AN-HK-13A

MC68HC05L10 AN ENHANCED VERSION OF L9 FOR HANDHELD EQUIPMENT APPLICATIONS

Prepared by Jamson Cheung
System Design & Application Department, Technical Operations
Motorola Semiconductors Hong Kong Ltd.

INTRODUCTION

The MC68HC05L10 (L10), new member of the DRAGONKAT™ family of Micro-Controllers Unit (MCU), is particularly tailored for application in handheld equipments, such as organizers, meter readers, inventory checkers, handheld diagnostic terminals, or personal communication products etc., where low power consumption and system optimization are the main concerns to the product designer. This enhanced version of MC68HC05L9 (L9) has upgraded features to meet the requirements in advanced applications. The similarities and differences between these two MCUs are being highlighted in this application note. In addition, software routines on some new features such as keyscan, LCD (Liquid Crystal Display) and Memory Management Unit (MMU) are also described in detail to guild users in their program development for a specific application.

FEATURE COMPARISON BETWEEN MC68HC05L9 AND MC68HC05L10

Both L10 and L9 belong to the 68HC05 microprocessor family so that they share the same instruction set. Since both of them serve for the same intended application, most of the circuit blocks are the same for ease of use. However, certain features have been added or enhanced in L10 to expand its capability. Their similarities and differences are highlighted in below.

The following on-chip circuits remain unchanged:

* Real time clock and its alarm function
* 16 bit programmable free run timer
* Serial communication interface (SCI)
* Tone generator
* LCD auto-display off
* Keyboard wake-up interrupt
* Separate external data and address bus
* High frequency voltage controlled oscillator with PLL (Phase Lock Loop) locking into refer ence frequency from 32.768 KHz crystal oscillator.

New Features Added or Features Enhanced to the MC68HC05L10:

* Serial peripheral interface (SPI). It provides a means to interface with other peripherals or MCU in a sophisticated system.

* Memory management unit (MMU). A hardware implementation to extend the memory addressing space from 64KB to 1MB for data storage. The number of address lines is therefore increased to 20.

* External ROM. Internal ROM disabled by hardware pin RDIS which gives the user a choice of using internal or external ROM.

* An additional external interrupt input.

* Two external pins only on PLL. RF pin in L9 PLL system is eliminated by RF resistor on-chip. No VCO (Voltage Controlled Oscillator) center frequency adjustment is required.

* Low voltage inhibit feature deleted. Due to the limitation of achieving high accuracy on internal reference voltage, it is recommended to use external low voltage detection circuit.

* Enhanced LCD driving capability. A programmable 32- or 41-multiplex backplane (or common) drive is built-in to expand the multiplex ratios while minimizing system chip count by fully utilizing the process ability of sustaining voltage. The segments are driven by a separate device (MC141511) which handles up to 128 segments each. The maximum segment driving capability is limited by the display RAM space located at $1C0 to $BFF in L10. Up to four MC141511 segment drivers of 41X128X4=20,992 pixels in total on a single LCD display panel or up to two MC141511 segment drivers in a split panel with the same display pixels may be cascaded. The built-in display RAM in MC141511 is organized in 8 bit vertical format. However, the 16 bytes of display RAM for the annunciator in the backplane 40th is arranged horizontally as shown in Figure 3. This configuration of the display RAM significantly eases programming for display of Chinese character in terms of changing the character pitch or performing horizontal scrolling.

* Less interference from bus. During internal memory access, the data bus is at high impedence, address bus at low state, R/W line at high state and P02 switched off. If STOP or WAIT instruction is executed during CPU accessing external memory, the address bus will hold at the location of the instruction following the STOP or WAIT until an interrupt or reset occurs. The interference induced by digital pulses from external bus lines or P02 is, therefore, eliminated during those operations. Furthermore, it also prevents the excess current contributed by the floating data lines and external memory devices during STOP or WAIT mode operation regardless of RIDS pin status.

PROGRAMMING PLL FOR BUS SPEEDS

The PLL system is shown in Figure 1. Upon power on or external reset, the CLKS bit in $1C register is cleared which set the CPU bus directly sourced from 32.768KHz crystal. To change the CPU bus speed from 32.768KHz, follow the sequence shown in flow chart Figure 2. The sampling rate of the PLL is 8KHz or sampling at every 125µs interval. If there is no error occurred at phase detector output after 32 consecutive samplings which equals to 4ms, a PLLI bit in $26 register will be set. It indicates that the PLL stablized at new bus frequency. For operations on those require accurate clock frequency and duty cycle, such as SCI or counter etc., should make sure that the PLLI is set before start execution of the program. The software routine below gives an example:

```
        BCLR    6,$1C       Select 16.384 KHz as internal bus frequency
        BSET    5,$27       Select 1.2288 MHz for PLL
DELAY   LDA     #$9         4.0 msec delay loop for min PLLI settle time
        DECA
        BNE     DELAY
STABLE  BRCLR   6,$26,STABLE Wait until PLL is stable
        BSET    6,$1C       Select PLL as the internal bus frequency
        JMP     SERVE       Go to service routine
```

INTERFACE BETWEEN MASTER L10 AND SLAVE MC141511 SEGMENT DRIVER

The connection between the L10 and the segment drivers in a single display panel is shown in Figure 4, and the simplified schematic for a split panel in Figure 5. The display RAM in the segment driver is configured in a "WHAT YOU STORE IS WHAT YOU SEE" (WYSIWYS) scheme which is the same as in L9. But the display RAM is to be accessed by the master L10 in 8-bit bytes oriented vertically instead of 5 bit arranged horizontally as in L9. The four segment drivers corresponding to the 2624 bytes (4X41X128) of display RAM located in L10 memory space $1C0 to $BFF are to be selected by four individual chip select pins CS1-CS4. Care should be taken on the system memory address decoding circuit. For details of the address correlation please refer to Table 6 "MCU Display Logical & Physical Address Translation" on page 6-9, MC68HC05L10 Product Preview and Figure 7 "Display RAM Configuration at 1:32 and 1:41 Multiplex Ratios" on page 7, of the MC141511 Product Preview. For better understanding the relationship an illustration is shown in Figure 6. Notice that the segment driver's data clock rate is 4.096KHz/2 so that there is a time lag of up to 500 usec after DON bit (bit 3 of $26 register) is set before the display data becomes valid.

PROGRAMMING FOR LCD DRIVE AND MMU

The program routine described here demonstrates an LCD driver that has large amount of symbols and font patterns stored in the external memory access via the MMU facility. Users have the flexibility of selecting various display symbols and font patterns or sizes. In this program, a display pattern shown in Figure 7 is chosen for demonstration. It consists of two rows of Chinese characters in 16X16 pixels, and one row of 21 alphanumerics in 8X6 pixels, the third row of characters is being scrolled softly from right to left. The flow chart of this program is shown in Figure 8a to 8c. It can be divided into three portions: The first portion is to search for the user's required symbol code and its font pattern; the second portion is to locate the display RAM address in L10 to its corresponding display pixel position on LCD; the last portion is to scroll the third character row located at 32nd to 39th backplane in 41 multiplex mode.

The font pattern is organized in a tabular structure so that the user can easily modify the symbols and font patterns to fit their requirements. The number of symbols and font patterns can be extended beyond 64 KB with the on-chip MMU. User can clearly notice from this program how easy it is to access the memory space beyond 64 KB.

PROGRAMMING FOR KEYPAD SCANNING

This KEYSCAN program routine demonstrates a 6x4 matrix in a MXN keypad (M in port A AND N in port B). In order to be versatile the number of keys and the definition of each individual key can be user specified. The flow chart of this program and the associated schematic are shown in Figure 9 and 10 respectively.

To adapt this program to a user specified keypad, user only needs to follow a simple procedure: Firstly, to assign the number of key M in port A and N in port B; secondly, to modify the keypad definition routine as required by the particular application; lastly, to enter the routine entry address into the KEYDEF table. The number of M and N is specified in the section of the program called NUMKEYA and MUNBERB respectively.

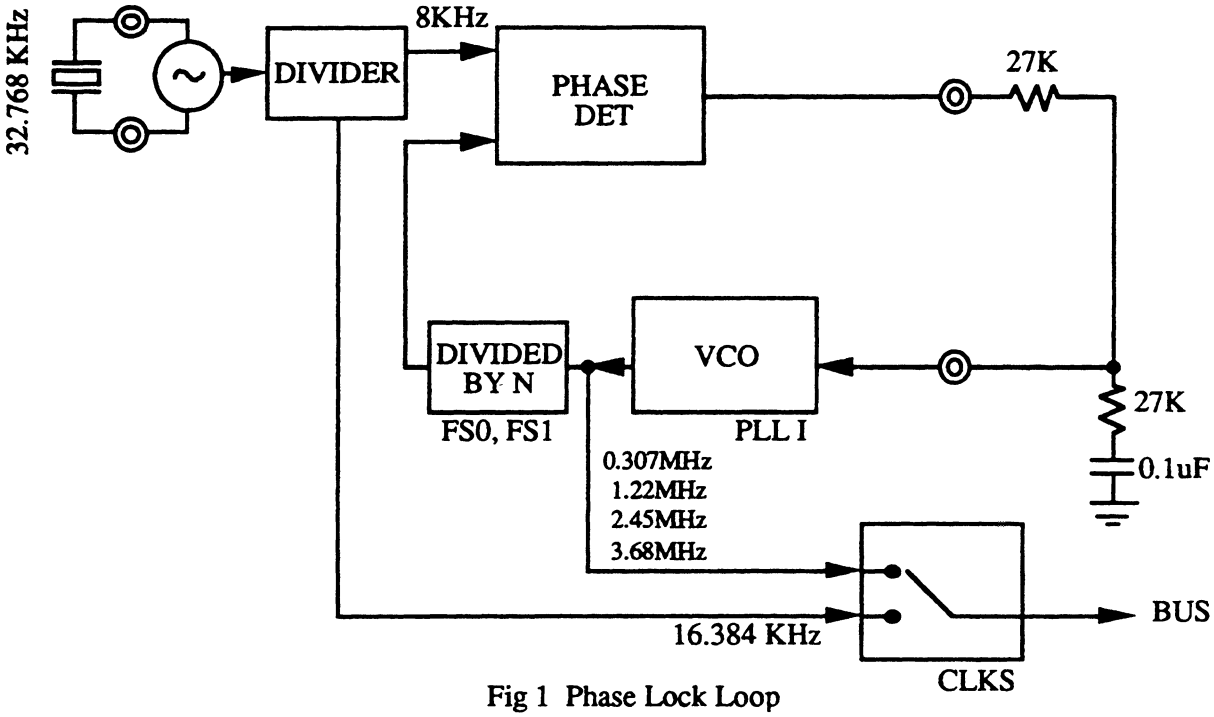
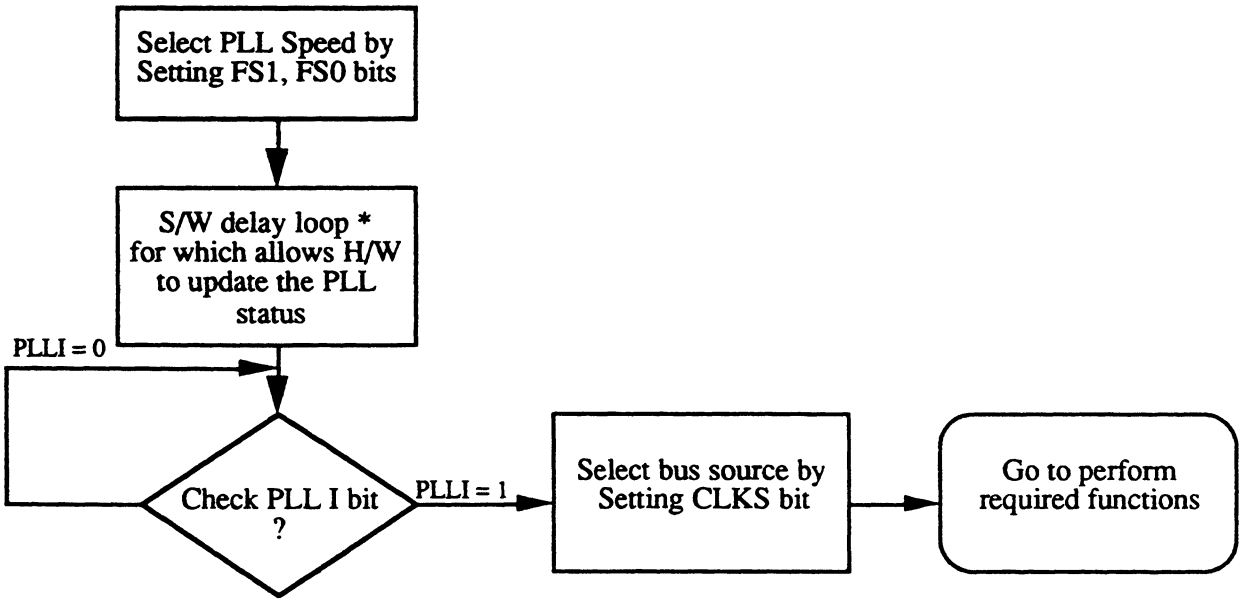


Fig 1 Phase Lock Loop



* L7, L9 need at least 1 msec delay for PLLI to change from 1 to 0 and 16ms from 0 to 1.
L10 needs at least 125 µsec delay for PLLI to change from 1 to 0 and 4ms from 0 to 1.

Fig 2 Flow Chart of Programming PLL for Bus Speed

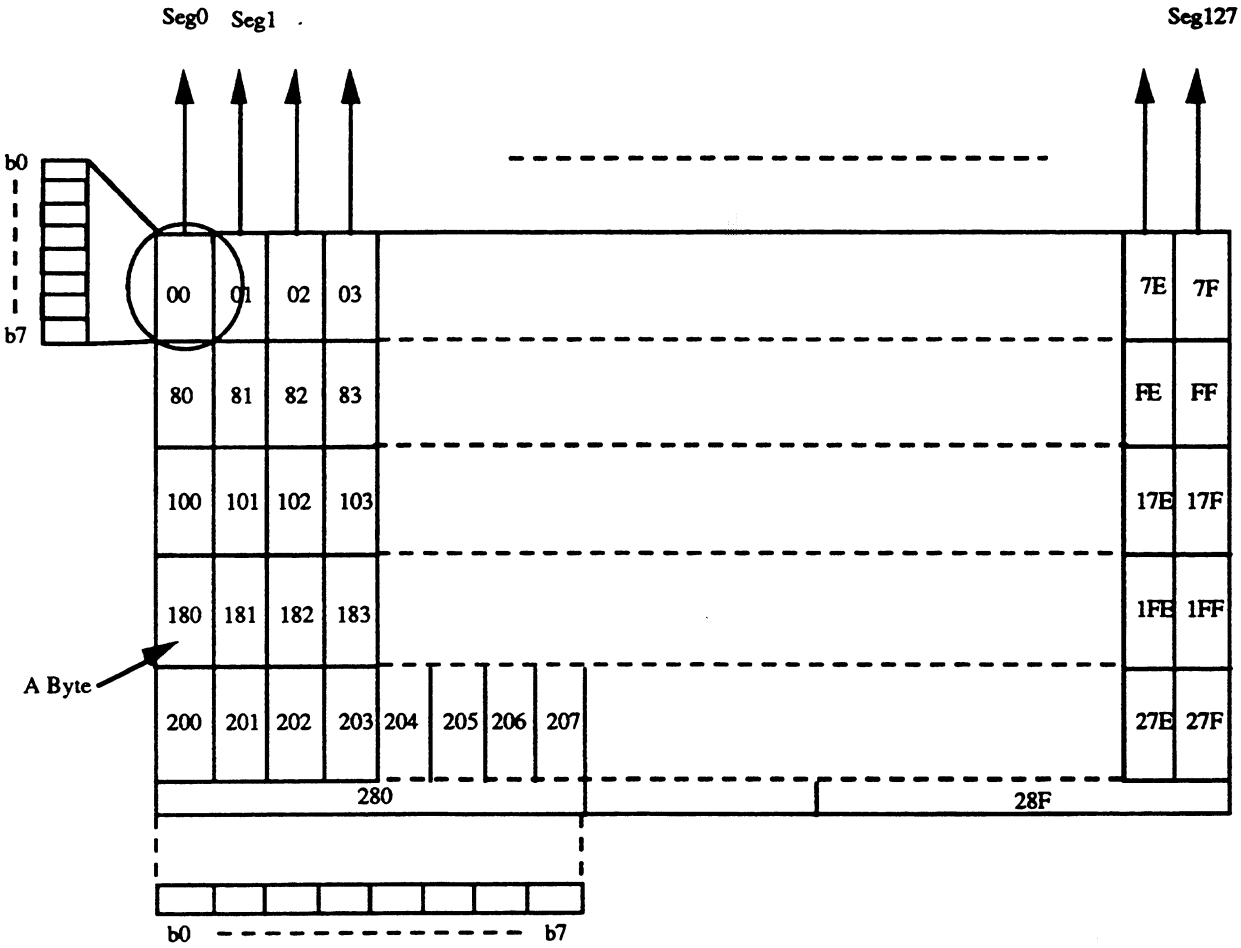


Fig 3 MC141511 Display RAM Configuration

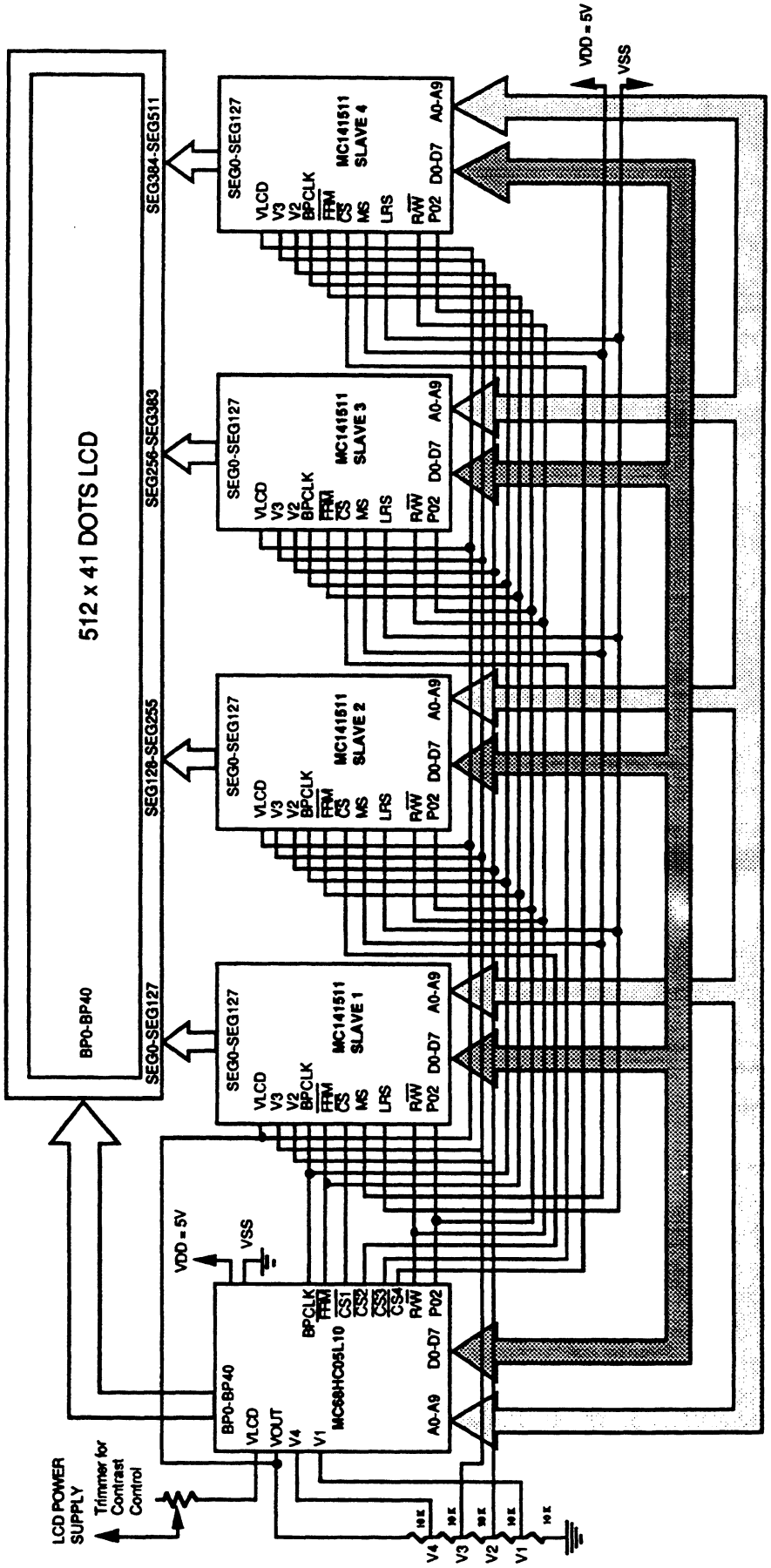


Fig 4 Full Capability of Master MCU and Slave LCD Drivers Pins Connection for 1:41 Mux and Left to Right Display Select

