

NOAA Western Region Computer Programs and
Problems NWS WRCP - No. 28



QC
874.3
.U68
no.28
c.2

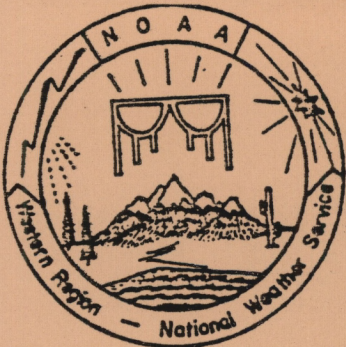
AFOS INTERACTIVE GRAPHICS

Salt Lake City, Utah
April 1981

U.S. DEPARTMENT OF
COMMERCE

/ National Oceanic and
Atmospheric Administration

/ National Weather
Service



PREFACE

This Western Region publication series is considered as a subset of our Technical Memorandum series. This series will be devoted exclusively to the exchange of information on and documentation of computer programs and related subjects. This series was initiated because it did not seem appropriate to publish computer program papers as Technical Memoranda; yet, we wanted to share this type of information with all Western Region forecasters in a systematic way. Another reason was our concern that in the developing AFOS-era there will be unnecessary and wasteful duplication of effort in writing computer programs in National Weather Service (NWS). Documentation and exchange of ideas and programs envisioned in this series hopefully will reduce such duplication. We also believe that by publishing the programming work of our forecasters, we will stimulate others to use these programs or develop their own programs to take advantage of the computing capabilities AFOS makes available.

We solicit computer-oriented papers and computer programs from forecasters for us to publish in this series. Simple and short programs should not be prejudged as unsuitable.

The great potential of the AFOS-era is strongly related to local computer facilities permitting meteorologists to practice in a more scientific environment. It is our hope that this new series will help in developing this potential into reality.

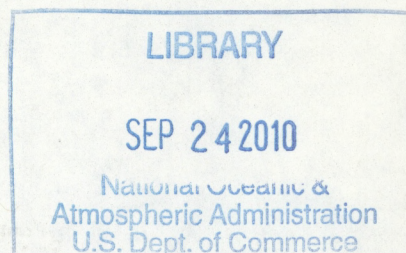
NOAA WESTERN REGION COMPUTER PROGRAMS AND PROBLEMS NWS WRCP

- 1 Standardized Format for Computer Series.
- 2 AFOS Crop and Soil Information Report Programs. Ken Mielke, July 1979.
- 3 Decoder for Significant Level Transmissions of Raobs. John A. Jannuzzi, August 1979.
- 4 Precipitable Water Estimate. Elizabeth Morse, October 1979.
- 5 Utah Recreational Temperature Program. Kenneth M. Labas, November 1979.
- 6 Normal Maximum/Minimum Temperature Program for Montana. Kenneth Mielke, December 1979.
- 7 Plotting of Ocean Wave Energy Spectral Data. John R. Zimmerman, December 1979.
- 8 Raob Plot and Analysis Routines. John Jannuzzi, January 1980.
- 9 The SWAB Program. Morris S. Webb, Jr., April 1980. (PB80-196041)
- 10 Flash-Flood Procedure. Donald P. Laurine and Ralph C. Hatch, April 1980. (PB80-298658)
- 11 Program to Forecast Probability of Summer Stratus in Seattle Using the Durst Objective Method. John Zimmerman, May 1980.
- 12 Probability of Sequences of Wet and Dry Days. Hazen H. Bedke, June 1980. (PB80-223340)
- 13 Automated Montana Hourly Weather Roundup. Joe L. Johnston, July 1980. (PB81-102576)
- 14 Lightning Activity Levels. Mark A. Mollner, July 1980. (PB81-108300)
- 15 Two Fortran Applications of Wind-Driven Ekman Water Transport Theory: Upwelling Index and Storm Tide. Kent S. Short, July 1980. (PB81-102568)
- 16 AFOS System Local Database Save and Rebuild Procedures or A Master Doomsday Program. Brian W. Finke, July 1980. (PB81-108342)
- 17 AFOS/RDOS Translator Subroutine. Morris S. Webb, Jr., August 1980. (PB81-108334)
- 18 AFOS Graphics Creation from Fortran. Alexander E. MacDonald, August 1980.
- 19 DATAKEYØ Repair Program. Paul D. Tolleson, August 1980. (PB81-102543)
- 20 Contiguous File Transfer from the DPCM to the DCM. Paul D. Tolleson, Sept. 1980. (PB81-128035)
- 21 Freezing Level Program. Kenneth B. Mielke, September 1980. (PB81-128043)
- 22 Radar Boresighting Verification Program. Thomas E. Adler, November 1980. (PB81-182677)
- 23 Accessing the AFOS Database. Matthew Peroutka, January 1981. (PB81 190266)
- 24 AFOS Word Processor. Morris S. Webb, Jr., February 1981.
- 25 Automated Weather Log for Terminal Forecasting. John A. Jannuzzi, February 1981.
- 26 Program to Compute Downwind Concentrations from a Toxic Spill. John Zimmerman, February 1981.
- 27 Animation of AFOS Graphics. Joe Wakefield and Jim Fors, April 1981.

QC
874.3
.U68
no. 28
c. 2

NOAA Western Region Computer Programs and Problems NWS WRCP - No. 28

AFOS INTERACTIVE GRAPHICS



Jim Fors
National Weather Service Western Region Headquarters
Salt Lake City, Utah

Don Laurine
Colorado Basin River Forecast Center, NWSWR
Salt Lake City, Utah

Sandy MacDonald
Environmental Research Laboratories
Boulder, Colorado

April 1981



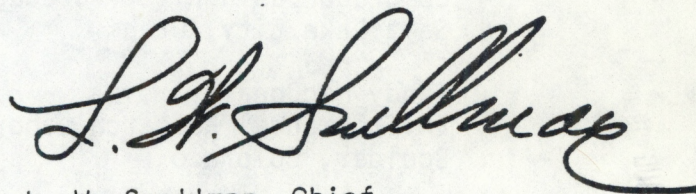
UNITED STATES
DEPARTMENT OF COMMERCE
Malcolm Baldrige, Secretary

NATIONAL OCEANIC AND
ATMOSPHERIC ADMINISTRATION
James P. Walsh, Acting Administrator

National Weather
Service
Richard E. Halkgren, Director



This Technical Memorandum has been
reviewed and is approved for
publication by Scientific Services
Division, Western Region.



L. W. Snellman, Chief
Scientific Services Division
Western Region Headquarters
Salt Lake City, Utah

AWs TECHNICAL LIBRARY
FL 4414
859 BUCHANAN STREET
SCOTT AFB IL 62225-5118



CONTENTS

	<u>Page</u>
List of Figures	iv
I. General Information	1
IIA. Application - Program INTERACT	2
IIB. Application - Program INXYRV	3
IIC. Application - Program INBUTF	4
III. Operating Instructions	4
IV. Cautions and Restrictions	5
V. Figures	6
VI. Complete Program Listing	10

FIGURES

	<u>Page</u>
Figure 1. Menu Displayed on GDM by INTERACT Program	6
Figure 2. Information Returned from System Call RDC	7
Figure 3. Formula to Calculate Image Space Coordinates	7
Figure 4. Formats for Interactive Vector and Text Strings	8
Figure 5. Standard and Special Character Sets Available with the INTERACT Program	9
Figure 6. Format for File IN	10

AFOS INTERACTIVE GRAPHICS

Jim Fors, SSD*
Don Laurine, CBRFC**
Sandy MacDonald, PROFS***

I. General Information

A. Summary

Graphic display systems such as AFOS bring many new features into Weather Service Offices. In Scientific Services Division (SSD), we have been very concerned in developing AFOS as a meteorological tool rather than just a communications device. We see innovative use of AFOS graphic display capabilities as one of these meteorological tools. In light of this, we have developed software to create AFOS graphics from application programs (WRCP-18) and animate AFOS graphics (WRCP-27). The programs described here add one more step to this process—the ability to interactively draw graphics on the GDM and have them stored into the AFOS database.

The value of interactive graphics software can be seen in many areas. One that immediately comes to mind is the problem of entering manually produced graphics into AFOS. Another is the potential of converting some of our current alphanumeric products into graphics. This might be useful for QPF forecasts, satellite discussions and the like.

The program documented here is a compilation of many different programs by many different authors. Also, there are several versions of interactive graphics packages around the country. The main difference between this and other packages is the ability to put text on the graphic and the ability to store the created graphic in the database.

