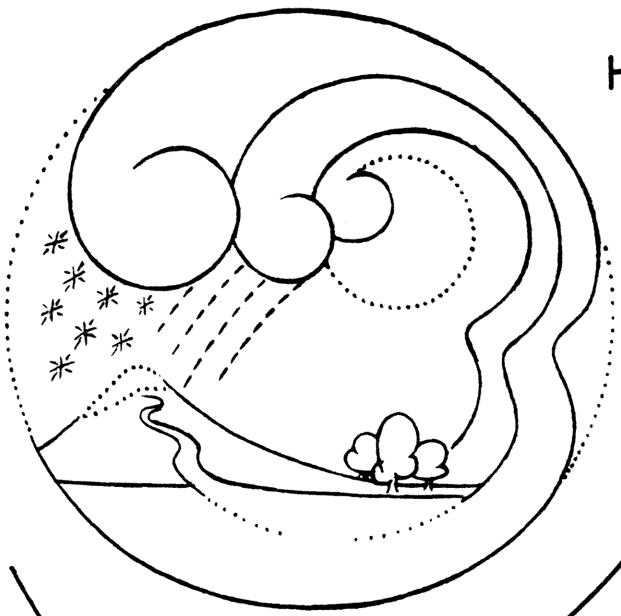
HYDRO TECHNICAL NOTE - 2

NOVEMBER 1983



HYDROLOGIC

RESEARCH

LABORATORY

CREATION OF AFOS GRAPHICS USING THE IBM 360/195

EDWARD J. VANBLARGAN

BILLY G. OLSEN

OFFICE

OF

HYDROLOGY

NATIONAL

WEATHER

SERVICE

NOAA

HYDROLOGIC RESEARCH LABORATORY

OFFICE OF HYDROLOGY

NATIONAL WEATHER SERVICE, NOAA

8060 13TH STREET

SILVER SPRING, MD 20910

CREATION OF AFOS GRAPHICS USING THE IBM 360/195

by

Edward J. VanBlargan
Hydrologic Research Laboratory, Office of Hydrology
National Weather Service, NOAA

and

Billy G. Olsen
Missouri Basin River Forecast Center, Central Region,
National Weather Service, NOAA

November 1983

CONTENTS

FIGURES

TABLE

ACKNOWLEDGEMENTS

The invaluable assistance of Roger Davis, AFOS Operations Division, in understanding the "Universal Transmission Format" and graphics display module is gratefully acknowledged. Many of the subroutines listed herein are modifications of routines originally coded by NMC Automation Division. The assistance of Gloria Dent from NMC in providing and explaining these routines is gratefully acknowledged. The equations used in 'AGPOLR' were extracted from work by Don Laurine, Colorado Basin River Forecast Center and 'UINTCH' and 'URELCH' are the work of George Smith. Also, the support, comments, and programming hints of George Smith, Charles Hoffeditz, and Joseph Ostrowski are sincerely appreciated. The sample display of precipitation reports was provided by courtesy of Joe Miketta, Middle Atlantic River Forecast Center. Finally, many special thanks to Ruth Ripkin for the preparation of this manuscript.

Creation of AFOS Graphics Using the IBM 360/195

by
Edward J. VanBlargan, Hydrologic Research Laboratory, Office of Hydrology
and
Billy G. Olsen, Missouri Basin River Forecast Center, Central Region
National Weather Service, NOAA

ABSTRACT. A software package is presented that can be used to create graphic products displayable on AFOS. The graphics are produced using the NOAA central computer, the IBM 360/195. This package utilizes all of the capabilities of the AFOS-GDM and can create any type of graphic product. It is a utility-type package that requires the user to supply a "main-driver" program. User notes, sample figures, and a sample program are included.

1. INTRODUCTION

This publication describes a software package and procedures developed to create a graphic product on the IBM 360/195 that can be displayed on the AFOS-GDM (Automation of Field Operations and Services-Graphic Display Module). This graphics package will allow many offices in the National Weather Service (NWS) to create and analyze data in a graphics form that was either previously not available graphically or available only in a less automated and efficient manner.

This graphics package may be utilized by any office with access to the NOAA central IBM 360/195 computer. The graphic product generated by this package may be displayed at any office with an AFOS-GDM. The graphic product is in a form that can be stored in the AFOS data base and displayed, or it can simply be displayed locally (i.e., from a floppy disk).

Creation of a graphic for many users, up to the present time, has implied creating the graphic locally in AFOS using an applications program in an available "ground" of the minicomputers. Creating graphics on the main frame IBM 360/195 is not meant as a replacement of that procedure. Rather, it is a procedure intended to give the user greater flexibility and efficiency in creating graphics. Use of applications programs in AFOS may be very constraining for any of the following situations:

a. Data to be displayed often resides in files only on the IBM 360/195.

b. No "space" is available to run an applications program due to the volume of data involved and/or no "grounds" are available to execute the applications program.

c. No time is available to manually execute an applications program, especially during severe weather.

Thus, creating graphics on the IBM 360/195 may be the most efficient and easy manner for certain data and times. The obvious restriction for this graphics package is that access to the IBM 360/195 is necessary.

The graphics package described in this paper is intended to be simple to use but flexible enough to handle a variety of graphic data and to allow the use of the features available in the AFOS-GDM. These features include zooming, reverse block characters, offsetting, and other capabilities. Descriptions of the use of this package hopefully allow ease of understanding for any level of user. Additionally, many "notes" and the source code are given to allow the experienced user the option to make sophisticated use and/or improvements of this graphics package.

2. THE AFOS-GDM

The primary display device for AFOS graphics is the AFOS-GDM (Graphic Display Module). The coordinate system on the GDM is a grid which has dimensions as follows: Y (vertical) extends from 0 to 3071 and X (horizontal) extends from 0 to 4095. The actual number of displayable pixels, however, at the 1:1 zoom level is one-sixteenth of the full coordinates, or 768 x 1024 (Y x X). The full coordinate grid can be examined by using the zoom capability. The user must be aware of the amount of memory required to display a very large graphic. A special memory called "Computer Language Store" (CLS) exists in the AFOS-GDM for graphics operations. Three separate graphic files can be displayed simultaneously as "overlays." The total CLS memory available for these three overlays is about 8K (or 8192 words where each word is 16 bits). A map background from the AFOS data base can also be displayed in addition to the three overlays and independent of the 8K alotted to them. The graphics package described herein will use the following CLS memory (approximately):

Subroutine AGALFA -	Each call uses 3 words plus 1 word for every 2 characters in the string plus 1 word if an "offset" is used.
Subroutine AGLINE -	Each new line (not each call) uses 4 words plus 2 words for each point (or possibly only 1 word if the coordinates are less than 64).

More complete information on the GDM and format and memory used by graphics can be found in the references [Ford Aerospace Corporation, 1976; MacDonald, 1981; Systems Development Office, 1978].

3. OVERVIEW OF PACKAGE USE

This graphics package is designed as a utility set of subroutines, much in the same way as the applications package produced by the Western Region Scientific Services Division described by MacDonald (1981), and commonly used in applications programs on AFOS. This means that the user must write a FORTRAN program which calls the subroutines of this package to create the graphic. The user-written main program will primarily access the data desired and pass it in proper sequence to the utility, graphic subroutines.

The utility subroutines can be broken down into two categories:

a. The actual graphic subroutines which must always be called.

b. Supporting subroutines which may or may not be called.

A list of the actual graphic subroutines with descriptions follows:

- AGALFA - Plots alphanumeric characters or words.
- AGLINE - Draws lines between specified points.
- AGUTF - Converts the data to "Universal Transmission Format" so that it is displayable and writes out the graphic data.

Typically, subroutines AGALFA and AGLINE will be called many times; a call to AGALFA must be made for each character string to be displayed, and AGLINE must be called for each point in the line being drawn. Subroutine AGUTF is called only once for each graphic product. Several graphic products may be created in the same program. Each product would have its own calls to AGALFA and AGLINE and then a call to AGUTF before the next product is started.

Various supporting subroutines are provided to produce transformations or computations often desired for the data. Calls to these support subroutines are necessary only if one desires the function they provide. A list of the support subroutines with descriptions follows:

- UINTCH - Converts integer values to their character representations.
- URELCH - Converts a floating point variable to a character representation.
- AGLENG - Calculates the total number of characters in the text string representation of a real number, given the desired number of decimal places.
- AGPOLR - Uses a polar stereographic projection to provide the user with conversion factors to convert any latitude and longitude to coordinates on the AFOS-GDM. NOTE: This subroutine provides only the conversion factors; the user must do the actual transformation of the latitude and longitude.
- AGCNTR - Contours and labels a given square grid point field. Given a gridded data array, grid location and coutour interval, this subroutine calculates contour vector locations, screen coordinates and pen up or down, then calls AGLINE to actually draw lines. Coordinates of contour intervals are determined and labels are then written using AGALFA.
- AGRGRD - Calculates a square grid of estimated data values, given a field of random data, the associated GDM screen coordinates and the desired square grid characteristics.

More complete descriptions of the uses and functions of all the subroutines are given in the next section.

