MPC REPORT NO. 92-9

ENHANCEMENT OF EXISTING ENGINEERING SOFTWARE

VOLUME NO. 2

UWGRAPH Computer-Aided Design and Drafting Library
Version 1.0

Developed under the sponsorship
of the
Wyoming Highway Department
and
Mountain-Plains Consortium

Chad Clancy
J.A. Puckett

Department of Civil Engineering
University of Wyoming
Laramie, Wyoming 82071

July 1992

Technical Report Documentation Page

1. Report No. MPC 92-9 Volume 2	2. Government Accession No.	3. Recipient's Catalog No.
4. Title and Subtitle Enhancement of Existing Engineering Software UWGRAPH Computer-Aided Design and Drafting Library - Version 1.0		5. Report Date July 1992
		6. Performing Organization Code
7. Author(s) J.A. Puckett and Chad Clancy		8. Performing Organization Report No.
9. Performing Organization Name and Address Department of Civil Engineering University of Wyoming Laramie, WY		10. Work Unit No. (TRAIS)
		11. Contract or Grant No.
12. Sponsoring Agency Name and Address Mountain-Plains Consortium North Dakota State University Fargo, ND		13. Type of Report and Period Covered Project Technical Report
		14. Sponsoring Agency Code

15. Supplementary Notes

Supported by a grant from the U.S. Department of Transportation, University Transportation Centers Program

16. Abstract

This documentation includes detailed information on each subroutine used to draw the entities supported by UWGRAPH. These descriptions include the definitions of the arguments, an example of the subroutine usage, and a graphical illustration of the arguments. Compilation instructions are included for using UWGRAPH with Unix and Microsoft Fortran compilers. A short section follows with programming hints. This section outlines the errors which are most common and likely to be made by new program developers. Several appendices are included which outline architecture, utilities, include files, and data files used by UWGRAPH, and finally a paper which overviews the UWGRAPH library is given.

17. Key Words computer-aided, UWGRAPH, software, engineering, design	18. Distribution Statement		
19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages 114	22. Price

ACKNOWLEDGEMENT

This report has been prepared with funds provided by the United States Department of Transportation to the Mountain-Plains Consortium (MPC). The MPC member universities include North Dakota State University, Colorado State University, University of Wyoming and Utah State University.

PREFACE

The development of graphics libraries have been undertaken by numerous groups ranging from computer manufacturers, software developers, and user groups, but to no one library addresses the needs of an application developer working in the area of civil engineering. Specifically, libraries are available for placing lines on a screen or other device, or perhaps viewing a model by translation/rotation/zoom. But such libraries are severely limited in their ability to easily produce high level entities such as dimension, notes, etc. Such entities are the foundation of civil engineering drafting. Furthermore, most libraries produce graphics to output devices, not directly to a CAD system. UWGRAPH was written to address both of these limitations. The library has the entities commonly used in drafting and will produce graphics to the screen and to drafting systems for editing and inclusion in production drawings.

UWGRAPH does not re-invent the wheel but rather reshapes it into a product of greater utility. The authors are sincerely hopeful that the library will be widely used to develop applications in civil engineering and hope that developers will share their products with the profession at-large. For only through a concerted effort will development efforts be most economical. There are so many possibilities for applications using UWGRAPH, but only through sharing the fruits of our efforts will we fully enter the realm of CADD (with two D's).

J.A. Puckett

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

PAGE

INTRODUCTION

Often a traditional procedure is used where a designer sketches results which are based on computer applications and transmits this information via hardcopy to a drafter, who reenters it into a CAD system. The inefficiencies of this are clear and an obvious opportunity exists for productive gain by linking design applications directly with CAD. Recent work by the Wyoming Highway Department has shown that by doing this, productivity ratios can exceed 20:1. UWGRAPH is a FORTRAN subroutine library to facilitate this linkage.

UWGRAPH is used in conjunction with an application which performs engineering calculations and/or drawing parameterization. The flexibility and familiarity of FORTRAN is combined with the tools necessary to produce graphics files automatically. The library is based on the graphics entities required for structural drafting. Many existing graphic systems are difficult to learn and implement. UWGRAPH simplifies and unifies the subroutine calls of three commonly used systems (DXF, Micro-CSL, and GKS) by combining low-level calls to produce graphical entities useful in engineering drawing, e.g., dimensions, notes, etc. There has been much effort to standardize computer graphics, but creating an all-encompassing graphics format that suits every application, type of drawing, and computer system is a difficult, if not impossible task. Hence, the UWGRAPH scope is directed to structural engineering but easily spans other disciplines.

The UWGRAPH library links to three graphic libraries commonly used in engineering. The purpose of the library is to allow the use of DXF, Micro-CSL and GKS libraries through one application program. Graphics can be created in one or more of these formats from a single set of subroutine calls. The graphical output of the application program can then be used by anyone who can display graphics in any one of the systems. The application programmer is relieved of the burden of learning more than one graphics system because the initialization, drawing definition, and termination is all consistently defined by the UWGRAPH library. Further, the application developer can use GKS to review drawings in development without accessing the CAD system.

UWGRAPH's first link is to AutoCad's Data Exchange Format (DXF), a standard commonly used for exchange of CADD information in both PC and workstation environments. The ASCII DXF file created by an application using UWGRAPH can be translated into the native format of many CADD systems.

The second link is to a graphics format supported by Intergraph using the MicroStation Customer Support Library (Micro-CSL). Micro-CSL is a workstation-based product and used with the Intergraph CADD software. A FORTRAN binding allows an interface to the capabilities of Micro-CSL. The output from this link to the library is an Intergraph design file which can be edited with Intergraph's MicroStation.

The third link is to the Graphical Kernal System (GKS) which was accepted by the *International Organization for Standardization* as a two-dimensional graphics standard in 1985. GKS has FORTRAN bindings for numerous hardware/software platforms including most all engineering workstations and DOS-based computers. GKS is used by some CAD systems for graphics display. GKS is different from the other graphics systems used by UWGRAPH because the output is sent directly to the screen as opposed to an output file. After the output is displayed on the screen, a metafile can be created and sent to a plotted used in conjunction with a metafile interpreter.

The UWGRAPH library supports the basic entities needed to create and dimension a drawing. These entities include lines, arcs, multiple point lines, labels, arrowheads and several types of dimensions. In creating these entities UWGRAPH does not fully use the capabilities of any one graphical system, but in general uses the capabilities common to all three systems. The result is a practical graphics link between an application and the three graphics libraries. This saves time and simplifies development.

The documentation which follows includes detailed information on each subroutine used to draw the entities supported by UWGRAPH. These descriptions include the definitions of the arguments, an example of the subroutine usage, and a graphical illustration of the arguments. Compilation instructions are included for using UWGRAPH with Unix and Microsoft Fortran compilers. A short section follows with programming hints. This section outlines the errors which are most common and likely to be made by new program developers. Several appendices are included which outline architecture, utilities, include files, and data files used by UWGRAPH, and finally a paper which overviews the UWGRAPH library is given.

USING UWGRAPH CALLS

Documentation Conventions

In the documentation of the **UWGRAPH** library, some conventional notation in the subroutine argument list and descriptions has been used:

1. **Bold letters** indicate a **DXF** dependency.
2. *Italic letters* indicate a *Micro-CSL* dependency.
3. Underlined letters indicate a **GKS** dependency.
4. Plain letters indicate that the subroutine argument is used by all three systems in the same manner.

A dependency means that the argument is not used by all of the systems or is used in a different way. For example, the argument XWMIN in the UWINIT call is only used by **GKS** and is ignored by the other graphic system calls. These *highlighted* arguments are system specific and can contain any value if the argument is not required for the system being used, but **MUST** be properly declared and passed in the subroutine calls. A standard include file called VARI.FOR contains all of the proper variable declarations required by an application using **UWGRAPH.** This file can be included at the beginning of the application program to ensure that the variables being passed are of the proper type.

Required Sections

A graphics file is created in three sections: Initialization, Entity Creation and Termination. In the initialization section, the necessary files are opened, defaults are read, windows are set, etc. There is only one call in the initialization section. In the entity creation section, the entities are written to a graphics file (or to the screen). One call is required for each entity created so there may be many calls in this section. The termination section closes all of the files, writes termination information to the output file and in the case of **GKS** allows some output manipulation. See the example code which follows and example applications in Appendix B for details.

Each subroutine call is documented by defining the routine's argument list. Here each variable is typed and dimensioned. Next an example which illustrates its usage is given. These examples include the code necessary to assign the arguments and issue the call. The initialization and termination process is common to all examples and is not repeated. The program in Table 1 illustrates the program structure and the initialization and termination requirements. Where indicated in the example code outlines, each subroutine call may be inserted, in any order. Note that the example below is written for an Intergraph workstation with **GKS**. Other installations will be required to change the workstation type parameter as noted in the comment lines.

Table 1. Example Program

```
      PROGRAM EXAMPLE
C
C     VARI.FOR types all variables used UWGRAPH argument lists.
C     Include this file in every subroutine with UWGRAPH calls.
      INCLUDE 'VARI.FOR'
C
C     Now, open up a file to output messages.
C
      OUTNUM = 3
      OPEN(UNIT=OUTNUM,FILE='MESSAGE.OUT',STATUS='UNKNOWN')
C
C     Set up the initialization parameters.
C
      OUTNAME = 'example'
      XWMIN   = 0
      XWMAX   = 90
      YWMIN   = 0
      YWMAX   = 90
      WKSID   = 1
      WSCON   = 1
C
C     The workstation type is system dependent, see your GKS
C     documentation for details.  The system is set to GKS below.
C     Note the system dependent code which is commented out in a
C     format which allows preprocessing.
C
      SYS     = 3
CI_G      WSTYP   = 12
CA_G      WSTYP   = 9701
      UNIT    = 'FT'
      IDBG0   = 1
C
C     Issue initialization call.
C      CALL UWINIT(INPUTDIR,OUTNAME,XWMIN,XWMAX,YWMIN,YWMAX,
     1     WKSID,WSCON,WSTYP,UNIT,SYS,IDBG0)
C
C     Initialize the default flags with this call.
C
      CALL USEDEF(ITYP1,ITYP2,ITYP3,ITYP4,ITYP5,ITYP6,ITYP7,ITYP9,
     1   ILEV1,ILEV2,ILEV3,ILEV4,ILEV5,ILEV6,ILEV7,ILEV9,
     2   ICOL1,ICOL2,ICOL3,ICOL4,ICOL5,ICOL6,ICOL7,ICOL9,
     3   WID1,WID2,WID3,WID4,WID5,WID6,WID7,WID9,
     4   SL4,SL5,SL6,SL7,
     5   ROT4,ROT5,ROT6,ROT7,
     6   SW4,SW5,SW6,SW7,
     7   HT4,HT5,HT6,HT7,
     8   IFONT)
C
C     Now, any graphical entity may be used.
C     You may insert the example code documented in each
C     subroutine here.
C     For example, draw a multiple point line.
C
      POINTS=7
      X(1)=20
      Y(1)=40
      X(2)=50
      Y(2)=40
      X(3)=50
      Y(3)=50
      X(4)=40
      Y(4)=50
      X(5)=40
      Y(5)=60
      X(6)=20
      Y(6)=60
      X(7)=X(1)
      Y(7)=Y(1)
      CALL UWMPLINE(X,Y,POINTS,ITYP3,WID3,ILEV3,ICOL3,SYS,IDBG3)
C
C     Issue termination call to close graphical system.
      CALL UWTERM(SYS,IDBG8)
C
```

```
CLOSE  (OUTNUM)
STOP
END
```

Subroutine Descriptions

Following each UWGRAPH subroutine the arguments are defined, typed and dimensioned. Then a short example is given along the graphic which was produced by the code. Last, the arguments are described graphically. The structure for each UWGRAPH subroutine documentation is consistent, except for the initialization and termination routines which perform no graphical drawing function.

INITIALIZATION - Initializes the appropriate graphics libraries for subsequent entity calls.

CALL UWINIT (INPUTDIR, OUTNAME, XWMIN, XWMAX, YWMIN, YWMAX, WKSID, WSCON, WSTYP, UNIT, SYS, IDBG0)

1. INPUTDIR (CHARACTER*35) : The directory name, located in the first line of the datafile BLOCK.DEF, is read and returned through this argument. The BLOCK.DEF datafile can be modified to contain the desired directory name. This directory can be used by the application program as the directory name for the input data files (use of this directory name is optional and the application is responsible for its use). See Appendix C for detail on BLOCK.DEF.

NOTE: This directory name will be echoed during the initialization procedure for approval or modification unless the first three characters of the directory name are 'xxx'. The 'xxx' characters must be removed by the application.

2. OUTNAME (CHARACTER*45) : This is the filename where the graphical data will be written. Do not use extensions. NOTE: The output directory name located in the second line of the datafile BLOCK.DEF will be placed at the beginning of this file name. The output directory name may be up to 35 characters long and the total number of characters in the output directory name and the output filename combined must be less than or equal to 45 characters.

3. XWMIN (REAL*4) : Minimum X coordinate for the output window in GKS.
* Used only by GKS

4. XWMAX (REAL*4) : Maximum X coordinate for the output window in GKS.
* Used only by GKS

5. YWMIN (REAL*4) : Minimum Y coordinate for the output window in GKS.
* Used only by GKS

6. YWMAX (REAL*4) : Maximum Y coordinate for the output window in GKS.
* Used only by GKS

7. WKSID (INTEGER*4) : Workstation identifier in GKS (similar to a file unit number in FORTRAN).
* Used only by GKS

8. WSCON (INTEGER*4) : Workstation connectivity in GKS (can be set to 1).

* Used only by **GKS**

9. WSTYP (INTEGER*4) : Workstation type number in **GKS** (for example WSTYP=12 for IG/**GKS** Window. Workstation - See **GKS** implementation specific Supplement to the GKS/C Reference Manual for applicable workstation type number).
* Used only by **GKS**

10. UNIT (CHARACTER*3) : String indicating the units being used.

a) 'IN' - Inches
b) 'MM' - Millimeters
c) 'FT' - Feet
d) 'MI' - Miles
e) 'M' - Meters
f) 'KM' - Kilometers
g) 'MIL' - Mils (i.e., 0.001 inch)
h) 'UM' - Microns
i) 'CM' - Centimeters
j) 'UIN' - Microinches

11. SYS (INTEGER*2) : System number for output.

1 = **DXF**
2 = **Micro-CSL**
3 = **GKS**

12. IDBG0 (INTEGER) : Debug flag for **UWGRAPH** initialization routine. If IDBG0 = 1 each argument in the initialization call will be written to the file debug.txt.

Table 2. Initialization Example

```
      PROGRAM EXAMPLE
      .
      .         (INITIALIZATION OF VARIABLES, ETC)
      .
      OUTNAME = 'testout'
      XWMIN = 0.0
      XWMAX = 190.0
      YWMIN = 0.0
      YWMAX = 190.0
      WKSID = 1
CI_G      WSTYP = 12
CA_G      WSTYP = 9701
      WSCON = 1
      UNIT = 'IN'
      SYS = 3
      IDBG0 = 1
      CALL UWINIT (INPUTDIR, OUTNAME, XWMIN, XWMAX, YWMIN,
     &  YWMAX, WKSID, WSCON, WSTYP, UNIT, SYS, IDBG0)
      .
      .
      .
C     ENTITY CALLS AND A TERMINATION CALL FOLLOW
```

LINE - A two point line is created.

CALL UWLINE (X1, Y1, X2, Y2, ITYP1, **WID1**, ILEV1, ICOL1, SYS, IDBG1)

1. X1 (REAL*8) : X coordinate of point 1.
2. Y1 (REAL*8) : Y coordinate of point 1.
3. X2 (REAL*8) : X coordinate of point 2.
4. Y2 (REAL*8) : Y coordinate of point 2.
5. ITYP1 (INTEGER*2) : Line type.

   1 = Solid
   2 = Dashed
   3 = Phantom
   4 = Centerline
   Negative integer = use default.

6. **WID1** (REAL*8) : Line width.

   Negative number = use default.
   ***N/A for DXF**

7. ILEV1 (INTEGER*2) : Viewing level of the line.

   Negative integer = use default.
   * N/A for **GKS**

8. ICOL1 (INTEGER*2) : Line color.

   1 = Black
   2 = Red
   3 = Green
   4 = Blue
   5 = Yellow
   6 = Magenta
   7 = Cyan
   8 = White
   Negative integer = use default

9. SYS (INTEGER*2) : System number for output.

   1 = **DXF**
   2 = **Micro-CSL**
   3 = **GKS**

10. IDBG1 (INTEGER) : Debug flag for **UWGRAPH** line routine. If IDBG1 = 1 each argument in the line call will be written to the file debug.txt.

Table 3. Line Entity Example

```
C       INITIALIZATION CALL
        .
        .
        .
        X1 = 10
        Y1 = 5
        X2 = 35
        Y2 = 10
        ITYP1 = 1
        WID1 = -1
        ILEV1 = -1
        ICOL1 = 4
        IDBG1 = 0
        CALL UWLINE (X1, Y1, X2, Y2, ITYP1, WID1, ILEV1, ICOL1,
     &  SYS, IDBG1)
        .
        .
        .
C       TERMINATION CALL
```

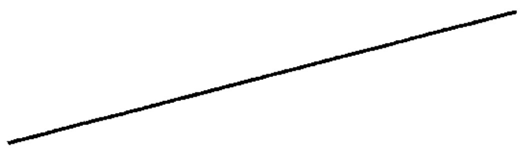


Figure 1.a

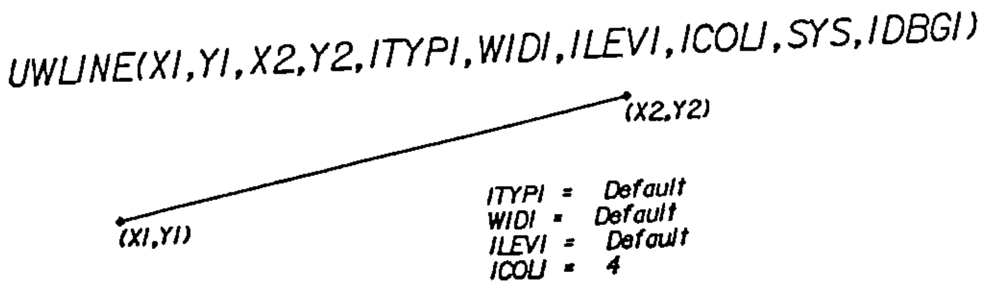


Figure 1.b

CIRCULAR ARC - An arc is drawn given a center, radius, start angle and an end angle.

CALL UWARC (XCENT, YCENT, RADIUS, DRAWDIR, STANGLE, EANGLE, ITYP2, **WID2**, ILEV2, ICOL2, SYS, IDBG2)

1. XCENT (REAL*8) : X coordinate of the center of the arc.
2. YCENT (REAL*8) : Y coordinate of the center of the arc.
3. RADIUS (REAL*8) : Radius of the arc.
4. DRAWDIR (INTEGER*2) : Draw direction from the start angle to the end angle.

   Negative integer = Counter clockwise

   Positive integer = Clockwise
5. STANGLE (REAL*8) : Start angle of the arc in degrees.
6. EANGLE (REAL*8) : End angle of the arc in degrees.
7. ITYP2 (INTEGER*2) : Arc line type.

   1 = Solid
   2 = Dashed
   3 = Phantom
   4 = Centerline
   Negative integer = use default.
8. **WID2** (REAL*8) : Arc line width.

   Negative number = use default.
   *** N/A for DXF**
9. ILEV2 (INTEGER*2) : Viewing level of the arc.

   Negative integer = use default.
   *** N/A for GKS**
10. ICOL2 (INTEGER*2) : Arc line color.

    1 = Black
    2 = Red
    3 = Green
    4 = Blue
    5 = Yellow
    6 = Magenta
    7 = Cyan
    8 = White
    Negative integer = use default

11. SYS (INTEGER*2) : System number for output.

1 = **DXF**
2 = **Micro-CSL**
3 = **GKS**

12. IDBG2 (INTEGER) : Debug flag for **UWGRAPH** arc routine. If IDBG2 = 1 each argument in the arc call will be written to the file debug.txt.

Table 4. Arc Entity Example

```
C       INITIALIZATION CALL
        .
        .
        .
        XCENT = 20.0
        YCENT = 28
        RADIUS = 10
        DRAWDIR = -1
        STANGLE = 90
        EANGLE = 180
        ITYP2 = 1
        WID2 = -1
        ILEV2 = 2
        ICOL2 = 4
        IDBG2 = 1
        CALL UWARC (XCENT, YCENT, RADIUS, DRAWDIR, STANGLE, EANGLE,
       &    ITYP2, WID2, ILEV2, ICOL2, SYS, IDBG2)
        .
        .
        .
C       TERMINATION CALL
```

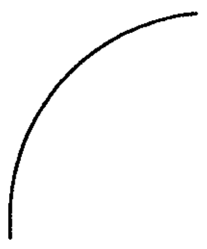


Figure 2.a

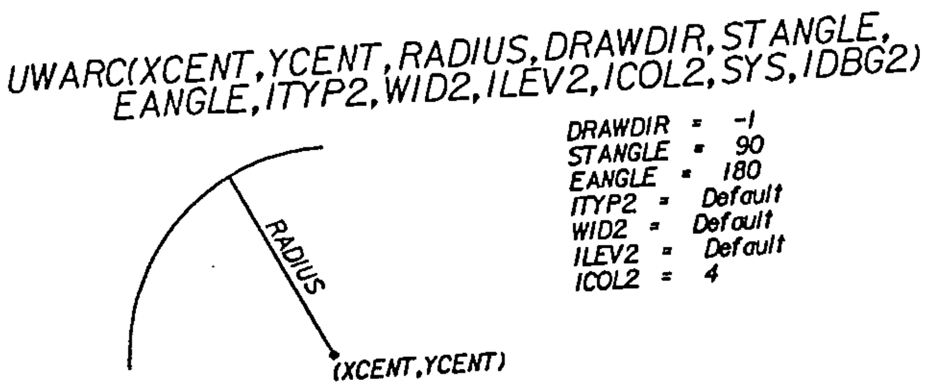


Figure 2.b

MULTIPLE POINT LINE - A line string is drawn given more than two points.

