Display terminals, although faster than typewriter devices, do not implicitly create records of the user's interactive sessions. Based on the premise that a display terminal session facility that also has the record-keeping functions of typewriter terminals would increase productivity, a research project was undertaken that has resulted in the session manager discussed. Experience with the system is summarized.

A time-sharing display terminal session manager

by J. M. McCrossin, R. P. O'Hara, and L. R. Koster

Display terminals are tending to supplant typewriter terminals as the interactive computing terminal of choice. To many interactive users, display terminals offer special features that increase productivity, especially for programmers. Part of the increased productivity comes from the display terminal itself (e.g., the capability of on-screen editing). To take full advantage of display terminals, however, the time-sharing system must provide support of a very different sort than that needed for typewriter terminals.

Most time-sharing systems are line oriented because they have been designed to be accessed from typewriter terminals. In a line-oriented system, the user accomplishes work by typing command statements one line at a time on a keyboard. The system responds with one or more lines of messages or other output. Thus a terminal session is a conversation between the user and the system, and is recorded on the typewriter terminal sheet, which serves as a hard copy record. If a user wants to review a command or output from earlier in a session, he merely looks back on the terminal sheet.

Although display terminals offer many advantages over typewriters, such as high-speed data display and quietness of opera-

tion, they also bring certain difficulties with them. Time-sharing terminal users who are familiar with the above-described typewriter support find their data vanishing each time the screen is erased (as though the terminal sheet had disappeared). Hard copy of program and session output becomes difficult to produce; for although data sets can be printed at local or system printers, the records of a terminal session usually cannot be produced. The screen of a display terminal often allows fewer characters than a typewriter's platen. Thus, output that contains more than eighty characters per line (such as compiler listings) overflows to the next line, and the screen becomes less intelligible.

Application programs, for example, can be specially coded to run on display terminals, and display terminal limitations, such as described above, can be avoided. Through the program, output can be formatted to fit the display screen and be retained in a buffer for redisplay upon request. Such full-screen programs have disadvantages in that they rarely support typewriter terminals, and they usually depend on the features and programming protocols of the display terminals they support. If a new, incompatible terminal is introduced, programs coded to run on the earlier display terminal must be recoded if they are to support the new terminal.

The typical approach to enhancing display terminal usage by time-sharing systems has been to treat each command or program separately; i.e., a line-oriented command is replaced by a full-screen command. After invoking such a command or program, the user is presented with one or more screens for data input and program output. When the command has finished, a blank screen or a main menu is typically redisplayed. Thus, a line-oriented time-sharing system has been replaced (at least partially) by a set of menus for command input and formatted command and program output. The display terminal session manager experiment discussed in this paper, in contrast, provides display support for the entire time-sharing session, not for the individual commands and programs that comprise it. This design philosophy springs from the fact that, to the time-sharing user, the relationships among commands that are entered during a terminal session are often as important as the commands themselves; there is continuity to the user/system conversation.

Now termed the "TSO session manager," the terminal display system that is the subject of this paper is based on an experimental system known as the Research Display Facility. The experimental system was developed at the IBM Thomas J. Watson Research Center in Yorktown Heights, New York, to provide comprehensive display support for the scientists and programmers who share the MVS/TSO system, that is, a Multiple Virtual Storage system with the Time Sharing Option.

Design objectives

As in most time-sharing systems, the TSO commands, editor, debugging facilities, and user programs are line oriented. The TSO user enters commands to the system in a string syntax, and the output that results from these commands is recorded on succeeding lines of a typewriter terminal paper or on a terminal screen. The TSO system resolves the differing programming requirements of the various terminals that can be used for a TSO session. Although these line-oriented programs and commands operate properly on both typewriter and display terminals, they do not utilize most of the features offered by display terminals.

