Copyright © 1977 American Telephone and Telegraph Company
The Bell System Technical Journal
Vol. 56, No. 2, February 1977
Printed in U.S.A.

1A Processor:

Control, Administrative, and Utility Software

By G. F. CLEMENT, P. S. FUSS, R. J. GRIFFITH, R. C. LEE, and R. D. ROYER

(Manuscript received July 16, 1976)

System software provided with the 1A Processor simplifies the task of using input/output devices, manages a portion of system time and memory resources, provides control over program interfaces, and provides an interactive facility that permits examination of system behavior. The impact of this software has been to simplify initial development of switching systems using the 1A Processor, to facilitate software maintenance, and to provide flexibility in adding system features. This article describes the capability and organization of the 1A Processor control, administrative, and utility software.

I. INTRODUCTION

Control, administrative, and utility software has been developed for the 1A Processor that simplifies the task of using input/output devices, managing system time and memory resources, providing control over the many program interfaces that exist, and finally providing an interactive facility that permits analysis of system behavior. Figure 1 provides a high-level block diagram of the manner in which the control, administrative, and utility features interact and fit into the software system.

A maintenance control structure has been provided with the 1A Processor to control system time and memory resources used by maintenance programs. These programs are executed concurrently with call processing. This facility allocates the resources associated with the maintenance function based on a priority arrangement, provides initialization between disjoint time intervals of program execution, and also provides a collection of general-purpose control functions. A standard interface provides access to these functions as well as to other control

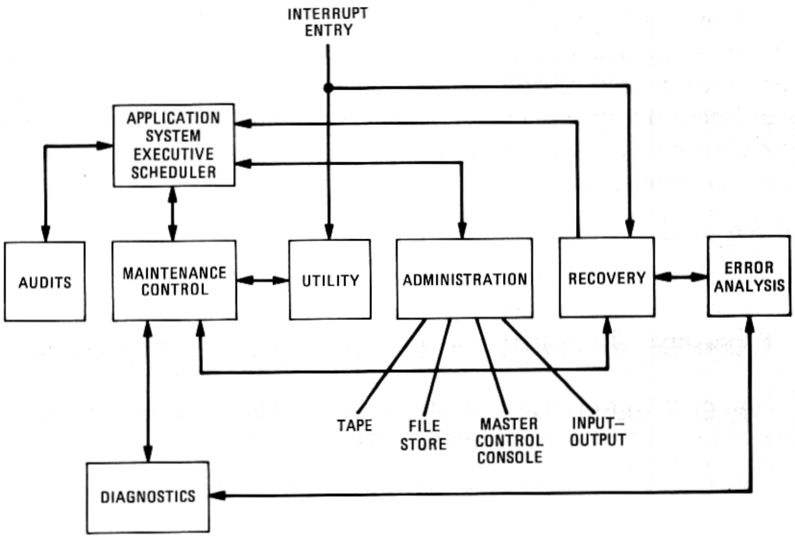


Fig. 1—1A Processor software.

features in order to simplify program design and improve reliability. A paging control feature manages access to the programs, which are stored in lower-cost disk memory. The paging controller manages the interaction between paged routines to insure that they are resident in core before they receive control. It also performs run-time link-editing functions to prepare newly loaded programs for execution.

The administrative features of the system were created to simplify the various programming tasks and to insure the most reliable design. The design relies on the use of standard interfaces that allow the using programs to be isolated from the details of the software or hardware functions being performed. Features have been provided for interfacing with input/output devices, the file store, tape subsystem, and the master control console.

Finally, utility features have been provided to allow extensive on-site analysis of system behavior. The features provided are similar to those in powerful laboratory utility systems and simulators. This is the first time a comprehensive set of utility functions has been included in a Bell System electronic switching system.

II. CONTROL SOFTWARE

2.1 Control software interfaces

The 1A Processor control programs allocate time and memory to programs called clients. The clients include: (*i*) maintenance programs that perform actions deferred from interrupt-level maintenance actions,[1] e.g., if, on interrupt-level, a program store is removed from service, a deferred action would be to diagnose the store; (*ii*) programs that execute various actions requested by the craft force, e.g., requests to reconfigure or diagnose the system or to perform the immediate utility functions described below; (*iii*) routine maintenance-related programs, such as processor error analysis or audit programs that verify the integrity of data structures; and (*iv*) library programs. Library programs are an open-ended set of functions that are added to the system on a temporary or special-use basis. Typical examples are programs to perform special hardware tests, such as those required when additional units are being added to an existing office. Many of the clients are common to all applications of the 1A Processor, e.g., a call-store diagnostic program. Clients may, however, be specific to a given application such as the program to diagnose the switching equipment within No. 4 ESS.

