RAYTHEON

QUALITY SOFTWARE

700 PROGRAMMING SYSTEMS

I/O MONITOR

DATE: October 1968

ID CODE: BNR

DRAWING: 391476

LABEL: MONB

AUTHOR: JMMC

SOURCE: SYM II

OBJECT: Absolute

PURPOSE

The Raytheon 703 Input /Output Software (IOS) provides the user with a powerful and flexible file oriented input/output system. Features of the system include device reassignment, simultaneous I/O capability, centralized I/O monitor, end action, and I/O macros. IOS provides a common I/O system for all users and alleviates the programmer's need to understand the machine language I/O operations of the 703 computer, while still allowing full use of the 703 capabilities.

IOS is able to perform I/O with any standard 703 peripheral device through the use of the 703 priority interrupt system, which allows rapid response to the necessary I/O interrupts. To further minimize time spent doing I/O the software is file oriented; i.e. all information necessary to perform the I/O operation is contained in a File Input/Output Table (FIOT, see 1.3) which is readily available to the device drivers. IOS operates entirely in the automatic interrupt mode which enables processing concurrent with I/O operations. This allows the user to make more efficient use of the computer than he might otherwise provide for himself.

Several I/O macros are available in IOS. These macros reduce the effort required to program efficient input/output. The user need only become acquainted with macros OPEN, DOIO, and STAT to communicate with any standard 703 device.

TABLE OF CONTENTS

1.1 IOS FUNCTIONS

Several operations are available in IOS. These are:

1. Read
2. Write
3. Rewind
4. Backspace
5. Write end of file
6. Search for end of file
7. Write skip
8. Status
9. Verify disk
10. Punch leader

Not all functions are available for every type of I/O device. For example, one cannot rewind the teletype. Refer to the driver descriptions for applicable functions.

IOS is an input/output control system which provides a user the facility to perform I/O operations which are effectively independent of the actual physical device being used. This is achieved by allowing different physical devices to be substituted for I/O operations without requiring re-assembly or recompilation of the program. For example, a program can be debugged using a small set of test data punched on cards and read from the card reader. After debug, during production runs, the input can be changed to magnetic tape to process the actual data files.

A user performs I/O by requesting data transfers on logical units, which can be related to any physical device. In a typical I/O request to IOS, the user specifies a Logical Unit Number (LUN), the address of the I/O data buffer and the number of words to be or written.

The logical unit is associated with a particular driver which performs a data transfer on a specific I/O device. The user can, at load time, cause this association to change so that a different driver and, therefore, physical device is referenced by the I/O request. This driver/device association is kept in a Peripheral Equipment Assignment Table (PEAT, see 1.2).

The device number or LUN assignment is made at system creation time. The IOS user uses the established LUNs and is free to reassign a given LUN to one of a set of devices, through X-RAY control directives. (These directives are described in the EXEC documentation).

All information pertaining to an I/O operation is kept in an 8-word File Input/Output Table, (FIOT - see 1.3). This table details the specific operation for the I/O request, as well as the LUN, I/O buffer, and the number of words. FIOT also provides temporary storage area as needed by the device driver.

1.1.1 IOS Calls

Macros are communication links between the programmer and IOS. All calls go through IOS subroutines to the appropriate device drivers. They translate information from an argument calling sequence to a FIOT, perform the I/O operation, provide error checking and end-action.

1.1.2 IOS Drivers

An I/O driver is a subroutine designed to perform data transfers between a specific peripheral device and the computer. An I/O driver is never directly used by a programmer, but only indirectly via IOS macros. Thus, for example, a programmer requests data to be read from some logical unit. IOS macros pass the FIOT address to the drivers, and the driver then extracts necessary information to start the I/O operation. The driver then exits back to the user program.

1.1.3 IOS Interrupt Service

Once initiated, I/O operations are processed independent of the user's program. Data transfers for devices on the Direct Memory Access (DMA) channel (such as disk and magnetic tape) are performed automatically by the hardware. The interrupt service routine portion of each driver performs all data transfer operations for devices on the Direct Input/Output (DIO) channel (such as; paper tape reader, teletype, card reader, etc.) When the I/O operation is finished, then interrupt service routine for DIO and DMA devices perform all operations required to conclude the I/O operation. Device status and ending buffer address are stored in the FIOT. At this time an end-action subroutine may be executed, which permits any arbitrary operation to be performed as a part of the interrupt service function.

1.2 PERIPHERAL EQUIPMENT ASSIGNMENT TABLE (PEAT)

IOS processes input and output operations via logical equipment. Logical equipment is a functional classification of I/O devices; such as, symbolic input devices, listing devices, binary output devices, etc. Therefore, since program reference to I/O are by function, different devices may be assigned to a logical unit without requiring a compensating change in the program reference.

For example, the binary input device may be the high speed paper tape reader for one job and the teletype for the next, but neither the I/O monitor nor the users program need be reassembled. The device may simply be reassigned by an XRAY directive (refer to the X-RAY EXEC documentation).

The peripheral equipment assignment table contains the information about the correspondence between logical and physical units. IOS uses this table to select the appropriate drivers when input or output requests are made.

1.2.1 Logical Unit Definitions

The PEAT shown in Figure 1 contains more equipment than is included with a typical system, but it is needed to show how logical units function. The first 7 units are system logical units that are referred to by name in the remainder of this manual and in other manuals.

