The Graphics System
Implementor's Guide

Editor: Gregory R. McArthur
# The Graphics System Implementor’s Guide

## Table of Contents

### CHAPTER 1: SYSTEM PLOT PACKAGE PHILOSOPHY

- Overview ................................................................. 1-1
- Formation ................................................................... 1-1
- Transportation ............................................................ 1-2
- Enhancements .............................................................. 1-2

### CHAPTER 2: IMPLEMENTING THE SYSTEM PLOT PACKAGE ON A TARGET COMPUTER

- Acquisition .................................................................. 2-1
- Support ........................................................................ 2-1
- Testing ......................................................................... 2-2
- Sample Picture ............................................................ 2-3

### CHAPTER 3: REQUIRED LOCALLY IMPLEMENTED SUPPORT Routines FOR TRANSPORTING THE NCAR SYSTEM PLOT PACKAGE

- Subroutine ULIBER .................................................... 3-1
- Subroutine WRITEB ...................................................... 3-1
- Subroutine ENCODE ..................................................... 3-1
- Subroutine GETCHR ..................................................... 3-2
- Subroutine SETCHR ..................................................... 3-2
- Subroutine PACKUM .................................................... 3-2
- Subroutine PERR ........................................................ 3-2
- Function LOC (X) ......................................................... 3-2
- Function ISHIFT .......................................................... 3-2
- Function IAND ............................................................. 3-3
- Function IOR ............................................................... 3-3
- Function INTT (X) ....................................................... 3-3
- Function IMACH (I) ...................................................... 3-3
- Function RMACH (I) ..................................................... 3-4
- Routine Implementations .............................................. 3-5
- Routine Implementations (Con’t) ................................... 3-6
- Routine Implementations (Cont) .................................... 3-7

### CHAPTER 4: HOW TO WRITE A METACODE TRANSLATOR TO ADD A GRAPHICS DEVICE TO THE SYSTEM

- Introduction to Metacode ............................................. 4-1
- Portable Translation ..................................................... 4-1
- Metacode Instructions .................................................. 4-1
- Metacode Instructions (Con’t) ....................................... 4-3
- OP CODE Definitions .................................................. 4-4
- Paper Efficiency .......................................................... 4-5
## CHAPTER 5: NCAR'S GRAPHICS METAFILE STRUCTURE

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>5-1</td>
</tr>
<tr>
<td>Metafile Structure</td>
<td>5-1</td>
</tr>
<tr>
<td>Old vs. New Metafile Structure</td>
<td>5-2</td>
</tr>
</tbody>
</table>

## CHAPTER 6: UNDERSTANDING AN NCAR SYSTEM PLOT PACKAGE

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose</td>
<td>6-1</td>
</tr>
<tr>
<td>Formation</td>
<td>6-1</td>
</tr>
<tr>
<td>Structure</td>
<td>6-1</td>
</tr>
<tr>
<td>Variables and Use Tables</td>
<td>6-2</td>
</tr>
</tbody>
</table>

## CHAPTER 7: PREPROCESSING TO CREATE A SYSTEM PLOT PACKAGE

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Introduction</td>
<td>7-1</td>
</tr>
<tr>
<td>Deck Structure</td>
<td>7-1</td>
</tr>
<tr>
<td>Mods.</td>
<td>7-1</td>
</tr>
<tr>
<td>Examples</td>
<td>7-2</td>
</tr>
</tbody>
</table>

## CHAPTER 8: UNDERSTANDING THE MASTER SYSTEM PLOT PACKAGE

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preprocessing</td>
<td>8-1</td>
</tr>
<tr>
<td>Parameters</td>
<td>8-1</td>
</tr>
<tr>
<td>Master System Plot Package Excerpts</td>
<td>8-2</td>
</tr>
<tr>
<td>Comments About Flags</td>
<td>8-2</td>
</tr>
<tr>
<td>Comments About Sets and Macros</td>
<td>8-3</td>
</tr>
<tr>
<td>Sets and Macros Defined</td>
<td>8-6</td>
</tr>
<tr>
<td>Comments About Codes</td>
<td>8-9</td>
</tr>
</tbody>
</table>

## CHAPTER 9: SAMPLE PLOTS

<table>
<thead>
<tr>
<th>Plot</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>AUTOGRAPH</td>
<td>9-2</td>
</tr>
<tr>
<td>CONRAN.</td>
<td>9-3</td>
</tr>
<tr>
<td>CONRAS.</td>
<td>9-3</td>
</tr>
<tr>
<td>CONRAQ.</td>
<td>9-3</td>
</tr>
<tr>
<td>CONREC.</td>
<td>9-4</td>
</tr>
<tr>
<td>CONRECOCK.</td>
<td>9-4</td>
</tr>
<tr>
<td>CONRECSCMSMH.</td>
<td>9-5</td>
</tr>
<tr>
<td>CONRECSUPR.</td>
<td>9-5</td>
</tr>
<tr>
<td>DASHCHAR.</td>
<td>9-6</td>
</tr>
<tr>
<td>DASHLINE.</td>
<td>9-6</td>
</tr>
<tr>
<td>DASHSMHT.</td>
<td>9-6</td>
</tr>
<tr>
<td>DASHSUPR.</td>
<td>9-6</td>
</tr>
<tr>
<td>HAFTON.</td>
<td>9-7</td>
</tr>
<tr>
<td>ISOSRF.</td>
<td>9-8</td>
</tr>
<tr>
<td>ISOSRFHR.</td>
<td>9-8</td>
</tr>
<tr>
<td>PWRTX</td>
<td>9-9</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----</td>
</tr>
<tr>
<td>PWRTY</td>
<td>9-10</td>
</tr>
<tr>
<td>PWRTZI</td>
<td>9-10</td>
</tr>
<tr>
<td>PWRTZS</td>
<td>9-10</td>
</tr>
<tr>
<td>PWRTZH</td>
<td>9-10</td>
</tr>
<tr>
<td>SCROLL</td>
<td>9-11</td>
</tr>
<tr>
<td>SURFACE</td>
<td>9-12</td>
</tr>
<tr>
<td>STRMLN</td>
<td>9-13</td>
</tr>
<tr>
<td>SUPMAP</td>
<td>9-14</td>
</tr>
<tr>
<td>THREEED</td>
<td>9-13</td>
</tr>
<tr>
<td>VELVCT</td>
<td>9-17</td>
</tr>
<tr>
<td>WINDOW</td>
<td>9-17</td>
</tr>
</tbody>
</table>
OVERVIEW
The system plot package is used to perform low-level graphics tasks, such as scaling, line drawing, and character drawing. It is written with the goal of being as fast and small as possible while constrained to provide adequate sophistication to support various scientific plotting requirements. It has been made portable so as to provide a common graphics interface on all computers at NCAR. This also allows this interface to be implemented at many other installations. The system plot package produces a high-level device-independent instruction set to support various plotters at NCAR. This instruction set can be interfaced to virtually all line-drawing plotters.

FORMATION
System plot packages are configured for various computers by using a preprocessor called FRED. The FRED preprocessor translates an extension of FORTRAN, called FRED into FORTRAN. The most important FRED features used in writing the master system plot package were macros, parameters, and conditional compilation. Included among the parameters in the master system plot package are vari-
ables describing the target machine (such as the number of bits in a character). Preprocessing the master system plot package with these variables properly set produces a system plot package for a given computer.

**Transportation**

Many experts in the field of transporting software today define software as being portable if the ratio of the amount of work to implement the software from scratch on a new system to the amount of work to transport the software to the new system is a large number. By this standard, the system plot package is portable if the implementor obtains a version configured for the target machine. The transportation process consists of implementing the required support routines for bit manipulation and I/O, testing these routines with the portable test package, testing the plot package and support routines with the sample program, and implementing the device-independent code translator for some plotter. For a person familiar with the target computer who also has software which drives a plotter, this can generally be done with roughly two weeks effort.

**Enhancements**

Once the system is working, efficiency can be upgraded by improving the required locally implemented routines, and flexibility can be added by expanding the translators. To reduce the core requirements and improve the running time of the system plot package, the support routines should be rewritten in tight machine language code. The routines which are used most often are, in order: INTT, IAND, ISHIFT, GETCHR, IOR, PACKUM, WRITEB, LOG, SETCHR, ENCODE, TIMACH, and RIMACH. The portable test program should be used to test the improved versions. As experience is gained with each translator, users will realize what improvements are needed to make optimum use of each plotter for various applications. Highly sophisticated translators can take several months to implement.
CHAPTER 2: IMPLEMENTING THE SYSTEM PLOT PACKAGE ON A TARGET COMPUTER

ACQUISITION
A system plot package must be configured with parts of the code tailored to match the hardware of the target computer. If the Scientific Computing Division's (SCD) software request form is properly completed by someone outside NCAR who is requesting the system plot package, the code received will match the target computer. To generate at NCAR a system plot package configured for a particular computer, the master system plot package is preprocessed using FRED (see System Plot Package Philosophy, Preprocessing to Create a System Plot Package, and Understanding the Master System Plot Package in this manual; FRED write-up, Chapter 13 of the NCAR Software Support Library).

SUPPORT
To run a system plot package, 14 support routines must be implemented (see Required Locally Implemented Support Routines for Transporting the NCAR System Plot Package in this manual). The implementation of these routines must be tested using the portable test program (TEST.12 on PORTLIB at NCAR; this is automatically sent to anyone requesting the plot package via the software distribution procedure). Hereafter, the system plot package is assumed to include the tested support routines.