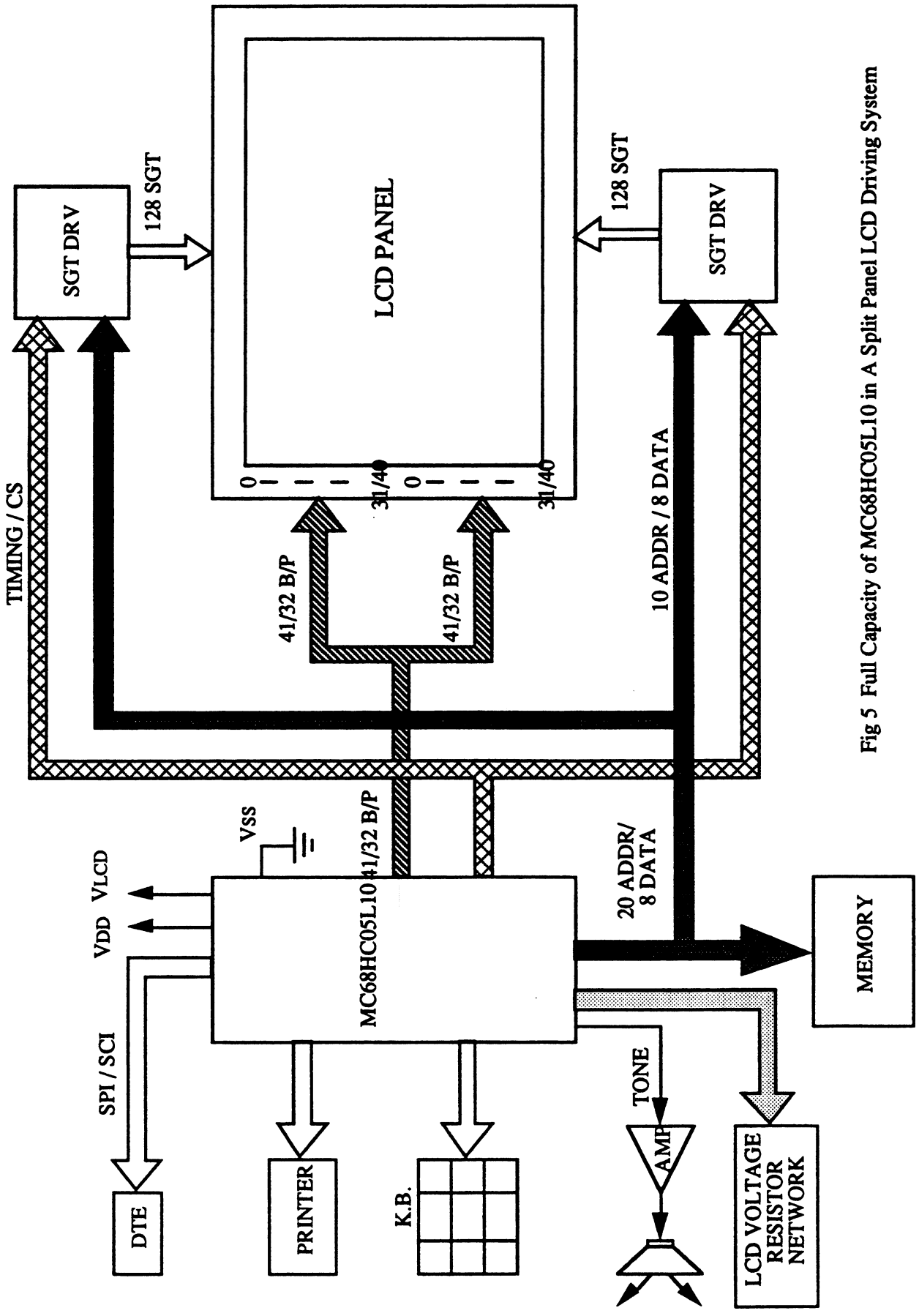
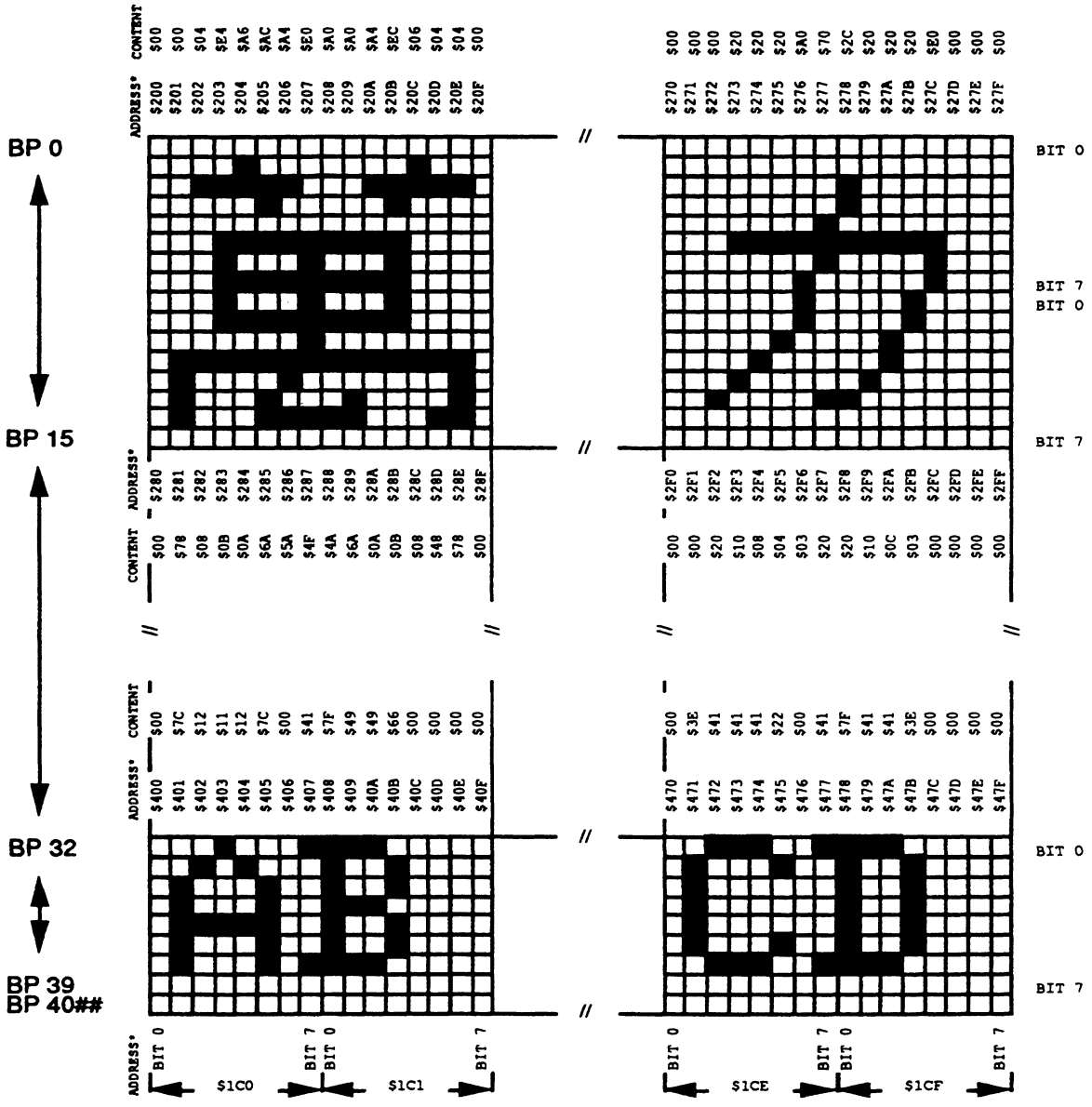


Fig 5 Full Capacity of MC68HC05L10 in A Split Panel LCD Driving System



Note: - * The display RAM address located in L10 for segment selected by $\overline{CS1}$.

- ## For annunciator feature.