The definition of each of these names is listed below:

LUN	Description	
0	SYSF	SYStem File. This is the unit from which the system library is read and written. In Figure 1 SYSF points to the disk.
1	SYSI	SYStem Input. SYSI is the unit for all directives input to system programs; such as the X-RAY EXEC or SYM II Assembler. In Figure 1 SYSI points to the teletype.

RESMON PEAT TABLE 09/12/68 PAGE 66

```
                    3419 *                    PEAT TABLE                                          VW 33000
                    3420 *  THE PHYSICAL UNIT ASSIGNMENT TABLE IS GENERATED                       VW 33010
                    3421 *  UP TO THE HIGHEST NUMBERED UNIT PRESENT IN THE                        VW 33020
                    3422 *  SYSTEM                                                                VW 33030
                    3423 *  PETE TABLE FORMAT IS AS FOLLOWS                                       VW 33040
                    3424 *                    ********************                                VW 33050
                    3425 *          PETE(0) *P   INR* DEV* UN *     WORD 1                        VW 33060
                    3426 *                    ********************                                VW 33070
                    3427 *          PETE(1) * DRIVER ADDRESS  *     WORD 2                        VW 33080
                    3428 *                    ********************                                VW 33090
3FD9 1200           3429 PEAT      BYTE  ASYSF,0                    0    SYSF                   VW 33100
3FDA 0000           3430           BYTE  ASYSI,0                    1    SYSI                   VW 33110
3FDB 1500           3431           BYTE  APRIN,0                    2    PRIN                   VW 33120
3FDC 1700           3432           BYTE  ALIST,0                    3    LIST                   VW 33130
3FDD 1500           3433           BYTE  ABIN,0                     4    BIN                    VW 33140
3FDE 1600           3434           BYTE  ABOUT,0                    5    BOUT                   VW 33150
3FDF 1200           3435           BYTE  ASCR,0                     6    SCR                    VW 33160
3FE0 1500           3436           BYTE  ALOGA,0                    7    LOG A                  VW 33170
3FE1 1600           3437           BYTE  ALOGB,0                    8    LOG B                  VW 33180
3FE2 1200           3438           BYTE  ALOGC,0                    9    LOG C                  VW 33190
3FE3 0E00           3439           BYTE  ALOGD,0                    10   LOG D                  VW 33200
3FE4 0F00           3440           BYTE  ALOGE,0                    11   LOG E                  VW 33210
3FE5 8000           3441           BYTE  X'80',0                    12   DUMMY PHYSICAL UNIT    VW 33220
3FE6 39F6           3442           DATA  M.DUM                                                  VW 33230
     0038           3443 TTYU1     EQU   TTYU+TTYU+TTYU+TTYU                                     VW 33240
     00E0           3444 TTYUX     EQU   TTYU1+TTYU1+TTYU1+TTYU1                                 VW 33250
3FE7 80E0           3445           BYTE  X'80'+TTYI,TTYUX           13   TELETYPE               VW 33260
3FE8 3DB4           3446           DATA  M.THSPT                                                VW 33270
                    3447           FALS  NUNITS=0                                               VW 33280
                    3448           TRUE  NMAG>0                                                 VW 33290
     0008           3449 MAGU1     EQU   MAGU+MAGU+MAGU+MAGU                                     VW 33300
     0020           3450 MAGUX     EQU   MAGU1+MAGU1+MAGU1+MAGU1                                VW 33310
3FE9 8420           3451           BYTE  X'80'+MAGI,MAGUX           14   MAG TAPE 0             VW 33320
3FEA 3CC6           3452           DATA  M.DMD                                                  VW 33330
3FEB 8421           3453           BYTE  X'80'+MAGI,MAGUX+1         15   MAG TAPE 1             VW 33340
3FEC 3CC6           3454           DATA  M.DMD                                                  VW 33350
3FED 8422           3455           BYTE  X'80'+MAGI,MAGUX+2         16   MAG TAPE 2             VW 33360
3FEE 3CC6           3456           DATA  M.DMD                                                  VW 33370
3FEF 8423           3457           BYTE  X'80'+MAGI,MAGUX+3         17   MAG TAPE 3             VW 33380
3FF0 3CC6           3458           DATA  M.DMD                                                  VW 33390
                    3459           ENDC                                                         VW 33400
```