CALL UWMPLINE (X, Y, POINTS, ITYP3, **WID3**, ILEV3, ICOL3, SYS, IDBG3)

1. X(I) (REAL*8 (340)) : I'th X coordinate.

2. Y(I) (REAL*8 (340)) : I'th Y coordinate.

3. POINTS (INTEGER*2) : Number of points in the line string.

4. ITYP3 (INTEGER*2) : Line string type.

   1 = Solid
   2 = Dashed
   3 = Phantom
   4 = Centerline
   Negative integer = use default.

5. **WID3** (REAL*8) : Line string width.

   Negative number = use default.
   *** N/A for DXF**

6. ILEV3 (INTEGER*2) : Viewing level of the line string.

   Negative number = use default.
   * N/A for **GKS**

7. ICOL3 (INTEGER*2) : Line string color.

   1 = Black
   2 = Red
   3 = Green
   4 = Blue
   5 = Yellow
   6 = Magenta
   7 = Cyan
   8 = White
   Negative integer = use default

8. SYS (INTEGER*2) : System number for output.

   1 = **DXF**
   2 = **Micro-CSL**
   3 = **GKS**

9. IDBG3 (INTEGER) : Debug flag for **UWGRAPH** multiple point line routine. If IDBG3 = 1 each argument in the multiple point line call will be written to the file debug.txt.

Table 5. Multiple Point Line Entity Example

```
C     INITIALIZATION CALL
      .
      .
      .
      X(1) = 10
      Y(1) = 58
      X(2) = 19
      Y(2) = 51
      X(3) = 25
      Y(3) = 57
      X(4) = 30
      Y(4) = 51
      POINTS = 4
      ITYP3 = -1
      WID3 = -1
      ICOL3 = 4
      ILEV3 = 4
      IDBG3 = 0
      CALL UWMPLINE (X, Y, POINTS, ITYP3, WID3, ILEV3, ICOL3,
     &  SYS, IDBG3)
      .
      .
      .
C     TERMINATION CALL
```

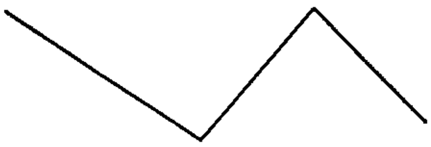


Figure 3.a

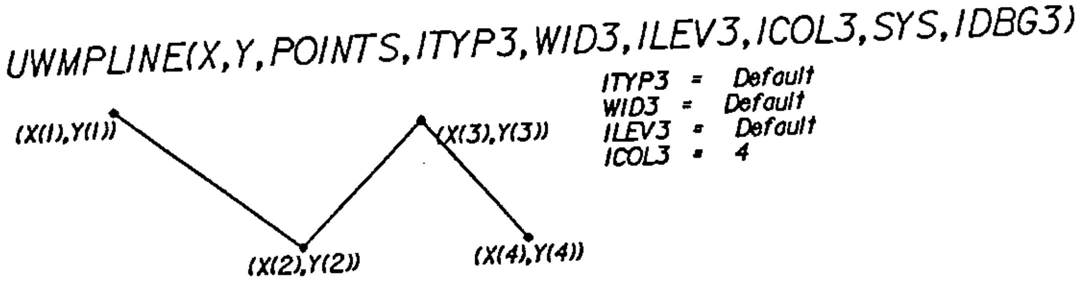


Figure 3.b

GENERAL NOTE - One or more strings of text are placed.

CALL UWNOTE (TCENX, TCENY, TOTCHR, NOSTR, TBUFFR, ITYP4, **WID4**, ILEV4, ICOL4, *SL4*, ROT4, HT4, *SW4*, JUST, **IFONT**, SYS, IDBG4)

1. TCENX(I) (REAL*8 (10)) : X coordinate of the I'th string of text.

2. TCENY(I) (REAL*8 (10)) : Y coordinate of the I'th string of text.

3. TOTCHR(I) (INTEGER*2 (10)) : Number of characters in the I'th string of text located in TBUFFR.

4. NOSTR (INTEGER*2) : Number of separate strings located in TBUFFR. Maximum of 10 strings.

5. TBUFFR (CHARACTER*72) : Text buffer containing combined text strings without spaces between the strings.

6. ITYP4 (INTEGER*2) : Line type for text.
   1 = Solid
   2 = Dashed
   3 = Phantom
   4 = Centerline
   Negative integer = use default.
   * *N/A for Micro-CSL*
   * N/A for **GKS**

7. **WID4** (REAL*8) : Line width for the note.

   Negative number = use default.
   * N/A for **GKS**
   * **N/A for DXF**

8. ILEV4 (INTEGER*2) : Viewing level of the note.

   Negative integer = use default.
   * N/A for **GKS**

9. ICOL4 (INTEGER*2) : Text color.

   1 = Black
   2 = Red
   3 = Green
   4 = Blue
   5 = Yellow
   6 = Magenta
   7 = Cyan
   8 = White
   Negative integer = use default

10. *SL4(I)* (REAL*8 (10)) : Character slant in degrees for the I'th text string (positive = CCW, negative = CW).

To use the default, set to -999.
* *N/A for Micro-CSL.*
* **GKS uses an italic font if SL4 > 0.0**

11. ROT4(I) (REAL*8 (10)) : I'th text string rotation angle in degrees (positive = CCW, negative = CW).

To use the default, set to -999.

12. HT4(I) (REAL*8 (10)) : Height of the I'th text string.

Negative number = use default

13. *SW4(I)* (REAL*8 (10)) : Width of the I'th string of text.

Negative number = use default.
* *N/A for Micro-CSL.*

14. JUST (INTEGER) : Text justification for the text strings (HORIZONTAL/VERTICAL).

1 =	LEFT/TOP
2 =	LEFT/CENTER
3 =	LEFT/BOTTOM
4 =	CENTER/TOP
5 =	CENTER/CENTER
6 =	CENTER/BOTTOM
7 =	RIGHT/TOP
8 =	RIGHT/CENTER
9 =	RIGHT/BOTTOM

15. **IFONT** (INTEGER) : Micro-CSL FONT NUMBER
* **N/A for DXF.**
* N/A for GKS.

16. SYS (INTEGER*2) : System number for output.

1 =	**DXF**
2 =	**Micro-CSL**
3 =	**GKS**

17. IDBG4 (INTEGER) : Debug flag for **UWGRAPH** general note routine. If IDBG4 = 1 each argument in the general note call will be written to the file debug.txt.

Table 6. General Note Entity Example

```
C       INITIALIZATION CALL
        .
        .
        .
        HT4(1) = 2.5
        HT4(2) = HT4(1)
        TCENX(1) = 14
        TCENY(1) = 70 + HT4(2) + 2
        TCENX(2) = 14
        TCENY(2) = 70
        NOSTR = 2
        TBUFFR = 'THIS IS A NOTEPlaced in two strings.'
        TOTCHR(1) = 14
        TOTCHR(2) = 22
        ITYP4 = 1
        WID4 = -1
        ICOL4 = 7
        ILEV4 = 5
        SL4(1) = 0.0
        SL4(2) = 0.0
        ROT4(1) = -999
        ROT4(2) = -999
        SW4(1) = 0.0
        SW4(2) = 0.0
        JUST = 3
        IDBG4 = 1
        CALL UWNOTE (TCENX, TCENY, TOTCHR, NOSTR, TBUFFR, ITYP4,
     &   WID4, ILEV4, ICOL4, SL4, ROT4, HT4, SW4, JUST, IFONT,
     &   SYS, IDBG4)
        .
        .
        .
C       TERMINATION CALL
```

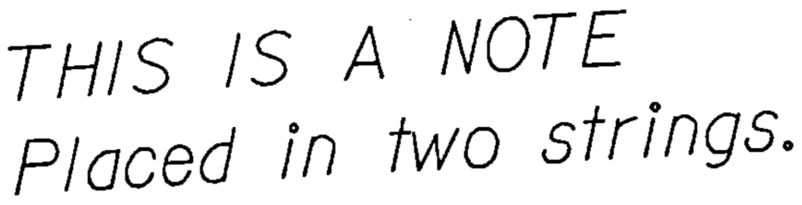


Figure 4.a

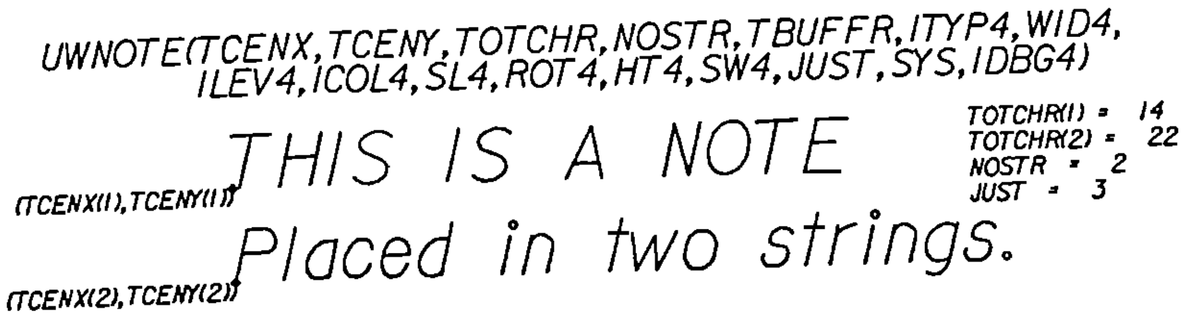


Figure 4.b

GENERAL DIMENSION CALL - A dimension with a break in the leader line for dimension text is drawn given two coordinates parallel to the object being dimensioned and the coordinate of the dimension's text. The value of the dimension text can be scaled and witness lines can be extended if necessary. NOTE: If there is not enough room for the dimension text between the witness lines the text will be placed outside the witness line on the XPT1,YPT1 side. If there is only room for the dimension text inside the witness lines (but not the leader and arrowheads) the dimension will be drawn with arrows out.

CALL UWDIM (DNUMX, DNUMY, XPT1, YPT1, XPT2, YPT2, ITYP5, WID5, ILEV5, ICOL5, *SL5*, ROT5, HT5, *SW5*, SCALE, DIMFLAG, EXT1, EXT2, SYS, IDBG5)

1. DNUMX (REAL*8) : X coordinate of the numeric part of the dimension.

2. DNUMY (REAL*8) : Y coordinate of the numeric part of the dimension.

3. XPT1 (REAL*8) : X coordinate of the first dimension point.

4. YPT1 (REAL*8) : Y coordinate of the first dimension point.

5. XPT2 (REAL*8) : X coordinate of the second dimension point.

6. YPT2 (REAL*8) : Y coordinate of the second dimension point.

7. ITYP5 (INTEGER*2) : Line type for the dimension.

   1 = Solid
   2 = Dashed
   3 = Phantom
   4 = Centerline
   Negative integer = use default.

8. **WID5** (REAL*8) : Line width for the dimension.

   Negative number = use default.
   ***N/A for GKS**
   ***N/A for DXF**

9. ILEV5 (INTEGER*2) : Viewing level of the dimension.

   Negative integer = use default.
   *** N/A for GKS**

10. ICOL5 (INTEGER*2) : Dimension text color.

    1 = Black
    2 = Red
    3 = Green

4 = Blue
5 = Yellow
6 = Magenta
7 = Cyan
8 = White
Negative integer = use default

11. *SL5(1)* (REAL*8 (10)) : Character slant in degrees for the dimension text (positive = CCW, negative = CW).

To use the default, set to -999.
* *N/A for **Micro-CSL**.*
* GKS uses an italic font if SL4 > 0.0.

12. ROT5(1) (REAL*8 (10)) : Dimension text rotation angle in degrees (positive = CCW, negative = CW).

To use the default, set to -999.

13. HT5(1) (REAL*8 (10)) : Height of the dimension text.

Negative number = use default

14. *SW5(1)* (REAL*8 (10)) : Width of the dimension text.

Negative number = use default.
* *N/A for **Micro-CSL**.*

15. SCALE (REAL*4) : Allows for scaling of the dimension text string value. For example: A drawing can be scaled by a factor of 10 so the desired dimension value must be divided by 10. The use of this parameter depends upon the next argument - DIMFLAG.

16. DIMFLAG (INTEGER*2) : Flag for dimension text format.

999 = Divide dimension length by SCALE and convert to fractional format. Applicable only if using units of feet or inches.
888 = Divide dimension length by SCALE and leave in floating point format.
OTHERWISE = The text string will reflect the true dimension length in floating point format.

17. EXT1 (REAL*8) : Distance to extend the witness line for the first point dimensioned.
NOTE: If EXT1 = -999.0 the witness line will not be drawn on the side of the first point dimensioned.

18. EXT2 (REAL*8) : Distance to extend the witness line for the second point dimensioned.
NOTE: If EXT2 = -999.0 the witness line will not be drawn on the side of the first point dimensioned.

19. SYS (INTEGER*2) : System number for output.

1 = **DXF**
2 = **Micro-CSL**
3 = **GKS**

20. IDBG5 (INTEGER) : Debug flag for **UWGRAPH** general dimension routine. If IDBG5 = 1 each argument in the general dimension call will be written to the file debug.txt.

Table 7. General Dimension Entity Example

```
C     INITIALIZATION CALL
      .
      .
      .
      XPT1 = 10
      YPT1 = 90
      XPT2 = 29
      YPT2 = YPT1
      HT5(1) = 0.8
      DNUMX = (XPT1 + XPT2) / 2.0
      DNUMY = YPT1 + 1.5 + HT5(1)
      EXT2 = 2.0
      EXT1 = 0.0
      WID5 = -1
      ICOL5 = 8
      ILEV5 = 3
      SL5(1) = -999
      ROT5(1) = -999
      SW5(1) = -1
      DIMFLAG = 999
      SCALE = 1.0
      IDBG5 = 1
      CALL UWDIM (DNUMX, DNUMY, XPT1, YPT1, XPT2, YPT2, ITYP5,
     &  WID5, ILEV5, ICOL5, SL5, ROT5, HT5, SW5, SCALE, DIMFLAG,
     &  EXT1, EXT2, SYS, IDBG5)
      .
      .
      .
C     TERMINATION CALL
```

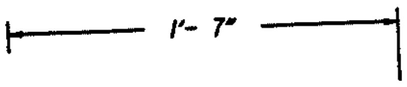


Figure 5.a

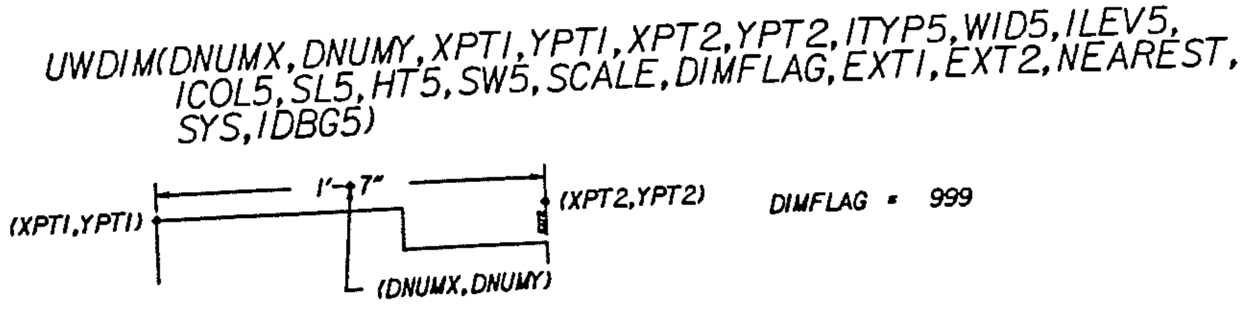

UWDIM(DNUMX,DNUMY,XPT1,YPT1,XPT2,YPT2,ITYP5,WID5,ILEV5,
ICOL5,SL5,HT5,SW5,SCALE,DIMFLAG,EXT1,EXT2,NEAREST,
SYS,IDBG5)
1'- 7"
(XPT1,YPT1)
(XPT2,YPT2)
(DNUMX,DNUMY)
DIMFLAG = 999

Figure 5.b

SECOND DIMENSION TYPE - A dimension is drawn with a continuous leader line drawn from witness line to witness line and a string of text or a numeric dimension placed at user defined coordinates. Witness line extensions and dimension value scaling is available with this dimension type also.

CALL UWDIM2 (ALDRX, ALDRY, MSGX, MSGY, XPT1, YPT1, XPT2, YPT2, TBUFFR, TOTCHR, ITYP6, **WID6**, ILEV6, ICOL6, *SL6*, ROT6, HT6, *SW6*, SCALE, DIMFLAG, EXT1, EXT2, SYS, IDBG6)

1. ALDRX (REAL*8) : Any X coordinate along the leader line.

2. ALDRY (REAL*8) : Any Y coordinate along the leader line.

3. MSGX (REAL*8) : X coordinate of the text message or numeric dimension.

4. MSGY (REAL*8) : Y coordinate of the text message or numeric dimension.

5. XPT1 (REAL*8) : X coordinate of the first dimension point.

6. YPT1 (REAL*8) : Y coordinate of the first dimension point.

7. XPT2 (REAL*8) : X coordinate of the second dimension point.

8. YPT2 (REAL*8) : Y coordinate of the second dimension point.

9. TBUFFR (CHARACTER*72) : Text if character string is to be placed, ignored if numeric dimension is to be placed.

10. TOTCHR(1) (INTEGER*2 (10)) : Number of characters in TBUFFR. To place a numeric dimension, set TOTCHR(1) = -1.

11. ITYP6 (INTEGER*2) : Line type for the dimension.

    1 = Solid
    2 = Dashed
    3 = Phantom
    4 = Centerline
    Negative integer = use default.

12. **WID6** (REAL*8) : Line width for the dimension.

    Negative number = use default.
    *** N/A for DXF**

13. ILEV6 (INTEGER*2) : Viewing level of the dimension.

    Negative integer = use default.
    * N/A for GKS

14. ICOL6 (INTEGER*2) : Dimension color.

1 = Black
2 = Red
3 = Green
4 = Blue
5 = Yellow
6 = Magenta
7 = Cyan
8 = White
Negative integer = use default

15. *SL6(1)* (REAL*8 (10)) : Character slant in degrees for the dimension text string (positive = CCW, negative = CW).

To use the default, set to -999.
* ***N/A for Micro-CSL.***
* GKS uses an italic font if SL4 > 0.0.

16. ROT6(1) (REAL*8 (10)) : Text string rotation angle in degrees (positive = CCW, negative = CW).

To use the default, set to -999.

17. HT6(1) (REAL*8 (10)) : Height of the dimension text string.

Negative number = use default

18. *SW6(1)* (REAL*8 (10)) : Width of the dimension text.

Negative number = use default.
* ***N/A for Micro-CSL.***

19. SCALE (REAL*4) : Allows for scaling of the dimension text string value. For example: A drawing can be scaled by a factor of 10 so the desired dimension value must be divided by 10. The use of this parameter depends upon the next argument - DIMFLAG.

20. DIMFLAG (INTEGER*2) : Flag for dimension text format.

999 = Divide dimension length by SCALE and convert to fractional format. Applicable only if using units of feet or inches.
888 = Divide dimension length by SCALE and leave in floating point format.
OTHERWISE = The text string will reflect the true dimension length in floating point format.

21. EXT1 (REAL*8) : Distance to extend the witness line for the first point dimensioned.
NOTE: If EXT1 = -999.0 the witness line will not be drawn on the side of the first point dimensioned.

22. EXT2 (REAL*8) : Distance to extend the witness line for the second point dimensioned.

NOTE: If EXT2 = -999.0 the witness line will not be drawn on the side of the first point dimensioned.

23. SYS (INTEGER*2) : System number for output.

1 = **DXF**
2 = **Micro-CSL**
3 = **GKS**

24. IDBG6 (INTEGER) : Debug flag for **UWGRAPH** second dimension routine. If IDBG6 = 1 each argument in the second dimension call will be written to the file debug.txt.

