The Summary of the MC68HC05L9 Microcomputer Applications Demo Board

Prepared by : Steve Ho, System Design/Applications Engineer
Motorola Semiconductors Hong Kong Ltd.

INTRODUCTION

The MC68HC05L9 HCMOS microcomputer is a member of Motorola's MC68HC05 family of low-cost single-chip microprocessors, which is particular suitable for Handheld applications with LC Display. This 8-bit microcomputer unit (MCU) contains useful peripherials such as two on-chip oscillators, serial interface system, liquid crystal display driver circuitry, real time clock, alarm, low voltage inhibit, auto display off, keyboard interrupt, external parallel address, data bus, and a tone generator etc. For particular description on the architecture and the hardware applications, please refers to article "An Application Specific MCU for Handheld Data Bank and Terminals" of IEEE Transaction on Consumer Electronic and "Data Sheet of MC68HC05L9".

This application note emphasizes in the software applications of L9 and it intends to serve as a general guidance to users on L9 software development in their specific application. Detailed description and usage of the key features of L9, such as LCD driver, real time clock ... etc can be found on the software routine. This application note also explains how to increase the LCD driving capability.

User can use this application demo board or make some modifications according to their real applications, to develop their own software and hardware if there is no EVM available.

There are three different programs that run on the MC68HC05L9 demo board. They are monitor, keyscan and real time clock programs. A description of the facilities available via the monitor software, a diagram of the demo board circuit, a listing of monitor code, keyscan code and real time clock code are also contain in this application note.

Individual program is executed by first configuring PC5-PC7 as follows before power on or external reset.

		PC5	PC6	PC7
Monitor program	:	1	0	0
Keyscan program	:	0	1	0
RTC program	:	0	0	1

MONITOR PROGRAM

A monitor program is available for MC68HC05L9, which when used with a monitor circuit, a power supply, and a video terminal will allow the user to write and debug user written program.

HARDWARE The monitor circuit requires +5V, +12V and -12V power supply, and all communications between the device and terminal take place via an RS-232 link.

Terminal setup is as follows:

9600 Baud, Full Duplex,
8 bit data and no parity.

A circuit diagram of the demo board is given in Figure 1. Demo board consists of one MC68HC05L9 and one MC68HC68L9, this allows a total display of 1520 LCD pixels. Slave LCD driver being used in the demo board is configured as slave 1 ($280-$32F). For detail description of how to increase the LCD driving capability, refers to the topic "Increasing the LCD driving capability of MC68HC05L9" in this application note.

Address $2000-$3FFF is allocated for demo board program while address $4000-$5FFF is allocated for 8 K SRAM. This configuration implies user create program should be downloaded to location $4000-$5FFF when running monitor program. Since PB6 (port B) is used by a program which resides within the MCU and is pulled high externally, PB6 should configure as input and pull high all the time.

MONITOR OPERATION When the MC68HC05L9 begins to execute the monitor program, the following message should appear on the monitor:

"HI!, I AM MC68HC05L9 FROM THE MOTOROLA HONG KONG IC DESIGN CENTRE"
"FOR HELP, TYPE H"

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:

R	Display of registers: HINZC AA XX PPPP
A	Display and change ACCA.
X	Display and change INDEX.
W	Write / examine memory Type **W** AAAA to begin, then type: = -- to re-examine current ^ -- to examine previous CR -- to examine next DD -- new data
C	Continue execution
G	Execute from address. Format is **G** AAAA. AAAA is any valid memory address.
S	Display machine state. All important I/O registers are displayed.
D	Dump 256 bytes of memory out of to the terminal. Format is **D** AAAA. AAAA is any valid memory address.
M	Fill memory between start and end address with data. Does a byte erase first. Type: **M** SSSS EEEE DD
L	Download S-record file into memory.

BREAKPOINTS Limited breakpoint capability is provided by the monitor. A SWI encountered in a user program will transfer control to the monitor, at the command level. The status of the CPU can be examined by mean of the **R** command, and/or the **D** command. Execution can then be resumed by replacing the SWI by a NOP, or any other instruction, and executing a **C** (Continue) command.

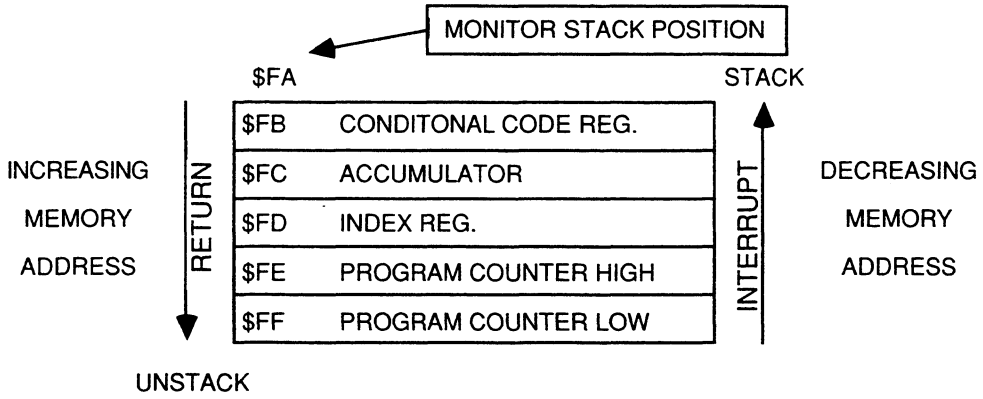
INTERRUPTS For evaluation purposes, most interrupts have vectors pointing into RAM, according to the following table.

Vector targets in RAM:

*	LVI int.	$BE
*	RTC int.	$C1
*	SCI int.	$C4
*	Timer int.	$C7
*	Keyboard int.	$CA
*	External int.	$CD

The SWI is used by the monitor and is therefore not available to the user. Each vectors is allocated 3 bytes of RAM, so that a JMP or BRA instruction can be placed in RAM pointing to the proper service routine.

Note: For commands **R, A, X, C** and **G,** it is assumed that the stack is as follows:



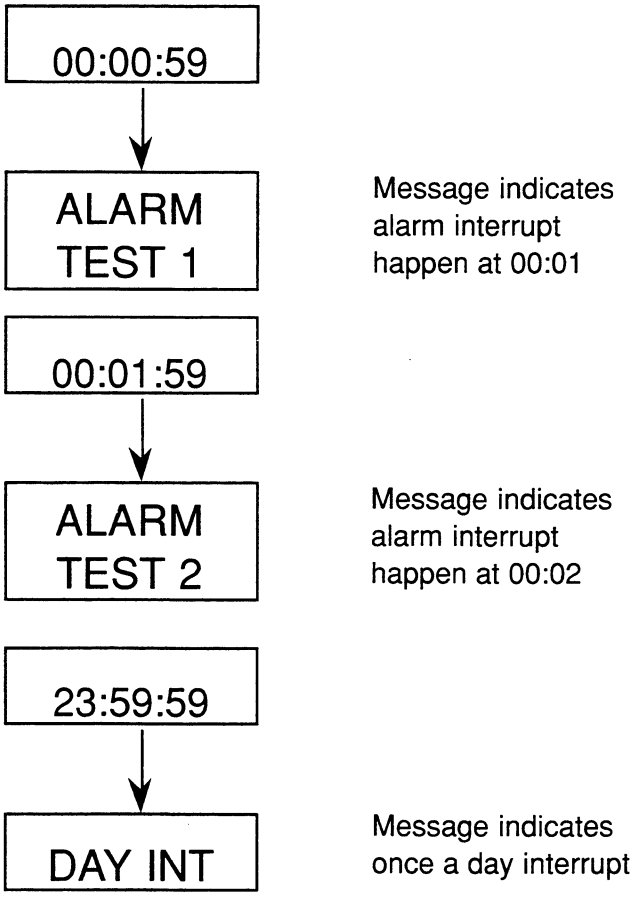
RAM locations $50 to $66 are reserved by the monitor and are not available to the user during monitor operation; Refer to monitor listing for detail description of the monitor program.

KEYSCAN PROGRAM

This program allows for scanning a 5 x 4 keypads; however, more complex routines can be developed from this. This program uses port A which has been implemented with keyboard interrupt feature and port B to build a 5 x 4 keypad (Figure 1). This program demonstrates how to implement dual function keypad and how to use the built in auto display off feature and STOP mode to minimize power consumption. Since DON (bit 4 of $0A) synchronizes with internal LCD clock, a maximum delay of 1 milli second for 1:16 multiplex, while maximum delay is 2 milli second for 1:8 multiplex is required after DON is set in order to turn on the LCD. This program also contains subroutine to transfer the display pattern from character ROM to a LCD panel. Refer to "Keyscan Program" listing for detailed description of the program.

REAL TIME CLOCK PROGRAM

This program implements a clock on a LCD panel using the real time clock once a second interrupt. This program use the same 5 x 4 keypads for the keyscan program except it only recognizes numeric keys. Alarm messages will display on the LCD panel at 00:01 and 00:02 hour to demonstrate the use of real time clock alarm interrupt. Whenever real time clock changes from 23:59:59 to 00:00:00, a message will display on the LCD panel which demonstrates the real time clock once a day interrupt; however, more complex routines can be developed from this.



Real time clock can be updated and the updating sequence is least significant digit of either hour, minute or second first. If the value after two digits have been entered is greater than what should expect (ie. hour should not greater than 24, minute and second should not greater than 60), user need to re-enter a new value for the real time clock. Refer to "Real Time Clock" listing for detailed description of the program.

The accuracy of the real time clock depends on the 32 KHz crystal frequency. Trimming the crystal can be done by first selecting PC0 (port C) as 2.048 KHz clock (ie. setting bit 5 of $0A), then trims the capacitor which is connected between OSC2 and ground until PC0 signal reaches 2.048 KHz.

INCREASING THE LCD DRIVING CAPABILITY OF THE MC68HC05L9 MICROCOMPUTER

INTRODUCTION The MC68HC68L9 is a LCD driver HCMOS chip. This HCMOS chip consists of 176 x 5 display RAM and is used with MC68HC05L9. For detailed description of the hardware feature of MC68HC68L9, please refers to "Data Sheet of MC68HC05L9".

The LCD driving capability of MC68HC05L9 is 16 characters in 5 x 8 dot matrix. Further expansion is possible by direct interfacing with three 80 pins LCD driver (MC68HC68L9) to achieve total 82 characters or 3280 pixels. This application note describes how to use master MCU (MC68HC05L9) and slave LCD driver (MC68HC68L9) together to expand the LCD driving capability.

DISPLAY RAM Address $200-$48F are reserved for display RAM. Master MCU (MC68HC05L9) consists of 128 x 5 bits of display RAM and slave LCD driver (MC68HC68L9) consists of 176 x 5 bits of display RAM. Address $200-$27F are reserved for master MCU and address $280-$48F are reserved for slave LCD drivers.

MASTER AND SLAVE CONFIGURATION Several pins in master MCU can be selected to output synchronous signal (Figure 2) to slave LCD drivers, they are PC0-PC4 of ports C. Before these pins are configured to output synchronous signal to slave LCD driver, bit 5 of address $0A of master MCU has to be set to one. After this bit is set, a delay of maximum one milli second is required for 1:16 multiplex before PC0-PC4 can communicate with slave LCD driver, and a delay of maximum of two milli second is required for 1:8 multiplex. Pins configuration PC0-PC4 of MC68HC05L9 will be as follows after LCD bit is set:

PC0->BPCLK: 2.048 KHz clock (1:16 multiplex) or 1.024 KHz (1:8 multiplex).
PC1->CNTRS: Periodic reset clock.
PC2->CS1 : Select display RAM address $280-$32F(slave 1).
PC3->CS2 : Select display RAM address $330-$3DF(slave 2).
PC4->CS3 : Select display RAM address $3E0-$48F(slave 3).

