Application Program

H20-0266-0

Remote Access Computing System (RAX) for the IBM System/360 System Description

RAX is a system that decreases turnaround time by providing immediate and sustained access to an IBM System/360 through the advanced techniques of (1) remote computing via terminals and (2) time sharing. It allows for concurrent remote entry of jobs coded in FORTRAN IV or Basic Assembler Language while compiling or executing other jobs. The user has the system at his command while entering his program or receiving program results or compilation diagnostics. This manual includes machine configuration and functions, program capabilities, system operation, and timing.

Copies of this and other IBM publications can be obtained through IBM branch offices. Address comments concerning the contents of this publication to IBM, Technical Publications Department, 112 East Post Road, White Plains, N.Y. 10601

© International Business Machines Corporation 1966

CONTENTS

INTRODUCTION

The ultimate yardstick for measuring the efficiency of any computing system is effective throughput, or the productive output of the system, furnished in response to user requirements.

One basic problem that must be solved to attain maximum productivity lies in what is termed "turnaround" time, or the total elapsed time that must occur between a request for service by the user, wherever located, and delivery of computed results to that user.

Turnaround time becomes a truly formidable obstacle when:

- System users are remote from the DP center — perhaps separated by many miles
- The requirements of these remote users exceed substantially the usual inquiry/response capabilities of conventional terminals. Each remote user may have need of the full power of the central computing system, to perform a variety of scientific or engineering calculations.
- Multiple remote users exist whose problems need to be solved swiftly, accurately, and with maximum economy.

The Remote Access Computing System (RAX) has been specifically tailored to meet the needs of system users. It accomplishes this objective by providing immediate and sustained access to an IBM System/360 to any system user through the advanced techniques of (1) remote computing via terminals and (2) time sharing. Turnaround time is minimized; the power of the central system is made available to each user to the degree required; and user satisfaction and efficiency are thus enhanced.

RAX allows for concurrent remote entry of jobs coded in FORTRAN IV or Basic Assembler Language while compiling or executing other jobs. By means of the remote computing and time-sharing techniques used, the user at his terminal has the system at his command while entering his program or receiving program results or compilation diagnostics. The ability to communicate input/output (I/O) data at program execution time allows for "conversing" with a user program.

RAX has these outstanding features:

- Remote-access time sharing
- Time-sliced execution, a software technique for improved user response on short jobs
- Interaction with the program during execution
- Program storage, retrieval, and maintenance

- Terminal command capability
- Concurrent processing of data from widely separated I/O devices at the computer site
- Ability to use three standard models of System/360
- Two languages available
- Ability to use two different multiplexors
- Two different terminals for I/O

RAX operates a variety of hardware configurations in a time-sharing, time-slicing mode of operation. The processing unit can be one of three models of System/360: the Model 30, 40, or 50. The multiplexor can be either the IBM 2701 Data Adapter Unit or 2702 Transmission Control. There is a choice of two terminals: the IBM 1050 Data Communication Terminal or 2260 Display Station. The online IBM 2540 Card Read Punch and 1403 Printer can be used to handle jobs concurrently with terminal operations. To satisfy the secondary storage needs of the system, IBM 2311 Disk Storage Drives and IBM 2400 tapes can be utilized. This modularity and compatibility enable RAX to satisfy many requirements in the remote computing and time-sharing area.

The programming languages available under RAX are FORTRAN IV, the widely accepted problem-oriented language for engineers, scientists, mathematicians and similar users, and the Basic Assembler Language, a symbolic coding language. (For more information on the programming languages, see Bibliography.)

In addition, the user has a terminal command language to allow control of the terminal and to implement the use of the above programming languages. By means of a simple and concise set of terminal commands, the RAX terminal (2260, 1050) user will be able to:

1. Enter a job into the system for compilation and execution.
2. Obtain diagnostic information or execution output.
3. Save the program(s) for later recall and usage.
4. Edit an incorrect source program with reentry of only the corrections, deletions, or insertions necessary prior to a rerun.

Because of its unique capabilities (960-character buffer, nondestructive cursor, 1053 print station, and multiline display), the 2260 can also:

1. Key in and display multiple lines before transmission to the computer.
2. Display multiple lines of output.

3. Advance page by page through the user's program in an update mode, adding corrections as desired and receiving new pages on demand.

4. Correct or delete single characters anywhere on the display during updating and before transmission.

5. Request selectively hard-copy output.

6. Transmit at a high data transfer rate for improved system efficiencies.

MACHINE CONFIGURATION AND FUNCTIONS

Minimum and Maximum System

RAX, with its modular systems design, permits a large number of systems configurations. The minimum-size memory configuration shown in Figure 1 consists of:

One System/360 Processing Unit, Model 2030 F (64K) with:

Interval Timer

Storage Protection

Decimal Arithmetic

Floating Point Arithmetic

First Selector Channel

One 1052 Printer-Keyboard, Model 8 (Console)

One 2821 Control Unit, Model 1

One 2540 Card Read Punch

One 1403 Printer, Model 2, 3, 7, or N1

One 2841 Storage Control

Two 2311 Disk Storage Drives

One 2701 Data Adapter Unit or 2702 Transmission Control

Up to four or fourteen 1050 Data Communications Terminals, depending on whether the 2701 or 2702 is used

