NOAA, NATIONAL WEATHER SERVICE, CENTRAL REGION

COMPUTER PROGRAMS AND PROBLEMS

NWS CRCP - NO. 9

ASSEMBLY LANGUAGE GRAPHICS LIBRARY (EGR1.LB) WITH FORTRAN INTERFACING

Thomas J. Egger
National Weather Service Office
Marquette, Michigan

July 1983

U.S. DEPARTMENT OF COMMERCE / National Oceanic and Atmospheric Administration / National Weather Service

NOAA, National Weather Service
Central Region Computer Programs and Problems Series

NWS CRCP No. 1 A Computer Program for Determining and Displaying the Geostrophic Winds on Lakes Superior and Michigan. R. Somrek. June 1980 (PB80 225675)

NWS CRCP No. 2 Program ALEMBIC. W. Sunkel. March 1982 (PB83 114488)

NWS CRCP No. 3 A Subroutine to Find and Extract Data from an AFOS Database Product. R. Somrek. May 1982 (PB83 118828)

NWS CRCP No. 4 Programs SAVALL and STORALL. W. Sunkel. July 1982. (PB83 118448)

NWS CRCP No. 5 Program ANALYZ. T. Schwein. Sept. 1982

NWS CRCP No. 6 A Regional Weather Depiction Plot. W. Sunkel. October 1982.

NWS CRCP No. 7 The Topeka Library (TOP.LB). W. Sunkel.. March 1983

NWS CRCP No. 8 A Multipurpose Weather Roundup Program. Warren E. Sunkel May 1983.

NOAA, NATIONAL WEATHER SERVICE, CENTRAL REGION
Computer Programs and Problems

NWS CRCP - No. 9

ASSEMBLY LANGUAGE GRAPHICS LIBRARY (EGR1.LB) WITH FORTRAN INTERFACING

Thomas J. Egger
National Weather Service Office
Marquette, Michigan

July 1983

UNITED STATES DEPARTMENT OF COMMERCE
Malcolm Baldrige, Secretary

National Oceanic and Atmospheric Administration
John V. Byrne, Administrator

National Weather Service
Richard E. Hallgren, Director

ASSEMBLY LANGUAGE GRAPHICS LIBRARY
(EGR1.LB)
WITH FORTRAN INTERFACING

Thomas J. Egger
National Weather Service Office
Marquette, Michigan

INTRODUCTION

The Assembly Language Graphics Library, EGR1.LB, is a series of subroutines that enables the programmer to generate graphic products using assembly language or FORTRAN. The library was designed to: (1) have faster routines than existing routines, (2) lessen core requirements, and (3) reduce disk space requirements. All of these goals could be obtained by the use of assembly language; it is much closer to the binary language the computer functions with, and therefore operates with great efficiency.

This is a dual-purpose library. Programs can be constructed using assembly language or FORTRAN. If assembly language is chosen, then the following routines can be selected: ELINE, ETXT, ECS, ECSR, ELEG, or ECLR. If the chosen language is FORTRAN, then the interfaces EFLINE, EFSLINE, EFTXT, EFSTXT, EFCS, EFCSR, EFLEG or EFCLR can be used.

Versatility was built-in to each subroutine. The programmer can select from a variety of display options including four different character formats, selective erasing, multiple zoom ratios; and the choice of console, screen and overlay.

All of the graphic routines in EGR1.LB are on-line. The subroutines cause immediate display on the graphic screen. No RDOS disk files are created.

The first version of the library is labeled EGR1.LB. Later versions will have a higher embedded numeral. Each subroutine is prefixed with the letter E to indicate its relationship to the library. If the second letter of the module name is F, then that module is an interface to FORTRAN. The only exception is EBIND.

LOADING

The EGR1.LB uses system calls. The system library, SYS.LB, must be available for linking. It is not necessary, however, to mention SYS.LB in the load line.

A typical load line for an assembly language program would appear as follows:

```
RLDR MAINPROGRAM SUBPROGRAMS EGR1.LB
```

A typical load line for a FORTRAN program appears as follows:

```
RLDR MAINPRGRAM SUBPROGRAMS EGR1.LB UTIL.LB FORT.LB
```

ERROR RETURNS

No error returns are offered. It was impractical to develop error returns because of the large number of input parameters.

FORMAT

The first section of this document is devoted to assembly language applications. Each subroutine is described in detail. The source listings follow each description. The second section describes the FORTRAN interfaces in detail. Source codes follow the descriptions. Examples of two complete assembly language programs are contined in Appendix 1. An example FORTRAN program is included in Appendix 2. These examples demonstrate the use of each subroutine.

The routines were assembled using Data General's Extended Assembler, ASM.SV. The assembler employed the symbol table ESYM.PS. A copy of the symbol table is enclosed in Appendix 3.

MEMORY REQUIREMENTS

Memory required by each subroutine is given in the listing by the ZREL value (the page zero relocatable memory), the NREL value (normally relocatable memory) and by the number of STACK variables. The values in the listing are given in octal (16-bit) words. It should be noted that the number of stack variables are assigned for the duration only that the routine is active. NREL and ZREL are values used during the complete program execution.

GRAPHIC COORDINATE DEFINITIONS

The image space used by all EGR1.LB routines is defined as 4096 points on the X-axis and 3072 points on the Y-axis. Acceptable values for X range from 0 to 4095, and for Y range from 0 to 3071.

ACKNOWLEDGMENTS

Some of the graphic routines in the EGR1.LB were developed with guidance from routines written in FORTRAN by the Western Region SSD.

Many people in different ways assisted the author of this library. I would like to single out in particular my professors at Northern Michigan University, Robert Helton and Terrence Seethoff, for igniting my interest in computer science. I also wish to express my gratitude to Bob van Haaren, Central Region SSD; Bob Somrek, WSFO Chicago; and John Hughes, WSFO Ann Arbor; for keeping that interest alive.

REFERENCES

Detailed discussion of the message formats for the GDM is contained in the engineering manual EHB13, WSO Theory of Operation, published by the Engineering Division, National Weather Service, Silver Spring, Maryland.

Details for graphics creation are described in AFOS Displays Programmers Reference, Ford Aerospace Corporation, WDL-TR7676A, Palo Alto, California.

I. IDENTIFICATION

Module name:	ECLR
Date:	October 23, 1982
Function:	Clear the display
Language:	Assembly
Memory locations:	ZREL 0 NREL 36 STACK 0

II. FUNCTIONAL SUMMARY

ECLR is a subroutine that will clear the display and delete the contents of CLS.

III. ENTRY POINTS

ECLR

IV. CALLING METHOD

Load AC2 with the word pointer to the parameter list; then do an indirect JSR to ECLR.

```
                LDA 2,PAR
                JSR @DUMMY
                  .

DUMMY:  ECLR
PAR:    .+1
        0
        0
        0
        0
```

V. INPUT

AC2 contains a word pointer to the parameter list. The parameter list is of the following design:

```
PAR:    .+1
        0    ;RDOS channel #
        0    ;CON console #
        0    ;SCR screen #
        0    ;OVY overlay #
```

VI. OUTPUT

The designated screen is cleared.

VII. ERROR RETURNS

NONE

VIII. REFERENCED EXTERNALS

NONE

```
  0001  ECLR
                ;*****************************************************************
                ;       ECLR.SR       TOM EGGER/MQT   10-23-82
                ;
                ;       SUBROUTINE TO CLEAR THE GDM SCREEN
                ;       PARAMETERS PASSED AFTER THE CALL:    RDOS CHANNEL #
                ;                                            CONSOLE #
                ;                                            SCREEN #
                ;                                            OVERLAY #
                ;
                ;*****************************************************************

                                    .TITL ECLR
                                    .ENT ECLR
        000001                      .TXTM 1
                                    .NREL

   00000'054435 ECLR:               STA 3,SAV        ;SAVE RETURN ADDRESS
   00001'155000                     MOV 2,3
   00002'021401                     LDA 0,1,3        ;CONSOLE #
 FU00003'000000                     HXL 1,0
   00004'024423                     LDA 1,CLSC       ;CLEAR SCREEN FUNCTION WORD
 FU00005'000000                     IOR 0,1
   00006'021402                     LDA 0,2,3         ;SCREEN #
 FU00007'000000                     HXL 0,0           ;LEFT 4
   00010'101120                     MOVZL 0,0         ;LEFT 1
   00011'101120                     MOVZL 0,0         ;LEFT 1
 FU00012'000000                     IOR 0,1
   00013'021403                     LDA 0,3,3         ;OVERLAY #
 FU00014'000000                     HXL 0,0           ;LEFT 4
 FU00015'000000                     IOR 0,1
   00016'044413                     STA 1,CL          ;STORE THE FINAL WORD
   00017'020411                     LDA 0,CLR         ;ADDRESS OF FINAL WORD
   00020'031400                     LDA 2,0,3         ;RDOS CHANNEL #
   00021'024405                     LDA 1,NB          ;2 BYTES
   00022'006017                     .SYSTM
   00023'016477                     .WRS 77
   00024'000406                     JMP ER
   00025'002410                     JMP @SAV
   00026'000002 NB:    2
   00027'010000 CLSC:  10000
   00030'000062'CLR:   CL*2
   00031'000000 CL:    0
   00032'006017 ER:    .SYSTM
   00033'006400        .ERTN
   00034'000776        JMP ER
   00035'000000 SAV:      0
                                .END

  0002  ECLR
CL      000031'
CLR     000030'
CLSC    000027'
ECLR    000000'
ER      000032'
HXL     000000U
IOR     000000U
NB      000026'
```

```
SAV     000035'
```

I. IDENTIFICATION

Module name:	ECS
Date:	December 1, 1982
Function:	Interrogate cursor without using "enter cursor" button
Language:	Assembly
Memory locations	ZREL 0 NREL 123 Stack 0

II. FUNCTIONAL SUMMARY

ECS is a subroutine that is used to obtain the cursor location without depressing the "enter cursor" button. The construction of this routine differs from ECSR only by changing the .RDC system call to .RDS. The advantage of this routine is that a secondary task does not need to be created. The cursor can be interrogated anywhere in the program without stopping the program.

III. ENTRY POINTS

ECS

IV. CALLING METHOD

Load AC2 with the parameter word pointer; then do an indirect JSR to ECS.

```
                    LDA 2,PAR
                    JSR @DUMMY
                    STA 1,XLOC
                    STA 2,YLOC
                        .

DUMMY:  ECS
XLOC:   0
YLOC:   0
PAR:    .+1
        0   ;RDOS channel #
        0   ;CON console #
        0   ;SCR screen #
```

V. INPUT

AC2 should contain the word pointer to the parameter list.

```
PAR:    .+1
        0    ;RDOS channel #
        0    ;CON console #
        0    ;SCR screen #
```

VI. OUTPUT

AC1 will contain the X location of the cursor.
AC2 will contain the Y location of the cursor.

VII. ERROR RETURNS

NONE

VII. REFERENCED EXTERNALS

NONE

```
0001  ECS
                ;*****************************************************************
                ;                  ECS.SR   TOM EGGER/MQT 12/01/82              *
                ;                                                               *
                ;   SUBROUTINE TO READ CURSOR POSITION                          *
                ;   ON GDM.                                                     *
                ;                                                               *
                ;          INPUT PARAMETERS VIA AC2:                            *
                ;              0 RDOS CHANNEL NUMBER                            *
                ;              1 CONSOLE NUMBER USUALLY 0                       *
                ;              2 SCREEN NUMBER 1-3                              *
                ;          OUTPUT PARAMETERS:                                   *
                ;              AC1= X LOCATION OF CURSOR                        *
                ;              AC2= Y LOCATION OF CURSOR                        *
                ;                                                               *
                ;*****************************************************************

                        .TITL ECS
                        .ENT ECS .
        000001          .TXTM 1
                        .NREL
                ;
  00000'054504 ECS:       STA 3,SAV           ;SAVE RETURN ADDRESS
  00001'155000            MOV 2,3             ;DISPLACE OFF AC3
  00002'020504            LDA 0,FUNC1         ;FUNCTION WORD 120000
  00003'025401            LDA 1,1,3           ;CONSOLE NUMBER
FU00004'000000            HXL 1,1             ;LEFT SHIFT 8
FU00005'000000            IOR 1,0             ;ADD TO FUNCTION WORD
  00006'025402            LDA 1,2,3           ;SCREEN NUMBER
FU00007'000000            HXL 0,1             ;LEFT 4
  00010'125120            MOVZL 1,1           ;LEFT 1
  00011'125120            MOVZL 1,1           ;LEFT 1
FU00012'000000            IOR 1,0             ;ADD TO FUNCTION WORD
  00013'152400            SUB 2,2             ;AC2=0
  00014'126400            SUB 1,1              ;AC1=0
  00015'025402            LDA 1,2,3            ;SCREEN NUMBER
FU00016'000000            HXL 2,1              ;LEFT SHIFT 12
  00017'125120            MOVZL 1,1            ;LEFT 1
  00020'125120            MOVZL 1,1            ;TOTAL SHIFT LEFT 14
FU00021'000000            IOR 1,2              ;ADD TO AC2
  00022'025401            LDA 1,1,3            ;CONSOLE NUMBER
FU00023'000000            HXL 1,1              ;LEFT 8
FU00024'000000            IOR 1,2              ;ADD TO AC2
  00025'025400            LDA 1,0,3            ;RDOS CHANNEL NUMBER
FU00026'000000            IOR 1,2              ;AC2=CHNL+CON+RCHN
  00027'024460            LDA 1,N10            ;10.
  00030'040465            STA 0,BUF            ;SAVE FUNC1
  00031'020463            LDA 0,IBUF           ;POINTER TO FUNCTION WORD
  00032'006017            .SYSTM               ;GET THE CURSOR POSITION
  00033'015077            .RDS 77
  00034'000401            JMP .+1              ;NORMAL RETURN
  00035'020462            LDA 0,BUF+2          ;X+ZOOM
FU00036'000000            HXR 2,0              ;RIGHT SHIFT 12
  00037'040454            STA 0,ZOOM
  00040'101005            MOV 0,0,SNR          ;SKIP IF ZOOM IS NOT ZERO
  00041'000437            JMP OUT
  00042'020455            LDA 0,BUF+2
  00043'024445            LDA 1,MASK           ;MASK FOR X
  00044'107400            AND 0,1              ;ELIMINATE ZOOM
```

```
 00045'020446                   LDA 0,ZOOM

 0002  ECS
 00046'152520                   SUBZL 2,2               ;+1
 00047'113000                   ADD 0,2                 ;ZOOM +1
 00050'020441                   LDA 0,N2                ;2047
 00051'106400                   SUB 0,1                 ;X-2047=AC1
 00052'102400                   SUB 0,0                 ;0
U00053'000000                   DIVS
 00054'030435                   LDA 2,N2                ;2047.
 00055'147000                   ADD 2,1
 00056'020443                   LDA 0,BUF+4             ;XTR
 00057'106400                   SUB 0,1                 ;
 00060'044425                   STA 1,XLOC              ;
 00061'024437                   LDA 1,BUF+3             ;Y
 00062'020430                   LDA 0,N3                ;1535.
 00063'106400                   SUB 0,1                 ;Y VAL  =AC1
 00064'152520                   SUBZL 2,2               ;+1
 00065'020426                   LDA 0,ZOOM
 00066'113000                   ADD 0,2                 ;ZOOM +1
 00067'102400                   SUB 0,0
U00070'000000                   DIVS                    ;DIVIDE AC0-AC1/AC2=AC1 REM AC0
 00071'030421                   LDA 2,N3                ;1535.
 00072'147000                   ADD 2,1
 00073'020427                   LDA 0,BUF+5             ;Y TRANSLATION
 00074'106400                   SUB 0,1
 00075'131000                   MOV 1,2                 ;Y LOCATION AC2
 00076'024407                   LDA 1,XLOC              ;X LOCATION AC1
 00077'006405                   JSR @SAV                ;RETURN FROM SUB
 00100'020417 OUT:              LDA 0,BUF+2             ;X NO ZOOM
 00101'105000                   MOV 0,1                 ;X LOCATION
 00102'030416                   LDA 2,BUF+3             ;Y LOCATION
 00103'006401                   JSR @SAV                ;RETURN FROM SUB
 00104'000000 SAV:      0
 00105'000000 XLOC:     0
 00106'120000 FUNC1:    120000
 00107'000012 N10:      10.
 00110'007777 MASK:     7777
 00111'003777 N2:       2047.
 00112'002777 N3:       1535.
 00113'000000 ZOOM:     0
 00114'000232'IBUF:     BUF*2
 00115'000000 BUF:      0               ;FUNCTION
 00116'000000           0               ;4
 00117'000000           0               ;ZOOM BITS 1-3 X LOC 4-15
 00120'000000           0               ;Y LOCATION 4-15
 00121'000000           0               ;X TRANSLATION
 00122'000000           0               ;Y TRANSLATION

                        .END

  0003  ECS
BUF     000115'
DIVS    000000U
ECS     000000'
FUNC1   000106'
HXL     000000U
HXR     000000U
IBUF    000114'
IOR     000000U
```

```
MASK    000110'
N10     000107'
N2      000111'
N3      000112'
OUT     000100'
SAV     000104'
XLOC    000105'
ZOOM    000113'
```

I. IDENTIFICATION

Module name:	ECSR
Date:	October 30, 1982
Function:	Interrogate cursor using "enter cursor" button
Language:	Assembly
Memory locations:	ZREL 0 NREL 123 STACK 0

II. FUNCTIONAL SUMMARY

ECSR is a subroutine that will determine the cursor location after depressing the "enter cursor" button.