TESTING
The system plot package is tested using the portable test program which produces the instructions for two sample pictures (TEST.PLOT on PORTLIB at NCAR; this is automatically sent to anyone requesting the plot package via the software distribution procedure). Care should be taken that the BLOCK DATA routine is loaded; this may require some effort on systems which only load routines which have been referenced. Implementors should carefully check the marked comments in the BLOCK DATA routine. When the program is run, unless the appropriate DATA statement in the BLOCK DATA routine is changed, the device-independent instructions are written on unit 8. These instructions should be dumped and checked against either the octal or hexadecimal dumps contained on comments in the test program. To plot these instructions, a translator must be written (see How to Write a Metacode Translator to Add a Graphics Device to the System in this manual). The pictures corresponding to the instructions produced by the test program follow.
SAMPLE PICTURE

NCAR PLOT PACKAGE TEST
CHAPTER 3: REQUIRED LOCALLY IMPLEMENTED SUPPORT ROUTINES FOR TRANSPORTING THE NCAR SYSTEM PLOT PACKAGE

The following routines are needed by the portable system plot package.

**SUBROUTINE ULIBER (IERR, MESS, LMESS)**

Prints error message or prints error number and error message

- IERR = error number (not printed if 0)
- MESS = message to be printed including carriage control
- LMESS = number of characters in MESS (< 130)

**SUBROUTINE WRITEB (MBUFA, MBLEN, MUNIT)**

Writes metacode to plotter or other unit which will serve as input to a metacode translator

- MBUFA = variable containing the address of the buffer to be written (MBUFA = LOC(buffer))
- MBLEN = number of buffer words that actually contain metacode data. If NCAR's standard metafile format is to be observed, WRITEB should do a fixed length binary write of 11,520 bits, regardless of the value of MBLEN. If MBLEN=0, a frame has been completed and the appropriate action is to put the data buffer out and backspace over it. In this way, the standard terminating record for a metafile will follow the last frame of the metafile.
- MUNIT = unit number where instructions are to be written

**SUBROUTINE ENCODE (NCHARS, IFMT, IRESLT, VAR)**

The NCHARS characters that would result from writing VAR using format IFMT are stored in IRESLT. Thus,

```
CALL ENCODE(4, 6H(F4.2), IRESLT, 1.)
```

would result in IRESLT containing the 4 character string "161.0", left justified. The system plot package and NCAR utilities always call ENCODE with an E, F, G, or I immediately following the (, so it is possible to implement ENCODE without a full-blown FORMAT cracker.
SUBROUTINE GETCHR (ICHARS, LEN, N, JCHAR)

Extracts character number N from character string ICHARS which is LEN characters long and returns it in JCHAR, right-justified with zero fill. If N is less than or equal to zero or greater than LEN, GETCHR should return the right-justified, zero-filled representation for a blank.

SUBROUTINE SETCHR (ICHARS, LEN, N, JCHAR)

Takes a right-justified, zero-filled character stored in JCHAR and puts it in character position N of character string ICHARS which is LEN characters long. The rest of JCHAR is unchanged. If N is less than or equal to zero or greater than LEN, SETCHR should do nothing.

SUBROUTINE PACKUM (MBPRS, NMBPRS, MBUFA)

Packs 16-bit bytes into a buffer

MBPRS = an array containing 16-bit bytes, one per word, right-justified, zero-filled

NMBPRS = number of such bytes. NMBPRS is always a number such that 16*NMBPRS bits fit exactly in an integral number of words

MBUFA = a variable containing the address of the start of the words into which the packed instructions are to be put. The bytes are not put in MBUFA; they are put starting at the location pointed to be MBUFA

SUBROUTINE PERROR

A fatal error has occurred in the system plot package. This routine should implement some graceful error exit.

FUNCTION LOC (X)

Address of X relative to the start of the program. If I and J are default-size integers stored in sequential locations in memory, then LOC(J) - LOC(I) = 1. Similarly, if M is a default size-dimensioned integer variable, then LOC(M(2)) - LOC(M(1)) = 1. LOC is an intrinsic function on NCAR's 7600 system and is, therefore, not included in the listings that follow.

FUNCTION ISHIFT (IWORD, N)

IWORD shifted by N bits. If N > 0, a left circular shift (ROTATE); if N < 0, a right end off shift. |N| ≤ word length.

3-2
**FUNCTION IAND (K1, K)**

The bit by bit logical product of K1 and K2

<table>
<thead>
<tr>
<th>Each K2 bit</th>
<th>( \text{IAND} )</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
</table>

| K1 | 0 | 0 | 0 |
| bit | 1 | 0:1 |

**FUNCTION IOR (K1, K)**

The bit by bit logical sum of K1 and K2

<table>
<thead>
<tr>
<th>Each K2 bit</th>
<th>( \text{IAND} )</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
</table>

| K1 | 0 | 0 | 1 |
| bit | 1 | 1:1 |

**FUNCTION INT (X)**

.\( \text{TRUE.} \) if \( X \) is an integer (Unless the user makes an error, the integers will be positive and \( \leq 2^{15} \), and unless the user changes the plotter resolution with \( \text{SETI} \), the integers will be positive and \( \leq 1024 \).)

.\( \text{FALSE.} \) Otherwise

If a floating point zero cannot be distinguished from an integer zero, .\( \text{FALSE.} \) should be returned for zero.

**FUNCTION I1MACH (I)**

This function is used to set up 16 machine constants:

\[
\begin{align*}
I1MACH(1) &= \text{the standard input unit} \\
I1MACH(2) &= \text{the standard output unit} \\
I1MACH(3) &= \text{the standard punch unit} \\
I1MACH(4) &= \text{the standard error message unit} \\
I1MACH(5) &= \text{the number of bits per integer storage unit} \\
I1MACH(6) &= \text{the number of characters per integer storage unit} \\
I1MACH(7) &= A, \text{ the base} \\
I1MACH(8) &= S, \text{ the number of base-A digits} \\
I1MACH(9) &= A^{S-1}, \text{ the largest magnitude}
\end{align*}
\]

Assume integers are represented in the \( S \)-digit, base-\( A \) form

\[
\text{SIGN}(X(S-1)A^{S-1}+...+X(1)A+X(0))
\]

where \( 0 \lt X(I) \lt A \) for \( I=0, ..., S-1 \).
Assume floating point numbers are represented in the T-digit, base-B form

\[ \text{SIGN}(B^{**E})*((X(1)/B)+...+(X(T)/B^{**T})) \]

where \( 0.\ LT.\ X(1) \), and \( EMIN.\ LE.\ LE.\ EMAX \)

I1MACH(10) = B, the base

Single precision:

I1MACH(11) = T, the number of base-B digits
I1MACH(12) = EMIN, the smallest exponent E
I1MACH(13) = EMAX, the largest exponent E

Double precision:

I1MACH(14) = T, the number of base-B digits
I1MACH(15) = EMIN, the smallest exponent E
I1MACH(16) = EMAX, the largest exponent E

**FUNCTION R1MACH (I)**

This function sets 5 single-precision machine constants.

R1MACH(1) = B**(EMIN-1), the smallest positive magnitude
R1MACH(2) = B**EMAX*(1-B**(-T)), the largest magnitude
R1MACH(3) = B**(-T), the smallest relative spacing
R1MACH(4) = B**(1-T), the largest relative spacing
R1MACH(5) = LOG10(B)

The following are implementations of these routines using NCAR system-dependent features. LOC is an inline function at NCAR, so it is not listed here.
ROUTINE
IMPLEMENTATIONS

*********** THESE ROUTINES ARE NOT PORTABLE ***********

SUBROUTINE ULIBER(IERR,MES,LMES)
DIMENSION MESS(1)
IF (IERR.EQ.0) WRITE(6,1) IERR
MESS=(LMES-1)/10+1
WRITE(6,2) (MESS(1),i=1,NWORDS)
RETURN
1 FORMAT(6HOIEMEEm=15)
2 FORMAT(13A10)
END

SUBROUTINE WRITEB(MBUFA,MBLEN,MUNIT)
DIMENSION IDUMMY(1)
DATA NWORDS/192/
ISTART = MBUFA-LOC(IDUMMY) + 1
IEND = ISTART + NWORDS - 1
BUFFER OUT(MUNIT,1) (IDUMMY(ISTART),IDUMMY(IEND))
1 IF (UNIT,MUNIT) 1,2,3,3
2 IF (MBLEN .EQ. 0) BACKSPACE MUNIT
RETURN
3 CALL ULIBER(0,32HOPLOT PACKAGE BUFFER WRITE ERROR,32)
STOP
END

SUBROUTINE ENCODE(NCHARS,IFMT,RESULT,VAR)
DIMENSION IFMT(1)
ENCODE(NCHARS,IFMT,RESULT) VAR
RETURN
END

SUBROUTINE GETCHR(ICHARS,LEN,N,JCHAR)
DIMENSION ICHARS(1)
IF(N.LT.1.OR.N.GT.LEN) GO TO 1
IWORD=(N-1)/10+1
IPOS=N-((IWORD-1)*10
JCHAR=AND(ISHIFT(ICHARS(IWORD),IPOS=6-60),77B)
RETURN
1 JCHAR=55B
RETURN
END

SUBROUTINE SETCHR(ICHARS,LEN,N,JCHAR)
DIMENSION ICHARS(1)
IF(N.LT.1.OR.N.GT.LEN) RETURN
IWORD=(N-1)/10+1
IPOS=N-((IWORD-1)*10
ICHARS(IWORD)=ISHIFT(IOR(JCHAR,AND(ISHIFT(ICHARS(IWORD),6=IPOS),
C RETURN
END
ROUTINE IMPLEMENTATIONS (Con't)

