Monitor Program for the MC68HC05B6 Microcomputer Unit

Prepared by: Bill Mathews, Product Engineer, Motorola Semiconductors Ltd, East Kilbride, Scotland

INTRODUCTION

The MC68HC05B6 HCMOS microcomputer is a member of Motorola'sMC68HC05 family of low-cost single-chip microprocessors. This 8-bit microcomputer (MCU) contains an on-chip oscillator, CPU, RAM, ROM, EEPROM, A/D, PULSE LENGTH Modulated outputs, I/O, a serial Communications Interface, Timer system and Watchdog timer.

A monitor program is available in the mask ROM of a 68HC05B6, (XC68HC05B6FN MONITOR), which when used with a monitor circuit module, a power supply, and a video terminal will allow the user to write and debug small portions of 68HC05B6 code. This application note contains a description of the facilities available via the monitor software, a diagram of the monitor circuit, and a listing of the monitor code.

HARDWARE

The monitor module requires a single(+5V) power supply, and all communications between the device and terminal take place via an RS-232 link. All 68HC05B6 I/O pins are available to the user to be configured as required.

Terminal setup is as follows:
9600 Baud, Half Duplex,
and either 7 bit data and parity '0'
or 8 bit data and no parity.

A circuit diagram of the monitor is given in Figure 1.

MONITOR OPERATION

The following sequence of operations should be followed:

1. Remove power supply from module
2. Insert XC68HC05B6FN MONITOR device
3. Place S1 in 'RESET' position
4. Apply +5V dc power supply
5. Connect video terminal
6. Place S1 in 'RUN' position

The 68HC05B6 will then begin to execute the monitor program, and the following message should appear on the monitor:

"Hi! I'm the MC 68HC805 B6 from MOTOROLA European Design Centre, Geneva."

A prompt "." will be displayed to indicate that the device is ready to receive commands from the terminal. If no message appears, then the setup of the terminal should be verified.

MONITOR COMMANDS

The following commands are available:

Command Description

R — Display the content of the registers in format.

HINZC AA XX PPPP ZZ = DD where

HINZC	=	Condition code register
AA	=	Contents of Accumulator
XX	=	Contents of Index Register
PPPP	=	Program Counter
ZZ,DD	=	User specified byte (which addresses f00 to fFF) and contents. Set up using the 'V' command. Reset initialises ZZ to f08 (A/D data register).

Note that this command assumes that the stack pointer is at address $FA.

A — Display/Change the Accumulator

The contents of the Accumulator are displayed, then the monitor waits for input of two hex digits (new accumulator contents). Typing a carriage return will return the program to the command mode with the contents of the Accumulator unchanged.

Note that this command assumes that the Stack Pointer is at address $FA.

X — Display/Change the Index Register

The contents of the Index Register are displayed, then the monitor waits for the input of two hex digits (new Accumulator contents). Typing a carriage return will return

the program to the command mode with the contents of the Index Register unchanged.

Note that this command assumes that the Stack Pointer is at address $FA.

M nnnn Examine/Change Memory

The contents of any address in the range $000 to $1FF (Register, RAM and EEPROM1) are displayed, and the program will await further input, namely:

.	Redisplay the contents of the current address
^	Display the contents of the previous address (nnnn-1)
<CR>	Open the next address (nnnn+1)
+	Increase the contents of the open location by 1
-	Decrease the contents of the open location by 1
/D	Replace the contents of the open location with the Ascii code for alphanumeric character D, and go to the next address.
nn	Replace the contents of the currently open address with two hex digits nn and go to the next address

Typing any other character will return the monitor to the command mode.

L nnnn List a block of memory starting at address nnnn. The default address (if nnnn is not specified) is $100. The data is displayed on screen as four blocks of eight by eight bytes, with the address printed at every sixteenth byte.

V nn Change the address of the page zero byte displayed with the R command with the hex byte specified by nn.

K nn Set Program and Erase times for operations on EEPROM1 where nn is milliseconds entered in decimal. The default values are 10ms. Typing 'enter' will skip the command.

P nn Program the entire EEPROM1 array (except address $100) with nn. This command may take some time to execute as firstly the entire array is erased, then each byte is programmed in turn.

If non-hex data is entered, then the following commands are available:

Z	Program the entire array (except address $100) with Data = Address, i.e. Address $100 = not programmed Address $101 = $01 Address $102 = $02 Address $103 = $03 ...etc.
P	Program a chequer-board pattern
Q	Program an inverse chequer-board pattern
	Any other character will exit this command.
E nnnn	Start execution at address nnnn
C	Continue program execution according to the current program counter, accumulator, index register and condition code register stored on the stack.

BREAKPOINTS AND INTERRUPTS

The SWI instruction may be used as a breakpoint. To continue following a breakpoint first replace the SWI with another command (such as NOP) then type 'C' to continue.

The interrupt vectors point to the RAM as shown below, and are spaced three bytes apart allowing the use of a JMP or BRA instruction to a service routine located in either RAM or EEPROM1.

Vector	Address	
SCI	$00DF	
Timer Overflow	$00E2	
Timer O/P CMP	$00E5	
Timer I/P CAP	$00E8	
IRQ	$00EB	
SWI	$08A6	Pointing to monitor for breakpoint
RESET	$0C22	Start of monitor code

