte contains the function, e.g. read, and the second byte the address. For the example of a read, the byte address is echoed, followed by the data held in that byte address. The flow diagrams and the MC68HC11 specific code for the MCM2814 manipulations are included in the appendices.

Modes of operation

When starting the software, upon successful initialisation the user will be prompted to select a mode of operation. There are two modes of operation of the system

(i) a single MCM2814 device tester - invoked by typing *TESTER*

(ii) a multiple MCM2814 device programmer - invoked by typing *PGMR*

Note: PCBug11 software resides in MCU RAM addresses $00-$FF, so transfers to/from the MCM2814 use MCU RAM addresses $100-1FF. During any read or transfer, MCU RAM address $123 corresponds to MCM2814 address $23, etc.

The MCM2814 uses a 3-gate EEPROM cell which allows simultaneous program and erase functions. This eliminates the need to erase an EEPROM byte before programming new data.

"TESTER" commands

In the "tester" mode, communication is established directly with the MCM2814 device on a byte-by-byte basis. (All addresses and data must be entered in Hex. format.)

PR1 dd xx	program address xx with data dd
READ	read addresses $00 through $FF and display
PROG dd	program addresses $00 through $FE with data dd (See Note below)
XFER	transfer RAM addresses $00 through $FF to MCM2814
QPROG dd	"quick program" addresses $00 through $FF with data dd (4 bytes at a time)
TEST	program address with data equal to byte address value
TPROG?	display current program time (x 100ms)
TPROG tt	change program time (x 100ms) - min is 100ms, max is 99.9ms
	(NB Default on any re-boot is 5ms)
DUMP	display RAM & registers
T?	display tester commands

```
*****************************************************************
*                MCM2814 TESTER MODE - COMMANDS                 *
*****************************************************************AB***

PR1 dd aa    - PROG ADDRESS aa WITH DATA dd
READ         - READ ALL BYTES
PROG dd      - PROG ALL BYTES WITH DATA dd
XFER         - TRANSFER RAM TO MCM2814
QPROG dd     - 'QUICK PROG' ALL BYTES WITH DATA dd
TEST         - PROG ALL BYTES TO THEIR ADDRESS
TPROG?       - DISPLAY CURRENT PROG TIME
TPROG tt     - UPDATE PROG TIME tt (x 100us)  (MAX=99.9ms/tt=999)
DUMP         - DISPLAY RAM & REGISTERS
T?           - DISPLAY THIS HELP SCREEN

PC    ACCA   ACCB   X      Y      CCR (SXHINZVC) SP        MCU:  68HC11
$0000 $40    $2C    $1000  $0100  $40 %.1.....  $01FF      RTS Level   :ON
                                                           State:  STOPPED
                                                           Base :  HEX
                                                           User RST  $XXXX
CLS                                                        User SWI  $XXXX
CLS                                                        User XIRQ $XXXX
>_
```

"PROGRAMMER" commands

In the "programmer" mode, communication is established indirectly with the MCM2814 device. The data to be programmed is edited or loaded in MCU RAM, then transferred on mass, and vice versa. In "programmer" mode, all 16 devices are programmed consecutively (even if all sockets are not occupied), but to reduce time 4 EEPROM bytes are always programmed simultaneously.

RAM manipulation commands (for more information, see Ref [4])

BF xx yy dd	block fill addresses xx through yy with data dd
MS xx dd	set address xx to data dd
MD xx yy	display addresses xx through yy
VERF SET xx yy dd	verify addresses xx through yy are all equal to data dd
LOADS filename.s19	load file from hard disk to RAM (Files need to be in standard Motorola S-record format - see Ref [2])

Program/Verify commands:

