1975

A microprocessor-based input/output system for an interactive computer

Wayne Elmer Jones
Iowa State University

INFORMATION TO USERS

This material was produced from a microfilm copy of the original document. While the most advanced technological means to photograph and reproduce this document have been used, the quality is heavily dependent upon the quality of the original submitted.

The following explanation of techniques is provided to help you understand markings or patterns which may appear on this reproduction.

1. The sign or "target" for pages apparently lacking from the document photographed is "Missing Page(s)". If it was possible to obtain the missing page(s) or section, they are spliced into the film along with adjacent pages. This may have necessitated cutting thru an image and duplicating adjacent pages to insure you complete continuity.

2. When an image on the film is obliterated with a large round black mark, it is an indication that the photographer suspected that the copy may have moved during exposure and thus cause a blurred image. You will find a good image of the page in the adjacent frame.

3. When a map, drawing or chart, etc., was part of the material being photographed the photographer followed a definite method in "sectioning" the material. It is customary to begin photoing at the upper left hand corner of a large sheet and to continue photoing from left to right in equal sections with a small overlap. If necessary, sectioning is continued again — beginning below the first row and continuing on until complete.

4. The majority of users indicate that the textual content is of greatest value, however, a somewhat higher quality reproduction could be made from "photographs" if essential to the understanding of the dissertation. Silver prints of "photographs" may be ordered at additional charge by writing the Order Department, giving the catalog number, title, author and specific pages you wish reproduced.

5. PLEASE NOTE: Some pages may have indistinct print. Filmed as received.

Xerox University Microfilms
300 North Zeeb Road
Ann Arbor, Michigan 48106

A microprocessor-based input/output system for an interactive computer

by

Wayne Elmer Jones

A Dissertation Submitted to the
Graduate Faculty in Partial Fulfillment of
The Requirements for the Degree of
DOCTOR OF PHILOSOPHY

Major: Electrical Engineering

Approved:

Signature was redacted for privacy.

In Charge of Major Work

Signature was redacted for privacy.

For the Major Department

Signature was redacted for privacy.

For the Graduate College

Iowa State University
Ames, Iowa

1975

TABLE OF CONTENTS

INTRODUCTION

The earliest computers consisted of a terminal and a processor. When a programmer wished to run a program, he loaded his program and data, ran the program, made any needed changes, and reran the program. The final two steps were repeated until the program operated correctly. This technique was excellent in terms of programmer efficiency because results were available very rapidly.

However, several disadvantages were present in this type of configuration. The computer was idle while the programmer was making changes. Every programmer was required to learn the intricacies of operating the computer. Every programmer had to come to a centralized location since there was usually only one terminal.

To correct some of these inadequacies, batch processing was developed. The programmer now submitted his program to some type of operations staff which ran the program on the machine and returned the program and any output to the programmer. This type of configuration improved the utilization of the computer and removed the burden of physically operating the computer from the programmer. However, elapsed time for a run increased causing programmer efficiency to suffer. The programmer often had forgotten the pattern of thought that led him to make certain changes

before the results of those changes could be returned to him.

In an effort to speed the response time and increase efficiency, a modification of batch processing called multiprogramming was developed. This allowed multiple programs to run on the same machine simultaneously. This process was accomplished by letting each program run for a fixed interval of time and swapping programs at the end of every such time slice. This concept decreased the elapsed time but did not provide a significant improvement over the standard batch processing technique because there were still too many people involved in the intermediate tasks.

The next technique developed was called remote job entry. In this mode the programmer entered his program from a remote terminal and the results were returned to him either from a centralized output device or possibly from an output device at the remote job entry point. This technique did not provide much improvement over the earlier batch processing methods because the elapsed time was not significantly decreased.

As an alternative it was suggested that the user receive a real-time response to his actions at a remote location. This technique is the basis of the interactive time-shared system. The system appears to interact with many users simultaneously by rapidly commutating the system's facilities among these users, each of whom is on-line at a remote

terminal[1,2].

A definition of a real-time response is now required if the system is to be adequately specified. A real-time response implies that the user receives a response to his activity while the reasons for an operation are still fresh in his mind. Delays greater than four seconds can be tolerated if they occur when a major break occurs in the user's thought pattern. Delays greater than two seconds should be avoided if the task requires a high level of concentration. The response speed needs to be proportional to the amount of detail involved in the operation. Operations like keystrokes and cursor movements should appear instantaneous to the user. However, care must be exercised to see that rapid response does not push the user[3,4].

The implementation of an interactive time-shared computer has been the goal of several efforts. The original interactive systems were of a centralized design. The percentage of time the processor was doing useful work for the user was typically very small. The processor spent most of its time doing bookkeeping tasks, program editing functions, time slicing activities and terminal control functions. The block diagram of a central processor configuration is shown in figure 1.

A similar situation arose in systems with a preprocessor type of configuration[5]. The preprocessor was used to

manipulate the data stream and thus simplify the I/O handling requirements for the main processor. The preprocessor typically removed part of the overhead from the main unit, but usually not a significant amount. The block diagram of a preprocessor configuration is shown in figure 2.

Another solution was the SYMBOL system with its distributed architecture and hardware control sequence[6]. This technique allowed the central processor to perform only users' work. However, the implementation of the control functions in hardware made the machine very difficult to service or to adapt to new equipment. A second problem was that the terminal interface requirements could be met only with a specialized and expensive type of terminal. A final problem was the relatively primitive level of communication between the SYMBOL system and the terminal. The block diagram of the SYMBOL system is shown in figure 3.