SUBROUTINE PACKUM(MBPRS,NMBPRS,MBUFA)
DIMENSION IDUMMY(1),MBPRS(NMBPRS)
I SUB=MBUFA-LOC(IDUMMY)+1
CALL BYTE$('DUMMY',ISUB),MBPRS,0,10,0,NMBPRS)
RETURN
END

SUBROUTINE PERROR
DATA IENTRY/0/
IF(IENTRY.NE.0) GO TO 2
IENTRY=1
CALL FLUSH
WRITE(6,3)
WRITE(6,4)
STOP
2 WRITE(0,3)
WRITE(0,5)
WRITE(0,4)
GO TO 1
3 FORMAT(2BH0FATAL ERROR IN PLOT PACKAGE)
4 FORMAT(32HIGNORE THE FOLLOWING ERROR TYPE)
5 FORMAT(23HCOULD NOT FLUSH BUFFER)
END

FUNCTION ISHFT(IWORD,N)
ISHFT=SHIFT(IWORD,N)
RETURN
END

FUNCTION IAND(K1,K2)
IAND=AND(K1,K2)
RETURN
END

FUNCTION IOR(K1,K2)
IOR=OR(K1,K2)
RETURN
END

FUNCTION INTT(K)
LOGICAL INTT
DATA MASK/3770000000000000000000000000000B/
INTT=IAND(IABS(K),MASK).EQ.0
IF(K,EQ.0) INTT=.FALSE.
RETURN
END

C LOC IS AN INTRINSIC FUNCTION ON THE NCAR SYSTEM.
INTEGER FUNCTION II MACH(I)
INTEGER IMACH(16)
C MACHINE CONSTANTS FOR THE CDC 6000/7000 SERIES
DATA IMACH( 1) / 5 /
DATA IMACH( 2) / 6 /
DATA IMACH( 3) / 3581 /
DATA IMACH( 4) / 6 /
DATA IMACH( 5) / 60 /
DATA IMACH( 6) / 10 /
DATA IMACH( 7) / 2 /
DATA IMACH( 8) / 48 /
DATA IMACH( 9) / 00007777777777777777B/
DATA IMACH(10) / 2 /
DATA IMACH(11) / 48 /
DATA IMACH(12) / -974 /
DATA IMACH(13) / 1070 /
DATA IMACH(14) / 96 /
DATA IMACH(15) / -927 /
DATA IMACH(16) / 1070 /
II MACH = IMACH(I)
RETURN
END

REAL FUNCTION R1 MACH(I)
REAL RMACH(5)
DATA RMACH(1) / 000140000000000000000000 B /
DATA RMACH(2) / 37767777777777777 B /
DATA RMACH(3) / 164040000000000000000000 B /
DATA RMACH(4) / 164140000000000000000000 B /
DATA RMACH(5) / 17164642023241175720 B /
R1 MACH = RMACH(I)
RETURN
END
INTRODUCTION TO METACODE

The device-independent instruction set (metacode) can be considered to be the hardware instructions for an imaginary plotter with extensive and extendable capabilities. This plotter has $2^{15}$ (or 32,768) addressable positions along each axis, multiple intensities and spot sizes, color, hardware characters in various sizes and fonts, and can have other required capabilities added on. The instruction set is 8-bit-byte-oriented and is more or less dictated by the nature of most minicomputers that will be used for the metacode translation.

PORTABLE TRANSLATION

A portable translator for translating NCAR metacode is available. This translator makes some efficiency sacrifices to achieve other goals. Because it is portable, the translator is easily moved from one host computer to another. Because it can reduce high-level metacode constructs to low-level ones, the least sophisticated plotters can be supported. Because the code has clearly marked interfaced points, a new plotter can be added to the system with a small effort.

METACODE INSTRUCTIONS

There are three types of instructions (see figure, next page). First is a four-byte positioning instruction to control pen movement containing a coordinate pair and a pen control bit. The next is a multibyte instruction, containing an operation code, the number of bytes to follow, and a character string. The last type of instruction is a two-byte increment instruction, for those users concerned with code compaction or very high resolution.

The positioning instruction is the basic line drawing instruction for the abstract plotter, reflecting the full precision of the device. This resolution corresponds to a picture size of almost three feet square with a resolution of one thousandth of an inch. The pen control bit position was chosen to facilitate translation on machines with short word lengths.

The multibyte instructions are used to control functions of the abstract plotter other than pen motion. Sixty-four operation codes are possible; half are reserved for use by the plot package designer, half are user-definable. The range of user-defined option codes is quite large, including picture descriptors, code compaction, links with interactive equipment, and metacode translator control. Users are, of course, required to add routines to the translator to react to these instructions; otherwise, they should be ignored.
The increment instruction can be used for code compaction, achieving up to a two-to-one compaction of the instructions for curves. (Other compaction methods are available through the multibyte instruction.)

**4 BYTE INSTRUCTION (ABSOLUTE POSITION)**

```
+---+---+
| 15-15 |
| X     Y |
|        |
```

**MULTIBYTE INSTRUCTION**

```
+---+---+---+---+
| 8-8-8-8 |
| OP CODE N+ BYTE1 ... BYTE N+ |
```

**2 BYTE INSTRUCTION (RELATIVE POSITION; subtract 2^5 from X and Y)**

```
+---+---+
| 6-6-6-6 |
| X     Y |
|        |
```

Postprocessors for various plotters should be easy to write because of the limited form of the metacode input. No complicated utilities, such as axis drawing or scaling, need to be written for noncompacted metacode. For plotters where system plot packages already exist, a translator can be written using some of the existing code with minor modifications.

Authors of codes to translate metacode into machine instructions for a given plotter are strongly encouraged to parameterize operation codes (op codes), as these may be changed as experience is gained. Changes in instruction format are considered extremely unlikely.

†NOTE: In multibyte instructions, if N is odd, an extra byte is added to the end of the character string so all instructions are a multiple of 16 bits in length. This byte should be ignored.
METACODE
INSTRUCTIONS
(Con't)

* 0 ≤ MX, MY ≤ 32,767. Scale these numbers to fit particular plotter used.

+ It is often possible to use (a stripped down version of) the existing
(locally written or manufacturer's) plot package to do these functions.
A pen control bit is found in the 2-byte instruction and the 4-byte instructions. An op code appears in the multibyte instructions. If the multibyte instruction is setting an option, the first byte of the character string tells which option is being set.

PEN CONTROL BIT
0 Pen up
1 Pen down

OP CODE

33 PWRIT call. The N characters to be plotted are in Byte 1 through Byte N. The location of the center of the first character is determined by the most recent pen position and centering option call.

34 FRAME advance. N is 0.

35 OPTN Call

<table>
<thead>
<tr>
<th>Byte 1</th>
<th>specifies which option is being set</th>
<th>Value found in</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Case (0=upper, 1=lower)</td>
<td>Byte 2</td>
</tr>
<tr>
<td>2</td>
<td>Intensity (0 through 255)</td>
<td>Byte 2</td>
</tr>
<tr>
<td>3</td>
<td>Orientation of characters in degrees</td>
<td>Bytes 3 &amp; 4</td>
</tr>
<tr>
<td>4</td>
<td>Character width</td>
<td>Bytes 3 &amp; 4</td>
</tr>
<tr>
<td>5</td>
<td>Font</td>
<td>Byte 2</td>
</tr>
<tr>
<td>6</td>
<td>Dash pattern (16 bits)</td>
<td>Bytes 3 &amp; 4</td>
</tr>
<tr>
<td>7</td>
<td>Spot size (0 through 32767)</td>
<td>Bytes 3 &amp; 4</td>
</tr>
<tr>
<td>8</td>
<td>Centering option</td>
<td>Byte 2</td>
</tr>
<tr>
<td></td>
<td>(0=left, 1=center, 2=right)</td>
<td>Bytes 2, 3, &amp; 4</td>
</tr>
<tr>
<td>9</td>
<td>Color</td>
<td></td>
</tr>
</tbody>
</table>

† Exact meanings of some options will be established when plotters are available at NCAR with these capabilities.

39 Range information

Byte 1 & 2 MINMX
3 & 4 MINMY
5 & 6 MAXMX
7 & 8 MAXMY

40 Points Mode, N is 2. If the byte 1 is 0, turn points mode off. If byte 1 is 1, turn points mode one with a simple point as the marker. If byte 1 is 2, turn points mode one with the character in byte 2 as the marker. When in points mode, the current marker is to be put at the pen position.
resulting from the normal execution of all normal 2-byte and 4-byte pen-up move and pen-down draw instructions.

**PAPER EFFICIENCY**

Besides OP CODEs 33, 34, and 35, most NCAR system plot packages produce a multibyte instruction with OP CODE=39. The 8 bytes following contain the approximate range of all plotting for the preceding frame (MXMIN, MYMIN, MXMAX, MYMAX). This can be very useful for minimizing paper waste on drum plotters. On other plotters, this instruction can be ignored. The location of this instruction is described in the next chapter NCAR's Graphics Metafile Structure of this manual. NCAR-supported metacode translators and portable translator shells contain proper logic to use this information if the ISIZE=3 option is set in the main program of the translator (and assuming the two READ statements in routine GETRNG are appropriate for the host machine).
CHAPTER 5: NCAR'S GRAPHICS METAFILE STRUCTURE

INTRODUCTION

This section describes the structure of a standard NCAR graphics metafile. Such a file is a stream of data records containing metacode instructions as put out by any of the supported metacode plot packages and as expected on input by any supported metacode translator or other metacode manipulator. The metacode instruction set itself is described in other documents in this manual.

