

Application Note

USER DEVICE DRIVER IMPLEMENTATION IN THE REAL TIME DISK OPERATING SYSTEM

017-000002-03

Ordering No. 017-000002

© Data General Corporation 1972, 1974, 1975

All Rights Reserved.

Printed in the United States of America

Rev. 03, February 1975

NOTICE

Data General Corporation (DGC) has prepared this manual for use by DGC personnel, licensees and customers. The information contained herein is the property of DGC and shall neither be reproduced in whole or in part without DGC prior written approval.

DGC reserves the right to make changes without notice in the specifications and materials contained herein and shall not be responsible for any damages (including consequential) caused by reliance on the materials presented, including but not limited to typographical, arithmetic, or listing errors.

(RDOS 00)	Original Release	-	September, 1972
(RDOS 01)	First Revision	-	November, 1972
(RDOS 02 & 03)	Second Revision	-	January, 1974
(RDOS 04)	Third Revision	-	February, 1975

This revision of application note 017-000002 is a major revision. A vertical bar on the outside margin of each page indicates substantially new, changed or deleted information.

TABLE OF CONTENTS

CHAPTER 1 - RDOS DRIVER IMPLEMENTATION

Introduction	1-1
Characteristics of Interrupts	1-1
Interrupt Priority Scheme	1-2
Interrupt Dispatch Program	1-4
Device Control Table (DCT) Structure	1-7
User File Table (UFT) Structure	1-11

CHAPTER 2 - OPERATING SYSTEM DEVICE IMPLEMENTATION

General	2-1
Reentrant Coding and Subroutine Linkage	2-3
General Subroutine Package (GSUB, MGSUB, BGSUB, AGSUB)	2-7
Save State Variables	2-11
Restore State Variables, Return a Frame	2-11
Move a Byte String	2-11
Load/Store a Byte	2-11
Exclusive Or	2-12
Clear a Block of Core	2-12
Move a Word String	2-12
Compare Two Word Strings	2-12
Single/Double Precision Integer Divide	2-13
Set/Reset Bits in a Word	2-13
Mapped Move a Byte String	2-13
Mapped Load and Store Bytes	2-14
Mapped Move a Word String	2-14
Mapped Clear a Block of Core	2-15
Generalized I/O Routines	2-16
Open (OPNO, OPNI)	2-20
Close (CLSO, CLSI)	2-21
Read Sequential (RDS)	2-21
Read Line (RDL)	2-22
Write Sequential (WRS)	2-23
Write a Line (WRL)	2-23
Get Master or Default Directory Name (GMDIR/ GDIRS)	2-24
I/O Buffer Module (IOBUF)	2-25
Common Input Device Interrupt Service (CISER)	2-26
Common Output Device Interrupt Service (COSER, STOUT)	2-27
Add a Bead to the String (ENQUE)	2-27

CHAPTER 2 - OPERATING SYSTEM DEVICE IMPLEMENTATION (Continued)

Terminate Bead from Head of List (DEQRQ)	2-27
Remove a Bead from the String (DEQUE)	2-28
Partial Input Interrupt Service (FINP)	2-28
Input a Character to the I/O Buffer (IBUF).	2-29
Priority Enqueue a Bead (PENQU)	2-29
Priority Enqueue a Bead and Input to an I/O Buffer (XIBUF)	2-29
Output a Character from the I/O Buffer (OBUF)	2-30
I/O Buffer Management	2-30
Retrieve a Character from the I/O Buffer (RCHR)	2-31
Add a Character to the I/O Buffer (ACHR)	3-32
Declaring the DCT Address	2-32
Creating a Peripheral Device Entry in a Directory	2-33
Updating the System Libraries	2-34
Creating a System Queue Entry	2-37
System Generation	2-37
Practical Hints for System Device Driver Implementation	2-38
Elements Required in User-Written I/O Routines	2-38
Examination of a System Device Driver	2-39

CHAPTER 3 - USER PROGRAM SERVICED INTERRUPTS

Servicing User Interrupts	3-1
User Device Driver Implementation at Run Time	3-1
Identify a User Interrupt Device (.IDEF)	3-2
Exit from a User Interrupt Routine (.UIEX)	3-4
Remove User Interrupt Servicing Program (.IRMV)	3-4
Set the Data Channel Map (.STMAP)	3-5
Modifying the Current Interrupt Mask (.SMSK)	3-6
Writing User Power Fail Service	3-6
Exit from a Power Fail Service Routine (.UPEX)	3-8
User Programmed Clock	3-8
Define a User Clock (.DUCLK)	3-8
Remove a User Clock (.RUCLK)	3-9
Examples of User Serviced Interrupts	3-10
Analog to Digital Converter	3-10
External Interrupt Recognition	3-16
Multiprocessor Communications Adapter	3-20

CHAPTER 1

RDOS DRIVER IMPLEMENTATION

INTRODUCTION

In all real time computer control systems, the CPU reacts to input data from a real world environment and provides output data to correct or control the environment. The incoming data is normally the result of a process device interrupt or an input-output operation completion interrupt. In general these interrupts differ from one another only in the way in which they are serviced.