B. Environment

The programs are written in Data General FORTRAN IV (one subroutine is written in assembly language). To run the program, an AFOS system with at least one graphic screen is required.

C. References

- 1) "AFOS Displays Programmers Reference", Ford Aerospace Corporation, WDL-TR7676A, Palo Alto, California.
- 2) MacDonald, Alexander E., 1981: "AFOS Graphics Creation from FORTRAN". WRCP-18, March, 22pp.
- 3) Wakefield, Joe, and Jim Fors, 1981: "Animation of AFOS Graphics". WRCP-27, April.

*SSD: National Weather Service Western Region Headquarters, Salt Lake City, Utah.
**CBRFC: Colorado Basin River Forecast Center, NWSWR, Salt Lake City, Utah.
***PROFS: Environmental Research Laboratories, NOAA, Boulder, Colorado.

IIA. Application - Program INTERACT

- A. The purpose of this program is to monitor cursor location input from the "Enter Cursor" button and take the appropriate action. All transactions that take place (vectors and text) are written to a disk file for use by other programs in this package. The console the operator is using and the AFOS product the graphic is to be stored in are specified in the ADM run line. Also, the ability to have a prestored legend put on the graphic is available to the user. Overlay 1 of the GDM screen closest to the ADM is used to draw the graphic.

The program initially draws a menu on the GDM (Figure 1). Placing the cursor in the appropriate box and hitting "Enter Cursor" puts the user in the mode specified (i.e., delete mode, text mode, etc.). The program then interprets the succeeding "Enter Cursor" entries appropriately (i.e., points in a vector, starting location of a text string, etc.). The graphic is stored in CLS memory but the menu shows up only at 1:1 (see WRCP-27 for a brief description of AFOS GDM hardware).

The location of the cursor is returned to the program from a call to the subroutine CURSR. Subroutine CURSR sets up the appropriate input for the system call .RDC---read cursor---and then calls the assembly subroutine RDCUR which executes the system call. The data returned from the system call is shown in Figure 2. The formulas for calculating the X, Y coordinates from this information are shown in Figure 3.

The program calls two subroutines (LN2D and RTEXT) to draw vectors or write text on the GDM. The format for these messages is shown in Figure 4.

The text to be put on the graphic is read from the ADM by the subroutine RDTEXT. A minor hardware modification and a one-word change in the RDOS operating system is required to use the text option. The user has available not only the standard character set but also the special character set shown in Figure 5. The details of these changes are not described here since approval to make these changes must be obtained from your Regional Headquarters and is made based on individual site requirements.

The end result of this program is a file called "IN". It contains the text and vector information specified by the user. The format of this file is shown in Figure 6. Text strings are written immediately to the disk file while the vector information is written as a group at the end of the program. A pair of -1's separates each vector string, and a pair of -2's indicates the end of file. INTERACT chains to a program called INXYRV upon completion.

The program uses the following subroutines and libraries:

- 1) CURSR.FR
- 2) RDCUR.SR
- 3) LN2D.FR
- 4) RTEXT.FR
- 5) RDTEXT.FR
- 6) UTIL.LB
- 7) AFOSE.LB

B. Machine Requirements

The program runs in less than 9K of memory. The save file is 9216 bytes in length and uses 3 I/O channels. The size of the output file varies depending on the size of the graphic created. As currently configured, the program will allow approximately 2000 vectors and an unlimited amount of text.

IIB. Application - Program INXYRV

A. Complete Program Description

This program converts the text and vector information resulting from the running of INTERACT into AFOS graphic format. Details of AFOS graphic format can be found in WRCP-18.

The program reads the AFOS output graphic that was specified in the ADM run line from file SK where it is written by INTERACT. It then calls subroutines COMMH and GPD1 to set up the AFOS header and the AFOS graphic product definition.

The program reads data from file IN one block at a time. Text information is converted to AFOS graphic format by a call to subroutine TEXT and point information is converted to relative vector format by subroutine MVEC. Subroutine NIB writes the processed graphic block to an output file called GP. When all the text and points are processed, the remaining space in the block is filled with -1's and is written to disk. Another block of all -1's is then written to disk. The program then chains to a program called INBUTF.

This program uses the following subroutines and libraries:

- 1) MVEC
- 2) TEXT
- 3) COMMH
- 4) GPD1
- 5) NIB
- 6) UTIL.LB
- 7) AFOSE.LB

B. Machine Requirements

The program runs in less than 9K of memory and the save file is 8192 bytes in length. The program uses two I/O channels. The output file—GP varies in length depending on the amount of text and number of vectors in the graphic.

IIC. Application - Program INBUTF

A. Complete Program Description

AFOS graphics are stored in the database in Universal Transmission Format. This is discussed in detail in WRCP-18. This program converts the file GP created by the program INXYRV one block at a time to Universal Transmission Format. The block of all -1's tells the program that the whole file has been processed. When the entire file is converted, it is stored in the database with a call to the BG.LB routine of FSTORE. A call to the BG.LB routine of FORK is made to alert the GDM that the graphic has been stored.

B. Machine Requirements

The program runs in less than 9K of memory. The save file is 6656 bytes in length. The program uses one I/O channel. The file GP is changed but no new files are created.

III. Operating Instructions

A. INTERACT

1) At ADM enter

RUN:INTERACT AA/C XXX/O N/H

where AA is the console number (be sure to use two digits);
if no AA is specified, 00 is assumed.

XXX is the output graphic.

N is the header number. If no N is specified, no header is brought up.

2) Menu will appear on screen 1 of the console specified.

3) To draw a vector,

- a) Place cursor in "NEXT" box and hit ENTER CURSOR.
- b) Move cursor to starting point of vector and hit ENTER CURSOR.
- c) Continue to draw vector by hitting ENTER CURSOR repeatedly at appropriate points.

4) To delete a vector,

- a) Place cursor in "DELETE" box. Each time ENTER CURSOR is hit, one vector will be deleted. They are deleted in the reverse order that they are created.

5) To add text,

- a) Place cursor in "TEXT" box and hit ENTER CURSOR.

- b) Place cursor at location where you want text string to start and hit ENTER CURSOR.
- c) Type in text string at ADM and hit ENTER and text string will appear on GDM (Max of 22 characters per string).

\$ Before text invokes large character set.

^ Before text invokes special character set.

- d) Move cursor to location for next string and repeat b) and C) until all text is entered.

- 6) Vectors and text can be intermixed in any order desired.
- 7) When finished, move cursor to "END" box and hit ENTER CURSOR.
- 8) When the graphic has been stored (about 30 seconds)...alert will go off at console saying "JOB INTERACT COMPLETED".

****NOTE:** If you desire a map background on the screen while drawing, it should be called up before RUN:INTERACT is initiated.

*******Instructions for Creating Header:**

Header is called up with menu and becomes a permanent part of the graphic. These headers are stored on disk and 9 different headers are allowed with a maximum of 4 lines per header. The headers are created as follows:

- 1) Enter at ADM

RUN:INTERACT AA/C XXX/0

- 2) Using "TEXT" mode, create your header (max of 4 lines).
- 3) Place cursor in "END" box and hit ENTER CURSOR.
- 4) At ADM enter:

RENAME:IN HEADERN

where N is the number that you will specify in your INTERACT run line (i.e., RENAME:IN HEADER3) with the /H switch.

IV. Cautions and Restrictions

As referenced earlier, the text mode will not function unless a minor hardware change is made and a one-word change is made to RDOS. If you enter the text mode without these changes, the program will hang and can only be killed with a CTRL-A at the Dasher. All other functions are operable without any changes.

Programming Credits:

GDCLR)
LN2D)
RTEXT)-----Don Laurine
CURSR)
RDCUR)

INTERACT-----Don Laurine, Mark Mathewson, Jim Fors

RDTEXT-----Jim Fors

XYRV)
MVEC)
COMMH)
GPD1)-----Sandy MacDonald, Jim Fors
TEXT)
NIB)
BUTF)

V. Figures

ENTER
DELETE
END
NEXT
TEXT

Figure 1. Menu displayed on GDM
by INTERACT program.