Fig 6 Relation Between Display RAM Data in L10 and Display Pattern

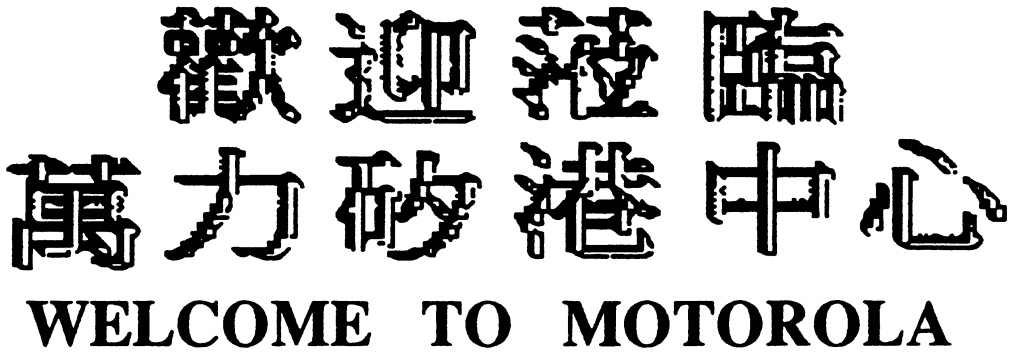


Fig 7 Display Pattern

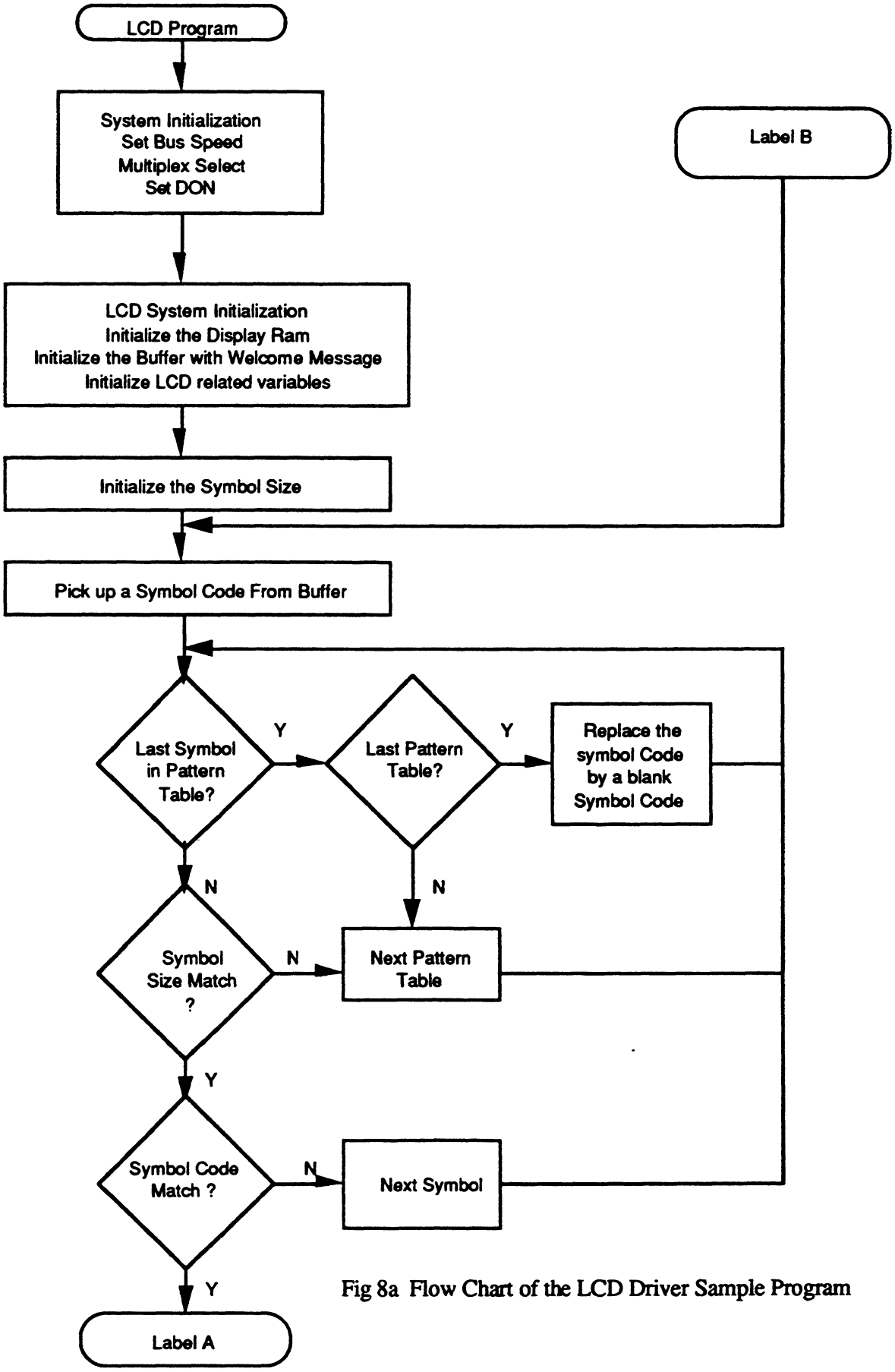


Fig 8a Flow Chart of the LCD Driver Sample Program

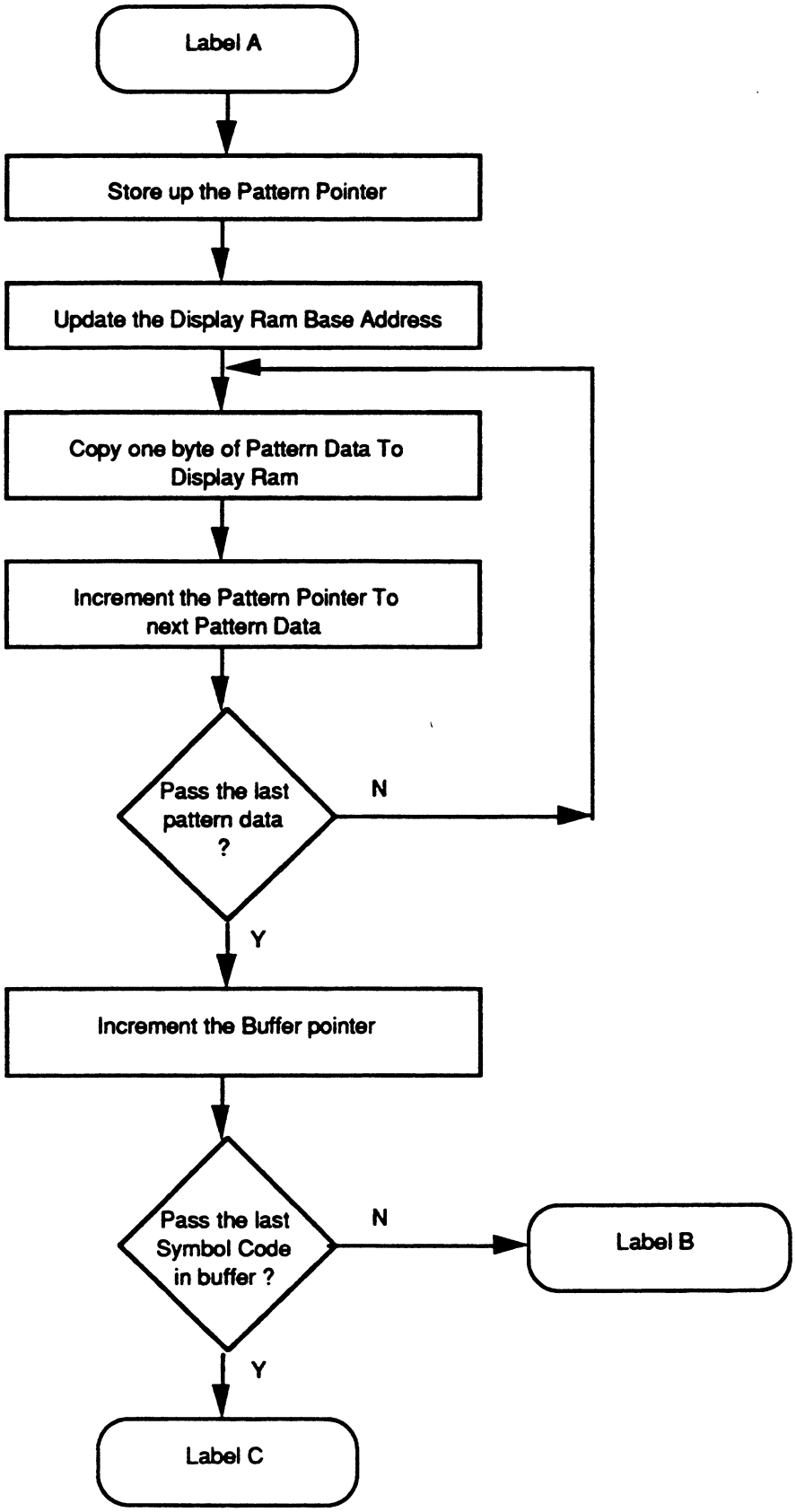


Fig 8b Flow Chart of the LCD Driver Sample Program

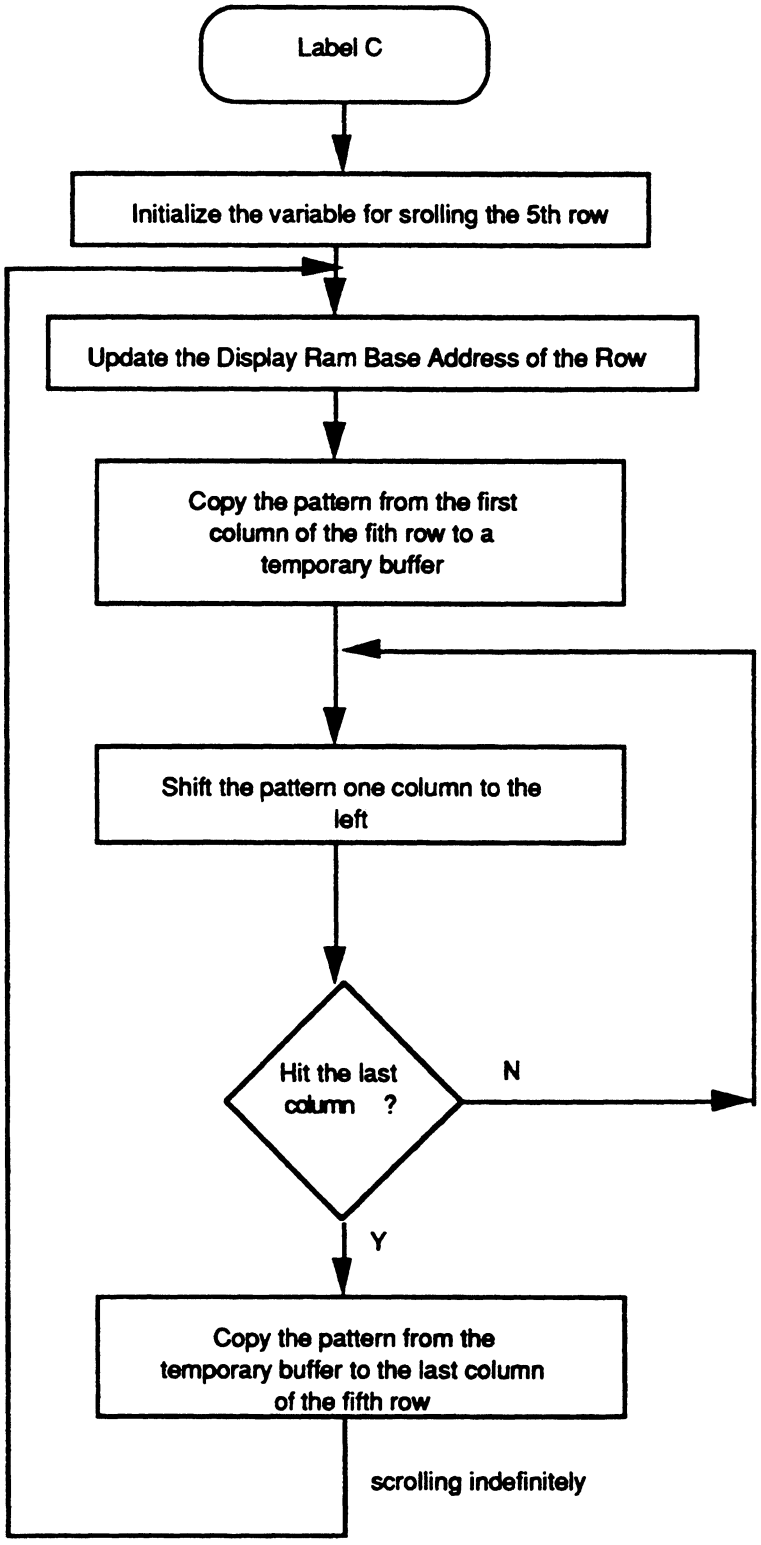


Fig 8c Flow Chart of the LCD Driver Sample Program

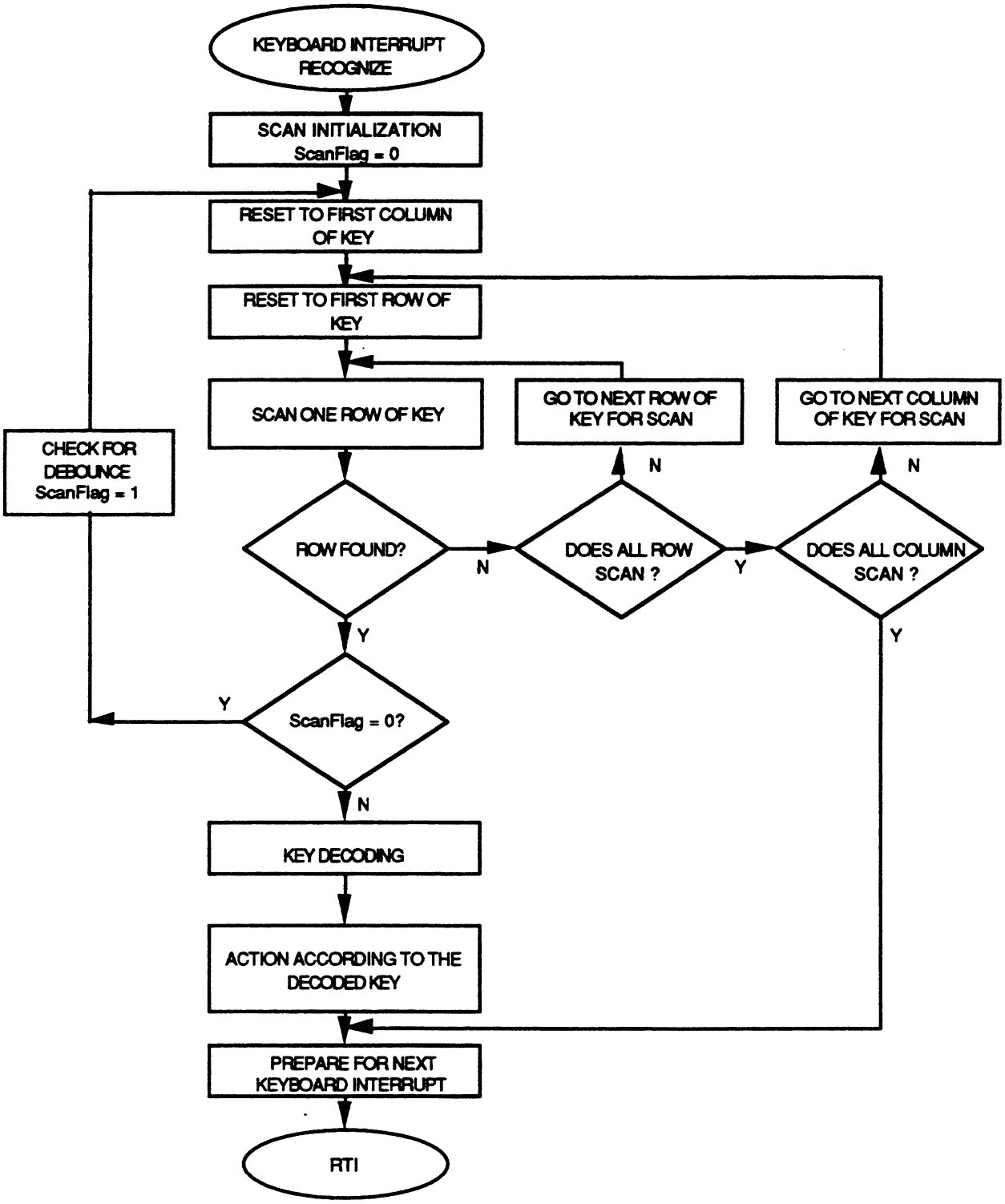


Fig 9 Flow Chart of the Keyscan Sample Program

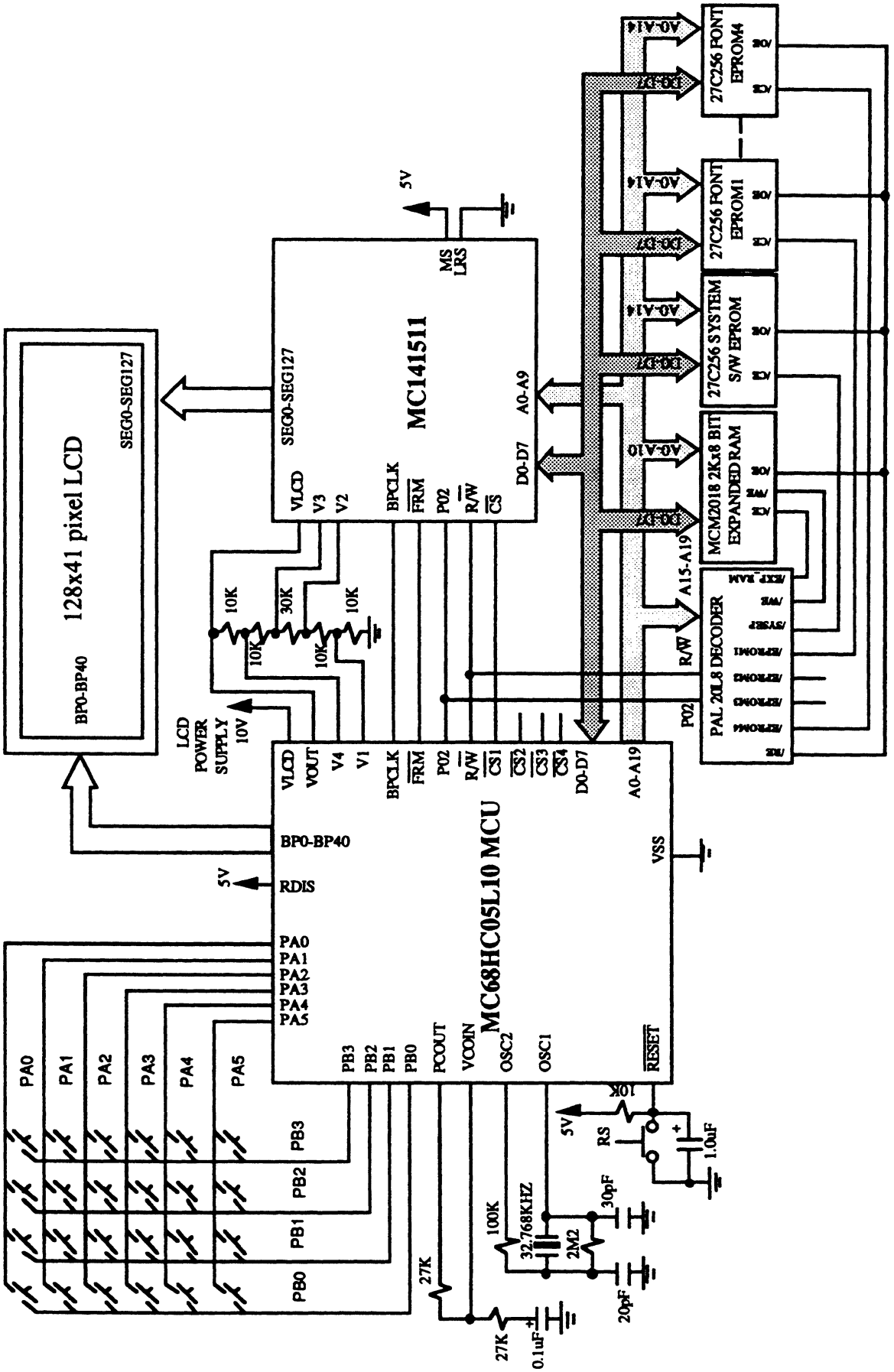


Fig 10 Application Circuitry for Keyscan Program, LCD Driver & Memory Management Unit Program

```
module L10APPLICATION;

title
'PAL20L8
R1.0/MAR26/91
THE ADDRESS DECODER FOR
THE L10 APPLICATION CIRCUITRY.';

L10APP device 'P20L8';


"declarations

        TRUE,FALSE = 1,0;
        H,L = 1,0;
        X,Z,C = .X.,.Z.,.C.;

        GND,VCC
                pin  12,24;

        "Input Port
        A8,A9,A10,A11,A12,A13,A14,A15,A16,A17,A18,A19
                pin  1,2,3,4,5,6,7,8,9,10,11,13;      "Address bus

        P02     pin  14;                              "System clock

        RW      pin  23;                              "Read/Write signal


        "Output Port
        _RE     pin  15;                              "Read Enable

        _WE     pin  16;                              "Write Enable

        _RAM    pin  17;                              "Expanded RAM Chip Enable
                                                       "$06000-$067FF

        _SYSEP  pin  18;                              "System S/W Eprom
                                                       "$00C00-$05FFF

        _EPROM1 pin  19;                              "Eprom 1 $08000-$0FFFF

        _EPROM2 pin  20;                              "Eprom 2 $10000-$17FFF

        _EPROM3 pin  21;                              "Eprom 3 $18000-$1FFFF

        _EPROM4 pin  22;                              "Eprom 4 $20000-$27FFF


        "Identifier Defintion
        Address = [A19,A18,A17,A16, A15,A14,A13,A12, A11,A10,A9,A8,
                   X,X,X,X,         X,X,X,X];
```

```
equations
        "Output Enable
        !_RE     = RW;


        "Write Enable
        !_WE     = P02 & !RW;


        "Expanded RAM CHIP ENABLE ( 2K x 8 bit RAM )
        !_RAM    = P02 & ( Address >= ^h06000 ) & ( Address <= ^h067FF );


        "_SYSEP  : SYSTEM S/W
        !_SYSEP  = P02 & ( Address >= ^h00C00 ) & ( Address <= ^h05FFF );


        "_EPROM1 : CHIP ENABLE OF FONT/DATA EPROM 1
        !_EPROM1 = P02 & ( Address >= ^h08000 ) & ( Address <= ^h0FFFF );


        "_EPROM2 : CHIP ENABLE OF FONT/DATA EPROM 2
        !_EPROM2 = P02 & ( Address >= ^h10000 ) & ( Address <= ^h17FFF );


        "_EPROM3 : CHIP ENABLE OF FONT/DATA EPROM 3
        !_EPROM3 = P02 & ( Address >= ^h18000 ) & ( Address <= ^h1FFFF );


        "_EPROM4 : CHIP ENABLE OF FONT/DATA EPROM 4
        !_EPROM4 = P02 & ( Address >= ^h20000 ) & ( Address <= ^h27FFF );
```

```
"Test Vectors

"Address Decoding Test Vector
test_vectors (
              [P02,A19,A18,A17,A16,A15,A14,A13,A12,A11,A10,A9,A8]
               ->[_RAM,_SYSEP,_EPROM1,_EPROM2,_EPROM3,_EPROM4]
             )

        "                                ~ ~ ~ ~
        "                              ~ E E E E
        "                              S P P P P
        "                            ~ Y R R R R
        "P A A A A A A A A A A       R S O O O O
        "0 1 1 1 1 1 1 1 1 1 A A     A E M M M M
        "2 9 8 7 6 5 4 3 2 1 0 9 8   M P 1 2 3 4

"RAM CHIP ENABLE : $06000-$067FF
        [H,L,L,L,L,L,H,H,L,L,X,X,X]->[L,H,H,H,H,H];

"SYSTEM EPROM CHIP ENABLE : $00C00-$06FFF
        [H,L,L,L,L,L,L,L,L,H,H,X,X]->[H,L,H,H,H,H];    "$00C00-$00FFF
        [H,L,L,L,L,L,L,L,L,H,X,X,X]->[H,L,H,H,H,H];    "$01000-$01FFF
        [H,L,L,L,L,L,L,L,H,L,X,X,X]->[H,L,H,H,H,H];    "$02000-$03FFF
        [H,L,L,L,L,L,L,H,L,X,X,X,X]->[H,L,H,H,H,H];    "$04000-$05FFF

"EPROM 1 CHIP ENABLE : $08000-$0FFFF
        [H,L,L,L,L,H,X,X,X,X,X,X,X]->[H,H,L,H,H,H];

"EPROM 2 CHIP ENABLE : $10000-$17FFF
        [H,L,L,L,H,L,X,X,X,X,X,X,X]->[H,H,H,L,H,H];

"EPROM 3 CHIP ENABLE : $18000-$1FFFF
        [H,L,L,L,H,H,X,X,X,X,X,X,X]->[H,H,H,H,L,H];

"EPROM 4 CHIP ENABLE : $20000-$27FFF
        [H,L,L,H,L,L,X,X,X,X,X,X,X]->[H,H,H,H,H,L];

"CHIP UN-SELECT
        [L,X,X,X,X,X,X,X,X,X,X,X,X]->[H,H,H,H,H,H];



"Read and Write Enable
test_vectors ( [RW,P02] ->[_RE,_WE] )
               [ H,  H] ->[  L,  H];
               [ H,  L] ->[  L,  H];
               [ L,  H] ->[  H,  L];
               [ L,  L] ->[  H,  H];


end L10APPLICATION;
```

```
M6805 Portable Cross Assembler  0.05   MS-DOS/PC-DOS  Page 1
 Mon Sep 09 11:26:57 1991
Command line:
 C:\PASM\PASM05.EXE -EQSUX -L keyscan.LST -O keyscan.OBJ keyscan.ASM
Options list:
ON  - b - Printing of macro definitions
ON  - c - Printing of macro calls
OFF - d - Placing of symbolic debugging information in COFF
ON  - e - Printing of macro expansions  (changed)
ON  - f - Printing of conditional directives
OFF - g - Printing of generated constants list
ON  - q - Expanding and printing of structured syntax  (changed)
ON  - s - Printing of symbol table  (changed)
ON  - u - Printing of conditional unassembled source  (changed)
ON  - x - Printing of cross reference table  (changed)
OFF - m - Suppress printing of error messages
ON  - w - Printing of warning messages
OFF - v - Suppress printing of updated status
OFF - y - Enabling of sgs extensions
ON  - o - Create object code
ON  -   - Formatting of source line listing
Create listing file - l - keyscan.LST
Change object file name - o - keyscan.OBJ

Xdefs:
  NONE

Xrefs:

RC00        RC01        RC02        RC03        RC10        RC11        RC12        RC13        RC20        RC21
RC22
RC23        RC30        RC31        RC32        RC33        RC40        RC41        RC42        RC43        RC50
RC51
RC52        RC53

Input file(s): keyscan.ASM (280 lines)

Output file: keyscan.OBJ
Listing file: keyscan.LST
```

```
M6805 Portable Cross Assembler  0.05  keyscan.ASM  Page 2
 Mon Sep 09 11:26:57 1991
Options - MD,MC,NOG,U,W,MEX,CL,FMT,O

LINE   S PC   OPCO OPERANDS S LABEL    MNEMO OPERANDS COMMENT
00001                        ********************************************************************************
00002                        *
00003                        *       FILE NAME:  KEYSCAN.ASM                 DATE: 27th Jan., 1991
00004                        *
00005                        *       FILE DESCRIPTION: KEYSCAN PROGRAM FOR MC68HC05L10
00006                        *                         KEYSCAN PROGRAM FOR M x N KEY PAD by PORTA and PORTB
00007                        *                         This program can be used for any M x N (8 x 8 max )
00008                        *                         keypad detection. User can run his/her own
00009                        *                         decoding scheme for his/her own application.
00010                        *
00011                        *       REVISION: 1.0
00012                        *
00013                        *       AUTHOR: JAMSON CHEUNG
00014                        *
00015                        *       Remark: This application is fully compiled by MOTOROLA PASM05
00016                        *               cross compiler
00017                        *
00018                        *       Program Latest Update:
00019                        *       Rev 1.0         Date: 30th Jan., 1991
00020                        *                       ORGINIAL RELEASE
00021                        *
00022
********************************************************************************
00023                        *
00024                        *
00025                                 OPT   NOP      SET NO PAGE HEADER FOR EACH PAGE
00026                        *
*
00027                        * KEYSCAN FOR THE MC68HC05L10 MCU
00028                        *
*
00029                                 OPT   MUL
00030                        *
00031                        *
00032                        *
00033                                 XREF  RC00,RC01,RC02,RC03
00034                                 XREF  RC10,RC11,RC12,RC13
00035                                 XREF  RC20,RC21,RC22,RC23
00036                                 XREF  RC30,RC31,RC32,RC33
00037                                 XREF  RC40,RC41,RC42,RC43
00038                                 XREF  RC50,RC51,RC52,RC53
00039
00040                        *
00041                        *       KEY PAD DEFINITION TABLE
00042                        *
00043  A 3000                         ORG   $3000
00044                        KEYDEF
00045  A 3000      00000000 X         FDB   RC00,RC01,RC02,RC03
00046  A 3008      00000000 X         FDB   RC10,RC11,RC12,RC13
00047  A 3010      00000000 X         FDB   RC20,RC21,RC22,RC23
00048  A 3018      00000000 X         FDB   RC30,RC31,RC32,RC33
00049  A 3020      00000000 X         FDB   RC40,RC41,RC42,RC43
00050  A 3028      00000000 X         FDB   RC50,RC51,RC52,RC53
00051                        *
00052                        * SYSTEM EQUATES
00053                        *
```

```
00054  A 3030       0007     A BIT7      EQU   $07
00055  A 3030       0006     A BIT6      EQU   $06
00056  A 3030       0005     A BIT5      EQU   $05
00057  A 3030       0004     A BIT4      EQU   $04
00058  A 3030       0003     A BIT3      EQU   $03
00059  A 3030       0002     A BIT2      EQU   $02
00060  A 3030       0001     A BIT1      EQU   $01
00061  A 3030       0000     A BIT0      EQU   $00
00062                         *
00063                         *
00064
********************************************************************************
00065                         *
00066                         *       I/O REGISTER DEFINITIONS
00067                         *
00068  A 3030       0000     A PORTA     EQU   $00       I/O PORT A OFFSET
00069  A 3030       0001     A PORTB     EQU   $01       I/O PORT B OFFSET
00070  A 3030       0002     A PORTC     EQU   $02       I/O PORT C OFFSET
00071  A 3030       0003     A PORTD     EQU   $03       I/0 PORT D OFFSET
00072  A 3030       0004     A PORTE     EQU   $04       I/O PORT E OFFSET
00073  A 3030       0005     A DDR       EQU   $05       DATA DIRECTION REGISTER OFFSET
00074  A 3030       0005     A DDRA      EQU   PORTA+DDR PORTA DDR
00075  A 3030       0006     A DDRB      EQU   PORTB+DDR PORTB DDR
00076  A 3030       0007     A DDRC      EQU   PORTC+DDR PORTC DDR
00077  A 3030       0009     A DDRE      EQU   PORTE+DDR PORTE DDR
00078                         *
00079                         *
00080                         *       CONTROL REGISTERS
00081                         *
00082  A 3030       001c     A CTRL$1C   EQU   $1C       CONTROL REGISTER $1C
00083  A 3030       0007     A TIMI      EQU   BIT7
00084  A 3030       0006     A CLKS      EQU   BIT6
00085  A 3030       0005     A IR1F      EQU   BIT5
00086  A 3030       0004     A IR2F      EQU   BIT4
00087  A 3030       0003     A PORTI     EQU   BIT3
00088  A 3030       0002     A SECF      EQU   BIT2
00089  A 3030       0001     A ALF       EQU   BIT1
00090  A 3030       0000     A RTCF      EQU   BIT0
00091
00092  A 3030       0026     A CTRL$26   EQU   $26       CONTROL REGISTER $26
00093  A 3030       0007     A TONE      EQU   BIT7
00094  A 3030       0006     A PLLI      EQU   BIT6
00095  A 3030       0005     A IR1O      EQU   BIT5
00096  A 3030       0004     A IR2O      EQU   BIT4
00097  A 3030       0003     A DON       EQU   BIT3
00098  A 3030       0002     A AUTO      EQU   BIT2
00099  A 3030       0001     A TONS      EQU   BIT1
00100  A 3030       0000     A MS        EQU   BIT0
00101
00102  A 3030       0027     A CTRL$27   EQU   $27       CONTROL REGISTER $27
00103  A 3030       0006     A FS1       EQU   BIT6
00104  A 3030       0005     A FS0       EQU   BIT5
00105  A 3030       0004     A KEYE      EQU   BIT4
00106  A 3030       0002     A SECE      EQU   BIT2
00107  A 3030       0001     A ALE       EQU   BIT1
00108  A 3030       0000     A RTCE      EQU   BIT0
00109
00110                         ********************************************************************************
00112                         ********************************************************************************
00113  A 0050                           org   $50       Variable Segment
00114  A 0050 cd    ffff     A KEYSUB    JSR   $FFFF
00115  A 0053       0051     A KEYADD    EQU   *-2
```

```
00116  A 0053 81                        RTS
00117
00118  A 0054      01       A KEYFLAG   RMB   1        Keyboard flag
00119  A 0055      01       A KEYA      RMB   1
00120  A 0056      01       A KEYB      RMB   1
00121  A 0057      01       A COLX      RMB   1
00122  A 0058      01       A ROWX      RMB   1
00123
00124                         *
00125                         * KEYPAD DEFINITION
00126                         * THE MAX. NUMBER OF DEFINITION FOR THIS KEYPAD IS 64
00127                         * SUPPOSE ALL THE LOWER ORDER OF THE PORT ARE USED UP FIRST
00128                         *
00129  A 0059      0005     A KEYNUMA   EQU   5        NUMBER OF PORT A ARE USED FOR KEYPAD
00130  A 0059      0004     A KEYNUMB   EQU   4        NUMBER OF PORT B ARE USED FOR KEYPAD
00131  A 0059      001f     A MASKPA    EQU   $FF!>(8-KEYNUMA) MASK FOR MASKING PORTA DATA
00132  A 0059      000f     A MASKPB    EQU   $FF!>(8-KEYNUMB) MASK FOR MASKING PORTB DATA
00133
00134
00135
00136                         ********************************************************************************
00138                         ********************************************************************************
00139  A 0de2                           ORG   $0DE2
00140  A 0de2 ad   06    0dea MAIN      BSR   INIT
00141  A 0de4 ad   10    0df6           BSR   KBINIT
00142  A 0de6 9a                        CLI
00143  A 0de7 8e                STOP    STOP
00144  A 0de8 20   fd    0de7           BRA   STOP
00145
00146                         *
00147                         * System Initialization
00148                         *
00149  A 0dea 1c   1c       A INIT      BSET  CLKS,CTRL$1C SELECT PLL CLOCK FOR CPU
00150  A 0dec 1c   27       A           BSET  FS1,CTRL$27 SELECT 7.3728 MHz PLL OUTPUT CLOCK FREQ
00151  A 0dee 1a   27       A           BSET  FS0,CTRL$27
00152  A 0df0 0d   26fd  0df0           BRCLR PLLI,CTRL$26,* WAIT UNTIL PLL CLOCK IS STABLE
00153  A 0df3 3f   00       A           CLR   PORTA    CLEAR ANY UNNECCESSARY KEYBOARD INTERRUPT
00154  A 0df5 81                        RTS
00155
00156
00157                         *
00158                         * Keyboard System Initialization
00159                         *
00160  A 0df6 3f   05       A KBINIT    CLR   DDRA     SELECT PORTA AS INPUT PORT
00161  A 0df8 3f   00       A           CLR   PORTA    PREPARE FOR KEYBOARD INTERRUPT
00162  A 0dfa a6   0f       A           LDA   #MASKPB
00163  A 0dfc b7   06       A           STA   DDRB     SELECT PORTB AS OUTPUT PORT
00164  A 0dfe 3f   01       A           CLR   PORTB
00165  A 0e00 81                        RTS
00166
00167                         ********************************************************************************
00169                         ********************************************************************************
00170                         *
00171                         * Keyboard Interrupt Service Routine
00172                         *
00173                         KEYISR
00174  A 0e01 3f   54       A           CLR   KEYFLAG
00175  A 0e03 a6   ff       A KEYSCAN   LDA   #$FF
00176  A 0e05 b7   01       A           STA   PORTB
00177  A 0e07 17   01       A           BCLR  KEYNUMB-1,PORTB START CHECKING FROM KEYPAD WITH HIGHEST PORTB
