Report No. 334

THE UNIVERSITY OF ILLINOIS ATTACHED MACHINE OPERATING SYSTEM: DESIGN CRITERIA AND PROGRAM LOGIC

by

Richard A. Wells

June, 1969

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN · URBANA, ILLINOIS

Report No. 334

THE UNIVERSITY OF ILLINOIS ATTACHED MACHINE OPERATING SYSTEM: DESIGN CRITERIA AND PROGRAM LOGIC*

by

Richard A. Wells

June, 1969

Department of Computer Science
University of Illinois
Urbana, Illinois 61801

* This work was supported in part by the National Science Foundation under Grant No. NSF-GP-7634 and was submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science, June, 1969.

ACKNOWLEDGMENT

The author is grateful for the invaluable advice and assistance of Professor H. G. Friedman, who supervised the AMOS project, and to those University representatives of IBM Corporation whose efforts were instrumental in the development of the system.

PREFACE

In the spring of 1968, an IBM 1800 computing system was installed in the Digital Computer Laboratory at the University of Illinois and attached to the System/360 Computer Complex.

Various factors dictated the development of a new operating system for the 1800. This paper is a summary of the design decisions made and the resulting program logic flow of that operating system.

TABLE OF CONTENTS

TABLE OF CONTENTS--CONTINUED

1. INTRODUCTION

In the spring of 1968, an IBM 1800 computing system was installed in the Digital Computer Laboratory at the University of Illinois. The primary purpose of the 1800 computer is to provide an extended analog processing capability to the University. Prior to the 1800, analog processing was done on the ILLIAC II computer, which was disassembled on August 31, 1968.

There were several reasons for the selection of an 1800 system. Most important was the requirement that analog processing be an integral part of the University of Illinois System/360 complex without disproportionately overloading the 360 computers. The problem of overload precluded the direct attachment of an analog control unit to a 360 computer; the alternative was a central processing unit dedicated to analog processing, with a hardware attachment to the System/360 complex for access to 360 facilities. In view of the centralization requirement, it was desirable that the analog processing CPU have the ability to utilize 360 peripheral equipment. Another obvious reason was the savings that would be realized by not having to duplicate peripheral devices.

The solution to these difficulties was the 1800 computer, interfaced to a 360 system as shown in Figure 1.

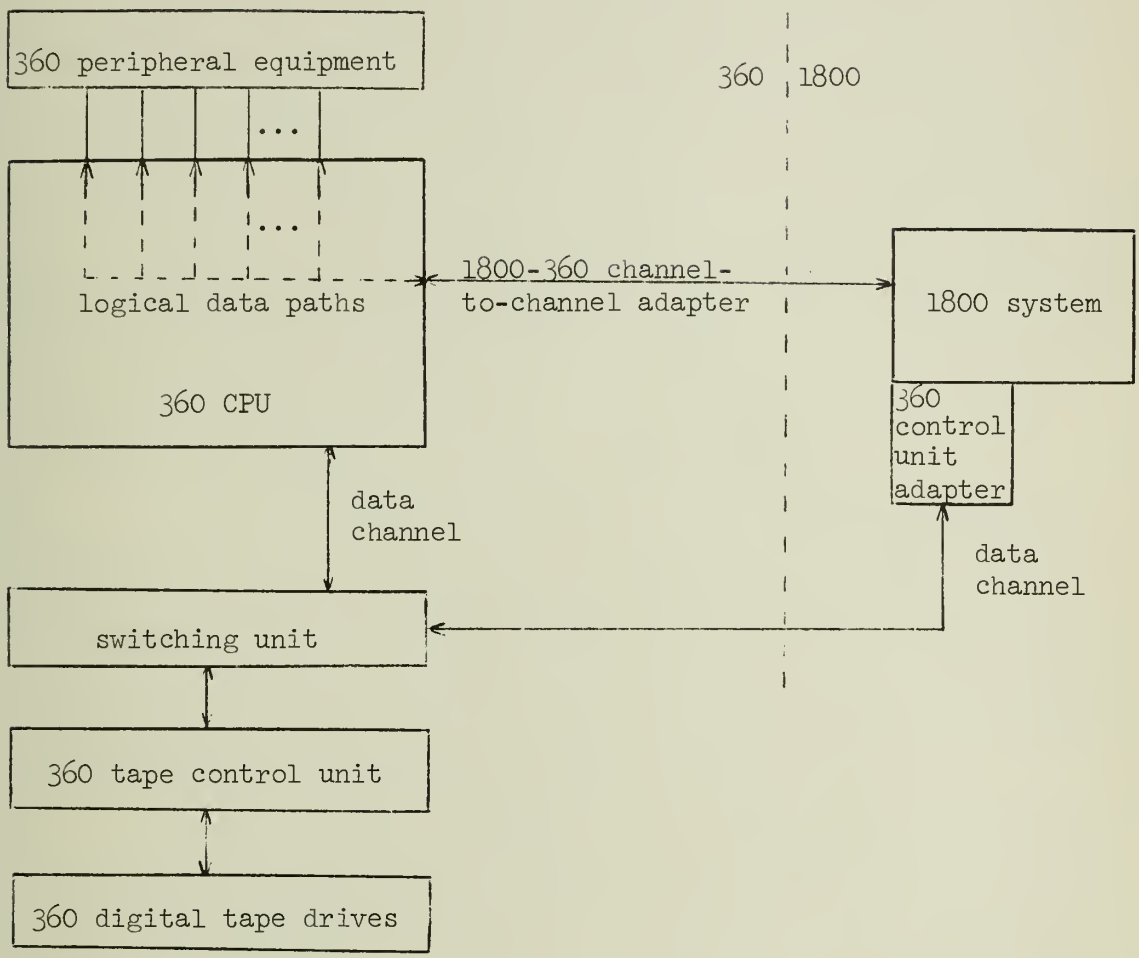


Figure 1.

Direct communication between the 1800 and 360 computers is provided by the 1800-360 channel-to-channel adapter, a device which permits either CPU to request data transmission[1] with the other by interrupt. The interrupted CPU responds to a request by determining in which direction the data flow is to occur and then issuing the appropriate command to the adapter, which initiates data transmission.

Since exclusive use of the adapter by the 1800 for analog processing purposes would require extensive 360 CPU attention (thus defeating the purpose of a separate CPU), a more direct interface to 360 input/output equipment was also necessary. This problem was solved by an 1800 special feature which allows the attachment of a 360 control unit directly to an 1800 data channel. Specifically, the 1800 channel was routed through a switching unit to a 360 tape control unit, allowing the 1800 access to 9-track digital tapes.[2]

This configuration permits the 1800 to obtain control and diagnostic information from the 360 computer via the adapter (also permitting the 360 to supervise execution of jobs on the 1800), and yet analog processing tasks can be performed on the 1800 independent of the 360 CPU (by using the 360 digital tape drives).

From a hardware standpoint, the configuration is ideal in that the best features of an independent CPU and a dependent control unit are combined, with the 1800 taking whatever form is necessary at any given time.

Footnotes

[1] at rates up to 250,000 bytes per second.

[2] maximum data rate of 180,000 bytes per second.

2. CONTROL PROGRAM DESIGN CRITERIA

A survey of currently available software systems for the 1800 quickly showed that none could support the hardware specifications outlined in Section 1. In particular, the two operating systems provided by IBM[1] for the 1800 are strictly for stand-alone 1800 systems. They each require 1800 disk storage, 1800 unit record equipment, and extensive amounts of core storage; furthermore, no provision is made for attached 360 facilities.

This situation necessitated the design and implementation of an 1800 operating system tailored to the specific needs of the University of Illinois. Several key requirements became obvious during the design stage of that operating system (now known as AMOS - Attached Machine Operating System). Those requirements are presented below.

2.1 1800 Hardware Minimization

In order to minimize duplication of 360 hardware on the 1800, it was necessary to design the 1800 control program so as to utilize existing 360 peripheral equipment via the channel-to-channel adapter. The method chosen involved allocation of the adapter into sixteen logical files, with each file "attached" to an appropriate 360 device by a control program on the 360 computer. Thus, such functions as system input (card images), system output (print line images), access to library subroutines, etc., were each assigned a logical file across the channel-to-channel adapter. As a result, the minimum 1800

hardware configuration necessary for AMOS operation included:

1800 central processing unit (1801 or 1802)

minimum of 8,000 words of core storage

1800-360 channel-to-channel adapter

1816 console typewriter

Initial Program Load (IPL); either a 1442 card reader

or a 1054 paper tape reader

Obviously, additional equipment was necessary for application programming purposes, such as the digital tape interface, analog/digital converters, etc. A complete description of the University of Illinois 1800 hardware configuration can be found in Department of Computer Science Report No. 280.[2]

2.2 System/360 Software Independence

Responses to 1800 requests across the channel-to-channel adapter must be performed by a supervisory program which resides in System/360 internal storage. Such a program should be responsible for satisfying 1800 requests, monitoring the status of the job executing on the 1800, and serving as an interface between the two computing systems.

Methods of communication across the adapter were organized in such a manner as to make the 1800 AMOS supervisor as logically independent of its 360 counterpart as possible. This decision was based on the modularity of 360 hardware, which has resulted in the availability of a multitude of 360 programming systems. As a result, future changes in System/360 software should have a minimum impact on

the AMOS system; only the 360-resident 1800 control program need be rewritten.

As a brief example, consider a request from the 1800 supervisor (AMOS) for the next system input (SYSIN) card image. AMOS expects the 360 program to respond by sending a card image across the adapter; AMOS cares not at all where the 360 program obtained the card image.

2.3 Data Reduction and Program Compilation

The fact that the 1800 instruction repetoire lacks both floating-point and character manipulation instructions suggests that 1800 application program compilations or assemblies and 1800 data reduction tasks should be removed from the 1800 to the attached System/360 computers. This decision was further enhanced by the availability of a 360-resident 1800 assembler, which proved to be easily adaptable to the AMOS system. Subsequent experience with the assembler has shown it to be quite reliable and far faster and more flexible than its stand-alone 1800 counterpart.

The lack of a FORTRAN compiler was not expected to be a serious problem since data reduction on the 1800 was to be minimized; experience to date has shown this decision to be essentially correct.

2.4 User Control/Communication

Perhaps the most serious problem encountered in the design of AMOS was the conflict between the 1800 user and the attached System/360 computer for control of the 1800.

The very nature of analog processing requires a high degree of "hands on" participation by humans. For example, consider the various sources of analog data: AM/FM tape recorders, process control and experimentation equipment, etc. In contrast, the lack of 1800 unit record devices and the requirement that 1800 job accounting be done on the System/360 dictated that jobs be routed to the 1800 from the 360 only. This implied that users present for the online handling of their 1800 jobs would be at the mercy of the System/360 job scheduling programs. To complicate the problem, many 1800 jobs would need access to 360 computation facilities prior to 1800 execution for assembly purposes; thus, such jobs would be in competition with the entire System/360 job queue, while the 1800 user would have no recourse but to wait for an unknown period of time for his job to become active on the 1800.

A possible answer to this situation involved the modification of the System/360 job scheduling routines so that jobs requiring 1800 processing would be expedited through the 360 job queue. Experience with this solution to date has shown it to be satisfactory for both the 1800 user and the System/360 job queue, since System/360 assembly of an 1800 program typically requires only a few seconds of 360 processing time, and as a result normal 360 job processing is not appreciably delayed.

This situation is expected to improve even further in the future for two reasons: the addition of System/360 multiprogramming support and the availability of a generalized analog/digital conversion program to the AMOS system. The latter program will eliminate the necessity of user-written programs (requiring 360 assembler time) for the great majority of 1800 users, therefore allowing most 1800 jobs to compete only with jobs in the 1800 queue. (This program is described more fully in Section 4.1.) System/360 multiprogramming support should make the 360 CPU more accessible to small, fast-running jobs, such as 1800 program assemblies.

The fact that this problem has not been as severe as first expected has given rise to the possibility of its other extreme: namely, the 1800 job arriving at the 1800 computer before the user has had time to prepare his equipment. There would appear to be many rather simple solutions to this situation should it appear. For example, one method might involve a system-imposed "hold" on certain 1800 jobs, which would prevent premature execution. At the operator's discrimination, such jobs could be "released" from hold status, allowing them to begin processing.

The overall solution to the 360/user control conflict was based on the fact that normal execution of an 1800 job could be disrupted in one of only four ways:

1) by the 1800 program itself
2) by the 1800 operator (typically a user)
3) by the System/360 computer
4) by lack of proper 360 reaction to an 1800 request.

Discounting system errors, methods 1), 2), and 4) are not applicable; only 3) need be considered.

The possibility of arbitrary 360 intervention was eliminated by requiring that the 360 control program (discussed in Section 2.2) be passive in nature, i.e., the 360 control program should communicate with the 1800 in two instances only:

1) in response to an 1800 request for information transfer across the channel-to-channel adapter;
2) whenever it becomes necessary to terminate execution of an 1800 job due to the existence of an abnormal condition detectable only by the System/360 computer.