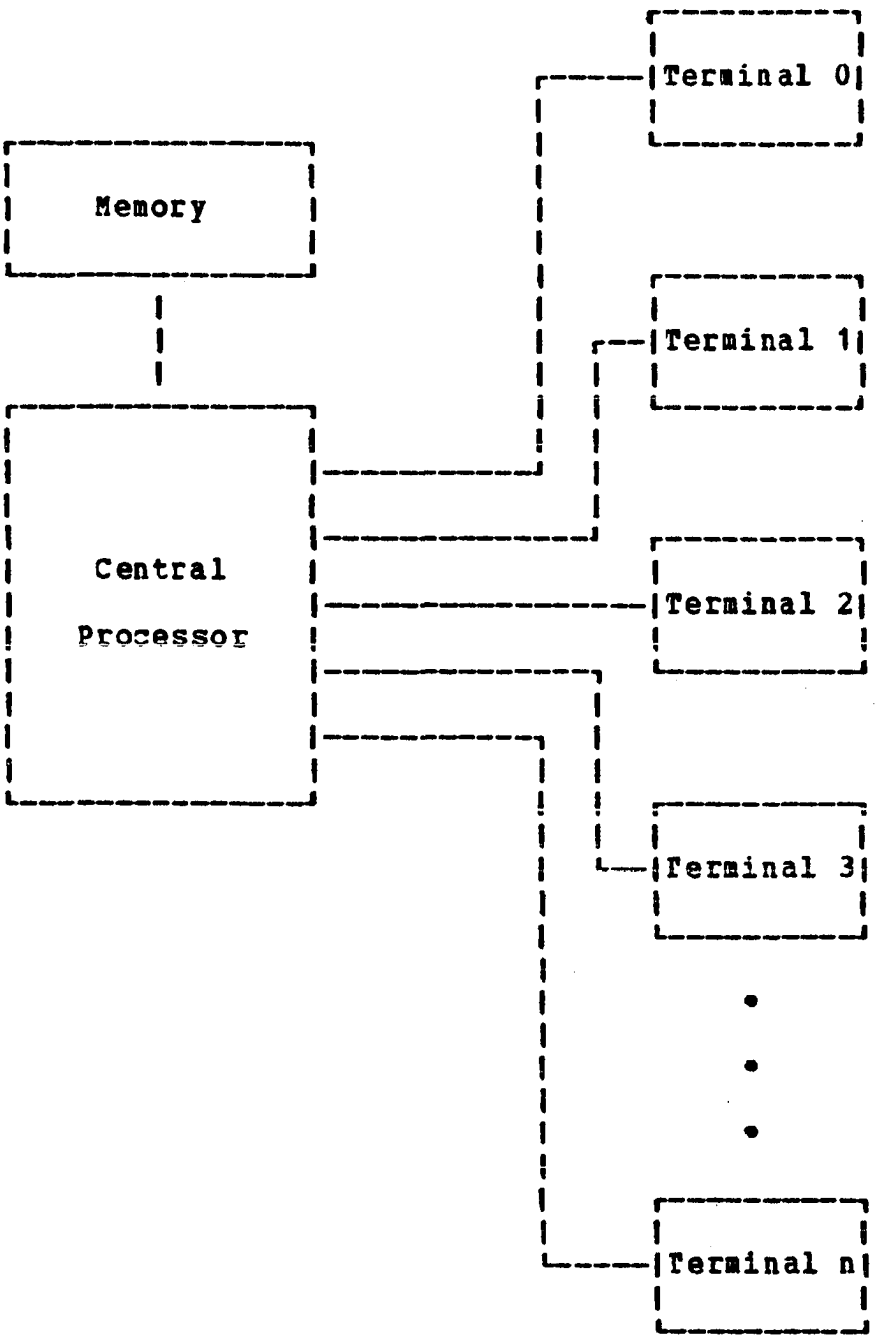

Fig. 1. Central Processor Configuration

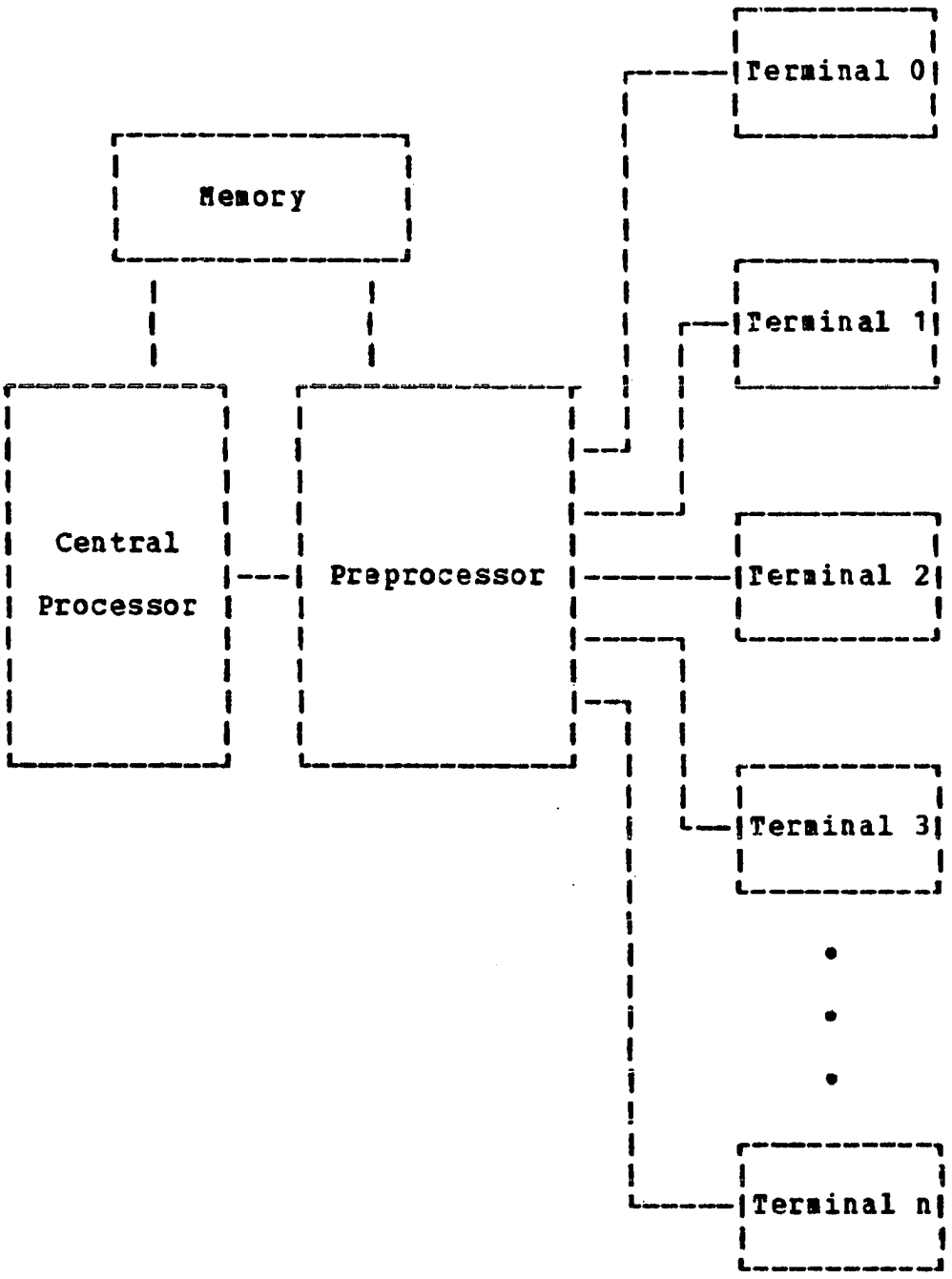

Fig. 2. Preprocessor Configuration

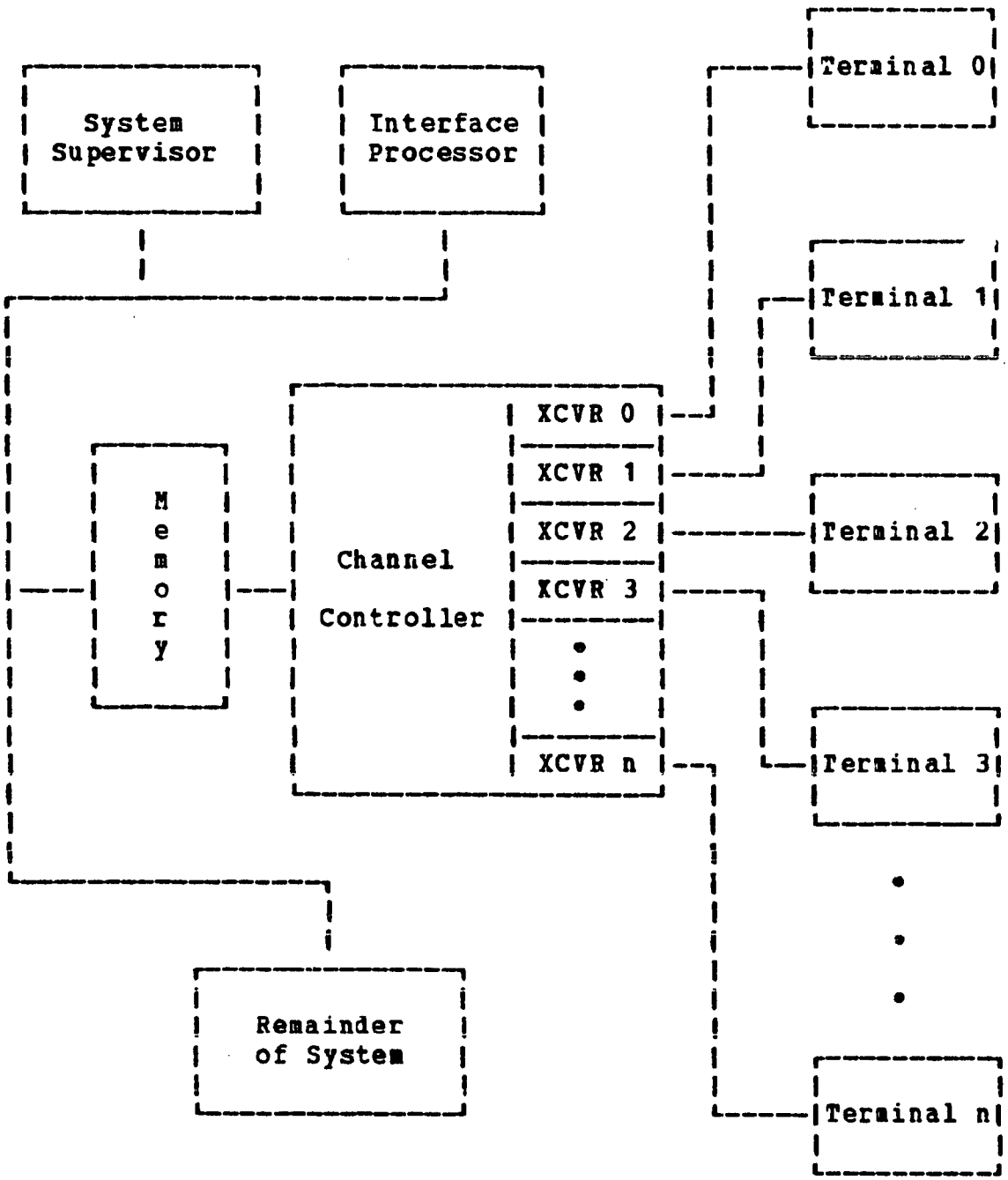

Fig. 3. SYMBOL System Configuration

REVIEW OF LITERATURE

Nearly all previous work involving I/O structures in interactive systems has involved the use of a general purpose computer with software control. The MULTICS system at the Massachusetts Institute of Technology is one example. It is implemented entirely in software on a General Electric 645 computer system[7,8]. One major fault of this system is a restriction on the type of terminals that can be supported, i.e., MULTICS supports only the IBM Model 2741 and the Teletype Model 37.

A second interactive system is THOR[9]. This system is implemented in software and utilizes a video display terminal. However, a special keyboard is required at the terminal to generate the control characters required by the system. During editing operations a portion of the user's text is displayed at the terminal. The user may edit the displayed text or request that a new portion of his text be displayed. The system also supports Teletypes, but the control and editing procedures are less general and more difficult to use than those for the video terminals.

Another interactive system is TALK[10]. It is implemented on a CDC 1700 computer. The system is capable of supporting Teletype devices and card reader/line printer pairs as I/O devices. However, on-line editing facilities are not provided.

An evaluation of commercially available interactive systems was undertaken by Ford Motor Company in 1970[11]. All of the systems utilized a software approach to I/O handling. Each system was evaluated in terms of command language and editing facilities. The user interface was defined as the major problem for nearly all of the evaluated systems.

A major thrust in the area of interactive computing has been the development of text editors. One system, QED, at the University of California in Berkeley is designed to support Teletypes and typewriters in a variety of editing operations[12]. Character editing operations are provided for intraline editing after the selected line has been accessed.

An editing system at Brandeis University is line oriented. No character editing is provided[13]. This system provides support for Teletypes and typewriters.

An editing system implemented by Bourne utilizes a set of codes for intraline editing[14]. ↑ means keep the current character and # means delete the current character. Other editing functions are supported on a line basis.

WYLBUR is the Stanford University Computation Center's editing system[15]. It is a large and powerful software system with only one major restriction. It supports only the IBM Model 2741 and Teletype Model 33 terminals. The system

does not contain provision for many of the functions of a video display, e.g., cursor movement.

The SYMBOL system represents a different approach to an interactive system[16,17,18,19,20,21]. Each processor within the system is utilized for a specific function and nearly all of the control is implemented in hardware. This system provides adequate support only for the SYMBOL terminal. Software can be used to support other I/O devices although not very efficiently.

PROBLEM DEFINITION

All of the previously described interactive systems are limited in the type of terminals that they are capable of supporting. Only one of the systems utilizes a video display and that display unit is special because of keyboard requirements.

In an effort to alleviate this problem, the research described in this disseration involves the determination of the type of I/O structure required to support many types of terminals in an interactive environment. This work includes the specification of command and editing functions for a variety of terminal types and the evaluation of various implementation methods for both control functions and data paths. Serviceability and understandability are also factors in such an evaluation because of the high cost of updating the I/O structure to support new terminals as they are introduced.

A principal emphasis of this work has been the development of an I/O subsystem which optimizes performance with interactive terminals, i.e., where average data rates are low and control functions are complex. Where the maximum data rate is an important factor, e.g., a bulk storage device, the subsystem does not represent the optimum I/O structure.

The availability of microprocessors adds a new dimension to this work. If a microprocessor with supporting hardware can be utilized as a special purpose processor in a large computer system without impairing performance, both serviceability and understandability of the system are improved. With an element, i.e., the microprocessor, common to many of the special purpose processors in the system, knowledge concerning any one of the special processors is an aid in understanding the operation of the others.

SYSTEM EVALUATION CRITERIA

Using information from existing systems, a set of criteria has been developed to evaluate the input-output section of interactive computers. Such a system should:

1. Interface directly to any asynchronous terminal type with a serial channel of up to 9600 baud including those terminals consisting of an interface and multiple devices.
2. Perform code conversion between the terminal character set and the system character set.
3. Provide on-line editing facilities for any type of compatible terminal without unreasonable time delays for the user.
4. Provide terminal control functions as required, e.g., carriage return, clear, line feed.
5. Provide detection and decoding facilities for system commands, e.g., run, initiate, load.
6. Require minimal control from the rest of the computer system.
7. Be interruptible so that it may be used with time slicing or virtual memory systems.
8. Provide a means of "pushing a program down" on a terminal so that a supervisory program

can be run on the same terminal and the "pushed down" program later completed.

9. Be simple to facilitate understanding and serviceability.
10. Include enough flexibility to handle new types of terminals as they are developed.
11. Be capable of being initialized without complex set-up procedures.

The flexibility requirement virtually eliminates the idea of hardware control sequences except in cases where there is little likelihood of change, e.g., buffer manipulation.

The criteria of rapid response, code conversion, and simple initialization preclude the use of pure software since execution time increases when software is used as the driver. The software must also be reloaded at initialization if a system failure caused the control program to be modified. A solution is to use a combination of software and hardware in the form of read-only memory containing control code sequences for the processors.

The serviceability criteria requires that the system be simple. However, the response speed requirement dictates that the system may need to respond to one terminal while it is handling another. One way that both of these criteria can be reasonably satisfied is with a distributed architecture.

With the concept of distributed architecture comes a new problem. Because each processor is constructed to perform a specific function, there may be little resemblance between it and the other processors in the system. Thus, serviceability must still be considered. A viable compromise would be to construct each processor from a standard part, e.g., a microprocessor, and surround it with the supporting hardware necessary to perform the required function. Each processor could then have a common element, but could still be customized to perform a specific task.

SYSTEM FUNCTION DEFINITION

Combining a distributed architecture in the I/O section with the preprocessor configuration of figure 3 results in an I/O subsystem capable of operating between a number of terminal types and a main computer system. Such a configuration is shown in figure 4.

With this organization, the main computer need not possess any knowledge concerning the terminal being utilized. When the main computer requires I/O, the I/O subsystem is informed and the main computer proceeds to other work. Using the terminal identification provided by the main computer with the I/O request, the I/O subsystem accesses the proper terminal, executes the I/O task, and informs the main computer at the time the task is completed or suspended.

This method of interconnection has two significant advantages. The main computer is involved in the I/O task only to the extent that it informs the I/O subsystem when an I/O task is pending and receives the I/O subsystem shutdown information when the task is halted. This allows the main computer to perform other work while I/O is in process on a terminal without time slicing activities. A second advantage is that the I/O subsystem can be utilized with any main computer system capable of supplying the necessary control information. Minor modifications will be required in the I/O subsystem to interface it with different types of main

computer systems. These modifications will be in the I/O subsystem/main computer interface, i.e., the control signals, the memory access signals, and possibly a data conversion if character sets differ.

The SYMBOL computer was utilized as a vehicle for further investigation of the I/O subsystem because SYMBOL's hardware I/O structure is a possible candidate for replacement by the subsystem. Availability of the SYMBOL system and general knowledge of the SYMBOL I/O structure were also important considerations. The detailed descriptions included in the remainder of this paper are defined in terms that are applicable for the I/O subsystem as it would be utilized with the SYMBOL computer. However, the concepts described have a much broader application.

To aid in the implementation of this I/O subsystem, a definition of the functions required of the subsystem is needed. Such a description follows:

1. Terminal control function, e.g., generating carriage returns at the end of a line of text if the terminal doesn't, controlling data rates to prevent data loss, and synchronizing the terminal's asynchronous information to allow the system to manipulate it.
2. Code conversion, i.e., the transformation between the symbolic code used by the terminal

and the symbolic code used by the system if they are not identical.

3. Control command detection, i.e., system commands and editing commands must be detected in the data stream.
4. Buffer manipulation, i.e., some type of buffer mechanism must be provided to smooth the data flow to the terminal and yet not present an excessive load to the system.
5. Data transfer, i.e., information must be moved between the terminals and their respective buffers. Information must also be moved between the buffers and main memory.
6. I/O subsystem control, i.e., the total data path must be monitored to assure that transfers are accomplished as efficiently and rapidly as possible.
7. Control command interpretation, i.e., system commands and editing commands must be interpreted to determine the proper response.
8. Separate control command channel, i.e., a discrete channel for control commands must be provided to allow system commands to be received during output.
9. Editing command execution, i.e., the receipt

of an editing command may necessitate an immediate update of the terminal display.

10. System command transmission, i.e., the rest of the system must be informed of the receipt of a system command.
11. Character echoing, i.e., the system must transmit any characters that are to be displayed back to the terminal because the system is interposed between the entry device and the display at the terminal in the full duplex mode of operation. Half duplex does not allow enough flexibility for an interactive system.
12. Pushdown operation, i.e., a supervisory routine can be run on a terminal and then return control to a user program in progress on the same terminal.

