TO: MSPM Distribution
FROM: R.M. Graham
SUBJECT: BF.0
DATE: 12/28/66

The attached issue of BF.0 contains only minor editorial changes from BF.0 dated 8/3/66.

Published: 12/28/66
(Supersedes: BF.0, 08/03/66)

Identification

Overview of the I/O System
V.A. Vyssotsky and J.L. Bash

Purpose

The I/O system contains the procedures by which user and other Multics programs communicate with external devices. This section provides a general description of the I/O system, a list of interfaces with procedures of other Multics systems, a description of how the I/O system uses peripheral devices, and a description of the internal structure and interfaces of the I/O system itself. For detailed information on a particular aspect of I/O, the relevant subsection following the overview should be read.

General Discussion

This section describes the following:

1. Hardware devices supported by the I/O system.
2. Calls that can be made to I/O and their functions.
3. I/O system modules.
4. I/O system interfaces with other Multics modules.
5. I/O procedures that execute as part of certain processes.
6. Data layouts for the various external media that are supported by the system.
7. Internal data required by the I/O system.
8. Internal procedures required by the I/O system.
9. Provisions for alterations and extensions to the I/O system.
10. Permissible user modification to the I/O system and restrictions and protective mechanisms protecting other users from such modifications.
11. I/O system handling of errors and exception conditions and permissible user modifications to this system feature.

I/O Devices Supported by the I/O system

In a typical 645 installation, three kinds of hardware devices are connected directly to ports on the memory controllers. These are:

1. Central Processing Units (CPU's)
2. Drum Controllers
3. General I/O Controllers (GIOC's)

The drum controllers provide hardware access between drums and memory. Hardware access between all other peripheral devices and memory is through the GIOC's. (See Figure 1.) When Multics is running on a 645, no process can pass any data, control, or status information between memory and any hardware device connected through the GIOC except by calls to the I/O system. Thus, only devices supported by I/O can be accessed by a process through GIOC. Table 1 lists devices supported by the initial version of Multics I/O.

The file system handles drum and drum controller procedures. However, I/O does contain facilities for using file system files as I/O media. For files kept in core and on drum, the I/O system provides indirect means of manipulation, acting as a file system user. For files kept on disc and tape, access is obtained through calls to I/O. To access a file on disc or tape, the file system calls I/O specifying the file. I/O accesses the file through the file system. The file system then calls I/O again to perform actual file reading. Figures 2 and 3 show file system-I/O interaction.

Basic Divisions of I/O

Procedures and data bases are divided into four groups, shown in Figure 4. I/O routines closest to the hardware constitute the GIOC Interface Module (GIM). The GIM has the following functions:

1. The GIM calls the file system to latch into core a data area in addressible storage, specified by the caller by segment and word number. The GIM notes the absolute core locations occupied by the data area.
2. If GIOC load is such that starting further transmission might cause transfer timing errors, the GIM delays initiation of I/O until load on the GIOC is reduced.

3. The GIM compiles a list of DCW's which, when decoded by the GIOC, will cause the desired I/O function.

4. The GIM associates particular GIOC channel numbers with the various devices.

5. The GIM initiates activity on an I/O channel and passes status returns for that channel to the caller of the GIM. As indicated in Figure 4, users do not call the GIM directly. The file system does call on the GIM directly, but only for I/O to the DSU250 disc.

Immediately above the GIM level are routines called device interface modules (DIM's). For each device, except the DSU250 disc, there is at least one DIM and, in some cases, more than one. The modules at this level accept such calls as read, write, attach, and detach. Their functions are:

1. Provide a degree of uniformity in I/O calls to the various devices by generating order codes to the devices. These order codes are passed to a particular device in a particular sequence through the GIM, which relays them without interpreting their meaning. The meaning of an order code must be interpreted by the device itself. For example, a printer DIM order code of octal 6^1 means slew two lines, while a tape DIM order code of octal 6^1 means set tape to low density.

2. Provide for default error recovery and exception handling, such as retrying write following a parity error in writing tape without user intervention.

3. Perform code conversion for many of the devices. For example, since the hardware interface to the ASCII printer does not accept ASCII characters, the printer DIM converts ASCII text for the printer into character codes acceptable to the interface.

4. Perform readahead and writebehind for many of the devices. For example, the typewriter DIM can accept a user message before it has received a read request from the user program.

5. Perform, for devices such as tape, blocking of data into uniform physical record sizes, supplying an identifying prefix for each record.

User programs call DIM's indirectly through the next level of I/O routines, the logical service modules. Thus, physical devices such as a tape DIM can be referenced symbolically at the logical service module level, rather than by direct

reference at the DIM level, which would tie the call to a specific device. Other functions of the logical service modules are:

1. Handling of logically structured data of various degrees of complexity. For example, if the user wishes to treat data on a data disc as composed of logical records, 72 characters each, a logical service routine provides the facility.

2. Checking for errors in user calls, supplying default conditions for missing arguments, and in general, providing a call to the DIM in the form expected by the DIM.

The topmost level of I/O contains utility routines. The routines are differentiated from routines at other levels of the system by the greater scope of the tasks performed. For example, a call to a logical service module might request a block of data or attachment of a particular reel of magnetic tape. A call to a utility routine might request printing of an output file or copying the contents of a tape into the file system. There are two classes of utility routines: daemons and specialized routines. Daemons are routines not for a particular user but for the installation, such as the card reader driver. Specialized routines are utility routines for particular users, e.g., the iocopy routine. Tables 2 through 5 list the modules of the I/O system at each of the four levels.

I/O Data Structures

The I/O system allows the user three modes of data representation: physical, linear, and sectional. Data in physical form reflects the physical characteristics of the recording medium. For example, card data handled in physical mode occurs in records which are all of the same length, as on cards. The same data read from a typewriter console arrives as a steady stream of characters without record boundaries. Because of problems inherent in handling data in physical mode, this mode should be avoided whenever possible.