An example of the latter condition is when the time estimate of an 1800 job has been exceeded.[3]

In this environment, the 1800 (and therefore the 1800 user) retains control of 1800 job execution unless the situation deteriorates to the point at which it becomes necessary for the 360 control program to intervene.

2.5 System Modularity

In the design and implementation of a new operating system, one must be prepared for unforeseen problems no matter how extensive the planning. These unknown problems can be minimized by applying the concepts of modularity and simplicity to the structure of the system. Modularity is obtained by centralizing similar logic paths and resolving idiosyncrasies of individual paths with the use of tables. This method

requires a minimum of code and the result is easy to modify or expand (it is far simpler to change or add to tables than it is to edit code).

A drawback to modular systems is the additional overhead involved in performing a given function, as exemplified by Operating System/360. Although the minimization of system overhead was quite important in the design of AMOS (due to the nature of analog/digital conversion processes), it was felt that the specific nature of the system (analog data processing) would more than cancel out any overhead introduced by modularity. In order to assure this result, AMOS was designed so that in certain instances normal system logic could be bypassed by the application program; also, a "necessary and sufficient" attitude was taken throughout the development of the system. Basically, this means that each component of AMOS is to perform a necessary function and shall be no more sophisticated than necessary.

2.6 Job Accounting

In order to keep accounting logic as straightforward and simple as possible, it was imperative that 1800 job accounting be centralized on the attached 360 computer within the existing accounting software used for System/360 jobs. The decision for centralized accounting was made prior to the installation of the 360 equipment, and the 360 accounting routines were therefore

written in a general manner, allowing for the insertion of logic for additional computers as they were added to the System/360 complex.

Footnotes

[1] IBM Time-Sharing Executive (TSX) System
IBM Multi-Programming Executive (MPX) System

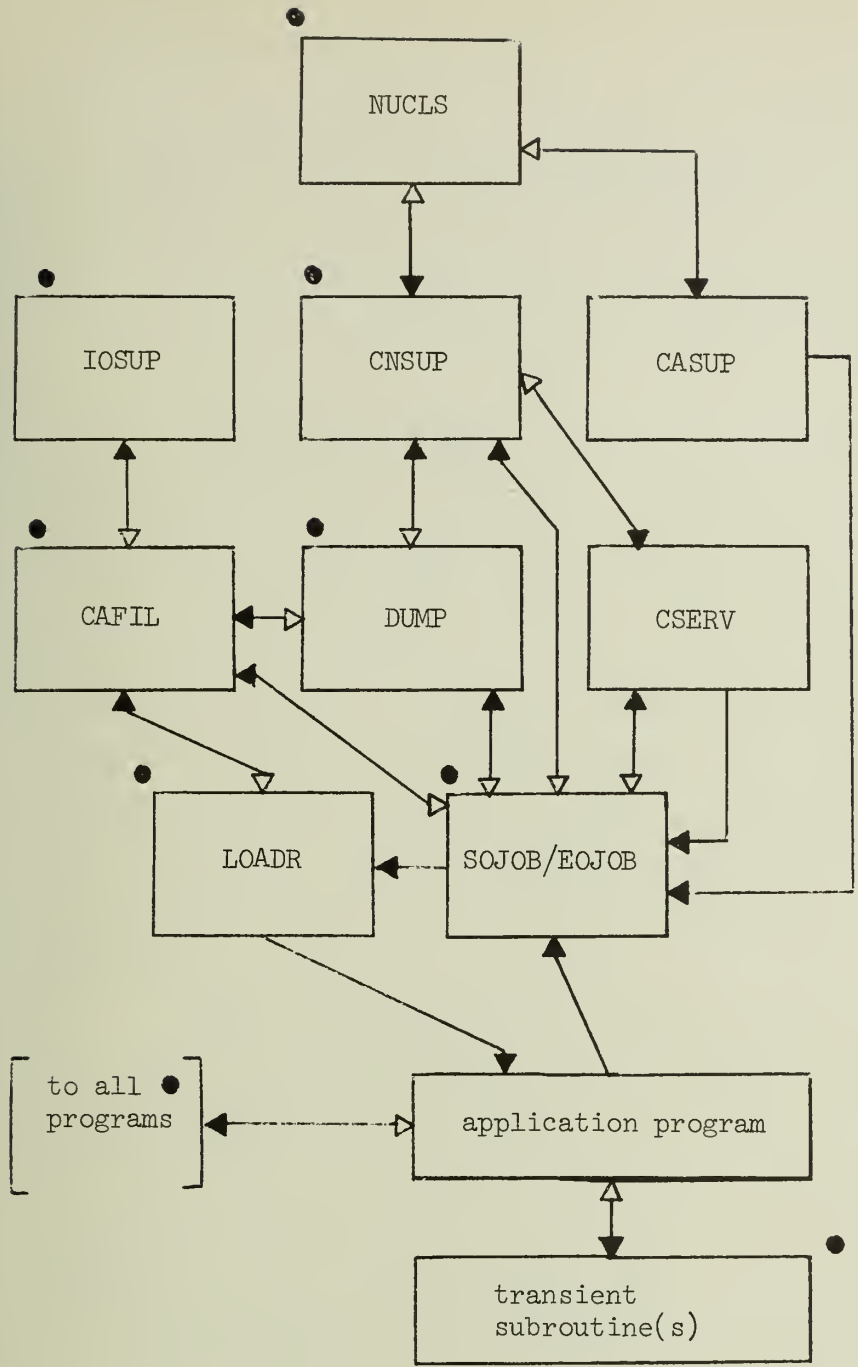
[2] Wells, R. A., Carter, C. E., and Friedman, H. G. "ILLINET Analog Facilities," Department of Computer Science, University of Illinois, Urbana, Illinois, Report No. 280, 1968.

[3] Time, line, and card estimates are monitored by the System/360 in view of Section 2.1.

3. CONTROL PROGRAM LOGIC

This section deals with detailed explanations of the principal programs which comprise the AMOS system. For clarity, AMOS/1800 will be defined as that collection of programs which are 1800 resident; AMOS/360 will denote the 360 control program described in Section 2.2.

Figure 2 gives a brief summary of the communication paths between the main components of AMOS/1800.



low core area (permanently resident programs) - i.e., AMOS/1800

high core area (transient programs) - programs obtained from previous 360 assemblies or AMOS/360 subroutine library

Figure 2.

 denotes call/return linkage in the direction of the darkened arrowhead.

 denotes "call/no return".

● means callable by an application program.

3.1 Basic Concepts

3.1.1 Logical Files of the 1800-360 Adapter

As mentioned earlier, effective utilization of 360 peripheral equipment is obtained by logically dividing the 1800-360 channel-to-channel adapter into sixteen distinct files and assigning each file (logically, by AMOS/360) to some destination or origin within the System/360 complex. For example, all system output data generated by the 1800 is routed by AMOS/1800 to logical file 3 of the adapter. The data is transferred across the adapter to AMOS/360, which is responsible for insuring that the data is eventually printed on a System/360 line printer.

Each of the files may be thought of as a sequential device much like a digital tape unit; i.e., data may be "written" on the file (sent to AMOS/360), the file may be "rewound" (logically repositioned by AMOS/360), and data may be "read" from the file (retrieved from AMOS/360). This concept is quite similar to the one used by the IBM Attached Support Processor program for communication between two System/360 computers and is obviously quite general in nature. Of course, the adapter hardware does not in itself accept such commands as "backspace file 8"; a detailed explanation of how this is done is given in Section 3.2.7.

3.1.2 The Event Control Word

One of the more important aspects of an operating system is its ability to effectively schedule and synchronize the various processes which can function concurrently with, and independently of, the execution of machine instructions within the CPU. These "asynchronous" processes include such items as the incrementing of an interval timer, the transmission of information to and from internal storage by a data channel, the typing of a message on the console typewriter by the operator, etc.

The method utilized by AMOS is similar to that used by Operating System/360, due to its utmost generality and simplicity, and is based on the sole assumption that the termination of an asynchronous process will be signalled by a CPU interrupt. The concept is centered on the Event Control Word (ECW), which may be defined as a location in internal storage (reserved by an application program or subroutine) whose contents reflect the status of a particular asynchronous process. By convention, if the contents of the ECW are zero, then the asynchronous process has not yet terminated (again by convention, a storage location does not become an ECW until the process with which it is associated has been started by the CPU). If the contents are non-zero, then the associated asynchronous process has terminated; furthermore (when applicable), the non-zero contents of the ECW reflect the terminating status of the operation (whether or not the process terminated normally and, if not, why not). The "event" in "Event Control Word" refers to the termination of the asynchronous process.

The location which is to become an ECW is generally zeroed by the application or AMOS program prior to initiation of the process with which the ECW is to be associated, and its address is then passed to the AMOS program which is to initiate the asynchronous process. After the process is started, control is returned to the calling program which may proceed with main line execution, initiate other processes (using different ECW's), etc. When the event of process termination occurs, the CPU is interrupted and the appropriate AMOS interrupt routine then sets the ECW associated with the event to non-zero, and suspended execution is resumed.

Since interrupts are transparent to an application program, no notice is given to the program with the exception that the contents of an ECW have been transformed from zero to non-zero. It is the responsibility of the application program to periodically interrogate ECW's of outstanding events with which the program is concerned.

This interrogation may be performed directly by simply testing the contents of the ECW location for zero. An alternative to this method is to call the AMOS "wait" routine, passing to it the address of the associated ECW. WAIT is a simple subroutine; it continually tests the contents of an ECW, and returns to the calling program only when the contents of the ECW have been changed to non-zero.

ECW's are used throughout the AMOS/1800 system to signal such events as the - to + voltage transfer of external experimentation equipment, analog/digital converter voltage overload, an operator request to enter a message from the console typewriter, etc. All data channel operations between internal storage and external input/output

devices signal their completion by ECW, as well as expired timer intervals, and completion of console message operator replies (see Section 3.2.3).

3.1.3 AMOS Subroutine Hierarchy

The transfer of control from one program to another is accomplished by the CALL statement, whose final form is the 1800 "branch and store instruction counter" (BSI) instruction. This method is used regardless of whether the called subroutine is "resident" or "transient." Resident subroutines are system (AMOS) programs only; the collection of resident programs makes up what has up to now been referred to as AMOS/1800. Transient subroutines are considered by AMOS/1800 to be application-oriented programs loaded from external storage immediately prior to the commencement of application program execution. These subroutines originate from AMOS/360, either from the AMOS/360 subroutine library or from the previous assembly of a source program on the 360 computer (see Section 3.2.7). Figure 2 illustrates the possible control paths within AMOS/1800.

3.2 AMOS/1800 Component Description

3.2.1 Control Supervisor (NUCLS)

The heart of the 1800-resident portion of the AMOS system is NUCLS, the 1800 control supervisor. NUCLS is the only truly conglomerate

program of the system, in that system tables as well as many independent sections of logic are all contained within the one source program.

The most important table in AMOS is the system vector table, which contains pointers to the various resident portions of AMOS/1800 as well as a great deal of other miscellaneous information (see Appendix A). The unit control blocks (one for each input/output device) are a series of tables which describe the various input/output devices attached to the 1800 computer. The unit control blocks are linked to the vector table by the device table, which is nothing more than an array of unit control block addresses indexed by device address.

NUCLS originates at location 0 within the 1800 computer and therefore is responsible for correct initialization of the various hardware-assigned internal storage locations, such as those reserved for interval timers and interrupt branch addresses. Since the 1800 is a nested interrupt computer, a unique section of machine status save/restore logic is provided for each interrupt priority level; these sections are pointed to by the interrupt branch table, which begins at storage location 8.

The logic of each interrupt routine is typically quite simple, since the only interrupts processed by the NUCLS routines are "expected", i.e., they occur as a result of an asynchronous process previously initiated by an AMOS program. Once machine status is saved, the process (usually an input/output device) causing the interrupt is interrogated and its terminating status (in the form of a non-zero 16-bit quantity) is obtained. The address of the Event Control Word associated with the device is then found via the appropriate

Unit Control Block and the 16 bits of status information are stored into the ECW, thus setting it non-zero. Machine status is then restored and suspended processing is resumed.

There are only two devices which can cause "unexpected" interrupts--the adapter and the console typewriter. These "unexpected" interrupts are routed to the appropriate AMOS program for processing; the called program is obliged to eventually return control to the interrupt routine so that machine status can be restored.

As stated earlier, several independent subroutines are contained within NUCLS. Among these is the WAIT routine described in Section 3.1.2.

All interval timer logic is also found within NUCLS, including the time-of-day (TIME) routine. TIME makes available the current local time (in seconds) by monitoring interval timer C, which runs continuously.