The principal goal of the session manager is to provide support that can extend all facilities of the IBM 3270 Display Terminal to these programs, and eliminate the operational problems commonly associated with display terminals on time-sharing systems. One objective of this support is to increase the productivity of the TSO user. The session manager exploits the facilities of the display terminal to save time for the TSO user. Many features of the session manager work together to achieve this objective, including the following:

- The user controls the data to be displayed or not displayed at the terminal.
- Unnecessary keystrokes are avoided by allowing the user to form new input from data displayed on the screen. The program function keys of the 3270 may be defined to represent often-used TSO commands, input sequences, commands to application programs, or session manager commands.
- The keyboard can be set to remain unlocked, so as to allow data entry at all times.
- Multitasking by the session manager allows the user to execute session manager commands while TSO commands are being processed.
- A journal of the entire TSO session allows the user to review previous command input and output. Without the session manager, a TSO user must often re-execute TSO commands just to have the output redisplayed, when this is possible, and it is often not possible.
- Multiple windows for viewing different data reduce the need for printing data on system or local printers. For example, a session manager user may view compiler messages and source listings, and edit a data set at the same time.

Another objective has been to provide display support while allowing all TSO commands and user-written programs to run unchanged. TSO comprises so many commands and programs that changing all of them to conform to a new display support facility

is not a practical alternative. For this reason, the session manager provides display support to these line-oriented commands and programs in a transparent manner.

It was also intended that the session manager functionally enhance TSO by fully utilizing the facilities of the 3270 display terminal. Thus, a user would perceive an enhanced TSO that makes full use of the 3270 display terminal, through the following control features:

- User-entered commands and other TSO inputs are highlighted, whereas system outputs are of normal intensity. The bright and normal intensities distinguish the two sides of the interactive conversation. When a session journal is copied to a system printer, the highlighted lines are overprinted to appear darker.
- The TSO command or program that is currently executing is similarly highlighted in the input-only journal. This is to enable the user to keep track of the currently executing command if he has typed in several commands at a time.
- An audible alarm indicates errors in session manager commands or messages from other TSO users or background job.
- The program function keys of the 3270 may be defined as TSO input or session manager commands.

The session manager provides session-wide system-level support. The session manager display support is available to the TSO user from log-on to log-off. All line-oriented TSO commands and modes, including EDIT, EDIT Input Mode, and TEST are covered together with any programs that are running under TSO. At the same time, the session manager allows the retaining of all capabilities the TSO user has on a typewriter terminal. The session manager provides the 3270 display user with the same kind of function provided to the typewriter user, such as an unlocked keyboard and full-width nonwrapping output review of an entire TSO session.

Some TSO commands and programs are written to run specifically on the 3270 display terminal. The session manager is designed to coexist with these full-screen programs. The Structured Programming Facility—a TSO program product—is an example of such a command in which the session manager steps aside when these display-oriented programs are executing. Thus full-screen programs are not affected by the presence of the session manager on the system.

As a final design objective, the session manager should give the TSO user control of the display terminal. The session manager command language should allow the TSO user to dynamically redefine part or all of his display environment at any time during a

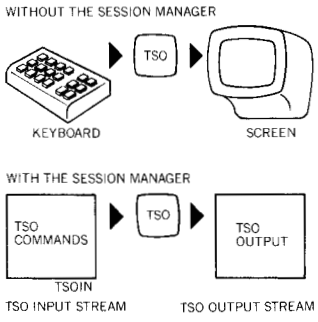
TSO session. None of the features of the display terminal (certain lines on the screen, certain program function keys, etc.) is to be reserved for use by the session manager. Thus, the TSO user can tailor the operation of the terminal to his personal needs and tastes, which may change from day to day or even during a single session.

System concepts

streams

The session manager incorporates several new concepts for the time-sharing user, which, together with some operational characteristics of the session manager, are described now. Views of the TSO user/system data flow with and without the session manager are shown in Figure 1. TSO receives its input from the terminal keyboard and sends its output to the 3270 display screen. With the session manager, TSO receives its input from and places its output in session manager *streams*. These streams reside in virtual storage in the TSO user's address space. A stream can be thought of as a "virtual sheet of paper" that can be read from and written upon.