The overall relationship of the common control software to other portions of the system is depicted in Fig. 2. The main executive program cyclically dispenses control to a set of programs. Each of these programs executes for a period of time, or "segment," that is limited by programming standards (enforced by run-time hardware checks) to a specified maximum duration. A typical program would require many segments

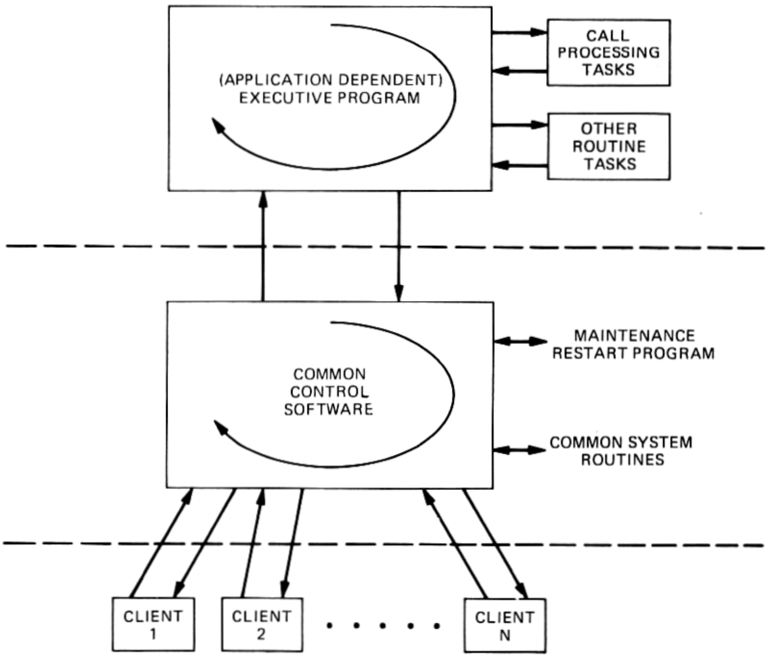


Fig. 2—Control software interfaces.

to run to completion. When the common control software receives control, it activates client programs in sequence (giving each a "segment" of time) until it has exhausted all the real time allocated to it or until no client requires a time segment. The control software provides initialization, execution, and cleanup routines to assist the client programs. Examples of these include routines that transfer nonresident client programs from magnetic tape or disk to core memory, routines that allow a client to suspend execution until a specified condition has been met, and routines to request the execution of another client.

2.2 Control software algorithms

2.2.1 Client scheduling

The first task of the control software is to determine which client, of those that are in competition for system resources, is to be scheduled for execution. If memory resources are available and requests for client execution have been received, the control software schedules the highest priority client that can be executed concurrently with all other clients that are already scheduled. Subsequent paragraphs will elaborate upon client scheduling.

Clients are identified uniquely by three attributes: class, job type, and job number. The class attribute groups clients that are generally similar. Clients from different classes can always be executed concurrently while clients from the same class may interfere with one another. In addition, the class attribute is the first determinant of client priority, since the scheduling algorithm seeks clients from one class first and then from the next class in order of classes as defined at program assembly. As an example of classes, all maintenance programs form one class, the utility programs are a second class, and library programs are a third. The job type within a class is a means of further establishing client priority since, within classes, clients are sought in order of job type. Two sample job types would be manually requested diagnostics and routine exercise programs whose execution is triggered at prespecified times. In this instance, the former job type would normally be higher priority; i.e., a manually requested diagnostic would be executed before a routine exercise. The relative priority of job types is predetermined at assembly time. The final determinant of priority is the job number. Within job types, the control software scans for "job requests" in order of job number. Again, this priority is fixed at assembly time. This would cause a central control diagnostic, say, to be scheduled before a program store diagnostic.

The remaining condition to be satisfied by the scheduling algorithm is to check for potential conflicts with clients that have already been scheduled. The potential for conflict exists because the maintenance programs alter the system configuration when first receiving control and assume that the configuration is unchanged from segment to segment. The overhead of completely reestablishing that same configuration for each time segment would be prohibitive. For example, a request to diagnose an unduplicated program store causes the maintenance program to copy the contents of that store into a spare program store. The spare store is then put in service in place of the store to be diagnosed. Copying the contents of a program store takes many segments. A potential conflict would occur in this example if a program store bus diagnostic were to be executed concurrently, for it would attempt to establish various store-bus configurations conflicting with those needed for the store diagnostic. The 1A Processor resolves this conflict by establishing blocking rules that prevent the concurrent execution of clients with conflicting unit configuration requirements. The structure of the rules, i.e., which units are affected by each client and which unit types are interfering, is established at assembly time. The scheduling algorithm checks for this blockage before activating a client that has been requested. A client is not scheduled while it is blocked.

2.2.2 Client supervision

The second basic task of the control software is to furnish run-time support to the clients that are scheduled for execution. These services include initiation and termination of execution at both the job and segment level, providing clients access to common service routines, and loading and linking pages for client programs that are not core resident.