In order to execute a graphic program, the user must supply a "main-driver" program and link in load modules of the utility subroutines. These are located in the following members on the IBM 360/195:

```
    NWS.AFOS.MAPBACK.LOAD(EVAFOSG) -- contains the graphic and general
                                        support routines
    NWS.AFOS.MAPBACK.LOAD(EVAGCTR) -- contains the countouring routines
```

The core requirements are 11K for member EVAFOSG only and another 64K to add member EVAGCTR. The JCL needed to utilize this package is:

```
    //   JOB    TIME=1,REGION=?
    //*MAIN    CARDS=(700,C)
    //   EXEC    XFORXCLG,IUPVER='R12.'
    //FORT.SYSIN    DD    *
              .
              .
          (USER-WRITTEN MAIN PROGRAM)
              .
              .
    /*
    //LKED.MYLIB    DD    DSN=NWS.AFOS.MAPBACK.LOAD,DISP=SHR
    //LKED.SYSIN    DD    *
     ENTRY MAIN
     INCLUDE   MYLIB(EVAFOSG)
     INCLUDE   MYLIB(EVAGCTR) --optional:  needed only if contouring is used
    /*
    //
```

Various samples of graphics created with this package are shown in Appendix A (figures A-1 through A-7). The programs used to create each sample are not given. Appendix C gives an example of a user-written, "main" program to create graphics. The JCL needed to execute this program is also given, and the graphic produced is shown in figure C-2. Appendix D gives information on locating and obtaining the source code for all of the graphics utility subroutines.

Any information used in this graphics package that is compatible with a certain AFOS load (e.g., the headers, which are for AFOS 3.15) will be updated as necessary as new AFOS loads become operational.

4. SUBROUTINE DESCRIPTIONS

This section gives complete descriptions of each subroutine in the graphics utility package. The subroutines are listed in three categories, actual graphic subroutines and two types of support subroutines, and are alphabetical within each category.

SUBROUTINE AGALFA

A. Description

Subroutine AGALFA allows the display of characters and special fonts (such as the hurricane symbol). The character string is passed in via an array (NCTEXT) along with its display location (IX,IY) on the AFOS-GDM. Subroutine AGALFA must be called separately for each character string to be displayed and the calls may be in any order. This subroutine allows various zoom levels, various sizes and block modes, and various "offset" distances to be used with each character string. Since only character values are used in the display, integer or real numbers to be displayed must be previously converted to character representations. Subroutines UINTCH and URELCH are available to accomplish this conversion.

B. Calling Sequence

```
CALL AGALFA (IX,IY,NCTEXT,NCHAR,ISIZE,IZT,IDELX,IDELY,WORK,WORK,LENGW)
```

C. Argument List

Variables	Input/ Output	Type	Dimension	Description
IX,IY	Input	I	1	The X and Y display coordinates of the lower left corner of the first character in the character string. Make sure they are within the display space, i.e., $0 \leq IX \leq 4095$ $0 \leq IY \leq 3071$
NCTEXT	Input	I	Variable	Array containing the character string to be displayed. (See Notes section below.)
NCHAR	Input	I	1	Number of characters in NCTEXT. Maximum is 113. If NCHAR is negative, then characters are not converted to ASCII. (See Notes.)
ISIZE	Input	I	1	Indicator of character size and type = 0, normal size--unblocked = 1, normal size--blocked = 2, normal size--reverse video blocked = 3, large (double) size -- unblocked. (See Notes.)

Variables	Input/ Output	Type	Dimension	Description
IZT	Input	I	1	Indicator of zoom threshold = 0, display at all zooms = 1, display at 4X or higher = 2, display at 9X or higher = 3, display at 16X or higher

IDELX & IDELY	Input	I	1	"Offset" values from the IX coordinates and IY coordinates, respectively.
				IDELX = 0, no offset from IX.
				IDELY = 0, no offset from IY.
				(See Notes.)

WORK	Input	R	LENGW	Work space array. (See Notes.)
WORK	Input	R	LENGW	Work space array. (See Notes.)
LENGW	Input	I	1	Maximum size of WORK array in full words.

D. Notes

1. The following notes more fully explain the array WORK. This array is passed into all three graphic routines -- AGALFA, AGLINE, and AGUTF. In any one call, the same array is passed in twice to AGALFA or AGLINE and once to AGUTF. WORK is filled with the graphic information and, therefore, should not be used anywhere else in the main program. The user must dimension WORK in the main program to any size desired. The maximum dimension of WORK should not exceed 4000, however, since the maximum size for a graphic product is roughly 16000 bytes.

2. In order to more fully explain variables NCTEXT and NCHAR above, the following points of note are listed:

   (a) Characters are assumed to be coded in EBCDIC unless NCHAR is negative. Internally, the subroutine will convert the EBCDIC code into ASCII. NCHAR usually will be positive since EBCDIC is the standard character code used in IBM.

   (b) Each array element in NCTEXT will contain 4 characters, unless NCTEXT is declared as a LOGICAL*1 variable. For example, the message "AFOS TEST" would be passed in as

   ```
                 NCTEXT(1)='AFOS'
                 NCTEXT(2)=' TES'
                 NCTEXT(3)='T'
                 NCHAR=9
   ```

   If UINTCH or URELCH is called, it is often helpful to declare NCTEXT as LOGICAL*1.

(c) The discussion of subroutine AGALFA up to this point has mainly concerned displaying "standard characters" including upper case letters, standard keyboard symbols, and numbers (in a character representation). Three additional "unique" categories are available for display:

(1) Spacing commands (Back, Forward, Down, Up, 1.5 New Line, and New Line)

(2) Special characters

(3) Lower case letters and certain "standard" symbols not available on some keyboards

These unique categories (and all displayable categories for that matter) have integer values associated with them as shown in table 1. These "unique" characters can be used by placing their integer values in NCTEXT and abiding by the following rules.

a. Always use the integer values indicated in table 1 and do not convert to character representations. For efficiency, declare NCTEXT as LOGICAL*1.

b. For spacing commands, simply place the integer value wherever it is desired in the NCTEXT string, i.e.,
NCTEXT(1) = 'A'
NCTEXT(2) = 13
NCTEXT(3) = 'B'
will display 'B' on a new line and immediately below 'A'.

c. The following example shows how to display "special" characters using 8 characters and the first location in NCTEXT.
NCTEXT(1) = 18 indicates special characters follow.
NCTEXT(2 thru 9) = integer values corresponding to each special character desired.
NCTEXT(10) = 17 indicates end of special characters.
NCHAR = 10.
Note: Special characters and standard characters can be mixed in the same string.

d. To use integer values for any standard character, simply pass the integer as a negative. For example,
NCTEXT(1) = 'A'
NCTEXT(2) = -97
NCHAR = 2
will produce Aa.

3. The following notes explain more fully the character size variable (ISIZE) and offset variables (IDELX,IDELY). As background information, one should recall that coordinates on the AFOS GDM are Y = 0 to 3071 and X = 0 to 4095. The number of displayable pixels, however, is one-quarter of that or 768 for Y and 1024 for X. Each normal size character uses 9 x 7 pixels (Y x X). To provide reasonable legibility, however, a 2-pixel spacing is used between characters, which essentially means each character is given an 11 x 9 pixel area (Y x X). From this, one can see that the maximum number of readable characters that will fit on the GDM is 69 down and 113 across . Also, special characters have only one size, 11 x 11.

"Blocking" refers to the clearing of a rectangular area beneath the character. "Reverse video" reverses the black and white on the character display block. When the block mode is used within a graphic product, a character can block out strings before it but not after it. The "Offset" values allow one to place a character string a certain distance from the IX,IY coordinate, independent of the zoom level. The "Offset" values refer to the number of pixels (that is, the 768 x 1024 grid) and can be (+) or (-). The maximum offset allowed for either IDELX or IDELY is ±63.

SUBROUTINE AGLINE

A. Description

Subroutine AGLINE will draw lines between successive screen coordinates. The lines may be actual ("pen down") or blank ("pen up"). A call to AGLINE must be made for each IX,IY coordinate and the calls must be in the same sequence as the coordinates appear to make up the line(s). This subroutine allows the use of different zoom levels for each line and the zoom feature may be disabled.

B. Calling Sequence

```
CALL AGLINE (IX,IY,IPEN,IZT,IZD,WORK,WORK,LENGW)
```

C. Argument List

Variables	Input/ Output	Type	Dimension	Description
IX,IY	Input	I	1	The X and Y coordinates for this point in the line.
IPEN	Input	I	1	Pen up or down indicator = 1, pen down, line segment is drawn. = 2, pen up; blank segment is drawn giving impression of starting a new line. = -2, start a new line. Similar to IPEN=2 except it is actually a new line (not just a blank segment) implying new zoom values. (See _Notes_ section.)
IZT	Input	I	1	Zoom threshold = 0, display at all zooms. = 1, display at 4X or higher. = 2, display at 9X or higher. = 3, display at 16X or higher. Used only when starting a new line. (See _Notes_ section.)

Variables	Input/ Output	Type	Dimension	Description
IZD	Input	I	1	Zoom disable = 0, zoom normally = 1, zoom is disabled and the line is anchored at the initial IX,IY of the line. Used only when starting a new line. (See Notes section.)
WORK	Input	R	LENGW	Work space array. (See Notes section in AGALFA description.)
WORK	Input	R	LENGW	Work space array (See Notes section in AGALFA description.)
LENGW	Input	I	1	Maximum size of WORK array in full words.

D. Notes

To more completely understand the use of AGLINE, one must recognize the difference between a new line and a simple "pen up" segment. A line consists of a series of segments drawn between consecutive X,Y points. The entire line is initially associated with a single zoom value. Within a line, there can be blank (pen up) segments. Blank segments are still part of the line although they graphically portray a new line. The important point, again, is that all segments in a single line have the same zoom value regardless of whether they are actually drawn or blank. Also, a new line is started (a) when IPEN = -2, (b) the first time AGLINE is called, or (c) when a call to AGALFA is made between calls to AGLINE.