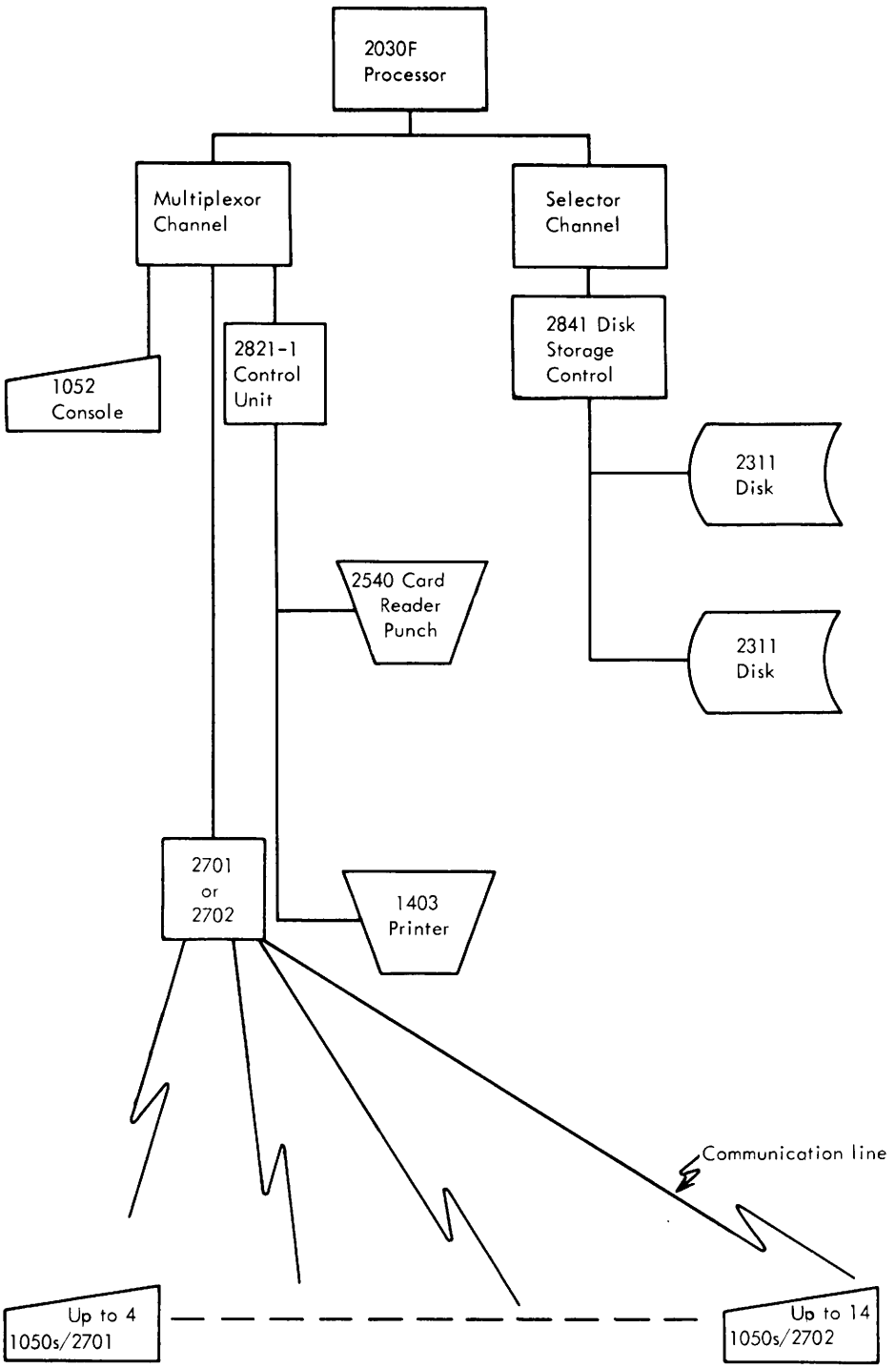

Figure 1. Minimum configuration

This system has all the language capabilities available to a larger configuration, except that the work file's/FILE control card of BPS FORTRAN would not be available. (No work file capabilities are allowed on the 64K core version of RAX.) Work files are defined as disk or tape files which may be used for the temporary functions of intermediate output, table spills, etc. They can also be used as master files. The work files are available only to jobs from the 2540, and record formats must conform to the BPS FORTRAN I/O restrictions.

The operation of and terminal configuration for RAX are highly memory-dependent. Larger memory results in more buffers and terminal control blocks, permitting the use of both the 1050 and 2260. Up to 63 terminals may be installed, in combination, with a maximum of sixty 1050s and eight 2260s. Larger memory will also allow up to 64K bytes for object program execution instead of 25K bytes.

The 128K system can have up to twenty-four 1050 terminals and four 2260 displays. A tradeoff on core storage will allow one 2260 to be added in place of three 1050s where it is desired to have more 2260s and fewer 1050s. For purposes of calculating the mix of terminal lines entering the system, 600 bytes of memory are required for each 1050 terminal and 1800 bytes per 2260 terminal after the basic I/O routines are incorporated.

While the figures above indicate the maximum number of 1050s and 2260s, the practical number of terminals depends heavily on the nature of the job mix and the installation turnaround requirements.

The maximum-size memory configuration supported by RAX is a 2050H Processing Unit (256K). A maximum configuration, as shown in Figure 2, consists of:

One System/360 Processor, Model 2050H with:

First Selector Channel

Second Selector Channel

One 1052 Printer-Keyboard, Model 7 (Console)

