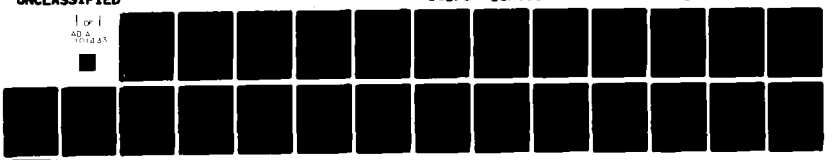
AD-A101 433 ARMY ENGINEER WATERWAYS EXPERIMENT STATION VICKSBURG MS F/G 9/2
TWO-DIMENSIONAL GRAPHICS COMPATIBILITY SYSTEM: VERSION 3.0. DEV--ETC(U)
JUL 78

UNCLASSIFIED DOD/DF-81/005A NL



END
DATE FILMED
8-81
DTIC

CATEGORY E
05-FS-R0200

DDF-81-0057

AD A101433

WATERWAYS EXPERIMENT STATION

Vicksburg, Miss.

DTIC
ELECTE
JUL 16 1981
S D
A

DTIC FILE COPY

This program is furnished by the Government and is accepted and used by the recipient with the express understanding that the United States Government makes no warranties, expressed or implied, concerning the accuracy, completeness, reliability, useability, or suitability for any particular purpose of the information and data contained in this program or furnished in connection therewith, and the United States shall be under no liability whatsoever to any person by reason of any use made thereof. The program belongs to the Government. Therefore, the recipient further agrees not to assert any proprietary rights therein or to represent this program to anyone as other than a Government program.

DEVICE IMPLEMENTATION GUIDELINES FOR 2-D GCS

This document has been approved for public release and sale; its distribution is unlimited.

for the WES Automatic Data Processing Center

81 7 15 069

6

TWO-DIMENSIONAL

GRAPHICS COMPATIBILITY SYSTEM,

Version 3.0.

DEVICE IMPLEMENTATION GUIDELINES.

12 26

18 DoD/IF

19 81/0002a

11 5 Jul[redacted]978

A

03[illegible]

8

Contents

	Page
Introduction	1
Graphics Status Area Initialization	2
Guidelines for Intelligent Devices	21
Implementation Sequence	23

Introduction

The following paragraphs develop guidelines for implementing new devices for 2-D GCS. Included will be the definition of device-dependent Graphics Status Area initialization values, functional specifications for each of the 2-D GCS device-dependent routines, a discussion of factors involved in supporting intelligent refresh devices, and a description of phased sequence for implementating new devices.

Graphics Status Area Initializations

In this section, each of the device-dependent elements of the Graphics Status Area (GSA) will be listed along with the procedures required to calculate the value, if appropriate.

KBEAMX Logical pen x-position in raster units. Default position is lower left corner of largest square on the plotting surface. If the screen is rectangular, the square is right- or bottom-justified.

KBEAMY Logical pen y-position in raster units. Default position is lower left corner of largest square on the plotting surface. If the screen is rectangular, the square is right- or bottom-justified.

TRSVX Ratio of screen to virtual units in x-direction. Default is the width of the UDAREA in raster units divided by the width of the UWINDO (100.).

TRSVY Ratio of screen to virtual units in y-direction. Default is the height of the UDAREA in raster units divided by the height of the UWINDO(100.).

KMINSX Minimum X screen window (UDAREA) boundary in raster units. Default is the same as KBEAMX.

KMAXSX Maximum X screen window (UDAREA) boundary in raster units. Default is the maximum address in the x-direction.

KMINSY Minimum Y screen window (UDAREA) boundary in raster units. Default is same as KBEAMY.

KMAXSY Maximum Y screen window (UDAREA) boundary in raster units. Default is the address of top boundary of largest square on plotting surface whichever is less.

KCOLOR Color or greyscale switch. Default is the actual color produced by the plot head when the device is initialized.

KHORSZ Horizontal hardware character position size in raster units. Default is the width of the smallest size character position.