SUBROUTINE AGUTF

A. Description

Subroutine AGUTF will write out the graphic data created by AGALFA and AGLINE to a punch stream. It must always be called, once only for each graphic product, after the last call to either AGALFA or AGLINE. More than one graphic product can be created in a program; each graphic created must have all its calls to AGALFA and AGLINE together, followed by a single call to AGUTF. This subroutine also checks the data for restricted characters, converts it where necessary to be compatible with the "Universal Transmission Format," and adds proper communications heading and ending. An option is provided for utilizing the synchronous routing to specific sites available in AFOS 3.15.

B. Calling Sequence

```
CALL AGUTF (NAFOS,NHTYPE,NROUTE,WORK)
```

C. Argument List

Variables	Input/ Output	Type	Dimension	Description
NAFOS	Input	I	9A1	Array containing a nine-character AFOS product identifier (referred to as the CCCNNNXXX). If product will only be displayed locally (not stored in data base), this can be any name desired.
NHTYPE	Input	I	1	Type of header to be attached to product: = 0, header is not written; user will supply one. = 1, header that is written can be directly processed and stored by AFOS with routing to local sites only. = -1, same as above (= 1) except specific routing site is specified in variable NROUTE. = 2, header that is written allows product to be stored as an RDOS file and displayed using DSP: command.

Variables	Input/ Output	Type	Dimension	Description
NROUTE	Input	I	3A1	Three character station or region address to which product will be routed. Valid only if NHTYPE = -1. Examples are 'EAS', 'FTW', and 'ALL' for Eastern Region, Ft. Worth, and all sites, respectively.
WORK	Input	R	Variable	Work space array; was also passed to AGALFA and/or AGLINE. (See Notes section in AGALFA description.)

D. Notes

Cautionary Note: AFOS Data Review Group approval is required prior to transmitting any graphic product on AFOS.

This section more fully describes the process of transferring the graphic product from the 360/195 to the AFOS GDM via Remote Job Entry (RJE). There are two methods to accomplish this; thus, there are two types of "headers" (NHTYPE) that subroutine AGUTF uses. These headers are attached to the graphic data to form the displayable graphic product. One must also keep in mind that when a punch stream from the 360/195 is being written, a punch identifier card is placed before and after the punch stream. Any method of displaying must be able to "strip" these punch identifiers off the graphic product. Also, one must consider the HASP Workstation Emulator (HAMLET) punch output stream parameters for proper output form when transferring via RJE. RJE must not translate or pad the punch stream. Further notes on the two methods follow.

Method 1

The graphic products from the 360/195 are brought back to the AFOS S/230 via a HAMLET (RJE) punch stream. RJE automatically sends this punch stream to an Asynchronous Line Multiplexor (ALM) port. This port in turn is connected to a second ALM port. Since the second ALM port has been initialized by the AFOS asynchronous directory, it is continuously monitored by AFOS. Because the punch stream is in the form expected by the asynchronous scheduler [i.e., header is 16 bytes consisting of:

'ZCZC CCCNNNXXX (carriage return) (line feed)'

and tailer is 'NNNN'], it is stored locally as an AFOS product (or optionally sent on the AFOS loop). This procedure is accomplished without halting RJE and utilizes standard AFOS and HAMLET features. A complete description of this technique is given by Olsen and Lillie (1983).

The HAMLET punch output stream assignment commands include three parameters that must be considered if output is to be in the proper form. The parameters are /T, /P, and /A. These values were set at HAMLET communications generation time (HCGEN) to have the following meaning in the current version of RJE:

> Inclusion of the /T parameter means no translation of EBCDIC to ASCII. If /P is included, the punch records will not be padded with blanks to 80 column records. If /A is included, the output stream will allow appending to disk files. This /A parameter must be included to allow static assignment of the punch stream to QTY:15.
>
> If the user wishes to statically assign the punch stream N1 to QTY:15 with no padding and no EBCDIC to ASCII translation, the command line in file 'STARTUP.CM' would read:

```
        A N1 QTY:15  /A/P/T
```

Refer to the Data General HASP II Workstation Emulator (HAMLET) Users Manual 093.000116-01 for clarification.

<u>Method 2</u>
The graphic product is transferred from RJE into an RDOS file. Then an applications program must be run to strip off the punch stream identifier cards. The graphic can be displayed using the DSP: command and/or can be stored in AFOS using the STORE command. The form of the graphic header is 18 bytes consisting of:

```
        'CCCNNNXXXaaamdhmtp'
```

When assigning the HAMLET punch output stream to the RDOS file name, one must consider the parameters (/T, /P) discussed in Method 1. An applications program to strip off the punch stream identifier is available under the name 'EVSTRIP'. This program may be obtained from the Office of Hydrology (FTS-427-7640) or from the source code in the library 'W.NWS.W23.EJV.SOURCE (EVSTRIP)'. This stripping program simulates AFOS by searching for a beginning 'ZCZC' and ending 'NNNN' which the user must provide in the main program. An example of doing this would be:

```
        LOGICAL*1 ZCZC(4),NNNN(4)
        DATA ZCZC/90,67,90,67/,NNNN/4*78/
              .
              .
              .
        WRITE(7,100) ZCZC
        CALL AGUTF (NAFOS,2,JUNK,WORK)
        WRITE(7,100) NNNN
100     FORMAT(4A1)
```

Note that the ZCZC and NNNN must be ASCII characters as reflected by the integer values in the DATA statement.

SUBROUTINE AGLENG

A. Description

Subroutine AGLENG calculates the total number of characters in the text string representation of a given real number. The subroutine must know the number of desired decimal places. The subroutine has a limit of 10 for the total number of characters in the representation of the real number. For further flexibility, a lower limit may be set in the calling program. In both cases, exceedence of the limit is a fatal error.

B. Calling Sequence

```
CALL AGLENG (VAL,NCHAR,NUMDEC,LIMIT)
```

C. Argument List

Variables	Input/ Output	Type	Dimension	Description
VAL	Input	R	1	The real number which is to be changed to its text string representation.
NCHAR	Output	I	1	Total number of characters in the text string representation.
NUMDEC	Input	I	1	Number of desired decimal places.
LIMIT	Input	I	1	Limit for number of characters in text string representation.

SUBROUTINE AGPOLR

A. Description

Subroutine AGPOLR determines the conversion factors necessary for converting latitude and longitude values to IX and IY coordinate values on the AFOS GDM. This is accomplished using equations for a polar stereographic projection. AGPOLR is usually called only once in a program and then each latitude and longitude is converted to the AFOS IX,IY coordinates by the user with the following four equations:

```
XM = 4680. * TAN (0.785369 - 0.5 * XLAT * DEGRAD)
YM = XLON * DEGRAD - ADJ * DEGRAD
IX = XM * COS (YM) * XA - XB
IY = (-XM * SIN (YM)) * YA - YB
```

where:	ADJ,XA,XB,YA, and YB are output from AGPOLR
	XLAT = latitude of point in degrees decimal (e.g., 37.80)
	XLON = longitude of point in degrees decimal
	DEGRAD = 0.017453; conversion factor for degrees to radians

Subroutine AGPOLR is also used by all of the local map background programs developed by the Office of Hydrology and described by Hoffeditz (1982). Therefore, use of this subroutine will make the graphic product compatible with local map backgrounds and all other commonly used backgrounds which use a polar stereographic projection.

B. Calling Sequence

```
CALL AGPOLR (TOP,BOT,XLEFT,RIGHT,ADJ,XA,YA,XB,YB)
```

C. Argument List

Variables	Input/ Output	Type	Dimension	Description
TOP	Input	R	1	Latitude of top of "window" area in degrees decimal.
BOT	Input	R	1	Latitude of bottom of "window" in degrees decimal.
XLEFT	Input	R	1	Longitude of left side of "window" in degrees decimal.
RIGHT	Input	R	1	Longitude of right side of "window" in degrees decimal.
ADJ	Output	R	1	Conversion factor for above equations.

Variables	Input/ Output	Type	Dimension	Description
XA	Output	R	1	Conversion factor
YA	Output	R	1	Conversion factor
XB	Output	R	1	Conversion factor
YB	Output	R	1	Conversion factor

SUBROUTINE UINTCH

A. Description

Subroutine UINTCH converts integer values to their character representations. Typical application is to prepare numbers for display using subroutine AGALFA.

B. Calling Sequence

```
CALL UINTCH (INT, MAXCHR, NCTEXT, NCHAR, IST)
```

C. Argument List

Variables	Input/ Output	Type	Dimension	Description
INT	Input	R	1	Integer value to be converted.
MAXCHR	Input	I	1	Maximum number of characters that NCTEXT can hold.
NCTEXT	Output	A	MAXCHR	Array containing the character representation of INT; right justified. (See Notes section.)
NCHAR	Output	I	1	The total number of characters filled in NCTEXT.
IST	Output	I	1	Status code. =0, All OK. =1, Array NCTEXT not large enough to hold value INT. The rightmost MAXCHAR characters of value INT are packed into NCTEXT.

D. Notes

Since the characters in NCTEXT are right justified, there will be leading blanks inserted if MAXCHR is greater than NCHAR. One can eliminate these blanks either by calling AGLENG or by passing the proper starting location in NCTEXT to AGALFA. For example, instead of passing the entire array into AGALFA, pass in NCTEXT(LOC) where LOC = MAXCHR-NCHAR+1. This example assumes NCTEXT is declared as LOGICAL*1.