The basic representation of data for most users is linear. Every device and medium supported by Multics I/O can be used for linear data. In the linear data mode, the input from a medium such as tape or cards is buffered by the I/O system to appear as a steady stream of characters (or words or bits). For example, data read from cards can be read 80 characters at a time, one character at a time, 119 characters at a time, or any set buffer size regardless of the physical medium's limitation of 80 characters.

Sectional representation of data superimposes logical record structure on linear representation. Logical records may be of arbitrary length and need not be correlated with the physical structure of the external medium. For example, logical records ranging from one bit to 10 or more can be punched into cards. When read back as sectional data, physical record divisions are suppressed and logical divisions retained. Logical data may be further divided into hierarchical groupings. The I/O system provides for treating a group of logical records as a single logical record or for treating a single record as separable into several independent data sets.

Random or sequential access is provided for both linear and sectional data. Random access is provided only to data on random access devices or in file system files; data on tape must be copied to a random access medium before random access capability is available to it.

Blocking and Buffering

Different media require different techniques of blocking and buffering, including readahead and writebehind facilities. These techniques have various features in common. For record-oriented devices such as magnetic tape, a particular fixed-record size is chosen for the medium, and data is fitted to fixed-record size. The I/O is programmed for all devices on the following assumption:

> On linear and sectional I/O we may read arbitrarily far ahead on the user's input medium and fall arbitrarily far behind in writing his output media, provided that:
>
> 1. Console I/O does not get out of phase with the user program without his permission, and
>
> 2. bulk media, such as tapes respond properly to sequences of I/O calls. In particular, a write-rewind-read must work properly in all cases, and any detach should force output buffers onto the medium.

The I/O system provides relevant status return information from the various devices and provides for user control of the degree of readahead and writebehind to be permitted.

User Calls to I/O

Calls a user program makes to I/O serve the following purposes:

1. To specify the external media required for this run (e.g., magnetic tape, tape reel number.)

2. To specify modes applicable to the use of the external media for this run (e.g. 9-track, 800 bits per inch, read only, etc.)

3. To position the medium to some point, such as the beginning, the end, the next item after the one we are now positioned to, etc.

4. To transmit data to and from the medium.

5. To obtain information from the I/O system about the various media, and their states and modes, such as end of tape.

6. To specify I/O action to be taken in certain exceptional conditions (e.g. "If we reach the end of the reel, pass control to...")

Each call to I/O can serve several functions. A read call primarily transmits data but can also reposition the medium and return status information from the medium.

To read data from a reel of magnetic tape, a user might follow these steps:

1. Negotiate with transactor (not part of I/O) for a tape frame.

2. Issue attach call, causing tape reel to be mounted, readied, and have a unique name associated with it for use in subsequent read calls.

3. For a standard tape, issue read calls to read data.

   For a non-standard tape, issue modal calls to change I/O system interpretation of subsequent read calls. Then issue normal read calls.

4. Check for end of data on tape in one of the following ways.

   a. Examine status return after each read call.

   b. Before starting to read, specify a procedure to the I/O to be invoked on end of data.

5. When end of data is reached, issue detach call, causing tape rewind and dismounting.

6. To mount a new tape, issue an attach call. If through processing, inform the transactor that the tape frame is being released.

Many variations can be made in the calling sequence given above. Possible calls are described in BF.1 and a list of call names is given in Table 6.

Flow of Control

When a user calls the I/O switch to read the 'next' logical record into a workspace from a sectional data tape, the call might be:

```
call read ('x', dptr, nitems)
```

where: 'x' indicates the place from which data is read, dptr indicates the workspace into which data is read, and nitems is the number of items to be read.

The I/O switch determines that 'x' is composed of logical records by a table lookup and forwards the call to the logical record formatter. The lrf makes a logical record header available to indicate length of the logical record. If logical record length is 80 and nitems = 100, lrf may make the following call:

```
call read ('..001',,dptr,80)
```

The name '..001' was obtained by the lrf from the attach table, described later. The I/O switch receives the call, performs a table lookup on '..001, and if satisfied, forwards the call to the tape device strategy module (dsm). The dsm checks its buffers for all, part, or none of the required data. If all data is available, the dsm delivers the data and returns. If none is available, the dsm can call the I/O switch with

```
call read ('..002',,buffer, 1056)
```

The call is forwarded by the I/O switch to the tape device control module (dcm). The dcm makes a call to the GIM that causes a physical tape read and then returns control through the I/O switch to the DSM. When the physical record arrives in the buffer, the dsm moves the data to the user's workspace and returns to the lrf.

The lrf replaces the logical record header by a call such as:

```
call read ('..001',,fcblock,16)
```

which causes the dsm to move data into lrf storage.

Figure 5 shows the flow of control for the example above.

Outer Calls and Outer Modules

A call to an I/O system procedure is made in one of two ways. The call may be made directly to the procedure which will perform the required function; these are known as inner calls. Or the call may be made to the I/O switch procedure for forwarding to the appropriate destination. Such calls are known as outer calls. Similarly, all modules of the I/O system called by the I/O switch are outer modules, and the remaining I/O system modules that are called directly are known as inner modules. The I/O switch can call all entry points in a given I/O system module or it cannot call the module at all.

Within a wide range, outer modules may be interchanged or removed. Inner modules do not have this flexibility. Each outer module contains all entry points necessary for proper functioning but not necessarily an entry point for every outer call. Outer calls are listed in Table 6 and outer modules in Table 7.

There is one exception to the rule that outer modules are called only through the I/O switch. Certain I/O modules may call outer modules by a direct call explicitly specified for this purpose.

Outer Modules and Their Functions

The I/O switch forwards to the outer module, notfounder, any call containing an ioname the switch cannot identify. Notfounder either rejects the call (as in the case of a read to an unknown device) or tentatively identifies the module to receive the call and reroutes the call to that module (as in the case of an attach call).

