A graphic job processor enables nonprogrammers to introduce application programs conversationally from a display console. Although not restricted to graphics applications, the processor makes the same display console available for both job definition and interaction with a graphics application program.

This paper discusses some of the factors considered in designing displays to elicit information from the user. The structure of the processor is then described, including its interface to the operating system under which it functions. It also discusses communication among the system operator, the console user, and the application programmer.

Conversational job control

by S. H. Brown

A graphic job processor has been designed to help a nonprogrammer display-console user to define and initiate computing system jobs in a conversational manner.[1] Although not restricted to graphics jobs, the processor is particularly useful to the engineer or scientist at the display console because:

- It automates many job control activities, speeding the job control process.
- It guides the nonprogrammer user, preventing some errors and minimizing the effects of others.
- It circumvents some job control activities requiring special knowledge and training.
- It enables the user to initiate an application program at the same display console that will be used to interact with the executing application program.
- It permits the user to run both graphics and nongraphics application programs on a demand basis at a time of his choosing.

Job control includes those activities associated with describing the job, defining its processing requirements, and getting it processed by the computer. (These operations are tabulated in more detail later in the paper.)

Job control activities have grown in complexity with newer data processing systems. Such systems make more efficient use of data processing equipment and reduce maintenance activities asso-

ciated with application programs. However, the graphics user cannot enter his job in an interactive way.

Using the graphic job processor, a nonprogrammer user can define jobs for computing systems supported by SYSTEM/360 Operating Systems capable of multiprogramming.[2,3] Throughout this paper, the emphasis is on *foreground* jobs, which we generally assume to be interactive, real-time applications run at higher priorities. The processor also accepts *background* jobs, which are usually run at lower priority in batch mode.

One version of the processor permits job control over telecommunications lines and is intended for use with a small, remotely located satellite computer. Throughout this paper, we concern ourselves only with the graphic job processor intended for local use. It should be noted, however, that the so-called local graphics terminal is seldom in the same room with the central processing unit and may not even be in the same building.

We first consider the graphic job processor from the point of view of the user. Then we discuss some of the factors considered in its functional relationship with the operating system. The final topic in the paper is communication among the nonprogrammer user, the system operator, and the application programmer.

Console user interface

The graphic job processor uses English phrases to elicit job control facts meaningful to the operating system from the user at the display console, guiding him where practicable. From this information, the processor usually constructs standard job control statements, which are read and interpreted by the operating system exactly as if they were entered in a job stream.

The activities required for job control are outlined in Table 1, which compares how the user initiates job control functions conventionally and how he does it using the graphic job processor. The operations required in using the graphic job processor are performed in the order shown except for LOGON, which is the first graphic job processor operation.

Users can perform operations other than those that result in job control statements. For example, selection of ENTER DATA permits the introduction of pure data. Users can also abort and restart a job definition.

The RECALL function allows the user to bring back previously entered information for re-use. Messages to the operator and data previously entered at the display console may be accepted by the user unseen, omitted, or redisplayed for review and possible modification. Redisplayed information is shown exactly as last seen. The present implementation represents a base from which it is possible to allow preservation of an entire terminal session for future RECALL. Even the writing of messages to the operator, such as requests to mount tape volumes, can be done automatically as part of the RECALL function.

Table 1 Graphic job processor capabilities

Job control activity	*Conventional method*	*Graphic job processor method*
Define job		
specify program or cataloged procedure	programmer codes JCL JOB and EXEC statements	user completes SPECIFY JOB STEP frame
describe data and devices	programmer codes JCL DD statement	user completes DESCRIBE DATA frame
enter data or control information	programmer codes as required	user completes ENTER DATA frame
negate job definition information	programmer discards statement	user invokes CANCEL JOB
amend job definition	programmer recodes or keypunches as required	user invokes RECALL
communicate with system operator	programmer writes on job request form	user completes WRITE MESSAGE frame
Get job into computer		
assemble input for run	programmer collates cards after keypunching	none
deliver job to computer room	programmer delivers or uses mail service	none
sign in job and get authorization	control clerk logs in job request form	user completes LOGON frame
schedule job	control clerk or job coordinator determines	user determines for foreground job
place job in job stream	system operator inserts in card reader	user invokes BEGIN JOB
Run job		
monitor during execution	user present if desired	user works with display terminal output
abort execution	system operator decides	user decides
Return results to user		
obtain final status and diagnose run	programmer receives with output and analyzes	user sees immediate display of messages
identify and collect output	system operator gathers output listings	unchanged
sign out job	control clerk logs out job request form	user completes LOGOFF frame
return input to user	programmer fetches or receives by mail	none
deliver output to user	programmer fetches or receives by mail	unchanged

limiting specifications

A basic philosophy was followed in the design of a graphic job processor that would provide these functions in an interactive, easy-to-use manner—that of seeking only essential information. The graphic job processor was designed so that user identification and the name of a cataloged procedure[4] is the only information always required. However, processing most jobs requires additional information peculiar to that job. In the operating system, cataloged procedures are often overridden with specific information, such as account numbers, input data characteristics, output data disposition, changed references to terminal devices and data sets, priorities, and many more.