KVERSZ Vertical hardware character position size in raster units. Default is the height of the smallest character position.

KLMRGN Left alphanumeric margin boundary in raster units. Default is normally zero (minimum plotting surface address in x-direction).

KRMRGN Right alphanumeric margin boundary in raster units. Default is the maximum plotting surface address in y-direction.

KTMRGN Top alphanumeric margin boundary in raster units. Default is the maximum plotting surface address in y-direction.

KBMRGN Bottom alphanumeric margin boundary in raster units. Default is normally zero (minimum plotting surface address in y-direction).

KGIN Graphics input device selector. Default is the primary graphics input device on the display.

KMINDX Minimum x-address of display surface in raster units. Default is zero.

KMAXDX Maximum x-address of display surface in raster units. Default for devices with variable size plotting surface is largest square using normal plotting medium.

KMINDY Minimum y-address of display surface in raster units. Default is zero.

KMAXDY Maximum y-address of display surface in raster units. Default for devices with variable size plotting surface is largest square using normal plotting medium.

TRASTX X-resolution in raster units per inch. Default is KMAXDX divided by width of plotting surface in inches.

TRASTY Y-resolution in raster units per inch. Default is KMAXDY divided by height of plotting surface in inches.

KTYMNU Device type for menu generation. Possible values are: Graphics device = 1, Alphanumeric device = 2, Batch device = 3.

KILS ILS device indicator. Possible values are: Intelligent device = 1, Unintelligent device = 0.

KBAUDR Speed of communication line in characters per second. Default is usual speed for device.

KPCHAR X,Y component buffer. Used by GCSPEN as a work area. Default should reflect an impossible or unusable value.

KPREVC Color which terminal is currently generating. Default is the same as KCOLOR.

UDIMEN(XMAX,YMAX)

XMAX Defines the maximum extent of a variable display surface dimension along the x-axis.

YMAX Defines the maximum extent of a variable display surface dimensions along the y-axis.

Comments

This routine is used to set the size of a variable plotting surface such as CALCOMP drum plotters. The value contained in each dimension which varies is converted from current device units to raster units and tested for legal value. If the new dimension is legal for the device, the resulting raster unit value is stored in KMAXDX or KMAXDY as appropriate.

For display-devices with a fixed size plotting surface, this routine need only set the UDAREA boundaries to the entire plotting surface. This should be done if the plotting surface is variable also.

This routine is not implementable at the display-device.

UERASE

Comments

This routine provides a clean plotting surface by either erasing the plotting surface or advancing the plotting medium to an unused frame. In implementing this routine, care must be taken to insure that the current pen/beam position remains unaltered but in the new plotting surface. This may require a call to the GCSPEN routine after the "erase" occurs.

This function may be supported at the display-device. In this case, only an indication that the erase must occur need be transmitted to the device. The beam position must still be insured.

URESET

Comments

This routine returns the Graphics Status Area to its initial condition. Aside from setting the device-dependent GSA elements to agree with those in the BLOCK DATA subroutine for this device before compilation, no further modifications are required.

This routine can not be supported at the display-device.

UWAIT(TIME)

TIME Indicates the amount of time which GCS will wait before sending further commands to the display.

Comments

This routine is usually implemented by sending a sufficient number of null-operation control characters across the communication line to cause the required delay.

The KBAUDR GSA element specifies the line speed in characters per second.

For batch devices, this routine is a null operation.

This routine can not be supported at the display.

GCSALF(ICHAR)

ICHAR Is one ASCII character, right-justified, zero-fill.

Comments

This routine causes one hardware character to be displayed after the terminal is placed in alpha mode and KKMODE is set to reflect alpha mode. If a hardware character generator is not available and if the device is not intelligent, a device-independent hardware character simulator version of GCSALF is available. If a hardware character generator or simulator is available at the display, the printable ASCII character is simply sent to the display. Only displayable ASCII characters are output from this routine. The receipt of control characters implies that the terminal be placed in alpha mode and that the character size be updated if necessary. Two control characters have special extra meaning for intelligent displays. A value of 001_8 indicates start of alpha string and 002_8 indicates end of alpha string. This allows strings of characters to be processed as one primitive operation at the display.