BIT																BIT	
0	1	2	3	4	7	8	9	10	13	14	15						
1	0	1	0	CONSOLE I.D.			CHAN										
				INPUT WD CT = 4													
ZOOM				X CURSOR										0	0		
				Y CURSOR										0	0		
				X TRANSLATE													
				Y TRANSLATE													

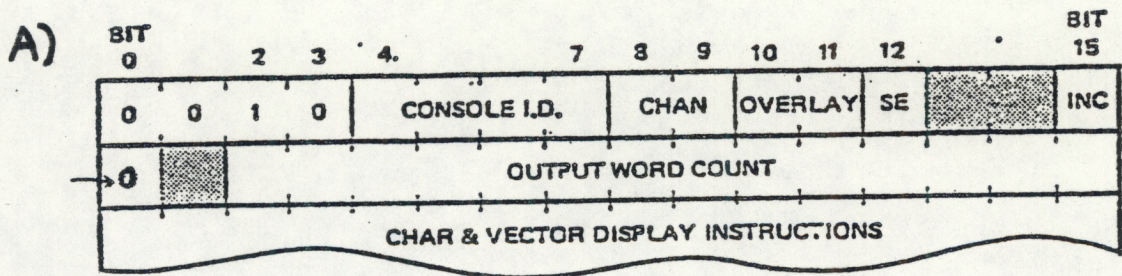
Figure 2. Information returned from system call RDC.

To calculate the image space coordinates from the data in the interrogate cursor position message, the following relationships apply:

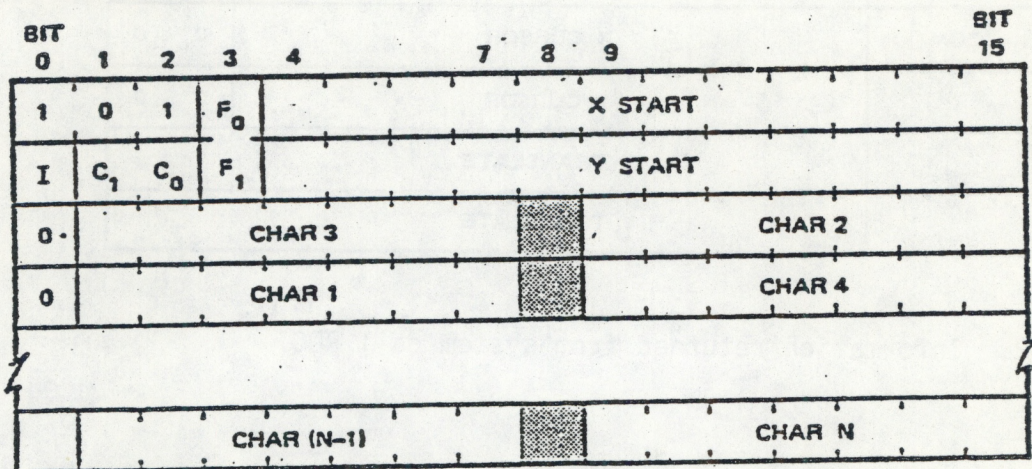
$$X_{\text{image}} = (X_{\text{cursor}} - 2047) / (\text{Zoom} + 1) + 2047 - X_{\text{translate}}$$

$$Y_{\text{image}} = (Y_{\text{cursor}} - 1535) / (\text{Zoom} + 1) + 1535 - Y_{\text{translate}}$$

Figure 3.



B) Character Instruction



C) Delta Segment Vector Instruction

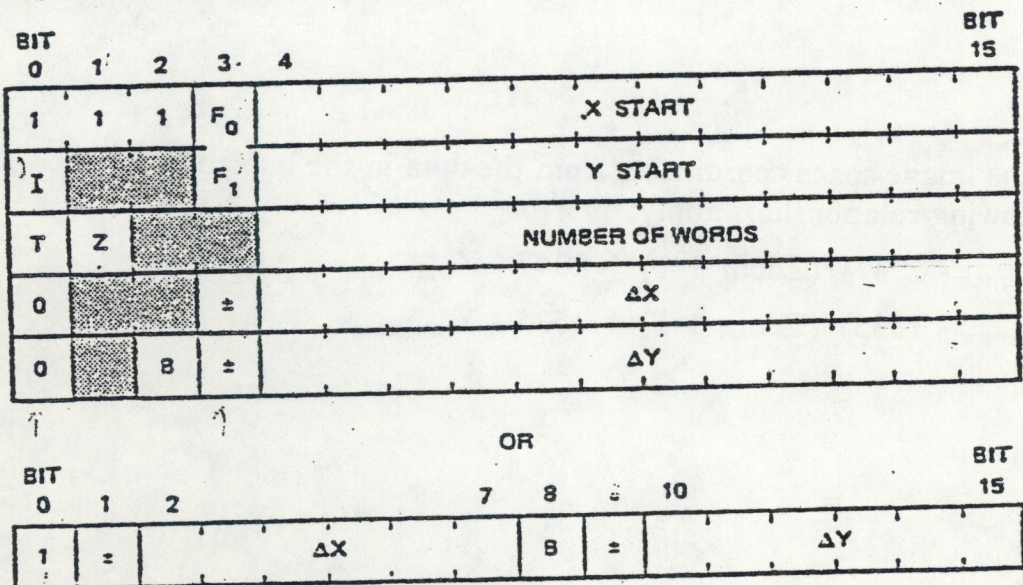


Figure 4. Formats for interactive vector and text strings.

STANDARD CHARACTER SET

	00	10	20	30	40	50	60	70	80	90	100	110	120
0	nul	down space			(	2	<	F	P	Z	d	n	x
1		up space			)	3	=	G	Q	[	e	o	y
2		1.5 new line		space	*	4	>	H	R	\	f	p	z
3		new line		!	+	5	?	I	S	]	g	q	{
4				"	,	6	@	J	T	^	h	r	!
5				#	-	7	A	K	U	_	i	s	}
6				\$	.	8	B	L	V	'	j	t	~
7		reset flag		%	/	9	C	M	W	a	k	u	£
8	back space	set flag		&	0	:	D	N	X	b	l	v	
9	fwd space			'	1	;	E	O	Y	c	m	w	

SPECIAL CHARACTER SET

	00	10	20	30	40	50	60	70	80	90	100	110	120
0	nul	down space	⊙	S	⊙	⌈	⊕	~	▲	H	M	Σ	5
1	⊕	up space	⊕	\$	⌈	⌈	⊕	∞	⌈	L	⌈	⌈	5
2	⊕	1.5 new line	⌈	⌈	⌈	⌈	⌈	⌈	⌈	⌈	⌈	⌈	⌈
3	⊕	new line	⌈	⌈	⌈	⌈	⌈	⌈	⌈	⌈	⌈	⌈	⌈
4	⊕	⊕	⌈	=	,	⌈	⌈	⌈	⌈	⌈	⌈	⌈	⌈
5	⊕	⊕	⌈	=	.	⌈	⌈	⌈	⌈	⌈	⌈	⌈	⌈
6	⊕	⊕	⌈	=	*	⌈	⌈	⌈	⌈	⌈	⌈	⌈	⌈
7	★	reset flag	⌈	⌈	⌈	⌈	⌈	⌈	⌈	⌈	⌈	⌈	⌈
8	back space	set flag	⌈	⌈	⌈	⌈	⌈	⌈	⌈	⌈	⌈	⌈	⌈
9	fwd space	⊕	∞	⌈	⌈	⌈	⌈	⌈	⌈	⌈	⌈	⌈	⌈

Figure 5. Standard and Special Character Sets available with the INTERACT Program.

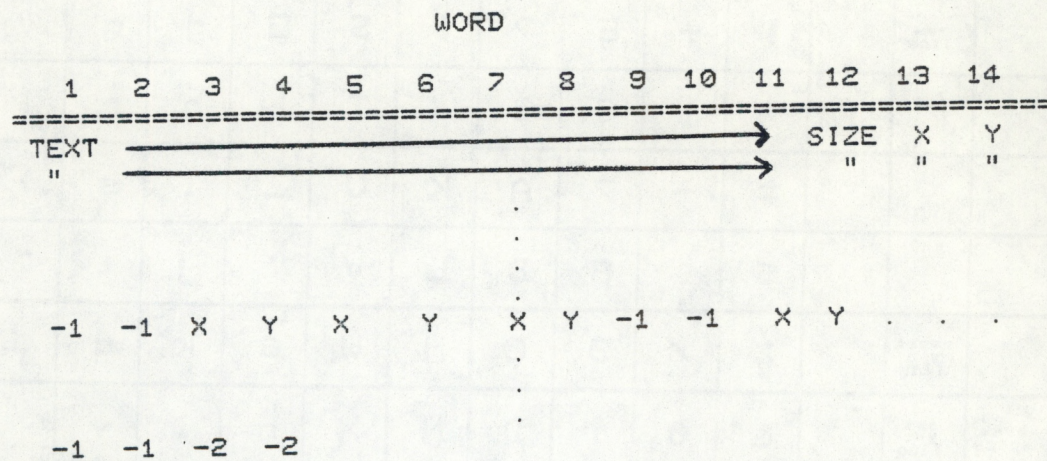


Figure 6. Format of File IN.