XFER	Transfer RAM contents to all MCM2814 devices
TPROG?	display current program time (x 100ms)
TPROG tt	change program time (x 100ms) - min is 100ms, max is 99.9ms (NB Default on any re-boot is 5ms)
DUMP	display RAM & registers
LOADP filename.s19	Transfers s19 file from hard disk to RAM to the MCM2814 devices, then verifies that they programmed correctly
READ nn	read addresses 00 through FF of MCM2814 device in programmer socket nn and display
CHK nn	verifies that contents of the MCM2814 device in socket number nn is equivalent to RAM contents
CHKALL	verifies contents of all 16 MCM2814 devices
P?	display programmer mode commands

```
*****************************************************************
*          MCM2814 PROGRAMMER MODE - COMMANDS                   *
*****************************************************************AB***

XFER         - TRANSFER RAM TO ALL DEVICES
TPROG?       - DISPLAY CURRENT PROG TIME
TPROG tt     - UPDATE PROG TIME tt (x 100us) (MAX=99.9ms/tt=999)
DUMP         - DISPLAY RAM & REGISTERS
LOADP file   - TRANSFERS FILE FROM PC TO ALL DEVICES AND CHECKS THEM
READ nn      - TRANSFERS CONTENTS OF DEVICE nn TO RAM AND DISPLAYS
CHK nn       - CHECKS CONTENTS OF DEVICE nn EQUALS RAM
CHKALL       - CHECKS CONTENTS OF ALL DEVICES EQUAL RAM
P?           - DISPLAY THIS HELP SCREEN


PC     ACCA    ACCB   X      Y      CCR (SXHINZVC) SP          MCU:  68HC11
$0000  $40     $2C    $1000  $0100  $40 %.1.....   $01FF       RTS Level   :ON
                                                               State:  STOPPED
                                                               Base :  HEX
CLS                                                            User RST  $XXXX
CLS                                                            User SWI  $XXXX
CLS                                                            User XIRQ $XXXX
>_
```

Note: MCM2814 byte address $FF has security or array write protection features as follows:

Data at $FF	Protected Addresses	No. of Bytes Protected
XXXX 00XX	No write protection	0
XXXX 01XX	$C0 - $FB	60
XXXX 10XX	$80 - $FB	124
XXXX 11XX	$40 - $FB	188

In the "tester" mode, the command *PROG* does not program this byte intentionally, but *QPROG*, *TEST* and *XFER* do. Use the *PR1* command to set this byte before performing any programming of the array. In the "programmer" mode, there are no special considerations for this byte so care must be taken, e.g. if a device in a programmer socket has a non-zero security byte then it will not verify correctly even if the data being programmed has the security byte equal to 00. However, if the operation is repeated it will succeed, since the first attempt will clear the security byte, and the second will program the data.

Conclusions

It has been shown that the M68HC11EVBU Universal Evaluation Board and the PCBug11 software environment make an ideal platform for providing a user-friendly system for testing and programming multiple MCM2814 devices.

References

[1] M68HC11 E Series Technical Data, Motorola MC68HC11E/D

[2] M68HC11EVBU Universal Evaluation Board User's Manual, Motorola M68HC11EVBU/AD1

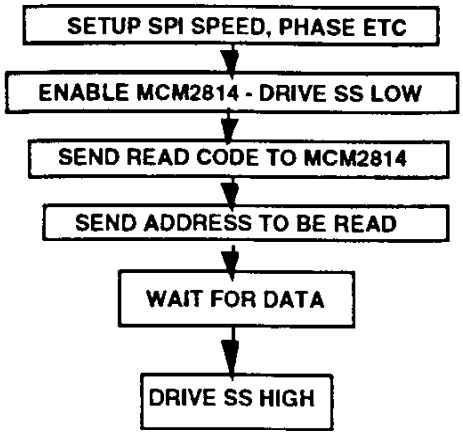
[3] MC68HC11 Bootstrap Mode, Motorola Application Note AN1060/D

[4] PCbug11 User Manual, Motorola M68PCBUG11/D2

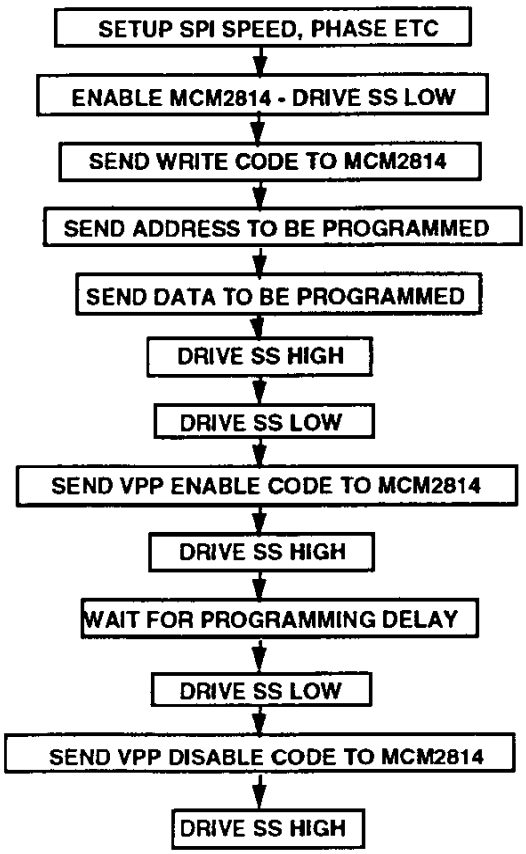
[5] MCM2814 Data Sheet, Motorola MCM2814/D