Slave LCD driver is used the same as static RAM and it is 5 bits a word. The display RAM is arranged in a "What You Store Is What You See" (WYSIWYS) scheme to greatly simplify software programming. Figure 3 shows how master MCU and slave LCD driver are connected, and Figure 4 shows the relation between display RAM Data and display.

Using bank switch technique, more than three slaves LCD driver can be connected to master MC68HC05L9 (Figure 5). Since 1:16 multiplex is the highest backplane ratio L9 can be selected, slaves connected to L9 can only increase the numbers of segment drive. And this implies LCD panel need to be built horizontally instead of vertically. By considering the way LCD panel is built, it is recommended that maximum six slaves should be connected to master MC68HC05L9.

```
*****************************************************************************
*                                                                           *
*       M O N I T O R PROGRAM                                               *
*                                                                           *
*****************************************************************************
*     THE MONITOR HAS THE FOLLOWING COMMANDS:
*
*     R --    PRINT REGISTERS.
*             FORMAT IS CCCCC AA XX PPPP
*
*     A --    DISPLAY AND CHANGE ACCA
*
*     X --    DISPLAY AND CHANGE INDEX
*
*     W --    WRITE/EXAMINE MEMORY (RAM).
*             TYPE W AAAA TO BEGIN,
*             THEN TYPE: =    --  TO RE-EXAMINE CURRENT
*                        ^    --  TO EXAMINE PREVIOUS
*                        CR   --  TO EXAMINE NEXT
*                        DD   --  NEW DATA
*             ANYTHING ELSE EXITS PROGRAM COMMAND.
*
*     C --    CONTINUE EXECUTION
*
*     G --    EXECUTE FROM ADDRESS.  FORMAT IS
*             G AAAA.  AAAA IS ANY VALID MEMORY ADDRESS.
*
*     S --    DISPLAY MACHINE STATE.  ALL IMPORTANT I/O
*             REGISTERS ARE DISPLAYED.
*
*     D --    XFER (DUMP) 256 BYTES OF MEMORY OUT OF TO THE TERMINAL
*
*     M --    FILL MEMORY BETWEEN START AND END ADDRESS WITH DATA.
*             DOES A BYTE ERASE FIRST
*             TYPE : M SSSS EEEE DD
*
*     L --    DOWNLOAD S-RECORD FILE INTO MEMORY.
*
*****************************************************************************
```

```
*******************************************************************************
*
*           SPECIAL EQUATES
*
*
CR          EQU     $0D         CARRIAGE RETURN
PROMPT      EQU     '>          PROMPT CHARACTER
FWD         EQU     CR          GO TO NEXT BYTE
BACKB       EQU     '^          GO TO PREVIOUS BYTE
SAME        EQU     '=          RE-EXAMINE SAME BYTE
*
*           CHARACTER CONSTANTS
LF          EQU     $0A         LINE FEED
BL          EQU     $20         BLANK
EOS         EQU     $00         END OF STRING
*
INITSP      EQU     $FF         INITIAL STACK POINTER VALUE
STACK       EQU     INITSP-5    TOP OF STACK
*
*           RAM VARIABLES
*
*
            ORG     $50
GET         RMB     5           4-BYTE NO-MANS LAND, SEE PICK AND DROP
*                               SUBROUTINES
XTEMP       RMB     1           X REG. TEMP FOR GETC,PUTC
DTEMP       RMB     3           START OF MODIFIABLE CODE (SEE DUMP)
DTEMPE      RMB     1           END OF MODIFIABLE CODE
TEMP        RMB     1
TEMP1       RMB     1
TEMP2       RMB     1
TOTAL       RMB     2           2 BYTE NUMBER/LAST PARAMETER
PTR         RMB     1
CMDPT       RMB     1           COMMAND NAME POINTER
ECHO        RMB     1           INPUT ECHO FLAG
*
SSBP        RMB     4           SSBP ADDRESSES
*
NUMBYT      RMB     1           NUMBER OF BYTES IN DIS-ASSEMBLY
*
```

```
*
*           I/0 REGISTER ADDRESSES
*
PORTA       EQU     $000        I/O PORT 0
PORTB       EQU     $001        I/O PORT 1
PORTC       EQU     $002        I/O PORT 2
DDR         EQU     4           DATA DIRECTION REGISTER OFFSET
*
*           EQUATES NEEDED FOR SCI
*
BAUD        EQU     $0D         SCI baud rate register
SCCR1       EQU     $0E         SCI control register 1
SCCR2       EQU     $0F         SCI control register 2
SCSR        EQU     $10         SCI status register
SCIDAT      EQU     $11         SCI data register
CHKSUM      EQU     PTR         Checksum byte
*
*           CONTROL REGISTERS
RTCF        EQU     $1C
CTRL1       EQU     $09         CONTROL REGISTER 1
CTRL2       EQU     $0A         CONTROL REGISTER 2
*******************************************************************************
            ORG     $800
*
*
*       STATE --- PRINT MACHINE STATE
*
*           A  B  C  D
*           DD DD DD DD
*
*       HEADER STRING FOR I/O REGISTER DISPLAY
*
IOMSG       FCB     CR,LF
            FCC     / A  B  C  D /
            FCB     CR,LF,EOS
*
STATE       CLRX
STATE2      LDA     IOMSG,X     GET NEXT CHAR
            CMP     #EOS        QUIT?
            BEQ     STATE3      YES, NOW PRINT VALUES
```

```
            JSR     PUTC        NO, PRINT CHAR
            INCX                BUMP POINTER
            BRA     STATE2      DO IT AGAIN
*
*       NOW PRINT VALUES UNDERNEATH THE HEADER
*
STATE3      CLRX
PIO         LDA     ,X          START WITH I/O PORTS
            JSR     PUTBYT
            JSR     PUTS
            INCX
            CPX     #4          END OF I/O?
            BNE     PIO         NO, DO MORE
*
            JSR     PUTS
            BRA     MONIT       ALL DONE
*
*       PCC --- PRINT CONDITION CODES
*
*       STRING FOR PCC SUBROUTINE
*
CCSTR       FCC     /HINZC/
*
PCC         LDA     STACK+1     CONDITION CODES IN ACCA
            ASLA                MOVE H BIT TO BIT 7
            ASLA
            ASLA
            STA     GET         SAVE IT
            CLRX
PCC2        LDA     #.
            ASL     GET         PUT BIT IN C
            BCC     PCC3        BIT OFF MEANS PRINT .
            LDA     CCSTR,X     PICKUP APPROPRIATE CHARACTER
PCC3        JSR     PUTC        PRINT . OR CHARACTER
            INCX                POINT TO NEXT IN STRING
            CPX     #5          QUIT AFTER PRINTING ALL 5 BITS
            BLO     PCC2
            RTS
*
*       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     MONIT1      ERROR, NO CHANGE
        STA     ,X          ELSE REPLACE WITH THE NEW VALUE
        BRA     MONIT1      NOW RETURN
*
*       REGS --- PRINT CPU REGISTERS
*
REGS    JSR     PCC         PRINT CC REGISTER
        JSR     PUTS        SEPARATE FROM NEXT STUFF
        CLR     GET+1       POINT TO PAGE ZERO,
        LDA     #STACK+2
        STA     GET+2
        JSR     OUT2HS      CONTINUE PRINT WITH A
        JSR     OUT2HS      X AND FINALLY THE
        JSR     OUT4HS      PROGRAM COUNTER
        BRA     MONIT
*
*       FALL INTO MAIN LOOP
*
*       MONIT --- PRINT PROMPT AND DECODE COMMANDS
*
MONIT1  EQU     *
MONIT   JSR     CRLF        GO TO NEXT LINE
        LDA     #PROMPT
        JSR     PUTC        PRINT THE PROMPT
        JSR     GETC        GET THE COMMAND CHARACTER
        AND     #%1111111 MASK PARITY
```

```
            JSR     PUTS    PRINT SPACE (WON'T DESTROY A)
            CMP     #'R     REGISTERS
            BEQ     REGS
            CMP     #'A
            BEQ     SETA    CHANGE A
            CMP     #'X
            BEQ     SETX    CHANGE X
            CMP     #'C
            BEQ     CONT    CONTINUE EXECUTION
            CMP     #'G     EXECUTE
            BEQ     EXEC
            CMP     #'M     MEMORY
            BEQ     MEMORY
            CMP     #'H     HELP INFORMATION
            BEQ     INFJMP
            CMP     #'S     DISPLAY MACHINE STATE
            BNE     MONIT2
            JMP     STATE   COMMANDS ARE GETTING TOO FAR AWAY
*
MONIT2      EQU     *
            CMP     #'D     DUMP MEMORY
            BEQ     XFR
            CMP     #'L     DOWNLOAD
            BEQ     DLOAD
            CMP     #'W     WRITE/EXAMINE RAM
            BEQ     PROGIT
INFJMP      JMP     INFO    Else display menu if none of above
XFR         JMP     DMEM
MEMORY      JMP     FILMEM
DLOAD       JMP     DOWN
*
*       EXEC --- EXECUTE FROM GIVEN ADDRESS
*
EXEC        JSR     GETBYT  GET HIGH NYBBLE
            BCS     MONIT   BAD DIGIT
            TAX             SAVE FOR A SECOND
            JSR     GETBYT  NOW THE LOW BYTE
            BCS     MONIT   BAD ADDRESS
            STA     STACK+5 PROGRAM COUNTER LOW
            STX     STACK+4 PROGRAM COUNTER HIGH
```

```
*
*         CONT --- CONTINUE USER PROGRAM
*
CONT      RTI
*
MONIT3    JMP       MONIT
*
*         WRITE/EXAMINE RAM
*
PROGIT    JSR       GETBYT    BUILD ADDRESS
BADONE    BCS       MONIT3    BAD HEX CHARACTER
          STA       GET+1
          JSR       GETBYT
          BCS       BADONE    BAD HEX CHARACTER
          STA       GET+2     ADDRESS IS NOW IN GET+1&2
MEM2      JSR       CRLF      BEGIN NEW LINE
          LDA       GET+1     PRINT CURRENT LOCATION
          JSR       PUTBYT
          LDA       GET+2
          JSR       PUTBYT
          JSR       PUTS      A BLANK, THEN
          JSR       PICK      GET THAT BYTE