One 2821 Control Unit, Model 1

One 2540 Card Read Punch

One 1403 Printer, Model 2, 3, 7 or N1

Two 2841 Disk Storage Control

Eight 2311 Disk Storage Drives

Two 2702 Transmission Control

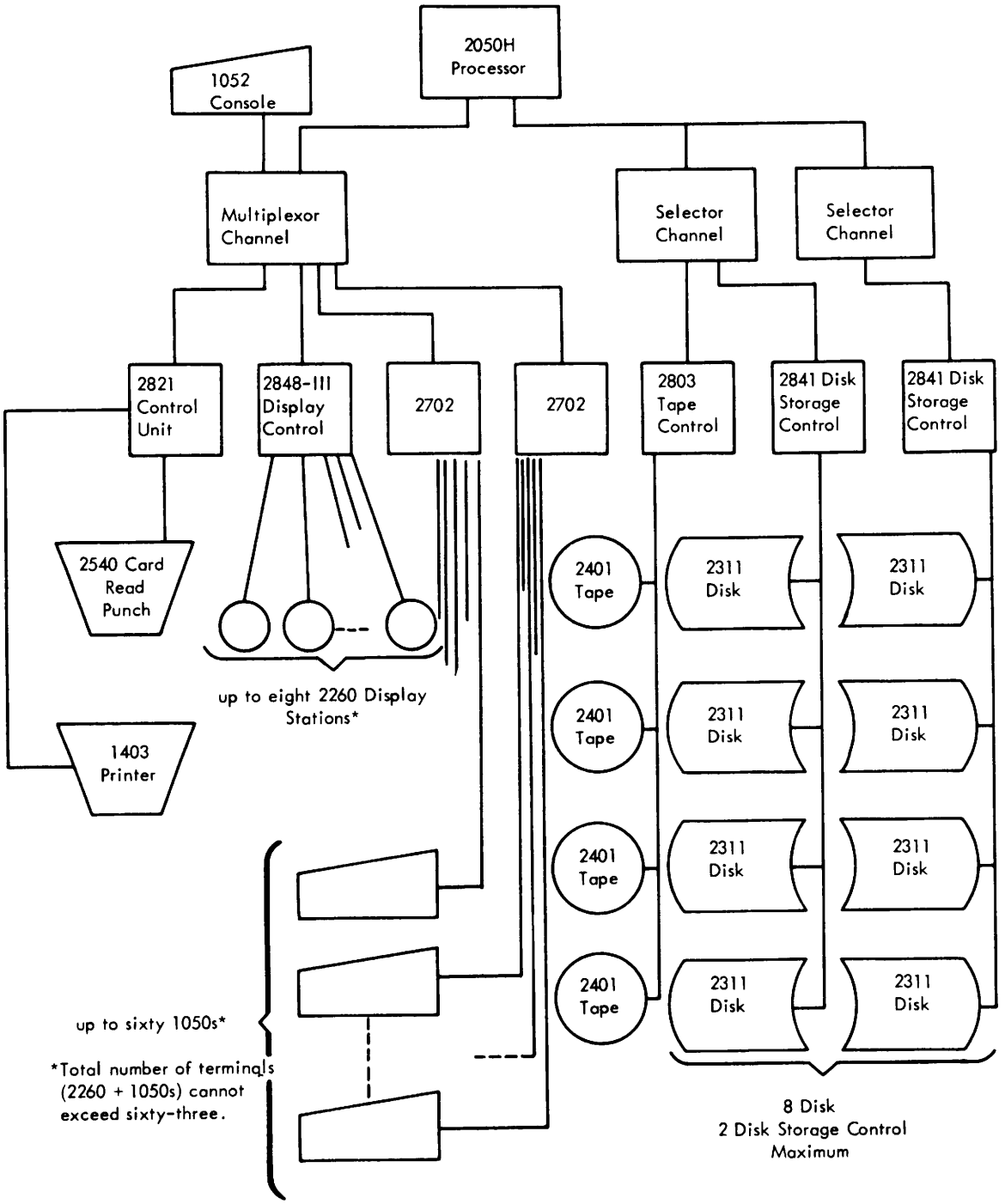

Figure 2. Maximum configuration

One 2803 Tape Control, Model 1

Four 2401 Magnetic Tape Units, Model 3

} Any combination of tape control and 2400 tape units allowed, except that a maximum of four tape units can be used by RAX.

One 2848 Display Control, Model 3 with:

Nondestructive Cursor

1053 Printer (if desired)

Channel Adapter

Eight 2260 Display Stations (maximum) with:

Alphameric Keyboard

} 63 maximum for both 1050 and 2260

Sixty 1050 Data Communications Terminals (maximum)

This configuration would have capabilities beyond those of the minimum system:

1. 65K core for the user's object program during execution.
2. Work file (disk or tape) available for use through the 2540 Card Read Punch. A maximum of four work files in any combination of tapes and/or disks can be used.
3. Use of a total of sixty-three 1050s or 2260s on one system.

Functions of System Components

Because RAX is a remote terminal system, terminal characteristics are very important. The 1050 and 2260 are the two terminals supported by RAX. In addition, the 2540 and 1403 can be used as input and output terminal devices.

The 1050 is a typewriter keyboard terminal with, optionally, paper tape or card I/O available. It operates at a maximum transmission rate of 14.8 characters per second (cps) across standard telephone lines or private communications lines attached via common-carrier data sets or IBM modems. A standard IBM SELECTRIC® typewriter is used for output with alphameric keyboard for input. Other means of I/O for the 1050 consist of paper tape or cards via the 1054 Paper Tape Reader, 1055 Paper Tape Punch, 1056 Card Reader, and 1058 Card Punch. Cards and tapes may be prepared offline on other devices or by utilizing the homeloop (nontransmitting) capability of the 1050.

The 2260 Display Station features a visual display unit, an alphameric keyboard for data and program entry, a nondestructive cursor for character modification on the face of the display unit, and multiple (twelve) 80-character lines for data and program entry and display. A high data transfer rate of 2560 cps, combined with its contention capabilities, makes it an efficient terminal for use with RAX.

Use of the 2260 is operationally simple and efficient. Because of the multiple 80-character lines, one can key in from one to twelve input statements and then visually verify them on the display unit before transmitting the lines. If a line must be corrected, the cursor can be moved to the point of erroneous data, the correction inserted, and the resulting information then transmitted to the system.