The remainder of NUCLS is composed of several frequently-used data conversion routines, including CVTIM (converts a binary integer in seconds to the EBCDIC form HH:MM:SS) and CVHEX (converts binary to hexadecimal EBCDIC form).

3.2.2 Console Typewriter Input/Output Supervisor (CNSUP)

The 1816 console typewriter is the primary method of communication between the 1800 user and his program. This device is completely controlled by CNSUP, due to the extreme difficulty involved with input/output between the 1816 and the 1800.

CNSUP requires that the typewriter always be in one of four logical states:

1) idle

2) reading

3) writing

4) carriage returning

There are five entry points to CNSUP; four are by interrupt and one is by direct call from an AMOS or application program (via CWRIT). A call to CWRIT is made when a program wishes to output a message to the typewriter; the mode is changed from "idle" to "writing" (unless the current mode is other than "idle", in which case CWRIT delays until the mode goes to "idle") and output of the message is initiated.

Interrupts from the typewriter are routed by NUCLS to four possible sections of logic within CNSUP, depending on the mode in effect at the time of the interrupt. If the mode at time of interrupt is "idle", then the cause of the interrupt is the 1800 operator (or user) who is requesting input of a message. Action taken by the program includes transfer of the mode from "idle" to "reading" and the initiation of message input.

The difficulty in utilization of the typewriter is due to the simplicity of the interface between the device and the 1800 CPU; there is no data channel or hardware buffer. As an illustration of this difficulty, consider the steps necessary to obtain characters typed by the operator:

1) CNSUP issues a read command to the 1816 and then exits.

2) Depression of a key by the operator interrupts the 1800; control is routed to CNSUP by NUCLS.

3) CNSUP obtains a binary bit pattern associated with the depressed key;

4) CNSUP translates this pattern to the appropriate EBCDIC character; and also

5) translates the pattern to the "tilt-rotate"[1] character output code;

6) sends the "tilt-rotate" code to the typewriter via a write command[2] (this prints the character associated with the depressed key) and then exits;

7) regains control via the interrupt which shows that the character has been printed; and

8) returns to step 1) for the next input character.

3.2.3 Control Message Handler (CSERV)

The function of CSERV is to process control messages passed to it from either CNSUP or SOJOB (the job scheduler). The syntax of control messages typically consists of a one-character verb followed by a "modifier" field made up of an arbitrary number of characters. One of these messages, the "reply" message, consists of an R,<text> syntax and is the form used for operator/program communication.

A program notifies CSERV that an impending operator-input message is to be expected by calling REPLY (an entry point within CSERV), passing as parameters two addresses; the first address points

to an input buffer area and the second to an Event Control Word. Action taken by REPLY basically consists of saving the two addresses. When CSERV is passed an R,<text> message by CNSUP, the <text> field is moved to the input buffer area and the ECW is set to non-zero. The application program requesting the "reply" message is thus informed of its presence by examination of the ECW.

Other control messages processed by CSERV are described in detail in section 6 of the "AMOS User's Guide, Version 2".[3]

3.2.4 Input/Output Supervisor (IOSUP)

It is the function of IOSUP to initiate data transfer between internal storage and all external input/output devices attached to the 1800 computer, with the sole exception of the 1816 console typewriter. IOSUP is invoked by direct call from an AMOS or application program; arguments passed to IOSUP include a device address, an Event Control Word address, and an Input/Output Control Command (IOCC)[4] to be executed.

Prior to executing the IOCC, IOSUP performs several tests related to the specified device in order to assure successful initiation of the operation. In addition, the specified ECW address is placed by IOSUP into the appropriate Unit Control Block and the ECW itself is zeroed.

Only after all tests have been satisfied does IOSUP execute the IOCC. Return is then made to the calling program, so that effective overlapping of CPU and input/output processes can be realized.

3.2.5 Program Loader (LOADR)

It is the responsibility of the LOADR program to transform application programs and transient subroutines into executable code within 1800 internal storage. Input to the "loader" is in the form of intermediate object records (output by the 360-1800 assembler described in Section 3.4) obtained across logical file 4 of the channel-to-channel adapter. Since the final 1800 internal storage origin of an application program or subroutine is unknown at assembly time, all programs processed by LOADR are expected to be of the relocatable format; "absolute" programs are not allowed.

The calling program is required to provide the loader with an origin address, which is normally the first location in the transient area of internal storage. The "main" program is then fetched (from adapter file 4) and loaded beginning at the specified location. When input data from logical file 4 signals the end of a program, all CALL statements are resolved by scanning a table of external entry points which is constructed during the loading process. This table is built beginning at the end of the transient area and expands toward the beginning of the transient area as more programs are loaded. See Figure 3.

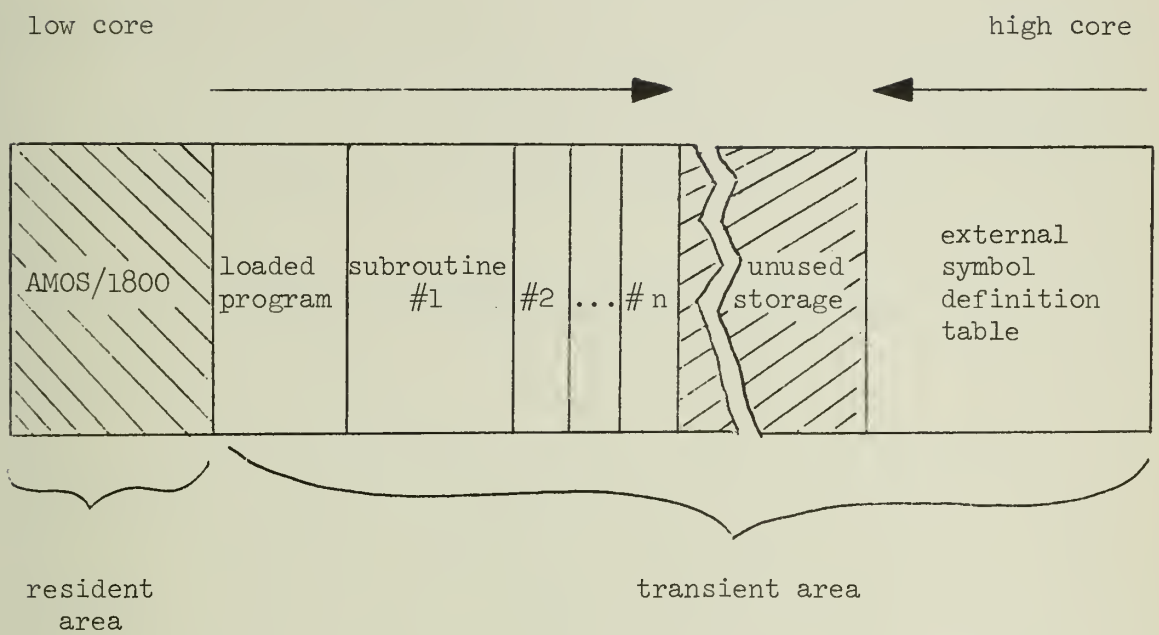


Figure 3.

Loaded programs are allowed to overlay the external symbol definition table only if the overlay is done in a non-destructive manner, i.e., those portions of the program(s) within the overlay may not be initialized with instructions or data.[5] This restriction only occurs with the very largest of programs and may otherwise be ignored.

When file 4 is exhausted, LOADR scans the external symbol table for CALL's to undefined (unloaded) programs. If such a reference is found, the name of the undefined program is sent to AMOS/360. If AMOS/360 can locate the requested program,[6] it is expected to honor further file 4 requests by LOADR with object records of the desired program.

When the program and all necessary subroutines have been loaded, LOADR transfers control to the program. The calling sequence is disguised so that return from the loaded program is to the program which invoked LOADR, rather than to LOADR itself.

3.2.6 1800 Job Scheduler (SOJOB/EOJOB)

The interpretation of control cards and the execution scheduling of application programs is the responsibility of the job scheduler SOJOB/EOJOB. SOJOB gains control when processing of a job has been requested by the 360 control program (AMOS/360). Logic consists of initializing various job-oriented flags and reading the system input stream (SYSIN - adapter logical file 1).

Control cards for SOJOB are denoted by the presence of $$ in card columns 1 and 2; cards of this type are read until a non-$$ card (an error condition) or a $$EXC card is detected. Other control cards accepted by SOJOB include $$JOB, $$(null), and $$<control statement>, where <control statement> is defined as a string of characters acceptable as console typewriter input (to CSERV; see Section 3.2.3). In other words, if V,04,ØN is an acceptable console typewriter input message, then $$V,04,ØN is a valid scheduler control card. Such statements are processed via CSERV just as if they originated from the console typewriter.

In the current version of the scheduler, a definite precedence relationship exists between the various control types. This precedence is:

<control statement> ⊃ $$JØB ⊃ $$EXC ⊃ $$ or $$EXC, etc.

where ⊃ means "must precede".

Detection of $$EXC,x causes SOJOB to relinquish control to the application program x, where x is either * or a symbolic program name. In the latter case, SOJOB requests AMOS/360 to position logical file 4 of the adapter to program x^7 before control is passed to LOADR for loading and execution of program x. * means that the program to be executed is already positioned on file 4 (previously assembled on the 360 computer) and that positioning is not necessary.

Direct (and therefore system error-free) return of the application program to SOJOB implies that further scanning for $$EXC cards is to be performed, so SOJOB takes an internal branch to the $$EXC logic described earlier.

The EOJOB portion of the scheduler is invoked at termination of application program processing. The nature of the return process (whether normal or abnormal) dictates whether or not a post-mortem dump is necessary; if so, EOJOB transfers to the DUMP program.

Other functions performed by EOJOB include the resetting of various hardware conditions and the logical deallocation of all peripheral devices which were used by the completed job. When all housekeeping has been completed, EOJOB informs AMOS/360 that the current job has terminated by placing a unique command into the adapter and then waiting for the AMOS/360 response to the command.

The response may or may not be forthcoming shortly; the clearing of the adapter by the 360 signifies initiation of a new job by AMOS/360. If no jobs are currently available for 1800 processing,

the 1800 "end of job" command is left "dangling" in the adapter controls. Since command initiation was by the 1800, the 360 selector channel to which the adapter is attached remains in a free and usable status. Eventually, receipt of the 360 response by EOJOB results in a transfer of control from EOJOB to SOJOB.

3.2.7 Adapter File Control (CAFIL)

CAFIL is a collection of routines which serialize and simplify data transfer across the most-used files of the channel-to-channel adapter. Each subroutine has a unique entry point (such as SYSIN, SYSMS, SYSPR, SYSPU, etc.) and is responsible for data transfer across the appropriate logical file. CAFIL itself controls files 0-3 only; the remaining files are utilized via a transient subroutine, although the file control logic discussed below is used by both.

The requested logical file is communicated to the 360 by use of the 1800 I/O command modifier bits. Four modifier bits are used for this purpose, which allows file addresses of from 0 to 15. The remaining four modifier bits are used in conjunction with the basic type of the 1800 command (either "read" or "write") to indicate the type of operation to be performed on the logical file by AMOS/360. A maximum of 32 commands can be defined in this manner.

For example, the 1800 can request AMOS/360 to "rewind" logical file 8 by using the appropriate modifier bit pattern with a "read" or "write" command, so long as the 360 uses the same scheme of bit patterns (strictly a programming convention) and clears the "read" or

"write" command from the adapter.

This is all possible because of the design of the adapter, in that the command and modifier bits used by the interrupting (initiating) computer are available to the interrupted (terminating) computer.

A more thorough treatise on the adapter may be found in the 1800 Functional Characteristics manual[8]; the modifier bit convention used by AMOS is found in Appendix A, Table 1.

3.2.8 Initialization Program (INIT)

The INIT program gains control when the AMOS/1800 system has been successfully "bootstrapped" into 1800 internal storage. Its purpose is to perform all functions necessary to prepare AMOS for initial job processing. Upon completion of these tasks, INIT transfers control to SOJOB; the internal storage within which INIT resides is then made available for use by application programs. See Figure 4.

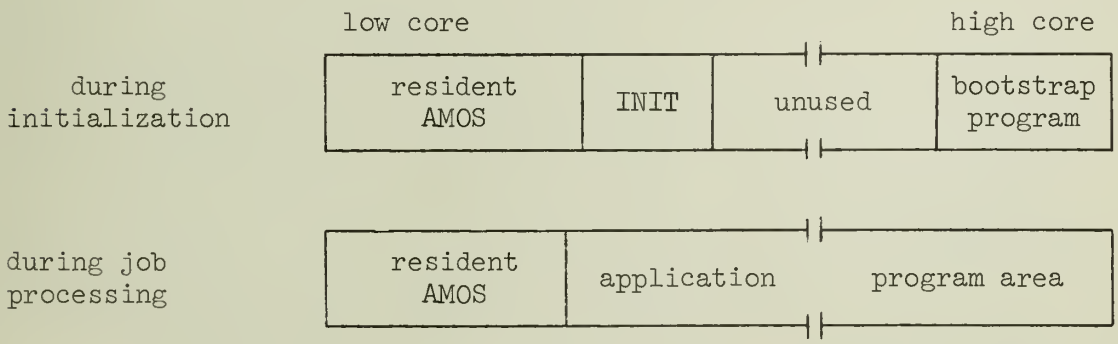


Figure 4.

