XEROX COMPUTER GRAPHICS

Timothy K. Dudley
Xerox Corporation
El Segundo, California
USA

ABSTRACT

Xerox Corporation has developed a graphics system capability which is comprised of a graphics language, GL-1, and a graphics tube (with optional accessories such as light pen, function keyboard, etc.) interfaced to an intelligent 'programmable' controller, which in turn interfaces to a standard IOP on a Sigma computer. This paper briefly describes the overall hardware system, dealing specifically with the Vector General Graphics Display System. It deals primarily with the GL-1 software, which may be classified into nine groups: initialization and termination routines, device routines, logical structuring routines, vector and character routines, mode setting commands, attention handling routines, control commands, keyboard data entry, and error handling.

RESUMÉ

La Xerox Corporation a réalisé un système graphique consistant en un langage graphique, GL-1, un tube cathodique (avec les accessoires à option comme un crayon éléctronique, un clavier à fonctions, etc.) connecté a un 'contrôleur intelligent' qui peut être programmé et qui, à son tour, est connecté à un calculateur-entrée-sortie IOP standard d'un ordinateur Sigma. Dans cette communication, on décrit brièvement le système d'équipement global et, plus spécifiquement, le Vector General Graphics Display System. On discute principalement la programmation GLI qui est divisée en neuf groupes: programmes pour mettre à la valeur initiale et pour terminer, programmes pour contrôler les appareils, programmes pour former les structures logiques, programmes pour représenter les caractères et les vecteurs, instructions pour disposer la mode d'opération, programmes pour manier l'attention, programmes pour commander la contrôle, programmes pour l'entrée de données par clavier, programmes pour résoudre des erreurs.

XEROX COMPUTER GRAPHICS

Timothy K. Dudley
Xerox Corporation
El Segundo, California
USA

INTRODUCTION

Xerox Corporation has developed a computer graphics capability comprised of a graphics language, GL-1, and a graphics tube with optional accessories such as light pen, function keyboard, etc., which is interfaced to an intelligent controller which in turn interfaces to a standard IOP on a Sigma computer. The system is executable in the background, foreground, or under time-sharing. GL-1 is re-entrant and runs on the Sigma 6, 7, and 9 computers under UTS. This paper deals with ghe GL-1 system running in conjunction with the Vector General Graphics Display System on a Sigma 6 under UTS.

HARDWARE DESCRIPTION

The overall hardware configuration is shown in Figure 1. The main computer is any standard Sigma 6, 7, or 9 configuration. The 'programmable' controller is an intelligent controller known as the System Control Unit (SCU). The graphics controller in this case is the VG display system.

VECTOR GENERAL DISPLAY SYSTEM

The VG display system is a CRT display with an interface unit, a display controller, a dual D/A converter, and a vector generator. Optional features include a character generator and three coordinate transformation generators: 2D, 2D with rotation, and 3D with rotation. Any of six interactive control devices may be connected to the system including a joystick, four A/N keyboards, sixteen or thirty-two function switches with manual interrupt, a data tablet, ten control dials, and four light pens. The character generator generates any one of four standard sizes in horizontal or vertical format, selected by the user. The 2D option provides hardware scaling and translation, 2D with rotation adds rotation in a single plane, and the 3D option generates three-dimensional constructs and displays them with scaling, translation, and rotation about any axis.

The refresh rate of the CRT is 60, 40, or 30 times a second, or asynchronously. The resolution is 4096 x 4096 over a fifteen inch

square display area, with thirty-two levels of beam intensity. Limited hardware clipping is made possible as follows: displacement registers allow the picture described in the image space (see Figure 2) to be shifted into the maximum picture space without distorting any vectors.

One VG controller may control up to four CRTs, four sets of function keys (sixty-four total), four A/N keyboards, four light pens, one data tablet, one joystick, and one set of ten control dials.

SYSTEM CONTROL UNIT

The System Control Unit (SCU) is essentially an intelligent programmable controller. The refresh and the refresh buffer are under SCU program congrol. The SCU handles attentions from the VG controller, and also handles communication protocol with the main computer. There is also limited interaction between the user and the display via the SCU without involving the main computer. The SCU is interfaced through a 7907 controller and a standard IOP channel. One VG controller is connected to one SCU, which is connected to one IOP channel.

