Stand-alone /remote graphic system

by MICHAEL D. RAPKIN and OTHMAN M. ABU-GHEIDA
IBM Corporation
Kingston, New York

The big revolution in computer usage is, by now, an old story. No longer do users, programmers and computer operators always work independently and separately, communicating only through voluminous printouts. Engineers and designers can now participate continuously in the execution of their problems through displays. By accelerating design iterations, they can solve their problems more quickly and, at the same time, improve the quality of their solutions.

The evolution in computer usage—the expansion of graphic and other display capabilities through an ever-broadening spectrum of devices—is less well known, however, and is continuing. These devices are becoming both more and less sophisticated. The large user with a large system at his disposal can now do more things. But probably even more important, the small user has not been forgotten.

This paper describes a system* developed to meet the needs of two particular classes of users—the "stand alone user and the remote user. (Figure 1)

In the remote display configuration, the effect of the substantially lower data transmission rates available with telecommunication facilities, as compared with those provided by a CPU channel, had to be compensated for in order to minimize any degradation in system response to operator actions. For this reason, it was necessary that specific functions, preferably those characterized as high-usage or conversational, be capable of being performed within the display subsystem. This defined a need for local data storage, I/O capabilities, decision making, and data processing capabilities within the subsystem. These capabilities were also required of a low-cost, stand-alone system. Further study indicated that the key system design criteria of the two configurations were also similar—particularly the following:

- Regeneration of the display from the CPU core, for ease of programming and rapid display unit control.
- Direct attachment of the display to the CPU without the use of an intermediary channel for efficient interrupt-handling.
- Facilities within the display-CPU interface to make efficient use of core and to minimize the number of interrupts generated by the display.

*IBM 1130/2250 Graphic Data Processing System
Fast, convenient, auxiliary-data storage.
Programming support for graphic functions—i.e., all high-activity conversational operations.
FORTRAN language support.
Non-programmer interface for display user.

The result was a low-cost, dual-purpose configuration for both types of users.

General description

The display is regenerated from computer core storage by cycle-stealing; i.e., once the display has been started by an I/O command, it operates asynchronously with the CPU program and other I/O devices. The display is designed not to interfere with the operations of other I/O devices attached to the computer system. In general, these devices will have very little effect on the performance of the display. The computer system is not changed in any way to accommodate the display. Thus, when the system is not being used for a graphic application, the computer can be used for other data processing functions. The remote configuration can be operated at two speeds: from 1200 baud to 2400 baud, and from 19.2K baud to 230.4K baud (2.5 to 30K characters per second).

The 1130/2250 is a self-sufficient graphic system. Its basic programming support, the support for the stand-alone configuration, is also the basic support for the remote configuration. In remote configurations, the system will handle the high-usage conversational and image-generation functions in an application, using the central System/360 for computing functions and access to large, central data bases.

The central processing unit of the satellite computer is a compact, desk-size binary computer. The system features a core storage capacity of up to 32K 16-bit words, core speeds of 3.6 or 2.2 µsec per word, a built-in disk file of 512,000 words, up to four additional disk files, card and paper tape I/O, a plotter, and line printers.

The display is composed of the display CRT and an interface for attachment to the controlling CPU. The display incorporates a 21-inch CRT (having a 12-inch by 12-inch usable area) and a program-controllable, fibre-optic light-pen with a pressure-operated tip switch. Optional features include an alphanumeric keyboard and a 32-key, programmable function keyboard. (Figure 2)

Some of the important performance characteristics of the display are:

- A total of 3,848 character positions: 52 lines with 74 characters per line.
- 1024 × 1024 addressable positions
- Up to 2000 characters or 2800 incremental vectors generated at a 40-cps regeneration rate; up to 2600 characters or 3700 incremental vectors generated at 30 cps.
- Absolute addressing—the capability to generate straight lines between any pair of the 1024 × 1024 positions on the display screen.
- Incremental addressing—the capability to position draw increments ranging between —64 and +64 raster units in X and Y from the current position.