To update the character size, on input the value of KSIZEF is compared to the value of KPREVS. If their values agree, then no size change need be processed. If their values disagree, the value of KSIZEF is stored in KPREVS and the new sizes are stored in KHORSZ and KVERSZ.

This function may be supported at an intelligent display-device either by sending single characters or by using 001_8 and 002_8 brackets to send character strings.

GCSARC(X,Y,ANGLE,RADIUS,ICHECK)

X,Y — Are the coordinates of the center of the circle defined by the arc.

ANGLE — Indicates the angle the arc will subtend and the direction of travel of the arc. A positive angle is counterclockwise; a negative angle is clockwise.

RADIUS — Specifies the radius of the circle.

ICHECK — Is set to 1 if the arc is not drawn by GCSARC. It is set to zero otherwise.

Comments

If hardware arc generation is available and if the arc can be drawn and does not intersect the window, the flag is set to zero and the appropriate instruction is sent to the display. If, for some reason, the arc can not be drawn at the display, the flag is set to 1. Reasons which might apply are lack of an arc generation facility at the device or intersection of the arc with the window.

This function may be supported at the device if the arc to be drawn does not intersect the window. If it does intersect the window, GCSARC may calculate the visible arc segments and issue several instructions to draw each segment. However, the instructions to perform these calculations are somewhat complex and it is usually left to UARC to draw the desired arc.

GCSBPK(IX,IY,ICHAR)

IX,IY Is the location to be marked by the character or dot.

ICHAR Is the ASCII code fill character for the designated raster position.

Comments

This routine maintains the internal raster buffer for raster devices such as line printers and alphanumeric terminals without cursor addressing. The buffer is a FORTRAN Common Area GCSBUF which contains storage for the entire frame or a page of the frame. The IX,IY values are mapped into a buffer character position and the character (after translation into internal computer code) is inserted in this position.

This routine only does packing. Flushing of the buffer is handled by GCSPEN.

This routine may be supported at the display-device if an intelligent raster-scan terminal or an unintelligent raster-scan terminal with local storage is being implemented. In this case, this routine should be eliminated and GCSPEN should send the packing request directly to the terminal.

GCSDSH(IX,IY)

IX,IY Raster unit location of the end of the dashed line.

Comments

This routine draws a dashed line based on the current dash specification contained in the KDASHT GSA variable. KDASHT values between one and nine indicate selection of various hardware dash modes. In none are available, a standard dash is simulated. A value of nine refers to hardware dotted line mode if available.

If a hardware dash is available, a call is made to GCSPEN with an opcode value of four to draw the line. If a hardware dash is not available or not requested, the desired dash sequence is simulated by calling GCSPEN to draw alternating solid and invisible lines.

This function is not supportable at the display-device.

GCSBEG

Comments

This routine performs any initialization functions required by the display. For intelligent devices, it may pass the initialization request to the device.

GCSEND

Comments

This routine performs any termination functions required by the display-device. For intelligent devices, it may pass the termination request to the device.

GCSFRA(INDEX,OPCODE)

INDEX Is the numeric identifier for the frame.

OPCODE Is the action to be performed related to the designated frame.

Comments

This routine processes the frame manipulation function requests of "open", "close", "show", and "unshow". Since these operations are currently supported only in an intelligent display, this function becomes a null routine for unintelligent devices. The function of this routine is primarily to reformat the requested operation as necessary and transmit it to the display.

This is an interface routine which communicates with a program executing in the display-device general processor if the display is intelligent.

GCSIN(ICOUNT,IBUFR)

ICOUNT Indicates the upper limit of the number of ASCII characters to be received as input.

IBUFR Is a buffer in which the ASCII characters will be placed right-justified, zero-fill.