When a significant event occurs, a signal is transmitted to the computer as an interrupt requiring a special subroutine to take appropriate action. Interrupts are usually assigned in order of urgency or priority, so that if two interrupts occur at the same moment, the more important interrupt is serviced first by the computer.

How well a computer is able to respond to interrupts generally determines the maximum capability of the real time system. A significant element in the responsive ability of any real time system is the inclusion of a powerful and flexible multi-priority interrupt control program.

This document outlines the methods of adding customer-assignable multi-priority servicing routines. These device drivers can be included as part of the resident RDOS operating system or they can be implemented as part of the user application program and attached to the interrupt dispatch program.

Statements in this manual which refer to RDOS can, unless otherwise qualified, be understood to apply to both mapped and unmapped versions of the operating system running in either NOVA® * or ECLIPSE™ ** computer environments.

CHARACTERISTICS OF INTERRUPTS

Interrupts can be generated by conditions internal to the computer hardware itself (power monitor option), or by events which originate in the plant or the environment that is being controlled (an external hardware interrupt like an analog-to-digital converter completion).

*NOVA is a registered trademark of Data General Corporation, Southboro, Massachusetts.

**ECLIPSE is a trademark of Data General Corporation, Southboro, Massachusetts.

CHARACTERISTICS OF INTERRUPTS (Continued)

Non-process interrupts may be caused by an error condition being detected, an interval timer run-down, an input-output (I/O) complete interrupt, etc. The I/O interrupt is characterized by the completion of an I/O device operation such as a paper tape or disk storage transfer function, which may involve the reading or writing of one character or word in the case of program control or multiple words in the case of data channel operation. The concept of this type of interrupt is based on the importance of keeping I/O devices active, thus improving job throughput.

Process interrupts reflect process conditions which have been detected and which require an immediate change in program execution, such as may be caused by the closing of an electrical switch or contact, a rise in temperature above a prescribed limit, etc.

INTERRUPT PRIORITY SCHEME

There are several ways in which priorities are determined for or assigned to devices on the I/O bus. An elementary priority is established by the hardware for devices that are requesting interrupts simultaneously: among those devices waiting with an active interrupt, the one which is physically closest to the processor on the bus is at the highest priority.

The most significant method is to specify which devices can interrupt a service routine currently in progress. This is done by using a 16 bit priority mask. Each device is wired to a particular bit of this mask word.

By means of the mask word, the interrupt servicing program can inhibit specified levels of interrupts and thus establish any priority structure. The biggest advantage of this means of priority level control is a near optimum priority response. To guarantee minimum response time to an interrupt, the mask bit assigned to this device should not be set for long periods of time. Through the judicious use of masking, data channels can be kept functioning for the transmission of data into and out of core storage while process interrupts are prevented from occurring. The function of masking is used to delay recognition of an interrupt (the important fact is that the interrupt is not lost).

INTERRUPT PRIORITY SCHEME (Continued)

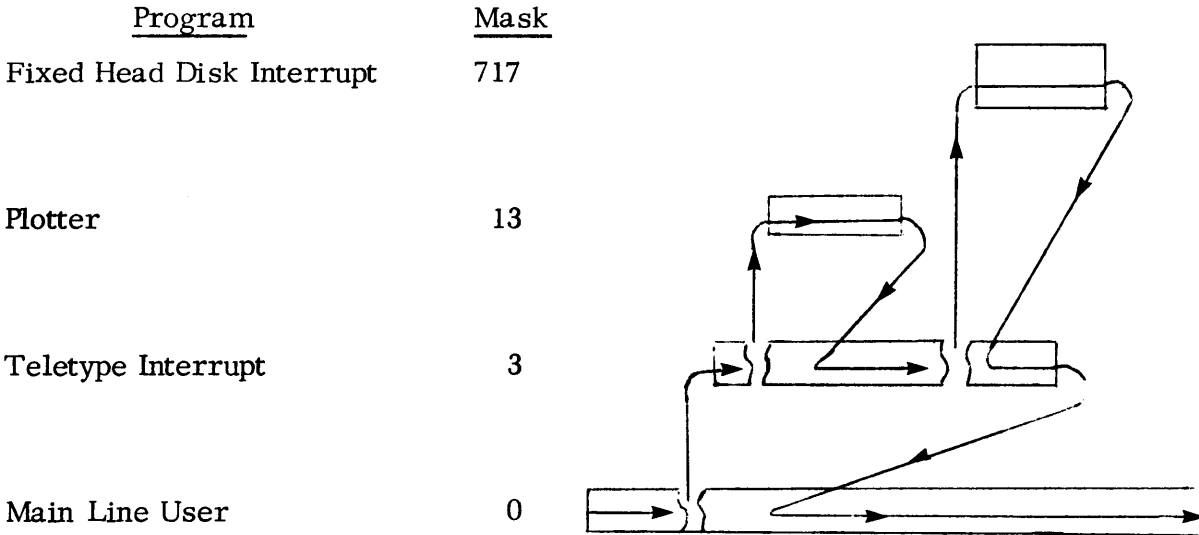
The interrupt mask assignments for standard devices supplied by DGC are as follows:

<u>Mask</u>	<u>Assigned Devices</u>
3	\$TTO, TTI, QTY (4060 quad multiplexer)
7	\$PTP
17	\$PLT, MCAT (4038 multiprocessor communications adapter transmitter), MCAR (4038 multiprocessor communications adapter receiver)
10	\$LPT
217	\$DPO, \$DP1 (IPB dual processor full duplex link)
617	moving head disk
717	fixed head disk
1717	IPB half-duplex directory interlock
1737	\$PTR
1777	\$CDR, magnetic tape, cassette

Although slower devices are assigned to higher numbered bits in the mask, there is no established priority, since the program can use any mask configuration. All devices which have their mask bits set will not cause an interrupt and are therefore regarded by the program as being of lower priority.

INTERRUPT PRIORITY SCHEME (Continued)

An example of multi-level priority interrupt servicing is shown below for an environment including a Teletype® *, plotter and fixed head disk.



INTERRUPT DISPATCH PROGRAM

At the precise moment an unmasked interrupt has been detected at the hardware level, the Interrupt Dispatch program (INTD) receives control to service the interrupt. The INTD program is assembled as an integral portion of the RDOS system and resides in core at all times.

The INTD program is designed to do the following:

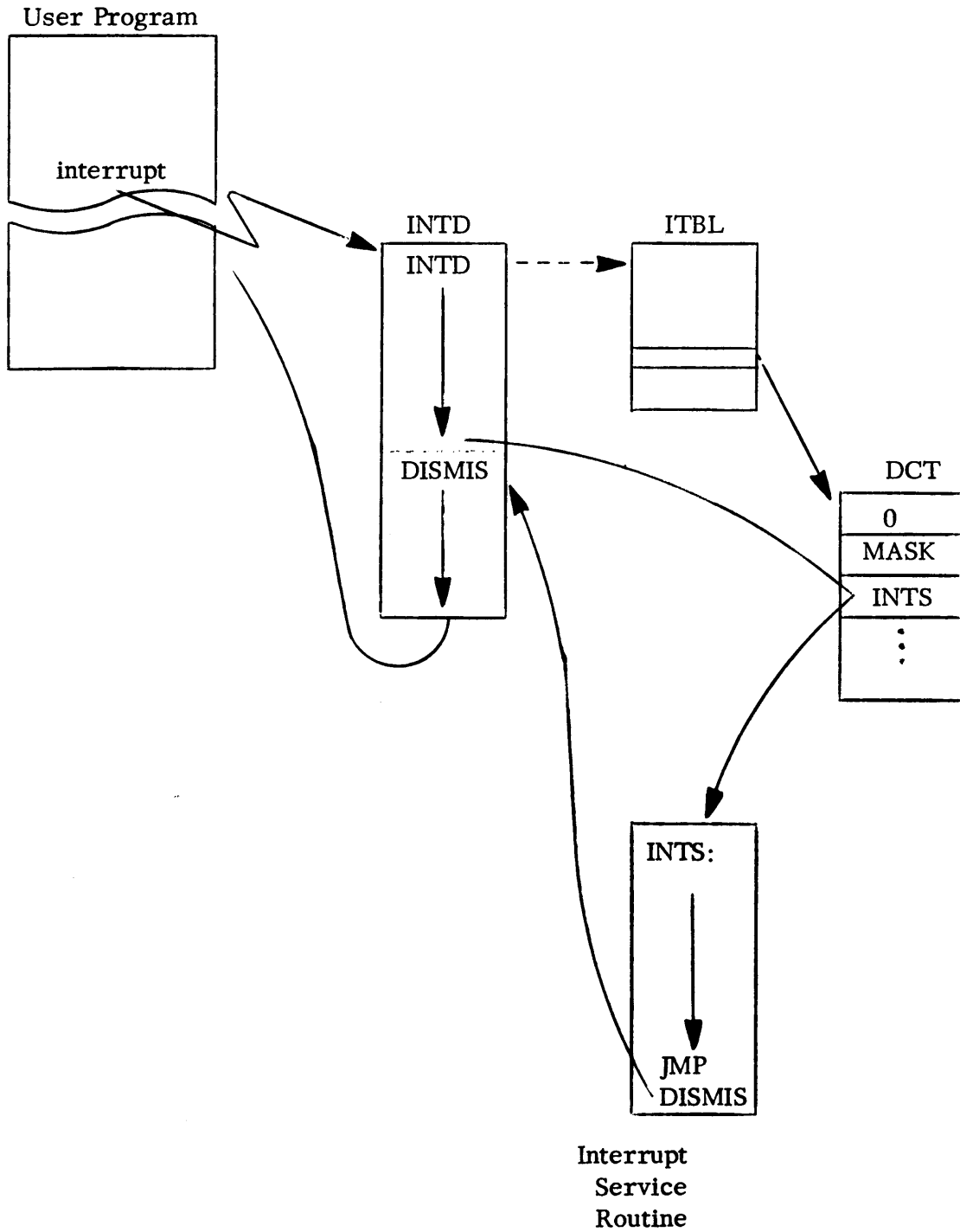
- Identify the interrupting device**
- Save the machine status and all** addressable registers
- Set up the new priority mask**
- Direct control to the proper servicing routine **
- Restore the previous priority mask **
- Restore the machine status and registers
- Return to the interrupted program

*Teletype is a registered trademark of Teletype Corporation, Skokie, Illinois.