          JSR       PUTBYT    AND PRINT IT
          JSR       PUTS      ANOTHER BLANK,
          JSR       GETBYT    TRY TO GET A BYTE
          BCS       MEM3      MIGHT BE A SPECIAL CHARACTER
          JSR       DROP      OTHERWISE, PUT IT AND CONTINUE
MEM4      JSR       BUMP      GO TO 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       #BACKB    GO BACK ONE BYTE?
          BNE       XMONIT    NO, EXIT MEMORY COMMAND
          DEC       GET+2     DECREMENT LOW BYTE
          LDA       GET+2     CHECK FOR UNDERFLOW
          CMP       #$FF
          BNE       MEM2      NO UNDERFLOW
          DEC       GET+1
```

```
        BRA     MEM2
INFO    CLRX
INFO1   LDA     DUMP,X      GET CHARACTER
        BEQ     XMONIT      IF 00,RETURN
        JSR     PUTC        OUTPUT CHAR
        INCX
        BEQ     INFO2       DO NEXT PAGE
        JMP     INFO1
INFO2   LDA     DUMP+$100,X  GET CHARACTER
        BEQ     XMONIT      IF 00,RETURN
        JSR     PUTC        OUTPUT CHAR
        INCX
        JMP     INFO2
*
*       CONVENIENT TRANSFER POINT BACK TO MONIT
*
XMONIT  JMP     MONIT       RETURN TO MONIT
*
DMEM    LDA     #$D6        OPCODE FOR LDA OFFSET,X
        STA     DTEMP       SET UP MODIFIABLE CODE
        LDA     #$81        OPCODE FOR RTS
        STA     DTEMPE
DMEM2   JSR     GETBYT      GET HIGH BYTE
        BCS     XMONIT      BAD DIGIT
        STA     DTEMP+1     SET UP HIGH ADDRESS
        STA     TEMP1       SAVE FOR LATER
        JSR     GETBYT      NOW THE LOW BYTE
        BCS     XMONIT      BAD ADDRESS
        STA     DTEMP+2     SET UP LOW BYTE
        STA     TEMP        SAVE FOR LATER
        CLRX
DMEMN   JSR     CRLF
        LDA     TEMP1
        JSR     PUTBYT      SHOW ADDRESS
        LDA     TEMP
        JSR     PUTBYT
        ADD     #$10
        STA     TEMP        UPDATE ADRESS
        BNE     LO
        LDA     TEMP1
```

```
            ADD     #$01
            STA     TEMP1
LO          JSR     PUTS
            JSR     PUTS
DMEM5       JSR     DTEMP     EXECUTE MODIFIED CODE
            JSR     PUTBYT    Echo character to terminal
            JSR     PUTS
            INCX
            BEQ     XMONIT    ALL DONE
            TXA
            AND     #$0F      END OF THE LINE
            BNE     DMEM5
            BRA     DMEMN

*       UTILITIES
*
*       PICK ---    GET BYTE FROM ANYWHERE IN MEMORY
*                   THIS IS A HORRIBLE ROUTINE (NOT MERELY
*                   SELF-MODIFYING, BUT SELF-CREATING)
*
*       GET+1&2 POINT TO ADDRESS TO READ,
*       BYTE IS RETURNED IN A
*       X IS UNCHANGED AT EXIT
*
PICK        STX     XTEMP     SAVE X
            LDX     #$D6      D6=LDA 2-BYTE INDEXED
            BRA     COMMON
*
*
*       DROP ---    PUT BYTE TO ANY MEMORY LOCATION.
*                   HAS THE SAME UNDESIRABLE PROPERTIES
*                   AS PICK
*       A HAS BTE TO STORE, AND GET+1&2 POINTS
*       TO LOCATION TO STORE
*       A AND X UNCHANGED AT EXIT
*
*
DROP        STX     XTEMP     SAVE X
            LDX     #$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
*
*       BUMP --- ADD ONE TO CURRENT MEMORY POINTER
*
*       A AND X UNCHANGED
*
BUMP    INC     GET+2   INCREMENT LOW BYTE
        BNE     BUMP2   NON-ZERO MEANS  NO CARRY
        INC     GET+1   INCREMENT HIGH NYBBLE
BUMP2   RTS
*
*
*       OUT4HS --- PRINT WORD POINTED TO AS AN ADDRESS, BUMP
*                  POINTER
*       X IS UNCHANGED AT EXIT
*
OUT4HS  BSR     PICK    GET HIGH NYBBLE
        BSR     PUTBYT  AND PRINT IT
        BSR     BUMP    GO TO NEXT ADDRESS
*
*       OUT2HS --- PRINT BYTE POINTED TO, THEN A SPACE. BUMP POINTER
*       X IS UNCHANGED AT EXIT
*
OUT2HS  BSR     PICK    GET THE BYTE
        BSR     PUTBYT
        BSR     BUMP    GO TO NEXT
        BSR     PUTS    FINISH UP WITH A BLANK
        RTS
*
*       PUTBYT --- PRINT A IN HEX
*       A AND X UNCHANGED
*
PUTBYT  STA     GET     SAVE A
```

```
            LSRA
            LSRA
            LSRA
            LSRA                    SHIFT HIGH NYBBLE DOWN
            BSR     PUTNYB          PRINT IT
            LDA     GET
            BSR     PUTNYB          PRINT LOW NYBBLE
            RTS
*
*       PUTNYB --- PRINT LOWER NYBBLE OF A IN HEX
*       A AND X UNCHANGED, HIGH NYBBLE OF A IS IGNORED.
*
PUTNYB  STA     GET+3           SAVE A IN YET ANOTHER TEMP
        AND     #$F             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
        LDA     #CR
        JSR     PUTC
        LDA     #LF
        BSR     PUTC
        LDA     GET             RESTORE
        RTS
*
*       PUTS --- PRINT A BLANK (SPACE)
*       A AND X UNCHANGED
*
PUTS    STA     GET             SAVE
        LDA     #BL
        BSR     PUTC
        LDA     GET             RESTORE
```

```
            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 CHARACTER TYPED. THE C-BIT IS
*       SET ON NON-HEX CHARACTERS; CLEARED OTHERWISE. X
*       UNCHANGED IN ANY CASE.
*
GETBYT  BSR     GETNYB  BUILD BYTE FROM 2 NYBBLES
        BCS     NOBYT   BAD CHARACTER IN INPUT
        ASLA
        ASLA
        ASLA
        ASLA            SHIFT NYBBLE TO HIGH NYBBLE
        STA     GET     SAVE IT
        BSR     GETNYB  GET LOW NYBBLE NOW
        BCS     NOBYT   BAD CHARACTER
        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.
*
GETNYB  BSR     GETC    GET THE CHARACTER
        AND     #%1111111 MASK PARITY
        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 9 AND A
        BLS     NOTHEX
```

```
GOTIT   CLC             C=0 MEANS GOOD HEX CHAR
        RTS
NOTHEX  LDA     GET+3   GET SAVED CHARACTER
        SEC
        RTS             RETURN WITH ERROR
*
*       S E R I A L   I / O   R O U T I N E S
*
*
*       DEFINITION OF SERIAL I/O LINES
*
*
*       GETC --- GET A CHARACTER FROM THE TERMINAL
*
*       A GETS THE CHARACTER TYPED, X IS UNCHANGED.
*
GETC    BRCLR   5,SCSR,*  WAIT FOR RDRF
        LDA     SCIDAT    Get character & clear RDRF bit
        TST     ECHO      ECHO DATA ?
        BEQ     GETCC
        JSR     PUTC      ECHO BACK
GETCC   AND     #$7F      MASK PARITY
        RTS               AND RETURN
*
*
*       X AND A UNCHANGED
*
PUTC    BRCLR   7,SCSR,*  WAIT FOR TDRE
        STA     SCIDAT    OUTPUT THE CHARACTER
        RTS
*
*******************************************************************************
*
*       RESET --- POWER ON RESET ROUTINE
*
RESET   EQU     *
        BSET    6,RTCF          SELECT PLL CLOCK FOR MCU
        LDA     #%01100000      SELECT 2.4 MHZ CLOCK
        STA     CTRL1
PLLST   BRCLR   6,CTRL2,*  WAIT UNTIL CLOCK IS STABLE
```

```
            LDA     #%00000100    9600 @ PH2 = 2.4 MHz
            STA     BAUD          Set baud rate for SCI
            LDA     #%00000000    Set SCI for 8 Data bits communication
            STA     SCCR1
            LDA     #%00001100    Enable transmitter and receiver
            STA     SCCR2
*
*       PRINT SIGN-ON MESSAGE
*
            LDA     #$FF
            STA     ECHO          ECHO DATA
            CLRX
BABBLE      LDA     MONMSG,X      GET NEXT CHARACTER
            CMP     #EOS          LAST CHAR?
            BEQ     MSTART        YES, START MONITOR
            JSR     PUTC          AND PRINT IT
            INCX                  ADVANCE TO NEXT CHAR
            BRA     BABBLE        MORE MESSAGE
MSTART      SWI
*
            JMP     MONIT1
*
*       MONMSG --- POWER UP MESSAGE
*
MONMSG      FCB     CR,LF,LF
            FCC     /HI! I AM MC68HC05L9 /
            FCC     /FROM THE MOTOROLA /
            FCC     /HONG KONG IC DESIGN CENTRE/
            FCB     CR,LF
            FCC     /FOR HELP ,TYPE H/
            FCB     EOS
DUMP        FCB     $0A,$0D
            FCC     /THE MONITOR HAS THE FOLLOWING COMMANDS:/
            FCB     $0A,$0D
            FCC     /D --  DUMP 256 BYTES OF MEMORY./
            FCB     $0A,$0D
            FCC     /G --  EXECUTE FROM ADDRESS./
            FCB     $0A,$0D
            FCC     /C --  CONTINUE EXECUTION./
            FCB     $0A,$0D
```

```
        FCC     /H -- HELP/
        FCB     $0A,$0D
        FCC     'L -- DOWNLOAD S-RECORD INTO MEMORY.'
        FCB     $0D,$0A
        FCC     *M -- MEMORY FILL WITH DATA.*
        FCB     $0A,$0D
        FCC     *W -- WRITE/EXAMINE RAM.*
        FCB     $0A,$0D
        FCC     /R -- REGISTER DISPLAY./
        FCB     $0A,$0D
        FCC     /A -- DISPLAY AND CHANGE ACCA./
        FCB     $0A,$0D
        FCC     /X -- DISPLAY AND CHANGE INDEX./
        FCB     $0A,$0D
        FCC     /S -- DISPLAY MACHINE STATE./
        FCB     $0A,$0D
        FCB     $00

MONITX  JMP     MONIT       RETURN TO MONITOR
FILMEM  JSR     GETBYT      BUILD START ADDRESS
        BCS     MONITX      BAD HEX CHARACTER
        STA     GET+1
        JSR     GETBYT
        BCS     MONITX      BAD HEX CHARACTER
        STA     GET+2       START ADDRESS IS NOW IN GET+1&2
        JSR     PUTS        PRINT SPACE
        JSR     GETBYT      BUILD END ADDRESS
        BCS     MONITX
        STA     SSBP+1
        JSR     GETBYT
        BCS     MONITX