The UPDATE mode of operation fully utilizes the display's capabilities. An activity request for program updating causes eight lines of the program to be displayed. The cursor is located at the beginning of the ninth line, the first line where additions to the program can be entered. These additions are referenced to the old program statements which have line numbers inserted by the system in positions 73-80 of the 80-character line. Four program modification lines can be entered before transmission. After transmission, the initial eight lines will be redisplayed and additional correction lines can be entered in lines 9-12. Where minor changes are desired for the original eight lines, the nondestructive cursor can be used to locate the modification and then new information entered in. The changed line would be identified by a # sign in the first position to the left of the line number. Upon completion of modifications to a page, the user can request the next eight lines, or page, of the program by typing in /PAGE. Termination of the update can occur at any time. A new update scan of the program will show the new source program renumbered and with the modifications inserted.

The multiline display capability of the 2260 is also utilized at output time. Depression of the ENTER key on the keyboard causes the display of output in a scroll fashion across the face of the scope. Continual depressions of the key cause additional output to be displayed.

The 2540 Card Read Punch and various models of the 1403 Printer can operate concurrently with remote terminal access to RAX. The interrupt-handling capability of the RAX supervisor permits these two units to operate effectively at full speed. Jobs submitted to the system from the 2540 can use BPS FORTRAN IV or Basic Assembly Language or they may be object program decks. The jobs from the online devices follow the same rules as terminal-originated jobs in vying for compilations and/or execution. These jobs can, however, be larger source programs as more input and output buffers are assigned by the system.

Basic to systems operation are the 2311 Disk Storage Drives, two of which are required by RAX. RAX, its library, scratch files, the language processors, and the user's individual work files and transaction files are located on one of the packs. The user's library is stored on the second pack. Additional disk drives available to the system allow for expanding the size of the library. RAX provides support for up to eight disk files on two selector channels.

In the large RAX systems (Model G and H memories), disks can be used as work files by the online 2540 user. This configuration requires at least one disk pack in addition to the two needed for the system and library. Work file availability permits intermediate results to be stored and later recalled for further processing, for referencing master files, and for updating of files. These file records cannot exceed maximum FORTRAN output record size.

If tapes exist on the RAX system, they are generally available for other operations on the System/360 during non-RAX usage. RAX supports any valid combination of tape control and 2400 tapes up to four tape units. While RAX is running, tapes can be used only as work files — again only for the online 2540 user. Where multiple tapes and disks exist on the same system, only four units (in any mix of tape and disk) can be used, because the only valid references to the work units are as FORTRAN logical I/O units 1-4, in the READ and WRITE statements.

PROGRAM CAPABILITIES

RAX is a terminal system designed to enable a user to communicate quickly and easily with a computer. To accomplish this communication, there must be a way for a user at a terminal to communicate with the computer and then, after establishing a conversation, to submit a problem statement in a form on which the computer can operate. The RAX terminal command language is the means whereby terminal-to-computer control is accomplished. Figure 3 indicates the valid commands and their combinations. The systems languages, the problem-oriented language FORTRAN IV, and the Basic Assembler Language, are the means by which a coherent problem is stated to the system.

An easy approach to understanding RAX is to view it as a file manipulation system. A user has at his command a language, the terminal command language, by which he controls the file processing activities. Files at his command are:

1. The input file. This file can be:

   a. Used to collect card images

   b. Used to obtain a copy of a file in the library

   c. Routed to the FORTRAN processor

   d. Entered into the library as a file

   e. Queried as to its contents

   f. Modified

2. The library files. These are files of source card images indexed by name and stored in a file library. They can be modified, displayed, copied into the input file, purged, and incorporated into the input stream to FORTRAN.

Activities	Commands Given under Activity Mode																		
	/BAP	/CANCEL	/CHANGE	/DATA	/DELETE	/DISPLAY	/END	/FTC	/ID	/INCLUDE	/INPUT	/INSERT	/JOB	/OFF	/PAGE	/PURGE	/RUN	/SAVE	/UPDATE
/DISPLAY																			
/ID																	X		
/INPUT	X	X		X	X	X	X	X	X	X		X	X	X					
/PURGE																			
/RUN																			
/SAVE																			
/UPDATE 1050	X	X	X	X	X	X	X	X		*X			X	X					
/UPDATE 2260	X	X		X			X	X	X	*X			X		X				

X = commands valid at stated activity time.

- = invalid usage, i.e., not used or an error. This symbol should appear in every cell without an X or be dropped.

* = only on input file.

Figure 3. Valid activity and command table

3. The transaction file. This is the temporary storage space for changes to be made to a file which the user wishes to modify.

Terminal Command Language

The control statements used to manipulate the files in RAX follow a standard format. Position 1 of the statement always contains a / (slash). Starting at position 2 is the appropriate terminal command. One or more blanks separate the command from the variable parameters, which must start before column 16. The parameter list items are separated by commas, the first blank indicating the end of the parameter list.

The following conventions are used in this manual to describe the terminal command language:

1. Uppercase letters and punctuation marks (except those explained in items 3 - 5 below) represent information that must be coded exactly as shown.

2. Lowercase letters and words are generic terms representing information that must be supplied; that is, a substitution must be made when coding a parameter or option so represented.

3. Information within brackets { } represents an option that may be included or omitted entirely, depending on program requirements.

4. When several choices shown on multiple lines are enclosed in braces [ ], one of the enclosed alternatives must be selected by the programmer, unless a default option is given.

5. Where several choices are possible for one parameter, the underlined choice is the default option.

When a user has his terminal attached to the processor and wishes to use the system, he indicates his presence and desire for service with a sign-on record. The record is in the following form:

/ID

This command can be used for job accounting information for the installation. Additional information will be supplied in the future to indicate how the job accounting information provided by the system can be utilized. No routine will be provided with the system to format and output accounting information collected on disk.