00178  A 0e09 b6   00       A REPEAT    LDA   PORTA    START SCANNING FROM THE BOTTOM ROW
00179  A 0e0b a4   1f       A           AND   #MASKPA
```

```
00180  A 0e0d a1   1f       A          CMP   #MASKPA
00181  A 0e0f 26   06    0e17          BNE   GOTIT     KEY FOUND IS Z=1
00182  A 0e11 34   01       A          LSR   PORTB     OTHERWISE, FOR NEXT COLUMN
00183  A 0e13 25   f4    0e09          BCS   REPEAT
00184  A 0e15 20   1b    0e32          BRA   DONE
00185
00186  A 0e17 00   5414  0e2e GOTIT    BRSET BIT0,KEYFLAG,NOSAVE
00187
00188  A 0e1a a6   0f       A          LDA   #MASKPB
00189  A 0e1c 43                      COMA
00190  A 0e1d ba   56       A          ORA   KEYB
00191  A 0e1f b7   56       A          STA   KEYB      SAVE KEYB
00192
00193  A 0e21 a6   1f       A          LDA   #MASKPA
00194  A 0e23 43                      COMA
00195  A 0e24 ba   55       A          ORA   KEYA
00196  A 0e26 b7   55       A          STA   KEYA      SAVE KEYA
00197
00198  A 0e28 ad   0b    0e35          BSR   DBOUNC    CHECK FOR NOISE
00199  A 0e2a 10   54       A          BSET  BIT0,KEYFLAG SET KEYBOARD FLAG
00200  A 0e2c 20   d5    0e03          BRA   KEYSCAN
00201
00202  A 0e2e ad   05    0e35 NOSAVE   BSR   DBOUNC    PAUSE
00203  A 0e30 ad   0e    0e40          BSR   DECODE    GO TO USER KEY DECODE ROUTINE
00204  A 0e32 ad   c2    0df6 DONE     BSR   KBINIT    PREPARE THE SCAN LINES FOR NEXT INTERRUPT
00205  A 0e34 80                      RTI
00206                          *
00207                          * DEBOUNCE ROUTINE
00208                          * DELAY FOR A SHORT PERIOD OF TIME
00209                          *
00210  A 0e35 a6   5a       A DBOUNC   LDA   #90
00211  A 0e37 ae   ff       A AGAIN1   LDX   #$FF
00212  A 0e39 5a                AGAIN    DECX
00213  A 0e3a 26   fd    0e39          BNE   AGAIN
00214  A 0e3c 4a                      DECA
00215  A 0e3d 26   f8    0e37          BNE   AGAIN1
00216  A 0e3f 81                      RTS
00217
00218
00219
00220
00221                          *
00222                          * DECODE ROUTINE
00223                          *
00224                          DECODE
00225  A 0e40 b6   55       A          LDA   KEYA      DETERMINE WHICH COLUMN IS DETECTED
00226  A 0e42 ae   ff       A          LDX   #$FF
00227  A 0e44 5c                CHKA     INCX
00228  A 0e45 44                      LSRA
00229  A 0e46 25   fc    0e44          BCS   CHKA
00230  A 0e48 bf   58       A          STX   ROWX      ROWX <= 7
00231
00232  A 0e4a b6   56       A          LDA   KEYB      DETERMINE WHICH ROW IS DETECTED
00233  A 0e4c ae   ff       A          LDX   #$FF
00234  A 0e4e 5c                CHKB     INCX
00235  A 0e4f 44                      LSRA
00236  A 0e50 25   fc    0e4e          BCS   CHKB
00237  A 0e52 bf   57       A          STX   COLX      COLX <= 7
00238
00239                          * CALCULATE THE OFFSET FROM KEYDEF TABLE WHICH IS EQUAL TO
00240                          * (ROWX * NUMBER OF COLUMN * 2) + (COLX * 2)
00241  A 0e54 b6   58       A          LDA   ROWX
```

```
00242  A 0e56 ae   04       A          LDX   #KEYNUMB
00243  A 0e58 58                       LSLX            MULTIPLE BY 2
00244  A 0e59 42                       MUL             PRODUCT <= 128 ( FOR 8 x 8 PAD )
00245
00246  A 0e5a bb   57       A          ADD   COLX
00247  A 0e5c bb   57       A          ADD   COLX      2 BYTE ADDRESS IN THE KEYDEF TABLE
00248                        *                            A = OFFSET OF THE ADDRESS FROM BEGINNING OF
00249                        *                            THE KEYDEF TABLE
00250  A 0e5e 97                       TAX
00251  A 0e5f d6   3000     A          LDA   KEYDEF,X LOAD THE UPPER BYTE OF THE ADDRESS
00252  A 0e62 b7   51       A          STA   KEYADD
00253  A 0e64 d6   3001     A          LDA   KEYDEF+1,X LOAD THE LOWER BYTE OF THE ADDRESS
00254  A 0e67 b7   52       A          STA   KEYADD+1
00255  A 0e69 bd   50       A          JSR   KEYSUB    EXECUTE RAM SUBROUTINE
00256  A 0e6b 81                       RTS
00257
00258
00259                        *
00260                        * DUMMY INTERRUPT SERVICE ROUTINE
00261                        *
00262                        SPIISR
00263                        RTCISR
00264                        SCIISR
00265                        TIMISR
00266                        EIRQISR
00267  A 0e6c 80             SWIISR    RTI
00268
00269
00270  A 0c00                          ORG   $0C00
00271  A 0c00       0e6c    A SPIIRQ   FDB   SPIISR    SPI INTERRUPT VECTOR
00272  A 0c02       0e6c    A RTCIRQ   FDB   RTCISR    REAL TIME CLOCK INTERRUPT VECTOR
00273  A 0c04       0e6c    A SCIIRQ   FDB   SCIISR    SCI INTERRUPT VECTOR
00274  A 0c06       0e6c    A TIMIRQ   FDB   TIMISR    TIMER INTERRUPT VECTOR
00275  A 0c08       0e01    A KEYIRQ   FDB   KEYISR    KEYBOARD INTERRUPT VECTOR
00276  A 0c0a       0e6c    A EIRQ     FDB   EIRQISR   EXTERNAL INTERRUPT VECTOR
00277  A 0c0c       0e6c    A SWIVCT   FDB   SWIISR    SOFTWARE INTERRUPT VECTOR
00278  A 0c0e       0de2    A          FDB   MAIN      RESET INTERRUPT VECTOR
00279                                   END
00280



Total number of errors: 0
Total number of warnings: 0
Total number of lines: 280


Number of bytes in section ASCT: 212

Number of bytes in program: 212

           CROSS REFERENCE TABLE
NAME     ATTRB S VALUE P:LINE LINE1....N

RC00       XREF X 0000               45
RC01       XREF X 0000               45
RC02       XREF X 0000               45
RC03       XREF X 0000               45
RC10       XREF X 0000               46
RC11       XREF X 0000               46
RC12       XREF X 0000               46
RC13       XREF X 0000               46
```

```
RC20      XREF X 0000              47
RC21      XREF X 0000              47
RC22      XREF X 0000              47
RC23      XREF X 0000              47
RC30      XREF X 0000              48
RC31      XREF X 0000              48
RC32      XREF X 0000              48
RC33      XREF X 0000              48
RC40      XREF X 0000              49
RC41      XREF X 0000              49
RC42      XREF X 0000              49
RC43      XREF X 0000              49
RC50      XREF X 0000              50
RC51      XREF X 0000              50
RC52      XREF X 0000              50
RC53      XREF X 0000              50
```

```
M6805 Portable Cross Assembler  0.05   MS-DOS/PC-DOS  Page 1
 Tue Sep 10 16:30:55 1991
Command line:
 C:\PASM\PASM05.EXE -EQSUX -L DPROG.LST -O DPROG.OBJ DPROG.ASM
Options list:
ON  - b - Printing of macro definitions
ON  - c - Printing of macro calls
OFF - d - Placing of symbolic debugging information in COFF
ON  - e - Printing of macro expansions  (changed)
ON  - f - Printing of conditional directives
OFF - g - Printing of generated constants list
ON  - q - Expanding and printing of structured syntax  (changed)
ON  - s - Printing of symbol table  (changed)
ON  - u - Printing of conditional unassembled source  (changed)
ON  - x - Printing of cross reference table  (changed)
OFF - m - Suppress printing of error messages
ON  - w - Printing of warning messages
OFF - v - Suppress printing of updated status
OFF - y - Enabling of sgs extensions
ON  - o - Create object code
ON  -   - Formatting of source line listing
Create listing file - l - DPROG.LST
Change object file name - o - DPROG.OBJ

Xdefs:
  NONE

Xrefs:
  NONE

Input file(s): DPROG.ASM (821 lines)

Output file: DPROG.OBJ
Listing file: DPROG.LST
```

```
M6805 Portable Cross Assembler  0.05  DPROG.ASM  Page 2
 Tue Sep 10 16:30:55 1991
Options - MD,MC,NOG,U,W,MEX,CL,FMT,O

LINE   S PC   OPCO OPERANDS S LABEL    MNEMO OPERANDS COMMENT
00001                       ********************************************************************************
00002                       *
00003                       * FILE NAME:  DPROG.ASM                    DATE: 8th Feb., 1991
00004                       *
00005                       * FILE DESCRIPTION: MC68HC05L10 MCU DEMONSTRATION PROGRAM
00006                       *                   a) ONE SLAVE LCD DRIVER PROGRAM
00007                       *                   b) USING MMU TO ACCESS THE MEMORY MAP THAT
00008                       *                      GREATER THAN 64KByte MEMORY
00009                       *
00010                       * REVISION: 1.0
00011                       *
00012                       * AUTHOR: JAMSON CHEUNG
00013                       *
00014                       * REMARK: THIS FILE CONTAINS THE SOURCE PROGRAM AND SOME BUFFERS
00015                       *         THIS PROGRAM MUST BE LOCATED IN BANK 0
00016                       *         THE SYMBOL PATTERN DATA FILES ARE
00017                       *                 a) DROM1.ASM
00018                       *                 b) DROM2.ASM
00019                       *
00020                       ********************************************************************************
00021                       *
00022                       *
00023                                OPT   NOP      SET NO PAGE HEADER FOR EACH PAGE
00024                       *
00025                       * ADDRESS LOCATION EQUATE
00026                       *
00027                       *
00028                       *       EXTERNAL EXPANDED RAM SECTION   $6000 - $67FF
00029                       *
00030  P 0000      6000     A EXTRAM   EQU   $6000
00031                       *
00032                       *
00033                       * INTERNAL DATA/PROGRAMMING ROM   $0DE2 - $3FFF
00034                       *
00035  P 0000      0de2     A INTROM   EQU   $0DE2
00036
00037                       *
00038                       * EXTERNAL FONT DATA ROM region        $10000-$14FFF
00039                       *
00040  P 0000      8000     A OFF6x8   EQU   $8000    COMBINE WITH MMU (PHYSICAL ADD=$10000)
00041  P 0000      c000     A OFF16x16 EQU   $C000    COMBINE WITH MMU (PHYSICAL ADD=$14000)
00042
00043  P 0000      8000     A TAB1     EQU   OFF6x8
00044  P 0000      8100     A TAB2     EQU   OFF6x8+$100
00045  P 0000      c000     A TAB3     EQU   OFF16x16
00046  P 0000      c100     A TAB4     EQU   OFF16x16+$100
00047                       *
00048                       * SYSTEM EQUATES
00049                       *
00050  P 0000      0007     A BIT7     EQU   $07
00051  P 0000      0006     A BIT6     EQU   $06
00052  P 0000      0005     A BIT5     EQU   $05
00053  P 0000      0004     A BIT4     EQU   $04
00054  P 0000      0003     A BIT3     EQU   $03
00055  P 0000      0002     A BIT2     EQU   $02
00056  P 0000      0001     A BIT1     EQU   $01
```

```
00057  P 0000      0000     A BIT0     EQU   $00
00058                        *
00059                        *       CONTROL REGISTERS
00060                        *
00061  P 0000      001c     A CTRL$1C  EQU   $1C      CONTROL REGISTER $1C
00062  P 0000      0007     A TIMI     EQU   BIT7
00063  P 0000      0006     A CLKS     EQU   BIT6
00064  P 0000      0005     A IR1F     EQU   BIT5
00065  P 0000      0004     A IR2F     EQU   BIT4
00066  P 0000      0003     A PORTI    EQU   BIT3
00067  P 0000      0002     A SECF     EQU   BIT2
00068  P 0000      0001     A ALF      EQU   BIT1
00069  P 0000      0000     A RTCF     EQU   BIT0
00070
00071  P 0000      0026     A CTRL$26  EQU   $26      CONTROL REGISTER $26
00072  P 0000      0007     A TONE     EQU   BIT7
00073  P 0000      0006     A PLLI     EQU   BIT6
00074  P 0000      0005     A IR1O     EQU   BIT5
00075  P 0000      0004     A IR2O     EQU   BIT4
00076  P 0000      0003     A DON      EQU   BIT3
00077  P 0000      0002     A AUTO     EQU   BIT2
00078  P 0000      0001     A TONS     EQU   BIT1
00079  P 0000      0000     A MS       EQU   BIT0
00080
00081  P 0000      0027     A CTRL$27  EQU   $27      CONTROL REGISTER $27
00082  P 0000      0006     A FS1      EQU   BIT6
00083  P 0000      0005     A FS0      EQU   BIT5
00084  P 0000      0004     A KEYE     EQU   BIT4
00085  P 0000      0002     A SECE     EQU   BIT2
00086  P 0000      0001     A ALE      EQU   BIT1
00087  P 0000      0000     A RTCE     EQU   BIT0
00088
00089                        *
00090                        *       MMU REGISTERS
00091                        *
00092  P 0000      0022     A MMUCB    EQU   $22      MMU COMMON BANK REGISTER
00093  P 0000      0020     A MMUPA1   EQU   $20      MMU BANK 1 OFFSET REGISTER
00094  P 0000      0021     A MMUPA2   EQU   $21      MMU BANK 2 OFFSET REGISTER
00095                        *
00096                        *
00097  P 0000      0080     A LCDCOL   EQU   128      LCD SIZE = LCDCOL x LCDROW = 128 x 40
00098  P 0000      0028     A M41LCDR  EQU   40       LCDROW = M41LCDR (41 mux) or M32LCDR (32 mux)
00099  P 0000      0020     A M32LCDR  EQU   32
00100  P 0000      00fa     A BUFSIZE  EQU   (LCDCOL/5)*(M41LCDR/8)*2
00101  P 0000      0020     A BLANK    EQU   $20      BLANK CODE
00102
00103
00104                        ********************************************************************************
00106                        ********************************************************************************
00107                        **
00108                        **        VARIOUS TABLE/DATA BUFFER DEFINITION
00109                        **
00110                        ********************************************************************************
00111
00112                        ********************************************************************************
00113                        **
00114                        **      SYMBOL PATTERN TABLE OFFSET DEFINITION
00115                        **
00116                        ********************************************************************************
00117  P 0000      0000     A XSIZE    EQU   0
00118  P 0000      0001     A YSIZE    EQU   1
00119  P 0000      0003     A TABRECSIZE EQU 3        2 BYTE SYMBOL CODE, 1 BYTE OFFSET
```

```
00120
00121                          ***************************************************************************
00123  A 0de2                           ORG   INTROM
00124                          ***************************************************************************
00125                          **
00126                          **      P_TAB - TABLE OF PATTERN SYMBOL PATTERN TABLE POINTER
00127                          **
00128                          **      EACH RECORD HAS 5 BYTE (P_TABRECSIZE)
00129                          **
00130                          **      P_TAB RECORD FORMAT
00131                          **              FIRST 2 BYTE : SYMBOL PATTERN TABLE POINTER
00132                          **              3rd BYTE     : CORRESPONDING MMU COMMON BANK DATA
00133                          **              4th BYTE     : CORRESPONDING MMU OFFSET REGISTER 1 DATA
00134                          **              5th BYTE     : CORRESPONDING MMU OFFSET REGISTER 2 DATA
00135                          **
00136                          ***************************************************************************
00137                            P_TAB
00138  A 0de2      8000      A P_REC0   FDB   TAB1      RECORD #0
00139  A 0de4      c8        A          FCB   $C8       COMMON BANK VALUE
00140  A 0de5      08        A          FCB   8         POSA1 VALUE
00141  A 0de6      00        A          FCB   0         POSA2 VALUE
00142                            E_REC0
00143  A 0de7      8100      A P_REC1   FDB   TAB2      RECORD #1
00144  A 0de9      c8        A          FCB   $C8       COMMON BANK VALUE
00145  A 0dea      08        A          FCB   8         POSA1 VALUE
00146  A 0deb      00        A          FCB   0         POSA2 VALUE
00147                            E_REC1