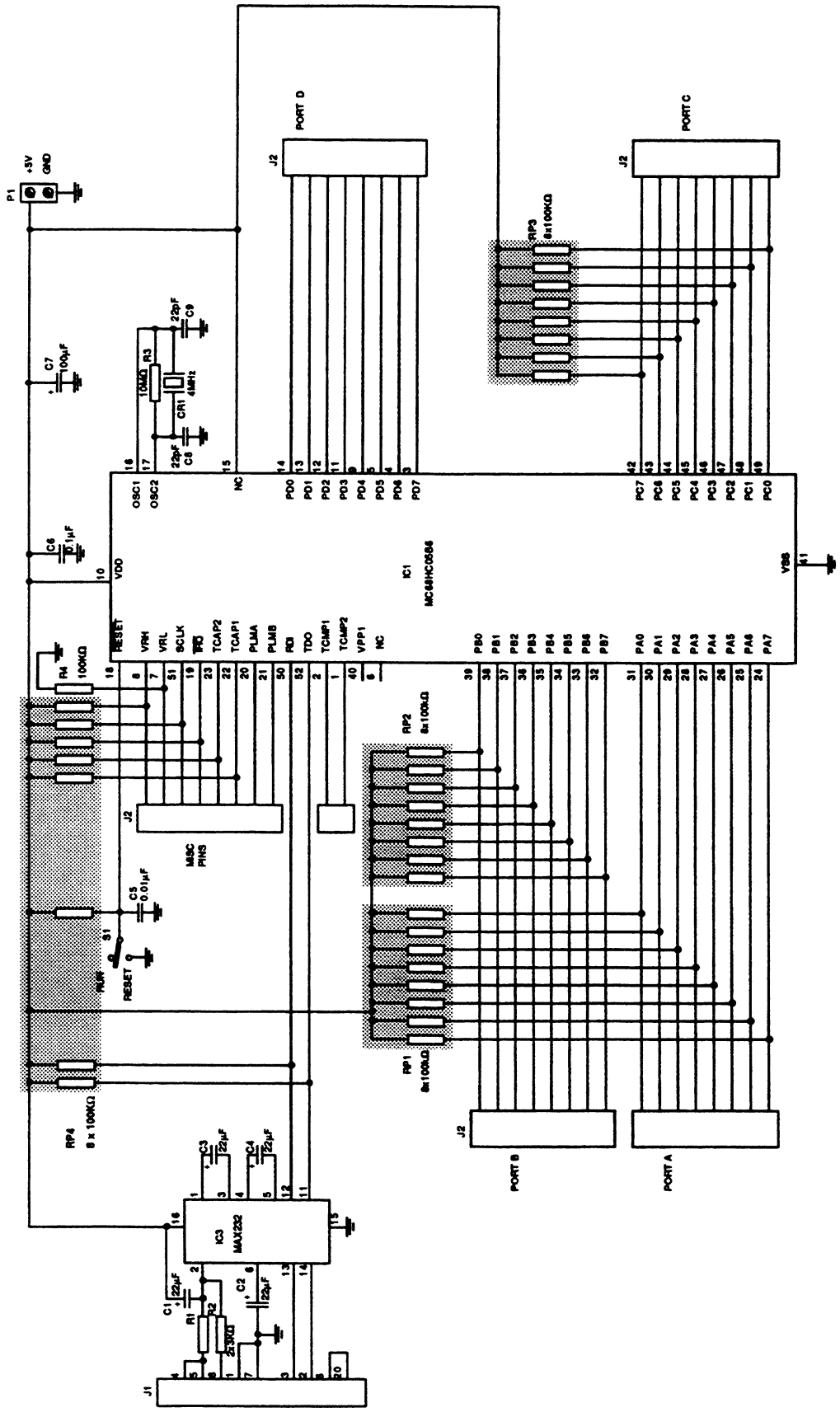

Figure 1 - 68HC05B6 Monitor Circuit Diagram

```
                TTL     Freescale Semiconductor, Inc. MONITOR
                OPT     P=58,LLE=118,CRE
*****************************************************************************
*
*           F I R M W A R E  F O R  T H E  M C 6 8 H C 0 5 B 6
            ==================================================

            MONITOR LISTING ONLY
            ====================

        I/O and INTERNAL registers definition


        I/O registers

)RTA    EQU     $00         port A.
)RTB    EQU     $01         port B.
)RTC    EQU     $02         port C.
)RTD    EQU     $03         port D.
)RA     EQU     $04         port A DDR.
)RB     EQU     $05         port B DDR.
)RC     EQU     $06         port C DDR.

        EEPROM register

:CONT   EQU     $07         EEPROM control register.
:1PGM   EQU     0
:1LAT   EQU     1
:1ERA   EQU     2

        A/D registers

)DATA   EQU     $08         A/D data register.
)STCT   EQU     $09         A/D status and control register.
:OCO    EQU     7                Conversion complete flag.

        PLM registers

:MA     EQU     $0A         pulse length mod reg A.
:MB     EQU     $0B         pulse length mod reg B.

        Miscellaneous register

:SC     EQU     $0C         Miscellaneous register.
WDOG    EQU     0               Watchdog control bit.
SM      EQU     1                Slow Mode.

        SCI registers

AUD     EQU     $0D         SCI baud register.
:CR1    EQU     $0E         SCI control register 1.
:CR2    EQU     $0F         SCI control register 2.
SBK     EQU     0                Send break bit.
:SR     EQU     $10         SCI status register.
```

```
  DRF   EQU     5       Receive data register full flag.
SCDAT   EQU     $11     SCI data register.
*
*       TIMER registers
*
TIMCTL  EQU     $12     Timer control register.
.TOIE   EQU     5       Timer overflow interrupt enable.
.OCIE   EQU     6       Timer output compares interrupt enable.
.ICIE   EQU     7       Timer input captures interrupt enable.
TIMST   EQU     $13     Timer status register.
.OCF2   EQU     3            Timer output compare 2 flag.
.ICF2   EQU     4            Timer input capture 2 flag.
.TOF    EQU     5            Timer overflow flag.
.OCF1   EQU     6            Timer output compare 1 flag.
.ICF1   EQU     7            Timer input capture 1 flag.
TIMIC1  EQU     $14     Timer input capture register 1 (16-bit).
TIMOC1  EQU     $16     Timer output compare register 1 (16-bit).
TIMCTR  EQU     $18     Timer free running counter (16-bit).
TIMALT  EQU     $1A     Timer alternate counter register (16-bit).
TIMIC2  EQU     $1C     Timer input capture register 2 (16-bit).
TIMOC2  EQU     $1E     Timer output compare register 2 (16-bit).
*
*
*       MEMORY MAP DEFINITION
*
*
TEST       EQU    $20                  TEST register
ROM0    EQU     $0020   Start address of ROM0.
RAM     EQU     $0050   Start address of RAM.
EEPROM  EQU     $0100   Start address of EEPROM 256 bytes. NOT USE IN B4
.SEC    EQU     0       Security bit.
ROM1    EQU     $0200   Start address of ROM1.
ROM1ND  EQU     $02BF   End address of ROM1.
ROM2    EQU     $0800   Start address of ROM2. $0F00 IN B4
ROMBOT  EQU     $1F00   Start address of bootstrap ROM2.
*
*
*       Miscellaneous definitions and equates
*
*
RAM1    EQU     RAM+1   Working memory.
RAMINT  EQU     RAM+10  RAM location used for interrupt test.
VECT    EQU     $1FE2   Start of bootstrap vectors.
*
*       Synthetized instructions
*
EOR     EQU     $D8     Exclusive or, 2 bytes indexed.
STA     EQU     $C7     Store A, extended mode.
                PAGE
*******************************************************************
*
*         M O N I T O R  P R O G R A M
*         ============================
*
*         Rev                         1987/07/22
```

```
QU   '1
REG  EQU  '3
*
*          Programmer :      O. Pilloud  MOT GVA
*
        THE MONITOR HAS THE FOLLOWING COMMANDS :


        R       DISPLAY OF REGISTERS  HINZC AA XX PPPP VV = DD
                WHERE DD IS THE CONTENT OF ANY DIRECT PAGE ADDRESS VV


        A       DISPLAY AND CHANGE ACCA


        X       DISPLAY AND CHANGE INDEX


        M.... MEMORY EXAMINE/CHANGE