        STA     SSBP+2      END ADDRES IS NOW IN SSBP+1:SSBP+2
        JSR     PUTS        PRINT SPACE
        JSR     GETBYT      GET DATA BYTE TO BE PUT IN MEMORY
        STA     SSBP
        JSR     CRLF        BEGIN NEW LINE
        LDA     #$C7
        STA     GET         "STA" EXTENDED
        LDA     #$81
        STA     GET+3       "RTS"
```

```
DONE2   LDA     SSBP    GET DATA
        JSR     DROP    WRITE IT
        JSR     BUMP    ADVANCE ADDRESS
        LDA     SSBP+1  GET END ADDRESS MSB
        CMP     GET+1
        BLO     DONE1   REACHED END ADDRESS
        BHI     DONE2   NOT DONE YET
        LDA     SSBP+2
        CMP     GET+2   COMPARE LSB
        BLO     DONE1   DONE.
        BRA     DONE2   DO NEXT BYTE
DONE1   JMP     MONIT
*
*       DOWNLOAD COMMAND
*
DOWN    JSR     CRLF    START NEW LINE ON TERMINAL
        CLR     ECHO    NO ECHO
*
LOADSS  CLR     NUMBYT  RESET END DUMP FLAG
*
*       WAIT FOR S-RECORDS
*
LOADS   JSR     GETC    YES, GET CHAR FROM HOST
        CMP     #'S     IS IT 'S' ?
        BNE     LOADS   NO, KEEP LOOKING
        JSR     GETC    YES, GET NEXT CHAR
        CMP     #'9     IS IT '9' ?
        BEQ     ENDLD   YES, END OF LOAD
        CMP     #'1     NO, IS IT '1' ?
        BNE     LOADS   NO, KEEP LOOKING
        BRA     GETLIN  YES
*
*       PROCESS S-RECORD BYTE COUNT
*
ENDLD   INC     NUMBYT  SET END LOAD FLAG
GETLIN  CLR     CHKSUM  RESET CHECKSUM
        BSR     EVBYT   GET NUMBER OF BYTES IN RECORD
        SUB     #$03    ADJUST FOR CKSUM, ADDR (3 BYTES)
        STA     CMDPT   STORE COUNT
*
```

```
*       GET DATA STORE ADDRESS
*
            BSR     EVBYT     GET LOAD ADDRESS MSB
            STA     GET+1     STORE IT
            BSR     EVBYT     GET LOAD ADDR LSB
            STA     GET+2     STORE NEW LSB
*
*       STORE DATA IN MEMORY
*
NEWBYT      DEC     CMDPT     ALL DATA BYTES INPUT ?
            BMI     GETCKS    YES, GET CHECKSUM
            BSR     EVBYT     NO, GET DATA BYTE
            JSR     DROP      STORE IT AT CURRENT ADDR
            JSR     BUMP      UPDATE ADDRESS
            BRA     NEWBYT    CONTINUE
*
*       VERIFY CHECKSUM
*
GETCKS      LDX     CHKSUM    TEMP SAVE CHECKSUM
            STX     CMDPT
            BSR     EVBYT     GET CHECKSUM FROM FILE
            TST     NUMBYT    END OF LOAD ?
            BNE     NOLOAD    YES, IGNORE CHECKSUM
            COMA              NO, COMPLEMENT IT
            CMP     CMDPT     CALCULATED EQUAL READ (CHECKSUM) ?
            BEQ     LOADS     YES, KEEP GOING
*
NOLOAD      EQU     *
*
ISERR8      EQU     *         NO, ERROR
            COM     ECHO      RE-ENABLE ECHO
            JMP     MONIT     RETURN
*
*       THIS ROUTINE WILL EVALUATE AN INPUT BYTE FROM
*       HOST/CASSETTE
*
EVBYT       JSR     GETBYT    READ DATA BYTE
            STA     TOTAL
            ADD     CHKSUM    ADD TO CHECKSUM
            STA     CHKSUM
```

```
        LDA     TOTAL
        RTS             RETURN
*
*       ORG     $1FF0
        FCB     $00,$BE LVI INTERRUPT
        FCB     $00,$C1 RTC INTERRUPT
        FCB     $00,$C4 SCI INTERRUPT
        FCB     $00,$C7 TIMER INTERRUPT
        FCB     $00,$CA KEYBOARD INTERRUPT
        FCB     $00,$CD EXTERNAL INTERRUPT
        FDB     MONIT1  SOFTWARE INTERRUPT
        FDB     RESET   POWER ON RESET
        END
```

```
*****************************************************************************
*                                                                           *
*                KEYSCAN PROGRAM                                            *
*                                                                           *
*****************************************************************************
*
*THIS  PROGRAM IS FOR THE EVALUATION OF KEYBOARD INTERRUPT,
*AUTO DISPLAY OFF  AND LCD DRIVER CIRCUIT
*IT RUNS AT 2.4 MHZ BUS FREQUENCY WITH PLL OUTPUT
*CLOCK IS SELECTED
*5 x 8 DOT MATRIX LCD IS USED AND 1:16 MULTIPLEX RATIO IS
*SELECTED
*
*****************************************************************************
            ORG     $50
BUF         RMB     16          16 CHARACTERS BUFFER
AKEY        RMB     1           VARIABLE FOR CHARACTERS A, B, C
DKEY        RMB     1           VARIABLE FOR CHARACTERS D, E, F
GKEY        RMB     1           VARIABLE FOR CHARACTERS G, H, I
JKEY        RMB     1           VARIABLE FOR CHARACTERS J, K, L
MKEY        RMB     1           VARIABLE FOR CHARACTERS M, N, O
PKEY        RMB     1           VARIABLE FOR CHARACTERS P, Q, R
SKEY        RMB     1           VARIABLE FOR CHARACTERS S, T, U
VKEY        RMB     1           VARIABLE FOR CHARACTERS V, W, X
YKEY        RMB     1           VARIABLE FOR CHARACTERS Y, Z
KEYA        RMB     1           ROW OF MATRIX KEYBOARD
KEYB        RMB     1           COLUMN OF MATRIX KEYBOARD
KEYFL       RMB     1
KEYCTR      RMB     1
GENFL       RMB     1
TMP         RMB     1
TMP1        RMB     1
TMP2        RMB     1
TMP3        RMB     1
TMP4        RMB     1
TMP5        RMB     1
STACK       RMB     1
CTRL1       EQU     $09         CONTROL REGISTER 1
CTRL2       EQU     $0A         CONTROL REGISTER 2
PORTA       EQU     00
```

```
PORTB   EQU   01
PORTAD  EQU   04
PORTBD  EQU   05
CHGEN   EQU   $1B00          STARTING LOCATION OF DISPLAY  PATTERN
NGEN    EQU   $1C00
DRAM    EQU   $200           DISPLAY RAM STARTING LOCATION


        ORG   $800
*************************************************************************
MAIN    BSR   INIT           INITIALIZE ALL NECESSARY BITS
        BSR   KEYW           PREPARE SCANNING LINES
        CLI                  ENABLE INTERRUPTS
STOP    STOP                 WAIT FOR KEYBOARD INTERRUPT
        BRA   STOP
*
*************************************************************************
*INITIALIZATION ROUTINE
INIT    BSET  6,$1C          SELECT PLL CLOCK FOR MCU
        LDA   #$FF
        STA   PORTAD
        STA   PORTA          CLEAR UNNECESSARY KEYBOARD
*                            INTERRUPT
        LDA   #%01110000     ENABLE KEYBOARD INTERRUPT
        STA   CTRL1          SELECT P02 FREQUENCY AS 2.4 MHZ
PLL     BRCLR 6,CTRL2,PLL    WAIT UNTIL CLOCK IS STABLE
        RTS
*
*************************************************************************
*ROUTINE TO UPDATE DISPLAY BUFFER AND DISPLAY RAM AT THE SAME
*TIME
PUTC    LDX   KEYCTR         LOAD DISPLAY BUFFER POINTER
        STA   BUF,X
        CLR   TMP1           UPDATE DISPLAY RAM
        JSR   BUTODR         UPDATE TOP ROW OF DISPLAY RAM
        LDA   #8
        STA   TMP1
        JSR   BUTODR         UPDATE BOTTOM ROW OF DISPLAY RAM
        RTS
*
```

```
*************************************************************************
*ROUTINE TO PREPARE KEYBOARD INTERRUPT
*PORT A IS INPUT
*PORT B IS OUTPUT
KEYW    CLR     PORTAD
        CLR     PORTA           PREPARE SCANNING LINES
        LDA     #$0F
        STA     PORTBD
        CLR     PORTB
        LDA     #'A-1           INITIALIZE CHARACTER KEYS
        STA     AKEY
        LDA     #'D-1
        STA     DKEY
        LDA     #'G-1
        STA     GKEY
        LDA     #'J-1
        STA     JKEY
        LDA     #'M-1
        STA     MKEY
        LDA     #'P-1
        STA     PKEY
        LDA     #'S-1
        STA     SKEY
        LDA     #'V-1
        STA     VKEY
        LDA     #'Y-1
        STA     YKEY
        CLR     KEYFL
        CLR     KEYCTR          RESET DISPLAY BUFFER POINTER TO ZERO
        JSR     CLRBUF          CLEAR BUFFER
        CLR     TMP1
        JSR     BUTODR
        LDA     #8
        STA     TMP1
        JSR     BUTODR          CLEAR DISPLAY RAM
        BSET    3,CTRL2         ENABLE AUTO DISPLAY OFF FEATURE
        LDA     #$02
        STA     $08             AUTO DISPLAY OFF TWO MINUTES
*                               AFTER EXECUTE STOP INSTRUCTION
        RTS
```

```
*
*******************************************************************************
*KEYSCAN ROUTINE
KEYSCN   BSET    4,CTRL2         TURN ON LCD
KEYSCN1  LDA     #$F7            START CHECKING FROM THE 4TH
*                                COLUMN
         STA     PORTB
REPEAT   LDA     PORTA
         AND     #$1F            START SCANNING FROM THE BOTTOM
*                                ROW
         CMP     #$1F
         BNE     GOTIT           KEY FOUND
         LSR     PORTB           REPEAT IF MORE COLUMN
         BCS     REPEAT
         CLR     KEYFL           PREPARE FOR NEXT SCAN
         CLR     PORTA           RESET KEYBOARD INTERRUPT
         CLR     PORTB
RETURN   RTI

GOTIT    BRSET   0,KEYFL,NOSAVE
         LDA     PORTB           SAVE KEY
         LDX     PORTA
         STA     KEYB
         STX     KEYA
         BSR     DBOUNC          CHECK FOR NOISE
         BSET    0,KEYFL
         BRA     KEYSCN1
NOSAVE   BSR     DBOUNC          PAUSE
         CLR     PORTA           PREPARE SCAN LINE
         CLR     PORTB
         CLR     KEYFL
         BRA     DECODE          GO TO USER KEY DECODE ROUTINE
*
*******************************************************************************
*DEBOUNCE ROUTINE
