PNNL-13196

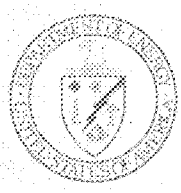
Pacific Northwest
National Laboratory
Operated by Battelle for the
U.S. Department of Energy

Low-Resolution Imaging using Optically Stimulated Luminescence

Project Manager:
SD Miller

Contributors:
Brion Burghard
Jim Skorpik
Rick Traub
Leslie Schwartz

November 22, 1999



Prepared for the U.S. Department of Energy
Under Contract DE-AC06-76RLO 1830

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor Battelle Memorial Institute, nor any of their employees, makes **any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights**. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof, or Battelle Memorial Institute. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

PACIFIC NORTHWEST NATIONAL LABORATORY
operated by
BATTELLE
for the
UNITED STATES DEPARTMENT OF ENERGY
under Contract DE-AC06-76RLO 1830

This document was printed on recycled paper.
(9/97)

PNNL-13196

Low-Resolution Imaging using Optically Stimulated Luminescence

Project Manager:
SD Miller

Contributors:
Brion Burghard
Jim Skorpik
Rick Traub
Leslie Schwartz

November 22, 1999

Prepared for the U.S. Department of Energy
Under Contract DE-AC06-76RLO 1830

EXECUTIVE SUMMARY

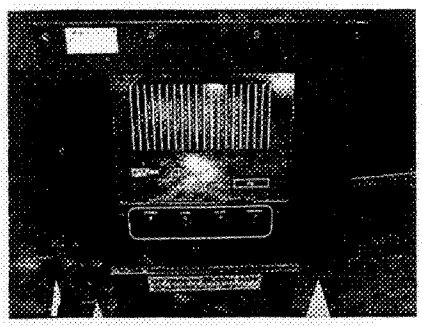
This report describes the development of a laboratory prototype low-resolution imaging system based on the Pacific Northwest National Laboratory (PNNL) developed Optically Stimulated Luminescence (OSL) technology. The PNNL OSL process uses light stimulation to measure the quantity of ionizing radiation exposure. When OSL materials are coated onto a flat two-dimensional format it forms the basis for measuring 2D ionizing radiation patterns analogous to conventional X-ray film. The prototype OSL imaging reader uses 1 square-centimeter light beams to form the image thus rendering the image "low-resolution". This technology has potential uses in nuclear arms control problems and specifically for Warhead and Fissile Material Transparency.

CONTENTS

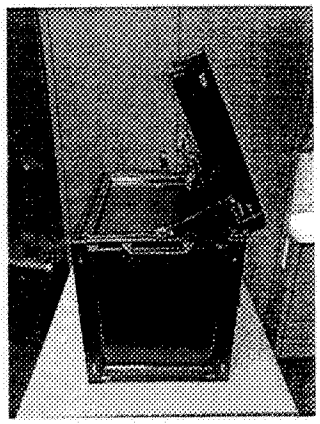
Figures

Appendix

1.0 INTRODUCTION



Reader – front view



Reader – side view

The Pacific Northwest National Laboratory (PNNL) has developed a new radiation detection technique known as Optically Stimulated Luminescence (OSL). The OSL technique is a patented and patent pending technology that relies on specially formulated crystalline phosphors storing the effects of radiation exposure. The OSL phosphors are uniquely able to store the radiation exposure information in the form of an image and then later, when subjected to light exposure, the image is rendered visible through the use of specialized readout equipment. The OSL phosphor image plate most closely resembles the X-ray film used in the medical and dental professions in form and function. Both image media take the form of a flat sheet. The method of storing and developing the radiation images set the two techniques apart. OSL phosphors materials do not require chemical processing as does X-ray film and therefore are more environmentally friendly. In addition to this important difference, the OSL image plates enable a more efficient conversion to digital form (a computer readable form), are reusable, and are much more tolerant of higher radiation exposures compared to X-ray film. OSL technology has a dynamic range of 8 orders-of-magnitude versus only 3 for X-ray film. X-ray film can easily be overexposed with large radiation exposures destroying the image whereas the OSL image plate continues to be useful even at large exposures.

The OSL technology has certain advantages over conventional X-ray film radiography in the Arms Control community because the technology can be made in a low-resolution form and therefore not reveal critical dimensions important to nuclear weapon design. The low-resolution characteristic can be achieved through a combination of image plate fabrication techniques and in the design of the image plate readout system. The fabrication of a low-

resolution image plate begins with creating a thick layer of large OSL crystals. The thickness of the image plate and the use of large phosphor particles decrease the image plates's ability to resolve small features.

The readout instrument that applies light to the image plate for the purposes of developing the stored radiation image into a visible form can also be tailored to producing a low-resolution image. The image plates are readout in discrete panels; light from 10 clusters of light-emitting diodes (LEDs) is applied to 10 small panels of the image plate for a brief moment while the radiation exposure information is collected. This process is repeated until the entire plate is read out. PNNL's engineers have devised a clever configuration that steps through the image plate and completes the readout in the shortest possible time. By controlling the size of light beam the dimensions of the panel become the limiting resolution dimension. A square-centimeter was chosen as the panel size and therefore the OSL image plates will not be able to resolve an image feature smaller than one centimeter.

1.1 Development of OSL Image Plates

An effort to determine the best approach to manufacturing OSL image plate sheets was initiated this fiscal year. Two approaches were considered namely; 1) Coating of a flexible plastic sheet and 2) Hot-pressing of polyethylene carefully mixed with the OSL-grade alumina. Both approaches produced well-dispersed image plates. Small manufacturing runs were procured in advance of the completion of the OSL image plate reader so that they would be available for testing. In late August the mechanical pieces were sufficiently completed so that the sheet feeder could be tested. It was found that the sheet feeder had trouble moving the coated sheets through the reader due to the "spring" action of the sheet. At the coated sheets tried to make the corner in the reader the sheet would push back due to the "spring" action and miss-feed. The hot-pressed sheets made of polyethylene did not have any trouble with the mechanical feed mechanism. It was then determined that a re-engineering of the OSL sheet reader might cost $20-50 K. Since this was beyond our budget we decided to go with the cast polyethylene sheets even though this approach was currently much more expensive to manufacture than the coated sheets. If continued funding exists we will look into alternative manufacturing processes that should make the OSL sheets much cheaper to manufacture.

1.2 Modeling and Development of a Collimator for Imaging Weapons Components

Differentiation of Point and Spherical Radiation Sources

The ability to distinguish between a point and spherical source of radiation that is contained in a metal drum or other container is important to non-intrusive inspection of weapons components. If the drum cannot be opened for inspection, imaging techniques can be used to help distinguish between the two types of source configuration.

Calculations

Calculations were performed using MCNP Version 4B.

The source container was modeled as a steel drum 69 cm high, having a radius of 23 cm. The walls of the drum were 0.13 cm thick. The elemental composition of the drum was iron having a density of 7.86 g/cm^3. The drum was lined with 7.62 cm (3 inch) thick cellotex, which was modeled as carbon having a density of 0.3 g/cm^3.

Two sources were modeled, a point source and a spherical source. The point source was modeled as a massless point in the center of the steel drum. The spherical source was modeled as a spherical shell centered at the center of the drum. The spherical source had an outer radius of 10 cm and was 0.163 cm thick. The spherical shell material was plutonium having a density of 19.86 g/cm^3.

The collimator was modeled as 0.3175 cm thick (1/8 inch) fins that were perpendicular to the axes of the steel drum. The inner radius of the fins was 0.01 cm from the outer radius of the steel drum. The collimator fins were either 2.5 or 4.0 cm long. The collimator fins had a 1 cm, center-to-center, spacing over a 20 cm distance. The collimators extended 10 cm above and below the center of the steel drum. The collimator was modeled as white tin having a density of 7.31 g/cm^3. The detector was modeled as an alumina/polyethylene film, 20 cm wide and 0.0762 cm (30 mil) thick. The density of the detector film was 1.22 g/cm^3.

2.0 RESULTS

The results of the calculations are shown in the figures on the next two pages. The difference between the figures is that for the upper figure on each page, the energy deposition in the film interval has been normalized to the average energy deposition over the whole detector film. For the lower figure of each page, the energy deposition has been normalized to the minimum energy deposition in the collimator intervals.

The figures show that it is possible to differentiate between point and spherical sources using alumina/polyethylene film as a detector when the radiation is collimated with tin sheets. The figures also show that the 4.0 cm long collimator is better than the 2.5 cm long collimator in two respects. First, the peak of the point source is narrower with the longer collimator. Second, the difference between the lowest and highest energy deposition for the 4.0 cm collimator is twice that of the 2.5 cm collimator.

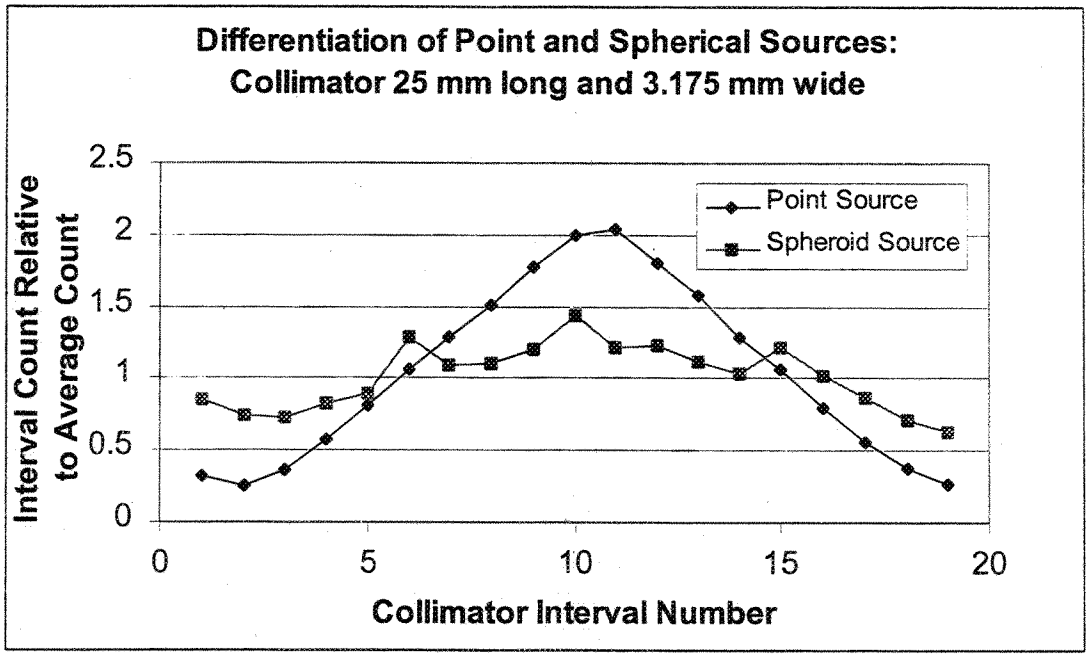


Figure 1.1 Differentiations of Point and Spherical Sources

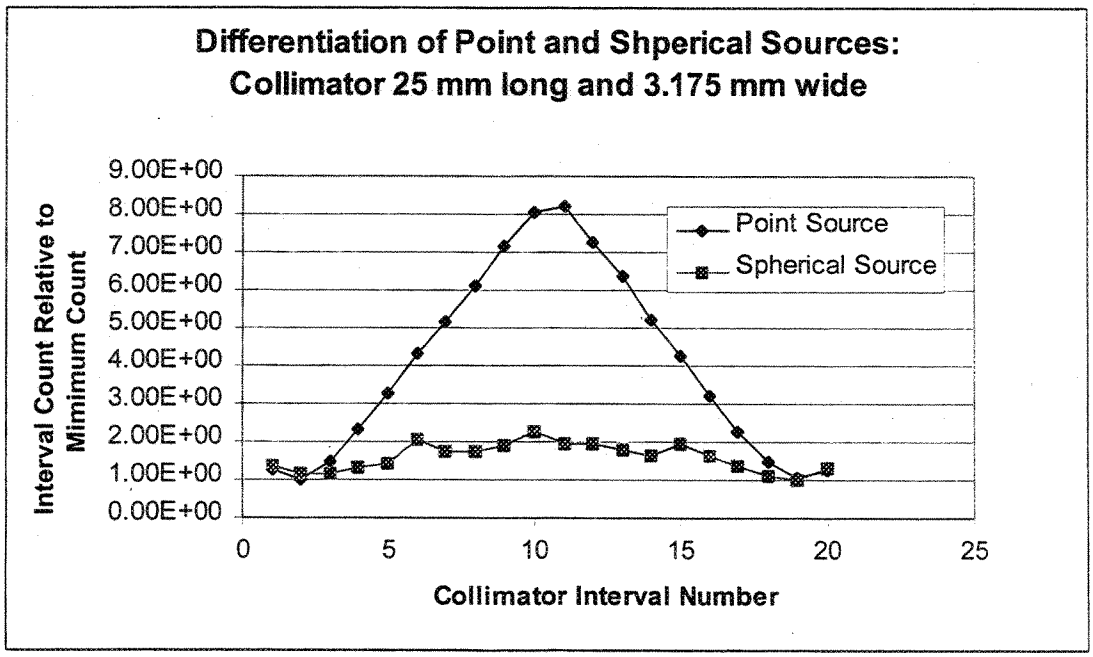


Figure 1.2 Differentiations of Point and Spherical Sources

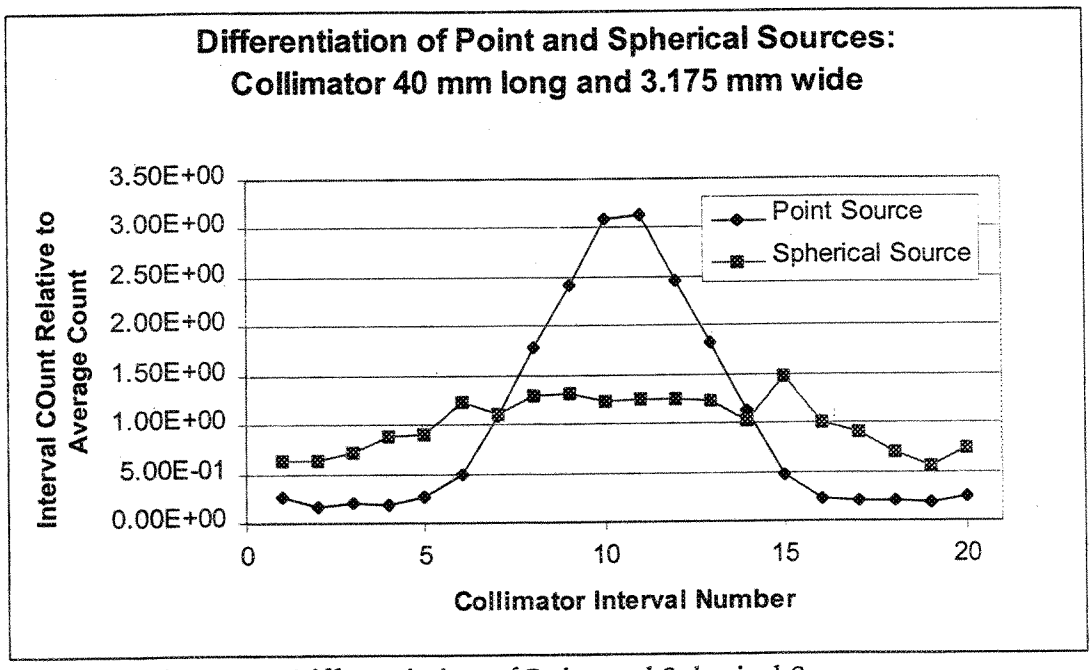


Figure 1.3 Figure 1.2 Differentiation of Point and Spherical Sources

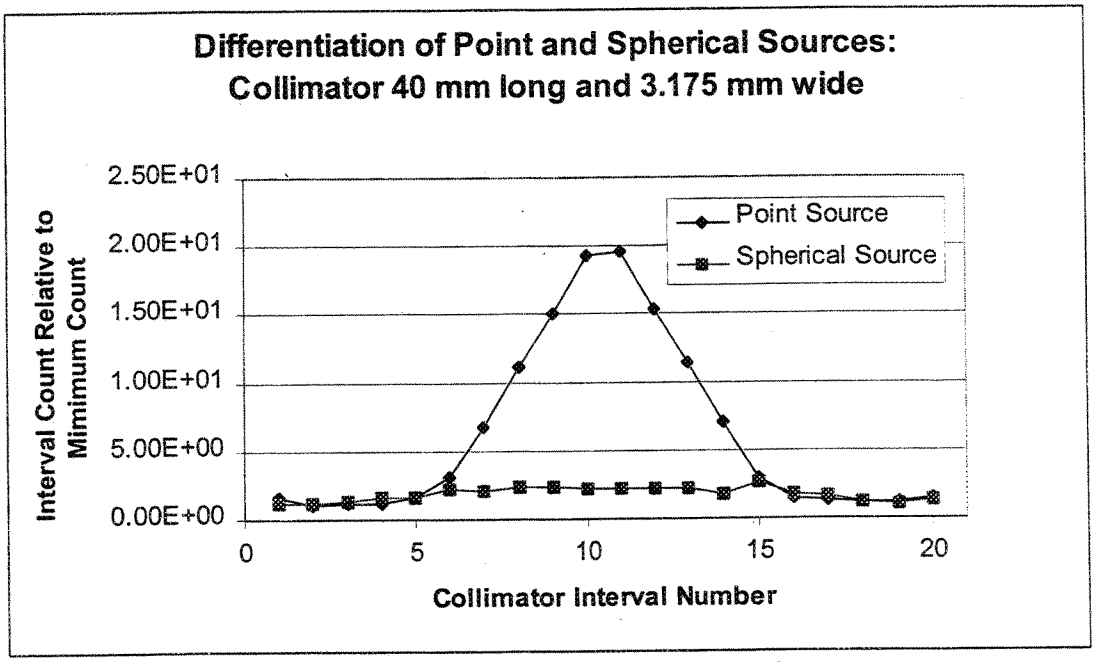


Figure 1.4 Figure 1.2 Differentiation of Point and Spherical Sources

APPENDIX A

OSL FILM SCANNING PROCEDURE

APPENDIX A

OSL FILM SCANNING PROCEDURE

1. When facing the front of the scanner (loading door side), the LED and PMT assembly must be driven to the left side of the scanner before loading the film. The operator lifts the loading door and inserts the short edge of the film (we'll call this edge the front edge of the film and the opposite short edge the rear edge of the film) through the front rollers until this front film edge makes contact with the rear rollers. The long edge of the film should be positioned to the right so that it is up against the guide on the plate between the LED's and the PMT's.

2. With the film in the load position described above, the operator lowers the door slightly and presses the load switch on the computer panel. The LOAD LED is turned on. The (30 cm long by 20 cm wide) film is indexed to the rear of the scanner unit until the rear edge of the film is even with the front of the rear row of masks (there are three masks in this row). The LOAD LED is turned off.

3. The operator closes and latches the door. The operator then presses the SCAN switch. The SCAN LED is turned on and the PMT's are turned on (a mechanical switch on the loading door enables/disables the PMT power).

4. The counters are reset and armed for a fixed period (nominally 6 s). The excitation LED's are immediately turned on for a fixed period (nominally 5 s). The LED period will always be shorter than the counter period. When the counter period times out, the values of the ten counters are transferred to the computer over the serial port and the rear edge of the film is indexed 1 cm toward the FRONT of the scanner.

5. Step 4 is repeated until the front edge of the film just covers the front row of masks (there are 4 masks in the front row). Assuming a 30 cm long film, this requires 38 cycles of step 4. Note that on the 38th cycle the film is not indexed toward the front of the scanner at the end of the counter period.

6. The LED and PMT assembly is then driven to the right side of the scanner.

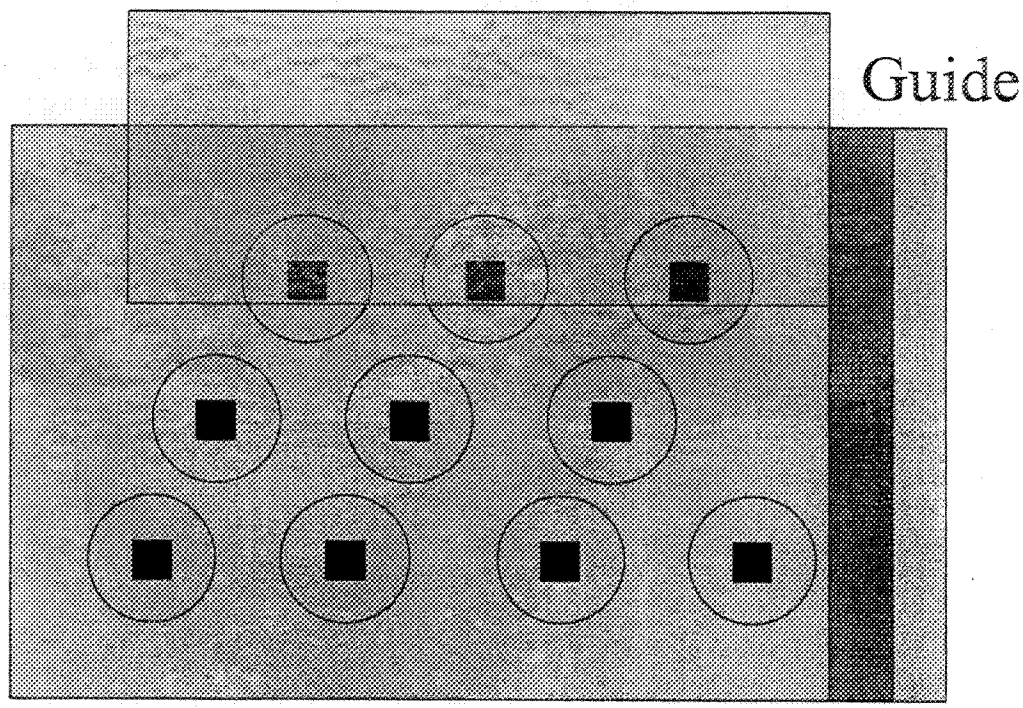
7. The counters are reset and armed for a fixed period (nominally 6 s). The excitation LED's are immediately turned on for a fixed period (nominally 5 s). When the counter period times out, the values of the ten counters are transferred to the computer over the serial port and the front edge of the film is indexed 1 cm toward the REAR of the scanner.

8. Step 7 is repeated until the rear edge of the film just covers the rear row of masks. Again, assuming a 30 cm long film, this requires 38 cycles of step 7. Note that on the 38th cycle the film is not indexed toward the rear of the scanner at the end of the counter period.

9. The PMT's are turned off and the LED and PMT assembly is driven to the left side of the scanner.

10. The SCAN LED is turned off. If desired, the operator may print a copy of the pseudo-colored image on the computer display.
Note: Printing is not currently implemented on the installed software. Instead, the Print function creates a text file of the 30 by 20 array of PMT counts. This is necessary to test and calibrate the unit.

11. When the SCAN LED is off, the operator may unlatch and open the loading door and press the UNLOAD switch. The film is indexed toward the front of the scanner unit. When the motor stops, the operator may pull the film out of the scanner and load a new film as described in step 1.



Front (Loading Side)