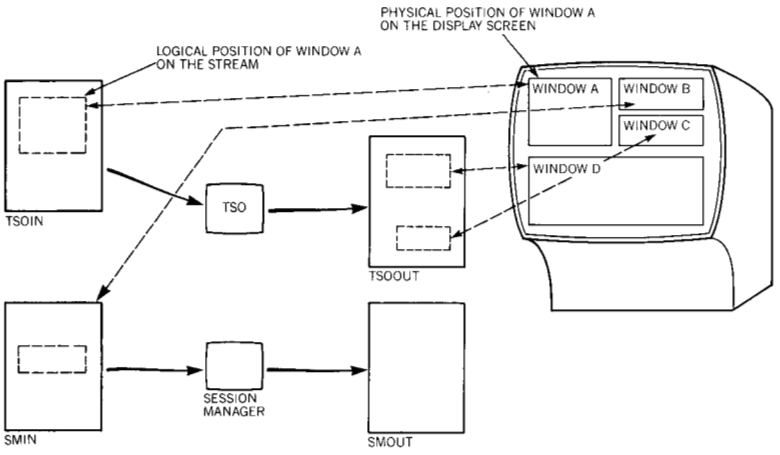
Figure 1 User/system data flows for TSO



Streams have a "top" and "bottom." Each line of data that enters a stream is placed in the next available line, starting at the top—just as lines of copy are typed on paper. The most recent line to be placed in the stream is considered to be the bottom of the stream. Streams, whose capacities are fixed when they are defined, gradually fill up with data. When a stream's capacity is exceeded, the stream "wraps around," and new data entering the stream replace the data at the top of the stream. The top of the stream now moves as new data enter, and thus contains the oldest data in the stream. Conceptually, there is no limit to the number of streams that each user may have. Among the streams given to session manager users as a default are the following:

- TSOIN This stream serves as the input source for TSO (commands, input data to application programs, etc.). Any data placed in this stream are read by TSO as though they had been typed at the terminal keyboard.
- TSOOUT Any output from TSO (READY prompts, messages, commands, or application output, etc.) is placed in this stream, as though it were the paper of a typewriter terminal. In addition, user input to TSO (copied from the TSOIN stream) is placed here to give a complete record of the terminal session. Thus, the TSOOUT stream looks identical to the terminal sheet produced by a TSO user at a typewriter terminal.
- SMIN This stream is used by the session manager in the same manner as the TSOIN stream is used by TSO. Any data placed here are interpreted as session manager commands.

Figure 2 Relationships between windows and streams



- SMOUT The session manager places error messages in this stream. Most session manager commands produce no output messages unless an error occurs.

The TSOOUT stream, as just described, contains a complete journal of the user's session. There is a command called SMCOPY. The SMCOPY TSO command processor allows the user to print on a system printer or copy to a data set all or part of any stream. In addition, this command can copy data sets into streams. The hard copy facility is especially useful in documenting program execution errors and program testing sessions under the TSO interactive debugging programs.

windows and scrolling

As session manager streams can be thought of as virtual sheets of paper, so session manager *windows* can be thought of as "virtual window panes" through which data in the stream can be viewed. The windows the user defines on the display screen are viewports on the data created and manipulated during a TSO session. Conceptually, the user may define as many windows as fit on the screen of the display. Each window is associated with a stream. Figure 2 shows a sample screen composed of four windows, each one for viewing a different stream. Note that two windows (C and D) view stream TSOOUT and that no window views stream SMOUT. There are no restrictions on the number of windows that may view a particular stream or whether a particular stream is viewed by a window. The presence of a window on a stream does not affect the data flow to and from that stream.

The process of moving a window over a stream is called *scrolling*. Each window may be moved left or right over the stream it is viewing, backward (up) toward the top of the stream, and forward (down) toward the bottom of the stream.

Windows operate in *locked* and *unlocked* modes. Just as the paper in a typewriter terminal moves to allow each new line to be typed, so an unlocked window moves to constantly display the newest data in the stream it is viewing. When a session manager scrolling command (listed, later in this paper) is issued, the window is locked (frozen) in position over the stream, and no longer moves to display new data. A locked window may be unlocked via the session manager UNLOCK command.

command and data entry

Windows serve the second function of command or data entry area on the display screen. In this mode, a cursor may be moved anywhere on the display screen, and new data may be entered. The entire screen may be made available as an input area. When the ENTER key or program function key is used to enter data, each modified line on the 3270 screen is transmitted to the system. The incoming data are then routed to one of the streams, based upon the window in which the data were entered. For example, a certain window on the screen may be defined as an input area for session manager commands, another window (or windows) may be defined as an input area for TSO commands, and still another window may be used to send comment lines to the session journal.

