THE DISPLAY PROTOCOL
A REFERENCE FOR THE PROTOCOL

Zigurd R. Mednieks

August 1981
WP-81-112

Part of a series on the Display Protocol graphics system and FEZ front end processor system.

INTERNATIONAL INSTITUTE FOR APPLIED SYSTEMS ANALYSIS
A-2361 Laxenburg, Austria

PREFACE

The Display Protocol is a unified graphics system for use at IIASA. It is based on existing systems to some extent. It consists of a device independent protocol interpreter and a small set of device dependent routines. Adapting it to a wide range of output devices requires little programmer effort. It is different from most existing systems in that it also handles devices with no graphics capability. It can convey both textual and graphic data in the same character stream. For a more detailed discussion of the design decisions that lead to the current Display Protocol see PP-79-1, "A NETWORK COMPATIBLE DISPLAY PROTOCOL", by B. Schweeger.

This is version 1.3 of this document, last modified 8/11/81

Other design goals include having a clean user input side interface, meshing with UNIX[1] design philosophy, containing ASCII as a subset, and being compact for efficient transmission over networks.

This document describes the Display Protocol as it stands, not as it might be, and so may be out of date on some subjects. However, most revisions of the protocol are expected to be backward compatible. While this document may not represent what is latest with the Display Protocol, it should serve the applications writer adequately.

[1]UNIX is a Trade/Service Mark of Bell Laboratories

1. FUNDAMENTALS

The Display Protocol is an eightbit protocol. The eighth bit of an eight bit byte distinguishes Display Protocol codes from their arguments and other ASCII characters.

1.1. The Format of Display Protocol Codes

Display Protocol control codes are distinguished from other characters in a stream by the high bit of an eight bit byte being "on" for these codes. Any arguments these codes may have and any other characters in the stream do not have this bit on. A request to the Display Protocol interpreter consists of a control code followed by arguments if needed. Codes may have varying numbers of arguments. They may have no arguments, they may have numeric arguments, and they may have string arguments. Upon receiving a code that requires arguments the interpreter consumes bytes as arguments until no more arguments can be consumed or until another Display Protocol command is encountered. Sending fewer arguments than minimally required by a command causes an error.

1.2. Default Values and the Top Level Window

Codes are interpreted in the context of the current environment. Environments are called windows. Windows are organized hierarchically. Physically, a child window lies inside its parent. Parents do not overwrite their childeren. Siblings and other, more distant, relatives are not known about and are overwritten.

The environment of the top level window is where interpretation begins. By convention, the default limits of

this window are bounded below by the resolution of the device and above by the addressing resolution of 16384. The origin (0, 0) is initially at the lower left corner of the device. There is no formula for picking these limits. They should, though, make sense for the device. Any pixel on a device of any resolution and size may be addressed by manipulating the limits of the windows. There is no provision for fourth quadrant addressing or negative addresses.

1.3. The Cursor

There is only one cursor. This cursor can be positioned explicitly, without side effects. It can be moved as a side effect of some action. Drawing a character moves the cursor to the position where the next character would be drawn. Drawing a vector moves the cursor from where it was to the end of the vector. Note that there are no separate graphics and character cursors.

1.4. The Format of Arguments to Display Protocol Codes

There are three basic types of arguments: string, absolute, and relative.

1.4.1. Characteristics of string arguments

String arguments usually contain the name of some object in the environment the interpreter maintains. Sometimes string arguments consist of a maximum of one character. Some, however, can vary in length. It is a good idea to terminate variable length strings with a Display Protocol no-op.

1.4.2. Characteristics of absolute arguments

Absolute arguments convey unsigned numeric values. Absolute arguments, like string arguments, consist of bytes that do not have the high bit on. Each number is carried by two consecutive bytes. The first byte is taken to contain the lower seven bits of an integer, and the following byte, the upper seven bits. That is, the least significant bits are sent first, and the most significant bits second. This constrains the interpreter to a maximum 16384 by 16384 resolution per window. There may be many windows per physical device.

The following is a C program fragment that stuffs two bytes with an absolute argument. Note that a bit shift must be performed to right justify the upper seven bits before they are stored in a byte.

```
#define LOWSEVEN 0177

int outvalue;
char *output;

output[n] = outvalue & LOWSEVEN;
output[n + 1] = (outvalue >> 7) & LOWSEVEN;
```

1.4.3. Characteristics of relative arguments

Relative arguments convey signed integer quantities. One byte holds each relative argument. The high bit must be

zero, the seventh bit contains the sign, and the six low bits contain the magnitude of the argument. Negative magnitudes are in twos complement. Note that relative arguments fall between -64 and +63.

The following C program fragment stuffs a byte with a relative argument. It works for both positive and negative quantities.

```
#define SIGNBIT 0100
#define LOWSIX 077

int outvalue, temp;
char *output;

temp = outvalue & LOWSIX;
output[n] = outvalue >= 0 ? temp : temp & SIGNBIT;
```

---

2. THE CONTROL CODES

The following section describes the Display Protocol control codes. Given are the full name of the op code, the symbolic name for the eight bit value of the opcode as in the C header file display.h, the type of arguments, and the number of arguments. A brief description of the effects of each control code follow the syntax summaries.