** This function is performed by the VCT instruction on ECLIPSE systems.

INTERRUPT DISPATCH PROGRAM (Continued)

The INTD program directs control to the correct servicing routine by using the correct entry in the system Interrupt Branch Table (ITBL). ITBL is a 66₁₀ word table, with displacements corresponding to device codes 1-63. That is, device code 1 is found at displacement 1 (i. e., at "ITBL"). Each displacement corresponding to a device in the system contains the address of that device's DCT. All other displacements contain -1.



Flow of Control During Interrupts

DEVICE CONTROL TABLE (DCT) STRUCTURE

Each device defined in the RDOS system must have the address of its DCT included in the interrupt branch table (ITBL) as explained in later sections. For interrupt servicing routines defined in the user's area, only the first three entries are needed. Words 7-26 of the DCT differ for byte and block oriented devices.

Word 0, DCTBS: In revisions prior to 04, this displacement continued the address of an eight word interrupt state save area not used by RDOS but reserved for compatibility with RTOS. In revision 04, bit zero is set to 1 if this device uses the data channel.

Word 1, DCTMS: Mask Word. Clear a bit for every priority considered higher than the priority of this device. The devices corresponding to the priority bits that are left cleared will be permitted to interrupt the current device. For a complete set of system mask words, see the list provided previously in INTERRUPT PRIORITY SCHEME.

Word 2, DCTIS: Address of the device interrupt service routine.

Word 3, DCTCH: Device characteristic word. A list of device characteristics is given in the following table.

<u>MNEMONIC</u>	<u>BIT</u>	<u>MEANING</u>
DCCPO	15	Device requiring leader/trailer
DCCGN	14	Device requiring tab simulation
DCIDI	13	Device requiring operator intervention
DCCNF	12	Device requiring form feed simulation
DCTO	11	Teletype output device
DCKEY	10	Keyboard input device (uncontrollable)
DCNAF	9	Device requiring nulls after form feeds
DCRAT	8	Device requiring rubouts after tabs
DCPCK	7	Device requiring even parity check on input, even parity computation on output
DCLAC	6	Device requiring line feeds after carriage returns
DCSPO	5	Spoolable device (LPT, PTP, PLT, TTO)
DCFWD	4	Full word device (any size greater than a byte).
DCFFO	3	Form feed sent on .OPEN
DCLTU	2	Convert lower to upper case ASCII
DCC80	1	Read 80 columns on input if set, 72 if reset. Send 80 characters on output, 72 if reset.
DCSPC	0	Spooling enabled if set to 1, disabled if reset

DEVICE CONTROL TABLE (DCT) STRUCTURE (Continued)

- Word 4, DCTCD: Device code.
- Word 5, DCTEX: Address of variable I/O instruction routine (reserved but unused on ECLIPSE systems.)
- Word 6, DCTDT: Address of the device command dispatch table; bit 0 set implies disk device. One entry for every RDOS I/O function. The table order must correspond exactly to the order of the function given below. Each entry is an address to the routine implementing the named RDOS function. If the device does not permit a command, a -1 should be entered in place of the address.

<u>MNEMONIC</u>	<u>DISPLACEMENT</u>	<u>MEANING</u>
OF	0	Open a file for reading/writing by one or more users.
CF	1	Close a file.
RS	2	Read Sequential
RL	3	Read Line
RR	4	Read Random
WS	5	Write Sequential
WL	6	Write Line
WR	7	Write Random
OA	10	Open a file for appending
RO	11	Open for reading only
EO	12	Exclusive read/write open
TO	13	Transparent (exclusive) open (does not alter file access info)

The following words, 7-26, are used with byte-oriented devices.

- Word 7, DCTST: Address of the device start routine. The device start routine specification is as follows:
- Input device: Activate the device and return
Output device: Character is passed in AC0
- Word 10, DCTBC: Size of device buffer in bytes.
- Word 11, DCTBP: Byte pointer to device buffer.
- Word 12, DCTPC: Base level (program) byte count.