Character generation is a programmable function allowing the user flexibility in the generation and use of character sets. Upper- and lower-case alphabets can be generated on the display screen through the use of the alphanumeric keyboard, or directly as part of a display program. Other character generation facilities provide extensive editing capabilities—the operator can overwrite, subscript or superscript a character, or overwrite a whole line of characters.

The display (Figure 3) attaches to the CPU via a storage access channel. Core storage is both space- and time-shared. Display commands and orders are stored in core storage, and are decoded and executed in the display interface. The dis-
display program (the set of display orders) therefore shares core storage with the CPU program, resulting in direct control over the buffer program, fast updating of the display image, and efficient graphic-system programming. Once the display has been started by an I/O command, display orders are accessed from core by stealing core memory cycles; i.e., the display operates independently of the CPU program, and both can be running simultaneously.

The display interface consists of a Memory Address Register, a Data Register for temporary data buffering, a Revert Register used in display image subrouting, and decoding logic. The display CRT has two deflection systems: the main deflection system for generating vectors, and a character deflection system for generating characters. CPU interrupts are caused by light-pen detects, depression of keys on either the alphanumeric or program-function keyboards, or by the display program orders. When an interrupt occurs, all interrupt data are read into CPU core by the execution of a single display command.

**Cycle Stealing and Interference**

Attachment of the 2250 Model 4 to 1130 system via the storage access channel permits the operation of the display asynchronously with the CPU. That is, once the 2250 has been started, it continues to execute display orders, similar to the operation of a channel, by stealing core storage cycles without CPU program intervention. The portion of the core having the display orders becomes a buffer for the display. These orders are accessed through the storage access channel and sent to the display up to 40 times per second. Since the display, I/O devices, and CPU are requesting core-storage cycles from a single source, some delay must occur. However, the design of the storage-access channel and the display interface prevents any significant interference with other I/O devices.

The following are some characteristics of cycle-stealing:

- The lowest-priority for cycle-stealing is the CPU. Some of the I/O devices require interrupt service within a specified length of time. To ensure that they obtain this service, the display is inhibited from cycle-stealing when interrupt service is required for these devices.
- When the display is drawing vectors, it steals one or two cycles to access the data related to the position of the vectors. The display will not cycle-steal during the time that a vector is being drawn.
- The maximum interference from cycle stealing occurs when the display is generating characters.
- The maximum interference with display operation occurs when the CPU has to continuously process interrupts for time-dependent devices. This interference will not normally be observed on the display.

**The display interface**

Because it was essential that the CPU be an effective processor of data in both system configurations, a display interface was included which reduces core requirements for the display, and handles interrupts quickly and efficiently.

A display program executed by this interface consists of orders and data. Orders either define the display operation or establish its "Mode." Order-defined operations include vector and point-plotting, branching, and CPU-interrupt generation. Three orders establish modes: Set Graphic, Set Character, and Set Light-Pen. The display is always in either Graphic or Character Mode, and in one of four pen modes.
Display Orders

Regeneration of the display program (Figure 4)

A combination of a Start Timer order at the beginning of the display program and a BRANCH order at the end of the program provides the regeneration cycle and ensures that the regeneration rate is no greater than 40 cycles per second.

There is no lower limit placed on the regeneration rate. However, display images regenerated at rates below 30 cycles per second will tend to "flicker."

Graphic mode

Either vector or point operations can be performed in Graphic Mode. In this mode, the display can receive, from the display program, either beam positioning or drawing orders, or an order to change mode. There are three basic beam-positioning orders which can be executed in Graphic Mode:

- Absolute positioning to any point \((X, Y)\) on the \(1024 \times 1024\) grid, with beam either on (to draw a vector or point) or off (to position the beam).
- Absolute positioning to any point, \(X\) or \(Y\), with beam on or off. This order moves the beam vertically or horizontally, and minimizes the use of CPU core in display images with a predominance of vertical and horizontal lines (Figure 5).