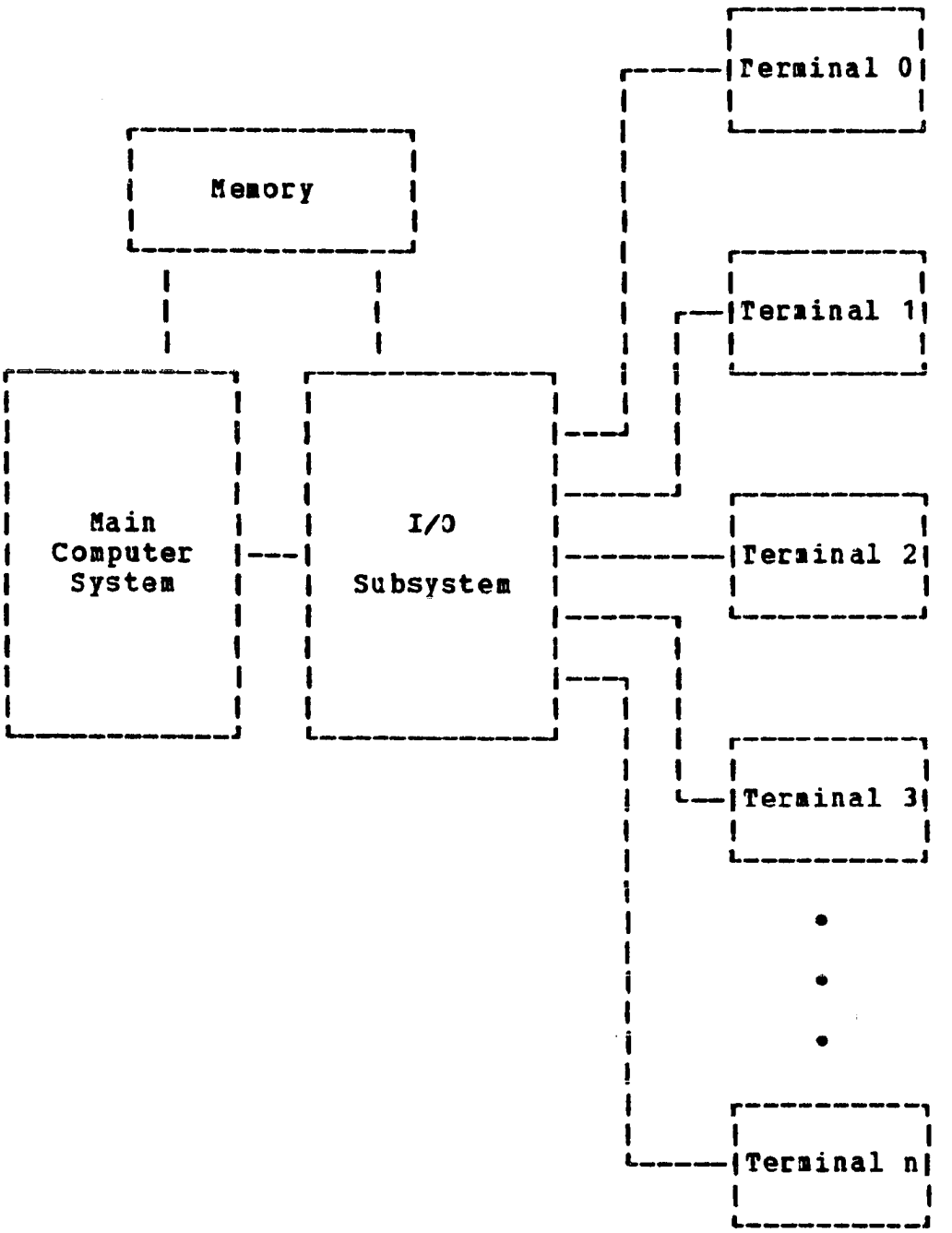

Fig. 4. I/O Subsystem Configuration

PROCESSOR FUNCTION ASSIGNMENTS

Accepting a distributed architecture as optimum leaves the problem of deciding where each of the previously defined functions should be performed.

The terminal control functions must be performed on a dedicated basis if they are to be adequate. If an effort is made to perform such tasks on a time-shared basis the complexity rises markedly. For example, it is necessary to prevent transmission of a new character from the terminal before the previous one has been handled. However, if such control is time-shared, high speeds and careful design are required to ascertain that every terminal is checked for such an overwrite condition. An individual terminal control for each terminal is more straightforward and allows control to be tailored for the terminal desired. Because a variety of terminal types is to be supported, the first application of a microprogrammed microprocessor becomes apparent. Standard hardware and a variety of control sequences for a variety of terminals effectively solves the problem. The name Terminal Controller has been chosen to describe this element in the architecture. One will be required for each terminal connected to the system.

The code transformation should take place in the Terminal Controller. Since needless complexity can be avoided if the rest of the system handles only one character

set, the code transformation must be done as close to the terminal as possible, i.e., in the Terminal Controller. An added advantage is that the Terminal Controller already requires an identification of the terminal type to determine the control requirements and this same identification can be used to control the code transformation.

Handling of control commands is the next function to be considered. These commands require detection at an early stage in the incoming data stream so that they may be routed separately. Because every character undergoes a code transformation, the control commands can be detected at the same time by deriving additional information from the code transformation. This information can then be used to control the routing of the control commands.

A final task that can be performed in the Terminal Controller is the job of echoing characters back to the terminal. If the character is not a part of a control command, it will need to be echoed and the most rapid response is obtained by doing the job in the Terminal Controller. Control commands may or may not need to be echoed and must be handled in a more complex manner.

The buffer concept and the requirement that the control command channel be separate dictate that there must be three buffer areas for each terminal. Two, A and B, will be utilized as data buffers. A third, C, will be utilized as a

holding area for control commands. This can be accomplished by locking out any new control commands until the previous one has been interpreted. This is not an unreasonable procedure because the user will need to know the result of the previous control command before another is entered.

The data transfers between the Terminal Controllers and their respective buffers will be handled by the Channel Controller. The Channel Controller will scan the Terminal Controllers and transfer characters between the Terminal Controllers and the appropriate buffers as required to keep the terminals and the system operating. It will utilize the information provided by the Terminal Controllers concerning control commands to initiate transfers to the C buffer.

To most effectively utilize the main memory, the data transfers between the buffers and the main memory will be performed on a burst basis to lessen the loading on the main memory. The processor which does this is called the Interface Processor.

The logical location for editing command execution is the Interface Processor. Because it manages transfers between the buffers and virtual memory, it has access to a copy of the information available to the user. This information is needed during editing tasks so that the information presented at the terminal may be updated correctly.

The interpretation of control commands is a simple task if each command is transmitted from the terminal as a unique code. However, this is clumsy because the keystrokes required will typically have little in common with the operation requested. If multiple keystrokes are permitted to construct a control command, another problem arises. Experienced users may be capable of using an abbreviated set of keystrokes while a new user may want to use very explicit commands to accomplish the same function. This implies that a variety of commands, e.g., "L‡, "LOAD‡, and "Control" X, may need to be converted to the same code for use within the system[22]. ‡ represents the end of a message.

To allow enough flexibility and still not cause unnecessary duplication of hardware, such a conversion can be accomplished after the control command is placed in the C buffer. Thus, only a single processor, the Control Interpreter, is needed to handle all the terminals in the system. The only restriction is that the same sequence can not have different meanings for different terminals. This restriction applies only after the code transformation is completed.

The I/O Supervisor is required to control the operations of the other processors in the system. Its tasks include:

1. Communicating with all the processors in the I/O subsystem to provide operating instructions.

2. Ascertaining when a processor in the I/O subsystem is done with its assigned task so that buffers and terminals may be reassigned to provide more work for that processor if any is available.
3. Communicating with the rest of the system to see what I/O tasks are ready to be performed.
4. Informing the rest of the system of system commands that have been received and interpreted.

The I/O Supervisor also has a responsibility to see that no information is lost when a program is made inactive on a terminal so that a supervisory routine may be executed. It must transfer all pertinent information to a safe location in the main memory. Then, upon request of the supervisory routine, the I/O Supervisor must restore the terminal to its original status by retrieving the data from the main memory and continuing the original operation.

The code transformation problem can be handled by an array of read-only memories, each programmed for a specfic type of device. The memory will be time-shared among the Terminal Controllers and thus will be able to scan Controllers at a rate equivalent to the cycle time of the memory array.

The input-output memory will be of the read/write variety and will contain the three buffers and a set of control words for each terminal. These control words will contain the information needed for the system to perform I/O tasks on the terminal.

A read-only memory containing control sequences will be attached to each processor. Provision for new terminals as they are introduced will require expansion of the control memory to provide additional control sequences. The Interface Processor and the Control Interpreter will require the most control memory modification if additional terminal types are supported. The Terminal Controller will require minor modifications and a new code transformation table will need to be added if the terminal's character set is different.

The block diagram of the I/O subsystem is shown in figure 5.

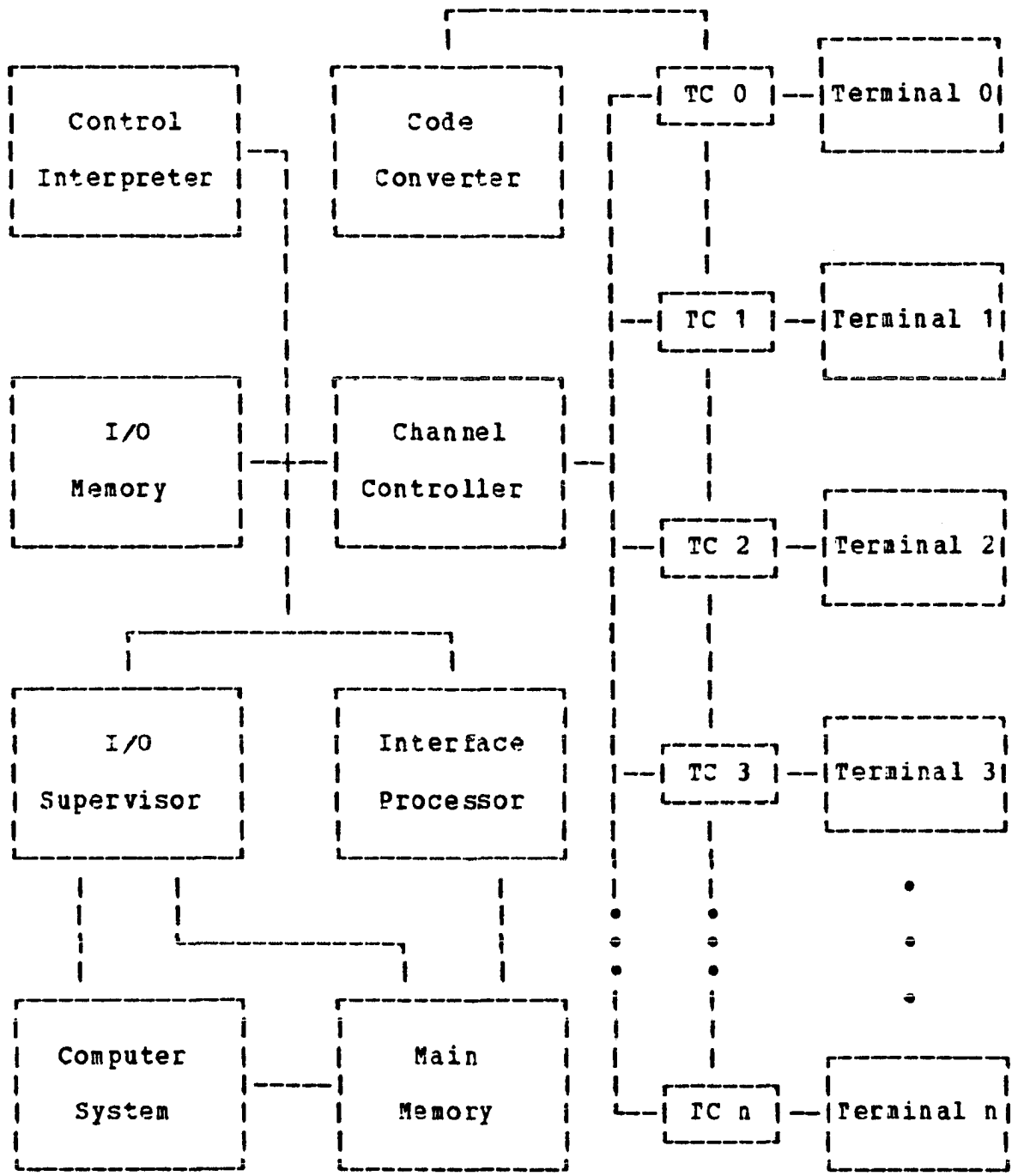

Fig. 5. I/O Subsystem Block Diagram

I/O SUBSYSTEM COMMUNICATION REQUIREMENTS

To provide a communication path between the various processors, a twenty-seven line control bus will be required. This bus is allocated as follows:

1. Five lines for the terminal number. These will be used to specify a terminal number for the various start and stop signals transmitted by the I/O Supervisor.
2. One line for the Terminal Controller start signal (TCSTRT) from the I/O Supervisor. This line in combination with the terminal number lines will start the selected Terminal Controller.
3. One line for the Terminal Controller quit signal (TCQUIT) from the I/O Supervisor. This line in combination with the terminal number lines will stop the selected Terminal Controller.
4. One line for the Interface Processor start signal (IPSTRT) from the I/O Supervisor. This line in combination with the terminal number lines will start the Interface Processor on the selected terminal's task.
5. One line for the Interface Processor quit

signal (IPQUIT) from the I/O Supervisor. This line will cause the Interface Processor to shut down, i.e., to suspend work on the current task.

