PRL

Technical Note

TN-90-69

WINDOWING CAPABILITY WITH HDS TERMINAL

BY

Surangi Shah, D.R.Kulkarni

1990

Physical Research Laboratory
Ahmedabad

DOCUMENT CONTROL AND DATA SHEET

1. **REPORT NO.** : PRL : TN-90-69

2. **TITLE AND SUBTITLE**

FORTRAN Programmes for windowing facility with HDS terminal (H-window)

3. **REPORT DATE** : June, 1990

4. **TYPE OF REPORT**: TECHNICAL NOTE

5. **PAGES** : 20

6. **PRICE** : Unpriced

7. **NO.OF REFERENCES** : 2

8. **AUTHORS** : Surangi M. Shah and D.R. Kulkarni

9. a) **PURPOSE** : Windowing facility for HDS terminal

b) **USEFUL FOR** : Application programmer (FORTRAN-77)

10. **ORIGINATING UNITS/DIVISIONS AND ADDRESS:**

Library
Physical Research Laboratory
Navrangpura
Ahmedabad 380 009

11. **SPONSORING AGENCY** : Nil

12. **ABSTRACT** : A package which offers windowing capabilities with HDS terminal has been developed and implemented for user of a high-level programming language FORTRAN-77. The package monitors various procedures related to creating different windows as specified by the application programmer. The package is quite robust, user-friendly and easy to use. It can be ported to any computer system to which HDS terminal have been connected. At present it is available to the users of the system DEC-1091.

13. **KEYWORDS** : Window management, HDS terminal, Application programming.

14. **DISTRIBUTION STATEMENT** : Distribution Unlimited

15. **SECURITY CLASSIFICATION** : Unclassified

WINDOWING CAPABILITY WITH HDS TERMINAL

by

Surangi M. Shah, D.R. Kulkarni

ABSTRACT

A package which offers windowing capabilities with HDS terminal has been developed and implemented for user of a high-level programming language FORTRAN-77. The package monitors various procedures related to creating different windows as specified by the application programmer. The package is quite robust,user-friendly and easy to use. It can be ported to any system to which HDS terminals have been connected. At present it is available to the users of the system DEC-1091.

Keywords :

1. Window management
2. HDS terminal
3. Application programming

WINDOWING CAPABILITY WITH HDS TERMINAL

I. Introduction

For any application programmer working with an interactive device such as display terminal, the windowing facility is a great boon. It enables one to present and distribute the Input/ Output properly on the screen. He can see both his input/output simultaneously and properly demarcated on the screen. It helps one to get additional helping prompts to enter data properly. It also presents the meaningful error messages on the screen along with the input and output. Thus the windowing facility increases considerably the productivity of the programmer and also helps him present his output in an attractive manner.

We have developed a software package namely H-WINDOW which offers windowing capability to the application programmer on the HDS (manufactured by Human Designed System) terminal. The HDS terminals are quite versatile and can operate both in ANSI text mode as well as graphics mode. In the ANSI text mode, these terminals can provide built-in windowing capabilities using different escape sequences. In order to make this facility available to the application programmer we have developed this package. Since no windowing package is available on the computer system DEC-1090, this package would be quite useful to all

application programmers using HDS terminals on the DEC computer system. Since the package has been written in a high-level language FORTRAN, it can be easily ported to any system to which HDS terminals have been connected. Before we describe the package we would like to mention the salient features of HDS terminal.

a) The HDS-3200 terminal can be made operational both in ANSI text mode as well as graphics mode.

b) In ANSI text mode it also simulates many other popular terminals VT100,VT220 etc.

c) It's a dual ported terminal.

d) It offers six different windows in the ANSI text mode.

e) The information contained in six windows is actually stored in an ANSI memory consisting of 96 lines. It is possible to assign these memory lines to different windows. If we ensure that the memory lines assigned to different windows do not overlap, the I/O operation performed to any window will not effect the contents of other windows.

f) Besides the six windows, terminal also has a one line window for the message line in the 25th line of the terminal. However this window cannot be manipulated through the program.