The logical record formatter (lrf)is the outer module handling sectional data, subdivisions of logical records, and hierarchies of records. This facility, called the subframe capability, is entirely concentrated in the lrf regardless of the external medium.

The broadcaster is an outer module that permits a single ioname to reference several destinations on output. For example, the name 'xy' may be defined by attach calls

to refer to a tape, line printer, two file system files, and four typewriters. A single write call to the broadcaster replaces 8 separate write calls to the output devices, with the broadcaster performing the fanout and handling returned status information.

Broadcast situations are set up by attach calls, each of which specifies one ioname and one device. When an ioname is first mentioned in an attach, the I/O system does not know whether or not that ioname will be used for broadcasting. An outer module, called the streamer, is used to catch such attach calls and associate the ioname directly with the device in a way that allows detection of a broadcast situation and invocation of the broadcaster.

The intervenor and stub are outer modules used for debugging. Any calls using a particular ioname can be routed to the intervenor before continuing along the normal I/O system path; the intervenor keeps a running record of calls containing a particular ioname. The stub is a null device. When a read request is sent to the stub, it reads no data but returns status information showing data read. The two modules can be used separately or together for checkout of logic of new I/O system modules and for checkout of I/O logic sequences in user procedures.

Outer modules concerned with input and output to file system storage are the file system interface module (fsim) and the segment interface module (sim). The fsim designates a file system file as an I/O device. The file can be regarded as random or sequential, linear or sectional. If data storage requires more than one file, the fsim automatically designates a group of files to hold the data. The segment interface module performs I/O to and from a single segment, attached by segment number. The sim is used by outer modules such as the fsim and lrf but can also be designated by a user who wishes, for example, to regard a PL/I array as an I/O device.

Every I/O-supported device is operated through a device interface module (DIM) that usually consists of device strategy module (DSM) and a device control module (DCM). DIM's and their submodules are outer modules. The DIM's handle all outer calls for linear I/O to the devices. The DIM's do not contain any subframe or broadcast capability. Thus, each DIM regards every I/O device as recording a string of elements with a beginning and end but no internal logical structure. If a device

has random access capability, the DIM supports random access. Every DIM supports sequential access.

When a DIM is divided into a DSM and DCM, the DCM provides physical mode capability. The DSM provides linear capability if the device is record-oriented (tape, card reader, disc pak). Synchronization management is largely concentrated in the DSM. Writebehind in any significant amount must be done via the DSM, and the DSM issues anticipatory read calls to the DCM. Buffering capability in the DCM and the GIM is extremely limited, and under no circumstances, does a DCM issue an anticipatory read to the GIM.

As user output is generated, it is placed in a file system file for later extraction and output to a physical device. The user is not concerned with the intermediate storage; he wishes status returns that reflect any difficulties with the actual output device. Outer modules that mimic the behavior and status returns of actual output devices and interface between the user and the file system interface module are called pseudodims. There is a pseudodim for each device supported by the I/O system.

Inner Modules

The inner modules of the I/O system are:

1. The utility routines, such as iocopy and the drivers.
2. The GIM
3. The I/O switch
4. Various small service routines.

Only the I/O switch and service routines are discussed here.

The I/O switch checks the ioname in each outer call against its entry in the attach table to determine the procedure to receive the call. The switch does not perform direct table lookup; this is the function of inner module, atsearch, which returns a pointer to the desired entry. Besides attach table search, the I/O switch does some argument validation.

Other inner modules are the data set interface modules. These are called by DCM's that operate devices over communications adaptors. The function of the data set interface modules is to analyse status reports from the computer during startup and shutdown of the communications link.

I/O System Data Bases

The central data table of I/O is the attach table. The attach table contains one entry for each ioname known to I/O at a given moment, in a given ring, and for a given process. The entry specifies a procedure to which calls containing that ioname are routed and a pointer to a data block used by the procedure to store information on transactions with that ioname. Other information in the attach table entry is type and use attributes and status information. Any outer module knowing a given ioname can retrieve all attach table information under that entry and all other information pointed to by the entry. This makes it possible to retrieve required information after a working process catastrophe such as an operation-not-complete fault.

The form of the data block pointed to by the attach table entry varies with the procedure associated with the ioname but must be completely determined when the procedure and the content of the attach table entry are known.

Two other tables are used to form attach table entries. One is the type table. When an attach call arrives, for example,

call attach ('x', 'tape9', '4321', 'rw')

'tape9' is looked up in the type table. Entry information on 9-track tapes, together with the call arguments, is used to form an attach table entry. A type table entry is structured as an attach table entry and is, in effect, the skeleton used to form the attach table entry. Type additions, deletions, and changes can be made in the type table by means of a version of the attach call directed to the type table instead of the attach table. This provides capability for testing new versions of I/O system modules and creating new I/O pseudodevices.

The attribute table provides a means of relating attributes appearing in a call to types implied by those attributes. For example, in the call

call attach ('x', 'tape9', '4321', 'ws')

where attribute 's' means sectional, ioname 'x' must be associated with logical record formatter, not the DIM for 9-track tapes. Attribute table entries have the same form as attach and type table entries.

Errors, Exceptions, Recovery and Status

The design criteria for exception handling in the I/O system are as follows:

1. Since the file and command systems are I/O system users, I/O must be able to return control to its caller under most circumstances to avert system shutdown. I/O may fail to return control only if:

   a. the caller does not want the return (e.g., caller specified abort on end of file)

   b. I/O system was scrambled before call (e.g., I/O switch deleted by user. In this case, we can safely assume that the user is not the file system.)

   c. Hardware failure other than I/O peripheral unit or subsystem failure.