Job initiation consists of assigning temporary memory to a client, loading that memory with input data received with the request to activate the client, loading the first page of the client program into core memory if necessary, and, finally, transferring to the start address of the client.

During client execution, control software service routines allow the client to specify special conditions that are to be established before it is given another execution segment. These conditions include special register settings, such as interrupt inhibits (which are restored to normal outside the client's segment), completion of output or disk transfer operations, or the passage of timed intervals. In addition, control routines intercept attempts by system routines to return control to clients that have activated them. This is required since the control program may have suspended or aborted client execution between the client's call to a system routine and the response of that routine. For example, a client may initiate a disk transfer. When the disk operation is complete, the client would receive control from the administrative program in order to record the completion of the operation. If, in the mean time, the client execution has been prematurely terminated because of a system error, then this attempted transfer from the administrative program would cause an improper flow of control. Therefore, the control program provides an interface between system programs and a client, intercepting all such responses and passing them to the client only if it is still in execution.

Client supervision also involves providing pages of nonresident programs as required. This is accomplished by intercepting program transfers off the currently loaded page, temporarily suspending client execution, transferring the necessary client program elements from disk to core, linking the elements together to form a new complete page, and reactivating the client.

Finally, client supervision requires providing a mechanism for client termination. Normally a client notifies the control program when it has completed execution. Under these circumstances, the control program simply releases resources that had been assigned to the client and restores hardware control registers and software status indicators to their normal values. A more complex situation arises when an abnormal condition occurs, such as a processor maintenance interrupt. This requires the control program to analyze the situation to determine whether the client either caused the abnormality or could have been adversely affected by the disruption. In either case, the client is terminated. On the one hand, this protects the system from potential further disruption. On the other, it guards against invalid results from clients. Before termination, the client is given control *once* at an abnormal termination routine that it has provided to handle the abort. The client attempts to restore the system to a normal state if it has established abnormal conditions. Should this "abort processing" by the client cause further disruption, the control program completely terminates client execution and relies upon recovery and integrity programs to restore normal system operation.

2.3 Control program organization

The control program consists of seven major sections, as shown in Fig. 3. Each of these program sections consists of reentrant, table driven code.

This structure has provided efficient and highly reliable operation. This is important, of course, since inefficiency in this control program could waste a significant fraction of the processor capacity. Furthermore, only the data tables need be changed to alter the client classes, job types, rules for analysis of abnormal situations that require early client termination, and most other characteristics described above. This means

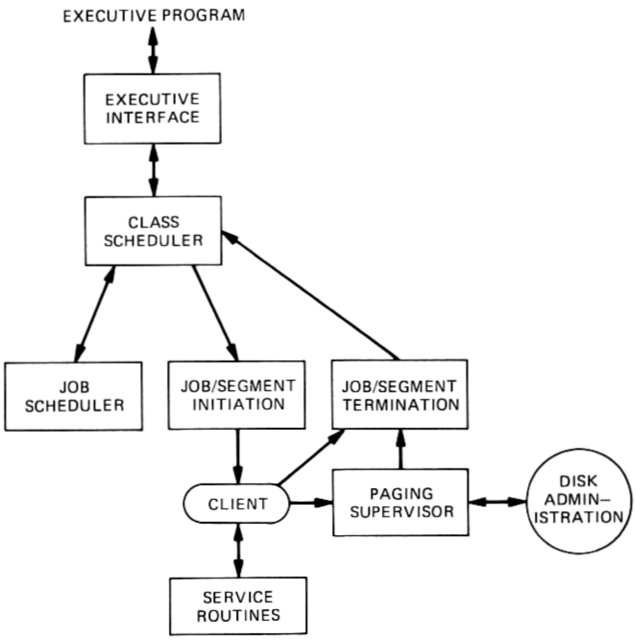


Fig. 3—Control program organization.

that relatively simple and easily identified changes allow this software to be adapted for rather different applications of the processor.

The basic data tables are illustrated in Fig. 4. There is one class table, one job-type table for each class, one set of job-request flags for each job type, one job-directory table for each job type, and one program-directory table. An extensive set of assembly language macros facilitates building and maintaining these tables. For paged programs, additional macros and pseudo-ops allow programmers to describe the program structure while writing software with minimal consideration of paging operations.