In generating the \(10 \times 7\) grid, only 31 words of core are required, whereas 62 words would have been required using the normal two-word, absolute-vector format. This two-to-one savings in core storage will also apply in the generation of bar charts, wiring diagrams, and integrated circuit layouts.

- Relative positioning with increments \(\Delta X\), \(\Delta Y\), up to \(+63\) raster units or \(-64\) raster units. The use of this order reduces core utilization for images with a large number of lines of \(\frac{3}{4}\) inch or less in length, and is necessary for image subroutines.

When the relative positioning of the beam causes it to exceed the bounds of the screen area and a total displacement of 1024 raster units beyond the perimeter is not exceeded, the vectors, points, or character strokes displaced will be blanked. Unless the overflow limit of 1024 raster units is exceeded, the displaced beam can be returned to the normal display area. The virtual image size is four times the actual screen size, and can be positioned anywhere within a region equivalent to nine screen areas. (Figure 7)

Figure 6 illustrates the importance of relative vectors. In the illustration, the resistor would be represented by a series of incremental vectors and stored in 1130 core storage as a graphic subroutine. Thus, even though the resistor appears in several places on the screen, the order appears only once in the display list.

Graphic subroutines

A graphic subroutine is a sequence of display
orders which displays a logical element or entity (such as a logic block, a resistor, a bolt, etc.). Graphic subroutine capability significantly reduces storage requirements for display images. Instead of requiring a copy of an image entity wherever it appears in the display image, the entity can be represented once by a graphic subroutine and can be generated as often as required by the execution of a "Branch" order. The display uses three orders to provide basic and multiple-level subroutine capability. (Figure 8)

Character generation

Character generation is a programmable function. Two character sizes can be displayed: .16 of an inch high and .24 of an inch high. Characters represented by their component strokes are organized into graphic subroutines and stored in 1130 core storage. Character generation is initiated by a "Set Character Mode" order. This order is followed by a series of "Branches" to character stroke subroutines.

The first branch order transfers program execution to the character-stroke subroutines. Up to two character strokes are contained with the 16-bit computer word. The last character stroke word of the character contains a revert bit, R, which performs the same function as the "Revert" order; i.e., it causes an automatic return to the display program. In addition, automatic character spacing results from the detection of the Revert bit.

Spacing to a new line is also automatic if the characters have been initially positioned by an absolute movement of the CRT beam. New line spacing is suppressed in the case of relative positioning. Special control codes within the character-stroke word are used to suppress spacing, position to a new line, insert a superscript or subscript, and reserve a location in GPU storage for later character placement.

Logical control orders

The logical control orders are used in the display program to reduce CPU program intervention, especially with respect to light-pen-detect interrupts and light-pen tracking. The control orders fall into three major categories: light pen control orders, conditional branch orders which
provide capability for logical decision making, and conditional interrupt orders which also supply logical decision-making capability and allow the CPU to be used effectively in support of display operation.

**Stand-alone programming support**

The Stand-Alone support, in addition to directly supporting the stand-alone configuration, is also the basis for the remote configuration. It has three components:

- Modifications to the 1130 Disk monitor system to allow the loading and execution of graphic programs.
- Extensions to the assembler to permit the symbolic coding of graphic programs.
- The Graphic Subroutine Package, a set of assembler language subroutines callable from both FORTRAN and Assembler Language, which perform image generation, image management, attention handling, support of the alphanumeric keyboard, and light-pen support.

The Graphic Subroutine Package is a set of assembler language subroutines which allow the FORTRAN or Assembler or Language programmer to create graphic images on the display. The displays can be constructed of lines, points, and characters. This package also furnishes the communication between the user and the program through routines related to the use of the light pen, function, keys, and the alphanumeric keyboard. The graphic subroutine package design is closely aligned to the design of the display interface, and its facilities make optimum use of the interface's functional capabilities. The structure of the graphic subroutine package can be best understood by looking first at the set of functions which are common to most graphic applications. These functions are shown in Figure 9.