00148  A 0dec      c000      A P_REC2   FDB   TAB3      RECORD #2
00149  A 0dee      c8        A          FCB   $C8       COMMON BANK VALUE
00150  A 0def      00        A          FCB   0         POSA1 VALUE
00151  A 0df0      08        A          FCB   8         POSA2 VALUE
00152                            E_REC2
00153  A 0df1      c100      A P_REC3   FDB   TAB4      RECORD #3
00154  A 0df3      c8        A          FCB   $C8       COMMON BANK VALUE
00155  A 0df4      00        A          FCB   0         POSA1 VALUE
00156  A 0df5      08        A          FCB   8         POSA2 VALUE
00157                            E_REC3
00158  A 0df6      60fa      A P_RECB   FDB   TABLANK   RECORD BLANK
00159  A 0df8      ff        A          FCB   $FF       COMMON BANK VALUE
00160  A 0df9      00        A          FCB   0         POSA1 VALUE
00161  A 0dfa      00        A          FCB   0         POSA2 VALUE
00162                            E_RECB
00163  A 0dfb      0000      A P_RECE   FDB   00        LAST RECORD INDICATOR
00164  A 0dfd      00        A          FCB   00        COMMON BANK VALUE
00165  A 0dfe      00        A          FCB   0         POSA1 VALUE
00166  A 0dff      00        A          FCB   0         POSA2 VALUE
00167                            E_RECE
00168  A 0e00      0e00      A P_END    EQU   *
00169
00170                          ***************************************************************************
00171                          **
00172                          **        P_TAB TABLE OFFSET DEFINITION
00173                          **
00174                          ***************************************************************************
00175
00176  A 0e00      0002      A P_OFFCB  EQU   2         P_TAB COMMON BANK DATA POINTER
00177  A 0e00      0003      A P_OFFPA1 EQU   3         P_TAB OFFSET REGISTER 1 DATA POINTER
00178  A 0e00      0004      A P_OFFPA2 EQU   4         P_TAB OFFSET REGISTER 2 DATA POINTER
00179  A 0e00      0005      A P_RECSIZE EQU  E_REC0-P_REC0
00180
00181
00182
```

```
00183                          ********************************************************************************
00184                          **
00185                          **      WELCOME BUFFER
00186                          **
00187                          ********************************************************************************
00188
00189  A 0e00      a140a140 A WELCOME  FDB   $A140,$A140,$C577,$AAEF,$BB59,$C17B,$A140,$A140
00190  A 0e10      a140b855 A          FDB   $A140,$B855,$A44F,$AABF,$B4E4,$A4A4,$A4DF,$A140
00191  A 0e20      00200057 A          FDB   ' ','W','E','L','C','O','M','E',' ','T','O',' '
00192  A 0e38      004d004f A          FDB   'M','O','T','O','R','O','L','A',' ',
00193  A 0e4c      0000     A          FDB   $00
00194  A 0e4e      0e4e     A WELEND   EQU   *
00195
00196                          ********************************************************************************
00197                          **
00198                          **        EXTERNAL RAM REGION - BUFFER DEFINITIONS
00199                          **
00200                          ********************************************************************************
00201  A 6000                          ORG   EXTRAM
00202  A 6000      fa       A BUF      RMB   BUFSIZE
00203
00204  A 60fa      00       A TABLANK  FCB   0
00205  A 60fb      00       A          FCB   0
00206  A 60fc      0020     A          FDB   BLANK    BLANK CODE NUMBER
00207  A 60fe      05       A          FCB   SYMBLANK-TABLANK
00208                        SYMBLANK
00209  A 60ff      fb       A          RMB   $100-(*-TABLANK)
00210  A 61fa      61fa     A TABLKE   EQU   *
00211
00212
00213
00214
00215
00216
00217
00218                          ********************************************************************************
00220                          ********************************************************************************
00221                          **
00222                          **        RAM SECTION - VARIABLE DEFINITIONS
00223                          **
00224                          ********************************************************************************
00225  A 0050                          ORG   $50
00226  A 0050 d6   ffff     A LDD      LDA   $FFFF,X
00227  A 0053 81                       RTS
00228  A 0054 d6   ffff     A MVV      LDA   $FFFF,X
00229  A 0057 d7   ffff     A STT      STA   $FFFF,X
00230  A 005a 81                       RTS
00231
00232  A 005b      0051     A SOURCE   EQU   LDD+1    THE 2-BYTE INDEX OF LDD RAM SUBROUTINE
00233  A 005b      0055     A MSRC     EQU   MVV+1    THE 2-BYTE INDEX OF MVV RAM SUBROUTINE
00234  A 005b      0058     A DESTINE  EQU   STT+1    THE 2-BYTE INDEX OF STT RAM SUBROUTINE
00235
00236  A 005b      01       A BUFPTR   RMB   1        BUFFER POINTER
00237  A 005c      01       A LCDPTR   RMB   1        LCD COLUMN POINTER
00238  A 005d      01       A TABPTR   RMB   1        SYMBOL PATTERN POINTER
00239  A 005e      01       A CHARROW  RMB   1        CHARSIZE = CHARROW x CHARCOL
00240  A 005f      01       A CHARCOL  RMB   1
00241  A 0060      01       A CHARBYT  RMB   1        NUMBER OF BYTE TO REPRESENT ONE SYMBOL
00242  A 0061      01       A COUNT    RMB   1        SYMBOL PATTERN SIZE COUNTER
00243  A 0062      02       A DEST     RMB   2        TEMPORARY 2 BYTE INDEX STORAGE
00244  A 0064      01       A TPTR     RMB   1        TEMPORARY SYMBOL PATTERN POINTER
00245  A 0065      01       A FOUND    RMB   1        FLAG FOR LOCATING THE SYMBOL PATTERN
```

```
00246  A 0066       01       A P_TPTR   RMB   1        SYMBOL PATTERN TABLE POINTER
00247  A 0067       02       A CHAR     RMB   2        CURRENT SYMBOL CODE
00248  A 0069       01       A TEMP     RMB   1        TEMPORARY VARIABLE 1
00249  A 006a       02       A TABCODE  RMB   2        TEMPORARY VARIABLE 2
00250  A 006c       03       A MMU      RMB   3        TEMPORARY STORAGE FOR MMU REGISTER
00251  A 006f       01       A DISPROW  RMB   1        CURRENT DISPLAY ROW
00252                          *                         (0..4 for 41 MUX;0..3 for 32 MUX )
00253  A 0070       01       A LCDROW   RMB   1        EQUAL TO 40 FOR 41 MUX; 32 FOR 32 MUX
00254
00255                          ********************************************************************************
00257                          ********************************************************************************
00258                          **
00259                          **        Name:   BLOCK - TEMPORARY BUFFER FOR SCROLLING
00260                          **
00261                          ********************************************************************************
00262  A 0100                           ORG   $0100
00263  A 0100       20       A BLOCK    RMB   32       TEMPORARY BUFFER ( 32 BYTE )
00264
00265                          ********************************************************************************
00267  A 1c30                           ORG   INTROM+WELEND
00268                          ********************************************************************************
00269                          **
00270                          **        Name:   RESET - MAIN PROGRAM
00271                          **
00272                          ********************************************************************************
00273  A 1c30 ad    0a    1c3c RESET    BSR   SYSINIT
00274  A 1c32 ad    1d    1c51         BSR   LCDINIT
00275  A 1c34 cd    1cc0     A         JSR   DISPBUF  DISPLAY 2 LINE OF CHINESE CHARACTERS
00276                          *                                  AND THEN A LINE OF ENGLISH MESSAGE
00277  A 1c37 cd    1e20     A KEEP_S   JSR   SCROLL5  SCROLLING THE 5th ROW
00278  A 1c3a 20    fb    1c37         BRA   KEEP_S   KEEP SCROLLING
00279
00280
00281                          ********************************************************************************
00282                          **
00283                          **        Name:   SYSINIT - SYSTEM INITIALIZATION
00284                          **
00285                          **        Return: A and X will be destroyed
00286                          **
00287                          ********************************************************************************
00288                          SYSINIT
00289
00290                          * SET UP BUS FREQUENCY
00291
00292  A 1c3c 1d    1c       A         BCLR  CLKS,CTRL$1C SELECT 32KHz CLOCK FOR CPU
00293  A 1c3e 1c    27       A         BSET  FS1,CTRL$27 SELECT 2.4576 MHz FOR PLL
00294  A 1c40 1b    27       A         BCLR  FS0,CTRL$27
00295
00296                          *        2 msec delay loop is necessary for
00297                          *               1) clearing the previous PLLI bit and
00298                          *               2) setting the current PLLI if PLL is locked
00299                          *
00300                          *        Current Bus speed is 16KHz
00301                          *        2 msec is equivalent to 32 bus cycle = 2 + 5*(3+3) cycle
00302                          *
00303  A 1c42 a6    05       A         LDA   #$5      2 cycle
00304  A 1c44 4a              PLLD      DECA          3 cycle
00305  A 1c45 26    fd    1c44         BNE   PLLD     DELAY COUNTER, 3 cycle
00306
00307  A 1c47 0d    26fd  1c47         BRCLR PLLI,CTRL$26,* WAIT UNTIL PLL CLOCK IS STABLE
00308  A 1c4a 1c    1c       A         BSET  CLKS,CTRL$1C SET THE BUS CLOCK = PLL
00309
```

```
00310  A 1c4c 16   26       A          BSET  DON,CTRL$26 TURN ON LCD
00311  A 1c4e 10   26       A          BSET  MS,CTRL$26 SELECT 1:41 MUX
00312
00313  A 1c50 81                       RTS
00314
00315                       ********************************************************************************
00316                       **
00317                       **        Name:   LCDINIT - LCD SUBSYSTEM INITIALIZATION
00318                       **
00319                       **        Return: A and X will be destroyed
00320                       **
00321                       ********************************************************************************
00322                       LCDINIT
00323  A 1c51 a6   10       A          LDA   #16      DISPLAY THE WELCOME MESSAGE
00324  A 1c53 b7   5f       A          STA   CHARCOL  FIRST TWO LINES ARE CHINESE CHARACTERS
00325  A 1c55 a6   02       A          LDA   #2       LAST LINE IS A ENGLISH MESSAGE
00326  A 1c57 b7   5e       A          STA   CHARROW
00327  A 1c59 a6   02       A          LDA   #2
00328  A 1c5b b7   60       A          STA   CHARBYT  2 BYTE SYMBOL CODE
00329
00330  A 1c5d 5f                       CLRX
00331  A 1c5e d6   0e01     A NEXTWEL  LDA   WELCOME+1,X INITIALIZE THE BUFFER WITH THE
00332  A 1c61 26   05     1c68         BNE   FILL     WELCOME MESSAGE UNTIL THE END OF THE
00333  A 1c63 d6   0e00     A          LDA   WELCOME,X RECORD IS DETECTED
00334  A 1c66 27   10     1c78         BEQ   FILLSP
00335
00336  A 1c68 d6   0e00     A FILL     LDA   WELCOME,X
00337  A 1c6b d7   6000     A          STA   BUF,X
00338  A 1c6e d6   0e01     A          LDA   WELCOME+1,X
00339  A 1c71 d7   6001     A          STA   BUF+1,X
00340  A 1c74 5c                       INCX
00341  A 1c75 5c                       INCX
00342  A 1c76 20   e6     1c5e         BRA   NEXTWEL
00343
00344  A 1c78 a3   fa       A FILLSP   CPX   #BUFSIZE FILL THE REST OF THE BUFFER SPACE
00345  A 1c7a 24   0e     1c8a         BHS   FILLDONE WITH SYMBOL BLANK CODE
00346  A 1c7c a6   00       A          LDA   #(BLANK!>8)!.$FF LOAD UPPER BYTE OF THE BLANK CODE
00347  A 1c7e d7   6000     A          STA   BUF,X
00348  A 1c81 a6   20       A          LDA   #BLANK!.$FF LOAD LOWER BYTE OF THE BLANK CODE
00349  A 1c83 d7   6001     A          STA   BUF+1,X
00350  A 1c86 5c                       INCX
00351  A 1c87 5c                       INCX
00352  A 1c88 20   ee     1c78         BRA   FILLSP
00353
00354                       FILLDONE
00355  A 1c8a 01   260d   1c9a         BRCLR MS,CTRL$26,MUX32 CHECK 32 MUX ? OR 41 MUX ?
00356
00357  A 1c8d 4f                MUX41  CLRA
00358  A 1c8e ae   0f       A          LDX   #$F
00359  A 1c90 d7   01c0     A R41INIT  STA   $1C0,X   CLEAR THE DATA IF THE BACKPLANE 40th
00360  A 1c93 5a                       DECX
00361  A 1c94 2a   fa     1c90         BPL   R41INIT
00362
00363  A 1c96 a6   05       A          LDA   #M41LCDR/8 5 BYTE ROW AVAILABLE
00364  A 1c98 20   02     1c9c         BRA   MUXINIT
00365
00366  A 1c9a a6   04       A MUX32    LDA   #M32LCDR/8 4 BYTE ROW AVAILABLE
00367
00368  A 1c9c b7   70       A MUXINIT  STA   LCDROW   INITIALIZE THE # OF ROW AVAILABLE
00369
00370  A 1c9e b7   69       A          STA   TEMP     TEMP = ROW COUNTER
00371
```

```
00372  A 1ca0 a6   02       A CLRLCD   LDA   #2       INITIALIZE THE DISPLAY RAM BASE ADDRESS
00373  A 1ca2 b7   58       A          STA   DESTINE
00374  A 1ca4 3f   59       A          CLR   DESTINE+1
00375
00376  A 1ca6 4f              NXTCOL     CLRA
00377  A 1ca7 5f                         CLRX
00378  A 1ca8 bd   57       A NXTCLR   JSR   STT      CLEAR THE LCD PANEL ROW BY ROW
00379  A 1caa 5c                         INCX
00380  A 1cab a3   80       A          CPX   #LCDCOL
00381  A 1cad 26   f9    1ca8          BNE   NXTCLR
00382  A 1caf b6   59       A          LDA   DESTINE+1
00383  A 1cb1 ab   80       A          ADD   #LCDCOL
00384  A 1cb3 b7   59       A          STA   DESTINE+1
00385  A 1cb5 b6   58       A          LDA   DESTINE
00386  A 1cb7 a9   00       A          ADC   #$0
00387  A 1cb9 b7   58       A          STA   DESTINE
00388
00389  A 1cbb 3a   69       A          DEC   TEMP
00390  A 1cbd 26   e7    1ca6          BNE   NXTCOL   EXIT WHEN IT HITS THE LAST ROW
00391  A 1cbf 81                       RTS
00392
00393                               ********************************************************************************
00395                               ********************************************************************************
00396                               **
00397                               **        Name:   DISPBUF -
00398                               **                        GET EACH SYMBOL FROM THE BUFFER,
00399                               **                        LOCATE THE THE CORRESPONDING SYMBOL PATTERN AND
00400                               **                        COPY TO DISPLAY RAM
00401                               **
00402                               **        Return: A and X will be destroyed
00403                               **
00404                               ********************************************************************************
00405                               DISPBUF
00406  A 1cc0 3f   6f       A          CLR   DISPROW  INITIALIZE ALL THE VARIABLES
00407  A 1cc2 3f   5b       A          CLR   BUFPTR
00408  A 1cc4 3f   5c       A          CLR   LCDPTR
00409  A 1cc6 3f   5d       A          CLR   TABPTR
00410  A 1cc8 3f   59       A          CLR   DESTINE+1 INITIALIZE THE STT RAM SUBROUTINE
00411  A 1cca a6   02       A          LDA   #2       2-BYTE INDEX IS SET TO FIRST BYTE
00412  A 1ccc b7   58       A          STA   DESTINE  OF DISPLAY RAM
00413
00414                               NEXTBUF
00415  A 1cce b6   6f       A          LDA   DISPROW  HIT THE 5th ROW ?
00416  A 1cd0 a1   04       A          CMP   #4
00417  A 1cd2 25   08    1cdc          BLO   C_DISP   DISPLAY A ROW OF 6x8 ENGLISH MESSAGE
00418  A 1cd4 a6   06       A          LDA   #6       AT THE 5th ROW
00419  A 1cd6 b7   5f       A          STA   CHARCOL
00420  A 1cd8 a6   01       A          LDA   #1
00421  A 1cda b7   5e       A          STA   CHARROW