2. For reliability, detailed status information must be available (e.g., ideally, we could say to the calling procedure, "Your tape record has lateral parity failures in characters 27, 28, 429, 430, 431 and 443".)

3. Status returns must not be given when not wanted.

4. Flexible exception handling disciplines must be available, such as ability to run a job involving three multireel tape files, each with a different labeling convention.

5. Default exception handling must be reasonably competent, so that at least 95% of user programmers can ignore exception handling.

Objectives 1 and 5 may conflict in cases where objective 5 says give up and abort, and objective 1 says return to caller. In all such cases, objective 1 must prevail.

The general design for exception handling is as follows. Every outer call to the I/O system has, as a final optional argument, a pointer to a status bit string. The caller may, upon return, examine the bit string directly, or alternatively, exception handlers may be called from within the I/O system each time status bits are set to 1. Default or user-supplied exception handlers are treated as outer modules. If a status bit relevant to an exception handler is set to 1 as a result of an outer call, the outer call is reissued to the exception handler, which then has available

to it everything the I/O knows about the transaction. If, for example, a user's special handler is attached to parity errors on tape 'x' and a parity error results from the call

```
call write ('x', data, 100)
```

then the I/O calls the special handler as follows:

```
call write ('x', 0, data, 100, sptr)
```

where sptr is a pointer to status concerning the transaction. The user's handler can retrieve attach table entry and associated data blocks for further information.

Like all outer modules, exception handlers must have sufficient entry points to receive control whenever corrective action will need to be taken.

I/O System Division between Processes

There is a device manager process for each physical device. A working process normally initiates I/O via a wakeup to the device manager process for the device. Some I/O procedures execute in the working process, others in the device manager, and others in both. In the latter case, the function performed and the mode of device attachment determine in which process a procedure executes. For example, procedures invoked by a user writing tape are shown in Figure 6. The lrf always executes as part of the manager but the DSM and DCM may execute in either process.

On a normal write request with writebehind, the request passes from working process to device manager at the DSM level. Part of the DSM and all the DCM execute in the manager. If a parity error is detected in writing a physical record on tape and default action is taken, it normally takes place within the device manager, so both DSM and DCM execute in the manager. Non-standard recovery action, however, executes in the working process, so both DSM and DCM execute in the working process.

Division of the I/O System between Rings

Every instruction executes as part of some process and as part of some ring of protection within the process. Most procedures of the GIM execute in the innermost ring, the hard core supervisor; the remainder in the administrative ring. Utility programs execute in the user base ring. The other I/O system procedures execute either in the

user base ring or the administrative ring, depending upon the procedure's callers. Callers in the hard core or administrative ring cause the I/O procedure called to execute in the administrative ring. Calls from any other ring cause the I/O procedure called to execute in the user base ring.

This technique makes I/O flexible and efficient for the unprivileged user, while preserving the integrity of I/O system procedures and data bases for privileged parts of Multics.

I/O System Interfaces to the Rest of Multics

The I/O user interface is called by Multics subsystems such as the answering service, shell, and file system for tape I/O. The file system calls the GIM directly for disc I/O. Various Multics modules can call the I/O mayday routine.

The I/O system, in turn, calls several Multics modules outside the I/O system. The fsim is a file system user. The typer calls linkage builder. Many I/O modules, including almost all DSM's and DCM's, call block, wakeup, and restart in the traffic controller. Every DCM calls the transactor and/or file system access control when a peripheral is first attached. The GIM calls the file system and interfaces with the interrupt interceptor, and the reader driver uses the file system and triggers the absentee job scheduler.

Alteration and Extension of the I/O System

The GIM is a privileged procedure, largely in the hard core ring. To change any part of the GIM, appropriate privilege is needed; the change is difficult to make and could cause a catastrophe. The GIM can only be modified on a system-wide basis; no user can have "his own version" of the GIM.

All other I/O procedures can be modified by the system or by a user. System changes are made by modifying the library of I/O procedures that resides in the administrative or user base ring or in both. Modification may include additions, deletions or substitutions. Users have several methods for modifying procedures. A user creates a new outer module by putting it into his working directory and entering it in the I/O type table by an attach call. A user substitutes his version of an outer module by putting it in his working directory and modifying the type table

entry to specify his version in place of the library version. All attach calls then are to his version. It is possible to replace a library module with a private version in an existing attachment. To do this, the user can detach and reattach the attachment; buffers and status will be handled properly in such a switch but the medium may be repositioned in an incorrect manner. The other alternative is to find the attach table entry pointing to the library module and change it to point to the private version. This insures against repositioning of the medium but the private version must be able to handle data layouts representing status, etc., exactly as did the library module.

Inner modules may be substituted on a system-wide basis with the same ease as outer modules. However, it is not convenient to substitute isolated inner modules within a process. Any I/O system inner modules relevant to a user process are already linked in by the time the user would be ready to change them. Hence, the change involves removing the library version of the module and then substituting a new version, rather than putting the new one in alongside the old as with outer modules.

Care must be taken in modifying I/O system modules. A new outer module, called through broadcaster or lrf and returning nonsense, can cause lrf or broadcaster to malfunction with catastrophic effect on the user process. However, used with care, the flexible substitution and addition of outer modules provide ease in handling special I/O requirements. For example, the user may provide his own merge for input from two or more typewriters into a single message sequence or provide pseudodevices and supply DSM's and DCM's for new, specialized I/O devices. To attach a random number source as a pseudodevice, the user codes the procedure to accept I/O system outer calls. If the procedure is named MYRND4, the user informs I/O of its existence by:

```
call attach ('rgen', 'newtype', 'myrnd4')
```

and can then attach as many ionames as he wishes to pseudodevices of his newly defined type, 'rgen'.