The user at the display generates an attention through the alphanumeric keyboard, program function keyboard, or light pen. The attention is processed by an attention-control function which consists of:

- System attention handling which recognizes attention and indicates to the program that an attention of a certain type has occurred.
- User attention control, which controls the program flow and determines if any further processing is needed.

The attention controller may require access to the problem model or data base in the system. Control from the attention controller may be passed to the application program which performs the arithmetic and logic processing on the data base. When this is completed, control is passed to a group of routines which generate and organize new display data. This new display data is placed in the buffer, thus modifying the display content and completing the cycle back to the user.

The Graphic Subroutine Package provides the FORTRAN programmer with all the necessary routines required for the image generation, image control, and attention handling functions of his application.

Image management and control routines allow the logical grouping and structuring of the basic display elements (lines, points, and characters). Any group of these elements becomes an entity. Thus the four lines forming a box become one box entity that the user can create, delete, modify, or group with another entity.

Each created entity or elements within an entity can be given a unique identification value which is returned to the program (by the at-

![Figure 9](https://www.computerhistory.org/)

**FIGURE 9**—Functions and data flow within a graphic application
Attention handling routines) when the entity is detected by the light pen.

The attributes of an entity can be dynamically controlled. A grid, for example, can be turned on or off. When the grid is displayed, it could be made detectable by the light pen (for generating a drawing) or undetectable when it is used as a background reference.

The collection of all entities is called an image entity, whose structure is defined by the series of calls to the image management subroutines. The elements within an entity are defined by a series of calls to the image generation subroutines. Some functions performed by image management routines are:

- Initialize the image construction area.
- Begin an entity, which defines the name, beginning, and attribute of the entity. This routine will usually be followed by calls to the image generation routines.
- End an entity.
- Delete an entity.
- Update an entity: this routine is used to add, change, or remove elements from an entity.
- Display an image entity.
- Stop the display.

The image generation subroutines are used to define the contents of an entity by converting the program input data into display format. Thus, an array of user X, Y, floating point data representing a graph is scaled, translated, scissored (eliminating any portions of the graph which are outside the display area of the display), and converted into vectors for display. By changing the scale factor, the same graph can be enlarged or made smaller. The graph can also be moved by changing the translation factor.

Some of the functions performed by the image generation routines are:

- Set the image generation control parameters of scaling, translation, scissoring, type and format of user data, and type and format of output data.
- Plot Line(s)
- Plot Points
- Plot Text
- Plot Grid
- Copy and Entity

Attention handling routines allow the program to specify the types of acceptable attentions (user actions), process the attentions when they occur, and provide the attention information to the user program.

The graphic subroutine package also includes the following auxiliary routines:

- Tracking subroutines which allow the user to draw lines, through the use of the light pen and a tracking symbol.
- Alphanumeric key support routine which allows the user to input and edit messages from the alphanumeric keyboard.

Remote configuration support

The programming support package for the remote configuration is structured such that the 1130 can handle all the graphic functions in an application—image management, image generation, attention-handling, and communication with the application program—and call on the central system for computational assistance and/or access to a large central data base. It is important to note how this structure relates to that illustrated in Figure 9. It can be seen that the functions of the graphic subroutine package are common to both configurations, and that the functions labeled “Model or Data Base Access” and “Application Processing” reside in the 1130 or in a central System/360, depending on whether we are dealing with a Stand-Alone or a Remote 1130/2250 System configuration. Thus, the Stand-Alone support is fundamental to supporting the Remote configuration.

The Remote Configuration support consists of two elements:

1. Data transmission and conversion subroutines which facilitate communication and interchange of data between an IBM System/360 and one or more 1130 Computing Systems.
2. The Satellite Graphic Job Processor (SGJP), which allows a remote display user to define, initiate and control a job which is either exclusively processed in System/360 or concurrently in System/360 and the 1130/2250 System.