Multiple lines of input may be entered at one time; this is especially useful under EDIT in the input mode, or when logically related sequences of commands are to be entered to perform such functions as compile, link edit, and call. Previous command inputs or outputs can be used to form new command entries. Data displayed in a window may be modified and overtyped through the use of the terminal editing keys. For example, if a user wants to delete several data sets, he might list his catalog, then move the cursor on the screen to the displayed listing and insert "delete" in front of each data set to be deleted. The ENTER key sends the several commands thus created to TSO for execution. In a similar manner, mistakes in a command entry can be corrected simply by correcting the erroneous characters, rather than retyping the entire command.

Without the session manager, when a display terminal user enters a TSO command, the terminal keyboard remains locked until the command completes execution, which forces the user to wait. With the session manager, the user can set the keyboard to remain unlocked, thereby causing the terminal to be always available for new command entry. The TSO user can enter new commands while previous commands are being processed. The additional commands are stacked in the TSOIN stream. Thus, the TSO user perceives a faster-running TSO system than before, and he is no longer required to remain idle while waiting for a command to execute. In addition, while TSO commands are being processed, the user may enter session manager commands. These commands execute immediately (in parallel with TSO), so that the user may

scroll back to view previous output, redefine the screen layout or program function keys, etc., while waiting for TSO commands to execute.

System software restrictions prevent the session manager from updating the display screen while the keyboard is unlocked. In an attempt to overcome this, the session manager provides a timer that controls the unlocking of the keyboard. By setting the timer to a value of ten seconds or so, it is possible to strike a workable compromise between an unlocked keyboard and an up-to-date screen. Many TSO commands can execute within ten seconds and have their output displayed. At the same time, longer running commands do not lock up the keyboard.

program function keys

The 3270 display terminal offers as a feature a keyboard with twelve program function keys. For those terminals, the session manager allows the user to define each key as a character string that is to be sent to a specified stream by the action of the key. For example, a given program function key might be defined as the TSO command LISTCAT to be sent to the TSOIN stream. By the action of that key, LISTCAT is sent to the specified stream, where it is interpreted as a TSO command, and subsequently executed by TSO, just as though it had been entered from the keyboard. In a similar manner, program function keys can be defined as session manager commands. This is done in the default session manager screen layout. The program function keys issue scrolling commands, for example. Optionally, symbolic arguments can be specified in program function key definitions. These arguments are then replaced by data from the screen when a key is pressed. For example, a certain program function key could be specified as "change /&1/&2/" to be sent to the TSOIN stream. If the user then types

abc xyz

and presses that key, the following EDIT subcommand is executed:

CHANGE /abc/xyz/

dynamic display support

All the facilities of the session manager are made available to the TSO user through the session manager command language, which gives the user complete control over the terminal. Actions possible under the TSO session manager range from scrolling a window to redefining the entire screen layout.

To aid in the dynamic redefinition of the display terminal environment, the session manager provides push-down stacks (last-in-first-out) where various portions of the current display environment may be pushed and later popped. For example, a user might push the existing program function key definitions onto a stack,

then redefine one or more such keys for some temporary task (repetitive command entry, a find/change sequence under TSO EDIT, etc.), then restore the previous program function key definitions by popping them off the stack.

The following session manager commands are presently implemented:

session manager commands

Environment definition commands

CHANGE CURSOR	Change the screen location of the cursor.
CHANGE FUNCTION	Change input and output streams for TSO, the session manager, and cross-memory messages from background jobs and other TSO users.
CHANGE PFK	Change the meaning of a program function key.
CHANGE STREAM	Change stream attributes.
CHANGE TERMINAL	Change terminal attributes.
CHANGE WINDOW	Change window attributes.
DEFINE WINDOW	Define new windows on the display screen.
DELETE WINDOW	Remove a window from the display screen.