The functions of INIT are as follows:

1) initialize various pointers in the System Vector Table;
2) read console switches for special AMOS options (if any);
3) compute partition (application program area) starting address;
4) protect resident AMOS internal storage locations;
5) initialize interval timer C for time-of-day purposes;
6) inform operator of successful initialization; and
7) transfer control to SOJOB.

3.2.9 Other AMOS/1800 Components

There are several additional programs which form an integral part of AMOS/1800; their more straightforward nature suggests that discussion of them can be done in a brief manner.

The INREQ program generates "intervention required" and "I/O error" messages for the various peripheral devices. It is generally invoked by IOSUP.

The DUMP program takes full core, partial core (with the boundaries specified by argument), or panel (registers and status) dumps by request. Return is to the calling program; thus DUMP can be used by application programs for "snapshot" purposes.

CASUP is the adapter interrupt handler; it gains control via NUCLS whenever an adapter interrupt occurs. If the interrupt was due to a 360-initiated command, the modifier bits presented by the 360 are examined and the appropriate action (currently, cancellation of the

1800 job) is taken. Otherwise, the interrupt is treated in much the same manner as any 1800 device terminating interrupt.

3.3 The System/360 Control Program

The routing of data between the various 360 facilities and the 1800 computer is handled by the System/360 control program (AMOS/360). The program currently used at the University of Illinois executes under control of the Attached Support Processor (ASP) system, which makes discussion of the program extremely difficult without assuming extensive knowledge of ASP on the reader's part. In view of this problem, and to emphasize the generality of AMOS/360, this section will deal primarily with the requirements and interfaces which must be met by an AMOS/360 program.

3.3.1 Channel-to-Channel Adapter Programming

Of primary concern to an AMOS/360 program is the 1800-360 channel-to-channel adapter as it appears to the System/360 computer. Unfortunately, the adapter does not quite conform to general System/360 device standards; the conflict occurs in the instance when the 360 issues a read (write) command to the adapter without realizing that the 1800 has previously issued a read (write) command. This clash of uncomplementary commands is unavoidable if both CPU's are allowed to initiate data transfer. The normal reaction of 360 hardware to such an occurrence would be to signal "command reject", informing the 360

program that a complementary command must be issued since the 1800 was at the adapter first. The actual signal given is "busy", however, which is misleading since no data transfer is taking place. The deceived program is the input/output supervisor of Operating System/360; "busy" to it is a temporary condition which will eventually disappear, and it therefore keeps trying the original 360 command (with no success).

Another less serious problem is the fact that the adapter can "independently" interrupt the 360 to signal the presence of an 1800 command. This signal must be passed to AMOS/360 so that the proper response can be given.

Both of these problems are solved by rather simple modifications to the OS/360 input/output and interrupt supervisors. Another less elegant method is to completely bypass both of these programs as far as control of the adapter is concerned.

3.3.2 Adapter Response Requirements

AMOS/360 is a passive program, in that the majority of its work is done in response to 1800 requests via the adapter. Typical logic consists of remaining in the wait state until the occurrence of an adapter interrupt. Upon notification of the interrupt, AMOS/360 senses the status of the adapter. The sense information obtained is the low-order 16 bits of the command which the 1800 issued to the adapter (see Appendix A, Table 1). From this data AMOS/360 can ascertain the "logical command", the "physical command" (read or write), and the "logical file" as specified by the 1800. All that remains is

for AMOS/360 to take the appropriate action, being sure to clear the adapter in the process.

For example, suppose that a program executing on the 1800 wishes to read the next card image from the input stream (SYSIN). This is accomplished by a call to the AMOS/1800 routine SYSIN, which in turn calls IOSUP, requesting that the following command be issued to the adapter:

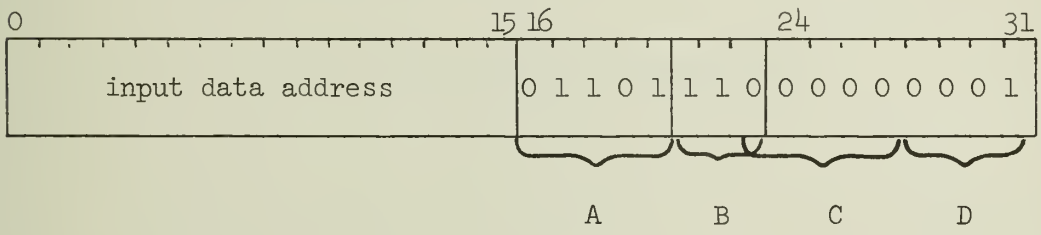


Figure 5.

The various command fields are described as follows:

Field A — 01101 is the 1800 address of the adapter (not needed by AMOS/360)

Field B — 110 is a "read" command (360 → 1800). This is the physical command.

Field C — 00000 is the logical command; as described in Appendix A, 00000 means only "read".

Field D — 0001 is the logical file address. By convention, SYSIN is assigned to logical file 1.

The resulting 360 interrupt notifies AMOS/360 of the request. The proper response is to first sense the adapter status, which results in the receipt of bits 16-31 by AMOS/360. The next step is to isolate the file number; if file 1 is logically attached to a 360 card reader, then AMOS/360 reads the next card from the reader. The data obtained is sent to the adapter with a 360 write command (complementing the 1800's read command), which initiates the data transfer and eventually clears the adapter.

3.3.3 Program Data Requirements

In order to process the various 1800 requests, AMOS/360 must have access to various forms of data, including the AMOS/1800 supervisor itself, which must be sent across the adapter during the 1800 restart ("bootstrap") process. In addition, a library of AMOS subroutines must be readily accessible. These data are generally placed on System/360 direct-access storage; as a result, it is possible to modify and/or update the AMOS system without accessing the 1800 computer at all.

3.3.4 Other Logic Provisions

In addition to processing 1800 requests, AMOS/360 may control 1800 device allocation, cancel jobs which are executing in an abnormal manner, and monitor various accounting data. Other programs in the 360

system may be responsible for allocating the 360 tapes used by the 1800 and for scheduling the order in which 1800 jobs are to be processed.

3.4 The 1800-360 Assembler

The availability of an 1800 assembler which executes on a System/360 computer affected some of the design decisions described earlier. Generally known by the acronym MASM, it was recently made available on a general basis as IBM Type III Program 360D-03.1.013.[9]

Although the assembler permits a superset of the statement repertoire of its stand-alone 1800 counterpart, several additions were made in order to adapt the assembler to AMOS requirements. The most important change involved the routing of 1800 object output to devices other than a 360 card punch (specifically, to 360 direct-access or tape storage). Other changes included the insertion of a primitive, non-parameterized macro facility, as well as several new pseudo-operations.

Footnotes

[1] In reference to the typing "ball" found in IBM Selectric typewriters.

[2] The 1816 hardware does not automatically assume responsibility for printing the character typed by the operator.

[3] "Attached Machine Operating System User's Guide, Version 2," Department of Computer Science, University of Illinois, Urbana, Illinois, 1968.

[4] IOCC's are the commands which initiate input/output between a device and internal storage.

[5] Only BSS and BES statements are allowed in this case.

[6] In a subroutine library located on System/360 disk storage.

[7] See Sections 3.2.7 and 3.3.

[8] "IBM 1800 Functional Characteristics," IBM Systems Reference Library Form No. A26-5918-6, International Business Machines Corporation, San Jose, California, 1968.

[9] More information may be obtained from:

Program Information Department
IBM Corporation
Poughkeepsie, New York

4. DEVELOPMENT PLANS

The AMOS system was essentially completed in December of 1968. Work continues in several areas, however, in order to expand the capabilities of the system and make it more useful. This section briefly outlines some of the current and proposed projects.

4.1 Analog/Digital Supervisor

Work is progressing on the development of an Analog-to-Digital Supervisor, which will execute under control of AMOS and exercise the full capabilities of the analog-to-digital converter and associated equipment. The program will be made up of two phases. The interpreter phase will accept and decode input parameters (or options) and the executor will perform the actual conversion, under user or program control. Each phase may be re-entered an indefinite number of times.

4.2 Assembler Macro Facility

A macro processor is currently being written for the assembler described in Section 3.4. The design of the macro processor is similar to the facility recently incorporated into the IBM 1800 TSX assembler (see Bibliography).

4.3 Transient Scheduler (SOJOB/EOJOB)

In order to provide more flexible job processing capability, consideration has been given to an expansion of the SOJOB/EOJOB program. In order to minimize internal storage requirements, the improved program would be a transient one (it would "roll in" and "roll out" of the application program area), as opposed to the current version, which is always resident.

4.4 Compiler Possibilities

Preliminary development work is being done on an 1800 FORTRAN subset compiler, which (like the 360-1800 assembler) will execute on the System/360 computer. Due to the availability of the System/360 for data reduction purposes, it is expected that floating-point logic (for example) will not be integrated into early versions of the compiler.

4.5 Multiprogramming

Another advantage of the Event Control Word concept is the relative ease in which it allows the introduction of multiprogramming into a system. For example, a lower priority job "y" (or background function) can access the CPU whenever the higher priority job "x" calls the WAIT routine; this action implies that x cannot utilize the CPU until the completion of the asynchronous process specified by the ECW argument, and therefore y may use the CPU until x's process has terminated.

"Background processing" (the simplest form of multiprogramming) is a definite possibility for a future version of AMOS. The method which will probably be chosen utilizes one of the lower-priority hardware interrupt levels which is currently unused.

4.6 1800/360 Parallel Computation

In order to allow System/360 "real time" analysis of data while execution of the corresponding 1800 job is continuing, an AMOS/360 supervisor which executes under control of Operating System/360 is currently being developed. 1800 jobs run under control of such a program will be able to schedule execution of programs on the 360 in a dynamic manner, with full communication and synchronization of the two computers controlled by AMOS/360 and AMOS/1800.

BIBLIOGRAPHY

"Attached Machine Operating System User's Guide, Version 2," Department of Computer Science, University of Illinois, Urbana, Illinois, 1968.

Fitsos, G. P. "Macro Assembly Program for the IBM 1800 TSX II System," International Business Machines Corporation, San Jose, California, 1967.

Gillette, W. L., Jr. "IBM Selector Channel - Principles of Operation," IBM Form No. L26-2034-2, International Business Machines Corporation, San Jose, California, 1968.

"IBM 1800 Assembler Language," IBM Systems Reference Library Form No. C26-5882-3, International Business Machines Corporation, San Jose, California, 1966.

"IBM 1800 Functional Characteristics," IBM Systems Reference Library Form No. A26-5918-6, International Business Machines Corporation, San Jose, California, 1968.

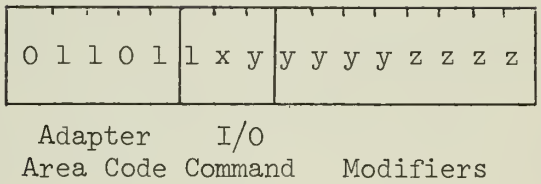
Wells, R. A., Carter, C. E., and Friedman, H. G. "ILLINET Analog Facilities," Department of Computer Science, University of Illinois, Urbana, Illinois, Report No. 280, 1968.

APPENDIX A

SYSTEM TABLES

Table 1. Adapter Command Bit Assignments

2nd word of 1800
I/O control command



The "I/O command" field (lxy) is either 101 (initialize write) or 110 (initialize read). The five y bits determine the logical command to be executed by AMOS/360 as well as the direction of any real or dummy data transfer. The four z bits address the file to be operated on (from 0 to 15).

The five y bits determine a number from 0 to 31; each number has assigned to it a logical command according to the following table. Note that commands 0-15 are read commands and each must therefore be answered by a 360 write command, whether or not meaningful data is to be transferred. The complementary condition exists for commands 16-31.

0	read (transfer data from 360 to 1800)
1*	read backward
2*	point (360 repositions file to previously "noted" point.)
3*	backspace record
4*	backspace file
5	rewind file
6	request 1800 initial program load (360 sends IPL data on subsequent reads.)