Data transmission and conversion subroutines

The data transmission and conversion subroutines make up what is called the processor-to-processor (PTOP) program. They are in-
voked via calls in the FORTRAN IV or the ASSEMBLER languages.

Separate sets of transmission subroutines are available in System/360 and in the 1130. They perform transmission control functions and insert the proper transmission-line control characters, enabling the programmer to perform data transmission without having detailed knowledge of telecommunications programming.

The transmission subroutines perform the following in either system:

- Initialize the communication line(s).
- Transmit and receive data via the communication lines.
- Test the status of a previously requested transmit or receive operation.
- Activate a user-written asynchronous routine in the other system.
- Terminate the communication linkage between the System/360 and the 1130 PTO programs.

The capability of terminating the communication link in one system at a time makes it possible, for example, for a new core load (in the 1130) to reinitialize communication with the transmission program existing in System/360. This would allow a user to monitor the progress of lengthy computation by receiving intermediate results, terminating communication, analyzing these results, and then reinitializing communication with perhaps a new set of input parameters.

In addition, the System/360 transmission subroutines enable the programmer to terminate the 1130 mainline program that is currently being executed.

Conversion subroutines are provided to resolve differences in the FORTRAN data formats of System/360 and the 1130. These subroutines can be invoked only from the System/360 program.

Figure 10 illustrates the sequence of operations and data flow for transmission from the 1130 to System/360. Figure 11 illustrates the same for transmission from System/360 to the 1130.

**The Satellite Graphic Job Processor**

The Satellite Graphic Job Processor (SGJP) is a program that elicits job control information from a user at the display unit, enabling him to process a job exclusively in the System/360 or concurrently in the System/360 and the subsystem. SGJP interprets the job control information
entered through the display unit and converts it into a language (Job Control Language) meaningful to the System/360 Operating System. The Job Control Language is then passed to the operating system to actually initiate the desired program. These services allow the non-programmer user to conveniently, rapidly define and start his job.

A job is defined as the fundamental unit of work for a computing system as seen by the user. A job may consist of one or more job steps, each of which requests the processing of a program or procedure.

Job control operations provide the job control information necessary to define jobs step-by-step, to describe data characteristics and device requirements related to job steps, and to start the processing of jobs. Job control information is presented to the operating system from the subsystem by means of SGJP.

The job control operations available with SGJP and their functions are listed below:

<table>
<thead>
<tr>
<th>OPERATION</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOG ON</td>
<td>Identifies the user to the operating system.</td>
</tr>
<tr>
<td>SPECIFY JOB STEP</td>
<td>Names a program or procedure to be executed in the System/360.</td>
</tr>
<tr>
<td>BEGIN PROCEDURE</td>
<td>Causes a named procedure to be processed as a foreground job in the System/360.</td>
</tr>
<tr>
<td>DESCRIBE DATA</td>
<td>Identifies data to be used in the specified System/360 job step.</td>
</tr>
<tr>
<td>BEGIN JOB</td>
<td>Starts the processing of the defined System/360 job.</td>
</tr>
<tr>
<td>SPECIFY 1130 PROGRAM</td>
<td>Names an 1130 program that is to run in conjunction with a program in the System/360.</td>
</tr>
<tr>
<td>WRITE MESSAGE</td>
<td>Sends a message to the System/360 operator and handles a reply to the 2250 user.</td>
</tr>
<tr>
<td>ENTER DATA</td>
<td>Allows 80-character records to be entered from the display unit or a card reader for use by the System/360 program.</td>
</tr>
<tr>
<td>CANCEL JOB</td>
<td>Deletes the job that is currently being defined.</td>
</tr>
<tr>
<td>RECALL</td>
<td>Allows the user to re-examine and modify previously completed job control operations.</td>
</tr>
<tr>
<td>LOG OFF</td>
<td>Completes user interaction with the display unit and frees it for the next user.</td>
</tr>
</tbody>
</table>

The above operations can be used to describe and start the processing of all types of application jobs (for example: a graphic program, an assembly, a service program, etc.) directly from the display unit. A job initiated at the subsystem can be designated to process as either a foreground or background job.