III. ENTRY POINTS

ECSR

IV. CALLING METHOD

Load AC2 with parameter word pointer; then do an indirect JSR to ECSR.

```
                LDA 2,PAR
                JSR @DUMMY
                STA 1,XLOC
                STA 2,YLOC
                    .

DUMMY:  ECSR
XLOC:   0
YLOC:   0
PAR:    .+1
        0   ;RDOS channel #
        0   ;CON console #
        0   ;SCR screen #
```

V. INPUT

AC2 should contain the word pointer to the parameter list.

```
PAR:    .+1
        0   ;RDOS channel #
        0   ;CON console #
        0   ;SCR screen #
```

Module name: ECSR Date: October 30, 1982

VI. OUTPUT

AC1 will contain the X location of the cursor.
AC2 will contain the Y location of the cursor.

VII. ERROR RETURNS

NONE

VIII. REFERENCED EXTERNALS

NONE

```
0001  ECSR
                ;*****************************************************************
                ;                 ECSR.SR   TOM EGGER/MQT 10/30/82              *
                ;                                                               *
                ;   SUBROUTINE TO READ ENTER CURSOR BUTTON                      *
                ;   ON GDM.                                                     *
                ;                                                               *
                ;          INPUT PARAMETERS VIA AC2:                            *
                ;              0 RDOS CHANNEL NUMBER                            *
                ;              1 CONSOLE NUMBER USUALLY 0                       *
                ;              2 SCREEN NUMBER 1-3                              *
                ;          OUTPUT PARAMETERS:                                   *
                ;              AC1= X LOCATION OF CURSOR                        *
                ;              AC2= Y LOCATION OF CURSOR                        *
                ;                                                               *
                ;*****************************************************************

                        .TITL ECSR
                        .ENT ECSR
        000001          .TXTM 1
                        .NREL
        031400          .DIO  .RDC=31400
                ;
  00000'054504 ECSR:      STA 3,SAV             ;SAVE RETURN ADDRESS
  00001'155000            MOV 2,3               ;DISPLACE OFF AC3
  00002'020504            LDA 0,FUNC1           ;FUNCTION WORD 120000
  00003'025401            LDA 1,1,3             ;CONSOLE NUMBER
FU00004'000000            HXL 1,1               ;LEFT SHIFT 8
FU00005'000000            IOR 1,0               ;ADD TO FUNCTION WORD
  00006'025402            LDA 1,2,3             ;SCREEN NUMBER
FU00007'000000            HXL 0,1               ;LEFT 4
  00010'125120            MOVZL 1,1             ;LEFT 1
  00011'125120            MOVZL 1,1             ;LEFT 1
FU00012'000000            IOR 1,0               ;ADD TO FUNCTION WORD
  00013'152400            SUB 2,2               ;AC2=0
  00014'126400            SUB 1,1               ;AC1=0
  00015'025402            LDA 1,2,3             ;SCREEN NUMBER
FU00016'000000            HXL 2,1               ;LEFT SHIFT 12
  00017'125120            MOVZL 1,1             ;LEFT 1
  00020'125120            MOVZL 1,1             ;TOTAL SHIFT LEFT 14
FU00021'000000            IOR 1,2               ;ADD TO AC2
  00022'025401            LDA 1,1,3             ;CONSOLE NUMBER
FU00023'000000            HXL 1,1               ;LEFT 8
FU00024'000000            IOR 1,2               ;ADD TO AC2
  00025'025400            LDA 1,0,3             ;RDOS CHANNEL NUMBER
FU00026'000000            IOR 1,2               ;AC2=CHNL+CON+RCHN
  00027'024460            LDA 1,N10             ;10.
  00030'040465            STA 0,BUF             ;SAVE FUNC1
  00031'020463            LDA 0,IBUF            ;POINTER TO FUNCTION WORD
  00032'006017            .SYSTM                ;GET THE CURSOR POSITION
  00033'031477            .RDC 77
  00034'000401            JMP .+1               ;NORMAL RETURN
  00035'020462            LDA 0,BUF+2           ;X+ZOOM
FU00036'000000            HXR 2,0               ;RIGHT SHIFT 12
  00037'040454            STA 0,ZOOM
  00040'101005            MOV 0,0,SNR           ;SKIP IF ZOOM IS NOT ZERO
  00041'000437            JMP OUT
  00042'020455            LDA 0,BUF+2
  00043'024445            LDA 1,MASK            ;MASK FOR X
```

```
 00044'107400                AND 0,1                 ;ELIMINATE ZOOM

0002  ECSR
 00045'020446                LDA 0,ZOOM
 00046'152520                SUBZL 2,2               ;+1
 00047'113000                ADD 0,2                 ;ZOOM +1
 00050'020441                LDA 0,N2                ;2047
 00051'106400                SUB 0,1                 ;X-2047=AC1
 00052'102400                SUB 0,0                 ;0
U00053'000000                DIVS
 00054'030435                LDA 2,N2                ;2047.
 00055'147000                ADD 2,1
 00056'020443                LDA 0,BUF+4             ;XTR
 00057'106400                SUB 0,1                 ;
 00060'044425                STA 1,XLOC              ;
 00061'024437                LDA 1,BUF+3             ;Y
 00062'020430                LDA 0,N3                ;1535.
 00063'106400                SUB 0,1                 ;Y VAL  =AC1
 00064'152520                SUBZL 2,2               ;+1
 00065'020426                LDA 0,ZOOM
 00066'113000                ADD 0,2                 ;ZOOM +1
 00067'102400                SUB 0,0
U00070'000000                DIVS                    ;DIVIDE AC0-AC1/AC2=AC1 REM AC0
 00071'030421                LDA 2,N3                ;1535.
 00072'147000                ADD 2,1
 00073'020427                LDA 0,BUF+5             ;Y TRANSLATION
 00074'106400                SUB 0,1
 00075'131000                MOV 1,2                 ;Y LOCATION AC2
 00076'024407                LDA 1,XLOC              ;X LOCATION AC1
 00077'006405                JSR @SAV                ;RETURN FROM SUB
 00100'020417 OUT:           LDA 0,BUF+2             ;X NO ZOOM
 00101'105000                MOV 0,1                 ;X LOCATION
 00102'030416                LDA 2,BUF+3             ;Y LOCATION
 00103'006401                JSR @SAV                ;RETURN FROM SUB
 00104'000000 SAV:      0
 00105'000000 XLOC:     0
 00106'120000 FUNC1:    120000
 00107'000012 N10:      10.
 00110'007777 MASK:     7777
 00111'003777 N2:       2047.
 00112'002777 N3:       1535.
 00113'000000 ZOOM:     0
 00114'000232*IBUF:     BUF*2
 00115'000000 BUF:      0          ;FUNCTION
 00116'000000           0          ;4
 00117'000000           0          ;ZOOM BITS 1-3 X LOC 4-15
 00120'000000           0          ;Y LOCATION 4-15
 00121'000000           0          ;X TRANSLATION
 00122'000000           0          ;Y TRANSLATION

                       .END

  0003  ECSR
BUF     000115'
DIVS    000000U
ECSR    000000'
FUNC1   000106'
HXL     000000U
HXR     000000U
IBUF    000114'
```

```
IOR     000000U
MASK    000110'
N0      000107'
N2      000111'
N3      000112'
OUT     000100'
SAV     000104'
XLOC    000105'
ZOOM    000113'
```

I. IDENTIFICATION

Module name:	ELEG
Date:	October 27, 1982
Function:	Display a legend
Language:	Assembly
Memory locations:	ZREL 0 NREL 112 STACK 0

II. FUNCTIONAL SUMMARY

The purpose of the ELEG subroutine is to create the legend block in the lower left (right) of the screen. Up to 32 characters may be entered.

III. ENTRY POINTS

ELEG

IV. CALLING METHOD

Load AC1 with the word pointer to the text. Laod AC2 with the word pointer to the parameter list. Do an indirect JSR to ELEG.

```
                LDA 1,TEX
                LDA 2,PAR
                JSR @DUMMY
                  .

DUMMY:  ELEG
TEX:    .TXT "TESTING <00>"
PAR:    .+1
        0
        .
```

V. INPUT

AC1 will contain a word pointer to the output characters. Enter an even number of characters, and make the last word binary <00>. AC2 contains the parameter pointer.

```
PAR:    .+1
        0    ;RDOS channel #
        0    ;CON console #
        0    ;SCR screen #
        0    ;OVY overlay #
```

Module name: ELEG Date: October 27, 1982

VI. OUTPUT

Legend characters are displayed as normal size block characters. The following are the screen coordinates for each overlay:

Overlay 1	X=16,	Y=48
Overlay 2	X=16,	Y=32
Overlay 3	X=16,	Y=16

VII. ERROR RETURNS

NONE

VIII. REFERENCED EXTERNALS

NONE

```
0001  ELEG
                ;*****************************************************************
                ;       ELEG.SR    TOM EGGER/MQT 10-27-82
                ;
                ;       SUBROUTINE TO DISPLAY A LEGEND MESSAGE      0  RDOS CHANNEL *
                ;                                                   1  CONSOLE *
                ;       AC1=WORD POINTER TO OUTPUT CHARACTERS       2  SCREEN *
                ;       AC2=PARAMETER POINTER                       3  OVERLAY
                ;*****************************************************************

                        .TITL ELEG
                        .ENT ELEG
        000001          .TXTM 1
                        .NREL

  00000'054441 ELEG:    STA 3,SAV               ;SAVE RETURN ADDRESS
  00001'044444          STA 1,WDPTR             ;SAVE THE WORD POINTER
  00002'020444          LDA 0,TXAD              ;POINT TO TEXT ADDRESS
  00003'040441          STA 0,TXSV              ;SAVE POINTER
  00004'020437          LDA 0,FUNC              ;LEGEND FUNCTION WORD
  00005'025001          LDA 1,1,2               ;CONSOLE *
FU00006'000000          HXL 1,1                 ;LEFT 8
FU00007'000000          IOR 1,0                 ;FUNCTION + CONSOLE *
  00010'025002          LDA 1,2,2               ;SCREEN *
FU00011'000000          HXL 0,1                 ;AC1 LEFT 4
  00012'125120          MOVZL 1,1               ;LEFT 1
  00013'125120          MOVZL 1,1               ;LEFT 1
FU00014'000000          IOR 1,0                 ;FUNCTION + SCREEN
  00015'025003          LDA 1,3,2               ;OVERLAY *
FU00016'000000          HXL 0,1                 ;AC1 LEFT 4   0=4
FU00017'000000          IOR 1,0                 ;FUNCTION + OVERLAY
  00020'040430          STA 0,ISN               ;FIRST OUTPUT WORD
  00021'026424 EL1:     LDA 1,@WDPTR            ;START OF TEXT
FU00022'000000          SN 1,1                  ;SKIP IF NON ZERO
  00023'000405          JMP EL2                 ;WRITE TO SCREEN
  00024'046422          STA 1,@TXAD             ;STORE IN OUT BUFFER
  00025'010421          ISZ TXAD                ;INCREMENT OUTBUFF
  00026'010417          ISZ WDPTR               ;INCREMENT INBUFF
  00027'000772          JMP EL1                 ;GET ANOTHER CHARACTER WORD
  00030'020417 EL2:     LDA 0,ISND              ;BYTE POINTER TO OUTPUT
  00031'031000          LDA 2,0,2               ;RDOS CHANNEL *
  00032'024410          LDA 1,N34               ;BYTES TO WRITE
  00033'006017          .SYSTM
  00034'016477          .WRS 77                 ;WRITE TO CHANNEL IN AC2
  00035'000401          JMP .+1                 ;SKIP THE ERROR RETURN
  00036'024406          LDA 1,TXSV              ;RESTORE
  00037'044407          STA 1,TXAD              ;POINTER
  00040'002401           JMP @SAV               ;RETURN FROM SUB

  00041'000000 SAV:     0
  00042'000042 N34:     34.
  00043'030000 FUNC:    30000
  00044'000000 TXSV:    0
  00045'000000 WDPTR:   0
  00046'000051'TXAD:    ISN+1
  00047'000120'ISND:    ISN*2
        000042 ISN:     .BLK 34.

                                .END
```

```
  0002  ELEG
EL1     000021'
EL2     000030'
ELEG    000000'
FUNC    000043'
HXL     000000U
IOR     000000U
ISN     000050'
ISND    000047'
N34     000042'
SAV     000041'
SN      000000U
TXAD    000046'
TXSV    000044'
WDPTR   000045'
```

I. IDENTIFICATION

Module name:	ELINE
Date:	November 5, 1982
Function:	Draw a line on the AFOS graphics screen
Language:	Assembly
Memory locations:	ZREL 0 NREL 150 STACK 0

II. FUNCTIONAL SUMMARY

ELINE is a subroutine that constructs a line (vector) from a point specified by X,Y position coordinates to a second point difined by delta deflection coordinates.

III. ENTRY POINTS

LINE1
LINE2

IV. CALLING METHOD

Load AC2 with the parameter word pointer; then do an indirect JSR to one of the two entry points. LINE1 will not store the image in CLS. LINE2 will store the image in CLS.

```
                LDA 2,PAR
                JSR @DUMMY
                  .

DUMMY:   LINE1
PAR:     .+1
          0
          .
```

V. INPUT

Ten input parameters are passed to this subroutine via a word pointer in AC2. The parameter list is of the form:

```
PAR:    .+1
        0    ;RDOS channel #
        0    ;X1
        0    ;Y1
        0    ;X2
        0    ;Y2
        0    ;SE = selective erase
        0    ;IZ = zoom
        0    ;CON = console #
        0    ;SCR = screen #
        0    ;OVY = overlay #
```

Module name: ELINE Date: November 5, 1982

VI. OUTPUT

A line is immediately drawn on the graphics screen.