III. ADMINISTRATIVE PROGRAMS

The 1A Processor administrative programs provide interfaces between the software and four types of hardware: the file stores, tape units, input/output devices, and the master control console. The hardware dependent code is concentrated in these programs. A macro is provided for each function performed by an administrative program to produce a standardized interface between programs. Each macro is expanded by the assembler into instructions to provide the data needed for the function and a transfer into the administrative program routine to perform the function. For operations that require significant amounts

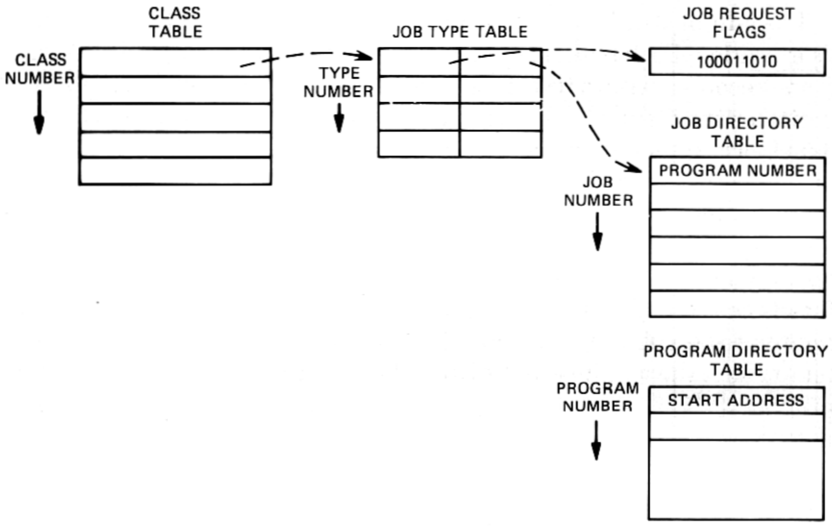


Fig. 4—Control program data tables.

of time for completion, such as data transfer to a file store, two exchanges of control occur between the using program and the administrative program. The first of these is a transfer of control from the using program to the administrative program. This is a request by the using program. Control is returned by the administrative program as soon as it has noted the request and associated control information and started the hardware action. The calling program is then able to continue processing if it does not depend upon completion of the request. Upon completion of the hardware action, the administrative program is called and transfers control to the using program at an address provided with the request. The using program records the completion of the request and then returns control to the administrative program.

3.1 File store administration

As described in Ref. 2 data are transferred between disk memory and core memory under control of the file store. The disk administration program provides the interface between the using programs and the duplicated file stores. The functions that it provides are: (*i*) read either disk, (*ii*) write both disks, (*iii*) write a specific disk, and (*iv*) read a specific disk. The first and second of these are used most frequently. The write or read operations that specify a particular disk are used by programs with duplicated data bases that are altered one copy at a time. This preserves the internal consistency of one copy while the other copy is undergoing a time-consuming alteration.

The administrative program tests the validity of a disk request and its execution. Each block of data on disk has an identifier associated with it and recorded within the block. A request to read or write a data block indicates the identifier that the using program has associated with the data block. If that identifier does not agree with the one recorded on the disk, an error condition is detected by the administrative program for write requests or by the file store for read requests. In either case, the administrative program notifies the using program that the requested operation cannot be completed. The administrative program also checks for other error indicators in the file-store hardware. The program attempts to repeat the operation if errors are indicated. If multiple attempts fail, the administrative program reports this failure to the using program. If the request has been completed without error, this is reported.

3.2 Tape administration

The tape-administration program provides the interface between using programs and the tape-unit hardware. The administrative program provides functions: (*i*) secure a tape for a using program, (*ii*) read a tape record, (*iii*) write a tape record, (*iv*) write an end of file, (*v*) position the tape as required, e.g., rewind to the beginning of the tape, move to the beginning of the next file, and locate a specific file, and (*vi*) release a tape. When a using program secures or releases a tape, the administrative program issues directives to the craftperson to mount or dismount a tape if necessary. It also receives the craftperson's input message that indicates when the required action has been completed.

The tape-administration program tests the validity of requested operations by comparing the tape specification contained in the using program's request to information obtained from a header label that is recorded at the beginning of each tape. It also repeats operations if error indicators are set by the tape hardware. Finally, the tape-administration program assures highly reliable recording of telephone billing data by automatically switching to a previously secured standby tape if a failure or end-of-tape occurs.

3.3 Input/output administration

The input/output administration program provides the interface between using programs and the input/output hardware and terminals. The 1A Processor is equipped with many input/output channels. There are also many programs that receive inputs or generate outputs. There is, however, no fixed association of programs and input/output channels. Many programs may be simultaneously directing output to the same logical channel. On the other hand, each member of a series of messages from a craftperson may be destined for a different program. Moreover, the format in which data are represented within programs is not convenient for manual interpretation and the converse is true also. Therefore, the input/output administration program must perform several functions in addition to providing an interface between using programs and the input/output hardware. On input, the program parses the input message to create a more readily usable format for user programs and determines which using program is to receive the message. On output, the program translates the data to a format that is more conveniently read by a craftperson and combines independent message streams for each channel. Ancillary functions allow messages to be rerouted to circumvent hardware malfunctions and allow outputs for a given channel to be monitored by other channels. The latter function allows the craft force flexibility in shifting allocation of responsibilities between office personnel.