Table 8. Second Dimension Entity Example

```
C     INITIALIZATION CALL
      .
      .
      .
      XPT1 = 10
      YPT1 = 110
      XPT2 = 22
      YPT2 = 114
      HT6(1) = 0.75
      SPALED = HT6(1) / 4.0
      ALDRX = (XPT1 + XPT2) / 2.0 - (4/12.65) * 2 * HT6(1)
      ALDRY = (YPT1 + YPT2) / 2.0 - (12/12.65) * 2 * HT6(1)
      MSGX = ALDRX
      MSGY = ALDRY + SPALED + HT6(1) / 2.0
      TBUFFR = 'Dimension Text'
      TOTCHR(1) = 16
      EXT2 = 0.0
      EXT1 = 0.0
      WID6 = -1
      ICOL6 = 8
      ILEV6 = 3
      SL6(1) = -999
      ROT6(1) = -999
      SW6(1) = -1
      IDBG6 = 1
      CALL UWDIM2 (ALDRX, ALDRY, MSGX, MSGY, XPT1, YPT1, XPT2,
     &  YPT2, TBUFFR, TOTCHR, ITYP6, WID6, ILEV6, ICOL6, SL6,
     &  ROT6, HT6, SW6, SCALE, DIMFLAG, EXT1, EXT2, SYS, IDBG6)
      .
      .
      .
C     TERMINATION CALL
```

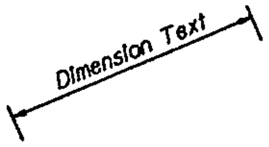


Figure 6.a

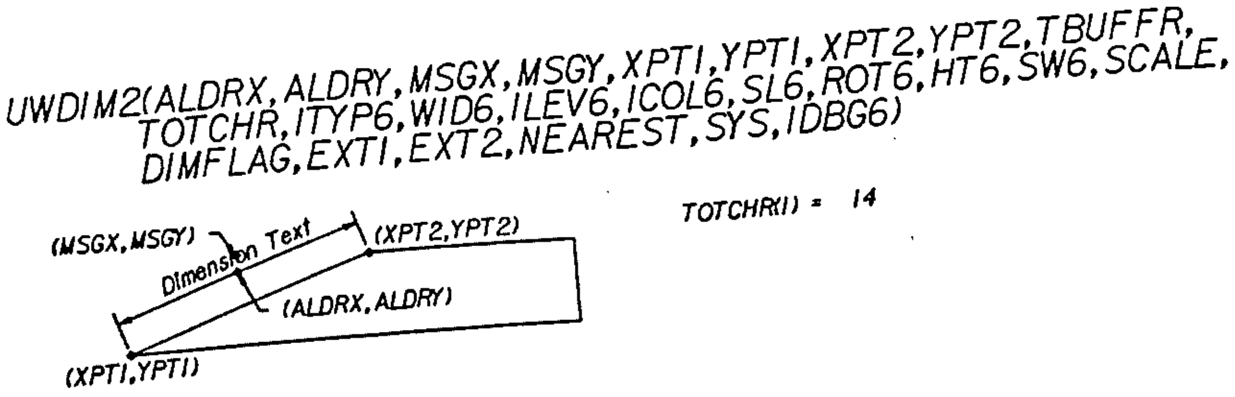

UWDIM2(ALDRX, ALDRY, MSGX, MSGY, XPT1, YPT1, XPT2, YPT2, TBUFFR,
TOTCHR, ITYP6, WID6, ILEV6, ICOL6, SL6, ROT6, HT6, SW6, SCALE,
DIMFLAG, EXT1, EXT2, NEAREST, SYS, IDBG6)
TOTCHR(1) = 14
(MSGX, MSGY)
Dimension Text
(XPT2, YPT2)
(ALDRX, ALDRY)
(XPT1, YPT1)

Figure 6.b

SUBSET VERSION OF UWDIM2 - Similar to UWDIM2, but allows more than one text string to be placed.

CALL HDDIM2 (ALDRX, ALDRY, TCENX, TCENY, XPT1, YPT1, XPT2, YPT2, TBUFFR, TOTCHR, NOSTR, ITYP6, WID6, ILEV6, ICOL6, *SL6*, ROT6, HT6, *SW6*, EXT1, EXT2, SYS, IDBG6)

1. ALDRX (REAL*8) : Any X coordinate along the leader line.
2. ALDRY (REAL*8) : Any Y coordinate along the leader line.
3. TCENX(I) (REAL*8) : X coordinate of the I'th text string.
4. TCENY(I) (REAL*8) : Y coordinate of the I'th text string.
5. XPT1 (REAL*8) : X coordinate of the first dimension point.
6. YPT1 (REAL*8) : Y coordinate of the first dimension point.
7. XPT2 (REAL*8) : X coordinate of the second dimension point.
8. YPT2 (REAL*8) : Y coordinate of the second dimension point.
9. TBUFFR (CHARACTER*72) : Contains combined character strings is to be placed.
10. TOTCHR(I) (INTEGER*2 (10)) : Number of characters in I'th string of TBUFFR.
11. NOSTR (INTEGER*2) : Number of strings to be placed.
12. ITYP6 (INTEGER*2) : Line type for the dimension.

    1 = Solid
    2 = Dashed
    3 = Phantom
    4 = Centerline
    Negative integer = use default.

13. **WID6** (REAL*8) : Line width for the dimension.

    Negative number = use default.
    *** N/A for DXF**

14. ILEV6 (INTEGER*2) : Viewing level of the dimension.

    Negative integer = use default.
    * N/A for GKS

15. ICOL6 (INTEGER*2) : Dimension color.

    1 = Black

2 = Red
3 = Green
4 = Blue
5 = Yellow
6 = Magenta
7 = Cyan
8 = White
Negative integer = use default

16. *SL6(1)* (REAL*8 (10)) : Character slant in degrees for the dimension text string (positive = CCW, negative = CW).

To use the default, set to -999.
* *N/A for* ***Micro-CSL***.
* GKS uses an italic font if SL4 > 0.0.

17. ROT6(1) (REAL*8 (10)) : Text string rotation angle in degrees (positive = CCW, negative = CW).

To use the default, set to -999.

18. HT6(1) (REAL*8 (10)) : Height of the dimension text string.

Negative number = use default

19. *SW6(1)* (REAL*8 (10)) : Width of the dimension text.

Negative number = use default.
* *N/A for* ***Micro-CSL***.

20. EXT1 (REAL*8) : Distance to extend the witness line for the first point dimensioned.
NOTE: If EXT1 = -999.0 the witness line will not be drawn on the side of the first point dimensioned.

21. EXT2 (REAL*8) : Distance to extend the witness line for the second point dimensioned.
NOTE: If EXT2 = -999.0 the witness line will not be drawn on the side of the first point dimensioned.

22. SYS (INTEGER*2) : System number for output.

1 = **DXF**
2 = **Micro-CSL**
3 = **GKS**

23. IDBG6 (INTEGER) : Debug flag for **UWGRAPH** subset dimension routine. If IDBG6 = 1 each argument in the subset dimension call will be written to the file debug.txt.

Table 9. Subset Dimension Entity Example

```
C     INITIALIZATION CALL
      .
      .
      .
      XPT1 = 10
      YPT1 = 132
      XPT2 = 22
      YPT2 = 136
      HT6(1) = 0.75
      HT6(2) = HT6(1)
      SPALED = HT6(1) / 4.0
      ALDRX = (XPT1 + XPT2) / 2.0 - (4/12.65) * 4 * HT6(1)
      ALDRY = (YPT1 + YPT2) / 2.0 - (12/12.65) * 4 * HT6(1)
      TCENX(1) = ALDRX
      TCENY(1) = ALDRY + SPALED + HT6(1) / 2.0
      TCENX(2) = TCENX(1)
      TCENY(2) = ALDRY - SPALED - HT6(1) / 2.0
      TBUFFR = 'DimensionText'
      TOTCHR(1) = 9
      TOTCHR(2) = 4
      NOSTR = 2
      EXT2 = 0.0
      EXT1 = 0.0
      WID6 = -1
      ICOL6 = 8
      ILEV6 = 3
      SL6(1) = -999
      ROT6(1) = 18.4
      SW6(1) = -1
      SL6(2) = -999
      ROT6(2) = 18.4
      SW6(2) = -1
      IDBG6 = 1
      CALL HDDIM2 (ALDRX, ALDRY, TCENX, TCENY, XPT1, YPT1, XPT2,
     &  YPT2, TBUFFR, TOTCHR, NOSTR, ITYP6, WID6, ILEV6, ICOL6,
     &  SL6, ROT6, HT6, SW6, EXT1, EXT2, SYS, IDBG6)
      .
      .
      .
C     TERMINATION CALL
```

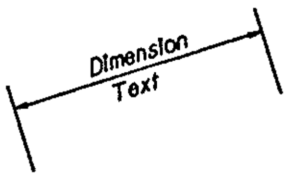


Figure 7.a

HDDIM2(ALDRX, ALDRY, TCENX, TCENY, XPT1, YPT1, XPT2, YPT2, TBUFFR, TOTCHR, NOSTR, ITYP6, WID6, ILEV6, ICOL6, SL6, ROT6, HT6, SW6, SCALE, DIMFLAG, EXT1, EXT2, NEAREST, SYS, IDBG6)

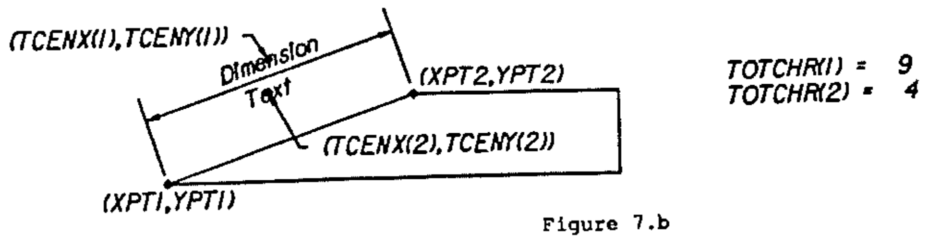


Figure 7.b

LABEL - A two segment leader is placed with an arrowhead at the tip and up to three text strings at the leader end.

CALL UWLABEL (TCENX, TCENY, TOTCHR, NOSTR, TBUFFR, TIPX, TIPY, MIDX, MIDY, ENDX, ENDY, ITYP7, **WID7**, ILEV7, ICOL7, *SL7*, ROT7, HT7, *SW7*, JUST, SYS, IDBG7)

1. TCENX(I) (REAL*8 (10)) : X coordinate of the I'th string of text.

2. TCENY(I) (REAL*8 (10)) : Y coordinate of the I'th string of text.

3. TOTCHR(I) (INTEGER*2 (10)) : Number of characters in the I'th string of text located in TBUFFR.

4. NOSTR (INTEGER*2) : Number of separate strings located in TBUFFR. Maximum of 10.

5. TBUFFR (CHARACTER*72) : Text buffer containing combined text strings without spaces between the strings.

6. TIPX (REAL*8) : X coordinate of the tip of the leader line (side where arrow is to be placed).

7. TIPY (REAL*8) : Y coordinate of the tip of the leader line (side where arrow is to be placed).

8. MIDX (REAL*8) : X coordinate of the middle point on the leader line.

9. MIDY (REAL*8) : Y coordinate of the middle point on the leader line.

10. ENDX (REAL*8) : X coordinate of the end point on the leader line.

11. ENDY (REAL*8) : Y coordinate of the end point on the leader line.

12. ITYP7 (INTEGER*2) : Line type for the label.

    1 = Solid
    2 = Dashed
    3 = Phantom
    4 = Centerline
    Negative integer = use default.

13. **WID7** (REAL*8) : Line width for the label.

    Negative number = use default.
    *** N/A for DXF**

14. ILEV7 (INTEGER*2) : Viewing level of the label.

    Negative integer = use default.
    * N/A for GKS

15. ICOL7 (INTEGER*2) : Label color.

1 = Black
2 = Red
3 = Green
4 = Blue
5 = Yellow
6 = Magenta
7 = Cyan
8 = White
Negative integer = use default

16. *SL7(I)* (REAL*8 (10)) : Character slant in degrees for the I'th text string (positive = CCW, negative = CW).

To use the default, set to -999.
* *N/A for **Micro-CSL**.*
* GKS uses an italic font if SL4 > 0.0.

17. ROT7(I) (REAL*8 (10)) : I'th text string rotation angle in degrees (positive = CCW, negative = CW).

To use the default, set to -999.

18. HT7(I) (REAL*8 (10)) : Height of the I'th text string.

Negative number = use default

19. *SW7(I)* (REAL*8 (10)) : Width of the I'th string of text.

Negative number = use default.
* *N/A for **Micro-CSL**.*

20. JUST (INTEGER) : Text justification for the text strings (HORIZONTAL/VERTICAL).

1 = LEFT/TOP
2 = LEFT/CENTER
3 = LEFT/BOTTOM
4 = CENTER/TOP
5 = CENTER/CENTER
6 = CENTER/BOTTOM
7 = RIGHT/TOP
8 = RIGHT/CENTER
9 = RIGHT/BOTTOM

21. SYS (INTEGER*2) : System number for output.

1 = **DXF**
2 = **Micro-CSL**
3 = **GKS**

22. IDBG7 (INTEGER) : Debug flag for **UWGRAPH** label routine. If IDBG7 = 1 each argument in the label call will be written to the file debug.txt.

See the example following the subset version of label description.

SUBSET VERSION OF LABEL - This label call has the same arguments as the first label call, but a text bracket is drawn automatically. Refer to the previous call for variable descriptions.

CALL HDLABEL (TCENX, TCENY, TOTCHR, NOSTR, TBUFFR, TIPX, TIPY, MIDX, MIDY, ENDX, ENDY, ITYP7, **WID7**, ILEV7, ICOL7, *SL7*, ROT7, HT7, *SW7*, JUST, SYS, IDBG7)

Table 10. Subset Label Entity Example

```
C     INITIALIZATION CALL
      .
      .
      .
      TBUFFR = 'This is a LabelWith A Text Bracket'
      TOTCHR(1) = 15
      TOTCHR(2) = 19
      NOSTR = 2
      HT7(1) = 1.25
      HT7(2) = HT7(1)
      SPALED = HT7(1) / 4.0
      TIPX = 10
      TIPY = 160
      MIDX = TIPX + 5.0 * HT7(1)
      MIDY = TIPY + 3.5 * HT7(1)
      ENDX = MIDX + 11.0 * HT7(1)
      ENDY = MIDY
      TCENX(1) = ENDX + HT7(1) / 2.0
      TCENY(1) = ENDY + SPALED / 2.0
      TCENX(2) = TCENX(1)
      TCENY(2) = ENDY - SPALED/2.0 - HT7(2)
      JUST = 3
      ICOL7 = 4
      ILEV7 = 3
      SL7(1) = -999
      ROT7(1) = -999
      SW7(1) = -1
      SL7(2) = -999
      ROT7(2) = -999
      SW7(2) = -1
      ITYP7 = 1
      WID7 = -1
      CALL HDLABEL (TCENX, TCENY, TOTCHR, NOSTR, TBUFFR, TIPX,
     &  TIPY, MIDX, MIDY, ENDX, ENDY, ITYP7, WID7, ILEV7, ICOL7,
     &  SL7, ROT7, HT7, SW7, JUST, SYS, IDBG7)
      .
      .
      .
C     TERMINATION CALL
```

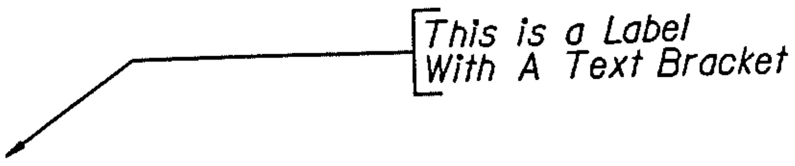


Figure 8.a

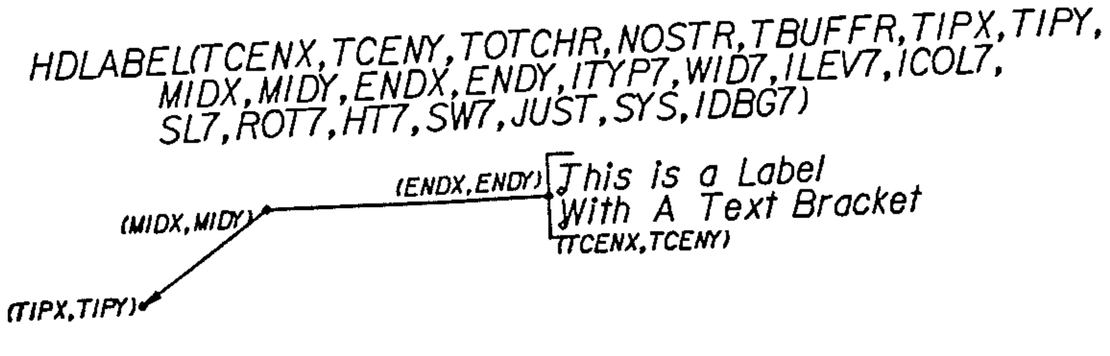

HDLABEL(TCENX,TCENY,TOTCHR,NOSTR,TBUFFR,TIPX,TIPY,
MIDX,MIDY,ENDX,ENDY,ITYP7,WID7,ILEV7,ICOL7,
SL7,ROT7,HT7,SW7,JUST,SYS,IDBG7)
(ENDX,ENDY)
This is a Label
With A Text Bracket
(MIDX,MIDY)
(TCENX,TCENY)
(TIPX,TIPY)

Figure 8.b

ARROWHEAD - Given the coordinates of a line arrowhead(s) are placed at the end(s) of the line.

CALL UWARROW (P1X, P1Y, P2X, P2Y, HT, SIDE, ITYP9, **WID9**, ILEV9, ICOL9, SYS, IDBG9)

1. P1X (REAL*8) : X coordinate of point 1 on the line.
2. P1Y (REAL*8) : Y coordinate of point 1 on the line.
3. P2X (REAL*8) : X coordinate of point 2 on the line.
4. P2Y (REAL*8) : Y coordinate of point 2 on the line.
5. HT (REAL*8) : Character height (used to scale the arrowhead; the arrowhead is drawn 1.5 characters long and 0.5 characters high)
6. SIDE (INTEGER) : Code for which side of the line to place the arrowhead.

   1 = Side of point 1
   2 = Side of point 2
   3 = Both sides

7. ITYP9 (INTEGER*2) : Line type.

   1 = Solid
   2 = Dashed
   3 = Phantom
   4 = Centerline
   Negative integer = use default.
   NOTE: UWARROW uses the same defaults as UWLABEL

8. **WID9** (REAL*8) : Line width.

   Negative number = use default.
   *** N/A for DXF**

9. ILEV9 (INTEGER*2) : Viewing level of the line.

   Negative integer = use default.
   * N/A for GKS

10. ICOL9 (INTEGER*2) : Line color.

    1 = Black
    2 = Red
    3 = Green
    4 = Blue
    5 = Yellow

6 = Magenta
7 = Cyan
8 = White
Negative integer = use default

11. SYS (INTEGER*2) : System number for output.

1 = **DXF**
2 = **Micro-CSL**
3 = **GKS**

12. IDBG9 (INTEGER) : Debug flag for **UWGRAPH** arrowhead routine. If IDBG9 = 1 each argument in the arrowhead call will be written to the file debug.txt.

Table 11. Arrowhead Entity Example

```
C     INITIALIZATION CALL
      .
      .
      .
      P1X  = 10
      P1Y  = 180
      P2X  = 35
      P2Y  = 185
      HT   = 3.00
      SIDE = 2
      ITYP9 = 1
      WID9 = -1
      ILEV9 = 3
      ICOL9 = 5
      CALL UWARROW (P1X, P1Y, P2X, P2Y, HT, SIDE, ITYP9, WID9, ILEV9, ICOL9, SYS,
     &  IDBG9)
      .
      .
      .
C     TERMINATION CALL
```

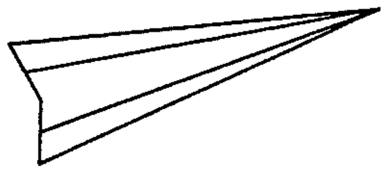


Figure 9.a

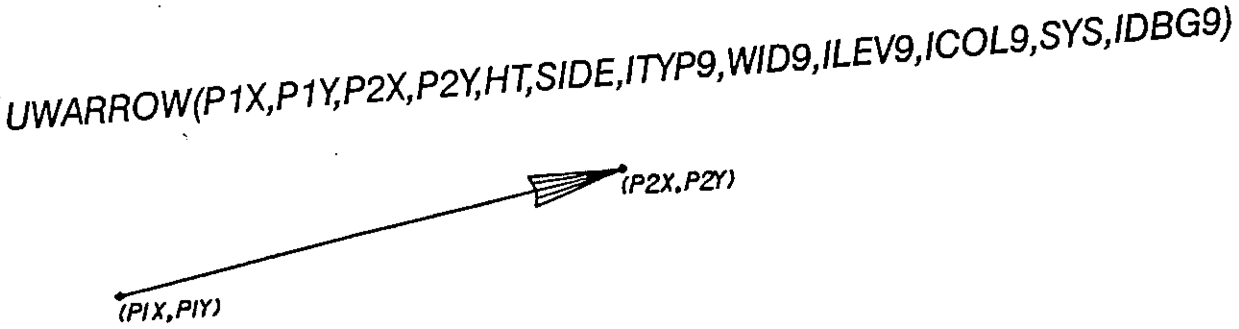

UWARROW(P1X,P1Y,P2X,P2Y,HT,SIDE,ITYP9,WID9,ILEV9,ICOL9,SYS,IDBG9)
(P2X,P2Y)
(PIX,PIY)

Figure 9.b

TERMINATION CALL:

CALL UWTERM(SYS, IDBG8)

1. SYS (INTEGER*2) : System number for output.

   1 = **DXF**
   2 = **Micro-CSL**
   3 = **GKS**

2. IDBG8 (INTEGER) : Debug flag for **UWGRAPH** termination routine. If IDBG8 = 1 each argument in the termination call will be written to the file debug.txt.

Table 13. Termination Example

```
C     INITIALIZATION CALL
      .
      .
      .
C     ENTITY CALLS
      .
      .
      .
      SYS = 3
      IDBG8 = 0
      CALL UWTERM(SYS, IDBG8)
C
      STOP
      END
```

COMPILING PROGRAMS

The following examples show the procedure for compiling and linking the first example program (TESTBOX.FOR) using three different compilers:

1. Microsoft FORTRAN Optimizing Compiler Version 4.01
2. Unix FORTRAN

MICROSOFT FORTRAN EXAMPLE

```
FL /c /4Nt /4I2 /FPi TESTBOX.FOR
LINK TESTBOX,,,UWGRAPHF
```

Note: To use the Microsoft Codeview debugger, include the options /Zi /Od in the compiling procedure and /CO in the linking procedure.

Required files:

1. TESTBOX.FOR
2. VARI.FOR
3. BLOCK.DEF
4. UWGRAPHF.LIB
5. *.cal subroutine call include files

File unit numbers used by the UWGRAPH system:

UNIT = 10 Opened in UWGRAPH.FOR (debug.txt file)
UNIT = 22 Opened in UWGRAPH.FOR (Defaults file)
UNIT = 100 DXF output file

UNIX FORTRAN EXAMPLE: (Using the GKS & CSL Libraries)

Note: All files in this example are located in the current working directory. The commands below form a UNIX makefile which is used to compile and link only the files which have been changed since the makefile was last run or when the object codes do not yet exist.
If this makefile is called 'makefile' the make is invoked by the command 'make'. If another filename is used the command 'make -f *makefilename*' must be used.