These are some of the features of the HDS terminals related to our work. The package uses all these facilities and offers an user-friendly windowing package on DEC system - 1091 in high level language FORTRAN which is widely used in P.R.L. The package consists of 45 subroutines over 1320

lines of FORTRAN code. These subroutines are divided in two parts.

a) Modules which implements HDS primitive operations.

b) Modules developed to offer windowing facilities. These windowing modules use the HDS primitives defined earlier. There are 16 such modules.

In the second section we describe FORTRAN subroutines which implements various HDS - primitives. In the third section we describe the structure of windowing package H-WINDOW. In the fourth section we give the detailed instructions regarding the usage of the package H-WINDOW for the application programmer. Appendix-I gives the list of other subroutines referenced by every module.

II. SUBROUTINES FOR HDS PRIMITIVES

In this section we describe various HDS primitive operations as implemented in high-level language FORTRAN-77. In general the HDS primitive operation is activated by sending a unique string of characters known as 'escape sequence' or 'control sequence' from the CPU to the terminal. We have devloped FORTRAN subroutines which send these well-defined escape sequences to the HDS terminal to exhibit the desired effect. These FORTRAN routines will be utilized to develop the package for offering windowing facilities. The brief description of these subroutines is given below. The string

given in the parentheses against the name of the subroutine represents the corresponding escape sequence. For detailed description for all the escape sequences and control sequences, refer to the programmer's manual for HDS 3200 terminal series.

1. Subroutine ATTOFF (ESC [0m)

   Sets all attributes off

2. Subroutine SELWIN(WIN,IDEV) (ESC [WIN;IDEV-Z)

   Selects window on device

   WIN --> Window number IDEV --> Active device

3. Subroutine CLSWIN (ESC [?2J)

   Clears window screen

4. Subroutine BORDER(BRDR) (ESC [BRDR -t)

   Draws border to window

   BRDR = 0 --> Draw border

   BRDR = 1 --> Erase border

5. Subroutine RWINSC(ON)

   Gives window screen video

   ON = . True. --> Reverse video (ESC [?5h)

   ON = . False. --> Normal video (ESC [?51)

6. Subroutine CRSRM (ESC [?61)

   Controls cursor movement according to window

7. Subroutine AUTWRP

   AUTVAR = . True. --> Autowrap on (ESC [?7h)

   AUTVAR = . False. --> Autowrap off (ESC [?71)

   Autowrap on means on terminal one line contains

either 80 or 132 characters so next character 81 or 133 will be displayed on next line

Autowrap off means 81 or 133 will be displayed in the same line on the position of 80 or 132 character.

8. Subroutine CLSCRN (ESC [2J)

Clears entire screen

9. Subroutine DEFWIN(IDEV,NUM) (ESC [IDEV;1;...6 -q)

Defines window on given communication device

IDEV --> Communication device on which you want to define window

NUM --> Number of windows you want to define at a time. Maximum six windows are allowed.

10. Subroutine FMFD (ESC [=23h)

Form feed

11. Subroutine SWEPER (ESC [0-~)

Resets all windows to default

12. Subroutine BELL

Rings the bell

BELL = char(7)

13. Subroutine SAVRES(DEV,SAVRE) (ESC [SAVRE;DEV-V)

To save/restore window

DEV --> Communication port

SAVRE --> Save or restore function

14. Subroutine COPYWN(FROMWI,TOWI) (ESC [FROMWI;TOWI +q)

Copy of one window into another window

FROMWI --> Window no. from which data is to be copied

TOWI --> Window no. to which data is to be copied.

15. Subroutine WSIZE(LIN1,LIN2,COL1,COL2)

ESC [1in1;lin2;co11;co12-w

Gives size of subjected window

LIN1 --> Starting row number

LIN2 --> Ending row number

COL1 --> Starting column number

COL2 --> Ending column number

16. Subroutine SCRNGE(SCR1,SCR2,NW) (ESC [SCR1;SCR2;NW+W)

Defines screen range of given window

SCR1 --> Starting row of screen

SCR2 --> Ending row of screen

NW --> Window number

17. Subroutine ANSIRG(WN,A1,A2) (ESC [WN;A1;A2-V)

Assigns ANSI memory range

WN --> Window number

A1 --> Starting memory line

A2 --> Ending memory line

18. Subroutine SCROLL(TOP,BOT) (ESC [TOP;BOT r)

Defines scrolling region.