VI. Complete Program Listing


```

R
TYPE DP1:INTERACT.FR
C*****
C
C      PROGRAM: INTERACT
C
C      AUTHOR: DON LAURINE CBRFC 12/79
C      MODIFIED BY: MARK MATHEWSON SSD 1/80
C      MODIFIED BY: JIM FORS SSD 12/80
C
C      PURPOSE: DRAW VECTORS AND PUT TEXT ON THE GDM INTERACTIVELY
C                PROGRAM CHAINS TO OTHER PROGRAMS WITH THE END
C                RESULT BEING THE GRAPHIC STORED IN THE DATABASE.
C
C      SUBROUTINES
C          GDCLR  CLEARS GDM
C          LN2D   DRAWS LINES
C          RTEXT  WRITES TEXT
C          CURSR  SETS UP CALL TO RDCUR
C          RDCUR  READS LOCATION OF CURSOR
C          RTEXT  READS TEXT FROM THE ADM
C
C      RLDR/P/N INTERACT GDCLR LN2D RTEXT CURSR RDCUR RTEXT
C      UTIL.LB FORT.LB AFOSE.LB
C
C*****\
C      INTEGER DAT(7),SW(2)
C      INTEGER HFLG
C      INTEGER NAME(4)
C      INTEGER IBUF(14),X,Y,COMMD(3,7),LINE(1:2,0:2000)
C      COMMON/ONE/ COMMD
C      DATA COMMD/'ENTER'      DELETE      END      NEXT TEXT '/'
C  A NUMBER SPECIFIED IN THE ADM RUNLINE IS APPENDED TO THIS FILENAME
C  IF A HEADER IS TO BE CALLED UP.
C      NAME(1)="HE"
C      NAME(2)="AD"
C      NAME(3)="ER"
C      HFLG=0
C      ICON=0
C      ICHNL=1
C      IOURL=1
C      IARSZ = 2000
C
C  *** OPEN CHANNEL TO GDM
C
C      CALL GCHN(ICHN,IER)
C      CALL OPENN(ICHN,"#GDM",0,IER)
C
C  *** OPEN CHANNEL TO DISK FILE (X,Y)
C
C      CALL GCHN(JCHN,IER)
C      CALL DFILW ("IN",IER)
C      CALL CRAND ("IN",IER)
C      CALL OPENN (JCHN,"IN",0,IER)
C
C  READ IN COMMAND LINE FROM ADM USING UTIL LIB-ROUTINES FCOM AND
C  COMCM. "O" SWITCH IS OUTPUT GRAPHIC NAME AND "C"
C  SWITCH IS CONSOLE NUMBER. H SWITCH IS HEADER FILE NUMBER.

```



```

          CALL FCOM(IC,IER)
950      CALL COMCM(IC,DAT,N,SW,IER)
          IF(IER.EQ.9)GO TO 951
          IF(ISWSET(SW,"O"))GO TO 953
          IF(ISWSET(SW,"C"))GO TO 952
          IF(ISWSET(SW,"H"))GO TO 954
          GO TO 950
C  CONVERT CONSOLE NUMBER TO INTEGER. DEFAULT IS CONSOLE ZERO
952      CONTINUE
          ICON=(ISHFT(IAND(DAT(1),177400K),-8)-60K)*10
          ICON=ICON+(IAND(DAT(1),377K)-60K)
          GO TO 950
C  WRITE OUTPUT FILENAME TO FILE SK
953      CONTINUE
          DAT(2)=IAND(DAT(2),177400K)+60K
C  WRITE THE OUTPUT GRAPHIC NAME TO DISK FILE SK
          CALL GCHN(KCHN,IER)
          CALL CRAND("SK",IER)
          CALL OPENN(KCHN,"SK",0,IER)
          CALL WRS(KCHN,DAT,4,IER)
          CALL KLOSE(KCHN,IER)
          GO TO 950
C  APPEND NUMBER TO HEADER FILE NAME
954      CONTINUE
          NAME(4)=IAND(DAT(1),177400K)
          CALL GCHN(KCHN,IER)
          CALL OPENN(KCHN,NAME,0,IER)
          HFLG=1
          GO TO 950
951      CONTINUE
C
C *** INITIALIZE COUNTERS
C
          LLCNT=0
          NPTS=-1
          IER=1
          IFLAG=1
C
C **** CLEAR CHANNEL DISPLAY
C
          CALL GDCLR (ICHN,ICON,ICHLN,IOURL,IER)
C
          IF(HFLG.EQ.0)GO TO 2
C  WRITE THE HEADER TO THE GRAPHIC AND TO THE DISK FILE. A MAX
C  OF 4 LINES OF HEADER IS ALLOWED.
          DO 1 I=1,4
              LL=28
              CALL RDS(KCHN,IBUF,LL,IER)
              IF(IER.NE.1)GO TO 2
              X=IBUF(13)
              Y=IBUF(14)
              NN=IBUF(12)
              CALL RTEXT(ICHN,X,Y,IBUF,0,NN,0,ICON,ICHLN,IOURL,IER)
              CALL WRS(JCHN,IBUF,28,IER)
              LLCNT=LLCNT+14
          1      CONTINUE
          2      CONTINUE
C
C *** DRAW MENU
C

```



```

      IBUF(1)=3700
      IBUF(3) = 4095
      DO 400 I=0,7
      IBUF(2) = 3071 - I*200
      IBUF(4) = IBUF(2)
      CALL LN2D(ICHN,IBUF,0,0,ICON,ICHNL,IOURL,IER)
400  CONTINUE
C
      IBUF(1)=3700
      IBUF(3)=3700
      IBUF(2)=3071
      IBUF(4)=1671
      CALL LN2D(ICHN,IBUF,0,0,ICON,ICHNL,IOURL,IER)
      IBUF(1)=4095
      IBUF(3)=4095
      CALL LN2D(ICHN,IBUF,0,0,ICON,ICHNL,IOURL,IER)
C
      X=3765
      Y=3120
      DO 405 I=1,7
      DO 410 J=1,3
      IBUF(J)=COMMD(J,I)
410  CONTINUE
      IBUF(4)=' '
      IBUF(5)=0
      Y=Y-200
      CALL RTEXT(ICHN,X,Y,IBUF,0,0,0,ICON,ICHNL,IOURL,IER)
      405  CONTINUE
C
C *** DRAW LINES AND TEXT
C
C PROGRAM RETURNS TO HERE WHEN "NEXT" OPTION IS USED.
C IT LOADS IN A PAIR OF -1'S TO DENOTE THIS. IT ALSO RESETS
C OPTION FLAG TO 1. OPTION FLAG IS INTERPRETED AS FOLLOWS:
C
C          IFLAG=1  DRAW VECTOR MODE
C          IFLAG=2  NO OP
C          IFLAG=3  DELETE MODE
C          IFLAG=4  NO OP
C          IFLAG=5  END
C          IFLAG=6  NEXT VECTOR
C          IFLAG=7  TEXT MODE
3
      NPTS=NPTS+1
      IF(NPTS.GT.IAR SZ)NPST=NPTS-3
      IF(NPTS.EQ.IAR SZ-2)GO TO 98
      LINE(1,NPTS)=-1
      LINE(2,NPTS)=-1
      IFLAG=1
C SOFTWARE COMES BACK TO HERE EACH TIME A NEW POINT IS ENTERED
C AND DRAWS THE LINE.
4
      IF(NPTS.LT.1)GO TO 12
      I1=NPTS
      IBUF(1)=LINE(1,I1-1)
      IBUF(2)=LINE(2,I1-1)
      IBUF(3)=LINE(1,I1)
      IBUF(4)=LINE(2,I1)
      IF( IBUF(1).EQ.-1.OR.IBUF(3).EQ.-1) GO TO 415
      CALL LN2D(ICHN,IBUF,0,0,ICON,ICHNL,IOURL,IER)
415  CONTINUE