METAFILE STRUCTURE

All records in the metafile have a fixed length of 11,520 bits. This is equivalent to 1440 8-bit bytes or 720 16-bit words, and is an integral multiple of the word length of any available computer.

The first 32 bits of each record are reserved for control information, and are divided into two 16-bit fields. The first field is the count of 8-bit metacode data bytes in the record. The count does not include the 32 bits of control information. The first 4 bits of the second 16-bit field are a data type identifier. A binary value of 0010 for this flag identifies the record as containing metacode. The next single bit is used to indicate frame boundaries. The bit is set to indicate that the record is the first data record of a new frame. The remaining 11 bits of the second 16-bit field are unused at the present time. Following the record control information are the metacode data.

Frames must begin on record boundaries. If the data do not fill the physical record, as will often be the case with the last data record of a frame, then the contents of the remaining space in the record are undefined. The last few metacode instructions of the last data record of a frame will typically be 16-bit no-op instructions (A020 in hex, or 120040 in octal). These will be included in the data count for the record; the number present will depend on the graphics image itself and the word size of the computer which generates the metafile. If the metafile includes a range instruction for each frame, it will immediately precede any no-op instructions at the end of a frame. The range instruction (which is 10 bytes long) will never cross record boundaries, hence, it will always be found in the last data record for a frame. There may (with a probability of less than 1%) be a few no-op instructions preceding the range instruction. The frame instruction itself will immediately precede whatever combination of range instruction and no-ops may be present.

A normal metafile; i.e., one resulting from a job that terminated normally, should have a single final record
with no data in it and zero in the byte count field. This allows multiple metafiles to be in a single data file ("file" in the FORTRAN sense), thus skirting the portability problems raised by the lack of the multifile dataset concept in FORTRAN 66. In order that metafiles from abnormally terminated jobs may be successfully processed, an end-of-file mark should be considered synonymous with the special terminating record.

OLD vs. NEW METAFILE STRUCTURE

For users familiar with NCAR'S original metafile structure, and possibly maintaining software based on that structure, the following differences should be noted. The old structure was based on record pairs, with the first record of each pair being a 16-bit (at least) byte count record, and the second being the accompanying variable length data record. These two records are now merged into a single fixed length record, the first 16 bits of which is the data byte count. The notion of a "zero-byte-count record" to partition frames is replaced by defining one of the second 16 bits of each record as a frame boundary flag. A single "zero-byte-count record" now terminates the metafile, whereas two consecutive such records did so previously. Finally, if the metafile contains a range instruction for each frame, it will now be found immediately following the frame instruction that terminates a given frame, instead of occupying a record by itself at the very end of the frame.
CHAPTER 6: UNDERSTANDING AN NCAR SYSTEM PLOT PACKAGE

PURPOSE
This section is intended to help programmers understand the code in an NCAR System Plot Package (NSPP). The term "an NSPP" is used because there are many versions configured for various levels of output code compaction. The output code is called "metacode," a device-independent plotter instruction set which is fully described in How to Write a Metacode Translator to Add a Graphics Device to the System.

An NSPP contains the low-level routines used by NCAR graphics utilities (contouring, etc.) to draw lines and characters on various plotters. The most complex task performed by an NSPP is the drawing of axes with numerical labels at the tick marks or grid lines. Any tasks which are more complex are done by utilities.

FORMATION
An NSPP is produced for a given machine architecture by specifying character and word sizes which are used in preprocessing a master system plot package written in FRED. The output of the preprocessor is a particular system plot package in FORTRAN. Thus, at NCAR, when errors are to be fixed, the changes are made in the master system plot package and the various system plot packages are reformed by using the preprocessor.

STRUCTURE
It is the nature of system plot packages that they do not flow a great deal; looping and conditional structures are seldom used. Rather, routines are entered, their arguments are examined, and entries made in storage local to the plot package. Thus, understanding an NSPP is based on understanding the arguments to the various routines and understanding the various variables, tables, and buffers of the system plot package. The routine arguments are described in the user documentation, The System Plot Package.

NSPP routines store all information needed for subsequent processing in a named common block. Local variables of a given routine are not used to store information needed in later calls. The routines are serially reusable but not reentrant. (Local variables are used for DO-loop indices, etc.) The following table describes the variables that may be in the common block. Not all NSPPs contain all the variables. All integer variables in the common block start with M and no other variables do. Except for SMALL, all real variables in the common block start with X or Y.
<table>
<thead>
<tr>
<th>VARIABLES</th>
<th>USE</th>
</tr>
</thead>
<tbody>
<tr>
<td>MMAJX, MMAJY, MMINX, MMINY, MXLAB, MYLAB, MFLG</td>
<td>GRIDAL arguments stored here so they will be in a known order for insertion in the instruction stream ONLY when ultracompact metacode is being produced.</td>
</tr>
<tr>
<td>MTYPE</td>
<td>Scaling type of the most recent SET call.</td>
</tr>
<tr>
<td>MX, MY</td>
<td>Plotter address of the pen location.</td>
</tr>
<tr>
<td>MXA, MYA, MXB, MYB</td>
<td>Plotter address corresponding to the first four arguments of the most recent SET call.</td>
</tr>
<tr>
<td>MTYPEX, MTYPEY</td>
<td>A decoding of MTYPE -- 0 = LINEAR, 1 = LOG</td>
</tr>
<tr>
<td>XXA, YYA, XXB, YYB, XXC, YYC, XXD, YYD</td>
<td>Exact copies of the first eight parameters of the most recent SET call.</td>
</tr>
</tbody>
</table>
| XFACTR, YFACTR, XADD, YADD | Numbers computed from the most recent SET call arguments so that real valued coordinates can be translated to integers by  

\[
MX = XFACTR \times XX + XADD \\
MY = YFACTR \times YY + YADD
\]

and similarly for Y. |
<p>| XX, YY | Most recent coordinate input to the PLOT package |
| MFMTX, MFMTY, MUMX, MUMY, MSIZX, MSIZY, MXDEC, MYDEC, MXOR | Most recent LABMOD inputs except that MXDEC = 0 and MYDEC = 0 are decoded and MXDEC = 1 and MYDEC = 1 become 0. |
| MOP(I), MNAME(I) | Option names are given in MNAME and their current values in MOP |
| MXOLD, MYOLD, MXMAX, Mymax, MXFAC, MYFAC | All used for increment instructions only. MXOLD and MYOLD are the plotter coordinates of the previous point, MXMAX and Mymax are the greatest distance an increment can move, and MXFAC and MYFAC are the number of plotter units per increment unit (generally 1, but can be more if compaction is important and high resolution is not). |</p>
<table>
<thead>
<tr>
<th><strong>MODEF</strong></th>
<th><strong>MTABLE(64)</strong></th>
<th><strong>MF2ER</strong></th>
<th><strong>MSHFTX, MSHFTY</strong></th>
<th><strong>MMGRX, MMGRIY, MMNRX, MMNRY</strong></th>
<th><strong>MASKD, MASKOP, MPWRIT</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>= 0 Flash routines have not been used</td>
<td>Translation table to change 6-bit characters to 8-bit characters (only on 6 bit-per-character machines).</td>
<td>= 0 no FLASH buffer overflow.</td>
<td>The power of two of the ratio between the resolution of the metacode address and the resolution the user is working in. In the default case, the user assumes the plotter is 1024 by 1024 (1024 = (2^{10})). Metacode addresses have 15 bits, so their capacity is 32,768. Thus, the default for (\text{MSHFTX} \text{AND MSHFTY}) is 5, and user integer coordinates are left shifted 5 to make plotter addresses.</td>
<td>Tick mark lengths (positive values point in).</td>
<td>For (\text{dd80}) plot package only. (\text{MASKD}) contains the various (\text{dd80}) instruction formats. (\text{MASKOP}) contains masks to set hardware options. (\text{MPWRIT}) contains masks for different numbers of trailing escape characters for character strings (which must be a multiple of 6 in length).</td>
</tr>
<tr>
<td>Variable</td>
<td>Description</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MCROUT</td>
<td>Number of metacode records that have been put out via PREOUT.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MFLCNT</td>
<td>Used to count the number of FLUSHB calls since last MBPRS initialization. It is used to avoid empty records which could otherwise be put out.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MFREND</td>
<td>FRAME sets to 1 to indicate last output call of a FRAME, and resets to zero before returning.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MFRLST</td>
<td>PREOUT manipulates, based on MFREND, so that it knows when a record is the first of a new frame.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MJXMIN, MJYMIN, MJXMAX, MJYMAX</td>
<td>Used to keep track of the range of the plotting address on the frame being created.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MNXSTO, MNYSTO, MXXSTO, MXYSTO</td>
<td>Used to hold MJXMIN,... after FLASH1 call, and restore them after FLASH2. MJXMIN,... are accumulated anew during FLASH saving, and stored in user FLASH buffer after FLASH2 call.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MPAIR1, MPAIR2</td>
<td>Two 16-bit pairs used to initialize each output record, so that PREOUT may format first 32 bits. They are actually put into MBPRS at proper times.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MPRINT</td>
<td>Unit number for printing error messages too extensive to be handled by ULIBER.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MSYBUF</td>
<td>Buffer to hold up to a few hundred metacode instructions.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MSBLEN</td>
<td>Word length of MSYBUF.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MNCFW</td>
<td>The number of characters per word on the host computer.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MINST</td>
<td>Holds instruction OP-CODE for the instruction being formed.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MBUFA</td>
<td>Contains the address of the buffer for the metacode instructions, either LOC(MSYBUF) or LOC(USER BUFFER) from a FLASH1 call.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Symbol</td>
<td>Description</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>----------</td>
<td>-----------------------------------------------------------------------------</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MBUFLU</td>
<td>The number of words of the buffer pointed to by MBUFA that have been filled with metacode or dd80 instructions.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MFWA, MLWA</td>
<td>Contains the first word address and the last word address for the FLASH buffers.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MIPAIR, MBPRS</td>
<td>MBPRS is used to store byte pairs of metacode until they can be packed in an integral number of words and placed in the buffer pointed to by MBUFA. MIPAIR tells how much of MBPRS has been used.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MBUFL</td>
<td>The length of the buffer pointed to by MBUFA.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MUNIT</td>
<td>Unit number for writing metacode.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SMALL</td>
<td>Smallest positive number on the host computer. This is used when non-positive numbers are plotted with log scaling.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
CHAPTER 7: PREPROCESSING TO CREATE A SYSTEM PLOT PACKAGE

INTRODUCTION

This chapter provides reference information for NCAR Graphics Project programmers, or for programmers with access to the NCAR CDC 7600 who wish to attempt construction of a custom graphics package.