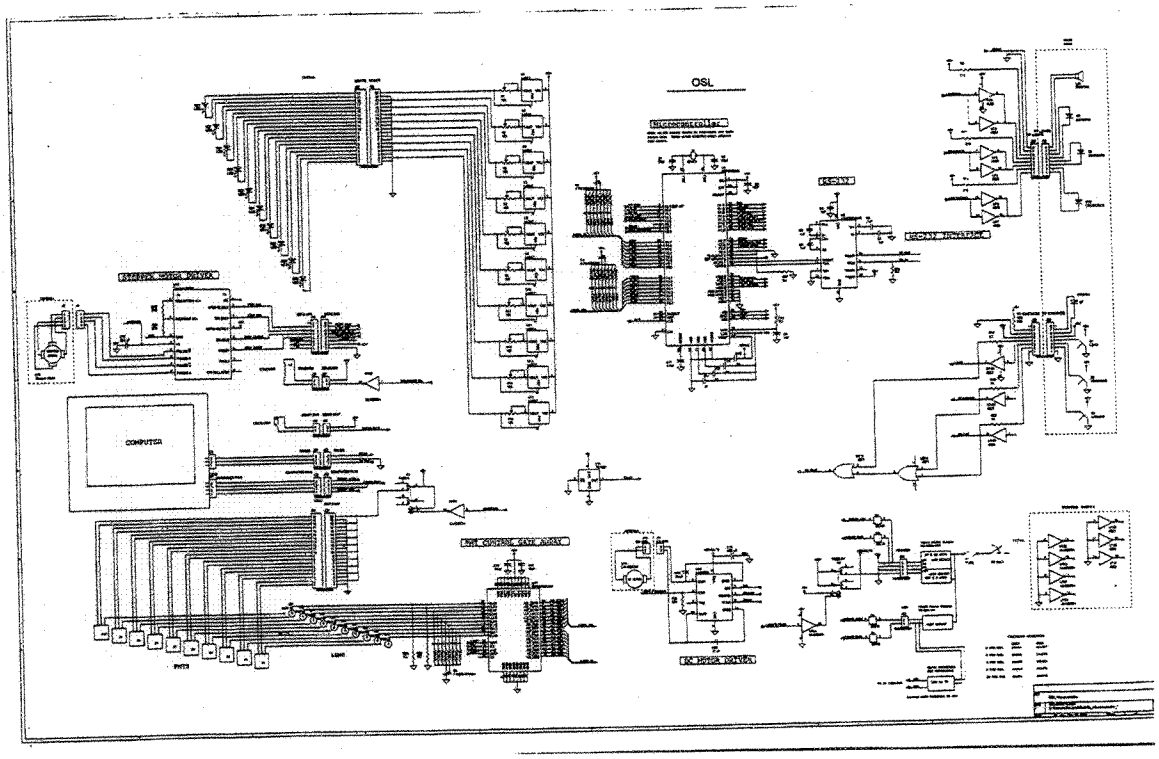
TOP VIEW OF SCANNER MASKS

APPENDIX B

ELECTRICAL 1

APPENDIX B

ELECTRICAL 1

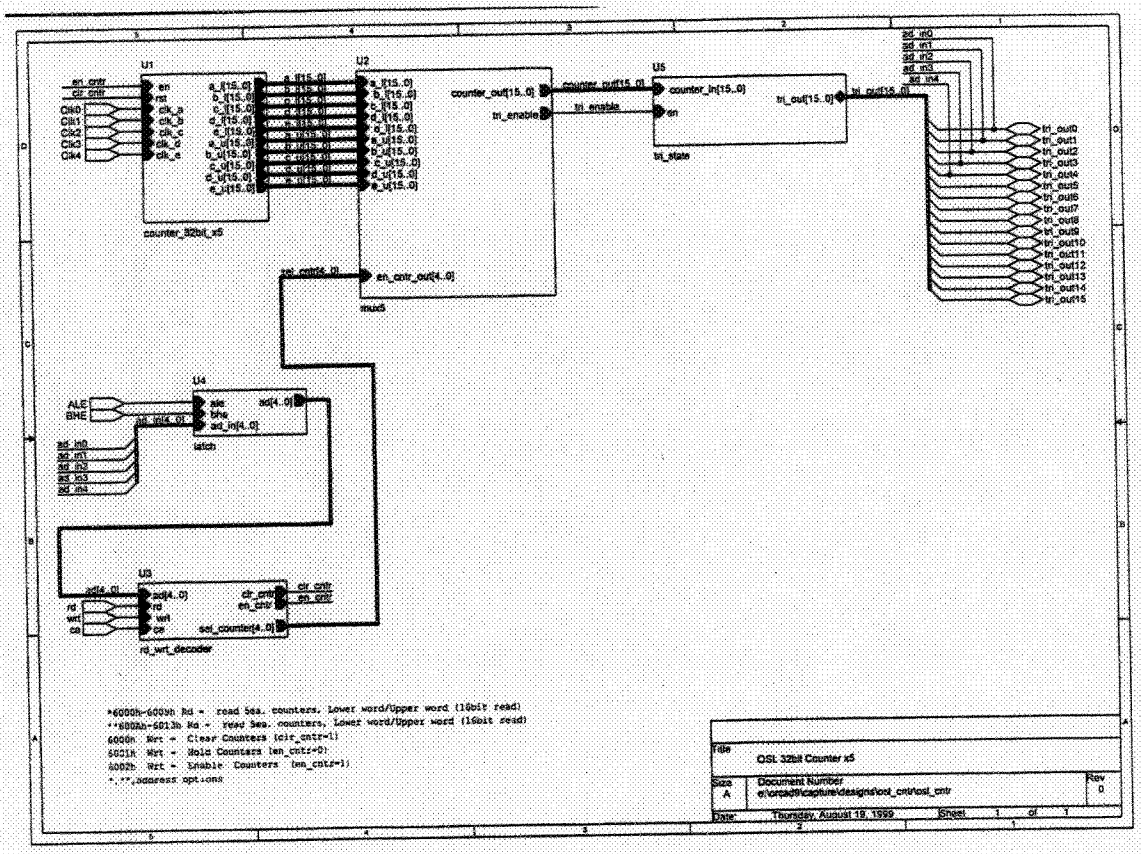


APPENDIX C

ELECTRICAL 2

APPENDIX C

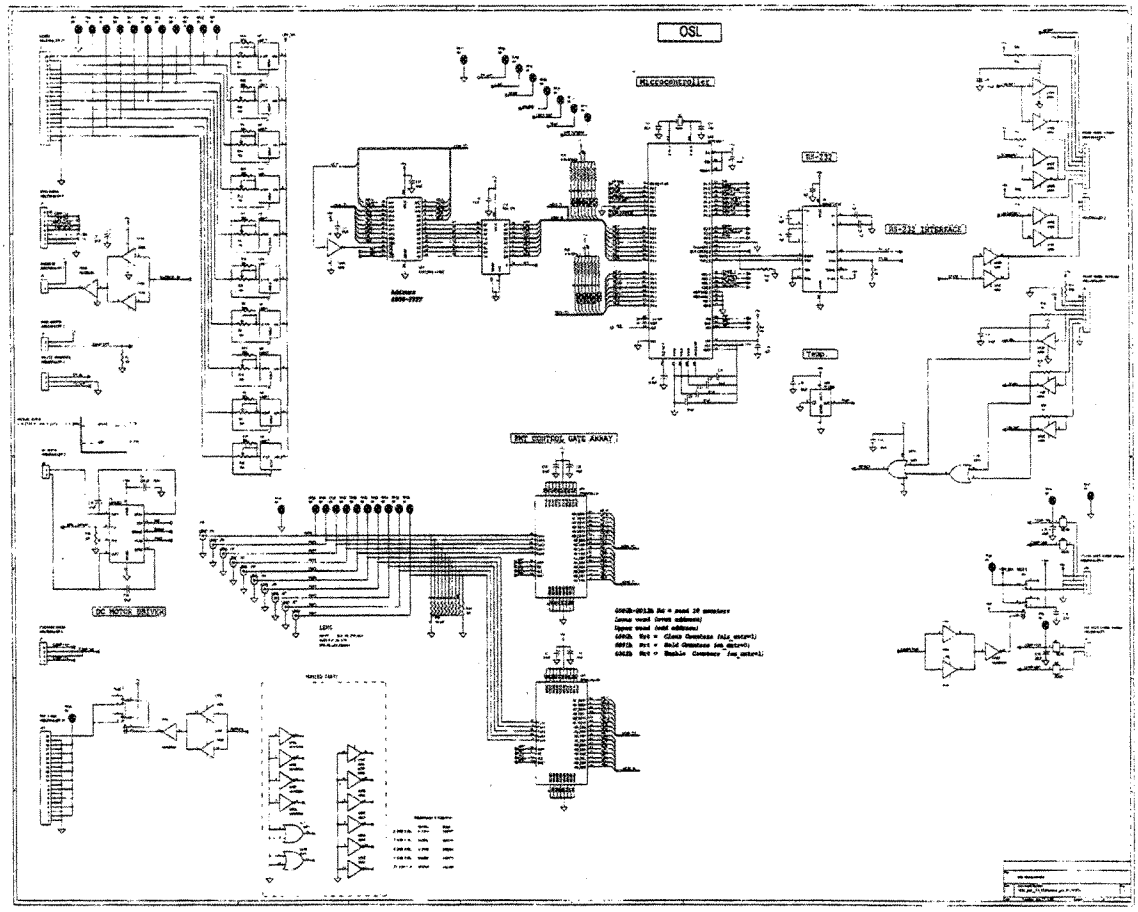
ELECTRICAL 2



APPENDIX D
Electronic 3

APPENDIX D

ELECTRONICS 3



APPENDIX E

COMPLETE SOURCE LISTING FOR THE LOW RESOLUTION OSL IMAGE READER

APPENDIX E

COMPLETE SOURCE LISTING FOR THE LOW RESOLUTION OSL IMAGE READER

```
'************************************************************************
******
'************************************************************************
******
'************************************************************************
******
'************************************************************************
******


************************************************************************
******
--    OSL Image Reader Host PC Source Code
--
************************************************************************
******
************************************************************************
******
-- OSLRead2.mak
--
************************************************************************
******
OSLREAD2.BAS
OSLTOUCH.BAS
GRAPHPAK\OSLSURF.BAS   ' (Source code for Crescent Graphpak library files
is not
GRAPHPAK\AXIS3G.BAS    ' distributable without license)
GRAPHPAK\FILLPOLY.BAS
GRAPHPAK\3DPLOTS.BAS
GRAPHPAK\GRAPH3.BAS
GRAPHPAK\DRAWCHAR.BAS
GRAPHPAK\DRAWTEXT.BAS
GRAPHPAK\SETVIDEO.BAS
GRAPHPAK\DISPLAY.BAS
GRAPHPAK\LOADFONT.BAS
GRAPHPAK\SUPPORT.BAS

--
************************************************************************
******
-- OSL2.doc
--
************************************************************************
******

oslread2+osltouch+3dplots+axis3G+display+drawchar+drawtext+fillpoly+grap
h3+loadfont+setvideo+support
oslread2.EXE /EX /NOD:brun45.lib
```

```
GRAPHPAK.LIB+BCOM45.LIB+QB.LIB


--
*************************************************************************
******
-- OSL2bc.bat
--
*************************************************************************
******
bc e:\qb45\oslread2.bas /e /x /v /w /o /t /ah /c:512
bc e:\qb45\osltouch.bas /e /x /v /w /o /t
bc e:\qb45\3DPlots.bas /e /x /v /w /o /t
bc e:\qb45\axis3g.bas /e /x /v /w /o /t
bc e:\qb45\display.bas /e /x /v /w /o /t
bc e:\qb45\drawchar.bas /e /x /v /w /o /t
bc e:\qb45\drawtext.bas /e /x /v /w /o /t
bc e:\qb45\fillpoly.bas /e /x /v /w /o /t
bc e:\qb45\graph3.bas /e /x /v /w /o /t
bc e:\qb45\loadfont.bas /e /x /v /w /o /t
bc e:\qb45\setvideo.bas /e /x /v /w /o /t
bc e:\qb45\support.bas /e /x /v /w /o /t

--
*************************************************************************
******
-- OSL2link.bat
--
*************************************************************************
******
bc e:\qb45\oslread2.bas /e /x /v /w /o /t /ah /c:512
link @osl2.doc


--
*************************************************************************
******
-- OSLread2.bas
--
*************************************************************************
******
'************************************************************************
*****************
' Portable Low-Resolution OSL Image Reader Application: OSLREAD2.EXE
' File name:      OSLREAD2.BAS
' Date:     September 30, 1999
' Author:   Brion Burghard, PNNL
' Compiler: Microsoft QuickBasic version 4.5
' Compile batch file: Osl2bc.bat
' Link batch file: Osl2link.bat (requires Osl2.doc)
' Third Party Software Libraries used:
'           Crescent GraphPak Professional
'            3DPlots.bas
'            axis3g.bas
'            display.bas
'            drawchar.bas
```

```
'             drawtext.bas
'             fillpoly.bas
'             graph3.bas
'             loadfont.bas
'             setvideo.bas
'             support.bas
'
'           DVRFUNC.BAS (1.5) - Microsoft QuickBASIC 4.0 Interface with
ELODEV 1.5 -
'                     the Elo device driver
'
'                  Copyright (C) 1988, 1990
'                      Elo TouchSystems, Inc.
'                      105 Randolph Road
'                      Oak Ridge, TN  37830
'                      (615) 482-4100
'
'                  Written on 1/08/88 by Michael Sigona
'                  Renamed as OSLtouch.bas by BJB
'*****************************************************************
*****************
DEFINT A-Z

DECLARE SUB DrawButton (Xwide AS INTEGER, Y AS INTEGER, msg AS STRING)

' Touch screen declarations
DECLARE FUNCTION FindDriver% ()
DECLARE SUB DriverInfo ()
DECLARE SUB OpenTouch ()
DECLARE SUB CloseTouch ()
DECLARE SUB EnableTouch ()
DECLARE SUB DisableTouch ()
DECLARE SUB SetMode (Mode%)
DECLARE SUB SetWaitTime (WaitTime%)
DECLARE FUNCTION BufferStatus% ()
DECLARE FUNCTION GetTouch% (X%, Y%, UT%)
DECLARE SUB FlushBuffer ()
DECLARE SUB InitDelay ()
DECLARE SUB ELODelay (Ms%)
DECLARE SUB ELOSound (Freq%)
DECLARE SUB NoSound ()
DECLARE SUB SetRange (Which%, XLow%, XHigh%, YLow%, YHigh%, ZLow%,
ZHigh%)
DECLARE SUB GetZAxis (Valid%, Z%)

' SetRange constants
CONST SETCALIBRANGE = 14, SETXLATERANGE = 15

' Mode bit constants
CONST BUFFERED = &H1, UNTOUCH = &H2, STREAM = &H4, XLATED = &H8
CONST ZONED = &H10


' Graphpak declarations
DECLARE SUB GPPause ()
DECLARE SUB Standby ()
DECLARE SUB Prepare ()
```

```
DECLARE SUB SetSpacing (SpacingH%, SpacingV%)
DECLARE SUB SetFont (FontNumber%)
DECLARE SUB Graph3 (GraphData!(), MainTitle$, Colors%(), Surface%)
DECLARE SUB DrawText (Xx%, Yy%, Text$, Angle%, Colr%, TextSize#)
DECLARE SUB SetVideo ()
DECLARE FUNCTION GetTextWidth% (Text$)
DECLARE SUB LoadFont (FontFile$)
DECLARE FUNCTION HercThere%

' Register declarations for touch screen use
' $INCLUDE: 'QB.BI'
DIM SHARED InRegsX AS RegTypeX, OutRegsX AS RegTypeX


'COMMON SHARED DriverInt%, InfoData AS InfoDataType

''=============================================================================
===
''Setup Screen, Fonts, Tiles, ...  (for Graphpak)
''=============================================================================
===
'$INCLUDE: 'e:\qb45\graphpak\Simple'              'see DEMOGPAK.BAS

' Dimension application variables
    CONST DATABLOCKSIZE = 43
    CONST MaxValue = 1000000
    CONST MinValue = 0
    DIM myMes AS STRING
    DIM myOut AS STRING
    DIM UpdateCurrent AS INTEGER
    DIM ComPort AS INTEGER
    REDIM OSLfilm&(29, 19)
    REDIM OSLfread(29, 19) AS INTEGER
    REDIM OSLdata&(29, 19, 9) ' Holds last 10 film counts
    DIM CurrentIndex% ' The last (current) index to OSLdata&

    DIM myvalue&(10)

    DIM wrow%
    DIM wcol%
    DIM wmyrow%
    DIM wmycol%
    DIM wcount%
    DIM wfnum%
    DIM wfilename$

    DIM colornumber(256) AS LONG
    DIM colorR(182) AS LONG
    DIM colorG(182) AS LONG
    DIM colorB(182) AS LONG
    DIM blue AS LONG
    DIM green AS LONG
    DIM red AS LONG
    DIM gi AS INTEGER
    DIM gj AS INTEGER
    DIM gk AS INTEGER
    DIM gl AS INTEGER
```

```
    DIM D1 AS INTEGER

    CONST OneWideWidth = 10
    CONST OneTallWidth = 10
    DIM Xstart AS INTEGER
    DIM Ystart AS INTEGER
    DIM Xend AS INTEGER
    DIM Yend AS INTEGER
    DIM Xpixels AS INTEGER
    DIM Ypixels AS INTEGER
    DIM myColumn AS SINGLE
    DIM myRow AS SINGLE
    DIM myBoxColor AS SINGLE
    DIM SheetsWide AS INTEGER
    DIM SheetsTall AS INTEGER
    DIM gm AS INTEGER
    DIM gn AS INTEGER
    DIM DivFactor AS SINGLE
    DIM mySheetX AS INTEGER
    DIM mySheetY AS INTEGER
    DIM isDirty AS INTEGER ' Screen needs to be refreshed
    DIM SurfacePlot%

SCREEN 0: WIDTH 80, 50
COLOR 15
CLS
PRINT "Standby... creating color bar"
GOSUB CreateColor

DIM myFnum AS INTEGER

ComPort = 1


MainProgramSection:

ON ERROR GOTO errorhandler
DIM mycomSTR$
mycomSTR$ = "COM" + MID$(STR$(ComPort), 2) + ":9600,N,8,1,DS0,RS,BIN"
OPEN mycomSTR$ FOR RANDOM AS #1
ON COM(ComPort) GOSUB ComCheck
COM(ComPort) ON
inCom% = 0

myFnum = FREEFILE
OPEN "OSLDATA.TXT" FOR APPEND AS #myFnum
CLOSE myFnum
KILL "OSLDATA.TXT"
OPEN "OSLDATA.TXT" FOR APPEND AS #myFnum

myFnum2 = FREEFILE
OPEN "osltouch.txt" FOR OUTPUT AS #myFnum2

IncrCounter& = 0

' set DriverInt% if ELODEV loaded
DriverInt% = FindDriver%
```

```
IF DriverInt% = 0 THEN
 BEEP: PRINT "ELODEV not installed."
 CLOSE
 END
END IF
PRINT "ELODEV installed!"
' get driver info
DriverInfo
PRINT "ELODEV version"; InfoData.DriverVersionMajor +
InfoData.DriverVersionMinor / 10;
PRINT "installed at interrupt "; HEX$(DriverInt%)
PRINT


' initialize ELODEV
SetMode XLATED + STREAM + UNTOUCH
SetRange SETXLATERANGE, 1, 80, 1, 25, 1, 15
OpenTouch
EnableTouch
CurrMenu = 3
isDirty = -1
GOSUB DisplayMainMenu

DO
 GOSUB WaitForData
LOOP UNTIL UCASE$(A$) = "Q"
CloseTouch
CLOSE
END


GetCursorPosition:
    CursorRow% = CSRLIN
    CursorCol% = POS(0)
RETURN


errorhandler:
 errors% = errors% + 1
 ' PRINT "Error "; ERR; " occurred"
 IF errors% > 100 THEN END
 RESUME NEXT
END

WaitForData:
       ' Write status to screen
       IF CurrMenu = 2 THEN
        GOSUB GetCursorPosition
        LOCATE 44, 1: PRINT SPACE$(50);
        LOCATE 44, 1: PRINT "Waiting for new data";
        LOCATE CursorRow%, CursorCol%
       END IF

       WHILE (TimeToUpdate% = 0) AND UCASE$(A$) <> "Q"
        IF GetTouch(Xx%, Yy%, UT%) THEN
         ' Operator pressed the touch screen
         GetZAxis ZAvail%, ZZ%
```

```
        'PRINT #myFnum2, "Menu: "; CurrMenu, "X: "; Xx%, "Y: "; Yy%,
"UT:"; UT%, "ZZ: "; ZZ%
        SELECT CASE CurrMenu

         CASE 1 ' Graphics screen
          SELECT CASE Xx%
           CASE IS >= 75
            SELECT CASE Yy%
           CASE 19 TO 21
            ' Print screen
            IF IncrCounter& = 75 THEN
             BEEP
             GOSUB PrintGraphics
            ELSE
             BEEP: BEEP
            END IF
           CASE IS >= 23
            ' Menu 3
            BEEP
            CurrMenu = 3
            isDirty = -1
           CASE ELSE
            END SELECT
          CASE ELSE
         END SELECT

         CASE 2 ' Diagnostics screen
          SELECT CASE Xx%
           CASE IS > 60
            SELECT CASE Yy%
           CASE 20 TO 22
            ' Print screen
            BEEP
            GOSUB PrintText
           CASE IS >= 23
            ' Menu 3
            CurrMenu = 3
            isDirty = -1
            BEEP
           CASE ELSE
            END SELECT
          CASE ELSE
         END SELECT

         CASE 3 ' Main Menu screen
          SELECT CASE Yy%
           CASE 11 TO 13
           ' Menu 1, Graphics
            CurrMenu = 1
            isDirty = -1
            BEEP
           CASE 15 TO 17
            ' Surface Plot
            ' Menu 4
            CurrMenu = 4
            isDirty = -1
            SurfacePlot% = 0
```

```
            BEEP
           CASE 19 TO 21
            ' Diagnostics
            BEEP
            CurrMenu = 2
            isDirty = -1
           CASE ELSE
          END SELECT

         CASE 4
          ' Surface Plot
          ' Go back to main menu
          CurrMenu = 3
          isDirty = -1

         CASE ELSE
        END SELECT
        ' Clear the port
        WHILE GetTouch(Xx%, Yy%, UT%)
        WEND
       END IF
       IF isDirty = -1 THEN
        GOSUB UpdateDisplay
       END IF
       A$ = INKEY$
      WEND
      TimeToUpdate% = 0
      GOSUB UpdateDisplay
RETURN


UpdateNumericTable:
 IF (IncrCounter& = 0) OR (isDirty = -1) THEN
  SCREEN 0: WIDTH 80, 50
  CLS
  COLOR 15
  ' Write the menu buttons to the screen

  LOCATE 46, 60
  PRINT CHR$(201);
  FOR Col% = 61 TO 79
   LOCATE 46, Col%
   PRINT CHR$(205);
  NEXT
  LOCATE 46, 80
  PRINT CHR$(187);
  FOR Row% = 47 TO 49
   LOCATE Row%, 60
   PRINT CHR$(186);
   PRINT SPACE$(7);
   IF Row% = 48 THEN
    PRINT "Menu "; : PRINT SPACE$(7); : PRINT CHR$(186);
   ELSE
    PRINT SPACE$(12); :  PRINT CHR$(186);
   END IF
  NEXT
  LOCATE 50, 60
```

```
 PRINT CHR$(200);
 FOR Col% = 61 TO 79
  PRINT CHR$(205);
 NEXT
 PRINT CHR$(188);
END IF
IF IncrCounter& = 75 THEN
 ' Draw print button
 COLOR 15
 LOCATE 40, 60
 PRINT CHR$(201);
 FOR Col% = 61 TO 79
  LOCATE 40, Col%
  PRINT CHR$(205);
 NEXT
 LOCATE 40, 80
 PRINT CHR$(187);
 FOR Row% = 41 TO 43
  LOCATE Row%, 60
  PRINT CHR$(186);
  PRINT SPACE$(7);
  IF Row% = 42 THEN
   PRINT "Print"; : PRINT SPACE$(7); : PRINT CHR$(186);
  ELSE
   PRINT SPACE$(12); :  PRINT CHR$(186);
  END IF
 NEXT
 LOCATE 44, 60
 PRINT CHR$(200);
 FOR Col% = 61 TO 79
  PRINT CHR$(205);
 NEXT
 PRINT CHR$(188);
END IF
 FOR Col% = 1 TO 30
  FOR Row% = 1 TO 20
   IF OSLfilm&(Col% - 1, Row% - 1) > 0 THEN
    myval! = (OSLfilm&(Col% - 1, Row% - 1) - MinValue) / (MaxValue -
MinValue)
    myval& = myval! * 100
   ELSE
    myval& = 0
   END IF
   LOCATE (31 - Col%), (1 + ((Row% - 1) * 4))
   PRINT USING "###"; myval&;
  NEXT
 NEXT
 LOCATE 37, 1
 PRINT USING "########   ########   ########   ########   ########";
myvalue&(1); myvalue&(2); myvalue&(3); myvalue&(4); myvalue&(5)
 PRINT USING "########   ########   ########   ########   ########";
myvalue&(6); myvalue&(7); myvalue&(8); myvalue&(9); myvalue&(10)
 PRINT
 PRINT USING "Temperature:           ###.#"; Temperature!
 PRINT USING "Records received: ##########"; DataCounter&
 PRINT USING "Increment count:         ###"; IncrCounter&
 PRINT USING "Checksum:                ###"; (chksum& AND &HFF)
```

```
RETURN


UpdateGraphics:

' Set screen mode to VGA, 320 by 200 pixels, 256 color palette
IF (IncrCounter& = 0) OR (isDirty = -1) THEN
 SCREEN 13: COLOR (0)
 PALETTE USING colornumber(0)
 CLS
'' Test palette by drawing vertical bars of each color on the screen
'gj = 0
'FOR gi = 1 TO 300
' PSET (gi, 0)
' mycolor$ = MID$(STR$(gj), 2)
' mydraw$ = "C" + mycolor$ + "D200"
' DRAW mydraw$
' gj = gj + 1
' IF gj > 180 THEN gj = 0
'NEXT
'
'' Wait for input
'DO
' A$ = INKEY$
'LOOP WHILE A$ = ""
'CLS
 ' Put a vertical color index bar on the right side of the screen
 FOR j = 0 TO 150
  FOR i = 300 TO 319
   PSET (i, j), INT((150 - j) / 150 * 180)
  NEXT
 NEXT

 ' Draw buttons
 FOR j = 176 TO 198
   PSET (301, j), 181
   PSET (302, j), 181
   IF j < 178 OR j > 195 THEN
    LINE -(318, j), 181
   END IF
   PSET (318, j), 181
   PSET (319, j), 181
 NEXT
 COLOR 181
 LOCATE 24, 39: PRINT "M";

END IF

'*************************************************************************
******

'
IF IncrCounter& = 75 THEN
 ' Draw print button
 FOR j = 152 TO 175
   PSET (301, j), 181
```

```
   PSET (302, j), 181
   IF j < 155 OR j > 174 THEN
    LINE -(318, j), 181
   END IF
   PSET (318, j), 181
   PSET (319, j), 181
 NEXT
 COLOR 181
 LOCATE 21, 39: PRINT "P";
END IF

SheetsWide = 1: SheetsTall = 1

' Update colored squares on the screen's upper left 300 by 200 pixels
'DO
SheetsWide = 1
SheetsTall = 1

 FOR X = 1 TO SheetsWide
  FOR Y = 1 TO SheetsTall
   mySheetX = X: mySheetY = Y
   ' Draw squares, 300 by 200

   FOR gj = 0 TO 199 STEP 10
    FOR gi = 0 TO 299 STEP 10
     Row% = gj / 10: Col% = gi / 10
     IF OSLfilm&(Col%, Row%) > 0 THEN
      ' Scale output to fit range 1 to 180 in 6 step log scale
      SELECT CASE OSLfilm&(Col%, Row%)
       CASE IS <= 10
      myval! = (CSNG(OSLfilm&(Col%, Row%)) / 10!) * 30!
       CASE IS <= 100
      myval! = 30! + ((CSNG(OSLfilm&(Col%, Row%)) - 10!) / 90!) * 30!
       CASE IS <= 1000
      myval! = 60 + ((CSNG(OSLfilm&(Col%, Row%)) - 100!) / 900!) * 30!
       CASE IS <= 10000
      myval! = 90 + ((CSNG(OSLfilm&(Col%, Row%)) - 1000!) / 9000!) * 30!
       CASE IS <= 100000
      myval! = 120! + ((CSNG(OSLfilm&(Col%, Row%)) - 10000!) / 90000!) *
30!
       CASE IS <= 1000000
      myval! = 150! + ((CSNG(OSLfilm&(Col%, Row%)) - 100000!) / 900000!)
* 30!
       CASE ELSE
      myval! = 180!
      END SELECT
     ELSEIF OSLfread(Col%, Row%) = 0 THEN
      myval! = -1
     ELSE
      myval! = 0
     END IF
     IF myval! > 180 THEN myval! = 180
     myBoxColor = INT(myval!)
     GOSUB PaintBox
    NEXT
   NEXT
  NEXT
```

```
 NEXT
 ' Frame the image
 PSET (0, 0), 181
 LINE -(300, 0)
 LINE -(300, 199)
 LINE -(0, 199)
 LINE -(0, 0)
RETURN

PaintBox:
 myRow = Row%
 myColumn = Col%

 ' Paint the square the requested color

 ' Determine which pixels need to be painted
 IF SheetsWide > SheetsTall THEN
  DivFactor = SheetsWide
 ELSE
  DivFactor = SheetsTall
 END IF
 Xstart = ((myColumn * OneWideWidth) / DivFactor) + ((mySheetX - 1) *
(300 / DivFactor))
 Ystart = (myRow * OneTallWidth) / DivFactor + ((mySheetY - 1) * (200 /
DivFactor))
 Xpixels = OneWideWidth / DivFactor
 Ypixels = OneTallWidth / DivFactor
 Xend = (Xstart + Xpixels - 1)
 IF Xstart < 1 THEN Xstart = 1
 IF Xend < 299 THEN Xend = Xend + 1
 IF Xend > 299 THEN Xend = 299
 IF Xend < Xstart THEN Xend = Xstart
 Yend = (Ystart + Ypixels - 1)
 IF Ystart < 1 THEN Ystart = 1
 IF Yend < Ystart THEN Yend = Ystart
 IF Yend < 199 THEN Yend = Yend + 1
 IF Yend > 199 THEN Yend = 199

 IF (myBoxColor < 0) THEN
  ' Square hasn't been read yet
  FOR gm = Xstart TO Xend
   FOR gn = Ystart TO Yend
    IF (gm = Xstart) OR (gm = Xend) OR (gn = Ystart) OR (gn = Yend) THEN
     PSET (gm, gn), 181
    ELSE
     PSET (gm, gn), 182
    END IF
   NEXT
  NEXT
 ELSE
  FOR gm = Xstart TO Xend
   FOR gn = Ystart TO Yend
    IF (gm = Xstart) OR (gm = Xend) OR (gn = Ystart) OR (gn = Yend) THEN
     IF IncrCounter& <> 75 THEN
      PSET (gm, gn), 181
     ELSE
      PSET (gm, gn), myBoxColor
```

```
     END IF
    ELSE
     PSET (gm, gn), myBoxColor
    END IF
   NEXT
  NEXT
 END IF
RETURN

CreateColor:
'*************************************************************************
******
' Create the color palette
' 256 levels available in palette, we'll just use 180 + 1 for white
'Make level 0 black, 181 white
colornumber(0) = 0

' Start with Violet
red = (144 / 255) * 63
green = 64
blue = 63
FOR i = 1 TO 30
 red = INT(((128! + (112! * (CSNG(i - 1) / 30!))) / 255!) * 63!)
 green = INT(((64! - (64! * (CSNG(i - 1) / 30!))) / 255!) * 63!)
 colornumber(i) = (65536 * blue) + (256& * green) + red
NEXT

blue = 63 / 2
green = 0
red = 63 / 2
' Transition from dark violet to light violet, 1 to 30
FOR i = 1 TO 30
 red = INT(((192! * (CSNG(i) / 30!)) / 255!) * 63!)
 colorR(i) = red
 colorG(i) = green
 colorB(i) = blue
 colornumber(i) = (65536 * blue) + (256& * green) + red
NEXT

' Transition from dark blue to light blue, 31 to 60
blue = 63
green = 0
red = 0
FOR i = 31 TO 60
 green = INT(((250! * (CSNG(i - 30!) / 30!)) / 255!) * 63!)
 red = INT(((166! * (CSNG(i - 30!) / 30!)) / 255!) * 63!)
 colorR(i) = red
 colorG(i) = green
 colorB(i) = blue
 colornumber(i) = (65536 * blue) + (256& * green) + red
NEXT

' Transition from dark green to light green, 61 to 90
blue = (60 / 255) * 63
green = (132 / 255) * 63
red = 0
FOR i = 61 TO 90
```

```
 blue = INT(((60! - (60! * (CSNG(i - 61) / 30!))) / 255!) * 63!)
 green = INT(((132! + (123! * (CSNG(i - 61) / 30!))) / 255!) * 63!)
 colorR(i) = red
 colorG(i) = green
 colorB(i) = blue
 colornumber(i) = (65536 * blue) + (256& * green) + red
NEXT

' Transition from dark yellow to light yellow, 91 to 120
blue = 0
green = (128 / 255) * 63
red = (128 / 255) * 63
FOR i = 91 TO 120
 red = INT(((128! + (127! * (CSNG(i - 91) / 30!))) / 255!) * 63!)
 green = INT(((128! + (127! * (CSNG(i - 91) / 30!))) / 255!) * 63!)
 colorR(i) = red
 colorG(i) = green
 colorB(i) = blue
 colornumber(i) = (65536 * blue) + (256& * green) + red
NEXT

' Transition from dark orange to light orange, 121 to 150
blue = 0
green = (96 / 255) * 63
red = (192 / 255) * 63
FOR i = 121 TO 150
 red = INT(((192! + (63! * (CSNG(i - 121) / 30!))) / 255!) * 63!)
 green = INT(((96! + (80! * (CSNG(i - 121) / 30!))) / 255!) * 63!)
 blue = INT(((48! * (CSNG(i - 121) / 30!)) / 255!) * 63!)
 colorR(i) = red
 colorG(i) = green
 colorB(i) = blue
 colornumber(i) = (65536 * blue) + (256& * green) + red
NEXT

' Transition from dark red to light red, 151 to 180
blue = 0
green = 0
red = 63
FOR i = 151 TO 180
 green = INT((((176! * (CSNG(i - 151) / 30!))) / 255!) * 63!)
 blue = INT((((176! * (CSNG(i - 151) / 30!))) / 255!) * 63!)
 colorR(i) = red
 colorG(i) = green
 colorB(i) = blue
 colornumber(i) = (65536 * blue) + (256& * green) + red
NEXT

' Make 181 = white
colornumber(181) = (65536 * 63&) + (256& * 63&) + 63&
 colorR(181) = 63
 colorG(181) = 63
 colorB(181) = 63

' Make 182 = less intense white
colornumber(182) = (65536 * 40&) + (256& * 40&) + 40&
 colorR(182) = 40
```

```
 colorG(182) = 40
 colorB(182) = 40

OPEN "oslcolor.txt" FOR OUTPUT AS #5
PRINT #5, "Red", "Green", "Blue"
FOR i = 0 TO 182
 PRINT #5, colorR(i), colorG(i), colorB(i)
NEXT
CLOSE 5
RETURN

PrintGraphics:
' Print the pseudo color image
' ***** Temporary *****
' Save the counts for debug, replace with print function later
   wfnum% = FREEFILE
   wcount% = wcount% + 1
   wfilename$ = "cnt" + MID$(STR$(wcount%), 2) + ".txt"
   OPEN wfilename$ FOR OUTPUT AS #wfnum%
   FOR wrow% = 0 TO 199 STEP 10
    FOR wcol% = 0 TO 299 STEP 10
     mywrow% = wrow% / 10: mywcol% = wcol% / 10
     PRINT #wfnum%, OSLfilm&(mywcol%, mywrow%)
    NEXT
   NEXT
   CLOSE wfnum%
RETURN

PrintText:
' Print the columns and rows
RETURN

DisplayMainMenu:
  'CurrMenu = 3
  'SCREEN 13: COLOR (0)
  'PALETTE USING colornumber(0)
  'CLS
  IF isDirty = -1 THEN

   SCREEN 13: COLOR (0)
   PALETTE USING colornumber(0)
   CLS

   ' Draw a frame
   PSET (0, 0), 40
   LINE -(319, 0), 40
   LINE -(319, 199), 40
   LINE -(0, 199), 40
   LINE -(0, 0), 40

   PSET (5, 5), 40
   LINE -(314, 5), 40
   LINE -(314, 194), 40
   LINE -(5, 194), 40
   LINE -(5, 5), 40

   COLOR 181
```

```
    msg$ = "Low-Resolution OSL Image Reader"
    LOCATE 5, (21 - (LEN(msg$) / 2))
    PRINT msg$;

    msg$ = "Main Menu"
    LOCATE 8, (20 - (LEN(msg$) / 2))
    PRINT msg$;
    GOSUB DrawMainMenuButtons
    isDirty = 0

  END IF
RETURN


DrawMainMenuButtons:
  ' Write the menu buttons to the screen
  ' Write the menu buttons to the screen
  COLOR 181
  CALL DrawButton(80, 50, "View Image")
  CALL DrawButton(80, 65, "Surface Plot")
  CALL DrawButton(80, 80, "Diagnostics")
RETURN


UpdateDisplay:

SELECT CASE CurrMenu
 CASE 1 ' Graphics
  GOSUB UpdateGraphics
 CASE 2 ' Debug
  GOSUB UpdateNumericTable
 CASE 3 ' Main Menu
  GOSUB DisplayMainMenu
 CASE 4 ' Surface Plot
  GOSUB DisplaySurfacePlot
 CASE ELSE
END SELECT

 isDirty = 0
RETURN

DisplaySurfacePlot:
IF SurfacePlot% = 0 THEN
 CLS
 CALL SetVideo                    'initiate graphics using the correct
Screen


'=====================================================================
==
 'Setup GPData% Array for Plaster Casting Surface Plots (GraphPak)

'=====================================================================
==
 GPDat%(4) = 0        'Turn off extra boundary
 GPDat%(10) = 80      'X-Axis angle
 GPDat%(11) = 60      'Y-Axis angle
```

```
 CALL SetSpacing(2, 2)

'=============================================================================
==
 'Setup for surface plot (GraphPak)

'=============================================================================
==
 'CALL Standby
 REDIM GraphData!(30, 20), Colors(600), Colr(6)
 Colr(1) = 5        'magenta
 Colr(2) = 1        'blue
 Colr(3) = 2        'green
 Colr(4) = 14       'yellow
 Colr(5) = 4        'red
 Colr(6) = 12       'light red

 FOR Y = 1 TO 20    'create the data for graph

     FOR X = 30 TO 1 STEP -1
       SELECT CASE OSLfilm&(X - 1, Y - 1)
        CASE IS <= 10
         GraphData!(X, Y) = CSNG(OSLfilm&(X - 1, Y - 1))
         myvalue% = 1
        CASE IS <= 100
         GraphData!(X, Y) = 10 + CSNG(OSLfilm&(X - 1, Y - 1)) / 10!
         myvalue% = 2
        CASE IS <= 1000
         GraphData!(X, Y) = 20 + CSNG(OSLfilm&(X - 1, Y - 1)) / 100!
         myvalue% = 3
        CASE IS <= 10000
         GraphData!(X, Y) = 30 + CSNG(OSLfilm&(X - 1, Y - 1)) / 1000!
         myvalue% = 4
        CASE IS <= 100000
         GraphData!(X, Y) = 40 + CSNG(OSLfilm&(X - 1, Y - 1)) / 10000!
         myvalue% = 5
        CASE IS <= 1000000
         GraphData!(X, Y) = 50 + CSNG(OSLfilm&(X - 1, Y - 1)) / 100000!
         myvalue% = 6
        CASE ELSE
         GraphData!(X, Y) = 60 + CSNG(OSLfilm&(X - 1, Y - 1)) / 1000000!
         myvalue% = 6
       END SELECT
       IF GraphData!(X, Y) > 100 THEN GraphData!(X, Y) = 100
       Colors(Y + (X - 1) * 20) = Colr(myvalue%)
     NEXT
 NEXT


  CALL Prepare
  MainTitle$ = "Low-Resolution_OSL Image Reader"
  CALL Graph3(GraphData!(), MainTitle$, Colors(), -1)
  SurfacePlot% = -1
  isDirty = 0
END IF
```

```
RETURN

ComCheck:
 IF inCom% = 1 THEN RETURN
 inCom% = 1
 ' Initialize the string to an empty string
 Tries% = 0
 B1$ = ""
 ' Look for 80H

 IF LOC(1) > 0 THEN
  B1$ = INPUT$(1, #1)
 END IF
 A$ = INKEY$
 IF B1$ <> CHR$(128) THEN
  inCom% = 0
  RETURN
 END IF

 IF UCASE$(A$) = "Q" THEN
  inCom% = 0
  RETURN
 END IF
 COM(ComPort%) STOP   'Inhibit COM interrupt

 ' We've got data ready, send "ACK"
 PRINT #1, CHR$(6);
 ' Write status to screen
 IF CurrMenu = 2 THEN
  GOSUB GetCursorPosition
  LOCATE 44, 1: PRINT SPACE$(50);
  LOCATE 44, 1: PRINT "Reading Data...";
  LOCATE CursorRow%, CursorCol%
 END IF

GetData:
        Tries% = Tries + 1
        ' Wait up to 2 seconds for data to come in
        ct! = TIMER
        bt! = TIMER
        B1$ = ""
        WHILE LEN(B1$) < DATABLOCKSIZE AND (bt! - ct! < 5 AND bt! >= ct!)
         IF LOC(1) > 0 THEN
          myMes = INPUT$(LOC(1), #1)
          B1$ = B1$ + myMes
         END IF
         bt! = TIMER
        WEND
        IF (bt! - ct! < 5) AND (bt! >= ct!) THEN
         ' Calculate the check sum
         chksum& = 0
         FOR i% = 1 TO (DATABLOCKSIZE - 1)
          newVal& = ASC(MID$(B1$, i%, 1))
          chksum& = chksum& + newVal&
         NEXT
         'IF Tries = 1 THEN chksum& = 0
         IF ASC(MID$(B1$, DATABLOCKSIZE, 1)) <> (chksum& AND &HFF) THEN
```

```
' Bad check sum, so ask for resend or start over
PRINT #1, CHR$(21);
IF Tries = 1 THEN
 ' Write status to screen
 IF CurrMenu = 2 THEN
  GOSUB GetCursorPosition
  LOCATE 44, 1: PRINT SPACE$(50);
  LOCATE 44, 1: PRINT "Bad data, will retry";
  LOCATE CursorRow%, CursorCol%
 END IF
 GOTO GetData
ELSE
 ' Write status to screen
 IF CurrMenu = 2 THEN
  GOSUB GetCursorPosition
  LOCATE 44, 1: PRINT SPACE$(50);
  LOCATE 44, 1: PRINT "Bad data again, restarting";
  LOCATE CursorRow%, CursorCol%
 END IF
 inCom% = 0
 COM(ComPort%) ON
 RETURN
END IF
ELSE
 ' Data is good, send "ACK" and update display, log data to file
 PRINT #1, CHR$(6);
 TimeToUpdate% = -1

 ' Update Data
 DataCounter& = DataCounter& + 1
 IncrCounter& = ASC(LEFT$(B1$, 1))
 Temperature! = ASC(MID$(B1$, 2, 1))
 IF IncrCounter& = 0 THEN
  REDIM OSLfilm&(29, 19)
  REDIM OSLfread(29, 19) AS INTEGER
 END IF
 ' Add data to current array
 FOR i% = 1 TO 10
   myvalue&(i%) = CVL(MID$(B1$, 3 + ((i% - 1) * 4), 4))
   IF IncrCounter& < 38 THEN
         ' First half of sheet
         SELECT CASE i%
          CASE IS < 4
           Col% = IncrCounter&
           Row% = 5 + ((i% - 1) * 6)
          CASE 4, 5, 6
           Col% = IncrCounter& - 4
           Row% = 3 + ((i% - 4) * 6)
          CASE ELSE
           Col% = IncrCounter& - 8
           Row% = 1 + ((i% - 7) * 6)
         END SELECT
   ELSE
         ' Second half of sheet
         SELECT CASE i%
          CASE IS < 4
           Col% = 75 - IncrCounter&
```

```
                Row% = 4 + ((i% - 1) * 6)
               CASE 4, 5, 6
                Col% = 71 - IncrCounter&
                Row% = 2 + ((i% - 4) * 6)
               CASE ELSE
                Col% = 67 - IncrCounter&
                Row% = ((i% - 7) * 6)
              END SELECT
            END IF
            IF Col% >= 0 AND Row% >= 0 AND Col% < 30 AND Row% < 20 THEN
             OSLfilm&(Col%, Row%) = myvalue&(i%)
             OSLfread(Col%, Row%) = 1
            END IF
          NEXT
          PRINT #myFnum, TIME$, DATE$, IncrCounter&, Temperature!;

          FOR ic% = 1 TO 9
           PRINT #myFnum, myvalue&(ic%);
          NEXT
          PRINT #myFnum, myvalue&(10)

          ' Check IncrCounter& for end of scan
          IF IncrCounter& = 75 THEN
           ' Archive the array
           CurrentIndex% = CurrentIndex% + 1
           IF CurrentIndex% = 10 THEN CurrentIndex% = 0
           FOR Row% = 0 TO 19
            FOR Col% = 0 TO 29
             OSLdata&(Col%, Row%, CurrentIndex%) = OSLfilm&(Col%, Row%)
            NEXT
           NEXT
          END IF
         END IF
        ELSE
         ' Write status to screen
         IF CurrMenu = 2 THEN
            GOSUB GetCursorPosition
            LOCATE 44, 1: PRINT SPACE$(50);
            LOCATE 44, 1: PRINT "Time out!, restarting";
            LOCATE CursorRow%, CursorCol%
         END IF
        END IF

 inCom% = 0
 COM(ComPort%) ON
RETURN

SUB DrawButton (Xwide AS INTEGER, Y AS INTEGER, msg AS STRING)
  ' For Screen 13 only
  DIM txtRow AS INTEGER
  DIM gY AS INTEGER
  DIM gX AS INTEGER

  ' Calculate closest text box row
  txtRow = INT((CSNG(Y) / 100!) * 25)
  gY = (txtRow * 8) - 5
  gX = INT(((CSNG(Xwide) / 100!) * 320) / 2)
```

```
  ' Draw a frame centered on closest text box
  PSET ((160 - gX), (gY - 8))
  LINE -((160 + gX), (gY - 8))
  LINE -((160 + gX), (gY + 8))
  LINE -((160 - gX), (gY + 8))
  LINE -((160 - gX), (gY - 8))

  PSET ((160 - gX) + 2, (gY - 6))
  LINE -((160 + gX) - 2, (gY - 6))
  LINE -((160 + gX) - 2, (gY + 6))
  LINE -((160 - gX) + 2, (gY + 6))
  LINE -((160 - gX) + 2, (gY - 6))

  COLOR 181
  LOCATE txtRow, (20 - (LEN(msg) / 2))

  PRINT msg

END SUB


--
************************************************************************
******
-- OSL2touch.bas
--
************************************************************************
******
'
************************************************************************
*
'
' DVRFUNC.BAS (1.5) - Microsoft QuickBASIC 4.0 Interface with ELODEV 1.5
-
'                     the Elo device driver
'
' Copyright (C) 1988, 1990
'                     Elo TouchSystems, Inc.
'                     105 Randolph Road
'                     Oak Ridge, TN  37830
'                     (615) 482-4100
'
' Written on 1/08/88 by Michael Sigona
'
' Revision History:
'
'   02/11/88 - MRS - Converted from the TurboBasic version
'   05/24/88 - MRS - Upgraded for QuickBASIC 4.0
'   11/09/88 - MRS - Changed FindDriver to not check interrupt 67
'   11/05/90 - MRS - Changed comments to "ELODEV 1.5"
'
' Notes:
'
'   Compile with Microsoft QuickBASIC Version 4.0 or later. Other
versions
```

```
'    exist for Turbo Basic and BASICA/GWBASIC.
'
'    DVRFUNC.BAS requires the INTERRUPTX assembly language routine
'    found in the QB.QLB and QB.LIB library files included with
QuickBASIC.
'    Enter QuickBASIC with the command "QB DVRFUNC /l". The link option
will
'    load the library and automatically link it to DVRFUNC.
'
'
*************************************************************************
*

DECLARE FUNCTION FindDriver% ()
DECLARE SUB DriverInfo ()
DECLARE SUB OpenTouch ()
DECLARE SUB CloseTouch ()
DECLARE SUB EnableTouch ()
DECLARE SUB DisableTouch ()
DECLARE SUB SetMode (Mode%)
DECLARE SUB SetWaitTime (WaitTime%)
DECLARE FUNCTION BufferStatus% ()
DECLARE FUNCTION GetTouch% (X%, Y%, UT%)
DECLARE SUB FlushBuffer ()
DECLARE SUB InitDelay ()
DECLARE SUB ELODelay (Ms%)
DECLARE SUB ELOSound (Freq%)
DECLARE SUB NoSound ()
DECLARE SUB SetRange (Which%, XLow%, XHigh%, YLow%, YHigh%, ZLow%,
ZHigh%)
DECLARE SUB GetZAxis (Valid%, Z%)

' SetRange constants
CONST SETCALIBRANGE = 14, SETXLATERANGE = 15

' Mode bit constants
CONST BUFFERED = &H1, UNTOUCH = &H2, STREAM = &H4, XLATED = &H8
CONST ZONED = &H10

' Register declarations
' $INCLUDE: 'QB.BI'
DIM SHARED InRegsX AS RegTypeX, OutRegsX AS RegTypeX



'COMMON SHARED DriverInt%, InfoData AS InfoDataType
'$INCLUDE: 'Common'

FUNCTION BufferStatus%
  InRegsX.ax = 7
  CALL INTERRUPTX(DriverInt%, InRegsX, OutRegsX)
  BufferStatus% = OutRegsX.ax <> 0
END FUNCTION

SUB CloseTouch
  InRegsX.ax = 2
  CALL INTERRUPTX(DriverInt%, InRegsX, OutRegsX)
```

```
END SUB

SUB DisableTouch
  InRegsX.ax = 4
  CALL INTERRUPTX(DriverInt%, InRegsX, OutRegsX)
END SUB

SUB DriverInfo
  InRegsX.ax = 0
  InRegsX.ds = VARSEG(InfoData): InRegsX.bx = VARPTR(InfoData)
  CALL INTERRUPTX(DriverInt%, InRegsX, OutRegsX)
END SUB

SUB ELODelay (Ms%)
  InRegsX.ax = 11
  InRegsX.bx = Ms%
  CALL INTERRUPTX(DriverInt%, InRegsX, OutRegsX)
END SUB

SUB ELOSound (Freq%)
  InRegsX.ax = 12
  InRegsX.bx = Freq%
  CALL INTERRUPTX(DriverInt%, InRegsX, OutRegsX)
END SUB

SUB EnableTouch
  InRegsX.ax = 3
  CALL INTERRUPTX(DriverInt%, InRegsX, OutRegsX)
END SUB

FUNCTION FindDriver%
  FOR Vec% = &H60 TO &H66          ' scan interrupt vector table
      DEF SEG = 0
      SegPtr = PEEK(Vec% * 4 + 3) * 256 + PEEK(Vec% * 4 + 2)
      OffPtr = PEEK(Vec% * 4 + 1) * 256 + PEEK(Vec% * 4) + 2
                          ' pointing to ELODEV?
      DEF SEG = SegPtr
      Found% = -1
      FOR i% = 0 TO 5              ' see if string "ELODEV" is at address
      IF MID$("ELODEV", i% + 1, 1) <> CHR$(PEEK(OffPtr + i%)) THEN
Found% = 0
      NEXT i%
      IF Found% THEN FindDriver% = Vec%: EXIT FUNCTION  ' found at Vec%
  NEXT Vec%
  FindDriver% = 0                  ' not found
END FUNCTION

SUB FlushBuffer
  InRegsX.ax = 9
  CALL INTERRUPTX(DriverInt%, InRegsX, OutRegsX)
END SUB

FUNCTION GetTouch% (X%, Y%, UT%)
  InRegsX.ax = 8
  CALL INTERRUPTX(DriverInt%, InRegsX, OutRegsX)
  X% = OutRegsX.bx: Y% = OutRegsX.cx: UT% = OutRegsX.dx <> 0
  GetTouch% = OutRegsX.ax <> 0
```

```
END FUNCTION

SUB GetZAxis (Valid%, Z%)
  InRegsX.ax = 16
  CALL INTERRUPTX(DriverInt%, InRegsX, OutRegsX)
  Z% = OutRegsX.bx
  Valid% = OutRegsX.ax <> 0
END SUB

SUB InitDelay
  InRegsX.ax = 10
  CALL INTERRUPTX(DriverInt%, InRegsX, OutRegsX)
END SUB

SUB NoSound
  InRegsX.ax = 13
  CALL INTERRUPTX(DriverInt%, InRegsX, OutRegsX)
END SUB

SUB OpenTouch
  InRegsX.ax = 1
  CALL INTERRUPTX(DriverInt%, InRegsX, OutRegsX)
END SUB

SUB SetMode (Mode%)
  InRegsX.ax = 5
  InRegsX.bx = Mode%
  CALL INTERRUPTX(DriverInt%, InRegsX, OutRegsX)
END SUB

SUB SetRange (Which%, XLow%, XHigh%, YLow%, YHigh%, ZLow%, ZHigh%)
  DIM Range%(5)

  IF (Which% < SETCALIBRANGE) OR (Which% > SETXLATERANGE) THEN BEEP:
PRINT "Illegal SetRange Argument": END
  Range%(0) = XLow%: Range%(1) = XHigh%
  Range%(2) = YLow%: Range%(3) = YHigh%
  Range%(4) = ZLow%: Range%(5) = ZHigh%
  InRegsX.ax = Which%
  InRegsX.ds = VARSEG(Range%(0)): InRegsX.bx = VARPTR(Range%(0))
  CALL INTERRUPTX(DriverInt%, InRegsX, OutRegsX)
END SUB

SUB SetWaitTime (WaitTime%)
  InRegsX.ax = 6
  InRegsX.bx = WaitTime%
  CALL INTERRUPTX(DriverInt%, InRegsX, OutRegsX)
END SUB
************************************************************************
******
--    END OF OSL Image Reader Host PC Source Code
--
************************************************************************
******
```

```
************************************************************************
******
--     OSL Image Reader Firmware Source Code
--
************************************************************************
******
************************************************************************
******
-- Linke.cmd
--
************************************************************************
******
START.OBJ, MENU.OBJ, FPGA.OBJ, DC.OBJ, STEP.OBJ, RUN.OBJ, HOST.OBJ,
RAMX.OBJ,&
HSIO.OBJ, CCR.OBJ, PWM.OBJ, &
INTS.OBJ, SIO.OBJ, EQU.OBJ, &
..\LIBX\INT_VEC.OBJ, ..\LIBX\INT_VEC1.OBJ, &
..\LIBX\DEBUG.OBJ, SUBS.OBJ &
TO MAIN &
ROM(2000H-2013H(INT_VEC),  &
    2018H-2019H(CCR),      &
    2030H-203FH(INT_VEC1), &
    2080H-4DFFH(START),    &
    4E00H-56FFH(SUBS),     &
    5700H-5FFFH(DEBUG))    &
RAM(1AH-1FFH(STACK))       &
STACKSIZE(32) &
IXREF
EXIT


--
************************************************************************
******
-- Linke.bat
--
************************************************************************
******
@ECHO OFF
REM -----KB(USE 3FFF AS UPPER LIMIT IN LINKE.CMD)
REM -----KB(USE 5FFF AS UPPER LIMIT IN LINKE.CMD)

CLS
\INTEL\8096\ASM96\RL96 &< LINKE.CMD
@ECHO OFF

DEL MAIN.ICE
RENAME MAIN MAIN.ICE
DS N
DEL *.LST

--
************************************************************************
******
```

```
--
*************************************************************************
******
-- 8086_reg.inc
--
*************************************************************************
******
              EXTRN  AX,AH,AL,BX,BU,BL,CX,CH,CL,DX,DH,DL,BP,SI,DT


--
*************************************************************************
******
-- Subs.inc
--
*************************************************************************
******
;****************** SUBS CONDITIONAL ASSEMBLY ******************
; SUBS FILE HAS "IF & ENDIF" STATEMENTS                         *
; USED TO FREE UP ADDTIONAL EPROM                               *
;*****************************************************************
;IF,ELSE ASSEMBLY FOR DELAY LOOPS AT DIFFERENT CRYSTALS
MHz_12              EQU  0FFH
MHz_16              EQU  00H
SPEED               EQU  MHz_16
;*****************************************************************
;SUBROUTINES TO INCLUDE/DISCLUDE
YES                 EQU  0FFH
NO                  EQU  00H

xSND_NoBITS         EQU  NO
xSND_RBITS          EQU  NO

xASCII2_HEX         EQU  NO
xHEX_ASCII          EQU  NO
xENTR_ASCIID_HEX    EQU  YES

xJSTFY_ASCII        EQU  NO
xBCD_ASCII          EQU  NO

xSND_STRING         EQU  NO
xSEND_CR            EQU  NO
xSEND_COLON         EQU  NO
xFIND_COLON         EQU  NO
xFIND_COMMA         EQU  YES

xLF_LOOP            EQU  NO

xBYTESWAP_LOOP      EQU  NO
xSTRING_SWAP        EQU  NO
xSTOP_START         EQU  YES
xFORM_2SCOMP        EQU  NO
xFORM_2SCOMP1       EQU  NO
xSND_AD_160010      EQU  NO
xSND_AD_161010      EQU  NO
;*****************************************************************
```

```
--
*****************************************************************************
******
-- Sfr.inc
--
*****************************************************************************
******
              EXTRN   SP,IMASK,IPEND,SBUF,IOS0,IOS1,PWM0_CNTRL
              EXTRN   P0,P1,P2,P3,P4,BAUD_REG,AD_RESULT,IOS2,R0
              EXTRN   IOC1,SPCON,IMASK1,IPEND1,SP_STAT,CCR
              EXTRN   IOC0,IOC2,WSR,HSI_MODE,TIMER2,TIMER1,AD_CMND
              EXTRN   HSI_STATUS,HSI_TIME,HSO_CMND,HSO_TIME
              EXTRN   PWM1_CNTRL,PWM2_CNTRL,IOC3,AD_TIME

--
*****************************************************************************
******
-- Subs.src
--
*****************************************************************************
******
$TITLE('SUBS LIBRARY')
$PAGELENGTH(999)

;********************* SUBS-87C196 *********************
; LIBRARY OF GENERAL PURPOSE ROUTINES                   *
;    -HAS "CONDITIONAL ASSEMBLY" --- SEE SUBS.INC       *
;*****************************************************
;                 ---SUBROUTINES--                      *
;                                                       *
; BSND_BNCD    SEND BIG BIN-BCD  (32-BITS) (BP=PTR)     *
; BIN___BCD    32 BIT BINARY-BCD CONVERSION (BP=PTR)    *
;   SEND_DGTS  SEND 5 PACKED BYTES  (BP=PTR)            *
; BCD_SPLIT    SPLIT PACKED BCD BYTE  (CL=BCD BYTE)     *
; SEND_PAIR    SEND BCD PAIR TO VIDEO  (FROM CX)        *
; SND_BTDEC    SEND BIN_BCD (8  BITS,BP=PTR)            *
; SND_BNBCD    SEND BIN_BCD (16 BITS,BP=PTR)            *
; SND_BNBCD5   SEND BIN_BCD (5 DIGITS,AX=DATA)          *
; SND_BNBCD4   SEND BIN_BCD (4 DIGITS,AX=DATA)          *
; SND_BNBCD3   SEND BIN_BCD (3 DIGITS,AX=DATA)          *
; SND_BNBCD2   SEND BIN_BCD (2 DIGITS,AL=DATA)          *
; BIN_BCD      16 BIT BIN/BCD (ENTER BX=DATA)           *
;                             (RESULT:AL,CH,CL)         *
; BCD_BIN      5 DIGIT BCD/BIN (ENTER AL,CX--BX=RESULT)*
;                                                       *
; SND_NoBITS   SEND BIT STRING FROM AX,CL=#             *
; SND_BITS     SEND BIT STRING-TWO NIBBLES,AL,MSB 1ST   *
; SND_RBITS    SEND BIT STRING-TWO NIBBLES,AL,LSB 1ST   *
; SND_CY       SEND CARRY OUT AS A BIT 0 OR 1           *
; SND_TABLE    SEND EPROM TBL (SI=PTR,CH=#,CL=# SP)     *
; SND_BYTE     SEND HEX BYTE FROM AL                    *
; SND_WORD     SEND HEX WORD FROM AX                    *
; SND_BYTE_MEM SEND HEX BYTE FROM MEMORY [BP]+          *
; SND_WORD_MEM SEND HEX WORD FROM MEMORY [BP]+          *
; SND_DWORD_MEM SEND HEX DOUBLE WORD FROM MEMORY [BP]++*
;                                                       *
```

```
; KYBD_ENTR     ENTER UNTIL CR (BP=PTR)                          *
; KYBD_ENTR_1   ENTER ONLY 1 DIGIT (AL=RESULT)                   *
; WAIT_KYBD     LOOP UNTIL KEY STROKE(USES CY AS FLG)            *
;                                                                *
; ASCII_HEX         CONVERT ASCII TO HEX @ [BP]=D1               *
; ASCII2_HEX        (2 DIGITS,AX=D2D1,BX=RESULT)                 *
; HEX_ASCII         AX=DATA,DT=RESULT PTR @ MSD                  *
; ENTR_SHOW_BYTE    BP=DATA PTR,SI=MSG PTR                       *
; ENTR_SHOW_WORD    BP=DATA PTR,SI=MSG PTR                       *
; ENTR1_SHOW_BYTE AL=DATA     ,SI=MSG PTR                        *
; ENTR_SHOW_ASCIID_HEXB 5 DIGIT KYBD ENTRY (33,32=20H)           *
;                   ENTER BP=BYTE DATA PTR, SI=MSG PTR           *
; ENTR_SHOW_ASCIID_HEXW 5 DIGIT KYBD ENTRY (33,32=20H)           *
;                   ENTER BP=WORD DATA PTR, SI=MSG PTR           *
; ENTR_SHOW_ASCIID_HEX  5 DIGIT KYBD ENTRY (33,32=20H)           *
;                   ENTER AX=DATA, EXIT AX=RESULT                *
; ENTR_ASCIID_HEX KEYBOARD ENTRY (33,32=20H)                     *
;                   ENTER DL=# DIGITS(2-5),AX=RESULT             *
; ENTR_ASCII_HEX  KEYBOARD ENTRY (41,33=A3H)                     *
;                   RESULT=AL:DX   CL=# ENTRIES                  *
; ENTR_ASCII_HEX1 KEYBOARD ENTRY (NO CR ECHO)                    *
; JSTFY_ASCII       RIGHT JUSTIFY ASCII ENTERED CHRS             *
; BCD_ASCII         5 PACKED BCD BYTES TO 10 ASCII CHARS         *
;                                                                *
; SND_NULL      SEND STRING UNTIL NULL [SI]                      *
; SND_STRING    SEND STRING (CX=#,SI=PTR)                        *
; SEND_CR       SEND STRING THROUGH CR [SI]                      *
; SEND_COLON    SEND STRING THROUGH : [SI]                       *
; FIND_COLON    FIND :[SI]                                       *
; FIND_COMMA    FIND :[DT],CX=# CHARS                            *
;                                                                *
; CLR_SCRN      CLEAR VIDEO SCREEN                               *
; HOME          SEND VIDEO CURSOR "HOME"                         *
; CMND_ERROR    SEND MESSAGE "COMMAND ERROR"                     *
; SP_LOOP       SEND OUT SPACES (CL=#)                           *
; LF_LOOP       SEND OUT LINE FEEDS (CL=#)                       *
; SND_CR_LF     SEND CR & LF TO VIDEO                            *
; SND_LF        SEND LF TO VIDEO                                 *
; SND_CR        SEND CR TO VIDEO                                 *
; SND_EQUAL     SEND EQUAL SIGN TO VIDEO                         *
; SND_BKSP      SEND BACKSPACE TO VIDEO                          *
; SND_COMMA     SEND COMMA TO VIDEO                              *
; SND_SP        SEND SPACE TO VIDEO                              *
; SND_DASH      SEND DASH CHAR TO CRT                            *
; SND_SLASH     SEND SLASH CHAR TO CRT                           *
; SND_COLON     SEND COLON CHAR TO CRT                           *
; SND_PLUS      SEND PLUS CHAR TO CRT                            *
; SND_PERIOD    SEND PERIOD CHAR TO CRT                          *
; SND_QMARK     SEND QUESTION MARK TO CRT                        *
; SND_BEEP      SEND BEEP/BELL TONE TO CRT                       *
; SND_STAR      SEND * TO CRT                                    *
; SND_ACK       SEND ACK CODE                                    *
; SND_NAK       SEND NAK CODE                                    *
; CR_CONT       SEND MSG "CR=CONTINUE"                           *
;                                                                *
; CLR_MEM           CLEAR MEMORY (DT=PTR,CX=#)                   *
; FILL_MEM          FILL MEMORY (DT=PTR,AL=VALUE,CX=#)           *
```

```
; BYTESWAP_LOOP   SWAP MSB,LSB (BP=PTR,CX=# WORDS)      *
; STRING_SWAP     REVERSE STRING (SI=PTR,CX=# BYTES)    *
; STOP_START      STOP/START DISPLAY WITH KEYSTROKE     *
; FORM_2SCOMP     2'S COMP OF DW([SI]=DW,[DT]=RESULT)   *
; FORM_2SCOMP1    2'S COMP STRING OF DW (AX,DX)         *
; REVERSE_AL      MAKE D7=D0,D1=D6,ETC  (D5=AB)         *
; SHLC            ROTATE LEFT WITH CY, AL=VALUE, CL=#   *
;                                                       *
; DLY_10us        12MHz or 16 MHz (CONDITIONAL ASM)     *
; DLY_25us        12MHz or 16 MHz (CONDITIONAL ASM)     *
; DLY_100us                    "                        *
; DLY_1ms                      "                        *
; DLY_10ms                     "                        *
; DLY_100ms                    "                        *
; DLY_500ms                    "                        *
; DLY_1sec                     "                        *
;                                                       *
; EN_AD_FAST      ENABLE A/D FAST CONVERSION MODE        *
; RD_AD_10        AX=RESULT (CL=Ch #)  (10-BITS)         *
; RD_AD_8         AL=RESULT (CL=Ch #)  ( 8-BITS)         *
; RD_AD_KC8       KC CHIP (AL=RESULT,CL=CH #) (8 BITS)   *
; SND_AD_8        SEND A/D-8 BITS   (0-5V)               *
; SND_AD8_5       SEND A/D-8 BITS   (0-5V)               *
; SND_AD8_10      SEND A/D-8 BITS   (0-10)               *
; SND_AD8_20      SEND A/D-8 BITS   (0-20V)              *
; SND_AD8_X       SEND A/D-8 BITS   (BX=RANGE VALUE)     *
; SND_AD_10       SEND A/D-10 BITS  (0-5,10 V)           *
; SND_AD_160010   SEND A/D-16 BITS  (0-10V)              *
; SND_AD_161010   SEND A/D-16 BITS  (-10 TO +10V)        *
;*******************************************************
;GLOBAL DECLARATIONS
                PUBLIC  SND_BNBCD,BSND_BNCD,BCD_SPLIT,SND_QMARK,SND_BNBCD2
                PUBLIC  SEND_PAIR,CLR_MEM,SND_NULL,SND_BNBCD3,ASCII_HEX
                PUBLIC  SND_BTDEC,SND_PERIOD,RD_AD_10,SND_TABLE,SND_BITS
                PUBLIC  SP_LOOP,SND_NULL,SND_PLUS,SND_BEEP,SND_CR_LF,SHLC
                PUBLIC  KYBD_ENTR,CMND_ERROR,CLR_SCRN,SND_WORD_MEM,SND_CY
                PUBLIC  HOME,SND_BYTE,SND_BYTE_MEM,SND_COMMA,SND_EQUAL
                PUBLIC  SND_LF,SND_SP,CR_CONT,SND_AD_8,SND_DASH,RD_AD_8
                PUBLIC
BYTE_CHK,SND_BKSP,SND_WORD,SND_SLASH,SND_CR,SND_ACK
                PUBLIC
ENTR_ASCII_HEX1,ENTR_ASCII_HEX,KYBD_ENTR_1,FILL_MEM
                PUBLIC
BIN_BCD,DLY_100us,DLY_1ms,DLY_10ms,DLY_100ms,SND_NAK
                PUBLIC  ENTR_SHOW_BYTE,ENTR_SHOW_WORD,SND_AD8_20,SND_AD8_X
                PUBLIC
SND_AD_10,WAIT_KYBD,SND_BNBCD4,SND_COLON,SND_BNBCD5
                PUBLIC
ENTR1_SHOW_BYTE,EN_AD_FAST,RD_AD_KC8,DLY_1sec,DLY_500ms
                PUBLIC  ENTR_SHOW_ASCIID_HEX,ENTR_SHOW_ASCIID_HEXB,SND_1
                PUBLIC  ENTR_SHOW_ASCIID_HEXW,SND_STAR,DLY_25us,SND_0
                PUBLIC
SND_AD8_5,SND_AD8_10,REVERSE_AL,SND_DWORD_MEM,DLY_10us

;EXTERNAL DECLARATIONS
                EXTRN  VIDEO_OUT,SOFT0_IN
```

```
$INCLUDE(SUBS.INC)
$INCLUDE(8086_REGS.INC)
$INCLUDE(SFR.INC)
              EXTRN  CODECRT_CLR,CODECRT_HOME,RAM_SCRATCH,RAM_SIOFLG

CSEG
;*******************************************************



;NOTE: HAS PROBLEMS--DSPLYS 11H AS 10H
;********************* BSND_BNCD ***********************
; BIG SEND-BINARY TO BCD                                *
; COMPUTE 32 BINARY BITS TO BCD AND SEND                *
; ENTER with:BP=DATA POINTER @ LSB                      *
;*******************************************************
BSND_BNCD:    CALL BIN___BCD              ;RESULT @ RAM_SCRATCH
              LD   BP,#RAM_SCRATCH
              CALL SEND_DGTS              ;ENTER BP=PTR
              RET
;*******************************************************



;********************* BIN___BCD ***********************
; BIG BINARY TO BCD CONVERSION---32 BITS & UPWARD       *
; DATA=4 BYTES IN RAM   (LSB------MSB)                  *
; EXAMPLE:   [64,00,00,00] GOES TO [00,01,00,00,00]     *
; ENTER with:BP=DATA POINTER @ LSB                      *
; EXIT  with:RESULT @ RAM_SCRATCH  (5 PACKED BCD BYTES) *
; NOTE:DESTROYS VALUE @ [BP]                            *
;*******************************************************
BIN___BCD:    PUSH SI
              LD   SI,#RAM_SCRATCH
              CLR  AX                     ;AX=0000
              ST   AX,[SI]                ;CLR RESULT LOCATIONS
              ST   AX,2[SI]
              ST   AX,4[SI]

              LDB  DL,#20H                ;32 BITS
ROTATE:       PUSH BP                     ;BP=DATA IN PTR
              LD   CX,#0405H              ;CH=#BIN BYTES,CL=#BCD BYTES
              CLRC                        ;CY=0
SHIFT:        LDB  AL,[BP]                ;START WITH LSB OF INPUT
              ADDCB AL,AL                 ;SAME AS RCL
              STB  AL,[BP]+
              DJNZ CH,SHIFT
              LD   BP,#RAM_SCRATCH        ;RESULT PTR
BCDOC:        LDB  AL,[BP]
;MUST FIND IF THERE IS A HALF CARRY FOR DAA ROUTINE
              PUSHF                       ;SAVE CY
              CLR  SI                     ;CLR HALF-CY FLG
              POPF                        ;RETREIVE CY
              JBC  AL,3,NO_HALF1
              INC  SI                     ;SI=0001 FOR HALF-CY
NO_HALF1:     ADDCB AL,AL                 ;2 x BCD + CY
```

```
                CALL DAA
                STB  AL,[BP]+
                DJNZ CL,BCDOC               ;# OF BCD BYTES

                POP  BP                     ;BP=INPUT PTR
                JC   DONE
                DJNZ DL,ROTATE
DONE:           POP  SI
                RET
;*********************************************************



;************************ DAA ***************************
; DECIMAL ADJUST REG AL                                  *
;   IF AL & 0F > 9  OR HALF-CY=1 THEN AL=AL+6            *
;   IF AL > 9F > OR CY=1 THEN AL=AL+60                   *
; ENTER with:HALF CARRY IN SI                            *
;*********************************************************
DAA:            PUSHF                       ;SAVE CY
                CALL CHK_LONIB
                POPF
                CALL CHK_HINIB
                RET

CHK_LONIB:      JBS  SI,0,ADD_6             ;CHK FOR HALF-CY
                PUSH AX
                ANDB AL,#0FH
                CMPB AL,#09H
                POP  AX
                JH   ADD_6                  ;JMP IF CY=1
                RET

ADD_6:          ADDB AL,#06H
                RET

CHK_HINIB:      JC   ADD_60
                CMPB AL,#9FH
                JH   ADD_60
                RET

ADD_60:         ADDB AL,#60H
                SETC
                RET
;*********************************************************



;********************* SEND_DGTS ************************
; SEND 10 BCD DIGITS     (5 PACKED RAM BYTES)            *
; RAM DATA FORMAT:(D2D1,D4D3,------,D10D9)               *
; ENTER with:BP=DATA POINTER @ D2D1                      *
;*********************************************************
SEND_DGTS:      ADD  BP,#0004H      ;START WITH MSB
```

```
                LDB  DL,#05H          ;# BCD BYTES
GGG:            PUSH DX
                LDB  CL,[BP]
                PUSH BP
                CALL BCD_SPLIT
                CALL SEND_PAIR
                POP  BP
                POP  DX
                DEC  BP               ;MEMORY POINTER
                DJNZ DL,GGG           ;# BCD BYTES
                RET
;*******************************************************



;********************* BCD_SPLIT ***********************
; CONVERT PACKED BCD BYTE INTO TWO BYTES                *
; ENTER with:CL=BCD2,BCD1                               *
; EXIT with :CX=3BCD2,3BCD1                             *
;*******************************************************
BCD_SPLIT:      LDB  AL,CL            ;GET BCD2 & BCD1
                ANDB AL,#0FH          ;AL=0,BCD1
                ADDB AL,#30H          ;AL=3,BCD1
                LDB  AH,AL            ;AH=3,BCD1
                LDB  AL,CL
                SHRB AL,#4            ;AL=0,BCD2
                ADDB AL,#30H          ;AL=3,BCD2
                LDB  CH,AL
                LDB  CL,AH            ;CX=3BCD2,3BCD1
                RET
;*******************************************************



;********************* SEND_PAIR ***********************
; SEND 2 ASCII DIGITS TO VIDEO                          *
; ENTER with:CL=3LSD                                    *
;            CH=3MSD                                    *
;*******************************************************
SEND_PAIR:      LDB  AL,CH
                CALL VIDEO_OUT        ;SEND MSD
                LDB  AL,CL
                CALL VIDEO_OUT        ;SEND LSD
                RET
;*******************************************************



;********************* SND_BTDEC ***********************
; SEND BINARY BYTE OUT AS DECIMAL                       *
; BYTE IS IN MEMORY [BP]                                *
; ENTER with:BP=DATA PTR      [BP]                      *
; EXAMPLE: [FFH] IS OUTPUTED AS 255                     *
;*******************************************************
```

```
SND_BTDEC:    LDB  AL,[BP]          ;GET LSB
              CLRB AH               ;MSB=00
              CALL SND_BNBCD3       ;BIN/BCD & SEND
              RET
;*******************************************************



;********************* SND_BNBCD **********************
; SEND BINARY TO BCD-----5 DIGITS OUT                  *
; COMPUTE 16 BITS AND SEND 5 DIGITS                    *
; ENTER with:BP=DATA PTR   [BP]                        *
;*******************************************************
SND_BNBCD:    LD   BX,[BP]
SND_BNBCDx:   CALL BIN_BCD          ;FROM BX
              ADDB AL,#30H
              CALL VIDEO_OUT        ;SEND DIGIT #5
              PUSH CX
              LDB  CL,CH
              CALL BCD_SPLIT
              CALL SEND_PAIR        ;SEND D4 & D3
              POP  CX
              CALL BCD_SPLIT
              CALL SEND_PAIR        ;SEND D2 & D1
              RET
;*******************************************************



;********************* SND_BNBCD5 *********************
; SEND BINARY TO BCD----5 DIGITS OUT                   *
; COMPUTE 16 BITS AND SEND 4 DIGITS                    *
; ENTER with:AX=DATA                                   *
;*******************************************************
SND_BNBCD5:   LD   BX,AX
              CALL SND_BNBCDx
              RET
;*******************************************************



;********************* SND_BNBCD4 *********************
; SEND BINARY TO BCD----4 DIGITS OUT                   *
; COMPUTE 16 BITS AND SEND 4 DIGITS                    *
; ENTER with:AX=DATA                                   *
;*******************************************************
SND_BNBCD4:   LD   BX,AX
              CALL BIN_BCD
              PUSH CX
              LDB  CL,CH
              CALL BCD_SPLIT
              CALL SEND_PAIR              ;SEND D4 & D3
              POP  CX
              CALL BCD_SPLIT
```

```
                CALL SEND_PAIR              ;SEND D2 & D1
                RET
;*******************************************************



;********************* SND_BNBCD3 ***********************
; SEND BINARY TO BCD----3 DIGITS OUT                    *
; COMPUTE 16 BITS AND SEND 3 DIGITS                     *
; ENTER with:AX=DATA                                    *
;*******************************************************
SND_BNBCD3:     LD   BX,AX
                CALL BIN_BCD
;SEND D3
                LDB  AL,CH
                ANDB AL,#0FH                ;AL=0,BCD3
                ADDB AL,#30H                ;AL=3,BCD3
                CALL VIDEO_OUT
;SEND D2,D1
                CALL BCD_SPLIT
                CALL SEND_PAIR
                RET
;*******************************************************



;********************* SND_BNBCD2 ***********************
; SEND BINARY TO BCD---2 DIGITS OUT                     *
; COMPUTE 16 BITS AND SEND 2 DIGITS                     *
; ENTER with:AL=DATA                                    *
;*******************************************************
SND_BNBCD2:     LDB  BL,AL
                CLRB BU
                CALL BIN_BCD
                CALL BCD_SPLIT
                CALL SEND_PAIR
                RET
;*******************************************************



;********************** BIN_BCD ************************
; BINARY TO BCD CONVERSION---16 BITS                    *
; ENTER with:BX=DATA                                    *
; RESULT:CL=TENS,UNITS                                  *
;        CH=THOUS,HUNDS                                 *
;        AL=0,TEN THOUS                                 *
;*******************************************************
BIN_BCD:        PUSH SI
                LDB  DL,#11H                ;16 BITS + 1
                CALL BCD
                LDB  CL,AL
```

```
                LDB   DL,#11H
                CALL  BCD
                LDB   CH,AL
                LDB   AL,BL
                POP   SI
                RET

BCD:            CLRC
                CLRB  AL
CVT:            DECB  DL
                JE    BCD_RET
;MUST FIND IF THERE IS A HALF CARRY FOR DAA ROUTINE
                CLR   SI                        ;CLR HALF-CY FLG
                JBC   AL,3,NO_HALF
                INC   SI                        ;SI=0001 FOR HALF-CY
NO_HALF:        ADD   BX,BX
                ADDCB AL,AL
                CALL  DAA
                JNC   CVT
                INC   BX
                BR    CVT

BCD_RET:        RET
;*******************************************************




;********************** BCD_BIN ************************
; 5 DIGITS-65535 OR LESS                                *
; ENTER with:AL=TEN THOUS                               *
;           :CH=THOUS/HUNDRS                            *
;           :CL=TENS/UNITS                              *
; EXIT with :BX=RESULT       (ALTERS DX)                *.
;*******************************************************
BCD_BIN:        CLR   BX                        ;CLR RESULT LOCATION
                LD    DX,#2710H                 ;10,000
                CALL  RNIBL
                LDB   AL,CH
                LD    DX,#03E8H                 ;1,000
                CALL  LNIBL
                LD    DX,#0064H                 ;100
                CALL  RNIBL
                LDB   AL,CL
                LDB   DL,#0AH                   ;10
                CALL  LNIBL
                LDB   DL,#01H                   ;1

RNIBL:          CMPB  AL,#00H
                JE    RNIBL_RET
                ADD    BX,DX                    ;UPDATE RESULT
                DECB  AL
                BR    RNIBL
RNIBL_RET:      RET

LNIBL:          CMPB  AL,#0AH
```

```
                BE    STAY_LNIBL
                BNH   LNIBL_RET
STAY_LNIBL:     ADD   BX,DX                    ;UPDATE RESULT
                SUBB  AL,#10H
                BR    LNIBL
LNIBL_RET:      RET
;*******************************************************



IF xSND_NoBITS
;********************* SND_NoBITS ***********************

; SEND WORD OUT AS INDIVIDUAL BITS-MSB OUT FIRST       *
; ENTER with:AX=WORD                                   *
;            :CL=#   (16 MAX)                          *
; EXAMPLE   AX=0068   RESULT: 00000000001101000        *
;*******************************************************
                PUBLIC  SND_NoBITS

SND_NoBITS:     SHL   AX,#1                    ;MSB INTO CY
                CALL  SND_CY
                DJNZ  CL,SND_NoBITS
                RET
;*******************************************************
ENDIF



;********************** SND_BITS ************************
; SEND TWO NIBBLES AS BIT STRING-START WITH MSB        *
; ENTER with:AL=DATA TO BE SENT                        *
; EXAMPLE   AL=68     RESULT: 0110 1000                *
;*******************************************************
SND_BITS:       PUSH  CX
                LDB   CH,#02H          ;# NIBBLES
NXT_LNIBBLE:    LDB   CL,#04H          ;# BITS/NIBBLE
NXT_LBIT:       SHLB  AL,#1
                CALL  SND_CY           ;SND OUT CY
                DJNZ  CL,NXT_LBIT
                CALL  SND_SP
                DJNZ  CH,NXT_LNIBBLE
                POP   CX
                RET
;*******************************************************



IF xSND_RBITS
;********************** SND_RBITS ***********************
; SEND TWO NIBBLES AS BIT STRING-START WITH LSB        *
; ENTER with:AL=DATA TO BE SENT                        *
; EXAMPLE   AL=68     RESULT: 0001 0110                *
;*******************************************************
```

```
                PUBLIC  SND_RBITS

SND_RBITS:      PUSH CX
                LDB  CH,#02H          ;# NIBBLES
NXT_RNIBBLE:    LDB  CL,#04H          ;# BITS/NIBBLE
NXT_RBIT:       SHRB AL,#1
                CALL SND_CY           ;SND OUT CY
                DJNZ CL,NXT_RBIT
                CALL SND_SP
                DJNZ CH,NXT_RNIBBLE
                POP  CX
                RET
;*******************************************************
ENDIF



;********************** SND_CY *************************
; SEND CARRY OUT AS A BIT=0 OR 1                       *
;*******************************************************
SND_CY:         JC   SND_1
SND_0:          PUSH AX
                LDB  AL,#30H
                CALL VIDEO_OUT
                POP  AX
                RET

SND_1:          PUSH AX
                LDB  AL,#31H
                CALL VIDEO_OUT
                POP  AX
                RET
;*******************************************************



;********************** SND_TABLE ***********************
; SEND TABLE BASED IN EPROM                            *
; ENTER with:SI=LOOKUP POINTER                         *
;           :CL=# SPACES                               *
;           :CH=# SENDS                                *
;*******************************************************
SND_TABLE:      LDB  AL,[SI]+         ;GET LSB PTR
                LDB  AH,[SI]+         ;GET MSB PTR
                PUSH SI
                LD   SI,AX
                CALL SND_NULL
                PUSH CX
                CALL SP_LOOP
                POP  CX
                POP  SI
                DJNZ CH,SND_TABLE
                RET
;*******************************************************
```

```
;********************** SND_BYTE ***********************
; BYTE IN AL IN HEX OUT TO VIDEO                        *
; ENTER with:AL=BYTE                                    *
;*******************************************************
SND_BYTE:     PUSH AX
              SHRB AL,#4          ;DO MSD
              CALL BYTE_CHK
              POP  AX
              ANDB AL,#0FH        ;DO LSD
              CALL BYTE_CHK
              RET

BYTE_CHK:     CMPB AL,#09H
              JH   GREATER_9
              ADDB AL,#30H        ;DECIMAL CODE
              CALL VIDEO_OUT
              RET

GREATER_9:    SUBB AL,#09H
              ADDB AL,#40H        ;ALPHABET CODE
              CALL VIDEO_OUT
              RET
;*******************************************************



;********************** SND_WORD ***********************
; SEND HEX WORD FROM AX TO VIDEO                        *
; ENTER with:AX=BYTE                                    *
;*******************************************************
SND_WORD:     PUSH AX
              LDB  AL,AH          ;SEND MSD
              CALL SND_BYTE
              POP  AX             ;SEND LSD
              CALL SND_BYTE
              RET
;*******************************************************



;******************** SND_BYTE_MEM *********************
; BYTE IN POINTED TO BY BP                              *
; ENTER with:BP=BYTE PTR   (BP INCREMENTED)             *
;*******************************************************
SND_BYTE_MEM: PUSH AX
              LDB  AL,[BP]+
              CALL SND_BYTE
              POP  AX
              RET
;*******************************************************
```

```
;******************** SND_WORD_MEM *********************
; ENTER with:BP=PTR @ LSB                               *
; EXIT  with:BP=BP+2                                    *
;*******************************************************
SND_WORD_MEM: PUSH AX
              LD   AX,[BP]+
              CALL SND_WORD
              POP  AX
              RET
;*******************************************************




;******************** SND_DWORD_MEM ********************
; ENTER with:BP=PTR @ LSB                               *
; EXIT  with:BP=BP+4                                    *
;*******************************************************
SND_DWORD_MEM: PUSH AX
              LD   AX,[BP]+
              PUSH AX
              LD   AX,[BP]+
              CALL SND_WORD              ;UPPER WORD
              POP  AX
              CALL SND_WORD              ;LOWER WORD
              POP  AX
              RET
;*******************************************************



;********************* KYBD_ENTR ***********************
; ACCOMODATES BACKSPACE,DELETE & ESCAPE KEYS            *
; IF 1ST CHAR=CR THEN BP IS NOT INCREMENTED & CL=00     *
; IGNORES 'NULL' ENTRIES                                *
;                                                       *
; ENTER with:BP=DATA POINTER                            *
; EXIT with :CL=# ENTRIES COUNTER(DOES NOT CNT CR)      *
;           :BP=1 ADDRS PAST CR                         *
;*******************************************************
KYBD_ENTR:    CLRB CL              ;CL=# ENTRIES CNTR
RETRY_KYBD:   CALL WAIT_KYBD
              CMPB AL,#0DH         ;CHK FOR CR
              BE   KRET            ;RET FIRST CHAR=CR
              BR   KSKIP
KDELETE:      CALL SND_SP          ;SEND SPACE TO WIPE OUT CHAR
              CALL SND_BKSP        ;SEND BACKSPACE
              BR   KNXT
KBKSP:        DECB CL
              DEC  BP
KNXT:         CALL WAIT_KYBD
              CMPB AL,#00H         ;IGNORE NULL CHAR ENTRY
```

```
                JE    KNXT
KSKIP:          CMPB  AL,#7FH          ;CHK FOR DELETE KEY
                JE    KDELETE
                CMPB  AL,#08H          ;CHK FOR BACKSPACE
                JE    KBKSP
                CMPB  AL,#1BH          ;CHK FOR ESCAPE KEY
                JE    KESCAPE
                STB   AL,[BP]+         ;SAVE KEYSTROKE
KESCAPE:        INCB  CL
                CMPB  AL,#0DH          ;END WITH CR-CR STORED
                JNE   KNXT
                DECB  CL               ;DO NOT COUNT CR
KRET:           RET
;*********************************************************



;******************** KYBD_ENTR_1 ************************
; ENTER ONLY 1 DIGIT                                     *
; EXIT with :CL=# ENTRIES COUNTER                        *
;           :AL=DIGIT ENTERED                            *
;*********************************************************
KYBD_ENTR_1:    PUSH BP
                LD    BP,#RAM_SCRATCH
                CALL  KYBD_ENTR
                LD    BP,#RAM_SCRATCH
                LDB   AL,[BP]
                POP   BP
                RET
;*********************************************************



;********************** WAIT_KYBD ************************
; LOOP UNTIL KEYSTROKE-KYBD INTERRUPT SETS CY            *
; EXIT with: AL=CHAR                                     *
;*********************************************************
WAIT_KYBD:      JBC   RAM_SIOFLG,0,HARD_UART
                BR    SOFT0_IN                 ;AL=RESULT

HARD_UART:      CLRC                           ;CY=0
LOOP_KYBD:      JNC   LOOP_KYBD                ;CY SET IN INTERRUPT ROUTINE
                RET
;*********************************************************



;********************** ASCII_HEX ************************
; ASCII DECIMAL-NO LETTERS                               *
; RAM=3D6,------3D2,3D1  (HIGH RAM END=D2D1)              *
; EXAMPLE                                                *
;       RAM=35,30                                        *
;       RESULT BX=0032                                   *
```

```
; MAX IN=65535                                                  *
; 5 DIGITS OF ASCII                                            *
; ENTER with:BP=PTR @ 3D1                                      *
;           :DL=# PAIRS  (3 MAX-DO TO BCD-BIN CONV)            *
; EXIT with :BX=RESULT                                         *
;**************************************************************
ASCII_HEX:    CLR  CX
              CALL LD_2DIGITS
              CLRB AL
              DECB DL              ;# PAIRS-3 MAX
              JE   CONVT
              LDB  CH,CL
              CALL LD_2DIGITS
              LDB  AL,CL
              LDB  CL,CH
              LDB  CH,AL
              CLRB AL
              DECB DL              ;# PAIRS