Comments

This is primarily an interface routine to the operating system function which actually obtains the character. If less than ICOUNT characters are received, the remaining buffer positions should be zeroed.

This function can not be supported at the display since it is really not device-dependent but operating system dependent. However, to insure correct linkage editing on all operating systems, it is grouped with the device-dependent routines.

For batch devices, this routine issues an error message and returns null characters.

GCSINN(ICOUNT,ISTRING)

ICOUNT Indicates the number of ASCII characters to be read from the terminal.

ISTRNG Is a buffer into which the ICOUNT characters will be placed.

Comments

This routine accepts strings of ASCII characters from the terminal. Prior to issuing an input operation using GCSIN, the terminal is placed in the appropriate mode to activate the keyboard input. If ICOUNT is less than or equal to zero, no input operation takes place.

This function can not be supported at the display.

GCSOUT(ICOUNT,IBUFR,IRESP)

ICOUNT Indicates the number of ASCII characters to be transmitted to the display.

IBUFR Is the buffer containing ICOUNT ASCII characters right-justified, zero-fill.

IRESP Is a count indicating the number of acknowledgement characters required from the terminal. This is only used for supporting intelligent devices.

Comments

This is the interface routine between GCS and the operating system routine which sends characters to the terminal. It is called by all GCS routines which need to transmit to the display. If ICOUNT = 0, the operating system is requested to flush any internal buffers.

This routine can not be supported at the display.

For batch devices, this routine performs only one function: It intercepts GCS error messages and displays them on the system print file.

For supporting intelligent displays, the IRESP arguments can be used to request an acknowledgement from the device before further characters are sent. It is usually used when sending functions which require considerable execution time at the display.

GCSPEN(IX,IY,IPEN)

IX,IY Is the end point of the line to be drawn in raster units.

IPEN Indicates the operation to be performed. Zero specifies a move; one specifies a solid line; two specifies a point; three specifies a buffer flush request; and four specifies a dashed line.

Comments

This routine is the basic line drawing routine for each device. Several variables have been set aside in the GSA for use by this routine. The array KPCHAR is for use as an internal work space and usually is where the command to be sent to the display is constructed. KPREVX and KPREVY are updated with the value IX and IY to always point to the current beam position when GCSPEN terminates. GCSPEN may also need to fetch additional information to insert intensity, blink, or other information into the command. The variable KKMODE is updated to reflect any mode changes initiated by GCSPEN. If the display-device is such that a separate invisible line mode is used, GCSPEN will set KMOVEF to one whenever he will exit with the display-device primed for a blank line. Otherwise, KMOVEF is set to zero.

This routine can be supported at an intelligent display-device. If such is to be the case, GCSPEN must still process buffer flush requests, must check for redundant "moves" to the current beam position, must maintain KKMODE settings as necessary, and reformat the line drawing instruction to be sent to the display.

For terminal devices, output from GCSPEN is sent through GCSOUT. For batch devices, output is written on the appropriate files either directly or using a device-specific support package.

Note: GCS assumes that the plotting surface is addressed from zero to some maximum number. If the actual address range contains negative numbers, it is the responsibility of GCSPEN to add in the appropriate displacement before building the display processor commands.

GCSSET(OPTION)

OPTION Is a standard form (four-character, left-justified, blank-fill) USET option passed from USET.

Comments

This routine performs hardware-supported mode setting. All USET requests are passed to GCSSET. If a particular option requires immediate mode setting at the display, it is recognized by GCSSET and the appropriate request is sent to the display.

This function can not be supported solely at the display. However, intelligent displays which support some mode setting will require a parallel routine to receive the requests sent by GCSSET.

GCSWHR(IX,IY,IC)

IX,IY Will contain the raster unit coordinates of the input cursor position.

IC Is a mode, status, or option character obtained during the input operation. If IC = 0 upon entry to GCSWHR, only the character is requested.