The following makefile also contains a m4 pre-processing section. The preprocessor is used to remove any system dependant comments that may reside in the source code. The source code naming conventions are as follows: source_name.F is the un-preprocessed code with the system dependent code commented out; source_name.f is the processed code (specific to a particular system) which has been created by the m4 preprocessor. The command: **include(m4_commands)** must appear at the top line of the source_name.F file. The m4_commands file contains the system dependent comments that will be striped out by m4. See Appendix D on preprocessing code.

```
FILES = testbox.f

OBJ = testbox.o

.SUFFIXES :
.SUFFIXES : .o .f .F

EDIR   = ./
LFLAGS =
FFLAGS =
MAIN   = testbox.exe
LIBS   = -lfgks -lhdl -lgks -ltools_s -lc_s -lmcsl -lm

#
#       linker
#
$(EDIR) $(MAIN): $(OBJ)
        $(LFLAGS) f77 $(OBJ) uwgraphf.lib $(LIBS) -o $(EDIR) $(MAIN)
# note:  there must be a tab at the beginning of the preceding line
#
#       compile to objects
#
.f .o:  $(OBJ)
        f77 $(FFLAGS) *.o $<
# note:  there must be a tab at the beginning of the preceding line
#
#       m4 preprocessor
#
.F .f:  $(FILES)
        m4 $? > $*.f
# note:  there must be a tab at the beginning of the preceding line
#
```

Required files:

1. testbox.F
2. uwgraphf.lib*
3. m4_commands
4. VARI.FOR*
5. BLOCK.DEF
6. The GKS Library
7. *.cal subroutine call include files*

*Note: These files should be symbolically linked to the directory from which the compile is executed to avoid using up disk space. This can be done with a command similar to the one below which links the uwgraph FORTRAN library to the current directory:

```
ln -s /usr/uwgraph/lib/uwgraph.lib .
```

File unit numbers used by the UWGRAPH system:

```
UNIT  =  10 Opened in UWGRAPH.FOR (debug.txt file)
UNIT  =  22 Opened in UWGRAPH.FOR (Defaults file)
```

Programming/Hints

Dimensioning:

Often the dimensioning text size on a sheet must remain constant even though various parts of the drawing must be to different scales. This must be taken into consideration when writing a program to create automated drawings. Perhaps the best way to provide this flexibility is to have separate scale factors for text and drawing entities. The text and whitespace between dimension lines and the drawing can be multiplied by a text scale factor which will be a function of the sheet scale. The drawing (excluding text) should use a different scale factor which might be a function of the available space on the sheet for the drawing or some other specified scale. It is wise to keep track of how much test space and white space is required by a particular drawing so there is enough space left for the rest of the drawing.

When using UWDIM to create small dimensions, if the text will not fit between the witness lines, the test coordinates are automatically changed by the dimensioning routine. The new text coordinates lie outside the witness lines on the side of the first point dimensioned. If it appears that the text for a dimension will not fit between the witness lines, leave some space for the text and/or arrowheads on the side of the first point dimensioned.

GKS Window Sizing

The initialization of GKS requires that the minimum and maximum X and Y window coordinates be specified. These coordinates should be set to the absolute minimum and maximum coordinates expected to be generated by the application. If the window height and width are not equal, the output will be distorted, therefore a square window size is recommended.

GKS Initialization

After **GKS** has been initialized, program execution must not be interrupted before termination. Within the **GKS** environment **GKS** has "full control" and a read from the keyboard or some other form of program interruption will not transfer program control external to **GKS** and will cause a lockup (unless input can be entered through another window). Therefore, all keyboard-type input must be completed before initialization or after termination.

Using an INTERPRO (Intergraph) Digitizer in GKS

When rotating or translating the output in **GKS**, digitizer input is required. The data button is used to select a point on the screen. The button should be pressed "crisply" because the screen may go blank if the button is pressed for too long. If the screen does go blank, a carriage return or an Esc key should return the display.

OUTPUT VIEWING

(Site Specific Details)

DXF

The process for converting a DXF file to the native format of a CADD system varies depending on the system being converted to. See the appropriate vendor supplied documentation for specific details.

Micro-CSL

Once a design file has been created, it can be viewed with the following procedure:

1. Login to a workstation.
2. From the directory where the design file is located type: mce *filename.*dgn
3. If using a digitizer, you must first activate the menu by typing: 'am = menu,cm'
4. Place the crosshairs on the origin of the digitizer and press the data button.
5. To fit the view to the window, select fit by pushing the inside red button while the crosshairs are over FIT on the digitizer pad. Locate a point on the screen and press the data button twice.
6. The window and zoom functions can be invoked with the digitizer with the command button and by locating screen points with the data button.

GKS

GKS output is displayed directly on the screen. The output can be manipulated by selecting a menu option from the screen menu. To select an option press the data button when the arrow is over the desired function. The available functions are:

- WINDOW (select window corners with pointing device)
- SCALE (input scale factors from menu or keyboard)
- TRANSLATE (input points with the pointing device)
- ROTATE (input an angle from the keyboard and a fixed point from the pointing device)
- PLOT (creates a **GKS** metafile with .plt extension)
- EXIT

If plot is selected, a metafile is created under the output filename and the user is returned to the menu. Plots can be made by using a plotter with a **GKS** metafile interpreter (such as **IPLOT**). The image as shown in the **GKS** window will be plotted. To create a plot, position the drawing so all of it can be seen and do the scaling at the plot menu level.

APPENDIX A

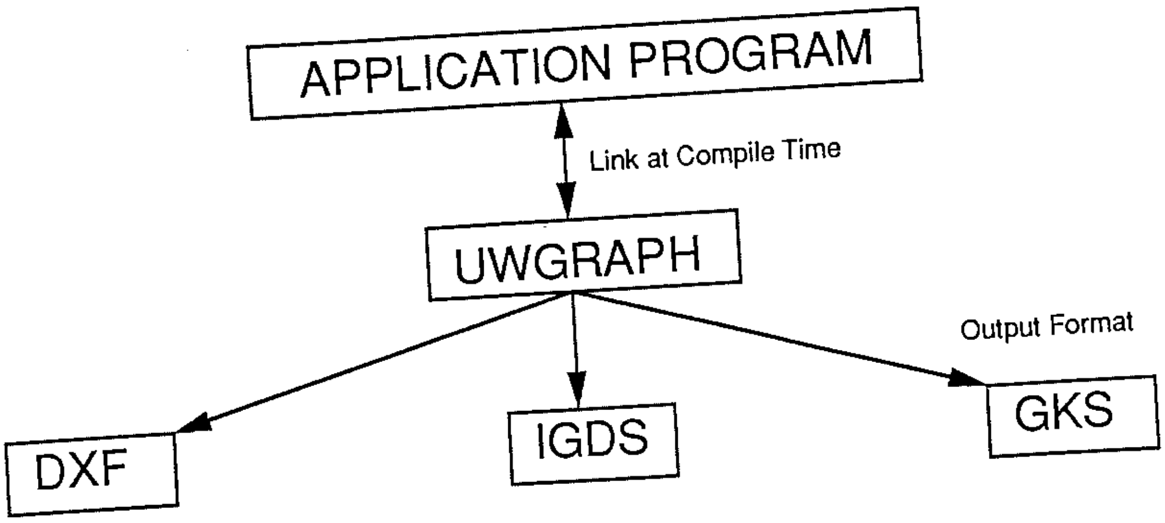


Figure A1 UWGRAPH Architecture

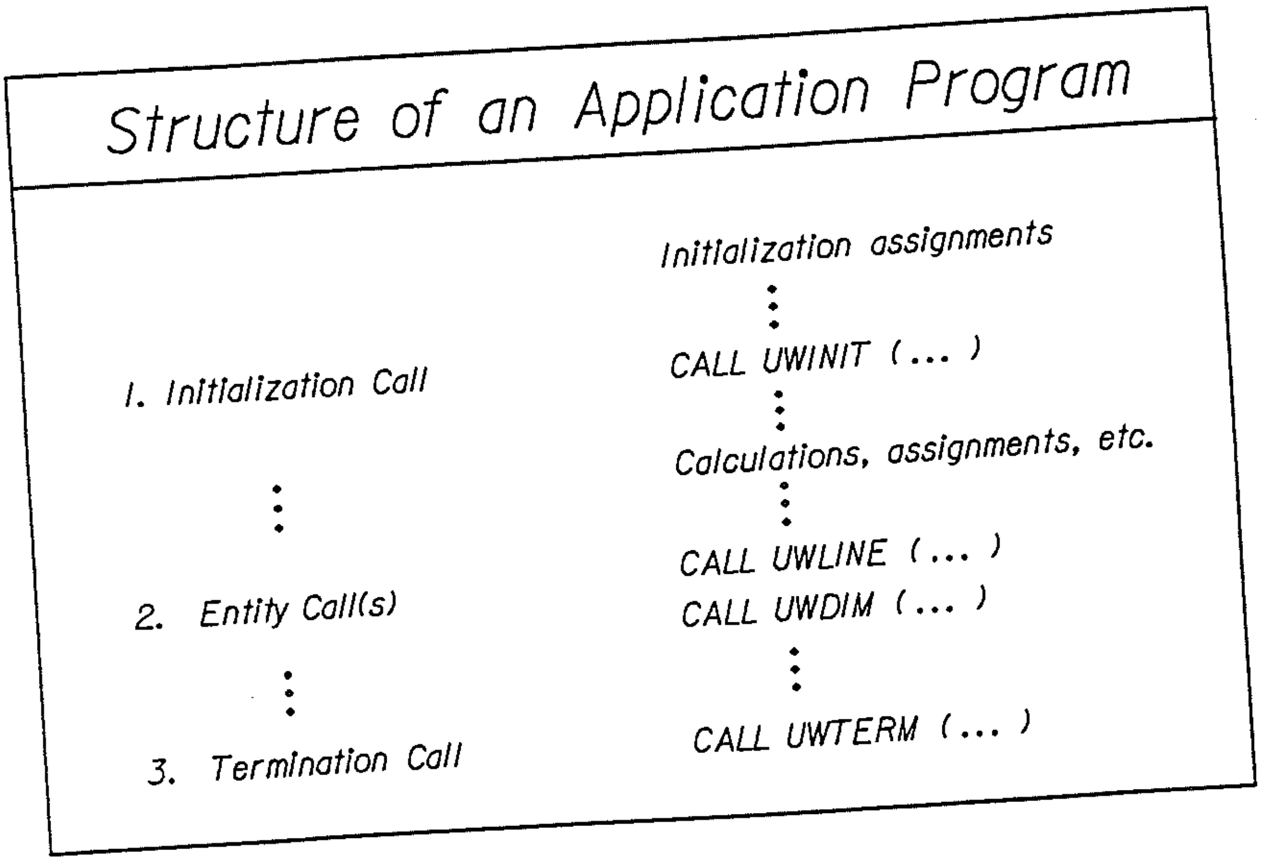
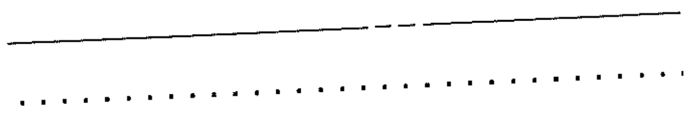


Figure A2 Structure of a typical application program

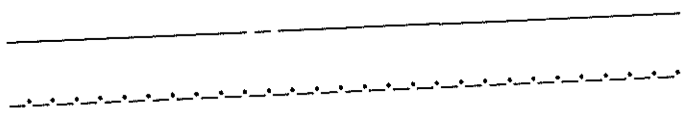
FALSE MAPPINGS

GKS does not support the use of phantom lines and centerlines so these types of lines are mapped to line styles that are available in **GKS**. The false mappings are as such.

Phantom line
is mapped to
Dotted line



Centerline
is mapped to
Dotdash line



The default color lookup table was used to select colors in Micro-CSL so the Intergraph color mappings are:

BLACK	**is mapped to**	**WHITE**
RED	is mapped to	RED
GREEN	is mapped to	GREEN
BLUE	is mapped to	BLUE
YELLOW	is mapped to	YELLOW
MAGENTA	**is mapped to**	**VIOLET**
CYAN	**is mapped to**	**BLUE**
WHITE	is mapped to	WHITE

If the colors do not map as shown, the reason may be that the colors are being displayed according to the level in which the entity was placed. Turning off the level symbology will remedy this problem.

Black is also mapped to white in the DXF system.

*.cal INCLUDE FILES

The following include files may be used in an application program to avoid having to make multiple changes to an argument list in the event of an argument list modification. The system dependant syntax of the include statement can be handled as shown in the following example (Microsoft, Apollo, VAX/VMS and Green Hills Fortran syntax for a uwinit.cal include):

```
CMSF$INCLUDE:'uwinit.cal'
CAPL%INCLUDE 'uwinit.cal'
CVAX      INCLUDE 'uwinit.cal'
CGHF      INCLUDE 'uwinit.cal'
```

After preprocessing, the source code will contain the appropriate syntax for the include statement.

```
uwinit.cal:
      CALL UWINIT(INPUTDIR,OUTNAME,XWMIN,XWMAX,YWMIN,
     &     YWMAX,WKSID,WSCON,WSTYP,UNIT,SYS,IDBG0)

usedef.cal:
      CALL USEDEF(ITYP1,ITYP2,ITYP3,ITYP4,ITYP5,ITYP6,ITYP7,
     1     ILEV1,ILEV2,ILEV3,ILEV4,ILEV5,ILEV6,ILEV7,
     2     ICOL1,ICOL2,ICOL3,ICOL4,ICOL5,ICOL6,ICOL7,
     3     WID1,WID2,WID3,WID4,WID5,WID6,WID7,
     4     SL4,SL5,SL6,SL7,
     5     ROT4,ROT5,ROT6,ROT7,
     6     SW4,SW5,SW6,SW7,
     7     HT4,HT5,HT6,HT7)

uwline.cal:
      CALL UWLINE(X1,Y1,X2,Y2,ITYP1,WID1,ILEV1,ICOL1,
     1     SYS,IDBG1)

uwarc.cal
      CALL UWARC(XCENT,YCENT,RADIUS,DRAWDIR,STANGLE,
     1     EANGLE,ITYP2,WID2,ILEV2,ICOL2,SYS,IDBG2)

uwmpline.cal
      CALL UWMPLINE(X,Y,POINTS,ITYP3,WID3,ILEV3,
     1     ICOL3,SYS,IDBG3)

uwnote.cal
      CALL UWNOTE(TCENX,TCENY,TOTCHR,NOSTR,TBUFFR,
     1     ITYP4,WID4,ILEV4,ICOL4,SL4,ROT4,HT4,SW4,
     2     JUST,SYS,IDBG4)

uwdim.cal
      CALL UWDIM(DNUMX,DNUMY,XPT1,YPT1,XPT2,YPT2,
     1     ITYP5,WID5,ILEV5,ICOL5,SL5,ROT5,HT5,SW5,
     2     SCALE,DIMFLAG,EXT1,EXT2,NEAREST,SYS,IDBG5)

uwdim2.cal
      CALL UWDIM2(ALDRX,ALDRY,MSGX,MSGY,XPT1,YPT1,XPT2,YPT2,
     1     TBUFFR,TOTCHR,ITYP6,WID6,ILEV6,ICOL6,SL6,ROT6,
     2     HT6,SW6,SCALE,DIMFLAG,EXT1,EXT2,NEAREST,SYS,
     3     IDBG6)

hddim.cal
      CALL HDDIM2(ALDRX,ALDRY,TCENX,TCENY,XPT1,YPT1,
     1     XPT2,YPT2,TBUFFR,TOTCHR,NOSTR,ITYP6,WID6,ILEV6,
     2     ICOL6,SL6,ROT6,HT6,SW6,EXT1,EXT2,SYS,IDBG6)

uwlabel.cal
      CALL UWLABEL(TCENX,TCENY,TOTCHR,NOSTR,TBUFFR,TIPX,
     1     TIPY,MIDX,MIDY,ENDX,ENDY,ITYP7,WID7,ILEV7,ICOL7,
     2     SL7,ROT7,HT7,SW7,JUST,SYS,IDBG7)

hdlabel.cal
      CALL HDLABEL(TCENX,TCENY,TOTCHR,NOSTR,TBUFFR,TIPX,
     1     TIPY,MIDX,MIDY,ENDX,ENDY,ITYP7,WID7,ILEV7,ICOL7,
     2     SL7,ROT7,HT7,SW7,JUST,SYS,IDBG7)

uwarrow.cal
      CALL UWARROW(P1X,P1Y,P2X,P2Y,HT,SIDE,ITYP9,WID9,
     1     ILEV9,ICOL9,SYS,IDBG9)

uwterm.cal
```

```
CALL UWTERM(SYS,IDBG8)
```

APPENDIX B

SAMPLE PROGRAMS

This appendix contains listings of two sample programs which demonstrate the UWGRAPH library. These programs must be preprocessed before being compiled. If m4 is being used as a preprocessor you must create the file m4_commands to indicate the comments that are to be stripped out. If you are using the STRIP preprocessor a STRIP.DAT file must be created and the first line of each program should be removed. See the appendix section on preprocessing and *.cal include files for more details.

TESTBOX PROGRAM

```
include(m4_commands)
      PROGRAM TESTBOX
C
C     This program draws a box and dimensions it using the
C     UWGRAPH library of graphics calls.
C
C     NOTE:  Key to system dependent comments:
C
C            MSF = MICROSOFT FORTRAN (IF USING, REPLACE CMSF WITH NOTHING)
C            VAX = VAX FORTRAN (IF USING, REPLACE CVAX WITH NOTHING)
C            GHF = Green Hills-FORTRAN (IF USING, REPLACE CGHF WITH NOTHING)
C            APL = APOLLO FORTRAN (IF USING, REPLACE CAPL WITH NOTHING)
C            IGS = IGES LIBRARIES BEING USED (REPLACE CIGS WITH NOTHING)
C            ITG = IGDS LIBRARIES BEING USED (REPLACE CITG WITH NOTHING)
C            GKS = GKS LIBRARIES BEING USED (REPLACE  WITH NOTHING)
C
C     First, declare the working variables with this statement.
C
CMSF$INCLUDE:'VARI.FOR'
CAPL%INCLUDE 'VARI.FOR'
CVAX      INCLUDE 'VARI.FOR'
CGHF      INCLUDE 'VARI.FOR'
      REAL WIDSCALE
C
C     Set debug flags.
C
      IDBG0 = 1
      IDBG1 = 1
      IDBG2 = 1
      IDBG3 = 1
      IDBG4 = 1
      IDBG5 = 1
      IDBG6 = 1
      IDBG7 = 1
      IDBG8 = 1
C
C     Now, open up a file to output messages.
C
      OUTNUM=41
      OPEN(UNIT=OUTNUM,FILE='MESSAGE.OUT',STATUS='UNKNOWN')
C
C     Set up the initialization parameters.
C
      OUTNAME = 'BOXOUT'
      XWMIN   = 0
      XWMAX   = 90
      YWMIN   = 0
      YWMAX   = 90
      WKSID   = 1
      WSCON   = 1
      WSTYP   = 12
      UNIT    = 'FT'
CIGS      SYS     = 1
CITG      SYS     = 2
CGKS      SYS     = 3
c     CALL UWINIT(INPUTDIR,OUTNAME,XWMIN,XWMAX,YWMIN,
c    1      YWMAX,WKSID,WSCON,WSTYP,UNIT,SYS,IDBG0)
CMSF$INCLUDE:'uwinit.cal'
CAPL%INCLUDE 'uwinit.cal'
CVAX      INCLUDE 'uwinit.cal'
CGHF      INCLUDE 'uwinit.cal'
```

```
C
C     Initialize the default flags with this call.
C
c      CALL USEDEF(ITYP1,ITYP2,ITYP3,ITYP4,ITYP5,ITYP6,ITYP7,
c     1   ILEV1,ILEV2,ILEV3,ILEV4,ILEV5,ILEV6,ILEV7,
c     2   ICOL1,ICOL2,ICOL3,ICOL4,ICOL5,ICOL6,ICOL7,
c     3   WID1,WID2,WID3,WID4,WID5,WID6,WID7,
c     4   SL4,SL5,SL6,SL7,
c     5   ROT4,ROT5,ROT6,ROT7,
c     6   SW4,SW5,SW6,SW7,
c     7   HT4,HT5,HT6,HT7)
CMSF$INCLUDE:'usedef.cal'
CAPL%INCLUDE 'usedef.cal'
CVAX      INCLUDE 'usedef.cal'
CGHF      INCLUDE 'usedef.cal'
C
C     Set the drawing scale.
C
      SCALE = 1.0
C
C     Initialize colors.
C
      ICOL3 = 8
      ICOL4 = 8
      ICOL5 = 5
C
C     Set linewidth scale.
C
      WIDSCALE = 0.3
C
C     Now, some graphics calls can be made.
C
C     Draw a multiple point line.
C
      POINTS=7
      X(1)=20
      Y(1)=40
      X(2)=50
      Y(2)=40
      X(3)=50
      Y(3)=50
      X(4)=40
      Y(4)=50
      X(5)=40
      Y(5)=60
      X(6)=20
      Y(6)=60
      X(7)=X(1)
      Y(7)=Y(1)
      WID3 = 0.01458 * WIDSCALE
c      CALL UWMPLINE(X,Y,POINTS,ITYP3,WID3,ILEV3,ICOL3,SYS,IDBG3)
CMSF$INCLUDE:'uwmpline.cal'
CAPL%INCLUDE 'uwmpline.cal'
CVAX      INCLUDE 'uwmpline.cal'
CGHF      INCLUDE 'uwmpline.cal'
C
C     Place dimension at left side.
C
      DNUMX=14
      DNUMY=50
      XPT1=20
      YPT1=40
      XPT2=20
      YPT2=60
      HT5(1)=1
      ROT5(1) = 90.0
      EXT1 = 0.0
      EXT2 = 0.0
      DIMFLAG = 999
      WID5 = 0.00833 * WIDSCALE
c      CALL UWDIM(DNUMX,DNUMY,XPT1,YPT1,XPT2,YPT2,ITYP5,WID5,
c     1      ILEV5,ICOL5,SL5,ROT5,HT5,SW5,SCALE,DIMFLAG,EXT1,
c     2      EXT2,NEAREST,SYS,IDBG5)
CMSF$INCLUDE:'uwdim.cal'
CAPL%INCLUDE 'uwdim.cal'
CVAX      INCLUDE 'uwdim.cal'
CGHF      INCLUDE 'uwdim.cal'
C
C     Place dimension at top-left.
C
      DNUMX=30
      DNUMY=64
      XPT1=20
      YPT1=60
      XPT2=40
      YPT2=60
      ROT5(1) = 0.0
c      CALL UWDIM(DNUMX,DNUMY,XPT1,YPT1,XPT2,YPT2,ITYP5,WID5,
c     1        ILEV5,ICOL5,SL5,ROT5,HT5,SW5,SCALE,DIMFLAG,EXT1,
c     2        EXT2,NEAREST,SYS,IDBG5)
CMSF$INCLUDE:'uwdim.cal'
CAPL%INCLUDE 'uwdim.cal'
CVAX       INCLUDE 'uwdim.cal'
CGHF       INCLUDE 'uwdim.cal'
C
C     Place dimension at top-right.
C
      DNUMX=45
      DNUMY=64
      XPT1=50
      YPT1=60
      XPT2=40
      YPT2=60
      EXT1 = 10.0
c      CALL UWDIM(DNUMX,DNUMY,XPT1,YPT1,XPT2,YPT2,ITYP5,WID5,
c     1        ILEV5,ICOL5,SL5,ROT5,HT5,SW5,SCALE,DIMFLAG,EXT1,
c     2        EXT2,NEAREST,SYS,IDBG5)
CMSF$INCLUDE:'uwdim.cal'
CAPL%INCLUDE 'uwdim.cal'
CVAX       INCLUDE 'uwdim.cal'
CGHF       INCLUDE 'uwdim.cal'
C
C     Place dimension on the right side.
C
      DNUMX=55
      DNUMY=45
      XPT1=50
      YPT1=50
      XPT2=50
      YPT2=40
      ROT5(1) = 90.0
c      CALL UWDIM(DNUMX,DNUMY,XPT1,YPT1,XPT2,YPT2,ITYP5,WID5,
c     1        ILEV5,ICOL5,SL5,ROT5,HT5,SW5,SCALE,DIMFLAG,EXT1,
c     2        EXT2,NEAREST,SYS,IDBG5)
CMSF$INCLUDE:'uwdim.cal'
CAPL%INCLUDE 'uwdim.cal'
CVAX       INCLUDE 'uwdim.cal'
CGHF       INCLUDE 'uwdim.cal'
C
C     Place a note.
C
      TCENX(1)=35
      TCENY(1)=72
      TOTCHR(1)=9
      NOSTR=1
      TBUFFR='TESTBOX 1'
      HT4(1)=4
      JUST = 5
      WID4 = 0.00833 * WIDSCALE
c      CALL UWNOTE(TCENX,TCENY,TOTCHR,NOSTR,TBUFFR,ITYP4,WID4,ILEV4,
c     1        ICOL4,SL4,ROT4,HT4,SW4,JUST,SYS,IDBG4)
CMSF$INCLUDE:'uwnote.cal'
CAPL%INCLUDE 'uwnote.cal'
CVAX       INCLUDE 'uwnote.cal'
CGHF       INCLUDE 'uwnote.cal'
C
C     Now, terminate graphics creation.
C
c      CALL UWTERM(SYS,IDBG8)
CMSF$INCLUDE:'uwterm.cal'
CAPL%INCLUDE 'uwterm.cal'
CVAX       INCLUDE 'uwterm.cal'
CGHF       INCLUDE 'uwterm.cal'
      STOP
      END
```