The system acknowledges its readiness to service this user by responding, "STATE ACTIVITY". The user then enters a control command, stating his desired activity, that is, a unit of work which the system will perform for him. An activity may be of such a nature as to require a file of input from the terminal, or it may be a task which can be performed with the command itself carrying all the necessary information. The description below of defined activities illustrates this contrast:

/INPUT

This activity results in collecting on the input work file all the cards following it. In this state, a data file may be created for filing away, or a source program file may be developed for processing. In this latter case a /JOB must follow the /INPUT to indicate the variable job parameters. (The /JOB command will be described in detail later.)

/UPDATE { name (lock) }

This activity request results in the locating of the named file "name" and a verification of the correctness of the given lock work "lock" before a modification of the file is allowed. Where the name and lock code are not given, the update reference is to the information in the input file.

/SAVE name (lock) { , x, y }

The /SAVE command causes the saving of the program in the user's input file under the name "name" and a lock code assigned by the user.

Not having the parameters x and y causes the entire program to be saved, while having the parameters means that line numbers x through y inclusive are to be saved. The lock code and the program name must be given. The program name is up to six alphameric characters, the first always being alphabetic. The lock code is up to four alphameric characters in length.

/PURGE name (lock)

This command causes deletion from the appropriate library index of program name after verification that the correct "lock" code is given.

/RUN

This command indicates that the job in the input file is to be submitted to a processor.

/OFF

This command causes a signoff from the system with charge accounting terminated.

After the response to "STATE ACTIVITY", acted on immediately by RAX, additional commands are necessary to clarify the particular options desired. These commands are discussed below (Figure 3 indicates which ones are valid under certain activity states):

1. /BAP $\left\{ \left\{ \text{NAME} = \text{xxxxxx} \right\}, \left\{ \left[\frac{\text{NOLIST}}{\text{LIST}} \right] \right\}, \left\{ \left[\frac{\text{NODECK}}{\text{DECK}} \right] \right\} \right.$

   This command will cause the Basic Assembler Language processor to operate on the following file when a /END RUN command is given. The optional parameters must be separated by commas and are interpreted similarly to those of the /FTC command. This command is allowed from a remote terminal, in which case the "DECK" and "NAME = xxxxxx" options are not available.

2. /CANCEL

   This command instructs the system to suspend processing on this activity. During output the terminal is periodically polled to allow the exercise of this option. Canceling of /INPUT activity does not destroy the input file.

3. /CHANGE $\left[\begin{matrix} \text{x, y} \\ \text{x} \end{matrix} \right]$

   /CHANGE is a specific request given in the /UPDATE state to add, delete, or correct line numbers x through y. A single parameter in a /CHANGE causes the lines following to be inserted after line number x. Parameters x, x cause a deletion of line x and insertion of the following lines at that point. x, y given in the /CHANGE

causes deletion of that segment of code and replacement by the following cards, if there are any.

4. /DATA

   Following this command are data lines to be used by the object program. This command signals the language processor that it is to terminate its compilation activity.

5. /DELETE

   This command causes the previous record to be removed from the input file.

6. /DISPLAY x,y, $\left\{ \text{name} \right\}$

   This is a request for the system to locate the file indexed under "name" and to display the lines numbered x through y on the terminal output device. If any of the lines exist in the sequence x through y, they are sent to the terminal. Otherwise, an error message is transmitted to the user. Not giving a file name causes a display of records from the terminal input file.

7. /END $\left\{ \text{RUN} \right\}$

   This command informs RAX that the input stream for the activity is complete. The system will return to "STATE ACTIVITY" status. The optional "RUN" parameter serves as a request to queue this input file for compilation and execution before a return to "STATE ACTIVITY" status. This card is not written on the input file.

8. /FTC $\left\{ \text{NAME=xxxxxx} \right\} \left\{ \right\}, \left[\frac{\text{NOLIST}}{\text{LIST}} \right] \left\{ \right\} \left\{ \right\}, \left[\frac{\text{NODECK}}{\text{DECK}} \right] \left\{ \right\} \left\{ \right\}, \left[\frac{\text{NOREF}}{\text{REF}} \right] \left\{ \right\}$

   This command causes the FORTRAN processor to operate on the following file when a /END RUN command is given. The optional parameters will be separated by commas and are interpreted as in BPS FORTRAN, except that the REF and NOREF are used instead of MAP and NOMAP. This command is allowed from a remote terminal, in which case the "DECK" and "NAME=xxxxxx" options are not available.

9. /INCLUDE name

   This is a request for RAX to locate the file indexed under name and to incorporate the contents of the file in the input stream to the FORTRAN processor in place of the /INCLUDE card. This card is written on the current input file.

10. /INSERT name

    This command causes a copy of the file "name" to be placed in the input file beginning at the location of the /INSERT.

11. /JOB $\left\{\left[\begin{matrix}\underline{\text{GO}}\\ \text{NOGO}\end{matrix}\right]\right\}\left\{,\left[\begin{matrix}\text{MAP}\\ \underline{\text{NOMAP}}\end{matrix}\right]\right\}\left\{,\text{TIME}=\text{xx}\right\}$

    This command is necessary from the terminal to indicate that a source program follows. A /FTC command with no optional parameters is forced following this command unless an explicit /FTC or /BAP command is given. The "GO" option indicates a desired execution, while "TIME = xx" is the limit for execution time in minutes, and the "MAP" option requests a loader storage map.

12. /PAGE

    This indicates a need to see the next page (eight lines) of a program in 2260 update mode of activity. The page will be displayed on the 2260 with line numbers appended on the right-hand end of the line. Corrections can then be entered on either the original line or lines 9-12. Correction lines will be indicated by a user-entered # sign after the last significant character of the statement and before the line number.