APPENDICES:

A. SPI flow charts



"Tester" mode - read one MCM2814 byte



"Tester" mode - program one MCM2814 byte

B: HC11 code for MCM2814 manipulation

```
; FILENAME: 2814APP.ASM/LST
;
; CODE FOR MCM2814 TESTER/PROGRAMMER
; BUILT FROM M68HC11EVBU (USING 68HC711E20)
;
; ANDREW BIRNIE
; MOTOROLA   EKB PRODUCT ANALYSIS   FEB 95
;
; NEEDS TO BE USED IN CONJUNCTION WITH
; PCBUG11 (VER340)
; PCBUG11 .P11 FILE "MCM2814.P11" AND
; PCBUG11 MACRO FILES "MCM2814WELC.MCR" & "2814APP.MCR"
;
; TWO MODES OF OPERATION - TESTER & PROGRAMMER
; TESTER MODE USES SPI SS SIGNAL AS DEVICE SS
; PROGRAMMER MODE USES PORTS B & C AS SS LINES
;
; DEFINE REGISTERS
;
SPCR EQU $28
SPSR EQU $29
SPDR EQU $2A
DDRD EQU $09
DDRC EQU $07
PORTD EQU $08
PORTB EQU $04
PORTC EQU $03
;
; DEFINE TEMP STORAGE/MARKERS
;
END EQU $28F          ; END FOR FUNCTION COMPLETE
PR_COMP EQU $28E      ; END FOR PROGRAMMING COMPLETE
CH_PASS EQU $28D      ; END FOR CHECKING DEVICE(S) - PASS
CH_FAIL EQU $28C      ; ....AND FAIL
DEV_ERROR EQU $28B    ; END FOR INPUT ERROR FOR DEVICE NUMBER
ADDR EQU $20C         ;
DATA EQU $202         ; TEMP STORAGE ETC
DEV EQU $204          ;
DEL EQU $208
LINE EQU $20A
NUM EQU $20E
PORT EQU $206
HUND EQU $20F
TOP EQU $100          ; MARKER FOR MCM2814 ARRAY TOP
BOT EQU $1FF          ; .....AND BOTTOM
;
;
                ORG $D000                   ; CODE IN MCU EPROM
;
; TPROG DISPLAY ROUTINE (DISPLAY 4 HEX NO. AS 4 DEC NO.)
; (BOTH MODES)
;
TPROG           LDD #$00                    ; CLEAR TEMP.....
                STD HUND                    ; CONVERSION COUNTER
                LDD DEL                     ; HEX DELAY
                CPD #$64                    ; IF DELAY < 100....
                BLO LOW                     ; .....BRANCH AWAY
                LDX #$64                    ;
                IDIV                        ; DIVIDE DEL BY 10
                STX HUND                    ; STORE NO. OF HUNDREDS
                LDAA HUND+1                 ; GET NO. OF HUNDREDS
                LDAB #$64                   ; 100 (DEC) IN HEX
                MUL
                SUBD DEL                    ; FIND DEL < 100
                LDAA #$00
                NEGB
LOW             LDX #$0A                    ; FIND NO. OF 10'S
                IDIV                        ;
                XGDX                        ; NO. OF 10'S IN ACCD
                LSLD                        ;
                LSLD                        ; SHIFT UP
                LSLD                        ;
                LSLD                        ;
                ABX
                XGDX                        ; REMAINDER IN ACCD
                LDAA HUND+1                 ; ADD ON NO. OF 100'S
                XGDX                        ; XFER BACK TO X
OUT             JMP END                     ; GET OUT - ANSWER IN X
;
```

```
; ROUTINE FOR CHANGING PROG TIME (X IS NEW DEL)
; (BOTH MODES)
;
                STX DEL                 ; STORE KEYBOARD ENTRY
                JMP END                 ; GET OUT
;
; CHECK "ONE 2814 = RAM" ROUTINE
; (PGMR MODE)
;
CHK1            STAB DEV                ; STORE KEYBOARD ENTRY
                JSR DEVICE              ; DECIDE WHICH DEVICE & SS LINE
                JSR PSETUP              ; SPI CONFIG
                LDY #TOP                ; GET 1ST ADDRESS
CH_LOOP         STY ADDR                ; ...TO BE READ
                JSR PREADS              ; READ A BYTE
                LDY ADDR
                CMPA $00,Y              ; DOES 2814 BYTE = RAM ?
                BNE GET_OUT             ; GET OUT - ERROR
                INY                     ; NEXT BYTE