6. Three lines for the Interface Processor to transmit shutdown information to the I/O Supervisor.
7. One line for the Channel Controller start signal (CCGO) from the I/O Supervisor. This line will cause the Channel Controller to begin scanning the Terminal Controllers.
8. One line for the Channel Controller quit signal (CCQUIT) from the I/O Supervisor. This line will cause the Channel Controller to stop scanning the Terminal Controllers.
9. Eight lines for the Channel Controller to transmit shutdown information to the I/O Supervisor. Three lines will be used to define the type of shutdown and five lines will be used to transmit the terminal number. A CCGO signal will be required from the I/O Supervisor after a shutdown before the Channel Controller will resume scanning the Terminal Controllers.
10. One line for the Control Interpreter start

signal (CISTRT) from the I/O Supervisor. This line in combination with the terminal number lines will start the Control Interpreter on the selected terminal's task.

11. One line for the Control Interpreter quit signal (CIQUIT) from the I/O Supervisor. This line will cause the Control Interpreter to shut down.
12. Three lines for the Control Interpreter to transmit shutdown information to the I/O Supervisor.

Because all of these lines are driven by only one output, the bus will not require wired-OR connections. An advantage to this bus configuration is that most of the processors need to monitor only selected lines of the bus and need not be concerned with the other lines. The only decoding required will be done by the I/O Supervisor to process shutdown information.

The shutdown information for the Channel Controller will be transmitted to the I/O Supervisor as one of the following codes. A terminal number, which is transmitted with the shutdown information, provides the terminal identification to the I/O Supervisor.

CCBFE This shutdown signals the I/O Supervisor that

the Channel Controller has stopped scanning because it has completed the processing required on the specified terminal and is ready for more work on that terminal. This shutdown also signals the I/O Supervisor that a buffer is empty during an editing task.

CCEOR This shutdown signals the I/O Supervisor that the Channel Controller has stopped scanning because it has completed the processing required on the specified terminal. Execution was stopped because an "end of record" character was encountered during processing. This shutdown also signals the I/O Supervisor that an editing task has been completed and the system is ready for more input from the specified terminal.

CCCC This shutdown signals the I/O Supervisor that the Channel Controller has encountered the end of a control command and has stored the command in the C buffer of the specified terminal.

CCSTAT This shutdown signals the I/O Supervisor that the Channel Controller has stopped scanning because it has completed a status update on the specified terminal.

CCERR This shutdown signals the I/O Supervisor that the Channel Controller has stopped scanning because it has detected an error on the specified terminal. An error code and the location of the error have been stored in the I/O Memory to facilitate diagnosis.

CCTCQT This shutdown signals the I/O Supervisor that the Channel Controller has stopped scanning because it has detected that the Terminal Controller on the specified terminal has shut down in response to a TCQUIT signal.

CCQT This shutdown signals the I/O Supervisor that the Channel Controller has stopped scanning in response to a quit signal from the I/O Supervisor. The terminal number is ignored when this shutdown is received by the I/O Supervisor.

The shutdown information for the Interface Processor will be transmitted to the I/O Supervisor as one of the following codes:

IPBFE This shutdown signals the I/O Supervisor that the Interface Processor has completed the processing required on its assigned job and is ready for more work. This shutdown also

signals the I/O Supervisor that a buffer is full during an editing task.

IPEOR This shutdown signals the I/O Supervisor that the Interface Processor has completed the processing required on its assigned job and is ready for more work. Execution was stopped because an "end of record" character was encountered during processing. This shutdown also signals the I/O Supervisor that an editing task has been completed and the system is ready for more input from the terminal as soon as a CCEOR shutdown is received from the Channel Controller for the terminal of interest.

IPPO This shutdown signals the I/O Supervisor that the Interface Processor is shutting down because it has encountered a page fault in the main memory and may be reassigned until the needed page is available.

IPERR This shutdown signals the I/O Supervisor that the Interface Processor is shutting down because it has detected an error condition. An error code and the location of the error have been stored in the I/O Memory to facilitate diagnosis.

IPQT This shutdown signals the I/O Supervisor that the Interface Processor is shutting down in response to a quit signal from the I/O Supervisor.

The shutdown information for the Control Interpreter will be transmitted to the I/O Supervisor as one of the following codes:

CIED This shutdown signals the I/O Supervisor that the Control Interpreter has completed the processing required on its assigned job and has detected an editing command.

CISC This shutdown signals the I/O Supervisor that the Control Interpreter has completed the processing required on its assigned job and has detected a system command.

CIERR This shutdown signals the I/O Supervisor that the Control Interpreter is shutting down because it has detected an error condition. An error code and the location of the error have been stored in the I/O Memory to facilitate diagnosis.

CIQT This shutdown signals the I/O Supervisor that the Control Interpreter is shutting down in response to a quit signal from the I/O Supervisor.

The communication path between the I/O Supervisor and the rest of the computer system requires eight lines for information and five lines for the terminal number. The allocation is as follows:

1. One line for the I/O subsystem start signal (IOSTRT) from the computer system. This line in combination with the terminal number lines will instruct the I/O Supervisor to initiate processing on the selected terminal's task.
2. One line for the I/O subsystem quit signal (IOQUIT) from the computer system. This line in combination with the terminal number lines will instruct the I/O Supervisor to halt processing on the selected terminal's task.
3. One line for the I/O subsystem save signal (IOSAVE) from the computer system. This line in combination with the terminal number lines will instruct the I/O Supervisor to halt processing on the selected terminal's task and store status information for the terminal in the main memory to allow restoration of the terminal's status at a later time.
4. One line for the I/O subsystem restore signal (IOREST) from the computer system. This line in combination with the terminal number lines

will instruct I/O Supervisor to retrieve the status information for the selected terminal from the main memory and resume processing on the selected terminal's task.

5. Four lines for the I/O Supervisor to transmit shutdown information to the computer system. These lines in combination with the terminal number lines will specify the type of shutdown generated for a particular terminal.
6. Five lines for the terminal number. These lines will be used to specify a terminal number for the communication path.

All of the above lines must be conditioned to meet the logic level requirements of the computer system. Provision must be made in the I/O Supervisor to generate any status signals that may be required by the computer system during these information exchanges.

The shutdown information for the I/O subsystem will be transmitted to the computer system as one of the following codes. A terminal number is transmitted with the shutdown information to provide terminal identification to the computer system.

IONOR The I/O task assigned by the computer system on the specified terminal has been completed satisfactorily.

IOERR The I/O task assigned by the computer system on the specified terminal has been terminated because of an error. An error code and the location of the error have been stored in the main memory to facilitate diagnosis.

IOPO The I/O task assigned by the computer system on the specified terminal has generated a page fault in the main memory. Processing will be suspended as soon as the buffer capacity is exceeded.

IOQT The I/O task assigned by the computer system on the specified terminal has been terminated in response to a quit signal from the computer system.

IOSC A system command has been detected by the I/O subsystem on the specified terminal and has been stored in the main memory pending further action by the computer system.

CLASSES OF TERMINAL TYPES SUPPORTED

The proposed I/O subsystem can support four broad classes of terminal types:

Class 1

Class 1 is composed of devices such as card readers and line printers. Although these types of devices do not lend themselves to interactive operation, they are still important to the user who desires to manipulate large data blocks on an interactive system without long I/O periods as would be required for keyboard entry or typewriter output. Status display is not provided for Class 1 terminals because there is no reasonable place for the presentation of such information. The system will support the use of simple editing and system commands. These will need to be provided as an input to the system where they will be interpreted as control commands and routed accordingly. The system commands which will be supported are:

•INIT†	Initiate activity on the terminal.
•CLEAR†	Clear the terminal's work area.
•LOAD†	Load a program into the terminal's work area.
•RUN†	Run the program in the terminal's work area.
•CANOUT†	Cancel the current output task on the terminal.
•PAUSE†	Halt processing on the terminal's work.
•CONTIN†	Continue processing on the terminal's work.
•Fm†	Initiate a special function call for terminal.
•TERM†	Terminate activity on the terminal.

where 'm' represents any integer between 0 and 15. These commands are all expressed in the long form. A short form

consisting of a one or two letter sequence will also be supported. The third form that will be supported is a single character representation, e.g., a multipunch will symbolize a LOAD command.

The system will support editing for Class 1 terminals only on a line basis. Each line is considered an autonomous unit. A line may be a fixed number of characters or may be delimited by a carriage return. This decision is a function of a flag set by two of the editing commands as defined below. The line editing commands which will be supported are:

▪CR‡	A line will be delimited by a carriage return.
▪L=n‡	A line will be considered as 'n' characters.
▪+n‡	Move the pointer forward 'n' lines.
▪-n‡	Move the pointer backward 'n' lines.
▪DLn‡	Delete 'n' lines following the pointer.
▪Dn‡	Display 'n' lines following the pointer.
▪SFxxxx‡	Search forward for 'xxxx'.
▪SBxxxx‡	Search backward for 'xxxx'.
▪OV‡	Enter overwrite mode.
▪IN‡	Enter insert mode.

where 'x' represents any character other than '▪' or '‡' and 'n' represents any integer between 0 and 999. The alphanumeric field for search operations may be up to eight characters in length. The alphanumeric field following ▪IN‡ will be padded with blanks on the right to correspond to a fixed line length if a line is defined as a fixed number of characters. If ▪IN‡ is not used, an overwrite will be performed. The 'insert' mode is exited whenever an editing command is encountered in the data stream from the terminal.

If 'n' is 0 the system will interpret 'n' as "to the beginning" or "to the end." If 'n' is omitted the system will assign 'n' a value of 1.

Class 2

Class 2 is composed of devices such as typewriters and Teletypes. Although more convenient than a Class 1 device for interactive activity, the Class 2 device's slower operating speed is a hindrance to optimum interactive performance.

In addition to the system commands for Class 1 devices, Class 2 terminals provide the user with the terminal's status upon request. Entering •STAT‡ at the terminal will cause the system to respond with the terminal's current status in the system. An example of a single character representation for Class 2 devices is "Control" X for a LOAD command.

The system will support editing commands for Class 2 terminals on a line or character basis. A line will be defined as for Class 1 devices. The line editing commands will be identical to those for Class 1 terminals. To allow the use of character editing, the system will respond to any line editing request by displaying the line selected by the current pointer position. The pointer will be positioned at the first character of the line.

To initiate the character editing mode the user will enter •CH‡. The character editing commands which will be

supported are:

▪+n‡	Move pointer forward 'n' characters.
▪-n‡	Move pointer backward 'n' characters.
▪DLn‡	Delete 'n' characters following the pointer.
▪Dn‡	Display 'n' characters following the pointer.
▪SFxxxx‡	Search forward for 'xxxx'.
▪SBxxxx‡	Search backward for 'xxxx'.
▪OV‡	Enter overwrite mode.
▪IN‡	Enter insert mode.

where 'x' represents any character other than '▪' or '‡' and 'n' represents any integer between 0 and 999. The alphanumeric field for search operations may be up to eight characters in length. If the alphanumeric field following ▪IN‡ causes the line to exceed the maximum length allowable, the system will generate a new line and pad it with blanks as necessary. If ▪IN‡ is not used, an overwrite will be performed. The 'insert' mode is exited whenever an editing command is encountered in the data stream from the terminal. If 'n' is 0 the system will interpret 'n' as "to the beginning" or "to the end." If 'n' is omitted the system will assign 'n' a value of 1.

To return to the line editing mode the user will enter ▪LN‡. The system will respond by displaying the line selected by the current pointer position. The pointer will be positioned at the first character of the line.

Class 3

Class 3 is composed of devices with a video display. These devices comprise the most useful type for interactive

operation because of their high speeds and convenient display. Most video terminals have some type of cursor to inform the user where the next character will appear on the screen.

Class 3 terminals utilize the same system command set as Class 2 devices. However, the extra keys on many Class 3 devices provide a concise method of generating system commands from one keystroke[23,24,25].

The system will support editing commands for Class 3 terminals in the same manner as utilized for Class 2 devices. The following single keystrokes will also be supported:

←	Move cursor left one character.
→	Move cursor right one character.
↑	Move cursor up one line.
↓	Move cursor down one line.
'HOME'	Move cursor to upper left corner of display.
'DELLINE'	Delete line selected by cursor.
'DELCHAR'	Delete character selected by cursor.
'INSLINE'	Insert line at cursor location.
'INSCHAR'	Insert character at cursor location.

If the user moves the cursor to a location containing information and does not select 'INSCHAR' or 'INSLINE', an overwrite will be performed.