391476B

```
RESMON                          PEAT TABLE                09/12/68  PAGE  67

                    3470              FALS  NUNITS-NMAG=0                                                    VW  33510
                    3471              TRUE  NDSK>0                                                         VW  33520
          0004      3472 DSKU1        EQU   DSKU+DSKU+DSKU+DSKU                                             VW  33530
          0010      3473 DSKUX        EQU   DSKU1+DSKU1+DSKU1+DSKU1                                         VW  33540
     3FF1 8310      3474              BYTE  X'80'+DSKI,DSKUX          18    DISK 0                       VW  33550
     3FF2 3CC6      3475              DATA  M.DMD                                                          VW  33560
                    3476              ENDC                                                                 VW  33570
                    3481              FALS  NUNITS-NMAG-NDSK0=0                                            VW  33620
                    3482              FALS  NPCH=0                                                         VW  33630
          0030      3483 PCHU1        EQU   PCHU+PCHU+PCHU+PCHU                                           VW  33640
          00C0      3484 PCHUX        EQU   PCHU1+PCHU1+PCHU1+PCHU1                                       VW  33650
     3FF3 80C0      3485              BYTE  X'80'+PCHI,PCHUX          19    HIGH SPEED TAPE PUNCH        VW  33660
     3FF4 3DB4      3486              DATA  M.THSPT                                                        VW  33670
                    3487              ENDC                                                                 VW  33680
                    3492              FALS  NUNITS-NMAG-NDSK0-NPCH=0                                       VW  33730
                    3493              FALS  NPTR=0                                                         VW  33740
          0034      3494 PTRU1        EQU   PTRU+PTRU+PTRU+PTRU                                           VW  33750
          00D0      3495 PTRUX        EQU   PTRU1+PTRU1+PTRU1+PTRU1                                       VW  33760
     3FF5 80D0      3496              BYTE  X'80'+PTRI,PTRUX          20  HIGH SPEED TAPE READER       VW  33770
     3FF6 3DB4      3497              DATA  M.THSPT                                                        VW  33780
                    3498              ENDC                                                                 VW  33790
                    3503              FALS  NUNITS-NMAG-NDSK0-NPCH-NPTR=0                                  VW  33840
                    3504              FALS  NCDR=0                                                         VW  33850
          0020      3505 CDRU1        EQU   CDRU+CDRU+CDRU+CDRU                                           VW  33860
          0080      3506 CDRUX        EQU   CDRU1+CDRU1+CDRU1+CDRU1                                       VW  33870
     3FF7 8980      3507              BYTE  X'80'+CDRI,CDRUX          21    CARD READER                  VW  33880
     3FF8 3BBA      3508              D     CARDR                                                          VW  33890
                    3509              ENDC                                                                 VW  33900
                    3514              FALS  NUNITS-NMAG-NDSK0-NPTR-NCDR-NPCH=0                             VW  33950
                    3515              FALS  NCPC=0                                                         VW  33960
          003C      3516 CPCU1        EQU   CPCU+CPCU+CPCU+CPCU                                           VW  33970
          00F0      3517 CPCUX        EQU   CPCU1+CPCU1+CPCU1+CPCU1                                       VW  33980
     3FF9 88F0      3518              BYTE  X'80'+CPCI,CPCUX          22    CARD PUNCH                   VW  33990
     3FFA 3AA4      3519              D     CARDP          CARD PUNCH  22  16 HEX                        VW  34000
                    3520              ENDC                                                                 VW  34010
                    3525              FALS  NUNITS-NMAG-NDSK0-NPTR-NCDR-NCPC-NPCH=0                        VW  34060
                    3526              FALS  NLPR=0                                                         VW  34070
          0014      3527 LPRU1        EQU   LPRU+LPRU+LPRU+LPRU                                           VW  34080
          0050      3528 LPRUX        EQU   LPRU1+LPRU1+LPRU1+LPRU1                                       VW  34090
     3FFB 8250      3529              BYTE  X'80'+LPRI,LPRUX          23    LINE PRINTER                 VW  34100
     3FFC 3A62      3530              D     M.LPR          LINE PRINTER 3  23   17 HEX                   VW  34110
```

LUN	Description	
2	PRIN	PRimary INput. PRIN is the unit for input to system programs where the input is neither a directive nor binary; i.e., SYM II accepts the input source text from PRIN. In Figure 1 PRIN points to the card reader.
3	LIST	LISTing. System programs use LIST for tabular output other than error messages. SYM II outputs the assembly listing on the LIST device. In Figure 1 LIST points to the line printer.
4	BIN	Binary INput. BIN is the system binary input unit. In Figure 1 BIN points to the card reader. Absolute or relocatable binary programs are input from this unit.
5	BOUT	Binary OUTput. BOUT is the system binary output unit. In Figure 1 BOUT points to the card punch. SYM II outputs absolute or relocatable binary text to the BOUT unit.
6	SCR	SCRatch. System programs use SCR as temporary storage. SYM II uses SCR to store intermediate text. In Figure 1 SCR points to the disk.
7-11	LOGA, LOGB, LOGC, LOGD, LOGE	The next 5 units, LOGA-E, are intended for users. Devices are divided into two classes; mass and non-mass. A mass device is one that can perform both read and write operations: i.e., disk, magnetic tapes, etc. Non-mass devices can perform only a read or write; i.e., card reader, card punch, etc. LOGA is the systems highest level non-mass input device. LOGB is the systems highest level non-mass output device. LOGC is the system mass unit. LOGD and E are assigned according to the assignment of other devices within the system. The user should look at the PEAT in his I/O monitor listing to determine how these units are assigned.
12	DUMMY	This logical unit performs no I/O but allows programs to be checked out without a peripheral device being available.