              JE   CONVT
              LDB  AL,[BP]
              ANDB AL,#0FH
CONVT:        CALL BCD_BIN
              RET

LD_2DIGITS:   LDB  AL,[BP]
              ANDB AL,#0FH         ;MASK UPPER NIBBLE
              LDB  CL,AL
              DEC  BP
              LDB  AL,[BP]
              SHLB AL,#4           ;MASK UPPER NIBBLE
              ADDB AL,CL
              LDB  CL,AL
              DEC  BP
              RET
;**************************************************************



IF xASCII2_HEX
;********************* ASCII2_HEX ***********************
; 2 ASCII DECIMAL-NO LETTERS                                   *
; EXAMPLE                                                      *
;      AX=32,35                                                *
;      RESULT BX=0016                                          *
; ENTER with:AX=3D23D1                                         *
; EXIT with :BX=RESULT                                         *
;**************************************************************
              PUBLIC  ASCII2_HEX

ASCII2_HEX:   ANDB AL,#0FH                ;MASK OFF UPPER NIBBLE
              ANDB AH,#0FH                ;MASK OFF UPPER NIBBLE
              SHLB AH,#4
              ADDB AL,AH
              LDB  CL,AL
              CLRB CH
              CLRB AL
```

```
                CALL BCD_BIN                    ;ENTER:AL=D5,CH=D4D3,CL=D2D1
                                                ;BX=RESULT
                RET
;*******************************************************
ENDIF


IF xHEX_ASCII
;********************* HEX_ASCII ***********************
; ASCII DECIMAL-NO LETTERS                              *
; EXAMPLE:AX=0064 (100 DECIMAL)                         *
;         [DT+00]=30H      D5                           *
;         [  +01]=30H      D4                           *
;         [  +02]=31H      D3                           *
;         [  +03]=30H      D2                           *
;         [  +04]=30H      D1                           *
; 5 DIGITS OF ASCII                                     *
; ENTER with:DT=RESULT PTR @ 3D5                        *
;           :AX=DATA IN BINARY HEX  (16 BITS)           *
; EXIT with :RESULT @ DT PTR                            *
;*******************************************************
                PUBLIC  HEX_ASCII

HEX_ASCII:      LD    BX,AX
                CALL BIN_BCD
                ADDB AL,#30H
                STB  AL,[DT]+                   ;STORE D5
                PUSH CX
                LDB  CL,CH
                CALL BCD_SPLIT                  ;RESULT CX=3D4,3D3
                STB  CH,[DT]+                   ;STORE D4
                STB  CL,[DT]+    ;STORE D3
                POP  CX
                CALL BCD_SPLIT
                STB  CH,[DT]+    ;STORE D2
                STB  CL,[DT]+    ;STORE D1
                RET
;*******************************************************
ENDIF


;******************* ENTR_SHOW_BYTE ********************
; SHOW OLD BYTE & ENTER NEW BYTE                        *
; CONVERTS ASCII NUMBERS TO HEX                         *
; MAX ENTRY=FF                                          *
; EXAMPLES:ENTER THE #12 (31,32)                        *
;          RESULT AX=000C                               *
; ENTER with:BP=RESULT PTR                              *
;           :SI=MSG PTR                                 *
; EXIT with :[BP]=RESULT                                *
;            BP PTS AHEAD                               *
;*******************************************************
ENTR_SHOW_BYTE:
                PUSH SI
                CALL SND_NULL                   ;SND MESSAGE
```

```
                LDB  AL,[BP]                ;DSPLY PREVIOUS VALUE
                CALL SND_BYTE               ;       "
                CALL SND_BKSP
                CALL SND_BKSP
;ENTER NEW VALUE
                CALL ENTR_ASCII_HEX         ;RESULT=AL:DX,CL=# ENTRIES
                CMPB CL,#00H                ;CHK FOR NO ENTRIES
                BE   EXIT_BYTE
                CMPB CL,#02H
                BE   SAVE_BYTE
                POP  SI
                BR   ENTR_SHOW_BYTE
;SAVE ENTERED VALUE
SAVE_BYTE:      STB  DL,[BP]                ;SAVE VALUE
EXIT_BYTE:      ADD  SP,#0002H
                INC  BP
                RET
;*******************************************************



;****************** ENTR_SHOW_WORD ********************
; SHOW OLD WORD & ENTER NEW WORD                       *
; CONVERTS ASCII NUMBERS TO HEX                        *
; MAX ENTRY=FFFF                                       *
; EXAMPLES:ENTER THE #0012 (30,30,31,32)               *
;          RESULT AX=000C                              *
;          ENTER THE #0100 (30,31,30,30)               *
;          RESULT AX=0064                              *
; ENTER with:BP=RESULT PTR                             *
;           :SI=MSG PTR                                *
; EXIT with :[BP]=RESULT                               *
;            BP PTS AHEAD                              *
;*******************************************************
ENTR_SHOW_WORD:
                PUSH SI
                CALL SND_NULL               ;SND MESSAGE
                LD   AX,[BP]                ;DSPLY PREVIOUS VALUE
                CALL SND_WORD               ;       "
                CALL SND_BKSP
                CALL SND_BKSP
                CALL SND_BKSP
                CALL SND_BKSP
;ENTER NEW VALUE
                CALL ENTR_ASCII_HEX         ;RESULT=AL:DX,CL=# ENTRIES
                CMPB CL,#00H                ;CHK FOR NO ENTRIES
                BE   EXIT_WORD
                CMPB CL,#04H
                BE   SAVE_WORD
                POP  SI
                BR   ENTR_SHOW_WORD
;SAVE ENTERED VALUE
SAVE_WORD:      ST   DX,[BP]                ;SAVE VALUE
EXIT_WORD:      ADD  SP,#0002H
                INC  BP
                INC  BP
```

```
              RET
;*********************************************************



;******************** ENTR1_SHOW_BYTE *******************
; SHOW OLD BYTE & ENTER NEW BYTE                         *
; CONVERTS ASCII NUMBERS TO HEX                          *
; MAX ENTRY=FF                                           *
; EXAMPLES:ENTER THE #12 (31,32)                         *
;          RESULT AX=000C                                *
; ENTER with:AL=DATA                                     *
;           :SI=MSG PTR                                  *
; EXIT with :AL=RESULT                                   *
;*********************************************************
ENTR1_SHOW_BYTE:
              PUSH SI
              CALL SND_NULL              ;SND MESSAGE
              CALL SND_BYTE              ;       "
              CALL SND_BKSP
              CALL SND_BKSP
;ENTER NEW VALUE
              CALL ENTR_ASCII_HEX        ;RESULT=AL:DX,CL=# ENTRIES
              CMPB CL,#00H               ;CHK FOR NO ENTRIES
              BE   EXIT1_BYTE
              CMPB CL,#02H
              BE   SAVE1_BYTE
              POP  SI
              BR   ENTR1_SHOW_BYTE
;SAVE ENTERED VALUE
SAVE1_BYTE:   LDB  AL,DL                 ;SAVE VALUE (DL=BYTE #1)
EXIT1_BYTE:   ADD  SP,#0002H
              RET
;*********************************************************



;****************** ENTR_SHOW_ASCIID_HEXB ***************
; CONVERTS ASCII NUMBERS TO HEX                          *
; ALWAYS 5 DIGITS                                        *
; MAX ENTRY=65535                                        *
; EXAMPLES:ENTER THE #00012 (30,30,30,31,32)             *
;          RESULT AX=000C                                *
;          ENTER THE #00100 (30,30,31,30,30)             *
;          RESULT AX=0064                                *
; ENTER with:BP=RESULT BYTE PTR                          *
;           :SI=MSG PTR                                  *
; EXIT with :[BP]=RESULT BYTE                            *
;            BP PTS AHEAD                                *
;*********************************************************
ENTR_SHOW_ASCIID_HEXB:
              PUSH AX
              PUSH BP
              LDB  AL,[BP]
              CLRB AH
```

```
              CALL ENTR_SHOW_ASCIID_HEX     ;AX=RESULT
              POP  BP
              STB  AL,[BP]
              POP  AX
              RET
;*******************************************************


;****************** ENTR_SHOW_ASCIID_HEXW **************
; CONVERTS ASCII NUMBERS TO HEX                         *
; ALWAYS 5 DIGITS                                       *
; MAX ENTRY=65535                                       *
; EXAMPLES:ENTER THE #00012 (30,30,30,31,32)            *
;          RESULT AX=000C                               *
;          ENTER THE #00100 (30,30,31,30,30)            *
;          RESULT AX=0064                               *
; ENTER with:BP=RESULT WORD PTR                         *
;            :SI=MSG PTR                                *
; EXIT with :[BP]=RESULT WORD                           *
;            BP PTS AHEAD                               *
;*******************************************************
ENTR_SHOW_ASCIID_HEXW:
              PUSH AX
              PUSH BP
              LD   AX,[BP]
              CALL ENTR_SHOW_ASCIID_HEX     ;AX=RESULT
              POP  BP
              ST   AX,[BP]
              POP  AX
              RET
;*******************************************************


;****************** ENTR_SHOW_ASCIID_HEX ***************
; CONVERTS ASCII NUMBERS TO HEX                         *
; ALWAYS 5 DIGITS                                       *
; MAX ENTRY=65535                                       *
; EXAMPLES:ENTER THE #00012 (30,30,30,31,32)            *
;          RESULT AX=000C                               *
;          ENTER THE #00100 (30,30,31,30,30)            *
;          RESULT AX=0064                               *
; ENTER with:AX=PAST RESULT                             *
; EXIT with :AX=RESULT                                  *
;            CL=# ENTERED                               *
;*******************************************************
ENTR_SHOW_ASCIID_HEX:
              PUSH SI
              PUSH AX
              CALL SND_NULL                 ;SND MESSAGE
              CALL SND_BNBCD5               ;       "
              CALL SND_BKSP
              CALL SND_BKSP
              CALL SND_BKSP
```

```
                CALL SND_BKSP
                CALL SND_BKSP
;ENTER NEW VALUE
                LDB  DL,#05H
                CALL ENTR_ASCIID_HEX
                CMPB CL,#00H                ;CHK FOR NO ENTRIES
                BE   EXIT1_ACD
                CMPB CL,#05H
                BE   SAVE1_ACD
                POP  AX
                POP  SI
                BR   ENTR_SHOW_ASCIID_HEX

;SAVE ENTERED VALUE
SAVE1_ACD:      ADD  SP,#0004H
                RET
EXIT1_ACD:      POP  AX
                ADD  SP,#0002H
                RET
;*****************************************************



IF xENTR_ASCIID_HEX
;****************** ENTR_ASCIID_HEX ********************
; CONVERTS ASCII NUMBERS TO HEX                       *
; MAX ENTRY=65535                                     *
; EXAMPLES:ENTER THE #12 (31,32)                      *
;          RESULT AX=000C                             *
;          ENTER THE #100 (30,31,30,30)               *
;          RESULT AX=0064                             *
; ENTER with:DL=# DIGITS TO BE ENTERED  (2-5 DIGITS)  *
; EXIT with :AX=RESULT                                *
;            CL=# ENTERED                             *
;*****************************************************
                PUBLIC  ENTR_ASCIID_HEX

ENTR_ASCIID_HEX: PUSH BP
                LD   BP,#RAM_SCRATCH
                CLRB AL
                STB  AL,[BP]                ;SET MSD=0
                LD   BP,#RAM_SCRATCH+1
                CALL KYBD_ENTR
                CMPB CL,#00H                ;CHK FOR NO ENTRIES
                BE   ASCIID_RET
                LD   BP,#RAM_SCRATCH+1
                ADDB BP,DL                  ;PT TO D1
                DECB BP                     ;
                JBS  DL,0,A_ODD             ;CHK IF ENTRY EVEN OR ODD
                SHRB DL,#1                  ;DIVIDE/2 TO GET # PAIRS
                PUSH CX
                CALL ASCII_HEX              ;BX=RESULT
                LD   AX,BX
                POP  CX                     ;CL=# ACTUAL ENTRIES
ASCIID_RET:     POP  BP
                RET
```

```
A_ODD:          DECB DL                         ;GET # PAIRS
                PUSH CX
                CALL ASCII_HEX                  ;BX=RESULT
                LD   AX,BX
                POP  CX                         ;CL=# ACTUAL ENTRIES
                POP  BP
                RET
;*****************************************************
ENDIF


;****************** ENTR_ASCII_HEX *********************
; CONVERTS ASCII NUMBERS AND LETTERS TO PACKED HEX      *
; ONLY EVEN # OF DIGITS MAY BE ENTERED  (6 MAX)         *
; EXAMPLE:                                              *
;       ENTER ASCII PAIR F1 AS (46,31)                  *
;       RESULT:DL=F1                                    *
; RESULT                                                *
;       AL=BYTE 3                                       *
;       DH=BYTE 2                                       *
;       DL=BYTE 1                                       *
;       CL=# KYBD ENTRIES                               *
;*****************************************************
ENTR_ASCII_HEX: PUSH BP
                LD   BP,#RAM_SCRATCH
                CALL KYBD_ENTR
                PUSH CX                         ;SAVE # DIGITS ENTERED
                CMPB CL,#00H                    ;CHK IF FIRST CHAR=CR
                JE   EASCII_RET
                CMPB CL,#01H                    ;ENTER Z=EXIT (1 CHAR)
                JE   EASCII_RET
                DEC  BP                         ;POINT TO LSD
                DEC  BP
                SHRB CL,#1      ;DIV CL BY 2
                CLR  DX                         ;CLR RESULT LOCATION
                CLRB AL                         ;CLR RESULT LOCATION
;DO FIRST PAIR
                PUSH CX
                CALL LD_DGTS_ASCII              ;RESULT IN CL
                LDB  DL,CL
                POP  CX
                DECB CL
                JE   EASCII_RET
;DO SECOND PAIR
                PUSH CX
                CALL LD_DGTS_ASCII              ;CL=RESULT
                LDB  DH,CL
                POP  CX
                DECB CL
                JE   EASCII_RET
;DO THIRD PAIR
                CALL LD_DGTS_ASCII              ;CL=RESULT
                LDB  AL,CL
EASCII_RET:     POP  CX
                POP  BP
```

```
            RET
;*********************************************************



;****************** ENTR_ASCII_HEX1 ********************
; NO ECHO OF CR                                          *
; CONVERTS ASCII NUMBERS AND LETTERS TO PACKED HEX       *
; ONLY EVEN # OF DIGITS MAY BE ENTERED  (6 MAX)          *
; EXAMPLE:                                               *
;       ENTER ASCII PAIR F1 AS (46,31)                   *
;       RESULT:DL=F1                                     *
; RESULT                                                 *
;       AL=D6D5                                          *
;       DH=D4D3                                          *
;       DL=D2D1                                          *
;       CL=# DIGITS ENTERED                              *
;*********************************************************
ENTR_ASCII_HEX1: PUSH BP
             LD   BP,#RAM_SCRATCH
             CALL KYBD_ENTR
             DEC  BP                      ;POINT TO LSD
             DEC  BP
             PUSH CX                      ;SAVE # DIGITS ENTERED
             SHRB CL,#1                   ;DIV CL BY 2
             CLR  DX                      ;CLR RESULT LOCATION
             CLRB AL                      ;CLR RESULT LOCATION
;DO FIRST PAIR
             PUSH CX
             CALL LD_DGTS_ASCII           ;RESULT IN CL
             LDB  DL,CL
             POP  CX
             DECB CL
             JE   ENT_ASCIIRET
;DO SECOND PAIR
             PUSH CX
             CALL LD_DGTS_ASCII           ;CL=RESULT
             LDB  DH,CL
             POP  CX
             DECB CL
             JE   ENT_ASCIIRET
;DO THIRD PAIR
             CALL LD_DGTS_ASCII           ;CL=RESULT
             LDB  AL,CL
ENT_ASCIIRET: POP  CX                     ;CL=# DIGITS ENTERED
             POP  BP
             RET
;*********************************************************



;******************* LD_DGTS_ASCII *********************
; DIGITS MAY BE EITHER ASCII NUMBERS OR ASCII LETTERS    *
; EXAMPLE: MEMORY=31,41                                  *
;          REG CL=1A                                     *
```

```
; ENTER with:BP=WORD PTR @ TOP BYTE                          *
; EXIT with:CL=RESULT                                        *
;          :BP POINTING 1 BYTE LOWER                         *
;*************************************************************
LD_DGTS_ASCII: PUSH AX
               LDB  AL,[BP]
               JBC  AL,6,ASCII_No          ;CHK IF IN 30'S OR 40'S
               ADDB AL,#09H                ;CONVERT LETTERS
ASCII_No:      ANDB AL,#0FH
               LDB  BL,AL
               DEC  BP
               LDB  AL,[BP]
               JBC  AL,6,ASCII_No1
               ADDB AL,#09H
ASCII_No1:     ANDB AL,#0FH
               SHLB AL,#4
               ADDB AL,BL
               LDB  CL,AL
               DEC  BP
               POP  AX
               RET
;*************************************************************



IF xJSTFY_ASCII
;******************* JSTFY_ASCII ***********************
; ENTER with:BP=START OF DIGITS --POINTS TO MSD              *
;           :AL=RQD # OF DIGITS                              *
;           :CL=ACTUAL # DIGITS ENTERED                      *
; RESULT EXAMPLE   D3,D2,D1===30,D3,D2,D1                   *
;*************************************************************
               PUBLIC  JSTFY_ASCII

JSTFY_ASCII:   PUSH BP              ;AL=RQD # DIGITS
               SUBB AL,CL           ;CL=# DIGITS ENTERED
               LDB  CH,AL
               LDB  AL,BL           ;ADJUST DI POINTER
               ADDB AL,CH           ;       '
               LDB  DL,AL           ;       '
               LDB  DH,BU           ;       '
               LD   DT,DX           ;       '
               PUSH CX
               CLRB CH
               LD   SI,BP
NXT_JASCII:    BMOV SI,CX           ;SI=SRC PTR,DT=DST PTR
               DEC  CX
               JNE  NXT_JASCII
               LDB  AL,#0DH
               STB  AL,[DT]         ;STORE CR
               POP  CX
               POP  BP
               LDB  AL,#30H         ;ASCII ZERO FILLOUT
               LD   DT,BP
               LDB  CL,CH
               CLRB CH
```

```
NXT_JASCII1:  BMOV SI,CX
              DEC  CX
              JNE  NXT_JASCII1
              RET
;*******************************************************
ENDIF


IF xBCD_ASCII
;******************** BCD_ASCII ************************
; PACKED BCD BYTES IN MEM TO SINGLE DIGIT ASCII IN MEM *
; 5 PACKED BCD BYTES TO 10 ASCII DIGITS                *
; EXAMPLE: [34,12,00,00,00] GOES TO [33,34,31,32,30...]*
;                                                      *
; ENTER with:BP=BCD DATA PTR                           *
; EXIT  with:ASCII RESULT @ RAM_SCRATCH                *
;*******************************************************
              PUBLIC  BCD_ASCII

BCD_ASCII:    LD   DT,#RAM_SCRATCH       ;RESULT PTR
              LDB  CL,#05H               ;# BCD BYTES
NXTBCDASCII:  PUSH CX
              LDB  CL,[BP]+
              CALL BCD_SPLIT
              ST   CX,[DT]+
              POP  CX
              DJNZ CL,NXTBCDASCII
              RET
;*******************************************************
ENDIF


;********************* SND_NULL ************************
; SEND ASCII STRING FROM EPROM UNTIL NULL CHAR         *
; ENTER with:SI=PTR                                    *
; EXIT with :SI POINTS AT NULL CHAR                    *
;*******************************************************
SND_NULL:     PUSH AX
SND_NULL1:    LDB  AL,[SI]+              ;GET MEM VALUE
              CALL VIDEO_OUT
              CLRB AL
              CMPB AL,[SI]               ;CHK MEM TO NULL
              JNE  SND_NULL1
              POP  AX
              RET
;*******************************************************


IF xSND_STRING
;********************* SND_STRING **********************
; SEND ASCII STRING                                    *
; ENTER with:CX=# CHARS                                *
;           :BP=PTR                                    *
```

```
; EXIT with :BP POINTS AT NEXT CHAR                              *
;*****************************************************
                PUBLIC  SND_STRING

SND_STRING:     LDB  AL,[BP]+                ;GET MEM VALUE
                CALL VIDEO_OUT
                DEC  CX
                JNE  SND_STRING
                RET
;*****************************************************
ENDIF


IF xSEND_CR
;********************** SEND_CR ************************
; SEND STRING THROUGH CR                                         *
; ENTER with:SI=PTR                                              *
; EXIT with :SI 1 ADDRS PAST CR                                  *
;*****************************************************
                PUBLIC  SEND_CR

SEND_CR:        LDB  AL,[SI]+
                PUSH AX
                CALL VIDEO_OUT
                POP  AX
                CMPB AL,#0DH
                JNE  SEND_CR
                RET
;*****************************************************
ENDIF


IF xSEND_COLON
;********************** SEND_COLON **********************
; SEND STRING THROUGH COLON                                      *
; ENTER with:SI=PTR                                              *
; EXIT with :SI 1 ADDRS PAST COLON                               *
;*****************************************************
                PUBLIC  SEND_COLON

SEND_COLON:     LDB  AL,[SI]+
                PUSH AX
                CALL VIDEO_OUT
                POP  AX
                CMPB AL,#3AH
                JNE  SEND_COLON
                RET
;*****************************************************
ENDIF


IF xFIND_COLON
;********************** FIND_COLON **********************
; FIND : IN EPROM STRING                                         *
```

```
; ENTER with:SI=PTR                                              *
; EXIT with :SI 1 ADDRS PAST COLON                               *
;***************************************************
              PUBLIC  FIND_COLON

FIND_COLON:   INC  SI
              LDB  AL,#3AH
              CMPB AL,[SI]                ;3AH=COLON
              JNE  FIND_COLON
              INC  SI                     ;PT NXT CHAR PAST :
              RET
;***************************************************
ENDIF


IF xFIND_COMMA
;************************* FIND_COMMA ************************
; FIND ,                                                       *
; ENTER with:DT=PTR                                            *
; EXIT with :DT 1 ADDRS PAST COMMA                             *
;           :CX # CHARS SEARCHED                               *
;*************************************************************
              PUBLIC  FIND_COMMA

FIND_COMMA:   CLR  CX
NXT_COMMA:    LDB  AL,[DT]+
              CMPB AL,#2CH                ;2CH=COMMA
              JE   FND_COMMA
              INC  CX
              BR   NXT_COMMA

FND_COMMA:    RET
;*************************************************************
ENDIF


;********************* CLR_SCRN ************************
;                                                       *
;*******************************************************
CLR_SCRN:     LDB  AL,#CODECRT_CLR        ;TELEVIDEO 925
              CALL VIDEO_OUT
              RET
;*******************************************************


;************************ HOME ***************************
; POSITION CURSOR AT UPPER LEFT CORNER OF CRT            *
;*******************************************************
HOME:         LDB  AL,#CODECRT_HOME       ;TELEVIDEO 925
              CALL VIDEO_OUT
              RET
;*******************************************************
```

```
;******************** CMND_ERROR ************************
; SEND MESSAGE                                           *
; ENTER with:AL=CMND VALUE                               *
;*********************************************************
CMND_ERROR:   PUSH AX                       ;SAVE CMND VALUE
              LD   SI,#C_LOOKUP             ;LOOKUP ADDRESS
              CALL SND_NULL
              POP  AX
              CALL SND_BYTE
              CALL SND_NULL
              RET

C_LOOKUP:     DCB  0AH,0DH,'CMND ERROR(VALUE=',00H
              DCB  'H) ',00H
;*********************************************************



;********************** SP_LOOP ************************
; SEND STRING OF SPACES TO VIDEO                         *
; ENTER with:CL=# SPACES                                 *
;*********************************************************
SP_LOOP:      PUSH AX
              LDB  AL,#20H                  ;ASCII SP CODE
SPL:          CALL VIDEO_OUT
              DJNZ CL,SPL
              POP  AX
              RET
;*********************************************************



IF xLF_LOOP
;********************** LF_LOOP ************************
; SEND STRING OF LINE FEEDS TO VIDEO                     *
; ENTER with:CL=# LF'S                                   *
;*********************************************************
              PUBLIC  LF_LOOP

LF_LOOP:      PUSH AX
              LDB  AL,#0AH                  ;ASCII LF CODE
LFL:          CALL VIDEO_OUT
              DJNZ CL,LFL
              POP  AX
              RET
;*********************************************************
ENDIF



;********************** SND_CR_LF ************************
; SEND TO VIDEO CR AND THEN LF                           *
```

```
;*******************************************************
SND_CR_LF:      CALL SND_CR
                CALL SND_LF
                RET
;*******************************************************



;********************** SND_LF ************************
; SEND TO VIDEO LF                                     *
;*******************************************************
SND_LF:         PUSH AX
                LDB  AL,#0AH
                CALL VIDEO_OUT
                POP  AX
                RET
;*******************************************************



;********************** SND_CR ************************
; SEND TO VIDEO CR                                     *
;*******************************************************
SND_CR:         PUSH AX
                LDB  AL,#0DH
                CALL VIDEO_OUT
                POP  AX
                RET
;*******************************************************


;********************** SND_EQUAL *********************
; SEND TO VIDEO EQUAL SIGN =                           *
;*******************************************************
SND_EQUAL:      PUSH AX
                LDB  AL,#3DH
                CALL VIDEO_OUT
                POP  AX
                RET
;*******************************************************



;********************** SND_BKSP **********************
; SEND TO VIDEO BACKSPACE                              *
;*******************************************************
SND_BKSP:       PUSH AX
                LDB  AL,#08H
                CALL VIDEO_OUT
                POP  AX
                RET
;*******************************************************
```

```
;********************** SND_COMMA ************************
; SEND TO VIDEO COMMA                                     *
;*********************************************************
SND_COMMA:     PUSH AX
               LDB  AL,#2CH
               CALL VIDEO_OUT
               POP  AX
               RET
;*********************************************************



;*********************** SND_SP **************************
; SEND SPACE TO VIDEO                                     *
;*********************************************************
SND_SP:        PUSH AX
               LDB  AL,#20H
               CALL VIDEO_OUT
               POP  AX
               RET
;*********************************************************



;*********************** SND_DASH ************************
; SEND TO VIDEO DASH                                      *
;*********************************************************
SND_DASH:      PUSH AX
               LDB  AL,#2DH
               CALL VIDEO_OUT
               POP  AX
               RET
;*********************************************************



;********************** SND_SLASH ************************
; SEND TO VIDEO SLASH CHAR /                              *
;*********************************************************
SND_SLASH:     PUSH AX
               LDB  AL,#2FH
               CALL VIDEO_OUT
               POP  AX
               RET
;*********************************************************



;********************** SND_COLON ************************
```

```
; SEND TO VIDEO COLON                                          *
;*************************************************************
SND_COLON:     PUSH AX
               LDB  AL,#3AH
               CALL VIDEO_OUT
               POP  AX
               RET
;*************************************************************



;********************** SND_PLUS ***********************
; SEND TO VIDEO PLUS SIGN +                                   *
;*************************************************************
SND_PLUS:      PUSH AX
               LDB  AL,#2BH
               CALL VIDEO_OUT
               POP  AX
               RET
;*************************************************************



;********************** SND_PERIOD *********************
; SEND TO VIDEO PERIOD CHAR .                                 *
;*************************************************************
SND_PERIOD:    PUSH AX
               LDB  AL,#2EH
               CALL VIDEO_OUT
               POP  AX
               RET
;*************************************************************



;********************** SND_QMARK **********************
; SEND TO VIDEO QUESTION MARK ?                               *
;*************************************************************
SND_QMARK:     PUSH AX
               LDB  AL,#3FH
               CALL VIDEO_OUT
               POP  AX
               RET
;*************************************************************



;********************** SND_BEEP ***********************
; SEND TO VIDEO BEEP/BELL TONE                                *
;*************************************************************
SND_BEEP:      PUSH AX
               LDB  AL,#07H
               CALL VIDEO_OUT
```

```
                POP  AX
                RET
;*******************************************************



;********************* SND_STAR ************************
; SEND *                                                *
;*******************************************************
SND_STAR:       PUSH AX
                LDB  AL,#2AH
                CALL VIDEO_OUT
                POP  AX
                RET
;*******************************************************



;********************* SND_ACK *************************
; SEND ACK CODE                                         *
;*******************************************************
SND_ACK:        PUSH AX
                LDB  AL,#06H
                CALL VIDEO_OUT
                POP  AX
                RET
;*******************************************************



;********************* SND_NAK *************************
; SEND NAK CODE                                         *
;*******************************************************
SND_NAK:        PUSH AX
                LDB  AL,#15H
                CALL VIDEO_OUT
                POP  AX
                RET
;*******************************************************



;********************* CR_CONT *************************
; SEND MESSAGE CR=CONTINUE                              *
;*******************************************************
CR_CONT:        LD   SI,#MSG_CRCONT
                CALL SND_NULL
                RET

MSG_CRCONT:     DCB  0AH,0AH,0DH,'CR=CONTINUE ',00H
;*******************************************************
```

```
;********************** CLR_MEM ************************
; ENTER with:DT=PTR FOR START OF LOCATION TO BE CLR'D  *
;           :CX=# BYTES TO BE CLEARED                  *
;*******************************************************
CLR_MEM:      CLRB AL                      ;AL=00
              CALL FILL_MEM
              RET
;*******************************************************




;********************** FILL_MEM ************************
; FILL MEMORY STRING                                   *
; ENTER with:DT=PTR FOR START OF LOCATION TO BE FILLED *
;           :AL=FILL VALUE                             *
;           :CX=# BYTES TO BE FILLED                   *
;*******************************************************
FILL_MEM:     STB  AL,[DT]+
              DEC  CX
              JNE  FILL_MEM
              RET
;*******************************************************



IF xBYTESWAP_LOOP
;********************** BYTESWAP_LOOP **********************
; MEMORY STRING WHERE THE MSB & LSB FOR EACH WORD IS    *
;     EXCHANGED                                         *
; EXAMPLE:                                              *
;        RAM+00=AA  to BB                               *
;        RAM+01=BB  to AA                               *
;        RAM+02=CC  to DD                               *
;        RAM+03=DD  to CC                               *
; ENTER with:BP=WORD PTR @ LSB                          *
;           :CX=# WORDS                                 *
;*******************************************************
              PUBLIC  BYTESWAP_LOOP

BYTESWAP_LOOP: PUSH AX
              PUSH BX
              CLRB BU
NXT_BSWAP:    LD   AX,[BP]
              LDB  BL,AH
      LDB  AH,AL
      AND  AX,#0FF00H
              OR   AX,BX
              ST   AX,[BP]+
              DEC  CX
              JNE  NXT_BSWAP
      POP  BX
```

```
      POP  AX
           RET
;*******************************************************
ENDIF


IF xSTRING_SWAP
;******************* STRING_SWAP ***********************
; MEMORY STRING WHERE THE MSB LSB PRIORITY IS REVERSED *
; EXAMPLE:                                             *
;        RAM+00=AA  to DD      (AA=MSB)                *
;        RAM+01=BB  to CC                              *
;        RAM+02=CC  to BB                              *
;        RAM+03=DD  to AA                              *
; ENTER with:SI=PTR @ MSB   (RAM+00)                   *
;           :CX=# BYTES                                *
;*******************************************************
             PUBLIC  STRING_SWAP

STRING_SWAP: PUSH DT
             PUSH SI
             PUSH CX
             LD   DT,#RAM_SCRATCH
NXT_SWAP:    BMOV SI,CX
             POP  CX
             POP  SI
;REVERSE ORDER OF STRING
             PUSH SI                      ;XCHG SI,DI
             LD   SI,DT
             POP  DT
             DEC  SI                      ;PT @ OLD LSB
STRING_LOOP: LDB  AL,[SI]
             STB  AL,[DT]+
             DEC  SI
             DEC  CX
             JNE  STRING_LOOP
             POP  DT
             RET
;*******************************************************
ENDIF


IF xSTOP_START
;******************** STOP_START ***********************
; STOP/START DISPLAY WITH KEYSTROKE                     *
;*******************************************************
             PUBLIC  STOP_START

STOP_START:  CLRB AL                      ;STOP DSPLY WITH KEYSTROKE
             EI
             NOP
             DI
             CMPB AL,#00H
             BE   DSPLY_GO
```

```
                EI
                CLRB AL
WAIT_START:     CMPB AL,#00H
                BE   WAIT_START             ;AWAIT START KEYSTROKE
                DI
DSPLY_GO:       RET
;*******************************************************
ENDIF


IF xFORM_2SCOMP
;******************** FORM_2SCOMP **********************
; FORM 2'S COMP OF A DOUBLE WORD IN RAM & WRITE IT BACK*
;     TO RAM   (LSB FIRST BEFORE & AFTER)              *
; ENTER with:SI=PTR FOR ABSOLUTE #    (LSB)           *
;           :DT=RESULT PTR                            *
; EXIT  with:RESULT TABLE @ DT PTR    (LSB FIRST)     *
;*******************************************************
                PUBLIC  FORM_2SCOMP

FORM_2SCOMP:    PUSH DX
                PUSH AX
                LD   DX,2[SI]               ;MSW
                LD   AX,[SI]                ;LSW
                CALL FORM_2SCOMP1
                ST   AX,[DT]+
                ST   DX,[DT]+
                POP  AX
                POP  DX
                RET
;*******************************************************
ENDIF


IF xFORM_2SCOMP1
;******************** FORM_2SCOMP1 *********************
; FORM 2'S COMP OF A DOUBLE WORD                       *
; ENTER with:AX=LSW                                    *
;           :DX=MSW                                    *
; EXIT  with:AX:DX RESULT                              *
;*******************************************************
                PUBLIC  FORM_2SCOMP1

FORM_2SCOMP1:   NOT  DX                     ;1'S COMP
                NOT  AX                     ;1'S COMP
                ADD  AX,#0001H              ;ADD 1
                ADDC DX,#0000H              ;ADD CY
                RET
;*******************************************************
ENDIF
```

```
;********************* REVERSE_AL ***********************
; FOR MIS WIRING OF DATA BITS                           *
;    MAKE D0=D7,D1=D6,ETC                               *
; EXAMPLE: D5 GOES TO AB                                *
;*******************************************************
REVERSE_AL:   CLRB AH
              JBC  AL,0,NO_0
              ORB  AH,#80H
NO_0:         JBC  AL,1,NO_1
              ORB  AH,#40H
NO_1:         JBC  AL,2,NO_2
              ORB  AH,#20H
NO_2:         JBC  AL,3,NO_3
              ORB  AH,#10H
NO_3:         JBC  AL,4,NO_4
              ORB  AH,#08H
NO_4:         JBC  AL,5,NO_5
              ORB  AH,#04H
NO_5:         JBC  AL,6,NO_6
              ORB  AH,#02H
NO_6:         JBC  AL,7,NO_7
              ORB  AH,#01H
NO_7:         LDB  AL,AH
              RET
;*******************************************************



;************************ SHLC ************************
; SHIFT LEFT WITH CY                                    *
; EXAMPLE: AL=D5,CL=07 THEN AL=EA                       *
; ENTER with: AL=VALUE                                  *
;             CL=# SHIFTS LEFT                          *
;*******************************************************
SHLC:         CLRC
NXT_ROTATE:   ADDCB AL,AL
              DJNZ CL,NXT_ROTATE
              RET
;*******************************************************



;********************** DLY_10us ************************
;                                                       *
;*******************************************************
VALUE_10us    EQU  0002H          ;16 MHz

DLY_10us:     PUSH CX
              LD   CX,#VALUE_10us
Dx_10us:      DEC  CX
              BNE  Dx_10us
              POP  CX
              RET
;*******************************************************
```

```
;********************** DLY_25us ************************
;                                                       *
;*******************************************************
IF SPEED
     VALUE_25us     EQU  0015H          ;12 MHz
ELSE
     VALUE_25us     EQU  0030H          ;16 MHz
ENDIF


DLY_25us:      PUSH CX
               LD   CX,#VALUE_25us
Dx_25us:       DEC  CX
               BNE  Dx_25us
               POP  CX
               RET
;*******************************************************



;********************** DLY_100us ***********************
;                                                       *
;*******************************************************
IF SPEED
     VALUE_100us    EQU  0037H          ;12 MHz
ELSE
     VALUE_100us    EQU  0041H          ;16 MHz
ENDIF


DLY_100us:     PUSH CX
               LD   CX,#VALUE_100us
Dx_100us:      DEC  CX
               BNE  Dx_100us
               POP  CX
               RET
;*******************************************************



;********************** DLY_1ms *************************
;                                                       *
;*******************************************************
IF SPEED
     VALUE_1ms      EQU  02D0H          ;12 MHz
ELSE
     VALUE_1ms      EQU  0390H          ;16 MHz
ENDIF


DLY_1ms:       PUSH CX
               LD   CX,#VALUE_1ms
```

```
Dx_1ms:         DEC  CX
                BNE  Dx_1ms
                POP  CX
                RET
;*******************************************************



;********************** DLY_10ms ***********************
;                                                      *
;*******************************************************
IF SPEED
     VALUE_10ms     EQU  1C80H          ;12 MHz
ELSE
     VALUE_10ms     EQU  1A00H          ;16 MHz
ENDIF


DLY_10ms:       PUSH CX
                LD   CX,#VALUE_10ms
Dx_10ms:        DEC  CX
                BNE  Dx_10ms
                POP  CX
                RET
;*******************************************************



;********************** DLY_100ms **********************
;                                                      *
;*******************************************************
DLY_100ms:      PUSH CX                        ;16 MHz
                LDB  CL,#0AH
Dx_100ms:       CALL DLY_10ms
                DJNZ CL,Dx_100ms
                POP  CX
                RET
;*******************************************************



;********************** DLY_500ms **********************
;                                                      *
;*******************************************************
DLY_500ms:      PUSH CX
                LDB  CL,#05H
DLYx_500ms:     CALL DLY_100ms
                DJNZ CL,DLYx_500ms
                POP  CX
                RET
;*******************************************************
```

```
;********************** DLY_1sec ************************
;                                                       *
;*******************************************************
DLY_1sec:      PUSH CX
               LDB  CL,#0AH
DLYx_1sec:     CALL DLY_100ms
               DJNZ CL,DLYx_1sec
               POP  CX
               RET
;*******************************************************
```

```
;********************** EN_AD_FAST **********************
; ENABLE A/D FAST MODE                                  *
;      IOC2.4=0    158 STATES @ 16MHz=19.8 us DEFAULT) *
;      IOC2.4=1     91 STATES @ 16MHz=11.4 us          *
;                   91 STATES @ 20MHz= 9.1 us          *
;*******************************************************
EN_AD_FAST:    ORB  IOC2,#10H              ;SET BIT 4
               RET
;*******************************************************
```

```
;********************** RD_AD_10 ************************
; 10-BITS    (KB or KC)                                 *
; USES POLLING METHOD(EOC INTERRUPT IS AN OPTION)       *
; 8 POSSIBLE CHANNELS-SHARE PINS WITH PORT #0           *
; PRESCALER                                             *
;      IOC2.4=0    158 STATES @ 16MHz=19.8 us DEFAULT) *
;      IOC2.4=1     91 STATES @ 16MHz=11.4 us          *
;                   91 STATES @ 20MHz= 9.1 us          *
; ENTER with:CL=A/D CH #                                *
; EXIT  with:AX=A/D RESULT  (10 BITS)                   *
;*******************************************************
RD_AD_10:      LDB  AD_CMND,CL             ;SET CH #
               ORB  CL,#08H                ;SET START BIT
               LDB  AD_CMND,CL             ;START A/D CONVERSION
WAIT_ADSTART:  LDB  AL,AD_RESULT           ;GET STATUS BYTE
               JBC  AL,3,WAIT_ADSTART      ;TAKE 8 STATE TIMES TO SET BIT
WAIT_ADDONE:   LDB  AL,AD_RESULT           ;GET STATUS BYTE
               JBS  AL,3,WAIT_ADDONE       ;91 OR 158 STATE TIMES
               LD   AX,AD_RESULT
               SHR  AX,#6                  ;A/D LSB INTO AL LSB
               ANDB CL,#0F7H               ;CLR START BIT
               RET
;*******************************************************
```

```
;********************** RD_AD_8 **********************
; 8-BIT RESULT-ROTATE OFF LOWER 2 BITS                *
; ENTER with:CL=A/D CH #                              *
; EXIT  with:AL=A/D RESULT  (8 BITS)                  *
;*****************************************************
RD_AD_8:      CALL RD_AD_10              ;AX=RESULT
              SHR  AX,#2                 ;10 BITS TO 8 BITS
              RET
;*****************************************************
```

```
;********************** RD_AD_KC8 **********************
; RD A/D FOR KC VERSION & 8-BIT INSTEAD OF 10          *
; NOTE:DO NOT USE "ORB  AD_CMND,#18H"                  *
;                                                      *
; ENTER with:CL=A/D CH #                               *
; EXIT  with:AL=A/D RESULT  (8 BITS)                   *
;******************************************************
RD_AD_KC8:    LDB  AD_CMND,CL            ;LD CH #
              ORB  CL,#18H               ;SET BOTH 8-BITS & START
              LDB  AD_CMND,CL            ;START A/D CONVERSION
              JBC  AD_RESULT,3,$         ;TAKE 8 STATE TIMES TO SET BIT
              JBS  AD_RESULT,3,$         ;91 OR 158 STATE TIMES
;WAIT_KCSTART: LDB  AL,AD_RESULT          ;GET STATUS BYTE
;              JBC  AL,3,WAIT_KCSTART     ;TAKE 8 STATE TIMES TO SET BIT
;WAIT_KCDONE:  LDB  AL,AD_RESULT          ;GET STATUS BYTE
;              JBS  AL,3,WAIT_KCDONE      ;91 OR 158 STATE TIMES
              LD   AX,AD_RESULT
              SHR  AX,#8                 ;A/D LSB INTO AL LSB
              ANDB CL,#0E7H              ;CLR START BIT
              RET
;******************************************************
```

```
;********************** SND_AD8_10 **********************
; COMPUTE 8 BIT A/D AND SEND                            *
; RANGE: 0 TO +10 VOLTS                                 *
; ENTER AL=DATA                                         *
; SEND    X.XX                                          *
;*******************************************************
RNG_0810      EQU  0188H                 ;10/255=39.2 mV

SND_AD8_10:   LD   BX,#RNG_0805          ;RANGE=10 OR 5 VOLTS
              CALL SND_AD8
              RET
;*******************************************************
```

```
;********************* SND_AD_8  ***********************
;********************* SND_AD8_5 ***********************
; COMPUTE 8 BIT A/D AND SEND                            *
; RANGE: 0 TO +5 VOLTS                                  *
; ENTER AL=DATA                                         *
; SEND    X.XX                                          *
;*******************************************************
RNG_0805      EQU  00C4H                ;05/255=19.6 mV

SND_AD8_5:
SND_AD_8:     LD   BX,#RNG_0805         ;RANGE=10 OR 5 VOLTS
SND_AD8:      CLRB AH                   ;AH=00
              MULU BX,AX                ;DATA x RES/BIT
              LD   AX,#0064H            ;0064H=100
              DIVU BX,AX                ;DIVIDE/100
              CALL BIN_BCD              ;ENTER WITH BX=DATA
;SND DATA TO CRT
              PUSH CX
              LDB  CL,CH
              CALL BCD_SPLIT
;             CALL SEND_PAIR            ;SEND D4D3
              LDB  AL,CL                ;SND D3
              CALL VIDEO_OUT
              CALL SND_PERIOD           ;SEND .
              POP  CX
              CALL BCD_SPLIT
              CALL SEND_PAIR            ;SEND D2D1
              RET
;*******************************************************



;********************* SND_AD8_20 **********************
; COMPUTE 8 BIT A/D AND SEND                            *
; RANGE: 0 TO +20 VOLTS                                 *
; A/D REF IS +5.0 VOLTS WITH INPUT RESISTOR DIVIDER x4  *
; ENTER AL=DATA                                         *
; SEND   XX.XX                                          *
;*******************************************************
RNG_0820      EQU  030DH                ;20/256=78.1 mV

SND_AD8_20:   LD   BX,#RNG_0820         ;RANGE=20 VOLTS
SND_AD8_X:    CLRB AH                   ;AH=00
              MULU BX,AX                ;DATA x RES/BIT
              LD   AX,#0064H            ;0064H=100
              DIVU BX,AX                ;DIVIDE/100
              CALL BIN_BCD              ;ENTER WITH BX=DATA
;SND DATA TO CRT
              PUSH CX
              LDB  CL,CH
              CALL BCD_SPLIT
              CALL SEND_PAIR            ;SEND D4D3
              CALL SND_PERIOD           ;SEND .
```

```
                POP  CX
                CALL BCD_SPLIT
                CALL SEND_PAIR              ;SEND D2D1
                RET
;*******************************************************



;********************* SND_AD_10 ***********************
; COMPUTE 10 BIT A/D AND SEND                           *
; RANGE: 0 TO +5                                        *
; ENTER AX=DATA                                         *
; SEND   X.XXX                                          *
;*******************************************************
RNG_1205        EQU  1313H                  ;05/1024=4883 uV

SND_AD_10:      LD   BX,#RNG_1205      ;RANGE=5 VOLTS
                MULU BX,AX                  ;DATA x RES/BIT
                LD   AX,#03E8H              ;03E8H=1000
                DIVU BX,AX                  ;DIVIDE/1000
                CALL BIN_BCD
;SND DATA TO CRT
;               ADDB AL,#30H                ;SND D5
;               CALL VIDEO_OUT
                PUSH CX
                LDB  CL,CH
                CALL BCD_SPLIT
                LDB  AL,CH                  ;SEND D4
                CALL VIDEO_OUT
                CALL SND_PERIOD             ;SEND .
                LDB  AL,CL
                CALL VIDEO_OUT              ;SND D3
                POP  CX
                CALL BCD_SPLIT
                CALL SEND_PAIR              ;SEND D2D1
                RET
;*******************************************************



IF xSND_AD_160010
;NOT CHK'D OUT
;******************** SND_AD_160010 ********************
; COMPUTE 16 BIT A/D AND SEND                           *
; RANGE: 0 TO +10                                       *
; ENTER AX=DATA                                         *
; SEND   X.XXX                                          *
;*******************************************************
                PUBLIC  SND_AD_160010

RNG_160010      EQU  05F6H                  ;10/65535=152.6 uV

SND_AD_160010:
                LD   BX,#RNG_160010         ;RANGE=10 VOLTS
                MULU BX,AX                  ;DATA x RES/BIT
                LD   AX,#2710H              ;2710H=10000
```

```
                DIVU BX,AX                      ;DIVIDE/10000
                CALL BIN_BCD
;SND DATA TO CRT
;               ADDB AL,#30H                    ;SND D5
;               CALL VIDEO_OUT
                PUSH CX
                LDB  CL,CH
                CALL BCD_SPLIT
                LDB  AL,CH                      ;SEND D4
                CALL VIDEO_OUT
                CALL SND_PERIOD                 ;SEND .
                LDB  AL,CL
                CALL VIDEO_OUT                  ;SND D3
                POP  CX
                CALL BCD_SPLIT
                CALL SEND_PAIR                  ;SEND D2D1
                RET
;*******************************************************
ENDIF


IF xSND_AD_161010
;NOT CHK'D OUT
;********************* SND_AD_161010 ******************
; COMPUTE 16 BIT A/D AND SEND                          *
; RANGE: -10 TO +10           +10=7FFF   (2'S COMP)    *
;                               0=0000                 *
; ENTER AX=DATA                -0=FFFF                 *
; SEND   X.XXX                -10=8000                 *
;*******************************************************
                PUBLIC  SND_AD_161010

RNG_161010      EQU  06ECH                      ;10/32768=305.2 uV

SND_AD_161010:
                JBC  AH,7,SKIP_161010           ;CHK SIGN BIT
                CALL SND_DASH                   ;SND MINUS SIGN
                NOT  AX                         ;1'S COMPLEMENT
SKIP_161010:    LD   BX,#RNG_161010             ;RANGE=10 VOLTS
                MULU BX,AX                      ;DATA x RES/BIT
                LD   AX,#2710H                  ;2710H=10000
                DIVU BX,AX                      ;DIVIDE/10000
                CALL BIN_BCD
;SND DATA TO CRT
;               ADDB AL,#30H                    ;SND D5
;               CALL VIDEO_OUT
                PUSH CX
                LDB  CL,CH
                CALL BCD_SPLIT
                LDB  AL,CH                      ;SEND D4
                CALL VIDEO_OUT
                CALL SND_PERIOD                 ;SEND .
                LDB  AL,CL
                CALL VIDEO_OUT                  ;SND D3
                POP  CX
                CALL BCD_SPLIT
                CALL SEND_PAIR                  ;SEND D2D1
```

```
            RET
;*******************************************************
ENDIF
END


--
*************************************************************************
******
-- Step.src
--
*************************************************************************
******
$TITLE('STEP')
$PAGELENGTH(999)

;********************** STEPPER MTR MODULE ********************
;INTELLIGIENT MOTION SYSTEMS MODULE, IM483                      *
;***************************************************************
;STEP_MTR_MENU                                                  *
;MENU_FIXED_CW                                                  *
;MENU_FIXED_CCW                                                 *
;MENU_SPEED                                                     *
;INIT_STEP_MTR                                                  *
;RUN_STEP_IN                                                    *
;RUN_STEP_OUT                                                   *
;RUN_STEP_AHEAD                                                 *
;RUN_STEP_BACK                                                  *
;RUN_STEPPER                                                    *
;RUN_MTR_FOREVER                                                *
;STEP_CLK                                                       *
;SET_DIR_CW                                                     *
;SET_DIR_CCW                                                    *
;EN_STEPPER                                                     *
;DIS_STEPPER                                                    *
;***************************************************************
; P1.7=DC DIR---------------OUT     (I/O)                       *
; P1.6=DC BRAKE-------------OUT     (I/O)                       *
; P1.5=LIGHTS PWR ON--------OUT     (I/O or PWM1)(IOC3.2=1 PWM) *
; P1.4=PMT PWR ON-----------OUT     (I/O or PWM2()IOC3.3=1 PWM) *
; P1.3=STEP RESET*----------OUT     (I/O)                       *
; P1.2=STEP ENABLE----------OUT     (I/O)                       *
; P1.1=STEP DIR-------------OUT     (I/O)                       *
; P1.0=STEP CLK-------------OUT     (I/O)                       *
;***************************************************************
            PUBLIC
STEP_MTR_MENU,INIT_STEP_MTR,RUN_STEP_IN,RUN_STEP_OUT
            PUBLIC  RUN_STEP_AHEAD,RUN_STEP_BACK

            EXTRN   WAIT_KYBD,SND_TABLE,CMND_ERROR,CLR_SCRN,DLY_10us
            EXTRN   DLY_1ms,ENTR_SHOW_WORD,SND_NULL

$INCLUDE(8086_REGS.INC)
$INCLUDE(SFR.INC)
            EXTRN   RAM_P1,DEFAULT_STEPS,RAM_STEP1CM,RAM_STEPCNTS
            EXTRN   RAM_STEPSPD,DEFAULT_SPEED,RAM_STEPIN,RAM_STEPOUT
            EXTRN   DEFAULT_STEPINx,DEFAULT_STEPOUTx,DEFAULT_STEP1CMx
```

```
                EXTRN   DEFAULT_STEPINy,DEFAULT_STEPOUTy,DEFAULT_STEP1CMy

CSEG
;*****************************************************************



;************************ STEP_MTR_MENU ************************
;                                                               *
;*****************************************************************
STEP_MTR_MENU: CALL MENU_PREP
               CALL EN_STEPPER
MENU_1:        CALL WAIT_KYBD              ;AL=RESULT
               CMPB AL,#01H                ;CNTRL A=RUN MTR CW FOREVER
               BE   M_CW
               CMPB AL,#02H                ;CNTRL B=RUN MTR CCW FOREVER
               BE   M_CCW
               CMPB AL,#04H                ;CNTRL D=RUN MTR CW FIXED
               BE   MENU_FIXED_CW
               CMPB AL,#05H                ;CNTRL E=RUN MTR CCW FIXED
               BE   MENU_FIXED_CCW
               CMPB AL,#13H                ;CNTRL S=SETUP
               BE   MENU_SETUP
               CMPB AL,#0DH                ;CR=RETURN
               BE   M_RET

;ERROR MESSAGE & TRY AGAIN
               CALL CMND_ERROR
               BR   MENU_1

M_RET:         RET

M_CW:          CALL CLR_SCRN
               LD   SI,#MSG_RUNNING
               CALL SND_NULL
               CALL SET_DIR_CW
               BR   RUN_MTR_FOREVER

M_CCW:         CALL CLR_SCRN
               LD   SI,#MSG_RUNNING
               CALL SND_NULL
               CALL SET_DIR_CCW
               BR   RUN_MTR_FOREVER

MSG_RUNNING:  DCB  'RUNNING STEPPER FOREVER',00H
;*****************************************************************



;************************ MENU_PREP ****************************
;                                                               *
;*****************************************************************
MENU_SENDS    EQU  06H                  ;INCLUDES TITLE
MENU_INDENTS  EQU  10H
```

```
MENU_PREP:      CALL CLR_SCRN
                LD   SI,#MENU_TABLE
                LDB  CL,#MENU_INDENTS       ;# OF INDENTS
                LDB  CH,#MENU_SENDS         ;# OF SENDS
                CALL SND_TABLE
                RET

;LOOKUP FROM EPROM
MENU_TABLE:     DCW  MENU_TITLE             ;POINTERS
                DCW  CW
                DCW  CCW
                DCW  FCW
                DCW  FCCW
                DCW  SETUP

MENU_TITLE:     DCB  'STEPPER MTR MENU',0AH,0DH,00H
CW:             DCB  'CNTRL A=RUN MTR CW FOREVER',0AH,0AH,0DH,00H
CCW:            DCB  'CNTRL B=RUN MTR CCW FOREVER',0AH,0AH,0DH,00H
FCW:            DCB  'CNTRL D=RUN MTR CW FIXED DISTANCE',0AH,0AH,0DH,00H
FCCW:           DCB  'CNTRL E=RUN MTR CCW FIXED DISTANCE',0AH,0AH,0DH,00H
SETUP:          DCB  'CNTRL S=SETUP',0AH,0AH,0DH,00H
;*************************************************************



;************************ MENU_FIXED_CW ***********************
;                                                            *
;*************************************************************
MENU_FIXED_CW: CALL CLR_SCRN
                CALL SET_DIR_CW
                LD   SI,#MSG_FIXED
                LD   BP,#RAM_STEPCNTS
                CALL ENTR_SHOW_WORD
                CALL RUN_STEPPER
                BR   STEP_MTR_MENU

MSG_FIXED:      DCB  'ENTER STEP DISTANCE (0000-FFFF)=',00H
;*************************************************************



;************************ MENU_FIXED_CCW **********************
;                                                            *
;*************************************************************
MENU_FIXED_CCW: CALL CLR_SCRN
                CALL SET_DIR_CCW
                LD   SI,#MSG_FIXED
                LD   BP,#RAM_STEPCNTS
                CALL ENTR_SHOW_WORD
                LD   CX,RAM_STEPCNTS
                CALL RUN_STEPPER
                BR   STEP_MTR_MENU
;*************************************************************
```

```
;************************* MENU_SETUP ***************************
;                                                               *
;***************************************************************
MENU_SETUP:    CALL CLR_SCRN
               LD   SI,#MSG_SPEED
               LD   BP,#RAM_STEPSPD        ;SPEED
               CALL ENTR_SHOW_WORD
               LD   BP,#RAM_STEP1CM        ;1 CM
               CALL ENTR_SHOW_WORD
               LD   BP,#RAM_STEPIN         ;IN
               CALL ENTR_SHOW_WORD
               LD   BP,#RAM_STEPOUT        ;OUT
               CALL ENTR_SHOW_WORD
               BR   STEP_MTR_MENU

MSG_SPEED:     DCB  'ENTER SPEED RATIO=',00H
               DCB  0AH,0DH,'ENTER 1 CM INCREMENT CNTS (0000-FFFF)=',00H
               DCB  0AH,0DH,'ENTER LOAD ADVANCE IN CNTS (0000-FFFF)=',00H
               DCB  0AH,0DH,'ENTER UNLOAD ADVANCE OUT CNTS (0000-
FFFF)=',00H
;***************************************************************



;************************* INIT_STEP_MTR ************************
;                                                               *
;***************************************************************
INIT_STEP_MTR:
               LD   RAM_STEPIN,#DEFAULT_STEPINx
               LD   RAM_STEPIN+2,#DEFAULT_STEPINy
               LD   RAM_STEPOUT,#DEFAULT_STEPOUTx
               LD   RAM_STEPOUT+2,#DEFAULT_STEPOUTy
               LD   RAM_STEP1CM,#DEFAULT_STEP1CMx
               LD   RAM_STEP1CM+2,#DEFAULT_STEP1CMy
               LD   RAM_STEPSPD,#DEFAULT_SPEED
               RET
;***************************************************************



;************************* RUN_STEP_IN **************************
;DURING LOAD                                                    *
;***************************************************************
RUN_STEP_IN:   PUSH CX
               CALL SET_DIR_CCW
               LD   CX,RAM_STEPIN
               LD   DX,RAM_STEPIN+2
               CALL RUN_STEPPER
               POP  CX
               RET
;***************************************************************
```

```
;************************ RUN_STEP_OUT ************************
;DURING UNLOAD                                                  *
;***************************************************************
RUN_STEP_OUT: PUSH CX
              CALL SET_DIR_CW
              LD   CX,RAM_STEPOUT
              LD   DX,RAM_STEPOUT+2
              CALL RUN_STEPPER
              POP  CX
              RET
;***************************************************************



;************************ RUN_STEP_AHEAD **********************
;RUN 1 cm CW                                                    *
;***************************************************************
RUN_STEP_AHEAD:
              PUSH CX
              CALL SET_DIR_CCW
              LD   CX,RAM_STEP1CM
              LD   DX,RAM_STEP1CM+2
              CALL RUN_STEPPER
              POP  CX
              RET
;***************************************************************



;************************ RUN_STEP_BACK ***********************
;RUN 1 cm CCW                                                   *
;***************************************************************
RUN_STEP_BACK:
              PUSH CX
              CALL SET_DIR_CW
              LD   CX,RAM_STEP1CM
              LD   DX,RAM_STEP1CM+2
              CALL RUN_STEPPER
              POP  CX
              RET
;***************************************************************



;************************ RUN_STEPPER *************************
;PULSE: LO-HI-LO                                                *
;ENTER with: DX=# STEPS-MSW                                     *
;          : CX=# STEPS-LSW                                     *
;***************************************************************
RUN_STEPPER:  INC  DX                      ;ACCOUNT FOR DJNZ DX CONCEPT
RUN_STEPPERx: CALL STEP_CLK
              DJNZW CX,RUN_STEPPERx
```

```
                DJNZW DX,RUN_STEPPERx
                RET
;*****************************************************************



;************************* RUN_MTR_FOREVER *********************
;EXIT WITH CNTRL_C                                               *
;*****************************************************************
RUN_MTR_FOREVER:
                CALL STEP_CLK
                BR   RUN_MTR_FOREVER
;*****************************************************************



;************************* SET_DIR_CW **************************
;P1.1                                                            *
;*****************************************************************
SET_DIR_CW:     ORB  RAM_P1,#02H
                STB  RAM_P1,P1
                RET
;*****************************************************************



;************************* SET_DIR_CCW *************************
;P1.1                                                            *
;*****************************************************************
SET_DIR_CCW:    ANDB RAM_P1,#0FDH
                STB  RAM_P1,P1
                RET
;*****************************************************************



;************************** EN_STEPPER *************************
;P1.2                                                            *
;*****************************************************************
EN_STEPPER:     ORB  RAM_P1,#04H
                STB  RAM_P1,P1
                RET
;*****************************************************************



;************************** DIS_STEPPER ************************
;P1.2                                                            *
;*****************************************************************
DIS_STEPPER:    ANDB RAM_P1,#0FBH
                STB  RAM_P1,P1
                RET
```

```
;*************************************************************


;*************************** STEP_CLK ************************
;PULSE: LO-HI-LO                                              .*
;*************************************************************
STEP_CLK:     PUSH CX
              ORB  RAM_P1,#01H          ;PULSE HI
              STB  RAM_P1,P1
              CALL DLY_10us
              ANDB RAM_P1,#0FEH         ;PULSE LO
              STB  RAM_P1,P1
              LD   CX,RAM_STEPSPD
DLY_CLK:      CALL DLY_10us             ;REP RATE
              DJNZW CX,DLY_CLK
              POP  CX
              RET
;*************************************************************
END


--
*************************************************************************
******
-- Start.src
--
*************************************************************************
******
$TITLE('START')
$PAGELENGTH(75)

;*************************** START ***************************
; AUTHOR: JR SKORPIK    BATTELLE NW-RICHLAND,WA                *
;                        07/99                                 *
;*************************************************************
; AT POWERUP & HARD RESETS,CODE EXECUTION WILL START HERE      *
;     -87C196 INTERNAL OTP @ 2080H                             *
;                                                              *
; NOTE: EMULATOR MUST BE MAPPED FOR USER/TARGET RAM            *
;       --- MAP 8000H LENGTH 32K USER                          *
;*************************************************************
; MAIN.ICE IS A LINKED EXECUTABLE THAT WILL FUNCTION FOR       *
;    EITHER THE TAG OR READER.                                 *
; uP CHECKS INPUT LINE ON STARTUP TO DETERMINE WHICH PROGRAM   *
;    TO RUN                                                    *
;*************************************************************
;                        --SUBROUTINES--                       *
; START         (BEGIN ALL EXECUTION HERE)                     *
;  INIT_IO      (ESTABLISH I/O LINES & SET OUTPUT VALUES)       *
;  INIT_HSO     (ESTABLISH 4 HIGH SPEED OUTPUTS & SET VALUES)   *
;  INIT_MEM     (INITIALIZE INTERNAL RAM)                       *
;  INIT_T1      (SETUP INTERNAL TIMER1)                         *
;  INIT_T2      (SETUP INTERNAL TIMER2)                         *
;  EN_FAST_T2   (ENABLE FAST COUNTING MODE)                     *
;  SET_EXTINT   (DIRECT I/O PIN SOURCE FOR INTERRUPT EXT_INT)   *
```

```
;  DIS_CLKOUT   (CLKOUT_DIS for KC & KD VERSIONS)                *
;  DIS_PWM0     (USE P2.5 AS GENERAL PURPOSE OUTPUT)             *
;  SET_RUNFLG                                                    *
;  CLR_RUNFLG                                                    *
;  OUT_P5                                                        *
;  IN_P5                                                         *