The graphic job processor reduces this extensive array of choices for the user. Default conditions are supplied in place of some choices. Also, making a given choice often eliminates the need for subsequent choices.

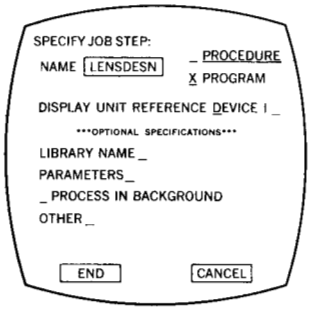
display frames

Job control information is elicited from the user by a sequence of display images called *frames*, which we define as a formatted presentation of information with allowance for optional responses by the user. When a frame is first displayed, it is comparable to an uncompleted application blank, as shown in Figures 1 and 2.

In the graphic job processor, a frame may assume variations (partial overlays) as the user's intentions become more apparent. For example, in the basic frame in which the user can DESCRIBE DATA, we assume that the data set to be used has been cataloged and that the system has sufficient information to gain access to it. However, if the user indicates that this is not the case, additional information, such as unit designation or number of records, is requested by means of an overlay of part of the frame. The manner in which overlays are manipulated may be changed by system programmers based on needs at a given installation.

Figure 1 Frame used primarily for commands and to select operations

Figure 2 Completed frame used to supply parameters for a selected operation



Although display screens have a high capacity for rapid textual output, human factors cannot be ignored. In the graphic job processor, we avoided overwhelming the user with large volumes of text. The larger character size available with the character generator of the IBM 2250 display console was used as much as possible, and an effort was made to keep frames uncluttered. Yet, since the display screen allows a purposeful arrangement of a sizeable amount of text, restrictions on the amount of text were not as stringent as they would have been, for example, on a typewriter-like terminal.

Experience with a previous display terminal system indicates that it is convenient to allow the user to supply as much information as possible in a given frame. For example, rather than request the user's name first and then his account number, it is preferable to request both items of information in the same LOGON frame.

The graphic display screen offers several methods for directing the attention of the user to a particular part of the display. In the graphic job processor, underlining is presently used to signify default options, which are provided automatically if the user makes no choice. Also, parts of the image are intensified to call attention to incorrect entries or possible errors. Blinking could

have been substituted for intensification. If color were available, errors could be indicated in red.

Data for the generation of frames is stored in separate program modules to permit changes to be made easily. For example, the English words and phrases used in the graphic job processor can be replaced by other languages, entry requests can be added or deleted, and frame formats can be modified to suit the particular installation.

Operating system interface

Many of the major decisions about the graphic job processor design involved its relationship to the operating system. Decisions related to such factors as the structure of the graphic job processor for best usage of system resources. Also considered was the question of which existing operating system facilities should be used by the processor and which should be duplicated in the processor.

At the beginning of the design process, we had to decide whether the graphic job processor should retain system resources continuously, or whether it should be brought into a region or partition as needed. If the graphic job processor were always present, it could always coordinate information, handle attention signals, and receive messages. There would be no interval during changeover from one program to another when the graphic job processor was not available. However, system resources, including main storage, allocated to the job processor would not be available to the user or to other tasks being executed in the system. And it is a requirement that one system resource, the display unit, be made available to the user.

The actual graphic job processor design is a compromise: the *communication program*, execution of which is treated as a separate task, remains in main storage throughout conversational job control operations; the *terminal program*, a copy of which supports each display console directly, is brought into main storage as needed.

communication program

The communication program, which requires less than 10K bytes of main storage, is initiated as a system task by an operator command before display console operations begin. The communication program

- Causes loading and initializing of the terminal program when the user is ready to define and initiate his job.
- Starts the user's job.
- Terminates the terminal program or the user's job.
- Communicates with the system operator, so that he can change operating parameters and can activate and de-activate terminals according to operating needs.
- Stores status information essential to the continuity of terminal program support for each display terminal between application program executions.

terminal program

A copy of the terminal program is brought into main storage by the communication program as it is needed to define a job at a particular 2250. The 2250 at which the user sits is allocated to the terminal program (no 2250 is ever allocated to the communication program). The terminal program

- Presents the frames to the user and accepts his responses.
- Checks for some types of errors as the user defines his job.
- Produces job control statements and submits them to the operating system.