                CPY #BOT+1              ; FINISHED YET ?
                BNE CH_LOOP             ; IF NOT GO AGAIN
                JMP CH_PASS             ; ALL FINISHED - DEVICE PASSES
GET_OUT         JMP CH_FAIL             ; ERROR MARKER
;
; READ CONTENTS OF ONE 2814 ROUTINE
; (PGMR MODE)
;
PREAD           STAB DEV                ; STORE KEYBOARD ENTRY
                JSR DEVICE              ; DECIDE WHICH DEVICE & SS LINE
                JSR PSETUP              ; SPI CONFIG
                LDY #TOP                ; GET TOP ADDR
RD_LOOP         STY ADDR                ; ....TO BE READ
                JSR PREADS              ; READ A BYTE
                LDY ADDR                ; GET ADDRESS
                STAA $00,Y              ; STORE DATA IN RAM
                INY                     ; ARRAY FINISHED YET
                CPY #BOT+1
                BNE RD_LOOP
                JMP END                 ; ALL FINISHED
;
; ROUTINE FOR CHECKING ALL 16 DEVICES = RAM
; (PGMR MODE)
;
CHKALL          LDAB #$00               ; START CHECKING FROM DEVICE #0
                STAB DEV
                JSR PSETUP              ; SPI CONFIG
                JSR DEVICE
CHA_LOOP        JSR CHECK               ; CHECK ONE DEVICE
                INC DEV                 ; ONTO NEXT DEVICE
                LDAB DEV
                CMPB #$10               ; FINISHED ALL F DEVICES ?
                BNE CHA_LOOP            ; IF NOT GO AGAIN
                JMP CH_PASS             ; ALL DEVICES OK
;
; ROUTINE FOR TRANSFER RAM TO ALL MCM2814s
; (PGMR MODE)
;
PXFER           LDAB #$00
                STAB DEV
                JSR PSETUP              ; SPI CONFIG
D_LOOP          JSR DEVICE
                LDY #TOP
                STY ADDR
PX_LP           JSR PPGM                ; PROG A BYTE
                LDY ADDR
                CPY #BOT+1              ; FINISHED YET ?
                BNE PX_LP               ; IF NOT GO AGAIN
                INC DEV
                LDAB DEV
                CMPB #$10
                BNE D_LOOP
                JMP PR_COMP             ; OUT WHEN DONE
;
; READ ALL BYTES
; (TESTER MODE)
;
TREAD           LDY #TOP                ; POINT TO TOP
                JSR TSETUP              ; SPI CONFIG
R_LOOP          JSR TREADS              ; READ 2814 BYTE
                STAA $00,Y              ; STORE IN MCU RAM
                INY                     ; NEXT BYTE
                CPY #BOT+1              ; AT ADDRESS FF YET?
                BNE R_LOOP              ; IF NOT, AGAIN
                JMP END                 ; ALL DONE? - OUT
;
; PROGRAM ONE BYTE (DATA IN ACCA, ADDR IN Y)
```

```
; (TESTER MODE)
;
TPR1            STAA DATA               ; SAVE DATA TO BE PROGGED
                JSR TSETUP              ; SPI CONFIG
                JSR TPGM                ; PROG A 2814 BYTE
                JSR TREADS              ; THEN READ IT BACK
                JMP PR_COMP             ; GET OUT
;
; PROGRAM ALL BYTES (DATA IN ACCA)
; (TESTER MODE)
;
TPRALL          STAA DATA               ; SAVE DATA TO BE PROGGED
                LDY #TOP                ; POINT TO 1ST LOCATION
                JSR TSETUP              ; SPI CONFIG
P_LOOP          JSR TPGM                ; PROG A 2814 BYTE
                INY                     ; NEXT BYTE
                CPY #BOT                ; IF ADDR = FF (SKIP SEC BYTE)
                BNE P_LOOP              ; THEN OUT, IF NOT - AGAIN
                JMP PR_COMP             ; GET OUT
;
; QUICK PROG ALL BYTES (IE 4 AT A TIME) (DATA IN ACCA)
; (TESTER MODE)
;
TQPR            STAA DATA               ; SAVE DATA TO BE PROGGED