VII. ERROR RETURNS

NONE

VIII. REFERENCED EXTERNALS

NONE

```
0001  ELINE
                ;*****************************************************************
                ;       ELINE.SR   TOM EGGER/MQT  11/5/82       PARAMETERS:  ,+1
                ;                                                  0   ICHN :RDOS CHNL
                ;       VECTOR PLOT SUBROUTINE                     1   X1   :FIRST X
                ;                                                  2   Y1   :FIRST Y
                ;       PARAMETERS ARE PASSED THROUGH AC2 VIA      3   X2   :SECND X
                ;       WORD POINTER                               4   Y2   :SECND Y
                ;                                                  5   SE   :ERASE
                ;       THE VECTOR IS WRITTEN TO SCREEN            6   IZ   :ZOOM
                ;       VIA WRS                                    7   CON  :CONSOLE *
                ;                                                  10  CHNL :CHNL *
                ;                                                  11  OVRLY:OVERLAY *
                ;*****************************************************************
                ;
                        .TITL ELINE
                        .ENT LINE1,LINE2
        000001          .TXTM 1
                        .NREL
                ;
   00000'054535 LINE1:  STA 3,SAV
   00001'020524         LDA 0,FUNC1        ;INTERACTIVE FUNCTION WORD NO CLS STORAG
   00002'040522         STA 0,FUNC         ;STORE IT AT FUNC
   00003'000404         JMP BEGIN
   00004'054531 LINE2:  STA 3,SAV          ;SAVE RETURN ADDRESS
   00005'020521         LDA 0,FUNC2        ;DISPLAY OVERLAY...LOADS CLS
   00006'040516         STA 0,FUNC         ;STORE IT AT FUNC
   00007'155000 BEGIN:  MOV 2,3            ;PARAMETER LIST IN AC3
   00010'020514         LDA 0,FUNC
   00011'031405         LDA 2,5,3          ;SELECTIVE ERASE
   00012'126400         SUB 1,1
 FU00013'000000         SEQ 1,2
   00014'024520         LDA 1,N12
 FU00015'000000         IOR 1,0
   00016'031407         LDA 2,7,3          ;CONSOLE NUMBER
 FU00017'000000         HXL 1,2            ;SHIFT LEFT 8 BITS AC2
 FU00020'000000         IOR 2,0            ;ADD TO FUNCTION WORD
   00021'031410         LDA 2,10,3         ;SCREEN NUMBER
 FU00022'000000         HXL 0,2            ;LEFT SIX
   00023'151120         MOVZL 2,2
   00024'151120         MOVZL 2,2
 FU00025'000000         IOR 2,0
   00026'031411         LDA 2,11,3
 FU00027'000000         HXL 0,2
 FU00030'000000         IOR 2,0
   00031'040506         STA 0,ISN          ;FIRST OUPUT WORD
   00032'020476         LDA 0,N5           ;5K
   00033'040505         STA 0,ISN+1        ;SECOND OUTPUT WORD
   00034'031406         LDA 2,6,3          ;ZOOM
 FU00035'000000         SZ 2,2             ;SKIP IF ZERO
   00036'000402         JMP .+2            ;SEE IF = 2
   00037'000407         JMP .+7            ;ZERO BIT
   00040'020471         LDA 0,N2           ;AC0=2
 FU00041'000000         SEQ 0,2            ;SKIP IF EQUAL
   00042'000402         JMP .+2            ;SET BIT
   00043'000403         JMP .+3            ;ZERO BIT
   00044'020467         LDA 0,N1           ;1B3
   00045'000402         JMP .+2            ;
   00046'102400         SUB 0,0            ;AC0=0
```

```
  00047'024460                LDA 1,FUNC3           ;FUNCTION WORD

 0002  ELINE
FU00050'000000                IOR 1,0
  00051'025401                LDA 1,1,3             ;FIRST X
FU00052'000000                IOR 1,0
  00053'040466                STA 0,ISN+2           ;THIRD OUTPUT WORD
  00054'102520                SUBZL 0,0
  00055'025406                LDA 1,6,3             ;ZOOM LEVEL
FU00056'000000                SLE 1,0
  00057'000403                JMP .+3
  00060'102400                SUB 0,0
  00061'000402                JMP .+2
  00062'020451                LDA 0,N1              ;BIT 12
FU00063'000000                IOR 0,1
  00064'021402                LDA 0,2,3             ;Y1
FU00065'000000                IOR 0,1
  00066'044454                STA 1,ISN+3           ;FORTH WORD
  00067'102400                SUB 0,0
  00070'101520                INCZL 0,0             ;AC0=2
  00071'040452                STA 0,ISN+4           ;2K FIFTH WORD
  00072'025401                LDA 1,1,3             ;X1
  00073'021403                LDA 0,3,3             ;X2
  00074'122400                SUB 1,0               ;X2-X1=AC0
  00075'101133                MOVZL# 0,0,SNC        ;SKIP <=0
  00076'000404                JMP .+4
  00077'100400                NEG 0,0               ;MAKE POSITIVE
  00100'024433                LDA 1,N1              ;BIT 12
FU00101'000000                IOR 1,0
  00102'040442                STA 0,ISN+5           ;DELTA X SIXTH WORD
  00103'025402                LDA 1,2,3             ;Y1
  00104'021404                LDA 0,4,3             ;Y2
  00105'122400                SUB 1,0
  00106'101133                MOVZL# 0,0,SNC        ;SKIP <=0
  00107'000404                JMP .+4
  00110'100400                NEG 0,0
  00111'024422                LDA 1,N1              ;BIT 12
FU00112'000000                IOR 1,0
  00113'040432                STA 0,ISN+6           ;WORD SEVEN
  00114'020422                LDA 0,ISND            ;BYTE POINTER TO ISN START
  00115'031400                LDA 2,0,3             ;RDOS CHANNEL NUMBER
  00116'024405                LDA 1,N14             ;NUMBER OF BYTES TO WRITE
  00117'006017                .SYSTM
  00120'016477                .WRS 77
  00121'000425                JMP ER
  00122'002413                JMP @SAV
  00123'000016 N14:    14.
  00124'000000 FUNC:    0
  00125'060000 FUNC1:   60000     ;DISPLAY INTERACTIVE
  00126'020000 FUNC2:    20000    ;DISPLAY OVERLAY FUNCTION WORD
  00127'160000 FUNC3:  160000    ;DELTA VECTOR FUNCTION WORD
  00130'000005 N5:     5
  00131'000002 N2:     2
  00132'000003 N3:     3
  00133'010000 N1:     1B3       ;12TH BIT FOR FORD AERO 4TH FOR DG
  00134'000010 N12:      1B12
  00135'000000 SAV:     0
  00136'000276'ISND:    .+1*2
  00137'000000 ISN:     0
  00140'000000          0
```

```
 00141'000000          0
 00142'000000          0

0003  ELINE
 00143'000000          0
 00144'000000          0
 00145'000000          0
 00146'006017 ER:      .SYSTM
 00147'006400          .ERTN
 00150'000776          JMP ER

                                .END

  0004  ELINE
BEGIN   000007'
ER      000146'
FUNC    000124'
FUNC1   000125'
FUNC2   000126'
FUNC3   000127'
HXL     000000U
IOR     000000U
ISN     000137'
ISND    000136'
LINE1   000000'
LINE2   000004'
N1      000133'
N12     000134'
N14     000123'
N2      000131'
N3      000132'
N5      000130'
SAV     000135'
SEQ     000000U
SLE     000000U
SZ      000000U
```

I. IDENTIFICATION

Module name:	ETXT
Date:	March 3, 1983
Function:	Generate text on the AFOS graphics screen
Language:	Assembly
Memory locations:	ZREL 0 NREL 316 STACK 0

II. FUNCTIONAL SUMMARY

ETXT is a subroutine that will display one or more (up to 180) characters on the screen. Special characters (the weather symbols) require the passage of 22-octal in the text array. All characters following the 22-byte will be interpreted as special characters. To return to the standard character set, pass 21-octal in the text array.

III. ENTRY POINTS

ETXT1
ETXT2

IV. CALLING METHOD

Load AC1 with a word pointer to the text array. Load AC2 with a word pointer to the parameter list. If the image is not to be stored in CLS, do an indirect JSR to ETXT1. To store the image in CLS, do an indirect JSR to ETXT2.

```
                    LDA 1,TEX
                    LDA 2,PAR
                    JSR @ET
                      .

ET:   ETXT1
TEX:  .+1
      .TXT "AFOS GRAPHICS "    ;even number of bytes
PAR:  .+1
       .    ;see parameter list example
```

V. INPUT

AC1 should contain the word pointer to the text array. AC2 should contain a word pointer to the input parameter list.

An example of the parameter list follows:

```
PAR:    .+1
        2       ;RDOS channel #
        200.    ;X start location
        300.    ;Y start location
        3       ;SIZE = double size
        0       ;SE = don't erase
        0       ;IZ = display at all zooms
        0       ;CON = the master console
        2       ;SCR = screen #2
        1       ;OVY = overlay #1
        7       ;NW = character words
```

VI. OUTPUT

Text is displayed on the screen.

VII. ERROR RETURNS

NONE

VIII. REFERENCED EXTERNALS

NONE

```
0001  ETXT
                ;*****************************************************************
                ;               ETXT.SR         TOM EGGER/MQT 3/3/83
                ;
                ;               CHARACTER INSTRUCTION SUBROUTINE
                ;
                ;               AC1=WORD POINTER TO TEXT
                ;
                ;               PARAMETERS PASSED THROUGH AC2:  ICHN
                ;                                               X START
                ;                                               Y START
                ;                                               SIZE
                ;                                               ERASE
                ;                                               CONSOLE *
                ;                                               SCREEN *
                ;                                               OVERLAY *
                ;                                               NW
                ;*****************************************************************
                ;
                ;
                        .TITL ETXT
                        .ENT ETXT1,ETXT2        ;TWO DIFFERENT ENTRY POINTS
      000001            .TXTM 1                 ;PACK TEXT BYTES LEFT TO RIGHT
                        .NREL
                ;
  00000'054532 ETXT1:   STA 3,SAV               ;ENTRY FOR FUNC1..SAVE RETURN ADDRESS
  00001'020536          LDA 0,FUNC1             ;INTERACTIVE FUNCTION WORD
  00002'040534          STA 0,FUNC              ;STORE IT AT FUNC
  00003'000404          JMP BEGIN               ;
  00004'054526 ETXT2:   STA 3,SAV               ;ENTRY FOR FUNC2..SAVE RETURN ADDRESS
  00005'020533          LDA 0,FUNC2             ;DISPLAY OVERLAY FUNCTION WORD
  00006'040530          STA 0,FUNC              ;STORE IT AT FUNC
  00007'044533 BEGIN:   STA 1,TXPTR             ;SAVE TEXT POINTER
  00010'155000          MOV 2,3                 ;MOVE PARAMETER LIST TO AC3
  00011'020537          LDA 0,TXAD              ;TEXT WORD POINTER
  00012'040517          STA 0,SVTX              ;SAVE THE POINTER
  00013'025411          LDA 1,11,3              ;NUMBER OF WORDS
  00014'044517          STA 1,CHWDS             ;CHARACTER WORDS
  00015'020526          LDA 0,N2                ;+2
  00016'123000          ADD 1,0                 ;CHWDS+2
  00017'040516          STA 0,OTWRD             ;# OF OUTPUT WORDS
  00020'020526          LDA 0,N10               ;+10.
  00021'123000          ADD 1,0                 ;NUMBER OF BYTES
  00022'101120          MOVZL 0,0               ;MULT *2
  00023'040511          STA 0,NBYTS             ;NUMBER OF BYTES
  00024'020512          LDA 0,FUNC              ;FUNCTION WORD FUNC1 OR FUNC2
  00025'031404          LDA 2,4,3               ;SELECTIVE ERASE
  00026'126400          SUB 1,1                 ;0
FU00027'000000          SEQ 2,1                 ;SKIP IF EQUAL AC1,AC2
  00030'024517          LDA 1,N12               ;12TH BIT
FU00031'000000          IOR 1,0                 ;ADD ON TO FUNCTION WORD
  00032'031406          LDA 2,6,3               ;CONSOLE *
FU00033'000000          HXL 1,2                 ;SHIFT LEFT 8 AC2   0=4 1=8 2=12
FU00034'000000          IOR 2,0                 ;ADD ON TO FUNCTION WORD
  00035'031407          LDA 2,7,3               ;SCREEN *
FU00036'000000          HXL 0,2                 ;SHIFT LEFT 4
  00037'151120          MOVZL 2,2               ;LEFT 1
  00040'151120          MOVZL 2,2               ;LEFT ONE MORE
FU00041'000000          IOR 2,0                 ;ADD TO FUNCTION WORD IN AC0
```

```
  00042'031410                LDA 2,10,3          ;OVERLAY #

 0002  ETXT
FU00043'000000                HXL 0,2             ;LEFT 4
FU00044'000000                IOR 2,0             ;ADD TO FUNCTION WORD
  00045'040505                STA 0,ISN           ;FIRST OUTPUT WORD
  00046'020467                LDA 0,OTWRD         ;NUMBER OF WORDS
  00047'040504                STA 0,ISN+1         ;SECOND OUTPUT WORD
  00050'102520                SUBZL 0,0           ;+1
  00051'031405                LDA 2,5,3           ;ZOOM RATIO
FU00052'000000                SNE 0,2             ;SKIP NOT EQUAL
  00053'000407                JMP .+7             ;GO SET BIT
  00054'020470                LDA 0,N3            ;AC0 +3
FU00055'000000                SEQ 0,2             ;SKIP IF EQUAL
  00056'000402                JMP .+2             ;DON'T SET BIT
  00057'000403                JMP .+3             ;SET BIT
  00060'102400                SUB 0,0             ;AC0=0
  00061'000402                JMP .+2             ;SKIP
  00062'020463                LDA 0,N4            ;183
  00063'024456                LDA 1,FUNC3         ;120000
FU00064'000000                IOR 1,0             ;ADD TO AC0
  00065'031401                LDA 2,1,3           ;X START
FU00066'000000                IOR 2,0             ;ADD TO AC0
  00067'040465                STA 0,ISN+2         ;THIRD OUTPUT
  00070'020453                LDA 0,N2            ;AC0=+2
  00071'031405                LDA 2,5,3           ;ZOOM RATIO
FU00072'000000                SNE 0,2             ;SKIP IF NOT EQUAL
  00073'000407                JMP .+7             ;
  00074'020450                LDA 0,N3            ;AC0=+3
FU00075'000000                SEQ 0,2             ;SKIP IF EQUAL
  00076'000402                JMP .+2             ;
  00077'000403                JMP .+3             ;
  00100'102400                SUB 0,0             ;AC0=0
  00101'000402                JMP .+2             ;
  00102'020443                LDA 0,N4            ;183
  00103'025403                LDA 1,3,3           ;CHARACTER SIZE
FU00104'000000                HXL 2,1             ;SHIFT LEFT 12
  00105'125120                MOVZL 1,1           ;LEFT ANOTHER ONE BIT
FU00106'000000                IOR 1,0             ;INCLUDE AC1 WITH AC0
  00107'025402                LDA 1,2,3           ;Y START
FU00110'000000                IOR 1,0             ;INCLUDE WITH AC1
  00111'040444                STA 0,ISN+3         ;FOURTH OUTPUT
                ;
  00112'026430 TX1:           LDA 1,@TXPTR        ;POINT TO START OF TEXT
  00113'046435                STA 1,@TXAD         ;AC1 TO AC3
  00114'010426                ISZ TXPTR           ;INCREMENT AC1
  00115'010433                ISZ TXAD            ;INCREMENT AC3 SKIP IF ZERO
  00116'014415                DSZ CHWDS           ;DECREMENT
  00117'000773                JMP TX1             ;GET ANOTHER CHARACTER
  00120'020431                LDA 0,ISND          ;BYTE POINTER TO OUTPUT
  00121'031400                LDA 2,0,3           ;RDOS CHANL #
  00122'024412                LDA 1,NBYTS         ;NUMBER OF BYTES TO WRITE
  00123'006017                .SYSTM              ;
  00124'016477                .WRS 77             ;WRITE TO CHNL IN AC2
  00125'000401                JMP .+1             ;ERROR RETURN
  00126'024403                LDA 1,SVTX          ;
  00127'044421                STA 1,TXAD          ;RESTORE POINTER
  00130'002402                JMP @SAV            ;RETURN FROM SUB
  00131'000000 SVTX:    0
  00132'000000 SAV:     0
```

```
 00133'000000 CHWDS:  0
 00134'000000 NBYTS:  0

0003  ETXT
 00135'000000 OTWRD:  0
 00136'000000 FUNC:   0
 00137'060000 FUNC1:  60000      ;INTERACTIVE FUNCTION WORD
 00140'020000 FUNC2:  20000      ;DISPLAY OVERLAY FUNC WORD
 00141'120000 FUNC3:  120000     ;CHARCTER GENERATOR FUNC WORD
 00142'000000 TXPTR:  0
 00143'000002 N2:     2
 00144'000003 N3:     3
 00145'010000 N4:     1B3
 00146'000012 N10:    10.        ;DECIMAL 10
 00147'000010 N12:    1B12
 00150'000156'TXAD:   ISN+4      ;TEXT WORD POINTER
 00151'000324'ISND:   .+1*2      ;BYTE POINTER
       000144 ISN:    .BLK 100.  ;RESERVE 100 WORDS

                           .END
```

```
  0004  ETXT
BEGIN   000007'
CHWDS   000133'
ETXT1   000000'
ETXT2   000004'
FUNC    000136'
FUNC1   000137'
FUNC2   000140'
FUNC3   000141'
HXL     000000U
IOR     000000U
ISN     000152'
ISND    000151'
N10     000146'
N12     000147'
N2      000143'
N3      000144'
N4      000145'
NBYTS   000134'
OTWRD   000135'
SAV     000132'
SEQ     000000U
SNE     000000U
SVTX    000131'
TX1     000112'
TXAD    000150'
TXPTR   000142'
```

I. IDENTIFICATION

Module names:	EFCSR EFCS
Dates:	October 30, 1982 December 1, 1982
Function:	Interrogate the cursor
Language:	Assembly (FORTRAN interfaces)
Memory locations:	ZREL 0 NREL 24 STACK 5

II. FUNCTIONAL SUMMARY

EFCSR is a subroutine that will interrogate the cursor location by depressing the "enter cursor" button. The EFCS subroutine will obtain the cursor location without depressing the "enter cursor" button.

III. ENTRY POINTS

EFCSR with the "enter cursor" button
EFCS without the "enter cursor" button

IV. CALLING METHOD

```
CALL EFCSR(ICHN,CON,SCR,X,Y)    ;with
CALL EFCS(ICHN,CON,SCR,X,Y)     ;without
```

V. INPUT

ICHN - RDOS channel number
CON - Console number
SCR - Screen number

All input parameters should be declared integer.

VI. OUTPUT

The X and Y coordinate position of the cursor is returned. X and Y variables should be declared integer.