Class 4

Class 4 is composed of a terminal interface and a number of I/O devices. If the I/O request specifies that the default device be used, the terminal interface must provide a default device identification when requested by the Terminal Controller to permit proper handling of data by the I/O

subsystem. If the I/O request contains a specific device number, the I/O subsystem can be configured without interrogation of the terminal interface. In this way any of the preceding terminal classes can be used in combination with a terminal interface to construct a more complex terminal.

DETAILED PROCESSOR DESCRIPTIONS

Terminal Controller

Each Terminal Controller will consist, in part, of a microprocessor driven by a read-only control memory. To supplement the microprocessor, five high speed registers will be required with the following characteristics:

IIREG A twelve bit register to store data produced by the code conversion during INPUT.

IOREG A twelve bit register to store data ready for the Channel Controller during INPUT.

OIREG An eight bit register to store data transmitted from the Channel Controller during OUTPUT.

OOREG A twelve bit register to store data produced by the code conversion during OUTPUT.

TMNID An eight bit register containing the terminal type identification.

The microprocessor utilized in the Terminal Controller is required to process one character every millisecond if the terminal is operating at 9600 baud. A four bit microprocessor with a 2 microsecond instruction cycle time is adequate for this mode of operation. Available microprocessors having this type of specification include the Intel 4040. Because the data registers are twelve bits in

many cases, the width of the data path through the microprocessor is not critical.

The data registers are necessary to buffer data since different portions of the subsystem operate at different rates. The low order eight bits in the data registers will be used to represent the data with the remainder reserved as indicators for escape symbols, carriage returns, and control commands.

A counter will be maintained in the Terminal Controller to determine if a carriage return should be inserted since the end of the line has been reached at the terminal. Since speed is not essential for this operation, the counter can be kept in one of the microprocessor's registers. If the counter is equal to zero, the Terminal Controller will generate a carriage return for the subsystem and for the terminal in addition to its normal work.

A number of flags will be required to indicate that the registers are valid, that characters should be echoed, that a '.' has been received from the terminal, that the terminal is in an I/O mode, that control commands can be accepted, and that the terminal is active. These may be either within the microprocessor or external depending on the capabilities of the microprocessor utilized.

Another element in the Terminal Controller is a buffer register for storing data as it is generated by the terminal.

This is necessary because under certain conditions some terminals continue to produce data after they have been instructed to stop sending information. This data would be lost if buffering was not provided. Each Terminal Controller will also contain the hardware necessary for asynchronous serial I/O, i.e., a Universal Asynchronous Receiver Transmitter or UART. A frequency divider will be required to generate a variety of baud rates. The proper baud rate will be determined by the contents of the TMNID register.

The Terminal Controller will be connected to the control bus to monitor the TCSTRT, TCQUIT, and terminal number lines. In the event that a TCQUIT is received from the I/O Supervisor, the Terminal Controller will instruct the terminal to stop sending information. After waiting sufficient time to insure that the terminal has stopped transmitting data, the Terminal Controller will transmit the information needed to restart itself to the Channel Controller with instructions to store the information in the Terminal Controller's control word. When this task is completed, a CCTCQT will be sent to the I/O Supervisor by the Channel Controller. When a TCSTRT signal is received by the Terminal Controller, the Channel Controller will retrieve the information stored in the Terminal Controller's control word and transmit it to the Terminal Controller so that the job may be started or resumed as the case may be.

The organization of the Terminal Controller is shown in figure 9.

Channel Controller

The Channel Controller will consist, in part, of a microprocessor driven by a read-only control memory. To supplement the microprocessor, two high speed registers will be required with the following characteristics:

CCWRD A sixty-four bit register to store the Channel Controller's control word for the selected terminal.

CCDATA An eight bit register to hold the data being transferred between the Channel Controller and the I/O memory.

The microprocessor utilized in the Channel Controller will have a minimum of 35 microseconds to process each character. An eight bit microprocessor with a 1 microsecond instruction cycle time can provide this data rate if care is exercised in the control program. Since it is very unlikely that all 32 terminals will be operating at 9600 baud simultaneously for extended periods of time, a microprocessor with a 2 microsecond instruction cycle time is adequate for this application most of the time. Available microprocessors having this type of specification include the Intel 8080. A faster microprocessor would allow more latitude in control

program construction and permit the Channel Controller to handle the worst case data rate without slowing the I/O subsystem.

The control word for the Channel Controller will contain the location of the next character position in the I/O memory and information describing the selected terminal's task. After each data transfer the control word is updated and written back into the I/O memory.

The Channel Controller will communicate with the Terminal Controllers over the Channel Control bus. This bus will consist of twelve bidirectional data lines, five address lines for selecting the appropriate Terminal Controller, a line activated by the selected Terminal Controller to indicate that a transfer can be accomplished, and a control line from the Channel Controller to enable the selected Terminal Controller's registers.

The selection of the Terminal Controller will be handled by a scanner implemented in hardware. Each Terminal Controller will be interrogated twice during each scan cycle, once for the INPUT mode and once for the OUTPUT mode.

Whenever the Channel Controller transmits a shutdown code to the I/O Supervisor, the I/O Supervisor must respond with a CCGO signal before the Channel Controller will resume scanning the Terminal Controllers. This method is needed to prevent a second Channel Controller shutdown while the first

is being processed by the I/O Supervisor.

The organization of the Channel Controller is shown in figure 10.

Control Interpreter

The Control Interpreter will consist, in part, of a microprocessor driven by a read-only control memory. To supplement the microprocessor, two high speed registers will be required with the following characteristics:

CIWRD A sixty-four bit register to store the Control Interpreter's control word for the selected terminal. The terminal type identification will be stored in this control word.

CIDATA A sixty-four bit register to store the portion of the C buffer being processed.

The microprocessor utilized in the Control Interpreter has no data rate problems. However, since the subsystem should respond to a control request rapidly, an eight bit unit with a 2 microsecond instruction cycle time is required. Available microprocessors having this type of specification include the Intel 8080. A four bit microprocessor is an alternative, but the search time for a table entry would increase.

The Control Interpreter will also require a read-only memory containing a table of the supported control commands for each type of terminal. There will be sufficient space in the Control Interpreter control word to store the interpreted version of the control commands for use by the I/O Supervisor.

The organization of the Control Interpreter is shown in figure 11.

Interface Processor

The Interface Processor will consist, in part, of a microprocessor driven by a read-only control memory. To supplement the microprocessor, nine high speed registers will be required with the following characteristics:

IPDATA A sixty-four bit register for temporary storage of data being transferred to or from the buffers in the I/O memory.

IPDATB A sixty-four bit register for temporary storage of data being transferred to or from the main memory.

BUFADD A sixteen bit register containing the current buffer location in the I/O memory.

THNID An eight bit register containing the terminal type identification.

DPSTRT A thirty-two bit register containing the

start location for the terminal display.

DPEND A thirty-two bit register containing the end location for the terminal display.

LNSTRT A thirty-two bit register containing the start location for the current line within the display.

LNEND A thirty-two bit register containing the end location for the current line within the display.

PNTADD A thirty-two bit register containing the current pointer location.

The microprocessor utilized in the Interface Processor will have a minimum of 2 milliseconds to handle a 64 character buffer. Providing this data rate while performing the necessary processing requires an instruction cycle time of 100 nanoseconds or less since a significant portion of the 2 milliseconds may be consumed by a page fault in the main memory. Available microprocessors having this type of specification include the Intel 3000 series.

The standard I/O functions involve transferring data between IPDATA and IPDATB while updating the address registers as required. The editing functions require more complex processing. In an effort to standardize this processing, a set of editing functions has been defined for the Interface Processor:

1. Move the pointer forward 'n' characters or to the end of the current line if fewer than 'n' characters follow the current pointer position in the line.
2. Move the pointer forward to the beginning of the following line and then forward for 'n' lines or to the end of the string if fewer than 'n' lines follow the current line.
3. Move the pointer forward to the beginning of the following line and then forward for 'n' characters or to the end of the line if there are fewer than 'n' characters in the line.
4. Search forward for a specified pattern or to the end of the current line if the pattern does not occur following the current pointer position in the line.
5. Search forward for a specified pattern or to the end of the string if the pattern does not occur following the current pointer position in the string.
6. Move the pointer backward 'n' characters or to the beginning of the current line if fewer than 'n' characters precede the current pointer position in the line.
7. Move the pointer backward to the beginning of

the current line and then backward for 'n' lines or to the beginning of the string if fewer than 'n' lines precede the current line.

8. Move the pointer backward to the beginning of the preceding line and then forward for 'n' characters or to the end of the line if there are fewer than 'n' characters in the line.
9. Search backward for a specified pattern or to the beginning of the current line if the pattern does not occur preceding the current pointer position in the line.
10. Search backward for a specified pattern or to the beginning of the string if the pattern does not occur preceding the current pointer position in the string.
11. Insert the incoming data into the existing string at the current pointer position.
12. Overwrite the existing string with the incoming data beginning at the current pointer position.
13. Delete 'n' characters following the current pointer position or to the end of the line if fewer than 'n' characters follow the current pointer position in the line.
14. Delete the current line and 'n' following

lines or the remainder of the string if fewer than 'n' lines follow the current line.

15. Display 'n' characters following the current pointer position or to the end of the line if fewer than 'n' characters follow the current pointer position.
16. Display the current line and 'n' following lines or the remainder of the string if fewer than 'n' lines follow the current line.

Utilization of these standard functions decreases the size of the control memory because each terminal type's control program can now be written as a series of linkages to these standard functions.

The organization of the Interface Processor is shown in figure 12.

I/O Supervisor

The I/O Supervisor will consist, in part, of a microprocessor driven by a read-only control memory. To supplement the microprocessor, five high speed registers will be required with the following characteristics:

IODATA A sixty-four bit register for temporary storage of control words as they are being updated and for data transfers during 'save' and 'restore' operations.

IOADD A sixteen bit register containing the location being processed in the I/O Memory during 'save' and 'restore' operations.

SAVADD A thirty-two bit register containing the location in the main memory for 'save' and 'restore' operations.

CNBREG A fourteen bit register for temporary storage of shutdown information from the other processors in the I/O subsystem.

COMREG A thirteen bit register for use in communicating with the main computer system.

The microprocessor utilized in the I/O Supervisor must respond to processor shutdowns with significant processing tasks of its own. An instruction cycle time of 2 microseconds is required to perform the work in a reasonable length of time. Long delays in the I/O Supervisor decrease the time available to the Channel Controller and the Interface Processor for the performance of their tasks. Available microprocessors having this type of specification include the Intel 8080.

The control sequence required for the I/O Supervisor can best be expressed by a flow table for each of the general I/O processes. When the I/O Supervisor receives an I/O request, the I/O state for the specified terminal is set to 0 and processing is begun.

If the I/O request is for INPUT, the flow table in figure 6 is utilized. The specified Terminal Controller is started and the Channel Controller begins to fill the A buffer. CCF is used in the flow table to indicate that the Channel Controller is filling the buffer. Because the B buffer is empty, E is entered in the flow table. When the A buffer is full, the Channel Controller stops scanning and transmits a CCBFE shutdown to the I/O Supervisor. The I/O Supervisor sets the I/O state to 1, adds the terminal to the Interface Processor queue to transfer the A buffer to the main memory, assigns the B buffer to the Channel Controller for the specified terminal, and transmits a CCGO to the

I/O STATE	BUFFER A	BUFFER B	SHUTDOWNS CCBFE	CCEOR	IPBFE	IPEOR
0	CCF	E	1	6	x	x
1	IPU	CCF	2	7	3	x
2	IPU	F	x	x	4	x
3	E	CCF	4	8	x	x
4	CCF	IPU	5	9	0	x
5	F	IPU	x	x	6	x
6	IPU	E	x	x	x	QUIT
7	IPU	F	x	x	8	x
8	E	IPU	x	x	x	QUIT
9	F	IPU	x	x	6	x

Fig. 6. INPUT Flow Table

Channel Controller. This is denoted in the flow table by IPU for the A buffer and CCF for the B buffer. F in a buffer column indicates that the specified buffer is full. An 'x' in the shutdown portion of the table indicates that the specfied shutdown can not occur.

If the I/O request is for OUTPUT, the flow table in figure 7 is utilized. Two new symbols appear in this flow table: IPF meaning the Interface Processor is filling the buffer; and CCU meaning the Channel Controller is transferring data out of the buffer.

I/O STATE	BUFFER A	BUFFER B	SHUTDOWNS IPBFE	IPEOR	CCBFE	CCEOR
0	IPF	E	1	7	x	x
1	CCU	IPF	3	8	2	x
2	E	IPF	4	6	x	x
3	CCU	F	x	x	4	x
4	IPF	CCU	5	9	0	x
5	F	CCU	x	x	1	x
6	E	CCU	x	x	x	QUIT
7	CCU	E	x	x	x	QUIT
8	CCU	F	x	x	6	x
9	F	CCU	x	x	7	x

Fig. 7. OUTPUT or EDITING Flow Table

The third type of I/O processing involves control command interpretation and editing command execution. Control command interpretation is handled using the flow table in figure 8. A new symbol appears in this flow table; CIU meaning the Control Interpreter is operating on the C buffer. When the C buffer is filled for a terminal, the Channel Controller's control word is modified to prevent the A and B buffers from being filled again since the buffers may be required for editing command execution. When the Control Interpreter shutdown is CISC the Channel Controller is again allowed to fill the A and B buffers. When the Control Interpreter shutdown is CIED a check is performed to verify that the terminal mode is INPUT and that the I/O state is less than 6. If these conditions are met, the A and B buffers are emptied and the editing command is executed using the flow table in figure 7.