SOFTWARE DESCRIPTION

Figure 3 shows the overall software structure and its relationship to the hardware. There are three pieces of software which are defined by function: the Graphics Control Program (GCP), the Graphics Access Method (GAM), and GL-1.

GRAPHICS CONTROL PROGRAM

The GCP resides in the SCU. It communicates to GL-1 via a 7907 and interface and the IOP. The SCU contains memory for the refresh list of the CRT(s), and the refresh is under GCP control. When data is received from GL-1 for insertion into the display list, the GCP does the insertion. It also responds to attentions (interrupts) from the VG controller and extracts appropriate information (such as what type of device caused the interrupt, where was the refresh currently executing, etc.) for transmittal to GL-1. The GCP also has facilities for queueing attentions.

GRAPHICS ACCESS METHOD

The GAM is part of the operation system and is essentially a graphics device handler. It provides features for defining a graphics device to the operating system, for transferring control to GL-1 in response to a graphics device interrupt, and for GL-1 to read from and write to the graphics device.

GL-1

GL-1 is not a language as such, but is rather a set of FORTRAN-callable subroutines which allow a user to interact with a graphics device. It is completely re-entrant. One user program may access one or more graphics devices, or any number of user programs may each access any number of graphics devices. However, more than one user may not access the same device. GL-1 may reside in the public library, user library, or it may be loaded with the user program. It is monitor independent, and can be modified to accomodate different

graphics devices such as storage tubes and raster-type devices. Each user of GL-1 has a separate context area which is always associated with him and contains information pertinent to his program.

The GL-1 calls are functionally independent...they are simple calls with a minimum number of arguments, and each call performs only one function. The calls are hopefully self-documenting. As will be seen, GL-1 provides the basic tools to interface to a graphic device attached to a Sigma computer. It is based on a table structure which can easily be expanded to incorporate new features or to correct oversights. It also maintains a hierarchial data structure for creating, omitting, and displaying graphic images, in addition to doing memory management for both the Sigma memory and the SCU memory.

The GL-1 calls which are available to the user can be classified into nine groups: initialization and termination, device routines, logical structuring routines, vector and character routines, mode-setting commands, attention handling routines, control commands, keyboard data entry, and error handling.

INITIALIZATION AND TERMINATION ROUTINES

BGNGL(AREA,SIZE)

where AREA is the address of the user context area, and SIZE is the size of the context area. This routine initializes a work area in the user program for GL-1 to use. It sets up a free space chain for memory allocation, clears the user context area to zeroes, and stores the context address in the user's TCB. This routine must be included in every program which uses GL-1.

TRMGL

This routine closes all the graphics devices associated with the user and zeroes the address of the context area in the TCB. It effectively sinks the ship.

GRAPHICS DEVICE ROUTINES

BGNGD(DCBUNIT[,HATID,SATID])

where DCBUNIT is the unit number assigned to the particular graphics device, HATID is the pointer to the Hardware Attribute Table, and SATID is the pointer to the Software Attribute Table. This routine is called to define a graphics device to the system and must contain a DCB unit number. It enables the user program to write to a specific graphics device. The HAT/SAT tables localize the hardware definition of the device and the memory configuration of the SCU. If multiple device configurations exist at an installation, the appropriate HAT/SATs are assembled as separate modules and placed in the GL-1 library.

SELGD(DCBUNIT)

where DCBUNIT is as described above. This call enables the user to select the device on which he wishes to display his picture.

TRMGD(DCBUNIT)

where DCBUNIT is as described above. This call closes the specified DCB to allow it to be used by other program elements.

LOGICAL STRUCTURING ROUTINES

A collection of GL-1 orders is called a block. A block may be a single contiguous set of memory words or a collection of these sets linked together. In either case, the block name refers to the entire collection of orders. Compound pictures are formed by connecting blocks to other blocks, yet blocks retain their identity even when connected to other blocks. Operations are available for creating and returning blocks to available space through the storage allocator. All elements within the block have the same input data type and scale factor. A block is named and may be displayed, deleted, modified, and/or used as a subroutine.