```

AWS TECHNICAL LIBRARY
 FL 4414
 859 BUCHANAN STREET
 SCOTT AFB IL 62225-5118


```

C
C *** READ CURSOR
C
  12 CALL CURSR(ICHN,ICON,ICHNL,X,Y,IER)
C THIS STATEMENT CHECKS TO SEE IF CURSOR WAS IN THE MENU AREA
  IF(X.GE.3700.AND.Y.GE.1671) GO TO 310
C THIS SENDS SOFTWARE TO APPROPRIATE LOCATION DEPENDING ON
C HOW IFLAG IS SET
  GO TO(500,550,600,650,98,3,700),IFLAG
C
C *** SET SELECTION OF MENU
C
  310 IFLAG=0
    DO 450 I=1,7
      J=3071-(8-I)*200
      IF(Y.GT.J) IFLAG=8-I
      IF(IFLAG.LE.0) GO TO 12
  450 CONTINUE
    IF(IFLAG.EQ.6) GO TO 3
    IF(IFLAG.EQ.5) GO TO 98
    IF(IFLAG.EQ.3)GO TO 600
    GO TO 12
C
C THIS SECTION PROCESS VARIOUS MENU SELECTIONS
C
C TEXT SECTION
  700 CALL RTEXT(IBUF,N)
    CALL RTEXT(ICHN,X,Y,IBUF,0,N,0,ICON,ICHNL,IOURL,IER)
    IBUF(13)=X
    IBUF(14)=Y
    CALL WRS(JCHN,IBUF,28,IER)
    LLCNT=LLCNT+14
    IF(LLCNT.GT.250)CALL WRS(JCHN,"12345678",8,IER)
    IF(LLCNT.GT.250)LLCNT=0
    GO TO 12
C LOAD X Y LOCATION AND GO BACK AND DRAW THE LINE
  500 NPTS=NPTS+1
    IF(NPTS.GT.IAR SZ)NPTS=NPTS-3
    IF(NPTS.EQ.IAR SZ-2)GO TO 98
    LINE(1,NPTS) = X
    LINE(2,NPTS) = Y
    GO TO 4
550 GO TO 12
C DELETE MODE
  600 IBUF(1)=LINE(1,NPTS-1)
    IBUF(2)=LINE(2,NPTS-1)
    IBUF(3)=LINE(1,NPTS)
    IBUF(4)=LINE(2,NPTS)
    NPTS=NPTS-1
    IFLAG=1
    CALL LN2D(ICHN,IBUF,1,0,ICON,ICHNL,IOURL,IER)
    GO TO 12
  650 GO TO 12
C
C *** WRITE PRESENT CONTOURS TO DISK)
C
C
C *** CLOSE CHANNEL TO GDM AND DISK FILE
C

```



```

98 CALL KLOSE(ICHN, IER)
   NPTS=NPTS+1
   LINE(1,NPTS)=-1
   LINE(2,NPTS)=-1
   NPTS=NPTS+1
C  WRITE THE END OF FILE
   LINE(1,NPTS)=-2
   LINE(2,NPTS)=-2
   NBYT=(NPTS+1)*4
   CALL WRS (JCHN, LINE, NBYT, IER)
   CALL KLOSE(JCHN, IER)
   CALL CHAIN("INXYRU.SU", IER)
STOP
END

```

R


```

R
TYPE DP1:RTEXT.FR
C*****
C
C      PROGRAM: RTEXT
C
C      PURPOSE: READ TEXT ENTERED FROM ADM AND INTERPRET
C      SPECIAL CONTRO CHARACTERS. RETURN STRING TO MAIN
C      PROGRAM.
C
C      AUTHOR: JIM FORS   DEC 1980
C
C*****
      SUBROUTINE RTEXT(LBUF,NN)
      DIMENSION JBUF(28),LBUF(14)
      NN=0
      IBUF=0K
C  OPEN A CHANNEL TO $ADM . THIS FILE IS LINKED TO APPROPRIATE
C  QTY PORT.
      CALL GCHN(ICHN,IER)
      CALL OPENN(ICHN,"$ADM",0,IER)
      N=0
C  READ IN TEXT FROM ADM ONE BYTE AT A TIME AND INTERPRET SPECIAL
C  CHARACTERS
      50  CONTINUE
          IBYT=1
          CALL RDS(ICHN,IBUF,IBYT,IER)
C  IF CHARACTER IS "$" SET TEXT SIZE TO 3 AND READ NEXT BYTE
          IF( IBUF.EQ.22000K)NN=3
          IF( IBUF.EQ.22000K)GO TO 50
C  IF CHARACTER IS A "^" SET THE SPECIAL CHARACTER FLAG
          IF( IBUF.EQ.57000K)IBUF=11000K
C  IF CHARACTER IS AN END OF TEXT GET OUT
          IF( IBUF.EQ.101400K)GO TO 51
C  IF CHARACTER IS A START OF TEXT READ THE NEXT BYTE
          IF( IBUF.EQ.101000K)GO TO 50
          N=N+1
          JBUF(N)=IBUF
          GO TO 50
      51  CONTINUE
          DO 2 I=1,14
              LBUF(I)=0
          2  CONTINUE
C  PUT ALL THE BYTES IN THE RIGHT HALF OF THE WORD
          DO 1 I=1,N
              JBUF(I)=ISHFT(JBUF(I),-8)
          1  CONTINUE
          CALL PACK(JBUF,N,LBUF)
          CALL KLOSE(ICHN,IER)
          LBUF(12)=NN
          RETURN
          END
R

```



```

R
TYPE DP2:GDCLR.FR
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C CLEAR DISPAY OF GDM
C
C RCHN - RDOS CHANNEL NUMBER
C
C CON - CONSOLE ID
C
C CHNL - CHANNEL NUMBER
C
C OURLY - OVERLAY NUMBER
C
C IER - ERROR RETURN
C
C
C 12/30/79
C DON LAURINE
C SLC/RFC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
SUBROUTINE GDCLR(RCHN,CON,CHNL,OURLY,IER)
INTEGER CON,CHNL,OURLY,CLRW,RCHN
C
NBYT=2
CLRW=10000K
IER=1
C
CLRW=IOR(CLRW,ISHFT(CON,8))
CLRW=IOR(CLRW,ISHFT(CHNL,6))
CLRW=IOR(CLRW,ISHFT(OURLY,4))
C
CALL WRS(RCHN,CLRW,NBYT,IER)
C
RETURN
END
R

```



```

TYPE DP2:LN2D.FR
SUBROUTINE LN2D(RCHN,IBUF,SE,IZ,CON,CHNL,OURLY,IER)
C*****
C
C      RCHN RDOS CHANNEL NUMBER
C      IBUF ARRAY CONTAINING POINTS IN THE LINE
C      SE    SELECTIVE ERASE-IF SET TO 1 A BLANK VECTOR IS
C            DRAWN
C      IZ    ZOOM LEVEL
C      CON   CONSOLE
C      CHNL  CHANNEL
C      OURLY OVERLAY
C
C*****
C      INTEGER SE,RCHN,CON,CHNL,OURLY,ISND(7),IBUF(1)
C
C  FUNCTION WORD FOR DISPLAY MESSAGE
C      IGDM=20000K
C      IGDM= IOR(IGDM,ISHFT(CON,8))
C      IGDM = IOR(IGDM,ISHFT(CHNL,6))
C      IGDM = IOR(IGDM,ISHFT(OURLY,4))
C      IF(SE.GT.0) CALL ISET(IGDM,3)
C
C      ISND(1) = IGDM
C  NUMBER OF WORDS IN RECORD
C      ISND(2) = 5K
C  RELATIVE VECTOR FUNCTION WORD
C      ISND(3) = 160000K
C
C      IF(IZ.EQ.1.OR.IZ.EQ.3) CALL ISET(ISND(3),12)
C  STARTING X LOCATION
C      ISND(3) = IOR(ISND(3),IBUF(1))
C      ISND(4) = 0K
C      IF(IZ.GT.1) CALL ISET(ISND(4),12)
C  STARTING Y LOCATION
C      ISND(4) = IOR(ISND(4),IBUF(2))
C
C  NUMBER OF DELTA X DELTA Y WORDS
C      ISND(5) = 2K
C
C      ITEMP = IBUF(3)-IBUF(1)
C      ISND(6) = 0K
C      IF(ITEMP.LT.0) CALL ISET(ISND(6),12)
C  DELTA X
C      ISND(6) = IOR(ISND(6),IABS(ITEMP))
C
C      ISND(7) = 0K
C      ITEMP = IBUF(4) - IBUF(2)
C      IF(ITEMP.LT.0) CALL ISET(ISND(7),12)
C  DELTA Y
C      ISND(7) = IOR(ISND(7),IABS(ITEMP))
C      NBYTES=14
C
C      CALL WRS(RCHN,ISND,NBYTES,IER)
C      RETURN
C      END
R

```


TYPE DP2:RTEXT.FR

SUBROUTINE RTEXT(RCHN,X,Y,IBUF,IZ,IS,SE,CON,CHNL,OURLY,IER)