DBOUNC   LDA     #60
AGAIN1   LDX     #$FF
AGAIN    DECX
         BNE     AGAIN
         DECA
```

```
            BNE     AGAIN1
            RTS
*
*******************************************************************************
*DECODE ROUTINE
DECODE      BSET    4,CTRL2         JUST TO MAKE SURE LCD IS TURNED ON
            BRCLR   0,KEYA,ROW0     FIRST ROW
            BRCLR   1,KEYA,ROW1     SECOND ROW
            BRCLR   2,KEYA,ROW2     THIRD ROW
            BRCLR   3,KEYA,THROW    FORTH ROW
            JMP     ROW4            FIFTH ROW
THROW       JMP     ROW3
ROW0        BRCLR   0,KEYB,RC00     SPACE
            BRCLR   1,KEYB,RC01     0
            BRCLR   2,KEYB,RC02     Y, Z
            BRCLR   3,KEYB,RC03     V, W, X
RC00        JMP     RC23
RC01        LDA     #'0
            JMP     FIN
RC02        JMP     YROW
RC03        JMP     VROW
ROW1        BRCLR   0,KEYB,RC10     7
            BRCLR   1,KEYB,RC11     8
            BRCLR   2,KEYB,RC12     9
            BRCLR   3,KEYB,RC13     S, T, U
RC10        LDA     #'7
            JMP     FIN
RC11        LDA     #'8
            JMP     FIN
RC12        LDA     #'9
            JMP     FIN
RC13        JMP     SROW
ROW2        BRCLR   0,KEYB,RC20     4
            BRCLR   1,KEYB,RC21     5
            BRCLR   2,KEYB,RC22     6
            BRCLR   3,KEYB,PPROW    P, Q, R
PPROW       JMP     PROW
RC20        LDA     #'4
            JMP     FIN
RC21        LDA     #'5
```

```
        JMP     FIN
RC22    LDA     #'6
        JMP     FIN
RC23    INC     KEYCTR
        LDA     KEYCTR
        CMP     #$10
        BNE     CCTR
        CLR     KEYCTR
CCTR    JMP     DONE
ROW3    BRCLR   0,KEYB,RC30     1
        BRCLR   1,KEYB,RC31     2
        BRCLR   2,KEYB,RC32     3
        BRCLR   3,KEYB,RC33     M, N, O
ROW4    BRCLR   0,KEYB,RC40     A, B, C
        BRCLR   1,KEYB,RC41     D, E, F
        BRCLR   2,KEYB,RC42     G, H, I
        BRCLR   3,KEYB,RC43     J, K, L
RC30    LDA     #'1
        BRA     FIN
RC31    LDA     #'2
        BRA     FIN
RC32    LDA     #'3
        BRA     FIN
RC33    INC     MKEY            ADVANCE TO NEXT CHARACTER
        LDA     MKEY
        CMP     #'P
        BNE     FIN
        LDA     #'M
        STA     MKEY
        BRA     FIN
RC40    INC     AKEY
        LDA     AKEY
        CMP     #'D
        BNE     FIN
        LDA     #'A
        STA     AKEY
        BRA     FIN
RC41    INC     DKEY
        LDA     DKEY
        CMP     #'G
```

```
        BNE     FIN
        LDA     #'D
        STA     DKEY
        BRA     FIN
RC42    INC     GKEY
        LDA     GKEY
        CMP     #'J
        BNE     FIN
        LDA     #'G
        STA     GKEY
        BRA     FIN
RC43    INC     JKEY
        LDA     JKEY
        CMP     #'M
        BNE     FIN
        LDA     #'J
        STA     JKEY
FIN     JSR     PUTC            UPDATE DISPLAY BUFFER
DONE    RTI
VROW    INC     VKEY
        LDA     VKEY
        CMP     #'Y
        BNE     FIN
        LDA     #'V
        STA     VKEY
        BRA     FIN
YROW    INC     YKEY
        LDA     YKEY
        CMP     #$5B
        BNE     FIN
        LDA     #'Y
        STA     YKEY
        BRA     FIN
SROW    INC     SKEY
        LDA     SKEY
        CMP     #'V
        BNE     FIN
        LDA     #'S
        STA     SKEY
        BRA     FIN
```

```
PROW    INC     PKEY
        LDA     PKEY
        CMP     #'S
        BNE     FIN
        LDA     #'P
        STA     PKEY
        BRA     FIN
*
*************************************************************************
*ROUTINE TO DISPLAY EIGHT BUF CHARACTERS ON ONE ROW OF DISPLAY *RAM
BUTODR  CLR     TMP4
        LDA     TMP1
        STA     TMP5
ALP0    BSR     NUORAL          GET OFFSET AND SET GENFL
        LDA     #$08            EIGHT ROWS FOR A CHARACTER
        STA     TMP3
ALP1    BRSET   0,GENFL,ALPH
        LDA     NGEN,X          GET NUM PATTERN
        BRA     ALPHF
ALPH    LDA     CHGEN,X         GET ALPH PATTERN
ALPHF   STX     TMP2            STORE PATTERN OFFSET
        LDX     TMP1            LOAD DRAM LOCATION OFFSET
        STA     DRAM,X
        INC     TMP2
        LDX     TMP2            ADVANCE TO NEXT DRAM LOCATION
        INC     TMP1
        DEC     TMP3
        BNE     ALP1            COMPLETE LOADING ONE CHARACTER
        LDA     TMP1
        ADD     #$08            ADVANCE TO NEXT DRAM LOC
        STA     TMP1
        INC     TMP5
        INC     TMP4
        LDA     TMP4
        CMP     #$08
        BNE     ALP0
        RTS
*
*************************************************************************
```

```
*TMP4 HAS THE OFFSET FOR DISPLAY BUFFER
*RETURN WITH GENFL SET
*AND OFFSET IN X
NUORAL  CLR     GENFL           CLEAR GEN FLAG
        LDX     TMP5            LOAD BUF OFFSET
        LDA     BUF,X           GET ASCII
        JSR     AMUL8           GET ROM PATTERN OFFSET
        CMP     #$40
        BHS     SETGEN          NOT A NUMBER
NRALD   RTS
SETGEN  BSET    0,GENFL
        BRA     NRALD
        RTS
*
*************************************************************************
*ROUTINE TO CLEAR DISPLAY BUFFER IE. #$40 -> BUFFER
CLRBUF  LDA     #$40
        CLRX
CLROP   STA     BUF,X
        INCX
        CPX     #$10
        BNE     CLROP
        RTS
*
*************************************************************************
*ASCII TO BIN AND MUL BY 8
*THIS ROUTINE CALCULATES THE STARTING LOCATION OF A DISPLAY
*PATTERN
AMUL8   STA     STACK
        CMP     #$40
        BHS     SUB40
        SUB     #$30
        BRA     JUO40
SUB40   SUB     #$40
JUO40   LSLA
        LSLA
        LSLA
        TAX
        LDA     STACK
        RTS
```

```
*
*******************************************************************************
        ORG     CHGEN
SPA     FCB     $00,$00,$00,$00,$00,$00,$00,$00
CHA     FCB     $04,$0A,$11,$11,$1F,$11,$11,$00
CHB     FCB     $1E,$11,$11,$1E,$11,$11,$1E,$00
CHC     FCB     $0E,$11,$10,$10,$10,$11,$0E,$00
CHD     FCB     $1C,$12,$11,$11,$11,$12,$1C,$00
CHE     FCB     $1F,$10,$10,$1E,$10,$10,$1F,$00
CHF     FCB     $1F,$10,$10,$1E,$10,$10,$10,$00
CHG     FCB     $0E,$11,$10,$17,$11,$11,$0E,$00
CHH     FCB     $11,$11,$11,$1F,$11,$11,$11,$00
CHI     FCB     $0E,$04,$04,$04,$04,$04,$0E,$00
CHJ     FCB     $07,$02,$02,$02,$02,$12,$0C,$00
CHK     FCB     $11,$12,$14,$18,$14,$12,$11,$00
CHL     FCB     $10,$10,$10,$10,$10,$10,$1F,$00
CHM     FCB     $11,$1B,$15,$15,$11,$11,$11,$00
CHN     FCB     $11,$11,$19,$15,$13,$11,$11,$00
CHO     FCB     $0E,$11,$11,$11,$11,$11,$0E,$00
CHP     FCB     $1E,$11,$11,$1E,$10,$10,$10,$00
CHQ     FCB     $0E,$11,$11,$11,$15,$12,$0D,$00
CHR     FCB     $1E,$11,$11,$1E,$14,$12,$11,$00
CHS     FCB     $0F,$10,$10,$0E,$01,$01,$1E,$00
CHT     FCB     $1F,$04,$04,$04,$04,$04,$04,$00
CHU     FCB     $11,$11,$11,$11,$11,$11,$0E,$00
CHV     FCB     $11,$11,$11,$11,$11,$0A,$04,$00
CHW     FCB     $11,$11,$11,$15,$15,$15,$0A,$00
CHX     FCB     $11,$11,$0A,$04,$0A,$11,$11,$00
CHY     FCB     $11,$11,$11,$0A,$04,$04,$04,$00
CHZ     FCB     $1F,$01,$02,$04,$08,$10,$1F,$00
        ORG     NGEN
CH0     FCB     $0E,$11,$13,$15,$19,$11,$0E,$00
CH1     FCB     $04,$0C,$04,$04,$04,$04,$0E,$00
CH2     FCB     $0E,$11,$01,$02,$04,$08,$1F,$00
CH3     FCB     $1F,$02,$04,$02,$01,$11,$0E,$00
CH4     FCB     $02,$06,$0A,$12,$1F,$02,$02,$00
CH5     FCB     $1F,$10,$1E,$01,$01,$11,$0E,$00
CH6     FCB     $06,$08,$10,$1E,$11,$11,$0E,$00
CH7     FCB     $1F,$01,$02,$04,$08,$08,$08,$00
CH8     FCB     $0E,$11,$11,$0E,$11,$11,$0E,$00
```

```
CH9     FCB     $0E,$11,$11,$0F,$01,$02,$0C,$00
        ORG     $1FF8
        FDB     KEYSCN
        ORG     $1FFE
        FDB     MAIN
        END
```

```
*****************************************************************************
*                                                                           *
*            REAL TIME CLOCK PROGRAM                                        *
*                                                                           *
*****************************************************************************
*THIS PROGRAM IS FOR THE EVALUATION OF REAL TIME CLOCK
*ALARM INTERRUPTS HAPPEN AT 00:01 AND 00:02
*ONLY NUMERIC KEY IS ACKNOWLEDGED WHEN RUNNING THIS PROGRAM
*****************************************************************************
            ABSOLUTE
            ORG     $50
BUF         RMB     16
KEYA        RMB     1
KEYB        RMB     1
KEYFL       RMB     1
KEYCTR      RMB     1
GENFL       RMB     1
TMP         RMB     1
TMPL        RMB     1
TMPH        RMB     1
TMP1        RMB     1
TMP2        RMB     1
TMP3        RMB     1
TMP4        RMB     1
TMP5        RMB     1
STACK       RMB     1
RTMP        RMB     1
STACK1      RMB     1
STACK2      RMB     1
ASCODE      RMB     1
PORTA       EQU     00
PORTB       EQU     01
PORTAD      EQU     04
```

```
PORTBD    EQU     05
CTRL1     EQU     $09       CONTROL REGISTER 1
CTRL2     EQU     $0A       CONTROL REGISTER 2
RTCFLG    EQU     $1C       REAL TIME CLOCK INTERRUPT FLAGS REGISTER
MINAL     EQU     $0B       MINUTE ALARM