The master system plot package is written in an extension of FORTRAN called FRED (see NCAR Software Support Library, Volume III, Chapter 13). The FRED preprocessor reads card images of its FORTRAN extension as input and creates a FORTRAN output file of card images.

DECK STRUCTURE

At this time, FRED runs only on NCAR's Control Data 7600 using NCAR's system. It is not portable. To create a system plot package, the following deck structure is used:

*JOB,ssss,ppppppp,user-name
*LIMIT,T=14S
*RUN,ED,BS=ULIB,BN=FRED
FETCH,S=(43510006),SN=PLOT.FRED,IL
(mods)
*EDIT,S=PSCR,D=PLIB,DN=name
*END

MODS

The mods are used to change the values of FRED-time variables. The nature of the mods determines the characteristics of the plot package produced. The FRED-time variables are described in Understanding the Master System Plot Package. The variables most commonly changed are listed in the following table:
<table>
<thead>
<tr>
<th>Name</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>LPLOTR</td>
<td>Specifies metacode output or dd80 output.</td>
</tr>
<tr>
<td>LCOMP</td>
<td>Specifies level of metacode compaction.</td>
</tr>
<tr>
<td>LPORT</td>
<td>Specifies if non-portable code for NCAR's system can be used or if portable code is required.</td>
</tr>
<tr>
<td>LNCPW</td>
<td>The number of characters that can be stored in a default length integer variable.</td>
</tr>
<tr>
<td>LPAIR</td>
<td>The number of 16-bit metacode packets that are stored before packing. Should be near 15 in magnitude and such that LPAIR*16 bits exactly fill an integral number of default-length integer words.</td>
</tr>
<tr>
<td>LOUTWD</td>
<td>The number of words exactly filled by LPAIR 16-bit packets.</td>
</tr>
<tr>
<td>LUMTCD</td>
<td>Unit number for metacode output.</td>
</tr>
</tbody>
</table>

Examples The following table gives a few sample mod decks for the production of typical plot packages. The card numbers listed should be verified against a listing of PLOTMASTER. The mod decks are in a form directly usable with the 7600 Editor.
<table>
<thead>
<tr>
<th>Computer</th>
<th>Output</th>
<th>Mods</th>
</tr>
</thead>
<tbody>
<tr>
<td>NCAR 7600</td>
<td>dd80</td>
<td>none</td>
</tr>
<tr>
<td>IBM 370</td>
<td>Metacode CHANGE 50, 199,</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;LPORT=0&quot;=&quot;LPORT=2&quot;, &quot;LPLOTR=0&quot;=&quot;LPLOTR=1&quot;</td>
</tr>
<tr>
<td>CRAY-1</td>
<td>Metacode CHANGE 50, 199,</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;LPORT=0&quot;=&quot;LPORT=2&quot;, &quot;LPLOTR=0&quot;=&quot;LPLOTR=1&quot;</td>
</tr>
<tr>
<td>PDP 11*</td>
<td>Metacode CHANGE 50, 199,</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>&quot;LPORT=0&quot;=&quot;LPORT=2&quot;, &quot;LPLOTR=0&quot;=&quot;LPLOTR=1&quot;</td>
</tr>
</tbody>
</table>

Please note that embedded blanks in the quoted strings above are necessary; they are indicated by $\_$. 

*using a compiler with default 16-bit integer variables.
CHAPTER 8: UNDERSTANDING THE MASTER SYSTEM PLOT PACKAGE

NOTE: This chapter provides reference information for NCAR Graphics Project programmers, or for programmers with access to the NCAR CDC 7600 who wish to attempt construction of a custom graphics package.

PREPROCESSING

The master system plot package is written in FRED and preprocessing it with FRED on ULIB results in the production of a system plot package with attributes determined by the state of various FRED-time variables. To understand the master system plot package, the FRED-time variables, plot package common block variables, and user argument lists must be understood. The plot package common block variables are described in Understanding an NCAR Plot Package. The user argument lists are described in the user documentation, The System Plot Package.

PARAMETERS

The FRED-time variables of the master system plot package fall into two classes: those which are never changed (such as L2(I) which is 2^i), and those which are changed to produce system plot packages with various characteristics. The following list contains the variables in the second class.

\[
\begin{align*}
L_{\text{PLOT}} &\quad L_{\text{NPW}} \\
L_{\text{COMP}} &\quad L_{\text{OUTWD}} \\
L_{\text{PACK}} &\quad L_{\text{PAIR}} \\
L_{\text{PORT}} &\quad L_{\text{SYBUF}} \\
L_{\text{OUTCD}} &\quad L_{\text{UMTCD}} \\
L_{\text{BUFFS}} &\quad L_{\text{FMTC}} \\
L_{\text{LAB}} &
\end{align*}
\]

†Most commonly changed variables

The following excerpt from the master system plot package contains explanations of all FRED-time variables. All FRED-time variables start with L and no other variables do.
SETS FOR DETERMINING THE TYPE OF CODE

LFORM  FORMAT FOR METACODE OUTPUT
0   FORTRAN BINARY OUTPUT
1   FORMATTED CARD IMAGES FOR BENCHMARKS

LPLOTR  TYPE OF PLOTTER CODE OUTPUT
1   METACODE - FOR ALL PLOTTERS
0   DD80 ONLY

LCOMP   TYPE OF COMPACITION. NON-ZERO ONLY WHEN LPLOTR IS TOO
0   NONE
1   INCREMENT INSTRUCTIONS USED, REDUNDANCIES OUT
2   HIGH LEVEL SUBSTITUTIONS USED FOR BACKGROUNDS

LPACK   TYPE OF ROUTINE PACKAGE BEING PRODUCED
0   A PLOT PACKAGE IS PRODUCED
1   MAKE A TRANSLATOR FOR THE DD80
2   MAKE A TRANSLATOR FOR THE CALCOMP
3   MAKE A TRANSLATOR FOR A PRINTER
4   MAKE A TRANSLATOR FOR A TEKTRONIX
5   MAKE A TRANSLATOR FOR DISTRIBUTION
6   MAKE A TRANSLATOR FOR EXAMINING METACODE

LPORT   PORTABILITY FLAG
0   NON-PORTABLE CODE
1   PORTABLE CODE TO A COMPUTER
   WITH 6 BITS PER CHARACTER
2   PORTABLE CODE TO A COMPUTER WITH 8 BITS PER CHARACTER
   WITH 8 BITS PER CHARACTER

LOUTCD  OUTPUT CODE
0   ASCII
1   EBCDIC

.SET LFOM=0
.SET LPLOTR=0
.SET LCOMP=1
.SET LPORT=0
.SET LOUTCD=0
.SET LPACK=0
F SET VARIABLES FOR CONSTANTS IN THE CODE
F MACROS FOR CONSTANTS IN THE CODE WHICH ARE FUNCTIONS OF OTHER THINGS
F MACRO AND SET DEFINITIONS
F NAME CONVENTION - ALL MACRO AND SET VARIABLES WILL START WITH L AND
F NOTHING ELSE WILL
F
F LADJST NUMBER TO PUT FLAG BITS IN OP CODES (11000000 BINARY)
F LBPOS MACRO TO BIAS A 2 BYTE NUMBER TO THE ALLOWED RANGE
F LBTBND MACRO TO BOUND INTEGERS TO LEGAL BYTE RANGE
F LBUFFS 0 THRU LBUFFS ALLOWED AS POINTER IN FLASH CALLS
F LBUF16 GET16 BUFFER SIZE IN NON-DD80 MC TRANSLATORS (THUS
F MAX PERMISSIBLE NUMBER OF 16-BIT PACKETS PER RECORD)
F LCAS OPTION NUMBER FOR CONTROLLING CHARACTER CASE
F LCENT OPTION NUMBER FOR SETTING CHARACTER CENTERING
F LCODE VALUE OF A FRED-TIME CONDITIONAL COMPILATION CODE,
F 0=ACTIVATE, 1=DEACTIVATE, 2=ELIMINATE
F LCOLR OPTION NUMBER FOR SETTING COLOR
F LDEC STANDARD DECREMENT FOR POSITION OF NUMERICAL LABELS
F ON AXES WHEN GENERATED BY GRIDAL
F LENGTH NUMBER OF CHARACTERS IN A FRED-TIME STRING
F LFMTC MAXIMUM NUMBER OF CHARACTERS ALLOWED IN THE FORMATS
F PASSED TO THE PLOT PACKAGE THROUGH A LABMOD CALL
F LFMTW NUMBER OF WORDS NEEDED FOR LFMTC CHARACTERS
F LFONT OPTION NUMBER FOR CONTROLLING CHARACTER FONT
F LFRAME FRAME OP CODE
F LFRST FRSTPT OP CODE
F LGRID SYSTEM OP CODE FOR GRIDAL CALL
F LHIGH HIGH INTENSITY (80 PCT OF AVAILABLE)
F LIMPOS IMPOSSIBLE NUMBER FOR INITIALIZING VARIABLES
F LINT OPTION NUMBER FOR CONTROLLING INTENSITY
F LLAB MAXIMUM NUMBER OF CHARACTERS IN GRIDAL LABEL (AND
F THUS THE LARGEST VALUE FOR NUMX AND NUMY IN LABMOD)
F LLABM SYSTEM OP CODE FOR LABMOD CALL
F LLOW LOW INSENSITIVITY (49.9 PCT OF AVAILABLE)
F LM(I) MACRO WHICH YIELDS A RIGHT-JUSTIFIED MASK I BITS LONG
F LMAXOP MAXIMUM NUMBER OF OPTIONS
F LMGR DEFAULT MAJOR TICK MARK LENGTH
COMMENTS
ABOUT SETS AND
MACROS (Con't)