DEVICE CONTROL TABLE (DCT) STRUCTURE (Continued)

Word 13, DCTPP:	Base level (program) byte pointer.
Word 14, DCTQL:	Link to device request bead chain.
Word 15, DCTDP:	Device data byte pointer.
Word 16, DCTDC:	Device data count.
Word 17, DCTQS:	Bead status word (described in GENERALIZED I/O ROUTINES).
Word 20, DCTBD:	Bead address (i. e., the starting address of the first bead, word 14)
Word 21, DCTQP:	Device queue starting address (initially -1).
Word 22, DCTT1:	First temporary for device control.
Word 23, DCTT2:	Second temporary for device control.
Word 24, DCTTO:	Timeout constant for input devices.
Word 25, DCTCC:	Output device column counter.
(Word 25, DCTPR:	Echo device DCT pair pointer, TTI only.)
Word 26, DCTLCL:	Output device line counter.
(Word 26, DCTLK:	Link to <u>TTR</u> , <u>TTI</u> <u>DCT</u> s only.) *
Word 27, DCTON:	"ON" DCB address for spooler. *
Word 30, DCTOF:	"OFF" DCB address for spooler. *
Word 31, DCTSL:	Link to next spoolable device DCT (-1 terminates the chain).
Word 32, DCTOP:	Device queue starting address for operator messages.
Word 33, DCTT3:	Temporary for operator message status word whose bits are defined as follows:

*set by RDOS.

DEVICE CONTROL TABLE (DCT) STRUCTURE (Continued)

<u>Bit</u>	<u>Meaning</u>
0	Set to 1 if "!" has been received.
15	Set if "F" or "B" has been received.

The following words, 7-26, are used with block transfer devices.

Word 7, DCTRD:	Read a Block.
Word 10, DCNB1:	High order portion of number of blocks on the device.
Word 11, DCSTR:	Device start routine.
Word 12, DCDST:	Set DST word (used for logical -to-physical disk block address computation).
Word 13, DCCRQ:	Current request pointer.
Word 14, DCTSZ:	Routine to perform disk sizing (not used by MTA/CAS).
Word 15, DCTRL:	Read last block.
Word 16, DCTRN:	Read next block.
Word 17, DCTIN:	Device initialization routine.
Word 20, DCTRS:	Device release routine.
Word 21, DCNBK:	Number of blocks on device (low order portion; see DCNB1).
Word 22, DCTNS:	Number of sectors per track.
Word 23, DCTNH:	Number of heads per unit.
Word 24, DCTMN:	Hash frame size (one of the following):

<u>Moving Head Disk Type</u>	<u>Frame Size (octal)</u>
4047	103
4048	155
4057	255
4231	255

DEVICE CONTROL TABLE (DCT) STRUCTURE (Continued)

<u>Aggregate Fixed Head Storage</u>	<u>Frame Size (decimal)</u>
128K	7
256K	13
384K	23
512K	31
640K	41
768K	47
896K	53
1024K	61
1152K	71
1280K	79
1480K	89
1536K	97
1664K	103
1792K	113
1920K	113
2048K	127

(For a description of hashing under RDOS, see the Stand Alone Disk Editor manual, 093-000092, Appendix B.)

- Word 25, DCHMP: First slot in data channel map (mapped systems only).
- Word 26, DCHNM: Number of slots needed in data channel map (mapped systems only).

USER FILE TABLE (UFT) STRUCTURE

User File Tables (UFTs) are tables used by the operating system to record file and device information for each file or device that is opened. There is one UFT for each channel defined to be available to a user program (via the .COMM TASK statement or RLDR switch /C). These tables reside in system space in mapped environments, and between NMAX and the operating system in unmapped single program environments (see Chapter 6 of the RDOS Reference Manual for further information about the location of these tables). UFT structure is defined in PARU. A subsection of each UFT, termed the UFD, is also defined in PARU; it consists of words 0-21₈ of each UFT. The structure and mnemonic assignments of UFTs are as follows:

Words 0 to 4, UFTFN: File name.
 Word 5, UFTEX: File extension.
 Word 6, UFTAT: File attributes.
 Word 7, UFTLK: Link access attributes word.
 Word 10, UFTBK: Number of the last block in the file (disk file).
 Word 11, UFTBC: Number of bytes in last block (disk file).
 Word 12, UFTAD: Device address of first block (disk file).
 Word 13, UFTAC: Year/day last accessed.
 Word 14, UFTYD: Year/day created.
 (Word 14, UFLAN, links only: Link alias name.)
 Word 15, UFTHM Hour/minute created.
 Word 16, UFTP1: Temporary location used by the system.
 Word 17, UFTP2: Temporary location used by the system.
 Word 20, UFTUC: File use count.
 Word 21, UFTDL: Link to DCT.
 Word 22, UFTDC: DCT address.
 Word 23, UFTUN: Unit number (disk file).
 Word 24, UFCA1: Current block address high order (disk file).
 Word 25, UFTCA: Current block address, low order (disk file).
 Word 26, UFTCB: Current block number (disk file).
 Word 27, UFTST: File status:

STER	1B15	ERROR DETECTED
STIOP	1B14	I/O IN PROGRESS
STFWR	1B13	FIRST WRITE FLAG
STDIU	1B12	DIRECTORY IN USE
STRWD	1B09	OPENED FOR MTA R/W BLOCK
ST7T	1B08	7 TRACK MAG TAPE
STINI	1B01	NO INIT FLAG
STCMK	1B00	SET = READ (BLKIO)
		INIT/RELEASE BIT
STOPN	1B2	DCB OPENED (MTA)

Word 30, UFEA1: Entry's block address, high order (disk file).
 Word 31, UFTEA: Entry's block address, low order (disk file).
 Word 32, UFNA1: Next block address, high order (disk file).
 Word 33, UFTNA: Next block address, low order (disk file).
 Word 34, UFLA1: Last block address, high order (disk file).
 Word 35, UFTLA: Last block address, low order (disk file).
 Word 36, UFTDR: SYS.DR DCB address.
 Word 37, UFFA1: First address, high order, (disk file).
 Word 40, UFTFA: First address, low order, (disk file).

Displacements in UFTs which refer to disk files are set to zero when the opened file described by the UFT is not a disk file.

CHAPTER 2

OPERATING SYSTEM DEVICE IMPLEMENTATION

GENERAL

Device driver support can be provided by the operating system in one of two ways:

- 1) by incorporating the driver into the operating system at source program level;
- 2) by identifying the driver to the operating system at run time via system call .IDEF.

The first of these means is the most efficient means of providing device interrupt service, since the same routines and data bases used by system devices are made available to the user device. This method requires a detailed understanding of system data bases, service routines, and device driver structure. These items are described in Chapters 1 and 2.

The second method, described in Chapter 3, is somewhat less efficient since it entails a duplication of certain functions by the user which are already provided by the system. Nonetheless, this method is comparatively simple to use, and it can be applied to both character and data channel devices.

The following is a list of I/O devices, their names and their device codes, that are standard system devices.

<u>Device Name</u>	<u>Device</u>	<u>Device Code</u>
\$CDR	card reader, primary controller	16
\$CDR1	card reader, secondary controller	56
CT _n	DGC cassette unit <u>n</u> , primary controller	34
CT1 _n	DGC cassette unit <u>n</u> , secondary controller	74
DK0	DGC NOVADISC, primary controller	20
DK1	DGC NOVADISC, secondary controller	60
DPh	moving head disk, n = 0 -3; primary controller	33
DP _n	moving head disk, n = 4 -7; secondary controller	73
\$DPI	input dual processor link	40
\$DPO	output dual processor link	41
\$LPT	line printer	17
\$LPT1	second line printer	57
MCAR	first multiprocessor communications adapter receiver	7
MCAR1	second multiprocessor communications adapter receiver	47

<u>Device Name</u>	<u>Device</u>	<u>Device Code</u>
MCAT	first multiprocessor communications adapter transmitter	6
MCAT1	second multiprocessor communications adapter transmitter	46
MT _n	magnetic tape unit <u>n</u> , primary controller	22
MT _{1n}	magnetic tape unit <u>n</u> , secondary controller	62
\$PLT	incremental plotter	15
\$PLT1	second incremental plotter	55
\$PTP	high speed paper tape punch	13
\$PTP1	second high speed paper tape punch	53
\$PTR	high speed paper tape reader	12
\$PTR1	second high speed paper tape reader	52
QTY	type 4060 asynchronous data multiplexor	30
RTC	real time clock	14
\$TTR/\$TTI	Teletype reader/keyboard or display terminal keyboard	10
\$TTR1/\$TTI1	second Teletype reader/keyboard or display terminal keyboard	50
\$TTP/\$TTO	Teletype printer/punch or terminal screen	11
\$TTP1/\$TTO1	second Teletype printer/punch or terminal screen	51

Device code 2 is not available to mapped RDOS (MRDOS) user devices, and device codes 14 and 36 are not available for use by any user devices selected to run under RDOS or MRDOS.

Changes to RDOS to incorporate another device into the operating system can be made at the source program level. This section describes briefly the required changes to add a device. However, to completely understand the process of adding a device driver, the user may need to review listings of those RDOS programs mentioned later that require modification. The source programs required are available from DGC.

During full initialization of the system, names of all single-file peripheral devices in the system are added to the system file directory, SYS.DR. The device name will be entered only if the user has forced the driver to be loaded.

The general procedure for adding a device driver at source level is as follows:

- . declare a DCT entry for the device
- . enable entry of the device name in SYS.DR
- . update the system library
- . create a system queue entry
- . perform a system generation and load.

The following sections describe each of these steps.

REENTRANT CODING AND SUBROUTINE LINKAGE

One of the basic problems that arises in multilevel priority interrupt programming is that different levels require use of the same subroutine. This means that a higher priority interrupt could interrupt a lower priority one before it has completely used a common subroutine and argument pointers, temporary core storage locations, etc., with the result that the return point for the lower level interrupt would be lost. Reentrancy, then, is defined as that property of a subroutine where use of the routine does not modify any of its locations; temporary values are stored outside the routine. Thus one user may be prevented temporarily from completing use of a reentrant routine because a higher priority user needs to use it. When the higher priority user has finished with the routine, the lower priority user completes his use of the routine at the point where he was interrupted.