;  SET_P5_OUTPUT                                                 *
;  SET_P5_INPUT                                                  *
;*************************************************************
; RAM_RUNFLG                                                     *
;     BIT #7=1,SIO TIMEOUT                                      *
;     BIT #6=                                                   *
;     BIT #5=                                                   *
;     BIT #4=                                                   *
;     BIT #3=1,UNLOADING                                        *
;     BIT #2=1,SCANNING                                         *
;     BIT #1=1,LOADING                                          *
;     BIT #0=1,RUN MODE                                         *
;                                                               *
; RAM_SIOFLG                                                    *
;     BIT #7=                                                   *
;     BIT #6=                                                   *
;     BIT #5=                                                   *
;     BIT #4=                                                   *
;     BIT #3=                                                   *
;     BIT #2=                                                   *
;     BIT #1=                                                   *
;     BIT #0=1 SOFT0, 0=HARDWARE UART                           *
;*************************************************************
               PUBLIC  START,SET_RUNFLG,CLR_RUNFLG,INIT_IO,RESTART
               PUBLIC  INIT_HSO
;              PUBLIC  OUT_P5,IN_P5,SET_P5_INPUT,SET_P5_OUTPUT

               EXTRN
INIT_SIO,MENU,CLR_MEM,EN_AD_FAST,SET_SOFTBAUD,SND_NULL
               EXTRN
CLR_HSO0,CLR_HSO1,CLR_HSO2,CLR_HSO3,INIT_HSI,CLR_SCRN
               EXTRN   INIT_STEP_MTR,RUN,DLY_1sec,SCAN_FILM,OUT_HSO

$INCLUDE(8086_REGS.INC)
$INCLUDE(SFR.INC)
               EXTRN   RAM_P1,RAM_P2,RAM_P3,RAM_P4,HWin0,STACK_VALUE
               EXTRN   HWin15,HWin1,RAM_RUNFLG,SIZE_LOWER,RAM_HSIO
               EXTRN   SIZE_UPPER,RAM_NoSCANSTEPS,No_SCAN_STEPS,RAM_RDDLY
               EXTRN   RAM_CNTTIME,CNT_TIME,RD_DLY
               EXTRN   WSI_P5_DIR,WSI_P5_DATA,WSI_P5_PIN

CSEG
;*************************************************************




;************************** START ****************************
; CODE EXECUTION ALWAYS STARTS HERE                            *
;  NOTE--ENSURE CCR=ADV INSTEAD OF ALE (FILE CCR.SRC)          *
```

```
;*************************************************************
START:          LD    SP,#STACK_VALUE       ;INIT STACK PTR
;For emulator usage
                PUSHA                       ;CLR ALL INTERRUPTS-EMULATOR
                LDB   IPEND,#00H            ;CLR IPEND
                LDB   IPEND1,#00H           ;CLR IPEND1
                LD    SP,#STACK_VALUE       ;REINIT STACK PTR

;Clr internal RAM
                CALL INIT_MEM               ;CLR INTERNAL RAM

;Set I/O Lines-must be after INIT_MEM
                CALL DIS_PWM0               ;USE P2.5 AS GENERAL PURPOSE
OUTPUT
                CALL INIT_HSO               ;CLR HIGH SPEED OUTPUTS(HSO.0-
3)
                CALL INIT_IO                ;RESET RAM_SHADOWS(P1-P4)

;Finish Initializing
                CALL INIT_SIO               ;HARDWARE UART=9600 BAUD
;               CALL SET_SOFTBAUD           ;SOFTWARE UART=9600 BAUD
;               CALL INIT_T1                ;INIT TIMER1-INTERNAL 1MHz
;               CALL INIT_T2                ;INIT TIMER2-CNT UP WITH EXT
CLK
                                            ;UP/DOWN MODE ENABLED
                                            ;CAN ALSO ENABLES FAST COUNTING
(4 MHz)
;               CALL EN_FAST_T2             ;ENABLE FAST COUNTING MODE-T2
(4 MHz)
                CALL SET_EXTINT             ;CONFIGURE EXT_INT PINS
                                            ;INT & INT1 HAVE MIN WIDTHS
;               CALL DIS_CLKOUT             ;DISABLE CLKOUT SIGNAL
;               CALL EN_AD_FAST             ;FAST CONVERSION MODE-91 STATES
;               CALL INIT_HSI               ;PULSE WIDTHS ON HSI (MUST LINK
WIDTHS.OBJ)

;USER SPECIFICS
                CALL INIT_STEP_MTR
RESTART:
                CALL CLR_SCRN
                LD    SI,#MSG_USER
                CALL SND_NULL
                CALL DLY_1sec               ;FOR USER TO HIT CNTRL C TO
ENTER DEBUG MODE
                CALL DLY_1sec
                EI                          ;GLOBAL INTERRUPT ENABLE
;               BR    MENU                  ;CHKOUT ONLY
                CALL CLR_SCRN
                LD    SI,#MSG_HOST
                CALL SND_NULL
                CALL SET_RUNFLG             ;OUTPUT TO HOST
                BR    RUN

MSG_USER:       DCB   'HIT CNTRL_C WITHIN 2 SECS TO GET DEBUG MENU',00H
MSG_HOST:       DCB   'RUN w/HOST-WAIT FP SWITCHES',00H
;*************************************************************
```

```
;************************** INIT_IO ****************************
; P0=INPUT ONLY                                                 *
; P1=QUASI-BIDIRECTIONAL        (IF USED AS INPUT WRT 1'S)      *
; P2=QUASI,INPUT OR OUTPUT                                      *
; P3=OPEN DRAIN BIDIRECTIONAL              "                    *
; P4=OPEN DRAIN BIDIRECTIONAL              "                    *
; OUTPUTS ARE STRONG LOW & WEAK HIGH                            *
; RESET STATE: P1,P3,P4=FF    P2=C1                             *
;***************************************************************
INIT_IO:      LDB  RAM_P1,#08H          ;SET PORT #1
              STB  RAM_P1,P1            ;----
              LDB  RAM_P2,#8FH          ;SET PORT #2
              STB  RAM_P2,P2            ;----
              LDB  RAM_P3,#08H          ;
              STB  RAM_P3,P3            ;----
              LDB  RAM_P4,#00H          ;
              STB  RAM_P4,P4            ;----
              RET
;***************************************************************
```

```
;************************** INIT_HSO ***************************
; CONFIGURE & SET HIGH SPEED OUTPUTS                            *
; HS0-3 ALWAYS OUTPUTS,HS4 & 5 CAN BE INPUTS OR OUTPUTS         *
; RESET STATE: HSO.0-.3=0                                       *
;***************************************************************
INIT_HSO:
;Configure HSO outputs-replaces HSI.2 & HSI.3
;             CALL EN_HSO4               ;ENABLE HSO4 AS OUTPUT
;             CALL EN_HSO5               ;ENABLE HSO5 AS OUTPUT
;Set HSO output states
;             CALL CLR_HSO0              ;
;             CALL CLR_HSO1              ;
;             CALL CLR_HSO2              ;
;             CALL CLR_HSO3              ;
;             CLRB RAM_HSIO

              LDB  RAM_HSIO,#0FH
              LDB  AL,RAM_HSIO          ;LEDS OFF
              CALL OUT_HSO
              RET
;***************************************************************
```

```
;************************** INIT_MEM ***************************
; CLR INTERNAL RAM (28H-FCH)      (PRESERVE STACK)              *
;                  (100H-1FCH)    (PRESERVE STACK)              *
```

```
;*************************************************************
INIT_MEM:       LD    CX,#SIZE_LOWER        ;LOWER RAM
                LD    DT,#0028H
                CALL  CLR_MEM
                LD    CX,#SIZE_UPPER        ;UPPER RAM
                LD    DT,#0100H
                CALL  CLR_MEM

                LD    RAM_NoSCANSTEPS,#No_SCAN_STEPS
                LDB   RAM_CNTTIME,#CNT_TIME
                LDB   RAM_RDDLY,#RD_DLY
                RET
;*************************************************************



;************************** INIT_T1 ****************************
; SETUP FREE-RUNNING TIMER #1-16 BITS                          *
; INPUT IS 1.33 usec CLOCK AT 12 MHz (EVERY 8 STATE TIMES)     *
;          1.00 usec CLOCK AT 16 MHz                           *
;           800 nsec CLOCK AT 20 MHz                           *
; UP CNTR                                                      *
; GENERATES INT AT 2000H                                       *
;*************************************************************
INIT_T1:        LDB   WSR,#HWin15           ;SET WINDOW=15
                LDB   AL,IOC1               ;RD IOC1
                ORB   AL,#04H               ;ENABLE T1 INT
                LDB   WSR,#HWin0            ;SET WINDOW=00
                STB   AL,IOC1               ;REWRITE IOC1
                RET
;*************************************************************



;************************** INIT_T2 ****************************
; SETUP TIMER #2-16 BITS                                       *
; INPUTS (EXTERNAL OR EXTERNAL, IOC3.0)                        *
;  -EXTERNAL (HSI.1 OR P2.3, IOC0.7)                           *
;     NORMAL MODE-EVERY 8 STATE TIMES,IOC2.0=0,RESET STATE     *
;       CNT RATE=376 KHz @ 12 MHz                              *
;                =500 KHz @ 16 MHz                             *
;                =625 KHz @ 20 MHz                             *
;     FAST MODE-EVERY 1 STATE TIME,IOC2.0=1                    *
;       CNT RATE=3 MHz @ 12 MHz                                *
;                =4 MHz @ 16 MHz                               *
;                =5 MHz @ 20 MHz                               *
;  -INTERNAL (NORMAL MODE OR FAST MODE AVAILABLE)              *
;      INPUT IS 1.33 usec CLOCK AT 12 MHz (EVERY 8 STATE TIMES)*
;                1.00 usec CLOCK AT 16 MHz                     *
;                 800 nsec CLOCK AT 20 MHz                     *
; MODES  (UP ONLY OR UP/DOWN, IOC2.1=0 for UP ONLY)            *
;                             IOC2.1=1 for UP/DOWN (UP=P2.6=0) *
;                             IOC2.1=0 upon RESET              *
```

```
; T2 CNTS BOTH POSITIVE & NEGATIVE SIMULTANEOUSLY                *
; CAN GEN INT ON FFFF-0000 OR 7FFF-8000(8000/7FFF)               *
; T2 VALUE CAN BE CAPTURED IN T2CAPture REG BY +EDGE ON P2.7     *
;                                                                *
; NOTE: IF "FAST MODE" USED DO NOT RESET TIMER2                  *
;                                                                *
; APPLICATION MODE IS:ENABLE INT,INPUT=P2.3,UP/DOWN MODE         *
;                     :NORMAL COUNTING MODE                      *
;*****************************************************************
INIT_T2:      LDB   WSR,#HWin15          ;SET WINDOW=15
              LDB   AL,IOC0              ;RD AL=IOC0
              LDB   AH,IOC1              ;RD AH=IOC1
              LDB   BL,IOC2              ;RD BL=IOC2
              ORB   AL,#00H              ;SET INPUT
SOURCE(HSI.1=80,P2.3=00)
              ORB   AH,#08H              ;ENABLE T2 INT
              ORB   BL,#02H              ;ENABLE UP/DOWN CNTR
                                         ;WIRE P2.6=1 ALSO FOR DOWN MODE
              ORB   BL,#01H              ;SET FAST COUNTING MODE
              LDB   WSR,#HWin0           ;SET WINDOW=00
              STB   AL,IOC0              ;REWRITE IOC0
              STB   AH,IOC1              ;REWRITE IOC1
              STB   BL,IOC2              ;REWRITE IOC2
;CLR TIMER2
              CLR   TIMER2               ;T2=0000
              RET
;*****************************************************************
```

```
;************************** EN_FAST_T2 ***************************
; ENABLE FAST COUNTING MODE                                       *
;     FAST MODE-EVERY 1 STATE TIME,IOC2.0=1                       *
;       CNT RATE=3 MHz @ 12 MHz                                   *
;               =4 MHz @ 16 MHz                                   *
;               =5 MHz @ 16 MHz                                   *
; T2 CNTS BOTH POSITIVE & NEGATIVE SIMULTANEOUSLY                 *
;                                                                 *
; NOTE: IF "FAST MODE" USED DO NOT RESET TIMER2 ???????          *
;*****************************************************************
EN_FAST_T2:   LDB   WSR,#HWin15          ;SET WINDOW=15
              LDB   BL,IOC2              ;RD BL=IOC2
              ORB   BL,#01H              ;SET FAST COUNTING MODE
              LDB   WSR,#HWin0           ;SET WINDOW=00
              STB   BL,IOC2              ;REWRITE IOC2
              RET
;*****************************************************************
```

```
;************************ SET_EXTINT *****************************
; EXT_INT=P0.7 OR P2.2          (VECTOR 200EH)                    *
; EXT_INT1=P2.2  ONLY           (VECTOR 203AH)                    *
```

```
;*************************************************************
SET_EXTINT:   LDB  WSR,#HWin15          ;SELECT WINDOW=15
              LDB  AL,IOC1              ;RD IOC1
              ORB  AL,#02H              ;SELECT P0.7 FOR EXTINT
              LDB  WSR,#HWin0           ;SELECT WINDOW=0
              STB  AL,IOC1              ;REWRITE IOC1
              RET
;*************************************************************



;************************ DIS_CLKOUT *************************
; DISABLE CLKOUT SIGNAL TO REDUCE SYSTEM NOISE                *
; 1=DISABLE  0=ENABLE          (RESET=0)                      *
;*************************************************************
DIS_CLKOUT:   LDB  WSR,#HWin1           ;SELECT WINDOW=1
              ORB  IOC3,#00000010B      ;SET BIT #1
              LDB  WSR,#HWin0           ;SELECT WINDOW=0
              RET
;*************************************************************



;************************ DIS_PWM0 ***************************
; SELECT P2.5 AS GENERAL PURPOSE OUTPUT PIN (PIN #39)         *
;    -IOC1.0=0 (GENERAL PURPOSE OUTPUT)      (RESET=1)        *
;*************************************************************
DIS_PWM0:     LDB  WSR,#HWin15          ;Set HWinDOW=15
              LDB  AL,IOC1
              LDB  WSR,#HWin0           ;Set HWinDOW=0
              ANDB AL,#0FEH             ;Set P2.5 AS GP OUTPUT
              STB  AL,IOC1              ;REWRITE IOC1
              RET
;*************************************************************



;************************ SET_RUNFLG *************************
;                                                             *
;*************************************************************
SET_RUNFLG:   LDB  RAM_RUNFLG,#01H      ;FLG=01
              RET
;*************************************************************



;************************ CLR_RUNFLG *************************
;                                                             *
;*************************************************************
CLR_RUNFLG:   LDB  RAM_RUNFLG,#00H      ;FLG=00
```

```
              RET
;*****************************************************************



;*************************** OUT_P5 ****************************
; WSI I/O PORT P5                                                *
; ENTER with: AL=DATA                                            *
;*****************************************************************
OUT_P5:       CALL SET_P5_OUTPUT
              STB  AL,WSI_P5_DATA
              RET
;*****************************************************************



;*************************** IN_P5 *****************************
; WSI I/O PORT P5                                                *
; EXIT  with: AL=DATA                                            *
;*****************************************************************
IN_P5:        LDB  AL,WSI_P5_PIN
              RET
;*****************************************************************



;************************** SET_P5_OUTPUT **********************
; WSI I/O PORT P5                                                *
; WRITE 1'S FOR EACH PIN=OUTPUT                                  *
;*****************************************************************
SET_P5_OUTPUT: PUSH AX
              LDB AL,#0FFH
              STB  AL,WSI_P5_DIR        ;I/O ADDRESS
              POP  AX
              RET
;*****************************************************************



;*************************** SET_P5_INPUT **********************
; WSI I/O PORT P5                                                *
; WRITE 0'S FOR EACH PIN=INPUT                                   *
;*****************************************************************
SET_P5_INPUT: CLRB AL
              STB  AL,WSI_P5_DIR        ;I/O ADDRESS
              RET
;*****************************************************************
END


--
*****************************************************************************
******
```

```
-- Sio.src
--
*************************************************************************
******
$TITLE('SERIAL PORT UTILITIES')
$PAGELENGTH(75)

;*************************** SIO ******************************
; SERIAL PORT UTILITIES-USED FOR DEBUG                          *
;     9600 BAUD                                                 *
;     8 BITS, NO PARITY                                         *
;************************************************************
;                     --HARDWARE SIO--                          *
; INIT_SIO          (9600 BAUD)                                 *
; INIT_SIO_9600     (9600 BAUD)                                 *
; INIT_SIO_19K      (19.2 KBAUD)                                *
; INIT_SIO_38K      (38.4 KBAUD)                                *
; INIT_SIO_56K      (56   KBAUD)                                *
; EN_SIO                                                        *
; DIS_SIO                                                       *
; GET_KYBD          (RESULT=AL, FOR DEBUG)                      *
; WAIT_KYBDTOUT                                                 *
; VIDEO_OUT         (FROM AL, FOR DEBUG)                        *
;                                                               *
;                     --SOFTWARE SIO--                          *
; SOFT0_IN          (RESULT=AL)       (P1.1)                    *
; SOFT0_OUT         (FROM AL)         (P1.0)                    *
;                                                               *
; SOFT1_OUT         (FROM AL)         (P1.7)                    *
; SOFT2_OUT         (FROM AL)         (HSO.0)                   *
;                                                               *
; DLY_1200                                                      *
; DLY_9600                                                      *
; SET_SOFTBAUD      (SOFT0=9600, SOFT1=9600, SOFT2=SOFT0)       *
;                                                               *
; CHKOUT_SIO        (SOFT TRANSMITTER)                          *
;                                                               *
; SIOOUT_CHKOUT                                                 *
; SIOIN_CHKOUT                                                  *
;                                                               *
; SET_SOFT_FLG      SOFT0 UART FLG      RAM_SIOFLG, BIT 0=1     *
; SET_HARD_FLG      HARDWARE UART FLG                     0     *
;************************************************************
; RAM_SIOFLG                                                    *
;    BIT #7=                                                    *
;    BIT #6=                                                    *
;    BIT #5=                                                    *
;    BIT #4=                                                    *
;    BIT #3=                                                    *
;    BIT #2=                                                    *
;    BIT #1=                                                    *
;    BIT #0=1 SOFT0, 0=HARDWARE UART                            *
;************************************************************
              PUBLIC  SOFT0_OUT,SOFT0_IN,INIT_SIO,SOFT1_OUT,SOFT2_OUT
              PUBLIC  SET_SOFTBAUD,GET_KYBD,VIDEO_OUT,CHKOUT_SIO
              PUBLIC  WAIT_KYBDTOUT,DLY_9600,SIO_MENU,SIOOUT_CHKOUT
              PUBLIC  EN_SIO,DIS_SIO,SET_SOFT_FLG,SET_HARD_FLG
```

```
                PUBLIC  RCV_DONE,RCV_LOOP,RD_RCVBIT

                EXTRN   CLR_SCRN,SET_HSO0,CLR_HSO0,MENU,RD_HOST,SND_NULL
                EXTRN   SND_TABLE,WAIT_KYBD,CMND_ERROR,SND_BYTE,DLY_1ms
                EXTRN   DLY_100us,DLY_100ms

$INCLUDE(8086_REGS.INC)
$INCLUDE(SFR.INC)
                EXTRN   BAUD1_9600,BAUD0_9600,RAM_BAUD,SOFT_9600
                EXTRN
STACK_VALUE,HWin0,HWin15,RAM_P1,RAM_BAUD1,RAM_RUNFLG
                EXTRN   RAM_SIOFLG
                EXTRN   BAUD0_19K,BAUD1_19K,BAUD0_38K,BAUD1_38K
                EXTRN   BAUD0_56K,BAUD1_56K,BAUD0_4800,BAUD1_4800

;FOLLOWING FOR "CONDITIONAL ASSEMBLY"---USES IF & ENDIF STATEMENTS
YES             EQU     0FFH
NO              EQU     00H
;               KB192   EQU YES                 ;YES=FF, SEE VIDEO.SRC
                KB192   EQU NO                  ;NO=00,  SEE VIDEO.SRC

xINIT_SIO_9600      EQU  YES
xINIT_SIO_19K       EQU  NO
xINIT_SIO_38K       EQU  NO
xINIT_SIO_56K       EQU  NO

CSEG
;*****************************************************************



;************************** INIT_SIO ****************************
; INITIALIZE SERIAL PORT                                        *
;    -USES INTERRUPT VECTOR                                     *
;*****************************************************************
INIT_SIO:       LDB  WSR,#HWin15            ;Set WINDOW=15
                LDB  AL,IOC1
                LDB  WSR,#HWin0             ;Set WINDOW=00
                ORB  AL,#00100000B          ;Set P2.0 to TXD
                STB  AL,IOC1                ;REWRITE IOC1
                LDB  BAUD_REG,#BAUD0_9600
                LDB  BAUD_REG,#BAUD1_9600
                LDB  SPCON,#01001001B       ;Enable receiver,Mode 1
                ORB  IMASK1,#00000010B      ;Enable receiver interrupt
                LDB  AL,#00H
                STB  AL,SBUF                ;Clear serial Port
                RET
;*****************************************************************



IF xINIT_SIO_9600
;************************* INIT_SIO_9600 ************************
; INITIALIZE SERIAL PORT @ 9600 KBAUD                           *
```

```
;     -USES INTERRUPT VECTOR                                          *
;*****************************************************************
              PUBLIC  INIT_SIO_9600

INIT_SIO_9600:
;             CALL INIT_SIO
              LDB  BAUD_REG,#BAUD0_9600
              LDB  BAUD_REG,#BAUD1_9600
              RET
;*****************************************************************
ENDIF


IF xINIT_SIO_19K
;************************* INIT_SIO_19K ************************
; INITIALIZE SERIAL PORT @ 19.2 KBAUD                             *
;     -USES INTERRUPT VECTOR                                      *
;*****************************************************************
              PUBLIC  INIT_SIO_19K

INIT_SIO_19K:
;             CALL INIT_SIO
              LDB  BAUD_REG,#BAUD0_19K
              LDB  BAUD_REG,#BAUD1_19K
              RET
;*****************************************************************
ENDIF


IF xINIT_SIO_38K
;************************* INIT_SIO_38K ************************
; INITIALIZE SERIAL PORT @ 38.4 KBAUD                             *
;     -USES INTERRUPT VECTOR                                      *
;*****************************************************************
              PUBLIC  INIT_SIO_38K

INIT_SIO_38K:
;             CALL INIT_SIO
              LDB  BAUD_REG,#BAUD0_38K
              LDB  BAUD_REG,#BAUD1_38K
              RET
;*****************************************************************
ENDIF


IF xINIT_SIO_56K
;************************* INIT_SIO_56K ************************
; INITIALIZE SERIAL PORT @ 56.4 KBAUD                             *
;     -USES INTERRUPT VECTOR                                      *
;*****************************************************************
              PUBLIC INIT_SIO_56K

INIT_SIO_56K:
;             CALL INIT_SIO
```

```
                LDB   BAUD_REG,#BAUD0_56K
                LDB   BAUD_REG,#BAUD1_56K
                RET
;***************************************************************
ENDIF



;*************************** EN_SIO ****************************
; ENABLE SIO INTERRUPT                                         *
;***************************************************************
EN_SIO:         ANDB IPEND1,#0FDH           ;CLR PENDING FIRST
                ORB  IMASK1,#02H
                RET
;***************************************************************



;*************************** DIS_SIO ***************************
; DISABLE SIO INTERRUPT                                        *
;***************************************************************
DIS_SIO:        ANDB IMASK1,#0FDH           ;CLR BIT #1
                RET
;***************************************************************



;*************************** GET_KYBD **************************
; READ VIDEO CHAR IN                                           *
;    -ECHOS CHAR BACK                                          *
;    -CHKS FOR CHAR "CNTRL C"                                  *
;    -CHKS FOR CHAR "CR"--NO ECHO                              *
; EXIT with:AL=RESULT                                          *
;           :CY=1                                              *
;***************************************************************
GET_KYBD:       LDB  AL,SBUF                ;Store SFR byte into AL
                JBS  RAM_RUNFLG,0,NO_CR     ;NO CHAR ECHO IF HOST PRESENT
                CMPB AL,#03H                ;CHK FOR CNTRL C
                BE   FND_CNTRL_C
                CMPB AL,#0DH                ;DO NOT ECHO CR
                BE   NO_CR
                CALL VIDEO_OUTx             ;ECHO CHAR
NO_CR:          SETC                        ;FLAG,CY=1
                RET

FND_CNTRL_C:    PUSHA                       ;CLRS IMASK,IMASK1
                LDB  IPEND,#00H             ;CLR IPEND
                LDB  IPEND1,#00H            ;CLR IPEND1
                ORB  IMASK1,#00000010B      ;Enable receiver interrupt
                LD   SP,#STACK_VALUE        ;RESET STACK PTR
                BR   MENU                   ;BACK TO "MENU" FILE
;***************************************************************
```

```
;************************ WAIT_KYBDTOUT ************************
; WAIT FOR CHAR RECEIVED(HARDWARE UART)-HAS TIMEOUT              *
; 3 SECOND TIMEOUT                                               *
; EXIT  with:AL=CHAR                                             *
;           :RAM_RUNFLG MSB=1 FOR TIMEOUT                        *
;           :           MSB=0 FOR NO TIMEOUT                     *
;****************************************************************
WAIT_KYBDTOUT: PUSH CX
               PUSH DX
               LDB  DL,#10H                ;TIMEOUT VALUE
               CLR  CX                     ;       "
               CLRC                        ;CY=0
WAIT_TOUT:     JC   EXIT_TOUT              ;INT WILL SET CY
               DJNZ CL,WAIT_TOUT
               DJNZ CH,WAIT_TOUT
               DJNZ DL,WAIT_TOUT
               ORB  RAM_RUNFLG,#80H        ;SET MSB FOR ERR FLG
               POP  DX
               POP  CX
               RET

EXIT_TOUT:     ANDB RAM_RUNFLG,#7FH        ;CLR MSB FOR NO TIMEOUT
               POP  DX
               POP  CX
               RET
;****************************************************************



;************************ VIDEO_OUT ****************************
; SEND ASCII CHAR OUT                                            *
; ENTER with:AL=CHAR                                             *
;****************************************************************
VIDEO_OUT:     JBC  RAM_SIOFLG,0,VIDEO_OUTx  ;BIT #0=1 FOR SOFT UART
               BR   SOFT0_OUT

VIDEO_OUTx:    JBC  SP_STAT,3,$            ;WAIT FOR TXE TO BE SET
               STB  AL,SBUF                ;Send byte
;DUMB TERMINAL NEEDS DELAY @ 19.2 KBaud-OTHERWISE OMIT
IF KB192
               CALL DLY_100us
               CALL DLY_100us
               CALL DLY_100us
               CALL DLY_100us
               CALL DLY_100us
               CALL DLY_100us
               CALL DLY_100us
               CALL DLY_100us
ENDIF
               RET
;****************************************************************
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;********************** SOFTWARE UARTS ************************
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;


;************************** SOFT0_IN ****************************
; SOFTWARE RCVR INPUT ON P1.1                                   *
; ENTER with:RAM_BAUD+00=BIT RATE-LSB                           *
;            :        +01=        -MSB                         *
; EXIT with:AL=DATA BYTE                                        *
;***************************************************************
SOFT0_IN:
GET_SOFT:     PUSH BX
              PUSH CX
              PUSH DX
              PUSH BP
              LD   BP,#RAM_BAUD
              LD   CX,[BP]              ;CX=FULL BIT DELAY
              LD   DX,CX
              SHR  DX,#1                ;DX=BAUD/2
              LDB  BL,#09H              ;# BITS TO RECEIVE+1
              LDB  AL,#00H
WAIT_RCV2:    LDB  AH,P1                ;WAIT FOR T0=0
              JBS  AH,1,WAIT_RCV2
GET_KYBD2:    DEC  DX                   ;DELAY 1/2 BIT TIME
              BNE  GET_KYBD2
              LDB  AH,P1
              JBS  AH,1,GET_SOFT        ;FALSE START

RCV_LOOP:     DEC  CX                   ;WAIT FULL BIT TIME
              BNE  RCV_LOOP
              DJNZ BL,RD_RCVBIT
;             CLRC                      ;CY=0
;             LDB  AH,P1
;             JBS  AH,1,RCV_DONE        ;EXIT IF T0=1 (STOP BIT)
;             SETC                      ;SET ERROR FLG (FRAME ERROR)
RCV_DONE:     CMPB AL,#03H              ;CHK FOR CNTRL C
              BE   FND_SCNTRL_C
              CMPB AL,#0DH              ;DO NOT ECHO CR
              BE   NO_S_CR
              CALL SOFT0_OUT            ;ECHO CHAR
NO_S_CR:      POP  BP
              POP  DX
              POP  CX
              POP  BX
              SETC                      ;FLAG,CY=1
              RET

RD_RCVBIT:    LDB  AH,P1
              SHRB AH,#1                ;SHIFT INTO BIT 0
              SHR  AX,#1                ;SHIFT RIGHT INTO AL
              LD   CX,[BP]              ;RELOAD DELAY REG
              BR   RCV_LOOP
```

```
FND_SCNTRL_C: PUSHA                         ;CLRS IMASK,IMASK1
              LDB  IPEND,#00H               ;CLR IPEND
              LDB  IPEND1,#00H              ;CLR IPEND1
              ORB  IMASK1,#00000010B        ;Enable receiver interrupt
              LD   SP,#STACK_VALUE          ;RESET STACK PTR
              BR   MENU                     ;BACK TO "START" FILE
;*************************************************************



;************************** SOFT0_OUT *************************
; SOFTWARE TRANSMIT OUT ON P1.0 (MUST BE ON LSB)               *
; IN "EQU.SRC" SET SOFTBAUD_9600=0042H   (16MHz)               *
; ENTER with:AL=DATA BYTE                                      *
;           :RAM_BAUD+00=BIT RATE-LSB                          *
;           :        +01=        -MSB                          *
;*************************************************************
SOFT0_OUT:    PUSH AX
              PUSH CX
              PUSH DX
              PUSH BP
              LDB  DL,#0AH                  ;# BITS OUT
              LD   BP,#RAM_BAUD             ;LOAD BIT RATE FROM RAM
              SHL  AX,#1                    ;LSB=0 FOR START BIT
              ORB  AH,#0EH                  ;SET BIT #10=1 FOR STOP BIT
NXT_SOFT:     JBC  AL,0,SND_V0
SND_V1:       ORB  RAM_P1,#01H              ;SET P0.0
              BR   BIT_VOUT
SND_V0:       ANDB RAM_P1,#0FEH             ;CLR P0.0
BIT_VOUT:     STB  RAM_P1,P1                ;BIT OUT
              SHR  AX,#1                    ;SHIFT RIGHT TO B0
              LD   CX,[BP]
BIT_VDLY:     DEC  CX
              BNE  BIT_VDLY
              DJNZ DL,NXT_SOFT
              POP  BP
              POP  DX
              POP  CX
              POP  AX
              RET
;*************************************************************



;NOT USED
;************************** SOFT1_OUT *************************
; SOFTWARE TRANSMIT OUT ON P1.7 (MUST BE ON MSB)               *
; IN "EQU.SRC" SET SOFTBAUD_9600=0042H   (16MHz)               *
; ENTER with:AL=DATA BYTE                                      *
;           :RAM_BAUD1+00=BIT RATE-LSB                         *
;           :        1+01=        -MSB                         *
;*************************************************************
SOFT1_OUT:    PUSH AX
```

```
                PUSH CX
                PUSH DX
                PUSH BP
                LDB  DL,#0AH                ;# BITS OUT
                LD   BP,#RAM_BAUD1          ;LOAD BIT RATE FROM RAM
                SHL  AX,#1                  ;LSB=0 FOR START BIT
                ORB  AH,#0EH                ;SET BIT #10=1 FOR STOP BIT
NXT_SND:        JBC  AL,0,SND_0
SND_1:          ORB  RAM_P1,#80H            ;SET P1.7
                BR   BIT_OUT
SND_0:          ANDB RAM_P1,#7FH            ;CLR P1.7
BIT_OUT:        STB  RAM_P1,P1              ;BIT OUT
                SHR  AX,#1                  ;SHIFT RIGHT TO BIT 0
                LD   CX,[BP]
SND_DLY:        DEC  CX
                BNE  SND_DLY
                DJNZ DL,NXT_SND

                POP  BP
                POP  DX
                POP  CX
                POP  AX
                RET
;***************************************************************



;NOT USED
;************************** SOFT2_OUT ***************************
; SOFTWARE TRANSMIT OUT ON HSO.0                                *
; IN "EQU.SRC" SET SOFTBAUD_9600=003CH    (16MHz)               *
; ENTER with:AL=DATA BYTE                                       *
;            :RAM_BAUD+00=BIT RATE-LSB                          *
;            :        +01=        -MSB                          *
;***************************************************************
SOFT2_OUT:      PUSH AX
                PUSH CX
                PUSH DX
                PUSH BP
                LDB  DL,#0AH                ;# BITS OUT
                LD   BP,#RAM_BAUD           ;LOAD BIT RATE FROM RAM
                SHL  AX,#1                  ;LSB=0 FOR START BIT
                ORB  AH,#0EH                ;SET BIT #10=1 FOR STOP BIT
NXT_HOUT:       JBC  AL,0,SND_H0
SND_H1:         CALL SET_HSO0               ;SET HSO.0
                BR   BIT_OUT
SND_H0:         CALL CLR_HSO0               ;CLR HSO.0
BIT_HOUT:       SHR  AX,#1                  ;SHIFT RIGHT TO BIT0
                LD   CX,[BP]
BIT_HDLY:       DEC  CX
                BNE  BIT_HDLY
                DJNZ DL,NXT_HOUT
                CALL DLY_9600
                POP  BP
                POP  DX
                POP  CX
```

```
                POP  AX
                RET
;***************************************************************



;************************** DLY_1200 ***************************
;                                                              *
;***************************************************************
DLY_1200:       PUSH CX
                LD   CX,#5800H
DLYx_1200:      DEC  CX
                BNE  DLYx_1200
                POP  CX
                RET
;***************************************************************



;************************** DLY_9600 ***************************
;                                                              *
;***************************************************************
DLY_9600:       PUSH CX
                LD   CX,#0B00H
DLYx_9600:      DEC  CX
                BNE  DLYx_9600
                POP  CX
                RET
;***************************************************************



;************************** SET_SOFTBAUD ************************
; SET BAUD RATE FOR A SOFTWARE UART                             *
; LOAD RAM_BAUD+00=BIT DELAY-LSB                                *
;               +01=           MSB                            *
;***************************************************************
SET_SOFTBAUD: LD   RAM_BAUD,#SOFT_9600
                LD   RAM_BAUD1,#SOFT_9600
                RET
;***************************************************************



;************************** CHKOUT_SIO **************************
; SOFT SIO                                                      *
;***************************************************************
CHKOUT_SIO:
;               CALL CLR_SCRN
NXT_A:          LDB  AL,#41H                ;ASCII "A"
;               CALL SOFT0_OUT              ;SOFTWARE SIO-9600 BAUD
```

```
;               CALL SOFT1_OUT              ;SOFTWARE SIO-9600 BAUD
                CALL VIDEO_OUTx
                CALL DLY_100ms
                BR   NXT_A
;*************************************************************




;************************** SIO_MENU ***************************
; SIO CHKOUT MENU                                             *
;*************************************************************
SIO_MENU:       CALL MENU_PREP
                EI
MENU_1:         CALL WAIT_KYBD              ;AL=RESULT
                CMPB AL,#0FH                ;CNTRL O=OUTPUT A"S
                BE   SIOOUT_CHKOUT
                CMPB AL,#09H                ;CNTRL I=INPUT CHAR
                BE   SIOIN_CHKOUT

;ERROR MESSAGE & TRY AGAIN
                CALL CMND_ERROR
                BR   MENU_1
;*************************************************************




;************************** MENU_PREP **************************
;                                                             *
;*************************************************************
MENU_SENDS      EQU  03H                    ;INCLUDES TITLE
MENU_INDENTS    EQU  10H

MENU_PREP:      CALL CLR_SCRN
                LD   SI,#MENU_TABLE
                LDB  CL,#MENU_INDENTS       ;# OF INDENTS
                LDB  CH,#MENU_SENDS         ;# OF SENDS
                CALL SND_TABLE
                RET

;LOOKUP FROM EPROM
MENU_TABLE:     DCW  MENU_TITLE             ;POINTERS
                DCW  MENU_OUT
                DCW  MENU_IN

MENU_TITLE:     DCB  'SIO CHKOUT MENU',0AH,0AH,0DH,00H
MENU_OUT:       DCB  'CNTRL O=CONTINUOUS OUTPUT CHAR A',0AH,0AH,0DH,00H
MENU_IN:        DCB  'CNTRL I=INPUT CHAR',0AH,0AH,0DH,00H
;*************************************************************




;************************ SIOOUT_CHKOUT ************************
;                                                             *
```

```
;*************************************************************
SIOOUT_CHKOUT:  CALL CLR_SCRN
NXT_OUT:        LDB  AL,#41H                ;ASCII "A"
                CALL VIDEO_OUT
                CALL DLY_1ms
                BR   NXT_OUT
;*************************************************************



;************************ SIOIN_CHKOUT ************************
;                                                             *
;*************************************************************
SIOIN_CHKOUT:  CALL CLR_SCRN
NXT_IN:        LD   SI,#MSG_INPUT
               CALL SND_NULL
               CALL WAIT_KYBD              ;RD CHAR
               CALL SND_BYTE
               BR   NXT_IN

MSG_INPUT:     DCB  0AH,0DH,'CHAR=',00H
;*************************************************************



;************************ SET_SOFT_FLG ************************
; RAM_SIOFLG: BIT 0=1 FOR SOFT0 UART IN/OUT                   *
;*************************************************************
SET_SOFT_FLG: ORB  RAM_SIOFLG,#01H      ;BIT 0=1
              RET
;*************************************************************



;************************ SET_HARD_FLG ************************
; RAM_SIOFLG: BIT 0=0 FOR HARDWARE UART IN/OUT                *
;*************************************************************
SET_HARD_FLG: ANDB RAM_SIOFLG,#0FEH     ;BIT 0=0
              RET
;*************************************************************
END



--
*************************************************************************
******
-- Run.src
--
*************************************************************************
******
$TITLE('RUN')
$PAGELENGTH(999)
```

```
;************************** RUN MODULE **************************
;                                                                *
;*****************************************************************
; RUN_MENU                                                       *
; RUN                                                            *
; FP_SWCH_INT                                                    *
; LOAD_FILM                                                      *
; SCAN_FILM                                                      *
; UNLOAD_FILM                                                    *
; LD_PTSNo                                                       *
; WAIT_DOOR_OPEN                                                 *
; WAIT_DOOR_CLOSED                                               *
; DLY_CNT_TIME                                                   *
; DLY_RD_DLY                                                     *
; MENU_SETUP                                                     *
; AUDIO_ON                                                       *
; AUDIO_OFF                                                      *
; DOUBLE_BEEP                                                    *
;*****************************************************************
; P0.7=FP SWCHS                    (EXT INT)                     *
; P0.6=LOAD                                                      *
; P0.5=SCAN                                                      *
; P0.4=UNLOAD                                                    *
; P0.3=DOOR SWCH                                                 *
; P0.2=                                                          *
; P0.1=TEMPERATURE                                               *
; P0.0=DC MTR CURRENT                                            *
;*****************************************************************
                PUBLIC
RUN,RUN_MENU,SCAN_FILM,FP_SWCH_INT,AUDIO_ON,AUDIO_OFF

                EXTRN   EN_EXTINT,DSPLY_CNTS,RD_FPGA_CNTS,CLR_FPGA_CNTS
                EXTRN
SET_SCAN_LED,CLR_SCAN_LED,SET_LOAD_LED,CLR_LOAD_LED
                EXTRN   SET_UNLOAD_LED,CLR_UNLOAD_LED,PMTS_ON,PMTS_OFF
                EXTRN
LIGHTS_ON,LIGHTS_OFF,RUN_STEP_IN,RUN_STEP_OUT,RD_TEMP
                EXTRN
RUN_DC_LEFT,RUN_DC_RIGHT,DLY_100ms,DLY_1sec,CLR_SCRN
                EXTRN   RUN_STEP_AHEAD,RUN_STEP_BACK,AUDIO_BEEP,WAIT_KYBD
                EXTRN   CMND_ERROR,SND_TABLE,ENTR_SHOW_WORD,ENTR_SHOW_BYTE
                EXTRN   SND_NULL,CLR_RUNFLG,SND_DATA_HOST,FILL_DUMMY_CNTS
                EXTRN
DIS_EXTINT,DLY_500ms,SET_RUNFLG,CLR_RAMx,SAVE_DATA_RAMx

$INCLUDE(8086_REGS.INC)
$INCLUDE(SFR.INC)
                EXTRN
RAM_P1,CNT_TIME,No_SCAN_STEPS,RAM_PTSCNTR,RAM_NOHOST
                EXTRN
RAM_NoSCANSTEPS,RD_DLY,RAM_CNTTIME,RAM_RDDLY,RAM_P2
                EXTRN   STACK_VALUE,RAM_RUNFLG,RAM_DATA

CSEG
;*****************************************************************
```

```
;*************************** RUN_MENU ***************************
;                                                                *
;*****************************************************************
RUN_MENU:       CALL MENU_PREP
MENU_1:         CALL WAIT_KYBD              ;AL=RESULT
                CMPB AL,#12H                ;CNTRL R=RUN SYSTEM
                BE   M_RUN
                CMPB AL,#0CH                ;CNTRL L=LOAD FILM
                BE   M_LOAD
                CMPB AL,#13H                ;CNTRL S=SCAN FILM
                BE   M_SCAN
                CMPB AL,#15H                ;CNTRL U=UNLOAD FILM
                BE   M_UNLOAD
                CMPB AL,#18H                ;CNTRL X=RUN SETUP
                BE   MENU_SETUP
                CMPB AL,#0DH                ;CR=RETURN
                BE   M_RET

;ERROR MESSAGE & TRY AGAIN
                CALL CMND_ERROR
                BR   MENU_1

M_RET:          RET

M_RUN:          CALL CLR_RUNFLG
                BR   RUN

M_LOAD:         CALL LOAD_FILM
                BR   RUN_MENU

M_SCAN:         CALL SCAN_FILM
                CALL WAIT_KYBD
                BR   RUN_MENU

M_UNLOAD:       CALL UNLOAD_FILM
                BR   RUN_MENU
;*****************************************************************



;************************ MENU_PREP ****************************
;                                                                *
;*****************************************************************
MENU_SENDS      EQU  06H                    ;INCLUDES TITLE
MENU_INDENTS    EQU  10H

MENU_PREP:      CALL CLR_SCRN
                LD   SI,#MENU_TABLE
                LDB  CL,#MENU_INDENTS       ;# OF INDENTS
                LDB  CH,#MENU_SENDS         ;# OF SENDS
                CALL SND_TABLE
                RET

;LOOKUP FROM EPROM
```

```
MENU_TABLE:    DCW  MENU_TITLE            ;POINTERS
               DCW  RUNS
               DCW  LOAD
               DCW  SCAN
               DCW  UNLOAD
               DCW  SETUP

MENU_TITLE:    DCB  'RUN MENU',0AH,0DH,00H
RUNS:          DCB  'CNTRL R=RUN SYSTEM',0AH,0AH,0DH,00H
LOAD:          DCB  'CNTRL L=LOAD FILM',0AH,0AH,0DH,00H
SCAN:          DCB  'CNTRL S=SCAN FILM',0AH,0AH,0DH,00H
UNLOAD:        DCB  'CNTRL U=UNLOAD FILM',0AH,0AH,0DH,00H
SETUP:         DCB  'CNTRL X=SETUP',0AH,0AH,0DH,00H
;*************************************************************



;***************************** RUN ****************************
;                                                             *
;*************************************************************
RUN:           LD   SP,#STACK_VALUE
               JBS  RAM_RUNFLG,0,SKIP_RUN
               CALL CLR_SCRN
               LD   SI,#MSG_RUN
               CALL SND_NULL
SKIP_RUN:      CALL EN_EXTINT             ;FP SWITCHES INTERRUPT
               EI
               BR   $                     ;WAIT FP INTERRUPT

MSG_RUN:       DCB  'RUN wo/HOST-WAITING FP SWITCHES',0AH,0AH,0DH,00H
;*************************************************************



;************************** FP_SWCH_INT ***********************
; P0.7=FP SWCHS                    (EXT INT)                  *
; P0.6=LOAD SWCH                                              *
; P0.5=SCAN SWCH                                              *
; P0.4=UNLOAD SWCH                                            *
; P0.3=DOOR SWCH                                              *
; P0.2=                                                       *
; P0.1=TEMPERATURE                                            *
; P0.0=DC MTR CURRENT                                         *
;                                                             *
; RAM_RUNFLG                                                  *
;    BIT #7=1,SIO TIMEOUT                                     *
;    BIT #6=                                                  *
;    BIT #5=                                                  *
;    BIT #4=                                                  *
;    BIT #3=1,UNLOADING                                       *
;    BIT #2=1,SCANNING                                        *
;    BIT #1=1,LOADING                                         *
;    BIT #0=1,RUN MODE                                        *
;*************************************************************
```

```
FP_SWCH_INT:
              LDB   AL,P0                    ;RD SWITCH PORT
              JBS   RAM_RUNFLG,2,STOP        ;CHK IF SCANNING
              JBS   AL,6,SW_LD
              JBS   AL,5,SW_SCAN
              JBS   AL,2,SW_SCAN
              JBS   AL,4,SW_UNLD
              RET

SW_LD:        CALL  LOAD_FILM
              JBS   RAM_RUNFLG,0,RECHK_CLOSED
              LD    SI,#MSG_CLOSED
              CALL  SND_NULL
RECHK_CLOSED: LDB   AL,P0                    ;WAIT DOOR CLOSED
              JBS   AL,3,DOOR_CLOSED
              CALL  AUDIO_BEEP
              CALL  DLY_500ms
              BR    RECHK_CLOSED
DOOR_CLOSED:  BR    RUN


SW_SCAN:      CALL  SCAN_FILM
              BR    RUN


SW_UNLD:
              JBS   RAM_RUNFLG,0,RECHK_OPEN
              LD    SI,#MSG_OPEN
              CALL  SND_NULL
RECHK_OPEN:   LDB   AL,P0                    ;WAIT DOOR OPEN
              JBC   AL,3,DOOR_OPEN
              CALL  AUDIO_BEEP
              CALL  DLY_500ms
              BR    RECHK_OPEN
DOOR_OPEN:    CALL  UNLOAD_FILM
              BR    RUN

;ANY SWITCH CAN ABORT A SCAN
STOP:         CALL  CLR_SCAN_LED
              CALL  PMTS_OFF
              BR    RUN

CONTINUE:     RET

MSG_CLOSED:   DCB   0AH,0AH,0DH,'WAITING DOOR CLOSED',00H
MSG_OPEN:     DCB   0AH,0AH,0DH,'WAITING DOOR OPEN',00H
;*************************************************************




;*************************** LOAD_FILM ************************
;                                                             *
;*************************************************************
LOAD_FILM:
              JBS   RAM_RUNFLG,0,SKIP_LD
              CALL  CLR_SCRN
```

```
                LD   SI,#MSG_LOAD
                CALL SND_NULL
SKIP_LD:        CALL DIS_EXTINT
                CALL SET_LOAD_LED
                CALL RUN_STEP_IN            ;ADVANCE STEPPER TO START
POSITION
                CALL CLR_LOAD_LED
                CALL DOUBLE_BEEP            ;OPERATOR NOTIFICATION
                RET

MSG_LOAD:       DCB  'LOADING FILM',00H
;****************************************************************



;************************** SCAN_FILM ***************************
;                                                               *
;****************************************************************
SCAN_FILM:
                JBS  RAM_RUNFLG,0,SKIP_SCAN
                CALL CLR_SCRN
                LD   SI,#MSG_SCAN
                CALL SND_NULL
SKIP_SCAN:      CALL CLR_FPGA_CNTS          ;PMT CNTS
                CALL CLR_RAMx               ;EXT DATA MEMORY
                CLRB RAM_NOHOST             ;HOST ALIVE FLG
                CLRB RAM_PTSCNTR            ;CNTR
                CALL PMTS_ON                ;PMT POWER ON
                CALL SET_SCAN_LED           ;SETS FLG
                CALL RUN_DC_RIGHT           ;TRANSLATE LIGHTS TO LIMIT
;               CALL FILL_DUMMY_CNTS        ;CHKOUT ONLY
;               LD   CX,#No_SCAN_STEPS      ;EACH DIRECTION
                LD   CX,RAM_NoSCANSTEPS     ;EACH DIRECTION
NXT_STEP0:      PUSH CX
                CALL CLR_FPGA_CNTS
                EI                          ;ABORT WITH FP SWITCHES
                NOP
                DI
                CALL LIGHTS_ON
                CALL DLY_CNT_TIME           ;STATIONARY TIME AT EACH
POSITION
                CALL LIGHTS_OFF
                CALL DLY_RD_DLY             ;WAIT TO RD CNTRS
                CALL RD_FPGA_CNTS           ;STORE @ RAM_CNTS=102H
                CALL LD_PTSNo               ;MOVE LOCATION # TO RAM_DATA
                CALL RD_TEMP                ;STORE @ RAM_TEMP=101H
                CALL SND_DATA_HOST          ;CHKS FLGS FOR HOST OR DEBUG
                                            ;HAS EI/DI
                CALL SAVE_DATA_RAMx         ;EXTERNAL RAMx DATA
MEMORY=8000H
                INCB RAM_PTSCNTR            ;CNTR @ RAM_PTSCNTS=100H
                POP  CX
                DEC  CX
                JE   REVERSE
                CALL AUDIO_ON
                CALL RUN_STEP_BACK          ;RUN 1 cm
```

```
                CALL AUDIO_OFF
                BR   NXT_STEP0

REVERSE:        CALL RUN_DC_LEFT            ;TRANSLATE LIGHTS
;               LD   CX,#No_SCAN_STEPS      ;EACH DIRECTION
                LD   CX,RAM_NoSCANSTEPS     ;EACH DIRECTION
NXT_STEP1:      PUSH CX
                CALL CLR_FPGA_CNTS
                EI                          ;FOR FP SWITCHES
                DI
                CALL LIGHTS_ON
                CALL DLY_CNT_TIME           ;STATIONARY TIME AT EACH
POSITION
                CALL LIGHTS_OFF
                CALL DLY_RD_DLY             ;WAIT TO RD CNTRS
                CALL RD_FPGA_CNTS           ;STORE @ RAM_CNTS=102H
                CALL LD_PTSNo               ;MOVE LOCATION # TO RAM_DATA
                CALL RD_TEMP                ;STORE @ RAM_TEMP=101H
                CALL SND_DATA_HOST          ;CHKS FLGS FOR HOST OR DEBUG
                                            ;HAS EI/DI
                CALL SAVE_DATA_RAMx         ;EXTERNAL RAMx DATA
MEMORY=8000H
                INCB RAM_PTSCNTR            ;CNTR @ RAM_PTSCNTS=100H
                POP  CX
                DEC  CX
                JE   SCAN_DONE
                CALL AUDIO_ON
                CALL RUN_STEP_AHEAD         ;RUN 1 cm
                CALL AUDIO_OFF
                BR   NXT_STEP1

SCAN_DONE:      CALL CLR_SCAN_LED           ;CLRS FLG
                CALL PMTS_OFF
                EI
                CALL DOUBLE_BEEP            ;OPERATOR NOTIFICATION
                RET

MSG_SCAN:       DCB  'SCANNING FILM',00H
;***************************************************************



;************************* UNLOAD_FILM ************************
;                                                              *
;***************************************************************
UNLOAD_FILM:
                JBS  RAM_RUNFLG,0,SKIP_UNLD
                CALL CLR_SCRN
                LD   SI,#MSG_UNLOAD
                CALL SND_NULL
SKIP_UNLD:      CALL DIS_EXTINT
                CALL SET_UNLOAD_LED
                CALL RUN_STEP_OUT
                CALL CLR_UNLOAD_LED
                CALL DOUBLE_BEEP            ;OPERATOR NOTIFICATION
                RET
```

```
MSG_UNLOAD:    DCB   'UNLOADING FILM',00H
;*************************************************************




;************************* LD_PTSNo ****************************
;                                                              *
;*************************************************************
LD_PTSNo:      LD    SI,#RAM_DATA          ;HAD TO DO IT THIS WAY DUE TO
LINKER
               LDB   AL,RAM_PTSCNTR        ;              "
               STB   AL,[SI]               ;              "
               RET
;*************************************************************



;NOT USED
;************************* WAIT_DOOR_OPEN **********************
; P0.7=FP SWCHS                      (EXT INT)                 *
; P0.6=LOAD SWCH                                               *
; P0.5=SCAN SWCH                                               *
; P0.4=UNLOAD SWCH                                             *
; P0.3=DOOR SWCH                                               *
; P0.2=                                                        *
; P0.1=TEMPERATURE                                             *
; P0.0=DC MTR CURRENT                                          *
;*************************************************************
WAIT_DOOR_OPEN:
               LDB   AL,P0
               JBC   AL,3,WAIT_DOOR_CLOSED
               RET
;*************************************************************



;NOT USED
;************************* WAIT_DOOR_CLOSED ********************
; P0.7=FP SWCHS                      (EXT INT)                 *
; P0.6=LOAD SWCH                                               *
; P0.5=SCAN SWCH                                               *
; P0.4=UNLOAD SWCH                                             *
; P0.3=DOOR SWCH                                               *
; P0.2=                                                        *
; P0.1=TEMPERATURE                                             *
; P0.0=DC MTR CURRENT                                          *
;*************************************************************
WAIT_DOOR_CLOSED:
               LDB   AL,P0
               JBS   AL,3,WAIT_DOOR_CLOSED
               RET
;*************************************************************
```

```
;************************* DLY_CNT_TIME ************************
;                                                               *
;***************************************************************
DLY_CNT_TIME:
;               LDB   CL,#CNT_TIME
                LDB   CL,RAM_CNTTIME
DLY_CTIME:      CALL  DLY_100ms
                DJNZ  CL,DLY_CTIME
                RET
;***************************************************************



;************************* DLY_RD_DLY ************************
;                                                               *
;***************************************************************
DLY_RD_DLY:
;               LDB   CL,#RD_DLY
                LDB   CL,RAM_RDDLY
DLY_RTIME:      CALL  DLY_100ms
                DJNZ  CL,DLY_RTIME
                RET
;***************************************************************



;************************* MENU_SETUP ************************
;                                                               *
;***************************************************************
MENU_SETUP:     CALL  CLR_SCRN
                LD    SI,#MSG_SETUP
                LD    BP,#RAM_NoSCANSTEPS
                CALL  ENTR_SHOW_WORD
                LD    BP,#RAM_CNTTIME
                CALL  ENTR_SHOW_BYTE
                LD    BP,#RAM_RDDLY
                CALL  ENTR_SHOW_BYTE
                BR    RUN_MENU

MSG_SETUP:      DCB   'ENTER # SCAN STEPS EACH WAY (0000-FFFF)=',00H
                DCB   0AH,0DH,'ENTER CNT TIME x 100ms (0000-FFFF)=',00H
                DCB   0AH,0DH,'ENTER RD DLY x 100ms (0000-FFFF)=',00H
;***************************************************************



;************************* AUDIO_ON ************************
;                                                               *
;***************************************************************
AUDIO_ON:       ANDB  RAM_P2,#7FH
                STB   RAM_P2,P2
                RET
```

```
;*************************************************************



;************************** AUDIO_OFF ***************************
;                                                              *
;*************************************************************
AUDIO_OFF:     ORB  RAM_P2,#80H
               STB  RAM_P2,P2
               RET
;*************************************************************



;************************** DOUBLE_BEEP *************************
;                                                              *
;*************************************************************
DOUBLE_BEEP:   CALL AUDIO_BEEP            ;OPERATOR NOTIFICATION
               CALL DLY_500ms
               CALL AUDIO_BEEP
               RET
;*************************************************************
END



--
*************************************************************************
******
-- Ramx.src
--
*************************************************************************
******
$TITLE('RAMX')
$PAGELENGTH(75)

;*************************** RAMx ********************************
; THE VOLATILE RAM IS OUTSIDE THE 87C196 & IS IN THE CYPRESS    *
;     CMOS 32Kx8 DEVICE                                        *
; MAPPED 8000-FFFF                                             *
;                                                              *
; NOTE: EMULATOR MUST BE MAPPED FOR USER/TARGET RAM            *
;       --- MAP 8000H LENGTH 32K USER                          *
;*************************************************************
; RAMx_MENU                                                    *
; CLR_RAMx        (CLR SRAM)                                   *
; SAVE_DATA_RAMx                                               *
; DSPLY_RAMx      (MEMORY DSPLY)                               *
;*************************************************************
               PUBLIC  RAMx_MENU,DSPLY_RAMx,CLR_RAMx,SAVE_DATA_RAMx

               EXTRN   WAIT_KYBD,SND_TABLE,CMND_ERROR,CLR_SCRN
               EXTRN   CLR_MEM,MEMx_DSPLY,VIDEO_OUT,DLY_9600
               EXTRN   SND_NULL,DLY_1ms,SND_CR_LF,SND_SP,CR_CONT
```

```
$INCLUDE(8086_REGS.INC)
              EXTRN
RAMx_BASE,RAM_SCRATCH,RAMx_SIZE,RAMx_PTR,No_BYTES_DATA
              EXTRN    RAM_DATA

CSEG
;*****************************************************************




;************************** RAMx_MENU ***************************
;                                                                *
;*****************************************************************
RAMx_MENU:    CALL MENU_PREP
MENU_1:       CALL WAIT_KYBD              ;AL=RESULT

              CMPB AL,#0BH                ;CNTRL K=CLR RAMx
              BE   KLR
              CMPB AL,#04H                ;CNTRL D=DSPLY RAMx
              BE   DSPLY_RAMx
              CMPB AL,#0DH                ;CR=RETURN
              BE   RETURN

;ERROR MESSAGE & TRY AGAIN
              CALL CMND_ERROR
              BR   MENU_1

KLR:          CALL CLR_RAMx
              BR   RAMx_MENU
RETURN:       RET
;*****************************************************************



;************************** MENU_PREP ***************************
;                                                                *
;*****************************************************************
MENU_SENDS    EQU  04H                    ;INCLUDES TITLE
MENU_INDENTS  EQU  10H

MENU_PREP:    CALL CLR_SCRN
              LD   SI,#MENU_TABLE
              LDB  CL,#MENU_INDENTS       ;# OF INDENTS
              LDB  CH,#MENU_SENDS         ;# OF SENDS
              CALL SND_TABLE
              RET

;LOOKUP FROM EPROM
MENU_TABLE:   DCW  MENU_TITLE             ;POINTERS
              DCW  MENU_CLR
              DCW  MENU_DSPLY
              DCW  MENU_EXIT

MENU_TITLE:   DCB  'RAMx-EXTERNAL RAM DATA MENU',0AH,0AH,0DH,00H
```

```
MENU_CLR:      DCB   'CNTRL K=CLR DATA MEMORY',0AH,0AH,0DH,00H
MENU_DSPLY:    DCB   'CNTRL D=DSPLY DATA MEMORY',0AH,0AH,0DH,00H
MENU_EXIT:     DCB   'CNTRL C=STOP/EXIT/MAIN MENU',0AH,0DH,00H
;*************************************************************



;************************** CLR_RAMx ***************************
;                                                             *
;*************************************************************
CLR_RAMx:      LD    CX,#RAMx_SIZE          ;# BYTES
               LD    DT,#RAMx_BASE+00H
               CALL  CLR_MEM
               LD    RAMx_PTR,#RAMx_BASE    ;INIT WRT PTR
               RET
;*************************************************************



;************************ SAVE_DATA_RAMx ***********************
;ENTER with: RAMx_PTR                                         *
;*************************************************************
SAVE_DATA_RAMx:
               LD    SI,#RAM_DATA           ;HAD TO DO IT THIS WAY DUE TO
LINKER
               LD    DT,RAMx_PTR
               LDB   CL,#No_BYTES_DATA
NXT_BYTE:      LDB   AL,[SI]+
               STB   AL,[DT]+
               DJNZ  CL,NXT_BYTE
               ADD   DT,#06H                ;LEAVE 00H FILLERS
               ST    DT,RAMx_PTR
               RET
;*************************************************************



;************************ DSPLY_RAMx ***************************
;                                                             *
;*************************************************************
DSPLY_RAMx:    CALL  CLR_SCRN
               LD    SI,#DATA_KEY
               CALL  SND_NULL
;              CALL  CR_CONT
               CALL  WAIT_KYBD

               LD    RAM_SCRATCH+0AH,#RAMx_BASE
               CALL  MEMx_DSPLY
               BR    RAMx_MENU

DATA_KEY:      DCB   'HEX READOUT KEYS',0AH,0AH,0DH
               DCB   '  +00=POSTION #',0AH,0DH
               DCB   '  +01=TEMPERATURE',0AH,0DH
               DCB   '  +02-29H=PMT CNTS',0AH,0DH
```

```
                DCB   '  +2A-2FH=00',0AH,0DH,00H
;*************************************************************
END



--
*************************************************************************
******
-- Pwm.src
--
*************************************************************************
******
$TITLE('PWM UTILITIES')
$PAGELENGTH(999)

;*************************** PWM-KC ****************************
; THIS MODULE ALSO REQUIRED FOR "DEBUG"                        *
; 87C196KB-1 DEDICATED PWM                                     *
; 87C196KC-3 DEDICATED PWM'S                                   *
;          -4 HSO CAM PWM'S                                    *
; ROUTINES FOR BOTH DEDICATED & HSO PWM'S                      *
;*************************************************************
;                        --HSO PWM'S--                         *
; LD_HSO_PWMS    (HSO.0-3 from [BP]-COMMON REP RATE)           *
;                (ACTIVATES PWM3-6)                            *
; LD1_PWM3       (HSO.0, AX=PULSE WIDTH, DX=PERIOD--T1 AS REF) *
; LD1_PWM4       (HSO.1, AX=PULSE WIDTH, DX=PERIOD--T1 AS REF) *
; LD2_PWM3       (HSO.0, AX=PULSE WIDTH, DX=PERIOD--T2 AS REF) *
; CLR_CAM        (FLUSH CAM-STOPS ALL HSO PWM'S)               *
;                                                              *
;                     --DEDICATED PWM'S--                      *
; SET_PWM_15K    (15KHz FIXED)                                 *
; SET_PWM_31K    (31KHz FIXED)                                 *
; EN_PWM0        (ENABLED @ RESET)                             *
; LD_PWM0        (AL=PWM VALUE--P2.5, 15 or 31KHz)             *
; STOP_PWM0                                                    *
; EN_PWM1        (DISABLED @ RESET)                            *
; LD_PWM1        (AL=PWM VALUE--P1.3, 15 or 31KHz)             *
; STOP_PWM1                                                    *
; EN_PWM2        (DISABLED @ RESET)                            *
; LD_PWM2        (AL=PWM VALUE--P1.4, 15 or 31KHz)             *
; STOP_PWM2                                                    *
;                                                              *
; PWM_MENU                                                     *
;  MENU_PREP                                                   *
;  CHKOUT_PWM         (DEDICATED PWM'S, PWM0-2)                *
;  CHKOUT_HSOPWM      (ONLY HSO.0, PWM3)                       *
;  CONST_HSO_PWMS     (4 CONSTANT HSO PWM3-6,1ms @ 100Hz)      *
;*************************************************************
                PUBLIC  LD_HSO_PWMS,CLR_CAM,PWM_MENU
                PUBLIC  EN_PWM0,LD_PWM0,STOP_PWM0
;               PUBLIC  EN_PWM1,LD_PWM1,STOP_PWM1
;               PUBLIC  EN_PWM2,LD_PWM2,STOP_PWM2
                PUBLIC  SET_PWM_15K,SET_PWM_31K
                PUBLIC  LD1_PWM3,LD1_PWM4,LD2_PWM3
                PUBLIC  CHKOUT_HSOPWM
```

```
                EXTRN   CLR_SCRN,SND_NULL,ENTR_SHOW_BYTE,ENTR1_SHOW_BYTE
                EXTRN   ENTR_ASCII_HEX,WAIT_KYBD,SND_CR_LF,EN_HSOINT
                EXTRN   CMND_ERROR,SND_TABLE

$INCLUDE(8086_REGS.INC)
$INCLUDE(SFR.INC)
                EXTRN   HWin0,HWin1,HWin15,RAM_SCRATCH

CSEG
;*************************************************************




;************************ LD_HSO_PWMS ************************
; 4 PWM'S FROM HSO.0-3                                        *
; T1 AS REF (1.00 us @ 16MHz)                                 *
; NOTE: INTERNAL EVENTS CAN ONLY BE "CANCELED" BY CLR_CAM     *
; SEQUENCE                                                    *
;   CLR TIMER1                                                *
;   FLUSH CAM & ENABLE CAM LOCK FOR REPETATIVE WAVEFORMS      *
;   ENABLE HSO INTERRUPT                                      *
;   LOAD FOLLOWING 6 CAM HSO CMNDS      (CAM CAN HOLD 8 CMNDS) *
;     (1) SET HSO.0-5 LINES @ T1=TIME_TO_SET                  *
;     (2) CLR HSO.0 @ T1=PULSE WIDTH+HSO_SET_TIME             *
;     (3) CLR HSO.1 @ T1=PULSE WIDTH+HSO_SET_TIME             *
;     (4) CLR HSO.2 @ T1=PULSE WIDTH+HSO_SET_TIME             *
;     (5) CLR HSO.3 @ T1=PULSE WIDTH+HSO_SET_TIME             *
;     (6) ENABLE CAM INTERRUPT WHICH CLRS T1                  *
;          -SETS RATE @ T1=HSO_REP_TIME+HSO_SET_TIME          *
;          -IF CMND #6 IS OMITTED THEN REP RATE=15Hz          *
;                                                             *
; ENTER with:BP=PULSE WIDTH PTR                               *
;           :[BP+00]=PULSE WIDTH HSO.0     (PWM3)             *
;           :[BP+02]=PULSE WIDTH HSO.1     (PWM4)             *
;           :[BP+04]=PULSE WIDTH HSO.2     (PWM5)             *
;           :[BP+06]=PULSE WIDTH HSO.3     (PWM6)             *
; EXIT  with:BP=BP+08                                         *
;*************************************************************