F LMNR DEFAULT MINOR TICK MARK LENGTH
F LNBPC NUMBER OF BITS PER CHARACTER
F LNBPW NUMBER OF BITS PER WORD ON TARGET COMPUTER (DERIVED
F FROM LPAIR AND LOUTWD).
F LNCPW NUMBER OF CHARACTERS PER WORD ON THE COMPUTER
F LNMASK NUMBER OF MASKS FOR DD80 INSTRUCTION FORMATION
F LNOOP 2 BYTE NO-OP INSTRUCTION (PEN UP INCREMENT OF 0)
F LNOPM NUMBER OF OPTION MASKS FOR DD80 INSTRUCTIONS
F LOC(I) INVERSE OF LOC FUNCTION, THAT IS, CAUSES A
F SUBROUTINE TO LOOK IN ADDRESS I FOR AN ARGUMENT.
F LOPEND LAST SYSTEM OP CODE
F LOPNUM NUMBER OF SYSTEM DEFINED OPTION NUMBERS
F LOPPR MACRO TO SHIFT OP CODE AND PUT IN FLAG BITS IN PAIR
F LOPST FIRST SYSTEM OP CODE
F LOPSTA FIRST SYSTEM OP CODE WITH FLAG BITS
F LOR OPTION NUMBER FOR CONTROLLING CHARACTER ORIENTATION
F LOUTWD NUMBER OF WORDS THAT LPAIR BYTE PAIRS FILL
F LPAIR NUMBER OF BYTE PAIRS SAVED BEFORE PACKING
F LPAIRF MACRO TO FORM BYTE PAIR FROM TWO BYTES AT RUN TIME
F (BYTES ARE FORCED TO BE IN THE RANGE 0 THRU 255)
F LPAIR1 THE 16-BIT PATTERNS USED TO INITIALIZE THE FIRST
F LPAIR2 2 WORDS OF MBPRS EVERY LRBITS OF GENERATED DATA, SO
F THAT PREOUT MAY FORMAT FIRST 32 BITS BEFORE OUTPUT.
F LPAT OPTION NUMBER FOR CONTROLLING DASHED LINE PATTERNS
F LPOINT POINT OP CODE FOR DD80 INSTRUCTIONS
F LPSYM OP CODE FOR PSYM WHEN DD IS ACTIVATED.
F LPTS OP CODE TO SWITCH TO POINT MODE
F LPWRIT CHARACTER OP CODE
F LRANG OP CODE FOR INSTRUCTION CONTAINING RANGE
F OF PLOT WITHIN A FRAME
F LRBITS SIZE, IN BITS, OF FIXED-LENGTH METACODE OUTPUT
F LRELD OP CODE TO SWITCH TO RELATIVE ADDRESSING (DEFERRED)
F LSBUFF POINTER TO SYSTEM BUFFER BOUNDS
F LSCR NUMBER OF ADDRESSABLE POSITIONS ON PLOTTER
F LSCRPD POWER (OF 2) OF DEFAULT SCREEN SIZE
F LSETOP OPTION OP CODE
F LSIZ OPTION NUMBER FOR CONTROLLING CHARACTER SIZE
F LSSIZ OPTION NUMBER FOR SPOT SIZE
F LSYBUF SYSTEM BUFFER LENGTH
F LTICK SYSTEM OP CODE FOR SETTING TICK LENGTH
F LUDD80 DD80 UNIT NUMBER ON NCAR SYSTEM
<table>
<thead>
<tr>
<th>Prefix</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>F LUMTCD</td>
<td>UNIT NUMBER FOR OUTPUTTING METACODE</td>
</tr>
<tr>
<td>F LUNORM</td>
<td>MACRO TO PERFORM THE INVERSE OF LNORM</td>
</tr>
<tr>
<td>F LUSROP</td>
<td>NUMBER OF USER DEFINABLE OPTIONS</td>
</tr>
<tr>
<td>F LVECT</td>
<td>VECTOR OP CODE</td>
</tr>
<tr>
<td>F LWDS(NC)</td>
<td>FUNCTION TO COMPUTE THE NUMBER OF WORDS NEEDED TO HOLD NC CHARACTERS</td>
</tr>
<tr>
<td>F L2(I)</td>
<td>ITH POWER OF 2</td>
</tr>
</tbody>
</table>
SETS AND MACROS DEFINED

.MACRO LOCI(IADDR)=IDUMMY(IADDR-LOC(IDUMMY)+1)
.MACRO LWES (ICHARS)= "(ICHARS-1)/LNCPW+1"
.MACRO LBPPPOS(I)=IAND(I+L2(15),"L2(16)-1")
.MACRO LBTBND(IBYT)=IAND(IBYT,"L2(8)-1")
.MACRO LCODE (ICCC)= "ADE(ICCC)"
.MACRO LENGTH(ICHARS)= "LOS(ICHARS)"
.MACRO LM(I)= "L2(I)-1"
.F LOPPR(IOP,INOC) DEFINED BELOW BECAUSE MACHINE NATURE NEEDED.
.MACRO LPAIRF(IL,IR)=IOR(ISHIFT(LBTBND(IL) ,8) ,LBTBND(IR))
.MACRO LUNORM(Il,I2)=ISHIFT(Il,I2)-L2(16-I2)
.F
.F SETS TYPE INTEGER L2(47)
.SET L2(1)=2
.REPEAT (I=2,47)
.SET L2(I)=L2(I-1)*2
.F
.IF LPORT .EQ. 0
.MA IAND=AND
.MA IOR=OR
.MA JLM2(K)=AND(K,JLM2)
.SET JLM2=777700000000000000B
.MA ISHIFT(K1,K2)=SHIFT(K1,K2)
.ENDIF
.F
.SET TYPE COUNTER INCR
.SET LIMPOS =-9999
.SET LNCPW =10
.SET LRBITS =11520
.SET INCR=0
.F PERSERVE THE ORDER OF LCAS THROUGH LPAT FOR quick DD80 RECOGNITION
.SET LCAS =INCR
.SET LINT =INCR
.SET LOR =INCR
.SET LSIZE =INCR
.SET LFONT =INCR
.SET LPAT =INCR
.SET LSSIZ =INCR
.SET LCENT =INCR
.SET LCOLR =INCR
. IF LN CPW. LE 2
. SET LDUMY =INCR
.ENDIF
. SET LOPNUM =INCR-1
. SET LUSROP =10
. SET LMAXOP =LOPNUM+LUSROP
F PLOTIT MUST BE CHANGED IF LVECT IS CHANGED.
. SET LVECT =1
. SET LFRST =1-LVECT
. SET LBUFFS =10
. SET LSBUFF =LBUFFS+2
. SET LOPST =L2(5)
. SET LOPEND =L2(6)-1
. SET LADJST =L2(7)+L2(6)
F
. TOP
. IF LCODE(MC).EQ.0 .OR. LPACK.EQ.1
. SET INCR=L2(5)+L2(6)
. SET LPWRIT =INCR
. SET LFRAME =INCR
. SET LSETOP =INCR
. SET LGRID =INCR
. SET LTICK =INCR
. SET LLABM =INCR
. SET LRANG =INCR
. SET LPTS =INCR
. SET LREL =INCR
. OTHERWISE
. SET LPWRIT =3
. SET LFRAME =5
. SET LPSYM =4
. SET LPTS = 6
. SET LREL = 9
F DASHLN =LMASK
. SET LNOPM =5
.ENDIF
F
. SET LUDD80 ="6774B"
SETS AND MACROS
DEFINED (Con't)

.SET LNMASK =7
.IF LCODE(MC) .EQ. 0
.SET LPAIR =15
.SET LOUTWD =4
.SET LNBPW ="(16*LPAIR-1)/LOUTWD + 1"
.SET LSYBUF ="(LRBITS-1)/LNBPW + 1"
.OTHERWISE
.SET LPAIR =5
.SET LOUTWD =3
.SET LPOINT =LFRST+LVECT+1
.SET LSYBUF =256
.ENDIF
.SET LUMTCD =8

.SET LFMMC =10
.SET LFMTW =LWDS(LFMMC)
.SET LLAB =20
.SET LLOW=127
.SET LHIGH=204

.SET LSCR =L2(15)-1
.SET LSCRD =10
.SET LDEC =LSCR/50
.SET LMGR =LSCR/85
.SET LMNR =LSCR/128
.IF LCODE(DD).EQ.0
.MACRO PREOUT=Q8QWRB(MBUFAMBIJFLUMUNTT)
.ENDIF
.IF LPORT .EQ. 0
.MACRO TRANS =Q8QTRN
.MACRO SPLTDA=Q8QSDA
.MACRO PUT42 =Q8QP42
.MACRO FLUSHB=Q8QFLB
.MACRO WRITEB=Q8QWRB
.MACRO SYSPLT=Q8QPP.C
.MACRO JISTFY=Q8QJUS
.ENDIF