Screen control commands

FIND	Find a text string in a stream and scroll a window to display it.
SCROLL	Scroll (move) a window over the stream it is viewing and then lock it in place.
UNLOCK	Unlock a window that has been locked in place.

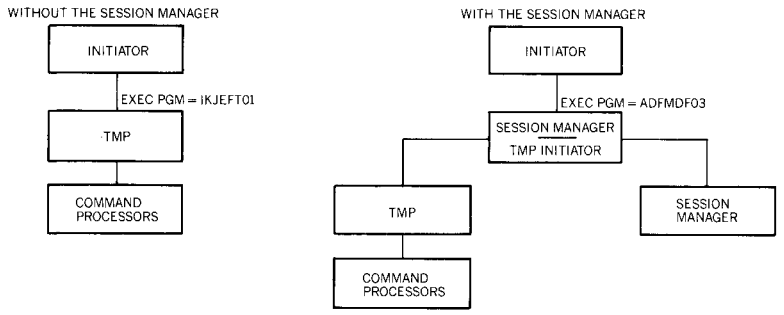
Session control commands

END	Terminate session manager display support and continue the TSO session.
PUT	Place a character string in a stream.
QUERY	Display status information.
RESET	Reset the screen and program function keys to the default settings.
RESTORE	Pop an element off one of the stacks.
SAVE	Push an element onto a stack.
SNAPSHOT	Take a "snapshot" of the display screen and place it in one of the streams for later printing.

TSO command processors

SMCOPY	Copy a stream to a TSO data set or vice versa. Print a data set or stream on a system printer.

Figure 3 System task structure



SMFIND	Search for a character string in a stream.
SMPUT	Place a character string in a stream.

TSO command procedures

The SMPUT command, given in the preceding list, allows the placement of session manager commands in TSO command procedures (CLISTs). Through such CLISTs, the user can build a repertoire of screen layouts and program function key definitions. These screen layouts are created by executing a CLIST that contains the session manager commands that define a given layout. For example, a user might build a CLIST with session manager commands that create a split screen and—as the last command in the CLIST—invoke TSO EDIT. Executing the CLIST then saves the current screen environment on one of the stacks, creates the split-screen layout and associated program function definitions, and starts the EDIT session. When the user leaves EDIT, the previous environment is restored. Similar CLISTs could be created to redefine the screen and program function keys for application program execution, TSO TEST sessions, etc.

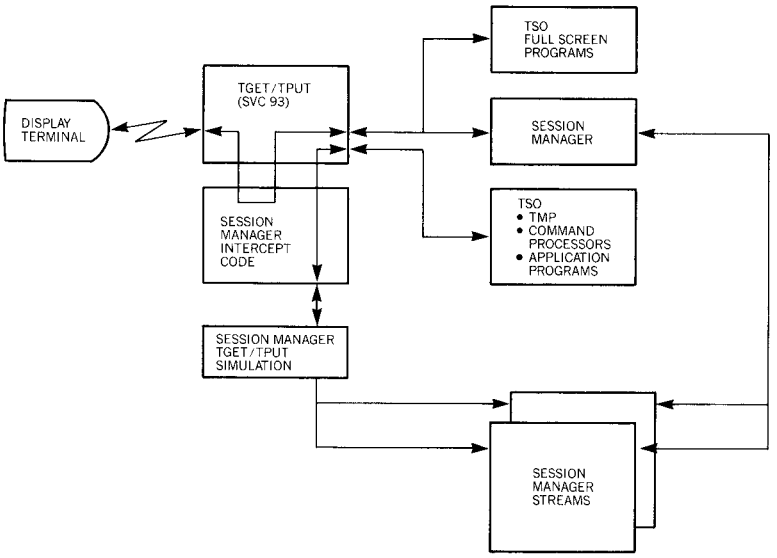
System implementation

MVS/TSO interface

When the session manager is not active, an initiator starts the Terminal Monitor Program (TMP) which controls the TSO user/system conversation. The TMP scans input lines from the terminal and attaches the appropriate command processors for interpretation and execution of the user's commands. The JCL of the log-on procedure specifies the name of the TMP to be started.