BGNBLK(NAME,[SIZE])
where NAME is the name of the block, and SIZE is the size of the block buffer in bytes. If SIZE is not included, 128 is assumed. This call begins the creation of a new block. All orders which follow this call and preceed the ENDBLK call will be associated with this named block.

ENDBLK(NAME)
where NAME is as above. This call ends creation of a block and places the block of graphic orders in the display buffer, ready to be displayed.

DSPBLK(NAME)
where NAME is as above. This routine causes the named block and its associated subroutines to be displayed. Any number of blocks may reside in the refresh buffer, but only one block and its associated subroutines may be displayed at one time.

INCBLK(NAME)
causes the named block to be included in the displayed image. It does this by causing a branch order to be inserted into the display list which branches to the block.

OMTBLK(NAME)
causes the named block to be omitted from the displayed image. In this case, the branch order in the display list branches around the block.

SELBLK(NAME)
selects and opens a previously defined block. This call is used for including, altering, or omitting a named element within a block which is no longer current.

LNKBLK(NAME)
connects the current block to the named block. The named block then becomes a subroutine to the current block and may be displayed or not by using the INCBLK or OMTBLK calls.

EXTBLK(NAME)
reopens an existing but previously ended block to enable the user to add graphic orders to it. It is closed again with ENDBLK.

RSTBLK(NAME)
releases the refresh buffer associated with the named block and begins creation of a new block to be identified by NAME. The new block must be of equal or shorter length than the old block. The block buffer size specified in the original BGNBLK is used.

TRMBLK(NAME)
releases all memory and the refresh buffer occupied by the specified block. After this call, the block no longer exists.

VECTOR AND CHARACTER ROUTINES

GL-1 contains subroutines for constructing each type of graphics display unit command and adding it to a specified block. If no space remains in the block, memory management will allocate additional storage. Commands include moving the beam, drawing lines, plotting points, drawing characters, and naming elements.

AMOVE(X,Y)
moves the beam to screen coordinate (X,Y). This has the same effect as CALL PLOT(X,Y,3) in the CALCOMP system.

MOVE(X,Y)
either moves the beam to screen coordinate (X,Y) or moves to the current beam position plus (X,Y), depending on the setting of a preset mode (see mode description below). This call corresponds to either an absolute or relative move.

DRAW(X,Y[,NUMBER])
where X is a list of x-coordinates, Y is a list of y-coordinates, and NUMBER is the number of elements in both lists. DRAW moves the beam with beam on, drawing as it moves. The X and Y lists may be absolute positions or relative displacements, depending on the preset mode setting. If NUMBER is missing, 1 is assumed, and the call becomes similar to CALCOMP'S CALL PLOT (X,Y,2)

PLOT(X,Y[,NUMBER])
where the arguments are as described in DRAW above. The only difference between PLOT and DRAW is that PLOT merely spots the points and no connecting lines are drawn. Again, if NUMBER is missing, it is assumed to be 1.

DRSGMT(XST,YST,XND,YND[,NUMBER])
where XST is a list of x-coordinates of the starting points of the line segments to be drawn, YST is the list of y-coordinates of the starting points, XND is the list of x-coordinates of the endpoints, YND is the list of the y-coordinates of the endpoints, and NUMBER is the number of elements in each list. This routine enables the user to draw sets of line segments. If number is absent, it is assumed to be 1.

PUTTXT(TEXT[,NUMBER])
where TEXT is a literal string or the address of a literal string, and NUMBER is the number of characters in the string.

This call allows the user to put text onto the screen. If NUMBER is absent, it is assumed to be 1.

NAMELT(NAME)
An element is defined as a graphic order and its associated data. Elements can be named, then altered, included, and omitted in much the same manner as blocks. The routine NAMELT takes the graphic order immediately following the call and builds an element table for it, then includes that table in the element chain.

INCELT(NAME)
includes the named element in the displayed image in the same manner as INCBLK.

OMTELT(NAME)
omits the named element in the displayed image in the same manner as OMTBLK.