TOP --> Top of scrolling region relative to current window

BOT --> Bottom of scrolling region relative to current window

19. Subroutine LINEDN(REPEAT) (ESC [repeat T)

Scrolls repeat number of times line down.

REPEAT --> Number of repetition

20. Subroutine LINEUP(REPEAT) (ESC [repeat S)

Scrolls number of lines up

REPEAT --> Number of repetition

21. subroutine SCRGUP(REPEAT) (ESC [repeat U)

Scrolls page of screen range up.

REPEAT --> Number of (page) repetition

22. Subroutine SCRGDN(REPEAT) (esc [repeat V)

Scrolls page of screen range down

REPEAT --> Number of (page) repetition

23. Subroutine TOPSCR(LIN,WIN) (ESC [lin;win-s)

Sets top of screen.

LIN --> Number of lines in window WIN that has to appear as the top line in the window's screen range.

WIN --> Number of windows to which the command applies.

24. Subroutine ERSEWN(ERASE) (ESC [ERASE j)

To erase character according to given values.

ERASE = 0 --> Erase from cursor position to end of the window

ERASE = 1 --> Erase from beginning to current cursor position.

ERASE = 2 --> Erase entire window.

25. Subroutine CURSOR(NR,NC) (ESC [NR;NC h)

Positions the cursor to given row number NR and column number NC

NR --> Row number of screen

NC --> Column number of screen

26. Subroutine BOLD(IFBOLD)

Output the following text in bold character

IFBOLD = . True. --> Bold attribute on (ESC [1m)

IFBOLD = . False. --> Bold attribute off (ESC [22m)

27. Subroutine BLINK(IFBLNK)

Blinking for the following text

IFBLNK = . True. --> Blinking attribute on (ESC [5m)

IFBLNK = . False. --> Blinking attribute off (ESC [25m)

28. Subroutine TRSTLN(SCRLIN,LINTYP,VIS)

ESC [SCRLIN;LINTYP;VIS -r Controls the display of terminal's status line.

SCRLIN --> Screen line number where the status and message line are displayed.

LINTYP = 0 --> Use current setting

1 --> Status line

2 --> Message line

3 --> Toggle between status and message line

VIS --> Visibility of status line

= 0 ; Use current setting

= 1 ; Turn off line

= 2 ; Turn on line

29. Subroutine CURTYP(CURTY)

Type of cursor and it's blinking rate

CURTY = 0 --> Fast blinking block cursor

=1 --> Fast blinking underline cursor(ESC[0/})

=2 --> Slow blinking block cursor(ESC[1/})

=3 --> Slow blinking underline cursor(ESC[2/})

=4 --> Solid block cursor(ESC[3/})

=5 --> Solid underline cursor(ESC[4/})

=6 --> No cursor(ESC[5/})

III. STRUCTURE OF THE PACKAGE H-WINDOW

There are seventeen windowing modules in the package. We give brief description of these modules.

1. Subroutine WINDW1

   This subroutine gives the following menu to select

   i) To use new window settings

   ii) To use old window settings

   Enter your choice either 1 or 2.

   A user can opt for old window settings if he/she has already defined his windows in the previous session. With this options the package just restores the earliar windowing specofications stored in a file.

2. Subroutine WINDW2

   Initilizes the specifications used to define window.

3. Subroutine WINDW3

   Accepts specifications to define window and stores it in array variable,writes in an user-defined file name.

4. Subroutine WINDW

If the user opts for the new window setting in the subroutine WINDW1, then this routine asks for the file name he may like to give. The information for creating the file itself will be asked by providing the layout of data-entry windows on the screen for the convenience of the user.

First window will ask the user regarding total number of windows he/she may like to define in his application program. Second window will ask the following specific details for each window such as 1) Title of window 2) Screen range 3) Cursor type 4) Screen VDU etc. Third window works as an on-line help and provides the user the necessary information for proper inputting of data to be fed in the above data-entry windows. If the user opts for old window settings, he/she will be asked to give the name of the file which stores the specification of the windows given in earlier session.

5. Subroutine READ

While using old window settings it is required to read the file containing the information about the windows required by the user. This module reads this information from the file, the name of which is provided by the user.

6. Subroutine WRITE

The sucessful execution of this subroutine will create a file which contains all the specification of the

windows required by the user. While using the new window settings, it is required to store the specifications of the windows offered by the user in a file. This module writes the information in the file, the name of which is provided earlier.