7	request current time (360 sends current time of day.)
8-15	currently, no operation
16	write (transfer data from 1800 to 360)
17*	note (360 sends 1800 reference to current file position.)
18	find (Pass 360 a name x; 360 honors subsequent file 4 reads with object data of program x.)
19*	forward space record
20*	forward space file
21*	write tape mark
22	query (Inform AMOS/360 of 1800 job termination; clearing of this command by AMOS/360 signifies request to process new job.)
23-31	currently, no operation

*currently not supported by AMOS/360

Table 2. System Vector Table (VECT)

The System Vector Table is always located beginning at internal storage location 23_{10}. Following the table, some of the more important non-address locations are discussed in more detail.

Location (dec/hex)		Symbolic Name	Contents
0023	0017	VIOSU	address of program IOSUP
0024	0018	VSOJO	address of program SOJOB
0025	0019	VEOJO	address of program EOJOB
0026	001A	VLOAD	address of program LOADR
0027	001B	VDUMP	address of program DUMP
0028	001C	VINRE	address of program INREQ
0029	001D	VIOER	address of program IOERR
0030	001E	VSYSI	address of program SYSIN
0031	001F	VSYSO	address of program SYSPR
0032	0020	VSYSP	address of program SYSPU
0033	0021	VSYSB	address of program SYSPB
0034	0022	VSYSM	address of program SYSMS
0035	0023	VCWRI	address of program CWRIT
0036	0024	VREPL	address of program REPLY
0037	0025	VKBTB	address of keyboard translate table
0038	0026	VMESS	address of console typewriter input message
0039	0027	VWAIT	address of WAIT routine
0040	0028	VTIME	address of TIME (of day) routine

Table 2.--Continued

Location (dec/hex)		Symbolic Name	Contents
0041	0029		not used
0042-3	002A-B	VTBAS	time base (used by TIME)
0044-5	002C-D	VCNCL	IOCC to cancel a job
0046-7	002E-F	VENAB	IOCC to enable all interrupts
0048-9	0030-1	VDISB	IOCC to disable all interrupts
0050	0032	VVTIM	address of routine CVTIM
0051	0033	VCONS	address of typewriter UCB
0052	0034	VCNIL	address of typewriter interrupt level
0053	0035	VMXDV	maximum number of I/O devices on 1800
0054	0036	VCEW	Channel Event Word
0055	0037	VDEVT	address of the Device Table
0056	0038	VCODE	job termination code
0057	0039	VERAD	address of program error
0058	003A	VFLAG	system job flags
0059	003B	VKEYS	panel key status
0060	003C	VDATA	panel data switch status
0061	003D	VPSTR	partition starting address
0062	003E	VPEND	partition ending address
0063-4	003F-40	VTECW	ECW's for timers A and B
0065	0041	VPECW	Process Interrupt ECW
0066	0042	VCECW	Analog Comparator ECW

VCEW, the channel event word, is used by IOSUP to indicate the status (whether active or inactive) of the various data channels attached to the 1800. If bit n of VCEW is on, then data channel n is active. Bit 0 is not used; its status of zero is typical of the non-data channel devices (see Table 4, symbol UCHAN).

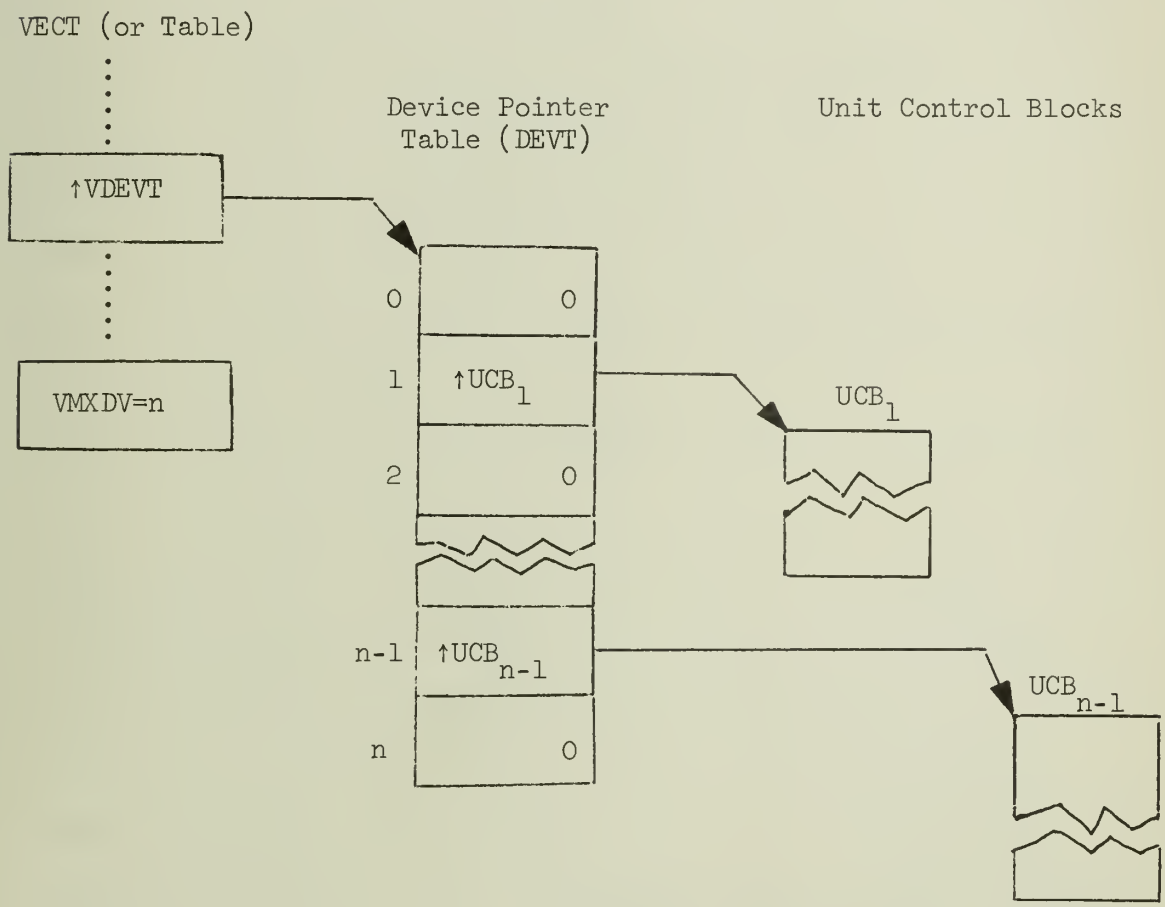
VCODE and VERAD are used to indicate the type and general location of errors encountered during job processing by AMOS. Exceptional conditions, when detected by AMOS program, result in the setting of VCODE to a unique positive number (the error code), and VERAD to the address where the error occurred (if available); control is then passed to EOJOB.

VFLAG is used by the job scheduler to indicate various options and requests made by the application program.

VKEYS and VDATA contain the status of the 1800 panel keys and data entry switches at the time of the last panel interrupt. They serve no special purpose for AMOS.

Table 3. Vector/Unit Control Block Interface

The following diagram illustrates how, given a device address, the corresponding Unit Control Block is located. For example, given an address m: if C(VDEVT)*+m contain 0, there is no such device (likewise if m > n). If C[C(VDEVT)+m] = y ≠ 0, then y is the address of the Unit Control Block for device x.



*C(x) means "contents of location x"

Table 4. Unit Control Block Format

There is one Unit Control Block (UCB) for each device attached to the 1800 system; a detailed description of the major entries is found following the table.

Displacement (dec/hex)		Symbolic Name	Definition
0000-1	0000-1	UIDSW	IOCC to sense the Device Status Word (DSW)
0002	0002	UDSW	Last DSW for this device
0003	0003	UCHAN	Data channel mask
0004	0004	UFLAG	Flag bits
0005	0005	UBUSY	"Device busy" DSW mask
0006	0006	UECW	Address of last ECW used for this device
0007	0007	UCMDA	Address of last IOCC used for this device
0008	0008	UCODE	Device area code
0009	0009	UNAME	Device name (in hexadecimal)
0010-1	000A-B	URDSW	IOCC to sense and reset the DSW
0012	000C	UDVND	Device termination mask
0013	000D	UIGNR	DSW bits which may be ignored
0014*	000E	UCYL	Current cylinder (if 2310 disk)
0015-7*	000F-11	UVOL	Currently mounted volume (if volume mountable device - disk or tape)

*optional

If the associated device is attached to data channel n, then bit n of UCHAN is 1. IOSUP verifies "channel ready" by ANDing UCHAN with VCEW, with a result of zero meaning that the associated data channel is not active. UCHAN is zero for non-channel devices, hence the test cannot fail for them.

UBUSY is a mask which isolates those bit(s) in the device's DSW (Device Status Word) which indicate the busy status of the device (used by IOSUP).

UDVND (another mask) is used by the NUCLS "POST" routine to determine whether or not a device interrupt was due to the termination of data channel input/output operation. If so, the terminating DSW is stored into the associated ECW (via UECW), thus setting it non-zero.

Table 5. Object Card Image Formats

This collection of tables describes the format of the object output of the 360-1800 assembler outlined in Section 3.4. Each "deck" produced by the assembler begins with a "TSX" record, followed by an arbitrary number of "text" records, and finally an "end" record. Each record is 60 words long; the corresponding punched card format is therefore in column binary, one record per card.*

Record Type	Word Displacement	Contents
TSX	5	number of external entry point definitions ($1 \leq n \leq 10$)
	9-10	symbolic name of entry point 1 (5 x 6 form)
	11	relative address of entry point 1
	12-38	same for entry points 2-10
text	0	relative origin of this text data
	2	number of text words in this record
	3-8	relocation bits for text words in this record
	9-n	text content of this record
end	3	relative entry point for program execution

*60 words x 16 bits per word = 960 bits
80 columns x 12 punches per column = 960 punches

There are two relocation bits for each word of text. 00 means that the corresponding text word is to be loaded without modification, 01 means the word contains an address which should be relocated, and 11 indicates that the next two words make up a "call" to an external entry point. The content of the two words is the symbolic name to be called in 5 x 6 form. These words are replaced (by LOADR) with a long-form BSI instruction when the address of the desired reference is available.

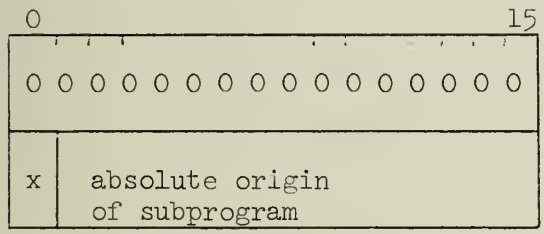
The "5 x 6" character format is detailed in this Appendix, Table 6.

Table 6. External Symbol Table Formats

The External Symbol Table is built by LOADR during the process of loading in an application program (see Section 3.2.5). The table is composed of a serial string of "origin", "definition", and "reference" elements.

One origin element is present for each separately assembled subprogram (main program or subroutine); "definition" and "reference" elements relate to external symbol definitions (ENT statements) and references (CALL statements), respectively.

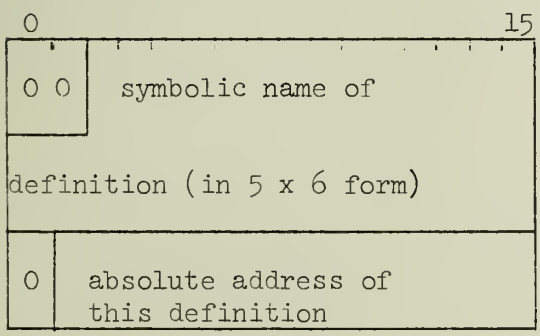
Origin Element



word 1 = 0

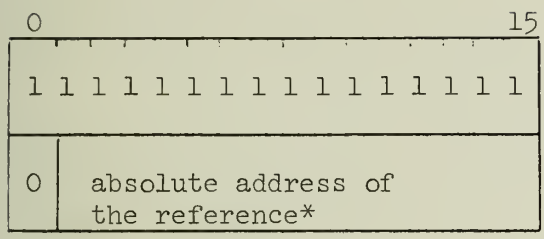
x = 1 means program was loaded so as to resolve a CALL reference

Definition Element



word 1 > 0

Reference Element



Symbols are limited to five characters. The 5 x 6 code name arises from "a maximum of five characters with six bits allocated for each character". Thus, a name in 5 x 6 form occupies the low order 30 bits of an 1800 doubleword; by convention, the high-order two bits are zero.

The code used for character representation is <u>not</u> BCD, but rather a truncated form of EBCDIC. This is possible because 11xxxxxx is the 8-bit EBCDIC form for all numerals and upper case alphabetics. In 5 x 6 code, the high-order '11' is not used.

---

*which initially contains the 5 x 6 form of the reference name

Table 7. AMOS Storage and Entry Point Map

The following table outlines the various programs which make up the resident portion of AMOS/1800; the approximate size of each program is given along with the various entry points (subprograms) found within each.