The remainder of the logical units are assigned to the various physical devices contained in the system. The following are currently assigned:

LUN	Assignment
13	Console Teletype
14	Magnetic Tape Unit 0
15	Magnetic Tape Unit 1
16	Magnetic Tape Unit 2
17	Magnetic Tape Unit 3
18	Disk Unit 0
19	High Speed Paper Tape Punch
20	High Speed Paper Tape Reader
21	Card Reader
22	Card Punch
23	Line Printer
24	Multiplex Teletype
25	Disk Unit 1
26	Disk Unit 2
27	Disk Unit 3
28	Plotter

Physical assignments for all logical units are made when the monitor is assembled for each system. These assignments are determined by what units are physically present in the system. The user can determine what these initial (or standard) assignments are by referring to the PEAT table as it appears in the listing for his monitor. Following is an example of how the first 12 words of the PEAT might appear:

LOC	Contents		Source Text		Comments Field	
3FD9	1200	3461 PEAT	BYTE	ASYSF,0	0	SYSF
3FDA	0D00	3462	BYTE	ASYSI,0	1	SYSI
3FDB	1500	3463	BYTE	APRIN,0	2	PRIN
3FDC	1700	3464	BYTE	ALIST,0	3	LIST
3FDD	1500	3465	BYTE	ABIN,0	4	BIN
3FDE	1600	3466	BYTE	ABOUT,0	5	BOUT
3FDF	1200	3467	BYTE	ASCR,0	6	SCR
3FE0	1500	3468	BYTE	ALOGA,0	7	LOG A
3FE1	1600	3469	BYTE	ALOGB,0	8	LOG B
3FE2	1200	3470	BYTE	ALOGC,0	9	LOG C
3FE3	0E00	3471	BYTE	ALOGD,0	10	LOG D
3FE4	0F00	3472	BYTE	ALOGE,0	11	LOG E

To determine the assignment of a particular unit, find the unit name (or number) in the comment field. The left byte of the contents field contains the hex unit number of the assignment. Convert it to decimal and look further down the table for that number in the comments field. For example, the device assigned to BIN (logical unit 4) has unit number 15_{16}. (15_{16} is the left byte of the pointer word corresponding to BIN). 15_{16} converts to decimal 21 which is the unit number of the card reader.

1.2.2 PEAT Table Description

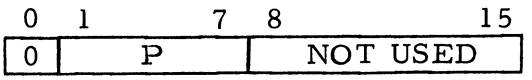
There are two types of entries in the PEAT. There are one word entries which define the correspondence between a logical unit and another logical unit, and there are two word entries which define the correspondence between a logical unit and a physical unit.

Entries within the PEAT are in logical unit number order; i.e., the first entry is for LUN 0, the second entry is for LUN 1, etc.

The table is in two sections. The first eleven entries, section 1, are always pointers to other LUNs. The remaining entries, section 2, are pointers to physical devices.

Numbers 0 - 11 may be considered logical unit numbers, whereas units 12 - 31 may be considered physical unit numbers, since they are not normally reassigned.

In section 1 there is one word per LUN, formatted as follows:

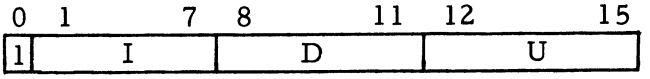


where:

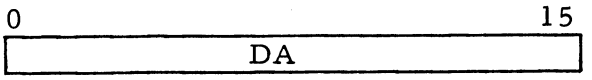
The 0 in bit 0 denotes a pointer to another unit.

P is a 7-bit field pointing to the location, relative to the top of the PEAT table, for the entry of the next LUN.

In section 2 there are two words per entry. If the unit has been reassigned, word one will have the same format as section 1, and word two will be unused. If the LUN has not been reassigned, the format is as follows:



Word 1



where:

The 1 denotes a pointer to a physical unit.

I is a 7-bit field used to denote the priority interrupt level for the device.

D is a 4-bit field used to denote the device code.

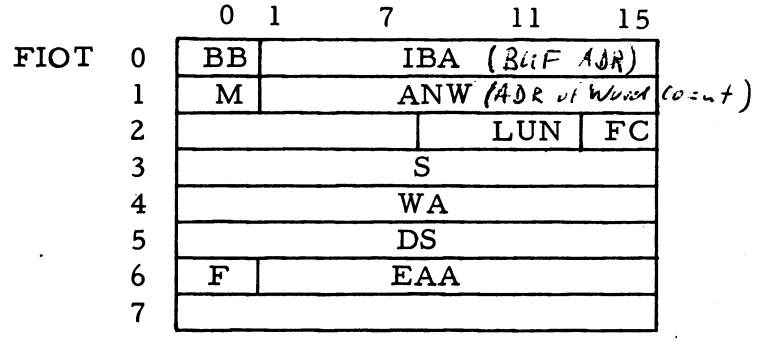
U is a 4-bit field used to denote the unit number of the device number; i.e., magnetic tape may be units 0, 1, 2, 3, etc.

DA is the device driver address for the LUN.

When IOS is seeking the physical unit corresponding to a logical unit, it will go from logical unit to logical unit until it finally arrives at an entry pointing to a physical unit (a 1 in bit position 0).

1.3 FILE INPUT/OUTPUT TABLE (FIOT)

The purpose of the FIOT is to provide user area for I/O operation description and temporary storage for device drivers. How each driver uses the temporary storage is described in the individual driver documentation. Each FIOT is 8 words in length and formatted as follows:



Word 0:

The Busy Bit (BB) is a one-bit field which is set to 1 while the I/O operation is being performed. It will be reset to 0 when the operation has completed. The Initial Buffer Address (IBA) is a 15-bit field which contains the word address of the first location of the data buffer to be transferred.