Table 1

HARDWARE DEVICES SUPPORTED BY INITIAL MULTICS I/O

Model No.	Description
DSU 250	Disc Storage Unit
DSU 150	Removable Disc Storage Unit
MTH 412	9 Track Tape Drive
MTH 311	7 Track Tape Drive
PRT 202	Printer with ASCII Character Set
CRZ 200	Single Stacker Card Reader
CPZ 200	300 CPM Card Punch
PTS 200	Paper Tape Reader and Punch
IBM 1050	Remote Typewriter
IBM 2741	Remote Typewriter
TTY 37	Teletype Model 37
801B	Automatic Telephone Dialing Unit
GE 115 CRZ 100 CPZ 101 PRT 100	Remote Satellite Printer, Punch, Reader Installation

Table 2

MODULES AT THE GIM LEVEL

Throughput Control
Core Latch Handler
DCW Compiler
DCW Postprocessor
I/O Initiator
I/O Cleanup
I/O Work Distributor
Channel Assigner
Common Peripheral Channel Multiplexor

Typewriter Adaptor Module
201A Adaptor Module
801B Adaptor Module
CPIA Adaptor Module
CPD's Adaptor Module

Table 3

MODULES AT THE DIM LEVEL

GE 115 DSM
GE 115 DCM
PTS 200 DSM
PTS 200 DCM
DSU 150 DSM
DSU 150 DCM
Standard Tape DIM
Flexible Tape DSM
Flexible 7 Track Tape DCM
Flexible 9 Track Tape DCM
PRT 202 Printer DSM
PRT 202 Printer DCM
CPZ 200 Punch DSM
CPZ 200 Punch DCM
CRZ 200 Reader DSM
CRZ 200 Reader DCM
Typewriter DSM
IBM 1050 DCM
IBM 2741 DCM
TTY 37 DCM
801B DIM

103A
103E Data Set Interface Modules
201A
202C

Various Exception Handlers

Table 4

MODULES AT THE LOGICAL SERVICE LEVEL

I/O Switch
Notfounder
Streamer
Broadcaster
Logical Record Formatter
File System Interface Module
Segment Interface Module
Intervenor
Stub
Typer
Attributer
Attach Table Search
Pseudoprinter DIM (PRT 202)
Pseudopunch DIM (CPZ 200)
Pseudoreader DIM (CRZ 200)
Pseudoprinter DIM (PRT 100)
Pseudopunch DIM (CPZ 101)
Pseudoreader DIM (CRZ 100)

Various Exception Handlers

Table 5

MODULES AT THE UTILITY LEVEL

PRT 202 Printer Driver
CPZ 200 Punch Driver
CRZ 200 Reader Driver
GE 115 Driver
iocopy
Mayday routine

Table 6

USER CALLS (OUTER CALLS)

attach
detach
noattach
trace

bounds
sizes
dmode
readsync
writesync
breaks

read
write
seek
tell
delete
first
tail

cleanup
order
status

trapon
trapoff
traprestore

Table 7

OUTER MODULES

GE 115 DSM
GE 115 DCM
PTS 200 DSM
PTS 200 DCM
DSU 150 DSM
DSU 150 DCM
Standard Tape DIM
Flexible Tape DSM
Flexible 7 Track Tape DCM
Flexible 9 Track Tape DCM
PRT 202 Printer DSM
PRT 202 Printer DCM
CPZ 200 Punch DSM
CPZ 200 Punch DCM
CRZ 200 Reader DSM
CRZ 200 Reader DCM
Typewriter DSM
IBM 1050 DCM
IBM 2741 DCM
TTY 37 DCM
801B DIM

Notfounder
Streamer
Broadcaster
Logical Record Formatter
File System Interface Module
Segment Interface Module
Intervenor
Stub
Typer
Attributer
Pseudoprinter DIM (PRT 202)
Pseudopunch DIM (CPZ 200)
Pseudoreader DIM (CRZ 200)
Pseudoprinter DIM (PRT 100)
Pseudopunch DIM (CPZ 101)
Pseudoreader DIM (CRZ 100)