C*****

C

C RCHN CHANNEL NUMBER
C X X LOCATION FOR START OF TEXT
C Y Y LOCATION FOR START OF TEXT
C IBUF ARRAY CONTAINING THE TEXT STRING
C IZ ZOOM LEVEL
C IS 0 NORMAL SIZE BLOCK
C 1 NORMAL SIZE REVERSE BLOCK
C 2 NORMAL SIZE
C 3 DOUBLE SIZE
C SE SELECTIVE EASE
C CON CONSOLE
C CHNL CHANNEL NUMBER OF GDM
C OURLY OVERLAY

C

C*****

C

INTEGER RCHN,X,Y,SE,CON,CHNL,OURLY,ISND(155),IBUF(1)

C

C FUNCTION WORD FOR DISPLAY MESSAGE

IGDM=20000K
IGDM=IOR(IGDM,ISHFT(CON,8))
IGDM=IOR(IGDM,ISHFT(CHNL,6))
IGDM=IOR(IGDM,ISHFT(OURLY,4))
IF(SE.GT.0) CALL ISET(IGDM,3)
ISND(1)=IGDM

C

C FUNCTION WORD FOR TEXT

ISND(3)=120000K
IF(IZ.EQ.1.OR.IZ.EQ.3) CALL ISET(ISND(3),12)

C XLOCATION FOR START OF TEXT

ISND(3)=IOR(ISND(3),X)

C

ISND(4)=0K
ISND(4)=IOR(ISND(4),ISHFT(IS,13))
IF(IZ.GT.1) CALL ISET(ISND(4),12)

C Y LOCATION FOR START OF TEXT

ISND(4)=IOR(ISND(4),Y)

C

K=0
DO 10 I=1,150
IF(IBUF(I).EQ.0) GO TO 20

C LOAD TEXT STRING

ISND(I+4)=IBUF(I)
K=K+1

10 CONTINUE

20 CONTINUE

C NUMBER OF WORDS

ISND(2)=K+2
NBYTES=(ISND(2)+2)*2

C

CALL WRS(RCHN,ISND,NBYTES,IER)

C

RETURN
END

R

TYPE DP2:RDCUR.SR

RDCUR.SR

```

;
;   SUBROUTINE( IBUF, IER )
;   DIMENSION IBUF(6)
;   FORTRAN SUBROUTINE TO READ ENTER CURSOR BUTTON
;   ON GDM OF AFOS SYSTEM. TOTALLY INDEPENDANT OF
;   AFOS RUNNING..
; INPUT DATA
;   IBUF(1) - FUNCTION WORD FOR INTERROGATE CURSOR
;             I.E. 120100 LOCATION OF CURSOR ON
;             CHANNEL 1 CONSOLE 0
;   IBUF(2) - CONSOLE ID ...BITS 4-7
;             CHANL NUM ...BITS 1-3
;             RDCS CHANNEL NUM ...BITS 8-15
; RETURNED FROM CALL
;   IBUF(2) - NUMBER OF WORDS ALWAYS 4
;   IBUF(3) - ZOOM ...BITS 1-3
;             X LOCATION OF CURSOR ...BITS 4-15
;   IBUF(4) - Y LOCATION OF CURSOR ...BITS 4-15
;   IBUF(5) - X TRANSLATION ...BITS 4-15
;   IBUF(6) - Y TRANSLATION ...BITS 4-15
; FOR DEFINITION OF ABOVE VARIABLES SEE ...PROGRAMMER'S
; REFERENCE GUIDE FOR GDM ADM FACC.
; PROGRAM WRITTEN BY GERY GROSFELD - HSD
;                   DON LAURINE - SLC/RFC
;
;                   12/19/79
; *****
```

```

.TITL RDCUR
.ENT RDCUR
.EXTD .CPYL, .FRET
.NREL
.DIO .RDC=31400
IBUF=-167 ;SET UP STACK
IER=IBUF+1
FS.=IER-IBUF+1
FS.

RDCUR:
JSR @.CPYL ;COPY ADDRESS TO STACK
LDA 2,IBUF,3 ;ADDRESS OF IBUF
MOVZL 2,0 ;OFFSET TO BYTE POINTER
LDA 2,1,2 ;LOAD CHANNEL AND CONSOLE ID
LDA 1,CNT ;# BYTES PASSED BACK FROM .RDC

STA 3,FPTR ;SAVE STACK POINTER

.SYSM
.RDC 77 ;CALL READ CURSOR ROUTINE
JMP RDERR

LDA 3,FPTR
LDA 2,C1 ;NO ERROR SO PASS IER=1
STA 2,@IER,3

JSR @.FRET ;RETURN
```


RDERR:

```
LDA 3,FPTR
STA 2,@IER,3      ;PASS BACK RDOS ERROR

JSR @.FRET        ;RETURN
```

```
C1:      000001
CNT:     10.
FPTR:    0
        .END
```

R

TYPE DP1:CURSR.FR

SUBROUTINE CURSR(RCHN,CON,CHNL,X,Y,IER)

C

INTEGER RCHN,CON,CHNL,X,Y,IBUF(6)

C

C *** SETUP FUNCTION WORD

C

```
IBUF(1) = 120000K
IBUF(1) = IOR(IBUF(1),ISHFT(CON,8))
IBUF(1) = IOR(IBUF(1),ISHFT(CHNL,6))
```

C

C *** SETUP CHANNEL ID AND CONSOLE NUMBER

C

```
IBUF(2) = 0K
IBUF(2) = IOR(IBUF(2),ISHFT(CHNL,14)) ;CHANNEL NUMBER
IBUF(2) = IOR(IBUF(2),ISHFT(CON,8))   ;CONSOLE NUMBER
IBUF(2) = IOR(IBUF(2),RCHN)           ;RDOS CHANNEL
```

C

C *** READ CURSOR

C

```
CALL RDCUR(IBUF,IER)
IF(IER.NE.1) RETURN
```

C

C *** SEPERATE ZOOM VALUE

C

```
IZOOM = ISHFT(IBUF(3),-12)
IF(IZOOM.EQ.0) GO TO 10
```

C

C *** TAKE OUT ZOOM VALUE TO GET X

C

```
CALL ICLR(IBUF(3),14)
CALL ICLR(IBUF(3),13)
CALL ICLR(IBUF(3),12)
```

C

C *** CALCULATE X,Y IN IMAGE COORDINATES

C

```
X = (IBUF(3) - 2047)/(IZOOM + 1) + 2047 - IBUF(5)
Y = (IBUF(4) - 1535)/(IZOOM + 1) + 1535 - IBUF(6)
RETURN
```

C

```
10 X = IBUF(3)
Y = IBUF(4)
RETURN
```

C

END

R

DIR DP1

R

TYPE INXYRV.FR

C*****

C

C PROGRAM: XYRV.FR

C

C AUTHOR: ALEXANDER E. MACDONALD

C

C MODIFIED FOR INTERACT BY: JIM FORS

C

C PURPOSE: TAKES A IMAGE COORD (XY) FILE AND CONVERTS IT TO
C A RELATIVE VECTOR GRAPHICS FILE. AFTER THIS IS
C RUN IT STILL MUST BE CONVERTED TO UNIVERSAL
C TRANSMISSION FORMAT.

C

C SUBROUTINES

C

C MVEC: CONVERTS A VECTOR STRING TO GRAPHIC FORMAT

C

C COMMH: SETS UP HEADER FOR GRAHIC FORMAT

C

C GPD1: SETS UP GRAPHIC PRODUCT DEFINITION

C

C TEXT: CONVERTS A TEXT STRING TO GRAPHIC FORMAT

C

C NIB: WRITE A BLOCK TO OUTPUT FILE

C

C VARIABLES: MUA - PARAMETER FOR "MOVE VECTOR ARRAY", THE
C SIZE OF THE RELATIVE VECTOR ARRAY IT
C IT - ARRAY TO HOLD VECTOR STRINGS
C LL - INPUT XY ARRAY
C GP - OUTPUT XY ARRAY
C NBL - NUMBER OF THE BLOCK IN INPUT FILE
C NBR - NUMBER OF THE BLOCK IN THE OUTPUT FILE
C NW - "NUMBER OF WORD" IN OUTPUT BUFFER

C

C RLDR/P/N INXYRV AMMVEC COMMH GPD1 TEXT AMNIB UTIL.LB
C FORT.LB AFOSE.LB

C

C*****

PARAMETER MUA=1000,NS=256
INTEGER LL(256), GP(256), IT(MUA),X,Y
CALL OPEN(2,"IN",2,IER)
NBR=-1
NBL=0
IV=0

CALL DFILW("GP",IER)
CALL OPEN(3,"GP",2,IER)