Program	Size (dec)	Entries (callable)
NUCLS	900	CVTIM, CVHEX, WAIT, TIME
CASUP	100	CASUP
CNSUP	350	CWRIT
CSERV	200	CSERV, REPLY
INREQ	100	INREQ, IOERR
IOSUP	100	IOSUP
KBTAB	125	—
LOADR	425	LOADR
CAFIL	375	SYSIN, SYSPR, SYSMS, SYSPU, SYSPB
DUMP	425	DUMP
SOJOB	325	SOJOB, EOJOB
	3425	

APPENDIX B

AMOS/1800 PROGRAM LOGIC FLOW DIAGRAMS

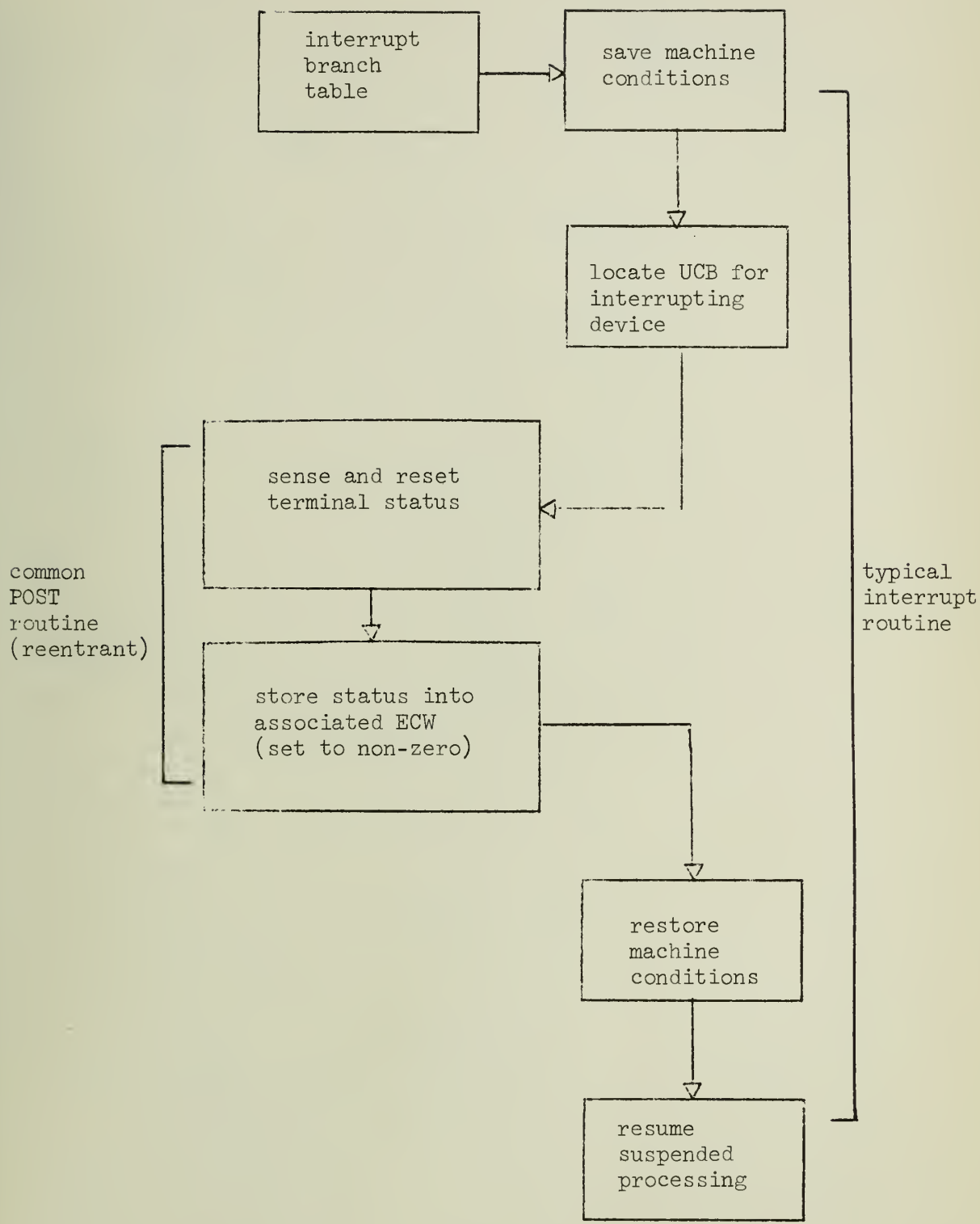


Figure B-1.

NUCLS Interrupt Logic

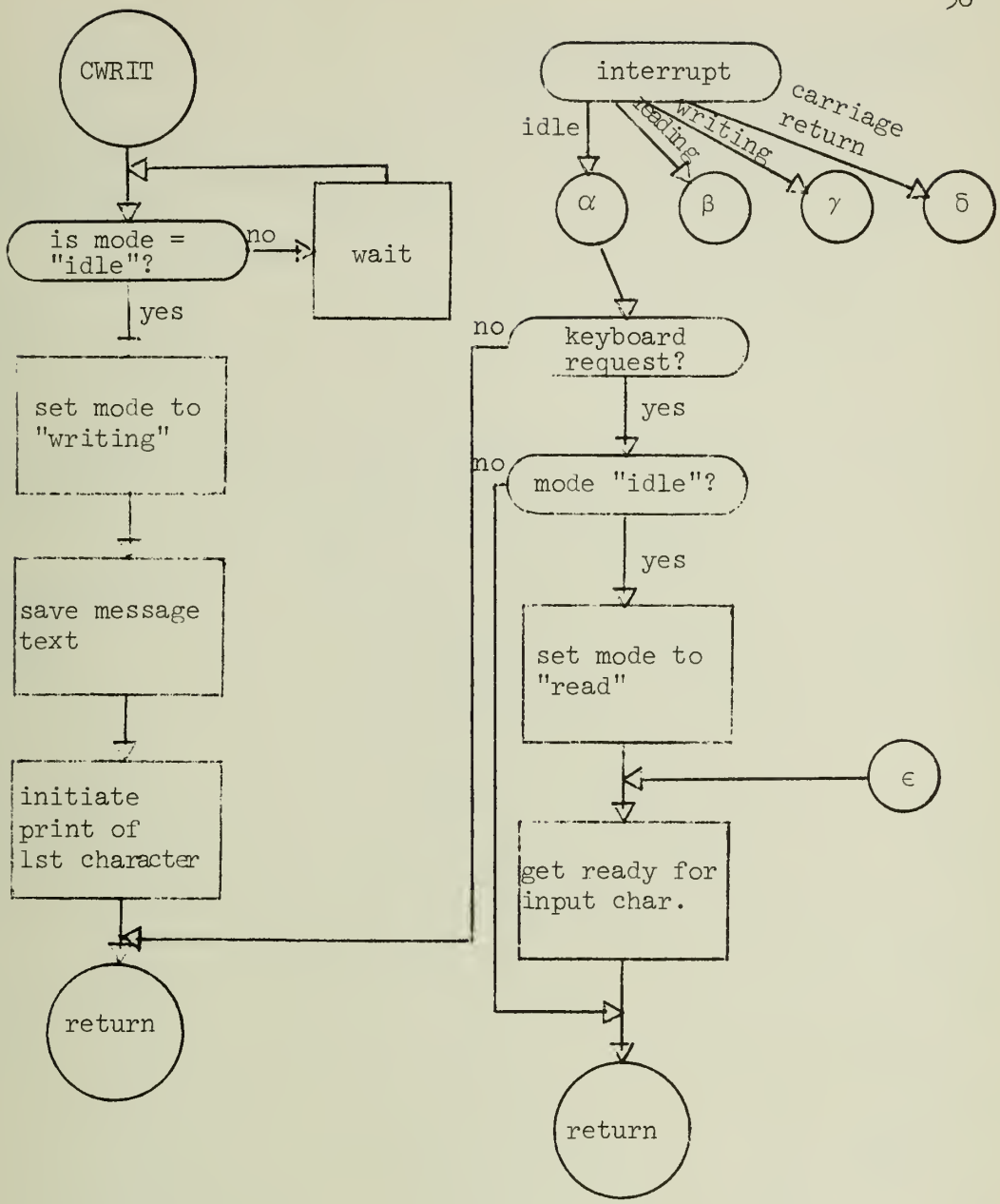


Figure B-2.

Console Typewriter Input/Output Supervisor (CNSUP)