                LDY #TOP                ; POINT TO 1ST LOCATION
                JSR TSETUP              ; SPI CONFIG
Q_LOOP          JSR QPROG               ; PROG 4 BYTES
                INY                     ; CALCULATE
                INY                     ; NEXT ADDRESS
                INY                     ; IE PREVIOUS
                INY                     ; PLUS FOUR
                CPY #BOT+1              ; LAST ADDR = FF ?
                BNE Q_LOOP              ; IF NOT, GO AGAIN
                JMP PR_COMP             ; GET OUT
;
; PROGRAM ALL BYTES TO THEIR BYTE ADDRESS
; (TESTER MODE)
;
TPADD           LDY #TOP                ; 1ST LOCATION
                JSR TSETUP              ; SPI CONFIG
PA_LOOP         STY DATA-1              ; "DATA" = LS 8 BITS OF Y
                JSR TPGM                ; PROG A BYTE
                INY                     ; NEXT BYTE
                CPY #BOT                ; LAST ADDR = FF ?
                BNE PA_LOOP             ; IF NOT, GO AGAIN
                JMP PR_COMP             ; GET OUT
;
; ROUTINE FOR TRANSFER RAM TO MCM2814
; (TESTER MODE)
;
TXFER           LDY #TOP                ; 1ST LOCATION
                JSR TSETUP              ; SPI CONFIG
TX_LP            LDAA $00,Y             ; GET RAM BYTE
                STAA DATA               ; STORE AS "DATA TO BE PROGGED"
                JSR TPGM                ; PROG A BYTE
                INY                     ; NEXT BYTE
                CPY #BOT+1              ; REACHED FF YET
                BNE TX_LP               ; IF NOT GO AGAIN
                JMP PR_COMP             ; GET OUT
;
; SUBR FOR SETUP OF SPI FOR PROGRAMMER MODE
; (PGMR MODE)
;
PSETUP          LDX #$1000              ; REGISTER OFFSET
                LDAA #$2F               ; SS HI, SCK LO, MOSI HI
                STAA PORTD,X            ; SS STAYS HI WHEN..
                LDAA #$38               ; ..DDRD5 IS SET
                STAA DDRD,X             ;
                LDAA #$5F               ; SETUP SPI AS MASTER
                STAA SPCR,X             ; & PHASE/SPEED
                LDAA #$FF               ; PORTS B & C = SS LINES
                STAA DDRC,X             ; SET PORT C AS O/P
                STAA PORTC,X            ; & B/C = FF SO NO DEVICES SELECTED
                STAA PORTB,X
                RTS
;
```

```
; PROG SUBROUTINE - PROGS 4 BYTES AT ONCE
; ADDRESS TO BE PROGGED IN Y-REG, DATA TO BE PROGGED IN RAM
; (PGMR MODE)
;
PPGM            LDAB LINE               ; SEND ONLY ONE
                LDY PORT                ; SS LINE OF ONE PORT
                STAB $00,Y              ; LOW AT A TIME
                LDY ADDR                ; GET FIRST ADDRESS
                LDAB #$A2               ; SEND WRITE CODE ..
                STAB SPDR,X             ; .. TO 2814
WAIT1           LDAA SPSR,X             ; WAIT FOR SPIF
                BPL WAIT1               ; FLAG TO BE SET
                NOP
                STY SPSR,X              ; Y-LO TO SPDR, IGNORE Y-HI
WAIT2           LDAA SPSR,X             ; WAIT FOR SPIF
                BPL WAIT2               ; FLAG TO BE SET
                NOP
                LDAB $00,Y              ; GET DATA BYTE
                STAB SPDR,X             ; SEND DATA TO 2814
WAIT3           LDAA SPSR,X             ; WAIT FOR SPIF
                BPL WAIT3               ; FLAG TO BE SET
                INY                     ; NEXT DATA BYTE
                LDAB $00,Y              ; GET DATA BYTE
                STAB SPDR,X             ; SEND DATA TO 2814
WAIT4           LDAA SPSR,X             ; WAIT FOR SPIF