BAND SHELL PROGRAM

```
include(m4_commands)
      PROGRAM SHELL
C
C     This program draws two views of a bandshell demonstrating
C     the use of UWGRAPH library of graphics calls.
C
C     NOTE:  Key to system dependent comments:
C
C            MSF = MICROSOFT FORTRAN (IF USING, REPLACE CMSF WITH NOTHING)
C            VAX = VAX FORTRAN (IF USING, REPLACE CVAX WITH NOTHING)
C            GHF = Green Hills-FORTRAN (IF USING, REPLACE CGHF WITH NOTHING)
C            APL = APOLLO FORTRAN (IF USING, REPLACE CAPL WITH NOTHING)
C            IGS = IGES LIBRARIES BEING USED (REPLACE CIGS WITH NOTHING)
C            ITG = IGDS LIBRARIES BEING USED (REPLACE CITG WITH NOTHING)
C            GKS = GKS LIBRARIES BEING USED (REPLACE  WITH NOTHING)
C
C     First, declare the working variables with this include statement.
C
CMSF$INCLUDE:'VARI.FOR'
CAPL%INCLUDE 'VARI.FOR'
CVAX      INCLUDE 'VARI.FOR'
CGHF      INCLUDE 'VARI.FOR'
      REAL WIDSCALE
C
C     Set debug flags.
C
      IDBG0 = 1
      IDBG1 = 1
      IDBG2 = 1
      IDBG3 = 1
      IDBG4 = 1
      IDBG5 = 1
      IDBG6 = 1
      IDBG7 = 1
      IDBG8 = 1
C
C     Now, open up a file to output messages.
C
      OUTNUM=51
      OPEN(UNIT=OUTNUM,FILE='MESSAGE.OUT',STATUS='UNKNOWN')
C
C     Set up the initialization parameters.
C
      OUTNAME = 'shellout'
      XWMIN   = 0
      XWMAX   = 110
      YWMIN   = 0
      YWMAX   = 110
      WKSID   = 1
      WSCON   = 1
      WSTYP   = 12
      UNIT    = 'FT'
CIGS      SYS     = 1
CITG      SYS     = 2
CGKS      SYS     = 3
c      CALL UWINIT(OUTNAME,XWMIN,XWMAX,YWMIN,YWMAX,WKSID,
c     1        WSCON,WSTYP,UNIT,SYS,IDBG0)
CMSF$INCLUDE:'uwinit.cal'
CAPL%INCLUDE 'uwinit.cal'
CVAX      INCLUDE 'uwinit.cal'
CGHF      INCLUDE 'uwinit.cal'
C
C     Initialize the default flags with this call.
C
c      CALL USEDEF(ITYP1,ITYP2,ITYP3,ITYP4,ITYP5,ITYP6,ITYP7,
c     1    ILEV1,ILEV2,ILEV3,ILEV4,ILEV5,ILEV6,ILEV7,
c     2    ICOL1,ICOL2,ICOL3,ICOL4,ICOL5,ICOL6,ICOL7,
c     3    WID1,WID2,WID3,WID4,WID5,WID6,WID7,
c     4    SL4,SL5,SL6,SL7,
c     5    ROT4,ROT5,ROT6,ROT7,
c     6    SW4,SW5,SW6,SW7,
c     7    HT4,HT5,HT6,HT7)
CMSF$INCLUDE:'usedef.cal'
CAPL%INCLUDE 'usedef.cal'
CVAX      INCLUDE 'usedef.cal'
CGHF      INCLUDE 'usedef.cal'
```

```
C
C     Initialize some colors.
C
      ICOL1=8
      ICOL2=8
      ICOL3=8
      ICOL4=5
      ICOL7=2
C
C     Set linewidth scale
C
      WIDSCALE = 0.15
C
      WID1 = 0.01458 * WIDSCALE
      WID2 = 0.01458 * WIDSCALE
      WID3 = 0.01458 * WIDSCALE
      WID4 = 0.00833 * WIDSCALE
      WID5 = 0.00833 * WIDSCALE
      WID6 = 0.00833 * WIDSCALE
      WID7 = 0.00833 * WIDSCALE
C
C     Now, some graphics calls can be made.
C
C     PLACE LOWER LEFT CORNER
C
      X(1)=8
      Y(1)=20
      X(2)=8
      Y(2)=16
      X(3)=10
      Y(3)=16
      POINTS=3
c       CALL UWMPLINE(X,Y,POINTS,ITYP3,WID3,ILEV3,ICOL3,SYS,IDBG3)
CMSF$INCLUDE:'uwmpline.cal'
CAPL%INCLUDE 'uwmpline.cal'
CVAX      INCLUDE 'uwmpline.cal'
CGHF      INCLUDE 'uwmpline.cal'
C
C     PLACE LOWER RIGHT CORNER
C
      X(1)=60
      Y(1)=16
      X(2)=62
      Y(2)=16
      X(3)=62
      Y(3)=20
      POINTS=3
c       CALL UWMPLINE(X,Y,POINTS,ITYP3,WID3,ILEV3,ICOL3,SYS,IDBG3)
CMSF$INCLUDE:'uwmpline.cal'
CAPL%INCLUDE 'uwmpline.cal'
CVAX      INCLUDE 'uwmpline.cal'
CGHF      INCLUDE 'uwmpline.cal'
C
C     PLACE BASE OF SHELL
C
      X(1)=10
      Y(1)=16
      X(2)=10
      Y(2)=20
      X(3)=60
      Y(3)=20
      X(4)=60
      Y(4)=16
      X(5)=X(1)
      Y(5)=Y(1)
      POINTS=5
c       CALL UWMPLINE(X,Y,POINTS,ITYP3,WID3,ILEV3,ICOL3,SYS,IDBG3)
CMSF$INCLUDE:'uwmpline.cal'
CAPL%INCLUDE 'uwmpline.cal'
CVAX      INCLUDE 'uwmpline.cal'
CGHF      INCLUDE 'uwmpline.cal'
C
C     PLACE LEFT STAIRS
C1
      X1=10
      Y1=16.8
      X2=16
      Y2=16.8
c       CALL UWLINE(X1,Y1,X2,Y2,ITYP1,WID1,ILEV1,ICOL1,SYS,IDBG1)
CMSF$INCLUDE:'uwline.cal'
CAPL%INCLUDE 'uwline.cal'
```

```
CVAX      INCLUDE 'uwline.cal'
CGHF      INCLUDE 'uwline.cal'
C2
      X1=10
      Y1=17.6
      X2=16
      Y2=17.6
c       CALL UWLINE(X1,Y1,X2,Y2,ITYP1,WID1,ILEV1,ICOL1,SYS,IDBG1)
CMSF$INCLUDE:'uwline.cal'
CAPL%INCLUDE 'uwline.cal'
CVAX      INCLUDE 'uwline.cal'
CGHF      INCLUDE 'uwline.cal'
C3
      X1=10
      Y1=18.4
      X2=16
      Y2=18.4
c       CALL UWLINE(X1,Y1,X2,Y2,ITYP1,WID1,ILEV1,ICOL1,SYS,IDBG1)
CMSF$INCLUDE:'uwline.cal'
CAPL%INCLUDE 'uwline.cal'
CVAX      INCLUDE 'uwline.cal'
CGHF      INCLUDE 'uwline.cal'
C4
      X1=10
      Y1=19.2
      X2=16
      Y2=19.2
c       CALL UWLINE(X1,Y1,X2,Y2,ITYP1,WID1,ILEV1,ICOL1,SYS,IDBG1)
CMSF$INCLUDE:'uwline.cal'
CAPL%INCLUDE 'uwline.cal'
CVAX      INCLUDE 'uwline.cal'
CGHF      INCLUDE 'uwline.cal'
C5
      X1=16
      Y1=20
      X2=16
      Y2=16
c       CALL UWLINE(X1,Y1,X2,Y2,ITYP1,WID1,ILEV1,ICOL1,SYS,IDBG1)
CMSF$INCLUDE:'uwline.cal'
CAPL%INCLUDE 'uwline.cal'
CVAX      INCLUDE 'uwline.cal'
CGHF      INCLUDE 'uwline.cal'
C
C     PLACE RIGHT STAIRS
C1
      X1=54
      Y1=16.8
      X2=60
      Y2=16.8
c       CALL UWLINE(X1,Y1,X2,Y2,ITYP1,WID1,ILEV1,ICOL1,SYS,IDBG1)
CMSF$INCLUDE:'uwline.cal'
CAPL%INCLUDE 'uwline.cal'
CVAX      INCLUDE 'uwline.cal'
CGHF      INCLUDE 'uwline.cal'
C2
      X1=54
      Y1=17.6
      X2=60
      Y2=17.6
c       CALL UWLINE(X1,Y1,X2,Y2,ITYP1,WID1,ILEV1,ICOL1,SYS,IDBG1)
CMSF$INCLUDE:'uwline.cal'
CAPL%INCLUDE 'uwline.cal'
CVAX      INCLUDE 'uwline.cal'
CGHF      INCLUDE 'uwline.cal'
C3
      X1=54
      Y1=18.4
      X2=60
      Y2=18.4
c       CALL UWLINE(X1,Y1,X2,Y2,ITYP1,WID1,ILEV1,ICOL1,SYS,IDBG1)
CMSF$INCLUDE:'uwline.cal'
CAPL%INCLUDE 'uwline.cal'
CVAX      INCLUDE 'uwline.cal'
CGHF      INCLUDE 'uwline.cal'
C4
      X1=54
      Y1=19.2
      X2=60
      Y2=19.2
c       CALL UWLINE(X1,Y1,X2,Y2,ITYP1,WID1,ILEV1,ICOL1,SYS,IDBG1)
CMSF$INCLUDE:'uwline.cal'
```

```
CAPL%INCLUDE 'uwline.cal'
CVAX       INCLUDE 'uwline.cal'
CGHF       INCLUDE 'uwline.cal'
C5
      X1=54
      Y1=20
      X2=54
      Y2=16
c       CALL UWLINE(X1,Y1,X2,Y2,ITYP1,WID1,ILEV1,ICOL1,SYS,IDBG1)
CMSF$INCLUDE:'uwline.cal'
CAPL%INCLUDE 'uwline.cal'
CVAX       INCLUDE 'uwline.cal'
CGHF       INCLUDE 'uwline.cal'
C
C     DRAW INNER ARC
C
      XCENT=35
      YCENT=20
      RADIUS=25
      DRAWDIR=-1
      STANGLE=0.0
      EANGLE=180.0
c       CALL UWARC(XCENT,YCENT,RADIUS,DRAWDIR,STANGLE,EANGLE,ITYP2,
c     1     WID2,ILEV2,ICOL2,SYS,IDBG2)
CMSF$INCLUDE:'uwarc.cal'
CAPL%INCLUDE 'uwarc.cal'
CVAX       INCLUDE 'uwarc.cal'
CGHF       INCLUDE 'uwarc.cal'
C
C     DRAW OUTER ARC
C
      RADIUS=27
c       CALL UWARC(XCENT,YCENT,RADIUS,DRAWDIR,STANGLE,EANGLE,ITYP2,
c     1     WID2,ILEV2,ICOL2,SYS,IDBG2)
CMSF$INCLUDE:'uwarc.cal'
CAPL%INCLUDE 'uwarc.cal'
CVAX       INCLUDE 'uwarc.cal'
CGHF       INCLUDE 'uwarc.cal'
C
C     DRAW PODIUM WITH TWO LINE STRINGS
C1
      X(1)=34
      Y(1)=20
      X(2)=34
      Y(2)=23.2
      X(3)=34.4
      Y(3)=23.2
      X(4)=34.4
      Y(4)=22.8
      X(5)=35.6
      Y(5)=22.8
      X(6)=35.6
      Y(6)=23.2
      X(7)=36
      Y(7)=23.2
      X(8)=36
      Y(8)=20.0
      POINTS=8
c       CALL UWMPLINE(X,Y,POINTS,ITYP3,WID3,ILEV3,ICOL3,SYS,IDBG3)
CMSF$INCLUDE:'uwmpline.cal'
CAPL%INCLUDE 'uwmpline.cal'
CVAX       INCLUDE 'uwmpline.cal'
CGHF       INCLUDE 'uwmpline.cal'
C
C2
      X(1)=34.4
      Y(1)=23.2
      X(2)=34.4
      Y(2)=24
      X(3)=35.6
      Y(3)=24
      X(4)=35.6
      Y(4)=23.2
      POINTS=4
c       CALL UWMPLINE(X,Y,POINTS,ITYP3,WID3,ILEV3,ICOL3,SYS,IDBG3)
CMSF$INCLUDE:'uwmpline.cal'
CAPL%INCLUDE 'uwmpline.cal'
CVAX       INCLUDE 'uwmpline.cal'
CGHF       INCLUDE 'uwmpline.cal'
C
C     DRAW HIDDEN PODIUM IN VIEW 2
```

```
C
      ITYP3=2
      X(1)=78
      Y(1)=20.0
      X(2)=78.0
      Y(2)=23.2
      X(3)=79.2
      Y(3)=23.2
      X(4)=79.2
      Y(4)=20.0
      POINTS=4
      ICOL3=5
c      CALL UWMPLINE(X,Y,POINTS,ITYP3,WID3,ILEV3,ICOL3,SYS,IDBG3)
CMSF$INCLUDE:'uwmpline.cal'
CAPL%INCLUDE 'uwmpline.cal'
CVAX      INCLUDE 'uwmpline.cal'
CGHF      INCLUDE 'uwmpline.cal'
C
      X(1)=78.6
      Y(1)=23.2
      X(2)=79.2
      Y(2)=24.0
      X(3)=79.2
      Y(3)=23.2
      POINTS=3
c      CALL UWMPLINE(X,Y,POINTS,ITYP3,WID3,ILEV3,ICOL3,SYS,IDBG3)
CMSF$INCLUDE:'uwmpline.cal'
CAPL%INCLUDE 'uwmpline.cal'
CVAX      INCLUDE 'uwmpline.cal'
CGHF      INCLUDE 'uwmpline.cal'
C
      ICOL3=8
      ITYP3=1
C
C
C     DRAW LINE STRING FOR PERIMETER OF SIDE VIEW
C
      X(1)=102
      Y(1)=20
      X(2)=102
      Y(2)=16
      X(3)=70
      Y(3)=16
      X(4)=70
      Y(4)=16.8
      X(5)=71
      Y(5)=16.8
      X(6)=71
      Y(6)=17.6
      X(7)=72
      Y(7)=17.6
      X(8)=72
      Y(8)=18.4
      X(9)=73
      Y(9)=18.4
      X(10)=73
      Y(10)=19.2
      X(11)=74
      Y(11)=19.2
      X(12)=74
      Y(12)=20
      X(13)=75
      Y(13)=20
      X(14)=75
      Y(14)=47
      POINTS=14
c      CALL UWMPLINE(X,Y,POINTS,ITYP3,WID3,ILEV3,ICOL3,SYS,IDBG3)
CMSF$INCLUDE:'uwmpline.cal'
CAPL%INCLUDE 'uwmpline.cal'
CVAX      INCLUDE 'uwmpline.cal'
CGHF      INCLUDE 'uwmpline.cal'
C
C     DRAW THE OUTER ARC FOR THE SIDE VIEW
C
      XCENT=75
      YCENT=20
      RADIUS=27
      DRAWDIR=-1
      STANGLE=0.0
      EANGLE=90.0
c      CALL UWARC(XCENT,YCENT,RADIUS,DRAWDIR,STANGLE,EANGLE,ITYP2,
c     1     WID2,ILEV2,ICOL2,SYS,IDBG2)
CMSF$INCLUDE:'uwarc.cal'
CAPL%INCLUDE 'uwarc.cal'
CVAX        INCLUDE 'uwarc.cal'
CGHF        INCLUDE 'uwarc.cal'
C
C      DRAW THE INNER ARC FOR THE SIDE VIEW
C
       RADIUS=25
       ITYP2=2
       ICOL2=5
        CALL UWARC(XCENT,YCENT,RADIUS,DRAWDIR,STANGLE,EANGLE,ITYP2,
c     1     WID2,ILEV2,ICOL2,SYS,IDBG2)
c
CMSF$INCLUDE:'uwarc.cal'
CAPL%INCLUDE 'uwarc.cal'
CVAX        INCLUDE 'uwarc.cal'
CGHF        INCLUDE 'uwarc.cal'
C
C      DRAW BOTTOM DASHED LINE ON THE SIDE VIEW
C
       X1=75
       Y1=20.0
       X2=100
       Y2=20.0
       ITYP1=2
       ICOL1=5
        CALL UWLINE(X1,Y1,X2,Y2,ITYP1,WID1,ILEV1,ICOL1,SYS,IDBG1)
c
CMSF$INCLUDE:'uwline.cal'
CAPL%INCLUDE 'uwline.cal'
CVAX        INCLUDE 'uwline.cal'
CGHF        INCLUDE 'uwline.cal'
       ICOL1=8
       ICOL2=8
C
C
C      PLACE THE TITLE
C
       TCENX(1)=50
       TCENY(1)=62
       TCENX(2)=50
       TCENY(2)=58
       TOTCHR(1)=10
       TOTCHR(2)=26
       NOSTR=2
       TBUFFR='BAND SHELLFor the Cheyenne Orchestra'
       HT4(1)=4
       HT4(2)=2
       JUST = 5
        CALL UWNOTE(TCENX,TCENY,TOTCHR,NOSTR,TBUFFR,ITYP4,WID4,
c    &           ILEV4,ICOL4,SL4,ROT4,HT4,SW4,JUST,SYS,IDBG4)
c
CMSF$INCLUDE:'uwnote.cal'
CAPL%INCLUDE 'uwnote.cal'
CVAX        INCLUDE 'uwnote.cal'
CGHF        INCLUDE 'uwnote.cal'
C
C      PLACE THE LABEL
C
       HT7(1) = 1.5
       TOTCHR(1)=14
       NOSTR=1
       TBUFFR='Quarter sphere'
       TIPX=35
       TIPY=47
       MIDX=38
       MIDY=50
       ENDX=40
       ENDY=50
       JUST = 2
       TCENX(1) = ENDX + HT7(1) / 2.0
       TCENY(1) = ENDY
        CALL UWLABEL(TCENX,TCENY,TOTCHR,NOSTR,TBUFFR,TIPX,
c    &           TIPY,MIDX,MIDY,ENDX,ENDY,ITYP7,WID7,
c    &           ILEV7,ICOL7,SL7,ROT7,HT7,SW7,JUST,SYS,IDBG7)
c
CMSF$INCLUDE:'uwlabel.cal'
CAPL%INCLUDE 'uwlabel.cal'
CVAX        INCLUDE 'uwlabel.cal'
CGHF        INCLUDE 'uwlabel.cal'
C
C      DIMENSION STAIRS ON MAIN VIEW
C
       ALDRX=13
```

```
      ALDRY=14
      MSGX=13
      MSGY=14.3
      XPT1=10
      YPT1=16
      XPT2=16
      YPT2=16
      TBUFFR='Stairs - 6 Ft.'
      TOTCHR(1)=14
      HT6(1)=.4
c       CALL UWDIM2(ALDRX,ALDRY,MSGX,MSGY,XPT1,YPT1,XPT2,
c     &              YPT2,TBUFFR,TOTCHR,ITYP6,WID6,ILEV6,ICOL6,
c     &              SL6,ROT6,HT6,SW6,SCALE,DIMFLAG,EXT1,EXT2,
c     &              NEAREST,SYS,IDBG6)
CMSF$INCLUDE:'uwdim2.cal'
CAPL%INCLUDE 'uwdim2.cal'
CVAX        INCLUDE 'uwdim2.cal'
CGHF        INCLUDE 'uwdim2.cal'
C
C     PLACE BOTTOM DIMENSIONS ON MAIN VIEW
C
      DNUMX=35
      DNUMY=14
      XPT1=16
      YPT1=16
      XPT2=54
      YPT2=16
      HT5(1)=.5
c       CALL UWDIM(DNUMX,DNUMY,XPT1,YPT1,XPT2,YPT2,ITYP5,
c     &              WID5,ILEV5,ICOL5,SL5,ROT5,HT5,SW5,SCALE,
c     &              DIMFLAG,EXT1,EXT2,NEAREST,SYS,IDBG5)
CMSF$INCLUDE:'uwdim.cal'
CAPL%INCLUDE 'uwdim.cal'
CVAX        INCLUDE 'uwdim.cal'
CGHF        INCLUDE 'uwdim.cal'
C
      DNUMX=38
      DNUMY=10
      XPT1=8
      YPT1=16
      XPT2=60
      YPT2=16
c       CALL UWDIM(DNUMX,DNUMY,XPT1,YPT1,XPT2,YPT2,ITYP5,
c     &              WID5,ILEV5,ICOL5,SL5,ROT5,HT5,SW5,SCALE,
c     &              DIMFLAG,EXT1,EXT2,NEAREST,SYS,IDBG5)
CMSF$INCLUDE:'uwdim.cal'
CAPL%INCLUDE 'uwdim.cal'
CVAX        INCLUDE 'uwdim.cal'
CGHF        INCLUDE 'uwdim.cal'
C
      DNUMX=61
      DNUMY=10
      XPT1=62
      YPT1=16
      XPT2=60
      YPT2=16
c       CALL UWDIM(DNUMX,DNUMY,XPT1,YPT1,XPT2,YPT2,ITYP5,
c     &              WID5,ILEV5,ICOL5,SL5,ROT5,HT5,SW5,SCALE,
c     &              DIMFLAG,EXT1,EXT2,NEAREST,SYS,IDBG5)
CMSF$INCLUDE:'uwdim.cal'
CAPL%INCLUDE 'uwdim.cal'
CVAX        INCLUDE 'uwdim.cal'
CGHF        INCLUDE 'uwdim.cal'
C
C
      DNUMX=73
      DNUMY=13
      XPT1=70
      YPT1=16
      XPT2=75
      YPT2=16
c       CALL UWDIM(DNUMX,DNUMY,XPT1,YPT1,XPT2,YPT2,ITYP5,
c     &              WID5,ILEV5,ICOL5,SL5,ROT5,HT5,SW5,SCALE,
c     &              DIMFLAG,EXT1,EXT2,NEAREST,SYS,IDBG5)
CMSF$INCLUDE:'uwdim.cal'
CAPL%INCLUDE 'uwdim.cal'
CVAX        INCLUDE 'uwdim.cal'
CGHF        INCLUDE 'uwdim.cal'
C
      DNUMX=89
      DNUMY=13
      XPT1=75
      YPT1=16
      XPT2=102
      YPT2=16
c      CALL UWDIM(DNUMX,DNUMY,XPT1,YPT1,XPT2,YPT2,ITYP5,
c     &          WID5,ILEV5,ICOL5,SL5,ROT5,HT5,SW5,SCALE,
c     &          DIMFLAG,EXT1,EXT2,NEAREST,SYS,IDBG5)
CMSF$INCLUDE:'uwdim.cal'
CAPL%INCLUDE 'uwdim.cal'
CVAX      INCLUDE 'uwdim.cal'
CGHF      INCLUDE 'uwdim.cal'
C
C     Now, terminate graphics creation.
C
c         CALL UWTERM(SYS,IDBG8)
CMSF$INCLUDE:'uwterm.cal'
CAPL%INCLUDE 'uwterm.cal'
CVAX      INCLUDE 'uwterm.cal'
CGHF      INCLUDE 'uwterm.cal'
c
        CLOSE (OUTNUM)
        STOP
        END
```