ALTELT(NAME)
replaces the named element with the graphic order which immediately follows the ALTELT call.

ORDER(ORDER)
allows the user to construct a graphics order and send it to the GCP. Very dangerous.

MODE SETTING COMMANDS

GL-1 contains operations for setting modes for blinking, intensity, character size and orientation, scaling, vector type, vector mode, blanking the beam, and modes for the light pen, A/N keyboard, and function keyboard.

SCALE(USRLMT[,SCRLMT])
where USRLMT is a four-word list specifying the limits of the user's data in user units, and SCRLMT is a four-word list specifying the screen limits in raster units. If SCRLMT is absent, full screen limits are assumed. If SCALE is not called, all input is assumed to be in raster units. Both lists have the following format:

word 1: x lower limit
word 2: y lower limit
word 3: x upper limit
word 4: y upper limit

VTYPE(TYPE[,AUTO])
where TYPE is on of "ABS", "REL", "INCL", or "INCH", and AUTO is either "X" or "Y". This routine sets a mode flag in the current block table. This affects such routines as MOVE and DRSGMT as to whether their lists contain actual coordinates or displacements. The data being passed to the SCU may be a time plot in which case the x values could be automatically incremented by a specified amount, which would necessitate only a y list, a starting x value, and x increment. This would save considerable space in SCU memory. Similar arguments hold for

auto y. The high order bits or the low order bits of either x or y may also be automatically incremented. Defaults are noe autoincrement and ABS mode.

VMODE(MODE)
where MODE is one of "LINE", "DASH", OR "DOT". Default is "LINE".

INTENS(INTS)
where INTS is an integer between 1 and 31. This selects the relative intensity of the beam. Default is 15.

BLINK
begins the blink mode. The selected data will blink at a rate of approximately twice per second.

NBLINK
terminates the blink mode.

BEAM, NBEAM
turns the beam on (BEAM) or off (NBEAM).

CHRSZE(HEIGHT)
where HEIGHT is one of .1, .2, .3, or .5 and produces characters of the following heights:

.1	yields	characters	.124523	inches	high (default)
.2	"	"	.183105	"	"
.3	"	"	.249023	"	"
.5	"	"	.468750	"	"

CHRORT(ORT)
where ORT is either 0 or 90 and specifies degrees of counter clockwise rotation of following character strings. Default is zero.

LPMODE(MODE)
where MODE is "SNGL" or "MULT". This routine sets up the SCU to handle light pen interrupts. If MODE is "SNGL", only single interrupts are recognized, and the light pen must be reset before another interrupt can be recognized. (See Attention Handling Routines.) If MODE is "MULT", then all light pen interrupts are acknowledged and queued. The number of multiple attentions allowed from the light pen is a parameter in the HAT/SAT tables. Default (and safer) mode is "SNGL".

KBMODE(MODE)
where MODE is as above. This routine works the same way as LPMODE except it is for the A/N keyboard.

FKMODE(MODE)
where MODE is as above. Works the same way as LPMODE except it is for the function keyboard.

ATTENTION HANDLING ROUTINES

The light pen, alphanumeric keyboard, and/or function keys may

be selectively enabled, reset, or disabled. The user program may request GL-1 to wait for an attention, or attentions can be queued in the SCU and returned when requested in the same order in which they occurred.

ENBLLP
  enables the light pen to allow it to cause interrupts.

RSETLP
  resets the light pen to allow more interrupts to be caused.

DSBLLP
  disables the light pen.

ENBLKB
  enables the A/N keyboard to allow it to cause interrupts.

RSETKB
  resets the A/N keyboard.

DSBLKB
  disables the A/N keyboard.

ENBLFK(KEY_i[,KEY_j...])
  enables the function keys selec ted in the parameter list.

DSBLFK(KEY_i[,KEY_j...])
  disables the selected function keys.

RSETFK
  resets all enabled function keys.

DSBLAL
  disables all attention sources.

CLRATN
  clears the attention queue in the SCU.