The input/output hardware and software are illustrated in Fig. 5. Note that format conversion and input message destination are governed by data tables called "catalogs." This means that changes in these may be introduced without changes to the input/output administration program. Output catalogs may be included in the using program and, thus, may be entirely under its control. Since the input catalogs are needed to determine the destination for input messages, they must all be available to the administration program. Using programs may, however, add or delete input catalogs at run-time.

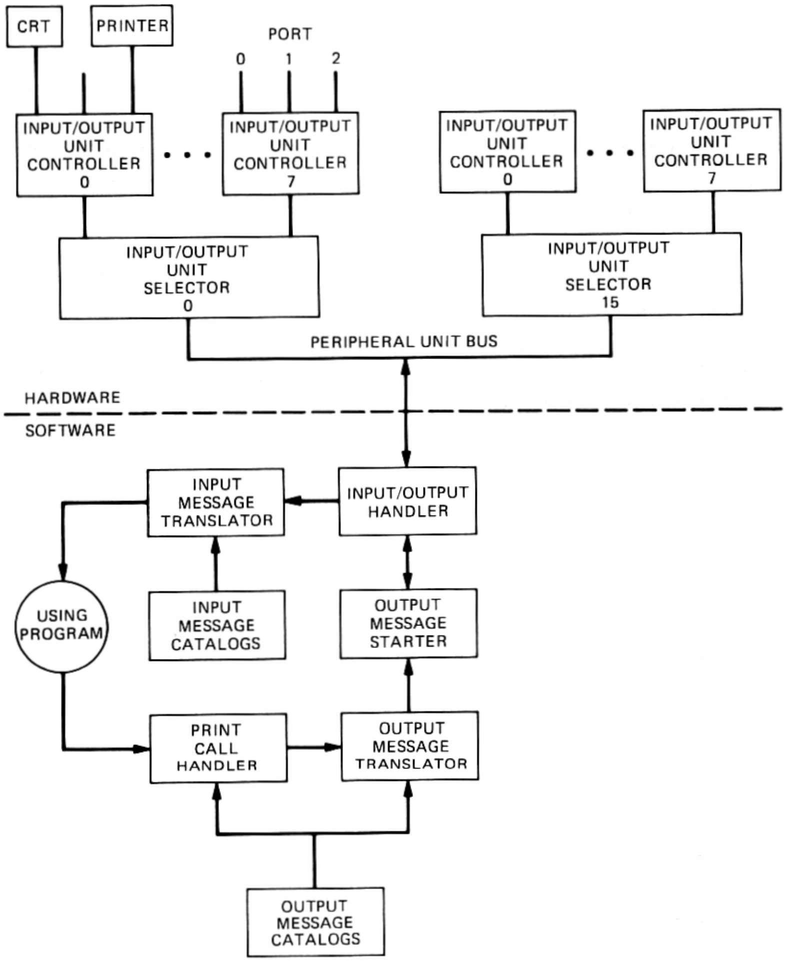
The messages associated with the 1A Processor conform to standards that are common to all Bell System electronic switching systems. The typical input format consists of a verb, noun, and optional modifier. Each input is a directive to the system to perform an action (specified by the verb) on a portion of the system (specified by the noun) with options (specified by the modifier). The verb and noun together determine the program that will receive the message. Data within the message is conveyed by key word and argument rather than by position. As an illustration, the following message directs the tape-administration program to allow (abbreviated as ALW) the use of tape-unit controller 2 (TUC 2) in the read-only mode (RO).

```
ALW:TUC 2:RO
```

Output messages follow similar formats.

3.4 Master control administration

The master control console administration program interfaces with the hardware master control console and processor peripheral interface.[2] One major function that it performs is to read keys and operate lamps on the master control console for other programs. It does this through


CRT
PRINTER
PORT
0
1
2
INPUT/OUTPUT UNIT CONTROLLER 0
INPUT/OUTPUT UNIT CONTROLLER 7
INPUT/OUTPUT UNIT CONTROLLER 0
INPUT/OUTPUT UNIT CONTROLLER 7
INPUT/OUTPUT UNIT SELECTOR 0
INPUT/OUTPUT UNIT SELECTOR 15
PERIPHERAL UNIT BUS
HARDWARE
SOFTWARE
INPUT MESSAGE TRANSLATOR
INPUT/OUTPUT HANDLER
USING PROGRAM
INPUT MESSAGE CATALOGS
OUTPUT MESSAGE STARTER
PRINT CALL HANDLER
OUTPUT MESSAGE TRANSLATOR
OUTPUT MESSAGE CATALOGS
Fig. 5—Input / output system.

a matrix of scanner and signal distributor points that is connected to the console. The program maintains a call-store image of this matrix to be used as backup in case of hardware failure. Another major program function is a periodic scan, every 1 second, of the console keys. Any changes are updated in the call-store image and reported to the appropriate client program. A third function is processing manual reports communicated via the power-control switches connected to the processor peripheral interface matrix. Every second the program scans the switches and reports any change to the appropriate fault-recovery program, which will then take an action such as removing a unit from service in response to a key operation. The program also changes the state of the power-switch lamps to inform the craftsperson of the system's action.