SAMPLE PROGRAMS ON DISK

The following list of sample programs residing on a DOS formatted floppy disk are distributed with the UWGRAPH software. These programs must be preprocessed before compiling. See the appendix on compiling programs and preprocessing for details on creating executables from these programs.

LIST OF SAMPLE PROGRAMS:

1. TESTBOX.F
2. BANDSHELL.F
3. UWTEST.F

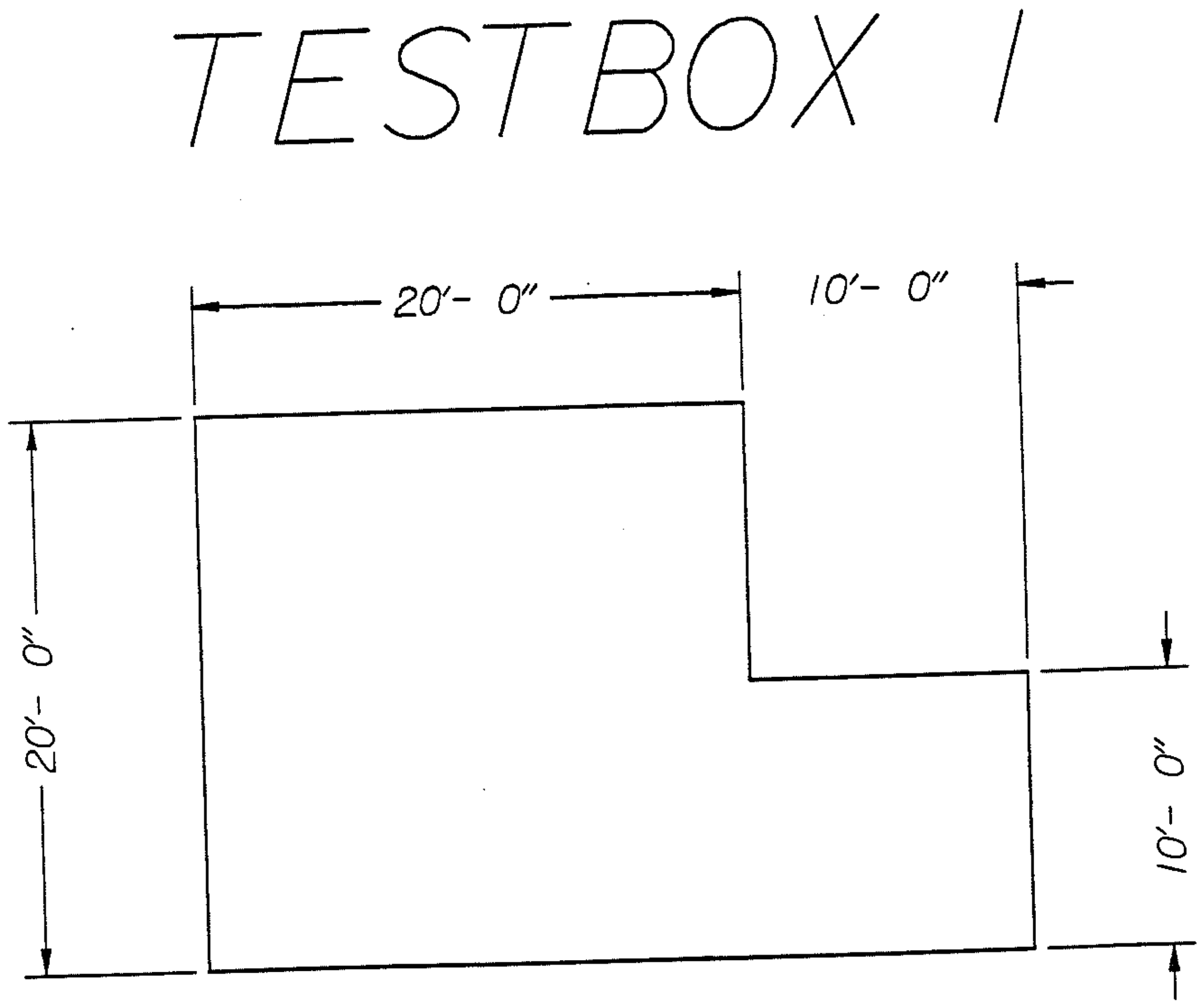


Figure B1. Testbox Output

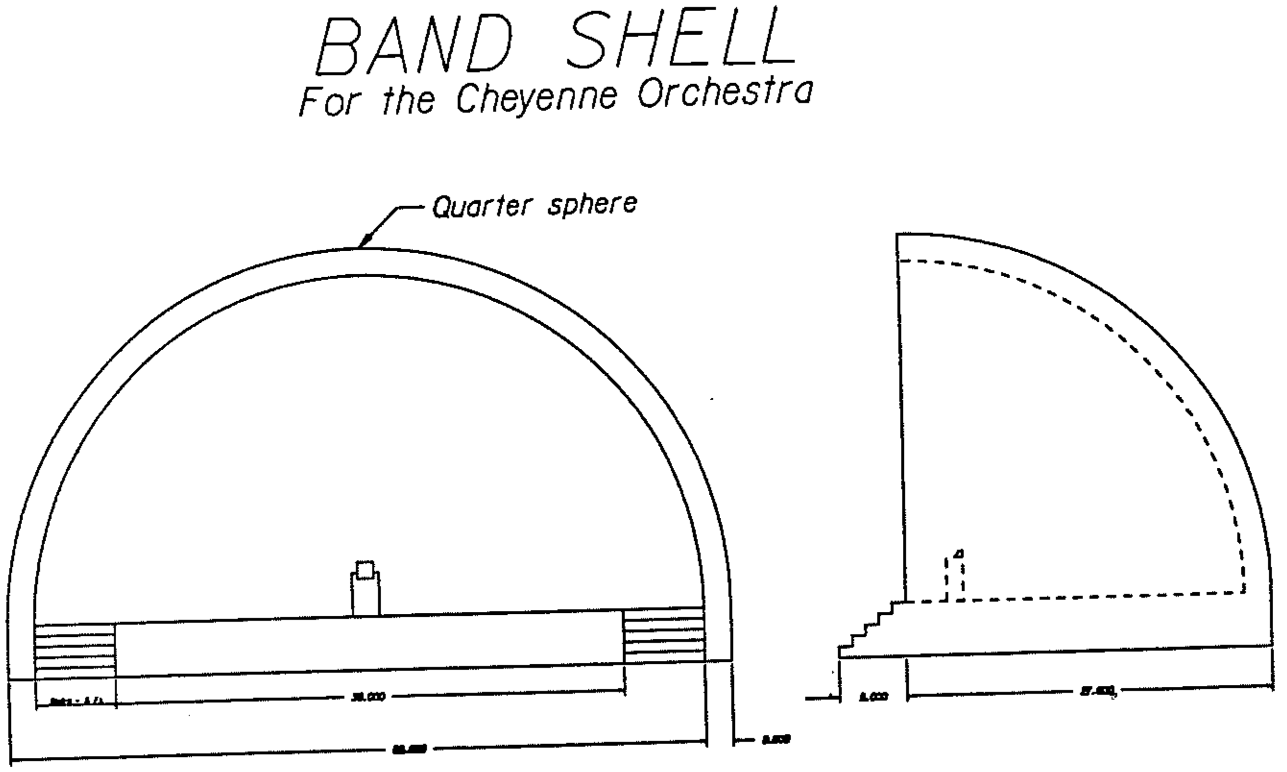


Figure B2. Band Shell Output

APPENDIX C

TECHNICAL REFERENCE TO UWGRAPH FILES

LIST OF DXF SPECIFIC SUBROUTINES:

1. DXINIT
   FUNCTION: Initializes the appropriate header table and block sections and opens the DXF output file.

   CALLS: HEADER, TABLES, BLOCKS

2. DXLINE
   FUNCTION: Creates a two point line in DXF

   CALLS: DXFCOL

3. DXFARC
   FUNCTION: Creates a circular arc in DXF given the center, radius, starting angle and ending angle.

   CALLS: DXFCOL

4. DXMPLINE
   FUNCTION: Creates a DXF line string

   CALLS: DXFCOL

5. DXNOTE
   FUNCTION: Places one or more strings of text in DXF

   CALLS: DXFCOL

6. DXDIM
   FUNCTION: Draws a DXF dimension with a break in the leader line for the dimension text

   CALLS: LASTC, DXLINE, ARROW, DXNOTE

7. DXDIM2
   FUNCTION: DXF routine to draw a continuous dimension line and place the dimension value or text at user defined coordinates.

   CALLS: LASTC, DXNOTE, FTINSYM, DXLINE, ARROW

8. DHDIM2
   FUNCTION: Similar to DXDIM2, but allows the user to place more than one text string

CALLS: DXNOTE, DXLINE, ARROW

9. DXLABEL
FUNCTION: Creates a two segment leader, an arrowhead, and up to three text strings at the end

CALLS: DXLINE, DXNOTE, ARROW

10. DHDLBL
FUNCTION: Similar to DXLABEL, but draws a bracket around the text

CALLS: DXLINE, DXMPLINE, DXNOTE, ARROW

11. DXTERM
FUNCTION: Terminates and closes DXF file

CALLS: None

12. DXFCOL
FUNCTION: Maps UWGRAPH colors to DXF colors

CALLS: None

13. HEADER
FUNCTION: Sets border and drawing limits to default values

CALLS: None

14. TABLES
FUNCTION: Sets standard line types and styles

CALLS: None

15. BLOCKS
FUNCTION: Writes the DXF Block section

CALLS: None

<u>LIST OF Micro-CSL SPECIFIC SUBROUTINES:</u>

1. GETCOL
FUNCTION: Maps UWGRAPH colors to Micro-CSL colors.

CALLS: CSL System

2. GETYPE
FUNCTION: Maps UWGRAPH line types to Micro-CSL line types.

CALLS: CSL System

3. ITARC

FUNCTION: Map uwgraph arc arguments to Micro-CSL arguments and call CSL system routines.

CALLS: GETCOL, GETYPE, CSL System

4. ITDIM

FUNCTION: Create dimension using Micro-CSL calls

CALLS: FTINSYM, ITLINE, UWARROW, ITNOTE, LASTC

5. ITDIM2

FUNCTION: Create second dimension type with Micro-CSL calls

CALLS: FTINSYM, ITLINE, UWARROW, ITNOTE, LASTC

6. ITINIT

FUNCTION: Initialize Micro-CSL graphics system

CALLS: FILECHK, LASTC, CSL System

7. ITJUST

FUNCTION: Map UWGRAPH text justification to Micro-CSL text justification

CALLS: None

8. ITLABEL

FUNCTION: Create a label in the using Micro-CSL elements

CALLS: ITNOTE, ITLINE, UWARROW

9. ITLINE

FUNCTION: Create a two point line using Micro-CSL calls

CALLS: GETCOL, GETYPE, CSL System

10. ITMPLINE

FUNCTION: Create a multiple point line using Micro-CSL calls

CALLS: GETCOL, GETYPE, CSL System

11. ITNOTE

FUNCTION: Create a text note using Micro-CSL System calls

CALLS: ITJUST, GETCOL, CSL System

12. ITTERM

FUNCTION: Terminate Micro-CSL graphics

CALLS: CSL System

13. THDIM2

FUNCTION: Create subset of dimension two dimension using Micro-CSL calls

CALLS: ITNOTE, LASTC, ITLINE, UWARROW

14. THDLBL

FUNCTION: Create bracketed label using Micro-C
SL calls

CALLS: ITNOTE, ITMPLINE, ITLINE, UWARROW

LIST OF GKS SPECIFIC SUBROUTINES:

1. ARROW

   FUNCTION: Create an arrowhead using GKS calls

   CALLS: GKLINE

2. GHDIM2

   FUNCTION: Create second dimension type using GKS calls

   CALLS: GKLINE, GKNOTE, ARROW

3. GHDLBL

   FUNCTION: Create a bracketed label using GKS calls

   CALLS: ARROW, GKLINE, GKMPLINE, GKNOTE

4. GKARC

   FUNCTION: Create a circular arc using GKS calls

   CALLS: GKCOL, GKTYP, GKS System

5. GKCOL

   FUNCTION: Map UWGRAPH colors to GKS colors

   CALLS: GKS System

6. GKDIM

   FUNCTION: Create first dimension type using GKS calls

   CALLS: FTINSYM, ARROW, GKLINE, GKNOTE, LASTC

7. GKDIM2

   FUNCTION: Create second dimension type using GKS calls

   CALLS: FTINSYM, ARROW, GKLINE, GKNOTE, LASTC

8. GKINIT

   FUNCTION: Initialize GKS System

   CALLS: GKS System

9. GKJUST

FUNCTION: Map UWGRAPH text justification to GKS text justification

CALLS: GKS System

10. GKLABEL

FUNCTION: Create a label using GKS elements

CALLS: ARROW, GKLINE, GKNOTE

11. GKLINE

FUNCTION: Draw a two point line in GKS

CALLS: GKCOL, GKTYP, GKS System

12. GKMPLINE

FUNCTION: Draw a multiple point line in GKS

CALLS: GKCOL, GKTYP, GKS System

13. GKNOTE

FUNCTION: Draw a text note in GKS

CALLS: GKCOL, GKJUST, GKS System

14. GKTERM

FUNCTION: Provide user interface to GKS and terminate GKS

CALLS: LASTC, FILECHK, GKS System

15. GKTYP

FUNCTION: Map UWGRAPH line types to GKS line types

CALLS: GKS System

LIST OF ROUTINES USED BY MORE THAN ONE SYSTEM:

1. FILECHK

   FUNCTION: Checks for the existence of the filename located in the variable FILEIN. The directory name must also be specified in the FILEIN variable or the local directory is assumed.

   CALLS: None

2. FTINCH

   FUNCTION: Converts a distance in feet residing in a REAL*4 typed variable to a character string. NEAREST is used to round the distance to a the nearest value as specified in this variable name (also in feet). The format of the character string is: # -# #/##

   CALLS: ROUNDIT,ROUNDUP,LASTC

3. FTINSYM

   FUNCTION: Converts a distance in feet residing in a REAL*4 typed variable to a character string. NEAREST is used to round the distance to a the nearest value as specified in this variable name (also in feet). This routine builds the distance string with ' and " symbols. The format of the character string is: #' -# #/##"

   CALLS: ROUNDIT,ROUNDUP,LASTC

4. LASTC

   FUNCTION: Counts the number of characters in a string before any trailing blanks.

   CALLS: None

5. READDIR

FUNCTION: Reads the directory names located in the first three lines of the data file BLOCK.DEF.

CALLS: none

6. ROUNDIT

FUNCTION: Rounds ROUNDED to the nearest NEAR.

CALLS: None

7. ROUNDUP

FUNCTION: Rounds <u>up</u> ROUNDED to the nearest NEAR.

CALLS: none

8. READDATA

FUNCTION: Reads data using free format from a text file which has been opened under the first argument. Lines in the text file may be commented out with a special character in the first column, other lines will be read into an array of reals. The number of items read is specifed by the third argument.

CALLS: None

9. REALIN

FUNCTION: REAL*8 function to convert character string to REAL*8 value

CALLS: None

10. SWITCH

FUNCTION: Switch two reals

CALLS: None

11. SYSNUM

FUNCTION: Write error message to file if wrong system number given

CALLS: None

MISCELLANEOUS FILES REQUIRED BY THE UWGRAPH LIBRARY:

1. BLOCK.DEF: Data file containing directory names and entity attribute defaults. This file is required by all three systems.

2. DEFAULT.FOR: Include file containing default variable declarations.

3. ITCOM.FOR: Include file containing unit conversion variable declarations.

4. seed.dgn, seed.cel, fontlib: Files required by **Micro-CSL** initialization.