                  +    INCREMENT DATA BY 1
                  -    DECREMENT DATA BY 1
                  .    RE-READ SAME ADDRESS
                  ¬    READ PREVIOUS ADDRESS
                  CR   READ NEXT ADDRESS
                  DD   CHANGE DATA
                  /D   CHANGE DATA WITH ASCII VALUE OF D
                ANYTHING ELSE EXITS MEMORY COMMAND


        C       CONTINUE EXECUTION


        E.... EXECUTE


        L.... LIST A 16 BY 16 BLOCK OF MEMORY
                IF NO ADDRESS SUPPLIED, EEPROM1 IS LISTED


        V..   CHANGE ADDRESS OF MEMORY DISPLAYED WITH R COMMAND


        K       SET WRITE AND ERASE CONSTANTS IN ms

        P..   PRESET EEPROM1 WITH SUPPLIED DATA, EXCEPT ADDR $100
                IF NON HEX DATA, THE FOLLOWING CAN OCCUR :

                  Z   PRESETS DATA = ADDRESS
                  P   CHECKER BOARD PATTERN (STARTING WITH 55)
                  Q   CHECKER BOARD PATTERN (STARTING WITH AA)

                FOR THE 3 CASES ABOVE, NO ERASE IS PERFORMED
          PAGE
           MEMORY MAP DEFINITIONS
```

```
*  Note : TEST is a write only register and its access is
*         authorized only when E1LAT is cleared. When E1LAT
*         is set, address $20 (P0EEP6) is accessed for writing.
*
*         DEFINITION OF MONITOR VALUES
*
STACK     EQU    $FA                 STACK FOR MONIT
*
*         Miscellaneous definitions and equates
*
HI        EQU    0                   hi byte offset
LO        EQU    1                   lo byte offset
LONG      EQU    $C4                 long timing factor (100 ms nominal)
SHORT     EQU    $14                 short timing factor (10 ms nominal)
RED       EQU    PLMA                red LED on PLMA
GREEN     EQU    PLMB                green LED on PLMB
E1BW      EQU    7                   bulk bit of TEST register
E6PGM     EQU    4                   EECONT
E6LAT     EQU    5                   EECONT
E6ERA     EQU    6                   EECONT
TDRE      EQU    7
MBIT      EQU    4                   8 data bits flag in SCCR1
ADON      EQU    5                   A/D converter control bit
*
*
* MONITOR DEFINITIONS
*
MPRT    EQU     '.
EPRT    EQU     '*
FWD     EQU     $0D
BACK    EQU     '¬
SAME    EQU     '.
PLUS    EQU     '+
MINUS   EQU     '-
ASCII   EQU     '/
SPACE   EQU     $20
CR      EQU     $0D
LF      EQU     $0A
BEEP    EQU     $07
EOT     EQU     $04
*
* MONITOR AND COMMON RAM DEFINITIONS
*
                ORG       RAM
*
COUNT   RMB     1         BINBCD
CHAR    RMB     1         CURRENT INPUT/OUTPUT CHARACTER
XTEMP   RMB     1         TEMP FOR GETC,PUTC
ATEMP   RMB     1         TEMP FOR GETC,PUTC
MEMADD  RMB     1         MEMORY ADDRESS FOR REGS
FLAG    RMB     1         ITEM COUNTER & TEMP FOR EEPROM R/W
GET     RMB     4         FOR PICK AND DROP SUBROUTINES
WRITEK  RMB     1         CONSTANT FOR EEPROM WRITE TIME
ERASEK  RMB     1         CONSTANT FOR EEPROM ERASE TIME
ASC     RMB     1         / FLAG
```

```
*
*
*            M O N I T O R   P R O G R A M

 ===========================================================================



              ORG       ROM0

       FCC /Rev /
       FCB  REV
       FCC /./
       FCB  REL
       FCC /   /

 ONE ROUTINE IN PAGE 0 - JUST FOR THE (C)HECK OF IT

 PCC --- PRINT CONDITION CODE REGISTER

CC     LDA     STACK+1   GET CCR
       ASLA              MOVE H BIT TO BIT 7
       ASLA
       ASLA
       STA     GET       SAVE IT
       CLRX
CC2    LDA     #'.
       ASL     GET       PUT BIT IN CARRY BIT
       BCC     PCC3      BIT OFF MEANS PRINT '.'
       LDA     CCSTR,X   PICKUP APPROPRIATE CHAR
CC3    JSR     PUTC      PRINT '.' OR CHAR
       INCX              POINT TO NEXT IN STRING
       CPX     #5        5 BITS ARE GOOD ENOUGH
       BLO     PCC2
       RTS

 Fill page 0 ROM

        FCC       /0123456789AB/

*****************************************************************************
        PAGE
             ORG     ROM2

  NOW, OTHER ROUTINES IN MAIN EEPROM

  TABLES

BTBL  FCB     $A,$14,$28,$50
```

```
CCSTR   FCC     /HINZC/
*
MSG     FCC     /Hi ! I'm the MC 68HC05 B6 /
        FCC     /From MOTOROLA European /
        FCC     /Design Center, Geneva/
        FCB     EOT
*
CMA     EQU     *       COMMON MESSAGE AREA
NGM     FCB     '?,BEEP,CR,LF,EOT
ERA     FCC     /Erase : /