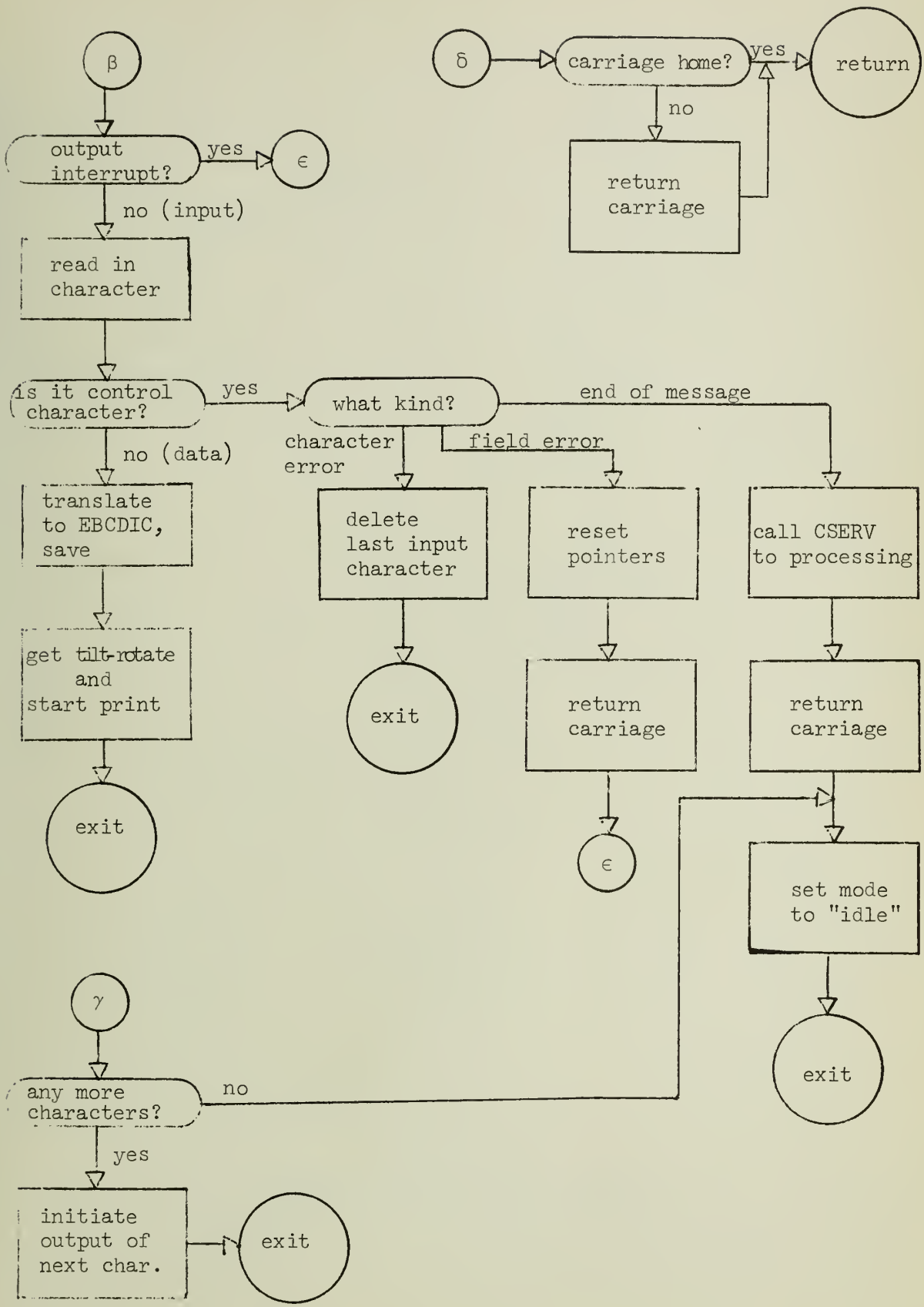


Figure B-2.--continued

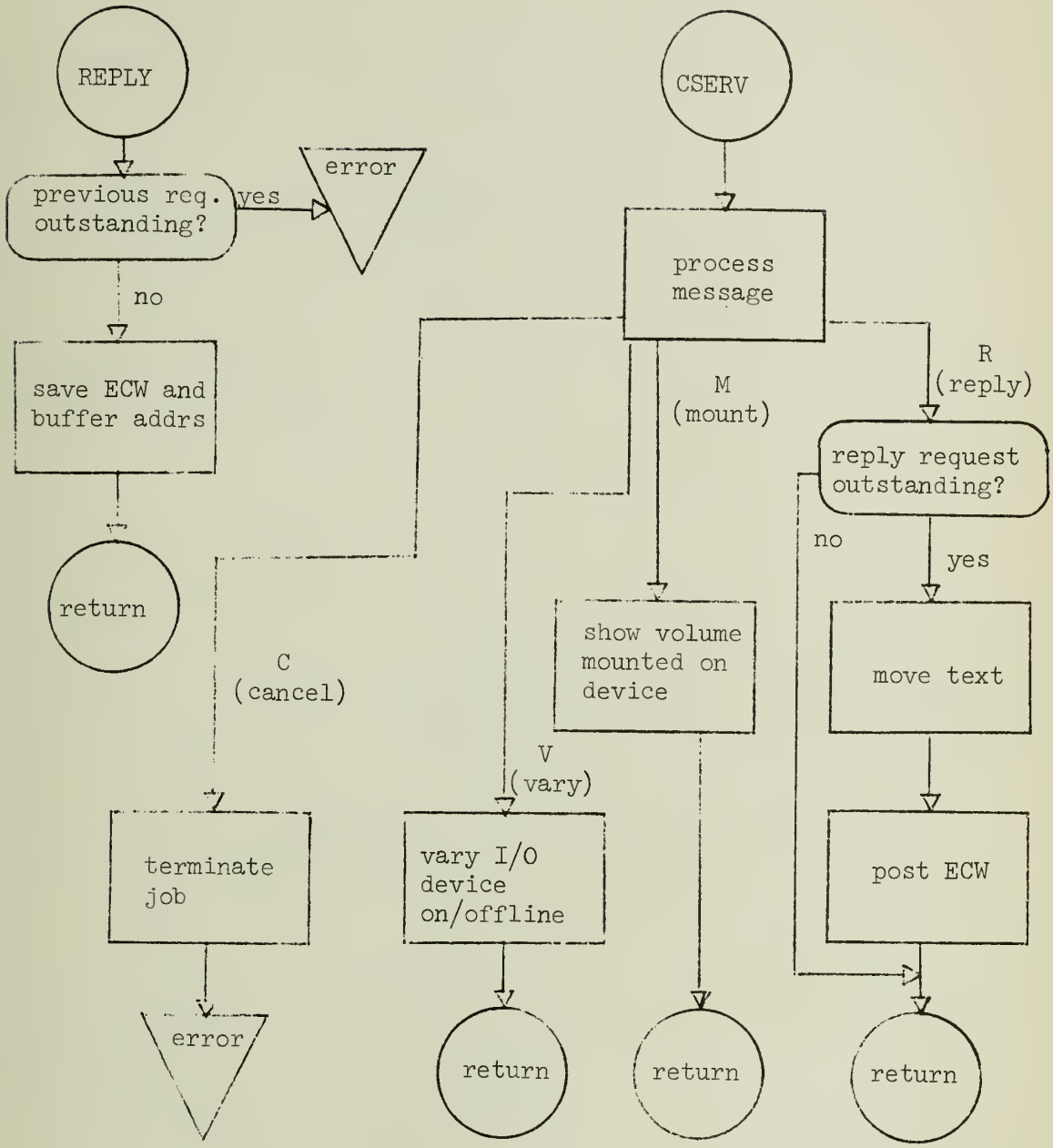


Figure B-3.

Console Message Handler (CSERV)

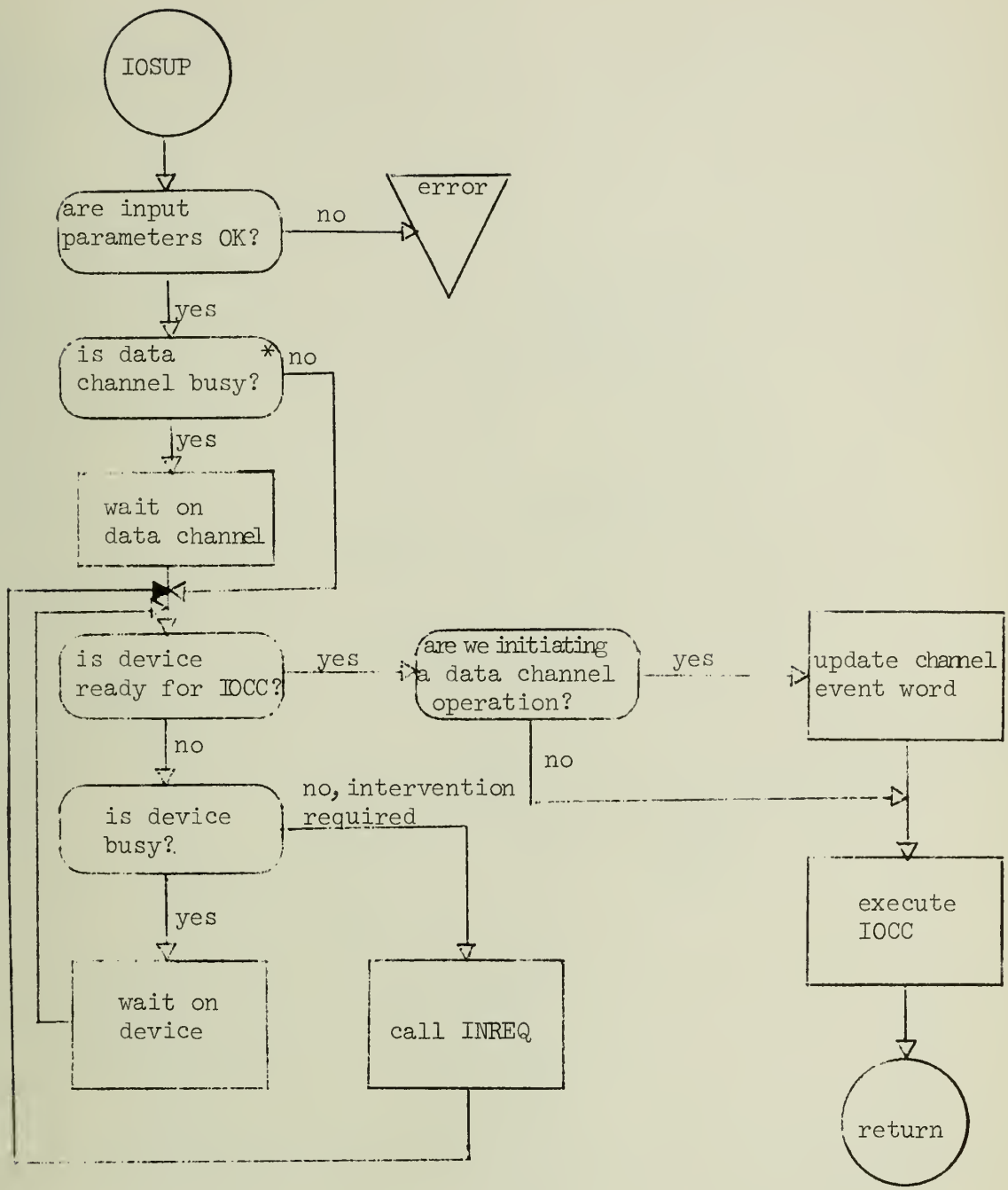


Figure B-4.

Input/Output Supervisor (IOSUP)

*devices not attached to a data channel automatically pass this test.

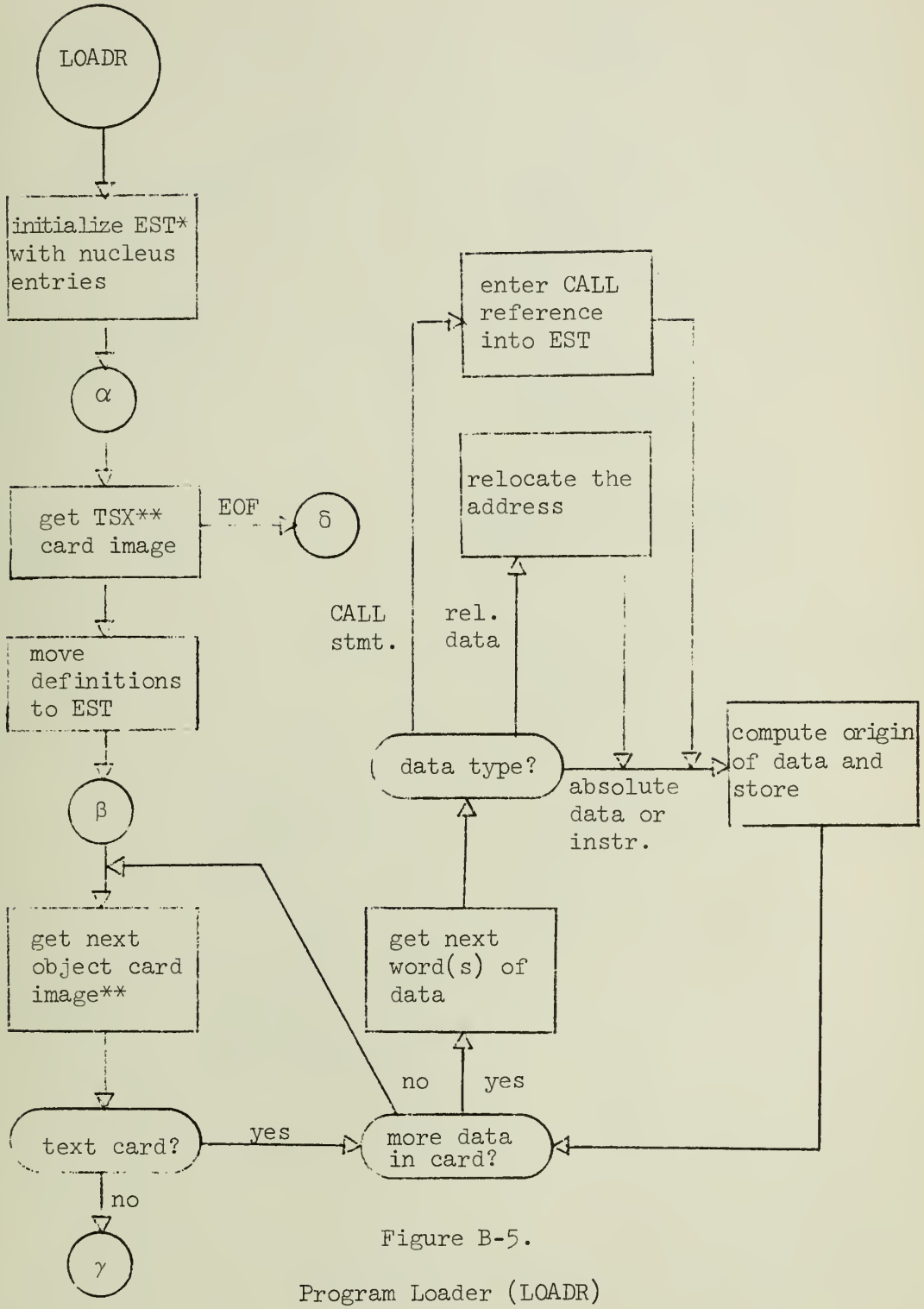


Figure B-5.