COMMENTS ABOUT CODES

CONDITIONAL COMPILATION CODES

A  ARGUMENT COMMENTS FOR USERS - ELIMINATE OR DEACTIVATE ONLY
AS  ASCII
C  FORTRAN COMMENTS
CA  CALCOMP CODE CREATION
D  COMMENTS FOR DD80 CODE CREATION
DD  DD80 CODE CREATION
EB  EBCDIC
EC  EIGHT BITS PER CHARACTER ON COMPUTER
EE  ERROR MESSAGES EXTENSIVE
ES  ERROR MESSAGES IN SHORT MODE
EX  EXAMINING METACODE (LPACK=6)
F  FRED-TIME COMMENT CARDS - ELIMINATE ONLY
HO  HOLLERITH INFORMATION FORMED FROM ASCII (USED IN TRANSLATOR ONLY)
I  COMMMENTS FOR II
II  INCREMENT INSTRUCTION CREATION
J  COMMENTS FOR SB
LV  LOCAL VARIABLES USED WHENEVER POSSIBLE TO REDUCE SETUP TIME
M  COMMENTS FOR METACODE CREATION
MC  METACODE CREATION
N  COMMENTS FOR NC
NC  NON-COMPACTING STATEMENTS FOR WHEN UC IS TURNED OFF
NP  NONPORTABLE CODE TO BE REPLACED BY PC CODE
OV  OVERFLOW TO DISK WHEN FLASH BUFFER FILLS
PC  PORTABILITY CODE
PR  PRINTER USED AS PLOTTER (USED IN TRANSLATOR ONLY — LPACK .NE. 0)
RA  MULTIBYTE RANGE INSTRUCTION CREATION
RF  REDUNDANT FRSTPT ELIMINATION
RO  REDUNDANT OPTION CODE ELIMINATION
RV  REDUNDANT (ZERO LENGTH) VECTOR ELIMINATION
SB  SIX BITS PER CHARACTER ON COMPUTER
SE  CODE TO PICK SB OR EC
TR  TRACE DEBUGGING CODE
U  COMMENTS FOR UC
UC  ULTRA-COMPACT OUTPUT CODE (FOR METACODE ONLY)
W  WARNING COMMENTS FOR IMPLEMENTORS - ELIMINATE OR DEACTIVATE ONLY

THESE CODES ARE ACTIVATED, DEACTIVATED, OR ELIMINATED ACCORDING TO
FLAGS SET IN THE MACRO DEFINITIONS. TO TUNE THIS PACKAGE TO A PARA-
TICULAR USE, OVERRIDE THAT SETTING WHERE MARKED BELOW.

8-9
<table>
<thead>
<tr>
<th>GLOBALS</th>
<th>PURPOSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>BYTMOV</td>
<td>Moves NCHAR characters from ICHAR to the instruction stream. If configured for a 6-bit machine and metacode is being produced, the characters are translated to 8-bit characters. If dd80 code is being produced, flag characters are resolved.</td>
</tr>
<tr>
<td>CHRTRN</td>
<td>Translation tables for various character sets. used in block data.</td>
</tr>
<tr>
<td>CCOM</td>
<td>Comments explaining the variables in the common block.</td>
</tr>
<tr>
<td>COMMON</td>
<td>The common block.</td>
</tr>
<tr>
<td>DRAWAL</td>
<td>Used in GRIDAL to draw axis lines.</td>
</tr>
<tr>
<td>DRAWLT</td>
<td>Used in GRIDAL to draw lines and ticks.</td>
</tr>
<tr>
<td>ERROR</td>
<td>Outputs error messages and causes termination.</td>
</tr>
<tr>
<td>FINDN</td>
<td>Finds a name in the MNAME table. Used in OPTN and GETOPT.</td>
</tr>
<tr>
<td>INITMB</td>
<td>Initializes MBPAIRS with MPAIR1 and MPAIR2; sets MPAIR to 2. Metacode packages only.</td>
</tr>
<tr>
<td>IMP</td>
<td>Marks places in code where implementors MUST change the code. Used in Block Data.</td>
</tr>
<tr>
<td>LOCSUB</td>
<td>Used to cause a local variable to be used to hold an argument to a subroutine.</td>
</tr>
<tr>
<td>PUT</td>
<td>Causes an instruction with OP CODE LCODE to be formed using MX and MY.</td>
</tr>
<tr>
<td>PUTPAIR</td>
<td>Puts a byte pair in the holding buffer and has the holding buffer dumped when it gets full.</td>
</tr>
<tr>
<td>TRN</td>
<td>Causes X and Y (which may be integer or floating) to be translated to plotter address units and put in MX and MY.</td>
</tr>
<tr>
<td>TRNPUT</td>
<td>TRN and PUT at once.</td>
</tr>
<tr>
<td>TRXY</td>
<td>Causes an X or Y value represented by Z (which equals 1HX or 1HY) to be translated from integer or floating point space to plotter address units and stored.</td>
</tr>
</tbody>
</table>

8-10
WARN  Outputs warnings.
LIST OF PLOTS

Contained on the following pages are samples of pictures generated by the Graphics Utilities, using the demonstration drivers.

AUTOGRAPH
CONRAN
CONRAQ
CONRAS
CONREC
CONRECQCK
CONRECSMTH
CONRECSUPR
DASHCHAR
DASHLINE
DASHSMTH
DASHSUPR
HAFTON
ISOSRF
ISOSRFHR
PWRITX
PWRITY
PWRZI
PWRZS
PWRZT
SCROLL
SRFACE
STRMLN
SUPMAP
THREED
VELVCT
WINDOW
SAMPLE PLOTS: AUTOGRAPH

DEMONSTRATING EZXY ENTRY IN AUTOGRAPH

DEMONSTRATING EZY ENTRY OF AUTOGRAPH

DEMONSTRATING EYX ENTRY OF AUTOGRAPH

DEMONSTRATING EZMY ENTRY OF AUTOGRAPH
SAMPLE PLOTS: CONRAN, CONRAQ, CONRAS
SAMPLE PLOTS: CONREC, CONRECQCK
SAMPLE PLOTS: DASHCHAR, DASHLINE, DASHSMTH, DASHSUPR

DEMONSTRATION PLOT FOR DASHCHAR
IPAT=DAADAOADAOOADAU. K 1
- -K -i - - -K Il- - ----
11 .1Y

IPAT=DDODOAAAAADDDDOD. K 2
- --

IPAT=DDODOAAAAADDDDOD. K 3
- --*

IPAT=DDODOAAAAADDDDOD. K 4
- --*

IPAT=DDODOAAAAADDDDOD. K 5
- --*

IN IPAT STRINGS, A AND S SHOULD BE INTERPRETED AS APOSTROPHE AND DOLLAR SIGN

DEMONSTRATION PLOT FOR DASHLINE
IPAT=0001100011111111

IPAT=111110011111100

IPAT=0001110001111111

IPAT=1111110011111100

IPAT=0001110001111111

IPAT=1111110011111100

IN IPAT STRINGS, A AND D SHOULD BE INTERPRETED AS APOSTROPHE AND DOLLAR SIGN
SAMPLE PLOTS: ISOSRF and ISOSRFHR
SAMPLE PLOTS: SCROLL

Demonstration Driver

Test of SCROLL

Demonstration Driver

Test of SCROLL
SAMPLE PLOTS: SURFACE

DEMONSTRATION PLOT FOR SURFACE ENTRY OF SURFACE

DEMONSTRATION PLOT FOR EZSURF ENTRY OF SURFACE
SAMPLE PLOTS: STMLN and THREED

DEMONSTRATION PLOT FOR ROUTINE STMLN

DEMONSTRATION PLOT FOR ROUTINE THREED
SAMPLE PLOTS: SUPMAP

SUPMAP DEMONSTRATION: STEREODRAPHIC PROJECTION

SUPMAP DEMONSTRATION: ORTHOGRAPHIC PROJECTION

SUPMAP DEMONSTRATION: LAMBERT EQUAL AREA PROJECTION

SUPMAP DEMONSTRATION: LAMBERT CONFORMAL CONIC PROJECTION
SAMPLE PLOTS: SUPMAP (Con't)