The terminal program operates in a 60K-byte region or partition, which is large enough to also contain the 44K-byte interpreter portion of the operating system. (The interpreter interprets the job control language and places the job in a queue to be executed.) When a job has been given to the system, the main storage space allocated to the terminal program is freed for subsequent re-allocation to the application program or for other uses. This approach maximizes the main storage available to the application program, since no part of the terminal program remains in the region or partition.

The application program operates independently of both the terminal program and the communication program.

system operation

Some of the remaining concepts of the graphic job processor can be better understood within the context of overall operation.

At the beginning of the terminal day, the system operator starts the communication program as a system task. At this time, he can amend information that was originally defined during system generation, such as the priority of the graphic job processor or background jobs.

The system operator next supplies to the system the channel address of all 2250 display consoles to be used for job control operations, distinguishing them from 2250 display consoles that are to be used as conventional input/output devices. From this point on, if an attention signal originates from any of the designated display consoles, the communication program causes the terminal program for that display console to be loaded into main storage and its execution to be started. It does this by giving a START command internally.

Note that the LOGON function is invoked by an initial (i.e., new or out-of-context) attention signal from any display console previously designated as an eligible terminal. On the 2250, this signal can be generated by a program function key or by an END or CANCEL key.

The terminal program presents the frames to the user, through which he defines the processing requirements for his foreground job. When the user selects BEGIN JOB, the interpreter portion of the operating system is brought into main storage to accept the job control statements that have been constructed. When interpreter processing is completed, execution of the terminal program is terminated, freeing all system resources that have been allocated

to it, including the 2250 display console. During its termination processing, the terminal program notifies the communication program of its pending termination. The user's job can then be initiated. The 2250 freed by the terminal program is allocated to the user's application program whether it is required or not.

When execution of the application program is terminated, the termination routines in the operating system record this fact in the communication program. The communication program restarts the terminal program for the particular graphic console. The user may then define another job or log off.

Although the graphic job processor is designed to provide control of a foreground job, background jobs can be defined and entered into the job queue. A background job is any job that does not use the foreground resources during its execution. Both foreground and background jobs may use any additional 2250 display devices not currently designated for job control. Furthermore, a foreground job is not required to use the 2250 display device at which it was defined, even though the device is reserved for it. For example, ordinary assemblies may be run in the foreground. Background jobs defined in the foreground are abandoned to the system, and the user is not informed when a background job is started or when it has been completed, since it is very likely that he may not be available to receive the information. If he were available, he probably would run it as a foreground job.

scheduling applications

During design of the graphic job processor, a question arose as to whether the operating system should schedule execution of the user's application program in the normal way, making use of the resource allocation facilities of the operating system, or whether the graphic job processor should load and transfer control to the application program. Since the operating system does not permit the definition or allocation of a data set as needed during execution, loading the application program into main storage would require pre-allocation of all required data sets and devices. Because it was not considered possible to predict these requirements, this approach was not taken. Depending on the multiprogramming option used, the same problem may occur to a lesser extent with main storage.

In scheduling jobs, the interpreter of the operating system obtains information from the job control statements and stores it in a group of control blocks that can be referred to by the operating system as needed during processing of the job. In designing the graphic job processor, consideration was given to having the processor fill in the needed data in these control blocks directly, rather than constructing job control statements. There are several advantages to having the graphic job processor produce job control statements as input to the operating system. For example, the graphic job processor can produce a listing of the job control statements it generates. Thus, checking this language, familiar to many operating system users, would be easier than checking storage dumps of control block information. Moreover, some of the rather

Table 2 Distribution of messages to system operator

Category	*Number of messages*
Internal program error	25*
Normal status conditions	3
System operator errors	3
Abnormal conditions due to configuration or malpractice	9

* Expected only as a result of installation modifications

Table 3 Messages to console user

Category	*Number of messages*
Advisory or status (to inform or reassure user)	3
LOGON or job definition (produced by either graphic job processor or user accounting routine to report conditions that usually must be corrected before user can proceed)	48
Explanations of why job could not be started (developed from selected system scheduler messages)	162*
Explanations of why job was terminated (developed from system completion codes)	193*

* Number depends on output of operating system as well as disposition or attribute codes in terminal program tables

complex functions of the interpreter would have to be duplicated, such as scanning and overriding of cataloged procedures and handling of errors, incurring additional overhead for the user.