Word 1:

Mode (M) is a one-bit field which has different meanings for various physical devices and is documented with each device driver. (See 2.1). ANW is a 15-bit field which is the Address of a location which contains the Number of Words to be input or output.

Word 2:

Bits 0 through 6 are temporary storage. The Logical Unit Number (LUN) is a 5-bit field contained in bits 7 through 11. The Function Code (FC) is a 4-bit field contained in bits 12 through 15, and defines the type of operation that is to be performed. Certain function codes perform special functions on specific devices; these are described in the applicable driver documentation. Function codes 9 and B_{16} will perform a read operation for every device capable of a read. Function code E_{16} will perform a write operation on every device capable of a write. Codes 9 and B_{16} are equivalent for every device except the teletype. To preserve device independence codes 9, B, and E should be used for all read and write operations.

Word 3:

Status is the status response word obtained from the device upon completion of each I/O operation. The response differs from device to device and is documented separately for each device. Refer to the appropriate driver documentation. For all devices bit 15 true means that an error has occurred during the I/O operation.

Word 4:

Working Address. Upon completion of the I/O operation word 4 will contain the byte address one greater than the buffer address of the last byte of data transferred. For certain devices this word contains the address of the next byte to be transferred throughout the I/O operation and may be monitored to determine the progress of the operation. Refer to individual driver documentation for details.

Word 5:

Word 5 serves as temporary storage for most drivers. For the disk driver, this word contains the Sector Address to be read or written. When performing device independent I/O, where the disk is a legitimate assignment for the unit, word 5 should be set to the desired sector. Then, if some other device is assigned, the disk sector specification will be ignored. The format for this word as a disk sector is described in the disk driver documentation.

Word 6:

Bit 0 of word six is used for record Format control. When bit 0 is false, those drivers which generate or read formatted records will do so. When bit 0 is true, all drivers generate or read unformatted records. For unformatted records the data in the external medium will be an exact image of the internal buffer. For formatted records certain drivers expect or supply additional formatting data. In most discussions bit 0 is referenced as "the special format bit" and unformatted records are called "special format records". Details of record formatting are described in the applicable driver documentation.

Bits 1 to 15 of word 6 are the End-Action Address (EAA). This address is the address of an end-action subroutine (or program) to which the I/O system will transfer when a particular I/O is completed. If this address is 0, no end action is specified. For a discussion of end action, refer to Section 1.6.

Word 7:

Word 7 is used by most drivers as temporary storage. It has a special use when the magnetic tape chain feature is present in the system, as described in the magnetic tape driver documentation.

1.4 BASIC IOS CALLS

IOS provides the user with 3 basic calls:

a. OPEN
b. DOIO
c. STAT

With these three calls the user can perform an I/O operation on any standard 703 peripheral device.

In the discussion to follow calling sequence for the various I/O subroutine calls are stated in SYM II assembly language format.

```
CALL  ARG1,ARG2...ARGN
```

This will be assembled as though it were written:

```
SMB     CALL
JSX     CALL
DATA    ARG1
DATA    ARG2
.............
DATA    ARGN + X'8000'
```

Note that the sign bit of the last argument is set. For more detail refer to the SYM II manual for an explanation of this form. Entry points to the I/O subroutines are through fixed link points in the System Linkage Table located in cells X'40' to X'7F'. These points must be defined to the assembler using equate cards. A listing of the System Linkage Table appears in Appendix D.

For example, the programmer who writes

```
DOIO   INFIOT
```

must include the definition

```
DOIO   EQU   X'44'
```

in his program. Programs written to execute on page 0 of the computer (locations 80_{16} through $7FF_{16}$) may omit the SMB instruction by using the form:

```
JSX    DOIO, INFIOT
```

Appendix A illustrates a sample input/output program which is written in the SYM I assembly language. The same program written in the SYM II assembly language is shown in Appendix B.

1.4.1 OPEN

The first call for any I/O operation is to subroutine OPEN. OPEN translates the arguments to the proper format and inserts them into the FIOT table.

OPEN has a fixed length calling sequence as follows:

```
OPEN   FIOT, BUF, WC, UNIT, OPER, MODE
```

where: FIOT is the first location of an 8-word array for the file description table.

BUF is the data buffer starting address.

WC is the address of the number of words.

UNIT is the logical unit number.

OPER is the type of operation (read, write, etc.).

MODE defines binary or alpha operation; 1 = binary; 0 = alpha.

It should be noted that the disk sector and the end-action address are not inserted into the FIOT by OPEN. These are less frequently used arguments and were eliminated to shorten the OPEN calling sequence. The user can assemble the arguments into the FIOT or store them at execution time.

The use of the OPEN call is optional, insofar as the FIOT may be assembled in the correct form. However, the OPEN call will initialize the FIOT, setting the busy bit off and clearing the status response word and may provide a convenient way of re-initializing. In addition, OPEN converts the specified logical unit into the actual physical unit number and stores it into bits 7 - 11 of word 2 of the FIOT, so that the using program may determine this assignment before beginning an I/O operation, if required.

1.4.2 DOIO

Actual I/O operations are initiated with calls to DOIO. DOIO changes the FIOT table as required, calls a driver to initiate the I/O operation, and returns to the user.