SECOND    EQU     $1F       SECOND REGISTER
MINUTE    EQU     $1E       MINUTE REGISTER
HOUR      EQU     $1D       HOUR REGISTER
DRAM      EQU     $200
CHGEN     EQU     $1B00
NGEN      EQU     $1C00

          ORG     $800
*************************************************************************
MAIN      BSR     INIT      P02 AND INTERRUPTS INITIALIZATION
          JSR     KEYW      PREPARE KEY SCAN
          BCLR    2,RTCFLG
          JSR     UDRTC     DISPLAY REAL TIME 23:59:59
          CLI               WAITING FOR KEYB AND RTC INTERRUPT
STOP      STOP
          BRA     STOP
*
*************************************************************************
*INITIALIZATION ROUTINE
INIT      BSET    6,RTCFLG        SELECT PLL CLOCK FOR MCU
          LDA     #$FF
          STA     PORTAD
          STA     PORTA           CLEAR UNNECESSARY INTERRUPT
          CLR     $0B
          CLR     $0C
          LDA     #%01110111      ENABLE KEYBOARD, ALARM AND ONCE
*                                 A SECOND & ONCE A DAY INTERRUPT
          STA     CTRL1
          RTS
*
*************************************************************************
*REAL TIME CLOCK INTERRUPT
RTCINT    BRSET   2,RTCFLG,OASEC     ONCE A SECOND INTERRUPT
          BRSET   1,RTCFLG,ALINT     ALARM INTERRUPT
          BRSET   0,RTCFLG,OADAY     ONCE A DAY INTERRUPT
```

```
*******************************************************************************
*ONCE A SECOND INTERRUPT
OASEC     BCLR    2,RTCFLG          RESET ONCE A SECOND INTERRUPT
          BRSET   1,RTCFLG,ALINT    CHECK FOR ALARM INTERRUPT
          BRSET   0,RTCFLG,OADAY CHECK FOR ONCE A DAY INTERRUPT
RTCRET    JSR     CLRBUF  CLEAR BUFFER
          JSR     DISRTC            RTC --> BUFFER
RTCIBB    CLR     TMP1
          JSR     BUTODR
          LDX     #8
          STX     TMP1
          JSR     BUTODR            BUFFER  --> DISPLAY RAM
RTCIBA    RTI
*
*******************************************************************************
*ONCE A DAY INTERRUPT
OADAY     BCLR    0,RTCFLG          RESET ONCE A DAY INTERRUPT
          JSR     DAYMESS
          BRA     RTCIBB    RETURN FROM INTERRUPT
*
*******************************************************************************
*ALARM INTERRUPT
ALINT     BCLR    1,RTCFLG          RESET ALARM INTERRUPT
          LDX     MINAL
          CPX     #1
          BEQ     UDBUF
          JSR     UDBUFF2
          BRA     RTCIBB
UDBUF     JSR     UDBUFF1
          BRA     RTCIBB
*
*******************************************************************************
*ROUTINE TO DISPLAY CURRENT TIME
UDRTC     CLR     KEYCTR
          LDX     #23       SET TIME AS 23:59:59
          STX     HOUR
          LDX     #59
          STX     MINUTE
          STX     SECOND
```

```
            BSET    0,MINAL         SET ALARM AT 00:01
            BSET    1,CTRL1         ENABLE ALARM INTERRUPT
            BSR     DISRTC          RTC --> BUFFER
            CLR     TMP1
            BSR     BUTODR
            LDX     #8
            STX     TMP1
            BSR     BUTODR
            RTS
*
*******************************************************************************
*ROUTINE TO DISPLAY SECOND, MINUTE AND HOUR ON LCD PANEL
*TMP1 HAS THE PROPER OFFSET FOR THE DISPLAY POSITION
*DISPLAY SEQUENCE: HOUR, MINUTE ,SECOND
*STACK1 IS USED TO STORE OFFSET OF RTC
*STACK2 IS USED TO STORE OFFSET OF DISPLAY BUF
DISRTC      CLRX
            CLR     STACK2
RTRP        LDA     HOUR,X  GET REAL TIME
            STX     STACK1  SAVE OFFSET
            JSR     A0      GET LDS AND MSD
            LDA     TMPH    GET MSD
            ADD     #$30    TO ASCII
            LDX     STACK2  PUT INTO BUF
            STA     BUF+$8,X
            INCX    ADVANCE TO NEXT BUF LOC
            LDA     TMPL    GET LSD
            ADD     #$30    TO ASCII
            STA     BUF+$8,XPUT INTO BUF
            LDA     #':     DISPLAY ":"
            INCX
            CPX     #$08    ONLY EIGHT CHARACTERS
            BEQ     RTDONE
            STA     BUF+$8,X
            INCX            INCREMENT BUF LOC
            STX     STACK2  SAVE IT IN STACK2
            LDX     STACK1  GET REAL TIME CLOCK OFFSET
            INCX
            BRA     RTRP
RTDONE      RTS
```

```
*
*******************************************************************************
*ROUTINE TO DISPLAY BUF CHARACTER 0-7 ON THE FIRST ROW OF DISPLAY
*RAM
BUTODR   CLR    TMP4
         LDA    TMP1
         STA    TMP5
ALP0     BSR    NUORAL          GET OFFSET AND SET GENFL
         LDA    #$08            EIGHT ROWS FOR A CHARACTER
         STA    TMP3
ALP1     BRSET  0,GENFL,ALPH
         LDA    NGEN,X          GET NUM PATTERN
         BRA    ALPHF
ALPH     LDA    CHGEN,X         GET ALPH PATTERN
ALPHF    STX    TMP2            STORE PATTERN OFFSET
         LDX    TMP1            LOAD DRAM LOCATION OFFSET
         STA    DRAM,X
         INC    TMP2
         LDX    TMP2            ADVANCE TO NEXT DRAM LOCATION
         INC    TMP1
         DEC    TMP3
         BNE    ALP1            COMPLETE LOADING ONE CHARACTER
         LDA    TMP1
         ADD    #$08            ADVANCE TO NEXT DRAM LOC
         STA    TMP1
         INC    TMP5
         INC    TMP4
         LDA    TMP4
         CMP    #$08
         BNE    ALP0
         RTS
*
*******************************************************************************
*TMP4 HAS THE OFFSET FOR BUF
*RETURN WITH GENFL SET
*AND OFFSET IN X
NUORAL   CLR    GENFL           CLEAR GEN FLAG
         LDX    TMP5            LOAD BUF OFFSET
         LDA    BUF,X           GET ASCII
         JSR    AMUL8           GET ROM PATTERN OFFSET
```

```
            CMP     #$40
            BHS     SETGEN      NOT A NUMBER
NRALD       RTS
SETGEN      BSET    0,GENFL
            BRA     NRALD
*
*****************************************************************************
*SUBROUTINE TO ENCODE BINARY NUMBER TO DECIMAL
*REG A CONTAINS NUMBER TO BE CONVERTED BEFORE CALLING THIS
*SUBROUTINE
*TMPL = STORING LSD
*TMH  = STORING MSD
A0          CLRX
A1          CMP     #$0A
            BHS     A2
            STA     TMPL
            STX     TMPH
            BRA     DON
A2          CLC
            SBC     #$0A
            INCX
            BRA     A1
DON         RTS
*
*****************************************************************************
*ROUTINE TO UPDATE REAL TIME CLOCK
RTUD        SUB     #$30        ASCII TO BINARY
            STA     ASCODE      STORE ACSII BEFORE CALLING SUBROUTINE
            JSR     CLRBUF
            JSR     DISRTC
            JSR     DISBUF      UPDTAE REAL TIME ON LCD
            LDA     ASCODE
            LDX     KEYCTR
            BEQ     KEYVL0      LSD OF HOUR
            CPX     #1
            BEQ     KEYVL1      MSD OF HOUR
            CPX     #2
            BEQ     KEYVL2      LSD OF MINUTE
            CPX     #3
            BEQ     KEYVL3      MSD OF MINUTE
```

```
        CPX     #4
        BEQ     KEYVL4    LSD OF SECOND
        CPX     #5
        BEQ     KEYVL5    MSD OF SECOND
        CLR     KEYCTR
        BRA     KEYVL0

KEYVL0  CLRX
        BSR     FIRCOM    MSD + NEW VALUE -> HOUR
        CMP     #23
        BHI     KEYCOM    HOUR MUST BE SMALLER THAN 24
        STA     HOUR
COMJOB  JSR     DISRTC    RTC -> BUF
        CLR     TMP1
        JSR     BUTODR
        LDX     #8
        STX     TMP1
        JSR     BUTODR    BUF -> DRAM
        INC     KEYCTR    NEXT MODIFICATION IS MSD
KEYCOM  RTS

KEYVL1  LDX     #1
        BSR     SECOM     LSD + NEW VALUE -> HOUR
        CMP     #23       HOUR MUST BE SMALLER THAN 24
        BHI     KEYCOM
        STA     HOUR
        BRA     COMJOB

KEYVL2  LDX     #3
        BSR     FIRCOM    MSD + NEW VALUE -> MINUTE
        STA     MINUTE
        BRA     COMJOB

KEYVL3  LDX     #4
        BSR     SECOM     LSD + NEW VALUE -> MINUTE
        CMP     #59       MINUTE MUST BE SMALLER THAN 59
        BHI     KEYCOM
        STA     MINUTE
        BRA     COMJOB
```

```
KEYVL4  LDX     #6
        BSR     FIRCOM      MSD + NEW VALUE -> SECOND
        STA     SECOND
        BRA     COMJOB

KEYVL5  LDX     #7
        BSR     SECOM       LSD + NEW VALUE -> SECOND
        CMP     #59
        BHI     KEYCOM      SECOND MUST BE SMALLER THAN 59
        STA     SECOND
        BRA     COMJOB

*******************************************************************************
*ROUTINE TO GET MSD FROM BUF THAT HAS BEEN UPDATE
*RTMP = MSD + LSD
FIRCOM  STA     STACK1
        LDA     BUF+8,X         MSD -> A REG.
        SUB     #$30            ASCII TO BINARY
        STA     RTMP
        JSR     ENCOD           THIS IS A MSD
        LDA     STACK1
        ADD     RTMP
        RTS
*
*******************************************************************************
*ROUTINE TO GET LSD FROM BUF
*RTMP = NEW MSD + LSD
SECOM   STA     RTMP
        JSR     ENCOD           CALCULATE NEW MSD
        LDA     BUF+8,X         GET LSD
        SUB     #$30            ASCII TO BINARY
        ADD     RTMP
        RTS
*