A method of reentrant coding using a system stack has been devised for RDOS to allow one subroutine to be entered at any time and from any interrupt level without loss of results. The operating system has several stacks used for the saving of state variables whenever a call to a system subroutine is executed. The structure of NOVA RDOS stacks and stack frame differs from those employed on ECLIPSE systems. On NOVA systems, each of these stacks is of a fixed length and stack frames are defined in the same manner for each stack. The stack frame is the basic increment of storage on a system stack.

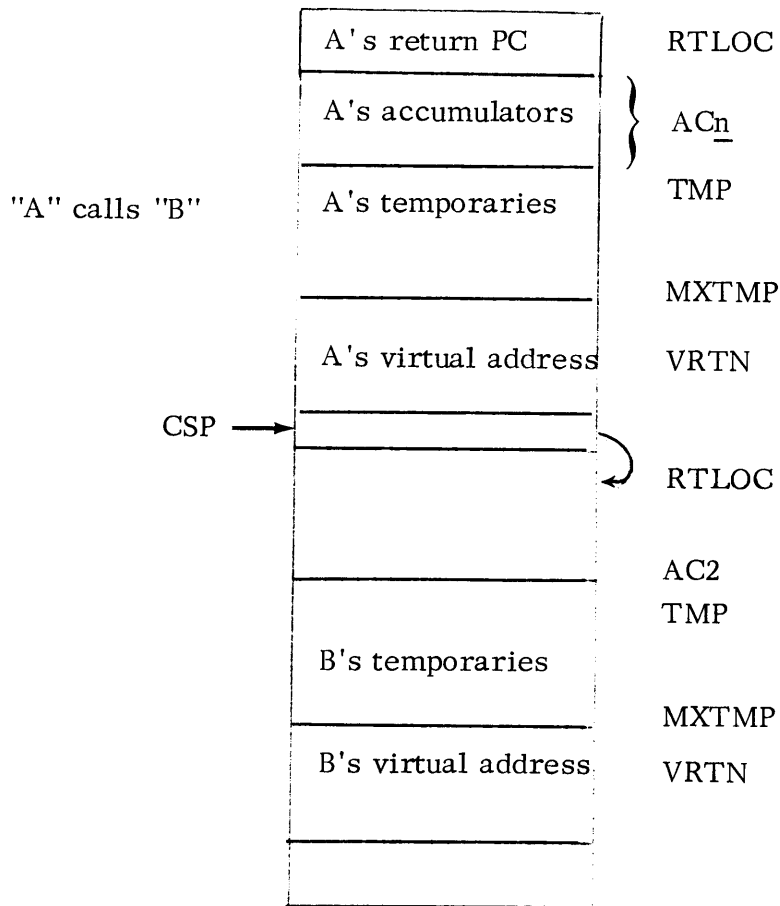
REENTRANT CODING AND SUBROUTINE LINKAGE (Continued)

Each NOVA system stack consists of 10 octal frames, each stack frame is 16 octal locations in length, and the locations in each frame have the following definitions:

<u>Displacement</u>	<u>Mnemonic</u>	<u>Contents</u>
-1	SP	Pointer to the beginning of this stack frame (i. e. , to RTLOC)
0	RTLOC	Return location.
1	AC0	AC0 contents if this routine calls another routine.
2	AC1	AC1 contents if this routine calls another routine.
3	AC2	AC2 contents if this routine calls another routine.
4	TMP	First general purpose storage location for use by this routine.
.		
.		
.		
13	MXTMP	Eighth and last general purpose storage location.
14	VRTN	Overlay virtual address if the caller is a system overlay.

REENTRANT CODING AND SUBROUTINE LINKAGE (Continued)

Thus if system subroutine A called system subroutine B, a portion of the system stack would contain the following items:



NOVA RDOS Stack Segment

Notice that B's return and AC displacements would not be used until and unless B calls out to another routine. Furthermore, A's AC storage and return storage locations are known to B as OAC_n and ORTN, and can be used by B, should B wish to return AC parameters to A. Consult PARS for a complete list of system stack mnemonic definitions.

RDOS maintains a pointer for the current system stack frame in location 10, CSP. Linkage to save subroutine state variables on a system stack and restore these variables is provided by routines contained in a utility package called GSUB (MGSUB/AGSUB/BGSUB). The save/restore routines are entitled SAV, SRTN, and RTN.