CHKATN(INFO)
  where INFO is a 16 word list describing the attention. This routine reads the attention queue from the SCU and builds the INFO table. This table contains such information as the DCB number of the interrupting graphics device, a mnemonic describing the type of attention, block and element names which were current when the interrupt occurred, function key pattern if the interrupt was caused by function keys, keyboard character if caused by the A/N keyboard, and (X,Y,Z) coordinates (in user units) of the light pen hit if caused by the light pen. This table is made available to the user. If no interrupt has occurred, the type of attention will be zero. Execution of the user program proceeds immediately after the call to CHKATN.

WTATN(INFO)
  where INFO is as described above. This routine is identical

to CHKATN except that execution of the user program is not resumed until an interrupt has occurred.

CONTROL COMMANDS

There are merely the START and STOP commands to the display controller.

KEYBOARD DATA ENTRY

GETTXT(ARRAY,NUMBER, ENDCOD)

where ARRAY is the name of the area in the user program where the text string is to be stored, NUMBER is the number of characters requested, and ENDCOD is returned by GL-1 as either "EOB" or "CMPL". "EOB" indicates that the end of block was hit before the correct number of characters was passed, and "CMPL" indicates that the requested number of characters was read.

ERROR HANDLING

GL-1 forgives the user and proceeds whenever possible. Anytime an error occurs in a GL-1 routine, a code is posted in the user's context area and also output to the LO device with a message. The error code indicates the type of error and which GL-1 routine posted it. If the error is irrecoverable, uncorrectable, or prevents GL-1 from proceeding, the job is aborted. In most cases, however, defaults are assumed and the user program is allowed to continue. The user can find out how many errors have occurred and what the first one was by calling the following routine:

ERROR(CODE,COUNT)

where CODE is as describe above and COUNT is the number of errors which have occurred since the call to BGNGL or the previous call to ERROR, whichever occurred later.

OTHER SUPPORTING ROUTINES

In addition to the routines described above, there are five basic subroutines which manage memory and data buffers, build and search chains of blocks and elements, and a routine which builds and maintains block tables and block buffers. There are also eight subroutines which provide supporting functions including argument handling, error reporting, initialization, and conversion routines to convert from real to integer, integer to real, ASCII to EBCDIC, EBCDIC to ASCII, and hexadecimal to EBCDIC.