IV. UTILITY SOFTWARE

4.1 Utility system objectives

The current systems using the 1A Processor are large in terms of both program size and data base requirements (about one million program and data words.) A very large effort has been required to debug all of the software features and to gather and output various kinds of data that are of interest. Since the program and data base are so large and new features are regularly added to the program, there will probably always be some latent errors in the system program. Because of this, a sophisticated utility capability has been included with the 1A Processor to aid in testing the initial and updated programs and, in anticipation of future needs, to sample data to analyze system behavior. A major objective of the utility features has been to safely provide full access to data in the system.

There are many potential uses for such utility features in an operational telephone switching machine. For example, the office craft force may wish to obtain information pertinent to determining why a specific software procedure is failing. Alternatively, information concerning the use of a particular trunk or trunk group in an office or information about the way in which the office data base is accessed or changed may point to the solution of a problem. The utility system allows these functions to be conveniently and safely inserted on a temporary basis into the operational program, and to be removed when the necessary information has been gathered.

The utility programs have been designed in a modular fashion for ease of modification and to limit the effect of the addition of new facilities or functions on previous features. Special hardware in the 1A central control facilitates efficient implementation of the utility system by providing the mechanisms for the utility software to receive and return control without disrupting the normal flow of control within the system.

4.2 Utility features

The capabilities of the utility package are discussed below.

4.2.1 Conditional utility functions

The WHEN function of the utility system permits the interruption of program flow and the transfer of control to utility software for the execution of the specified utility operations when specific conditions have been met. The conditions permitted for the WHEN function are:

(*i*) When a specific program address is reached.
(*ii*) When a specific memory location is read or written.

The first condition is implemented by using a special instruction that is loaded by the utility software in place of the instruction normally resident at the program address. This instruction saves the system registers and transfers control to the utility software. The second condition uses the matching circuitry of the central control to compare data addresses to the specified value. A successful match causes the central control to save its registers and transfer to the utility software.

Additional control can be maintained over the execution of utility functions in a WHEN clause by the use of the IF and ELSE functions. The IF function controls the utility operation on the basis of the evaluation of a logical expression included in the command. The following features are permitted in the evaluation:

(*i*) One of six operators may be specified for comparing two memory or register locations.
(*ii*) Several levels of indirect addressing may be specified, with or without indexing, for both sides of the expression.
(*iii*) The evaluation may take into account from one to all bits in the registers or memory locations being evaluated by the expression.

If the expression is evaluated as true, the utility commands that follow the IF clause are performed. However, if the expression is evaluated as false, no functions are performed unless the ELSE command was part of the original message. If ELSE is specified, and the expression is false, the utility commands following the ELSE are executed. A number of complex conditional expressions can be active in the system simultaneously.

4.2.2 Immediate utility functions

The capability is provided in the utility software to permit almost all addressable locations in the system to be displayed on output devices, to be initialized to a specified value, or to be moved from one addressable location to essentially any of the memory systems provided by the 1A Processor. In addition, the capability is provided to freeze large amounts of data on the occurrence of a particular event or to provide control of oscilloscope operation being performed by the office craft people. Most of these features can be performed in direct response to input messages as immediate utility functions or in conjunction with the conditional features of the system described in the preceding section. Each of the above functions are described in more detail in the following paragraphs.

The DUMP function provides the capability of displaying portions of the following addressable 1A Processor locations on an output device:

(*i*) Call and program-store memory.
(*ii*) File-store memory.
(*iii*) Standard label tapes.
(*iv*) Special memory locations provided for the utility user.
(*v*) Central control internal registers.

The DUMP feature permits memory contents to be printed on a system teletypewriter. It is also possible to display data on the master control console. The determination of the address at which the DUMP is to begin can be influenced by both indirectness and indexing options. The capability is provided for displaying from one to all bits in the locations being dumped. Output may be in either binary or decimal format. Through the use of the "DUMP on interrupt" command, data may be selectively displayed on any interrupt level. This selection may be refined down to a dump on any particular interrupt source.

The LOAD function provides the capability of initializing portions of the following addressable 1A Processor locations:

(*i*) Call and program store memory.
(*ii*) Special memory locations provided for the utility user.

The LOAD feature permits up to 128 locations to be initialized. The same address determination and formating features that are provided by the DUMP command are available on LOAD.

The COPY function provides the ability to move data from one storage medium in the 1A Processor to another without affecting the data at the source location. The same address determination and formating features provided for the DUMP function are also available on COPY.