00422
00423  A 1cdc be   5b       A C_DISP   LDX   BUFPTR   LOAD THE SYMBOL CODE TO VARIABLE CHAR
00424  A 1cde d6   6000     A          LDA   BUF,X
00425  A 1ce1 b7   67       A          STA   CHAR
00426  A 1ce3 d6   6001     A          LDA   BUF+1,X
00427  A 1ce6 b7   68       A          STA   CHAR+1
00428
00429  A 1ce8 ad   2a    1d14          BSR   FPTOLCD  FIND PATTERN AND DISPLAY TO LCD
00430
00431  A 1cea b6   5c       A          LDA   LCDPTR   CHECK WHETHER ENOUGH LCD COLUMN SPACE
00432  A 1cec bb   5f       A          ADD   CHARCOL  FOR ANOTHER SYMBOL
00433  A 1cee a1   80       A          CMP   #LCDCOL
00434  A 1cf0 23   15    1d07          BLS   NOADJ
```

```
00435
00436  A 1cf2 3f   5c       A          CLR   LCDPTR   INITIALIZE TO FIRST COLUMN OF NEXT ROW
00437
00438  A 1cf4 be   5e       A          LDX   CHARROW
00439  A 1cf6 3c   6f       A NEXTROW  INC   DISPROW  UPDATE THE STT RAM SUBROUTINE BASE
00440  A 1cf8 b6   59       A          LDA   DESTINE+1 ADDRESS OFFSET AND UPDATE THE ROW
00441  A 1cfa ab   80       A          ADD   #LCDCOL  COUNTER
00442  A 1cfc b7   59       A          STA   DESTINE+1
00443  A 1cfe b6   58       A          LDA   DESTINE
00444  A 1d00 a9   00       A          ADC   #0
00445  A 1d02 b7   58       A          STA   DESTINE
00446
00447  A 1d04 5a                       DECX
00448  A 1d05 26   ef    1cf6         BNE   NEXTROW
00449
00450  A 1d07 b6   5b       A NOADJ    LDA   BUFPTR   UPDATE THE BUFFER POINTER AND POINTING
00451  A 1d09 bb   60       A          ADD   CHARBYT  TO NEXT SYMBOL CODE
00452  A 1d0b b7   5b       A          STA   BUFPTR
00453
00454  A 1d0d b6   6f       A          LDA   DISPROW  HIT THE LAST ROW ?
00455  A 1d0f b1   70       A          CMP   LCDROW
00456  A 1d11 25   bb    1cce         BLO   NEXTBUF
00457  A 1d13 81                       RTS
00458
00459                        ********************************************************************************
00461                        ********************************************************************************
00462                        **
00463                        **        Name:   FPTOLCD -
00464                        **                        FIND SYMBOL LOCATION AND COPY THE PATTERN TO DISPLAY
00465                        **                        RAM
00466                        **                        IF SYMBOL CODE FAILS TO LOCATE, A EQUIVALENT SIZE
00467                        **                        OF SPACE WILL BE COPIED TO DISPLAY RAM
00468                        **
00469                        **        Return: A and X will be destroyed
00470                        **
00471                        ********************************************************************************
00472                        FPTOLCD
00473  A 1d14 b6   22       A          LDA   MMUCB    THE SYMBOL PATTERN MIGHT BE LOCATED
00474                        *                                 IN NON BANK 0
00475  A 1d16 b7   6c       A          STA   MMU      SAVE UP THE MMU REGISTERS BEFORE
00476  A 1d18 b6   20       A          LDA   MMUPA1   SEARCHING START
00477  A 1d1a b7   6d       A          STA   MMU+1
00478  A 1d1c b6   21       A          LDA   MMUPA2
00479  A 1d1e b7   6e       A          STA   MMU+2
00480
00481  A 1d20 cd   1d92     A TRY2     JSR   FINDPAT  SEARCH AND FIND THE LOCATION OF
00482                        *                                 SYMBOL ROM PATTERN
00483  A 1d23 27   18    1d3d         BEQ   DISP     FOUND THE SYMBOL PATTERN ?
00484
00485                        * IF THE PATTERN IS NOT FOUND, SKIP WITH A BLANK SYMBOL
00486
00487  A 1d25 a6   00       A SKIPB    LDA   #(BLANK!>8)!.$FF LOAD THE UPPER BYTE OF THE BLANK CODE
00488  A 1d27 b7   67       A          STA   CHAR
00489  A 1d29 a6   20       A          LDA   #BLANK!.$FF LOAD THE LOWR BYTE OF THE BLANK CODE
00490  A 1d2b b7   68       A          STA   CHAR+1
00491
00492  A 1d2d b6   5f       A          LDA   CHARCOL  BUILD THE BLANK WITH THE EQUIVALENT
00493  A 1d2f ae   00       A          LDX   #XSIZE   SIZE
00494  A 1d31 d7   60fa     A          STA   TABLANK,X
00495  A 1d34 b6   5e       A          LDA   CHARROW
00496  A 1d36 ae   01       A          LDX   #YSIZE
00497  A 1d38 d7   60fa     A          STA   TABLANK,X
```

```
00498  A 1d3b 20   e3    1d20          BRA   TRY2
00499
00500  A 1d3d cd   1d4d     A DISP     JSR   DISPLAY  COPY TO DISPLAY RAM
00501
00502  A 1d40 b6   6c       A          LDA   MMU      RESTORE THE MMU REGISTERS
00503  A 1d42 b7   22       A          STA   MMUCB
00504  A 1d44 b6   6d       A          LDA   MMU+1
00505  A 1d46 b7   20       A          STA   MMUPA1
00506  A 1d48 b6   6e       A          LDA   MMU+2
00507  A 1d4a b7   21       A          STA   MMUPA2
00508  A 1d4c 81                       RTS
00509
00510                       ********************************************************************************
00512                       ********************************************************************************
00513                       **
00514                       **        Name:   DISPLAY -
00515                       **                        COPY THE FOUND SYMBOL PATTERN TO DISPLAY RAM
00516                       **
00517                       **        Return: A and X will be destroyed
00518                       **
00519                       ********************************************************************************
00520                       DISPLAY
00521  A 1d4d b6   5f       A          LDA   CHARCOL
00522  A 1d4f b7   61       A          STA   COUNT    INITIALIZE THE SYMBOL PATTERN SIZE
00523  A 1d51 b6   58       A          LDA   DESTINE  COUNT
00524  A 1d53 b7   62       A          STA   DEST     SAVE UP THE CURRENT DISPLAY RAM ROW
00525  A 1d55 b6   59       A          LDA   DESTINE+1 BASE ADDRESS IN CASE OF PATTERN SIZE
00526  A 1d57 b7   63       A          STA   DEST+1   OTHER THAN 1 BYTE HEIGHT
00527
00528  A 1d59 b6   5e       A N_COL    LDA   CHARROW
00529  A 1d5b b7   69       A          STA   TEMP     INITIALIZE VARIABLE FOR CURRENT ROW
00530  A 1d5d b6   5d       A          LDA   TABPTR
00531  A 1d5f b7   64       A          STA   TPTR     INITIALIZE VARIABLE FOR PATTERN
00532                       *                                 POINTER
00533  A 1d61 be   64       A N_ROW    LDX   TPTR
00534  A 1d63 bd   50       A          JSR   LDD      LOAD THE SYMBOL PATTERN BYTE BY BYTE
00535  A 1d65 be   5c       A          LDX   LCDPTR
00536  A 1d67 bd   57       A          JSR   STT      COPY THE SYMBOL PATTERN BYTE TO DISPLAY
00537  A 1d69 3a   69       A          DEC   TEMP     RAM
00538  A 1d6b 27   14    1d81          BEQ   COL_UP   EXIT IF SIGNLE ROW SYMBOL PATTERN
00539
00540  A 1d6d b6   64       A          LDA   TPTR
00541  A 1d6f bb   5f       A          ADD   CHARCOL
00542  A 1d71 b7   64       A          STA   TPTR
00543
00544  A 1d73 b6   59       A          LDA   DESTINE+1 UPDATE THE DISPLAY RAM ADDRESS TO NEXT
00545  A 1d75 ab   80       A          ADD   #LCDCOL  ROW
00546  A 1d77 b7   59       A          STA   DESTINE+1
00547  A 1d79 b6   58       A          LDA   DESTINE
00548  A 1d7b a9   00       A          ADC   #0
00549  A 1d7d b7   58       A          STA   DESTINE
00550  A 1d7f 20   e0    1d61          BRA   N_ROW    GO TO ANOTHER ROW IF NECESSARY
00551
00552  A 1d81 b6   62       A COL_UP   LDA   DEST     RESTORE THE DISPLAY RAM ADDRESS BACK
00553  A 1d83 b7   58       A          STA   DESTINE  TO THE ORGINAL ADDRESS
00554  A 1d85 b6   63       A          LDA   DEST+1
00555  A 1d87 b7   59       A          STA   DESTINE+1
00556
00557  A 1d89 3c   5d       A          INC   TABPTR   UPDATE THE POINTER TO NEXT SYMBOL
00558                       *                                 PATTERN BYTE
00559  A 1d8b 3c   5c       A          INC   LCDPTR   UPDATE THE DISPLAY RAM COLUMN POINTER
00560
```

```
00561  A 1d8d 3a   61       A          DEC   COUNT    DECREMENT THE COLUMN COUNT BY 1
00562  A 1d8f 26   c8    1d59          BNE   N_COL
00563
00564  A 1d91 81                       RTS
00565
00566                      ********************************************************************************
00568                      ********************************************************************************
00569                      **
00570                      **        Name:   FINDPAT -
00571                      **
00572                      **        SEARCH ALL SYMBOL PATTERN ROM MEMORY UNTIL THE RIGHT PATTERN IS
FOUND
00573                      **        EACH PAGE OF SYMBOL PATTERN ROM MEMORY IS STORE IN THE P_TAB.
00574                      **        EACH RECORD OF THE P_TAB CONTAINS
00575                      **                ADDRESS OF THE SYMBOL PATTERN ROM TABLE
00576                      **                CORRESPONDING MMU COMMON BANK VALUE
00577                      **                CORRESPONDING MMU POSA1 VALUE
00578                      **                CORRESPONDING MMU POSA2 VALUE
00579                      **        THE LAST RECORD HAS ALL ZERO ENTRY
00580                      **
00581                      **        RETURN  : A DESTROY
00582                      **                  X DESTROY
00583                      **                  Z = 1 IF FOUND
00584                      **                  Z = 0 IF NOT FOUND
00585                      **
00586                      ********************************************************************************
00587                      FINDPAT
00588
00589  A 1d92 3f   66       A          CLR   P_TPTR   INITIALIZE THE P_TAB POINTER
00590  A 1d94 10   65       A          BSET  0,FOUND  INITIALIZE THE FLAG
00591
00592                      NEXT_PTAB
00593  A 1d96 be   66       A          LDX   P_TPTR   CHECK FOR NULL RECORD
00594  A 1d98 d6   0de2     A          LDA   P_TAB,X
00595  A 1d9b 26   14    1db1          BNE   LDSET
00596  A 1d9d d6   0de3     A          LDA   P_TAB+1,X
00597  A 1da0 26   0f    1db1          BNE   LDSET
00598
00599  A 1da2 d6   0de4     A          LDA   P_TAB+P_OFFCB,X
00600  A 1da5 26   0a    1db1          BNE   LDSET
00601  A 1da7 d6   0de5     A          LDA   P_TAB+P_OFFPA1,X
00602  A 1daa 26   05    1db1          BNE   LDSET
00603  A 1dac d6   0de6     A          LDA   P_TAB+P_OFFPA2,X
00604  A 1daf 27   26    1dd7          BEQ   P_EXIT   EXIT IF NULL RECORD IS FOUND
00605
00606  A 1db1 d6   0de2     A LDSET    LDA   P_TAB,X  INITIALIZE THE BASE ADDRESS TO THE
00607  A 1db4 b7   51       A          STA   SOURCE   PAGE COINTAINS THE FOUND SYMBOL
00608  A 1db6 d6   0de3     A          LDA   P_TAB+1,X PATTERN
00609  A 1db9 b7   52       A          STA   SOURCE+1
00610  A 1dbb d6   0de4     A          LDA   P_TAB+P_OFFCB,X SET THE MMU REGISTERS
00611  A 1dbe b7   22       A          STA   MMUCB
00612  A 1dc0 d6   0de5     A          LDA   P_TAB+P_OFFPA1,X
00613  A 1dc3 b7   20       A          STA   MMUPA1
00614  A 1dc5 d6   0de6     A          LDA   P_TAB+P_OFFPA2,X
00615  A 1dc8 b7   21       A          STA   MMUPA2
00616
00617  A 1dca cd   1dda     A CHKPTAB  JSR   FOUNDTAB SEARCH THE PAGE OF SYMBOL CODES WHICH
00618                       *                              DEFINES IN P_TAB
00619  A 1dcd 27   08    1dd7          BEQ   P_EXIT   FOUND IF Z = 1; EXIT IF Z = 0
00620  A 1dcf b6   66       A          LDA   P_TPTR
00621  A 1dd1 ab   05       A          ADD   #P_RECSIZE GO TO NEXT P_TAB RECORD
00622  A 1dd3 b7   66       A          STA   P_TPTR
```

```
00623  A 1dd5 20   bf    1d96          BRA   NEXT_PTAB
00624
00625  A 1dd7 3d   65       A P_EXIT   TST   FOUND    RETURN Z=1
00626  A 1dd9 81                       RTS
00627
00628                      ********************************************************************************
00630                      ********************************************************************************
00631                      **
00632                      **        Name:   FOUNDTAB -
00633                      **
00634                      **        LOCATE THE SYMBOL PATTERN ROM WITH THE SYMBOL CODE DEFINED IN
00635                      **        VARIABLE CHAR
00636                      **        ALL THE SYMBOL PATTERN ENTRIES ARE STORE IN A TABLE AND WITH THE
00637                      **        FOLLOWING FORMAT
00638                      **        FIRST TWO BYTE DEFINES THE SYMBOL PATTERN SIZE ( X bits x Y byte)
00639                      **        EACH ENTRY OF THIS TABLE CONTAINS
00640                      **                TWO BYTE CODE REPRESENT THE SYMBOL/CHARACTER CODE
00641                      **                PATTERN OFFSET FROM THE BEGINNING OF THE PAGE
00642                      **        THE LAST RECORD HAS ALL ZERO ENTRY
00643                      **
00644                      **
00645                      **
00646                      **        RETURN  : A DESTROY
00647                      **                  X DESTROY
00648                      **                  Z = 1 FOUND IF THE PATTERN SIZE AND CODE ARE MATCHED
00649                      **                  Z = 0 OTHERWISE
00650                      **
00651                      ********************************************************************************
00652                      FOUNDTAB
00653  A 1dda ae   00       A          LDX   #XSIZE   LOAD THE PATTERN WIDTH SIZE TO CHECK
00654  A 1ddc bd   50       A          JSR   LDD
00655  A 1dde b1   5f       A          CMP   CHARCOL
00656  A 1de0 26   3b    1e1d          BNE   RECEXIT  EXIT IF NOT MATCH
00657  A 1de2 ae   01       A          LDX   #YSIZE   LOAD THE PATTERB HEIGHT SIZE TO CHECK
00658  A 1de4 bd   50       A          JSR   LDD
00659  A 1de6 b1   5e       A          CMP   CHARROW
00660  A 1de8 26   33    1e1d          BNE   RECEXIT  EXIT IF NOT MATCH
00661
00662  A 1dea ae   02       A          LDX   #2       IF MATCH, UPDATE THE SYMBOL PATTERN
00663  A 1dec bf   5d       A          STX   TABPTR   POINTER AND START SEARCHING
00664
00665  A 1dee be   5d       A CHKTAB   LDX   TABPTR   LOAD THE 2-BYTE SYMBOL CODE FROM TABLE
00666  A 1df0 bd   50       A          JSR   LDD
00667  A 1df2 b7   6a       A          STA   TABCODE
00668  A 1df4 5c                       INCX
00669  A 1df5 bd   50       A          JSR   LDD
00670  A 1df7 b7   6b       A          STA   TABCODE+1
00671
00672  A 1df9 3d   6a       A          TST   TABCODE
00673  A 1dfb 26   04    1e01          BNE   C_CHKTAB
00674  A 1dfd 3d   6b       A          TST   TABCODE+1
00675  A 1dff 27   1c    1e1d          BEQ   RECEXIT  EXIT IF HIT THE NULL ENTRY
00676
00677                         C_CHKTAB
00678  A 1e01 b1   68       A          CMP   CHAR+1   SYMBOL CODE MATCH ?
00679  A 1e03 26   10    1e15          BNE   N_REC
00680  A 1e05 b6   6a       A          LDA   TABCODE
00681  A 1e07 b1   67       A          CMP   CHAR
00682  A 1e09 26   0a    1e15          BNE   N_REC    GO TO NEXT SYMBOL CODE IF NOT MATCH
00683
00684  A 1e0b 5c                       INCX           SYMBOL CODE MATCH
00685  A 1e0c bd   50       A          JSR   LDD
```

```
00686  A 1e0e 97                          TAX
00687  A 1e0f b7   5d       A             STA   TABPTR   SET TABPTR TO THE PAGE OFFSET
00688  A 1e11 3f   65       A             CLR   FOUND    CLEAR THE FLAG AND EXIT
00689  A 1e13 20   08    1e1d             BRA   RECEXIT
00690
00691  A 1e15 b6   5d       A N_REC       LDA   TABPTR   TRY NEXT SYMBOL CODE
00692  A 1e17 ab   03       A             ADD   #TABRECSIZE
00693  A 1e19 b7   5d       A             STA   TABPTR
00694  A 1e1b 20   d1    1dee             BRA   CHKTAB
00695
00696  A 1e1d 3d   65       A RECEXIT     TST   FOUND
00697  A 1e1f 81                          RTS
00698
00699