C READ IN FIRST BLOCK
CALL RDBLK(2,NBL,LL,1,IER)

C READ IN THE AFOS GRAPHIC NAME FROM THE FILE SK
IT(1)="NM"
IT(2)="CG"
IT(3)="PH"

CALL GCHN(KCHN,IER)
CALL OPENN(KCHN,"SK",0,IER)
CALL RDS(KCHN,IT(4),4,IER)
CALL KLOSE(KCHN,IER)
IT(6)="00"

C SET UP COMM HEADER FOR GRAHIC
CALL COMMH(GP,IT,NW,NS)

C SET UP GRAPHIC PRODUCT DEFINITION
CALL GPD1(GP,NW,NS,IFLG)

C FILL THE ARRAY WITH -1'S
DO 500 IJ=1,MUA
IT(IJ)=-1


```

500    CONTINUE
C THIS SECTION READS IN ALL THE TEXT INFORMATION AND CONVERTS
C IT TO GRAPHIC FORMAT
C
      IS=2
905    DO 645 IIS=16,254,14
901    IF(LL(IIS-15).EQ.-1)GO TO 900
      IS=IS+7
      NNN=LL(IIS-4)
      X=LL(IIS-3)
      Y=LL(IIS-2)
      CALL TEXT(NNN,GP,X,Y,LL(IIS-15),NW,NS)
645    CONTINUE
      NBL=NBL+1
C READ IN ANOTHER BLOCK
      CALL RDBLK(2,NBL,LL,1,IER)
      IS=2
      GO TO 905
900    CONTINUE
C THIS LOOP CONVERTS X,Y COORDINATES INTO AFOS GRAPHIC
C FORMAT. THIS LOOP PROCESSES ONE BLOCK AT A TIME. A -1 IN THE FILE
C INDICATES THE START OF A NEW VECTOR. A -2 IN THE FILE INDICATES
C THAT THIS IS THE END OF THE FILE
400    CONTINUE
      DO 100 I=IS,128
      RLAT=LL(I*2-1)
      RLON=LL(I*2)
C CHECK FOR THE START OF A NEW VECTOR
      IF(RLAT.NE.-1)GO TO 40
C HOWEVER IF THIS VECTOR HAS ONLY ONE POINT IN IT
C THROW IT AWAY.
      IF(IT(3).EQ.-1)IV=0
      IF(IT(3).EQ.-1)GO TO 100
C PROCESS THE VECTOR
      CALL MVEC(GP,IT,NW,NS,NBR)
C REFILL ARRAY WITH -1'S.
      DO 300 IJ=1,MVA
      IT(IJ)=-1
300    CONTINUE
      IV=0
      GO TO 100
40    CONTINUE
C CHECK IF THIS IS THE END OF THE FILE
      IF(RLAT.NE.-2)GO TO 50
      NNW=NNW+1
C IF THIS IS THE END FILL REMAINDER OF BLOCK WITH -1'S
      DO 105 II=NNW,256
      GP(II)=-1
105    CONTINUE
      NBR=NBR+1
C WRITE IT TO DISK
      CALL WRBLK(3,NBR,GP,1,IER)
C NOW FILL ANOTHER BLOCK COMPLETELY WITH -1'S AND WRITE IT
      DO 910 II=1,256
910    GP(II)=-1
      NBR=NBR+1
      CALL WRBLK(3,NBR,GP,1,IER)
      CALL RESET
      CALL CHAIN("INBUTF.SU",IER)
      STOP LLGP

```



```

50    CONTINUE
      IV=IV+1
C  IV IS THE INDEX VARIABLE FOR THE VECTOR STRING ARRAY IT
      IF(IV.GT.MVA)STOP MVA
      IT(IV)=RLAT
      IV=IV+1
      IF(IV.GT.MVA) STOP MVA
      IT(IV)=RLON
100   CONTINUE
C  READ IN THE NEXT BLOCK AND START OVER
      NBL=NBL+1
C  NOW READ IN NEXT INPUT BLOCK
      CALL RDBLK(2,NBL,LL,1,IER)
      IS=1
      GO TO 400
      STOP
      END

```

R

```

TYPE DP2R
:AMMVEC.FR
C  THIS SUBROUTINE CREATES A RELATIVE VECTOR FORMAT
C  FROM THE ARRAY OF X,Y POINTS PASSED TO IT
C
      SUBROUTINE MVEC(IGA,IVEC,NW,NS,NBR)
C  IVEC IS A SEQUENCE OF X AND Y VALUES ENDING
C  WITH -1 S.
      INTEGER TB(256)
      DIMENSION IGA(NS),IVEC(1)
      INTEGER DX,DY
      MASK=377K
      NW=NW+1
C  IF AT END OF BLOCK WRITE IT TO DISK
      IF(NW.GT.256)CALL NIB(IGA,NW,NBR)
C  KEY WORD FOR RELATIVE VECTORS
      IGA(NW)=141400K
      NW=NW+1
      IF(NW.GT.256)CALL NIB(IGA,NW,NBR)
      IX=IVEC(1)
      IGA(NW)=IX
      NW=NW+1
      IF(NW.GT.256)CALL NIB(IGA,NW,NBR)
      JX=IVEC(2)
      IGA(NW)=JX
      NW=NW+1
      IF(NW.GT.256)CALL NIB(IGA,NW,NBR)
      NBT=NBR
      NTEMP=NW
      NWORD=0
      DO 1 IW=3,999.2
      IF(IVEC(IW).EQ.-1)GO TO 1000
      NX=IVEC(IW)
      NY=IVEC(IW+1)
      DX=NX-IX
      DY=NY-JX
      NW=NW+1

```



```

        IF(NW.GT.256)CALL NIB(IGA,NW,NBR)
        NWORD=NWORD+1
C   IF VECTOR HAS AN X OR Y COMPONENT GREATER THAN 63
C   PIXELS TWO WORDS MUST BE USED
        IF(DX.LT.-63 .OR. DY.LT.-63)GO TO 20
        IF(DX.GT.63 .OR. DY.GT.63) GO TO 20
        IGA(NW)=DY
        IGA(NW)=IAND(IGA(NW),MASK)
        DX=ISHFT(DX,8)
        IGA(NW)=IOR(DX,IGA(NW))
        CALL ICLR(IGA(NW),7)
        CALL ISET(IGA(NW),15)
        IX=NX
        JX=NY
        GO TO 1
20      CONTINUE
        IGA(NW)=DX
        NWORD=NWORD+1
        IF(DX.GE.0)GO TO 70
        CALL ICLR(IGA(NW),15)
        CALL ICLR(IGA(NW),14)
        CALL ICLR(IGA(NW),13)
70      CONTINUE
        NW=NW+1
        IF(NW.GT.256)CALL NIB(IGA,NW,NBR)
        IGA(NW)=DY
        IX=NX
        JX=NY
        IF(DY.GE.0)GO TO 71
        CALL ICLR(IGA(NW),15)
        CALL ICLR(IGA(NW),14)
        CALL ICLR(IGA(NW),13)
71      CONTINUE
        CALL ICLR(IGA(NW),13)
1       CONTINUE
1000    CONTINUE
C   IF VECTOR CROSSES A BLOCK HAVE TO GO BACK TO PREVIOUS
C   BLOCK TO PUT IN NUMBER OF WORDS IN VECTOR
        IF(NBR.EQ.NBT)GO TO 400
        CALL ROBLK(3,NBR,TB,1,IER)
        TB(NTEMP)=NWORD
        CALL WRBLK(3,NBR,TB,1,IER)
        GO TO 401
400     CONTINUE
        IGA(NTEMP)=NWORD
401     CONTINUE
        RETURN
        END
R

```



```

R
TYPE COMMH.FR
      SUBROUTINE COMMH(IGA,IT,NW,NS)
      DIMENSION IGA(NS),IT(1)
      NW=0
      DO 1 I=1,NS
      IGA(I)=0
1      CONTINUE
      DO 2 I=1,6
      NW=NW+1
      IGA(I)=IT(I)
2      CONTINUE
      IGA(7)=-1
      IGA(8)=-1
      NW=8
      RETURN
      END
R

```

```

TYPE GPD1.FR
      SUBROUTINE GPD1(IGA,NW,NS)
      DIMENSION IGA(NS)
      INTEGER RP4,RPPN,GPD,RELU
      IMAX=10000K
      JMAX=6000K
      GPD=140400K
      RELU=141400K
      IGA(NW+1)=2400K
      IGA(NW+2)=GPD
      IGA(NW+4)=IMAX
      IGA(NW+5)=JMAX
      IGA(NW+6)=0
      IGA(NW+7)=0
      NW=NW+8
      RETURN
      END
R