*******************************************************************************
*ROUTINE TO UPDATE DISPLAY RAM WITH CURRENT TIME ON REAL TIME
*CLOCK
DISBUF  CLR     TMP1
        JSR     BUTODR
        LDX     #8
```

```
            STX     TMP1
            JSR     BUTODR
            RTS
*
*******************************************************************************
*KEYSCAN ROUTINE
KEYSCN      LDA     #$F7        START CHECKING FROM THE 4TH COLUMN
            STA     PORTB
REPEAT      LDA     PORTA
            AND     #$1F        START SCANNING FROM THE BOTTOM ROW
            CMP     #$1F
            BNE     GOTIT       KEY FOUND
            LSR     PORTB       REPEAT IF MORE COLUMN
            BCS     REPEAT
            CLR     KEYFL       PREPARE FOR NEXT SCAN
            CLR     PORTA
            CLR     PORTB
RETURN      RTI

GOTIT       BRSET   0,KEYFL,NOSAVE
            LDA     PORTB       SAVE KEY IN MEMORY
            LDX     PORTA
            STA     KEYB
            STX     KEYA
            BSR     DBOUNC      CHECK FOR NOISE
            BSET    0,KEYFL
            BRA     KEYSCN
NOSAVE      BSR     DBOUNC      PAUSE
            CLR     PORTA       PREPARE SCAN LINE
            CLR     PORTB
            CLR     KEYFL
            BRA     DECODE      GO TO USER KEY DECODE ROUTINE

*******************************************************************************
*DEBOUNCE ROUTINE
DBOUNC      LDA     #60
AGAIN1      LDX     #$FF
AGAIN       DECX
            BNE     AGAIN
            DECA
```

```
            BNE     AGAIN1
            RTS
*
*************************************************************************
*DECODE ROUTINE
DECODE      BSET    4,CTRL2         JUST TO MAKE SURE LCD IS TURNED ON
            BRCLR   0,KEYA,ROW0     FIRST ROW
            BRCLR   1,KEYA,ROW1     SECOND ROW
            BRCLR   2,KEYA,ROW2     THIRD ROW
            BRCLR   3,KEYA,THROW
            JMP     ROW4            FIFTH ROW
THROW       JMP     ROW3
ROW0        BRCLR   0,KEYB,RC00
            BRCLR   1,KEYB,RC01     0
            BRCLR   2,KEYB,RC02
            BRCLR   3,KEYB,RC03
RC00        JMP     DONE
RC01        LDA     #'0
            JMP     FIN1
RC02        JMP     DONE
RC03        JMP     DONE
ROW1        BRCLR   0,KEYB,RC10     7
            BRCLR   1,KEYB,RC11     8
            BRCLR   2,KEYB,RC12     9
            BRCLR   3,KEYB,RC13
RC10        LDA     #'7
            JMP     FIN1
RC11        LDA     #'8
            JMP     FIN1
RC12        LDA     #'9
            JMP     FIN1
RC13        JMP     DONE+OFF1
ROW2        BRCLR   0,KEYB,RC20     4
            BRCLR   1,KEYB,RC21     5
            BRCLR   2,KEYB,RC22     6
            BRCLR   3,KEYB,RC23     SPACE
RC20        LDA     #'4
            JMP     FIN1
RC21        LDA     #'5
            JMP     FIN1
```

```
RC22        LDA     #'6
            JMP     FIN1
RC23        INC     KEYCTR
            LDA     KEYCTR
            CMP     #$10
            BNE     CCTR
            CLR     KEYCTR
CCTR        BRA     DONE
ROW3        BRCLR   0,KEYB,RC30     1
            BRCLR   1,KEYB,RC31     2
            BRCLR   2,KEYB,RC32     3
            BRCLR   3,KEYB,RC33
ROW4        BRCLR   0,KEYB,RC40
            BRCLR   1,KEYB,RC41
            BRCLR   2,KEYB,RC42
            BRCLR   3,KEYB,RC43
RC30        LDA     #'1
            BRA     FIN1
RC31        LDA     #'2
            BRA     FIN1
RC32        LDA     #'3
            BRA     FIN1
RC33        BRA     FIN
RC40        BRA     FIN
RC41        BRA     FIN
RC42        BRA     FIN
RC43        BRA     FIN
FIN         BRA     DONE
FIN0        BRA     DONE
FIN2        BRA     DONE
FIN1        JSR     RTUD
DONE        JMP     RETURN
*
*******************************************************************************
*ASCII TO BIN AND MUL BY 8
*X REGISTER DESTROY
AMUL8       STA     STACK
            CMP     #$40
            BHS     SUB40
            SUB     #$30
```

```
            BRA     JUO40
SUB40       SUB     #$40
JUO40       LSLA
            LSLA
            LSLA
            TAX
            LDA     STACK
            RTS
*
*******************************************************************************
*ROUTINE TO PREPARE KEYBOARD INTERRUPT
*PORT A IS  INPUT
*PORT B IS  OUTPUT
KEYW        CLR     PORTAD      PORTA A AS INPUT
            CLR     PORTA       PREPARE SCANNING LINES
            CLR     PORTB
            CLR     KEYFL
            CLR     KEYCTR
            LDA     #$0F        PORT B AS OUTPUT
            STA     PORTBD
            JSR     CLRBUF      CLEAR DISPLAY BUFFER
            CLR     TMP1
            JSR     BUTODR
            LDA     #8
            STA     TMP1
            JSR     BUTODR      CLEAR DISPLAY RAM
            BSET    4,CTRL2     TURN ON LCD
            RTS
*
*******************************************************************************
*MULTIPLY BY 10
ENCOD       STA     STACK
            LSL     RTMP
            LDA     RTMP
            LSL     RTMP
            LSL     RTMP
            ADD     RTMP
            STA     RTMP
            LDA     STACK
            RTS
*
```

```
*******************************************************************************
*ROUTINE TO CLEAR BUFFER #$40 —> BUFFER
CLRBUF    LDA     #$40
          CLRX
CLROP     STA     BUF,X
          INCX
          CPX     #$10
          BNE     CLROP
          RTS
*
*******************************************************************************
*UPDATE BUFFER WITH THE ONCE A DAY INTERRUPT MESSAGE "DAY INT"
*WHEN ONCE A DAY INTERRUPT HAPPENS AT 00:00:00
DAYMESS   CLRX
DAYLOP    LDA     D1,X
          STA     BUF,X
          INCX
          CPX     #$10
          BNE     DAYLOP
          RTS
D1        FCB     $40,$40,$40,$40,$40,$40,$40,$40
          FCC     /DAY/
          FCB     $40,$40
          FCC     /INT/
*
*******************************************************************************
*UPDATE BUFFER WITH THE FIRST ALARM MESSAGE "ALARM   TEST1"
*WHEN ALARM INTERRUPT HAPPENS AT 00:01
UDBUFF1   CLRX
BU1LOP    LDA     T1,X
          STA     BUF,X
          INCX
          CPX     #$10
          BNE     BU1LOP
          LDX     #02
          STX     MINAL
          BSET    1,CTRL1
          RTS
```

```
T1          FCC     /ALARM/
            FCB     $40,$40,$40
            FCC     /TEST1/
            FCB  $  40,$40,$40
*
*******************************************************************************
*UPDATE BUFFER WITH THE SECOND ALARM MESSAGE "ALARM   TEST2"
*WHEN ALARM INTERRUPT HAPPENS AT 00:02
UDBUFF2     CLRX
BU2LOP      LDA     T2,X
            STA     BUF,X
            INCX
            CPX     #$10
            BNE     BU2LOP
            LDX     #01
            STX     MINAL
            BSET    1,CTRL1   ENABLE ALARM INTERRUPT
            RTS

T2          FCC     /ALARM/
            FCB     $40,$40,$40
            FCC     /TEST2/
            FCB     $40,$40,$40
*
*******************************************************************************
            ORG     CHGEN
SPA         FCB     $00,$00,$00,$00,$00,$00,$00,$00
CHA         FCB     $04,$0A,$11,$11,$1F,$11,$11,$00
CHB         FCB     $1E,$11,$11,$1E,$11,$11,$1E,$00
CHC         FCB     $0E,$11,$10,$10,$10,$11,$0E,$00
CHD         FCB     $1C,$12,$11,$11,$11,$12,$1C,$00
CHE         FCB     $1F,$10,$10,$1E,$10,$10,$1F,$00
CHF         FCB     $1F,$10,$10,$1E,$10,$10,$10,$00
CHG         FCB     $0E,$11,$10,$17,$11,$11,$0E,$00
CHH         FCB     $11,$11,$11,$1F,$11,$11,$11,$00
CHI         FCB     $0E,$04,$04,$04,$04,$04,$0E,$00
CHJ         FCB     $07,$02,$02,$02,$02,$12,$0C,$00
CHK         FCB     $11,$12,$14,$18,$14,$12,$11,$00
CHL         FCB     $10,$10,$10,$10,$10,$10,$1F,$00
CHM         FCB     $11,$1B,$15,$15,$11,$11,$11,$00
```

```
CHN     FCB     $11,$11,$19,$15,$13,$11,$11,$00
CHO     FCB     $0E,$11,$11,$11,$11,$11,$0E,$00
CHP     FCB     $1E,$11,$11,$1E,$10,$10,$10,$00
CHQ     FCB     $0E,$11,$11,$11,$15,$12,$0D,$00
CHR     FCB     $1E,$11,$11,$1E,$14,$12,$11,$00
CHS     FCB     $0F,$10,$10,$0E,$01,$01,$1E,$00
CHT     FCB     $1F,$04,$04,$04,$04,$04,$04,$00
CHU     FCB     $11,$11,$11,$11,$11,$11,$0E,$00
CHV     FCB     $11,$11,$11,$11,$11,$0A,$04,$00
CHW     FCB     $11,$11,$11,$15,$15,$15,$0A,$00
CHX     FCB     $11,$11,$0A,$04,$0A,$11,$11,$00
CHY     FCB     $11,$11,$11,$0A,$04,$04,$04,$00
CHZ     FCB     $1F,$01,$02,$04,$08,$10,$1F,$00

        ORG     NGEN
CH0     FCB     $0E,$11,$13,$15,$19,$11,$0E,$00
CH1     FCB     $04,$0C,$04,$04,$04,$04,$0E,$00
CH2     FCB     $0E,$11,$01,$02,$04,$08,$1F,$00
CH3     FCB     $1F,$02,$04,$02,$01,$11,$0E,$00
CH4     FCB     $02,$06,$0A,$12,$1F,$02,$02,$00
CH5     FCB     $1F,$10,$1E,$01,$01,$11,$0E,$00
CH6     FCB     $06,$08,$10,$1E,$11,$11,$0E,$00
CH7     FCB     $1F,$01,$02,$04,$08,$08,$08,$00
CH8     FCB     $0E,$11,$11,$0E,$11,$11,$0E,$00
CH9     FCB     $0E,$11,$11,$0F,$01,$02,$0C,$00
COL     FCB     $00,$0C,$0C,$00,$00,$0C,$0C,$00
        ORG     $1FF2
        FDB     RTCINT
        ORG     $1FF8
        FDB     KEYSCN
        ORG     $1FFE
        FDB     MAIN
        END
```