Figure 3 illustrates the location of the session manager in the MVS system task structure. The session manager log-on procedure specifies the name of a session manager module instead of the name of the TMP. When this module is executed, a new task—called the session manager/TMP initiator—is created. This task then starts the TMP and the session manager tasks, which run as sister tasks. The session manager intercepts all TGET/TPUT mes-

Figure 4 Data flow through TGET/TPUT (SVC 93)



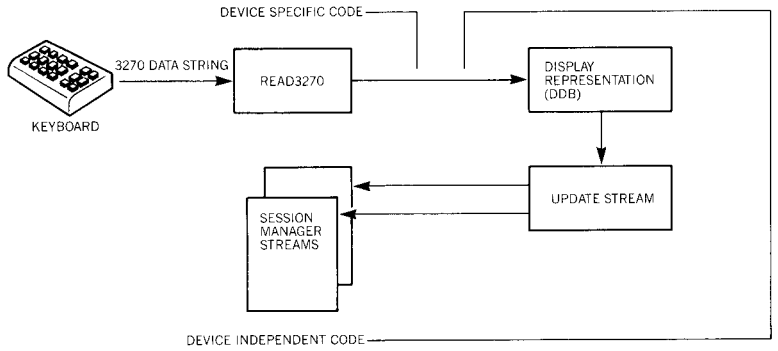
sages issued by the TMP or command processor and directs them to or from the appropriate session manager stream. This interception takes place in a session manager routine that receives control from exits in the TGET/TPUT routine (SVC 93).

Figure 4 shows the Input/Output (I/O) data paths to and from the terminal. Each time a TGET or TPUT is issued, the session manager intercept code (in the branch from SVC 93) determines whether the session manager task is active. If so, the intercept code determines whether the I/O request has been made on behalf of the session manager. I/O from the session manager and full-screen applications are not intercepted, but return to SVC 93 for normal processing.

The session manager maintains a device-independent display representation, termed the Display Description Buffer (DDB). The DDB contains sufficient information to partially or completely regenerate the display screen image when required. The device-independent routines of the session manager manipulate this data structure.

On input from the terminal, the input data are mapped into the DDB. On output to the terminal, the device output data string is generated from the internal representation in the DDB. The DDB describes the status of the display screen and the current environment as defined by the user and includes such things as the number of windows, program function key definitions, and cursor

Figure 5 Session manager display input operation



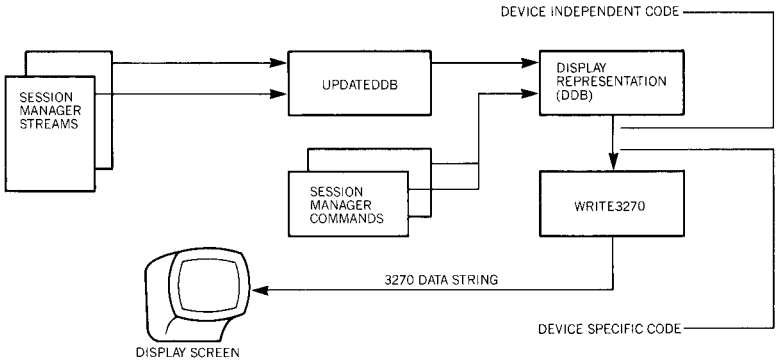
position. Pointed to by the DDB are the Window Description Blocks (WINBLOCKS), one for each window defined on the screen, which include descriptions of the placement and size of the window, the lines currently displayed in it, and the attributes of each line (is the line nondisplayable, highlighted, etc.) as well as attributes of the window itself (alarm, mode, overlap count, locked/unlocked state, etc.)

The DDB and WINBLOCKS provide the input information to the I/O modules (READ/WRITE3270) to build the proper 3270 data string when an update of the screen is required. The session manager does not rewrite the entire display screen each time a screen write is done; only those lines that need updating are rewritten.