Name: no-op
Symbol name: DPNOP
Octal value: 0200
Arguments: none

The no-op does nothing. It is used to terminate variable length argument lists.

Name: limit
Symbol name: DPLIMIT
Octal value: 0203
Arguments: Type: numeric Min: 2 Max: 4

Changes the origin and the limits of the coordinates for the current window. If two arguments are given, they are taken to be the upper bounds of of the coordinates, and the lower bounds are assumed to be (0, 0). If four arguments are given, the first two are taken to be the lower bounds, and the upper two, the upper bounds. Note that, by manipulating the limits of a window, the scaling, resolution, and clipping properties of that window are manipulated.

Name: clear
Symbol name: DPCLEAR
Octal value: 0206
Arguments: none

Clears the window. The physical effect of this code might be a formfeed for hardcopy devices.

Name: plot absolute
Symbol name: DPPLOTA
Octal value: 0207
Arguments: Type: numeric Min: 2

Draws a vector from the current cursor position to the coordinates specified. Note that any even number of arguments may be given. All seven bit quantities following this code will be taken as arguments. A no-op can be used to terminate a series of argument pairs.

Name: plot relative
Symbol name: DPPLOTR
Octal value: 0210
Arguments: Type: relative Min: 2

Draws a vector from the current cursor position to the specified coordinates. Note that any even number of arguments may be given. All seven bit quantities following this code will be taken as arguments. A no-op can be used to terminate a series of argument pairs.

Name:move absolute
Symbol name: DPMOVEA
Octal value: 0211
Arguments: Type: absolute Min: 2 Max: 2

Moves the cursor to the specified coordinates.

Name: move relative
Symbol name: DPMOVER
Octal value: 0212
Arguments: Type: Min: 2 Max: 2

Moves the cursor to the specified coordinates relative to the current position.

Name: set font absolute
Symbol name: DPSFONTA
Octal value: 0216
Arguments: Type: numeric and string Min: n.a. Max: n.a.

Binds a font to a font number. The first argument is the font number. The remaining arguments are the name of the font. Be sure this argument list is terminated.

Name: change font
Symbol name: DPCHFONT
Octal value: 0217
Arguments: Type: numeric Min: 1 Max: 1

Changes the current font to the font associated with the specified font number.

Name: new window
Symbol name: DPNEWWIN
Octal value: 0225
Arguments: Type: string and numeric Min: 5 Max: 5

Creates a window. The first and second arguments are the name of the window. The last four arguments are the coordinates of the lower left and upper right corners of the new window. To create a window with a one character name, the first two bytes carry the name and a null byte. This window is heirarchically below the current window.

Name: enter window
Symbol name: DPENTWIN
Octal value: 0226
Arguments: Type: string Min: 2 Max: 2

Enters the specified window. The interpreter will now evaluate codes in the environment of the specified window. To enter a window with a one character name, the two argument bytes carry the name and a null byte.

Name: exit window
Symbol name: DPEXWIN
Octal value: 0227
Arguments: none

Exits the current window hierarchically upward. The exited window continues to exist.

Name: kill window
Symbol name: DPKILWIN
Octal value: 0230
Arguments: Type: string Min: 1 Max: 2

Removes the specified window from the current window's environment. Only children of the current window can be killed.

Name: delete character right
Symbol name: DPDCHAR
Octal value: 0234
Arguments: Type: string Min: 1 Max: 1

Deletes the character to the right of the cursor. For variable width fonts, the character must be specified. For fixed width fonts, any character can be successfully used as an argument and the action is identical to that of a typical delete-char function. Note that character size is deduced from the current font.

Name: erase character right
Symbol name: DPECHAR
Octal value: 0235
Arguments: Type: string Min: 1 Max: 1

Clears a region the size of the specified character to the right of the cursor.

Name: insert character right
Symbol name: DPICHAR
Octal value: 0236
Arguments: Type: string Min: 1 Max: 1

Inserts the specified character to the right of the cursor.

Name: move over character right
Symbol name: DPMRCHR
Octal value: 0237
Arguments: Type: string Min: 1 Max: 1

Moves right the width of the specified character.

Name: move over character left
Symbol name: DPMLCHR
Octal value: 0240
Arguments: Type: string Min: 1 Max: 1

Moves left the width of the specified character.

Name: delete to beginning of line
Symbol name: DPDBOL
Octal value: 0241
Arguments: none

Deletes to the beginning of the line.

Name: delete to end of line
Symbol name: DPDEOL
Octal value: 0242
Arguments: none

Deletes to the end of the line.

Name: delete line from up
Symbol name: DPDLINU

Octal value: 0243
Arguments: none

Deletes the current line and scrolls lines down from above.

Name: delete line from down
Symbol name: DPDLIND
Octal value: 0244
Arguments: none

Deletes the current line and scrolls lines up from below. This behaves like a typical delete-line function.

Name: insert line up
Symbol name: DPILINU
Octal value: 0245
Arguments: none

Insert a blank line and scroll lines above up.

Name: insert line down
Symbol name: DPLIND
Octal value: 0246
Arguments: none

Insert a blank line and scroll lines below down.