Various Exception Handlers

Figure 1

SAMPLE HARDWARE CONNECTIONS BETWEEN DEVICES

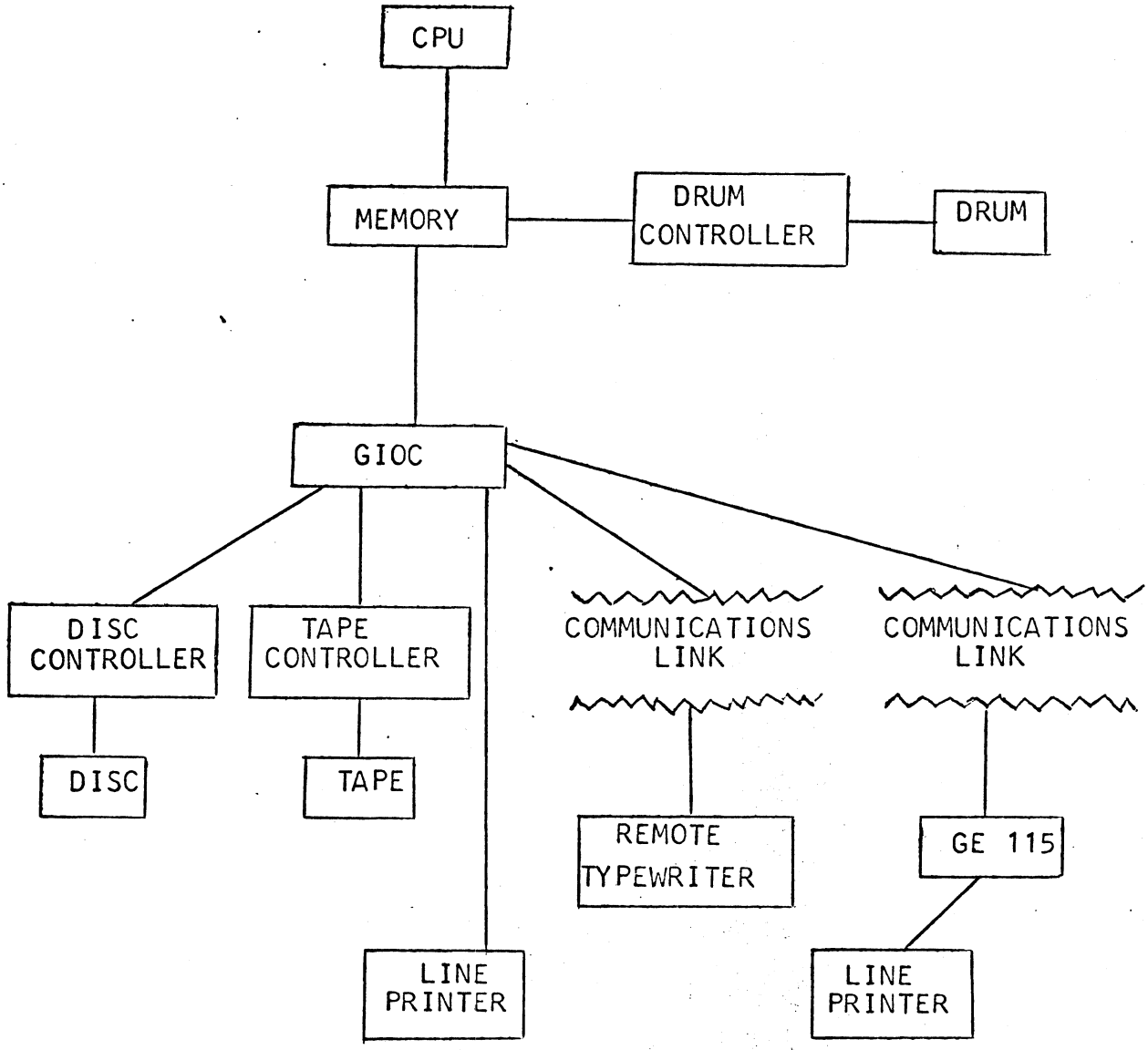
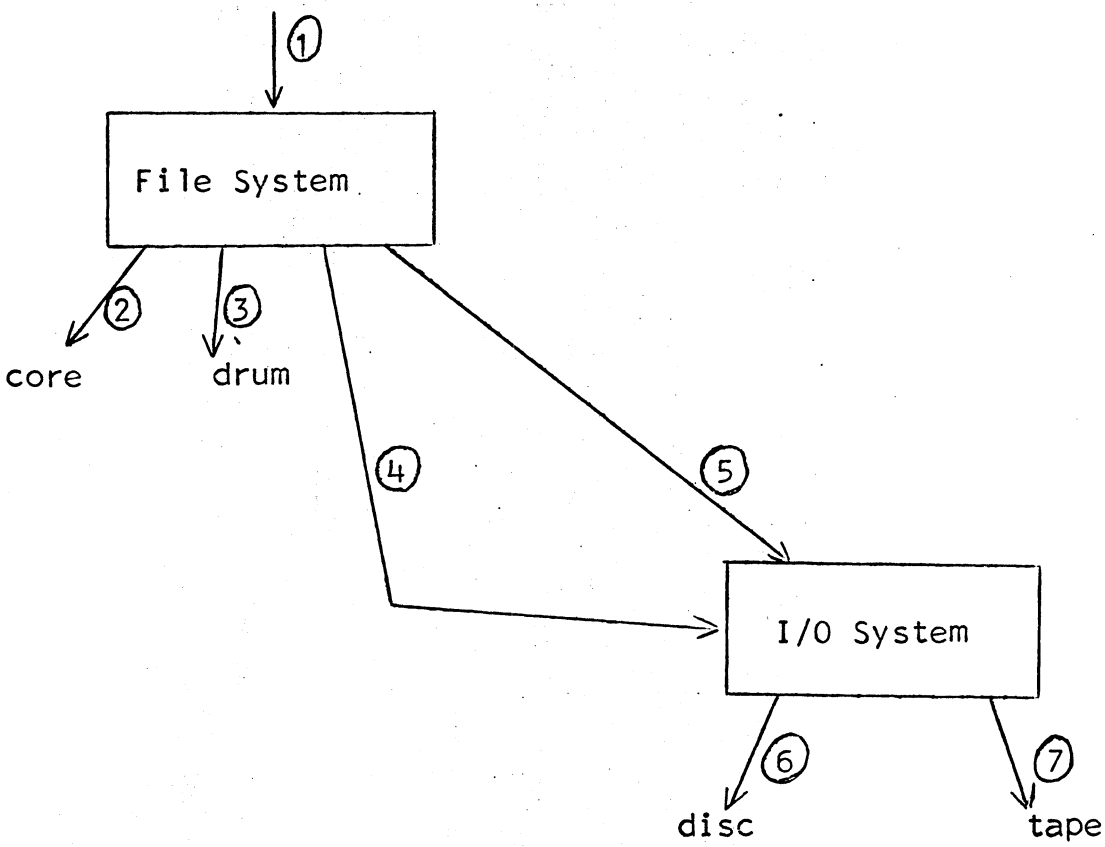


Figure 2

STORAGE AND RETRIEVAL OF FILE SYSTEM FILES



(1) Request for file system action (by user or Multics component).