SUBROUTINE URELCH

A Description

Subroutine URELCH converts real values to their character representations. The real values that can be input are those for which the absolute value is less than 2**31. The maximum number of decimal places that can be requested is 9.

B. Calling Sequence

```
CALL URELCH (VALUE, MAXCHR, NCTEXT, NUMDEC, NCHAR, IST)
```

C. Argument List

Variables	Input/ Output	Type	Dimension	Description
VALUE	Input	R	1	Real value to be converted.
MAXCHR	Input	I	1	Maximum number of characters that NCTEXT can hold.
NCTEXT	Output	A	MAXCHR	Array containing the character representation of VALUE. (See Notes section in UINTCH.)
NUMDEC	Input	I	1	Number of decimal places (maximum 9) to be filled in NCTEXT. =0, the decimal point is the rightmost character in NCTEXT. = -1, no decimal point is stored in NCTEXT.
NCHAR	Output	I	1	The total number of characters filled in NCTEXT.
IST	Output	I	1	Status code. =0, All OK. =1, MAXCHR not large enough to hold VALUE. =2, NUMDEC greater than 9; NUMDEC reset to 9 and NCTEXT filled. =3, Value out of valid range, i.e., absolute value greater than or equal to 2**31.

SUBROUTINE AGCNTR

A. Description

Subroutine AGCNTR contours and labels a given square grid point field. Given a gridded data array, grid location and contour interval, AGCNTR calculates contour vector locations, screen coordinates and pen up or down, then calls AGLINE to actually draw lines. Coordinates of contour intervals are determined and labels are then written using AGALFA. Subroutine AGRGRD is available to transform random data to a square grid point field.

B. Calling Sequence

CALL AGCNTR (GRID,IZERO,IZT,WORK,WORK,LENGW)

C Argument List

Variables	Input/ Output	Type	Dimension	Description
GRID	Input	R	Variable	Array containing the square grid point field to be analyzed.
IZERO	Input	I	1	Indicator for plotting zero contour. = 0, zero contour not plotted = 1, zero contour plotted
IZT	Input	I	1	Zoom level
WORK	Input	R	LENGW	Work space array. (See Notes section in AGALFA description.)
WORK	Input	R	LENGW	Work space array. (See Notes section in AGALFA description.)
LENGW	Input	I	1	Maximum size of WORK array in full words.

D. Notes

1. This subroutine calculates the coordinates on a grid square (four grid points) where the desired contour values intersect the grid square edges. It begins with the lower left grid square and proceeds left to right, then up to the next row. The subroutine will handle the possible one through four point grid square intersections for any one given contour interval. No contour will be drawn for a one point intersection (i.e.,one grid point equal to the contour interval). Selection of possible alternative vectors for three and four point intersections are resolved based on the estimated magnitude of the contour value at the center of the square.

2. In general, the smoothness of the actual contour lines can be improved by increasing the dimensions of the input GRID array. "Bull's eye" type data depictions from the AGCNTR subroutine are more difficult to control, since they are a function not only of mesh size, but also of contour interval, magnitude of individual values of input data, and reporting network density.

3. Contour parameters are controlled through the labeled common AGCNPR, which is described on page 23. This common should be filled in the main program.

4. Variable IZT, the zoom level, is the same as IZT described in the AGALFA description.

SUBROUTINE AGRGRD

A. Description

Subroutine AGRGRD calculates a square grid (array GRID) of estimated data values given a field of random data and associated GDM screen coordinates and the desired square grid parameters (labeled common AGCNPR).

B. Calling Sequence

```
CALL AGRGRD (GRID,IXD,IYD,DATA,NUMOBS)
```

C. Argument List

Variables	Input/ Output	Type	Dimension	Description
GRID	Output	R	Variable	Array containing the estimated square grid point field to be input to subroutine AGCNTR.
IXD,IYD	Input	I	Variable	Arrays containing the pixel (X,Y) coordinates on the AFOS GDM corresponding to each point in DATA.
DATA	Input	R	Variable	Array containing the random (i.e., observed) data.
NUMOBS	Input	I	1	Number of reports used for estimation of the square grid field, i.e., the dimension of DATA.

D. Notes

1. See common block AGCNPR description for desired square grid parameter list.

2. In estimating desired grid point values, this subroutine uses a simple technique which locates the closest observed value in each quadrant, and estimates the grid point value based on $1/d^2$ weighting (where d is the distance in pixels from location of observed value to grid point). If an observed value happens to have the same coordinates as the desired grid point value, the subroutine uses this observed value.

COMMON BLOCK: AGCNPR

A. Description

Common block AGCNPR contains contour control parameters for subroutines AGCNTR and AGRGRD. It should be filled in the main program if contouring is used. The size of the common block is 9 words.

B. Listing

```
COMMON/AGCNPR/ NX,NY,ISTEP,UI,DU,XMIN,YMIN,XRANGE,YRANGE
```

C. Contents

Variables	Type	Dim.	Word Pos.	Description
NX,NY	I	1	1,2	Number of X and Y coordinates of the GRID array (square grid point field).
ISTEP	I	1	3	Number of screen coordinates between grid points.
UI	R	1	4	Value of starting contour (normally used to nearest tenth of a unit).
DU	R	1	5	Contour interval (normally used to nearest tenth of a unit).
XMIN,YMIN	R	1	6,7	X and Y screen coordinates of lower left grid point.
XRANGE, YRANGE	R	1	8,9	X and Y screen coordinate distance of the grid.

D. Notes

Basically one coordinate system (GRID) is being converted to another (the AFOS GDM) and the relationship between them is the variable ISTEP. The screen coordinate reference variables (XMIN, YMIN, XRANGE, and YRANGE) are set by the user to allow flexibility in contouring over the entire AFOS GDM or just a part of it. Thus, there are no set numbers for these variables. For example, to use roughly only the lower left quadrant of the screen, one may have NX=40, NY=30, XMIN=YMIN=0, XRANGE=2000, YRANGE=1500, and ISTEP=50.

REFERENCES

Data General Corporation, 1979. HASP II Workstation Emulator (HAMLET) Users Manual 093.000116-01.

Ford Aerospace Corporation, 1976. Programmers Reference Guide: Graphic Display Module Alphanumeric Display Module. Document WDL-TR7676A, Palo Alto, California, pp. 3-1 through 3-13 and 6-1 through 6-9.

Hoffeditz, Charles N., 1982. AFOS Map Background User Guide, Office of Hydrology, Silver Spring, Maryland, pp. 1-18.

MacDonald, Alexander E., 1981. AFOS Graphics Creation from FORTRAN. NOAA Western Regional Computer Programs and Problems, NWS-WRCP No. 18, 22 pp.

Olsen, Billy G. and Lillie, Dale G., 1983. Automatic Distribution of AFOS Products Created at the NOAA Central Computer Facility via Hamlet (RJE) Punch Stream. NOAA Technical Memorandum NWS CR-70, NWS Central Region, 20 pp.

Systems Development Office, 1978. Universal Transmission Format Version 2.1. National Weather Service Headquarters, Silver Spring, Maryland, 18 pp.