00700                                     ********************************************************************************
00702                                     ********************************************************************************
00703                                     **
00704                                     **        Name:   SCROLL5 -
00705                                     **                        SCROLLING THE 5th ROW OF DISPLAY RAM TO LEFT
00706                                     **                        SIX COLUMN
00707                                     **
00708                                     **        Return: A and X will be destroyed
00709                                     **
00710                                     ********************************************************************************
00711                                     SCROLL5
00712  A 1e20 a6   04       A             LDA   #4       INITIALIZE THE VARIABLES
00713  A 1e22 b7   6f       A             STA   DISPROW
00714  A 1e24 a6   06       A             LDA   #6
00715  A 1e26 b7   5f       A             STA   CHARCOL
00716  A 1e28 a6   01       A             LDA   #1
00717  A 1e2a b7   5e       A             STA   CHARROW
00718  A 1e2c ad   04    1e32             BSR   SCROLLEFT SCROLLING THE ROW OF DISPLAY RAM TO
00719                                     *                                 LEFT BY A COLUMN
00720  A 1e2e cd   1e6c     A             JSR   PAUSE    PAUSE FOR EACH SYMBOL PATTERN MOVE
00721  A 1e31 81                          RTS
00722
00723                                     ********************************************************************************
00725                                     ********************************************************************************
00726                                     **
00727                                     **        Name:   SCROLLEFT -
00728                                     **                        SCROLLING THE SPECIFIED ROW (SINGLE ROW) BY THE
NUMBER
00729                                     **                        OF COLUMN STATED IN VARIABLE CHARCOL
00730                                     **
00731                                     **        Return: A and X will be destroyed
00732                                     **
00733                                     ********************************************************************************
00734                                     SCROLLEFT
00735  A 1e32 a6   02       A             LDA   #2       CALCULATE THE LCD ADDRESS OF THE ROW
00736  A 1e34 b7   51       A             STA   SOURCE
00737  A 1e36 3f   52       A             CLR   SOURCE+1
00738
00739  A 1e38 be   6f       A             LDX   DISPROW
00740  A 1e3a b6   52       A INCR        LDA   SOURCE+1 UPDATE THE BASE DISPLAY RAM ADDRESS
00741  A 1e3c ab   80       A             ADD   #LCDCOL  ACCORDING TO THE VARIABLE DISPROW
00742  A 1e3e b7   52       A             STA   SOURCE+1
00743  A 1e40 b6   51       A             LDA   SOURCE
00744  A 1e42 a9   00       A             ADC   #0
00745  A 1e44 b7   51       A             STA   SOURCE
00746  A 1e46 5a                          DECX
00747  A 1e47 26   f1    1e3a             BNE   INCR
00748
```

```
00749  A 1e49 b6   52       A           LDA   SOURCE+1 DUPLICATE THE DISPLAY RAM ADDRESS
00750  A 1e4b b7   59       A           STA   DESTINE+1 TO DESTINE
00751  A 1e4d ab   01       A           ADD   #1       INCREMENT THE DISPLAY RAM ADDRESS
00752  A 1e4f b7   56       A           STA   MSRC+1   1 AND STORE TO MSRC
00753
00754  A 1e51 b6   51       A           LDA   SOURCE
00755  A 1e53 b7   58       A           STA   DESTINE
00756  A 1e55 a9   00       A           ADC   #0
00757  A 1e57 b7   55       A           STA   MSRC
00758
00759  A 1e59 bd   50       A PICKBYT   JSR   LDD      MOVE THE COLUME 0th DATA TO A BUFFER
00760  A 1e5b c7   0100     A           STA   BLOCK
00761
00762  A 1e5e 5f                        CLRX
00763  A 1e5f bd   54       A MVVLCD    JSR   MVV      SHIFT ALL THE DATA PATTERN BY
00764  A 1e61 5c                        INCX           A COLUMN UNTIL THE LAST COLUMN IS HIT
00765  A 1e62 a3   7f       A           CPX   #$7F
00766  A 1e64 26   f9    1e5f           BNE   MVVLCD
00767
00768  A 1e66 c6   0100     A           LDA   BLOCK    MOVE THE DATA FROM COLUME 0TH BEFORE
00769  A 1e69 bd   57       A           JSR   STT      TO THE LAST COLUMN
00770
00771  A 1e6b 81                        RTS
00772
00773
00774
00775                           ********************************************************************************
00776                           **
00777                           **        Name:   PAUSE() -- PAUSE FOR ONE SECOND
00778                           **
00779                           ********************************************************************************
00780                           PAUSE
00781
00782  A 1e6c 5f                        CLRX
00783                           PAUSE0
00784  A 1e6d a6   d0       A           LDA   #$D0
00785  A 1e6f 5c                PAUSE1  INCX
00786  A 1e70 26   fd    1e6f           BNE   PAUSE1
00787  A 1e72 4a                        DECA
00788  A 1e73 26   fa    1e6f           BNE   PAUSE1
00789  A 1e75 81                        RTS
00790
00791                           ********************************************************************************
00793                           ********************************************************************************
00794                           **
00795                           **        Name:   DUMINT() -- DUMMY INTERRUPT SERVICE ROUTINE
00796                           **
00797                           ********************************************************************************
00798                           DUMINT
00799
00800  A 1e76 80                        RTI
00801
00802
00803
00804
00805                           ********************************************************************************
00807                           ********************************************************************************
00808                           **
00809                           **        SYSTEM RESET VECTOR
00810                           **
00811                           ********************************************************************************
00812  A 0c00                           ORG   $0C00
```

```
00813  A 0c00      1e76     A SPIIRQ   FDB   DUMINT   SPI INTERRUPT VECTOR
00814  A 0c02      1e76     A RTCIRQ   FDB   DUMINT   REAL TIME CLOCK INTERRUPT VECTOR
00815  A 0c04      1e76     A SCIIRQ   FDB   DUMINT   SCI INTERRUPT VECTOR
00816  A 0c06      1e76     A TIMIRQ   FDB   DUMINT   TIMER INTERRUPT VECTOR
00817  A 0c08      1e76     A KEYIRQ   FDB   DUMINT   KEYBOARD INTERRUPT VECTOR
00818  A 0c0a      1e76     A EIRQ     FDB   DUMINT   EXTERNAL INTERRUPT VECTOR
00819  A 0c0c      1e76     A SWIVCT   FDB   DUMINT   SOFTWARE INTERRUPT VECTOR
00820  A 0c0e      1c30     A          FDB   RESET    RESET INTERRUPT VECTOR
00821                                  END


Total number of errors: 0
Total number of warnings: 0
Total number of lines: 821


Number of bytes in section ASCT: 1278

Number of bytes in program: 1278
```

```
M6805 Portable Cross Assembler  0.05   MS-DOS/PC-DOS  Page 1
 Mon Sep 09 10:43:21 1991
Command line:
 C:\PASM\PASM05.EXE -EQSUX -L DROM1.LST -O DROM1.OBJ DROM1.ASM
Options list:
ON  - b - Printing of macro definitions
ON  - c - Printing of macro calls
OFF - d - Placing of symbolic debugging information in COFF
ON  - e - Printing of macro expansions  (changed)
ON  - f - Printing of conditional directives
OFF - g - Printing of generated constants list
ON  - q - Expanding and printing of structured syntax  (changed)
ON  - s - Printing of symbol table  (changed)
ON  - u - Printing of conditional unassembled source  (changed)
ON  - x - Printing of cross reference table  (changed)
OFF - m - Suppress printing of error messages
ON  - w - Printing of warning messages
OFF - v - Suppress printing of updated status
OFF - y - Enabling of sgs extensions
ON  - o - Create object code
ON  -   - Formatting of source line listing
Create listing file - l - DROM1.LST
Change object file name - o - DROM1.OBJ

Xdefs:
  NONE

Xrefs:
  NONE

Input file(s): DROM1.ASM (189 lines)

Output file: DROM1.OBJ
Listing file: DROM1.LST
```

```
M6805 Portable Cross Assembler  0.05  DROM1.ASM  Page 2
 Mon Sep 09 10:43:21 1991
Options - MD,MC,NOG,U,W,MEX,CL,FMT,O

LINE   S PC   OPCO OPERANDS S LABEL    MNEMO OPERANDS COMMENT
00001                       ********************************************************************************
00002                       *
00003                       * FILE NAME:  DROM1.ASM                    DATE: 8th Feb., 1991
00004                       *
00005                       * FILE DESCRIPTION: MC68HC05L10 MCU DEMONSTRATION SYMBOL PATTERN
00006                       *                   DATA FILE 1
00007                       *
00008                       * REVISION: 1.0
00009                       *
00010                       * AUTHOR: JAMSON CHEUNG
00011                       *
00012                       * REMARK: THIS FILE MUST BE IN CONJUCTION WITH THE PROGRAM DPROG.ASM
00013                       *         THIS FILE CONTAINS TWO SYMBOL PATTERN TABLES. BOTH SYMBOL
00014                       *         PATTERN ARE 8 x 6 DOTS SYMBOLS.
00015                       *
00016                       ********************************************************************************
00017                       ********************************************************************************
00018                       *
00019                       *       SYMBOL PATTERN TABLE FORMAT
00020                       *               FIRST TWO BYTE: PATTERN SIZE
00021                       *                               FIRST  BYTE : NUMBER OF DOTS IN X-AXIS
00022                       *                         SECOND BYTE : NUMBER OF BYTES IN Y-AXIS
00023                       *                                       ( 8 DOTS A BYTE )
00024                       *
00025                       *               3 BYTE RECORD FOLLOWING:
00026                       *                               1st TWO BYTE: SYMBOL CODE
00027                       *                         3rd     BYTE: PATTERN OFFSET FROM THE
00028                       *                                       BEGINNING OF THE TABLE
00029                       *
00030                       *               THE RECORD ENDING WITH A NULL RECORD
00031                       *
00032                       *               THE BYTE FOLLOWING THE NULL RECORD ARE THE SYMBOL PATTERN
00033                       *               DATA LOCATION
00034                       *
00035                       ********************************************************************************
00036                       *
00037                       *
00038                                OPT   NOP      COMPILER - LISTING OPTION
00039                       *
00040                       * SYMBOL PATTERN TABLE ADDRESS EQUATE
00041                       *
00042  P 0000      8000     A OFFSET   EQU   $8000
00043
00044  P 0000      8000     A TAB1LOC  EQU   OFFSET+$000
00045  P 0000      8100     A TAB2LOC  EQU   OFFSET+$100
00046
00047                       * NB :    PHYSCIAL ADDRESS = LOGICAL ADDRESS + POSAx * $1000
00048                       *
00049                       *              LOGICAL ADDRESS         PHYSICAL ADDRESS    CB  POSA1  POSA2
00050                       * TAB1LOC       $8000                   $10000              C8   8       0
00051                       * TAB2LOC       $8100                   $11000              C8   8       0
00052                       *
00053
00054
00055                       *
00056                       * SYMBOL PATTERN TABLE 1
```

```
00057                          *
00058  A 8000                            ORG   TAB1LOC
00059  A 8000       06       A TAB1      FCB   SYMEXCL-SYMSP8 NUMBER OF DOTS IN X-AXIS
00060  A 8001       01       A           FCB   1        NUMBER OF BYTES IN Y-AXIS
00061                          *
00062                          * SYMBOL CODE AND SYMBOL PATTERN DATA OFFSET RECORD
00063                          *
00064  A 8002       002053   A           FCB   $00,' ',SYMSP8-TAB1
00065  A 8005       002159   A           FCB   $00,'!,SYMEXCL-TAB1
00066  A 8008       00225f   A           FCB   $00,'",SYMDQUO-TAB1
00067  A 800b       002365   A           FCB   $00,'#,SYMSQ-TAB1
00068  A 800e       00246b   A           FCB   $00,'$,SYMDOL-TAB1
00069  A 8011       002571   A           FCB   $00,'%,SYMPER-TAB1
00070  A 8014       002677   A           FCB   $00,'&,SYMAMP-TAB1
00071  A 8017       00277d   A           FCB   $00,$27,SYMSQUO-TAB1
00072  A 801a       002883   A           FCB   $00,'(,SYMOPBK-TAB1
00073  A 801d       002989   A           FCB   $00,'),SYMCLBK-TAB1
00074  A 8020       002a8f   A           FCB   $00,'*,SYMAST-TAB1
00075  A 8023       002b95   A           FCB   $00,'+,SYMPLUS-TAB1
00076  A 8026       002c9b   A           FCB   $00,$2C,SYMCOMA-TAB1
00077  A 8029       002da1   A           FCB   $00,'-,SYMMINU-TAB1
00078  A 802c       002ea7   A           FCB   $00,'.,SYMDOT-TAB1
00079  A 802f       0030ad   A           FCB   $00,'0,SYM0-TAB1
00080  A 8032       0031b3   A           FCB   $00,'1,SYM1-TAB1
00081  A 8035       0032b9   A           FCB   $00,'2,SYM2-TAB1
00082  A 8038       0033bf   A           FCB   $00,'3,SYM3-TAB1
00083  A 803b       0034c5   A           FCB   $00,'4,SYM4-TAB1
00084  A 803e       0035cb   A           FCB   $00,'5,SYM5-TAB1
00085  A 8041       0036d1   A           FCB   $00,'6,SYM6-TAB1
00086  A 8044       0037d7   A           FCB   $00,'7,SYM7-TAB1
00087  A 8047       0038dd   A           FCB   $00,'8,SYM8-TAB1
00088  A 804a       0039e3   A           FCB   $00,'9,SYM9-TAB1
00089  A 804d       003ae9   A           FCB   $00,':,SYMCOL-TAB1
00090  A 8050       000000   A           FCB   $00,$00,$00 NULL RECORD OF TAB1
00091                          *
00092                          * SYMBOL PATTERN DATA
00093                          *
00094  A 8053       00000000 A SYMSP8    FCB   $00,$00,$00,$00,$00,$00 SPACE
00095  A 8059       0000004f A SYMEXCL   FCB   $00,$00,$00,$4F,$00,$00 !
00096  A 805f       00000700 A SYMDQUO   FCB   $00,$00,$07,$00,$07,$00 "
00097  A 8065       00147f14 A SYMSQ     FCB   $00,$14,$7F,$14,$7F,$14 #
00098  A 806b       00242a7f A SYMDOL    FCB   $00,$24,$2A,$7F,$2A,$12 $
00099  A 8071       00231308 A SYMPER    FCB   $00,$23,$13,$08,$64,$62 %
00100  A 8077       00364956 A SYMAMP    FCB   $00,$36,$49,$56,$20,$58 &
00101  A 807d       00000503 A SYMSQUO   FCB   $00,$00,$05,$03,$00,$00 '
00102  A 8083       00001c22 A SYMOPBK   FCB   $00,$00,$1C,$22,$41,$00 (
00103  A 8089       00004122 A SYMCLBK   FCB   $00,$00,$41,$22,$1C,$00 )
00104  A 808f       0022147f A SYMAST    FCB   $00,$22,$14,$7F,$14,$22 *
00105  A 8095       0008083e A SYMPLUS   FCB   $00,$08,$08,$3E,$08,$08 +
00106  A 809b       00000050 A SYMCOMA   FCB   $00,$00,$00,$50,$30,$00 ,
00107  A 80a1       00080808 A SYMMINU   FCB   $00,$08,$08,$08,$08,$08 -
00108  A 80a7       00000060 A SYMDOT    FCB   $00,$00,$00,$60,$60,$00 .
00109  A 80ad       003e5149 A SYM0      FCB   $00,$3E,$51,$49,$45,$3E 0
00110  A 80b3       0000427f A SYM1      FCB   $00,$00,$42,$7F,$40,$00 1
00111  A 80b9       00426151 A SYM2      FCB   $00,$42,$61,$51,$49,$46 2
00112  A 80bf       00214145 A SYM3      FCB   $00,$21,$41,$45,$4B,$31 3
00113  A 80c5       00181412 A SYM4      FCB   $00,$18,$14,$12,$7F,$10 4
00114  A 80cb       00274545 A SYM5      FCB   $00,$27,$45,$45,$45,$39 5