VII. ERROR RETURNS

NONE

Module names: EFCSR
EFCS

Dates: October 30, 1982
December 1, 1982

VIII. REFERENCED EXTERNALS

EGR1.LB

ECSR interrogates the cursor position by depressing "enter cursor" button

ECS interrogates the cursor position without depressing the "enter cursor" button

FORT.LB

.CPYL copies the return address of the calling program

.FRET returns to the calling program

```
0001  EFCSR
                ;**********************************************************
                ;            EFCSR.SR  TOM EGGER 10-30-82                *
                ;                                                        *
                ;     FORTRAN INTERFACE TO ASSEMBLY ECSR                 *
                ;                                                        *
                ;     CALLING SEQUENCE:                                  *
                ;                                                        *
                ;           CALL EFCSR(ICHN,CON,SCR,X,Y)                 *
                ;                                                        *
                ;**********************************************************
                ;
                          .TITL EFCSR
                          .ENT EFCSR
                          .EXTD .CPYL,.FRET
                          .EXTN ECSR
                          .NREL
                                                    ;SET UP STACK
      177611               ICHN=-167                ;INPUT
      177612               CON=-166                 ;INPUT
      177613               SCR=-165                 ;INPUT
      177614               X=-164                   ;RETURNED
      177615               Y=-163                   ;RETURNED
      000005               FS.=5
00000'000005              FS.
                ;
00001'006001$EFCSR:       JSR @.CPYL               ;COPY ADDRESS TO STACK
00002'023611              LDA 0,@ICHN,3            ;RDOS CHANNEL #
00003'040417              STA 0,PAR+1              ;STORE IT
00004'023612              LDA 0,@CON,3             ;CONSOLE #
00005'040416              STA 0,PAR+2              ;STORE IT
00006'023613              LDA 0,@SCR,3             ;SCREEN #
00007'040415              STA 0,PAR+3              ;STORE IT
00010'054410              STA 3,FSP                ;SAVE STACK POINTER
00011'030410              LDA 2,PAR                ;PARAMETER POINTER
00012'006405              JSR @ECS                 ;GO TO ASSEMBLY SUB ECSR
00013'034405              LDA 3,FSP                ;RESTORE POINTER
00014'047614              STA 1,@X,3               ;RETURNED X VALUE
00015'053615              STA 2,@Y,3               ;RETURNED Y VALUE
00016'006002$             JSR @.FRET               ;RETURN TO FORTRAN
                ;
00017'077777 ECS:   ECSR
00020'000000 FSP:   0     ;STACK POINTER
00021'000022'PAR:   .+1
00022'000000        0     ;ICHN
00023'000000        0     ;CONSOLE
00024'000000        0     ;SCREEN

                                    .END

  0002  EFCSR
CON     177612
ECS     000017'
ECSR    000017'X
EFCSR   000001'
FSP     000020'
FS.     000005
ICHN    177611
PAR     000021'
```

```
SCR     177613
X       177614
Y       177615
.CPYL   000001$X
.FRET   000002$X
```

```
0001  EFCS
                ;*****************************************************
                ;           EFCS.SR   TOM EGGER 12-01-82            *
                ;                                                   *
                ;     FORTRAN INTERFACE TO ASSEMBLY ECS             *
                ;                                                   *
                ;     CALLING SEQUENCE:                             *
                ;                                                   *
                ;           CALL EFCS(ICHN,CON,SCR,X,Y)             *
                ;                                                   *
                ;*****************************************************
                ;
                        .TITL EFCS
                        .ENT EFCS
                        .EXTD .CPYL,.FRET
                        .EXTN ECS
                        .NREL
                                                    ;SET UP STACK
      177611             ICHN=-167                  ;INPUT
      177612             CON=-166                   ;INPUT
      177613             SCR=-165                   ;INPUT
      177614             X=-164                     ;RETURNED
      177615             Y=-163                     ;RETURNED
      000005             FS.=5
00000'000005            FS.
                ;
00001'006001$EFCS:      JSR @.CPYL                 ;COPY ADDRESS TO STACK
00002'023611            LDA 0,@ICHN,3              ;RDOS CHANNEL #
00003'040417            STA 0,PAR+1                ;STORE IT
00004'023612            LDA 0,@CON,3               ;CONSOLE #
00005'040416            STA 0,PAR+2                ;STORE IT
00006'023613            LDA 0,@SCR,3               ;SCREEN #
00007'040415            STA 0,PAR+3                ;STORE IT
00010'054410            STA 3,FSP                  ;SAVE STACK POINTER
00011'030410            LDA 2,PAR                  ;PARAMETER POINTER
00012'006405            JSR @ECSS                  ;GO TO ASSEMBLY SUB ECSR
00013'034405            LDA 3,FSP                  ;RESTORE POINTER
00014'047614            STA 1,@X,3                 ;RETURNED X VALUE
00015'053615            STA 2,@Y,3                 ;RETURNED Y VALUE
00016'006002$           JSR @.FRET                 ;RETURN TO FORTRAN
                ;
00017'077777 ECSS:  ECS
00020'000000 FSP:   0     ;STACK POINTER
00021'000022'PAR:   .+1
00022'000000        0     ;ICHN
00023'000000        0     ;CONSOLE
00024'000000        0     ;SCREEN

                                .END

  0002  EFCS
CON     177612
ECS     000017'X
ECSS    000017'
EFCS    000001'
FSP     000020'
FS.     000005
ICHN    177611
PAR     000021'
```

```
SCR     177613
X       177614
Y       177615
.CPYL   000001$X
.FRET   000002$X
```

I. IDENTIFICATION

Module name:	EFLEG
Date:	October 30,1982
Function:	Display a legend
Language:	Assembly (FORTRAN interface)
Memory locations:	ZREL 0 NREL 23 STACK 5

II. FUNCTIONAL SUMMARY

The purpose of the EFLEG subroutine is to create the legend block in the lower left (right) of the screen. Up to 32 characters may be entered. ELFEG is the interface to FORTRAN for ELEG.

III. ENTRY POINTS

EFLEG

IV. CALLING METHOD

```
CALL EFLEG(IBUF,ICHN,CON,SCR,OVY)
```

V. INPUT

IBUF	-	An array containing the text
ICHN	-	The RDOS channel number
CON	-	The console number
SCR	-	The screen number
OVY	-	The overlay number

VI. OUTPUT

Legend characters are displayed as normal size block characters. The following are the screen coordinates for each overlay:

Overlay 1	X=16,	Y=48
Overlay 2	X=16,	Y=32
Overlay 3	X=16,	Y=16

Module name: EFLEG Date: October 30,1982

VII. ERROR RETURNS

NONE

VIII. REFERENCED EXTERNALS

EGR1.LB
ELEG displays the legend
FORT.LB
.CPYL saves the return address of the calling program on the stack
.FRET returns to the calling program

```
0001  EFLEG
                ;*****************************************************
                ;          EFLEG.SR  TOM EGGER 10-30-82             *
                ;                                                   *
                ;     FORTRAN INTERFACE TO ASSMBLY SUB ELEG         *
                ;                                                   *
                ;     CALL SEQUENCE:                                *
                ;                                                   *
                ;           CALL EFLEG(IBUF,ICHN,CON,SCR,OVY)       *
                ;                                                   *
                ;*****************************************************
                ;
                          .TITL EFLEG
                          .ENT EFLEG
                          .EXTD .CPYL,.FRET
                          .EXTN ELEG
                          .NREL
                                                  ;SET UP THE STACK
      177611              IBUF=-167               ;ADDRESS OF TEXT ARRAY
      177612              ICHN=IBUF+1             ;ADDRESS OF RDOS CHANNEL #
      177613              CON=ICHN+1              ;ADDRESS OF CONSOLE #
      177614              SCR=CON+1               ;ADDRESS OF SCREEN #
      177615              OVY=SCR+1               ;ADDRESS OF OVERLAY #
      000005              FS.=OVY-IBUF+1
00000'000005              FS.
                ;
00001'006001$EFLEG:       JSR @.CPYL              ;COPY ADDRESS TO STACK
00002'023612              LDA 0,@ICHN,3           ;RDOS CHANNEL #
00003'040415              STA 0,PAR+1             ;STORE ON PARAMETER LIST
00004'023613              LDA 0,@CON,3            ;CONSOLE #
00005'040414              STA 0,PAR+2             ;STORE ON PARAMETER LIST
00006'023614              LDA 0,@SCR,3            ;LOAD SCREEN #
00007'040413              STA 0,PAR+3             ;STORE ON PAR LIST
00010'023615              LDA 0,@OVY,3            ;LOAD OVERLAY #
00011'040412              STA 0,PAR+4             ;STORE ON PAR LIST
00012'025611              LDA 1,IBUF,3            ;ADDRESS OF TEXT
00013'030404              LDA 2,PAR               ;PARAMETER POINTER
00014'006402              JSR @EL                 ;GO TO ASSEMBLY SUB ELEG
00015'006002$             JSR @.FRET              ;RETURN TO FORTRAN
                ;
00016'077777 EL:    ELEG
00017'000020'PAR:   .+1
00020'000000        0    ;RDOS CHANNEL
00021'000000        0    ;CONSOLE
00022'000000        0    ;SCREEN
00023'000000        0    ;OVERLAY
                                  .END

  0002  EFLEG
CON     177613
EFLEG   000001'
EL      000016'
ELEG    000016'X
FS.     000005
IBUF    177611
ICHN    177612
OVY     177615
PAR     000017'
SCR     177614
```

```
.CPYL   000001$X
.FRET   000002$X
```

I. IDENTIFICATION

Module names:	EFLINE EFSLINE
Date:	November 1, 1982
Function:	Draw a line on the screen
Language:	Assembly (FORTRAN interfaces)
Memory locations:	ZREL 0 NREL 44 STACK

II. FUNCTIONAL SUMMARY

EFSLINE is a subroutine that constructs a line on the screen; the image is stored in CLS. EFLINE will also construct a line on the screen; the image is not stored in CLS.

III. ENTRY POINTS

EFLINE image is not stored in CLS
EFSLINE image is stored in CLS

IV. CALLING METHOD

```
CALL EFLINE(ICHN,X1,Y1,X2,Y2,SE,IZ,CON,SCR,OVY)   ;without CLS
CALL EFSLINE(ICHN,X1,Y1,X2,Y2,SE,IZ,CON,SCR,OVY)  ;with CLS
```

V. INPUT

ICHN	-	RDOS channel #
X1	-	First X
Y1	-	First Y
X2	-	Second X
Y2	-	Second Y
SE	-	Selective erase
IZ	-	Zoom ratio
CON	-	Console #
SCR	-	Screen #
OVY	-	Overlay #

All input variables should be declared integer.

Module names: EFLINE
EFSLINE

Date: November 1, 1982

VI. OUTPUT

A line is immediately drawn on the graphics screen.

VII. ERROR RETURNS

NONE

VIII. REFERENCED EXTERNALS

EGR1.LB

LINE1 draws a line wihtout CLS storage
LINE2 draws a line with CLS storage

FORT.LB

.CPYL copies the return address of the calling program
.FRET returns to the calling program

```
0001  EFLIN
                ;*****************************************************************
                ;              EFLINE.SR TOM EGGER/MQT 11/01/82                 *
                ;                                                               *
                ;     FORTRAN INTERFACE TO ASSEMBLY ELINE PROGRAM                *
                ;     INTERACTIVE FUNCTION WORD...NO CLS STORAGE                *
                ;     CALLING SEQUENCE:                                         *
                ;                                                               *
                ;          CALL EFLINE(ICHN,X1,Y1,X2,Y2,SE,IZ,CON,SCR,OVY)      *
                ;                                                               *
                ;                                                               *
                ;*****************************************************************

                            .TITL EFLINE
                            .ENT EFLINE
                            .EXTD .CPYL,.FRET
                            .EXTN LINE1
       000001               .TXTM 1
                            .NREL
                                                ;SET UP THE STACK
       177611       ICHN=-167                   ;RDOS CHANNEL
       177612       X1=ICHN+1                   ;LINE
       177613       Y1=X1+1                     ;ENDPOINTS
       177614       X2=Y1+1                     ;
       177615       Y2=X2+1                     ;
       177616       SE=Y2+1                     ;ERASE 0=NO 1=YES
       177617       IZ=SE+1                     ;ZOOM RATIOS
       177620       CON=IZ+1                    ;CONSOLE # 0=MASTER
       177621       SCR=CON+1                   ;SCREEN # 1,2,3
       177622       OVY=SCR+1                   ;OVERLAY #
       000012       FS.=OVY-ICHN+1
00000'000012        FS.
                ;
00001'006001$EFLINE:    JSR @.CPYL              ;COPY ADDRESS TO STACK
00002'023611            LDA 0,@ICHN,3           ;RDOS CHNL
00003'040430            STA 0,PAR+1             ;STORE ON PARAMETER LIST
00004'023612            LDA 0,@X1,3             ;FIRST X
00005'040427            STA 0,PAR+2             ;STORE IT
00006'023613            LDA 0,@Y1,3             ;FIRST Y
00007'040426            STA 0,PAR+3             ;STORE IT
00010'023614            LDA 0,@X2,3             ;
00011'040425            STA 0,PAR+4             ;
00012'023615            LDA 0,@Y2,3             ;
00013'040424            STA 0,PAR+5             ;
00014'023616            LDA 0,@SE,3             ;ERASE
00015'040423            STA 0,PAR+6             ;STORE IT
00016'023617            LDA 0,@IZ,3             ;ZOOM
00017'040422            STA 0,PAR+7             ;STORE IT
00020'023620            LDA 0,@CON,3            ;CONSOLE #
00021'040421            STA 0,PAR+10            ;STORE
00022'023621            LDA 0,@SCR,3            ;SCREEN #
00023'040420            STA 0,PAR+11            ;
00024'023622            LDA 0,@OVY,3            ;OVERLAY #
00025'040417            STA 0,PAR+12            ;STORE IT
                ;
00026'030404            LDA 2,PAR               ;PARAMETER POINTER
00027'006402            JSR @EL                 ;GO TO ASSMBLY SUB ELINE
00030'006002$           JSR @.FRET              ;RETURN TO FORTRAN
                ;
```

```
 00031'077777 EL:   LINE1   ;ELINE ENTRY POINT...LINE WITH NO CLS STORAGE

 0002  EFLIN
  00032'000033'PAR:  .+1
  00033'000000       0        ;ICHN
  00034'000000       0        ;X1
  00035'000000       0        ;Y1
  00036'000000       0        ;X2
  00037'000000       0        ;Y2
  00040'000000       0        ;SE
  00041'000000       0        ;IZ
  00042'000000       0        ;CON
  00043'000000       0        ;SCR
  00044'000000       0     ;OVY
                ;
                ;
                                .END

  0003  EFLIN
CON     177620
EFLIN   000001'
EL      000031'
FS.     000012
ICHN    177611
IZ      177617
LINE1   000031'X
OVY     177622
PAR     000032'
SCR     177621
SE      177616
X1      177612
X2      177614
Y1      177613
Y2      177615
.CPYL   000001$X
.FRET   000002$X
```

```
0001  EFSLI
                ;*****************************************************************
                ;              EFSLINE.SR TOM EGGER/MQT 11/01/82                *
                ;                                                               *
                ;    FORTRAN INTERFACE TO ASSEMBLY ELINE PROGRAM                *
                ;    USES DISPLAY OVERLAY FUNCTION WORD...CLS STORAGE           *
                ;    CALLING SEQUENCE:                                          *
                ;                                                               *
                ;         CALL EFSLINE(ICHN,X1,Y1,X2,Y2,SE,IZ,CON,SCR,OVY)      *
                ;                                                               *
                ;                                                               *
                ;*****************************************************************

                                .TITL EFSLINE
                                .ENT EFSLINE
                                .EXTD .CPYL,.FRET
                                .EXTN LINE2
        000001                  .TXTM 1
                                .NREL
                                                        ;SET UP THE STACK
        177611          ICHN=-167               ;RDOS CHANNEL
        177612          X1=ICHN+1               ;LINE
        177613          Y1=X1+1                 ;ENDPOINTS
        177614          X2=Y1+1                 ;
        177615          Y2=X2+1                 ;
        177616          SE=Y2+1                 ;ERASE 0=NO 1=YES
        177617          IZ=SE+1                 ;ZOOM RATIOS
        177620          CON=IZ+1                ;CONSOLE # 0=MASTER
        177621          SCR=CON+1               ;SCREEN # 1,2,3
        177622          OVY=SCR+1               ;OVERLAY #
        000012          FS.=OVY-ICHN+1
00000'000012            FS.
                ;
00001'006001$EFSLINE:   JSR @.CPYL              ;COPY ADDRESS TO STACK
00002'023611            LDA 0,@ICHN,3           ;RDOS CHNL
00003'040430            STA 0,PAR+1             ;STORE ON PARAMETER LIST
00004'023612            LDA 0,@X1,3             ;FIRST X
00005'040427            STA 0,PAR+2             ;STORE IT
00006'023613            LDA 0,@Y1,3             ;FIRST Y
00007'040426            STA 0,PAR+3             ;STORE IT
00010'023614            LDA 0,@X2,3             ;
00011'040425            STA 0,PAR+4             ;
00012'023615            LDA 0,@Y2,3             ;
00013'040424            STA 0,PAR+5             ;
00014'023616            LDA 0,@SE,3             ;ERASE
00015'040423            STA 0,PAR+6             ;STORE IT
00016'023617            LDA 0,@IZ,3             ;ZOOM
00017'040422            STA 0,PAR+7             ;STORE IT
00020'023620            LDA 0,@CON,3            ;CONSOLE #
00021'040421            STA 0,PAR+10            ;STORE
00022'023621            LDA 0,@SCR,3            ;SCREEN #
00023'040420            STA 0,PAR+11            ;
00024'023622            LDA 0,@OVY,3            ;OVERLAY #
00025'040417            STA 0,PAR+12            ;STORE IT
                ;
00026'030404            LDA 2,PAR               ;PARAMETER POINTER
00027'006402            JSR @EL                 ;GO TO ASSMBLY SUB ELINE
00030'006002$           JSR @.FRET              ;RETURN TO FORTRAN
                ;
```

```
  00031'077777 EL:   LINE2   ;ELINE ENTRY POINT...LINE WITH CLS STORAGE

 0002  EFSLI
  00032'000033'PAR:  .+1
  00033'000000       0        ;ICHN
  00034'000000       0        ;X1
  00035'000000       0        ;Y1
  00036'000000       0        ;X2
  00037'000000       0        ;Y2
  00040'000000       0        ;SE
  00041'000000       0        ;IZ
  00042'000000       0        ;CON
  00043'000000       0        ;SCR
  00044'000000       0     ;OVY
                 ;
                 ;
                                .END
```

```
  0003  EFSLI
CON     177620
EFSLI   000001'
EL      000031'
FS.     000012
ICHN    177611
IZ      177617
LINE2   000031'X
OVY     177622
PAR     000032'
SCR     177621
SE      177616
X1      177612
X2      177614
Y1      177613
Y2      177615
.CPYL   000001$X
.FRET   000002$X
```

I. IDENTIFICATION

Module names:	EFTXT EFSTXT
Date:	November 1, 1982
Function:	Generate text on the AFOS graphic screen
Language:	Assembly (FORTRAN interface)
Memory language:	ZREL 0 NREL 45 STACK 13

II. FUNCTIONAL SUMMARY

EFTXT and EFSTXT are subroutines that will display up to 180 characters on the graphics screen. Special characters (the weather symbols) require the passage of 22-octal in the text array. All characters following the 22-byte will be interpreted as special characters. To return to the standard character set pass 21-octal in the text array.