Comments

This routine performs "positioning"-type input operations from the input device specified by GSA variable KGIN or the closest available equivalent. Whatever the form of the input coordinates when received, they must be converted to standard raster unit format. The mode, status, or option character is returned as received to the calling routine.

This routine can not be supported at the display, but may require a parallel function to execute in an intelligent device.

Guidelines for Intelligent Devices

Display-devices which have computing power of their own may be termed intelligent. Appropriate use of the intelligence can greatly increase the capabilities of the entire system. Most such display-devices have a refresh capability. Thus much of the capacity of the general processor of the capability of the general processor of the device is involved with display list management.

In GCS, a display list management facility has been provided to support dynamic pictures in a device-independent manner to the extent that the program will execute even while using dynamic instructions on a non-refresh device. This is the framing facility. A more commonly used name for this feature is transformed, segmented, unstructed display list. Four functions are suppported for these frames: open (UFRAME), close (UFREND), make visible (USHOW), and make invisible (UNSHOW). These four routines all call the device-dependent routine GCSFRA with the number of the frame to initiate the requested operation. GCSFRA passes these requests to the intelligent terminal. If no named frame is open, an unnamed frame is maintained which is always visible.

In the display, an open operation starts up a new segment for the indicated frame number. If a segment for that frame is already being displayed, a double buffering takes place which continues showing the elder of the two frames with identical numbers until the newer frame definition is complete (indicated by receiving a close (UFREND) command). At this time, the elder frame is discarded and the newer frame is displayed. A frame may be constructed either visibly or invisibly. If a frame is constructed visibly, it will be shown when completed. It can then be made invisible by executing an UNSHOW command. If a frame is constructed invisibly, it will not be displayed until a USHOW command is received. Subsequent USHOW and UNSHOW commands can turn off and on defined frames as requested.

The logical organization of the display list is as above. The actual physical organization and technique of memory management will depend on the particular display-device. Some will more naturally use a paging memory management scheme; others may use some type of linked list.

In addition to display list management routines, the intelligent device general processor will also need to interface available hardware functions with the device-dependent routines of GCS and to provide simulators as desired

for device-dependent functions not available in the particular hardware under consideration.

In designing the format in which the device-dependent GCS routines communicate with the display-device, it is wise to select an encoding scheme that does not violate ASCII coding conventions. This requires that only seven bits be used as data and the parity-bit reserved for parity or marking even if the particular operating system does not require it. Following this procedure will allow connection of the terminal with the host-computer over standard communication lines without special communications software.

Implementation Sequence

GCS is a modular system. Since many of the devices supported by GCS are relatively unsophisticated, many of the GCS device-dependent libraries contain simulators for many of the hardware functions. By using these simulators, it is possible to shorten the initial implementation time for new devices allowing later replacement of the simulators as time permits.

The following scheme for phased implementation of new devices has been designed to maximize the benefits mentioned above.

Phase 1: Identify the required initialization values for each device-dependent GSA variable and insert them in the Block Data and URESET subroutines.

Phase 2: For intelligent devices, design the communication data structure to be used between the device-dependent GCS routines and the program executing in the display-device. This program should be developed in parallel to the device-dependent GCS routines in the remaining phases if not already available.

Phase 3: Implement the GCSBEG, GCSSET, and GCSEND functions if necessary.

Phase 4: Implement the UERASE and GCEPEN routines. The GCSOUT routine can usually be used without modification from one of the other libraries. Simulators or null routines can be used for the remaining functions.

Phase 5: Implement the GCSALF routines if a hardware character generator is available. This would replace the hardware character simulator.

Phase 6: Implement the GCSFRA routine, if the terminal is intelligent. If not, the null version of the GCSFRA routine is appropriate.

Phase 7: Implement GCSIN, GCSINN, and GCSWHR functions if the device is interactive. These would then replace the null routines which probably were used until this phase.

Phase 8: Implement all other functions which do not yet support the capabilities of the device.

By following the sequence above, experience has shown that implementation of new devices should proceed smoothly.

DA
FILM
8