APPENDIX A. SAMPLE FIGURES

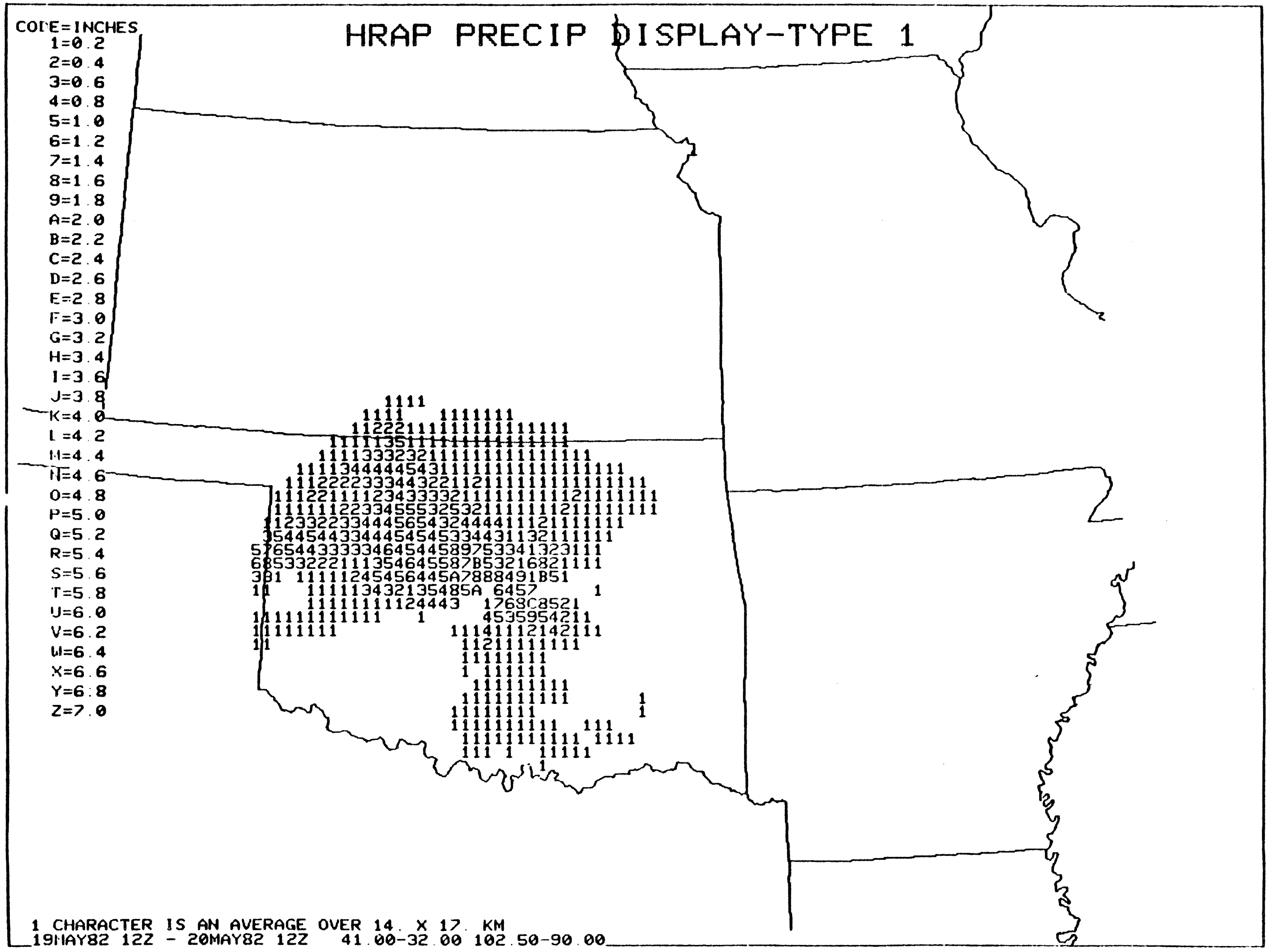


Figure A-1. Sample display from HRAP (Hydrological Rainfall Analysis Program).

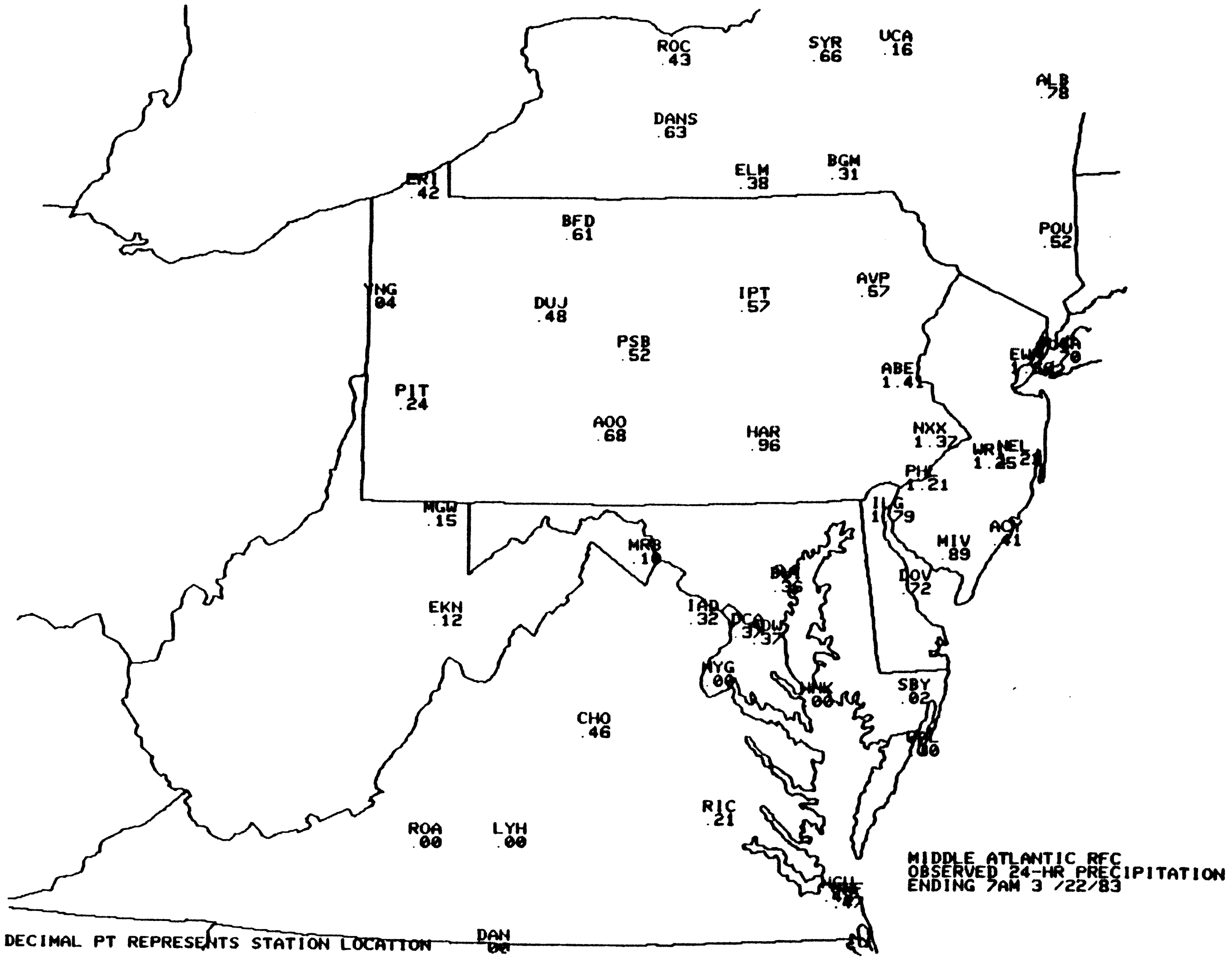


Figure A-2. Sample display of precipitation reports (in inches). Courtesy of Middle Atlantic RFC.

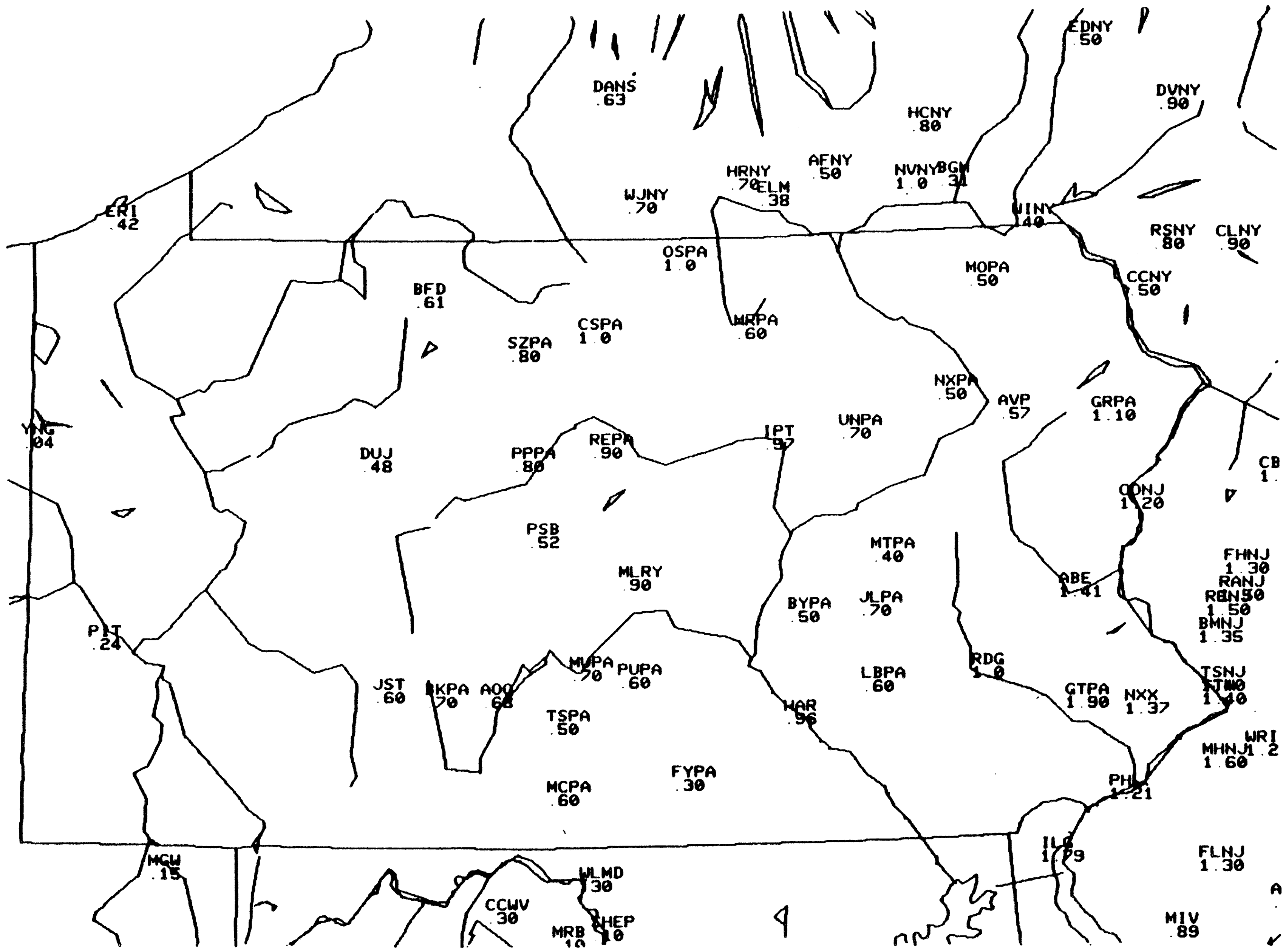


Figure A-3. Sample display of precipitation at 4:1 zoom (from figure A-2) showing additional precipitation and river data. Courtesy of Middle Atlantic RFC.