HSO_SET_TIME    EQU  03E8H                  ;1K us-MUST BE LONG ENOUGH
                                            ;TO ENSURE ALL CAM IS LOADED
                                            ;BEFORE FIRST EVENT OCCURS
HSO_REP_TIME    EQU  2710H                  ;us (2710H=10Kus=100Hz)

LD_HSO_PWMS:    PUSH AX
                LDB  WSR,#HWin15            ;SWITCH WINDOWS
                LDB  AL,IOC2                ;RD IOC2
                CLR  TIMER1                 ;CLR TIMER1
                LDB  WSR,#HWin0             ;SWITCH WINDOWS
                ORB  AL,#0C0H               ;FLUSH CAM & ENABLE LOCKED
ENTRIES
                STB  AL,IOC2                ;               "
;Enable "HSO INTERRUPT" -HSO interrupt handler clrs TIMER1
                CALL EN_HSOINT              ;ENABLE HSO INTERRUPT
;Ld "HSO_SET_TIME" for HSO.0-5
                LDB  HSO_CMND,#0ACH         ;HSO.0-5=1,LOCK CAM,T1 REF
```

```
                                        ;BIT #7=CAM_LOCKED(1=LOCKED)
                                        ;BIT #6=TIMER_SEL(0=T1,1=T2)
                                        ;BIT #5=PIN_CMD(1=SET,0=CLR)
                                        ;BIT #0-3=CMND_TAG (PG 8-19)
;Ld PULSE WIDTH for HSO.0 from [BP]
            LD   HSO_TIME,#HSO_SET_TIME ;LD Lo TIME (T1 REF)
            SKIP R0                     ;DLY 8 STATE TIMES BEFORE NXT
LD
            SKIP R0                     ;           "
            LDB  HSO_CMND,#80H          ;HSO.0=0,LOCKED,T1 AS REF
                                        ;BIT #7=CAM_LOCKED(1=LOCKED)
                                        ;BIT #6=TIMER_SEL(0=T1,1=T2)
                                        ;BIT #5=PIN_CMD(1=SET,0=CLR)
                                        ;BIT #0-3=CMND_TAG (PG 8-19)
            LD   AX,[BP]+               ;AX=PULSE WIDTH
            ADD  HSO_TIME,AX,#HSO_SET_TIME  ;LD Hi TIME (T1 REF)
;Ld PULSE WIDTH for HSO.1 from [BP]
            LDB  HSO_CMND,#81H          ;HSO.1=0,LOCKED,T1 AS REF
            LD   AX,[BP]+               ;AX=PULSE WIDTH
            ADD  HSO_TIME,AX,#HSO_SET_TIME  ;LD Hi TIME (T1 REF)
            SKIP R0                     ;DLY 8 STATE TIMES BEFORE NXT
LD
            SKIP R0                     ;           "
;Ld PULSE WIDTH for HSO.2 from [BP]
            LDB  HSO_CMND,#82H          ;HSO.2=0,LOCKED,T1 AS REF
            LD   AX,[BP]+               ;AX=PULSE WIDTH
            ADD  HSO_TIME,AX,#HSO_SET_TIME  ;LD Hi TIME (T1 REF)
            SKIP R0                     ;DLY 8 STATE TIMES BEFORE NXT
LD
            SKIP R0                     ;           "
;Ld PULSE WIDTH for HSO.3 from [BP]
            LDB  HSO_CMND,#83H          ;HSO.3=0,LOCKED,T1 AS REF
            LD   AX,[BP]+               ;AX=PULSE WIDTH
            ADD  HSO_TIME,AX,#HSO_SET_TIME  ;LD Hi TIME (T1 REF)
            SKIP R0                     ;DLY 8 STATE TIMES BEFORE NXT
LD
            SKIP R0                     ;           "
;Ld "HSO_REP_TIME" to generate interrupt which clrs TIMER1
            LDB  HSO_CMND,#9CH          ;HSO.0=0,LOCKED,T1 AS REF
                                        ;BIT #7=CAM_LOCKED(1=LOCKED)
                                        ;BIT #6=TIMER_SEL(0=T1,1=T2)
                                        ;BIT #5=PIN_CMD(1=SET,0=CLR)
                                        ;BIT #4=ENABLE INTERRUPT
                                        ;BIT #0-3=CMND_TAG (PG 8-19)
            LD   AX,#HSO_REP_TIME       ;AX=REP RATE TIME us
            ADD  HSO_TIME,AX,#HSO_SET_TIME  ;LD INTERRUPT TIME
            POP  AX
            RET
;*************************************************************


;************************ LD1_PWM3 ****************************
; PULSE TRAIN FROM HSO.0 PIN (PIN #28) with Timer1 AS REF      *
; USES CAM-8 ENTRIES POSSIBLE                                  *
```

```
; T1 AS REF (1.00 us @ 16MHz)                                    *
;           ( .80 us @ 20MHz)                                    *
; USE CAM LOCK FOR REPETATIVE WAVEFORM                           *
; ENTER with:AX=PWM "PULSE WIDTH"--usecs                         *
;           :DX=PWM "PERIOD"-------usecs                         *
;*****************************************************************
OFFSET1       EQU  0005H                ;FOR TIMER1 RESET-SETS
                                        ;REP RATE
LD1_PWM3:     ADD  AX,#OFFSET1
              PUSH AX
              LDB  WSR,#HWin15          ;Set HWinDOW=15
              LDB  AL,IOC2              ;RD IOC2
              CLR  TIMER1               ;TIMER1=0000
              LDB  WSR,#HWin0           ;Set HWinDOW=0
;REWRITE IOC2
              ORB  AL,#0C0H             ;FLUSH CAM & ENABLE LOCKED
ENTRIES
              STB  AL,IOC2              ;CAM_CLR=IOC2.7
              POP  AX
;Enable "HSO INTERRUPT" -HSO interrupt handler clrs TIMER1
              CALL EN_HSOINT            ;ENABLE HSO INTERRUPT

;PROGRAM HSO UNIT-TIMER1 as reference
;PROGRAM ABSOLUTE "TIME TO SET"
              LDB  HSO_CMND,#0A0H       ;HSO.0=1,LOCK CAM,T1 REF
                                        ;BIT #7=CAM_LOCKED(1=LOCKED)
                                        ;BIT #6=TIMER_SEL(0=T1,1=T2)
                                        ;BIT #5=PIN_CMD(1=SET,0=CLR)
                                        ;BIT #0-3=CMND_TAG (PG 8-19)
              PUSH DX
              SUB  DX,AX                ;TIME TO SET=PERIOD-WIDTH
              LD   HSO_TIME,DX          ;LD Lo TIME (T1 REF)
              POP  DX
              SKIP R0                   ;DLY 8 STATE TIMES BEFORE NXT
LD
              SKIP R0                   ;            "
;PROGRAM ABSOLUTE "TIME TO CLR"
              LDB  HSO_CMND,#80H        ;HSO.0=0,LOCKED,T1 AS REF
                                        ;BIT #7=CAM_LOCKED(1=LOCKED)
                                        ;BIT #6=TIMER_SEL(0=T1,1=T2)
                                        ;BIT #5=PIN_CMD(1=SET,0=CLR)
                                        ;BIT #0-3=CMND_TAG (PG 8-19)
              LD   HSO_TIME,DX          ;LD Hi TIME (T1 REF)
              SKIP R0                   ;DLY 8 STATE TIMES BEFORE NXT
LD
              SKIP R0                   ;            "
;PROGRAM "PERIOD"-SET INTERRUPT WHICH WILL RESET TIMER1
              LDB  HSO_CMND,#90H        ;
                                        ;BIT #7=CAM_LOCKED(1=LOCKED)
                                        ;BIT #6=TIMER_SEL(0=T1,1=T2)
                                        ;BIT #5=PIN_CMD(1=SET,0=CLR)
                                        ;BIT #4=ENABLE INTERRUPT
;             ADD  DX,#OFFSET1
              SUB  DX,#OFFSET1
              LD   HSO_TIME,DX          ;PERIOD + OFFSET
              RET
;*****************************************************************
```

```
;************************ LD1_PWM4 ****************************
; PULSE TRAIN FROM HSO.1 PIN (PIN #29) with Timer1 AS REF      *
; USES CAM-8 ENTRIES POSSIBLE                                  *
; T1 AS REF (1.00 us @ 16MHz)                                  *
; USE CAM LOCK FOR REPETATIVE WAVEFORM                         *
; ENTER with:AX=PWM "PULSE WIDTH"--usecs                       *
;           :DX=PWM "PERIOD"-------usecs                       *
;************************************************************
LD1_PWM4:     PUSH AX
              LDB  WSR,#HWin15          ;Set HWinDOW=15
              LDB  AL,IOC2              ;RD IOC2
              CLR  TIMER1               ;TIMER1=0000
              LDB  WSR,#HWin0           ;Set HWinDOW=0
;REWRITE IOC2
              ORB  AL,#0C0H             ;FLUSH CAM & ENABLE LOCKED
ENTRIES
              STB  AL,IOC2              ;CAM_CLR=IOC2.7
              POP  AX
;Enable "HSO INTERRUPT" -HSO interrupt handler clrs TIMER1
              CALL EN_HSOINT            ;ENABLE HSO INTERRUPT

;PROGRAM HSO UNIT-TIMER1 as reference
;PROGRAM ABSOLUTE "TIME TO SET"
              LDB  HSO_CMND,#0A1H       ;HSO.1=1,LOCK CAM,T1 REF
                                        ;BIT #7=CAM_LOCKED(1=LOCKED)
                                        ;BIT #6=TIMER_SEL(0=T1,1=T2)
                                        ;BIT #5=PIN_CMD(1=SET,0=CLR)
                                        ;BIT #0-3=CMND_TAG (PG 8-19)
              PUSH DX
              SUB  DX,AX                ;TIME TO SET=PERIOD-WIDTH
              LD   HSO_TIME,DX          ;LD Hi TIME (T1 REF)
              POP  DX
              SKIP R0                   ;DLY 8 STATE TIMES BEFORE NXT
LD
              SKIP R0                   ;            "
;PROGRAM ABSOLUTE "TIME TO CLR"
              LDB  HSO_CMND,#81H        ;HSO.1=0,LOCKED,T1 AS REF
                                        ;BIT #7=CAM_LOCKED(1=LOCKED)
                                        ;BIT #6=TIMER_SEL(0=T1,1=T2)
                                        ;BIT #5=PIN_CMD(1=SET,0=CLR)
                                        ;BIT #0-3=CMND_TAG (PG 8-19)
              LD   HSO_TIME,DX          ;LD Hi TIME (T1 REF)
              SKIP R0                   ;DLY 8 STATE TIMES BEFORE NXT
LD
              SKIP R0                   ;            "
;PROGRAM "PERIOD"-SET INTERRUPT WHICH WILL RESET TIMER1
              LDB  HSO_CMND,#91H        ;
                                        ;BIT #7=CAM_LOCKED(1=LOCKED)
                                        ;BIT #6=TIMER_SEL(0=T1,1=T2)
                                        ;BIT #5=PIN_CMD(1=SET,0=CLR)
                                        ;BIT #4=ENABLE INTERRUPT
```

```
                ADD   DX,#OFFSET1
                LD    HSO_TIME,DX           ;PERIOD + OFFSET
                RET
;*****************************************************************



;NOT USED
;************************* LD2_PWM3 ****************************
; PULSE TRAIN FROM HSO.0 PIN (PIN #28) with T2CLK AS REFERENCE *
; USES CAM-8 ENTRIES POSSIBLE                                  *
; T2CLK AS REF-LIMITED RESOLUTION DUE TO T2 COUNTING SPEED     *
; USE CAM LOCK FOR REPETATIVE WAVEFORM                         *
; ENTER with:AX=PWM "PULSE WIDTH"                              *
;           :DX=PWM "PERIOD"                                   *
;*****************************************************************
OFFSET2         EQU   0005H                 ;FOR TIMER2 RESET-SETS
                                            ;REP RATE
LD2_PWM3:       PUSH  AX
                LDB   WSR,#HWin15           ;Set HWinDOW=15
                LDB   AL,IOC2               ;RD IOC2
                LDB   WSR,#HWin0            ;Set HWinDOW=0
;REWRITE IOC2
                ANDB  AL,#07CH              ;NORMAL CNT MODE,CNT UP
                ORB   AL,#0C0H              ;FLUSH CAM & ENABLE LOCKED
ENTRIES
                STB   AL,IOC2               ;CAM_CLR=IOC2.7
                POP   AX

                CLR   TIMER2                ;TIMER2=0000

;PROGRAM HSO UNIT-TIMER2 as reference
;PROGRAM ABSOLUTE "TIME TO SET"
                LDB   HSO_CMND,#0E0H        ;HSO.0=1,LOCK CAM,T2 REF
                                            ;BIT #0-3=CMND_TAG (PG 8-19)
                                            ;BIT #7=CAM_LOCKED(1=LOCKED)
                                            ;BIT #6=TIMER_SEL(0=T1,1=T2)
                                            ;BIT #5=PIN_CMD(1=SET,0=CLR)
                PUSH  DX
                SUB   DX,AX                 ;TIME TO SET=PERIOD-WIDTH
                LD    HSO_TIME,DX           ;LD Hi TIME (T1 REF)
                POP   DX
                SKIP  R0                    ;DLY 8 STATE TIMES BEFORE NXT
LD
                SKIP  R0                    ;            "
;PROGRAM ABSOLUTE "TIME TO CLR"
                LDB   HSO_CMND,#0C0H        ;HSO.0=0,LOCKED,T2 AS REF
                                            ;BIT #0-3=CMND_TAG (PG 8-19)
                                            ;BIT #7=CAM_LOCKED(1=LOCKED)
                                            ;BIT #6=TIMER_SEL(0=T1,1=T2)
                                            ;BIT #5=PIN_CMD(1=SET,0=CLR)
                LD    HSO_TIME,DX           ;LD Hi TIME (T2 REF)
                SKIP  R0                    ;DLY 8 STATE TIMES BEFORE NXT
LD
                SKIP  R0                    ;            "
;PROGRAM ABSOLUTE TIMER2 RESET-DETERMINES "PERIOD"
```

```
                LDB   HSO_CMND,#0CEH          ;RESET TIMER2(CMND_TAG=0E)
                                              ;BIT #7=CAM_LOCKED(1=LOCKED)
                                              ;BIT #6=TIMER_SEL(0=T1,1=T2)
                ADD   DX,#OFFSET2
                LD    HSO_TIME,DX             ;PERIOD + OFFSET
                RET
;***************************************************************



;************************* CLR_CAM *****************************
; FLUSH CAM & ENABLE LOCKED CAM                                 *
;***************************************************************
CLR_CAM:        PUSH AX
                LDB   WSR,#HWin15             ;Set HWinDOW=15
                LDB   AL,IOC2
                LDB   WSR,#HWin0              ;Set HWinDOW=0
;REWRITE IOC2
                ORB   AL,#0C0H                ;FLUSH CAM & ENABLE LOCKED
ENTRIES
                STB   AL,IOC2
                POP   AX
                RET
;***************************************************************



;$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$
;$$$$$$$$$$$$$$$$$ START OF DEDICATED PWM'S $$$$$$$$$$$$$$$$$$$$
;$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$$



;************************ SET_PWM_15K **************************
; SET REP RATE-SAME FOR PWM0,PWM1 & PWM2                        *
;     (1) SELECT DIVIDE BY 2 PRESCALER (INTERNAL OSC)           *
;        -IOC2.2=1    (11.8 KHz @ 12MHz, 15.63 KHz @ 16MHz)    *
;         IOC2.2=0 upon RESET                                   *
;***************************************************************
SET_PWM_15K:    LDB   WSR,#HWin15             ;Set HWinDOW=15
                LDB   AL,IOC2                 ;AL=IOC2
                LDB   WSR,#HWin0              ;Set HWinDOW=0
;ENABLE DIVIDE BY 2 PRESCALER
                ORB   AL,#04H                 ;SET IOC2.2
                STB   AL,IOC2                 ;REWRITE IOC2
                RET
;***************************************************************



;************************ SET_PWM_31K **************************
```

```
; SET REP RATE-SAME FOR PWM0,PWM1 & PWM2                          *
;    (1) SELECT DIVIDE BY 1 PRESCALER                             *
;       -IOC2.2=0   (23.6 KHz @ 12MHz, 31.25 KHz @ 16MHz)        *
;        IOC2.2=0 upon RESET                                     *
;*****************************************************************
SET_PWM_31K:  LDB  WSR,#HWin15          ;Set HWinDOW=15
              LDB  AL,IOC2              ;AL=IOC2
              LDB  WSR,#HWin0           ;Set HWinDOW=0
;SELECT DIVIDE BY 1 PRESCALER
              ANDB AL,#0FDH             ;CLR IOC2.2
              STB  AL,IOC2              ;REWRITE IOC2
              RET
;*****************************************************************



;************************** EN_PWM0 ****************************
; SELECT P2.5 AS PWM OUTPUT PIN (PIN #39)                         *
;    -IOC1.0=1 (ENABLED)          (RESET=1)                       *
;*****************************************************************
EN_PWM0:      LDB  WSR,#HWin15          ;Set HWinDOW=15
              LDB  AL,IOC1
              LDB  WSR,#HWin0           ;Set HWinDOW=0
              ORB  AL,#01H              ;Set P2.5 AS PWM OUTPUT
              STB  AL,IOC1              ;REWRITE IOC1
              RET
;*****************************************************************



;************************** LD_PWM0 ****************************
; SET HIGH LEVEL TIME---PULSE WIDTH                               *
; 8-BIT CNTR INCREMENTED EVERY STATE TIME                         *
; PWM OUTPUT=1 WHEN CNTR=0                                        *
; PWM OUTPUT=0 WHEN CNTR=PWM VALUE---COMPARATOR GOES Lo           *
; ZERO DUTY CYCLE:PWM=0                                           *
; MAX  DUTY CYCLE:PWM=FF                                          *
; ENTER with:AL=PWM VALUE     (DETERMINES PULSE WIDTHS-Hi TIME)*
;                             (Hi TIME=PWM_CNTRLx2/Fosc)          *
;*****************************************************************
LD_PWM0:      STB  AL,PWM0_CNTRL
              RET
;*****************************************************************



;************************** STOP_PWM0 **************************
; DEDICATED PWM                                                   *
; ZERO DUTY CYCLE:PWM=0                                           *
;*****************************************************************
STOP_PWM0:    LDB  PWM0_CNTRL,#00H
              RET
```

```
;*************************************************************

;*************************** EN_PWM1 ***************************
; SELECT P1.3 AS PWM OUTPUT PIN  (PIN #22)                     *
;     IOC3.2=1 (ENABLED)         (RESET=0)                     *
;*************************************************************
EN_PWM1:      LDB  WSR,#HWin1            ;Set HWinDOW=1
              ORB  IOC3,#04H             ;Set P1.3 AS PWM1 OUTPUT
              LDB  WSR,#HWin0            ;Set HWinDOW=0
              RET
;*************************************************************

;*************************** LD_PWM1 ***************************
; SET HIGH LEVEL TIME---PULSE WIDTH                            *
; SET HIGH LEVEL TIME                                          *
; 8-BIT CNTR INCREMENTED EVERY STATE TIME                      *
; PWM OUTPUT=1 WHEN CNTR=0                                     *
; PWM OUTPUT=0 WHEN CNTR=PWM VALUE---COMPARATOR GOES Lo        *
; ZERO DUTY CYCLE:PWM=0                                        *
; MAX  DUTY CYCLE:PWM=FF                                       *
; ENTER with:AL=PWM VALUE     (DETERMINES PULSE WIDTHS-Hi TIME)*
;                             (Hi TIME=PWM_CNTRLx2/Fosc)       *
;*************************************************************
LD_PWM1:      LDB  WSR,#HWin1            ;Set HWinDOW=1
              STB  AL,PWM1_CNTRL
              LDB  WSR,#HWin0            ;Set HWinDOW=0
              RET
;*************************************************************

;*************************** STOP_PWM1 *************************
; DEDICATED PWM                                                *
; ZERO DUTY CYCLE:PWM=0                                        *
;*************************************************************
STOP_PWM1:    LDB  WSR,#HWin1            ;Set HWinDOW=1
              LDB  PWM1_CNTRL,#00H
              LDB  WSR,#HWin0            ;Set HWinDOW=0
              RET
;*************************************************************

;*************************** EN_PWM2 ***************************
; SELECT P1.4 AS PWM OUTPUT PIN  (PIN #23)                     *
```

```
;      IOC3.3=1 (ENABLED)       (RESET=0)                              *
;*************************************************************
EN_PWM2:       LDB   WSR,#HWin1              ;Set HWinDOW=1
               ORB   IOC3,#08H               ;Set P1.4 AS PWM2 OUTPUT
               LDB   WSR,#HWin0              ;Set HWinDOW=0
               RET
;*************************************************************



;************************** LD_PWM2 ***************************
; SET HIGH LEVEL TIME                                          *
; 8-BIT CNTR INCREMENTED EVERY STATE TIME                      *
; PWM OUTPUT=1 WHEN CNTR=0                                     *
; PWM OUTPUT=0 WHEN CNTR=PWM VALUE---COMPARATOR GOES Lo        *
; ZERO DUTY CYCLE:PWM=0                                        *
; MAX  DUTY CYCLE:PWM=FF                                       *
; ENTER with:AL=PWM VALUE      (DETERMINES PULSE WIDTHS-Hi TIME)*
;                              (Hi TIME=PWM_CNTRLx2/Fosc)      *
;*************************************************************
LD_PWM2:       LDB   WSR,#HWin1              ;Set HWinDOW=1
               STB   AL,PWM2_CNTRL
               LDB   WSR,#HWin0              ;Set HWinDOW=0
               RET
;*************************************************************



;************************** STOP_PWM2 *************************
; DEDICATED PWM                                                *
; ZERO DUTY CYCLE:PWM=0                                        *
;*************************************************************
STOP_PWM2:     LDB   WSR,#HWin1              ;Set HWinDOW=1
               LDB   PWM2_CNTRL,#00H
               LDB   WSR,#HWin0              ;Set HWinDOW=0
               RET
;*************************************************************



;************************** PWM_MENU **************************
; CALLED FROM "MENU & DEBUG"                                   *
;*************************************************************
PWM_MENU:      CALL  MENU_PREP
MENU_1:        CALL  WAIT_KYBD               ;AL=RESULT

               CMPB  AL,#10H                 ;CNTRL P=DEDICATED PWM'S CHKOUT
               BE    CHKOUT_PWM              ;      3 PWM'S
               CMPB  AL,#08H                 ;CNTRL H=HSO PWM CHKOUT
               BE    CHKOUT_HSOPWM
               CMPB  AL,#0BH                 ;CNTRL K=CONSTANT HSO PWM'S
               BE    CONST_HSO_PWMS
```

```
;ERROR MESSAGE & TRY AGAIN
              CALL CMND_ERROR
              BR   MENU_1
;*************************************************************



;************************** MENU_PREP ***************************
;                                                              *
;*************************************************************
MENU_SENDS    EQU  04H                    ;INCLUDES TITLE
MENU_INDENTS  EQU  10H

MENU_PREP:    CALL CLR_SCRN
              LD   SI,#MENU_TABLE
              LDB  CL,#MENU_INDENTS       ;# OF INDENTS
              LDB  CH,#MENU_SENDS         ;# OF SENDS
              CALL SND_TABLE
              RET

;LOOKUP FROM EPROM
MENU_TABLE:   DCW  MENU_TITLE             ;POINTERS
              DCW  MENU_PWM
              DCW  MENU_HSO
              DCW  MENU_KHSO

MENU_TITLE:   DCB  'PWM MENU',0AH,0AH,0DH,00H
MENU_PWM:     DCB  'CNTRL P=DEDICATED PWM CHKOUT',0AH,0AH,0DH,00H
MENU_HSO:     DCB  'CNTRL H=HSO PWM CHKOUT',0AH,0AH,0DH,00H
MENU_KHSO:    DCB  'CNTRL K=CONSTANT HSO PWMS',0AH,0AH,0DH,00H
;*************************************************************



;************************** CHKOUT_PWM **************************
; INTERNAL CLOCK AS REFERENCE-TIMER1                           *
; PULSE TRAIN OUTPUTS ON PWM0,PWM1 & PWM2                      *
;*************************************************************
CHKOUT_PWM:   CALL CLR_SCRN
              CALL SET_PWM_15K            ;SELECT SLOW REP RATE
              CALL EN_PWM0                ;SELECT OUTPUT PINS
              CALL EN_PWM1
              CALL EN_PWM2
NXT_PWM:      LD   SI,#MSG_PWM
              CALL SND_NULL

              LD   BP,#RAM_SCRATCH+0AH              ;DO PWM0
              LDB  WSR,#HWin15
              LDB  RAM_SCRATCH+0AH,PWM0_CNTRL
              LDB  WSR,#HWin0
              CALL ENTR_SHOW_BYTE
              STB  RAM_SCRATCH+0AH,PWM0_CNTRL

              LDB  WSR,#HWin1                       ;DO PWM1
```

```
                LDB   AL,PWM1_CNTRL
                LDB   WSR,#HWin0
                CALL  ENTR1_SHOW_BYTE
                LDB   WSR,#HWin1
                STB   AL,PWM1_CNTRL

                LDB   AL,PWM2_CNTRL                    ;DO PWM2
                LDB   WSR,#HWin0
                CALL  ENTR1_SHOW_BYTE
                LDB   WSR,#HWin1
                STB   AL,PWM2_CNTRL
                LDB   WSR,#HWin0

                CALL  SND_CR_LF                        ;REPEAT
                CALL  WAIT_KYBD
                BR    NXT_PWM

MSG_PWM:        DCB   'SET DEDICATED PWM DUTY CYCLE'
                DCB   '(0%=00,100%=FF-----Z=EXIT)'
                DCB   0AH,0DH,'-REP RATE(IOC2.2)=15 or 31KHz',00H
                DCB   0AH,0DH,'    PWM0=',00H
                DCB   0AH,0DH,'    PWM1=',00H
                DCB   0AH,0DH,'    PWM2=',00H
;*************************************************************



;************************ CHKOUT_HSOPWM ************************
; INTERNAL T1 CLOCK AS REFERENCE  (1us @ 16MHz)                *
; PULSE TRAIN OUTPUT ON HSO.0=PIN 28    (PWM3)                 *
;*************************************************************
CHKOUT_HSOPWM: CALL CLR_SCRN
NXT_HSO:        LD    SI,#MSG_HSO
                CALL  SND_NULL
                CALL  ENTR_ASCII_HEX          ;ENTER Hi TIME
                PUSH  DX                      ;DX=RESULT
                CALL  SND_NULL
                CALL  ENTR_ASCII_HEX          ;ENTER Lo TIME
                                              ;DX=PERIOD
                POP   AX                      ;AX=PULSE WIDTH
                CALL  EN_HSOINT
                CALL  LD1_PWM3                ;T1 AS REF
                CALL  SND_CR_LF
                CALL  WAIT_KYBD
                BR    NXT_HSO                 ;REPEAT

MSG_HSO:        DCB   0AH,0DH,'HSO.0 PULSE WIDTH-us(XXXX)=',00H
                DCB   0AH,0DH,'HSO.0 PERIOD-us(YYYY)=',00H
;*************************************************************



;********************** CONST_HSO_PWMS ***********************
```

```
; CONSTANT 1ms PULSE WIDTH OUT TO HSO.0-.3     (PWM3-6)          *
; REP RATE=100Hz                                                 *
;****************************************************************
CONST_HSO_PWMS:
              CALL CLR_SCRN
              LD   SI,#MSG_KPWMS
              CALL SND_NULL

              CALL CLR_CAM                ;FLUSH CAM

              LD   AX,#03E8H              ;PULSE WIDTH=1000us
              LD   BP,#RAM_SCRATCH        ;LD RAM WITH PW'S
              ST   AX,[BP]+
              ST   AX,[BP]+
              ST   AX,[BP]+
              ST   AX,[BP]
              LD   BP,#RAM_SCRATCH        ;RESET PTR
              CALL LD_HSO_PWMS            ;SND 4 PWM'S OUT
              CALL WAIT_KYBD
              BR   PWM_MENU

MSG_KPWMS:    DCB  'OUTPUTTING 4 HSO PWMS-1ms WIDTH @ 100Hz ',00H
;****************************************************************
END
```

```
--
*******************************************************************************
******
-- Menu.src
--
*******************************************************************************
******
$TITLE('MENU')
$PAGELENGTH(999)

;************************* MENU MODULE ************************
; SYSTEM MENU        (07/99)                                    *
;   -ACCESS WITH EXTERNAL TERMINAL & "CNTRL C"                  *
;*************************************************************
; MENU                                                         *
; MENU_RD_TEMP                                                 *
; RD_TEMP                                                      *
; LIGHTS_ON                                                    *
; LIGHTS_OFF                                                   *
; PMTS_ON                                                      *
; PMTS_OFF                                                     *
; SOLENOID_ON                                                  *
; SOLENOID_OFF                                                 *
; SET_SCAN_LED                                                 *
; CLR_SCAN_LED                                                 *
; SET_LOAD_LED                                                 *
; CLR_LOAD_LED                                                 *
; SET_UNLOAD_LED                                               *
; CLR_UNLOAD_LED                                               *
; FLASH_LEDS                                                   *
```

```
; AUDIO_BEEP                                                    *
; USER_DSPLY_CHKOUT                                             *
; DSPLY_SWITCHES                                                *
;***************************************************************
              PUBLIC  MENU,RD_TEMP
              PUBLIC  LIGHTS_ON,LIGHTS_OFF,PMTS_ON,PMTS_OFF,SET_SCAN_LED
              PUBLIC
CLR_SCAN_LED,SET_LOAD_LED,CLR_LOAD_LED,SET_UNLOAD_LED
              PUBLIC  CLR_UNLOAD_LED,AUDIO_BEEP,FLASH_LEDS

              EXTRN   CLR_SCRN,SND_NULL,CMND_ERROR,DEBUG,WAIT_KYBD
              EXTRN   SND_TABLE,SND_SP,SND_CR_LF,SND_EQUAL,SND_BNBCD2
              EXTRN   SND_WORD_MEM,SND_BYTE,DLY_1sec,RD_AD_8,OUT_HSO
              EXTRN   DC_MTR_MENU,STEP_MTR_MENU,INIT_IO,STOP_DC_MTR
              EXTRN   RUN_MENU,SND_BITS,DSPLY_FPGA_CNTS,CLR_FPGA_CNTS
              EXTRN
EN_FPGA_CNTS,DLY_500ms,AUDIO_ON,AUDIO_OFF,RAMx_MENU

$INCLUDE(8086_REGS.INC)
$INCLUDE(SFR.INC)
              EXTRN   STACK_VALUE,RAM_P3,RAM_P4,RAM_P2,RAM_P1,RAM_HSIO
              EXTRN   No_CHS,RAM_CNTS,RAMx_FPGA,RAM_RUNFLG,RAM_TEMP

CSEG
;***************************************************************



;*************************** MENU ******************************
; MAIN SYSTEM MENU                                              *
; HOST-TO-uP PROTOCOL                                           *
;     (1) WAKEUP CMND=?    (QMARK)                              *
;     (1) WAKEUP CMND=?                                         *
;           -uP SNDS ACK                                        *
;     (3) FUNCTION CMND... XX         (RAM_HOST+0)              *
;     (4) PARAMETER                   (RAM_HOST+1)              *
;     (3) CHKSUM                      (RAM_HOST+2)              *
;           -uP SNDS ACK/NAK                                    *
;***************************************************************
MENU:         LD    SP,#STACK_VALUE       ;RESET STACK POINTER
              CALL  STOP_DC_MTR
              CALL  EN_FPGA_CNTS
              CALL  INIT_IO
MENU_0:       CALL  MENU_PREP
              EI
MENU_1:       CALL  WAIT_KYBD             ;AL=RESULT
              CMPB  AL,#12H               ;CNTRL R=RUN MENU
              BE    M_RUN
              CMPB  AL,#04H               ;CNTRL D=DC MTR MENU
              BE    M_DC_MTR
              CMPB  AL,#13H               ;CNTRL S=STEPPER MTR MENU
              BE    M_STEP_MTR
              CMPB  AL,#05H               ;CNTRL E=EXTERNAL RAMx MENU
              BE    M_RAMxMENU
              CMPB  AL,#18H               ;CNTRL X=RD PMT CNTR
              BE    DSPLY_FPGA_CNTS
```

```
                CMPB AL,#0BH                ;CNTRL K=CLR PMT CNTR
                BE   M_FPGACLR
                CMPB AL,#14H                ;CNTRL T=RD TEMPERATURE
                BE   MENU_RD_TEMP
                CMPB AL,#0FH                ;CNTRL O=LIGHTS ON
                BE   M_ILL_ON
                CMPB AL,#10H                ;CNTRL P=LIGHTS OFF
                BE   M_ILL_OFF
                CMPB AL,#06H                ;CNTRL F=PMTS ON
                BE   M_PMT_ON
                CMPB AL,#07H                ;CNTRL G=PMTS OFF
                BE   M_PMT_OFF
                CMPB AL,#0AH                ;CNTRL J=USER DSPLY CHKOUT
                BE   USER_DSPLY_CHKOUT
                CMPB AL,#0CH                ;CNTRL L=DSPLY SWITCHES
                BE   DSPLY_SWITCHES
                CMPB AL,#1AH                ;CNTRL Z=DEBUG
                BE   DEBUG

;ERROR MESSAGE & TRY AGAIN
                CALL CMND_ERROR
                BR   MENU_1

M_RUN:          CALL RUN_MENU
                BR   MENU

M_DC_MTR:       CALL DC_MTR_MENU
                BR   MENU_0

M_STEP_MTR:     CALL STEP_MTR_MENU
                BR   MENU_0

M_RAMxMENU:     CALL RAMx_MENU
                BR   MENU_0

M_FPGACLR:      CALL CLR_FPGA_CNTS
                BR   MENU_0

M_ILL_ON:       CALL LIGHTS_ON
                BR   MENU_0

M_ILL_OFF:      CALL LIGHTS_OFF
                BR   MENU_0

M_PMT_ON:       CALL PMTS_ON
                BR   MENU_0

M_PMT_OFF:      CALL PMTS_OFF
                BR   MENU_0

;M_SOL_ON:       CALL SOLENOID_ON
;                BR   MENU_0

;M_SOL_OFF:      CALL SOLENOID_OFF
;                BR   MENU_0
;*****************************************************************
```

```
;************************* MENU_PREP ***************************
;                                                              *
;*************************************************************
MENU_SENDS     EQU  0FH                    ;INCLUDES TITLE
MENU_INDENTS   EQU  10H

MENU_PREP:     CALL CLR_SCRN
               LD   SI,#MENU_TABLE
               LDB  CL,#MENU_INDENTS       ;# OF INDENTS
               LDB  CH,#MENU_SENDS         ;# OF SENDS
               CALL SND_TABLE
               RET

;LOOKUP FROM EPROM
MENU_TABLE:    DCW  MENU_TITLE             ;POINTERS
               DCW  M_RUNMENU
               DCW  M_DC
               DCW  M_STEP
               DCW  M_RAMX
               DCW  M_CNTS
               DCW  M_CLR
               DCW  M_TEMP
               DCW  M_ION
               DCW  M_IOFF
               DCW  M_PMTON
               DCW  M_PMTOFF
;              DCW  M_SOLON
;              DCW  M_SOLOFF
               DCW  M_CHKOUT
               DCW  M_SWCHS
               DCW  M_EXIT

MENU_TITLE:    DCB  'OSL DEBUG MENU (07/28/99)',0AH,0DH,00H
M_RUNMENU:     DCB  'CNTRL R=RUN MENU',0AH,0DH,00H
M_DC:          DCB  'CNTRL D=DC MTR MENU',0AH,0DH,00H
M_STEP:        DCB  'CNTRL S=STEPPER MTR MENU',0AH,0DH,00H
M_RAMX:        DCB  'CNTRL E=RAMx DATA MENU',0AH,0DH,00H
M_CNTS:        DCB  '    CNTRL X=RD PMT CNTS',0AH,0DH,00H
M_CLR:         DCB  '    CNTRL K=CLR PMT CNTS',0AH,0DH,00H
M_TEMP:        DCB  '    CNTRL T=RD TEMP',0AH,0DH,00H
M_ION:         DCB  '    CNTRL O=LIGHTS ON',0AH,0DH,00H
M_IOFF:        DCB  '    CNTRL P=LIGHTS OFF',0AH,0DH,00H
M_PMTON:       DCB  '    CNTRL F=PMT POWER ON',0AH,0DH,00H
M_PMTOFF:      DCB  '    CNTRL G=PMT POWER OFF',0AH,0DH,00H
;M_SOLON:       DCB  '    CNTRL H=SOLENOID ON',0AH,0DH,00H
;M_SOLOFF:      DCB  '    CNTRL I=SOLENOID OFF',0AH,0DH,00H
M_CHKOUT:      DCB  '    CNTRL J=BEEPER,LEDS CHKOUT',0AH,0DH,00H
M_SWCHS:       DCB  '    CNTRL L=DSPLY SWITCHES',0AH,0DH,00H
M_EXIT:        DCB  '    CNTRL C=STOP/EXIT/MAIN MENU',0AH,0DH,00H
;*************************************************************
```

```
;************************** MENU_RD_TEMP ***********************
;RD & DSPLY TEMP SENSOR                                         *
;***************************************************************
MENU_RD_TEMP: CALL CLR_SCRN
NXT_TEMP:     LD   SI,#MSG_TEMP
              CALL SND_NULL
              CALL RD_TEMP
              CALL SND_BYTE
              CALL DLY_1sec
              BR   NXT_TEMP

MSG_TEMP:     DCB  0AH,0DH,'TEMP(00-FF)=',00H
;***************************************************************



;***************************** RD_TEMP *************************
;EXIT with: AL=TEMPERATURE                                      *
;***************************************************************
RD_TEMP:      LDB  CL,#01H              ;IGNORE 1st READING
              CALL RD_AD_8
              LDB  CL,#01H              ;CH #
              CALL RD_AD_8
              STB  AL,RAM_TEMP
              RET
;***************************************************************



;***************************** LIGHTS_ON ***********************
;GREEN LED LIGHTS=ON                                            *
;***************************************************************
LIGHTS_ON:
              ORB  RAM_P1,#20H
              STB  RAM_P1,P1
              RET
;***************************************************************



;************************** LIGHTS_OFF *************************
;GREEN LED LIGHTS=OFF                                           *
;***************************************************************
LIGHTS_OFF:   ANDB RAM_P1,#0DFH
              STB  RAM_P1,P1
              RET
;***************************************************************



;***************************** PMTS_ON *************************
;PMTS POWER=ON                                                  *
;***************************************************************
```

```
PMTS_ON:        ORB   RAM_P1,#10H
                STB   RAM_P1,P1
                RET
;***************************************************************


;****************************** PMTS_OFF ************************
;PMTS POWER=OFF                                                 *
;***************************************************************
PMTS_OFF:       ANDB RAM_P1,#0EFH
                STB   RAM_P1,P1
                RET
;***************************************************************


;*************************** SOLENOID_ON ************************
;                                                               *
;***************************************************************
SOLENOID_ON:    ORB   RAM_P2,#40H
                STB   RAM_P2,P2
                RET
;***************************************************************


;************************** SOLENOID_OFF ************************
;                                                               *
;***************************************************************
SOLENOID_OFF:   ANDB RAM_P2,#0BFH
                STB   RAM_P2,P2
                RET
;***************************************************************


;************************* CLR_SCAN_LED *************************
; HSO3=---------------------OUT                                 *
; HSO2=SCANNING LED*--------OUT                                 *
; HSO1=LOADING LED*---------OUT                                 *
; HSO0=UNLOADING LED*-------OUT                                 *
;***************************************************************
CLR_SCAN_LED:   ORB   RAM_HSIO,#04H
                LDB   AL,RAM_HSIO
                CALL  OUT_HSO
                ANDB  RAM_RUNFLG,#0FBH      ;BIT 2=1
                RET
;***************************************************************
```

```
;************************ SET_SCAN_LED ************************
;                                                             *
;*************************************************************
SET_SCAN_LED: ANDB RAM_HSIO,#0FBH
              LDB  AL,RAM_HSIO
              CALL OUT_HSO
              ORB  RAM_RUNFLG,#04H        ;BIT 2=1
              RET
;*************************************************************



;************************ CLR_LOAD_LED ************************
;                                                             *
;*************************************************************
CLR_LOAD_LED: ORB  RAM_HSIO,#02H
              LDB  AL,RAM_HSIO
              CALL OUT_HSO
              ANDB RAM_RUNFLG,#0FDH       ;BIT 2=1
              RET
;*************************************************************



;************************ SET_LOAD_LED ************************
;                                                             *
;*************************************************************
SET_LOAD_LED: ANDB RAM_HSIO,#0FDH
              LDB  AL,RAM_HSIO
              CALL OUT_HSO
              ORB  RAM_RUNFLG,#02H        ;BIT 2=1
              RET
;*************************************************************



;************************ CLR_UNLOAD_LED ************************
;                                                             *
;*************************************************************
CLR_UNLOAD_LED:
              ORB  RAM_HSIO,#01H
              LDB  AL,RAM_HSIO
              CALL OUT_HSO
              ANDB RAM_RUNFLG,#0F7H       ;BIT 2=1
              RET
;*************************************************************



;************************ SET_UNLOAD_LED ************************
;                                                             *
;*************************************************************
SET_UNLOAD_LED:
```

```
                ANDB RAM_HSIO,#0FEH
                LDB  AL,RAM_HSIO
                CALL OUT_HSO
                ORB  RAM_RUNFLG,#08H        ;BIT 2=1
                RET
;***************************************************************



;************************* FLASH_LEDS **************************
;FROM HOST.SRC                                                  *
;***************************************************************
FLASH_LEDS:
                ANDB RAM_HSIO,#0F0H
                LDB  AL,RAM_HSIO
                CALL OUT_HSO
                LDB  AL,#04H                ;# AUDIO BEEPS
ERR_LOOP:       CALL AUDIO_ON
                CALL DLY_500ms
                CALL AUDIO_OFF
                DJNZ AL,ERR_LOOP
                CALL SET_SCAN_LED
                RET
;***************************************************************



;************************* AUDIO_BEEP **************************
;                                                               *
;***************************************************************
AUDIO_BEEP:     ANDB RAM_P2,#7FH
                STB  RAM_P2,P2
;               CALL DLY_1sec
                CALL DLY_500ms
                ORB  RAM_P2,#80H
                STB  RAM_P2,P2
                RET
;***************************************************************



;************************ USER_DSPLY_CHKOUT ********************
;                                                               *
;***************************************************************
USER_DSPLY_CHKOUT:
                CALL CLR_SCRN
                LD   SI,#MSG_FPCHKOUT
                CALL SND_NULL
                CALL AUDIO_BEEP

                CALL SET_LOAD_LED
                CALL DLY_1sec
                CALL DLY_1sec
                CALL CLR_LOAD_LED
```

```
                CALL SET_SCAN_LED
                CALL DLY_1sec
                CALL DLY_1sec
                CALL CLR_SCAN_LED

                CALL SET_UNLOAD_LED
                CALL DLY_1sec
                CALL DLY_1sec
                CALL CLR_UNLOAD_LED
                BR   MENU_0

MSG_FPCHKOUT: DCB  'ACTIVATING LEDS & BEEPER',00H
;*************************************************************



;************************ DSPLY_SWITCHES ************************
; P0.7=FP SWCHS                        (EXT INT)                 *
; P0.6=LOAD SWCH                                                 *
; P0.5=SCAN SWCH                                                 *
; P0.4=UNLOAD SWCH                                               *
; P0.3=DOOR SWCH                                                 *
; P0.2=                                                          *
; P0.1=TEMPERATURE                                               *
; P0.0=DC MTR CURRENT                                            *
;*************************************************************
DSPLY_SWITCHES:
                CALL CLR_SCRN
                LD   SI,#MSG_SWCHS
                CALL SND_NULL
AGAIN_P0:       LD   SI,#MSG_P0
                CALL SND_NULL
                LDB  AL,P0
                CALL SND_BITS
                CALL DLY_1sec
                BR   AGAIN_P0

MSG_SWCHS:      DCB  'FP SWITCHES',0AH,0DH
                DCB  '  BIT 6=LOAD',0AH,0DH
                DCB  '  BIT 5=SCAN',0AH,0DH
                DCB  '  BIT 4=UNLOAD',0AH,0DH
                DCB  '  BIT 3=DOOR',00H
MSG_P0:         DCB  0AH,0DH,'PORT P0(7-0)=',00H
;*************************************************************
END



--
*************************************************************************
******
-- Ints.src
--
*************************************************************************
******
```

```
$TITLE('INTERRUPT HANDLERS')
$PAGELENGTH(999)

;*************************** INTS ******************************
; THIS FILE IS THE "INTERRUPT HANDLER"                          *
;    ALL INTERRUPT TYPES WILL VECTOR HERE-EXCEPT "GET_KYBD"     *
;                                                               *
; INTERRUPT STRUCTURE                                           *
;    -AFTER INTERRUPT NO FURTHUR INTERRUPT CALLS UNTIL AFTER    *
;     FIRST INSTRUCTION OF SERVICE ROUTINE                      *
;    -PUSHA INSTRUCTION SAVES PSW,INT_MASK1 & WSR & THEN CLRS   *
;     PSW,INT_MASK, & INT_MASK1(EI BIT=0) BLOCKING FURTHUR INTS*
;    -POPA RESTORES THAT OF PUSHA & THE LAST INSTRUCTION "RET"  *
;     WILL EXECUTE BEFORE A PENDING INTERRUPT                   *
;***************************************************************
;                        --SUBROUTINES--                        *
; INT_EXT          (INTERRUPT HANDLER P0.7)                     *
; INT_EXT1         (INTERRUPT HANDLER P2.2)                     *
; INT_T1           (INTERRUPT HANDLER TIMER1)                   *
; INT_T2           (INTERRUPT HANDLER TIMER2-P2.3)              *
; INT_HSI0         (INTERRUPT HANDLER HSI.0 PIN)                *
; INT_SWTIMER      (INTERRUPT HANDLER CAM 4 SOFTWARE TIMERS)    *
; INT_HSOEVENT     (INTERRUPT HANDLER HSO EVENTS)               *
; INT_ADEOC        (INTERRUPT HANDLER A/D END-OF-CONVERSION)    *
; TRAP             (EXTRANEOUS INTERRUPTS)                      *
; EN_EXTINT        (ENABLE INTERRUPT-CLRS PENDING ALSO)         *
; EN_EXTINT1                              "                     *
; EN_T2INT                                "                     *
; EN_T1INT                                "                     *
; EN_HSI0INT                              "                     *
; EN_SWTIMER                              "                     *
; EN_HSOINT                               "                     *
; EN_ADEOC                                "                     *
; DIS_EXTINT       (DISABLE INTERRUPT)                          *
; DIS_EXTINT1             "                                     *
; DIS_T2INT               "                                     *
; DIS_T1INT               "                                     *
; DIS_HSI0INT             "                                     *
; DIS_SWTIMER             "                                     *
; DIS_HSOINT              "                                     *
; DIS_ADEOC               "                                     *
;***************************************************************
              PUBLIC  INT_EXT,INT_EXT1,INT_T2,EN_EXTINT,DIS_T1INT
              PUBLIC  EN_EXTINT1,TRAP,EN_T2INT,EN_T1INT,INT_HSI0
              PUBLIC  INT_T1,DIS_EXTINT,DIS_EXTINT1,DIS_T2INT
              PUBLIC  EN_HSI0INT,DIS_HSI0INT,INT_SWTIMER,EN_SWTIMER
              PUBLIC  DIS_SWTIMER,INT_HSOEVENT,EN_HSOINT,DIS_HSOINT
              PUBLIC  INT_ADEOC,EN_ADEOC,DIS_ADEOC

              EXTRN   SND_NULL,CLR_SCRN,VIDEO_OUT,FP_SWCH_INT
              EXTRN   DLY_100ms

$INCLUDE(8086_REGS.INC)
$INCLUDE(SFR.INC)
              EXTRN   HWin0,HWin15

CSEG
```

```
;*****************************************************************



;************************** INT_EXT ****************************
; INTERRUPT ON P0.7     (ALSO "WAKEUP"-FROM IDLPD #2)            *
; SELECT THIS PIN IN I0C1                                        *
; + EDGE (NEEDS MINIMUM WIDTH (>2 STATE TIMES--125ns or 167ns)  *
; APPLICATION: FP SWITCH                                         *
;*****************************************************************
INT_EXT:        CALL FP_SWCH_INT
                RET
;*****************************************************************



;NOT USED
;************************** INT_EXT1 ***************************
; INTERRUPT ON PIN P2.2                                          *
; + EDGE (NEEDS MINIMUM WIDTH (>2 STATE TIMES--125ns or 167ns)  *
; APPLICATION:                                                   *
;*****************************************************************
INT_EXT1:       SETC
                RET
;*****************************************************************



;NOT USED
;************************** INT_T1 *****************************
; TIMER 1 OVERFLOW INTERRUPT HANDLER  (FFFF-0000)                *
;    MUST BE LOADED IN WINDOW 15                                 *
;         LDB  WSR,#0FH                                          *
;         LD   TIMER1,#....                                      *
;         LDB  WSR,#00H                                          *
;    MUST BE READ IN WINDOW 0                                    *
; INTERNAL UP CNT TIMER (1.33 usecs AT 12 MHz)                   *
;                       (1.00 usecs AT 16 MHz)                   *
; APPLICATION:                                                   *
;*****************************************************************
INT_T1:         PUSHA
                POPA
                RET
;*****************************************************************



;NOT USED
;************************** INT_T2 *****************************
```

```
; TIMER 2 OVERFLOW INTERRUPT HANDLER  (FFFF-0000)                *
; 2 POSSIBLE MODES-CNTS BOTH EDGES                               *
;   -UP/DOWN CNTR    (IOC2.1=1,  P2.6=1 for UP)                  *
;   -UP CNTR ONLY    (IOC2.1=0)                                  *
; APPLICATION:ENCODER CNTS OVERFLOW                              *
;****************************************************************
INT_T2:       PUSHA
              POPA
              RET
;****************************************************************



;NOT USED
;************************** INT_HSI0 **************************
; INTERRUPT ON HSI.0 PIN                                         *
; + EDGE (NEEDS MINIMUM WIDTH (>2 STATE TIMES--125ns or 167ns)  *
; APPLICATION:                                                   *
;****************************************************************
INT_HSI0:     PUSHA
              POPA
              RET
;****************************************************************



;NOT USED
;************************** INT_SWTIMER ***********************
; INTERRUPT-CAM 4 SOFTWARE TIMERS                                *
; CHECK ISO1 TO SEE WHICH OF 4 TIMERS OVERFLOWED                 *
;    BIT #3=SWTF3                                                *
;    BIT #2=SWTF2                                                *
;    BIT #1=SWTF1                                                *
;    BIT #0=SWTF0                                                *
; APPLICATION:                                                   *
;****************************************************************
INT_SWTIMER:  PUSHA
              LDB  AL,IOS1                ;RD IOS1
              JBS  AL,0,INT_SWT0
              JBS  AL,1,INT_SWT1
              JBS  AL,2,INT_SWT2
INT_SWT3:     NOP
              POPA
              RET
INT_SWT2:     NOP
              POPA
              RET
INT_SWT1:     NOP
              POPA
              RET
INT_SWT0:     NOP
              POPA
              RET
;****************************************************************
```

```
;************************ INT_HSOEVENT ************************
; INTERRUPT-HSO CAM EVENTS                                      *
; CHECK ISO2 TO SEE WHICH OF 6 HSO CAM EVENTS OCCURRED          *
;    BIT #5=HSO5 EVENT                                          *
;    BIT #4=HSO4 EVENT                                          *
;    BIT #3=HSO3 EVENT                                          *
;    BIT #2=HSO2 EVENT                                          *
;    BIT #1=HSO1 EVENT                                          *
;    BIT #0=HSO0 EVENT                                          *
; APPLICATION: ONLY HSO.0 YIELDS AN EVENT                       *
;*************************************************************
INT_HSOEVENT: PUSHA                         ;SAVE WSR
              LDB  WSR,#HWin15
              CLR  TIMER1                   ;SET TIMER1=0000
              POPA                          ;RESTORE PREVIOUS WSR
              RET

;USE THE FOLLOWING FOR MULTIPLE HSO EVENTS
              PUSHA
              LDB  AL,IOS2                  ;RD IOS2
              JBS  AL,0,INT_HSO0
              JBS  AL,1,INT_HSO1
              JBS  AL,2,INT_HSO2
INT_HSO3:     NOP
              POPA
              RET
INT_HSO2:     NOP
              POPA
              RET
INT_HSO1:     NOP
              POPA
              RET
INT_HSO0:     LDB  WSR,#HWin15
              CLR  TIMER1                   ;SET TIMER1=0000
              POPA                          ;RESTORES WSR
              RET
;*************************************************************



;************************** INT_ADEOC *************************
; A/D END-OF-CONVERSION                                         *
;*************************************************************
INT_ADEOC:    SETC                          ;CY=1
              RET
;*************************************************************



;***************************** TRAP ***************************
; CATCHES ANY EXTRANEOUS INTERRUPTS                             *
```

```
;*************************************************************
TRAP:          CALL CLR_SCRN
               LD   SI,#MSG_INTERR
               CALL SND_NULL
               CLRC
SELF:          BNC  SELF                    ;EXIT WITH RST/CNTL C

MSG_INTERR:    DCB  0AH,0AH,0DH,'ERROR-EXTRANEOUS INTERRUPT ',00H
;*************************************************************




;*************************** EN_EXTINT ************************
; ENABLE INT_EXT INTERRUPT                                    *
;*************************************************************
EN_EXTINT:     ANDB IPEND,#7FH              ;CLR PENDING FIRST
               ORB  IMASK,#80H
               RET
;*************************************************************




;*************************** EN_EXTINT1 ***********************
; ENABLE INT_EXT1 INTERRUPT                                   *
;*************************************************************
EN_EXTINT1:    ANDB IPEND1,#0DFH            ;CLR PENDING FIRST
               ORB  IMASK1,#20H
               RET
;*************************************************************




;*************************** EN_T2INT *************************
; ENABLE TIMER 2 OVERFLOW INTERRUPT                           *
;*************************************************************
EN_T2INT:      ANDB IPEND1,#0EFH            ;CLR PENDING FIRST
               ORB  IMASK1,#10H
               RET
;*************************************************************




;*************************** EN_T1INT *************************
; ENABLE TIMER 1 OVERFLOW INTERRUPT                           *
;*************************************************************
EN_T1INT:      ANDB IPEND,#0FEH             ;CLR PENDING
               ORB  IMASK,#01H
               RET
;*************************************************************
```

```
;***************************** EN_HSI0INT ***********************
; ENABLE HSI.0 PIN INTERRUPT                                     *
;****************************************************************
EN_HSI0INT:   ANDB IPEND,#0EFH          ;CLR PENDING
              ORB  IMASK,#10H
              RET
;****************************************************************



;************************** EN_SWTIMER **************************
; ENABLE CAM SOFTWARE TIMER INTERRUPT                            *
;****************************************************************
EN_SWTIMER:   ANDB IPEND,#0DFH          ;CLR PENDING
              ORB  IMASK,#20H
              RET
;****************************************************************



;************************** EN_HSOINT ***************************
; ENABLE CAM HSO EVENT INTERRUPT                                 *
;****************************************************************
EN_HSOINT:    ANDB IPEND,#0F7H
              ORB  IMASK,#08H
              RET
;****************************************************************



;***************************** EN_ADEOC *************************
; ENABLE A/D END-OF-CONVERSION INTERRUPT                         *
;****************************************************************
EN_ADEOC:     ANDB IPEND,#0FDH          ;CLR PENDING
              ORB  IMASK,#02H
              RET
;****************************************************************



;***************************** DIS_EXTINT ***********************
; DISABLE INT_EXT INTERRUPT                                      *
;****************************************************************
DIS_EXTINT:   ANDB IMASK,#7FH
              RET
```

```
;*************************************************************

;*************************** DIS_EXTINT1 ***********************
; DISABLE ENT_EXT1 INTERRUPT                                   *
;*************************************************************
DIS_EXTINT1:  ANDB IMASK1,#0DFH
              RET
;*************************************************************

;**************************** DIS_T2INT ************************
; DISABLE TIMER 2 OVERFLOW INTERRUPT                           *
;*************************************************************
DIS_T2INT:    ANDB IMASK1,#0EFH
              RET
;*************************************************************

;**************************** DIS_T1INT ************************
; DISABLE TIMER 1 OVERFLOW INTERRUPT                           *
;*************************************************************
DIS_T1INT:    ANDB IMASK,#0FEH
              RET
;*************************************************************

;************************** DIS_HSI0INT **********************
; DISABLE HSI.0 PIN INTERRUPT                                  *
;*************************************************************
DIS_HSI0INT:  ANDB IMASK,#0EFH
              RET
;*************************************************************

;************************** DIS_SWTIMER **********************
; DISABLE CAM SOFTWARE TIMER INTERRUPT                         *
;*************************************************************
DIS_SWTIMER:  ANDB IMASK,#0DFH
              RET
;*************************************************************
```

```
;************************* DIS_HSOINT ***************************
; DISABLE CAM HSO EVENT INTERRUPT                                 *
;*************************************************************
DIS_HSOINT:   ANDB IMASK,#0F7H
              RET
;*************************************************************



;************************* DIS_ADEOC ****************************
; DISABLE A/D END-OF-CONVERSION INTERRUPT                         *
;*************************************************************
DIS_ADEOC:    ANDB IMASK,#0FDH
              RET
;*************************************************************
END


--
*****************************************************************************
******
-- Hsio.src
--
*****************************************************************************
******
$TITLE('HSIO')
$PAGELENGTH(999)

;***************************** HSIO *********************************
; THIS MODULE REQUIRED FOR "DEBUG"                                  *
; HIGH SPEED INPUTS                                                 *
;    HSI0-3                                                         *
; HIGH SPEED OUTPUTS                                                *
;    HSO0-3 ARE NORMAL OUTPUTS                                      *
;    HSO4-5 CAN BE ENABLED AS OUTPUTS  (REPLACES HSI2 & HSI3)       *
;*************************************************************
;                          --SUBROUTINES--                          *
; RD_HSI       LOW NIBBLE AL=HSI0-3                                 *
; RD_HSO       LOW NIBBLE AL=HSO0-3                                 *
; OUT_HSO      LOW NIBBLE AL=HSO0-3                                 *
;                                                                   *
; SET_HSO0     HSO0=1                                               *
; CLR_HSO0     HSO0=0                                               *
; SET_HSO1     HSO1=1                                               *
; CLR_HSO1     HSO1=0                                               *
; SET_HSO2     HSO2=1                                               *
; CLR_HSO2     HSO2=0                                               *
; SET_HSO3     HSO3=1                                               *
; CLR_HSO3     HSO3=0                                               *
; SET_HSO4     HSO4=1                                               *
; CLR_HSO4     HSO4=0                                               *
; SET_HSO5     HSO5=1                                               *
```

```
; CLR_HSO5     HSO5=0                                                  *
;                                                                     *
; EN_HSO4      ENABLE HSO4 AS AN OUTPUT  (REPLACES HSI2)              *
; EN_HSO5      ENABLE HSO5 AS AN OUTPUT  (    "    HSI3)              *
;*********************************************************************
;FOLLOWING FOR "CONDITIONAL ASSEMBLY"---USES IF & ENDIF STATEMENTS
YES                 EQU  0FFH
NO                  EQU  00H

;SUBROUTINE "INCLUDE LIST"
xRD_HSI             EQU  YES              ;USED IN DEBUG
xRD_HSO             EQU  YES              ;      "
xOUT_HSO            EQU  YES              ;      "
xSET_HSO0           EQU  YES
xCLR_HSO0           EQU  YES
xSET_HSO1           EQU  YES
xCLR_HSO1           EQU  YES
xSET_HSO2           EQU  YES
xCLR_HSO2           EQU  YES
xSET_HSO3           EQU  NO
xCLR_HSO3           EQU  NO
xSET_HSO4           EQU  NO
xCLR_HSO4           EQU  NO
xSET_HSO5           EQU  NO
xCLR_HSO5           EQU  NO
xEN_HSO4            EQU  NO
xEN_HSO5            EQU  NO

$INCLUDE(8086_REGS.INC)
$INCLUDE(SFR.INC)
                    EXTRN   HWin0,HWin15

CSEG
;*********************************************************************



IF xRD_HSI
;*************************** RD_HSI ******************************
; RD HSI FROM HSI_STATUS                                             *
; HSI_STATUS                                                         *
;      BIT #7=HSI.3 PIN                                              *
;      BIT #6=                                                       *
;      BIT #5=HSI.2 PIN                                              *
;      BIT #4=                                                       *
;      BIT #3=HSI.1 PIN                                              *
;      BIT #2=                                                       *
;      BIT #1=HSI.0 PIN                                              *
;      BIT #0=                                                       *
; EXIT:AL=0,HSI  (DATA IN LOW NIBBLE)                                *
;*********************************************************************
                    PUBLIC  RD_HSI

RD_HSI:             LDB  AH,HSI_STATUS          ;RD HSI_STATUS
                    CLRB AL                     ;AL=00
                    JBC  AH,7,NXT_4
```

```
                ORB   AL,#08H
NXT_4:          JBC   AH,5,NXT_2
                ORB   AL,#04H
NXT_2:          JBC   AH,3,NXT_1
                ORB   AL,#02H
NXT_1:          JBC   AH,1,NXT_0
                ORB   AL,#01H
NXT_0:          RET                             ;AL=RESULT IN LOW NIBBLE
;***************************************************************
ENDIF



IF xRD_HSO
;************************** RD_HSO ****************************
; RD HSO.0-.3 OUTPUT LINES                                     *
;                                                              *
; EXIT with:AL=0,HSO PORT VALUE                                *
;***************************************************************
                PUBLIC  RD_HSO

RD_HSO:         LDB   AL,IOS0                   ;RD IOS0 REGISTER
                ANDB  AL,#0FH
                RET
;***************************************************************
ENDIF



IF xOUT_HSO
;************************** OUT_HSO ***************************
; SET HSO.0-.3 OUTPUT LINES AS A NIBBLE WRT                    *
;                                                              *
; ENTER with:AL=0,HSO PORT VALUE                               *
;***************************************************************
                PUBLIC  OUT_HSO

OUT_HSO:        PUSH AX
                LDB   AH,IOS0                   ;RD IOS0 REGISTER
                ANDB  AH,#0F0H
                ORB   AL,AH
                LDB   WSR,#HWin15               ;SELECT HWin=15
                STB   AL,IOS0                   ;WRT TO IOS0
                LDB   WSR,#HWin0                ;SELECT HWin=0
                POP   AX
                RET
;***************************************************************
ENDIF



IF xSET_HSO0
```

```
;************************ SET_HSO0 ******************************
; SET HSO0                                                      *
;***************************************************************
                PUBLIC  SET_HSO0

SET_HSO0:       PUSH AX
                LDB  AL,IOS0                ;RD IOSO REGISTER
                LDB  WSR,#HWin15            ;SELECT HWin=15
                ORB  AL,#01H                ;SET HSO0
                STB  AL,IOS0                ;WRT TO IOS0
                LDB  WSR,#HWin0             ;SELECT HWin=0
                POP  AX
                RET
;***************************************************************
ENDIF



IF xCLR_HSO0
;************************ CLR_HSO0 ******************************
; CLR HSO0                                                      *
;***************************************************************
                PUBLIC  CLR_HSO0

CLR_HSO0:       PUSH AX
                LDB  AL,IOS0                ;RD IOSO REGISTER
                LDB  WSR,#HWin15            ;SELECT HWin=15
                ANDB AL,#0FEH               ;CLR HSO0
                STB  AL,IOS0                ;WRT TO IOS0
                LDB  WSR,#HWin0             ;SELECT HWin=0
                POP  AX
                RET
;***************************************************************
ENDIF



IF xSET_HSO1
;************************ SET_HSO1 ******************************
; SET HSO1                                                      *
;***************************************************************
                PUBLIC  SET_HSO1

SET_HSO1:       PUSH AX
                LDB  AL,IOS0                ;RD IOSO REGISTER
                LDB  WSR,#HWin15            ;SELECT HWin=15
                ORB  AL,#02H                ;SET HSO1
                STB  AL,IOS0                ;WRT TO IOS0
                LDB  WSR,#HWin0             ;SELECT HWin=0
                POP  AX
                RET
;***************************************************************
ENDIF
```

```
IF xCLR_HSO1
;************************ CLR_HSO1 ******************************
; CLR HSO1                                                      *
;***************************************************************
              PUBLIC  CLR_HSO1

CLR_HSO1:     PUSH AX
              LDB  AL,IOS0                ;RD IOSO REGISTER
              LDB  WSR,#HWin15            ;SELECT HWin=15
              ANDB AL,#0FDH               ;CLR HSO1
              STB  AL,IOS0                ;WRT TO IOS0
              LDB  WSR,#HWin0             ;SELECT HWin=0
              POP  AX
              RET
;***************************************************************
ENDIF



IF xSET_HSO2
;************************ SET_HSO2 ******************************
; SET HSO2                                                      *
;***************************************************************
              PUBLIC  SET_HSO2

SET_HSO2:     PUSH AX
              LDB  AL,IOS0                ;RD IOSO REGISTER
              LDB  WSR,#HWin15            ;SELECT HWin=15
              ORB  AL,#04H                ;SET HSO2
              STB  AL,IOS0                ;WRT TO IOS0
              LDB  WSR,#HWin0             ;SELECT HWin=0
              POP  AX
              RET
;***************************************************************
ENDIF



IF xCLR_HSO2
;************************ CLR_HSO2 ******************************
; CLR HSO2                                                      *
;***************************************************************
              PUBLIC  CLR_HSO2

CLR_HSO2:     PUSH AX
              LDB  AL,IOS0                ;RD IOSO REGISTER
              LDB  WSR,#HWin15            ;SELECT HWin=15
              ANDB AL,#0FBH               ;CLR HSO2
              STB  AL,IOS0                ;WRT TO IOS0
              LDB  WSR,#HWin0             ;SELECT HWin=0
              POP  AX
              RET
;***************************************************************
```

```
ENDIF



IF xSET_HSO3
;************************ SET_HSO3 ****************************
; SET HSO3                                                     *
;*************************************************************