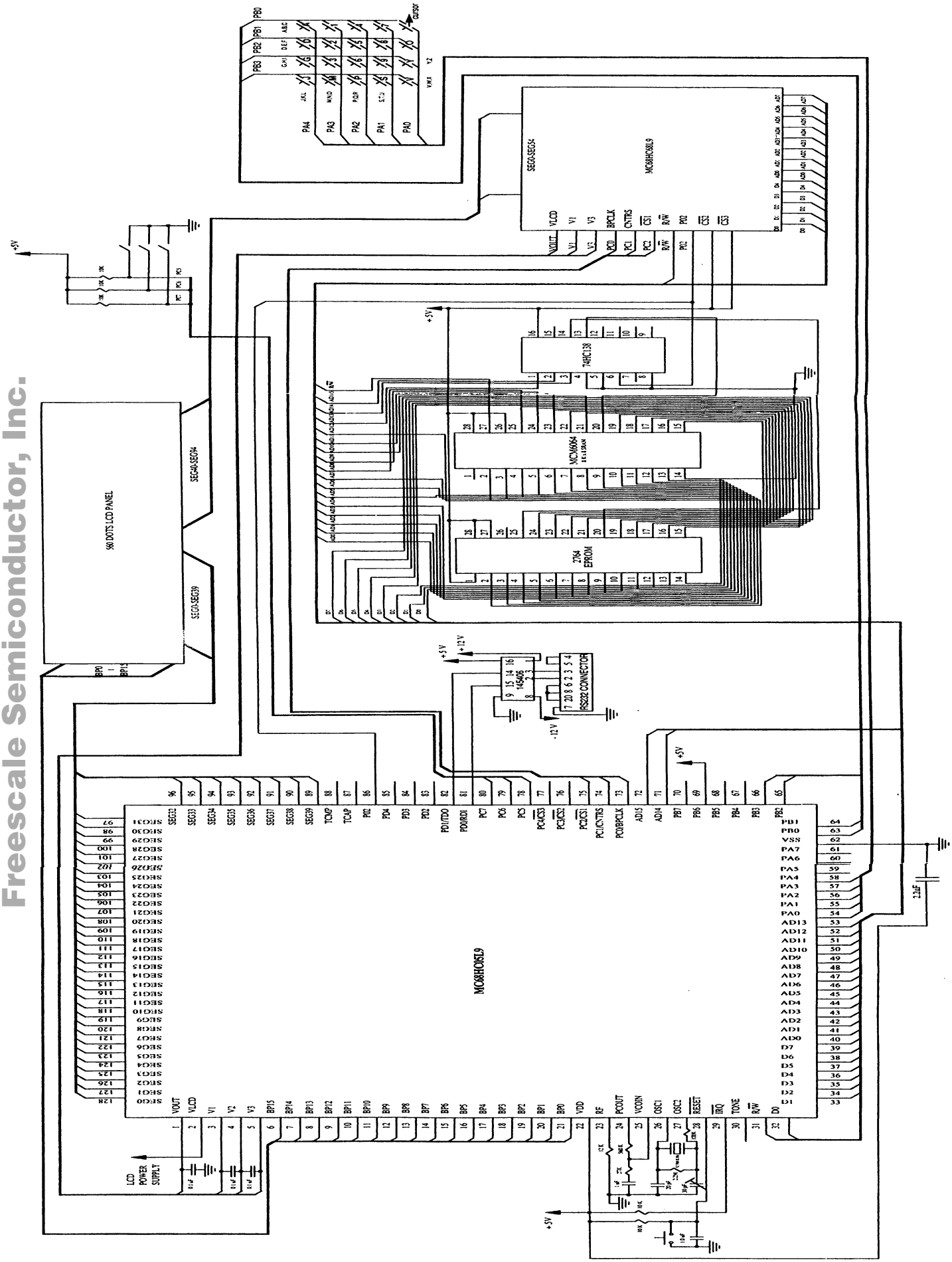


FIGURE 1.19 APPLICATION DEMO BOARD CIRCUIT SCHEMATIC

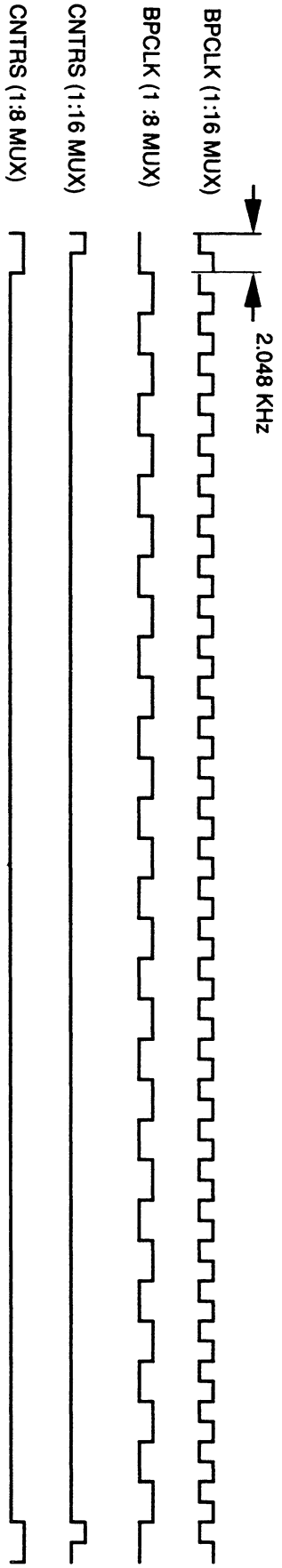


FIGURE 2 SYNCHRONOUS SIGNAL FOR SLAVE LCD DRIVER

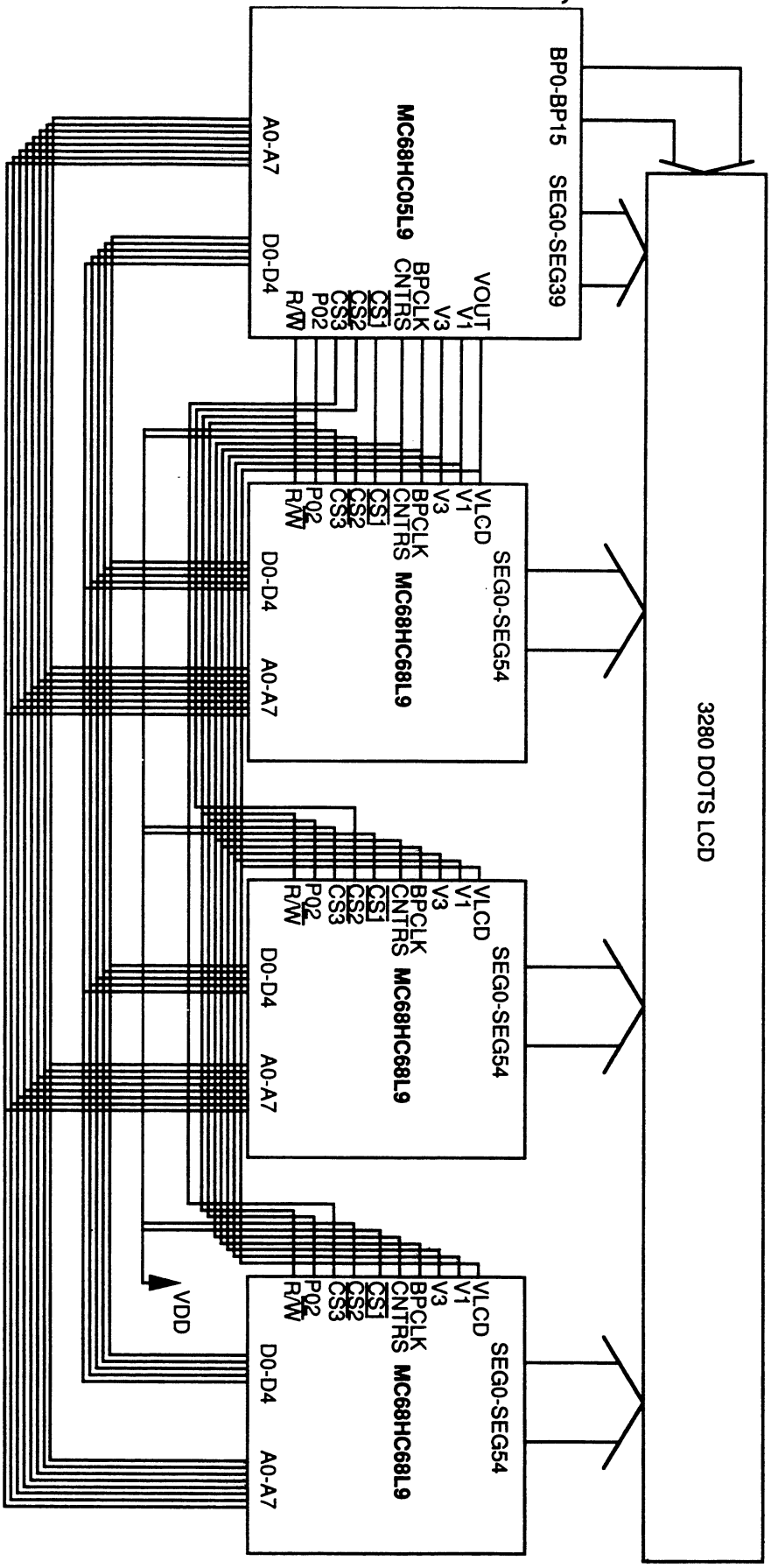
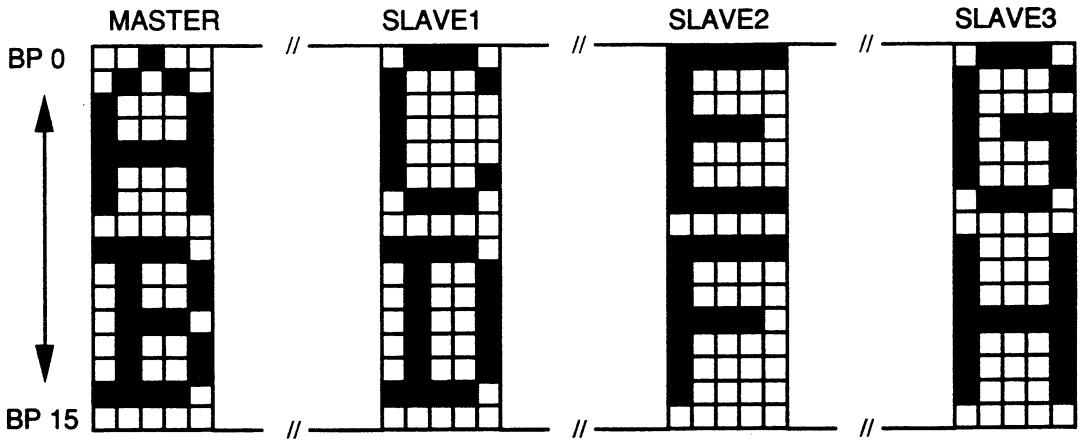


FIGURE 3 MASTER MCU AND SLAVE LCD DRIVERS PINS CONNECTION



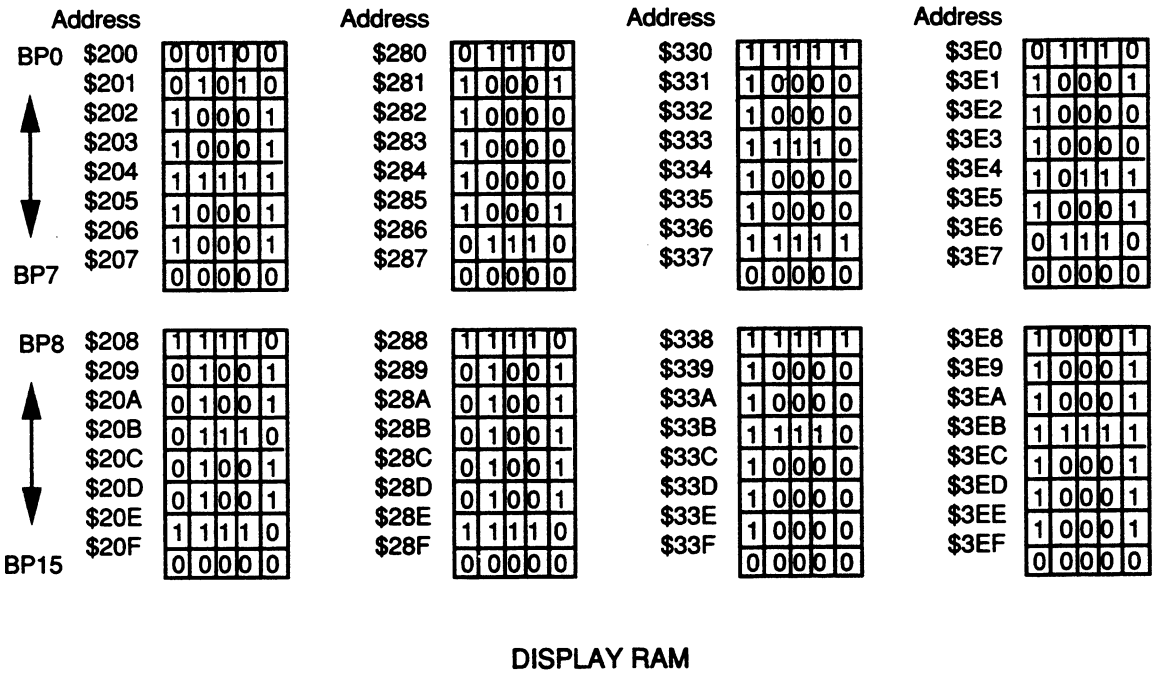


FIGURE 4 RELATION BETWEEN DISPLAY RAM DATA AND DISPLA

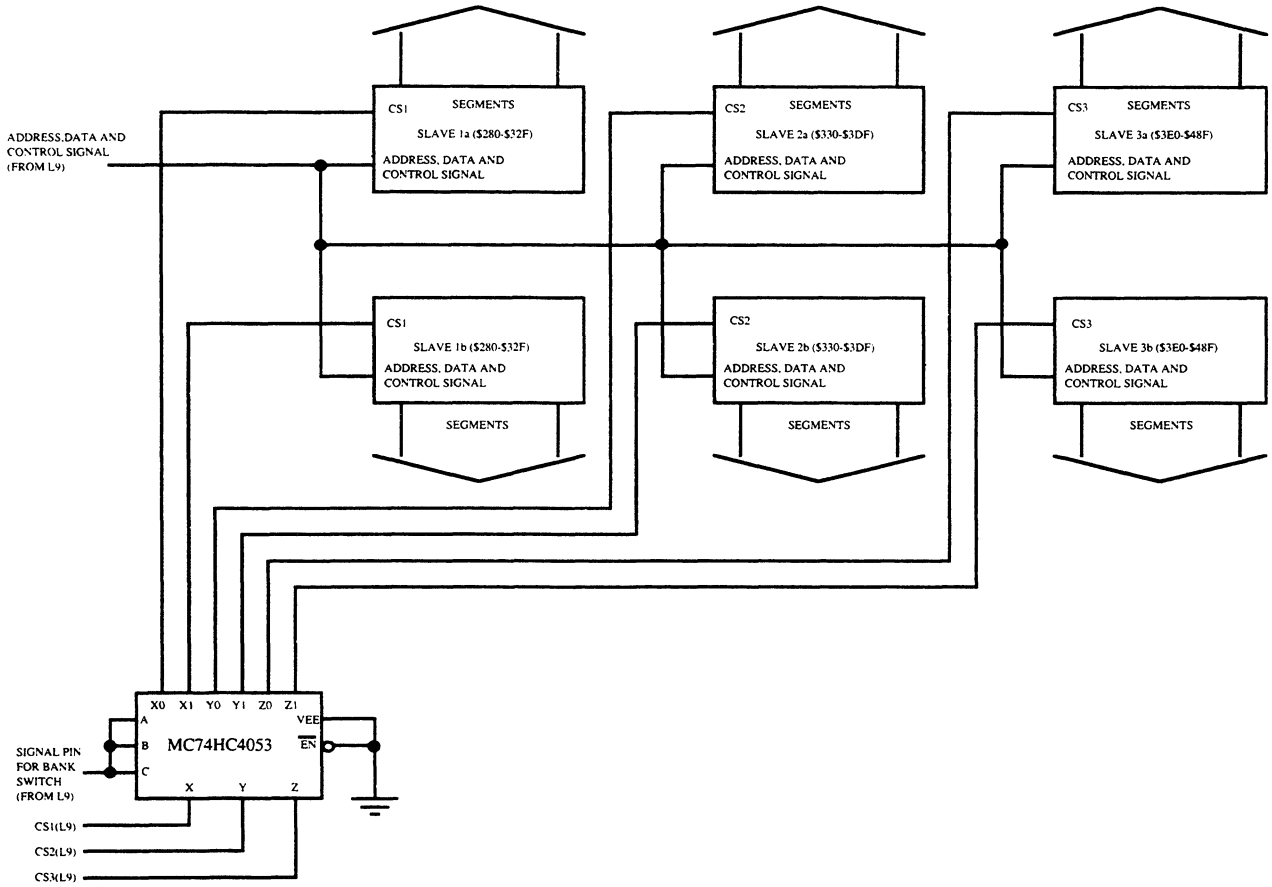


FIGURE 5 USE BANK SWITCHING TECHNIQUE TO EXPAND THE LCD DRIVING CAPABILITY OF L9

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