III. ENTRY POINTS

ETXT1 if the characters are not to be stored in CLS
ETXT2 if the characters are to be stored in CLS

IV. CALLING METHOD

```
CALL EFTXT(IBUF,ICHN,X,Y,SIZE,SE,IZ,CON,SCR,OVY,NW)   ;without CLS
CALL EFSTXT(IBUF,ICHN,X,Y,SIZE,SE,IZ,CON,SCR,OVY,NW)  ;with CLS
```

V. INPUT

IBUF	-	text array
ICHN	-	RDOS channel #
X	-	X start location
Y	-	Y start location
SIZE	-	character size
SE	-	selective erase
IZ	-	zoom ratio
CON	-	console #
SCR	-	screen #
OVY	-	overlay #
NW	-	character words

All input variables should be declared integer.

VI. OUTPUT

Text is displayed on the screen.

VII. ERROR RETURNS

NONE

VIII. REFERENCED EXTERNALS

EGR1.LB
ETXT1 display the text and do not store in CLS
ETXT2 display the text and do store in CLS
FORT.LB
.CPYL copies the return address of the calling program
.FRET returns to the calling program

```
0001  EFTXT
                ;*****************************************************************
                ;       EFTXT.SR          TOM EGGER 11-01-82                      *
                ;                                                                 *
                ;       FORTRAN INTERFACE TO ASSEMBLY SUB ETXT                    *
                ;       USES INTERACTIVE FUNCTION WORD..NO CLS STORAGE            *
                ;       CALLING SEQUENCE:                                         *
                ;                                                                 *
                ;       CALL EFTXT(IBUF,ICHN,X,Y,SIZE,SE,IZ,CON,SCR,OVY,NW)       *
                ;                                                                 *
                ;                                                                 *
                ;*****************************************************************
                ;
                        .TITL EFTXT
                        .ENT EFTXT
                        .EXTD .CPYL,.FRET
                        .EXTN ETXT1
                        .NREL
                                                ;SET THE STACK
      177611            IBUF=-167
      177612            ICHN=IBUF+1
      177613            X=ICHN+1
      177614            Y=X+1
      177615            SIZE=Y+1
      177616            SE=SIZE+1
      177617            IZ=SE+1
      177620            CON=IZ+1
      177621            SCR=CON+1
      177622            OVY=SCR+1
      177623            NW=OVY+1
      000013            FS.=NW-IBUF+1
00000'000013            FS.
                ;
00001'006001$EFTXT:     JSR @.CPYL              ;RETURN ADDRESS
00002'023612            LDA 0,@ICHN,3           ;RDOS CHANNEL #
00003'040431            STA 0,PAR+1             ;STORE IT
00004'023613            LDA 0,@X,3              ;X START LOCATION
00005'040430            STA 0,PAR+2             ;STORE IT
00006'023614            LDA 0,@Y,3              ;Y START LOCATION
00007'040427            STA 0,PAR+3             ;STORE IT
00010'023615            LDA 0,@SIZE,3           ;CHARACTER MODE
00011'040426            STA 0,PAR+4
00012'023616            LDA 0,@SE,3             ;SELECTIVE ERASE
00013'040425            STA 0,PAR+5
00014'023617            LDA 0,@IZ,3             ;ZOOM RATIO
00015'040424            STA 0,PAR+6
00016'023620            LDA 0,@CON,3             ;CONSOLE #
00017'040423            STA 0,PAR+7
00020'023621            LDA 0,@SCR,3            ;SCREEN #
00021'040422            STA 0,PAR+10
00022'023622            LDA 0,@OVY,3            ;OVERLAY #
00023'040421            STA 0,PAR+11
00024'023623            LDA 0,@NW,3             ;NUMBER OF WORDS
00025'040420            STA 0,PAR+12
00026'025611            LDA 1,IBUF,3            ;START ADDRESS OF TEXT
00027'030404            LDA 2,PAR               ;PARAMETER POINTER
00030'006402            JSR @ET                 ;GO TO ETXT ROUTINE
00031'006002$           JSR @.FRET              ;RETURN TO FORTRAN
                ;
```

```
 00032'077777 ET:     ETXT1     ;ENTRY POINT TO ETXT...TEXT WITH NO CLS STORAGE

  0002  EFTXT
  00033'000034'PAR:   .+1
        000012        .BLK 10.

                                .END

  0003  EFTXT
CON     177620
EFTXT   000001'
ET      000032'
ETXT1   000032'X
FS.     000013
IBUF    177611
ICHN    177612
IZ      177617
NW      177623
OVY     177622
PAR     000033'
SCR     177621
SE      177616
SIZE    177615
X       177613
Y       177614
.CPYL   000001#X
.FRET   000002#X
```

```
0001  EFSTX
                ;*****************************************************************
                ;       EFSTXT.SR          TOM EGGER 11-01-82                     *
                ;                                                                 *
                ;       FORTRAN INTERFACE TO ASSEMBLY SUB ETXT                    *
                ;       USES DISPLAY OVERLAY FUNCTION WORD..CLS STORAGE           *
                ;       CALLING SEQUENCE:                                         *
                ;                                                                 *
                ;       CALL EFSTXT(IBUF,ICHN,X,Y,SIZE,SE,IZ,CON,SCR,OVY,NW)      *
                ;                                                                 *
                ;                                                                 *
                ;*****************************************************************
                ;
                        .TITL EFSTXT
                        .ENT EFSTXT
                        .EXTD .CPYL,.FRET
                        .EXTN ETXT2
                        .NREL
                                                ;SET THE STACK
      177611            IBUF=-167
      177612            ICHN=IBUF+1
      177613            X=ICHN+1
      177614            Y=X+1
      177615            SIZE=Y+1
      177616            SE=SIZE+1
      177617            IZ=SE+1
      177620            CON=IZ+1
      177621            SCR=CON+1
      177622            OVY=SCR+1
      177623            NW=OVY+1
      000013            FS.=NW-IBUF+1
00000'000013            FS.
                ;
00001'006001$EFSTXT:    JSR @.CPYL              ;RETURN ADDRESS
00002'023612            LDA 0,@ICHN,3           ;RDOS CHANNEL #
00003'040431            STA 0,PAR+1             ;STORE IT
00004'023613            LDA 0,@X,3              ;X START LOCATION
00005'040430            STA 0,PAR+2             ;STORE IT
00006'023614            LDA 0,@Y,3              ;Y START LOCATION
00007'040427            STA 0,PAR+3             ;STORE IT
00010'023615            LDA 0,@SIZE,3           ;CHARACTER MODE
00011'040426            STA 0,PAR+4
00012'023616            LDA 0,@SE,3             ;SELECTIVE ERASE
00013'040425            STA 0,PAR+5
00014'023617            LDA 0,@IZ,3             ;ZOOM RATIO
00015'040424            STA 0,PAR+6
00016'023620            LDA 0,@CON,3             ;CONSOLE #
00017'040423            STA 0,PAR+7
00020'023621            LDA 0,@SCR,3            ;SCREEN #
00021'040422            STA 0,PAR+10
00022'023622            LDA 0,@OVY,3            ;OVERLAY #
00023'040421            STA 0,PAR+11
00024'023623            LDA 0,@NW,3             ;NUMBER OF WORDS
00025'040420            STA 0,PAR+12
00026'025611            LDA 1,IBUF,3            ;START ADDRESS OF TEXT
00027'030404            LDA 2,PAR               ;PARAMETER POINTER
00030'006402            JSR @ET                 ;GO TO ETXT ROUTINE
00031'006002$           JSR @.FRET              ;RETURN TO FORTRAN
                ;
```

```
 00032'077777 ET:    ETXT2    ;ENTRY POINT TO ETXT...TEXT WITH CLS STORAGE

 0002  EFSTX
  00033'000034'PAR:  .+1
        000012       .BLK 10.

                              .END
```

```
  0003  EFSTX
CON     177620
EFSTX   000001'
ET      000032'
ETXT2   000032'X
FS.     000013
IBUF    177611
ICHN    177612
IZ      177617
NW      177623
OVY     177622
PAR     000033'
SCR     177621
SE      177616
SIZE    177615
X       177613
Y       177614
.CPYL   000001$X
.FRET   000002$X
```

I. IDENTIFICATION

Module name:	EFCLR
Date:	October 28, 1982
Function:	Clear the display
Language:	Assembly (FORTRAN interface)
Memory locations:	ZREL 0 NREL 22 STACK 4

II. FUNCTIONAL SUMMARY

EFCLR is a subroutine that will clear the display and delete the contents of CLS.

III. ENTRY POINTS

EFCLR

IV. CALLING METHOD

```
CALL EFCLR(ICHN,CON,SCR,OVY)
```

V. INPUT

ICHN	-	RDOS channel number
CON	-	Console number
SCR	-	Screen number
OVY	-	Overlay number

VI. OUTPUT

The designated screen is cleared.

VII. ERROR RETURNS

NONE

Module name: EFCLR Date: October 28, 1982

VIII. REFERENCED EXTERNALS

EGR1.LB

ECLR clears the screen

FORT.LB

.CPYL copies the return address of the calling program
.FRET returns to the calling program

```
0001  EFCLR
                ;*****************************************************************
                ;         EFCLR.SR   TOM EGGER/MQT 10-28-82                     *
                ;                                                               *
                ;    FORTRAN INTERFACE TO ASSEMBLY ECLR PROGRAM                 *
                ;                                                               *
                ;    CALLING SEQUENCE:                                          *
                ;                                                               *
                ;         CALL EFCLR(ICHN,CON,SCR,OVY)                          *
                ;                                                               *
                ;*****************************************************************
                ;
                          .TITL EFCLR
                          .ENT EFCLR
                          .EXTD .CPYL,.FRET
                          .EXTN ECLR
                          .NREL
                                                        ;SET UP THE STACK
      177611              ICHN=-167                     ;RDOS CHANNEL
      177612              CON=ICHN+1                    ;CONSOLE #
      177613              SCR=CON+1                     ;SCREEN # 1,2,3
      177614              OVY=SCR+1                     ;OVERLAY #
      000004              FS.=OVY-ICHN+1
00000'000004              FS.
                ;
00001'006001$EFCLR:       JSR @.CPYL                    ;COPY ADDRESS TO STACK
00002'023611              LDA 0,@ICHN,3                 ;RDOS CHANNEL #
00003'040414              STA 0,PAR+1                   ;STORE ON PARAMETER LIST
00004'023612              LDA 0,@CON,3                  ;CONSOLE #
00005'040413              STA 0,PAR+2                   ;STORE IT
00006'023613              LDA 0,@SCR,3                  ;SCREEN #
00007'040412              STA 0,PAR+3                   ;STORE IT
00010'023614              LDA 0,@OVY,3                  ;OVERLAY #
00011'040411              STA 0,PAR+4                   ;STORE IT
                ;
00012'030404              LDA 2,PAR                     ;PARAMETER POINTER
00013'006402              JSR @EC                       ;GO TO ASSEMBLY SUB ECLR
00014'006002$             JSR @.FRET                    ;RETURN TO FORTRAN
                ;
00015'077777 EC:    ECLR
00016'000017'PAR:   .+1
00017'000000        0     ;RDOS CHANNEL
00020'000000        0     ;CONSOLE
00021'000000        0     ;SCREEN
00022'000000        0     ;OVERLAY
                ;
                          .END

  0002  EFCLR
CON     177612
EC      000015'
ECLR    000015'X
EFCLR   000001'
FS.     000004
ICHN    177611
OVY     177614
PAR     000016'
SCR     177613
.CPYL   000001$X
```

```
.FRET   000002$X
```

I. IDENTIFICATION

Module names:	EPBYT unknown author
	EGBYT unknown author
Date:	Unknown
Functions:	EPBYT puts a byte in CORE
	EGBYT gets a byte from CORE
Language:	Assembly
Memory locations:	ZREL 0 NREL 21 STACK 0 (EPBYT)
	ZREL 0 NREL 14 STACK 0 (EGBYT)

II. FUNCTIONAL SUMMARY

EPBYT is a subroutine that will store a byte in CORE.
EGBYT is a subroutine that will get a byte from CORE.

III. ENTRY POINTS

EPBYT
EGBYT

IV. CALLING METHOD

The EPBYT routine is entered in the following manner: load AC0 with the byte (right adjusted) to be stored, load AC2 with the byte pointer to the storage in CORE, then do an indirect JSR to EPBYT.

```
                        LDA 0,BYT   ;right adjusted
                        LDA 2,BTPT  ;byte pointer
                        JSR @DUMMY
DUMMY:    EPBYT             .
BTPT:     .+1*2
          .BLK xx     ;xx is the number of words reserved
```

The EGBYT routine is entered in the following manner: load AC2 with the byte pointer (bytes packed left to right).

```
                        LDA 2,BTPT
                        JSR @DUMMY
                        STA 0,BYT
                            .
DUMMY:    EGBYT
BTPT:     .+1*2
            .       ;storage area
BYT:      0
```

Module names: EPBYT
EGBYT

Date: unknown

V. INPUT

EPBYT requires the byte right adjusted in AC0 and the byte pointer in AC2. EGBYT requires the byte pointer in AC2.

VI. OUTPUT

EPBYT a byte is stored in CORE
EGBYT a byte (right adjusted) is returned from CORE

VII. ERROR RETURNS

NONE

VIII. REFERENCED EXTERNALS

NONE

```
0001  EPBYT
                ;
                ;               EPBYT.SR      UNKNOWN AUTHOR
                ;
                ;
                                .TITL EPBYT
                                .ENT EPBYT
                                .NREL
                ;
                ;ROUTINE:        EPBYT
                ;ENTRY FORM:     OTHER MODULE VIA JSR @ PBT
                ;PURPOSE:        STORE A BYTE IN CORE
                ;ENTER WITH:     AC0, BYTE RIGHT ADJUSTED, AC2, BYTE POINTER
                ;RETURN WITH:    AC0 & AC2, UNCHANGED AC1 & AC3, DESTROYED

00000'054417 EPBYT:     STA 3,SAV2    ;SAVE RETURN ADDRESS
00001'151220            MOVZR 2,2     ;GET ADDRESS
00002'025000            LDA 1,0,2     ;AND WHAT'S THERE
00003'151013            MOV# 2,2,SNC
00004'125300            MOVS 1,1      ;SWAP FOR LEFT HALF INSERT
00005'041000            STA 0,0,2     ;SAVE AC0 FOR A SEC
00006'020412            LDA 0,LMSK
00007'107400            AND 0,1
00010'021000            LDA 0,0,2     ;RESTORE
00011'107000            ADD 0,1
00012'151013            MOV# 2,2,SNC
00013'125300            MOVS 1,1        ;SWAP BACK
00014'045000            STA 1,0,2       ;SAVE IN CORE
00015'151100            MOVL 2,2        ;RESTORE AC2
00016'002401            JMP @ SAV2      ;AND SPLIT

00017'000000 SAV2:    0
00020'177400 LMSK:    177400

                                .END

  0002  EPBYT
EPBYT   000000'
LMSK    000020'
SAV2    000017'
```

```
0001  EGBYT
                ;
                ;               EGBYT.SR      UNKNOWN AUTHOR
                ;
                ;
                                .TITL EGBYT
                                .ENT EGBYT
                                .NREL

                ;ROUTINE:                EGBYT
                ;ENTRY FROM:             OTHER MODULE VIA JSR @ GBT
                ;PURPOSE:                GET A BYTE FROM CORE
                ;ENTER WITH:             AC2, BYTE POINTER (BYTES PACKED LEFT TO RIGHT;
                ;RETURN WITH:            AC0,BYTE RIGHT ADJUSTED
                ;                        AC2,BYTE POINTER UNCHANGED
                ;                        AC1 & 3, CONTENTS DESTROYED

  00000'054413 EGBYT:     STA 3,SAV1      ;SAVE RETURN ADDRESS
  00001'102400            SUB 0,0         ;CLEAR AC0
  00002'151220            MOVZR 2,2       ;GET ADDRESS
  00003'021000            LDA 0,0,2       ;AND WHAT'S THERE
  00004'151013            MOV* 2,2,SNC
  00005'101300            MOVS 0,0
  00006'024404            LDA 1,RMSK
  00007'123400            AND 1,0     ;
  00010'151100            MOVL 2,2    ;RESTORE AC2
  00011'002402            JMP @ SAV1

  00012'000377 RMSK:  377
  00013'000000 SAV1:  0

                                .END

   0002  EGBYT
EGBYT   000000'
RMSK    000012'
SAV1    000013'
```

I. IDENTIFICATION

Module name:	EFXRN
Date:	March 3, 1983
Function:	Generates a random number
Language:	Assembly
Memory location:	ZREL 0 NREL 22 STACK 1 ZREL 0 NREL 36 STACK 0 (for assembly RAND)

II. FUNCTIONAL SUMMARY

EFXRN is a FORTRAN interface that will generate random numbers. The input of a seed value is not necessary. The initial seed value is obtained from the system clock.

III. ENTRY POINTS

EFXRN

IV. CALLING METHOD

```
CALL EFXRN(NUM)
```

V. INPUT

NONE

VI. OUTPUT

A random number from -32767 to +32767 is returned in NUM.

VII. ERROR RETURNS

NONE

VIII. REFERENCED EXTERNALS

EGR1.LB

RAND the assembly random number generator written originally by Data General as .RAND.