When the user types input data on the 3270 screen and presses the ENTER key, the terminal transmits to the system a character string that contains device control information as well as the actual characters that have been typed. Each modified line of data on the screen causes a separate field to be created in the input data. The session manager I/O module (READ3270) determines (for each of these fields) into which window the data have been entered. Each input field is pointed to by its containing WINBLOCK. Figure 5 illustrates the control flow of the session manager modules during a display input cycle.

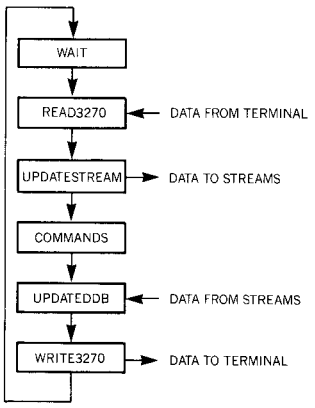
Later in the cycle, the input data—now stripped of control characters—are placed in the appropriate session manager stream by UPDATESTREAM. Each window definition specifies a stream to receive input data. The data are placed in this indicated stream for each input field. Data placed in the TSOIN stream are sent later to TSO where they are interpreted as command or other input. In the same manner, data sent to the SMIN stream are interpreted as session manager commands. Data sent to other streams are displayed but do not cause any program action.

Figure 6 Session manager display output operation



Movement of data into the streams and the execution of session manager commands cause the DDB to be updated. During an output operation, the session manager module WRITE3270 translates display requirements from the internal form stored in the DDB to 3270 device protocol. Figure 6 illustrates this translation. The data string generated is written to the 3270 display, thereby causing the screen to reflect the internal representation in the DDB.

Figure 7 The session manager task program loop



The session manager routines perform the program loop illustrated in Figure 7. At the top of the loop, there is a wait for input from the display terminal. When the user presses the ENTER key and the data are transmitted to the system, module READ3270 is called to map the input into the DDB/WINBLOCK structure.

Module UPDATESTREAM is then called to distribute the data to the proper streams. Module COMMANDS is called to execute any pending session manager commands that may have been queued from the previous UPDATESTREAM call. Module UPDATEDDB is called to bring the display representation up to date. The changes reflected have been caused by the previous session manager command execution, new data in a given stream as viewed by a window, or by the automatic movement of a window over the stream it is viewing. Module WRITE3270 is called to output the new screen image (via the TPUT SVC with the FULLSCREEN option).

The session manager routines just described are designed to execute under both the session manager and TSO tasks. They receive control under the TSO task each time a TGET or TPUT is issued and intercepted. The simulation of the TGET/TPUT is also performed at this time by directing the necessary data to/from the session manager streams. The existence of the session manager task that is running in parallel with the TSO task provides for data entry and display while TSO commands are being processed.

Experience and concluding remarks

The TSO session manager is based on an experimental display facility designed for the IBM Thomas J. Watson Research Center in Yorktown Heights, New York, where a substantial amount of experience has been gained by using the experimental system. Measurements were made by executing a number of typical TSO scripted sessions with and without the session manager. Results have shown that resource consumption in terms of MVS service units necessary to perform a script is slightly less when the session manager is active. In addition, a given script can be performed in an average of thirty percent less elapsed time. The session manager does not cause any additional resource consumption and it avoids calls to the terminal access method because multiple lines of output are written as a single screen write operation. Without the session manager, the TSO address space is swapped on each input request. The session manager allows the user to enter multiple input lines at a time, thus avoiding swaps, and a user can consume more resources per minute because he can be more productive than can a TSO user without the session manager. Typically, a session manager user has a larger paging demand due to the additional code, control blocks, and streams of the session manager. Although the working set is only slightly larger, the total virtual storage requirement is significantly larger, because the session manager streams reside in virtual storage. TSOOUT is typically the largest stream because it has been defined as 200 K bytes in the experimental system.

user survey

A survey of the experimental system reveals the following as the most practical features:

- Scrolling via program function keys.
- Session manager command invocation from TSO CLISTs.
- Multiple window definitions.
- User-defined screen layouts and program function key definitions.
- Hard copy of terminal session.
- Retention of messages from other users and background jobs.