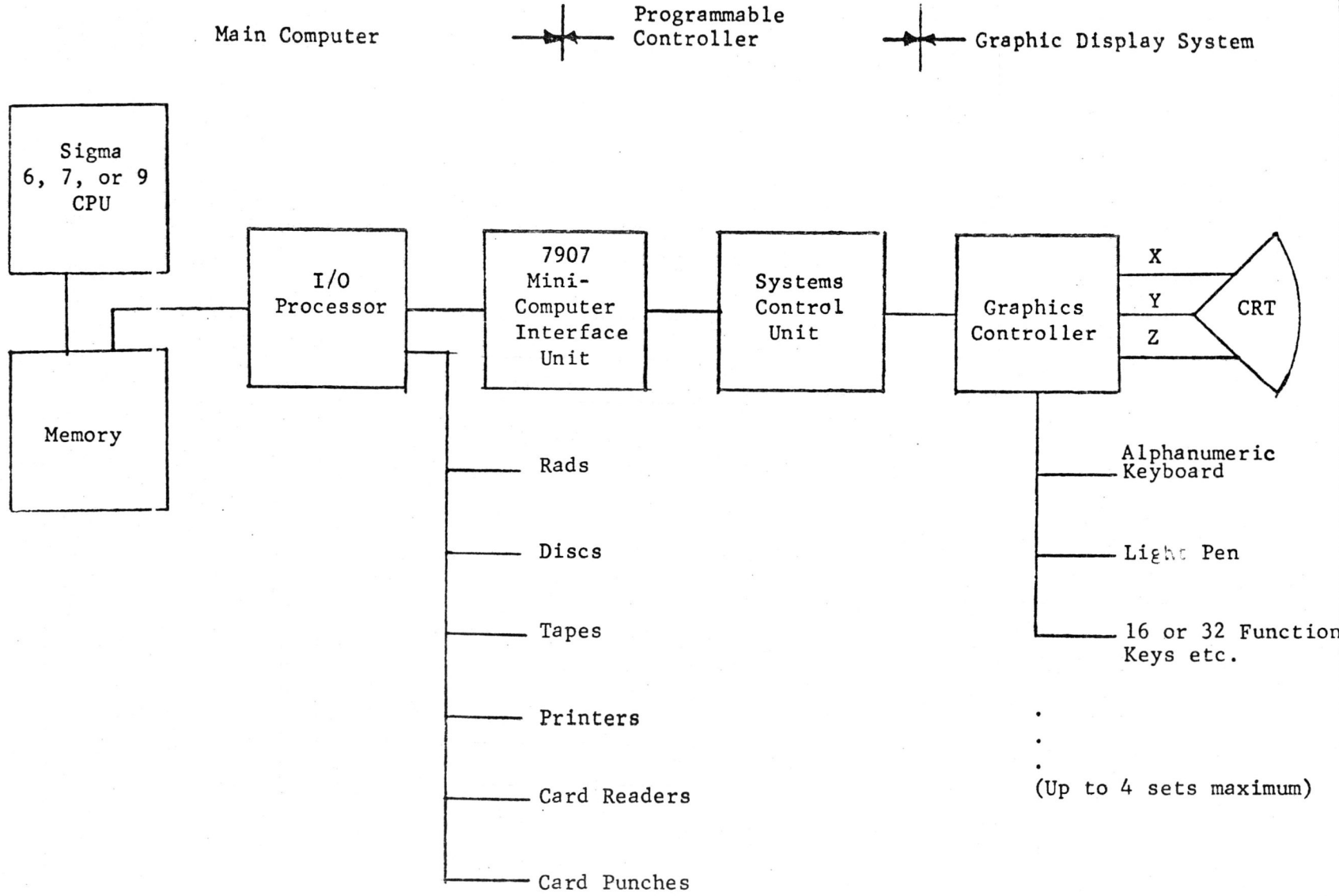

FIGURE 1. HARDWARE CONFIGURATION

TITLE

SHEET . OF

XEROX

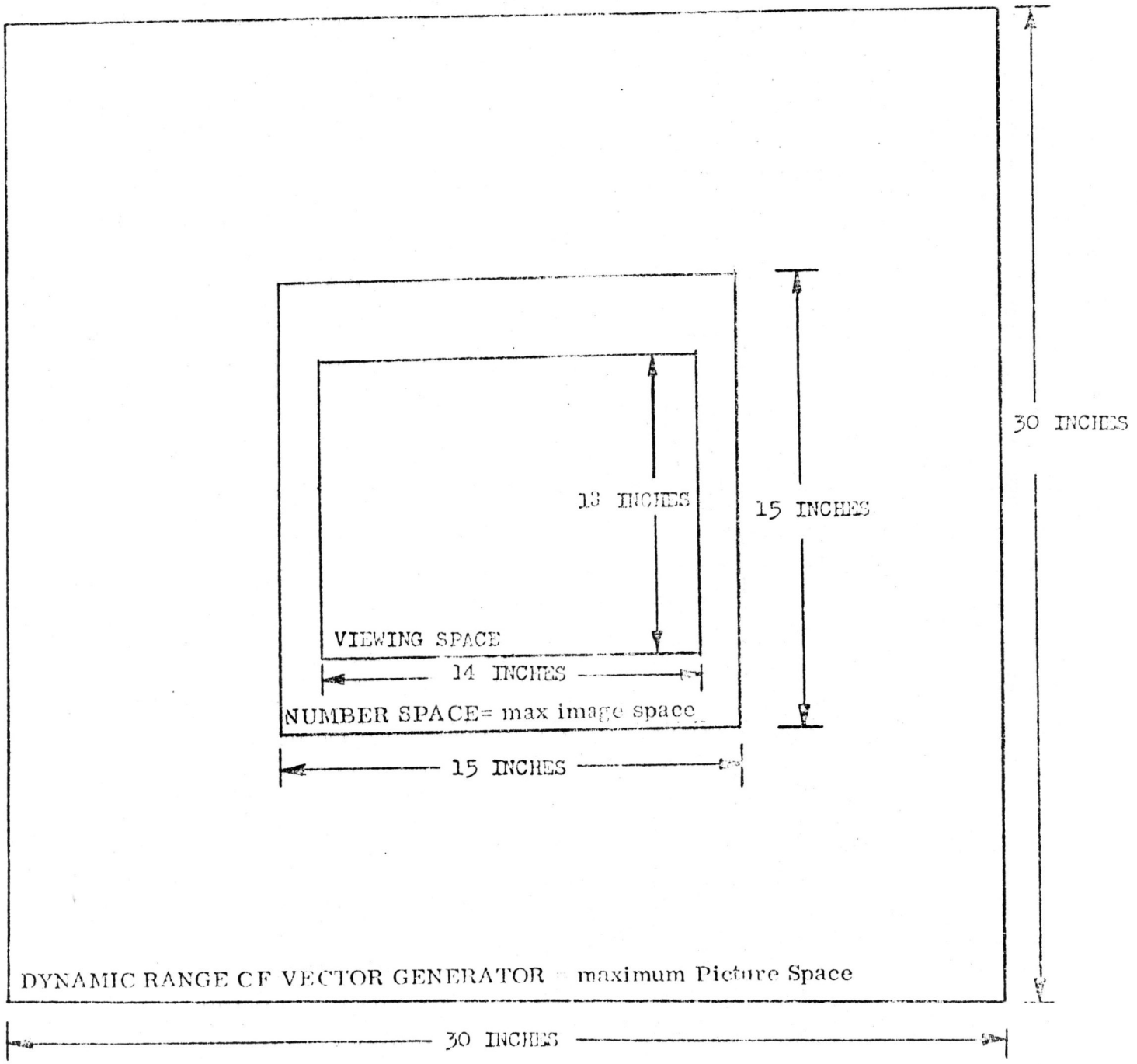

Figure 2. Image Areas, 21-Inch Display

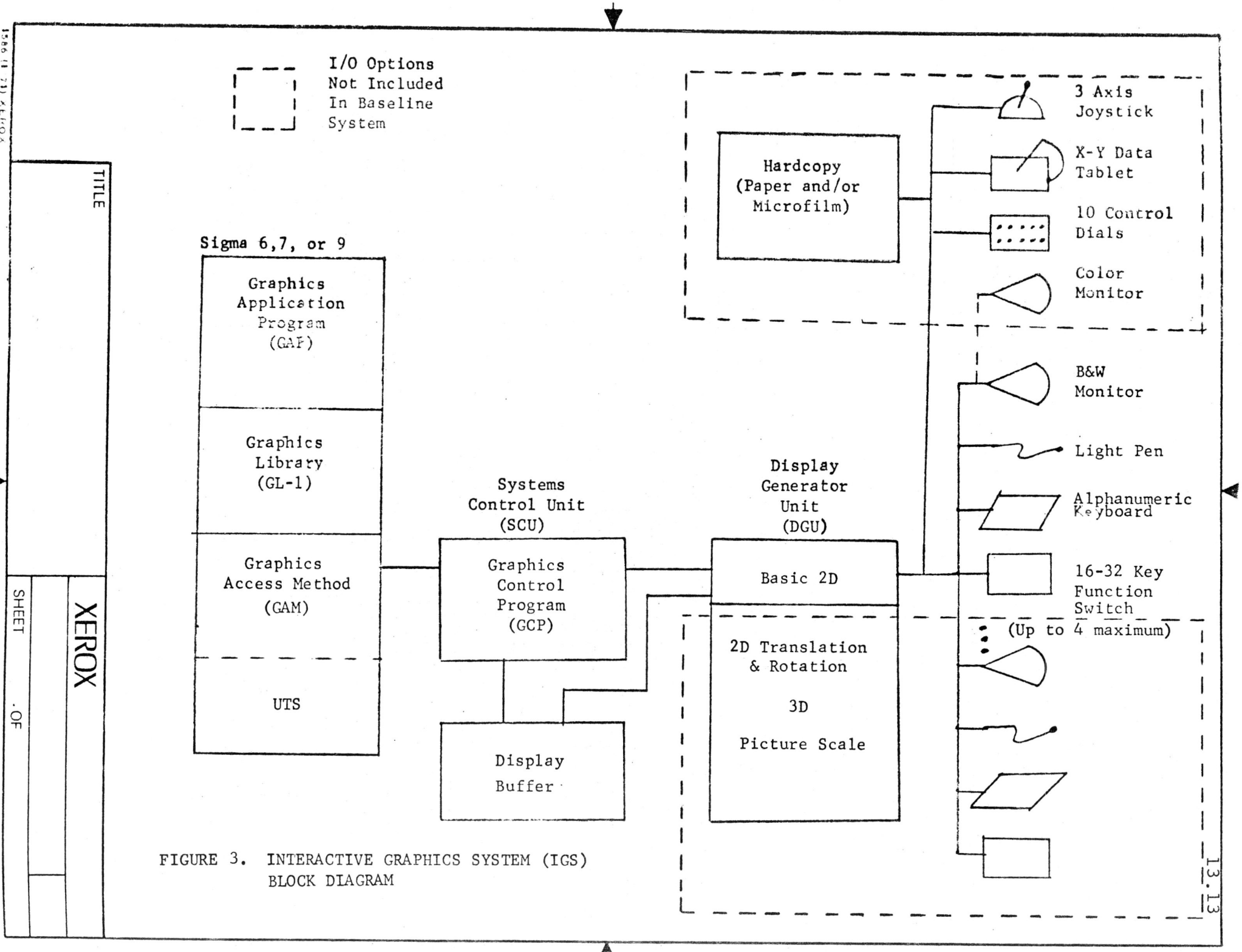

FIGURE 3. INTERACTIVE GRAPHICS SYSTEM (IGS) BLOCK DIAGRAM

7.1.2.1 GL-1 Call Summary

GL-1 Initialization and Termination (2)

```
BGNGL (area,size)
TRMGL
```

Graphics Device (3)

```
BGNGD (dcb[,MAT,
  SAT])
SELGD (dcb)
TRMGD (dcb)
```

Logical Structure (10)

```
BGNBLK (blockname
  [,buffersize])
ENDBLK (blockname)
DSPBLK (blockname)
INCBLK (blockname)
OMTBLK (blockname)