Job Processing Languages

Problems can be stated to RAX in two programming languages: Basic Programming Support FORTRAN IV and Basic Assembler Language.

FORTRAN IV is the widely accepted and used problem-oriented language which follows the familiar notation of mathematics. It allows a quick and simple statement of a problem in terminology easily learned by the scientist, engineer, mathematician, and similar users. As a specially designed language to simplify man-machine interaction, it is more an efficient language for the user to state his problem solution than a language to utilize the full machine capabilities.

By contrast, the Basic Assembler Language (BAL) allows specific machine characteristics to be exploited by use of specific discrete machine instructions. BAL simplifies the use of machine instructions by allowing easier mnemonic references to machine operation codes and symbols used in a program. Since all BAL programs execute in the problem state, no privileged instructions can be used. No supervisor call (SVC) instructions may be used. All I/O must be performed by FORTRAN programs.

The two languages are described in detail in the IBM System Reference Library manuals, IBM System/360 Basic Programming Support FORTRAN IV (C28-6504) and Basic Assembler Language (C28-6503). Because of the availability of these manuals no further reference will be made to the capabilities available to users of the languages. Each language's range, restriction, and precision are covered in the referenced publications.

SYSTEM OPERATION

Internal Operation

The technique of operation used in this time-sharing system is shown in Figure 4, a diagram of a single job flow.

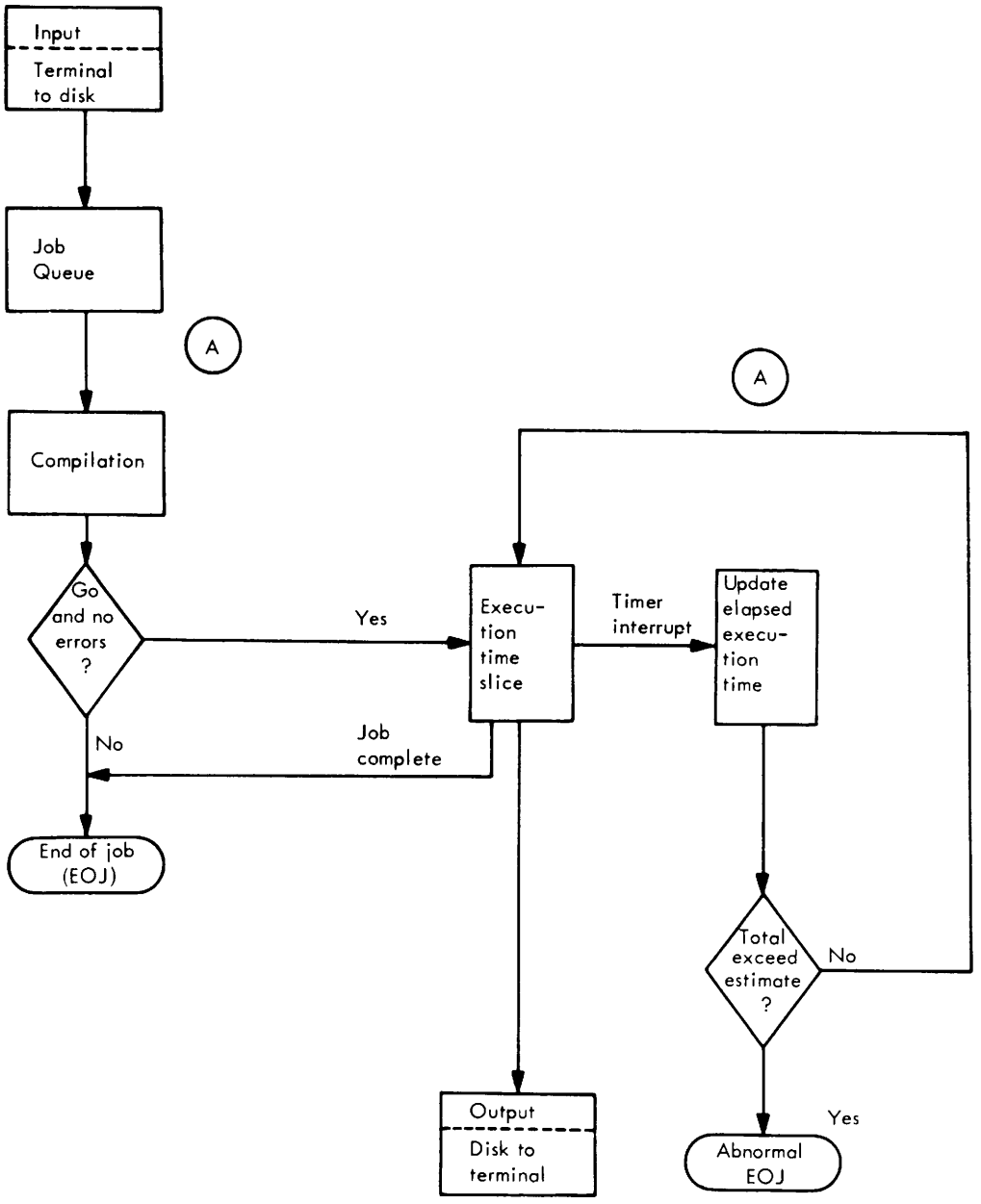

Figure 4. Internal systems operation — single job flow

Essentially, the procedure is to accumulate each line of a user's program until the entire program, consisting of source statements and data, is received. Following termination of input, the job is queued for compilation and/or execution. A scheduling algorithm determines when processing unit access is provided, and at that time, the user's program is compiled to its completion. The system then returns control to the system supervisor to determine whether an execution time slice is to be given to the in-core user. Not finding any errors and receiving a "GO" request causes an increment of execution time to be given to the program. After the time slice is over, the system checkpoints the user program, if more execution is necessary, and then goes to the scheduler to determine the next user. While the compilation/execution is taking place, terminal input and output continue to take place.