CONTROL STATE	BUFFER C		SHUTDOWNS		
		CCCC	CIED	CISC	IPEOR
A	CCF	B	x	x	x
B	CIU	x	C	A	x
C	CCF	D	x	x	A
D	F	x	x	x	B

Fig. 8. CONTROL Flow Table

To maintain information on what types of work are ready to be performed three queues will be used in the I/O Supervisor[26]. The I/O queue will contain all terminals in I/O modes. The IP queue will contain all terminals needing service by the Interface Processor. The CI queue will contain all terminals needing service by the Control Interpreter. Whenever the Interface Processor or the Control Interpreter completes a task, the I/O Supervisor looks in the appropriate queue to find a different task.

Another responsibility of the I/O Supervisor is analysis of available information to determine the terminal type identification and updating of the subsystem control words to reflect that information[27].

The organization of the I/O Supervisor is shown in figure 13.

Code Converter

The Code Converter consists of an array of read-only memory programmed to provide the necessary code conversion. Each terminal type will require 512 words of twelve bits. The code conversion will be controlled by a scanner which will look at the code converter registers in each Terminal Controller and see if a transformation can be executed. If possible, the array will be accessed by the scanner using the data in the Terminal Controller's OIREG or buffer register as an address and the returned data stored in the Terminal

Controller's OOREG or IIREG. In this manner the code conversion can be performed very rapidly because the microprocessor in the Terminal Controller is not involved.

The organization of the Code Converter is shown in figure 14.

I/O Memory

The I/O Memory will be organized in words of 8 characters or 64 bits. If the three low order bits of an address are all zeroes, the memory will consider the address to refer to a word and read or write 64 bits of information. If the three low order bits of an address are not all zeroes, the memory will consider the address to refer to a character and read or write only 8 bits of information. The memory will consider only the eight low order lines of the data path to be significant in this case.

The organization of the I/O Memory is shown in figure 15.