The calling sequence is as follows:

```
DOIO    FIOT, BUF, WC
```

where: BUF is the data buffer starting address.

WC is the address of the number of words.

DOIO has a variable length calling sequence. If BUF or WC do not change from the previous call or have been set by OPEN, they need not be included. However, if WC is to be changed, BUF must be included.

1.4.3 STAT

STAT provides the user with a means of waiting until an I/O operation is complete. STAT has a variable length calling sequence as follows:

```
STAT    FIOT, ERR
```

When a call to STAT is made, it will loop until the FIOT busy bit is off. If the ERR argument is not included, STAT will make a return to the next instruction following the calling sequence. If argument ERR is in the calling sequence, STAT will load the accumulator with the proper error processor return, and transfer to the ERR address. If the user wished to return from

to the ERR address. If the user wished to return from his ERR routine to the program sequence, he can do this by saving the accumulator as a return, then returning by loading the return into the index and executing a JMP*2 (see sample program in Appendices A and B). If there were no errors, STAT returns to the next instruction following the ERR argument.

1.5 SPECIAL IOS CALLS

The following calls perform special functions; such as Rewind, Write-End-of-File, and Backspace. Not all devices can perform the specified operation; for instance, the teletype will not rewind. If the device cannot perform the specified operation, the subroutine will return with the sign of the accumulator negative. The program can test this condition and determine an appropriate course of action. For example, the Loader accepts input from the BIN device (logical unit 4). If an input error occurs, the Loader backspaces the unit. If the unit was mag tape, it backspaces and the Loader re-reads the record. If the unit is not mag tape and cannot backspace, the Loader requests operator intervention to reposition the record.

The disk unit cannot perform the special functions; however, special calls for the disk return with the accumulator positive, since such functions as backspace and write end-of-file may be simulated by adjusting the sector address in the FIOT and adjusting the disk logical file pointers, which may be done arbitrarily and then allowing the program to drop through the special calls. Refer to the disk driver documentation.

The only information which must be in the FIOT for special operation is the logical unit number (bits 7 - 11 of word 2).

1.5.1 Device Status

The physical status of any I/O device may be interrogated at any time. The call is

```
STUS   FIOT
```

The device's physical status response is returned in the accumulator. The FIOT is not disturbed by this operation, except to substitute the actual physical device number for the logical unit, if this has not already been done. An I/O operation in progress will not be disturbed.

1.5.2 Write End-of-File

A file mark is defined for most I/O devices and may be written on all output devices except the disk, line printer, and plotter. For magnetic tape the file mark is hardware supplied. For paper tape an ASCII "Bell" (X'87') character is the file mark. For cards a 2-7-8-9 punch is the file mark. All input devices upon sensing a file mark conclude the I/O operation and return End-of-File status (bit 10 of the status word true). The write end-of-file call is

```
WEOF    FIOT
```

The FIOT will be busy until the operation is complete.

1.5.3 Seek End-of-File

A file mark is sought on the specified unit. Only the magnetic tape can perform this function. Refer to the magnetic tape driver documentation. The call is

```
SEOF    FIOT
```

1.5.4 Backspace

The specified unit is backspaced one record. Only the magnetic tape can perform this operation. The call is

```
BKSP    FIOT
```

1.5.5 Write-Skip

The specified magnetic tape unit will erase forward three inches of tape. The call is

```
WSKP    FIOT
```

1.5.6 Rewind

The specified magnetic tape unit is rewound. The call is

```
REWD    FIOT
```

1.6 END ACTION

The End Action feature permits program operation to be synchronized to I/O device operation. It allows sharing of time, otherwise wasted during I/O operations, with other tasks. A core resident program can initiate an I/O operation and then exit to allow the system to initiate another task. Then, when the I/O operation is complete, the system will return to the resident program at the end-action address. It can start another I/O operation and then exit. The system will restore the interrupted task.

End action may also be used within a single task; for example, to operate a double buffered I/O function.

End action is requested by specifying a non-zero service address in word 6 of the FIOT. Upon completion of the I/O operation, the I/O system will transfer to that address, after servicing but before restoring the final interrupt of the I/O operation. Then, when the end action subroutine exits, the system automatically restores the interrupted task.

End action subroutines are largely arbitrary in form and function. Any legal operation, including I/O, may be performed as part of an end-action subroutine. The system transfers to the end action address with the return on the index, so that the end action service routine may be in the form of a subroutine:

```
STX     RETSAVE
.
.
.
.
.
LDX     RETSAVE
JSX *   0
```

Of course, the SUBR and EXIT pseudo operations of SYM II may be used.

The end action service routine may also simply exit through the system EXIT link, cell X'40'. The system intercepts this exit and restores the interrupt. Thus, the service routine may have the form of a main program.

There are two restrictions on an end action service routine which affects the initiation of new I/O operations. When the new I/O operation is on the same device or on a device with the same interrupt level, as the device causing the end action, DOIO does not return after initiating the I/O, but rather exits the end action. This frees the interrupt level for the new I/O operation. Practically, this means that the last thing which should be done in an end action service routine is to initiate new I/O for the same device. It is emphasized that this restriction does not apply to I/O for devices on other interrupt levels, which return normally from the DOIO call.