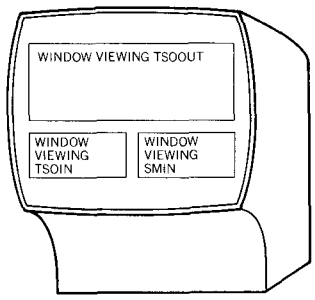
The following are features with which users of the experimental session manager experienced difficulty:

- The session manager's wealth of function makes it somewhat complex and, therefore, confusing to the first-time user.
- The screen cannot be updated when the keyboard is unlocked.
- Confusion sometimes occurs when certain full-screen TSO programs temporarily leave the full-screen mode.

examples of use

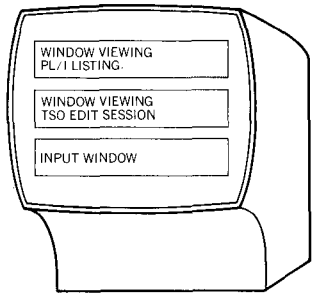
Given here is a close-up view of several typical example applications of the session manager. Consider first the personalizing of the screen layout.

Figure 8 Example of personalizing the screen layout



As illustrated in Figure 8, the user has written a TSO CLIST that redefines the default session manager screen layout. The CLIST consists of a number of SMPUT commands that place session manager commands that are needed to create the personalized screen layout in the SMIN stream. (Refer to the previously given description of the SMPUT TSO command in this paper.) Here the user has allocated most of the screen to a window for viewing the TSOOUT stream (which contains the complete TSO session journal), and has placed this window at the top of the screen. Two other windows at the screen bottom are used to view the TSOIN stream (which contains all TSO commands) and the SMIN stream (which contains all session manager commands). Program function keys have been defined to issue session manager scrolling commands.

Figure 9 Example of correcting errors in a PL/I program



In Figure 9 is shown an example of the use of the session manager for correcting errors in a PL/I program. Here, a PL/I compiler listing is viewed in the upper part of the screen simultaneously with a TSO EDIT session in the central portion of the screen. The user has written a TSO CLIST that does the following things:

- Copies the compiler listing to the TSOOUT stream by using the TSO LIST command.
- Splits the window for viewing the TSOOUT stream into two windows that show TSOOUT stream.
- Locks the top window over the beginning of the PL/I listing.
- Unlocks the center window to follow the TSO session.
- Defines program function keys to scroll over the listing.
- Calls TSO EDIT to modify the PL/I source program.

While one is editing the PL/I source program, the compiler output that contains the error messages and source listing is readily available for review via session manager scrolling commands. The need to copy information from the screen with pencil and paper has been eliminated, and the need to print the listing at a system or local printer has also been greatly reduced. The user scrolls through the listing, using TSO EDIT to correct errors in the source data set as required. Corrections appear in the input window at the bottom of the screen. After editing has been completed, the screen returns to its original format via the session manager RESTORE command (described earlier in this paper).

The improving of programmer productivity was the principal design goal of the experiment that resulted in the session manager. Through direct measurements and informal surveys we have found that the session manager has contributed greatly to increased user productivity on TSO. As another objective, the session manager experiment was also designed to explore display support for a range of TSO users, from scientists to system programmers. This objective has also been shown to have been met through the wide acceptance of the system.

GENERAL REFERENCES

1. *IBM OS/VS2 MVS TSO 3270 Extended Display Support-Session Manager*, SC28-0912 (Program Product 5740-XE2), IBM Corporation, Data Processing Division, White Plains, New York 10604 (1977).
2. *IBM TSO Display Support and Structured Programming Facility*, SH20-1975 (Program Product 5740-XT8), IBM Corporation, Data Processing Division, White Plains, New York 10604 (1976).
3. J. Martin, *Design of Man-Computer Dialogues*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey (1973).
4. W. M. Newman and R. F. Sproull, *Principles of Interactive Computer Graphics*, McGraw-Hill Book Co., New York, New York (1973).

Reprint Order No. G321-5074.