SELBLK (blockname)
LNKBLK (blockname)
EXTBLK (blockname)
RSTBLK (blockname)
TRMBLK (blockname)
```

Vectors & Characters (11)

```
AMOVE (x,y)
MOVE (x,y)
DRAW (x,y[,number])
PLOT (x,y[,number])
DRSGMT (xstart,ystart,
  xend,yend[,number])
PUTTXT (text[,number])
MAMELT (elementname)
ALTELT (elementname)
INCELT (elementname)
OMTELT (elementname)
ORDER (order)
```

Mode Setting Commands (13)

```
SCALE (userlimits[,
  screenlimits])
VTYPE (vectortype[,auto])
VMODE (vectormode)
INTENS (intensity)
BLINK
NBLINK
BEAM
NBEAM
CHRSZE (charactersize)
CHRORT (orientation)
LPMODE (lpmode)
KBMODE (kbmode)
FKMODE (fkmode)
```

Attention Handling (13)

```
ENBLLP
RSETLP
DSBLLP
ENBLKB
RSETKB
DSBLKB
ENBLFK (fnkey[,fnkey,
  ...])
RSETFK
DSBLFK (fnkey[,fnkey,
  ...])
DSBLAL
CHKATN (atninfo)
WTATN (atninfo)
CLRATN
```

Keyboard Data Entry (1)

```
GETTXT (array,number,
  endcode)
```

Control Commands (3)

```
START
STOP
CMMND (command)
```

Error Handling (1)

```
ERROR (error,errorcount)
```

C O R R E C T I O N S

TO

XEROX COMPUTER GRAPHICS

by

Timothy K. Dudley
Xerox Corporation
U.S.A.

1. All references to "SYSTEM CONTROL UNIT" should read "MICROPROGRAMMABLE CONTROLLER".

2. All references to "SCU" should read "MPC".