              PUBLIC  SET_HSO3

SET_HSO3:     PUSH AX
              LDB  AL,IOS0              ;RD IOSO REGISTER
              LDB  WSR,#HWin15          ;SELECT HWin=15
              ORB  AL,#08H              ;SET HSO3
              STB  AL,IOS0              ;WRT TO IOS0
              LDB  WSR,#HWin0           ;SELECT HWin=0
              POP  AX
              RET
;*************************************************************
ENDIF



IF xCLR_HSO3
;************************ CLR_HSO3 ****************************
; CLR HSO3                                                     *
;*************************************************************
              PUBLIC  CLR_HSO3

CLR_HSO3:     PUSH AX
              LDB  AL,IOS0              ;RD IOSO REGISTER
              LDB  WSR,#HWin15          ;SELECT HWin=15
              ANDB AL,#0F7H             ;CLR HSO3
              STB  AL,IOS0              ;WRT TO IOS0
              LDB  WSR,#HWin0           ;SELECT HWin=0
              POP  AX
              RET
;*************************************************************
ENDIF



IF xSET_HSO4
;************************ SET_HSO4 ****************************
; SET HSO4                                                     *
;*************************************************************
              PUBLIC  SET_HSO4

SET_HSO4:     PUSH AX
              LDB  AL,IOS0              ;RD IOSO REGISTER
              LDB  WSR,#HWin15          ;SELECT HWin=15
              ORB  AL,#10H              ;SET HSO4
              STB  AL,IOS0              ;WRT TO IOS0
              LDB  WSR,#HWin0           ;SELECT HWin=0
```

```
                POP  AX
                RET
;*****************************************************************
ENDIF



IF xCLR_HSO4
;************************ CLR_HSO4 ******************************
; CLR HSO4                                                       *
;*****************************************************************
                PUBLIC  CLR_HSO4

CLR_HSO4:       PUSH AX
                LDB  AL,IOS0                ;RD IOSO REGISTER
                LDB  WSR,#HWin15            ;SELECT HWin=15
                ANDB AL,#0EFH               ;CLR HSO4
                STB  AL,IOS0                ;WRT TO IOS0
                LDB  WSR,#HWin0             ;SELECT HWin=0
                POP  AX
                RET
;*****************************************************************
ENDIF



IF xSET_HSO5
;************************ SET_HSO5 ******************************
; SET HSO5                                                       *
;*****************************************************************
                PUBLIC  SET_HSO5

SET_HSO5:       PUSH AX
                LDB  AL,IOS0                ;RD IOSO REGISTER
                LDB  WSR,#HWin15            ;SELECT HWin=15
                ORB  AL,#20H                ;SET HSO5
                STB  AL,IOS0                ;WRT TO IOS0
                LDB  WSR,#HWin0             ;SELECT HWin=0
                POP  AX
                RET
;*****************************************************************
ENDIF



IF xCLR_HSO5
;************************ CLR_HSO5 ******************************
; CLR HSO5                                                       *
;*****************************************************************
                PUBLIC  CLR_HSO5

CLR_HSO5:       PUSH AX
                LDB  AL,IOS0                ;RD IOSO REGISTER
                LDB  WSR,#HWin15            ;SELECT HWin=15
```

```
                ANDB AL,#0DFH               ;CLR HSO5
                STB  AL,IOS0                ;WRT TO IOS0
                LDB  WSR,#HWin0             ;SELECT HWin=0
                POP  AX
                RET
;*****************************************************************
ENDIF


IF xEN_HSO4
;************************** EN_HSO4 ****************************
; ENABLE HSO4 AS AN OUTPUT-REPLACES HSI2                        *
;*****************************************************************
                PUBLIC  EN_HSO4

EN_HSO4:        PUSH AX
                LDB  WSR,#HWin15            ;SET HWin=15
                LDB  AL,IOC1                ;READ IOC1
                LDB  WSR,#HWin0             ;SET HWin=0
                ORB  AL,#10H                ;ENABLE HSO4 AS OUTPUT
                STB  AL,IOC1                ;WRT TO IOC1
                POP  AX
                RET
;*****************************************************************
ENDIF


IF xEN_HSO5
;************************** EN_HSO5 ****************************
; ENABLE HSO5 AS AN OUTPUT-REPLACES HSI3                        *
;*****************************************************************
                PUBLIC  EN_HSO5

EN_HSO5:        PUSH AX
                LDB  WSR,#HWin15            ;SET HWin=15
                LDB  AL,IOC1                ;READ IOC1
                LDB  WSR,#HWin0             ;SET HWin=0
                ORB  AL,#40H                ;ENABLE HSO5 AS OUTPUT
                STB  AL,IOC1                ;WRT TO IOC1
                POP  AX
                RET
;*****************************************************************
ENDIF
END


--
*****************************************************************************
******
-- Host.src
--
*****************************************************************************
******
$TITLE('HOST')
```

```
$PAGELENGTH(999)

;************************* HOST MODULE ************************
; JR Skorpik          (08/99)                                  *
;**************************************************************
; OSL uP & HOST INTERFACE-9600 & 8N1                           *
;                                                              *
;uP TO HOST PROTOCOL                                           *
;   uP SNDS A RQST BYTE                                        *
;         RQST_SNDCNTS=80H                                     *
;   uP WAITS FOR "ACK" FROM HOST                               *
;      -TIMES OUT & RESENDS RQST, X NUMBER OF RETRYS           *
;   uP RCVS "ACK" & SNDS                                       *
;         LOCATION POSITION CNT                                *
;         TEMPERATURE                                          *
;         40 BYTE DATA FIELD (10 CNTRS, 4 BYTES EACH)          *
;         CHKSUM                                               *
;   uP WAITS "ACK"--DOES 1 RESEND OF CNTS ON "NAK"             *
;**************************************************************
; SND_DATA_HOST                                                *
; SND_CMND_HOST                                                *
; WAIT_ACK                                                     *
;**************************************************************
              PUBLIC  SND_DATA_HOST

              EXTRN   VIDEO_OUT,WAIT_KYBDTOUT,AUDIO_BEEP,DSPLY_CNTS
              EXTRN   FLASH_LEDS

$INCLUDE(8086_REGS.INC)
$INCLUDE(SFR.INC)
              EXTRN   RAM_HOST,RAM_RUNFLG,RQST_SNDDATA,RAM_DATA,CODE_ACK
              EXTRN   No_HOST_TRYS,No_BYTES_DATA,RAM_NOHOST

CSEG
;**************************************************************



;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;                                                              ;
;                    uP-to-HOST PRIMITIVES                     ;
;                                                              ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;



;************************ SND_DATA_CNTS ***********************
;42 BYTES WITH CHKSUM                                          *
;PROTOCOL                                                      *
;  SND CMND                                                    *
;  WAIT ACK-1 RESEND IF TIMEOUT                                *
;  SND LOCATION CNT                                            *
;  SND TEMPERATAURE                                            *
;  SND 40 BYTES (10 EA 4 BYTE CNTRS)                           *
;  SND CHKSUM                                                  *
```

```
;  WAIT ACK-1 RESEND IF NAK                                          *
;                                                                    *
;ENTER with: RAM_LOCATION @ 100H                                     *
;          : RAM_TEMP @ 101H                                         *
;          : RAM_CNTS @ 102H                                         *
;EXIT  with: CY=1 OK, NO ERROR                                       *
;EXIT  with: CY=1 OK, NO ERROR                                       *
;          : RAM_NOHOST=01 FOR ERROR                                 *
;*********************************************************************
SND_DATA_HOST:
              JBS  RAM_RUNFLG,0,HOST_MODE
              CALL DSPLY_CNTS
              RET

HOST_MODE:    CMPB RAM_NOHOST,#00H        ;CHK IF HOST PRESENT
              BE   HOST_ALIVE
              RET

HOST_ALIVE:   LDB  AL,#RQST_SNDDATA
              CALL SND_CMND_HOST          ;FROM AL---HAS RETRYS (1.8 SEC
TOUT)
              JC   DATA_OUT               ;CY=1, OK
              RET

              CLRB CH                     ;FLG
DATA_OUT:     LDB  CL,#No_BYTES_DATA
              LD   SI,#RAM_DATA
              CLRB AH                     ;CLR CHKSUM
NXT_BYTE:     LDB  AL,[SI]+
              ADDB AH,AL                  ;AH=CHKSUM
              CALL VIDEO_OUT
              DJNZ CL,NXT_BYTE
              INCB CH                     ;# TRYS

              LDB  AL,AH                  ;SND CHKSUM
              CALL VIDEO_OUT
              CALL WAIT_ACK               ;CY=1 OK
              JC   CNTS_OK
              CALL AUDIO_BEEP             ;HOST ERROR INDICATOR
              CALL AUDIO_BEEP
              CMPB CH,#01H
              BE   DATA_OUT
              LDB  RAM_NOHOST,#01H        ;SET FLG
              CLRC
CNTS_OK:      RET
;*********************************************************************




;************************ SND_CMND_HOST **********************
; EXIT with: CY=1 FOR VALID "ACK"                                    *
;          : CY=0, TIMEOUT OR INVALID ACK                            *
;*********************************************************************
SND_CMND_HOST:
              LD   BL,#No_HOST_TRYS+1
RETRY_HOST:   PUSH AX                     ;RQST CMND
```

```
                CALL VIDEO_OUT              ;FROM AL
                CALL WAIT_ACK               ;WAIT ACK FROM HOST
                POP  AX
                JC   HOST_OK
                DJNZ BL,RETRY_HOST
                CALL FLASH_LEDS             ;VISUAL ERROR INDICATOR
                CLRC
                RET

HOST_OK:        SETC
                RET
;*****************************************************************



;************************** WAIT_ACK ****************************
; WAIT FOR "ACK" CODE BACK FROM HOST                             *
; EXIT with: CY=1 FOR VALID "ACK"                                *
;          : CY=0, TIMEOUT OR INVALID ACK                        *
;*****************************************************************
WAIT_ACK:       EI
                CALL WAIT_KYBDTOUT          ;WAIT WITH TIMEOUT
                DI                          ;FP SWITCHES
                JBS  RAM_RUNFLG,7,TIMEOUT   ;MSB=1 FOR TIMEOUT
                CMPB AL,#CODE_ACK
                BE   ACK_YES
;THIS SHOULD NOW BE A NAK
                CLRC                        ;CY=0
                RET

TIMEOUT:        CLRB AL
                CLRC                        ;CY=0
                RET

ACK_YES:        SETC                        ;CY=1
                RET
;*****************************************************************
END



--
*************************************************************************
******
-- Hardware.src
--
*************************************************************************
******
$TITLE('FPGA')
$PAGELENGTH(999)

;************************* FPGA MODULE **************************
;ALTERA FPGA-10 EACH 32-BIT CNTRS                                *
;*****************************************************************
;DSPLY_FPGA_CNTS                                                 *
;DSPLY_CNTS                                                      *
```

```
;RD_FPGA_CNTS                                                    *
;CLR_FPGA_CNTS                                                   *
;EN_FPGA_CNTS                                                    *
;HOLD_FPGA_CNTS                                                  *
;FILL_DUMMY_CNTS                                                 *
;*****************************************************************
;RAM_CNTS+00=CH #0        (FPGA ADDRS=0000)                      *
;RAM_CNTS+04=CH #1                       2                       *
;RAM_CNTS+08=CH #2                       4                       *
;RAM_CNTS+0C=CH #3                       6                       *
;RAM_CNTS+10=CH #4                       8                       *
;RAM_CNTS+14=CH #5                       A                       *
;RAM_CNTS+18=CH #6                       C                       *
;RAM_CNTS+1C=CH #7                       E                       *
;RAM_CNTS+20=CH #8                      10                       *
;RAM_CNTS+24=CH #9                      12                       *
;*****************************************************************
              PUBLIC  DSPLY_FPGA_CNTS,RD_FPGA_CNTS,CLR_FPGA_CNTS
              PUBLIC  EN_FPGA_CNTS,HOLD_FPGA_CNTS,DSPLY_CNTS
              PUBLIC  FILL_DUMMY_CNTS

              EXTRN   CLR_SCRN,SND_NULL,SND_CR_LF,SND_SP,SND_BNBCD2
              EXTRN   SND_EQUAL,SND_DWORD_MEM,DLY_1sec,STOP_START
              EXTRN   SND_BNBCD3,SND_SLASH,SP_LOOP,SND_BYTE

$INCLUDE(8086_REGS.INC)
$INCLUDE(SFR.INC)
              EXTRN
RAMx_FPGA,RAM_CNTS,No_CHS,RAM_PTSCNTR,RAM_NoSCANSTEPS
              EXTRN   RAM_TEMP

CSEG
;*****************************************************************



;************************ DSPLY_FPGA_CNTS ***********************
;FROM MENU                                                       *
;RD CNTS FROM FPGA & DSPLY                                       *
;EXIT WITH CNTRL_C                                               *
;*****************************************************************
DSPLY_FPGA_CNTS:
              CALL CLR_SCRN
RD_AGAIN:     CALL RD_FPGA_CNTS          ;RESULT @ RAM_CNTS
              LD   SI,#MSG_CNTS
              CALL SND_NULL
              CLRB CH
              LDB  CL,#No_CHS            ;# PMT'S
              LD   BP,#RAM_CNTS
NXT_CH:       CALL SND_CR_LF
              CALL SND_SP
              CALL SND_SP
              LDB  AL,CH
              PUSH CX
              CALL SND_BNBCD2            ;CH #
              CALL SND_EQUAL             ;=
```

```
                CALL SND_DWORD_MEM          ;CNTS DOUBLE WORD, INCS BP
                POP  CX
                INCB CH
                DJNZ CL,NXT_CH
                CALL SND_CR_LF
                CALL SND_CR_LF
                CALL DLY_1sec               ;UPDATE TIME
                CALL STOP_START             ;STOP ON KYBD HIT
                BR   RD_AGAIN

MSG_CNTS:       DCB  'FPGA PMT CNTS',00H
;*****************************************************************



;************************** DSPLY_CNTS ************************
;FROM RUN MODE--CHKOUT                                          *
;ENTER with: CNTS @ RAM_CNTS                                    *
;*****************************************************************
DSPLY_CNTS:     CALL SND_CR_LF
                LDB  AL,RAM_PTSCNTR         ;CNTR
                CLRB AH
                CALL SND_BNBCD3             ;SND SCAN STEP #
                CALL SND_SLASH              ;SND /
                LDB  AL,RAM_NoSCANSTEPS     ;SND TOTAL # SCAN STEPS
                SHLB AL,#1
                DECB AL
                CLRB AH
                CALL SND_BNBCD3             ;SND TEMPERATURE
                LDB  CL,#0DH
                CALL SP_LOOP
                LD   SI,#MSG_TEMP
                CALL SND_NULL
                LDB  AL,RAM_TEMP
                CALL SND_BYTE
                CALL SND_CR_LF
                LDB  CL,#No_CHS             ;# PMT'S
                SHRB CL,#1                  ;DIVIDE BY 2
                LDB  CH,CL
                LDB  DL,#02H                ;# LINES
                LD   BP,#RAM_CNTS
NXT_CNTR:       PUSH CX
                PUSH DX
                CALL SND_DWORD_MEM          ;CNTS DOUBLE WORD, INCS BP
                CALL SND_SP
                CALL SND_SP
                POP  DX
                POP  CX
                DJNZ CL,NXT_CNTR
                CALL SND_CR_LF
                LDB  CL,CH
                DJNZ DL,NXT_CNTR
                RET

MSG_TEMP:       DCB  'TEMPERATURE(00-FF)=',00H
;*****************************************************************
```

```
;************************ RD_FPGA_CNTS ************************
;EXIT with: RAM_CNTS+00=CH #0        (FPGA ADDRS=0000)          *
;         : RAM_CNTS+04=CH #1                    2             *
;         : RAM_CNTS+08=CH #2                    4             *
;         : RAM_CNTS+0C=CH #3                    6             *
;         : RAM_CNTS+10=CH #4                    8             *
;         : RAM_CNTS+14=CH #5                    A             *
;         : RAM_CNTS+18=CH #6                    C             *
;         : RAM_CNTS+1C=CH #7                    E             *
;         : RAM_CNTS+20=CH #8                   10             *
;         : RAM_CNTS+24=CH #9                   12             *
;                                                              *
;NOTE:EVEN ADDRESSES=LO WORD                                   *
;     : ODD ADDRESSES=HI WORD                                  *
;**************************************************************
RD_FPGA_CNTS: LD   DT,#RAM_CNTS
              LD   BP,#RAMx_FPGA
              LDB  CL,#No_CHS
NXT_FPGA:     LD   AX,[BP]              ;CAN'T USE [BP]+
              INC  BP
              ST   AX,[DT]+
              LD   AX,[BP]              ;CAN'T USE [BP]+
              INC  BP
              ST   AX,[DT]+
              DJNZ CL,NXT_FPGA
              RET
;**************************************************************


;************************ CLR_FPGA_CNTS ************************
;WRT TO FPGA ADDRS=0000 CLRS ALL CNTRS                         *
;**************************************************************
CLR_FPGA_CNTS:
              LD   BP,#RAMx_FPGA
              ST   AX,[BP]              ;WRT FOR RESET
              RET
;**************************************************************


;************************ EN_FPGA_CNTS ************************
;WRT TO FPGA ADDRS=0002 TO ENABLE                              *
;**************************************************************
EN_FPGA_CNTS: LD   BP,#RAMx_FPGA+02H
              ST   AX,[BP]              ;WRT FOR RESET
              RET
;**************************************************************
```

```
;************************ HOLD_FPGA_CNTS ************************
;WRT TO FPGA ADDRS=0001 TO ENABLE                                 *
;************************************************************
HOLD_FPGA_CNTS:
              LD    BP,#RAMx_FPGA+01H
              ST    AX,[BP]                ;WRT FOR RESET
              RET
;************************************************************



;************************ FILL_DUMMY_CNTS ***********************
;FILL RAM_CNTS                                                    *
;************************************************************
FILL_DUMMY_CNTS:
              LD    BP,#RAM_CNTS
              CLRB  AL
              LDB   CL,#80H
NXT_DUM:      ST    AX,[BP]+               ;WRT FOR RESET
              INCB  AL
              DJNZ  CL,NXT_DUM
              RET
;************************************************************
END



--
*****************************************************************************
******
-- Fpga.src
--
*****************************************************************************
******
$TITLE('FPGA')
$PAGELENGTH(999)

;************************ FPGA MODULE ************************
;ALTERA FPGA-10 EACH 32-BIT CNTRS                                *
;************************************************************
;DSPLY_FPGA_CNTS                                                 *
;DSPLY_CNTS                                                      *
;RD_FPGA_CNTS                                                    *
;CLR_FPGA_CNTS                                                   *
;EN_FPGA_CNTS                                                    *
;HOLD_FPGA_CNTS                                                  *
;FILL_DUMMY_CNTS                                                 *
;************************************************************
;RAM_CNTS+00=CH #0         (FPGA ADDRS=0000)                     *
;RAM_CNTS+04=CH #1                       2                       *
;RAM_CNTS+08=CH #2                       4                       *
;RAM_CNTS+0C=CH #3                       6                       *
;RAM_CNTS+10=CH #4                       8                       *
;RAM_CNTS+14=CH #5                       A                       *
;RAM_CNTS+18=CH #6                       C                       *
```

```
;RAM_CNTS+1C=CH #7                        E                        *
;RAM_CNTS+20=CH #8                       10                        *
;RAM_CNTS+24=CH #9                       12                        *
;*****************************************************************
              PUBLIC  DSPLY_FPGA_CNTS,RD_FPGA_CNTS,CLR_FPGA_CNTS
              PUBLIC  EN_FPGA_CNTS,HOLD_FPGA_CNTS,DSPLY_CNTS
              PUBLIC  FILL_DUMMY_CNTS

              EXTRN   CLR_SCRN,SND_NULL,SND_CR_LF,SND_SP,SND_BNBCD2
              EXTRN   SND_EQUAL,SND_DWORD_MEM,DLY_1sec,STOP_START
              EXTRN   SND_BNBCD3,SND_SLASH,SP_LOOP,SND_BYTE

$INCLUDE(8086_REGS.INC)
$INCLUDE(SFR.INC)
              EXTRN
RAMx_FPGA,RAM_CNTS,No_CHS,RAM_PTSCNTR,RAM_NoSCANSTEPS
              EXTRN   RAM_TEMP

CSEG
;*****************************************************************



;************************ DSPLY_FPGA_CNTS **********************
;FROM MENU                                                       *
;RD CNTS FROM FPGA & DSPLY                                       *
;EXIT WITH CNTRL_C                                               *
;*****************************************************************
DSPLY_FPGA_CNTS:
              CALL CLR_SCRN
RD_AGAIN:     CALL RD_FPGA_CNTS          ;RESULT @ RAM_CNTS
              LD   SI,#MSG_CNTS
              CALL SND_NULL
              CLRB CH
              LDB  CL,#No_CHS            ;# PMT'S
              LD   BP,#RAM_CNTS
NXT_CH:       CALL SND_CR_LF
              CALL SND_SP
              CALL SND_SP
              LDB  AL,CH
              PUSH CX
              CALL SND_BNBCD2            ;CH #
              CALL SND_EQUAL             ;=
              CALL SND_DWORD_MEM         ;CNTS DOUBLE WORD, INCS BP
              POP  CX
              INCB CH
              DJNZ CL,NXT_CH
              CALL SND_CR_LF
              CALL SND_CR_LF
              CALL DLY_1sec              ;UPDATE TIME
              CALL STOP_START            ;STOP ON KYBD HIT
              BR   RD_AGAIN

MSG_CNTS:     DCB  'FPGA PMT CNTS',00H
;*****************************************************************
```

```
;************************* DSPLY_CNTS ************************
;FROM RUN MODE--CHKOUT                                          *
;ENTER with: CNTS @ RAM_CNTS                                    *
;***************************************************************
DSPLY_CNTS:   CALL SND_CR_LF
              LDB  AL,RAM_PTSCNTR        ;CNTR
              CLRB AH
              CALL SND_BNBCD3            ;SND SCAN STEP #
              CALL SND_SLASH             ;SND /
              LDB  AL,RAM_NoSCANSTEPS    ;SND TOTAL # SCAN STEPS
              SHLB AL,#1
              DECB AL
              CLRB AH
              CALL SND_BNBCD3            ;SND TEMPERATURE
              LDB  CL,#0DH
              CALL SP_LOOP
              LD   SI,#MSG_TEMP
              CALL SND_NULL
              LDB  AL,RAM_TEMP
              CALL SND_BYTE
              CALL SND_CR_LF
              LDB  CL,#No_CHS            ;# PMT'S
              SHRB CL,#1                 ;DIVIDE BY 2
              LDB  CH,CL
              LDB  DL,#02H               ;# LINES
              LD   BP,#RAM_CNTS
NXT_CNTR:     PUSH CX
              PUSH DX
              CALL SND_DWORD_MEM         ;CNTS DOUBLE WORD, INCS BP
              CALL SND_SP
              CALL SND_SP
              POP  DX
              POP  CX
              DJNZ CL,NXT_CNTR
              CALL SND_CR_LF
              LDB  CL,CH
              DJNZ DL,NXT_CNTR
              RET

MSG_TEMP:     DCB  'TEMPERATURE(00-FF)=',00H
;***************************************************************



;************************ RD_FPGA_CNTS ************************
;EXIT with: RAM_CNTS+00=CH #0        (FPGA ADDRS=0000)          *
;         : RAM_CNTS+04=CH #1                    2              *
;         : RAM_CNTS+08=CH #2                    4              *
;         : RAM_CNTS+0C=CH #3                    6              *
;         : RAM_CNTS+10=CH #4                    8              *
;         : RAM_CNTS+14=CH #5                    A              *
;         : RAM_CNTS+18=CH #6                    C              *
;         : RAM_CNTS+1C=CH #7                    E              *
```

```
;          : RAM_CNTS+20=CH #8                          10          *
;          : RAM_CNTS+24=CH #9                          12          *
;                                                                   *
;NOTE:EVEN ADDRESSES=LO WORD                                        *
;     : ODD ADDRESSES=HI WORD                                       *
;*******************************************************************
RD_FPGA_CNTS: LD   DT,#RAM_CNTS
              LD   BP,#RAMx_FPGA
              LDB  CL,#No_CHS
NXT_FPGA:     LD   AX,[BP]              ;CAN'T USE [BP]+
              INC  BP
              ST   AX,[DT]+
              LD   AX,[BP]              ;CAN'T USE [BP]+
              INC  BP
              ST   AX,[DT]+
              DJNZ CL,NXT_FPGA
              RET
;*******************************************************************



;************************* CLR_FPGA_CNTS ************************
;WRT TO FPGA ADDRS=0000 CLRS ALL CNTRS                              *
;*******************************************************************
CLR_FPGA_CNTS:
              LD   BP,#RAMx_FPGA
              ST   AX,[BP]              ;WRT FOR RESET
              RET
;*******************************************************************



;************************** EN_FPGA_CNTS ************************
;WRT TO FPGA ADDRS=0002 TO ENABLE                                   *
;*******************************************************************
EN_FPGA_CNTS: LD   BP,#RAMx_FPGA+02H
              ST   AX,[BP]              ;WRT FOR RESET
              RET
;*******************************************************************



;************************ HOLD_FPGA_CNTS ************************
;WRT TO FPGA ADDRS=0001 TO ENABLE                                   *
;*******************************************************************
HOLD_FPGA_CNTS:
              LD   BP,#RAMx_FPGA+01H
              ST   AX,[BP]              ;WRT FOR RESET
              RET
;*******************************************************************
```

```
;************************ FILL_DUMMY_CNTS ***********************
;FILL RAM_CNTS                                                  *
;*************************************************************
FILL_DUMMY_CNTS:
              LD   BP,#RAM_CNTS
              CLRB AL
              LDB  CL,#80H
NXT_DUM:      ST   AX,[BP]+              ;WRT FOR RESET
              INCB AL
              DJNZ CL,NXT_DUM
              RET
;*************************************************************
END
```

```
--
*************************************************************************
******
-- Equ.src
--
*************************************************************************
******
$TITLE('EQU MODULE')
$PAGELENGTH(75)

;************************ EQUATE MODULE *************************
; DEFINES SYSTEM GLOBAL CONSTANTS & ALLOCATES SYSTEM RAM        *
;*************************************************************
; 87C196KB              87C196KC              87C196KD          *
;     12/16 MHz             16 MHz                20 MHz        *
;     232 RAM               488 RAM               1K RAM        *
;       232 Lower Regs        232 Lower Reg File    232 Lower Reg*
;         0 Upper Regs        256 Upper Reg Ram     768 Upper Reg*
;      8K EPROM(3FFF)       16K OTP(5FFF)        32K OTP(9FFF)  *
;                             -BMOVI                            *
;                             -XCH/XCHB                         *
;                             -TIJMP                            *
;                                                               *
; Upper Regs-Accessed using indirect or indexed addressing      *
; Vertical Windowing-maps sections of Register Ram into the     *
;    upper section of lower Reg File in 32,64 or 128 bytes      *
;         87C196KC=16  32 byte VWindows   See Appendix C        *
;                  = 8  64 byte VWindows                        *
;                  = 4 128 byte VWindows                        *
;                                                               *
; WSI PERIPHERAL CHIP                                           *
;    RAM-2K:         6000-67FFH                                 *
;    I/O PORT P5:    6802,4,6H      (SEE BOTTOM EQU.SRC)        *
;    I/O PORT P6:    6803,5,7H                                  *
;*************************************************************
;GLOBALS
              PUBLIC  AX,AL,AH,BX,BU,BL,CX,CH,CL,DX,DH,DL,PWM0_CNTRL
              PUBLIC  BP,SI,DT,SP,IMASK,IPEND,SBUF,IOS0,IOS1,CCR
              PUBLIC  P0,P1,P2,P3,P4,BAUD_REG,AD_RESULT,IOS2,R0
              PUBLIC  IOC1,SPCON,STACK_VALUE,IMASK1,IPEND1,SP_STAT
              PUBLIC  IOC0,IOC2,WSR,HSI_MODE,TIMER2,TIMER1,AD_CMND
```

```
                PUBLIC  HSI_STATUS,HSI_TIME,HSO_CMND,HSO_TIME
                PUBLIC  PWM1_CNTRL,PWM2_CNTRL,IOC3,AD_TIME

                PUBLIC  RAM_SCRATCH,RAM_P1,RAM_P2,RAM_P3,RAM_P4,RAM_T1
                PUBLIC  RAM_PWM0
;               PUBLIC
RAM_PWM1,RAM_PWM2,RAM_PWM3,RAM_PWM4,RAM_PWM5,RAM_PWM6
                PUBLIC  RAM_BAUD,RAM_BAUD1,RAM_T2,RAM_RUNFLG,RAM_SCRATCH1
                PUBLIC  RAM_HSIO,RAM_SIOFLG

                PUBLIC  CODECRT_CLR,CODECRT_HOME,HWin0,HWin15,HWin1
                PUBLIC
BAUD0_9600,BAUD1_9600,SIZE_LOWER,SOFT_9600,SOFT_1200
                PUBLIC  BAUD0_19K,BAUD1_19K,BAUD0_38K,BAUD1_38K,BAUD0_4800
                PUBLIC  BAUD1_4800,BAUD0_57K,BAUD1_57K,SIZE_UPPER
                PUBLIC  V128_0,V128_1,V128_2,V128_3,No_PW_CHS,No_PW_EDGES
                PUBLIC  CODE_ACK,CODE_NAK

;USER PUBLICS
                PUBLIC  RAM_WSI,WSI_P5_PIN,WSI_P5_DIR,WSI_P5_DATA
                PUBLIC  RAM_CNTS,RAMx_FPGA,RAM_STEPOUT,RAM_TEMP,RAM_DATA
                PUBLIC  RAM_STEPSPD,RAM_STEP1CM,RAM_STEPIN,RAM_STEPCNTS
                PUBLIC  RAM_PTSCNTR,RAM_NoSCANSTEPS,RAM_CNTTIME,RAM_RDDLY
                PUBLIC  RAM_NOHOST,RAMx_PTR

                PUBLIC
DEFAULT_PWM0,CURRENT_LIMIT,DEFAULT_SPEED,RD_DLY,No_CHS
                PUBLIC
No_SCAN_STEPS,CNT_TIME,DEFAULT_STEPINx,DEFAULT_STEPOUTx
                PUBLIC  DEFAULT_STEP1CMx,DEFAULT_STEP1CMy,DEFAULT_STEPINy
                PUBLIC  DEFAULT_STEPOUTy,No_HOST_TRYS,No_BYTES_DATA
                PUBLIC  RQST_SNDDATA,RAMx_BASE,RAMx_SIZE
;*****************************************************************



;******************* SPECIAL FUNCTION REGISTERS ****************
; SFR'S                                                        *
; ADDRESSES 0000-0019                                          *
;*****************************************************************
; HORIZONTAL "WINDOW #0"
                R0              EQU   00H:WORD        ;R/W
                AD_RESULT       EQU   02H:WORD        ;R
                AD_CMND         EQU   02H:BYTE        ; /W
                HSI_MODE        EQU   03H:BYTE        ; /W
                HSI_TIME        EQU   04H:WORD        ;R
                HSO_TIME        EQU   04H:WORD        ;W
                HSI_STATUS      EQU   06H:BYTE        ;R
                HSO_CMND        EQU   06H:BYTE        ; /W
                SBUF            EQU   07H:BYTE        ;R/W
                IMASK           EQU   08H:BYTE        ;R/W
                IPEND           EQU   09H:BYTE        ;R/W
                TIMER1          EQU   0AH:WORD        ;R
                IOC2            EQU   0BH:BYTE        ; /W
                TIMER2          EQU   0CH:WORD        ;R/W
                BAUD_REG        EQU   0EH:BYTE        ; /W
```

```
                P0              EQU  0EH:BYTE        ;R
                P1              EQU  0FH:BYTE        ;R/W
                P2              EQU  10H:BYTE        ;R/W
                SPCON           EQU  11H:BYTE        ; /W
                SP_STAT         EQU  11H:BYTE        ;R
                IPEND1          EQU  12H:BYTE        ;R/W
                IMASK1          EQU  13H:BYTE        ;R/W
                WSR             EQU  14H:BYTE        ;R/W
                IOC0            EQU  15H:BYTE        ; /W
                IOS0            EQU  15H:BYTE        ;R
                IOC1            EQU  16H:BYTE        ; /W
                IOS1            EQU  16H:BYTE        ;R
                IOS2            EQU  17H:BYTE        ;R
                PWM0_CNTRL      EQU  17H:BYTE        ; /W
                SP              EQU  18H:WORD        ;R/W
                CCR             EQU  2018H:BYTE      ;R/W
                P3              EQU  1FFEH:BYTE      ;R/W
                P4              EQU  1FFFH:BYTE      ;R/W
;HORIZONTAL "WINDOW #1"
                AD_TIME         EQU  03H:BYTE        ;R/W
                IOC3            EQU  0CH:BYTE        ;R/W
                PWM1_CNTRL      EQU  16H:BYTE        ;R/W
                PWM2_CNTRL      EQU  17H:BYTE        ;R/W
;*************************************************************



;********************* LOWER REGISTER FILE **********************
; 00-FFH  KC & KD    256 BYTES                                   *
;                                                               *
; DSW'S AUTOMATICALLY ADJUSTED TO EVEN BOUNDARIES               *
; DSL'S AUTOMATICALLY ADJUSTED TO QUAD BOUNDARIES               *
;                                                               *
; SOME INSTRUCTIONS REQUIRE LONGS(LONG WORDS) (MULU,DIVU,ETC)   *
;    -ADDRESS DIVISIBLE BY 4                                    *
;    -CAN USE BX,DX & SI    (SUBS ALSO HAS A DECLARED LONG=DW)  *
;                                                               *
; LINKER HAS STACKSIZE=32    RAM=<E0H                           *
;*************************************************************
RSEG
;8086 REGS  (RAM ADDRS 001A-0027)
                AX:  DSW 1                  ;001A
                AL   EQU AX:BYTE
                AH   EQU (AX+1):BYTE

                BX:  DSW 1                  ;001C (OK FOR LONG'S)
                BL   EQU BX:BYTE
                BU   EQU (BX+1):BYTE

                CX:  DSW 1                  ;001E
                CL   EQU CX:BYTE
                CH   EQU (CX+1):BYTE

                DX:  DSW 1                  ;0020 (OK FOR LONG'S)
                DL   EQU DX:BYTE
                DH   EQU (DX+1):BYTE
```

```
                BP:  DSW 1                  ;0022
                SI:  DSW 1                  ;0024 (OK FOR LONG'S)
                DT:  DSW 1                  ;0026

;DSW'S AUTOMATICALLY ADJUSTED TO EVEN BOUNDARIES
;RAM ADDRS (0028-00FF)  (REMEMBER STACK IS DOWNWARD FROM 00FF)
                RAM_SCRATCH:  DSB 10H       ;GENERAL PURPOSE(SUBS & DEBUG)
                RAM_SCRATCH1: DSB 10H       ;
                RAM_P1:       DSB 01H       ;I/O PORT SHADOWS
                RAM_P2:       DSB 01H       ;    "
                RAM_P3:       DSB 01H       ;    "
                RAM_P4:       DSB 01H       ;    "
                RAM_BAUD:     DSW 01H       ;SOFTWARE SIO #1
                RAM_BAUD1:    DSW 01H       ;SOFTWARE SIO #2
                RAM_T1:       DSW 01H       ;TIMER1 CNTUP
                RAM_T2:       DSW 01H       ;TIMER 2 CNTDWN VALUE
                RAM_RUNFLG:   DSB 01H
                RAM_SIOFLG:   DSB 01H       ;HARD/SOFT UART FLG(01=SOFT)

                RAM_PWM0:     DSB 01H       ;DUTY CYCLE
;               RAM_PWM1:     DSB 01H
;               RAM_PWM2:     DSB 01H
;               RAM_PWM3:     DSW 01H       ;PWM3 SHADOW
;               RAM_PWM4:     DSW 01H       ;PWM4 SHADOW
;               RAM_PWM5:     DSW 01H       ;PWM5 SHADOW
;               RAM_PWM6:     DSW 01H       ;PWM6 SHADOW
;               RAM_WIDTH:    DSW 04H       ;HSI PULSE WIDTHS
                RAM_HSIO:     DSB 01H       ;HSIO SHADOW
;************************* USER RAM ***************************
                RAM_STEPCNTS: DSW 02H       ;# STEPPER MTR CNTS
                RAM_STEP1CM:  DSW 02H       ;# STEPPER MTR CNTS
                RAM_STEPIN:   DSW 02H       ;# STEPPER MTR CNTS
                RAM_STEPOUT:  DSW 02H       ;# STEPPER MTR CNTS
                RAM_STEPSPD:  DSW 01H       ;STEEPER MTR SPEED
                RAM_NoSCANSTEPS: DSW 01H    ;EACH DIRECTION
                RAMx_PTR:     DSW 01H       ;EXT RAMx WRT PTR
                RAM_CNTTIME:  DSB 01H       ;CNTR
                RAM_RDDLY:    DSB 01H       ;CNTR
                RAM_NOHOST:   DSB 01H       ;HOST ALIVE FLG
                RAM_PTSCNTR:  DSB 01H       ;LOCATION CNTR
;*************************************************************



;NOT USED
;********************** UPPER REGISTER FILE *******************
; 100H-1FFH KC   256 BYTES                                     *
; 100H-3FFH KD   768 BYTES                                     *
;                                                              *
; NO WINDOWING:                                                *
;    ACCESS RAM USING "INDIRECT" OR "INDEXED" ADDRESSING       *
;        LDB  AL,[BP]     INDIRECT                            *
;        LDB  AL,5[BP]    INDEXED                             *
;    NOT VALID IS REGISTER DIRECT                              *
;        LDB  AL,AH       DIRECT                              *
;                                                              *
```

```
;                                                                    *
; VERTICAL WINDOWING-87C196KC & KD                                   *
; 87C196KC=16  32 byte VWindows   See Appendix C-88                  *
;         = 8  64 byte VWindows                                      *
;         = 4 128 byte VWindows    =512??                            *
; REGISTER DIRECTS ON ADDRESSES WITHIN LOW REG FILE WINDOW           *
;      ACCESSES VWINDOW IN UPPER REG RAM IF WINDOW IS ACTIVE         *
; INDIRECT OR INDEXED THAT USES ADDRESS WITHIN THE LOW REG           *
;      FILE OR VWINDOW ACESSES THE ACTUAL MEMORY LOCATION            *
; MAPPING:                                                           *
;       32 BYTE WINDOWS TO LOWER REF FILE:(00E0-00FF)                *
;       64 BYTE WINDOWS TO LOWER REF FILE:(00C0-00FF)                *
;      128 BYTE WINDOWS TO LOWER REF FILE:(0080-00FF)                *
; EXAMPLE-WHEN VWINDOWING ENABLE                                     *
;         PUSHA               ;SAVE WSR                            *
;         LDB WSR,VW128_3     ;ENABLE/SELECT VWINDOW 3             *
;         ADD 40H,80H         ;MEM_WORD(40H) + MEM_WORD(180H)      *
;                             ;REGISTER DIRECT                     *
;         ADD 40H,80H[0]      ;MEM_WORD(40H) + MEM_WORD(80H+0)     *
;                             ;REGISTER INDIRECT                   *
;         ADD 40H,180H[0]     ;MEM_WORD(40H) + MEM_WORD(180H+0)    *
;         POPA                ;RELOAD WSR                          *
;*********************************************************************
             V128_0        EQU  10H     ;ADDRS=0000
             V128_1        EQU  11H     ;ADDRS=0080
             V128_2        EQU  12H     ;ADDRS=0100
             V128_3        EQU  13H     ;ADDRS=0180
;*********************************************************************



;************************** CONSTANTS ***************************
; KEEP STACK ON EVEN ADDRESSES                                       *
;*********************************************************************
;----87C196 Constants
             STACK_VALUE   EQU  100H    ;STACK POINTER-196KC (KEEP OUT
OF UPPER REG)
;            STACK_VALUE   EQU  200H    ;STACK POINTER-196KC
;            STACK_VALUE   EQU  400H    ;STACK POINTER-196KD ????
; BAUD1_MSB=80, BAUD0_LSB=(Fosc/BAUDRATEx16)-1
             BAUD0_4800    EQU  0CFH    ;BAUD VALUE-4800 @ 16MHz
             BAUD1_4800    EQU  80H     ;      "
;            BAUD0_9600    EQU  4DH     ;BAUD VALUE-9600 @ 12MHz
;            BAUD1_9600    EQU  80H     ;      "
             BAUD0_9600    EQU  67H     ;BAUD VALUE-9600 @ 16MHz
             BAUD1_9600    EQU  80H     ;      "
;            BAUD0_9600    EQU  82H     ;BAUD VALUE-9600 @ 20MHz
;            BAUD1_9600    EQU  80H     ;      "
             BAUD0_19K     EQU  34H     ;BAUD VALUE-19.2K @ 16MHz
             BAUD1_19K     EQU  80H     ;      "
;            BAUD0_19K     EQU  40H     ;BAUD VALUE-19.2K @ 20MHz
;            BAUD1_19K     EQU  80H     ;      "
             BAUD0_38K     EQU  19H     ;BAUD VALUE-57.6K @ 16MHz
             BAUD1_38K     EQU  80H     ;      "
;            BAUD0_38K     EQU  20H     ;BAUD VALUE-57.6K @ 20MHz
;            BAUD1_38K     EQU  80H     ;      "
```

```
                BAUD0_57K       EQU   10H       ;BAUD VALUE-57.6K @ 16MHz
                BAUD1_57K       EQU   80H       ;       "
;               BAUD0_57K       EQU   15H       ;BAUD VALUE-57.6K @ 20MHz
;               BAUD1_57K       EQU   80H       ;       "
                HWin0           EQU   00H       ;HORIZ WINDOW #0
                HWin1           EQU   01H       ;HORIZ WINDOW #1
                HWin15          EQU   0FH       ;HORIZ WINDOW #15
                CODECRT_CLR     EQU   1AH       ;FOR DEBUG
                CODECRT_HOME    EQU   00H       ;FOR DEBUG
                MENU_INDENTS    EQU   10H       ;FOR SUBS.SRC, "SND_TABLE"
                SOFT_1200       EQU   0238H     ;1200 BAUD-SOFTWARE DLY (16MHz)
;               SOFT_9600       EQU   0047H     ;9600 BAUD-SOFTWARE DLY (16MHz)
                SOFT_9600       EQU   0048H     ;9600 BAUD-SOFTWARE DLY (16MHz)
;               SOFT_9600       EQU   0037H     ;9600 BAUD-SOFTWARE DLY (12MHz)
;               SPEED           SET CRYSTAL SPEED IN SUBSX.INC
                CODE_ACK        EQU   06H
                CODE_NAK        EQU   15H

;STACK IN UPPER RAM <200H
;               SIZE_LOWER      EQU   0D8H      ;28-FFH    (LOWER RAM)
;               SIZE_UPPER      EQU   0F0H      ;100-1F0H (UPPER RAM)
;STACK IN LOWER RAM <100H
                SIZE_LOWER      EQU   0C8H      ;28-F0H    (LOWER RAM)
                SIZE_UPPER      EQU   100H      ;100-1FFH (UPPER RAM)

;User Constants
                No_PW_CHS       EQU   01H       ;# PULSE WIDTH CHS
                No_PW_EDGES     EQU   No_PW_CHS+No_PW_CHS
                DEFAULT_PWM0    EQU   20H       ;DC MTR SPEED, SETS PULSE WIDTH

;External RAM particulars
                RAMx_FPGA       EQU   6000H     ;EXTERNAL RDS/WRTS    (MIN 6000
KC,A000 KD)
                RAMx_BASE       EQU   8000H     ;EXTERNAL RDS/WRTS    (MIN 6000
KC,A000 KD)
                RAMx_SIZE       EQU   8000H     ;EXTERNAL RDS/WRTS    (MIN 6000
KC,A000 KD)
                No_CHS          EQU   0AH       ;# PMT CHANNELS
                RAM_DATA        EQU   100H      ;uP RAM ADDRS
                RAM_TEMP        EQU   101H      ;uP RAM ADDRS
                RAM_CNTS        EQU   102H      ;uP RAM ADDRS
                No_BYTES_DATA   EQU   42        ;LOCATION #, TEMP,10 CNTRS @ 4
BYTES EACH
                CURRENT_LIMIT   EQU   40H       ;FOR WHEN DC MTR HITS
MECHANICAL STOP
                                                ;.33V OK, 2.2V @ STOP

                DEFAULT_SPEED EQU   0004H     ;STEPPER BD SPEED
                DEFAULT_STEPINy  EQU  0003H     ;STEPPER BD LOAD-MSW
                DEFAULT_STEPINx  EQU  0C8C0H    ;STEPPER BD LOAD-LSW
                DEFAULT_STEPOUTy EQU  0005H     ;STEPPER BD UNLOAD-MSW
                DEFAULT_STEPOUTx EQU  0000H     ;STEPPER BD UNLOAD-LSW
                DEFAULT_STEP1CMy EQU  0000H     ;STEPPER BD 1 CM-MSW
                DEFAULT_STEP1CMx EQU  28A0H     ;STEPPER BD 1 CM-LSW

;               No_SCAN_STEPS EQU   50        ;# INSPECTION STEPS IN EACH
DIRECTION
```

```
                No_SCAN_STEPS EQU   5       ;# INSPECTION STEPS IN EACH
DIRECTION
;               CNT_TIME      EQU   50      ;REST TIME AT EACH SPOT x 100ms
                CNT_TIME      EQU   10      ;REST TIME AT EACH SPOT x 100ms
                RD_DLY        EQU    5      ;RD DLY AFTER LIGHTS OFF x
100ms

                No_HOST_TRYS  EQU   01H     ;# RETRYS FOR uP TO HOST WAKEUP
                RQST_SNDDATA  EQU   80H     ;uP TO HOST

;NOT USED---WSI I/O Particulars for Ports P5 & P6
                RAM_WSI       EQU   6000H           ;RAM BASE ADDRESS (2K)
                WSI_P5_BASE   EQU   6802H           ;P5 BASE ADDRESS
                WSI_P5_PIN    EQU   WSI_P5_BASE+0   ;PIN REGISTER-OUT
READBACK
                WSI_P5_DIR    EQU   WSI_P5_BASE+2   ;DIRECTION REGISTER
                WSI_P5_DATA   EQU   WSI_P5_BASE+4   ;DATA REGISTER
                WSI_P6_BASE   EQU   6803H           ;P6 BASE ADDRESS
                WSI_P6_PIN    EQU   WSI_P6_BASE+0   ;PIN REGISTER-OUT
READBACK
                WSI_P6_DIR    EQU   WSI_P6_BASE+2   ;DIRECTION REGISTER
                WSI_P6_DATA   EQU   WSI_P6_BASE+4   ;DATA REGISTER
;*****************************************************************
END


--
*************************************************************************
******
-- Dc.src
--
*************************************************************************
******
$TITLE('DC')
$PAGELENGTH(999)

;************************ DC MTR MODULE ************************
;USE PWM0                                                      *
;*************************************************************
; DC_MTR_MENU                                                  *
; DSPLY_CURRENT                                               *
; INIT_DC_MTR                                                 *
; RUN_DC_RIGHT        (RUN TO MECHANICAL WALL & STOP)         *
; RUN_DC_LEFT                    "                            *
; RUN_MTR                                                     *
; RUN_MTR_FOREVER                                             *
; STOP_DC_MTR                                                 *
; SET_DIR_CW                                                  *
; SET_DIR_CCW                                                 *
; RD_MTR_CURRENT                                              *
;*************************************************************
; P2.7=BEEP OUT------------OUT    (I/O)                       *
; P2.6=SOLENOID ON---------OUT    (I/O)                       *
; P2.5=DC PWM0-------------OUT    (I/O or PWM1)(IOC3.2=1 PWM) *
; P2.4=--------------------IN     (INPUT)                     *
; P2.3=--------------------IN     (T2 INPUT)                  *
; P2.2=--------------------IN     (INPUT-EXT INT1)            *
```

```
; P2.1=Rx RS232 HOST--------IN     (HARDWARE SIO-9600 BAUD)     *
; P2.0=Tx RS232 HOST--------OUT                 "               *
;                                                               *
; P1.7=DC DIR---------------OUT    (I/O)                        *
; P1.6=DC BRAKE-------------OUT    (I/O)                        *
; P1.5=LIGHTS PWR ON--------OUT    (I/O or PWM1)(IOC3.2=1 PWM)  *
; P1.4=PMT PWR ON-----------OUT    (I/O or PWM2()IOC3.3=1 PWM)  *
; P1.3=STEP RESET-----------OUT    (I/O)                        *
; P1.2=STEP ENABLE----------OUT    (I/O)                        *
; P1.1=STEP DIR-------------OUT    (I/O)                        *
; P1.0=STEP CLK-------------OUT    (I/O)                        *
;                                                               *
; P0.7=FP SWCHS                    (EXT INT)                    *
; P0.6=LOAD                                                     *
; P0.5=SCAN                                                     *
; P0.4=UNLOAD                                                   *
; P0.3=DOOR SWCH                                                *
; P0.2=                                                         *
; P0.1=TEMPERATURE                                              *
; P0.0=DC MTR CURRENT                                           *
;***************************************************************
              PUBLIC  DC_MTR_MENU,INIT_DC_MTR,STOP_DC_MTR
              PUBLIC  RUN_DC_RIGHT,RUN_DC_LEFT,RUN_DC_FOREVER

              EXTRN   WAIT_KYBD,SND_TABLE,CMND_ERROR,CLR_SCRN,SND_BYTE
              EXTRN
SET_PWM_15K,LD_PWM0,EN_PWM0,STOP_PWM0,RD_AD_8,SND_NULL
              EXTRN   ENTR_SHOW_BYTE,DLY_100ms,SND_CR_LF

$INCLUDE(8086_REGS.INC)
$INCLUDE(SFR.INC)
              EXTRN   RAM_P1,CURRENT_LIMIT,RAM_PWM0,DEFAULT_PWM0,RAM_P2

CSEG
;***************************************************************




;************************* DC_MTR_MENU ************************
;                                                               *
;***************************************************************
DC_MTR_MENU:  CALL MENU_PREP
MENU_1:       CALL WAIT_KYBD              ;AL=RESULT
              CMPB AL,#01H                ;CNTRL A=RUN MTR LEFT TO LIMIT
              BE   M_LEFT
              CMPB AL,#02H                ;CNTRL B=RUN MTR RIGHT TO LIMIT
              BE   M_RIGHT
              CMPB AL,#0BH                ;CNTRL K=STOP MTR
              BE   M_STOP
;             CMPB AL,#13H                ;CNTRL S=SET MTR SPEED
;             BE   MENU_SPEED
              CMPB AL,#12H                ;CNTRL R=RD CURRENT LIMIT
              BE   DSPLY_CURRENT
              CMPB AL,#0DH                ;CR=RETURN
              BE   M_RET
```

```
;ERROR MESSAGE & TRY AGAIN
              CALL CMND_ERROR
              BR   MENU_1

M_RET:        RET

M_LEFT:       CALL CLR_SCRN
              CALL SET_DIR_CW
              CALL RUN_DC_LEFT
              BR   DC_MTR_MENU

M_RIGHT:      CALL CLR_SCRN
              CALL SET_DIR_CW
              CALL RUN_DC_RIGHT
              BR   DC_MTR_MENU

M_STOP:       CALL STOP_DC_MTR
              BR   DC_MTR_MENU
;*****************************************************************



;************************* MENU_PREP ****************************
;                                                                *
;*****************************************************************
MENU_SENDS    EQU  05H                  ;INCLUDES TITLE
MENU_INDENTS  EQU  10H

MENU_PREP:    CALL CLR_SCRN
              LD   SI,#MENU_TABLE
              LDB  CL,#MENU_INDENTS     ;# OF INDENTS
              LDB  CH,#MENU_SENDS       ;# OF SENDS
              CALL SND_TABLE
              RET

;LOOKUP FROM EPROM
MENU_TABLE:   DCW  MENU_TITLE           ;POINTERS
              DCW  CW
              DCW  CCW
;             DCW  SPEED
              DCW  CURRENT
              DCW  STOP

MENU_TITLE:   DCB  'DC MTR MENU',0AH,0DH,00H
CW:           DCB  'CNTRL A=RUN MTR LEFT TO LIMIT',0AH,0AH,0DH,00H
CCW:          DCB  'CNTRL B=RUN MTR RIGHT TO LIMIT',0AH,0AH,0DH,00H
;SPEED:        DCB  'CNTRL S=SET MTR SPEED',0AH,0AH,0DH,00H
CURRENT:      DCB  'CNTRL R=RD MTR CURRENT',0AH,0AH,0DH,00H
STOP:         DCB  'CNTRL K=STOP MTR',0AH,0AH,0DH,00H
;*****************************************************************



;NOT USED
;************************* MENU_SPEED ***************************
;SET MTR SPEED                                                   *
```

```
;*************************************************************
MENU_SPEED:    CALL CLR_SCRN
               LD   SI,#MSG_SPEED
               LD   BP,#RAM_PWM0
               CALL ENTR_SHOW_BYTE
               BR   DC_MTR_MENU

MSG_SPEED:     DCB  'ENTER MTR SPEED DUTY CYCLE(00=0%, FF=100%)=',00H
;*************************************************************



;************************ DSPLY_CURRENT ***********************
;EXIT WITH CNTRL C                                            *
;*************************************************************
DSPLY_CURRENT:
               CALL CLR_SCRN
NXT_CURRENT:   LD   SI,#MSG_CURRENT
               CALL SND_NULL
               CALL RD_MTR_CURRENT
               CALL SND_BYTE
               CALL DLY_100ms
               CALL DLY_100ms
               CALL DLY_100ms
               BR   NXT_CURRENT

MSG_CURRENT:   DCB  0AH,0DH,'CURRENT (00-FF)=',00H
;*************************************************************



;************************ INIT_DC_MTR *************************
;                                                             *
;*************************************************************
INIT_DC_MTR:   CALL SET_PWM_15K          ;15 KHz REP RATE (OR 31KHz)
               CALL EN_PWM0              ;SELECT P2.5 FOR PWM0 OUT
               LDB  RAM_PWM0,#DEFAULT_PWM0
               RET
;*************************************************************



;************************ RUN_DC_LEFT *************************
;                                                             *
;*************************************************************
RUN_DC_LEFT:   CALL SET_DIR_CCW
               CALL RUN_MTR
               RET
;*************************************************************



;************************ RUN_DC_RIGHT ************************
```

```
;RUN TO LIMIT & STOP                                                    *
;***************************************************************
RUN_DC_RIGHT: CALL SET_DIR_CW
              CALL RUN_MTR
              RET
;***************************************************************



;*************************** RUN_MTR ****************************
;RUN MTR INTO MECHANICAL STOP                                          *
;  -DETERMINE MECHANICAL STOP BY MONITORING CURRENT LIMIT              *
; P2.7=BEEPER---------------OUT     (I/O)                             *
; P2.6=SOLENOID ON----------OUT     (I/O)                             *
; P2.5=DC PWM---------------OUT     (I/O or PWM1)(IOC3.2=1 PWM)       *
; P2.4=---------------------IN      (INPUT)                           *
; P2.3=---------------------IN      (T2 INPUT)                        *
; P2.2=---------------------IN      (INPUT-EXT INT1)                  *
; P2.1=Rx RS232 HOST--------IN      (HARDWARE SIO-9600 BAUD)          *
; P2.0=Tx RS232 HOST--------OUT                 "                     *
;***************************************************************
RUN_MTR:
;             LDB  AL,RAM_PWM0            ;SPEED
;             CALL LD_PWM0                ;START DC MTR
              CALL RUN_DC_FOREVER
              CALL DLY_100ms
MTR_ON:       CALL RD_MTR_CURRENT
              CMPB AL,#CURRENT_LIMIT
              JNH  MTR_ON                 ;<=
              CALL STOP_DC_MTR
              RET
;***************************************************************



;************************ RUN_DC_FOREVER ************************
;STOP WITH CNTRL C OR MENU CHOICE                                      *
;***************************************************************
RUN_DC_FOREVER:
              ORB  RAM_P2,#20H
              STB  RAM_P2,P2
;             LDB  AL,RAM_PWM0
;             CALL LD_PWM0                ;START DC MTR
              RET
;***************************************************************



;************************** STOP_DC_MTR *************************
;                                                                      *
;***************************************************************
STOP_DC_MTR:
;             CALL STOP_PWM0
              ANDB RAM_P2,#0DFH
```

```
                STB   RAM_P2,P2
                RET
;***************************************************************



;************************ SET_DIR_CW ************************
;                                                              *
;***************************************************************
SET_DIR_CW:     ORB   RAM_P1,#80H
                STB   RAM_P1,P1
                RET
;***************************************************************



;************************ SET_DIR_CCW ***********************
;                                                              *
;***************************************************************
SET_DIR_CCW:    ANDB RAM_P1,#7FH
                STB   RAM_P1,P1
                RET
;***************************************************************



;************************ RD_MTR_CURRENT ********************
;EXIT with: AL=VALUE                                           *
;***************************************************************
RD_MTR_CURRENT:
                LDB   CL,#00H                ;CH #
                CALL RD_AD_8
                RET
;***************************************************************
END


--
*****************************************************************************
******
-- Ccr.src
--
*****************************************************************************
******
$TITLE('CCR')
$PAGELENGTH(75)

;***************************** CCR ******************************
; CHIP CONFIGURATION REGISTER (CCR)-READ ONLY AT RESET          *
;      -87C196 INTERNAL OTP @ 2018H                             *
;***************************************************************
; DEFAULT=1F              (SEE MANUAL APPENDIX C-16)            *
;                                                               *
; BIT #7=LOC1         (PROGRAMMING PROTECTION)                  *
```

```
; BIT #6=LOC0                    "                                        *
; BIT #5=IRC1          (#WAIT STATES,00=1,01=2,10=3,11=RDY PIN)          *
; BIT #4=IRC0                    "                                        *
; BIT #3=ALE/AVD       (ALE=1)                                           *
; BIT #2=WR            (16 BIT WRITE CYCLES)                             *
; BIT #1=BW           BUSWIDTH (0=8 BITS) (1=BW PIN,16 BITS=1)           *
; BIT #0=PD            (IDLPD #2 POWERDOWN ENABLE=1)                     *
;                                                                        *
; NOTE: FOR WSI CHIP CCR=F1                                              *
;************************************************************************
CSEG
;2018H                                      ;RESET=FF
;             DCB  0F1H                     ;SET FOR 8-BITS & AVD
;             DCB  0F9H                     ;SET FOR 8-BITS & ALE
              DCB  0FFH                     ;16 BIT

;2019H
              DCB  20H                      ;RESERVED-MUST CONTAIN 20H
;************************************************************************
END


************************************************************************
******
--    END OF OSL Image Reader Firmware Source Code
--
************************************************************************
******
```

```
************************************************************************
******
--          Altera Field Programmable Gate Array Source Code
--
************************************************************************
******
************************************************************************
******
-- tri_state.vhd
--
************************************************************************
******
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity tri_state is
  port(counter_in  : in std_logic_vector(15 downto 0);
       en : in std_logic;
       tri_out  : inout std_logic_vector(15 downto 0));
end;
architecture behavior of tri_state is
begin
process (counter_in, en)
  begin
    if en = '1' then
      tri_out <= counter_in;
    else
      tri_out <= (others=>'Z');
    end if;
  end process;
end;


--
************************************************************************
******
-- rd_wrt_decoder.vhd
--
************************************************************************
******
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
use ieee.std_logic_unsigned.all;
entity rd_wrt_decoder is
  port (ad : in std_logic_vector (4 downto 0);
            rd,wrt,ce : in std_logic;
            clr_cntr : out std_logic;
            en_cntr : out std_logic;
            sel_counter : out std_logic_vector (4 downto 0));
  end;

architecture behavior of rd_wrt_decoder is
begin
  process (rd,wrt,ce)
    begin
```

```
      clr_cntr <= '0';
      sel_counter <= (others=>'0');
      if wrt='0' and ce ='0' then
            case ad is
            when "00000" => clr_cntr <= '1';
            when "00001" => en_cntr <= '0';
            when "00010" => en_cntr <= '1';
            when others => en_cntr <= '1';
                           clr_cntr <= '0';
            end case;
      end if;
    if rd='0' and ce ='0' then
    sel_counter <= ad;
    else
    sel_counter <= "11111";
      end if;
 end process;
end;

--
**************************************************************************
******
-- Mux10.vhd
--
**************************************************************************
******
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.NUMERIC_STD.all;
entity mux5 is
      port(a_L,b_L,c_L,d_L,e_L: in std_logic_vector(15 downto 0);
            a_U,b_U,c_U,d_U,e_U: in std_logic_vector(15 downto 0);
            counter_out : out std_logic_vector(15 downto 0);
            tri_enable : out std_logic:='0';
         en_cntr_out : in std_logic_vector (4 downto 0));
end;
architecture behavior of mux5 is
begin
  process (a_L,b_L,c_L,d_L,e_L,a_U,b_U,c_U,d_U,e_U,en_cntr_out)
           begin
                       case en_cntr_out is
                       --when "00000" => counter_out <= a_L;
                       --                              tri_enable <=
'1';
                       --when "00010" => counter_out <= b_L;
                       --                              tri_enable <=
'1';
                       --when "00100" => counter_out <= c_L;
                       --                              tri_enable <=
'1';
                       --when "00110" => counter_out <= d_L;
                       --                              tri_enable <=
'1';
                       --when "01000" => counter_out <= e_L;
                       --                              tri_enable <=
'1';
                       --when "00001" => counter_out <= a_U;
```

```
                        --                                  tri_enable <=
'1';
                        --when "00011" => counter_out <= b_U;
                        --                                  tri_enable <=
'1';
                        --when "00101" => counter_out <= c_U;
                        --                                  tri_enable <=
'1';
                        --when "00111" => counter_out <= d_U;
                        --                                  tri_enable <=
'1';
                        --when "01001" => counter_out <= e_U;
                        --                                  tri_enable <=
'1';
                        when "01010" => counter_out <= a_L;
                                                            tri_enable <=
'1';
                        when "01100" => counter_out <= b_L;
                                                            tri_enable <=
'1';
                        when "01110" => counter_out <= c_L;
                                                            tri_enable <=
'1';
                        when "10000" => counter_out <= d_L;
                                                            tri_enable <=
'1';
                        when "10010" => counter_out <= e_L;
                                                            tri_enable <=
'1';
                        when "01011" => counter_out <= a_U;
                                                            tri_enable <=
'1';
                        when "01101" => counter_out <= b_U;
                                                            tri_enable <=
'1';
                        when "01111" => counter_out <= c_U;
                                                            tri_enable <=
'1';
                        when "10001" => counter_out <= d_U;
                                                            tri_enable <=
'1';
                        when "10011" => counter_out <= e_U;
                                                            tri_enable <=
'1';
                        when others => counter_out <= x"0000";
                                                            tri_enable <=
'0';
                end case;
            end process;
end;

--
*********************************************************************
******
-- Latch.vhd
```

```
--
**************************************************************************
******
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity latch is
  port(ale,bhe: in std_logic;
       ad_in: in std_logic_vector (4 downto 0);
       AD: out std_logic_vector (4 downto 0));
end;
architecture behavior of latch is
begin
process (ale,bhe)
  begin
  if bhe = '0' then
    if (ale = '1' and ale'event) then
       AD <= ad_in;
    end if;
  end if;
  end process;
end;

--
**************************************************************************
******
-- Counter_10.vhd
--
**************************************************************************
******
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.NUMERIC_STD.all;
-- Project OSL, 32bit counter
  entity Counter_32bit_x5 is
  port (en,rst,clk_a,clk_b,clk_c,clk_d,clk_e: in std_logic;
            a_L,b_L,c_L,d_L,e_L: out std_logic_vector(15 downto 0);
            a_U,b_U,c_U,d_U,e_U: out std_logic_vector(15 downto 0));
end entity Counter_32bit_x5;
-- insert after entity declaration
architecture behavior of Counter_32bit_x5 is
   signal count_1L,count_2L,count_3L,count_4L,count_5L: unsigned(15
downto 0);
   signal count_1U,count_2U,count_3U,count_4U,count_5U: unsigned(15
downto 0);
begin
  process (rst,en,clk_a,clk_b,clk_c,clk_d,clk_e)
  begin
    if rst='1' then
      count_1L <= x"0000";
      count_1U <= x"0000";
    elsif (clk_a='1' and clk_a'event)then
      if en='1' then
        count_1L <= count_1L + "1";
            if count_1L = x"ffff" then
            count_1U <= count_1U + "1";
            end if;
```

```
        end if;
    end if;
    if rst='1' then
      count_2L <= x"0000";
      count_2U <= x"0000";
    elsif (clk_b='1' and clk_b'event)then
      if en='1' then
      count_2L <= count_2L + "1";
            if count_2L = x"ffff" then
            count_2U <= count_2U + "1";
            end if;
        end if;
    end if;
    if rst='1' then
      count_3L <= x"0000";
      count_3U <= x"0000";
    elsif (clk_c='1' and clk_c'event)then
      if en='1' then
      count_3L <= count_3L + "1";
            if count_3L = x"ffff" then
            count_3U <= count_3U + "1";
            end if;
        end if;
    end if;
    if rst='1' then
      count_4L <= x"0000";
      count_4U <= x"0000";
    elsif (clk_d='1' and clk_d'event)then
      if en='1' then
      count_4L <= count_4L + "1";
            if count_4L = x"ffff" then
            count_4U <= count_4U + "1";
            end if;
        end if;
       end if;
    if rst='1' then
      count_5L <= x"0000";
      count_5U <= x"0000";
    elsif (clk_e='1' and clk_e'event)then
      if en='1' then
      count_5L <= count_5L + "1";
            if count_5L = x"ffff" then
            count_5U <= count_5U + "1";
            end if;
        end if;
    end if;
end process;
        a_L <= std_logic_vector(count_1L);
        b_L <= std_logic_vector(count_2L);
        c_L <= std_logic_vector(count_3L);
        d_L <= std_logic_vector(count_4L);
        e_L <= std_logic_vector(count_5L);
        a_U <= std_logic_vector(count_1U);
        b_U <= std_logic_vector(count_2U);
        c_U <= std_logic_vector(count_3U);
        d_U <= std_logic_vector(count_4U);
        e_U <= std_logic_vector(count_5U);
```