        FCB     EOT
WRI     FCC     /Write : /
        FCB     EOT
MS      FCC     / ms /
        FCB     EOT
*
*
* SETA --- EXAMINE / CHANGE ACCUMULATOR
*
SETA    LDX     #STACK+2 POINT TO A
        BRA     SETANY
*
* SETX --- EXAMINE / CHANGE INDEX
*
SETX    LDX     #STACK+3 POINT TO X
*
* SETANY - PRINT £X| AND CHANGE IF REQUESTED
*
SETANY  LDA     ,X      PICK UP THE DATA
        JSR     PUTBYT  AND PRINT IT
        JSR     PUTS    AND A SPACE
        JSR     GETBYT  CHANGE ?
        BCS     BLEEP   ERROR, NO CHANGE
        STA     ,X      ELSE REPLACE WITH NEW VALUE
        BRA     MAIN    NOW RETURN
*
* REGS --- PRINT CPU REGISTERS
*
REGS    JSR     PCC     PRINT CCR
        JSR     PUTS    SEPARATE FROM NEXT STUFF
        CLR     GET+1+HI POINT TO PAGE ZERO
        LDA     #STACK+2
        STA     GET+1+LO POINT TO A ON STACK
        JSR     OUT2HS  NOW PRINT A
        JSR     OUT2HS  AND X ,
        JSR     OUT4HS  THE PC,
        JSR     DISADD  THE CURRENT ADDRESS
        LDA     #'=
        JSR     PUTC    PRINT '='
        JSR     PUTS    SPACE
        CLR     GET+1+HI CLR ADDRESS CARRY
        JSR     PRDAT   FINALLY THE DATA AND A SPACE
*       FALL INTO MAIN LOOP
*
* MAIN --- PROMPT , GET AND DECODE COMMAND
```

```
        JSR     CRLF        START A NEW LINE
        LDA     #MPRT
        JSR     PUTC        PRINT THE PROMPT
        JSR     GETC        GET COMMAND
        JSR     PUTS        PRINT SPACE

        CMP     #'A         CHANGE A
        BEQ     SETA
        CMP     #'X         CHANGE X
        BEQ     SETX
        CMP     #'R         REGISTERS
        BEQ     REGS
        CMP     #'E         EXECUTE
        BEQ     EXEC
        CMP     #'C         CONTINUE
        BEQ     CONT
        CMP     #'M         MEMORY
        BEQ     MEMORY
        CMP     #'L         LIST BLOCK OF MEM
        BEQ     LISTR
        CMP     #'V         ADDRESS CHANGE
        BEQ     MEMDIS
        CMP     #'P         PRESET EEPROM1
        BEQ     PRESTR
        CMP     #'K         SET CONSTANTS
        BEQ     CONSTR
        FALL INTO BLEEP

BLEEP -- PRINT '?' AND PROTEST

EEP   LDX     #NGM-CMA ?+BEEP MESSAGE
      JSR     PUTMSG
      BRA     MAIN


STR   JMP     LIST
ESTR  JMP     PRESET
NSTR  JMP     CONSTS

EXEC --- EXECUTE FROM GIVEN ADDRESS

EC    JSR     GETBYT    GET HIGH BYTE
      BCS     BLEEP     NOT A HEX DIGIT
      AND     #$1F      MAX ADDR $1FFF
      TAX               SAVE FOR A SECOND
      JSR     GETBYT    NOW THE LOW BYTE
      BCS     BLEEP     BAD ADDRESS
      STA     STACK+5   PC LOW
      STX     STACK+4   PC HIGH

CONT --- CONTINUE USER PROGRAM

NT    RTI               COULDN'T BE SIMPLER
```

```
* MEMDIS - MEMORY ADDRESS CHANGE (FOR REGISTER DISPLAY)
*
MEMDIS  JSR     DISADD    PRINT PRESENT ADDRESS
        JSR     GETBYT    GET NEW VALUE
        BCS     BLEEP     RETURN FOR NON VALID INPUT
        STA     MEMADD
        BRA     MAIN      RETURN
*
* MEMORY - MEMORY EXAMINE / CHANGE
*
MEMORY  JSR     MEM8      GET ADDRESS
        BCS     BLEEP     NOT HEX CHAR
MEM2    JSR     CRLF      BEGIN NEW LINE
        JSR     PRADD     PRINT CURRENT ADDRESS
        JSR     PRDAT     AND ASSOCIATED DATA
        JSR     GETBYT    TRY TO GET A BYTE
        BCS     MEM3      MIGHT BE A SPECIAL CHAR
MEMA    JSR     DROP      OTHERWISE, PUT IT AND CONTINUE
MEM4    JSR     BUMP      GOTO NEXT ADDRESS
        BRA     MEM2      AND REPEAT
*
MEM3    CMP     #SAME     RE-EXAMINE SAME ?
        BEQ     MEM2      YES, RETURN WITHOUT BUMPING
        CMP     #FWD      GO TO NEXT ?
        BEQ     MEM4      YES, BUMP THEN LOOP
        CMP     #BACK     GO BACK ONE BYTE ?
        BEQ     MEM5      YES, GO DECREMENT ADDRESS
        CMP     #ASCII    NEXT VALUE ASCII ?
        BEQ     MEM9      GO READ VALUE
        CMP     #PLUS     INCREMENT DATA ?
        BEQ     MEM6      YES, GO READ DATA
        CMP     #MINUS    DECREMENT DATA ?
        BNE     BLEEP     NO, EXIT MEMORY COMMAND
*
        JSR     PICK      GET THE DATA BYTE
        DECA
        BRA     MEM7      AND GO PUT IT BACK
MEM6    JSR     PICK      GET THE DATA BYTE
        INCA
MEM7    JSR     DROP      PUT IT BACK
        BRA     MEM2      READY
*
MEM5    DEC     GET+1+LO  DECREMENT LOW BYTE
        LDA     GET+1+LO  CHECK FOR UNDERFLOW