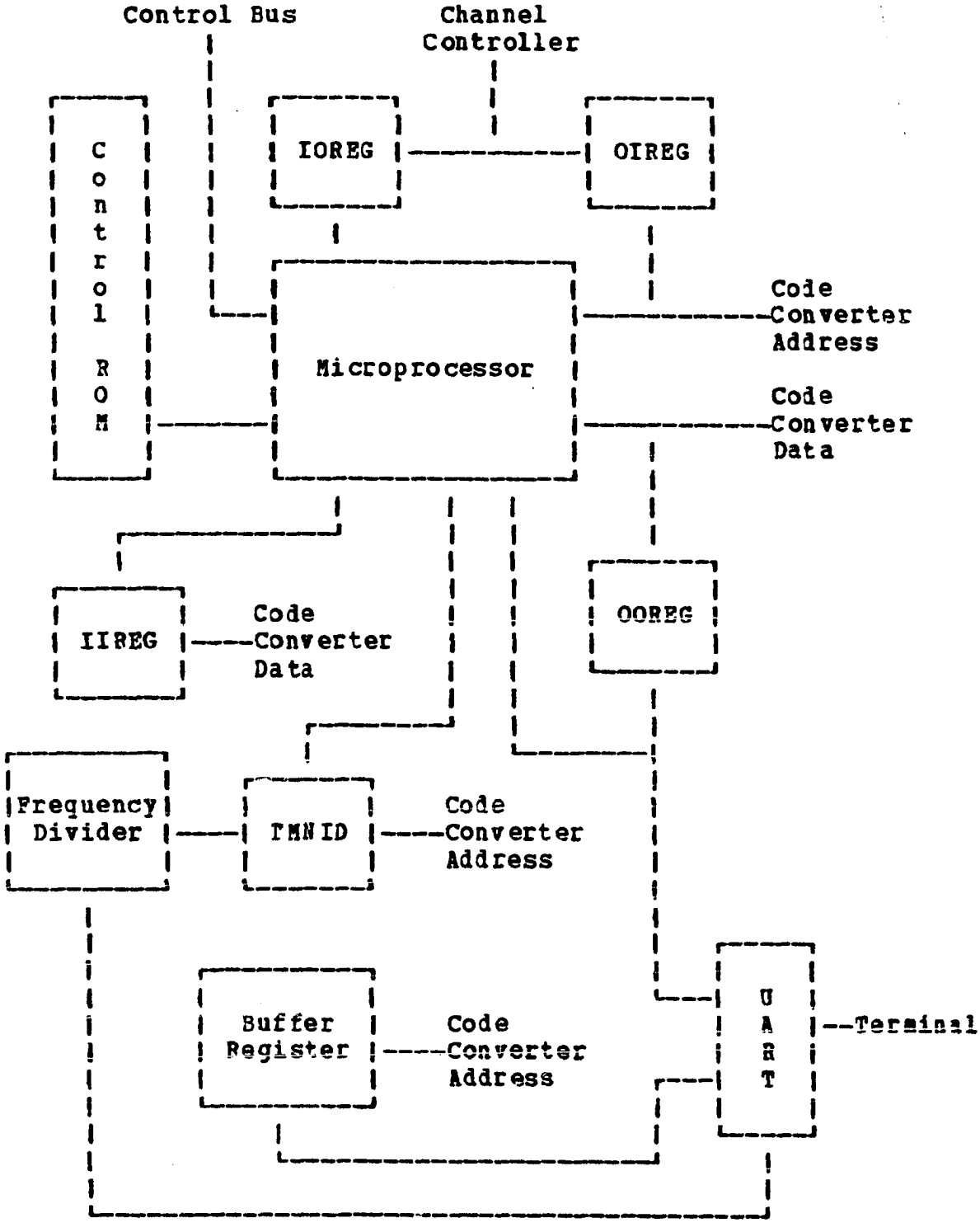

Fig. 9. Terminal Controller Block Diagram

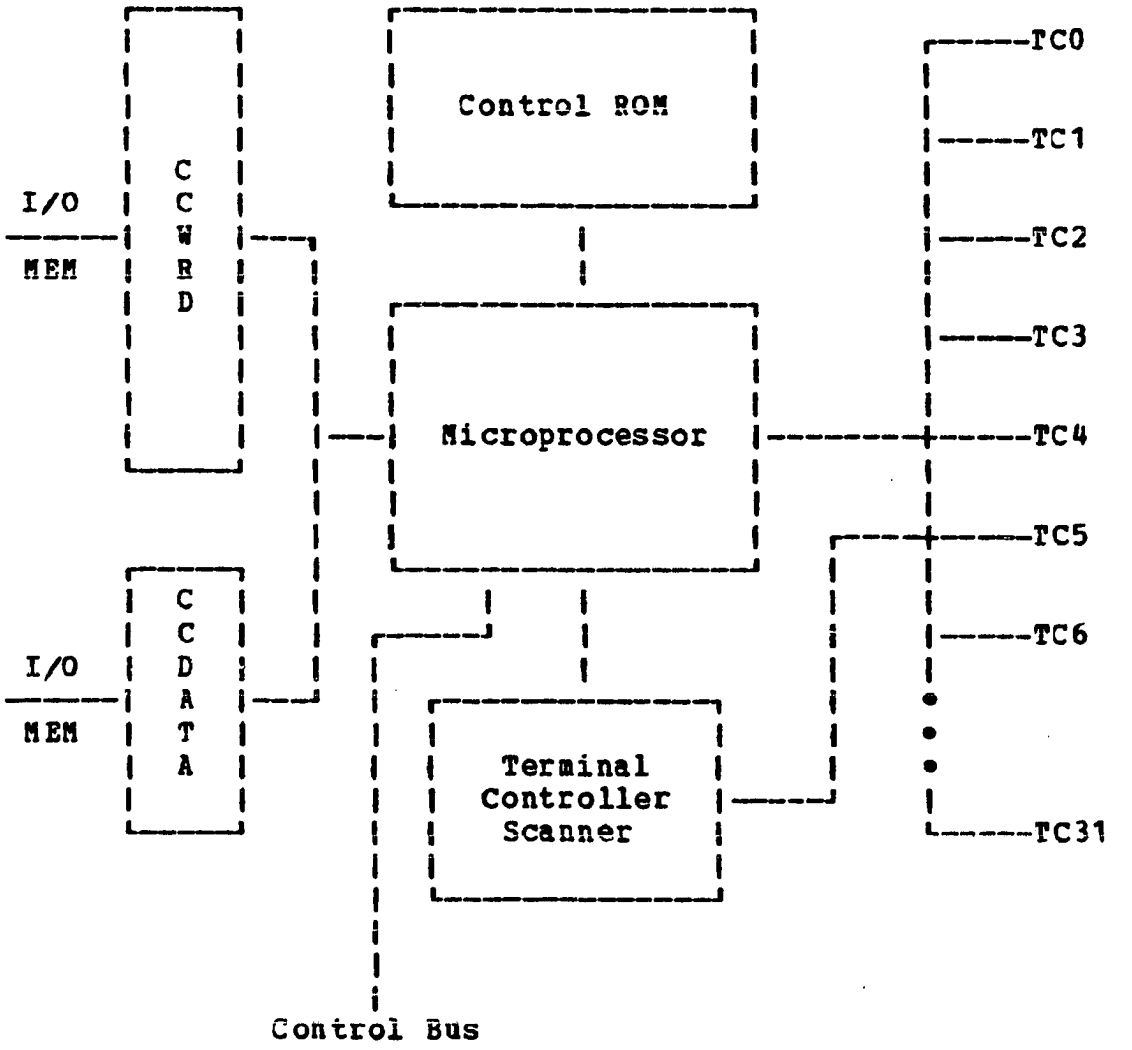

Fig. 10. Channel Controller Block Diagram

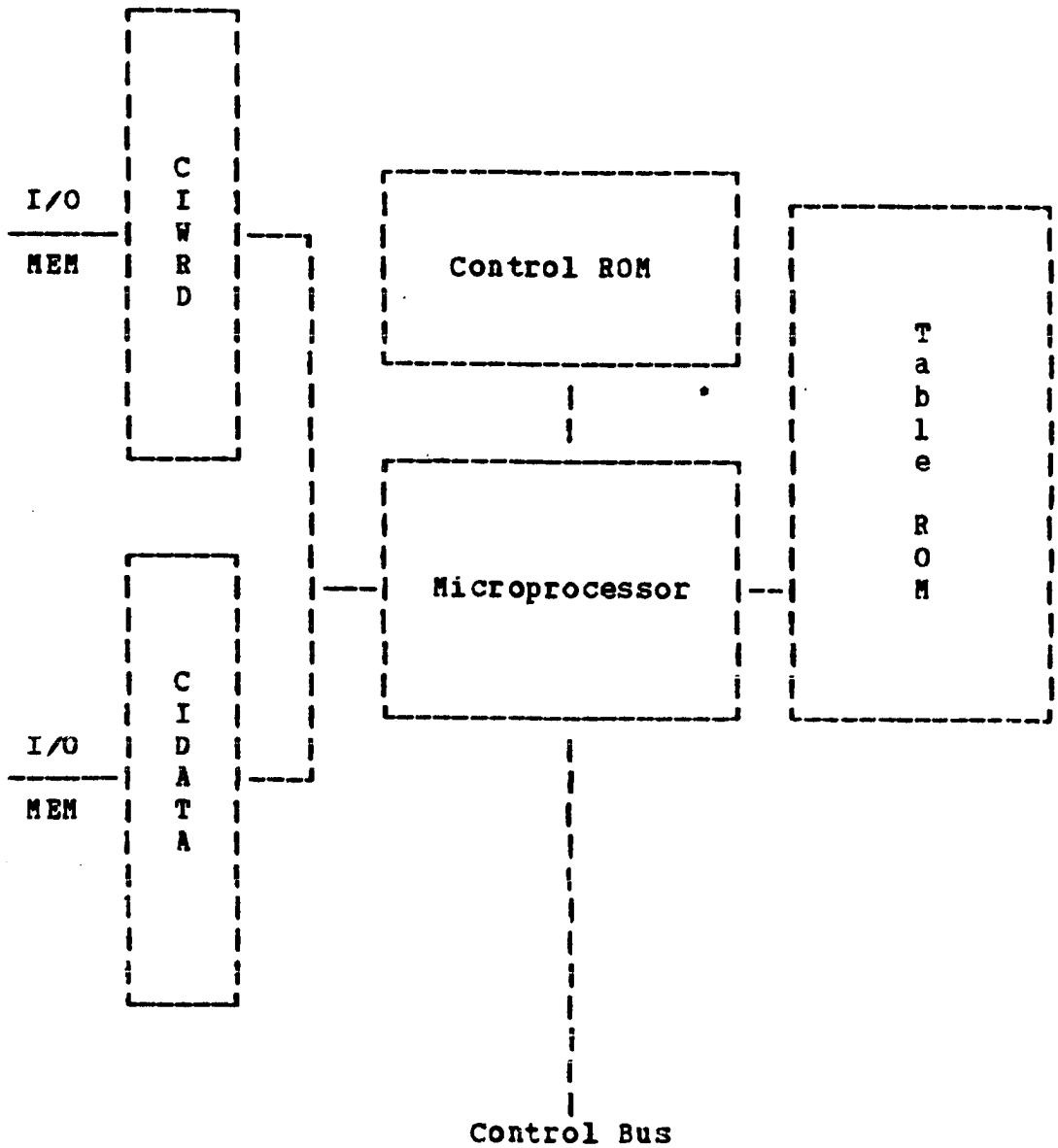

Fig. 11. Control Interpreter Block Diagram

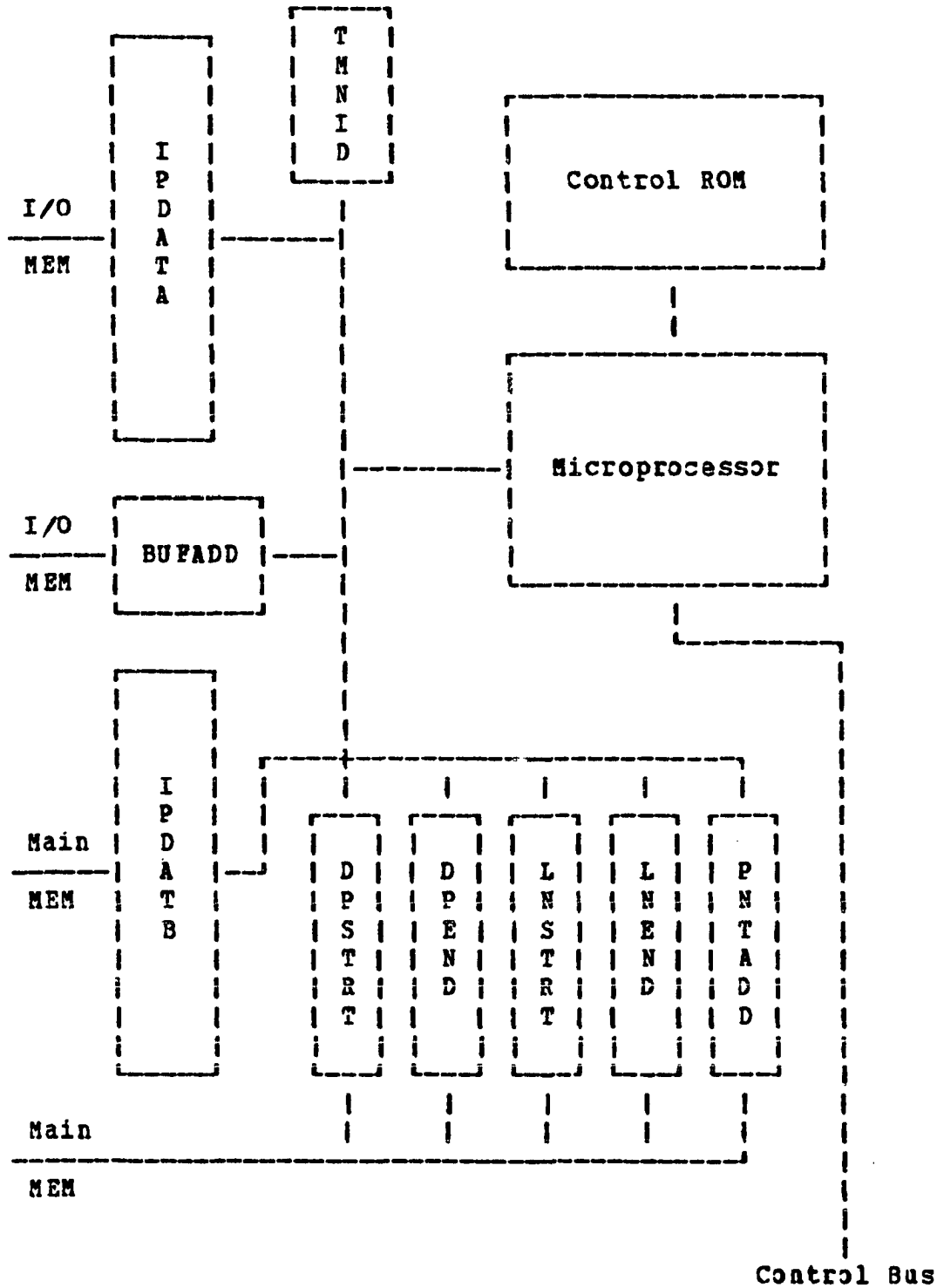

Fig. 12. Interface Processor Block Diagram

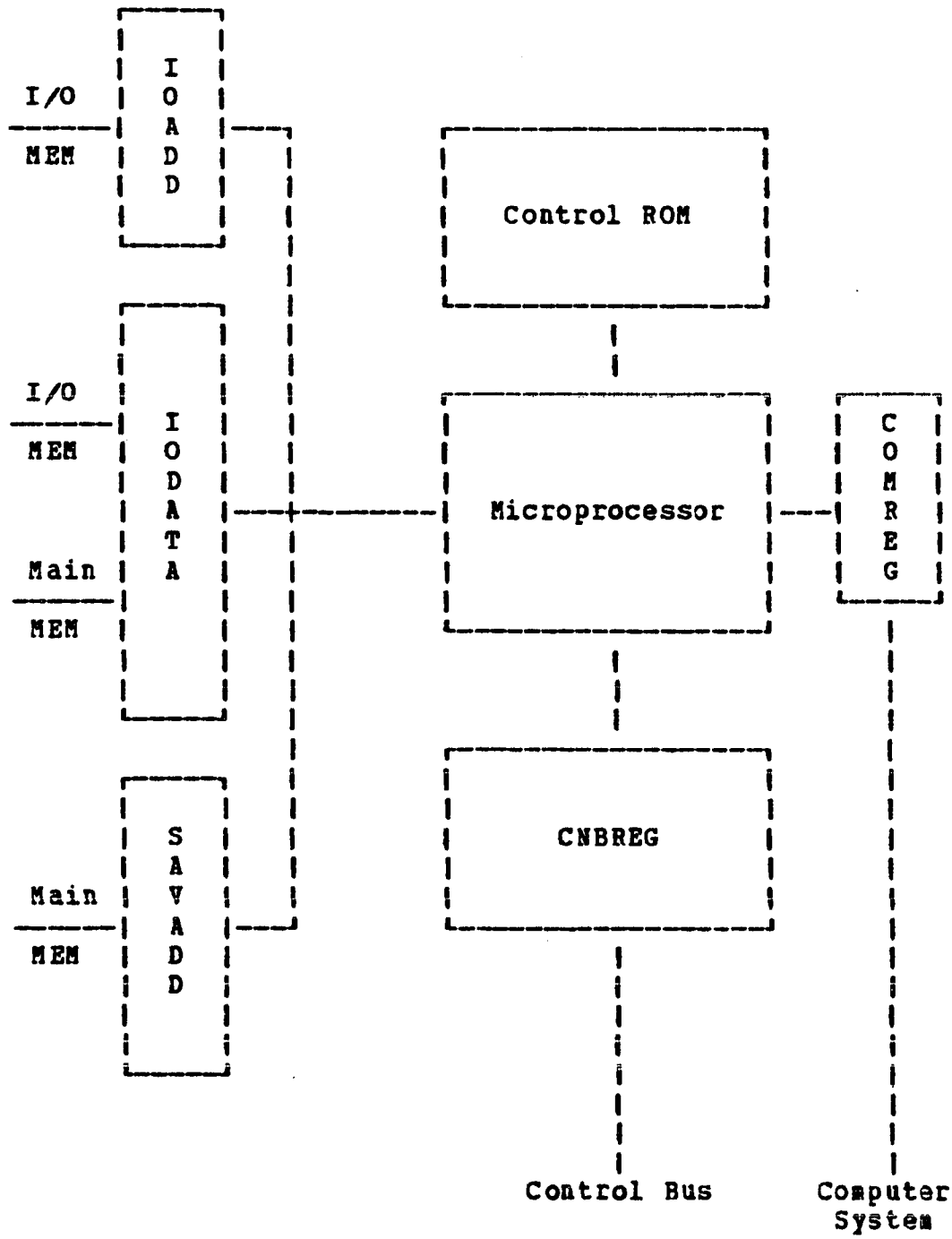

Fig. 13. I/O Supervisor Block Diagram

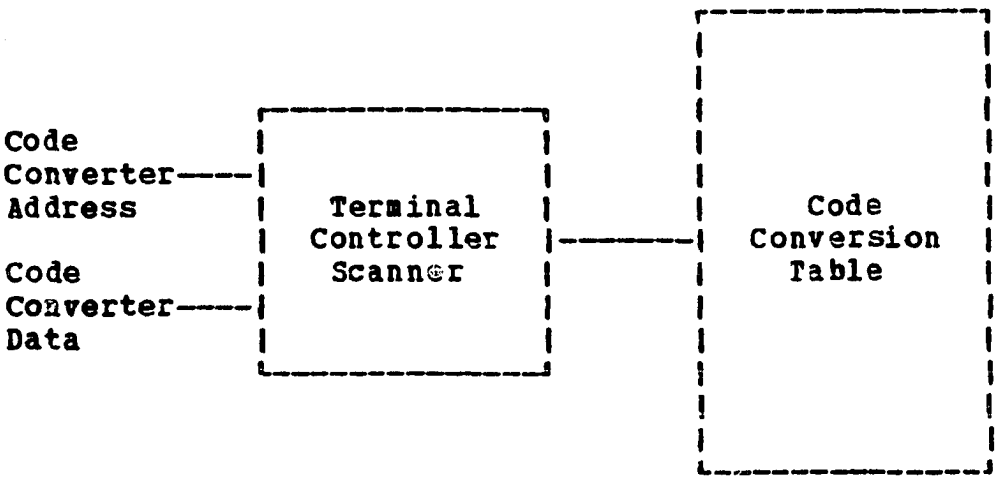

Fig. 14. Code Converter Block Diagram

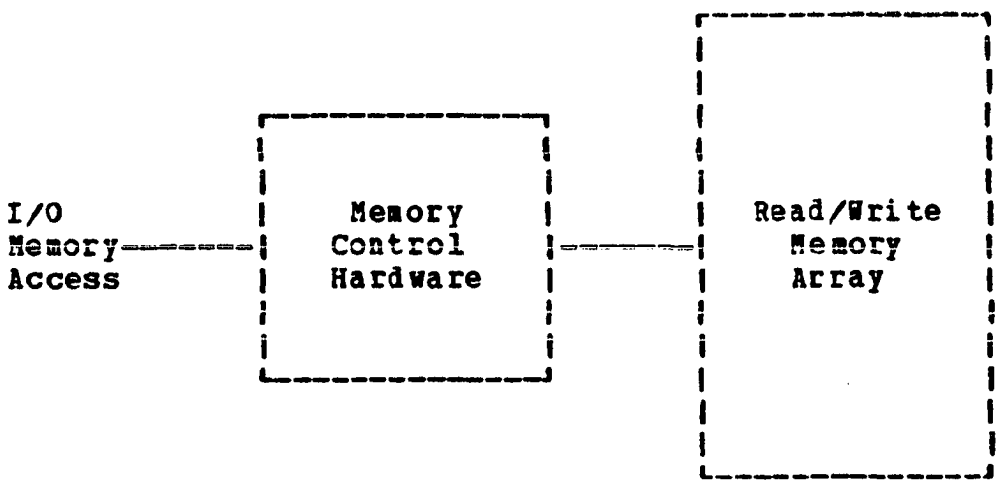

Fig. 15. I/O Memory Block Diagram

CONCLUSIONS

The proposed I/O subsystem satisfies nearly all of the criteria set forth in an early chapter of this paper. Thirty-two terminals operating at 9600 baud produce a maximum of 28,000 characters per second. This rate forces the Channel Controller to perform a character transfer every 35 microseconds and the Interface Processor to handle a 64 character buffer every 2 milliseconds. Although both of these speeds are within the range of current technology, extra consideration needs to be devoted to the microprocessor in the Channel Controller. Further information on this problem appears in the Appendix. The 35 microsecond minimum is not typical since it is unlikely that all 32 terminals will be operating at 9600 baud for extended periods of time. If an average of 70 microseconds can be allotted for each character transfer, a slower but more easily implemented Channel Controller results through the use of a slower but simpler microprocessor.