7. Subroutine DWIN

This routine will display a specified window on the screen as per the specification in the file.

8. Subroutine TIT

This subroutine accepts the title for each window. The title contains maximum 25 characters.

9. Subroutine WINNO

There are maximum six windows one can define. This subroutine indicates the window number in 2nd data-entry window so that, the user will know for which window the data is being fed.

10. Subroutine MEMORY

ANSI memory has maximum 96 lines. The memory range is devided into different parts depending on how many windows user may like to define in his application program. If the number of window to be defined is 1 then memory range can be 1 to 96. If it's 2 then range has to be divided in to two equal parts i.e it can be 1-48 and 49-96 etc. This routine divides memory lines into number of parts equal to number of windows required by the user.

11. Subroutine NUMWIN

This routine asks for the total number of windows required by the user in the first data-entry screen window. At the same time, it displays the message in the third data-entry window,warning that the number of windows has to be between 1 to 6.

12. Subroutine CRSRTY

This subroutine accepts the type of cursor to be used for a particular window. There are six different types of cursor. When the user presses ' H' for help, the system will display a window which will give different types of cursor with corresponding associated values.

13. Subroutine VDU

This subroutine accepts the information regarding the type of screen required by window. There are two types of screens possible. The user may press 'N' for normal screen and 'R' for reverse video.

14. Subroutine SCREEN

This subroutine accepts screen range for a particular window. The screen range is specified by giving the row range and column range. It may be noted that the windows which should be displayed simultaneously should not have their screen ranges overlapped. This routine also displays the helpful messages in the third data-entry window for proper entry of the screen ranges.

15. Subroutine SMCPWB

This subroutine converts lower-case letter into an upper-case letter.

16. Subroutine INTGER

This subroutine converts string of numeric character into a corresponding integer number.

17. Subroutine ENTRY

This is main routine of the package. For using package H-WINDOW user has to invoke the routine ENTRY.

These windowing routines make use of the other HDS primitive subroutines described in the section 2. Appendix I gives the list of subroutines referenced by the subroutines described above.

IV. USAGE OF THE PACKAGE H-WINDOW

Using HDS primitives and windowing modules the package H-WINDOW offers windowing capability to the application programmer on HDS terminal. Since usage of FORTRAN-77 is high, this package is written in language FORTRAN-77.

In order to use this package, a user has to invoke the outer most subroutine ENTRY in his application program. This routine asks whether the user intends to create a file for the new specifications of the windows or he/she wants to use an existing file created earlier. If he opts for the existing file,the contents of the file are utilized to present various windows on the screen. If he wants to

create a new file, the new specifications will be asked. In any case he has to mention the file name which he will either use or create.

As described earlier for entering the new specifications of the windows, the system presents the user a screen with three data-entry windows. The specifications of the windows are given in terms of following attributes.

1) No. of windows required

2) The detail of each window as given below

a) Window title b) Screen range

c) Cursor type d) Screen VDU

This information is stored in the file (with a user-defined name) which is later used for displaying your windows. The first window in the data-entry screen will ask for the number of windows desired by the user. At the same time it gives a helping message in the third data-entry window warning the user that the number of windows should be between 1 to 6. Having entered the number of windows desired, the cursor moves automatically to the next field to get the title of the window. Each window is supposed to have a title which should be a string of maximum 25 characters. The next attribute will be the definition of window in terms of screen range. While giving the screen range for various windows, it may be worth remembering that the windows whose screen ranges overlap can not be displayed simultaneously. Further there are certain restrictions for specifying the row and column ranges for different windows.

1) For columns, the range should be between

$$2 <= N <= 79$$

For rows, the range should be between

$$1 <= N <= 24$$

This range allocation enables the package to offer windows with borders.

2) Also the difference between starting number and ending number should be greater than or equal to 3.

3) Entry should be always an unsigned integer

4) Entry should be of maximum two digits.

The package is, of course, robust against invalid data and incorporates data-validation for the ranges fed by the user. The invalid data (i.e. the one which is not in conformity with above restrictions) is immiediatly refused by the package. Also the package enables the user to modify his/her input if required. Next the user has to specify the cursor type that he prefers in the window by giving the appropriate number between 0 and 6. The last attribute of the window is the screen type which may be 'NORMAL' or 'REVERSE'.