Module name: EFXRN Date: March 3, 1983

FORT.LB

.CPYL copies the return address of the calling program
.FRET provides return to the calling program

NOTE: The Data General random number generator .RAND had a "bug" in it. The "bug" was fixed by the author, and renamed RAND. .RAND resides in the Data General MATH.LB.

```
0001  EFXRN
                ;*****************************************************************
                ;   EFXRN.SR            TOM EGGER  3-3-83                        *
                ;   FORTRAN INTERFACE TO REPAIRED DATA GENERAL RANDOM NUMBER GEN *
                ;*****************************************************************
                ;
                        .TITL EFXRN
                        .ENT EFXRN
                        .EXTD .CPYL,.FRET
                        .EXTN RAND
                        .NREL
                ;                               ;SET UP STACK
      177611 NUM=-167                           ;STORAGE ADDRESS OF RANDOM NUMBER
      000001 FS.=1
00000'000001 FS.
                ;
00001'006001$EFXRN:     JSR @.CPYL              ;SAVE REGISTERS
00002'054420            STA 3,FSP               ;STORE THE STACK POINTER
00003'020412            LDA 0,SEED              ;LOAD THE SEED VALUE
00004'101004            MOV 0,0,SZR             ;SKIP NEXT IF ZERO
00005'000407            JMP .+7                 ;ALREADY HAVE A SEED
00006'006017            .SYSTM                  ;INITIAL SEED COMES FROM CLOCK
00007'021003            .GTOD                   ;GET THE TIME OF DAY
00010'000401            JMP .+1                 ;ERROR RETURN
00011'107000            ADD 0,1                 ;AC0 SECONDS AC1 MINUTES
00012'147000            ADD 2,1                 ;AC2 HOURS
00013'044402            STA 1,SEED              ;STORE THE SEED
00014'006405            JSR @XRN                ;JUMP TO ASSEMBLY RANDOM NUMBER GEN
00015'000000 SEED:      0
00016'034404            LDA 3,FSP               ;SAVE THE STACK POINTER
00017'043611            STA 0,@NUM,3            ;STORE IT ON STACK AT NUM
00020'006002$           JSR @.FRET              ;RETURN TO FORTRAN
                ;
00021'077777 XRN:  RAND             ;ENTRY POINT TO ASMBLY RANDOM ROUTINE
00022'000000 FSP:  0                ;RESERVED FOR STACT POINTER

                                    .END
```

```
  0002  EFXRN
EFXRN   000001'
FSP     000022'
FS.     000001
NUM     177611
RAND    000021'X
SEED    000015'
XRN     000021'
.CPYL   000001$X
.FRET   000002$X
```

I. IDENTIFICATION

Module name:	EBIND
Date:	November 28,1982
Function:	Binary to decimal conversion
Language:	Assembly
Memory locations:	ZREL 1 NREL 40 STACK 2
	ZREL 0 NREL 51 STACK 0 (.BIND)

II. FUNCTIONAL SUMMARY

EBIND is a FORTRAN interface to the Data General routine .BIND. A Binary number is input, then converted to a decimal ASCII character string.

III. ENTRY POINTS

EBIND

IV. CALLING METHOD

```
CALL EBIND(INBIN,IDEC)
```

V. INPUT

An integer value is passed through INBIN.

VI. OUTPUT

IDEC is an ASCII character string terminated by a null word. The string has the following form:

```
+DDDDD(NULL)
-DDDDD(NULL)
```

VII. ERROR RETURNS

NONE

Module name: EBIND Date: November 28, 1982

VIII. REFERENCED EXTERNALS

EGR1.LB
- .BIND the Data General conversion routine resident in EGR1.LB
- EPBYT puts a byte in CORE routine

FORT.LB
- .CPYL copies the return address of the calling routine
- .FRET returns to the calling program

```
0001  EBIND
                ;**********************************************************
                ;     EFBIND.SR     TOM EGGER 11-28-82
                ;     FORTRAN INTERFACE TO D.G. BINARY TO DECIMAL
                ;     CONVERSION      CALL EBIND(INBIN,IDEC)
                ;**********************************************************
                      .TITL EBIND
                      .ENT EBIND,.PTCH
                      .EXTD .CPYL,.FRET
                      .EXTN .BIND,EPBYT
      000001          .TXTM 1
                      .ZREL
                ;
00000-000020'.PTCH:  PTCH
                      .NREL
                ;
      177611 INBIN= -167      ;ADDRESS OF INPUT BINARY
      177612 IDEC= -166       ;ADDRESS OF OUTPUT DECIMAL ARRAY
      000002 FS.=2
00000'000002 FS.
00001'006001$EBIND:     JSR @.CPYL        ;SAVE REGISTERS AND ACCUMULATORS
00002'054423            STA 3,FSP         ;SAVE FORTRAN STACK POINTER
00003'020430            LDA 0,BTPT        ;BYTE POINTER TO OUTPUT
00004'040426            STA 0,BTP
00005'027611            LDA 1,@INBIN,3     ;INPUT
00006'006422            JSR @BND
00007'034416            LDA 3,FSP          ;RESTORE STACK POINTER
00010'031612            LDA 2,IDEC,3
00011'024423            LDA 1,OTPT
00012'045000            STA 1,0,2
00013'024422            LDA 1,OTPT+1
00014'045001            STA 1,1,2
00015'024421            LDA 1,OTPT+2
00016'045002            STA 1,2,2
00017'006002$           JSR @.FRET
00020'054406 PTCH:      STA 3,SAV         ;SAVE RETURN ADDRESS
00021'030411            LDA 2,BTP         ;BYTE POINTER
00022'006407            JSR @PB           ;PUT A BYTE IN CORE
00023'010407            ISZ BTP
00024'002402            JMP @SAV          ;RETURN
00025'000000 FSP:  0
00026'000000 SAV:  0
00027'000066 N66:  66
00030'077777 BND:  .BIND
00031'077777 PB:   EPBYT
00032'000000 BTP:  0
00033'000070'BTPT:  .+1*2
      000005 OTPT:  .BLK 5


                                .END

  0002  EBIND
BND     000030'
BTP     000032'
BTPT    000033'
EBIND   000001'
EPBYT   000031'X
FSP     000025'
```

```
FS.     000002
IDEC    177612
INBIN   177611
N66     000027'
OTPT    000034'
PB      000031'
PTCH    000020'
SAV     000026'
.BIND   000030'X
.CPYL   000001$X
.FRET   000002$X
.PTCH   000000-
```

APPENDIX 1

EXAMPLE ASSEMBLY LANGUAGE PROGRAMS

Two complete assembly language programs, EBOX and EDUM, are included in this appendix. They demonstrate the use of the EGR1.LB routines. The source files were generated at the ADM console using the AFOS editor. The following commands initiated the editor:

```
M:F/EBOX.SR
M:F/EDUM.SR
```

The programs were then assembled using the Data General Extended Assembler, ASM.SV. The following commands initiated the assembler:

```
ASM EBOX
ASM ESYM.PS/S EDUM     ;EDUM uses the symbol table
```

No assembly errors were noted. It was time to load:

```
RLDR EBOX EGR1.LB
RLDR EDUM EGR1.LB
```

To run the programs just type the name of the program at the dasher.

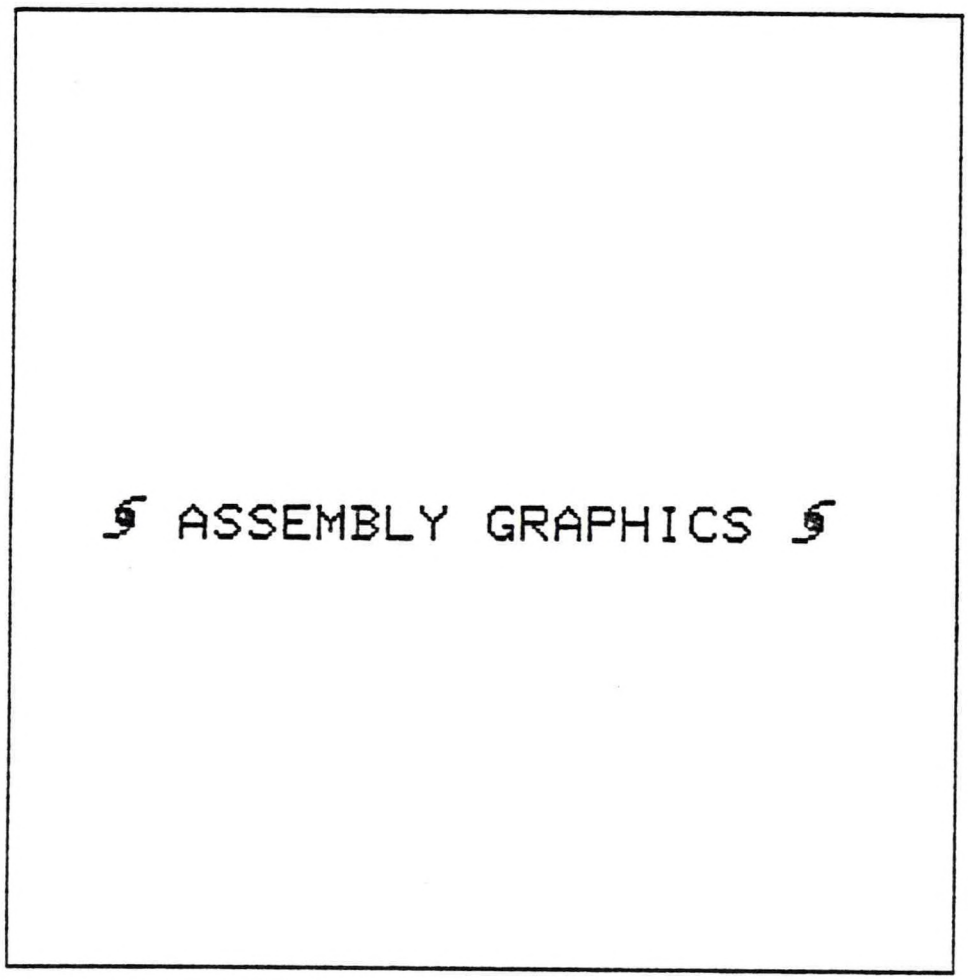
EBOX when activated will display a box on screen 2 of the master console. Near the center of the box is the text AFOS GRAPHICS. Preceding and succeeding the text are hurricane symbols. Clear the screen, then hit the "enter cursor" button; the image is restored. Move the cursor to the bottom of the screen (a zero Y value) and depress the "enter cursor" button. The image is erased and the program terminates.

EDUM when executed will display four lines that make a border around screen 2 of the master console. At the center of the screen are displayed the current X and Y coordinate values of the cursor's location. If the cursor is moved the values will change instantly. Informational text is contained in the legend block.

```
0001  EBOX
                ;**********************************************************
                ;       EBOX.SR    TJE/MQT 11/07/82
                ;       ASSEMBLY PROGRAM TO DEMONSTRATE THE USE
                ;       EGR.LB ROUTINES
                ;**********************************************************
                ;
                        .TITL EBOX
                        .EXTN LINE2,ETXT2,ECSR,ECLR,ELEG
                        .ENT START
      000001            .TXTM 1
                        .NREL

00000'020507 START:     LDA 0,NGDM          ;BYTE POINTER TO GDM
00001'006017            .SYSTM
00002'014001            .OPEN 1             ;OPEN ON CHANNEL #1
00003'000512            JMP ER              ;ERROR RETURN
00004'030427 BOX1:      LDA 2,PAR           ;WORD POINTER TO PARAMETERS
00005'006441            JSR @EL             ;DRAW OVER
00006'024427            LDA 1,PAR+2         ;X1
00007'020440            LDA 0,N2000         ;2000.
00010'107000            ADD 0,1
00011'044424            STA 1,PAR+2         ;NEW X1
00012'044424            STA 1,PAR+3         ;NEW Y1
00013'030420            LDA 2,PAR           ;PARAMETER POINTER
00014'006432            JSR @EL             ;DRAW UP
00015'024422            LDA 1,PAR+4         ;X2
00016'020422            LDA 0,PAR+5         ;Y2
00017'040420            STA 0,PAR+4         ;NEW X2
00020'044420            STA 1,PAR+5         ;NEW Y2
00021'030412            LDA 2,PAR           ;PARAMETER POINTER
00022'006424            JSR @EL             ;DRAW BACK
00023'020424            LDA 0,N2000         ;2000.
00024'024411            LDA 1,PAR+2         ;X1
00025'106400            SUB 0,1
00026'044407            STA 1,PAR+2         ;NEW X1
00027'044407            STA 1,PAR+3         ;NEW Y1
00030'030403            LDA 2,PAR           ;PARAMETER POINTER
00031'006415            JSR @EL             ;DRAW DOWN
00032'000416            JMP TEXT
                ;
00033'000034'PAR:   .+1
00034'000001        1       ;RDOS CHANNEL # IN USE
00035'001750        1000.   ;X1 IN DECIMAL
00036'001750        1000.   ;Y1     "
00037'005670        3000.   ;X2     "
00040'001750        1000.   ;Y2     "
00041'000000        0       ;NO ERASE
00042'000000        0       ;ALL ZOOMS
00043'000000        0       ;MASTER CONSOLE
00044'000002        2       ;SCREEN #2
00045'000001        1       ;OVERLAY #1
                ;
00046'077777 EL:    LINE2       ;ENTRY POINT FOR ELINE STORED IN CLS
00047'003720 N2000: 2000.
                ;
00050'024420 TEXT:      LDA 1,TEX           ;TEXT POINTER
00051'030403            LDA 2,PAR1          ;TEXT PARAMETER POINTER
00052'006415            JSR @ET             ;DISPLAY TEXT
```

```
00053'000445                JMP LEGD            ;GO TO LEGEND

0002  EBOX
            ;
00054'000055'PAR1:    .+1
00055'000001          1     ;RDOS CHANNEL #
00056'002260          1200. ;X START
00057'003554          1900. ;Y START
00060'000003          3     ;DOUBLE SIZE
00061'000000          0     ;NO ERASE
00062'000000          0     ;DISPALY AT ALL ZOOMS
00063'000000          0     ;MASTER CONSOLE
00064'000002          2     ;SCREEN #2
00065'000001          1     ;OVERLAY #1
00066'000015          13.   ;NUMBER OF WORDS
            ;
00067'077777 ET:      ETXT2  ;STORED IN CLS
00070'000071'TEX:     .+1
                      .TXT "<22><171><21> ASSEMBLY GRAPHICS <22><171><21><00>"

00071'011171
00072'010440
00073'040523
00074'051505
00075'046502
00076'046131
00077'020107
00100'051101
00101'050110
00102'044503
00103'051440
00104'011171
00105'010400
00106'000000
            ;
00107'000220'NGDM:    .+1*2
                      .TXT "DP0:$GDM"

00110'042120
00111'030072
00112'022107
00113'042115
00114'000000
00115'006017 ER:            .SYSTM
00116'006400                .ERTN
00117'000776                JMP ER
            ;
00120'024750 LEGD:          LDA 1,TEX           ;TEXT WORD POINTER
00121'030417                LDA 2,PAR2          ;LEGEND PARAMETER LIST
00122'006413                JSR @ELG            ;DISPLAY LEGEND
00123'030415                LDA 2,PAR2          ;INTERROGATE CURSOR PAR LIST
00124'006412                JSR @ECS            ;READ CURSOR
00125'151005                MOV 2,2,SNR         ;SKIP IF Y >0
00126'000402                JMP OUT             ;TERMINATE
00127'000655                JMP BOX1            ;DRAW AGAIN
00130'030410 OUT:           LDA 2,PAR2          ;ECLR PARAMETER LIST
00131'006406                JSR @ECL            ;CLEAR THE SCREEN
00132'006017                .SYSTM              ;AND
00133'004400                .RTN                ;RETURN TO CLI
00134'000761                JMP ER              ;NEVER TAKEN
            ;
00135'077777 ELG:     ELEG
```

```
00136'077777 ECS:   ECSR
00137'077777 ECL:   ECLR

0003  EBOX
00140'000141'PAR2:  .+1   ;PAR2 SHARED AMONG ELEG,ECSR,ECLR
00141'000001        1     ;RDOS CHANNEL #
00142'000000        0     ;CONSOLE #
00143'000002        2     ;SCREEN #
00144'000001        1     ;OVERLAY FOR ECLR & ELEG


      000000'             .END START

 0004  EBOX
BOX1    000004'
ECL     000137'
ECLR    000137'X
ECS     000136'
ECSR    000136'X
EL      000046'
ELEG    000135'X
ELG     000135'
ER      000115'
ET      000067'
ETXT2   000067'X
LEGD    000120'
LINE2   000046'X
N2000   000047'
NGDM    000107'
OUT     000130'
PAR     000033'
PAR1    000054'
PAR2    000140'
START   000000'
TEX     000070'
TEXT    000050'
```