The FREEZE function provides the ability to save the contents of one or more program or call stores (65,536 26-bit words) for later analysis. The contents of the store may be frozen at the time of a program interrupt, before memory is initialized as part of system recovery, or on other occurrences of interest. This function is implemented by maintaining a duplicate memory block for the area to be frozen and, when the event of interest occurs, inhibiting changes to the duplicate memory. This data may be selectively displayed on a teletypewriter or copied onto tape for later analysis.

The SYNC function provides the ability to produce a sync pulse at a coaxial connector on the central control when a desired event occurs. The SYNC command is triggered by the central control match circuit and is used in conjunction with the conditional utility features described previously. It is expected that the SYNC pulse will be essential for hardware repair procedures using an oscilloscope.

4.2.3 Control functions

A data set capability has been designed into the generic utility package to permit message sequences to be created, saved, and manipulated. This procedure permits any messages that can normally be input on the teletypewriter to be saved on disk memory under a name specified by the user. The named sequence of messages can be edited or can be input to the system with one command. This feature permits the craft force to build a library of convenient functions for use in routine procedures and reduces the likelihood of errors due to improper messages.

The utility software has a feature that enables field-operations personnel to "overwrite" or change the program of the 1A Processor. These changes are introduced without disruption of the system, while it is in operation. The altered data is restored to its initial value automatically if a system malfunction occurs. The craft force is allowed to make the change permanent after a test interval has been completed. A set of control features is also provided to permit the craft force to control the initiation and disabling of conditional utility functions.

The utility features included with the 1A Processor provide a high degree of capability for determining how and why the system operates as it does. In some cases, this flexibility, if carelessly used, makes it possible to disrupt system operation. To minimize this possibility, a number of safeguards have been built into the utility functions. For example, addresses used for utility actions must be in predefined ranges specified on the input request. Software controls also insure that utility functions cannot take excessive amounts of real time from the system. If too much time is taken, the active utility functions are automatically deactivated. If invalid program actions are detected, software recovery[1] removes the utility functions. Finally, a mechanism is provided so that utilities can be quickly disabled through manually initiated interrupt at the master control console.

4.2.4 Off-line functions

The off-line features of the utility system provide the 1A Processor with the ability to remove redundant hardware units from normal system operation and to use them to build an off-line system. This off-line system can then be manipulated from the active system using the special features described below as well as the utility conditionals and commands described previously. The testing of this configuration does not interfere with normal activity of the system and permits utility functions to be performed with negligible real-time expenditure from the on-line processing hardware. If recovery from a fault in the active system requires use of a unit in the off-line system, this unit is automatically returned to service. The off-line facilities are designed to be used for the isolation of difficult hardware and software problems that cannot be solved by normal maintenance facilities.[1] The following functions are provided for the off-line configuration:

The start and stop off-line CC functions are the basic control of the off-line CC. Program execution begins with the instruction that is currently contained in internal CC registers and stops upon command from the active CC without disrupting internal CC registers.

The transfer function causes the off-line CC to transfer program control to a specified address. A return option can be specified that later permits control to be returned to the point after the transfer. The return function causes control to be passed to the instruction following the last executed transfer function, which specified a return option. The loop function causes the program in the off-line system to be modified so that a loop is created and control is passed to the top of the loop.

4.3 Utility examples

The following example is provided to briefly illustrate the functions of the utility system provided with the 1A Processor:

```
WHEN:ADR 14000000
IF:ADR 1234, SIZE 6, DISP 4; EQ; 45
DUMP:ADR 1235
ELSE:
LOAD:ADR 1237:25
```

After the instruction at address 14000000 has been executed, the contents of address 1234 is tested. If the 6-bit field (SIZE 6) that is displaced 4 bit positions (DISP 4) from the right-hand end is equal to 45, the contents of address 1235 will be printed. Otherwise, the contents of address 1237 will be set to 25.

The flow of operations that would occur to process this utility request is shown in Fig. 6. Briefly, after it is determined that the input message is a request for a utility operation, any currently active conditional utility functions are deactivated to release memory for the parsing routines that are brought from disk to error-check the message syntax and to build the necessary utility data structures. For this example, if the syntax of the input message was correct, any previously active conditions would be reactivated. The new conditional command would not itself be activated until a specific control request was received by the utility system.