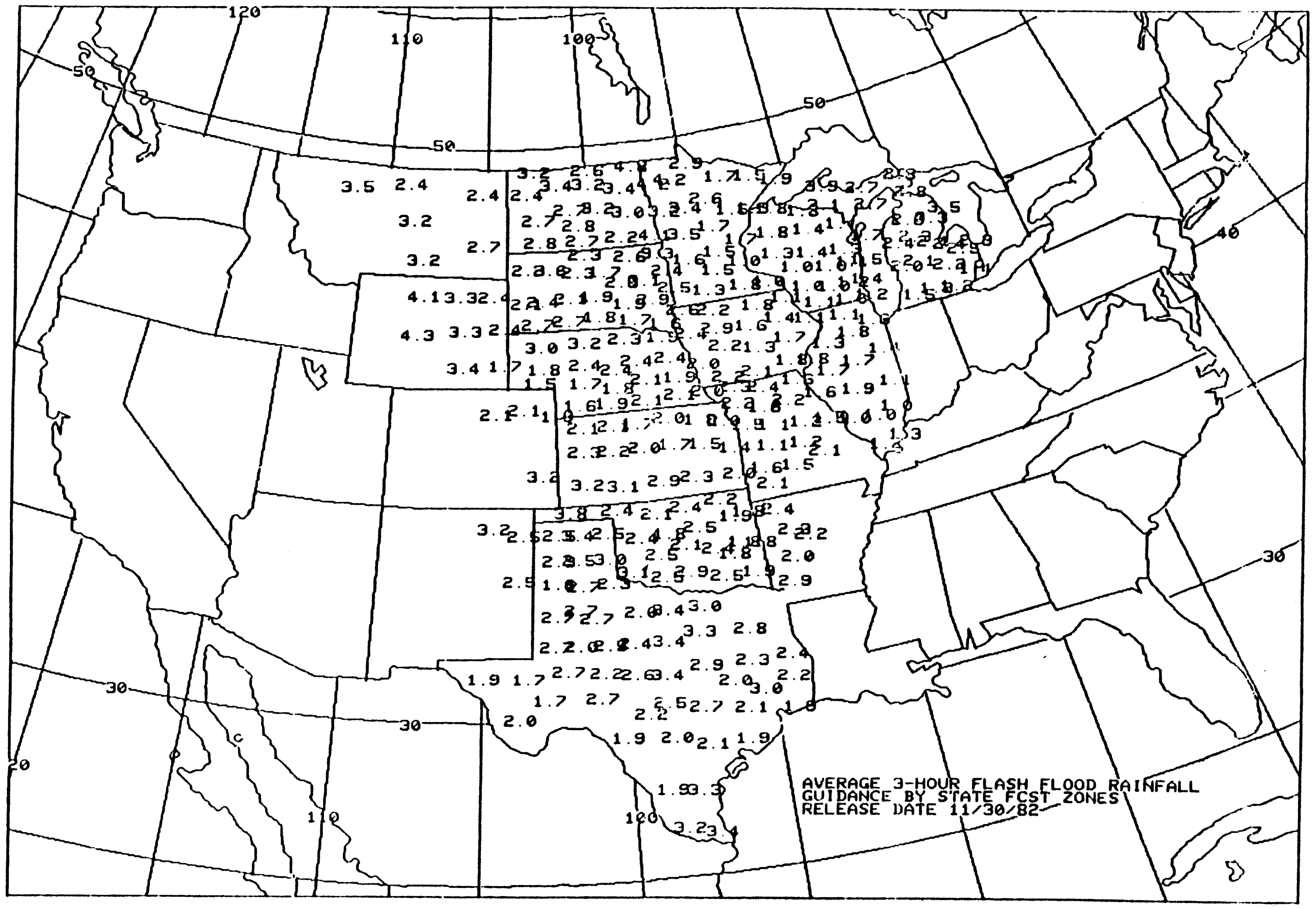

AVERAGE 3-HOUR FLASH FLOOD RAINFALL
GUIDANCE BY STATE FCST ZONES
RELEASE DATE 11/30/82
120
110
100
50
40
30

Figure A-4. Sample display of Flash Flood Guidance from four RFC areas at 1:1 zoom.

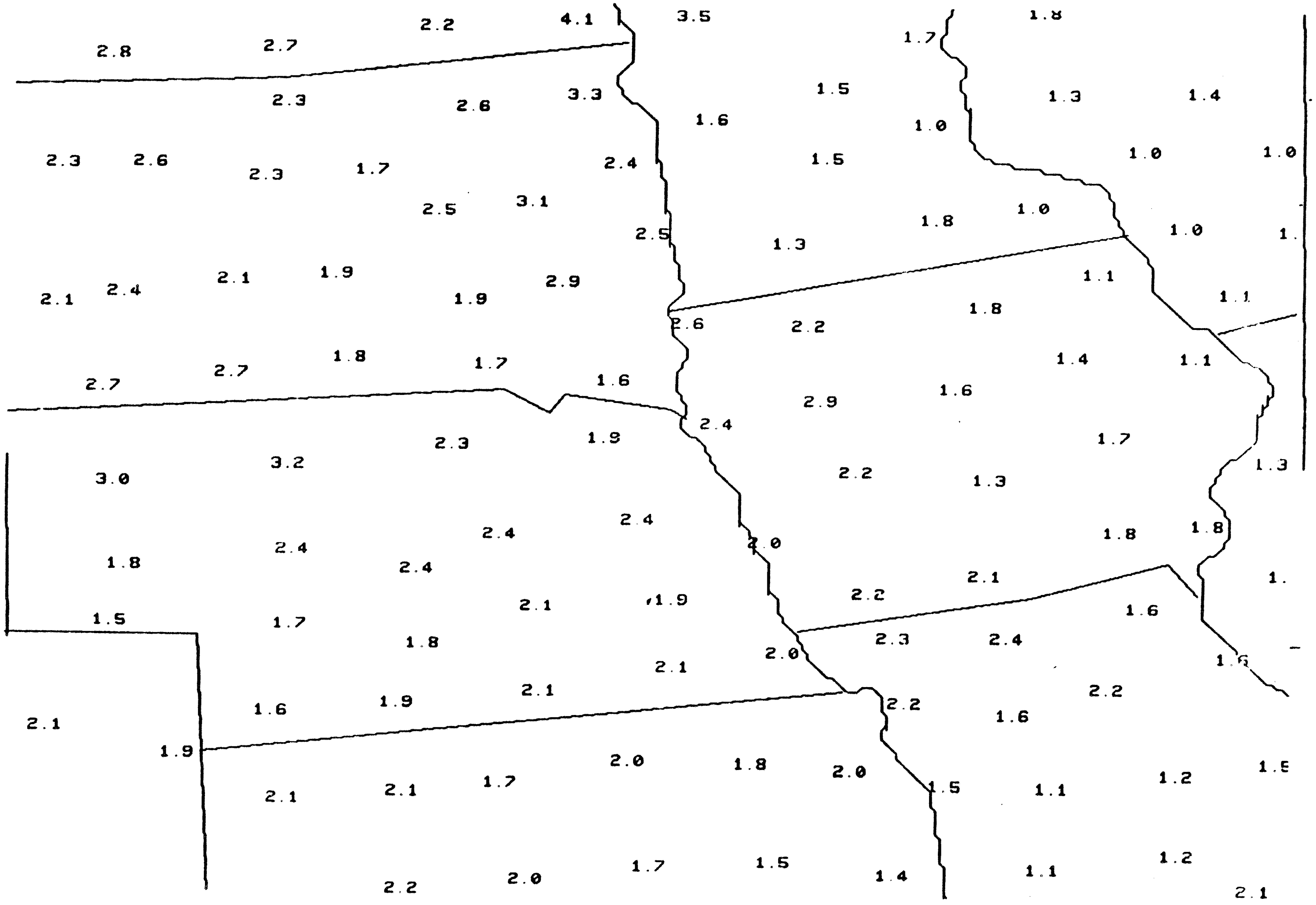


Figure A-5. Sample display of Flash Flood Guidance at 16:1 zoom (from figure A-4).