Program Loader (LOADR)

---

*External Symbol Table: see Appendix A and Section 3.2.5

**from logical file 4 of the adapter

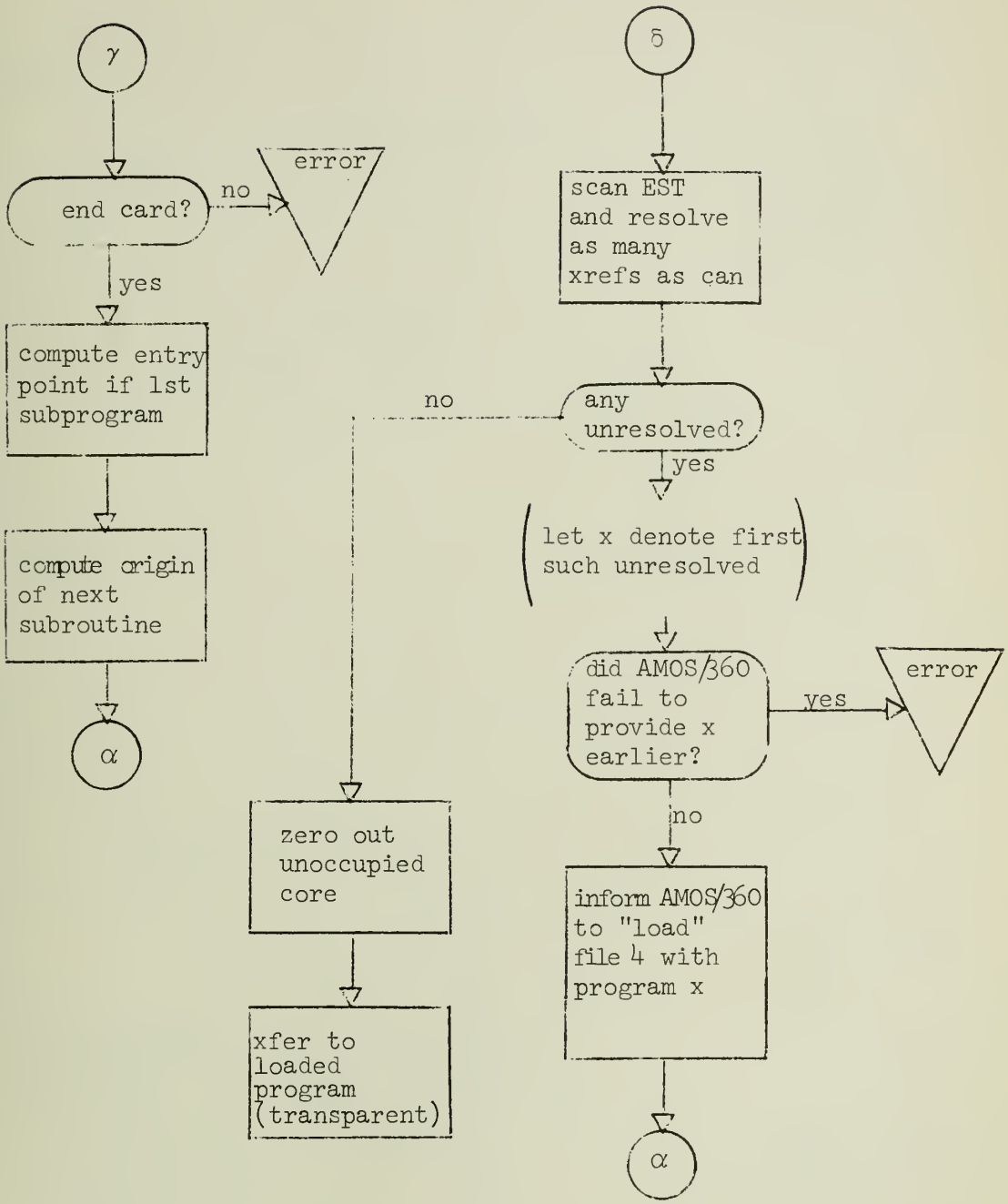


Figure B-5.--continued

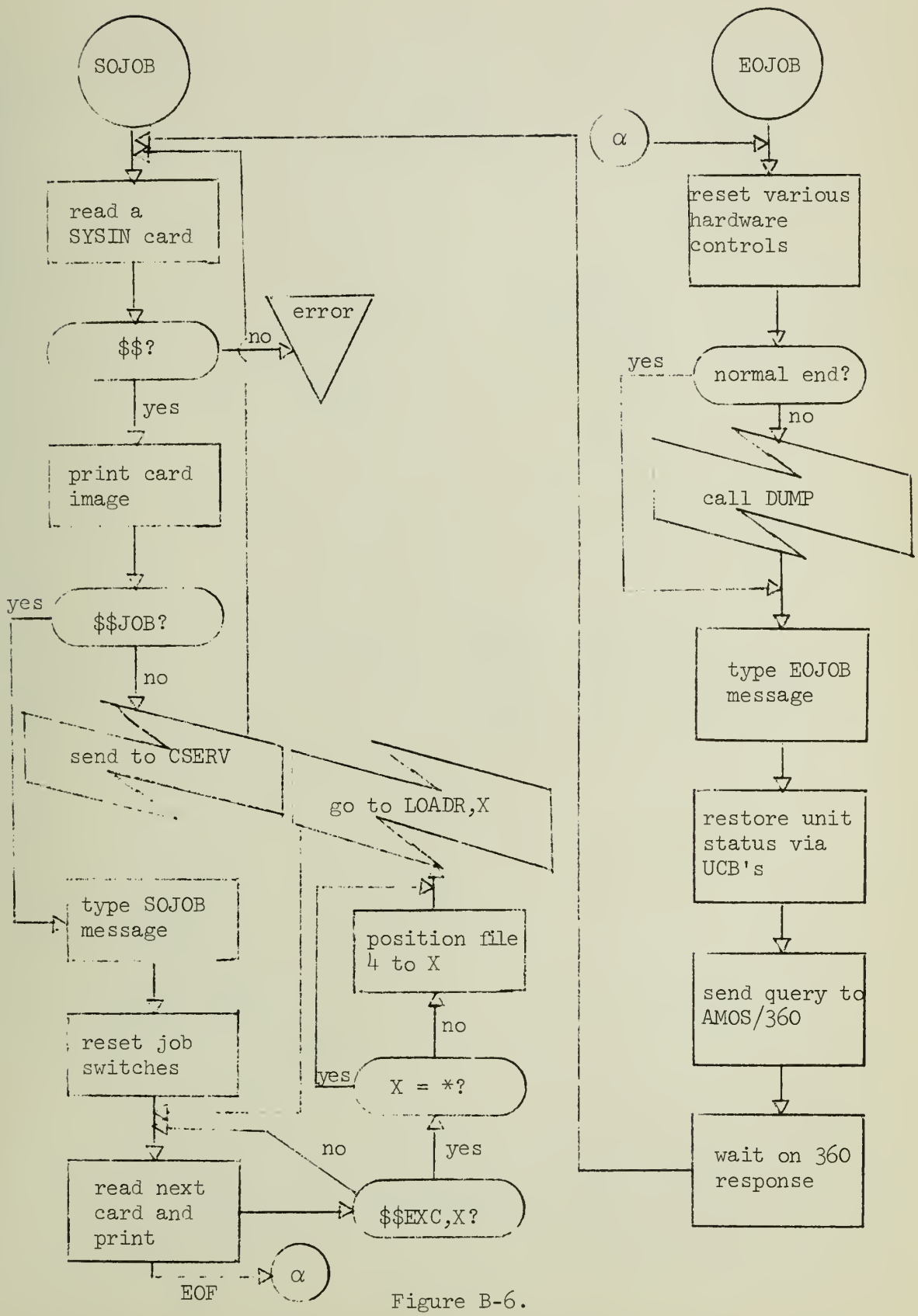
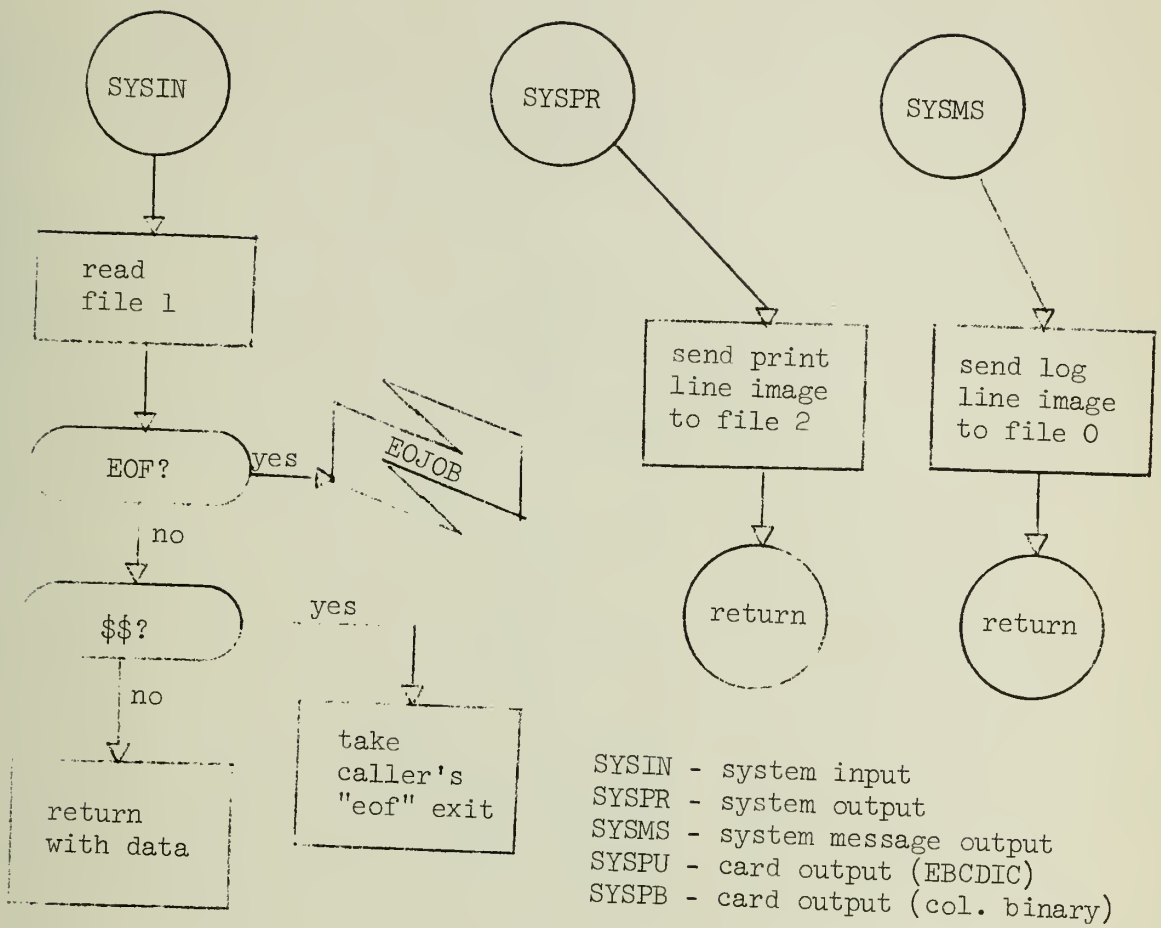


Figure B-6.

1800 Job Scheduler (SOJOB/EOJOB)



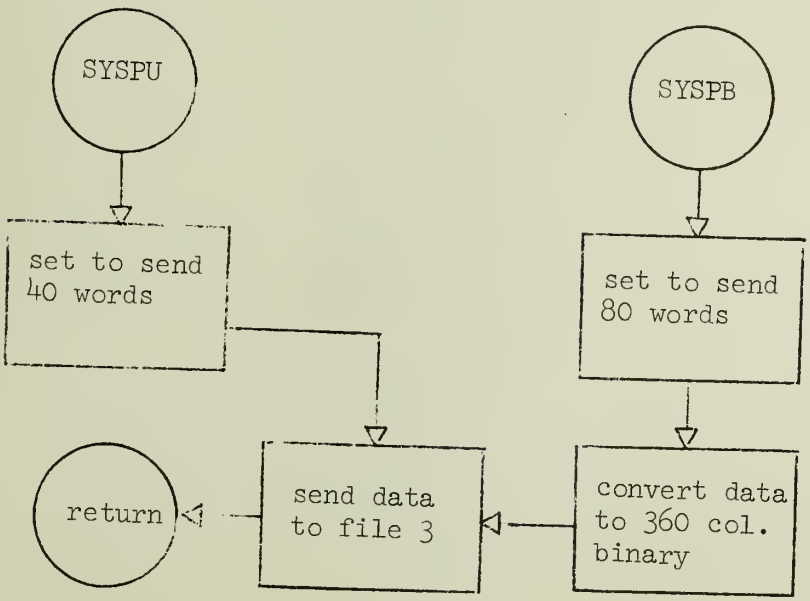


Figure B-7.

Adapter File Control (CAFIL)

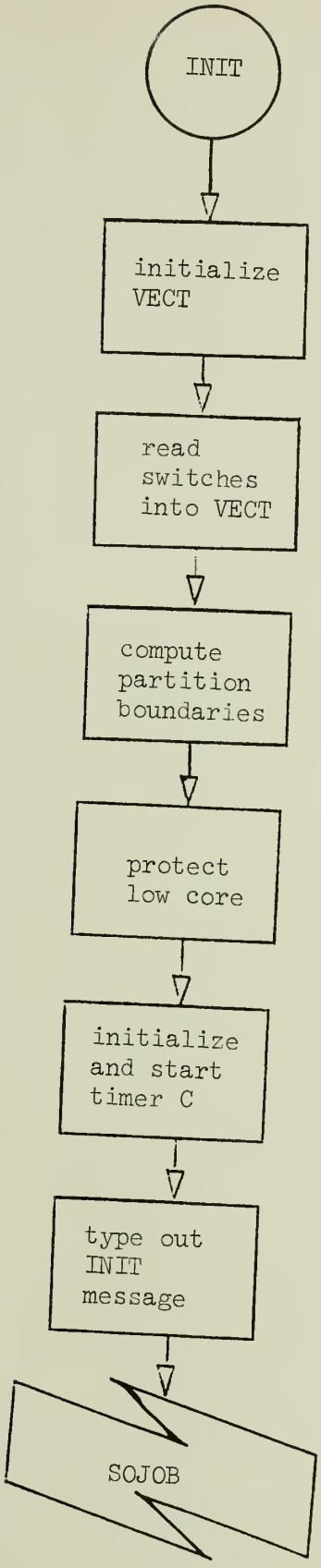


Figure B-8.

Initialization Program (INIT)