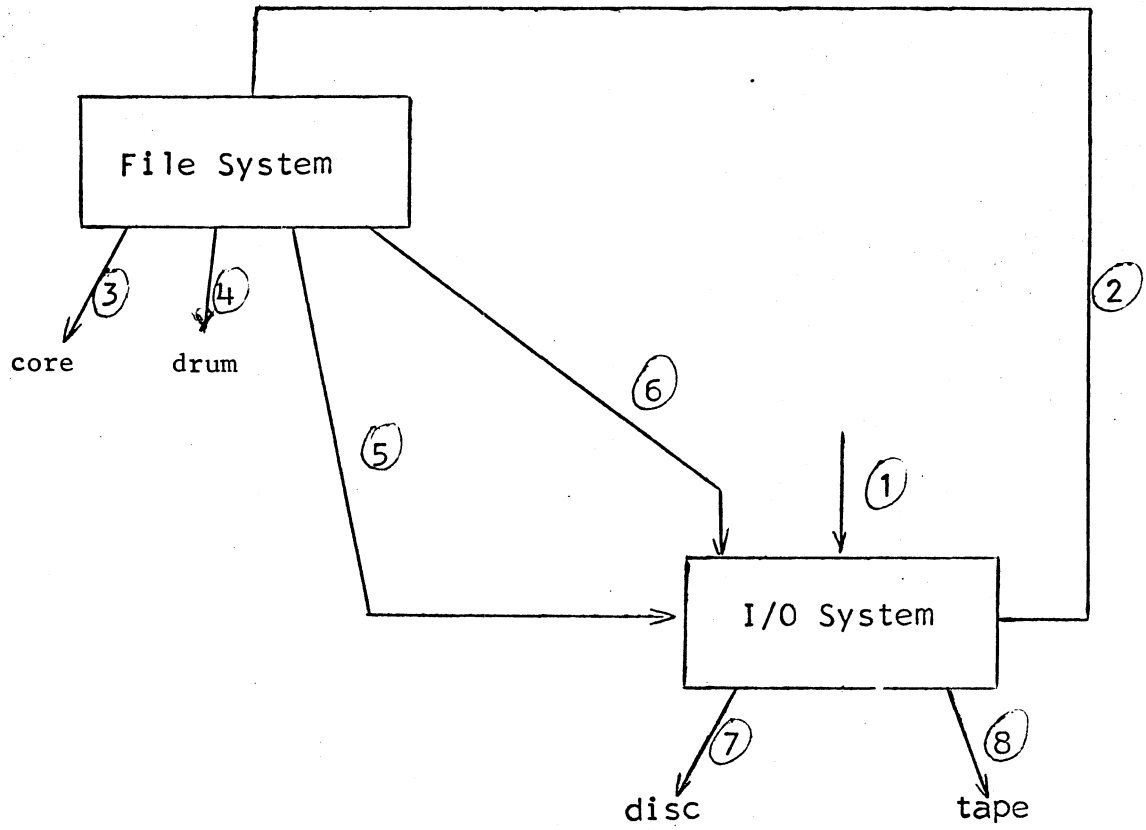
(2)(3) File system satisfies request from core or drum if file is in core or on drum.

(4)(5) If file is on disc or tape, file system calls upon I/O system.

(6)(7) I/O system runs disc or tape as required by file system.

Figure 3

FILE SYSTEM FILES AS I/O MEDIA



(1) Request for I/O using file system file (by user or Multics component)

(2) I/O system calls upon file system to make data available.

(3)(4) If file is in core or drum, file system satisfies request.

(5)(6) If file is on disc or tape, file system calls upon I/O system.

(7)(8) I/O system runs disc or tape as required by file system.