```


TYPE TEXT.FR

```
SUBROUTINE TEXT(NSIZE,IGA,IL,JL,IT,NW,NS)
  DIMENSION IGA(NS),IT(1)
  INTEGER ASCI
  ASCI=142400K
  NN=NW+3
  IGA(NW+1)=ASCI
  IGA(NW+2)=IL
  IF(NSIZE.EQ.0)CONTINUE
  IF(NSIZE.EQ.1)CALL ISET(IGA(NW+2),14)
  IF(NSIZE.EQ.2)CALL ISET(IGA(NW+2),15)
  IF(NSIZE.EQ.3)CALL ISET(IGA(NW+2),14)
  IF(NSIZE.EQ.3)CALL ISET(IGA(NW+2),15)
  IGA(NW+3)=JL
  NW=NW+3
  DO 1 I=1,100
    IF(IT(I).EQ.0)GO TO 50
    IGA(NW+1)=IT(I)
    NW=NW+1
1  CONTINUE
50  CONTINUE
    RETURN
    END
```

R

R

TYPE AMNIB.FR

C

C

SUBROUTINE NIB IS USED BY THE PROGRAM TO WRITE

C

OUT A BLOCK

SUBROUTINE NIB(IG,NW,NBR)

COMMON/CHAN/ LCHN

DIMENSION IG(256)

NBR=NBR+1

CALL WRBLK(3,NBR,IG,1,IER)

NW=1

RETURN

END

R


```

DIR DP1
R
TYPE INBUTF.FR
C*****
C
C   PROGRAM: BUTF.FR
C
C   AUTHOR: ALEXANDER E. MACDONALD
C   MODIFIED FOR INTERACT BY: JIM FORS
C
C   PURPOSE:  CONVERTS ANY GRAPHIC FILE TO UNIVERSAL TRANSMISSION
C             FORMAT ONE BLOCK AT A TIME
C
C   NO USER WRITTEN SUBROUTINES
C
C   VARIABLES:  IW - INPUT WORD BUFFER
C               OF - INTEGER OVERFLOW FILE
C               IBN - INPUT BLOCK NUMBER
C               BB - BYTE INTERMEDIATE BUFFER
C               IX - COUNTER FOR EXCESS BYTES IN AN INPUT BLOCK
C               JMP - USED TO "JUMP" IF NEXT BYTE IS USER LOADED 20K
C
C   RLDR/P/N INBUTF BG.LB UTIL.LB FORT.LB AFOSE.LB
C
C*****
C   INTEGER IW(256),OF(188),BB(700)
C   MASK=377K
C   LZ=177400K
C   JMP=0
C   NEND=0
C   CALL OPEN(2,"GP",2,IER)
C   IBN=0
C   DO 90 I=1,188
90      OF(I)=-1
400      CONTINUE
C   READ IN FIRST BLOCK
C   CALL ROBLK(2,IBN,IW,1,IER)
C   IF BLOCK IS ALL -1'S IT IS THE LAST BLOCK
C   IF(IW(1).EQ.-1 .AND. IW(256).EQ.-1)NEND=1
C   UNPACK THE BLOCK
C   CALL UNPACK(IW,512,BB)
C   WITH BYTE BUFFER LOADED WE PUT "-1S" IN OVERFLOW OF BYTE BUFFER
C   DO 103 I=513,700
C   BB(I)=-1
103      CONTINUE
C   IX KEEPS TRACK OF THE NUMBER OF OVERFLOW BYTES
C   IX=0
C   IF THIS IS THE END BLOCK OF -1'S SKIP THIS LOOP
C   IF(NEND.EQ.1)GO TO 940
C   THIS LOOP INSERTS A 020K 014K FOR 203K
C   CHANGES A REGULAR 020K TO A 020K 020K
C   DO 105 I=1,698
C   IF(BB(I).EQ.377K .AND. BB(I+1).EQ.377K .AND. BB(I+2).EQ.
X 377K .AND. I.GT.512)GO TO 940
C   JMP=JMP-1
C   IF(JMP.GE.0)GO TO 105
C   IF(BB(I).NE.203K)GO TO 30
C   BB(I)=20K
C   IP2=I+2
C   NTT=BB(I+1)
C   DO 106 J=IP2,700
C   NT=BB(J)

```



```

      BB(J)=NTT
      NTT=NT
106    CONTINUE
      BB(I+1)=14K
      IX=IX+1
      GO TO 105
30     CONTINUE
      IF(BB(I).NE.20K)GO TO 105
      JMP=1
      IP2=I+2
      NTT=BB(I+1)
      DO 108 J=IP2,700
      NT=BB(J)
      BB(J)=NTT
      NTT=NT
108    CONTINUE
      BB(I+1)=20K
      IX=IX+1
105    CONTINUE
940    CONTINUE
C NOW BYTE BUFFER IS LOADED TO SOME SIZE 512 OR GREATER WIHT
C -1S FROM 700 BACK TO DATA END
C
C WE NOW HAVE TO PUT THE OVERFLOW DATA AT THE BEGINNING OF THE
C BYTE BUFFER AND MOVE DOWN THE DATA WHICH WAS JUST READ IN
C FROM THE INPUT FILE
C
C FIRST WE MUST COMPUTE HOW MANY BYTES ARE IN THE OVERFLOW FILE
      N=0
      DO 160 I=1,188
      IF(OF(I).EQ.-1)GO TO 161
      N=N+1
160    CONTINUE
161    CONTINUE
C IXT IS THE LENGTH OF THE EXPANDED INPUT BYTE BUFFER
      IXT=IX+512
      IN=IXT+N
      IO=IXT
C WE MOVE EVERY BYTE IN THE BYTE BUFFER DOWN N SPACES
      IF(IN.EQ.IO)GO TO 300
299    CONTINUE
      BB(IN)=BB(IO)
      IN=IN-1
      IO=IO-1
      IF(IO.EQ.0)GO TO 300
      GO TO 299
C LOAD FIRST N POSTIONS OF BYTE BUFFER WITH OVERFLOW BYTES
300    CONTINUE
      IF(N.EQ.0)GO TO 175
      DO 170 I=1,N
      BB(I)=OF(I)
170    CONTINUE
175    CONTINUE
C REPACK THE BLOCK
      CALL PACK(BB,512,IN)
      IF(NEND.EQ.1)GO TO 500
C NOW LOAD OVERFLOW FILE WITH OVERFLOW FROM CURRENT BYTE BUFFER
      DO 200 I=513,700
      J=I-512
      OF(J)=BB(I)

```



```

200      CONTINUE
C NOW WRITE MODIFIED BLOCK BACK TO FILE
      CALL WRBLK(2,IBN,IW,1,IER)
      IBN=IBN+1
      GO TO 400
500      CONTINUE
C WE MUST NOW SEARCH BACKWARD TO FIND END OF DATA TO INSERT
C A 203K BYTE. MAY HAVE TO BACK UP ALL THE WAY INTO THE PREVIOUS
C BLOCK
      I=257
60      CONTINUE
      I=I-1
      IF(I.GE.1)GO TO 61
      IBN=IBN-1
      CALL RDBLK(2,IBN,IW,1,IER)
      GO TO 500
61      IF(IW(I).EQ.177777K)GO TO 60
      IW(I+1)=203K
      JMP=IAND(IW(I),MASK)
      IF(JMP.EQ.377K)IW(I)=IAND(IW(I),LZ)
      CALL WRBLK(2,IBN,IW,1,IER)
      CALL RESET
C NOW STORE IT INTO THE DATA BASE AND ALERT THE ADM
      CALL FSTORE("GP",0,IER)
      CALL FORK("INTERACT ",IER)
      STOP BUTF
      END

```

R

NOAA CENTRAL LIBRARY



3 8398 1013 1879 2

NOAA SCIENTIFIC AND TECHNICAL PUBLICATIONS

The National Oceanic and Atmospheric Administration was established as part of the Department of Commerce on October 3, 1970. The mission responsibilities of NOAA are to assess the socioeconomic impact of natural and technological changes in the environment and to monitor and predict the state of the solid Earth, 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 series; prediction and outlook periodicals; technical manuals, training papers, planning reports, and information series; 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 (D822)
ENVIRONMENTAL DATA AND INFORMATION SERVICE
NATIONAL OCEANIC AND ATMOSPHERIC ADMINISTRATION
U.S. DEPARTMENT OF COMMERCE**

**6009 Executive Boulevard
Rockville, MD 20852**