5. prdfpilnk: one line text file required for Micro-CSL structure mapping

BLOCK.DEF Defaults Data File

```
xxx~/input/
./
/usr/ip32/mstation/
#
#NOTE1:  The first two of the above directory names are the default
#input and output directories, respectively.
#The input directory must contain input data files for applications
#(if the application requires the data files to be in that directory).
#The output directory is where the graphics file will be sent.
#The third directory is used in csl routines for the seed cell/dgn
#file to be attached to the design file.  If csl routines
#are not being called the third directory name will be ignored.
#Three x's in the first three columns will flag the directory reading
#program not to echo this directory name when read (this is useful if
#the application does not use this the input directory name)
#These directory defaults may be modified by the user and must
#end with a trailing /.
#
#The data listed below is for the default entity attributes
#and is separated for each of the three systems.  There is one
#comment indicating the graphics system followed by a one line
#entity header for each entity and the data for that entity.  The
#order of the data is as follows:
#
#  Line Type, Line Width, Level, Color, Char. Slant, String Rotation,
#  Char. Height, String Width
#
#Each default attribute must be on one line. Use care in modifying this
#file.  Adding or subtracting lines can have a detrimental effect on
#the ability for UWGRAPH to run properly.
#
#--------> SYSTEM NUMBER ONE:  DXF <--------
#DXF LINE DEFAULTS NEXT 8 LINES
      1.000000
  1.750000E-01
      1.000000
      8.000000
  0.000000E+00
  0.000000E+00
  0.000000E+00
  0.000000E+00
#DXF ARC DEFAULTS NEXT 8 LINES
      1.000000
  1.750000E-01
      1.000000
      8.000000
  0.000000E+00
  0.000000E+00
  0.000000E+00
  0.000000E+00
#DXF MULTIPLE POINT LINE DEFAULTS NEXT 8 LINES
      1.000000
  1.750000E-01
      1.000000
      8.000000
  0.000000E+00
  0.000000E+00
  0.000000E+00
  0.000000E+00
#DXF GENERAL NOTE DEFAULTS NEXT 8 LINES
```

```
        1.000000
    1.750000E-01
        3.000000
        8.000000
    0.000000E+00
    0.000000E+00
    2.190000E-01
    0.000000E+00
#DXF DIMENSION DEFAULTS NEXT 8 LINES
        1.000000
    1.000000E-01
        3.000000
        8.000000
    0.000000E+00
    0.000000E+00
    1.560000E-01
    0.000000E+00
#DXF DIMENSION DEFAULTS NEXT 8 LINES
        1.000000
    1.000000E-01
        3.000000
        8.000000
    0.000000E+00
    0.000000E+00
    1.560000E-01
    0.000000E+00
#--------> SYSTEM NUMBER TWO:  IGDS <--------
#IGDS LINE DEFAULTS NEXT 8 LINES
        1.000000
    2.500000E-03
        1.000000
        8.000000
    0.000000E+00
    0.000000E+00
    0.000000E+00
    0.000000E+00
#IGDS ARC DEFAULTS NEXT 8 LINES
        1.000000
    1.460000E-03
        1.000000
        8.000000
    0.000000E+00
    0.000000E+00
    0.000000E+00
    0.000000E+00
#IGDS MULTIPLE POINT LINE DEFAULTS NEXT 8 LINES
        1.000000
    2.500000E-03
        1.000000
        8.000000
    0.000000E+00
    0.000000E+00
    0.000000E+00
    0.000000E+00
#IGDS GENERAL NOTE DEFAULTS NEXT 8 LINES
        1.000000
    2.500000E-03
        3.000000
        8.000000
    0.000000E+00
    0.000000E+00
    1.820000E-02
    0.000000E+00
#IGDS DIMENSION LINE DEFAULTS NEXT 8 LINES
```

```
       1.000000
   1.750000E-03
       3.000000
       8.000000
   0.000000E+00
   0.000000E+00
   1.300000E-02
   0.000000E+00
#IGDS GENERAL LABEL DEFAULTS NEXT 8 LINES
       1.000000
   1.750000E-03
       3.000000
       8.000000
   0.000000E+00
   0.000000E+00
   1.300000E-02
   0.000000E+00
#--------> SYSTEM NUMBER THREE:  GKS <--------
#GKS LINE DEFAULTS NEXT 8 LINES
       1.000000
   1.460000E-03
       1.000000
       8.000000
   0.000000E+00
   0.000000E+00
   0.000000E+00
   0.000000E+00
#GKS ARC DEFAULTS NEXT 8 LINES
       1.000000
   1.460000E-03
       1.000000
       8.000000
   0.000000E+00
   0.000000E+00
   0.000000E+00
   0.000000E+00
#GKS MULTIPLE POINT LINE DEFAULTS NEXT 8 LINES
       1.000000
   1.460000E-03
       1.000000
       8.000000
   0.000000E+00
   0.000000E+00
   0.000000E+00
   0.000000E+00
#GKS GENERAL NOTE DEFAULTS NEXT 8 LINES
       1.000000
   1.460000E-03
       3.000000
       8.000000
   0.000000E+00
   0.000000E+00
   1.820000E-02
   0.000000E+00
#GKS DIMENSION LINE DEFAULTS NEXT 8 LINES
       1.000000
   8.340000E-04
       3.000000
       8.000000
   0.000000E+00
   0.000000E+00
   1.300000E-02
   0.000000E+00
#GKS GENERAL LABEL DEFAULTS NEXT 8 LINES
```

```
        1.000000
    8.340000E-04
        3.000000
        8.000000
    0.000000E+00
    0.000000E+00
    1.300000E-02
    0.000000E+00
#IGDS DEFAULT FONT NUMBER NEXT LINE
24
```

APPENDIX D

APPENDIX D - PREPROCESSING

Although every effort has been made to write system independent FORTRAN code, some system dependencies were required. Two preprocessing procedures have been used to maintain the UWGRAPH library in a system independent fashion. System dependencies are 'commented' out with special character string starting with the character 'C', for example, CVAX. The preprocessor removes the string based on a developer-defined file which instructs the preprocessor. The first procedure described is based on the UNIX m4 utility. The second preprocessor described, called STRIP.FOR is FORTRAN based and is more portable, but less powerful than its m4 counterpart.

UNIX UTILITIES

m4: A processor provided in SysV and BSD Unix Environments. Functionality includes: arguments, conditional testing, arithmetic capabilities, string and substring functions, and file manipulations.

make: A utility which automates the many activities of program development. The development information, such as inter-file dependencies and command sequences, is stored in a file which *make* interprets.

TYPE CONVENTIONS

The textual information is set in helvetica which is the same type as this line is printed.

```
The example commands and code are set in Courier which is the
same type as this line is printed.
```

The program names, utilities, etc. are set in italics which is the same type as this line is printed.

INTRODUCTION

Programming languages such as FORTRAN and C have become standardized through national efforts and the standardized language specifications are readily available. Hence the programmer should make every effort to conform to the standards so the resulting code can be easily ported to a variety of operating systems. Most computer vendors support additional language features not included in the standard language specifications. Such features include: file access procedures, system calls to operating system procedures, special calls to hardware such as the system clock, etc., and special features related to hardware such as vectorization/parallelization commands.

Often it is simply not possible (or very convenient) to develop an application without the use of language or commands which are specific to a particular operating system. If such features are employed then the appropriate language must be substituted when the code is ported to a new operating system. Sometimes the equivalent function is not available on a target system and significant re-coding must be performed. This leads to multiple versions of the program and the multiple versions lead to increased effort in maintenance, tests, and verification.

The procedures outlined here allow the programmer to form one version of a program source code which includes all the system dependencies. This source code has special syntax which is interpreted by a preprocessor (m4) to produce the system dependent code. The operational procedures of the preprocess compile, and link has been automated with the Unix make utility. The process is illustrated in Figure 1. The development level of source code which included only those dependencies for a particular system is referred to as source_2.

PROCEDURES

The procedure for maintaining a system independent code and the conversion to a system dependent code is outlined below.

1. The source_1 program and associated subroutines are written with system dependencies "commented out" with special strings, such as cunx, cvms, and cibm. When *m4* interprets the special strings then it will "un-comment" the appropriate lines and leave the other lines commented to be ignored by the compiler. For example:

```
c          The underscode represents blanks, see below.
cunx____   WRITE(ounit, 100)
cunx__100  FORMAT(10x, 'This line is written in the Unix
cunx_____&                Version')
```

In the above, the cunx string will be removed allowing the compiler to include this statement. The special strings are defined in a file called m4_commands. As many search and replace operations may be included as necessary. For example, special strings such as end-of-line comments and embedded comment delimeters may be included and modified by *m4* to the system dependent syntax.

<u>Special note:</u> m4 does not replace a special string with blanks, therefore blanks may be inserted in the source_1 code to hold the blank after string removal. For example, insert blanks after the special string (same number of blanks as special string length) so when the string is removed the source code is shifted by the appropriate number of spaces. The spaces are represented by the underscores (___) in the example above.

2. The source_1 is preprocessed by *m4* to give the system dependent code source_2. This is achieved by:

```
m4 source_1 > source_2
```

3. The source_2 code is transferred to the target system to be compiled and linked in the usual manner.

The above procedure may be described with a makefile created for the source_1 code. An example of how to code this makefile is given in the following section.

EXAMPLE

This section contains an example of how to manually perform the procedures outlined previously by typing commands at the command line. This example is followed by the automated approach using the *make* utility.

1. There are three source_1 programs required for this example. These are: main.F, sub1.F, and sub2.F. Note that the convention suggested and used later in the makefile is source_1 has an extension of .F and source_2 have an extension of .f. These programs are listed below and are contained on the DOS format floppy included at the back of this report.

main.F

```
include(m4_commands)
c    the above include places the contents of the file
c    m4_commands in this file prior to m4 processing.
c    These do not appear in the source_2 code.

     program main

     print *,  ' starting test program '

cvms print *,   ' this statement should print *,  in vms
cvms&               version '

cunx print *,  ' this statement should print *,  in Unix
cunx&               version '

     print *,  ' this statement should print *,  in any
    &               version '


c    simple subroutines for illustration
c    note _eol is a special string for in-line comments

     call sub1            _eol comment goes here

     call sub2

     stop
     end
```

sub1.F

```
include(m4_commands)
      subroutine sub1

      print *, ' called subroutine sub1 '

      return

      end
```

sub2.F

```
include(m4_commands)
      subroutine sub2

      print *, ' called subroutine sub2 '

      return

      end
```

m4_commands

```
define(cunx,      )
define(_eol, !)
```

2. Use m4 to interpret the source_1 file into source_2 files. At the command line type:

```
m4 main.F > main.f
m4 sub1.F > sub1.f
m4 sub2.F > sub2.f
```

3. Review the contents of main.f, sub1.f and sub2.f. Note the "cunx" has been translated to blanks and the "_eol" string has been translated to the comment delimeter "!".

4. Compile and link these programs in the usual manner.

The above procedure has been automated with the makefile main.mak which serves as an example of how to write similar makefiles for other programs.

main.mak

```
#   source_1 code must have an extension of .F
#   source_2 code is assigned an extension of .f
#   objects have an extension of .o
#   executable has the file name assigned in MAIN
#


FILES =    main.f\
            sub1.f\
            sub2.f

FILES2 =   main.F\
            sub1.F\
            sub2.F

OBJECTS =       main.o\
                sub1.o\
                sub2.o

EDIR =
LFLAGS =
FFLAGS = -c
MAIN = main.exe

.SUFFIXES :
.SUFFIXES : .o .f .F

#    linker
$(EDIR)$(MAIN):      $(OBJECTS)
     f77 $(LFLAGS) $(OBJECTS) -o $(EDIR)$(MAIN)

#    compile to objects
.f.o:     $(OBJECTS)
     f77 $(FFLAGS) *.o  $<

#    m4 preprocessor
.F.f:     $(FILES)
     m4 $? > $*.f
```

This procedure is invoked by typing:

```
     make -f main.mak
```

at the command line. This makefile will cause the make utility to check the dependencies and date/time stamps on all the files required to produce the source_2 code in an executable form. Only operations which are required due to a source_1 modification will be performed.

For example, assume the file sub1.F was edited and saved. The make utility, combined with the main.mak file, detects that only this file has changed and performs m4 processing to produce sub1.f. The sub1.f is recompiled to produce sub1.o and finally, the main.o, sub1.o and sub2.o is linked to produce the executable.

In order to modify the main.mak file for another code, change the file names in the appropriate macros. For large systems, this list of files is easily created at the Unix command level. For example,

```
ls -1 *.F > temp_file
```

then edit temp_file as necessary and merge with the new makefile.

MODIFICATION OF PREPROCESSING

The preprocessing with *m4* is controlled by the special string defined to *m4* in the file m4_commands. The m4_commands file contains the *m4* processing commands which are included at the start of the source_1 files via the *m4* include statement. The include inserts the file m4_commands prior to *m4* processing. Hence, once the include statement has been included in all source_1 files, then only the m4_commands file must be altered to invoke different *m4* processing, i.e. to create codes with difference system dependencies. For example, the m4_commands file given above causes *m4* to replace the string "cunx" with "   " which results in a Unix version of the code. To create a VMS version, change "cunx" in the m4_commands file to "cvms". Other *m4* commands are available for a variety of purposes. These commands should be described in detail in the Unix documentation.

STRIP.FOR PREPROCESSOR

STRIP.FOR is a preprocessor written in FORTRAN designed to process FORTRAN source code containing system dependencies. This preprocessor can be used in any environment where a FORTRAN compiler is available (the m4 preprocessor is a UNIX specific utility). The strip program will strip out the system dependent comments at the beginning of the line. These comments are located in the file STRIP.DAT and can be up to 7 characters long. The last line in the STRIP.DAT file must contain an asterisk in the first column to indicate the end of the datafile. For example the STRIP.DAT file might look something like this:

```
CUNX
CGKS
CDEBUG
*
```

NOTE: The maximum number of strip strings is 20 and the strings are only striped if they reside at the beginning of a line. The strings will be replaced with '' (nothing) so the source code should be written accordingly.

Upon execution of the STRIP program the user is prompted for input and output filenames. The following is an example of source code before and after STRIP preprocessing: (assume the example STRIP.DAT datafile is being used)

Before preprocessing:

```
      PROGRAM TESTSTRIP
C
CVAX      THIS LINE WOULD CONTAIN VAX/VMS CODE
CUNX      THIS LINE WOULD CONTAIN UNIX CODE
C
CIGS      SYS = 1
CITG      SYS = 2
CGKS      SYS = 3
C
CDEBUG      WRITE(*,*) ' SYSTEM NUMBER = ',SYS
C
      STOP
      END
```

After preprocessing:

```
      PROGRAM TESTSTRIP
C
CVAX      THIS LINE WOULD CONTAIN VAX/VMS CODE
      THIS LINE WOULD CONTAIN UNIX CODE
C
CIGS      SYS = 1
CITG      SYS = 2
      SYS = 3
C
      WRITE(*,*) ' SYSTEM NUMBER = ',SYS
C
      STOP
      END
```

Note: A program called STRIPS.FOR is also available which strips comments from multiple files. Recent work has also been completed to provide the filter type string searching and replacement functionality of m4 with a C program. This program can be compiled in any environment and works well in makefiles. This program is called m4r.c and will be put on **UWGRAPH's** standard distribution file set. Instructions on it's use are contained in the source code.

APPENDIX E

STRIP.FOR PREPROCESSOR

STRIP.FOR is a preprocessor written in FORTRAN designed to process FORTRAN source code containing system dependencies. This preprocessor can be used in any environment where a FORTRAN compiler is available (the m4 preprocessor is a UNIX specific utility). The strip program will strip out the system dependent comments at the beginning of the line. These comments are located in the file STRIP.DAT and can be up to 7 characters long. The last line in the STRIP.DAT file must contain an asterisk in the first column to indicate the end of the datafile. For example the STRIP.DAT file might look something like this:

UTILITY 1: SUBROUTINE USEDEF

UTILITY DESCRIPTION: The following subroutine assigns negative values into the variables (that can be defaulted) so the defaults are used unless otherwise changed after the call.

INCLUDE FILES: none

LOWER LEVEL CALLS: none

ARGUMENT LIST: (ITYP1,ITYP2,ITYP3,ITYP4,ITYP5,ITYP6,ITYP7,
ILEV1,ILEV2,ILEV3,ILEV4,ILEV5,ILEV6,ILEV7,
ICOL1,ICOL2,ICOL3,ICOL4,ICOL5,ICOL6,ICOL7,
WID1,WID2,WID3,WID4,WID5,WID6,WID7,
SL4,SL5,SL6,SL7,
ROT4,ROT5,ROT6,ROT7,
SW4,SW5,SW6,SW7,
HT4,HT5,HT6,HT7)

ARGUMENT DESCRIPTION: (for arguments not previously defined)

See the subroutine definitions section

UTILITY 2: SUBROUTINE DRAWNUT

UTILITY DESCRIPTION: This utility uses lines and arcs to draw an elevation of a nut of width NW and of height NH. The nut can be drawn with the flat side down or the flat side up.

INCLUDE FILES: VARI.FOR

LOWER LEVEL CALLS: UWLINE, UWARC

ARGUMENT LIST: (NH, NW, ISECT, XC, YC, DIR, ITYP1, WID1, ILEV1, ICOL1, ITYP2, WID2, ILEV2, ICOL2, SYS)

ARGUMENT DESCRIPTION: (for arguments not previously defined)

NH (REAL*4): Height of the nut.
NW (REAL*4): Width of the nut.
ISECT (INTEGER*2): Number of visible sections for the nut elevation.
DIR (INTEGER*2): Flag for direction to draw the nut.
1 = flat side down
-1 = flat side up
XC (REAL*4): X coordinate of the left side of the nut.
YC (REAL*4): Y coordinate of the flat of the nut.

UTILITY 3: SUBROUTINE DRAWTHR

UTILITY DESCRIPTION: This utility uses lines to draw a bolt threads of width TW and of height TH. The threads can be drawn with a taper at the top, a taper at the bottom or with no taper. This utility will draw the threads with a vertical axis only.

INCLUDE FILES: VARI.FOR

LOWER LEVEL CALLS: UWLINE, UWMPLINE

ARGUMENT LIST: (TH, TW, XCT, YCT, DIRT, ITYP1, WID1, ILEV1, ICOL1, SYS)

ARGUMENT DESCRIPTION: (for arguments not previously defined)

TH (REAL*4): Height of the threads.
TW (REAL*4): Width of the threads.
DIR (INTEGER*2): Flag for placement of bolt taper.
1 = place taper on the top
0 = no taper
-1 = place taper on the bottom
XC (REAL*4): X coordinate of the left side of the bolt.
YC (REAL*4): Y coordinate of the bottom of the bolt.

UTILITY 4: SUBROUTINE BOXCOORD

UTILITY DESCRIPTION: This subroutine fills the X and Y arrays with the coordinates to draw a rectangle with lower left.

INCLUDE FILES: VARI.FOR

LOWER LEVEL CALLS: NONE

ARGUMENT LIST: (XLL, YLL, L, H, X, Y)

ARGUMENT DESCRIPTION: (for arguments not previously defined)

XLL (REAL*4): X coordinate of the left side of the rectangle.
YLL (REAL*4): Y coordinate of the bottom of the rectangle.
L (REAL*4): Length (horizontal dimension) of the rectangle.
H (REAL*4): Height (vertical dimension) of the rectangle.

UTILITY 5: SUBROUTINE BLDDIM

UTILITY DESCRIPTION: This subroutine builds two or three part dimensions separated by 'x' in a fractional format. For example: 1' -6 3/4" x 10 5/16" x 2"

INCLUDE FILES: NONE

LOWER LEVEL CALLS: LASTC, FTINCH, FTINSYM

ARGUMENT LIST: (M1, M2, M3, CBLD)

ARGUMENT DESCRIPTION: (for arguments not previously defined)

M1 (REAL*4): First dimension value in feet.
M2 (REAL*4): Second dimension value in feet.
M3 (REAL*4): Third dimension value in feet (or 0.0 if two part dimension to be created).
CBLD (CHARACTER*65): Serves as both an input and output variable. The first three characters of this variable are significant in its role as an input argument. If the first character is an 'X' then the first part of the dimension is created without " or ' symbols. The second and third characters of this variable are used in the same way. CBLD is also used to return the dimension string.

APPENDIX F

APPENDIX - F UWGRAPH OVERVIEW PAPER

The following paper was presented at the FHWA Bridge Engineering Conference held in May of 1991 at Denver, CO. It provides an overview of the capabilities and architecture of UWGRAPH. It is reproduced with the permission of the Transportation Research Board.

LINKING COMPUTER AIDED ENGINEERING PROCEDURES WITH COMPUTER DRAFTING

J.A. Puckett
Associate Professor
Dept. of Civil Engr.
University of Wyoming
Laramie, WY 82071

Chad Clancy
Research Assistant
Dept. of Civil Engr.
University of Wyoming
Laramie, WY 82071

David Pope
State Bridge Engineer
Wyoming Department of Transportation
Cheyenne, WY 82002

ABSTRACT

Often a traditional design procedure is used where a designer sketches results which are based on computer applications and transmits this information via hardcopy to a drafter, who reenters it into a CAD system. The inefficiencies are clear and an obvious opportunity exists for productive gain. Recent work by the Wyoming Department of Transportation (WDT) has shown that productivity ratios can greater than 20:1 by linking design applications directly with CAD. A graphics library (UWGRAPH) has been developed to facilitate this linkage.

UWGRAPH is used in conjunction with a design application which performs engineering calculations and/or drawing parameterization. UWGRAPH links to three graphic formats commonly used in engineering DXF (AutoCAD), MICRO-CSL (Intergraph) and GKS (screen graphics) in one application program. The flexibility and familiarity of FORTRAN is combined with the tools necessary to produce graphics files automatically. The library is based on the graphics entities required for structural drafting. UWGRAPH simplifies and unifies the subroutine calls by combining low-level calls to produce graphical entities useful in engineering drawing, e.g. dimensions, notes, etc. The UWGRAPH scope is directed toward bridge engineering but easily spans other disciplines.

INTRODUCTION

Often a traditional procedure is used where a designer sketches results which are based on computer applications and transmits this information via hardcopy to a drafter, who reenters it into a CAD system. The inefficiencies are clear and an obvious opportunity exists for productive gain. Recent work by the Wyoming Department of Transportation (WDT) has shown that productivity ratios can exceed be greater than 20:1 by linking design applications directly with CAD. A graphics library (UWGRAPH) has been developed to facilitate this linkage.

UWGRAPH is used in conjunction with a design application which performs engineering calculations and/or drawing parameterization. The flexibility and familiarity of FORTRAN is combined with the tools necessary to produce graphics files either to the computer screen or to a drafting/design file which can be used by a CAD system. The library is based on the graphical entities required for structural drafting.

The difficulty with many existing graphic systems is they are often difficult to learn and implement. UWGRAPH simplifies and unifies the subroutine calls by using low-level calls, combined with intermediate calculation, to produce graphical entities useful in engineering drawing, e.g. dimensions, notes, etc. Thus, only one call is used in UWGRAPH for each entity, for example, a dimension. All the calculations and decisions involved with leader type and placement, text justification and location, etc. are automatically performed in the desired graphical format. Hence, UWGRAPH performs a great deal of work for the program developer.