PAGE 01

```
0001  EDUM
                ;**********************************************************
                ;       EDUM.SR EXAMPLE PROGRAM  TJE/MQT 3-7-83
                ;**********************************************************
                ;
                                .TITL EDUM
                                .EXTN ECS,.BIND,ELEG,EPBYT,LINE2,ETXT1 ;EXNAL MODULES EGR1.LB
                                .ENT EDUM,.PTCH                 ;ENTRY POINTS
        000001                  .TXTM 1                         ;PACK TEXT BYTES LEFT TO RIGHT
                                .ZREL                           ;PAGE ZERO RELOCATION
00000-000126' .PTCH:    PTCH
                                .NREL                   ;NORMAL RELOCATION
00000'020433 EDUM:              LDA 0,NGDM              ;GDM BYTE POINTER
00001'006017                    .SYSTM
00002'014002                    .OPEN 2                 ;OPEN GDM
00003'000401                    JMP .+1                 ;SKIP ERROR RETURN
00004'024437                    LDA 1,TEX               ;PUT LEGEND ON SCREEN #2
00005'030457                    LDA 2,PAR               ;LEGEND PARAMETER WORD POINTER
00006'006433                    JSR @EL                 ;JUMP TO LEGEND SUBROUTINE
00007'030464                    LDA 2,PAR1              ;LINE PARAMETER WORD POINTER
00010'006432                    JSR @LN                 ;JUMP TO LINE ROUTINE
00011'024461                    LDA 1,N4095             ;NEW X1
00012'044463                    STA 1,PAR1+2            ;STORE AT NEW X1 ON PAR LIST
00013'024456                    LDA 1,N3071             ;NEW Y1
00014'044462                    STA 1,PAR1+3            ;STORE ON LINE PAR LIST
00015'030456                    LDA 2,PAR1              ;AC2 = PAR WORD POINTER
00016'006424                    JSR @LN                 ;DRAW SECOND LINE
00017'126400                    SUB 1,1                 ;AC1=0 NEW X2
00020'044457                    STA 1,PAR1+4            ;STORE ON LIST
00021'024450                    LDA 1,N3071             ;NEW Y2
00022'044456                    STA 1,PAR1+5            ;STORE ON LIST
00023'030450                    LDA 2,PAR1              ;PAR WORD POINTER
00024'006416                    JSR @LN                 ;DRAW THIRD LINE
00025'126400                    SUB 1,1                 ;NEW X1 AND Y1
00026'044447                    STA 1,PAR1+2            ;STORE ON LIST
00027'044447                    STA 1,PAR1+3            ;STORE ON LIST
00030'030443                    LDA 2,PAR1              ;PARAMETER WORD POINTER
00031'006411                    JSR @LN                 ;DRAW LAST LINE
00032'000454                    JMP READ
00033'000070'NGDM:      .+1*2                   ;BYTE POINTER
                        .TXT "DP0:$GDM"
00034'042120
00035'030072
00036'022107
00037'042115
00040'000000
00041'077777 EL:        ELEG                    ;LEGEND ROUTINE
00042'077777 LN:        LINE2                   ;LINE DRAWING CLS STORED
00043'000044'TEX:       .+1                     ;WORD POINTER
                        .TXT "TERMINATE PROGRAM AT (+0+0)     "
00044'052105
00045'051115
00046'044516
00047'040524
00050'042440
00051'050122
00052'047507
00053'051101
00054'046440
```

```
00055'040524

0002  EDUM
00056'020050
00057'025460
00060'025460
00061'024440
00062'020040
00063'020000
00064'000065'PAR:    .+1
00065'000002         2    ;CHANNEL *
00066'000000         0    ;CONSOLE *
00067'000002         2    ;SCREEN *
00070'000001         1    ;OVERLAY *
00071'005777 N3071:  3071.      ;SCREEN COORDINATES
00072'007777 N4095:  4095.      ;IN DECIMAL
00073'000074'PAR1:   .+1
00074'000002         2          ;CHANNEL *
00075'000000         0          ;X1
00076'000000         0          ;Y1
00077'007777         4095.      ;X2
00100'000000         0          ;Y2
00101'000000         0          ;SELECTIVE ERASE = NO
00102'000000         0          ;ZOOM RATIOS
00103'000000         0          ;CONSOLE *
00104'000002         2          ;SCREEN *
00105'000001         1          ;OVERLAY *

            ;
00106'020435 READ:          LDA 0,BTPT          ;BYTE POINTER TO X-Y TEXT
00107'040432                STA 0,BTP           ;SAVE IT
00110'030754                LDA 2,PAR           ;PARAMETER LIST FOR CURSOR
00111'006425                JSR @EC             ;READ CURSOR
00112'050422                STA 2,YVAL          ;SAVE Y VALUE
00113'125014                MOV# 1,1,SZR        ;IS X ZERO
00114'000403                JMP .+3             ;JUMP IF NOT
00115'151015                MOV# 2,2,SNR        ;IS Y ZERO
00116'000455                JMP OUT             ;QUIT
00117'006420                JSR @BND            ;CONVERT X TO ASCII
00120'024414                LDA 1,YVAL          ;RESTORE Y VALUE
00121'006416                JSR @BND            ;CONVERT Y TO ASCII
00122'024420                LDA 1,TEX2          ;WORD POINTER TO X-Y TEXT
00123'030435                LDA 2,PAR2          ;PARAMETER WORD POINTER
00124'006411                JSR @ETX            ;DISPLAY X-Y VALUES
00125'000761                JMP READ
00126'054405 PTCH:          STA 3,SAV
00127'030412                LDA 2,BTP           ;BYTE POINTER
00130'006410                JSR @PB             ;PUT A BYTE IN CORE
00131'010410                ISZ BTP             ;INCREMENT POINTER
00132'002401                JMP @SAV            ;RETURN TO GET ANOTHER BYTE
00133'000000 SAV:    0
00134'000000 YVAL:   0
00135'077777 ETX:    ETXT1      ;DISPLAY X-Y TEXT NO CLS STORAGE
00136'077777 EC:     ECS        ;INTERROGATE CURSOR ENTRY
00137'077777 BND:    .BIND      ;BINARY TO DECIMAL ASC
00140'077777 PB:     EPBYT      ;PUT A BYTE IN CORE
00141'000000 BTP:    0
00142'000144'TEX2:    .+2       ;WORD POINTER
00143'000310"BTPT:    .+1*2     ;BYTE POINTER
      000014          .BLK 12.  ;RESERVED FOR X-Y TEXT
```

```
00160'000161'PAR2:    .+1
00161'000002          2          ;CHANNEL *

0003  EDUM
00162'003100          1600.      ;X LOCATION
00163'002734          1500.      ;Y LOCATION
00164'000000          0          ;CHARACTER SIZE
00165'000000          0          ;NO ERASE
00166'000000          0          ;ZOOM RATIOS
00167'000000          0          ;CONSOLE *
00170'000002          2          ;SCREEN *
00171'000001          1          ;OVERLAY
00172'000014          12.        ;NW

00173'006017 OUT:     .SYSTM     ;TERMINATE PROGRAM
00174'004400          .RTN
00175'000401           JMP .+1

      000000'                .END EDUM

  0004  EDUM
BND     000137'
BTP     000141'
BTPT    000143'
EC      000136'
ECS     000136'X
EDUM    000000'
EL      000041'
ELEG    000041'X
EPBYT   000140'X
ETX     000135'
ETXT1   000135'X
LINE2   000042'X
LN      000042'
N3071   000071'
N4095   000072'
NGDM    000033'
OUT     000173'
PAR     000064'
PAR1    000073'
PAR2    000160'
PB      000140'
PTCH    000126'
READ    000106'
SAV     000133'
TEX     000043'
TEX2    000142'
YVAL    000134'
.BIND   000137'X
.PTCH   000000-
```

+01604+01464

TERMINATE PROGRAM AT (+0+0)

APPENDIX 2

EXAMPLE FORTRAN PROGRAM

ELN4 is a complete program written in FORTRAN. The source file, ELN4.FR, was generated at an ADM using the AFOS editor. The following command was used to initiate the editor:

```
M:F/ELN4.FR
```

The program was then compiled with the FORTRAN compiler:

```
FORT ELN4
```

Loading the program was accomplished in the following format:

```
RLDR/P ELN4 EGR1.LB UTIL.LB FORT.LB
```

To execute the program, just type the name of the program at the dasher.

ELN4 will display a triangle with the statement AFOS and a thunderstorm symbol within its borders on screen 1. Screen 2 will display two intersecting vectors with the same statement in double size characters. The screen 1 image will be in reverse video.

At this time it is good to note the two different display modes. Screen 2's image is not destroyed by tampering with the zoom controls, while screen 1's image is. Now depress the "enter cursor" button for screen 2. If the cursor location is not in the lower left of the screen, the image is restored; otherwise the program will terminate.

```
; DGC FORTRAN IV REV 05.10ES

; C
; C     FORTRAN PROGRAM TO DEMONSTRATE
; C     ASSEMBLY INTERFACE SUBROUTINES  ELN4.FR    11-06-82
; C
;       INTEGER IBUF(34)
; C
;       CALL OPENN(0,"$GDM",0,IER)        ;OPEN GDM ON CHANNEL 0
; C
;  10   CALL EFLINE(0,3000,3000,1000,1000,0,0,0,2,1)  ;NO CLS STORAGE
;       CALL EFLINE(0,3000,1000,1000,3000,0,0,0,2,1)  ;NO CLS STORAGE
; C
;       IBUF(1)="AF"
;       IBUF(2)="OS"
;       IBUF(3)=" <22>"       ;START SPECIAL CHARACTERS
;       IBUF(4)="<51><21>"    ;THUNDERSTORM SYMBOL..END SPECIAL CHARACTERS
; C
;       CALL EFTXT(IBUF,0,1800,2000,3,0,0,0,2,1,4) ;NO CLS STORAGE
; C
;       CALL EFSLINE(0,1000,1000,2000,1000,0,0,0,1,1);CLS
;       CALL EFSLINE(0,1000,1000,1500,3000,0,0,0,1,1);CLS
;       CALL EFSLINE(0,1500,3000,2000,1000,0,0,0,1,1);CLS
; C
;       CALL EFSTXT(IBUF,0,1475,2000,1,0,0,0,1,1,4)  ;CLS
; C
;       CALL EFCSR(0,0,2,IX,IY)           ;INTERROGATE CURSOR ON SCR 2
;       IF(IX.LT.100 .AND. IY.LT.100)CALL EXIT
; C
;       CALL EFCLR(0,0,1,1)     ;CLEAR SCREEN 1
;       CALL EFCLR(0,0,2,1)     ;CLEAR SCREEN 2
; C
;       IBUF(1)="LE"
;       IBUF(2)="GE"
;       IBUF(3)="ND"
;       IBUF(4)="  "
; C
;       CALL EFLEG(IBUF,0,0,2,1)  ;OUTPUT LEGEND
;       GOTO 10
;       STOP
;       END

  0001  .MAIN

                ; DGC FORTRAN IV REV 05.10ES


                ; C

                ; C     FORTRAN PROGRAM TO DEMONSTRATE

                ; C     ASSEMBLY INTERFACE SUBROUTINES  ELN4.FR    11-06-82

                ; C

                ;       INTEGER IBUF(34)

                ; C
```

```
                ;       CALL OPENN(0,"$GDM",0,IER)       ;OPEN GDM ON CHANNEL 0
                        .NREL
                        .TITL   .MAIN
                        .ENT    .MAIN
                        .NREL
      000001            .TXTM   1
                        .EXTU
                        .EXTN   .I
               .F1:
00000'000000 .F2:       0
      000000            .CSIZ   0
00001'000010            FS.
               .MAIN:
00002'006001$           JSR     @.FALO
00003'000420'           A1.
00004'000011            V.+0
00005'000042            42
00006'002401            JMP     @.+1
00007'000010'           L1.
               L1.:

00010'002401            JMP     @.+1
00011'000015'           L2.
               L3.:     .TXT    "$GDM"
00012'022107
00013'042115
00014'000000
               L2.:
                        .EXTN   OPENN
00015'006002$           JSR     @.FCAL
00016'077777            OPENN
00017'000004            4
00020'000436'           .C3
00021'000012'           L3.
00022'000436'           .C3
00023'000014            V.+3            ;IER

               ; C

               ;  10   CALL EFLINE(0,3000,3000,1000,1000,0,0,0,2,1)  ;NO CLS STORAGE
               L4.:
                        .EXTN   EFLINE

0002  .MAIN
 00024'006002$          JSR     @.FCAL
 00025'077777           EFLINE
 00026'000012           12
 00027'000436'          .C3
 00030'000435'          .C4
 00031'000435'          .C4
 00032'000434'          .C5
 00033'000434'          .C5
 00034'000436'          .C3
 00035'000436'          .C3
 00036'000436'          .C3
 00037'000433'          .C6
 00040'000437'          .C2

                ;       CALL EFLINE(0,3000,1000,1000,3000,0,0,0,2,1)  ;NO CLS STORAGE
```

```
                        .EXTN   EFLINE
00041'006002$           JSR     @.FCAL
00042'000025'           EFLINE
00043'000012            12
00044'000436'           .C3
00045'000435'           .C4
00046'000434'           .C5
00047'000434'           .C5
00050'000435'           .C4
00051'000436'           .C3
00052'000436'           .C3
00053'000436'           .C3
00054'000433'           .C6
00055'000437'           .C2

                ; C

                ;       IBUF(1)="AF"
00056'006004$           JSR     @.FSUB
00057'000003            3
00060'000011            V.+0                    ;IBUF
00061'000017            VS.+1
00062'000437'           .C2
00063'002401            JMP     @.+1
00064'000067'           L5.
                L6.:    .TXT    "AF"
00065'040506
00066'000000
                L5.:
00067'006005$           JSR     @.LD0
00070'000065'           L6.
00071'043617            STA     0,@TS.+1,3

                ;       IBUF(2)="OS"
00072'006004$           JSR     @.FSUB
00073'000003            3
00074'000011            V.+0                    ;IBUF
00075'000017            VS.+1
00076'000433'           .C6
00077'002401            JMP     @.+1
00100'000103'           L7.
                L10.:   .TXT    "OS"
00101'047523
00102'000000

0003  .MAIN
                L7.:
00103'006005$           JSR     @.LD0
00104'000101'           L10.
00105'043617            STA     0,@TS.+1,3

                ;       IBUF(3)=" <22>"     ;START SPECIAL CHARACTERS
00106'006004$           JSR     @.FSUB
00107'000003            3
00110'000011            V.+0                    ;IBUF
00111'000017            VS.+1
00112'000432'           .C7
00113'002401            JMP     @.+1
00114'000117'           L11.
                L12.:   .TXT    " <22>"
```

```
00115'020022
00116'000000
            L11.:
00117'006005$       JSR     @.LD0
00120'000115'       L12.
00121'043617        STA     0,@TS.+1,3

            ;       IBUF(4)="<51><21>"   ;THUNDERSTORM SYMBOL..END SPECIAL CHARACTERS
00122'006004$       JSR     @.FSUB
00123'000003        3
00124'000011        V.+0                        ;IBUF
00125'000017        VS.+1
00126'000431'       .C10
00127'002401        JMP     @.+1
00130'000133'       L13.
            L14.:   .TXT    "<51><21>"
00131'024421
00132'000000
            L13.:

            ; C
00133'006005$       JSR     @.LD0
00134'000131'       L14.
00135'043617        STA     0,@TS.+1,3

            ;       CALL EFTXT(IBUF,0,1800,2000,3,0,0,0,2,1,4) ;NO CLS STORAGE
                    .EXTN   EFTXT
00136'006002$       JSR     @.FCAL
00137'077777        EFTXT
00140'000013        13
00141'100012        @V.+1                       ;IBUF
00142'000436'       .C3
00143'000430'       .C11
00144'000427'       .C12
00145'000432'       .C7
00146'000436'       .C3
00147'000436'       .C3
00150'000436'       .C3
00151'000433'       .C6
00152'000437'       .C2
00153'000431'       .C10

            ; C

            ;       CALL EFSLINE(0,1000,1000,2000,1000,0,0,0,1,1);CLS

0004  .MAIN
                    .EXTN   EFSLINE
00154'006002$       JSR     @.FCAL
00155'077777        EFSLINE
00156'000012        12
00157'000436'       .C3
00160'000434'       .C5
00161'000434'       .C5
00162'000427'       .C12
00163'000434'       .C5
00164'000436'       .C3
00165'000436'       .C3
00166'000436'       .C3
00167'000437'       .C2
```

```
00170'000437'          .C2

                ;      CALL EFSLINE(0,1000,1000,1500,3000,0,0,0,1,1);CLS
                       .EXTN   EFSLINE
00171'006002$          JSR     @.FCAL
00172'000155'          EFSLINE
00173'000012           12
00174'000436'          .C3
00175'000434'          .C5
00176'000434'          .C5
00177'000426'          .C13
00200'000435'          .C4
00201'000436'          .C3
00202'000436'          .C3
00203'000436'          .C3
00204'000437'          .C2
00205'000437'          .C2

                ;      CALL EFSLINE(0,1500,3000,2000,1000,0,0,0,1,1);CLS
                       .EXTN   EFSLINE
00206'006002$          JSR     @.FCAL
00207'000172'          EFSLINE
00210'000012           12
00211'000436'          .C3
00212'000426'          .C13
00213'000435'          .C4
00214'000427'          .C12
00215'000434'          .C5
00216'000436'          .C3
00217'000436'          .C3
00220'000436'          .C3
00221'000437'          .C2
00222'000437'          .C2

              ; C

                ;      CALL EFSTXT(IBUF,0,1475,2000,1,0,0,0,1,1,4)  ;CLS
                       .EXTN   EFSTXT
00223'006002$          JSR     @.FCAL
00224'077777           EFSTXT
00225'000013           13
00226'100012           @V.+1                    ;IBUF
00227'000436'          .C3
00230'000425'          .C14
00231'000427'          .C12
00232'000437'          .C2

0005  .MAIN
00233'000436'          .C3
00234'000436'          .C3
00235'000436'          .C3
00236'000437'          .C2
00237'000437'          .C2
00240'000431'          .C10

              ; C

                ;      CALL EFCSR(0,0,2,IX,IY)          ;INTERROGATE CURSOR ON SCR 2
                       .EXTN   EFCSR
00241'006002$          JSR     @.FCAL
```