                BPL WAIT4               ; FLAG TO BE SET
                INY                     ; NEXT DATA BYTE
                LDAB $00,Y              ; GET DATA BYTE
                STAB SPDR,X             ; SEND DATA TO 2814
WAIT5           LDAA SPSR,X             ; WAIT FOR SPIF
                BPL WAIT5               ; FLAG TO BE SET
                INY                     ; NEXT DATA BYTE
                LDAB $00,Y              ; GET DATA BYTE
                STAB SPDR,X             ; SEND DATA TO 2814
WAIT6           LDAA SPSR,X             ; WAIT FOR SPIF
                BPL WAIT6               ; FLAG TO BE SET
                INY                     ; POINT TO NEXT DATA BYTE
                STY ADDR
                LDY PORT
                LDAB #$FF               ; ENSURE ALL...
                STAB $00,Y              ; ...SS LINES SENT HIGH
                LDAB LINE               ; SEND ONLY ONE
                STAB $00,Y              ; LOW AT A TIME
                LDAB #$A6               ; SEND VPP ENABLE CODE ..
                STAB SPDR,X             ; .. TO 2814
WAIT7           LDAA SPSR,X             ; WAIT FOR SPIF
                BPL WAIT7               ; FLAG TO BE SET
                NOP
                LDAB #$FF               ; ENSURE ALL...
                STAB $00,Y              ; ...SS LINES SENT HIGH
                JSR DELAY               ; WAIT FOR PROG TIME
                LDAB LINE               ; SEND ONLY ONE
                STAB $00,Y              ; LOW AT A TIME
                LDAB #$A4               ; SEND VPP DISABLE ..
                STAB SPDR,X             ; .. CODE TO 2814
WAIT8           LDAA SPSR,X             ; WAIT FOR SPIF
                BPL WAIT8               ; FLAG TO BE SET
                LDAB #$FF               ; ENSURE ALL...
                STAB $00,Y              ; SENT HIGH
                RTS
;
```

```
; READ SUBROUTINE - READS ONE BYTE, ADDRESS IN ADDR
; EXIT SUBR WITH MCU RAM FULL OF DATA
; (PGMR MODE)
;
PREADS          LDAB LINE               ; SEND ONLY ONE
                LDY PORT                ; SS LINE OF ONE PORT
                STAB $00,Y              ; LOW AT A TIME
                LDY ADDR                ; GET FIRST ADDRESS
                LDAB #$A7               ; SEND READ CODE..
                STAB SPDR,X             ; .. TO 2814
WAITA           LDAA SPSR,X             ; WAIT FOR SPIF
                BPL WAITA               ; FLAG TO BE SET
                STY SPSR,X              ; Y-LO TO SPDR
WAITB           LDAA SPSR,X             ; WAIT FOR SPIF
                BPL WAITB               ; FLAG TO BE SET
                STAB SPDR,X             ; DUMMY SEND
WAITC           LDAA SPSR,X             ; WAIT FOR REAL
                BPL WAITC               ; DATA OUT
                STAB SPDR,X             ; DUMMY SEND
WAITD           LDAA SPSR,X             ; WAIT FOR REAL
                BPL WAITD               ; DATA OUT
                LDAA SPDR,X             ; REAL DATA NOW
                LDAB #$FF               ; ENSURE ALL...
                LDY PORT                ; ....SS LINES....
                STAB $00,Y              ; SENT HIGH
                RTS
;
; DEVICE SUBROUTINE
; INTERPRETS WHICH DEVICE TO EXERCISE & SETS UP SS LINE
; ENTER WITH "DEV" = 0,1,2,...,E,F
; (PGMR MODE)
;
DEVICE          LDAB #$00
                STAB NUM                ; CLEAR "NUM" TEMP STORE
                LDAB #$FE               ; 1ST SS LINE IS BIT 0 OF PORT B
                STAB LINE
                LDAB DEV                ; IS IT DEVICE 0 ?
NXT             CMPB NUM