        CMP     #$FF
        BNE     MEM2      NO UNDERFLOW
        DEC     GET+1+HI
        LDA     GET+1+HI  SAME FOR HIGH BYTE
        CMP     #$FF
        BNE     MEM2      OK
        LDA     #$1F      HIGHEST ADDRESS IS $1FFF
        STA     GET+1+HI
        BRA     MEM2
*
MEM9    JSR     GETC      READ 1 BYTE
```

```
        BRA     MEMA
MEM8    JSR     GETBYT    BUILD ADDRESS
        BCS     MEND      NOT HEX CHAR
        STA     GET+1+HI
        JSR     GETBYT
        BCS     MEND
        STA     GET+1+LO ADDRESS IS NOW COMPLETE
END     RTS              AND IN GET+1 HI & LO

 BULKW -- BULK WRITE EEPROM1 - DATA IS IN A - A IS UNCHANGED
          BULK OPERATION BEING DESABLED IN USER MODE, THE
          BULK OPERATIONS ARE PERFORMED BY SUCCESSIVE
          BYTE OPERATIONS.

ULKW    LDX     #01
        STX     GET+1+LO
        STX     GET+1+HI SET UP ADDRESS  $101
NTBTR   BSR     BYTEW     BYTE WRITE
        JSR     BUMP
        BRCLR   1,GET+1+HI,NXTBTR LOOP
        RTS

 BULKE -- BULK ERASE OF EEPROM1 - A IS UNCHANGED
          SEE ABOVE ABOUT BULK OPS

ULKE    CLR     GET+1+LO
        LDX     #01       SET UP ADDRESS
        STX     GET+1+HI
NTBTE   BSR     BYTER     BYTE ERASE
        JSR     BUMP
        BRCLR   1,GET+1+HI,NXTBTE LOOP
        RTS

 PRESET - EEPROM SET OR ERASE

NESET   JSR     GETBYT    GET DATA BYTE
        BCS     BLEEPZ    MAY BE SPECIAL COMMAND
        BSR     BULKE     ERASE E2PROM
        BSR     BULKW     WRITE E2PROM
NINT    JMP     MAIN      RETURN

NEEPZ   CMP     #'Z       DATA = ADDRESS ?
        BEQ     DADD
        CMP     #'P       CHCKBOARD 5'S ?
        BEQ     PS5
        CMP     #'Q       CHCKBOARD A'S ?
        BEQ     PSA       ELSE FALL INTO BLEEP


NEEPK   LDA     SCDAT     CLEAR RX STATUS
NEEPR   JMP     BLEEP


 DADD --- PRESET EEPROM WITH DATA = ADDRESS
```

```
*           (ADDRESS $100 UNCHANGED)
*
DADD    LDA     #$01      BOTTOM OF EEPROM
        STA     GET+1+HI
DADLP   STA     GET+1+LO ADDRESS LSB
        BSR     BYTEW     WRITE DATA IN A
        INCA              NEXT ADDRESS, NEXT DATA
        BNE     DADLP     NO OVERFLOW YET
DADND   BRA     MAINT     JOB DONE
*
*
* CHECKBOARD PATTERNS
*
*
PS5     LDA     #$55      START VALUE
        BRA     PSC
PSA     LDA     #$AA
*
PSC     LDX     #01
        STX     GET+1+LO
        STX     GET+1+HI
*
        STA     ATEMP
AX25    LDA     #$0F
        AND     GET+1+LO
        BNE     TNC
        COM     ATEMP
*
TNC     LDA     ATEMP
        JSR     BYTEW
        JSR     BUMP
        BRCLR   1,GET+1+HI,AX25
        BRA     DADND
*
*
* BYTEW -- BYTE WRITE TO EEPROM1 - DATA IS IN A
*          ADDRESS IN GET+1&2 - A IS UNCHANGED
*
BYTEW   STA     FLAG      SAVE A
        BSET    .E1LAT,EECONT PROG LATCH ENABLE
        JSR     DROP1     PUT ADDRESS + DATA
        BSET    .E1PGM,EECONT
        LDA     WRITEK    GET WRITE TIMING CONSTANT
        JSR     ADJDEL
        BCLR    .E1LAT,EECONT END OP
        LDA     FLAG      RESTORE A
        RTS
*
* BYTER -- BYTE ERASE - A IS UNCHANGED AND ADDRESS
*          IS IN GET+1 HI & LO
*
BYTER   STA     FLAG      SAVE DATA
        BSET    .E1LAT,EECONT PROG LATCH ENABLE
        BSET    .E1ERA,EECONT
        JSR     DROP1     PUT ADDRESS
```

```
        BSET    .E1PGM,EECONT
        LDA     ERASEK    GET ERASE TIMING CONSTANT
        JSR     ADJDEL
        BCLR    .E1LAT,EECONT
        LDA     FLAG      RESTORE A
        RTS
;
; LIST --- LIST 4 BLOCKS OF 64 BYTES ON ONE PAGE
;
LIST    JSR     MEM8      GET START ADDRESS
        BCC     GOL       GOOD ADDRESS
        LDA     #01
        STA     GET+1+HI  SET EEPROM1 ADDRESS
        CLR     GET+1+LO
;
GOL     JSR     CRLF
        CLRX              COUNTER 256 BYTES
LL1     JSR     PRADD     PRINT ADDRESS
        JSR     PUTS      AND A SPACE
LNEXT   JSR     PRDAT     PRINT DATA
        JSR     BUMP      NEXT BYTE
        STX     XTEMP     A GOOD THOUGHT FOR
        JSR     ADJ10     UNBUFFERED TERMINALS
        LDX     XTEMP
        DEX               DATA COUNT DECREMENT
        TXA
        AND     #%00001111 TEST FOR 16TH DATA BYTE
        BEQ     LINE
        AND     #%00000111 TEST FOR 8TH DATA BYTE
        BNE     LNEXT
        JSR     PUTS      INSERT EXTRA 2 SPACES
        JSR     PUTS