One 2311 used in the system holds the RAX input (SYSIN) function. To build up this SYSIN file, the system collects characters, one after the other, in a terminal buffer until an entire source statement line is formed. Then a "buffer-to-disk" operation is performed to build up the input file on disk. When the job is accepted for compilation by the system, the SYSIN file is the source of the program fed to the compiler for translation.

User Operation

The following examples illustrate procedures for entering a job into the RAX system. The examples show how a user might proceed in entering, modifying, and saving a FORTRAN program. They are oriented primarily around the use of the 1050 keyboard terminal, although use with the 2540 and 2260 is not precluded. Differences existing in use of the different terminals are described. The capabilities illustrated will be available in RAX, but the messages, format, etc., are not of the final form.

Figure 5 (job input) and Figure 6 (job update) show the RAX terminal commands and program statements for a user program. Comments to the right of some statements clarify what is happening. Alongside each FORTRAN statement is an internal statement number generated by RAX. This is used to reference updates. It is not printed out for the user at the time of initial entry but is shown to indicate the internal numbering and how it is used.

Figure 5 shows a typical sequence of events during a terminal session. At the beginning of the session the user signs on with /ID. This starts his job accounting. The system then responds with a request for the user to state his activity. /INPUT would indicate building up an input file, with the following /JOB indicating that a processor (FORTRAN) is to be called. Other job parameters may be given on this card. Following the /JOB card is the FORTRAN program. After line number 6 the user gives a /DISPLAY to verify line numbers. Typed out for him immediately is the name of the file, the line number, and then the statement(s).

Terminal and FORTRAN Statements	Internal Statement * Number	Comments
/ID		(starts user accounting routine)
"STATE ACTIVITY"		(system response)
/INPUT		(to indicate activity is to build up an input file)
/JOB	(1)	(to indicate job parameters)
A = 1.	(2)	
B = 2.	(3)	
C = 3.	(4)	
-----	(5)	
-----	(6)	
/DISPLAY 2, 3		(immediately executed and not recorded on SYSIN)
L.0002 . . . . . A = 1.		(line 2 & 3 typed out)
L.0003 . . . . . B = 2.		(line 2 & 3 typed out)
/DELETE		(delete line 5)
-----	(6)	(new line 5)
-----	(7)	
-----	(8)	
END	(9)	(normal FORTRAN END)
/INCLUDE ABC	(10)	(included source for subroutine retrieved at compilation time)
-----	(11)	
-----	(12)	
END	(13)	
/DATA	(14)	
/INCLUDE DATA	(15)	(include file of data previously stored)
/END RUN	(16)	(request for queuing and execution)

*Statement # assigned internally by RAX but not typed out for user.

Figure 5. Job input

After the /DISPLAY command is satisfied, the system reverts to the input mode to accept additional statements. /DELETE at this time deletes line number 6, not the /DISPLAY command. A new line 6, and succeeding lines, can be entered until the FORTRAN end statement terminates the program. At this point /INCLUDE ABC is entered. This line is queued on disk as another addition to the source program. At compilation time, when the compiler receives this line, the monitor locates the file ABC and transmits it one line at a time for processing. By /INCLUDE a user can insert saved statements inline and effectively exceed the installation-defined limit of statements in his input file. The

/DATA is used to indicate to the processor that data follows. In this example the data is included from a previously saved data file. /END RUN causes queuing of the program for compilation. Absence of the RUN parameter would cause a system request to "STATE ACTIVITY". After the task is completed, the system issues a request for "STATE ACTIVITY". Figure 6 shows a request for an /UPDATE of the input file. A /CHANGE 5,5 is given to eliminate line number 5. The following source statement is inserted in its place. Another change eliminates lines 10-12 but five lines are inserted to replace them. /DISPLAY is desired suddenly and is implemented immediately, displaying the unchanged file. A /END terminates the process and causes the actual updating to take place.

```
"STATE ACTIVITY"
/UPDATE                                  (requested activity)
/CHANGE 5,5                              (removed line 5)
      -----                              (insert this line in place of 5)
/CHANGE 10,12                            (remove line 10-12 inclusive)
      -----                              (insert these 5 lines where line
                                          10-12 was)
      -----
      -----
      -----
      -----
/DISPLAY 10,12                           (displays old line 10-12)
L.0010 = --------
L.0011 = --------
L.0012 = --------
/END RUN

"PRE-EDIT IN PROGRESS"                   (systems message)
"UPDATE PACKAGE ACCEPTED"                (systems message)
"UPDATE COMPLETE"                        (new file built with change
                                          cards)
"STATE ACTIVITY"                         (return to activity state)
/SAVE NEWN (LOC)                         (save new file)
"NEWN SAVED"                             (systems message)
```

Figure 6. Job update

The system's output message is typed out as each updating phase occurs. Finally, an "UPDATE COMPLETE" indicates that the change cards have been used to build up a new file. The system asks whether further action is desired by responding "STATE ACTIVITY". The user requests that his input file be placed in the file library by entering the command /SAVE. The system responds that the /SAVE has been performed.

TIMING

The nature of a time-sharing, slicing operation is such that it is not possible to supply definitive timings. For accurate and valid timings one would have to know the number of statements per program, the number of terminals active, the mix of compile or executes, whether the online I/O devices are running, and a host of other details.

The only information that can be supplied on throughput timings is contained in Figures 7 and 8, which show the operation of the Lockheed-Georgia RACF* system running on a Model 50. A system summary, by terminal, of various operating statistics and a queue delay summary are given. The number of lines in each program is not available to develop throughput figures. These figures can be considered fairly representative of how RAX will perform on short jobs. (The maximum input is under 500 cards and the average is considerably less.)