STANDARDIZATION EFFORTS AND OBJECTIVES

There has been much effort to standardize computer graphics. CORE, GKS, PHIGS, DXF are examples of standardized graphics library and file formats (6). But the creation of an all-encompassing graphics format that suits every application, type of drawing, and computer system is difficult. Further, a single system which adequately supports the development environment may not support the production environment. For example, a library is used which produces an DXF file which must be interpreted by a CAD system prior to viewing the results. Because the file is converted to the native CAD format, this approach is well suited for production. However, it offers the development engineer an extremely cumbersome approach.

A library based solely on a screen/plot presentation system, like GKS or PHIGS, produces fine screen graphics, almost instantaneously, upon execution of the application. This approach is amenable to both development and the production engineers who wish to rapidly iterate design programs. However these libraries produce drawing files in formats required by only screen and plotting devices, and can not be easily edited by a drafter for inclusion in a set of plans. This is a major limitation of this programming approach as the greatest productivity potential lies with linking the application to CAD directly.

The intent of the UWGRAPH library is to merge both of the favorable attributes of the screen/presentation libraries with libraries which automate the linkage to CAD. Specifically, the objectives of the UWGRAPH development effort is to develop a graphic library to:

- ❐ Produce structural drafting which can be extended to other disciplines.
- ❐ Provide an environment for rapid program development.
- ❐ Provide an environment fast enough for the design engineer iterating on design solutions.
- ❐ Produce a file which can be interpreted or used directly in CAD systems.
- ❐ Use existing standardized graphics and CAD formats linked to a common library.

LIBRARY ARCHITECTURE

The UWGRAPH library links to three graphic libraries commonly used in engineering. The purpose of the library is to allow the use of DXF, MICRO-CSL and GKS libraries through one application program. The library architecture is illustrated in Figure 1. Graphics can be created in one or more of these three formats from a single set of subroutine calls. This is helpful in that the graphical output of the application program can be used by one who has the capability of displaying graphics in any or all of the systems. A simple software switch is available to direct the application to the desired format. The application programmer is relieved of the burden of learning more than one graphics system because the initialization, drawing definition, and termination is all consistently defined by the UWGRAPH library. Further, the application developer can use GKS to review drawings in development without accessing the CAD system. This permits program development on inexpensive graphics devices and can help to eliminate scheduling conflicts on the CAD systems.

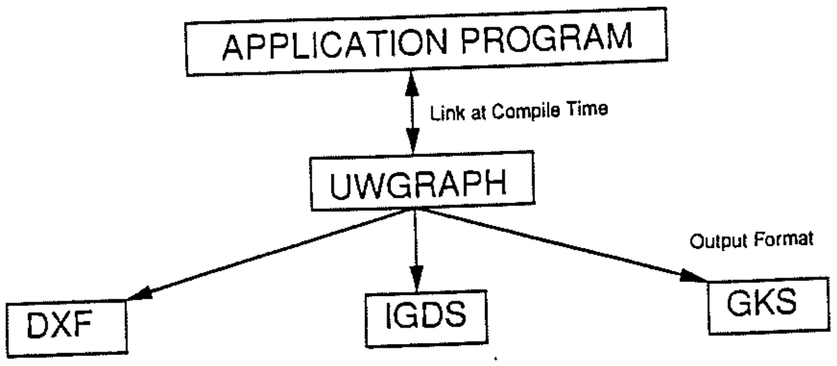


Figure 1. UWGRAPH Architecture

The first link is to the Data Exchange Format (DXF), a defacto standardized file format used by the popular AutoCAD system. UWGRAPH links to a graphics library to create an ASCII file in the DXF format. Most CAD systems have DXF interpreters to convert the DXF file into their native format. Once in the native format, the drawing can be edited in the usual manner. Thus, atypical and site specific details can be easily added and merged with other drawings as required.

The second link is to a graphics format supported by Intergraph using the Microstation Customer Support Library (MICRO-CSL) (4). MICRO-CSL is a workstation-based product and used with the Intergraph CAD software. A FORTRAN binding interfaces the capabilities of MICRO-CSL. The output from this link to the library is an MICRO-CSL design file which can be edited with an Intergraph CAD system. Approximately forty State DOTs, including WHD, use Intergraph systems. Hence, the MICRO-CSL was selected for UWGRAPH because of this important constituency.

The third link is to the Graphical Kernel System (GKS) which was accepted by the *International Organization for Standardization* as a two-dimensional graphics standard in 1985 (4,5). FORTRAN GKS bindings are available for numerous hardware/software platforms including most all engineering workstations and DOS-based computers. GKS is used by some CAD systems to display graphics. GKS is different from the other graphics systems used by UWGRAPH because the output is sent directly to the screen as opposed to an output file. Alternatively, the graphical output can be displayed on the screen, then a metafile can be created and sent to a plotter used in conjunction with a metafile interpreter.

ENTITY SPECIFICS

The UWGRAPH library supports the basic entities needed to create and dimension a drawing. These entities include lines, arcs, multiple point lines, labels, text, and several types of dimensions. In the creation of these entities UWGRAPH does not fully use the capabilities of any one graphical system, but in general uses the capabilities common to all three systems. Extended capabilities have been included by programming within UWGRAPH. For example, the DXF dimension entity is not used but is created using line and text entities common to all systems. This approach results in a practical graphics link between an application and the three graphics libraries. This functionality saves time and simplifies development. Table 1 illustrates the basic entities/procedures available in UWGRAPH.

Other higher level utilities have been developed using the basic entities. These utilities are included and described in Table 2.

It is hopeful that many agencies will use UWGRAPH to develop applications which can be shared. In application development is it inevitable that programmers will develop other high level utilities which can be properly documented and also shared with others.

Table 1. Procedures in UWGRAPH (1)

INITIALIZATION: Initializes the appropriate graphics libraries for subsequent entity calls.
LINE: Creates a two point line.
CIRCULAR ARC: Creates an arc given a center, radius, start angle and an end angle.
MULTIPLE POINT LINE: Draws a line string given more than two points.
GENERAL NOTE: Places strings of text.
GENERAL DIMENSION CALL: Creates a dimension with a break in the leader to place the dimension text. (if there is enough room for the text between witness lines)
SECOND DIMENSION TYPE: Creates a dimension which has a leader arrow drawn from witness line to witness line with no break and a string of text or a numeric dimension placed at user defined coordinates.
SUBSET VERSION OF UWDIM2: Creates a dimension similar to UWDIM2, but allows more than one text string to be placed.
LABEL: Creates a text label with a two segment leader with an arrowhead at the tip and up to three text strings at the leader end. Text justification is controlled in several ways.
SUBSET VERSION OF LABEL: Creates same label as above but automatically draws a bracket around the text associated with the label.
TERMINATION: Terminates all drawing processes.

Table 2. High Level Utilities (1)

DRAWNUT: Uses lines and arcs to draw an elevation of a nut of width NW and of height NH. The nut can be drawn with the flat side down or the flat side up.
DRAWTHR: Uses lines to draw bolt threads of width TW and of height TH. The threads can be drawn with a taper at the top, a taper at the bottom or with no taper. This utility will draw the threads with a vertical axis only.
BOXCOORD: Fills the X and Y arrays with the coordinates to draw a rectangle.
BLDDIM: Builds two or three part dimensions separated by 'x' in a fractional format. For example: 1' -6 3/4" x 10 5/16" x 2".

EXAMPLES

Two examples are presented. The first is a simple example which illustrates the usage of UWGRAPH. The example, called testbox, performs most of the operations required in a typical drawing. The FORTRAN listing of testbox is given in the Appendix and the output is illustrated in Figure 2. The drawing process has three distinct parts: initialization, drawing, and termination. The initialization is performed with a single call of UWINIT. This initializes the target system. The termination is also performed with a call to UWTERM. In between initialization and termination, any drawing entity may be called. The testbox program uses line, text, and dimemsion entities.

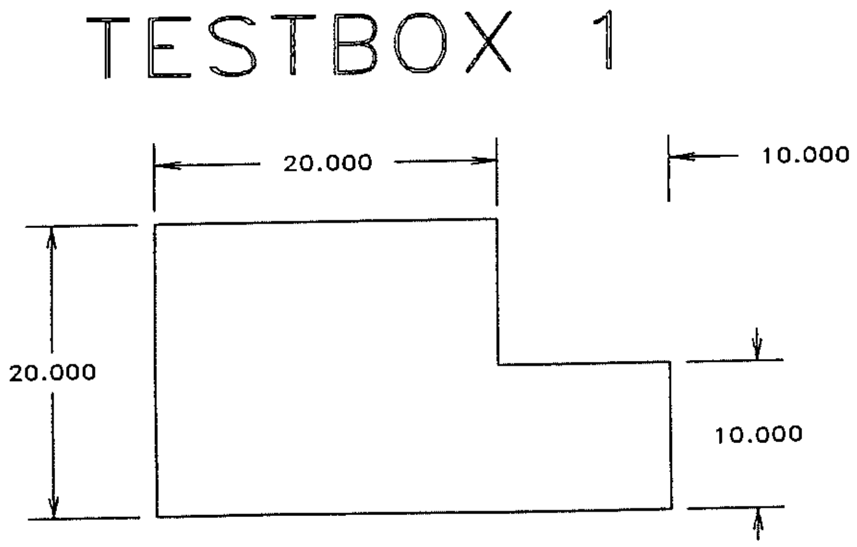
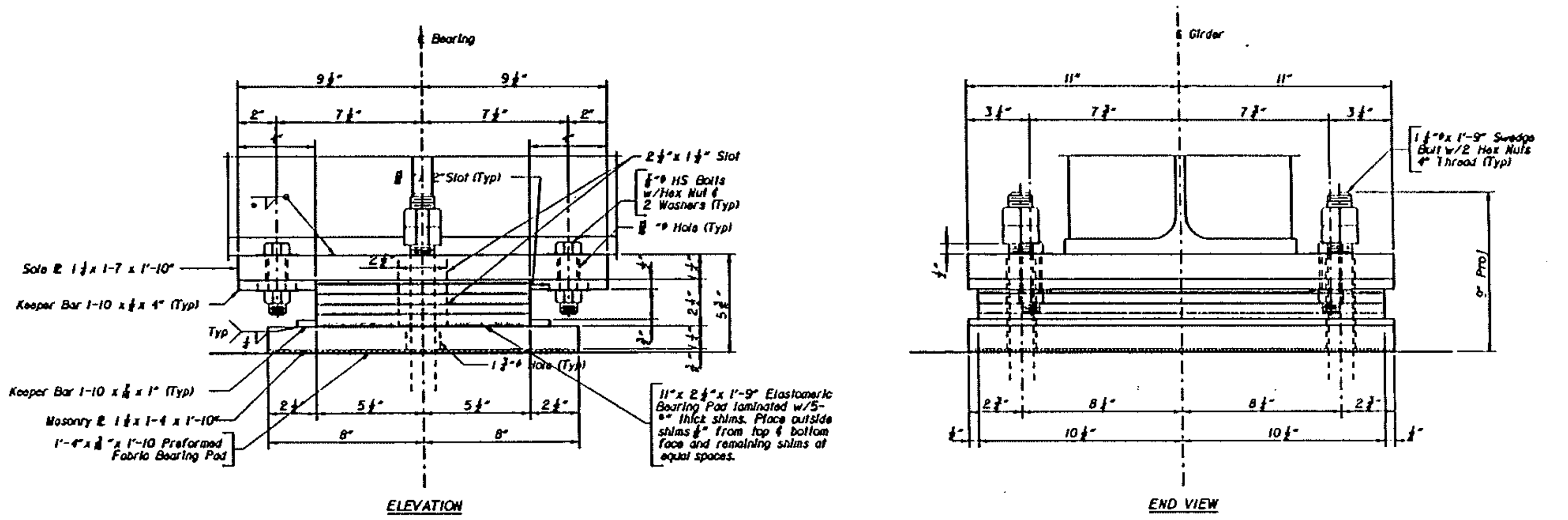


Figure 2. Output From Program Testbox

The second example is the bearing detail shown in Figure 3. This design detail was selected for automation because of its limited scope and the existence of application to perform the design. This was the initial effort to test UWGRAPH in a production environment. In this application, a few design parameters are entered to a design program which generates the details necessary for drafting. The library is used to automatically generate the drafting file. This illustration was developed from an MICRO-CSL (Intergraph CAD) file. The generation of the design and drawing took approximately five minutes plus ten minutes for cleanup and merging with the project plans. If the design was performed with the same application but the drawing was "manually" illustrated in the CAD system the time required is approximately 4.5 hours, which gives a productivity ratio of approximately 18.



℄ Bearing
2 ½"x 1 ½" Slot
2" Slot (Typ)
⅞"Φ HS Bolts w/Hex Nut & 2 Washers (Typ)
"Φ Hole (Typ)
Sole ℗ 1 ½ x 1-7 x 1'-10"
Keeper Bar 1-10 x ½ x 4" (Typ)
Typ
Keeper Bar 1-10 x ⅝ x 1" (Typ)
Masonry ℗ 1 ½ x 1-4 x 1'-10"
1'-4" x ⅛" x 1'-10 Preformed Fabric Bearing Pad
1 ¾"Φ Hole (Typ)
11"x 2 ½"x 1'-9" Elastomeric Bearing Pad laminated w/5-⅛" thick shims. Place outside shims ¼" from top & bottom face and remaining shims at equal spaces.
ELEVATION
℄ Girder
1 ½"Φx 1'-9" Swedge Bolt w/2 Hex Nuts 4" Thread (Typ)
9" Proj
END VIEW

Figure 3. Bearing Details

PRESENT WORK

Presently UWGRAPH is being used to create design and graphical illustrations for reinforced box culverts, bridge geometry, screed elevations tables and web cutting diagrams. The initial work on the box culvert application was performed using the IGES format(30) and is implemented in the BRASS system (2). The bridge geometry program is presently under development. Code writing is underway at the time of this writing. This program includes: general plan/elevation drawings and substructure layout. The screed elevation and web cutting diagram program is completed and is being tested by WDT engineers.

ACKNOWLEDGEMENT AND AVAILABILITY

The UWGRAPH library was developed under the sponsorship of the Wyoming Department of Transportation and the Mountain Plains Consortium at North Dakota State University which is sponsored by the United States Department of Transportation. The items presented in this paper do not necessarily reflect the viewpoint of the project sponsors.

The UWGRAPH source code is available upon request. The code has been sucessfully ported to two version of Unix, Intergraph CLIX 3.1 and Apollo Domain SR10.3. Other ports are presently being investigated at the time of this writing.

REFERENCES

1. Clancy, C. and Puckett, J.A., *UWGRAPH Graphics Library*, Wyoming Highway Department, Cheyenne, WY. May 14, 1990.

2. Puckett, J.A. and Guenther, P.W., *Linking Civil Engineering Design and Drafting Software via IGES*, Journal of Computing in Civil Engineering, ASCE, Vol.3, No. 3, July, 1989, pp. 228.

3. National Bureau of Standards, *Initial Graphics Exchange Specification*, Publication PB86-199759, Version 3.0, 1986.

4. Intergraph Corp., *Intergraph GKS/C Reference Manual*, DSYS027, Third Edition, Huntsville, AL, August, 1988.

5. Hopgood, F.R.A., Duce, D.A., Gallop, J.R., and Sutcliffe, D.C., *Introduction to the Graphical Kkernal System (GKS)*, Second Edition, Academic Press, Harcourt Brace Jovanovich, London, New York, 1986.

6. Dewey, B.R., *Computer Graphics for Engineers*, Harper and Row, New York, NY, 1988.

APPENDIX -- Program Listing

```
      PROGRAM TESTBOX
C
C     This program draws a box and dimensions it using the
C     UWGRAPH library of graphics calls.
C
C     First, declare the working variables with this include statement.
C
CMSF$INCLUDE:'VARI.FOR'
CVAX      INCLUDE 'VARI.FOR'
CUNX      INCLUDE 'VARI.FOR'
C
C     Now, open up a file to output messages.
C
      OUTNUM=5
      OPEN(UNIT=OUTNUM,FILE='MESSAGE.OUT',STATUS='UNKNOWN')
C
C     Set up the initialization parameters.
C
      OUTNAME = 'BOXOUT'
      XWMIN   = 0
      XWMAX   = 90
      YWMIN   = 0
      YWMAX   = 90
      WKSID   = 1
      WSCON   = 1
      WSTYP   = 12
      UNIT    = 'FT'
CDXF      SYS     = 1
CITG      SYS     = 2
CGKS      SYS     = 3
      CALL UWINIT(OUTNAME,XWMIN,XWMAX,YWMIN,YWMAX,WKSID,
     1     WSCON,WSTYP,UNIT,IDBG0,SYS)
C
C     Initialize the default flags with this call.
C
      CALL USEDEF(ITYP1,ITYP2,ITYP3,ITYP4,ITYP5,ITYP6,ITYP7,
     1   ILEV1,ILEV2,ILEV3,ILEV4,ILEV5,ILEV6,ILEV7,
     2   ICOL1,ICOL2,ICOL3,ICOL4,ICOL5,ICOL6,ICOL7,
     3   WID1,WID2,WID3,WID4,WID5,WID6,WID7,
     4   SL4,SL5,SL6,SL7,
     5   ROT4,ROT5,ROT6,ROT7,
     6   SW4,SW5,SW6,SW7,
     7   HT4,HT5,HT6,HT7)
C
C     Now, some graphics calls can be made.
C
C     Draw a multiple point line.
C
      POINTS=7
      X(1)=20
      Y(1)=40
      X(2)=50
      Y(2)=40
      X(3)=50
      Y(3)=50
      X(4)=40
      Y(4)=50
      X(5)=40
      Y(5)=60
      X(6)=20
      Y(6)=60
      X(7)=X(1)
      Y(7)=Y(1)
      CALL UWMPLINE(X,Y,POINTS,ITYP3,WID3,ILEV3,ICOL3,IDBG3,SYS)
C
C     Place dimension at left side.
```

```
C
      DNUMX=14
      DNUMY=50
      XPT1=20
      YPT1=40
      XPT2=20
      YPT2=60
      HT5(1)=1
      CALL UWDIM(DNUMX,DNUMY,XPT1,YPT1,XPT2,YPT2,ITYP5,WID5,
     1      ILEV5,ICOL5,SL5,ROT5,HT5,SW5,IDBG5,SYS)
C
C     Place dimension at top-left.
C
      DNUMX=30
      DNUMY=64
      XPT1=20
      YPT1=60
      XPT2=40
      YPT2=60
      CALL UWDIM(DNUMX,DNUMY,XPT1,YPT1,XPT2,YPT2,ITYP5,WID5,
     1      ILEV5,ICOL5,SL5,ROT5,HT5,SW5,IDBG5,SYS)
C
C     Place dimension at top-right.
C
      DNUMX=45
      DNUMY=64
      XPT1=50
      YPT1=60
      XPT2=40
      YPT2=60
      CALL UWDIM(DNUMX,DNUMY,XPT1,YPT1,XPT2,YPT2,ITYP5,WID5,
     1      ILEV5,ICOL5,SL5,ROT5,HT5,SW5,IDBG5,SYS)
C
C     Place dimension on the right side.
C
      DNUMX=55
      DNUMY=45
      XPT1=50
      YPT1=50
      XPT2=50
      YPT2=40
      CALL UWDIM(DNUMX,DNUMY,XPT1,YPT1,XPT2,YPT2,ITYP5,WID5,
     1      ILEV5,ICOL5,SL5,ROT5,HT5,SW5,IDBG5,SYS)
C
C     Place a note.
C
      TCENX(1)=32
      TCENY(1)=72
      TOTCHR(1)=9
      NOSTR=1
      TBUFFR='TESTBOX 1'
      HT4(1)=4
      CALL UWNOTE(TCENX,TCENY,TOTCHR,NOSTR,TBUFFR,ITYP4,WID4,ILEV4,
     1      ICOL4,SL4,ROT4,HT4,SW4,IDBG4,SYS)
C
C     Now, terminate graphics creation.
C
        CALL UWTERM(IDBG8,SYS)
        STOP
        END
```

Glossary of Terms

application program: a program which uses the UWGRAPH library to produce automated drawings.

argument: a variable that is passed to a subroutine through a list.

compiler: a program which interprets source code and creates object and/or executable code.

dependency: an entity attribute that is specific to a particular system.

default: an entity attribute read in from a datafile which is used if flagged to do so.

design file: (.DGN file) the type of graphics file created by the Intergraph CAD system.

digitizer: an electromagnetic sensitive tablet and a puck with crosshairs and buttons used as a computer input device.

drawing parameterization: the process of determining the dimensions and the attributes of a drawing.

entity: a graphical element, such as a line or text

entity creation: the section of an application program where one or more entity calls are made.

GKS: Graphical Kernal System, an international standard for two dimensional graphics accepted by the International Organization for Standardization.

high level utility: an application subroutine which uses the UWGRAPH library and is generic enough to be useful in several applications.

initialization: the first section of an application program which initializes the graphic system; this section opens files, reads defaults and sets window sizes among other functions.

leader line: a line used both for dimensions and labels; the leader line associated with a dimension spans between witness lines; the leader line associated with labels goes from the label text to the tip of the arrow.

library: a collection of related object modules which have been combined in one file.

m4: a UNIX utility used as a preprocessor for the source code; this processor can be used to strip out system dependent comments.

make: a UNIX based utility used to keep object and executable code up to date; the make utility checks the date stamps on files to determine wether or not they need to be compiled before the link operation.

metafile: a text file created by GKS which can be interpreted and converted to numerous formats used by output (graphics) devices.

Micro-CSL: MicroStation customer support library: the graphics library used to create MicroStation graphics primitives.

termination: the last section of an application program which closes the graphics file.

witness line: a line used in dimensioning to show an endpoint of the dimension.

INDEX