Figure 4

GROSS DIVISION OF THE I/O SYSTEM

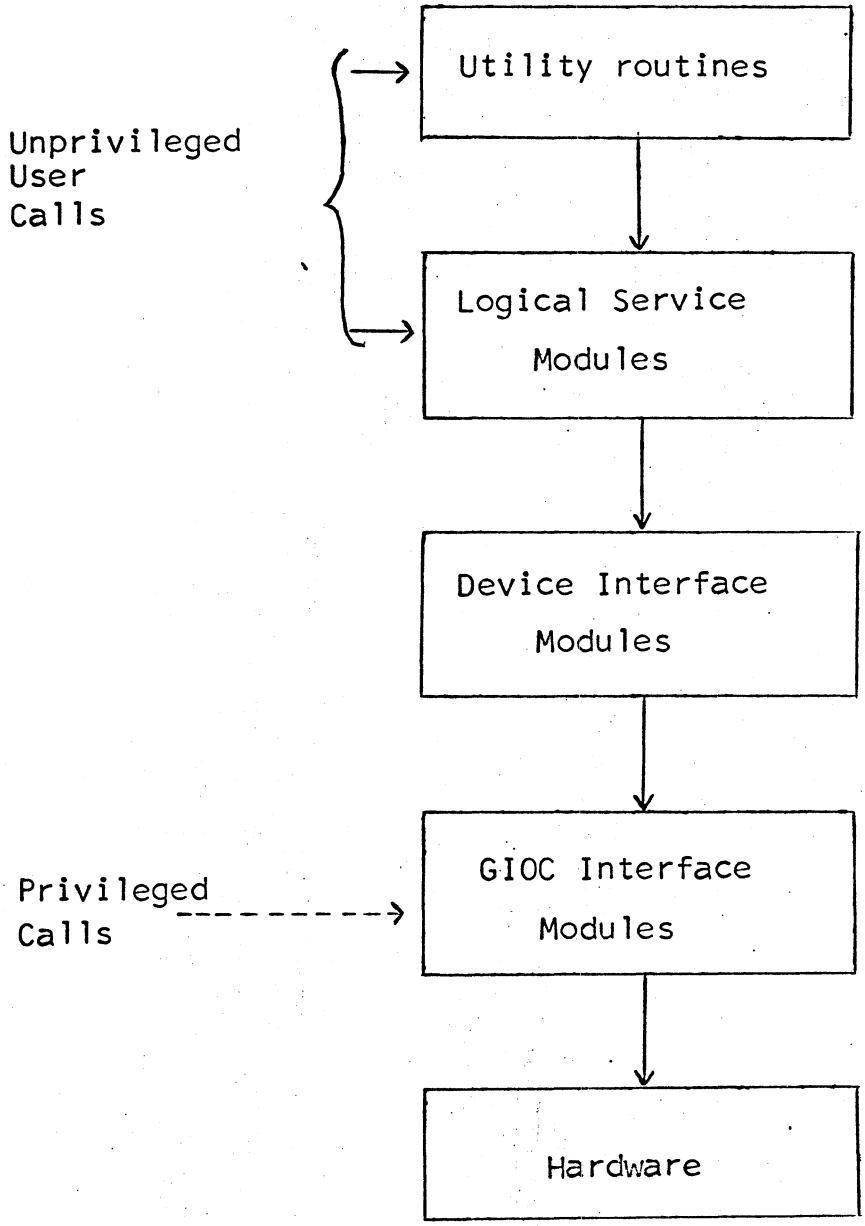
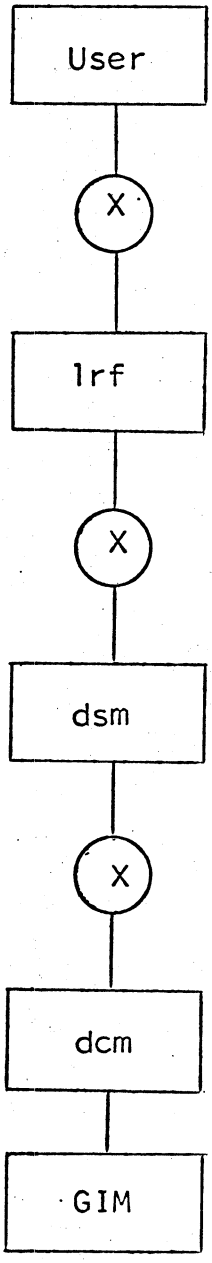
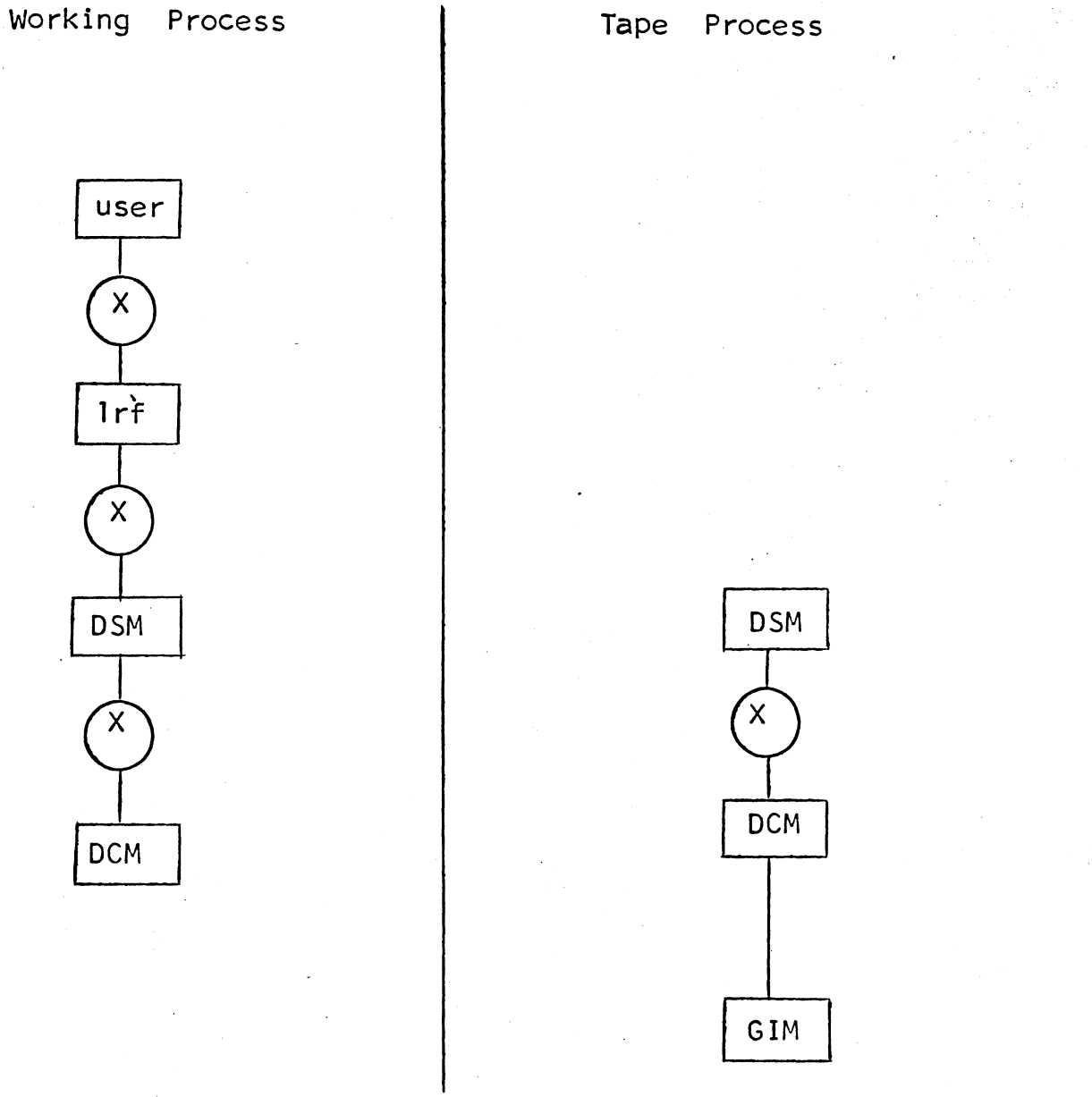


Figure 5



(X) indicates I/O switch.

Figure 6


Working Process
Tape Process
user
X
lrf
X
DSM
X
DCM
DSM
X
DCM
GIM