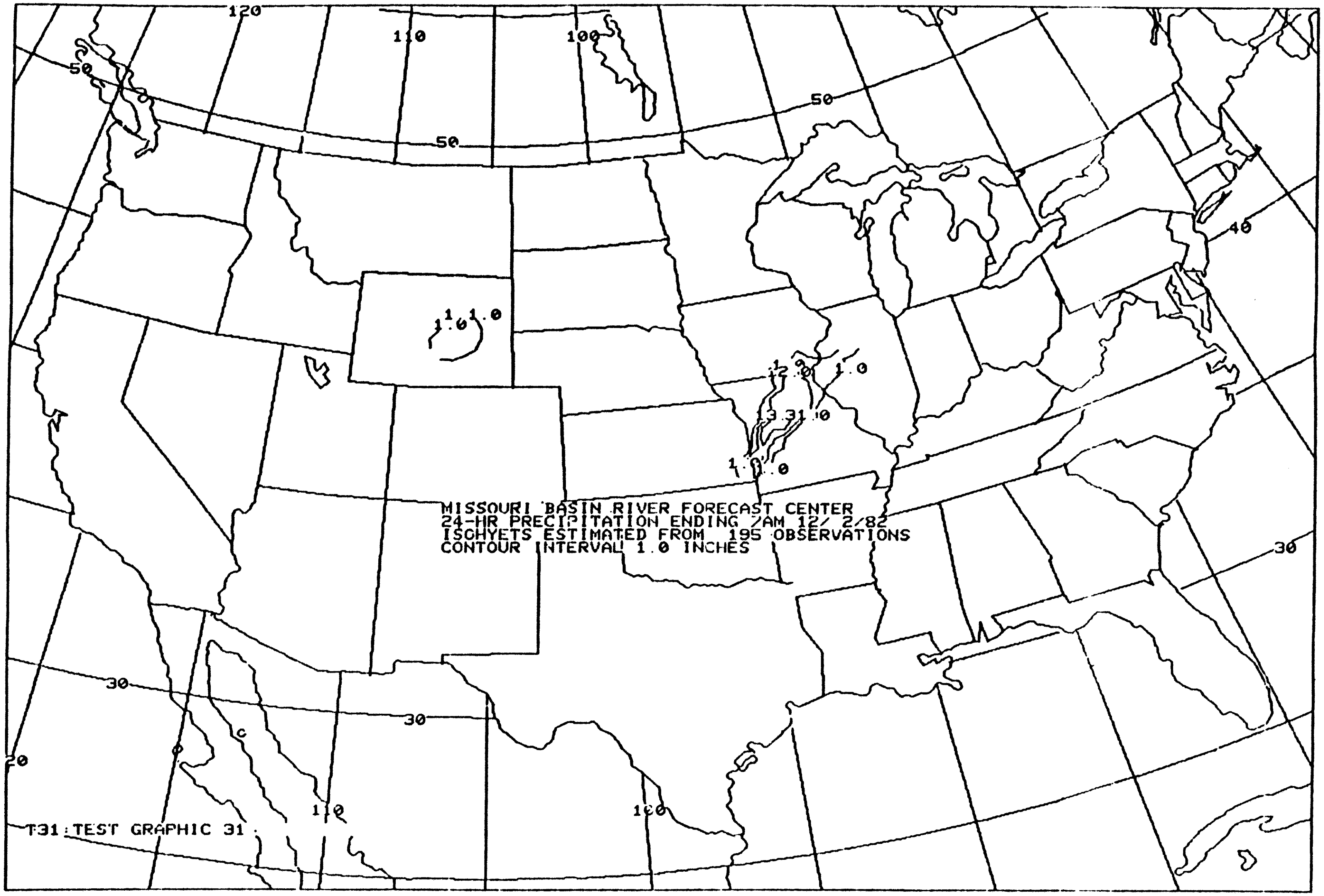

120
110
100
50
50
50
40
30
30
30
20
110
100
MISSOURI BASIN RIVER FORECAST CENTER
24-HR PRECIPITATION ENDING 7AM 12/ 2/82
ISOHYETS ESTIMATED FROM 195 OBSERVATIONS
CONTOUR INTERVAL 1.0 INCHES
1.0
1.0
2.0
1.0
3.0
1.0
T31:TEST GRAPHIC 31

Figure A-6. Contouring for 24-hour precipitation (inches) in the Missouri River Basin RFC area at 1:1 zoom.

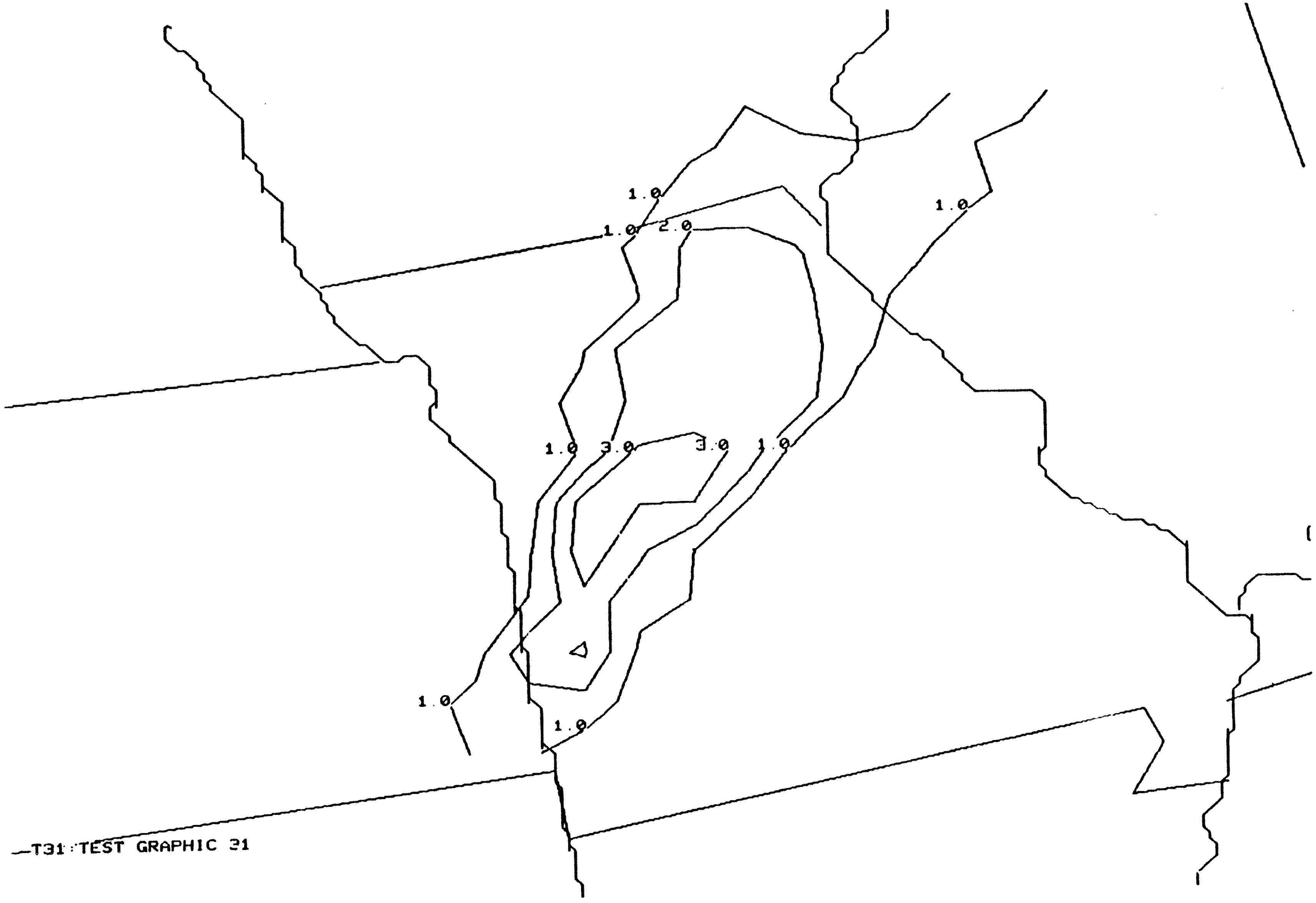


Figure A-7. Contouring for 24-hour precipitation at 25:1 zoom (from figure A-6).

APPENDIX B. TABLES

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 c flag		%	/	9	C	M	W	a	k	u	▨
8	back space	set c flag		&	0	:	D	N	X	b	l	v	
9	fwd space			'	1	;	E	O	Y	c	m	w	

SPECIAL CHARACTER SET

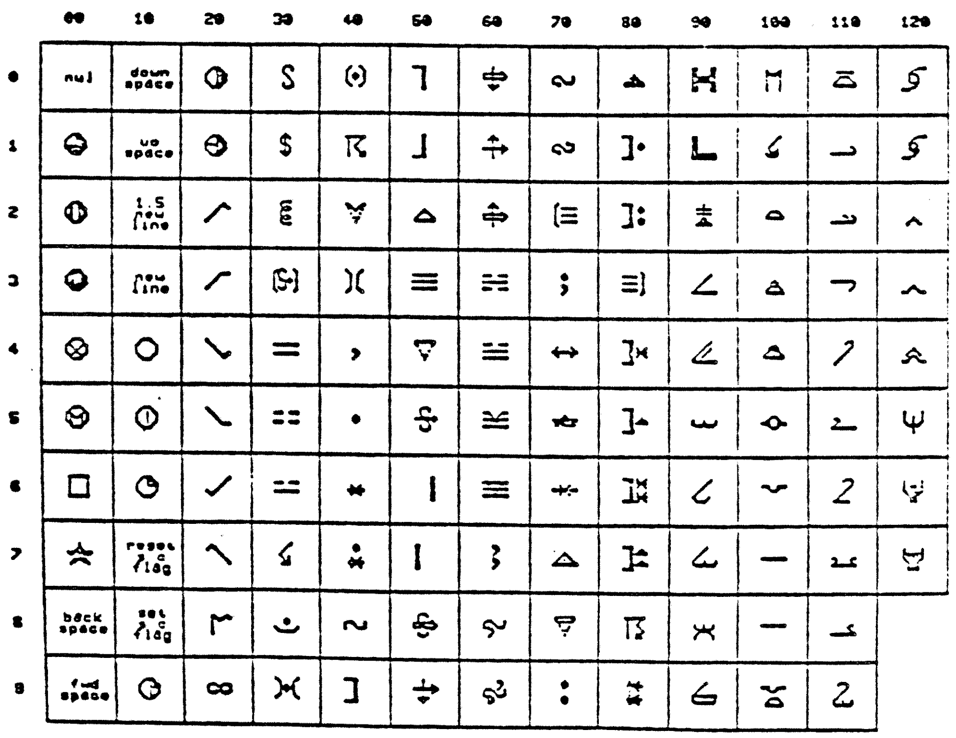


Table B-1. The standard and special character sets with their decimal values. Value equals column + row. Reprinted from MacDonald (1981).