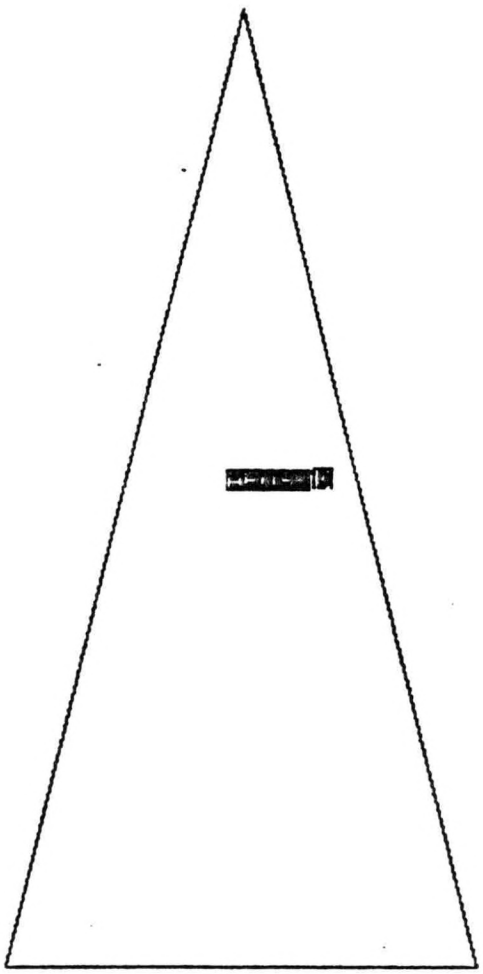
```
00242'077777          EFCSR
00243'000005          5
00244'000436'         .C3
00245'000436'         .C3
00246'000433'         .C6
00247'000015          V.+4            ;IX
00250'000016          V.+5            ;IY

                ;     IF(IX.LT.100 .AND. IY.LT.100)CALL EXIT
00251'021615          LDA     0,T.+4,3        ;IX
00252'006006$         JSR     @.LD1
00253'000424'         .C15
00254'152620          SUBZR   2,2
00255'133400          AND     1,2
00256'113100          ADDL    0,2
00257'106003          ADC     0,1,SNC
00260'126401          SUB     1,1,SKP
00261'126000          ADC     1,1
00262'031616          LDA     2,T.+5,3        ;IY
00263'006005$         JSR     @.LD0
00264'000424'         .C15
00265'045617          STA     1,TS.+1,3
00266'126620          SUBZR   1,1
00267'107400          AND     0,1
00270'147100          ADDL    2,1
00271'142003          ADC     2,0,SNC
00272'102401          SUB     0,0,SKP
00273'102000          ADC     0,0
00274'025617          LDA     1,TS.+1,3
00275'123400          AND     1,0
00276'101004          MOV     0,0,SZR
00277'000403          JMP     .+3
00300'002401          JMP     @.+1
00301'000305'         L15.

              ; C
                      .EXTN   EXIT
00302'006002$         JSR     @.FCAL
00303'077777          EXIT
00304'000000          0
              L15.:


                ;     CALL EFCLR(0,0,1,1)     ;CLEAR SCREEN 1
                      .EXTN   EFCLR
00305'006002$         JSR     @.FCAL
00306'077777          EFCLR

0006  .MAIN
00307'000004          4
00310'000436'         .C3
00311'000436'         .C3
00312'000437'         .C2
00313'000437'         .C2

                ;     CALL EFCLR(0,0,2,1)     ;CLEAR SCREEN 2
                      .EXTN   EFCLR
00314'006002$         JSR     @.FCAL
00315'000306'         EFCLR
00316'000004          4
```

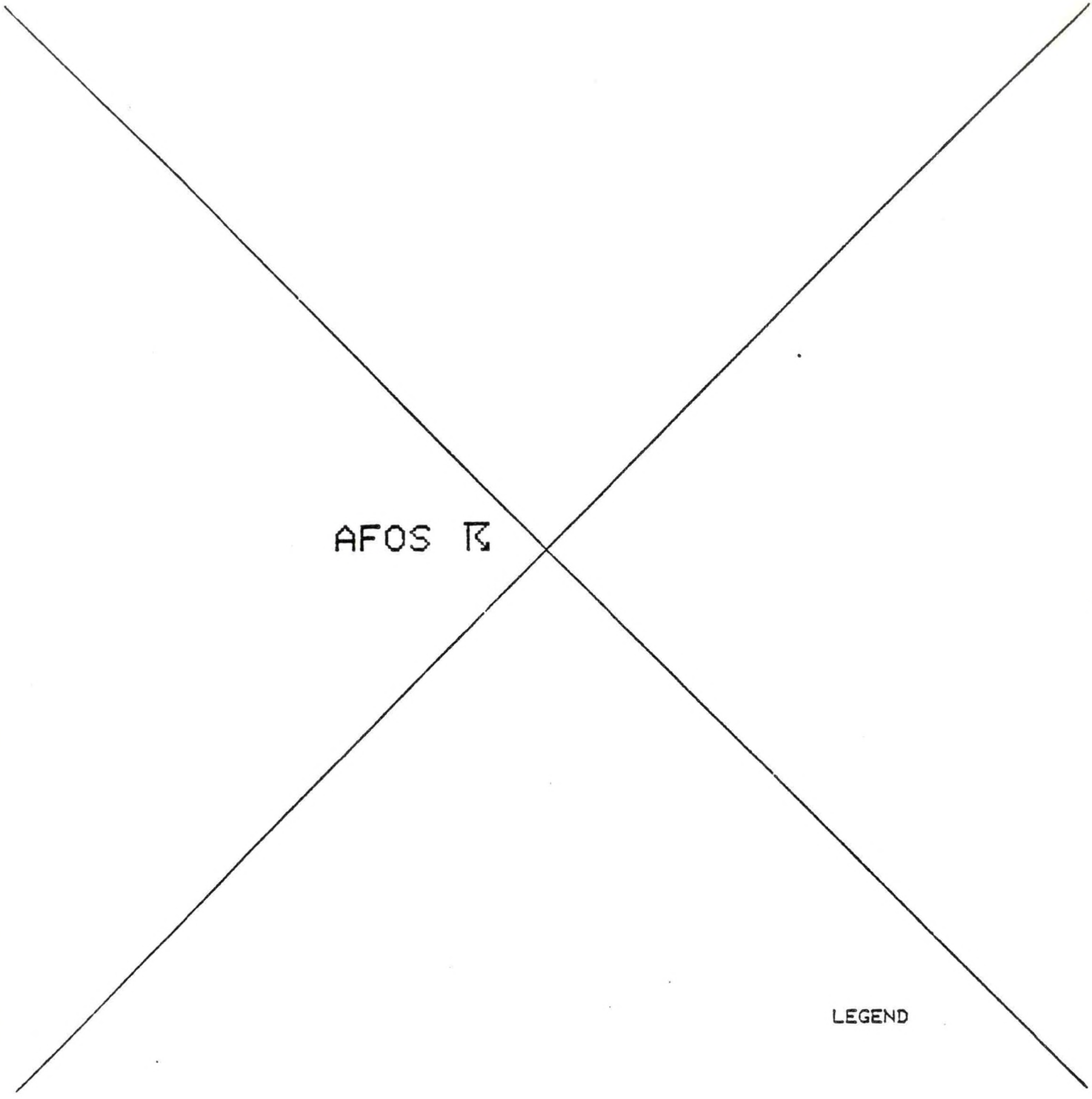
```
00317'000436'          .C3
00320'000436'          .C3
00321'000433'          .C6
00322'000437'          .C2

                ; C

                ;      IBUF(1)="LE"
00323'006004$          JSR     @.FSUB
00324'000003           3
00325'000011           V.+0                        ;IBUF
00326'000017           VS.+1
00327'000437'          .C2
00330'002401           JMP     @.+1
00331'000334'          L16.
                L17.:  .TXT    "LE"
00332'046105
00333'000000
                L16.:
00334'006005$          JSR     @.LD0
00335'000332'          L17.
00336'043617           STA     0,@TS.+1,3

                ;      IBUF(2)="GE"
00337'006004$          JSR     @.FSUB
00340'000003           3
00341'000011           V.+0                        ;IBUF
00342'000017           VS.+1
00343'000433'          .C6
00344'002401           JMP     @.+1
00345'000350'          L20.
                L21.:  .TXT    "GE"
00346'043505
00347'000000
                L20.:
00350'006005$          JSR     @.LD0
00351'000346'          L21.
00352'043617           STA     0,@TS.+1,3

                ;      IBUF(3)="ND"
00353'006004$          JSR     @.FSUB
00354'000003           3
00355'000011           V.+0                        ;IBUF
00356'000017           VS.+1
00357'000432'          .C7
00360'002401           JMP     @.+1
00361'000364'          L22.
                L23.:  .TXT    "ND"

0007   .MAIN
00362'047104
00363'000000
                L22.:
00364'006005$          JSR     @.LD0
00365'000362'          L23.
00366'043617           STA     0,@TS.+1,3

                ;      IBUF(4)="  "
00367'006004$          JSR     @.FSUB
00370'000003           3
```

```
00371'000011            V.+0                        ;IBUF
00372'000017            VS.+1
00373'000431'           .C10
00374'002401            JMP     @.+1
00375'000400'           L24.
                L25.:   .TXT    "  "
00376'020040
00377'000000
                L24.:

                ; C
00400'006005$           JSR     @.LD0
00401'000376'           L25.
00402'043617            STA     0,@TS.+1,3

                ;       CALL EFLEG(IBUF,0,0,2,1)  ;OUTPUT LEGEND
                        .EXTN   EFLEG
00403'006002$           JSR     @.FCAL
00404'077777            EFLEG
00405'000005            5
00406'100012            @V.+1                       ;IBUF
00407'000436'           .C3
00410'000436'           .C3
00411'000433'           .C6
00412'000437'           .C2

                ;       GOTO 10
00413'002401            JMP     @.+1
00414'000024'           L4.

                ;       STOP
00415'006007$           JSR     @.STOP
                        .TXT    ""
00416'000000

                ;       END
00417'006003$           JSR     @.FRET
                A1.:                        ;IBUF
00420'000003            3
00421'000401            401
00422'000001            1
00423'000042            42
00424'000144 .C15:      000144
00425'002703 .C14:      002703
00426'002734 .C13:      002734
00427'003720 .C12:      003720
00430'003410 .C11:      003410
00431'000004 .C10:      000004
00432'000003 .C7:       000003

0008  .MAIN
00433'000002 .C6:       000002
00434'001750 .C5:       001750
00435'005670 .C4:       005670
00436'000000 .C3:       000000
00437'000001 .C2:       000001
00440'000042 .C1:       000042

      000010            FS.=10
      000000            SFS.=0
```

```
          177611          T.=-167
          000011          V.=200+T.
          177616          TS.=T.+5
          177611          FTS.=T.+0
          000016          VS.=V.+5
          000011          FVS.=V.+0
                          .END

  0009    .MAIN
A1.       000420'
EFCLR     000315'X
EFCSR     000242'X
EFLEG     000404'X
EFLIN     000042'X
EFSLI     000207'X
EFSTX     000224'X
EFTXT     000137'X
EXIT      000303'X
FS.       000010
FTS.      177611
FVS.      000011
L10.      000101'
L11.      000117'
L12.      000115'
L13.      000133'
L14.      000131'
L15.      000305'
L16.      000334'
L17.      000332'
L1.       000010'
L20.      000350'
L21.      000346'
L22.      000364'
L23.      000362'
L24.      000400'
L25.      000376'
L2.       000015'
L3.       000012'
L4.       000024'
L5.       000067'
L6.       000065'
L7.       000103'
OPENN     000016'X
SFS.      000000
TS.       177616
T.        177611
VS.       000016
V.        000011
.C1       000440'
.C10      000431'
.C11      000430'
.C12      000427'
.C13      000426'
.C14      000425'
.C15      000424'
.C2       000437'
.C3       000436'
.C4       000435'
.C5       000434'
.C6       000433'
```

```
.C7      000432'
.F1      000000'
.F2      000000'
.FALO    000001$X
.FCAL    000002$X
.FRET    000003$X
.FSUB    000004$X
.I       077777 X

  0010   .MAIN
.LD0     000005$X
.LD1     000006$X
.MAIN    000002'
.STOP    000007$X
```




AFOS
LEGEND

APPENDIX 3

```
;PERMANENT SYMBOL TABLE TO BE USED WITH ASM.SV

;ESYM.PS                11-08-82

;TWO ACCUMULATOR COMPARISONS  -  COMPARE 2 NUMBERS OF THE SAME SIGN

.DALC       SLT=  ADCZ#   0,0,SNC  ;SKIP ON LESS THAN
.DALC       SLE=  SUBZ#   0,0,SNC  ;SKIP ON LESS THAN OR EQUAL TO
.DALC       SEQ=  SUB#    0,0,SZR  ;SKIP ON EQUAL
.DALC       SNE=  SUB#    0,0,SNR  ;SKIP ON NOT EQUAL
.DALC       SGT=  SUBZ#   0,0,SZC  ;SKIP ON GREATER THAN
.DALC       SGE=  ADCZ#   0,0,SZC  ;SKIP ON GREATER THAN OR EQUAL TO


;ONE ACCUMULATOR COMPARISON - USED FOR 15 BIT SIGNED INTEGERS

.DALC       SZ=   MOV#    0,0,SZR  ;SKIP IF ZERO
.DALC       SN=   MOV#    0,0,SNR  ;SKIP IF NONZERO
.DALC       SMZ=  ADDO#   0,0,SEZ  ;SKIP IF MINUS OR ZERO
.DALC       SPZ=  MOVL#   0,0,SCZ  ;SKIP IF PLUS OR ZERO
.DALC       SPN=  ADDO#   0,0,SBN  ;SKIP IF PLUS NONZERO
.DALC       SMN=  MOVL#   0,0,SNC  ;SKIP IF MINUS NONZERO

;      ADDITIONAL INSTRUCTIONS
.DALC       IOR=100410             ;IOR ACS,ACD  1 1=1
.DALC       XOR=100510             ;XOR ACS,ACD  1 1=0
.DALC       HXL=101410             ;HXL n,AC     N=0,3
.DALC       HXR=101510             ;HXR N,AC     N=0,3
.DALC       ADI=100010             ;ADI N,AC     N=0,3
.DALC       SBI=100110             ;SBI N,AC     N=0,3
.DALC       DIV=153710             ;UNSIGNED DIVIDE
.DALC       MUL=143710             ;UNSIGNED MULTIPLY
.DALC       DIVS=157710            ;SIGNED DIVIDE
.DALC       MULS=147710            ;SIGNED MULTIPLY
;
;WHEN BUILDING THE GRAPHICS LIBRARY MANY OF THE ABOVE INSTRUCTIONS
;WERE USED.  ALL LOADING WAS ACCOMPLISHED VIA ASM.SV
;
;
R
```

APPENDIX 4

EGR.LB - GLOSSARY

RDOS CHANNEL #	Reffered to as ICHN is an identifier of a logical file in which I/O operations are performed.
SIZE	Refers to the character size and display format. 0=normal size block character 1=normal size reverse video 2=normal size 3=double size character
SE	Selective erase if set to one specifies that the processed image is to be erased from the contents of refresh memory.
IZ	Supress displays as a function of zoom ratios. 0=display at all ratios 1=display at 4:1 or higher 2=display at 9:1 or higher 3=display at 16:1 or higher
CON	Console number. 0=master console.
SCR	Screen number. 1,2,3
OVY	Overlay number. 1,2,3
NW	Number of character words. Two characters per word.
X,Y	Defines the image space coordinate start location of a vector or the lower left corner of the first character.
CLS	Computer language storage.
R	

NOAA SCIENTIFIC AND TECHNICAL PUBLICATIONS

The National Oceanic and Atmospheric Administration was established as part of the De of Commerce on October 3, 1970. The mission responsibilities of NOAA are to assess the socioecon ict of natural and technological changes in the environment and to monitor and predict the state of the th, the oceans and their living resources, the atmosphere, and the space environment of the Earth.

The major components of NOAA regularly produce various types of scientific and technical information in the following kinds of publications:

PROFESSIONAL PAPERS — Important definitive research results, major techniques, and special investigations.

CONTRACT AND GRANT REPORTS — Reports prepared by contractors or grantees under NOAA sponsorship.

ATLAS — Presentation of analyzed data generally in the form of maps showing distribution of rainfall, chemical and physical conditions of oceans and atmosphere, distribution of fishes and marine mammals, ionospheric conditions, etc.

TECHNICAL SERVICE PUBLICATIONS — Reports containing data, observations, instructions, etc. A partial listing includes data serials; prediction and outlook periodicals; technical manuals, training papers, planning reports, and information serials; and miscellaneous technical publications.

TECHNICAL REPORTS — Journal quality with extensive details, mathematical developments, or data listings.

TECHNICAL MEMORANDUMS — Reports of preliminary, partial, or negative research or technology results, interim instructions, and the like.

Information on availability of NOAA publications can be obtained from:

ENVIRONMENTAL SCIENCE INFORMATION CENTER
ENVIRONMENTAL DATA AND INFORMATION SERVICE
NATIONAL OCEANIC AND ATMOSPHERIC ADMINISTRATION
U.S. DEPARTMENT OF COMMERCE

Rockville, MD 20852