SUPMAP DEMONSTRATION: AZIMUTHAL EQUIDISTANT PROJECTION

SUPMAP DEMONSTRATION: MOLLWEIDE TYPE PROJECTION

SUPMAP DEMONSTRATION: MERCATOR PROJECTION

SUPMAP DEMONSTRATION: CYLINDRICAL EQUIDISTANT PROJECTION
SAMPLE PLOTS: VELVCT and WINDOW

DEMONSTRATION PLOT FOR ENTRY EZVEC OF VELVCT

DEMONSTRATION PLOT FOR ENTRY VELVCT OF VELVCT

DEMONSTRATION PLOT FOR WINDOW

7 4 1
8 2
9 6 3
INDEX

Abnormally terminated job, 5-2
ACQUISITION, 2-1
Added,
   new plotter, 4-1
Added flexibility, 1-2
Architecture,
   machine, 6-1
Argument lists, 8-1
Arguments,
   routine, 6-1
Axis drawing, 4-2
Base-B form, 3-4
Bit manipulation, 1-2
Block,
   named common, 6-1
BLOCK DATA routine, 2-1
Boundaries,
   record, 5-1
Character drawing, 1-1
Characters,
   hardware, 4-1
Code,
   matching, 2-1
tailored, 2-1
Code compaction,
   output, 6-1
CODE DEFINITIONS,
   OP, 4-4
Codes,
   user-defined option, 4-1
CODÉS,
   OP, 4-5
Common block,
   named, 6-1
Compaction,
   output code, 6-1
Computer,
   target, 2-1
Conditional structures, 6-1
Configuration,
   target machine, 1-2
Constants,
   machine, 3-3
Constructs,
   metacode, 4-1
Control,
   pen, 4-1
Core requirements, 1-2
Cracker,
   format, 3-1
Current marker, 4-5
Data records, 5-1
DATA routine,
   BLOCK, 2-1
Data type identifier, 5-1
DECK STRUCTURE, 7-1
Deck structure, 7-1
DEFINITIONS,
   OP CODE, 4-4
Device-independent, 1-1
Device-independent instructions, 2-1, 4-1
Double precision, 3-4
Drawing,
   character, 1-1
   line, 1-1
   axis, 4-2
Drum plotters, 4-5
Dumps,
   hexadecimal, 2-1
   octal, 2-1
EFFICIENCY,
   PAPER, 4-5
ENCODE,
   Subroutine, 3-1
ENHANCEMENTS, 1-2
Error,
   fatal, 3-2
Examples, 7-2
   .FALSE., 3-3
Fatal error, 3-2
Flexibility,
   added, 1-2
Floating point numbers, 3-4
Fonts, 4-1
Form,
   software request, 2-1
   base-B, 3-4
Format,
   metafile, 3-1
Format cracker, 3-1
FORMATION, 1-1
FORMATION, 6-1
Four-byte positioning instruction, 4-1
Frame, 5-2
FRED, 1-1
FRED-time variables, 8-1
Function IMACH, 3-3
FUNCTION IAND, 3-3
FUNCTION INTT (X), 3-3
Function IOR, 3-3
Function ISHIFT, 3-2
FUNCTION LOC (X), 3-2
Function R1MACH, 3-4
GETCHR,
Subroutine, 3-2
Graphics,
low-level, 1-1
Graphics interface,
portable, 1-1
GRAPHICS METAFILE STRUCTURE,
NCAR'S, 5-1
Hardware, 2-1
Hardware characters, 4-1
Hexadecimal dumps, 2-1
Host machine, 4-5
I1MACH,
Function, 3-3
IAND,
Function, 3-3
Identifier,
data type, 5-1
Implementation, 2-1
IMPLEMENTATIONS,
ROUTINE, 3-5
IMPLEMENTATIONS (Con't),
ROUTINE, 3-6
IMPLEMENTATIONS (Cont),
ROUTINE, 3-7
Increment instruction,
two-byte, 4-1
Increment instruction, 4-2
Instruction,
four-byte positioning, 4-1
two-byte increment, 4-1
increment, 4-2
range, 5-1
Instruction set, 1-1
Instructions,
device-independent, 2-1, 4-1
INSTRUCTIONS,
METACODE, 4-1
Instructions,
multibyte, 4-1
no-op, 5-1
INSTRUCTIONS (Con't),
METACODE, 4-3
Interface,
portable graphics, 1-1
Interfaced points, 4-1
INTRODUCTION, 5-1
INTRODUCTION, 7-1
INTRODUCTION TO METACODE, 4-1
INTT (X),
FUNCTION, 3-3
I/O, 1-2
IOR,
Function, 3-3
ISHIFT,
Function, 3-2
Job,
abnormally terminated, 5-2
Line drawing, 1-1
Lists,
argument, 8-1
LOC (X),
FUNCTION, 3-2
Looping, 6-1
Low-level graphics, 1-1
Machine,
host, 4-5
Machine architecture, 6-1
Machine configuration,
target, 1-2
Machine constants, 3-3
Manipulation,
bit, 1-2
Manipulator,
metacode, 5-1
Marker,
current, 4-5
Master system plot package, 2-1
MASTER SYSTEM PLOT PACKAGE,
UNDERSTANDING THE, 8-1
Matching code, 2-1
Metacode, 4-1
METACODE,
INTRODUCTION TO, 4-1
Metacode constructs, 4-1
METACODE INSTRUCTIONS, 4-1
METACODE INSTRUCTIONS (Con't), 4-3
Metacode manipulator, 5-1
Metacode translation, 4-1
Metafile format, 3-1
METAFILE STRUCTURE, 5-1
METAFILE STRUCTURE,
NCAR'S GRAPHICS, 5-1
METAFILE STRUCTURE,
OLD vs. NEW, 5-2
MODS, 7-1
Multibyte instructions, 4-1
Named common block, 6-1
NCAR system-dependent, 3-4
NCAR'S GRAPHICS METAFILE STRUCTURE, 5-1
Needed routines, 3-1
NEW METAFILE STRUCTURE, 5-2
OLD vs., 5-2
New plotter added, 4-1
No-op instructions, 5-1
Numbers,
  floating point, 3-4
Octal dumps, 2-1
Old structure, 5-2
OLD vs. NEW METAFILE STRUCTURE, 5-2
OP CODE DEFINITIONS, 4-4
OP CODES, 4-5
Option codes,
  user-defined, 4-1
Options,
  setting, 4-4
Output code compaction, 6-1
OVERVIEW, 1-1
Package,
  master system plot, 2-1
PACKUM,
  Subroutine, 3-2
PAPER EFFICIENCY, 4-5
Paper waste, 4-5
PARAMETERS, 8-1
Pen control, 4-1
PERROR,
  SUBROUTINE, 3-2
PERROR,
  Subroutine, 3-2
PHILOSOPHY,
  SYSTEM PLOT PACKAGE, 1-1
PICTURE,
  SAMPLE, 2-2
PICTURE,
  SAMPLE, 2-3
Picture size, 4-1
Plot package,
  master system, 2-1
PLOT PACKAGE,
  UNDERSTANDING THE MASTER SYSTEM, 8-1
PLOT PACKAGE PHILOSOPHY,
  SYSTEM, 1-1
PLOTS,
  SAMPLE, 9-1
Plotter added,
  new, 4-1
Plotters,
  drum, 4-5
Point numbers,
  floating, 3-4
Points,
  interfaced, 4-1
Portability problems, 5-2
Portable graphics interface, 1-1
Portable test program, 1-2
PORTABLE TRANSLATION, 4-1
Positioning instruction,
  four-byte, 4-1
Postprocessors, 4-2
Precision,
  double, 3-4
Precision,
  single, 3-4
PREPROCESSING, 8-1
Preprocessor, 1-1
Problems,
  portability, 5-2
  portable test, 1-2
  test, 2-1
PURPOSE, 6-1
R1MACH,
  Function, 3-4
Range instruction, 5-1
Record boundaries, 5-1
Records,
  data, 5-1
Request form,
  software, 2-1
Requirements,
  core, 1-2
Resolution, 4-1
Right-justified, 3-2
Routine,
  BLOCK DATA, 2-1
Routine arguments, 6-1
ROUTINE IMPLEMENTATIONS, 3-5
ROUTINE IMPLEMENTATIONS (Con't), 3-6
ROUTINE IMPLEMENTATIONS (Cont), 3-7
Routines,
  support, 2-1
Routines,
  needed, 3-1
SAMPLE PICTURE, 2-2
SAMPLE PICTURE, 2-3
SAMPLE PLOTS, 9-1
Scaling, 1-1
Scaling, 4-2
Set,
  instruction, 1-1
SETCHR,
  Subroutine, 3-2
Setting options, 4-4
Single precision, 4-4
Size,
  picture, 4-1
Sizes, various, 4-1
Software, transporting, 1-2
Software request form, 2-1
STRUCTURE, METAFILE, 5-1
STRUCTURE, NCAR'S GRAPHICS METAFILE, 5-1
Structure, old, 5-2
STRUCTURE, OLD vs. NEW METAFILE, 5-2
STRUCTURE, DECK, 7-1
Structure, deck, 7-1
Structures, conditional, 6-1
Subroutine ENCODE, 3-1
Subroutine GETCHR, 3-2
Subroutine PACKUM, 3-2
SUBROUTINE PERROR, 3-2
Subroutine PERROR, 3-2
Subroutine SETCHR, 3-2
Subroutine ULIBER, 3-1
Subroutine WRITEB, 3-1
SUPPORT, 2-1
Support routines, 2-1
System plot package, master, 2-1
SYSTEM PLOT PACKAGE, UNDERSTANDING THE MASTER, 8-1
SYSTEM PLOT PACKAGE PHILOSOPHY, 1-1
System-dependent, NCAR, 3-4
Tailored code, 2-1
Target computer, 2-1
target machine configuration, 1-2
T-digit, 3-4
Terminated job, abnormally, 5-2
Test program, portable, 1-2
Test program, 2-1
TESTING, 2-1
THE MASTER SYSTEM PLOT PACKAGE, UNDERSTANDING, 8-1
Translation, metacode, 4-1
TRANSLATION, PORTABLE, 4-1
Translators, 1-2
TRANSPORTATION, 1-2
Transporting software, 1-2
.TURE., 3-3
Two-byte increment instruction, 4-1
Type identifier, data, 5-1
ULIBER, Subroutine, 3-1
User-defined option codes, 4-1
Variables, FRED-time, 8-1
various sizes, 4-1
Waste, paper, 4-5
WRITEB, Subroutine, 3-1
(X), FUNCTION LOC, 3-2
(X), FUNCTION INTT, 3-3
Zero-filled, 3-2