The second restriction applies when the end action service routine is interrupting the interrupt service routine for the device for which the new I/O request is being made. When this occurs, the I/O in progress on that device cannot complete and, therefore, the new I/O operation cannot be initiated. In this event DOIO does not set the FIOT busy bit, but returns a status response of X'00FF' in word 3 of the FIOT. A STAT call with an error return will take the error return for this condition. This condition is virtually unrecoverable; however, it can occur normally only when two users are simultaneously attempting to use the same device . Cautious programming should avoid this situation. A possible solution to this situation, should it occur, would be to initiate a dummy I/O operation on an available device for example, the console typewriter whose end action would attempt to initiate the I/O on the unavailable device.

1.7 DEVICE DRIVER REFERENCES

I.D. Code	Title	Drawing
BPU	Disk and Magnetic Tape Driver	391040
BSU	Card Punch Driver	390018
BSV	Card Reader Driver	390019
BSW	Line Printer Driver	390020
BUU	Teletype Multiplexer Driver	391909
BYM	Teletype - High Speed Paper Tape I/O Driver	392292
BYP	Plotter Driver	392295

```
??
CK
```

APPENDIX A

SYM I SAMPLE PROGRAM

```
  ??
SA

                    *       TEST PROGRAM TO READ A RECORD FROM
                    *       THE BINT UNIT AND TYPE OUT DONE.
                            ORIG  2048
                    READ    EQU   9
                    WRIT    EQU   X'E'
                    BINT    EQU   2
                    BIN     EQU   1
                    ALPH    EQU   0
                    LIST    EQU   3
                    EXIT    EQU   64
                    OPEN    EQU   66
                    DOIO    EQU   68
                    STAT    EQU   70
  0800  0080        STRT    SMB   OPEN
  0801  2042                JSX   OPEN,RFIT,RBUF,RWC,BINT,READ,BIN
* 0802  37FF
*
  0803  37FF
* 0804  37FF
  0805  0001
  0806  0009
  0807  8001

  0808  0080                SMB   OPEN

  0809  2042                JSX   OPEN,TFIT,RBUF,RWC,LIST,WRIT,ALPH

*
  080A  37FF
*
  080B  3003
* 080C  3004
  080D  0003
  080E  000E
  080F  8000

  0810  0080                SMB   DOIO

  0811  2044                JSX   DOIO,RFIT

*
  0812  3802

  0813  0080                SMB   STAT

  0814  2046                JSX   STAT,RFIT,RERR

*
  0815  3012
* 0816  3FFF
```

APPENDIX A
(Continued)

SYM I SAMPLE PROGRAM

```
  0817  0080          SMB   DOIO

  0818  2044          JSX   DOIO,TFIT,FMES,TWO
* 0819  300A
* 081A  37FF
* 081B  3FFF

  081C  0080          SMB   STAT

  081D  2046          JSX   STAT,TFIT
* 081E  3819

  081F  0080          SMB   EXIT

  0820  2040          JSX   EXIT

                RBUF  RES   10
                RFIT  RES   8
  0833  0002    TWO   D     2

* 0834  77FF    RERR  STW   RER

  0835  0080          SMB   DOIO,TFIT

  0836  2044          JSX   DOIO,TFIT
* 0837  381E

* 0838  9034          LDX   RER

  0839  1802          JMP * 2

  083A  0000    RER   DATA  0

                TFIT  RES   8
  0843  C4CF    FMES  D     'DO','NE'
  0844  CEC5                            D O
                                        N E

  0845  000A    RWC   DATA  10

                      END
```

APPENDIX A
(Continued)

SYM I SAMPLE PROGRAM

PSEUDO LISTING

```
0800  0080  STRT
0801  2042
0802  082B
0803  0821
0804  0846
0805  0001
0806  0009
0807  8001
0808  0080
0809  2042
080A  083B
080B  0821
080C  0846
080D  0003
080E  000E
080F  8000
0810  0080
0811  2044
0812  882B
0813  0080
0814  2046
0815  082B
0816  8834
0817  0080
0818  2044
0819  083B
081A  0843
081B  8833
081C  0080
081D  2046
081E  883B
081F  0080
0820  2040
0821  0000  RBUF
0822  0000
0823  0000
0824  0000
0825  0000
0826  0000
0827  0000
0828  0000
0829  0000
082A  0000
082B  0000  RFIT
082C  0000
082D  0000
082E  0000
082F  0000
0830  0000
0831  0000
0832  0000
0833  0002  TWO
0834  703A  RERR
0835  0080
0836  2044
0837  883B
0838  903A
0839  1802
083A  0000  RER
083B  0000  TFIT
083C  0000
083D  0000
083E  0000
083F  0000
0840  0000
0841  0000
0842  0000
0843  C4CF  FMES
0844  CEC5
0845  000A  RWC
```