        BRA     LNEXT
MEC9BH  JMP     MAIN
;
LINE    BSR     ASCIIR    PUT ASCII LINE
        CPX     #00
        BEQ     MEC9BH
        JSR     CRLF
        CMPX    #128      TEST FOR 128 BYTE PAGE
        BNE     NOPAGE
        JSR     CRLF      ADD EXTRA LINE IF TRUE
NOPAGE  BRA     LL1
;
ASCIIR  LDA     GET+1+LO  DECREMENT MEMORY POINTER
        SUB     #16
        STA     GET+1+LO
        BCC     NOBORO
        DEC     GET+1+HI
        LDA     GET+1+HI
        AND     #$1F      JUST IN CASE
        STA     GET+1+HI
NOBORO  TXA               BACK UP BYTE COUNTER
        ADD     #16
        TAX
```

```
        JSR     PUTS        SEPARATE WITH 2 SPACES
        JSR     PUTS
LPASC   BSR     PUTASC      PUT CHAR
        JSR     BUMP
        DEX
        TXA
        AND     #$0F
        BEQ     LINER
        BRA     LPASC       FINISH THIS LINE
LINER   RTS
*
PUTASC  JSR     PICK
        AND     #$7F
        CMP     #$20
        BHS     DISP        '.' BELOW $20
        LDA     #'.
DISP    CMP     #$7F
        BNE     DISP1
        LDA     #$20        SPACE FOR DEL ($7F)
DISP1   JSR     PUTC
        RTS
*
*
* DISPLAY AND CHANGE WRITE AND ERASE CONSTANTS
*
CONSTS  CLR     FLAG        0=ERA; 1=WRI
        LDX     #ERA-CMA
        BRA     CM1
CML     LDX     #WRI-CMA
CM1     JSR     CRLF
        JSR     PUTMSG      PUT EITHER WRI OR ERA
        LDA     ERASEK
        BRCLR   7,FLAG,CM2
        LDA     WRITEK
CM2     BSR     BINBCD
        JSR     PUTBYT      PRINT BCD VALUE
        JSR     PUTS
        LDX     #MS-CMA
        JSR     PUTMSG      PRINT ms
*
        JSR     GETNYB      GET MS NYBBLE
        BCS     NWER        IF ERROR
        CMP     #$9
        BHI     NWER        BCD  INPUT !
        STA     GET         SAVE MSD
*
        JSR     GETNYB      GET LS NYBBLE
        BCS     NWER        IF ERROR
        CMP     #$9
        BHI     NWER        BCD INPUT !
        BSR     BCDBIN      SAVE IN BINARY
        TSTA                CMP #$00
        BEQ     NWER        MIN IS 1 ms
        BRSET   7,FLAG,CM3 WRI K
        STA     ERASEK
```

```
        DEC     FLAG
        BRA     CML
CM3     STA     WRITEK
CM4     JMP     MAIN

 WER    CMP     #CR         CR ?
        BNE     CMD
        BRCLR   7,FLAG,CMX
        BRA     CM4
 MD     JMP     BLEEP


  U T I L I T I E S


 CDBIN  CLRX                POINT TO CONV TABLE
 CBIL   LSR     GET
        BCC     BCBIL1
        ADD     BBTBL,X
 CBIL1  INX
        CPX     #$4
        BNE     BCBIL
        RTS     A HAS BCD NB


 INBCD  CLR     COUNT       FUTURE BCD
        STA     CHAR        SAVE ENTRY PARAMETER
 BCLOP  LDA     CHAR
        BEQ     DONEB
        DEC     CHAR
        INC     COUNT

        LDA     COUNT       OVF  9 => A
        AND     #$0F
        CMP     #$0A
        BNE     BBC1        IF NOT
        LDA     COUNT
        ADD     #$06        ADJUST
        STA     COUNT

 BC1    BRA     BBCLOP
 ONEB   LDA     COUNT       GET BCD VALUE
        RTS


  PICK --- GET BYTE FROM ANYWHERE IN MEMORY
           THIS IS A HORRIBLE ROUTINE (NOT MERELY
           SELF MODIYING, BUT ALSO SELF CREATING)

           GET+1 HI & LO POINT TO ADDRESS TO BE READ.
           BYTE IS RETURNED IN A
           X IS UNCHANGED AT EXIT
```

```
PICK    STX     XTEMP       SAVE X
        LDX     #$D6        $D6 = LDA 2-BYTE INDEXED
        BRA     COMMON
*
* DROP --- PUT BYTE TO ANY MEMORY LOCATION
*           HAS THE SAME UNDESIRABLE PROPERTIES
*           AS PICK
*
*           A HAS BYTE TO STORE AND GET+1 HI & LO POINTS
*           TO LOCATION TO STORE, A AND X ARE
*           UNCHANGED AT EXIT
*
*           THE FLOW IS DIFFERENT WHETHER FOR RAM OR EEPROM
*
DROP    BSR     TSTEE       WITHIN EEPROM ?
        BCC     DROP1       NOT EEPROM
        JSR     BYTER       ERASE BYTE
        JSR     BYTEW       WRITE BYTE
        RTS
DROP1   STX     XTEMP       SAVE X
        LDX     #$D7        $D7 = STA 2-BYTE INDEXED
*
*
COMMON  STX     GET         PUT OPCODE IN PLACE
        LDX     #$81        $81 = RTS
        STX     GET+3       NOW THE RETURN
        CLRX                WE WANT ZERO OFFSET
        JSR     GET         EXECUTE THIS MESS
        LDX     XTEMP       RESTORE X
        RTS                 AND EXIT
*
* TSTEE -- TEST THE ADDRESS IN GET+1 HI & LO AND RETURN