This procedure will be automatically repeated so that the attributes of all the windows desired by the user are obtained. Also while entering the information the user gets appropriate and helpful prompts in the third window of the data-entry screen to enable him to feed the proper entry. It also refuses to accept an invalid entry and proceeds further only if the entry is valid. This package is quite

user- friendly and robust against invalid data.

As soon as the data entry for all the windows is over, package clears the screen and shows the first user-defined window along with all other windows which can be displayed simultaneously. At this stage the application program comes out of the package H-window which has now defined the windowing facility as per his requirements.

During the application program, the user may select any of the windows defined earlier before performing the desired I/O operation. The selection of the desired window is possible by invoking the subroutine SELWIN(WIN,IDEV). The argument WIN denotes the window number desired for I/O operation and IDEV is the number of the active device. Usually to get the output on the screen, the IDEV is always 9.

Thus before every I/O operation on the display terminal, the user has to select the suitable window to present his output in a attractive layout desired by way of window definitions.

References :

1. HDS 3200 Programmer's manual,Published by
Human Designed System, U.S.A.

2. HDS 3200 Owner's manual,Published by
Human Designed System, U.S.A.

We describe here the structure of all windowing modules in terms of other subroutines referenced by them.

1. The subroutine ENTRY

   This is the outer most module of the package to be referenced for invoking windowing facility. It references only one subroutine viz. WINDW.

2. Subroutine WINDW

   Subroutine WINDW1 ; Subroutine DWIN ;subroutine DEFWIN

   Subroutine NUMWIN ; Subroutine WINDW2 ; Subroutine WINDW3

   Subroutine READ ; Subroutine SWEPER ; Subroutine SELWIN

   Subroutine CURSOR

3. Subroutine WINDW1(NM)

   Subroutine DWIN ; Subroutine CLSCRN ; Subroutine DEFWIN

   Subroutine DEFWIN ; Subroutine SELWIN ; Subroutine CLSWIN

4. Subroutine WINDW3(NUM)

   Subroutine WRITE ; Subroutine TIT ; Subroutine WINNO

   Subroutine CRSRTY ; Subroutine VDU ; Subroutine SCREEN

5. Subroutine DWIN(NUMBR,WNDO,LRG1,LRG2,

   CLM1,CLM2,RVNR1,CRSR)

Subroutine MEMORY ; Subroutine ATTOFF ; Subroutine WSIZE

Subroutine SCRNGE ; Subroutine RWINSC ; Subroutine CURTYP

Subroutine CRSRM ; Subroutine BORDER ; Subroutine AUTWRP

6. Subroutine TIT(K,TITL)

Subroutine SMCPWB ; Subroutine SELWIN ; Subroutine CLSWIN

Subroutine CURSOR ; Subroutine BOLD ; Subroutine BLINK

Subroutine BELL

7. Subroutine WINNO(K,IUK)

Subroutine SELWIN ; Subroutine CLSWIN ; Subroutine CURSOR

8. Subroutine SWEPER

Subroutine ATTOFF ; Subroutine SELWIN ; Subroutine RWINSC

Subroutine CLSCRN ; Subroutine DEFWIN

9. Subroutine MEMORY(NUM,WIN)

Subroutine ANSIRG

10. Subroutine NUMWIN(NUM)

Subroutine SELWIN ; Subroutine CLSWIN ; Subroutine BOLD

Subroutine BLINK

11. Subroutine CRSRTY(K,CRS)

Subroutine SELWIN ; Subroutine CLSWIN ; Subroutine

BOLD

Subroutine CURSOR ; Subroutine BLINK ; Subroutine SMCPWB

Subroutine CURTYP

12. Subroutine VDU(K,RVN1)

Subroutine SMCPWB ; Subroutine SELWIN ; Subroutine CLSWIN

Subroutine CURSOR ; Subroutine BOLD ; Subroutine BLINK

13. Subroutine SCREEN(K,NUM,LR1,LR2,CL1,CL2)

Subroutine INTGER ; Subroutine SMCPWB ; Subroutine SELWIN

Subroutine CLSWIN ; Subroutine CURSOR ; Subroutine BOLD

Subroutine BLINK

--------- xx ---------