                BEQ PTB                 ; IF SO BRANCH AWAY
                INC NUM                 ; IF NOT, IS IT DEVICE 1
                SEC
                ROL LINE                ; FOR DEVICE 1, SS IS BIT 1 OF PORT B
                LDAA NUM
                CMPA #$08               ; TIME TO MOVE ONTO PORT C YET ?
                BNE NXT
                LDAB #$FE
                STAB LINE               ; 9TH SS LINE IS BIT 0 OF PORT C
                LDAB DEV
NXT1            CMPB NUM                ; FIND CORRECT SS LINE
                BEQ PTC                 ; FOR DEVICE NUMBERS 8-F
                INC NUM
                SEC
                ROL LINE
                LDAA NUM
                CMPA #$10               ; DEVICE NUMBER > F
                BNE NXT1
                JMP DEV_ERROR           ; DEVICE ERROR - GET OUT PTB
                LDD #$1004              ; SS LINE IN PORT B
                STD PORT
                BRA QUIT
PTC             LDD #$1003              ; SS LINE IN PORT C
                STD PORT
QUIT            RTS
;
; CHECK "ONE 2814 = RAM" SUBROUTINE
; (PGMR MODE)
;
CHECK           LDY #TOP                ; GET 1ST ADDRESS
CK_LOOP         STY ADDR                ; ...TO BE READ
                JSR PREAD               ; READ A BYTE
                CMPA $00,Y              ; DOES 2814 BYTE = RAM ?
                BNE JMP_OUT             ; GET OUT - ERROR
                LDY ADDR
                INY                     ; NEXT BYTE
                CPY #BOT+1              ; FINISHED YET ?
                BNE CK_LOOP             ; IF NOT GO AGAIN
                RTS
JMP_OUT         JMP CH_FAIL             ; ERROR MARKER
;
```

```
; DELAY FOR PROGRAMMING
;
DELAY           LDX DEL                 ; DELAY ="DEL" x 100us
LOOP2           LDAA #$27               ; INNER LOOP
LOOP1           DECA                    ; OF DELAY IS
                BNE LOOP1               ; 100us
                DEX
                BNE LOOP2
                LDX #$1000              ; RESET REGISTER OFFSET
                RTS
;
;
; SUBR FOR SETUP OF SPI
; (TESTER MODE)
;
TSETUP          LDX #$1000              ; OFFSET
                LDAA #$2F               ; SS HI, SCK LO, MOSI HI
                STAA PORTD,X            ; SS STAYS HI WHEN..
                LDAA #$38               ; ..DDRD5 IS SET
                STAA DDRD,X             ; SS GEN O/P
                LDAA #$5F               ; SETUP SPI AS MASTER
                STAA SPCR,X             ; & PHASE/SPEED
                LDAA #$FF
                STAA $07,X
                STAA $03,X              ; ENSURE PGMR SS LINES
                STAA $04,X              ; ARE ALL DISABLED
                RTS
;
; PROG SUBROUTINE
; ADDRESS TO BE PROGGED IN Y, DATA IN "DATA"
; (TESTER MODE)
;
TPGM            BCLR PORTD,X,$20        ; DRIVE SS LOW
                LDAB #$A2               ; SEND WRITE CODE ..
                STAB SPDR,X             ; .. TO 2814
WAITJ           LDAA SPSR,X             ; WAIT FOR SPIF
                BPL WAITJ               ; FLAG TO BE SET
                STY SPSR,X              ; Y-LO TO SPDR, IGNORE Y-HI
WAITK           LDAA SPSR,X             ; WAIT FOR SPIF
                BPL WAITK               ; FLAG TO BE SET
                LDAB DATA               ; GET DATA BYTE
                STAB SPDR,X             ; SEND DATA TO 2814
WAITL           LDAA SPSR,X             ; WAIT FOR SPIF
                BPL WAITL               ; FLAG TO BE SET
                BSET PORTD,X,$20        ; DRIVE SS HIGH
VPPEN           BCLR PORTD,X,$20        ; DRIVE SS LOW
                LDAB #$A6               ; SEND VPP EN CODE ..