*           WITH CARRY SET IF IT IS A VALID EEPROM1
*           ADDRESS, AND CLEARED OTHERWISE.
*
TSTEE   LDX     GET+1+HI
        CPX     #01         TEST HI BYTE
        BNE     NOEE        NOT EEPROM1
        SEC
        RTS
NOEE    CLC
        RTS
*
*
* BUMP --- ADD ONE TO CURRENT MEMORY POINTER
*           A AND X UNCHANGED
*
BUMP    INC     GET+1+LO INCREMENT LOW BYTE
        BNE     BUMP2    NON-ZERO MEANS NO CARRY
        INC     GET+1+HI INCREMENT HIGH NYBBLE
BUMP2   RTS
*
* OUT4HS - PRINT BYTE POINTED TO AS AN ADDRESS AND
*           BUMP POINTER - X IS UNCHANGED AT EXIT
*
```

```
      BSR     PICK        GET NEXT BYTE
      BSR     PUTBYT      AND PRINT IT
      BSR     BUMP        GO TO NEXT ADDRESS
*     FALL INTO OUT2HS
*
 OUT2HS - PRINT BYTE POINTED TO, THEN A SPACE,
          BUMP POINTER. X IS UNCHANGED AT EXIT

UT2HS BSR     PICK        GET THE BYTE
      BSR     PUTBYT
      BSR     BUMP        GO TO NEXT
      BSR     PUTS        FINISH UP WITH A SPACE
      RTS

 DISADD - PRINT ONE BYTE ADDRESS IN MEMADD

ISADD LDA     MEMADD      GET ADDRESS
      STA     GET+1+LO    SET UP TO PRINT
      BSR     PRADD1      PRINT ADDRESS (PAGE 0)
      RTS

 PRADD -- PRINT CURRENT ADDRESS FROM GET+1 HI & LO

RADD  LDA     GET+1+HI    PRINT CURRENT LOCATION
      AND     #$1F        max $1FFF
      STA     GET+1+HI    CONVENIENTLY RESTORE 1X
      BSR     PUTBYT
RADD1 LDA     GET+1+LO
      BSR     PUTBYT
      BSR     PUTS        THEN A SPACE
      RTS

 PRDAT -- PRINT DATA POINTED TO BY GET+1 HI & LO

RDAT  BSR     PICK        GET THAT BYTE
      BSR     PUTBYT      PRINT IT
      BSR     PUTS        ANOTHER SPACE
      RTS

 PUTBYT - PRINT £A| IN HEX - A AND X UNCHANGED

UTBYT STA     GET         SAVE A
      LSRA
      LSRA
      LSRA
      LSRA                SHIFT HIGH NYBBLE DOWN
      BSR     PUTNYB      PRINT IT
TSN   LDA     GET
      BSR     PUTNYB      PRINT LOW NYBBLE
      RTS

 PUTNYB - PRINT LOWER NYBBLE OF A IN HEX
          A AND X UNCHANGED
          HIGH NYBBLE OF A IGNORED
```

```
PUTNYB STA     GET+3     SAVE A IN YET ANOTHER TEMP
       AND     #$0F      MASK OFF HIGH NYBBLE
       ADD     #'0       ADD ASCII ZERO
       CMP     #'9       CHECK FOR A-F
       BLS     PUTNY2
       ADD     #'A-'9-1  ADJUSTMENT FOR HEX A-F
PUTNY2 JSR     PUTC
       LDA     GET+3     RESTORE A
       RTS
*
* CRLF --- PRINT CARRIAGE RETURN - LINE FEED
*          A AND X UNCHANGED
*
CRLF   STA     GET       SAVE A
       LDA     #CR
       JSR     PUTC
       LDA     #LF
       JSR     PUTC
       LDA     GET       RESTORE A
       RTS
*
* PUTS --- PRINT A SPACE - A AND X UNCHANGED
*
PUTS   STA     GET       SAVE A
       LDA     #SPACE
       JSR     PUTC
       LDA     GET       RESTORE A
       RTS
*
* GETBYT - GET A HEX BYTE FROM TERMINAL
*
*          A GETS THE BYTE TYPED IF IT WAS A VALID HEX
*          NUMBER, OTHERWISE A GETS THE LAST CHAR TYPED.
*          THE C-BIT IS SET ON NON HEX CHARS, CLEARED
*          OTHERWISE.  X IS UNCHANGED IN ANY CASE.
*
GETBYT BSR     GETNYB    BUILD BYTE FROM 2 NYBBLES
       BCS     NOBYT     NON HEX CHAR
       ASLA
       ASLA
       ASLA
       ASLA              SHIFT NYBBLE TO HIGH NYBBLE
       STA     GET       SAVE IT
       BSR     GETNYB    GET LOW NYBBLE NOW
       BCS     NOBYT     NON HEX CHAR
       ADD     GET       C-BIT CLEARED
NOBYT  RTS
*
* GETNYB - GET HEX NYBBLE FROM TERMINAL
*
*          A GETS THE NYBBLE TYPED IF IT WAS IN THE RANGE 0-F,
*          OTHERWISE A GETS THE CHARACTER TYPED. THE C-BIT IS
*          SET ON NON HEX CHARACTERS, CLEARED OTHERWISE.
*          X IS UNCHANGED
*
```

```
B BSR     GETC      GET THE CHARACTER
        STA     GET+3     SAVE IT JUST IN CASE
        SUB     #'0       SUBTRACT ASCII ZERO
        BMI     NOTHEX    WAS LESS THAN '0'
        CMP     #9
        BLS     GOTIT
        SUB     #'A-'9-1  FUNNY ADJUSTMENT
        CMP     #$F       TOO BIG ?
        BHI     NOTHEX    WAS GREATER THAN 'F'
        CMP     #9        CHECK BETWEEN ASCII 9 AND A
        BLS     NOTHEX
GOTIT   CLC               C=0 MEANS GOOD HEX CHAR
        RTS
NOTHEX  LDA     GET+3     GET SAVED CHAR
        SEC
        RTS               RETURN WITH 'ERROR'
*
* ADJDEL - DELAY FOR EEPROM ROUTINES = TO £A| ms
*
ADJ10   LDA     #10