After deciding to use the interpreter, it became necessary to decide on a mechanism for providing it with the job control statements produced by the graphic job processor. The statements could be written out onto a disk unit and read in again by the interpreter. However, this approach would reduce system performance when a large number of display consoles are active. Fortunately, the design of the operating system permits the graphic job processor to use the interpreter function directly, eliminating any intermediate steps.

other considerations

The graphic job processor as presently implemented waits until the user selects BEGIN JOB before calling the interpreter. This function, if done earlier, could overlap the user's "think" time, but that would have several disadvantages. Job control statements already released to the interpreter could not be altered, and information resulting in requests to override cataloged procedures would have to be provided in the correct order. In addition, bringing the interpreter into main storage at this time could tax storage space to an extent that might require use of a smaller but slower version of the interpreter or a larger partition or region for the terminal program and interpreter.

An approach called "back-to-back" jobs was considered to minimize the time required to reschedule the terminal program after the user application program has been terminated. Two jobs would be scheduled at the same time, the user-defined job first, followed by the terminal program. The difficulty here is that if the terminal program were scheduled as a normal job, it would not be executed with the privileges of a system task.

processing times and storage requirements

Since the graphic job processor provides terminal services to assist display console users, it uses a measurable amount of computer resources. The execution times and the main storage required for initiation, execution, and termination of the user's application programs, of course, remain the same.

However, time is required to initiate, execute, and terminate the terminal program itself. Defining a job at the display console typically requires between 360 and 3000 milliseconds of central processing unit (CPU) time on a SYSTEM/360 Model 50. The major time factor in the present implementation is that required to initiate the terminal program using an internal START command. This may take minutes if contention for serially reusable resources is at a high level. It is conceivable that this time (and subsequent termination time) can be reduced to a once-a-day frequency by some sort of save-and-restore facility.

Another cost of using the graphic job processor is the unavailability of the 10 K bytes of main storage required by the communication program, which is not available to other jobs during graphic job control operations.

User-operator-programmer interface

In addition to the interfaces between the graphic job processor and the user, and between the graphic job processor and the operating system, there must also be an interface between the graphic job processor communication program and the system operator. The graphic job processor is designed so that messages originating from either the terminal program or the communication program appear at the system console typewriter to help the system operator control the system. These messages fall into the categories indicated in Table 2.

Messages that have no value to the system operator but are useful to the application programmer (who may or may not be the user at the console) are stored in a system message block data set. Some messages that might also be useful to the system operator are supplied both to him and to the system message block data set.

Only messages of interest to the user are displayed on the 2250. They may be one to five lines of information, such as the general type of message, actual data referred to, cause of message, and suggested action. Emphasis is placed on the suggested action or remedy, where known. These messages to the user are categorized in Table 3.

Messages explaining why the job could not be started or why it was terminated are derived from operating system messages that

are translated by the terminal program so as to be readily understood by the nonprogrammer user. When the terminal program regains control after termination of the user's application program, it locates the system message block data set and scans it. Messages significant to the user are extracted and translated. When an operating system message is translated, the original message number is also supplied to the user so that he can find a more technical description in a reference manual if he chooses.

preventing errors

The sequencing of frames is designed to prevent errors. For example, BEGIN JOB is not made available for selection until the user has successfully defined at least one job step. Thus, he cannot start a job that has no job step. The design of the graphic job processor in this way eliminates the need for many types of error messages. Nonetheless, it must duplicate some operating system error checking to support meaningful interaction during job description, since the operating system would detect some errors too late to permit recovery. In general, the graphic job processor detects the following types of errors:

- Too little or missing information
- Too much or conflicting information
- Specification of nonexistent or misspelled procedures
- Some syntax errors, such as unpaired parentheses, embedded blanks or special characters, overly long fields, and data that is improperly numeric or alphabetic

Summary

Graphic display users, typically engineers and scientists, need the ability to process jobs on a computing system at a time and in a manner of their own choosing. The graphic job processor is designed for such users and requires little knowledge of data processing. Rather than passively accepting preplanned input, the graphics job processor interacts with and guides the user, performing some job control activities without user's direction. One of the design objectives was to prevent certain errors and to reduce the effects of others.

CITED REFERENCE AND FOOTNOTES

1. The graphic job processor, program number 360S-RC-541, is available through IBM Branch Offices.
2. B. I. Witt, "The functional structure of OS/360, Part II, Job and task management," *IBM Systems Journal* **5**, No. 1, 12–19 (1966).
3. Those operating system options capable of multiprogramming with a fixed number of tasks (MFT) and a variable number of tasks (MVT).
4. After processing requirements have been specified in a set of job control statements and the set has been cataloged, theoretically only a request to execute the cataloged procedure is needed to run a job in the batched mode.