00115  A 80d1       003c4a49 A SYM6      FCB   $00,$3C,$4A,$49,$49,$30 6
00116  A 80d7       0001f109 A SYM7      FCB   $00,$01,$F1,$09,$05,$03 7
00117  A 80dd       00364949 A SYM8      FCB   $00,$36,$49,$49,$49,$36 8
00118  A 80e3       00064949 A SYM9      FCB   $00,$06,$49,$49,$29,$1E 9
```

```
00119  A 80e9      00000024 A SYMCOL   FCB   $00,$00,$00,$24,$00,$00 :
00120
00121
00122                        *
00123                        * SYMBOL PATTERN 2
00124                        *
00125  A 8100                        ORG   TAB2LOC
00126  A 8100      06       A TAB2     FCB   SYMB-SYMA NUMBER OF DOTS IN X-AXIS
00127  A 8101      01       A          FCB   1        NUMBER OF BYTES IN Y-AXIS
00128                        *
00129                        * SYMBOL CODE AND SYMBOL PATTERN DATA OFFSET RECORD
00130                        *
00131  A 8102      004153   A          FCB   $00,'A,SYMA-TAB2
00132  A 8105      004259   A          FCB   $00,'B,SYMB-TAB2
00133  A 8108      00435f   A          FCB   $00,'C,SYMC-TAB2
00134  A 810b      004465   A          FCB   $00,'D,SYMD-TAB2
00135  A 810e      00456b   A          FCB   $00,'E,SYME-TAB2
00136  A 8111      004671   A          FCB   $00,'F,SYMF-TAB2
00137  A 8114      004777   A          FCB   $00,'G,SYMG-TAB2
00138  A 8117      00487d   A          FCB   $00,'H,SYMH-TAB2
00139  A 811a      004983   A          FCB   $00,'I,SYMI-TAB2
00140  A 811d      004a89   A          FCB   $00,'J,SYMJ-TAB2
00141  A 8120      004b8f   A          FCB   $00,'K,SYMK-TAB2
00142  A 8123      004c95   A          FCB   $00,'L,SYML-TAB2
00143  A 8126      004d9b   A          FCB   $00,'M,SYMM-TAB2
00144  A 8129      004ea1   A          FCB   $00,'N,SYMN-TAB2
00145  A 812c      004fa7   A          FCB   $00,'O,SYMO-TAB2
00146  A 812f      0050ad   A          FCB   $00,'P,SYMP-TAB2
00147  A 8132      0051b3   A          FCB   $00,'Q,SYMQ-TAB2
00148  A 8135      0052b9   A          FCB   $00,'R,SYMR-TAB2
00149  A 8138      0053bf   A          FCB   $00,'S,SYMS-TAB2
00150  A 813b      0054c5   A          FCB   $00,'T,SYMT-TAB2
00151  A 813e      0055cb   A          FCB   $00,'U,SYMU-TAB2
00152  A 8141      0056d1   A          FCB   $00,'V,SYMV-TAB2
00153  A 8144      0057d7   A          FCB   $00,'W,SYMW-TAB2
00154  A 8147      0058dd   A          FCB   $00,'X,SYMX-TAB2
00155  A 814a      0059e3   A          FCB   $00,'Y,SYMY-TAB2
00156  A 814d      005ae9   A          FCB   $00,'Z,SYMZ-TAB2
00157  A 8150      000000   A          FCB   $00,$00,$00 NULL RECORD OF TAB2
00158                        *
00159                        * SYMBOL PATTERN DATA
00160                        *
00161  A 8153      007c1211 A SYMA     FCB   $00,$7C,$12,$11,$12,$7C A
00162  A 8159      00417f49 A SYMB     FCB   $00,$41,$7F,$49,$49,$36 B
00163  A 815f      003e4141 A SYMC     FCB   $00,$3E,$41,$41,$41,$22 C
00164  A 8165      007f4141 A SYMD     FCB   $00,$7F,$41,$41,$41,$3E D
00165  A 816b      007f4949 A SYME     FCB   $00,$7F,$49,$49,$49,$41 E
00166  A 8171      00ff0909 A SYMF     FCB   $00,$FF,$09,$09,$09,$01 F
00167  A 8177      003e4149 A SYMG     FCB   $00,$3E,$41,$49,$49,$3A G
00168  A 817d      007f0808 A SYMH     FCB   $00,$7F,$08,$08,$08,$7F H
00169  A 8183      0000417f A SYMI     FCB   $00,$00,$41,$7F,$41,$00 I
00170  A 8189      00204041 A SYMJ     FCB   $00,$20,$40,$41,$3F,$01 J
00171  A 818f      007f0814 A SYMK     FCB   $00,$7F,$08,$14,$22,$41 K
00172  A 8195      007f4040 A SYML     FCB   $00,$7F,$40,$40,$40,$40 L
00173  A 819b      007f020c A SYMM     FCB   $00,$7F,$02,$0C,$02,$7F M
00174  A 81a1      007f0408 A SYMN     FCB   $00,$7F,$04,$08,$10,$7F N
00175  A 81a7      003e4141 A SYMO     FCB   $00,$3E,$41,$41,$41,$3E O
00176  A 81ad      007f0909 A SYMP     FCB   $00,$7F,$09,$09,$09,$06 P
00177  A 81b3      003e4151 A SYMQ     FCB   $00,$3E,$41,$51,$21,$5E Q
00178  A 81b9      007f0919 A SYMR     FCB   $00,$7F,$09,$19,$29,$46 R
00179  A 81bf      00464949 A SYMS     FCB   $00,$46,$49,$49,$49,$31 S
00180  A 81c5      0001017f A SYMT     FCB   $00,$01,$01,$7F,$01,$01 T
```

```
00181  A 81cb      003f4040 A SYMU     FCB   $00,$3F,$40,$40,$40,$3F U
00182  A 81d1      001f2040 A SYMV     FCB   $00,$1F,$20,$40,$20,$1F V
00183  A 81d7      003f4038 A SYMW     FCB   $00,$3F,$40,$38,$40,$3F W
00184  A 81dd      00631408 A SYMX     FCB   $00,$63,$14,$08,$14,$63 X
00185  A 81e3      00070870 A SYMY     FCB   $00,$07,$08,$70,$08,$07 Y
00186  A 81e9      00615149 A SYMZ     FCB   $00,$61,$51,$49,$45,$43 Z
00187
00188                                   END
00189



Total number of errors: 0
Total number of warnings: 0
Total number of lines: 189


Number of bytes in section ASCT: 478

Number of bytes in program: 478
```

```
M6805 Portable Cross Assembler  0.05   MS-DOS/PC-DOS  Page 1
 Mon Sep 09 10:43:45 1991
Command line:
 C:\PASM\PASM05.EXE -EQSUX -L DROM2.LST -O DROM2.OBJ DROM2.ASM
Options list:
ON  - b - Printing of macro definitions
ON  - c - Printing of macro calls
OFF - d - Placing of symbolic debugging information in COFF
ON  - e - Printing of macro expansions  (changed)
ON  - f - Printing of conditional directives
OFF - g - Printing of generated constants list
ON  - q - Expanding and printing of structured syntax  (changed)
ON  - s - Printing of symbol table  (changed)
ON  - u - Printing of conditional unassembled source  (changed)
ON  - x - Printing of cross reference table  (changed)
OFF - m - Suppress printing of error messages
ON  - w - Printing of warning messages
OFF - v - Suppress printing of updated status
OFF - y - Enabling of sgs extensions
ON  - o - Create object code
ON  -   - Formatting of source line listing
Create listing file - l - DROM2.LST
Change object file name - o - DROM2.OBJ

Xdefs:
  NONE

Xrefs:
  NONE

Input file(s): DROM2.ASM (157 lines)

Output file: DROM2.OBJ
Listing file: DROM2.LST
```

```
M6805 Portable Cross Assembler  0.05  DROM2.ASM  Page 2
 Mon Sep 09 10:43:45 1991
Options - MD,MC,NOG,U,W,MEX,CL,FMT,O

LINE   S PC   OPCO OPERANDS S LABEL    MNEMO OPERANDS COMMENT
00001                       ********************************************************************************
00002                       *
00003                       * FILE NAME:  DROM2.ASM                    DATE: 8th Feb., 1991
00004                       *
00005                       * FILE DESCRIPTION: MC68HC05L10 MCU DEMONSTRATION SYMBOL PATTERN
00006                       *                   DATA FILE 2
00007                       *
00008                       * REVISION: 1.0
00009                       *
00010                       * AUTHOR: JAMSON CHEUNG
00011                       *
00012                       * REMARK: THIS FILE MUST BE IN CONJUCTION WITH THE PROGRAM DPROG.ASM
00013                       *         THIS FILE CONTAINS TWO SYMBOL PATTERN TABLES. BOTH SYMBOL
00014                       *         PATTERN ARE 16x16 DOTS SYMBOLS.
00015                       *
00016                       ********************************************************************************
00017                       ********************************************************************************
00018                       *
00019                       *       SYMBOL PATTERN TABLE FORMAT
00020                       *               FIRST TWO BYTE: PATTERN SIZE
00021                       *                               FIRST  BYTE : NUMBER OF DOTS IN X-AXIS
00022                       *                         SECOND BYTE : NUMBER OF BYTES IN Y-AXIS
00023                       *                                       ( 8 DOTS A BYTE )
00024                       *
00025                       *               3 BYTE RECORD FOLLOWING:
00026                       *                               1st TWO BYTE: SYMBOL CODE
00027                       *                         3rd     BYTE: PATTERN OFFSET FROM THE
00028                       *                                       BEGINNING OF THE TABLE
00029                       *
00030                       *               THE RECORD ENDING WITH A NULL RECORD
00031                       *
00032                       *               THE BYTE FOLLOWING THE NULL RECORD ARE THE SYMBOL PATTERN
00033                       *               DATA LOCATION
00034                       *
00035                       ********************************************************************************
00036                       *
00037                       *
00038                                OPT   NOP      COMPILER - LISTING OPTION
00039                       *
00040                       *
00041                       * SYMBOL PATTERN TABLE ADDRESS EQUATE
00042                       *
00043  P 0000      c000     A OFFSET   EQU   $C000
00044
00045  P 0000      c000     A TAB3LOC  EQU   OFFSET
00046  P 0000      c100     A TAB4LOC  EQU   OFFSET+$100
00047
00048                       *
00049                       * NB :    PHYSCIAL ADDRESS = LOGICAL ADDRESS + POSAx * $1000
00050                       *
00051                       *               LOGICAL ADDRESS         PHYSICAL ADDRESS    CB  POSA1  POSA2
00052                       * TAB3LOC       $C000                   $14000              C8   0      8
00053                       * TAB4LOC       $C100                   $14100              C8   0      8
00054                       *
00055
00056                       *
```

```
00057                          * SYMBOL PATTERN TABLE 3
00058                          *
00059  A c000                                ORG   TAB3LOC
00060  A c000  10          A TAB3    FCB   (YEN-FU)/2 NUMBER OF DOTS IN X-AXIS
00061  A c001  02          A         FCB   2        NUMBER OF BYTE IN Y-AXIS
00062  A c002  c577        A         FDB   $C577
00063  A c004  17          A         FCB   FU-TAB3
00064  A c005  aaef        A         FDB   $AAEF
00065  A c007  37          A         FCB   YEN-TAB3
00066  A c008  bb59        A         FDB   $BB59
00067  A c00a  57          A         FCB   EI-TAB3
00068  A c00b  c17b        A         FDB   $C17B
00069  A c00d  77          A         FCB   LEM-TAB3
00070  A c00e  b855        A         FDB   $B855
00071  A c010  97          A         FCB   MAN-TAB3
00072  A c011  a44f        A         FDB   $A44F
00073  A c013  b7          A         FCB   LEK-TAB3
00074  A c014  0000        A         FDB   $00      NULL RECORD OF TAB3
00075  A c016  00          A         FCB   $00
00076                          *
00077                          * SYMBOL CODE AND SYMBOL PATTERN DATA OFFSET RECORD
00078                          *
00079  A c017  00745e74 A FU        FCB   $00,$74,$5E,$74,$80,$74,$5E,$74
00080  A c01f  0010080c A           FCB   $00,$10,$08,$0C,$F8,$08,$18,$00
00081  A c027  0004027f A           FCB   $00,$04,$02,$7F,$55,$7F,$55,$00
00082  A c02f  40300804 A           FCB   $40,$30,$08,$04,$03,$0E,$78,$00
00083
00084  A c037  00009080 A YEN       FCB   $00,$00,$90,$80,$00,$F8,$04,$02
00085  A c03f  0000fc04 A           FCB   $00,$00,$FC,$04,$04,$FC,$00,$00
00086  A c047  00003e31 A           FCB   $00,$00,$3E,$31,$60,$6F,$68,$64
00087  A c04f  62405f40 A           FCB   $62,$40,$5F,$40,$41,$41,$00,$00
00088
00089  A c057  00102404 A EI        FCB   $00,$10,$24,$04,$86,$4C,$E4,$14
00090  A c05f  0000c40c A           FCB   $00,$00,$C4,$0C,$06,$04,$00,$00
00091  A c067  00412110 A           FCB   $00,$41,$21,$10,$00,$00,$7F,$20
00092  A c06f  25292129 A           FCB   $25,$29,$21,$29,$25,$21,$20,$00
00093
00094  A c077  00fc243c A LEM       FCB   $00,$FC,$24,$3C,$24,$E4,$00,$20
00095  A c07f  100cea28 A           FCB   $10,$0C,$EA,$28,$E8,$08,$00,$00
00096  A c087  007f243c A           FCB   $00,$7F,$24,$3C,$24,$27,$20,$00
00097  A c08f  3c243d01 A           FCB   $3C,$24,$3D,$01,$3D,$24,$3C,$00
00098
00099  A c097  000004e4 A MAN       FCB   $00,$00,$04,$E4,$A6,$AC,$A4,$E0
00100  A c09f  a0a0a4ec A           FCB   $A0,$A0,$A4,$EC,$06,$04,$04,$00
00101  A c0a7  0078080b A           FCB   $00,$78,$08,$0B,$0A,$6A,$5A,$4F
00102  A c0af  4a6a0a0b A           FCB   $4A,$6A,$0A,$0B,$08,$48,$78,$00
00103
00104  A c0b7  00000020 A LEK       FCB   $00,$00,$00,$20,$20,$20,$A0,$70
00105  A c0bf  2c202020 A           FCB   $2C,$20,$20,$20,$E0,$00,$00,$00
00106  A c0c7  00002010 A           FCB   $00,$00,$20,$10,$08,$04,$03,$20
00107  A c0cf  20100c03 A           FCB   $20,$10,$0C,$03,$00,$00,$00,$00
00108
00109
00110                          *
00111                          * SYMBOL PATTERN 4
00112                          *
00113  A c100                                ORG   TAB4LOC
00114  A c100  10          A TAB4    FCB   (KON-GI)/2 NUMBER OF DOTS IN X-AXIS
00115  A c101  02          A         FCB   2        NUMBER OF BYTES IN Y-AXIS
00116  A c102  aabf        A         FDB   $AABF
00117  A c104  14          A         FCB   GI-TAB4
00118  A c105  b4e4        A         FDB   $B4E4
```

```
00119  A c107      34       A          FCB   KON-TAB4
00120  A c108      a4a4     A          FDB   $A4A4
00121  A c10a      54       A          FCB   CHUN-TAB4
00122  A c10b      a4df     A          FDB   $A4DF
00123  A c10d      74       A          FCB   SIN-TAB4
00124  A c10e      a140     A SYMSP16  FDB   $A140
00125  A c110      94       A          FCB   WHITE-TAB4
00126  A c111      0000     A          FDB   $00      NULL RECORD OF TAB2
00127  A c113      00       A          FCB   $00
00128                         *
00129                         * SYMBOL PATTERN DATA
00130                         *
00131  A c114      00008848 A GI       FCB   $00,$00,$88,$48,$28,$18,$08,$08
00132  A c11c      8040a030 A          FCB   $80,$40,$A0,$30,$28,$A0,$60,$00
00133  A c124      0001001f A          FCB   $00,$01,$00,$1F,$11,$11,$1F,$40
00134  A c12c      20100805 A          FCB   $20,$10,$08,$05,$02,$05,$00,$00
00135
00136  A c134      00224488 A KON      FCB   $00,$22,$44,$88,$00,$20,$A0,$64
00137  A c13c      3e247ea4 A          FCB   $3E,$24,$7E,$A4,$24,$20,$20,$00
00138  A c144      00201008 A          FCB   $00,$20,$10,$08,$04,$01,$00,$7F
00139  A c14c      41414547 A          FCB   $41,$41,$45,$47,$71,$02,$00,$00
00140
00141  A c154      0000e010 A CHUN     FCB   $00,$00,$E0,$10,$10,$10,$10,$FE
00142  A c15c      10101010 A          FCB   $10,$10,$10,$10,$E0,$00,$00,$00
00143  A c164      00000302 A          FCB   $00,$00,$03,$02,$02,$02,$02,$7F
00144  A c16c      02020202 A          FCB   $02,$02,$02,$02,$03,$00,$00,$00
00145
00146  A c174      00008040 A SIN      FCB   $00,$00,$80,$40,$A0,$00,$00,$60
00147  A c17c      30000000 A          FCB   $30,$00,$00,$00,$00,$80,$40,$00
00148  A c184      00010000 A          FCB   $00,$01,$00,$00,$01,$02,$06,$0C
00149  A c18c      08080804 A          FCB   $08,$08,$08,$04,$00,$01,$00,$00
00150
00151  A c194      00000000 A WHITE    FCB   $00,$00,$00,$00,$00,$00,$00,$00
00152  A c19c      00000000 A          FCB   $00,$00,$00,$00,$00,$00,$00,$00
00153  A c1a4      00000000 A          FCB   $00,$00,$00,$00,$00,$00,$00,$00
00154  A c1ac      00000000 A          FCB   $00,$00,$00,$00,$00,$00,$00,$00
00155
00156                                  END
00157



Total number of errors: 0
Total number of warnings: 0
Total number of lines: 157


Number of bytes in section ASCT: 395

Number of bytes in program: 395
```

NOTE

NOTE

Literature Distribution Centers:
USA: Motorola Literature Distribution; P.O. Box 20912; Phoenix, Arizona 85036.
EUROPE: Motorola Ltd.; European Literature Centre; 88 Tanners Drive, Blakelands, Milton Keynes, MK14 5BP, England.
JAPAN: Nippon Motorola Ltd.; 4-32-1, Nishi-Gotanda, Shinagawa-ku, Tokyo 141, Japan.
ASIA PACIFIC: Motorola Semiconductors H.K. Ltd.; Silicon Harbour Center, No. 2 Dai King Street, Tai Po Industrial Estate, Tai Po, N.T., Hong Kong.

MOTOROLA
PRINTED IN USA (1993) MPS/POD

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