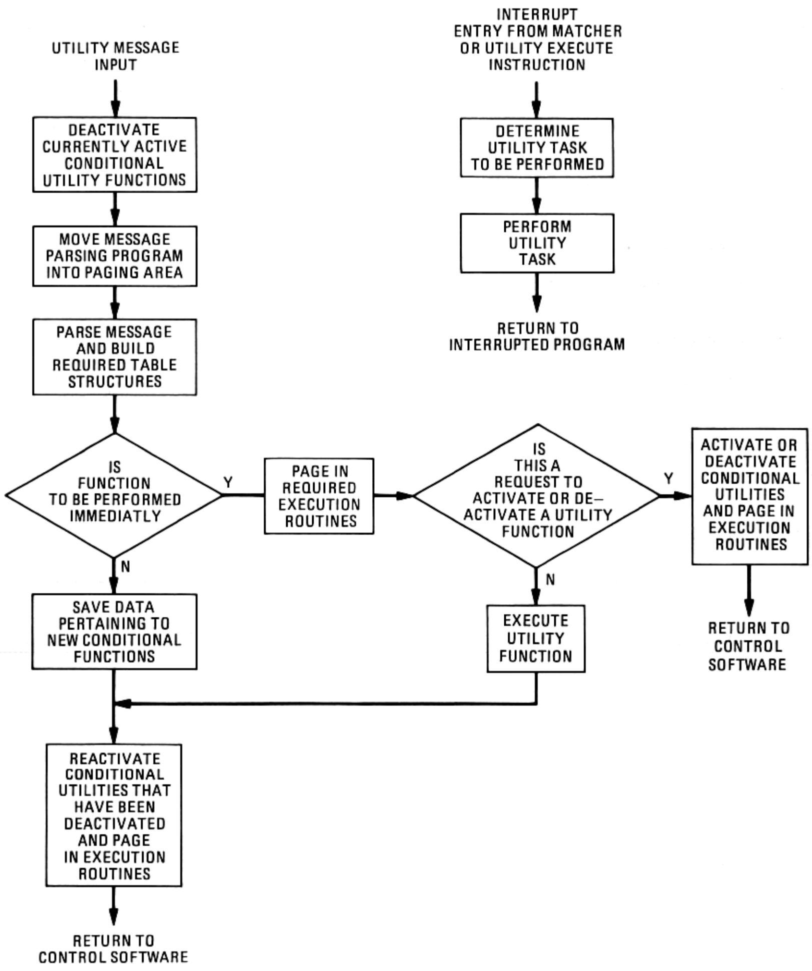

UTILITY MESSAGE INPUT
DEACTIVATE CURRENTLY ACTIVE CONDITIONAL UTILITY FUNCTIONS
MOVE MESSAGE PARSING PROGRAM INTO PAGING AREA
PARSE MESSAGE AND BUILD REQUIRED TABLE STRUCTURES
IS FUNCTION TO BE PERFORMED IMMEDIATLY
Y
PAGE IN REQUIRED EXECUTION ROUTINES
N
SAVE DATA PERTAINING TO NEW CONDITIONAL FUNCTIONS
REACTIVATE CONDITIONAL UTILITIES THAT HAVE BEEN DEACTIVATED AND PAGE IN EXECUTION ROUTINES
RETURN TO CONTROL SOFTWARE
INTERRUPT ENTRY FROM MATCHER OR UTILITY EXECUTE INSTRUCTION
DETERMINE UTILITY TASK TO BE PERFORMED
PERFORM UTILITY TASK
RETURN TO INTERRUPTED PROGRAM
IS THIS A REQUEST TO ACTIVATE OR DE-ACTIVATE A UTILITY FUNCTION
Y
ACTIVATE OR DEACTIVATE CONDITIONAL UTILITIES AND PAGE IN EXECUTION ROUTINES
RETURN TO CONTROL SOFTWARE
N
EXECUTE UTILITY FUNCTION

Fig. 6—Basic program flow of utility software.

V. CONCLUSION

The control, administrative, and utility software of the 1A Processor has improved electronic switching system development by centralizing and standardizing functions common to multiple applications. This has simplified initial development and debugging of system software. The effort required for software maintenance has been reduced. It has provided flexibility for the addition of system features.

The software provides a balance between specialized functions required for extremely reliable long-term operation and general-purpose capability. For example, the control software protects the system from disruptions due to interactions between programs that could conflict, but it still permits concurrent execution of independent programs. As an example, within the administrative area, the disk-file system checks a data-identification field before completing disk reads or writes in order to prevent data mutilation or use of erroneous data. Since checks are performed by hardware or by administrative software, user programs bear only a portion of the responsibility to assure system reliability. Finally, the utility system provides a safe means by which the craft force may obtain special outputs or modify the system.

VI. ACKNOWLEDGMENTS

The work described in this article was accomplished through the contributions of many people at Bell Laboratories. The authors would like especially to acknowledge the assistance of E. P. Schan in the preparation of this article.

REFERENCES

1. P. W. Bowman et al, "1A Processor: Maintenance Software," B.S.T.J., this issue, pp. 255–287.
2. C. F. Ault et al., "1A Processor: Memory Systems," B.S.T.J., this issue, pp. 181–205.
3. A. H. Budlong et al., "1A Processor: Control System," B.S.T.J., this issue, pp. 135–179.