APPENDIX B

SYM II SAMPLE PROGRAM

```
                    1 *         TEST PROGRAM TO READ A RECORD FROM
                    2 *         THE BINT UNIT AND TYPE OUT DONE.
                    3           ORIG  2048
         0009       4 READ      EQU   9
         000E       5 WRIT      EQU   X'E'
         0001       6 BINT      EQU   1
         0001       7 BIN       EQU   1
         0000       8 ALPH      EQU   0
         0003       9 LIST      EQU   3
         0040      10 DONE      EQU   64
         0042      11 OPEN      EQU   66
         0044      12 DOIO      EQU   68
         0046      13 STAT      EQU   70
  0800 0080        14 STRT      OPEN  RFIT,RBUF,RWC,BINT,READ,BIN
S 0801 2042
  0802 082B
  0803 0821
  0804 0844
  0805 0001
  0806 0009
  0807 8001
  0808 0080        15           OPEN  TFIT,RBUF,RWC,LIST,WRIT,ALPH
S 0809 2042
  080A 083A
  080B 0821
  080C 0844
  080D 0003
  080E 000E
  080F 8000
  0810 0080        16           DOIO  RFIT
S 0811 2044
  0812 882B
  0813 0080        17           STAT  RFIT,RERR
S 0814 2046
  0815 082B
  0816 8833
  0817 0080        18           DOIO  TFIT,FMES,=2
S 0818 2044
  0819 083A
  081A 0842
  081B 8845
  081C 0080        19           STAT  TFIT
S 081D 2046
  081E 883A
  081F 0080        20           DONE
S 0820 2040
  0821             21 RBUF      RES   10
  082B             22 RFIT      RES   8
  0833 7039        23 RERR      STW   RER
  0834 0080        24           DOIO  TFIT
S 0835 2044
  0836 883A
  0837 9039        25           LDX   RER
  0838 1802        26           JMP * 2
  0839 0000        27 RER       D     0
  083A             28 TFIT      RES   8
```

APPENDIX B
(Continued)
SYM II SAMPLE PROGRAM

```
0842 C4CF    29 FMES     TEXT  'DONE'
0843 CEC5
0844 000A    30 RWC      D     10
     0800    31          END   STRT

0845 0002


NO ERRORS


ALPH     0000  BIN      0001  BINT     0001  DOIO     0044
DONE     0040  FMES     0842  LIST     0003  OPEN     0042
RBUF     0821  READ     0009  RER      0839  RERR     0833
RFIT     082B  RWC      0844  STAT     0046  STRT     0800
TFIT     083A  WRIT     000E


NEXT
```

APPENDIX C

703 Character Codes

Internal Code	Graphic	Hollerith Card Code
A0	blank	blank
A1	!	11-2-8
A2	"	0-5-8
A3	#	0-7-8
A4	$	11-3-8
A5	%	11-7-8
A6	&	12-2-8
A7	'	4-8
A8	(	0-4-8
A9	)	12-4-8
AA	*	11-4-8
AB	+	12
AC	,	0-3-8
AD	-	11
AE	.	12-3-8
AF	/	0-1
B0	0	0
B1	1	1
B2	2	2
B3	3	3
B4	4	4
B5	5	5
B6	6	6
B7	7	7
B8	8	8
B9	9	9
BA	:	5-8
BB	;	11-6-8
BC	<	12-6-8
BD	=	3-8
BE	>	6-8
BF	?	12-2-8

Internal Code	Graphic	Hollerith Card Code
C0	@	0-2-8
C1	A	12-1
C2	B	12-2
C3	C	12-3
C4	D	12-4
C5	E	12-5
C6	F	12-6
C7	G	12-7
C8	H	12-8
C9	I	12-9
CA	J	11-1
CB	K	11-2
CC	L	11-3
CD	M	11-4
CE	N	11-5
CF	O	11-6
D0	P	11-7
D1	Q	11-8
D2	R	11-9
D3	S	0-2
D4	T	0-3
D5	U	0-4
D6	V	0-5
D7	W	0-6
D8	X	0-7
D9	Y	0-8
DA	Z	0-9
DB	[	12-5-8
DC	\	0-6-8
DD	]	11-5-8
DE	↑	7-8
DF	←	2-8

APPENDIX D

System Linkage Table

Cell (Base 16)	Cell Use
40	EXIT
42	OPEN
44	DOIO
46	STAT
48	BKSP
4A	WKSP
4C	SEOF
4E	RWND
50	WEOF
52	STUS
54	END Lower limit of available core (end X-RAY in Basic)
55	RBEG Restart address for RELO programs
56	Address of interrupt routine for TTY
57	ENB
58	STYP System type word
59	PEAT Address of system Peat Table
5A	ULIM Upper limit of available core*
5B	ENDM Upper or lower limit of Basic or Standard I/O Subsystems, respectively
5C	CORE Address of last cell in core
5D	DATE
61	NAME
65	JMP*0
66	LLIM Lowest cell loaded by loader
67	JOB Control Word
68	Unassigned
69	Unassigned
6A	QUEU Monitor Queue Subroutine
6C	QFCH Monitor Queue Fetch Argument Subroutine
6E	TYPE Type message subroutine
70	MTSF Masked test system flag word
72	LOAD Load a processor
74	DVEC Disk Vector Pointer
75	ULIM2 Upper limit of available core*
76	P.MAP Processor Map sector on disk
77	LOAD Load X-RAY Flag
78	ABSLOAD Absolute program load
7A	RLRS RELO program restart
7C	RESTL Resident task table linkage
7D	PFS Power Fail Safe Linkage
7F	RDTF Resident Disk Table size

*(Below resident task, blocks, etc.)