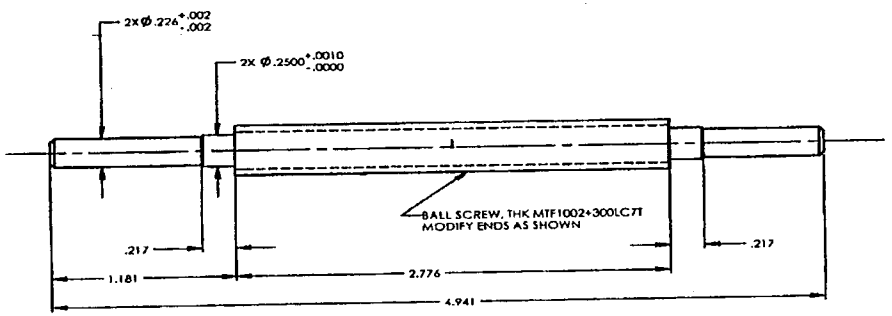
```
end behavior;

************************************************************************
******
--          END OF Altera Field Programmable Gate Array Source Code
--
************************************************************************
******


************************************************************************
******
--          END OF OSL Image Reader Source Code Listing
--
************************************************************************
******
```

APPENDIX F

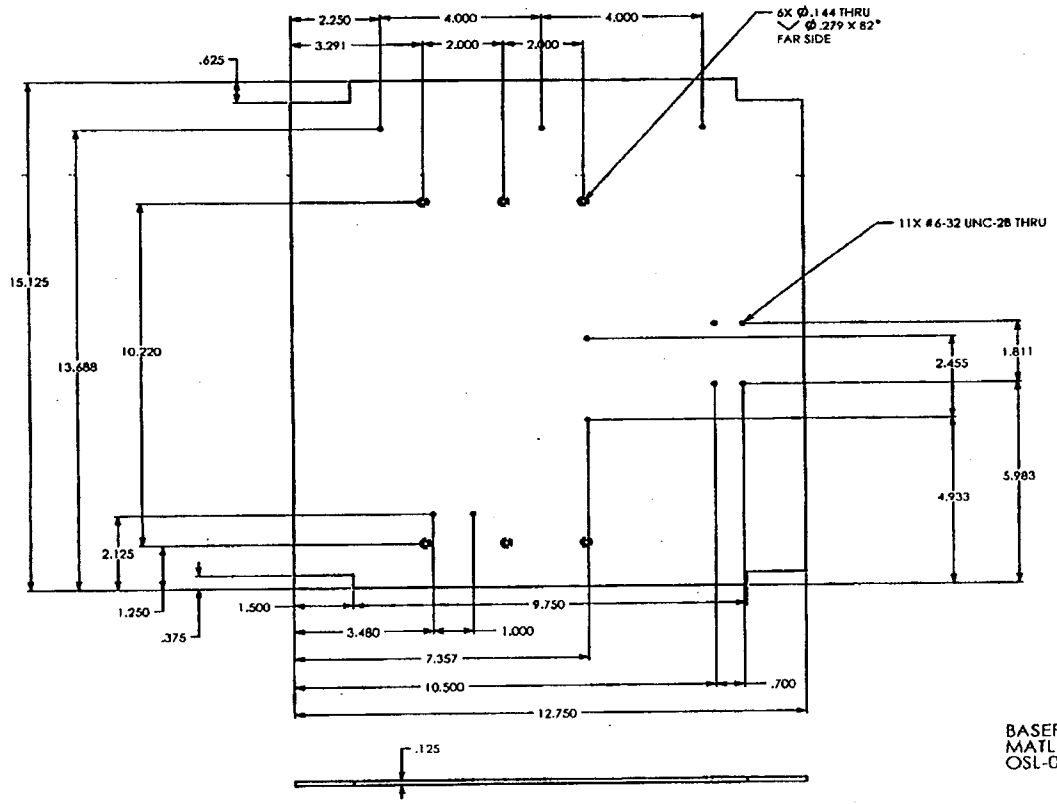
ENGINEERING DRAWINGS

APPENDIX F

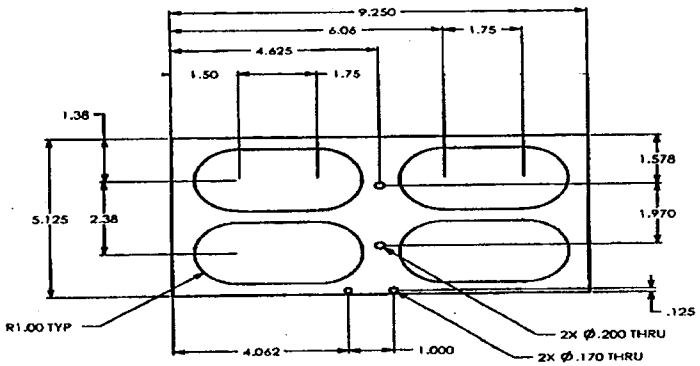
ENGINEERING DRAWINGS

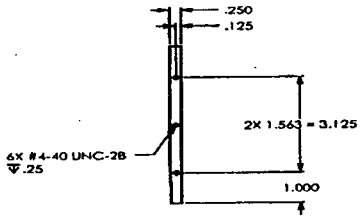


BALLSCREW
OSL-02

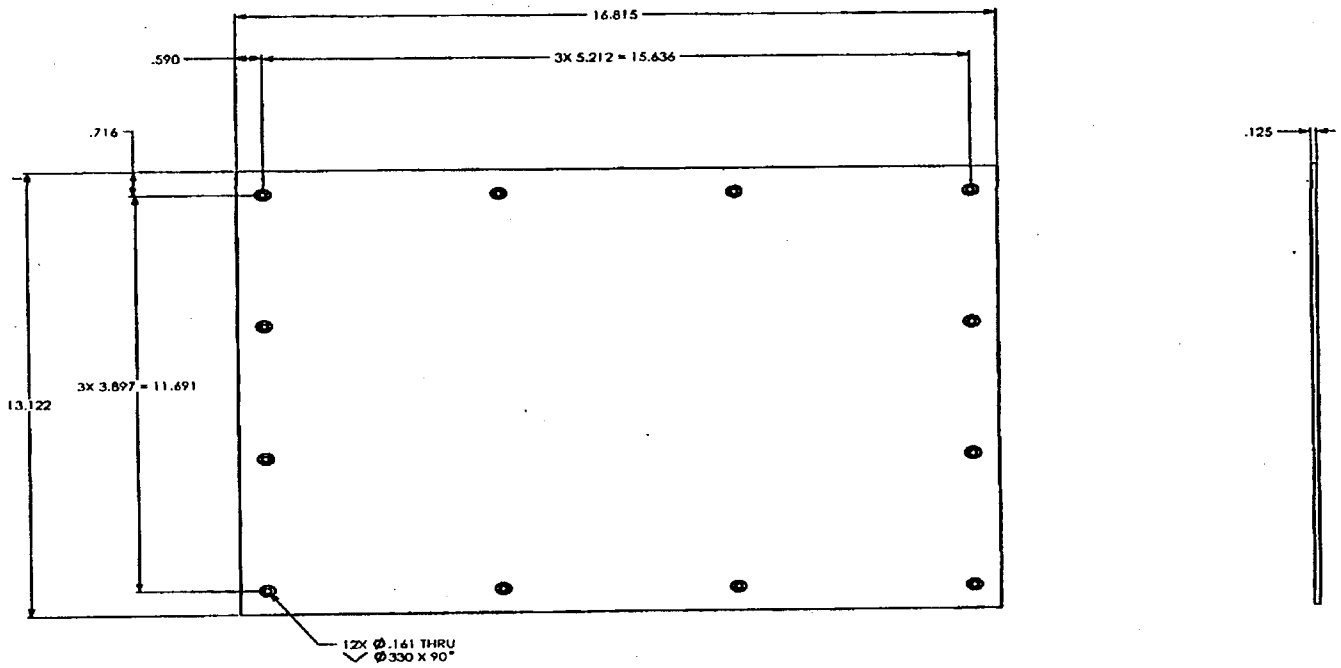


BASEPLATE
MATL: ALUMINUM
OSL-03

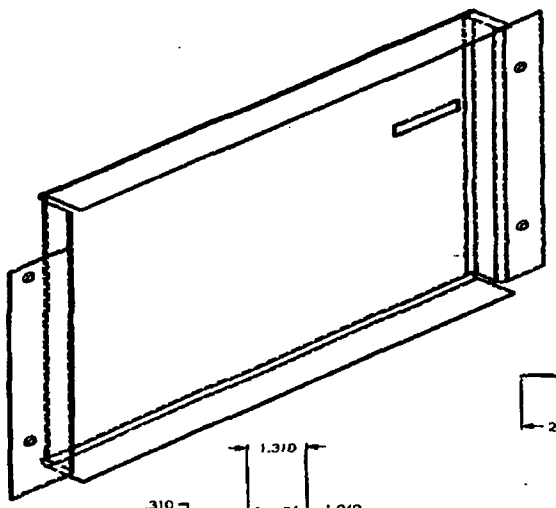


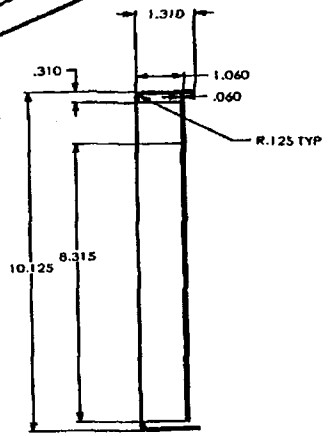


BOTTOM MOUNTING PLATE
MATL: ALUMINUM
OSL-06

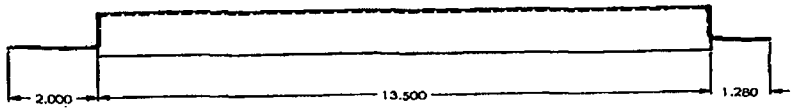


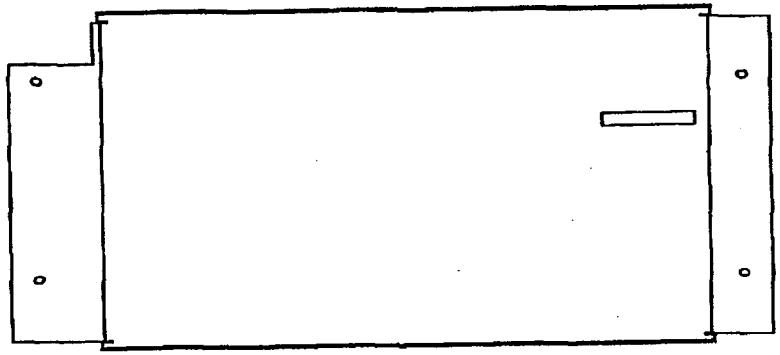
OUTER BOTTOM PLATE
MATL: ALUMINUM
OSL-07

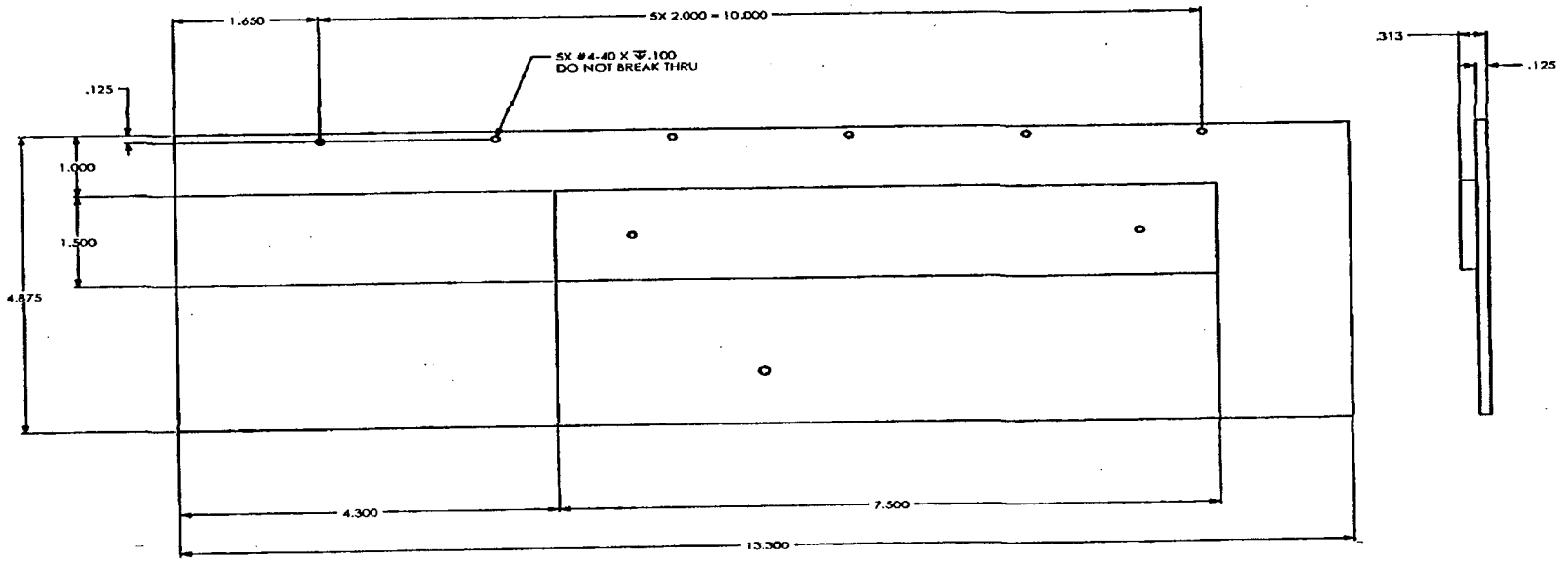




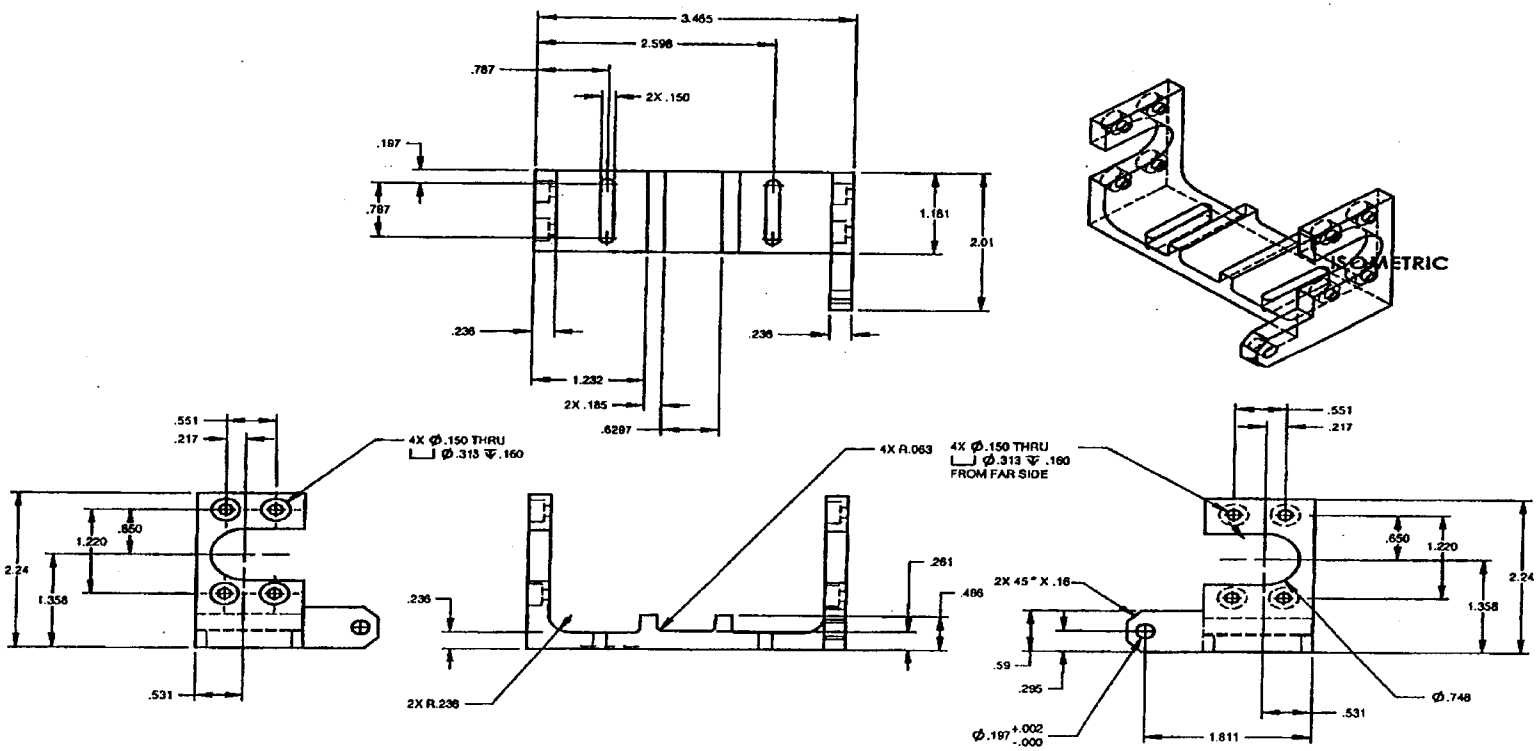
COMPUTER COVER
MATL: .06 ALUMINUM
OSL-09