**System-user communication**

To enable the selection and performance of job control operations, SGJP establishes communication between the user at the subsystem and the operating system by means of displayed frames.

The frames are displays that request the entry of job control information from the user. Each information request is indicated by a word or phrase. The entry area related to the individual request is indicated by a short underscore or a rectangular box where an entry is to be made.

The frames are displayed in a logical sequence (that is, only as they are applicable to the user's job). Through interaction with SGJP, the user's responses to the frames convey the information necessary to process his jobs.

The particular information to be supplied in response to the frames (such as accounting code, procedure name, etc.) depends upon installation and user requirements.

There are two types of frames used for com-
munication between the user and SGJP. These are the select frame and the parameter frame.

The select frame

The select frame presents the job control operations available to a user at each point during definition of his job. It is from this frame that the user selects the next operation he wants to perform. The select frame is divided into three areas: the selection area, the history area, and the message area (see Figure 12).

The selection area contains the list of the job control operations currently available to the user and then entry areas for their selection. This list varies as operations are selected and processed, and guides the user by presenting only the operations that are applicable to his job. An entry area for selecting an operation is indicated by a short underscore preceding the name of the operation. For an operation for which the user must provide information, the area in which the user is to enter information is denoted by a short underscore following the name of the operation.

The history area contains a sequential list of the 14 most recent job control operations selected by the user. As the user completes each operation, an entry for that operation is added to the list. If there already are 14 operations in the list, the oldest operation is removed from the history area as each new operation is added to the list. Each operation is preceded by a 3-digit history number to indicate the sequence in which the operation was performed during the current session.

The message area is used to display status information and diagnostic messages.

The parameter frame

The parameter frame asks the user to supply values, called parameters, that are necessary to complete a selected operation. It is divided into three areas: the description area, the message area, and the key area (see Figure 13).

The description area contains the name of the selected operation, all parameter requests, and the entry areas associated with the requests.

If the parameter request is one for which information can be entered from the keyboard, the request is followed by a short underscore. In addition, if the user must provide the information, the entry area following the request is enclosed in a box.

If the parameter request may be selected with the keyboard or with the light pen, the request is preceded by a short underscore. In some cases, the user is given the choice of two or more options that will satisfy a parameter request. Normally, one of the options is completely underscored. The underscored option is called the default option, and this option will take effect if no other option is selected for the request. The message area is used to display status information, diagnostic messages, or replies from the system operator.

The key area displays the words END and CANCEL which may be used to perform the END and CANCEL functions with the light pen. Because they perform the same functions as the END and CANCEL keys on the alphanumeric keyboard, they are referred to as the END key and CANCEL key.
Application example

To illustrate the application environment and to demonstrate the sequence of operations required to execute a graphics application from a remote site, the following application example of a sample job that is initiated and processed by a user at the subsystem is presented. Illustrations of the frames that accompany the job control operations at the display unit are included.

John Doe, a user, is sitting at a display unit that is not in use. He wishes to process a job to design a lens at the display unit. This job consists of one System/360 job step that is executed in conjunction with a related 1130 program.

The System/360 job step calls for execution of a program named LENSDESN which will perform the calculations for a lens display. LENSDESN uses a previously created data set named LENSSAVE. The subsystem is referenced as DEVICE by the program. The data set is referenced as OUTPUT and is to be retained at the end of the job step.