APPENDIX C. SAMPLE PROGRAM

```
//WH3EJVTP JOB
//   TIME=1,REGION=256K
//*MAIN CARDS=(700,C)
//*MAIN ORG=AFOS1
//   EXEC  XFORXCLG,IUPVER='R12.'
//FORT.SYSIN  DD  *
C *****************************************************************************
C
C TEST OF AFOS GRAPHICS FROM IBM 360/195
C
C THIS PROGRAM WILL WRITE A TITLE WITH A DATE; THEN WILL DRAW A RIGHT
C TRIANGLE LABELING EACH SIDE (WITH THE HYPOTENUSE BEING COMPUTED).
C ALSO PUT A BORDER AROUND PLOT.
C
      DOUBLE PRECISION TITLE(2),LABEL(2)
      LOGICAL*1 NCTEXT(113),LLABEL(16),ZCZC(4),NNNN(4)
      DIMENSION NAFOS(3),IX(4),IY(4),WORK(1000)
C
      DATA TITLE/8HTHIS IS ,6HA TEST/
      DATA LABEL/8HY=6X=4Z=,1H /
      DATA NCTEXT(2),NCTEXT(5)/1H/,1H//
      DATA ZCZC/90,67,90,67/,NNNN/4*78/
      DATA IX/1000,1000,1400,1000/,IY/1000,1600,1000,1000/
C
      EQUIVALENCE (LABEL(1),LLABEL(1))
C
      LW=1000
C TITLE
      CALL AGALFA(2000,2900,TITLE,14,0,0,0,0,WORK,WORK,LW)
C
C DATE
      IMO=1
      IDA=25
      IYR=83
      CALL UINTCH(IMO,1,NCTEXT(1),NUSE,IER)
      IERSUM=IER
      CALL UINTCH(IDA,2,NCTFXT(3),NUSE,IER)
      IERSUM=IERSUM+IER
      CALL UINTCH(IYR,2,NCTEXT(6),NUSE,IER)
      IERSUM=IERSUM+IER
      IF (IERSUM.EQ.0) CALL AGALFA(2072,2866,NCTEXT,7,0,0,0,0,
     $ WORK,WORK,LW)
C
C RIGHT TRIANGLE
      DO 100 I=1,4
 100  CALL AGLINE(IX(I),IY(I),1,0,0,WORK,WORK,LW)
C
C PLOT BORDER
      CALL AGLINE(0,0,2,0,0,WORK,WORK,LW)
      CALL AGLINE(4095,0,1,0,0,WORK,WORK,LW)
      CALL AGLINE(4095,3071,1,0,0,WORK,WORK,LW)
      CALL AGLINE(0,3071,1,0,0,WORK,WORK,LW)
      CALL AGLINE(0,0,1,0,0,WORK,WORK,LW)
C
C LABELS
      CALL AGALFA(892,1200,LLABEL,3,0,0,0,0,WORK,WORK,LW)
      CALL AGALFA(1036,966,LLABEL(4),3,0,0,0,0,WORK,WORK,LW)
C
C HYPOTENUSE
      Z=(6**2 + 4**2)**0.5
      CALL URELCH(Z,3,LLABEL(9),1,NUSE,IER)
      IF (IER.EQ.0) CALL AGALFA(1328,1200,LLABEL(7),5,0,0,0,0,
     $ WORK,WORK,LW)
C
C CALL AGUTF
C USE HEADER TYPE 2
      WRITE(7,200) ZCZC
      CALL AGUTF(NAFOS,2,NROUTE,WORK)
      WRITE(7,200) NNNN
 200  FORMAT(4A1)
C
      STOP
      END
/*
//LKED.MYLIB  DD  DSN=NWS.AFOS.MAPBACK.LOAD,DISP=SHR
//LKED.SYSIN  DD  *
 ENTRY MAIN
 INCLUDE MYLIB(EVAFOSG)
/*
//
```

Figure C-1. Listing of test program.

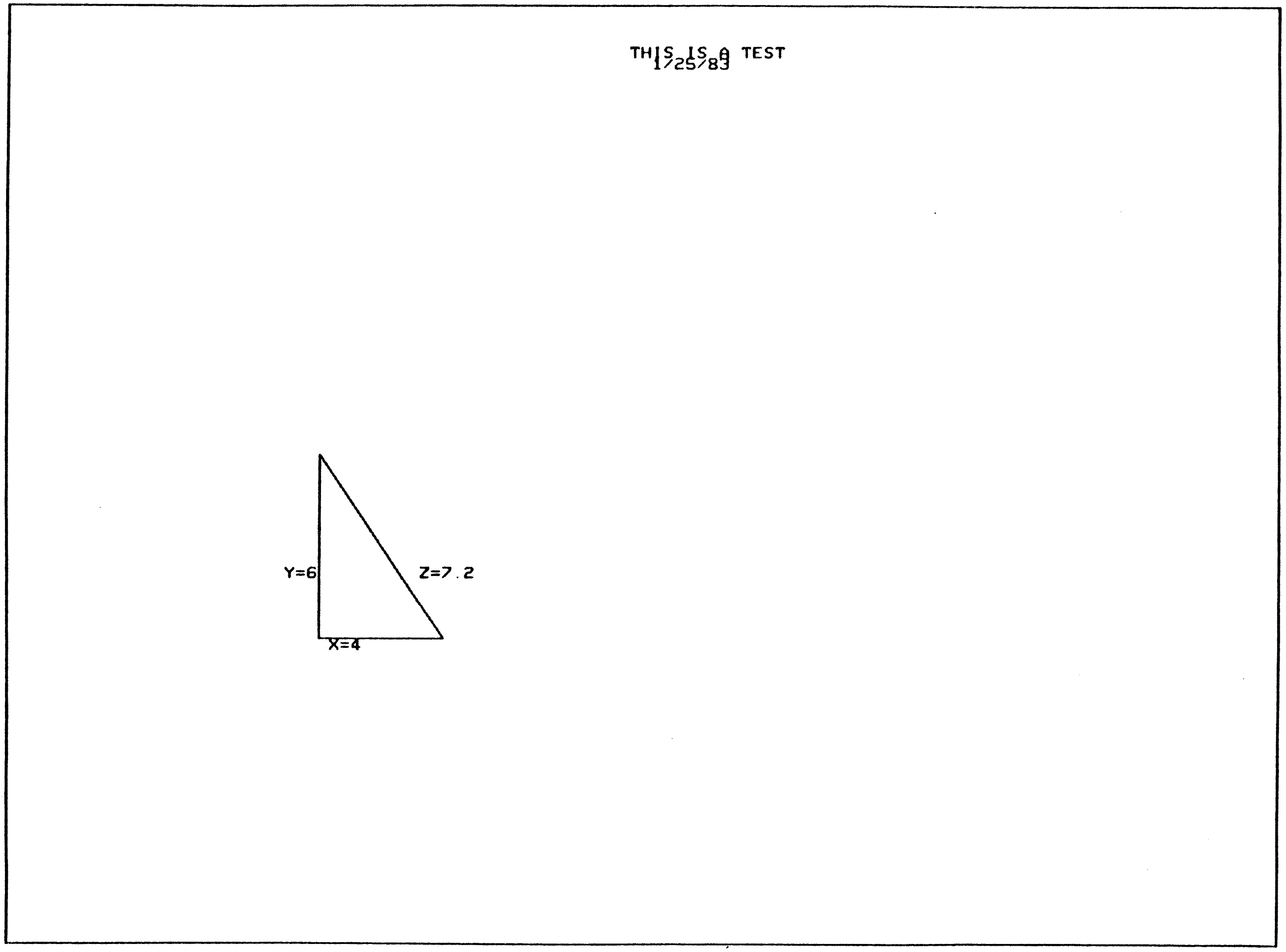


Figure C-2. Sample display created with test program.

APPENDIX D. SOURCE CODE

The locations of the source code for the subroutines are shown below along with information on "how to" list them.

1. Library - NWS.AFOS.MAPBACK.SOURCE contains:

Member Name	Subroutine
AFOSG	AGALFA, AGLINE, AGUTF
AGPOLR	AGPOLR

The above can be listed out using the PDS command on TSO or with a batch job using the NWS procedure NWSLISTM which is:

```
//   EXEC   NWSLISTM,DSNAME='NWS.AFOS.MAPBACK.SOURCE',MEMBER=XXXXXXX
     where XXXXXXX=name of member to be printed
```

2. Library - NWS.MKC.PROD.PANVALET.SOURCE contains:

Member Name	Subroutine
AGCONTUR	AGCNTR
AGLENGTH	AGLENG
AGRANGRD	AGRGRD
AGSPLTXY	SPLTXY (not user-callable)

The above can be listed out with the following batch job:

```
//    JOB    REGION=100K,TIME=1
//STEP1    EXEC    PROC=PAN#1
//PANDO1     DD    DSN=NWS.MKC.PROD.PANVALET.SOURCE,DISP=SHR
//SYSPRINT  DD    SYSOUT=A
++WRITE    PRINT,XXXXXXXX (where XXXXXXX=member name)
  (Repeat above card for each member to be listed.)
/*
//
```

3. Library - NWS.RFS5.FCST.SOURCE contains:

Member Name	Subroutine
UINTCH	UINTCH
URELCH	URELCH

The above can be listed with the same batch job shown in 2 above. The third card must be changed to:

```
//PANDO1   DD   DSN = NWS.RFS5.FCST.SOURCE, DISP=SHR
```