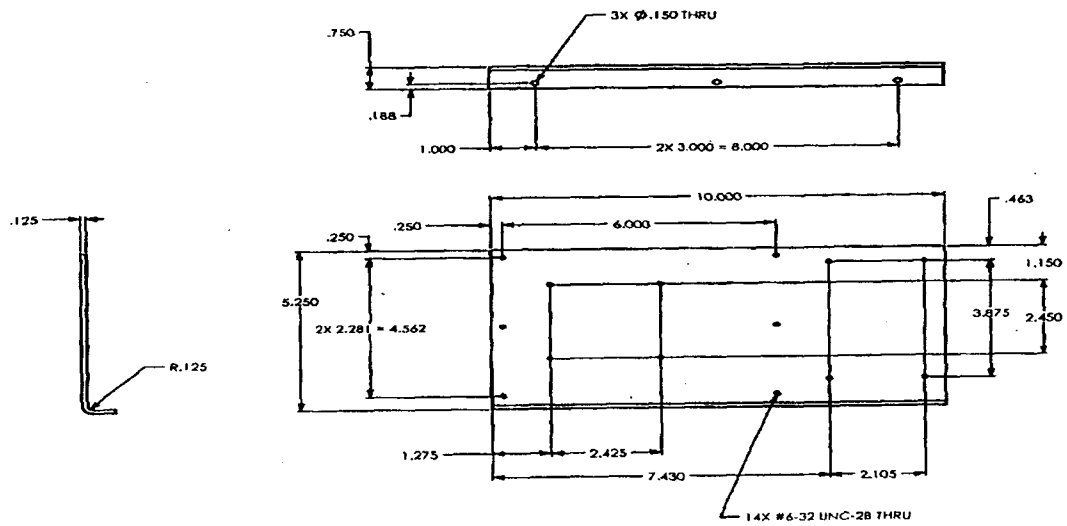




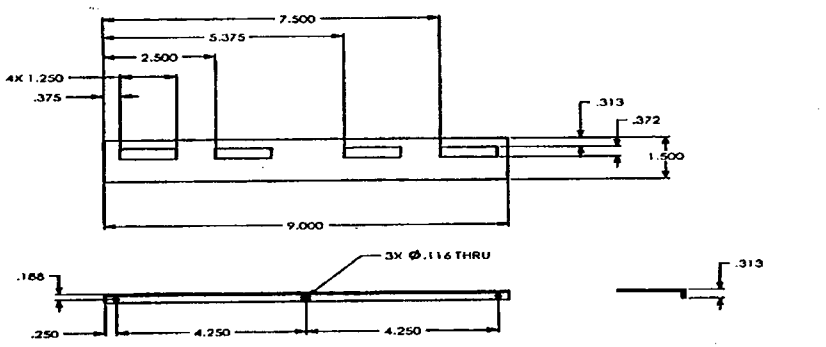
DOOR
MATL: ALUMINUM
OSL-10



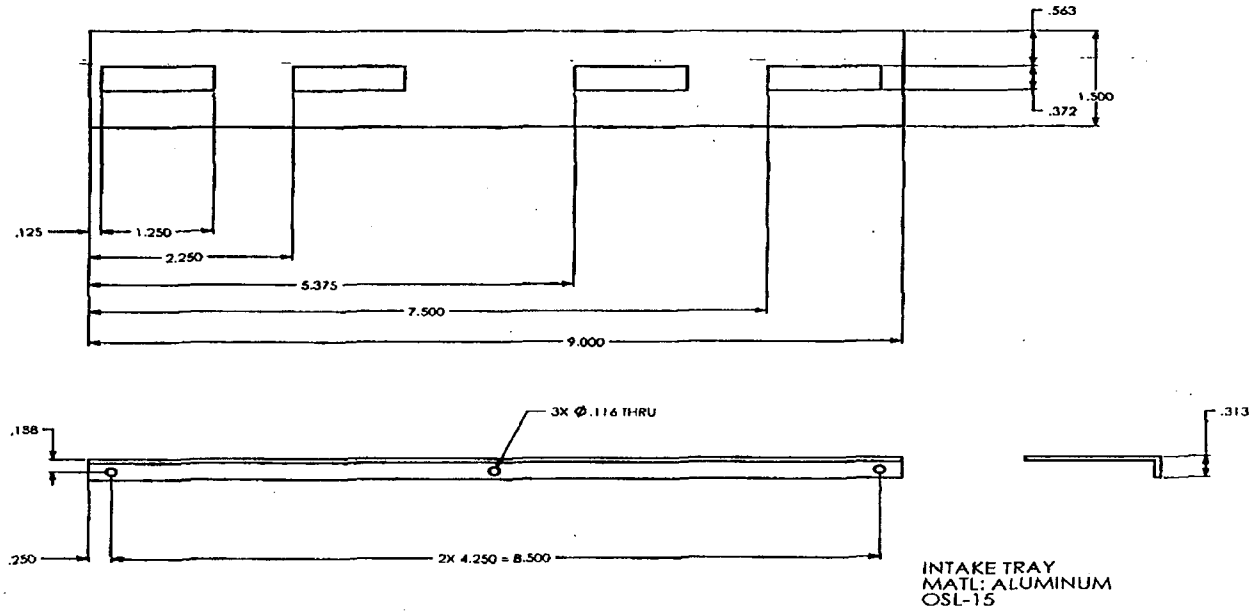
DRIVE SUPPORT
MATL: ALUMINUM
OSL-11



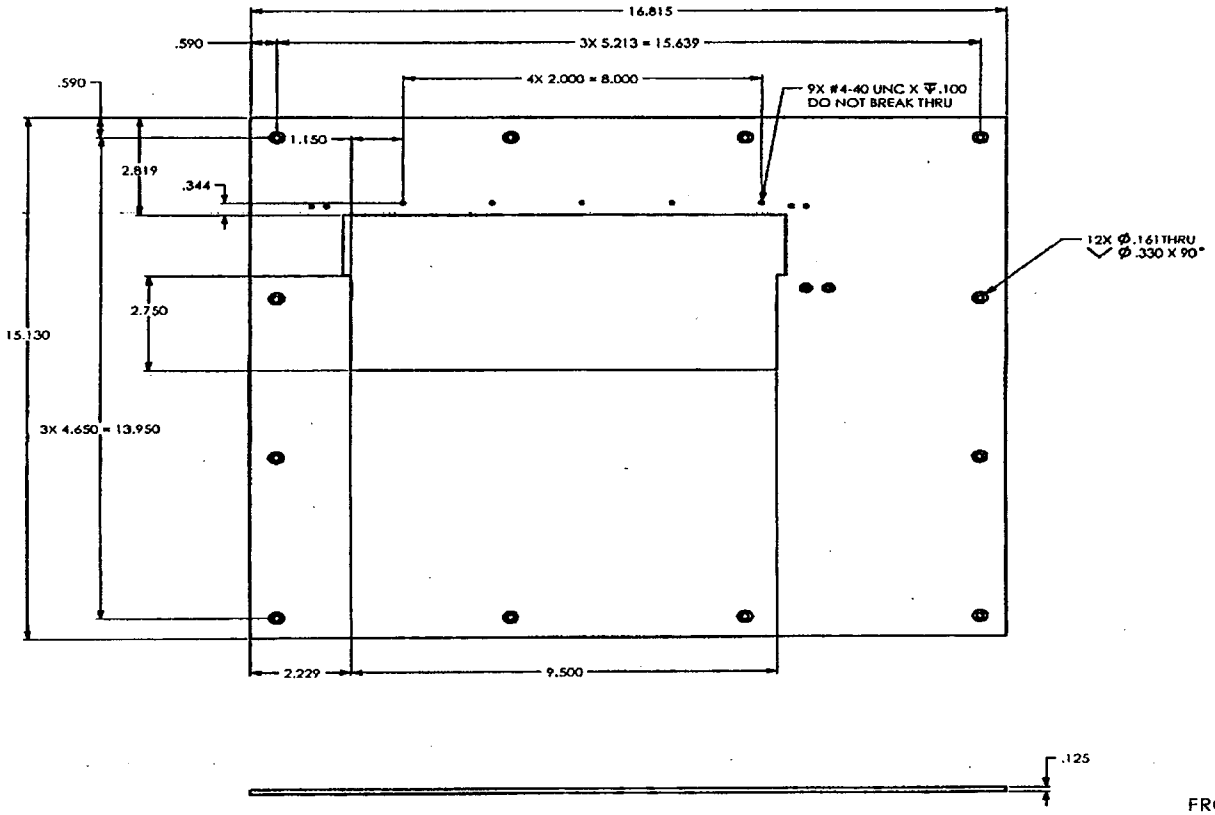
ELECTRONIC SUPPORT PLATE
MATL: ALUMINUM
OSL-12



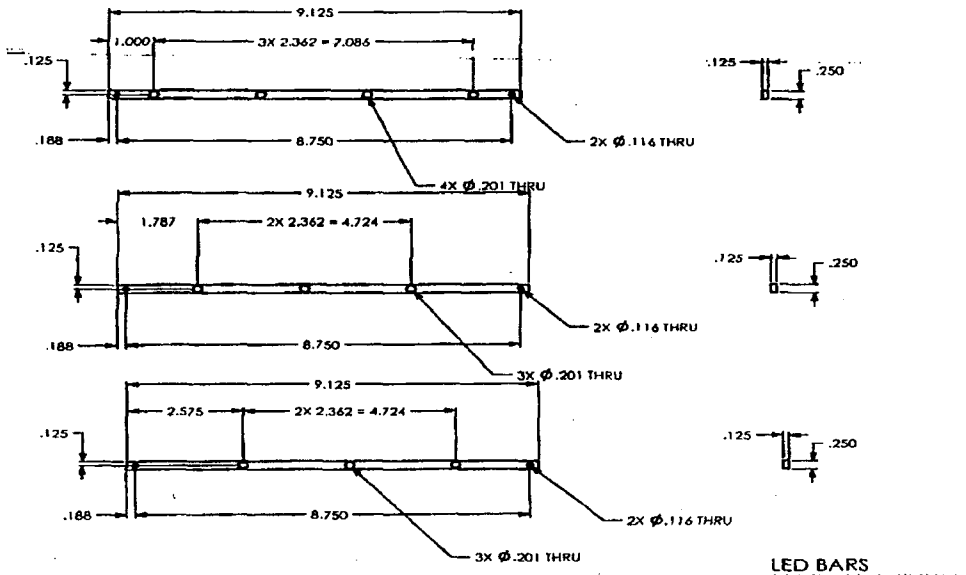
INTAKE TRAY 2
MATL: ALUMINUM
OSL-14



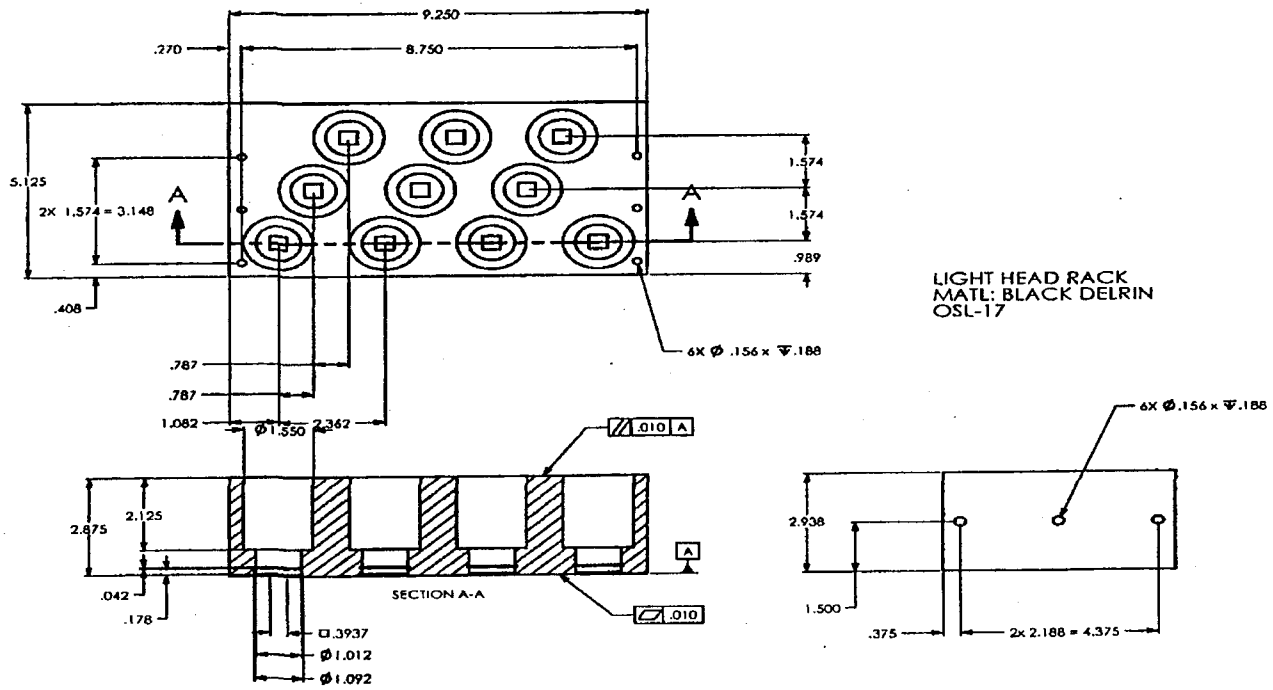
INTAKE TRAY
MATL: ALUMINUM
OSL-15

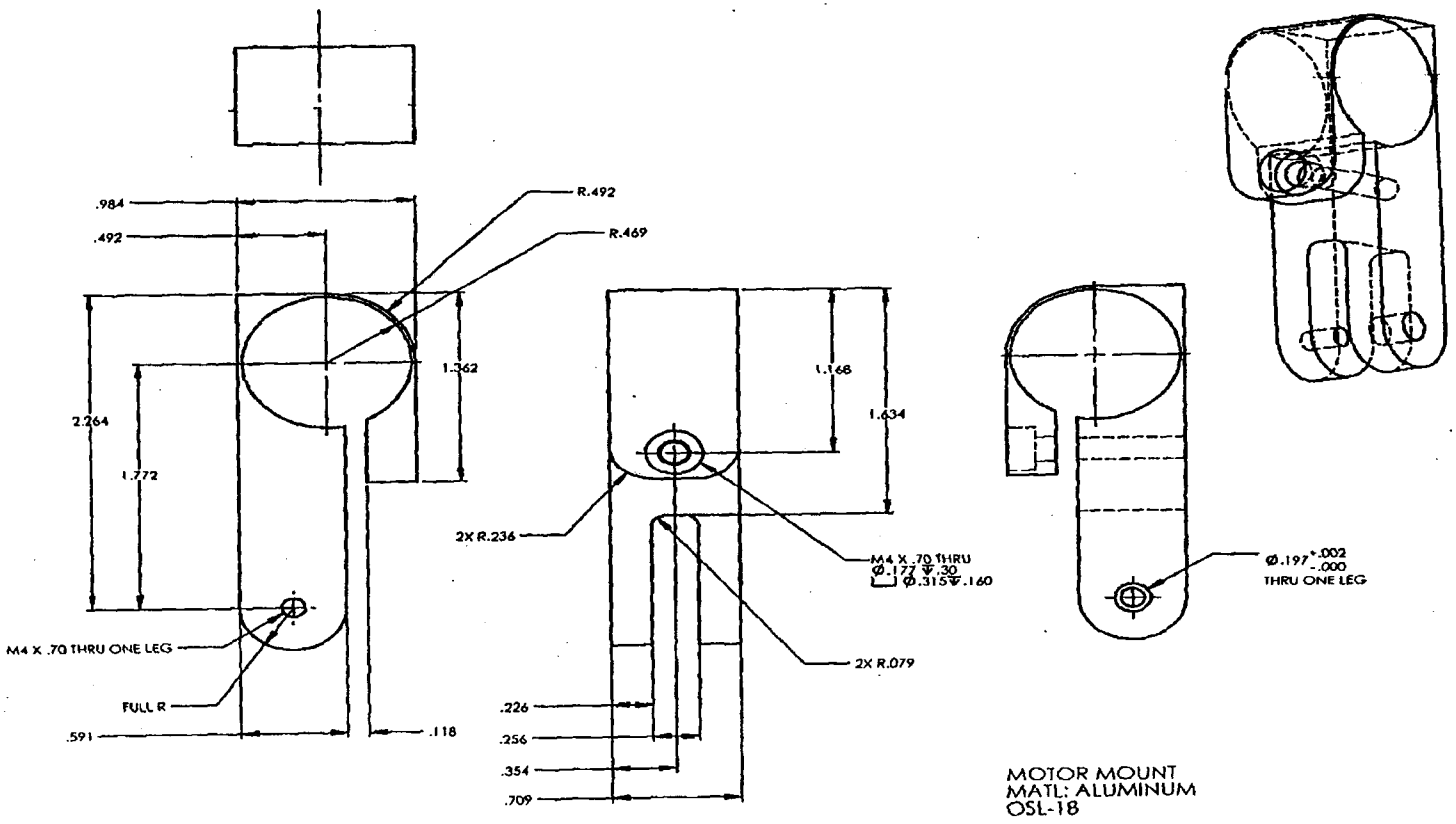


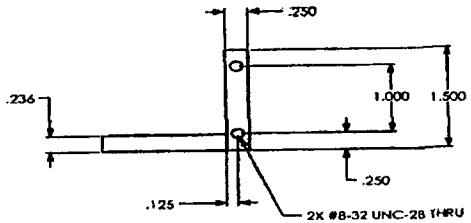
FRONTPLATE
MATL: ALUMINUM
OSL-13

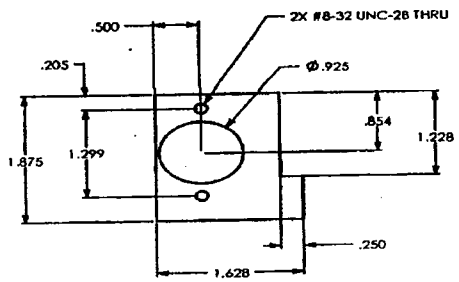


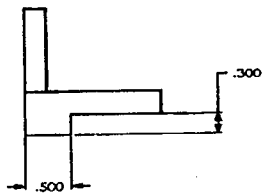
LED BARS
MATL: ALUMINUM
OSL-16


9.250
.270
8.750
5.125
2X 1.574 = 3.148
A
A
1.574
1.574
.989
.408
6X Ø .156 x ↧.188
.787
.787
1.082
Ø1.550
2.362
.010 A
2.875
2.125
A
.042
.178
SECTION A-A
.010
□.3937
Ø1.012
Ø1.092
LIGHT HEAD RACK
MATL: BLACK DELRIN
OSL-17
6X Ø.156 x ↧.188
2.938
1.500
.375
2x 2.188 = 4.375

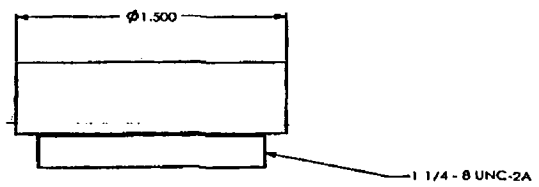

.984
.492
R.492
R.469
1.362
2.264
1.772
M4 X .70 THRU ONE LEG
FULL R
.591
.118
1.168
1.634
2X R.236
M4 X .70 THRU
Ø.177 ↧.30
Ø.315 ↧.160
2X R.079
.226
.256
.354
.709
Ø.197 +.002 -.000
THRU ONE LEG
MOTOR MOUNT
MATL: ALUMINUM
OSL-18

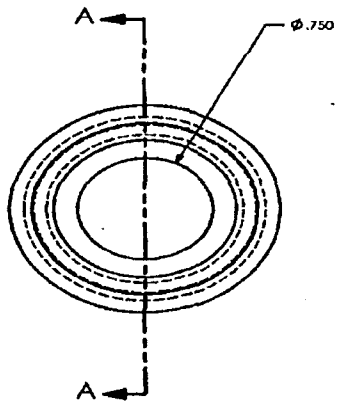


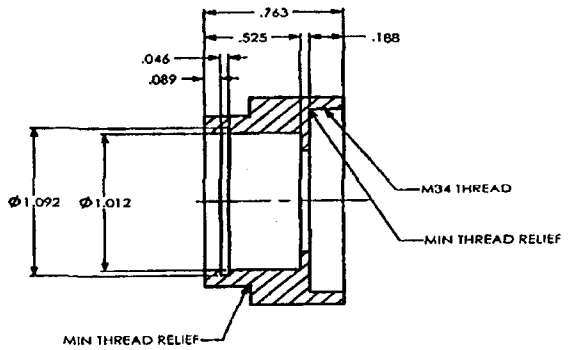




MOVEMENT PLATE
MATL: ALUMINUM
OSL-19

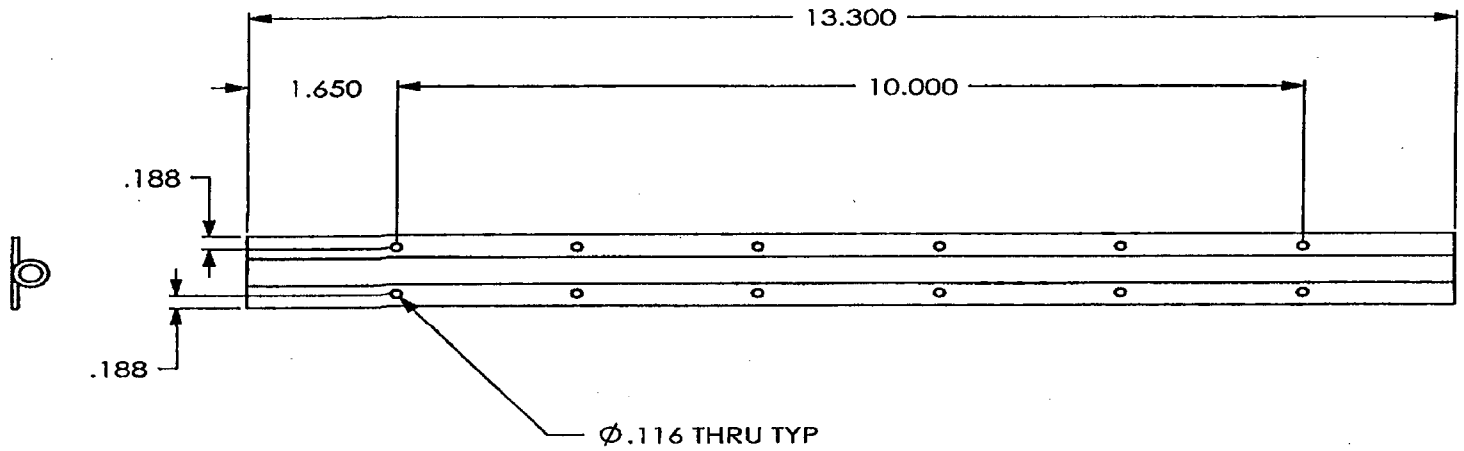




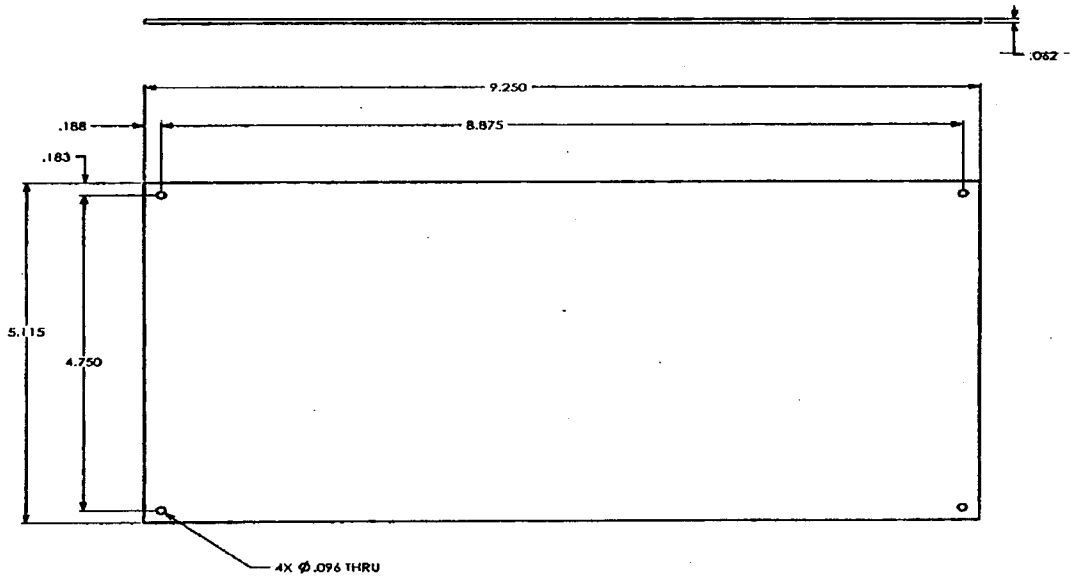


SECTION A-A
SCALE 2 : 1

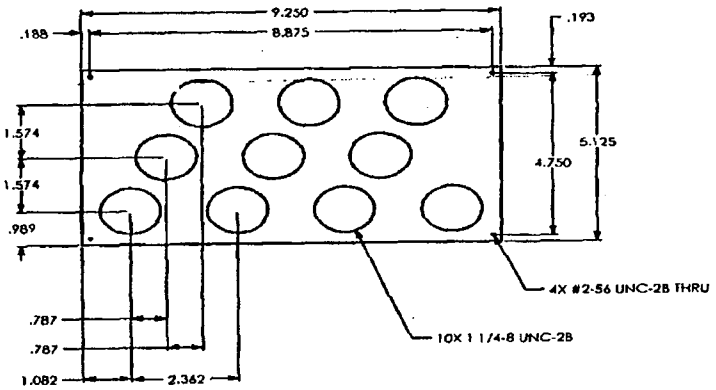
FILTER MOUNT
MATL: ALUMINUM
OSL-20

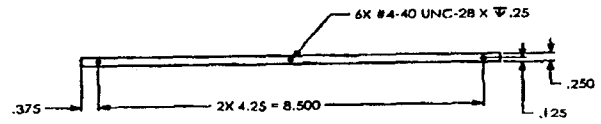


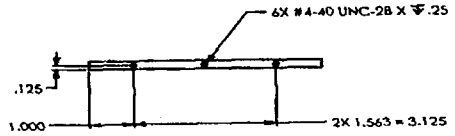
PIANO HINGE
OSL-21



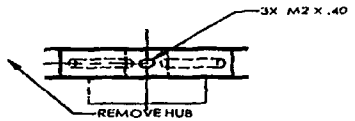
PLASTIC COVER
OSL-22

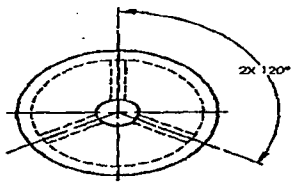




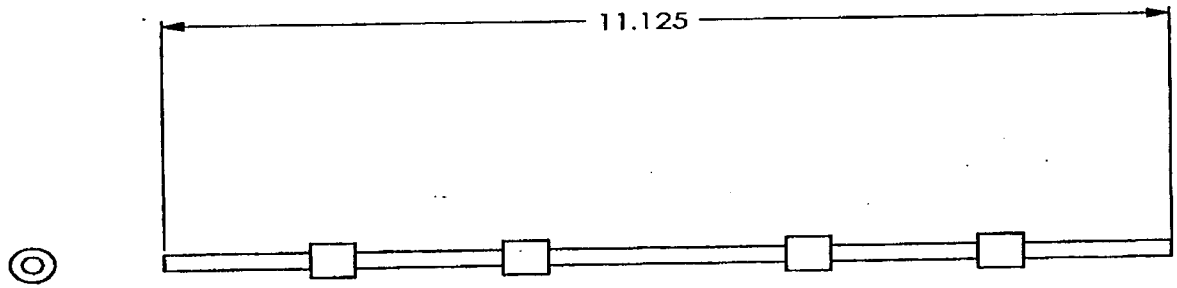


PMT MOUNTING PLATE
MATL: ALUMINUM
OSL-23

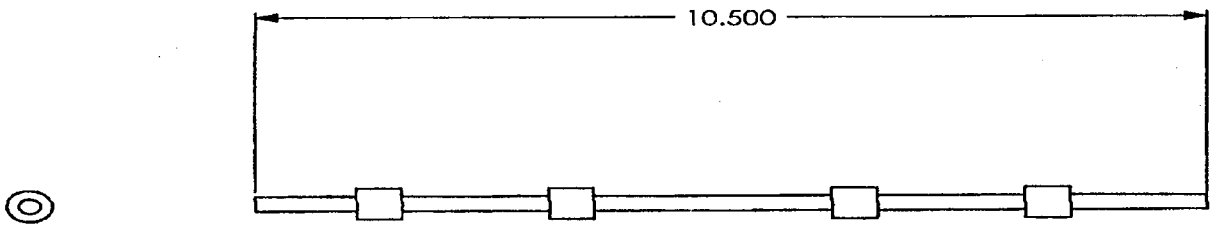




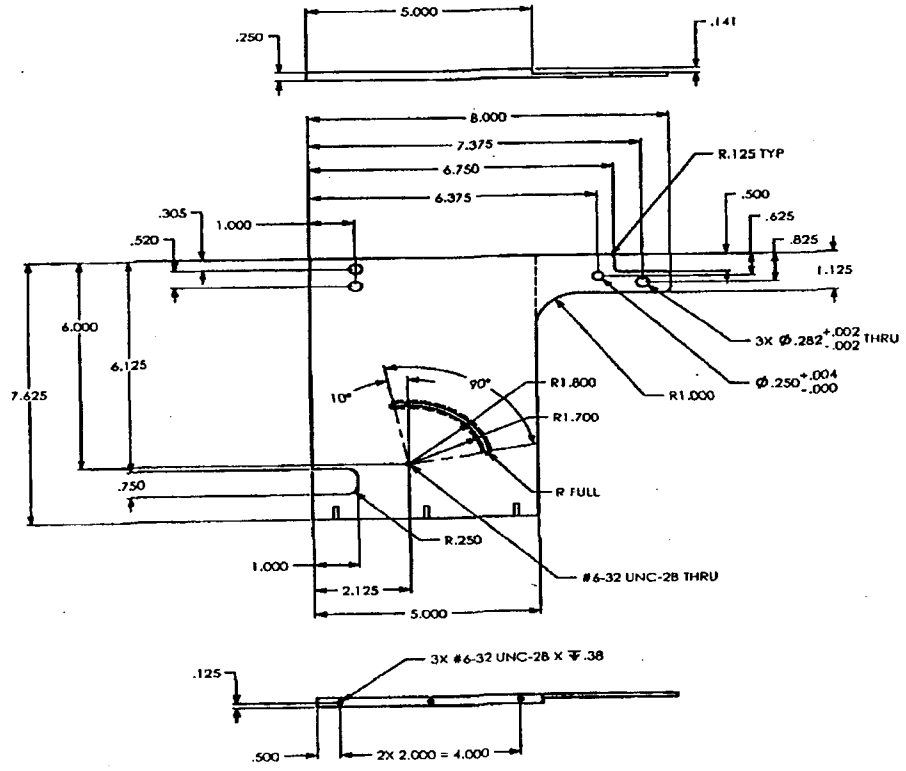
PULLEY
OSL-24

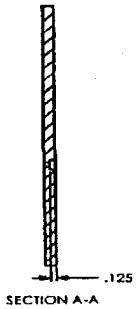


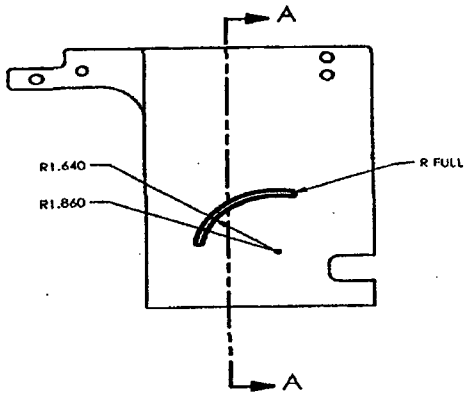
ROLLER 1
OSL-25



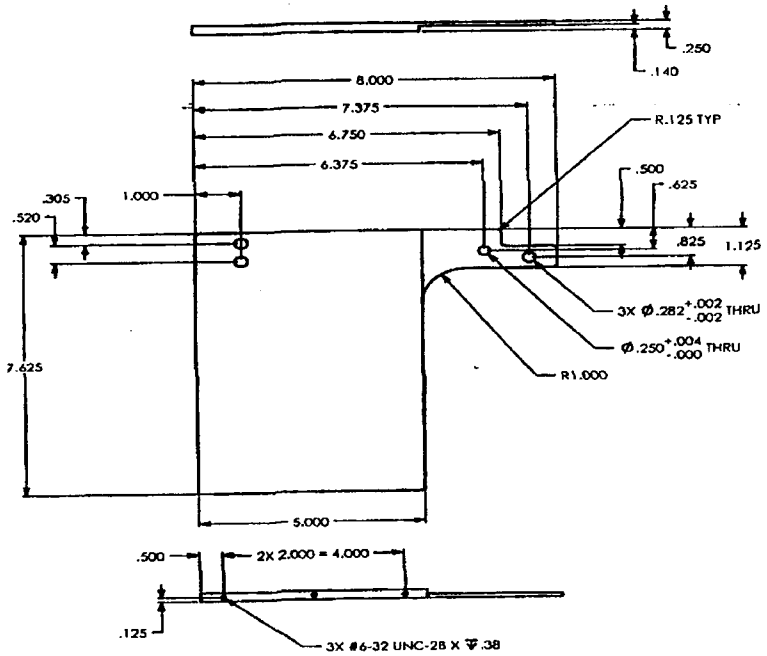
ROLLER 2
OSL-26



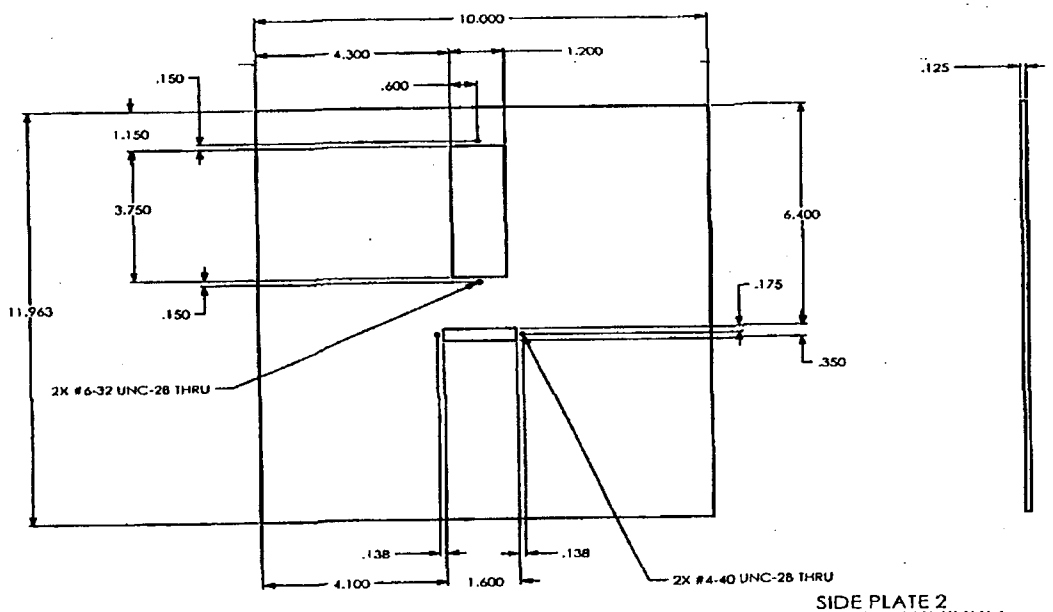




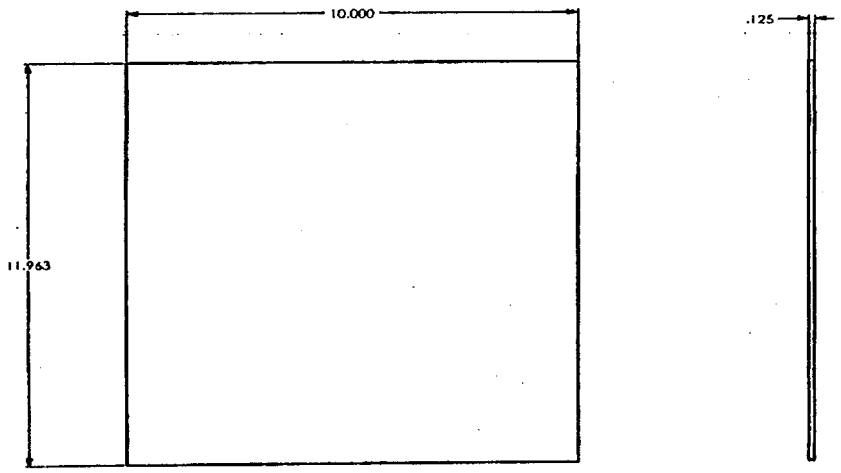
ROLLER SUPPORT
MATL: ALUMINUM
OSL-27



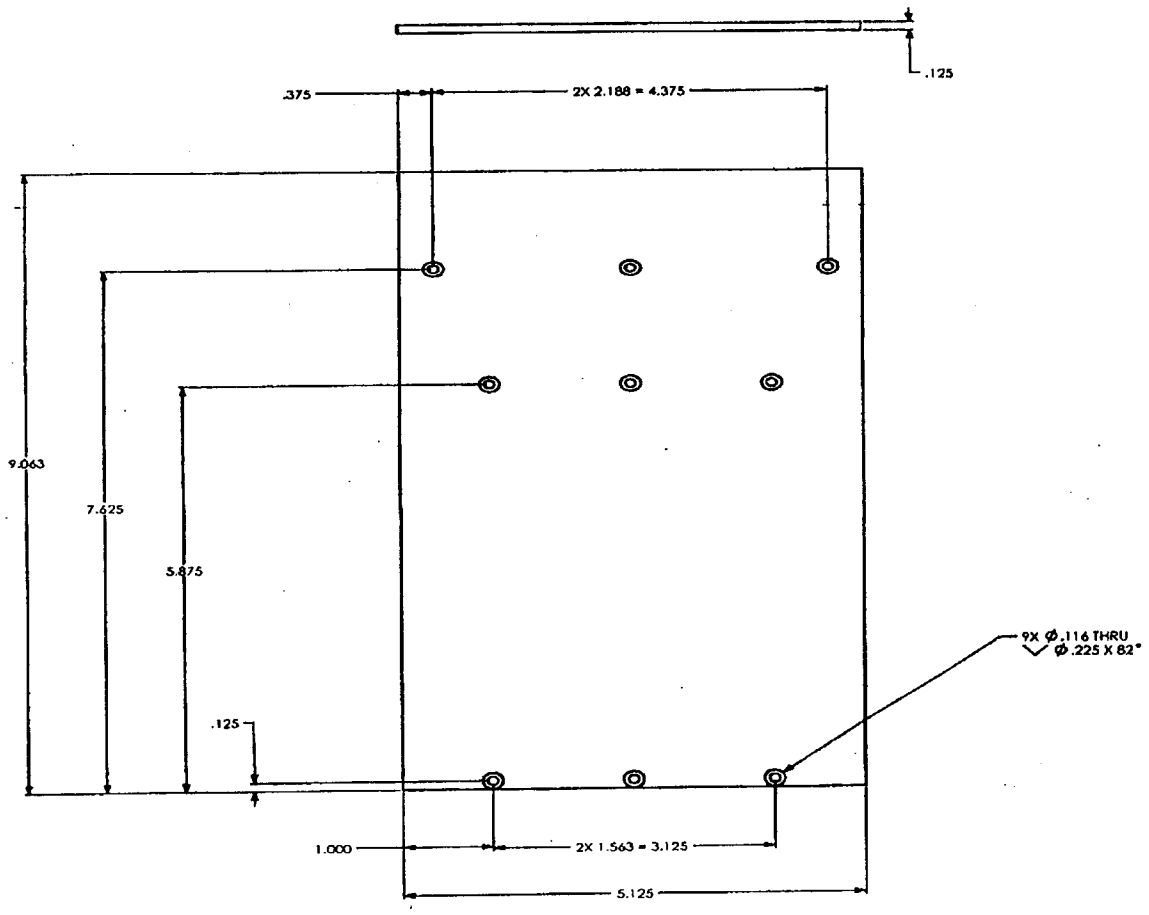
ROLLER SUPPORT 2
MATL: ALUMINUM
OSL-28



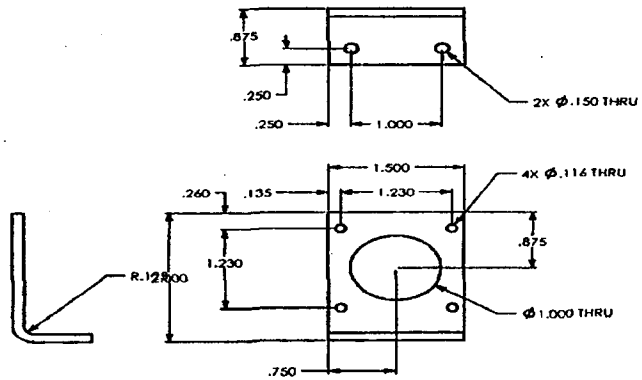
SIDE PLATE 2
MATL: ALUMINUM
OSL-29



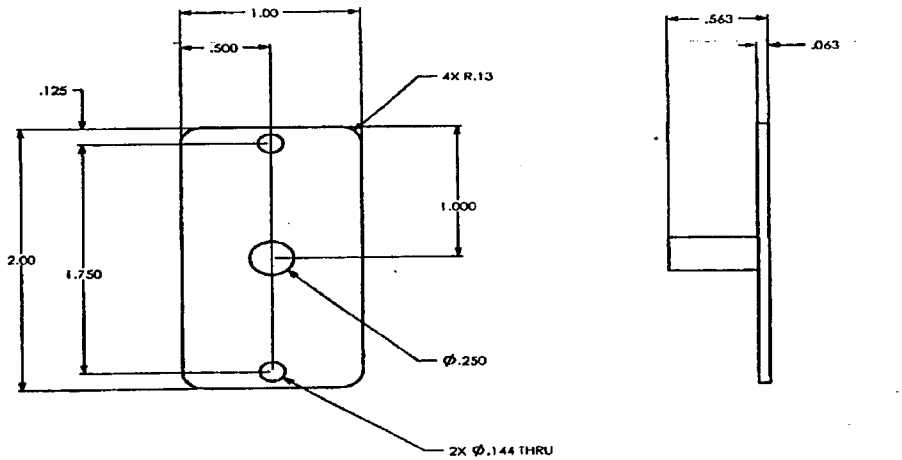
SIDE PLATES
MATL: ALUMINUM
OSL-30



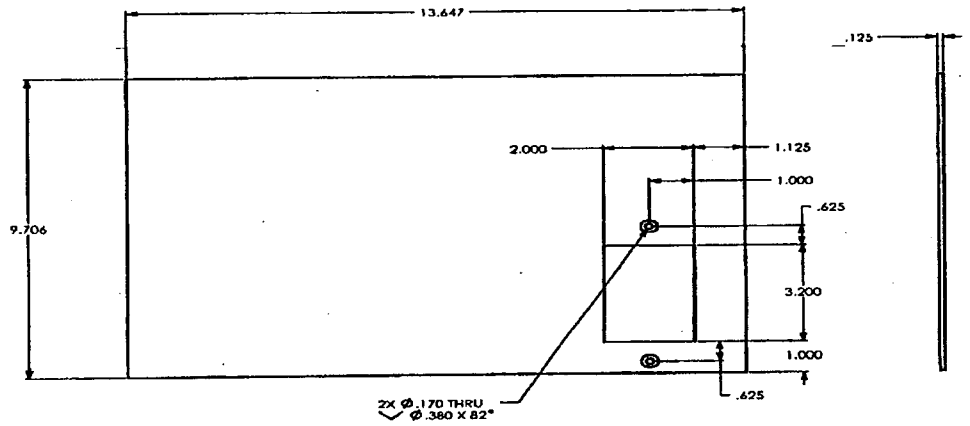
SIDEWALL
MATL: ALUMINUM
OSL-31



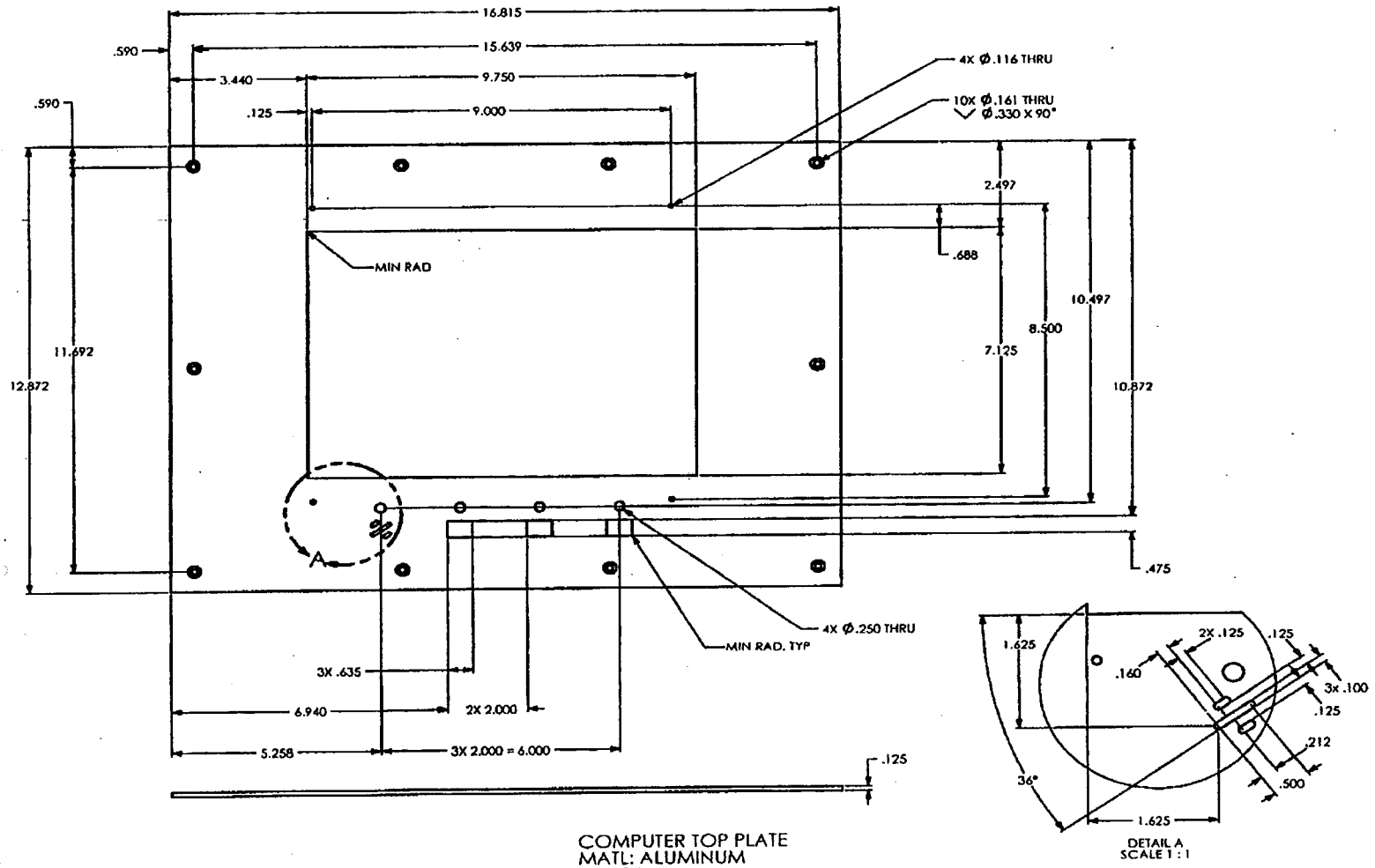
STEPPER MOTOR SUPPORT
MATL: ALUMINUM
OSL-32



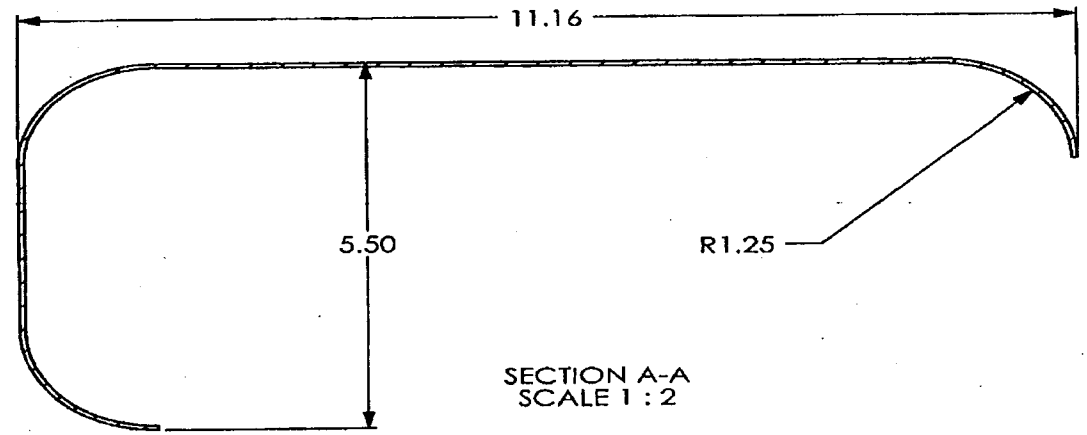
TENSION PLATE
MATL: ALUMINUM
OSL-33



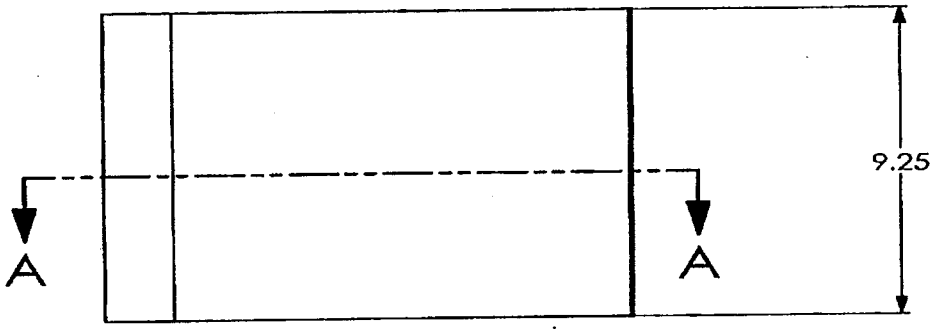
TOP PLATE
MATL: ALUMINUM
OSL-34



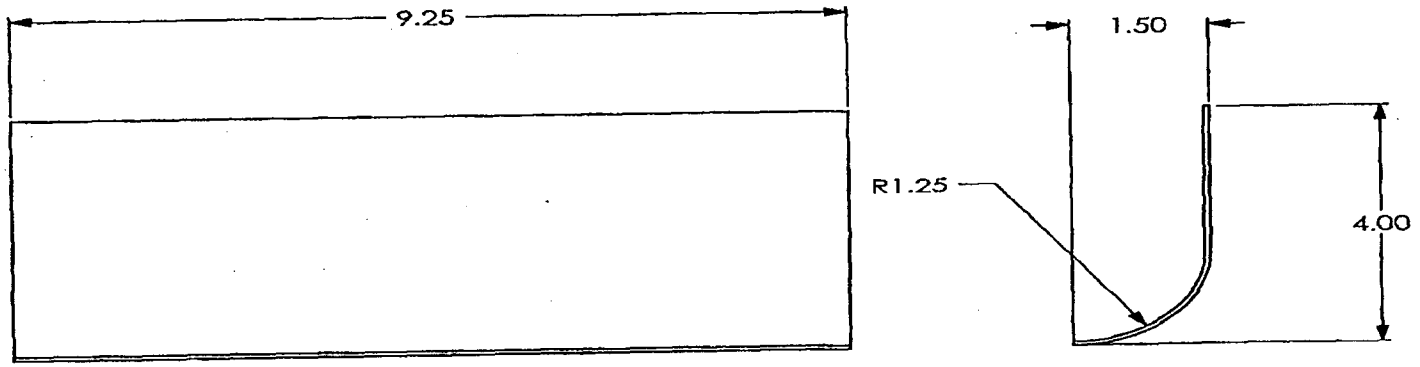
COMPUTER TOP PLATE
MATL: ALUMINUM
OSL-34



SECTION A-A
SCALE 1 : 2



TRACK 1
MATL: ALUMINUM
OSL-35



TRACK 2
MATL: ALUMINUM
OSL-36

DISTRIBUTION LIST -

DOE/NN - 20
David Spears
Michael O'Connell

DOE/NN - 40
Michael Newman

DTRA/OST
Maj John Anton

PNNL
Gordon Dudder (10)
Hal Undem
Jennifer Tanner
Anthony Peurrung
James Fuller
John Smoot
Chuck Batishko
Ron Hockey