Figure 7 shows the system summary, by terminal, for the batch terminal (2540) and twenty-four 1050 terminals. The bottom two lines show a summary of activity for the batch and twenty-four terminals and for just the twenty-four 1050 terminals.

Column 1 shows the number of jobs input on this day over a 13-hour-and-56-minute period. The number of jobs completely entered is indicated in column 3. After a complete job is entered, it can be canceled out by the user before going to the compiler. Column 5 shows the number of jobs going to the compiler, while column 8 indicates the jobs that compiled, did not have disastrous compile errors, and had an execution request.

The last three columns in the accounting report give pertinent job times for a particular terminal. Column 9 shows net execute time and the sum of all the time slices of execution accrued by a program, while column 10 highlights the elapsed time between the start and end of execution. Total processing unit seconds are the sum of columns 6-9 and are shown in column 11.

*The prototype of RAX.

DAY BEGAN AT 0744/JOB RECEIVED AT 2140.

UNIT	(1) NUMBER JOBS INPUT	(2) INPUT TIME MINUTES	(3) NUMBER JOBS QUEUED	(4) QUEUE DELAY SECONDS	(5) NUMBER COMPILES	(6) COMPILE TIME SECONDS	(7) AVERAGE COMPILE SECONDS	(8) NUMBER JOBS IN EXECUTION	(9) NET EXECUTE SECONDS	(10) GROSS EXECUTE SECONDS	(11) TOTAL CPU SECONDS
BATCH	70	79.7	67	274.2	67	992.2	14.8	41	3130.8	6698.5	4123.0
01	32	140.4	30	134.1	30	249.4	8.3	28	897.6	1675.6	1147.0
02	16	45.7	16	59.4	16	79.7	5.0	12	295.6	545.3	375.3
03	30	203.0	25	114.2	25	261.0	10.4	21	302.1	303.4	563.1
04	12	55.8	11	108.4	11	78.3	7.1	11	612.5	930.0	690.8
05	39	121.2	38	172.8	38	165.8	4.4	34	600.1	1028.5	765.9
06	12	99.0	10	57.9	10	89.6	9.0	8	7.3	8.0	96.9
07	28	75.5	24	138.2	24	101.1	4.2	22	24.0	25.3	125.1
08	20	146.8	18	91.6	18	137.1	7.6	16	661.8	1453.7	798.9
09	44	111.2	39	139.5	39	331.1	8.5	36	2389.1	4286.2	2720.2
10	0	0.0	0	0.0	0	0.0	0.0	0	0.0	0.0	0.0
11	32	245.3	28	163.5	28	216.3	7.7	24	75.3	76.9	291.6
12	21	90.3	17	359.3	17	98.6	5.8	12	1385.8	3227.2	1484.4
13	25	129.6	22	86.6	22	155.9	7.1	20	142.8	351.0	298.7
14	15	114.9	15	76.6	15	160.7	10.7	15	7.9	9.1	168.6
15	33	172.1	30	190.6	30	259.4	8.6	25	299.6	1757.7	559.0
16	20	107.0	17	31.9	17	89.6	5.3	15	814.1	1865.5	903.7
17	25	102.9	20	217.3	20	85.5	4.3	19	205.7	306.4	291.2
18	24	142.0	22	147.4	22	125.6	5.7	17	9.7	10.6	135.3
19	34	155.4	33	153.8	33	190.6	5.8	31	20.1	22.5	210.7
20	9	19.4	9	71.6	9	56.1	6.2	9	57.6	58.1	113.7
21	11	102.1	11	18.8	11	104.7	9.5	8	15.1	15.7	119.8
22	33	80.3	28	85.3	28	99.4	3.5	22	644.9	1401.4	744.3
23	43	67.7	36	308.5	36	117.8	3.3	30	8.8	11.6	126.6
24	3	9.0	2	4.9	2	11.8	5.9	2	14.0	1825.4	25.8
		HOURS		MINUTES		MINUTES			MINUTES	MINUTES	MINUTES
TOTAL	631	43.6	568	53.4	568	71.0	7.5	478	210.4	464.9	281.3
RACF	561	42.3	501	48.9	501	54.4	6.5	437	158.2	353.3	212.6

Figure 7. System summary by terminal

AVERAGE QUEUE DELAY = 5.9 SECONDS HIGH QUEUE DELAY= 129.9 SECONDS

384 QUEUES BELOW AVERAGE 117 QUEUES ABOVE AVERAGE

TIME INTERVAL	JOBS QUEUED
0.00 - 6.50 SECONDS	393
6.50 - 12.99 SECONDS	52
12.99 - 19.49 SECONDS	18
19.49 - 25.98 SECONDS	15
25.98 - 32.48 SECONDS	6
32.48 - 38.97 SECONDS	5
38.97 - 45.47 SECONDS	3
45.47 - 51.96 SECONDS	3
51.96 - 58.46 SECONDS	1
58.46 - 64.95 SECONDS	0
64.95 - 71.45 SECONDS	0
71.45 - 77.94 SECONDS	0
77.94 - 84.44 SECONDS	2
84.44 - 90.93 SECONDS	0
90.93 - 97.43 SECONDS	1
97.43 - 103.92 SECONDS	0
103.92 - 110.42 SECONDS	1
110.41 - 116.91 SECONDS	0
116.91 - 123.40 SECONDS	0
123.40 - 129.90 SECONDS	1

Figure 8. Terminal queue delay summary

BIBLIOGRAPHY

IBM System/360 BPS FORTRAN IV Language (C28-6504)

BPS Basic Assembler (C28-6503)

BPS Operating Guide for Basic Assembler and Utilities (C28-6557)

BPS FORTRAN IV Programmer's Guide (C28-6583)

Printed in U.S.A. H20-0266-0

IBM

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601
(USA Only)

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
(International)