ADJDEL  LDX     #83       CONSTANT
AL1     BRCLR   4,ADSTCT,*+3 DUMMY
        BRCLR   4,ADSTCT,*+3 DUMMY
        BRCLR   4,ADSTCT,*+3 DUMMY
        BRN     *          DITTO
        DECX
        BNE     AL1
        DECA
        BNE     ADJDEL    LOOP  A  TIMES
        RTS
*
*
* PUTMSG - PRINT THE MESSAGE POINTED TO BY X
*
PUTMSG  LDA     CMA,X     GET NEXT CHARACTER
        CMP     #EOT
        BEQ     NDMSG
        BSR     PUTC      SEND CHAR
        INX
        BRA     PUTMSG
NDMSG   RTS
*
*
*
*          S E R I A L    I / O    R O U T I N E S
*
*
*  Initialise the SCI
*
SCINIT     BCLR    MBIT,SCCR1          8 data bits
           LDA     #%11000000          baud rate 9600
           STA     BAUD
           LDA     #%00001100          TE / RE
           STA     SCCR2               end of init
           STA     SCSR                clear TDRE & TC bits
```

```
            RTS
*
*
* GETC :  Routine GETC services the SCI, it does that by polling
*         the RDRF (received data ready flag). It returns with
*         the byte of data in ACCA.
*
*
GETC        BRCLR   .RDRF,SCSR,*        Possibly wait for char
GDATA       LDA     SCDAT               get data & clear RDRF
            CMP     #'/                 NEXT CHAR ASCII ?
            BNE     CHARS
            DEC     ASC                 FLAG IT
            RTS
CHARS       BRSET   7,ASC,SCHAR
            CMP     #$40
            BLS     NOCHAR
            AND     #%1011111           UPPER CASE
SCHAR       CLR     ASC
NOCHAR      RTS
*
*
* PUTC :  Routine PUTC services the SCI. It polls the TDRE
*         (Tramsmit Data Register Empty), and puts the char
*         when true.
*
PUTC        BRCLR   TDRE,SCSR,*         WAIT
            STA     SCDAT
            RTS
*
*
* ================ ENTRY ======================================
*
* MONIT - ENTRY POINT FROM RESET
*
MONIT   LDA     #10         10 ms BY DEFAULT
        STA     ERASEK      SET ERASE DEFAULT TIME
        STA     WRITEK      SET WRITE DEFAULT TIME
        JSR     SCINIT      INIT SCI
        LDA     #ADDATA
        STA     MEMADD      DISPLAY ADR BY DEFAULT
        JSR     CRLF        START A BRAND NEW LINE
        CLRX                POINT TO START OF MESSAGE
BABBLE  LDA     MSG,X       GET NEXT CHARACTER
        BRCLR   4,PORTD,BAB1 ROM MESSAGE
        LDA     EEPROM+1,X GET NEXT CHAR (EEPROM1 MESSAGE)
BAB1    CMP     #EOT
        BEQ     BABND       IF END OF MESSAGE
        BSR     PUTC        PRINT IT
        INCX                POINT TO NEXT CHAR
        BNE     BABBLE      MORE !
BABND   JSR     CRLF        SEPARATE MESSAGE FROM COMMANDS
        SWI                 GO TO MONITOR ROUTINES
        BRA     MONIT       LOOP AROUND
*
```

```
      ROM1 BURN IN TEST ROUTINE.
* SET UP REQUIRED NB OF ITERATION IN $70:$71
* AND DATA TO BE PROGRAMMED IN $72.
* NOTE : MAXIMUM NB OF ITERATION IS $7FFF.
*
*
ABCNT EQU   $70
ABDAT EQU   $72
*
ABCD    JSR     CRLF
ABL     LDA     ABDAT
        JSR     BULKE
        JSR     BULKW
        LDA     ABCNT+LO
        DECA
        STA     ABCNT+LO
        CMP     #$FF
        BNE     NOBURO
        DEC     ABCNT+HI
        BMI     NDAB
NOBURO  LDA     ABCNT+HI
        JSR     PUTBYT
        LDA     ABCNT+LO
        JSR     PUTBYT
        JSR     CRLF
        BRA     ABL
NDAB    SWI
*
* ============================================================================
*
*         VECTORS
*
*          The unused vectors point to RAM, so as to be available
*          for test purposes (RAM Bootloader, SCI loader). Their
*          positionning allows 10 bytes for the stack, that is 2
*          interrupt levels, or 1 interrupt and 2 subroutine levels.
*
           FDB     STACK-9-18          SCI
           FDB     STACK-9-15          TIM OVF
           FDB     STACK-9-12          TIM OUT COMP
           FDB     STACK-9-9           TIM IN CAP
           FDB     STACK-9-6           IRQ
           FDB     MAIN                SWI
           FDB     MONIT               RESET
*
* ============================================================================
*
*               E N D
*
*****************************************************************
              END
```

This page intentionally left blank.

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