The 1130 program, named LPGM, operates in the subsystem in conjunction with the System/360 program. LPGM contains the specifications for a "thin lens" design, except for two user-supplied parameters that specify the aperture and focal lengths of the lens. During processing, LPGM accepts data entered from the 2250 Display Unit and transmits it to the System/360 program where computations for the lens display are performed. When the System/360 program has completed the calculations, it transfers the results to LPGM. LPGM then displays the results on the 2250 screen.

Note that, during definition of the job, the user recognizes an error in a job control operation he has already completed and uses the RECALL operation to correct the error. He then completes the definition of the job and starts processing.

The first thing the user must do is identify himself to the operating system and provide his account number of KG505301. By performing the CANCEL function from the keyboard he obtains the LOG ON frame. First, he enters his name in the frame from the alphanumeric keyboard. Then, he positions the cursor on the screen to ACCOUNT and enters his account identification from the keyboard. To obtain a list of the job control operations that he performs, he also selects the PRINTED RECORD option.

At this point, the user has completed the LOG ON operation. He performs the END function to indicate that he has finished entering information on the LOG ON frame.

A select frame now appears on the screen. Displayed in this frame are the various job control operations the user can perform at this time. The first entry in the history area of the frame reflects the LOG ON operation he has just completed. The user now selects SPECIFY JOB STEP in order to define the System/360 program.
The SPECIFY JOB STEP frame now appears on the screen. This frame requests certain information about a job step, such as the name of the procedure or program, subsystem reference, and other optional specifications.

The user begins SPECIFY JOB STEP by entering the name of the program (LENSDESN) from the alphanumeric keyboard. To indicate that his job step is a program, he enters the name after the PROGRAM NAME option. Then, since this System/360 program will be processed in conjunction with an 1130 program (LPGM), he enters the subsystem reference, FT49F001, from the alphanumeric keyboard. The subsystem reference is a symbolic name by which the user's System/360 program refers to the subsystem. The user then enters from the keyboard the parameters (aperture of 5.0 and focal length of +4.2) necessary for his program. The frame now appears as follows:

Since all information necessary for his job has been entered in the frame, the user performs the END function to indicate that the SPECIFY JOB STEP operation is complete.

A second select frame appears on the screen displaying the job control operations now available to the user. The second entry in the history area reflects the SPECIFY JOB STEP operation. To describe his data set, the user selects DESCRIBE DATA.

The DESCRIBE DATA frame now appears on the screen. This frame requests the information necessary for the user to identify his data set, such as the data set name, the data reference by which the System/360 program refers to the data set, and other specifications.

The user begins the DESCRIBE DATA operation by entering the name of his data set, LENSSAVE, from the alphanumeric keyboard. Then, he positions the cursor to DATA REFERENCE and enters the name OUTPUT from the alphanumeric keyboard. Options for status and disposition can now be specified. The user knows that the data set is CATALOGED; i.e., it already exists and can be found automatically by the operating system. The user does not have to specify CATALOGED, however, because it is a default option (note underscore on frame). Furthermore, the user does not have to specify a disposition since he wishes to retain the data set and the operating system (if he specifies no option) will assume the disposition already assigned to the data set (KEEP).
The user now performs the END function to indicate that the DESCRIBE DATA frame has been completed.

A third select frame is now displayed on the screen. At this point, however, the user suddenly realizes that he meant to specify a focal length of 4.02 (instead of 4.20) as the parameter for his lens specification in the SPECIFY JOB STEP operation. The user decides to correct the operation in which he provided the focal length parameter. To re-examine the operations in his job, he selects RECALL on the select frame.

He performs the END function and the SPECIFY JOB STEP frame is now displayed as it appeared when the user had completed this operation earlier in his job.

The user positions the cursor to the entry area following PARAMETERS on the frame. He uses the ADVANCE key to position the cursor to the desired point of change and enters 02 in place of the previous 20. The frame now appears as follows:

Since this was the only change he wished to make, he performs the END function and the RECALL frame is displayed again.