On-line editing facilities are provided with acceptable response speeds. The above character rates are maximums and editing operations typically do not require high speed exchanges between the system and the terminal. Thus, the response times should not be degraded because of long editing execution times.

By placing the terminal control functions in the Terminal Controller, the rest of the I/O subsystem is not involved with terminal operation and thus can be utilized for other purposes. The detection and decoding facilities for system commands allow the user to control the system with commands that are easily understood. The code conversion problem is converted to an addressing operation on a read-only memory.

The capability to interrupt an I/O operation, run a supervisory program on the terminal, and then complete the I/O operation provides a convenient means of communication between the system and the terminal even during I/O tasks. Programs may be stacked more than one deep on a terminal simply by having a second supervisory routine push down the first one, etc.

Simplicity is another feature of the subsystem. With only registers external to the microprocessors, the operation of each processor is straightforward. The flexibility is also enhanced by this type of design. Since the control sequences are provided by read-only control memories, initialization is only a matter of setting up the standard control words and starting the I/O subsystem.

This project is completed to the extent that the subsystem is functionally specified. Detailed design was not included in the effort because the microprocessor field is a

rapidly evolving technology and the newest products offer magnitudes of improvement in performance over the capabilities of the earlier types. Only recently have devices become available that possess the performance characteristics required for this I/O subsystem.

This I/O subsystem as specified can operate with any main computer system capable of supplying control information and data to the subsystem and accepting data from the subsystem. The code conversion tables and the control memories for the I/O Supervisor and the Interface Processor will need to be modified to meet the specifications of the main computer system, but the general subsystem will remain unchanged.

Further work in this area should include consideration of the scheduling algorithms needed for optimum performance, particularly with regard to editing commands and the Interface Processor. Another important consideration is the serviceability aspect. With many dedicated processors in the subsystem, complex diagnostic programs must be used to verify correct performance. Since timing plays an important role, this testing must be done dynamically. As an exhaustive test would be impossible, a compromise procedure must be developed that will run in a reasonable length of time and still verify that the subsystem operates as intended.

ACKNOWLEDGEMENTS

The author wishes to thank Professor T. A. Smay and Professor R. J. Zingg for their suggestions, encouragement, and patience throughout this project.

LITERATURE CITED

1. Bell, C. Gordon and Gold, Michael M. "An Introduction to the Structure of Time-shared Computers." In Advances in Information Systems Science, pp. 161-272. Edited by Julius T. Tou. New York: Plenum Press, 1972.

2. Bull, G. M. and Packham, S. F. G. Time-sharing Systems. London: McGraw Hill Book Company Limited, 1971.

3. Martin, James. Design of Man-Computer Dialogues. Englewood Cliffs, N.J.: Prentice-Hall, Inc., 1973.

4. Miller, Robert B. "Response Time in Man-computer Conversational Transactions." Proceedings of the 1968 Fall Joint Computer Conference. Washington, D.C.: Thompson Book Company, 1968.

5. Mills, David L. "Preprocessors in a Data Communication Computer Environment." Proceedings of the ACM Symposium on Problems in the Optimization of Data Communications Systems. New York: Association for Computing Machinery, 1969.

6. Smith, W. R.; Rice, Rex; Chesley, Gilman D.; et al. "SYMBOL: A Large Experimental System Exploring Major Hardware Replacement of Software." Proceedings of the 1971 Spring Joint Computer Conference. Montvale, N.J.: AFIPS Press, 1971.

7. Saltzer, J. H. and Ossanna, J. F. "Remote Terminal Character Stream Processing in MULTICS." Proceedings of the 1970 Spring Joint Computer Conference. Montvale, N.J.: AFIPS Press, 1970.

8. Saltzer, J. H. and Ossanna, J. F. "Technical and Human Engineering Problems in Connecting Terminals to a Time-sharing System." Proceedings of the 1970 Fall Joint Computer Conference. Montvale, N.J.: AFIPS Press, 1970.

9. McCarthy, John; Brian, Dow; Feldman, Gary; and Allen, John. "THOR--A Display-based Time-sharing System." Proceedings of the 1967 Spring Joint Computer Conference. Washington, D.C.: Thompson Book Company, 1967.

10. Kingslake, R. "TALK-An Interactive System for a Small Computer." Software--Practice and Experience 1 (October 1971): 391-401.

11. Borman, P. R. and Grimm, E. H. III. "An Analysis of Selected Interactive Systems." Interactive Computing. Berkshire, England: INFOTECH Information Limited, 1972.

12. Deutsch, L. Peter and Lampson, Butler W. "An Online Editor." Communications of the ACM 10 (December 1967): 793-9, 803.

13. Benjamin, Arthur J. "An Extensible Editor for a Small Machine with Disk Storage." Communications of the ACM 15 (August 1972): 742-7.

14. Bourne, S. R. "A Design for a Text Editor." Software--Practice and Experience 1 (January 1971): 73-81.

15. Fajman, Roger and Borgelt, John. "WYLBUR: An Interactive Text Editing and Remote Job Entry System." Communications of the ACM 16 (May 1973): 314-22.

16. Jones, Wayne E. "The Role of the Interface Processor in the SYMBOL IIR Computer System." Special Report NSF-OCA-GJ33097-CL7304. Cyclone Computer Laboratory. Iowa State University. Ames, Iowa, 1973.

17. "SYMBOL 2R Channel Controller Flow Charts." Unpublished document. Fairchild Semiconductor Research and Development Laboratory. Palo Alto, Calif., 1970.

18. "SYMBOL 2R Interface Processor Flow Charts." Unpublished document. Fairchild Semiconductor Research and Development Laboratory. Palo Alto, Calif., 1970.

19. "SYMBOL 2R Job Control Flow Charts." Unpublished document. Fairchild Semiconductor Research and Development Laboratory. Palo Alto, Calif., 1970.

20. "SYMBOL 2R Job Controller Specification." Unpublished document. Fairchild Semiconductor Research and Development Laboratory. Palo Alto, Calif., 1969.

21. "SYMBOL 2R Memory Specifications--SP028E." Unpublished document. Fairchild Semiconductor Research and Development Laboratory. Palo Alto, Calif., 1970.

22. Wasserman, Anthony I. "The Design of 'Idiot Proof' Interactive Programs." Proceedings of the 1973 National Computer Conference and Exposition. Montvale, N.J.: AFIPS Press, 1973.

23. Hayman, E. "Design Criteria for CRT Alphanumeric Displays." 1969 International Symposium on Man-Machine Systems. New York: Institute of Electrical and Electronics Engineers, 1969.

24. McLaughlin, Richard A. "Alphanumeric Display Terminal Survey." Datamation 19 (November 1973): 71-92.

25. McLaughlin, Richard A. "Fast Interactive Hardcopy Terminals." Datamation 19 (October 1973): 77-80.

26. Gotterer, Malcolm H. "On the Development of a Supervisory Sequencing Routine." On Line Data Processing. New York: Institute of Electrical and Electronics Engineers, 1963.

27. Ossanna, Joseph F. "Identifying Terminals in Terminal-oriented Systems." Second Symposium on Problems in the Optimization of Data Communications Systems. New York: Association for Computing Machinery, 1971.

RELATED LITERATURE

Barrett, E. E. "Memory Considerations for an On-line Processor." On Line Data Processing. New York: Institute of Electrical and Electronics Engineers, 1963.

Black, W. Wayne. An Introduction to On-line Computers. London: Gordon and Breach, Science Publishers, Ltd., 1971.

Boies, S. J. "User Behavior on an Interactive Computer System." IBM Systems Journal 13 (January 1974): 2-18.

Bratman, Harvey; Martin, Hiram G.; and Perstein, Ellen Clark. "Program Composition and Editing with an On-line Display." Proceedings of the 1968 Fall Joint Computer Conference. Washington, D.C.: Thompson Book Company, 1968.

Chu, W. W. "A Study of Asynchronous Time Division Multiplexing for Time-sharing Computer Systems." Proceedings of the 1969 Fall Joint Computer Conference. Montvale, N.J.: AFIPS Press, 1969.

Fein, Louis. "Assessing Computing Systems." On Line Data Processing. New York: Institute of Electrical and Electronics Engineers, 1963.

Fitzsimmons, Thomas F. "ASCII Extension and Expansion and Their Impact on Data." Second Symposium on Problems in the Optimization of Data Communications Systems. New York: Association for Computing Machinery, 1971.

Fraser, A. G. "On the Interface Between Computers and Data Communication Systems." Communications of the ACM 15 (July 1972): 566-73.

Groner, Gabriel F. "Display Terminals Can Help People to Use Computers." Proceedings of the 1973 National Computer Conference and Exposition. Montvale, N.J.: AFIPS Press, 1973.

Hansen, Wilfred J. "User Engineering Principles for Interactive Systems." Proceedings of the 1971 Fall Joint Computer Conference. Montvale, N.J.: AFIPS Press, 1971.

Meadow, Charles T. Man-Machine Communication. New York: John Wiley and Sons, Inc., 1970.

Parnas, David L. "On the Use of Transition Diagrams in the Design of a User Interface for an Interactive Computer System." Proceedings of the 24th National Conference. New York, N.Y.: Association for Computing Machinery, 1969.

Rice, David E. and Van Dam, Andries. "An Introduction to Information Structures and Paging Considerations for On-line Text Editing Systems." In Advances in Information Systems Science, pp. 93-158. Edited by Julius T. Tou. New York: Plenum Press, 1972.

Rosenblum, Stanley R. "Progress in Control Procedure Standardization." Second Symposium on Problems in the Optimization of Data Communications Systems. New York: Association for Computing Machinery, 1971.

Ross, H. McGregor. "The British Standard Data Code and How to Exploit It." Computer Journal 13 (August 1970): 223-9.

Teperman, A. and Katzenelson, J. "A Format Editor." Software--Practice and Experience 2 (July 1972): 219-30.

Van Dam, Andries and Rice, David E. "On-line Text Editing: A Survey." Computing Surveys 3 (September 1971): 93-114.

Wilkes, Mary Allen. "Scroll Editing: An On-line Algorithm for Manipulating Long Character Strings." IEEE Transactions on Computers 19 (November 1970): 1009-15.

Williams, Robin. "A Survey of Data Structures for Computer Graphics Systems." Computing Surveys 3 (March 1971): 1-21.

APPENDIX

The performance characteristics of the I/O subsystem are affected by the response speed of all of the processors within the subsystem. However, the most critical performance specifications are in the Channel Controller. More than 100 instructions can be executed for each character in the Terminal Controllers and the Interface Processor without degrading the response time of the system. In the Channel Controller only about 10 instructions can be executed for each character if all 32 terminals are operating at their maximum rates. Maximum rates can occur only during standard I/O operations, i.e., without control commands. When control commands appear in the data stream from a terminal, that terminal's average data rate decreases since the user is utilizing much of the time deciding which operation to perform. The worst case for the Channel Controller is during the INPUT mode because checks must be performed on the high order bits from the Terminal Controller to determine the proper routing of characters. During the OUTPUT mode these checks are unnecessary because all characters are transferred to the Terminal Controller.

To justify the feasibility of a microprocessor-based Channel Controller, a control sequence for an Intel 8080 microprocessor was constructed to perform the worst case operation described above. The control sequence is shown in

figure 16. The execution time for the sequence shown is 73 microseconds with a 2 microsecond instruction cycle time.

```
Execution
Time (us)  Label  Mnemonic  Operand      Description

   5       TOP:   OUT       HOLD     ;Wait for scanner to
                                      ; release HOLD line.
   5              IN        GETCW    ;Fetch control word from
                                      ; I/O Memory and load
                                      ; selected byte of CCWRD
                                      ; into accumulator.
   3.5            CPI       COH      ;Check INPUT flag (bit 8)
                                      ; and CONTROL flag
                                      ; (bit 7).
   5              JNC       CONCN    ;Branch to control routine
                                      ; if bit 7 and bit 8 are
                                      ; not both '1'.
   5              IN        CINFO    ;Fetch 4 high order bits
                                      ; from TC IOREG.
   3.5            CPI       00H      ;Check that the bits are
                                      ; all '0'.
   5              JNZ       BEGCN    ;Branch to control routine
                                      ; if any bits are not '0'.
   5              IN        FCHAR    ;Fetch character from
                                      ; from TC IOREG.
   3.5            CPI       0EH      ;Check if end of record.
   5              JZ        EOR      ;Branch to shutdown
                                      ; routine if end of
                                      ; record.
   5              OUT       SCHAB    ;Load CCDATA with
                                      ; character and transfer
                                      ; to I/O Memory.
   5              IN        BUFAD    ;Fetch buffer address
                                      ; from CCWRD.
   2.5            INR       A        ;Increment buffer address.
   5              OUT       STCRW    ;Store buffer address in
                                      ; CCWRD and return control
                                      ; word to I/O Memory.
   5              JC        BFE      ;Branch to shutdown
                                      ; routine if buffer full.
   5              JMP       TOP      ;Return to top of program.
  ---
  73us
```

Fig. 16. Channel Controller Standard INPUT Control Program