                STAB SPDR,X             ; .. TO 2814
WAITM            LDAA SPSR,X            ; WAIT FOR SPIF
                BPL WAITM               ; FLAG TO BE SET
BSET            PORTD,X,$20             ; DRIVE SS HIGH
                JSR DELAY               ; PROG DELAY (DEF 5ms)
                BCLR PORTD,X,$20        ; DRIVE SS LOW
                LDAB #$A4               ; SEND VPP DIS ..
                STAB SPDR,X             ; .. CODE TO 2814
WAITN           LDAA SPSR,X             ; WAIT FOR SPIF
                BPL WAITN               ; FLAG TO BE SET
                BSET PORTD,X,$20        ; DRIVE SS HIGH RTS
;
; READ SUBROUTINE
; ENTER SUBR WITH START ADDRESS IN Y
; EXIT WITH DATA IN ACCA
; (TESTER MODE)
;
TREADS          BCLR PORTD,X,$20        ; DRIVE SS LOW
                LDAB #$A7               ; SEND READ CODE..
                STAB SPDR,X             ; .. TO 2814
WAITP           LDAA SPSR,X             ; WAIT FOR SPIF
                BPL WAITP               ; FLAG TO BE SET
                STY SPSR,X              ; Y-LO TO SPDR
WAITQ           LDAA SPSR,X             ; WAIT FOR SPIF
                BPL WAITQ               ; FLAG TO BE SET
                STAB SPDR,X             ; DUMMY SEND
WAITR           LDAA SPSR,X             ; WAIT FOR REAL
                BPL WAITR               ; DATA OUT
                STAB SPDR,X             ; DUMMY SEND
WAITS           LDAA SPSR,X             ; WAIT FOR REAL
                BPL WAITS               ; DATA OUT
                LDAA SPDR,X             ; REAL DATA NOW
                BSET PORTD,X,$20        ; DRIVE SS HIGH
                RTS
;
;
```

```
; QUICK PROG SUBROUTINE
; DATA TO BE PROGGED IN "DATA", ADDRESS IN Y-REG
; (TESTER MODE)
;
QPROG           BCLR PORTD,X,$20        ; DRIVE SS LOW
                LDAB #$A2               ; SEND WRITE CODE ..
                STAB SPDR,X             ;  .. TO 2814
WAITQ2          LDAA SPSR,X             ; WAIT FOR SPIF
                BPL WAITQ2              ; FLAG TO BE SET
                STY SPSR,X              ; Y-LO TO SPDR, IGNORE Y-HI
WAITQ3          LDAA SPSR,X             ; WAIT FOR SPIF
                BPL WAITQ3              ; FLAG TO BE SET
                LDAB DATA               ; GET DATA BYTE
                STAB SPDR,X             ; SEND DATA 1 TO 2814
WAITQ4          LDAA SPSR,X             ; WAIT FOR SPIF
                BPL WAITQ4              ; FLAG TO BE SET
                STAB SPDR,X             ; SEND DATA 2 TO 2814
WAITQ5          LDAA SPSR,X             ; WAIT FOR SPIF
                BPL WAITQ5              ; FLAG TO BE SET
                STAB SPDR,X             ; SEND DATA 3 TO 2814
WAITQ6          LDAA SPSR,X             ; WAIT FOR SPIF
                BPL WAITQ6              ; FLAG TO BE SET
                STAB SPDR,X             ; SEND DATA 4 TO 2814
WAITQ7          LDAA SPSR,X             ; WAIT FOR SPIF
                BPL WAITQ7              ; FLAG TO BE SET
BSET            PORTD,X,$20             ; DRIVE SS HIGH
                JMP VPPEN               ; DO VPP BIT IN PROG SUBR
;
```

How to reach us:

USA/EUROPE: Motorola Literature Distribution;
P.O. Box 20912; Phoenix, Arizona 85036. 1–800–441–2447

JAPAN: Nippon Motorola Ltd.: Tatsumi–SPD–JLDC, Toshikatsu Otsuki,
6F Seibu–Butsuryu–Center, 3–14–2 Tatsumi Koto–Ku, Tokyo 135, Japan. 03–3521–8315

MFAX: RMFAX0@email.sps.mot.com –TOUCHTONE (602) 244–6609
INTERNET: http://Design–NET.com

HONG KONG: Motorola Semiconductors H.K. Ltd.; 8B Tai Ping Industrial Park,
51 Ting Kok Road, Tai Po, N.T., Hong Kong. 852–26629298

MOTOROLA

AN4003/D