The next operation (DESCRIBE DATA) is shown after “CURRENT ITEM” and the SPECIFY JOB STEP has been added to the history area of the frame. The frame appears as follows:
Since the user has no changes to make in the DESCRIBE DATA operation he performs the END function. The system assumes he wishes to retain the operation since ACCEPT is the default option. Because this was the last operation in the history area that could be recalled, a select frame appears on the screen so that the user can continue his job.

At this point, the user has completed all specifications needed for his job. Therefore, he selects BEGIN JOB to indicate that his job is ready to start processing.

A message containing an identification number given to the job by the system is displayed in the message area of the above select frame. In this sample the message returned is:

"JOB SCHEDULED AS J0240001"

The user performs an END function to acknowledge the message.

The SPECIFY 1130 PROGRAM parameter frame now appears on the screen and the user enters the name of his 1130 program (LPGM) from the alphanumeric keyboard.

The user now performs the END function to indicate the frame is complete and his job begins processing.

Using the aperture and focal length parameters provided as part of the SPECIFY JOB STEP operation, the System/360 program performs calculations and transmits data to the 1130 necessary to display an image, such as the one that follows:

The 1130 program is designed to accept additional information entered from the alphanumeric keyboard or with the light pen. The data are transmitted to the System/360 where new calculations are performed and information needed to modify the lens display is returned to the 1130. By providing new data and manipulating his display, the user designs a lens that meets his requirements. When the user has completed designing the lens, he terminates his program in a manner specified by the installation.

When the job has completed processing, a select frame automatically appears on the screen with a message indicating that the job has been completed. Since the user had only this one job to process, he selects LOG OFF. (If the user had wanted to define additional jobs, he could have continued by selecting another job control operation.)
The LOG OFF frame now appears on the screen. The frame contains a message from the accounting routine asking the user to supply the number of jobs he has executed since he logged on. Since the user has executed only one job, he positions the cursor to the entry area after TEXT and types the number “1.” The user wants to leave the 2250 Unit available for SGJP operation by a different person, so he does not designate the DISCONNECT THE SUBSYSTEM option. The frame appears as follows:

Having entered the required information, the user performs the END function. The screen now goes blank. The LOG ON frame is made available to another user when the CANCEL key is depressed at the keyboard.

Communication requirements

The important question in the design of a remote graphic system is whether the 300-cps or higher-speed (5000 to 30,000 cps) transmission between the 1130 and System/360 is sufficient for graphic applications. To answer this question, it is important, first, to recognize that the satellite system is not always communicating with the System/360. During the input phase of the application, the system is in a stand-alone mode for some duration of time within an application. During this mode, there is no transmission to System/360 and the response time due to operator actions will be very fast.

The effect of the data rate on response time should, therefore, be considered only when the system is communicating to System/360, and the relationship of this time to the total application cycle time is important. Figure 14 illustrates this principle. Specifically, the turnaround time for transmission to the System/360 computation or access of data in System/360, and then transmission from System/360 to the satellite computer must be considered. Depending on the type of application, the justification of the 300 cps or higher speed is based on any or all of the following:

- The amount of transmitted data
- The ratio of System/360 computation or data access time to transmission time
- The ease with which the data display can be overlapped with transmission. For example, a quick response time can be achieved by starting to display portions of the data as soon as the data is received in the subsystem.

Last and most important, it must be recognized that the 1130 computing capability reduces the frequency of communication between the dis-
play and System/360. If some operator action causes slower response than is normally acceptable to the operator, it will be tolerated as long as it does not occur frequently.

In general, 300 cps should be satisfactory for analysis-type applications in which small amounts of data (messages, control parameters, single graphs, etc.) are transmitted between the subsystem and the central processor; higher transmission rates may be needed for applications requiring frequent transmission of large amounts of data.

ACKNOWLEDGMENTS
The following individuals made important contributions to this paper: Mr. D. B. Arnold, Product Publications, Mr. R. L. Bean, Programming Publications, and Mr. J. M. Ettinger and his group, Remote Graphic System Support.