REENTRANT CODING AND SUBROUTINE LINKAGE (Continued)

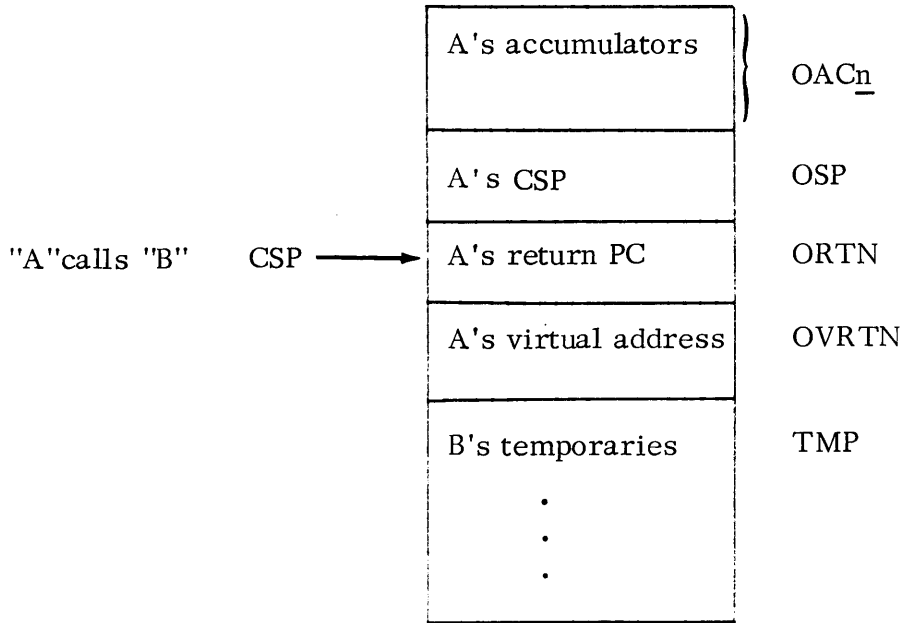
Calls to these routines are defined in the system parameters as RSAVE, SRTRN and RTRN.

On ECLIPSE systems, the RTN machine language instruction is used for system stack frame popping; a macro, RSAVE, is defined using instruction SAVE for stack pushing. This permits stack frames to be variable in length and speeds up the use of the stack. The current frame pointer, CSP, is maintained in location 41. Each system stack in a ECLIPSE environment is 250 octal words long; the interrupt stack is only 100 octal words in length. The fixed portion of each ECLIPSE system stack frame is defined as follows:

<u>Displacement</u>	<u>Mnemonic</u>	<u>Contents</u>
-4	OAC0	AC0 contents if this routine calls another routine.
-3	OAC1	AC1 contents if this routine calls another routine.
-2	OAC2	AC2 contents if this routine calls another routine.
-1	OSP	Current stack pointer (CSP) for this frame if this routine calls another routine.
0	ORTN	PC contents if this routine calls another routine.
1	OVRTN	Overlay virtual address if the caller is a system overlay.
2	TMP	First temporary storage location of the called routine (if this routine calls out).
.		Additional variable storage of called routine.
.		
.		

REENTRANT CODING AND SUBROUTINE LINKAGE (Continued)

Thus if system subroutine A called system subroutine B, a portion of the system stack would contain the following items:



ECLIPSE RDOS Stack Segment

GENERAL SUBROUTINE PACKAGE (GSUB, MGSUB, BGSUB, AGSUB)

GSUB (MGSUB, BGSUB, AGSUB) is a core resident module which contains a series of utilities for the system's use; GSUB is used by NOVA RDOS, MGSUB is used by NOVA mapped RDOS (MRDOS), BGSUB is used by unmapped ECLIPSE RDOS, and AGSUB is used by mapped ECLIPSE RDOS. These utilities can be called by user-routines and are used extensively by the system in such areas as device drivers (e.g., card reader) and overlays (e.g., RING1 and RING2). Each of these utilities is self-contained, i.e., none of them calls out to other routines and none of them save variables outside a system stack frame (except the byte-handling and compare-word routines, as noted later). The following list gives the entry names and uses of each of the utilities in the GSUB, MGSUB, BGSUB, AGSUB modules.

The column labeled "CPU" describes the type of computer system which can issue this call. When a call is appropriate to a mapped environment (e.g., MVMFW) yet is listed as being issuable in a non-mapped system, that call is equated with a similar call (e.g., MVWD) which is appropriate in an unmapped environment. This permits certain calls like MVMFW to be issued in both environments.

GENERAL SUBROUTINE PACKAGE (GSUB, MGSUB, BGSUB, AGSUB) (Continued)

CPU types and their keys are as follows:

- 1 - Unmapped NOVA computer.
- 2 - Mapped NOVA computer.
- 3 - Unmapped ECLIPSE computer.
- 4 - Mapped ECLIPSE computer.

<u>NAME</u>	<u>CPU</u>	<u>USE</u>
RSAVE	1,2	Save state variables on a system stack.
RETN	1,2	Restore state variables, return the frame to the system stack.
MVBYT	1,2,3,4	Move a byte string.
MTMVB	1,3	Move bytes with a mapped <u>to</u> address.
MFMVB	1,3	Move bytes with a mapped <u>from</u> address.
LDBT	1,2	Load a byte.
LDCHR	1,2	Load a byte.
STBT	1,2	Store a byte.
STCHR	1,2	Store a byte.
MLDBT	1,2,4	Load a byte.
MSTBT	1,2,4	Store a byte.
MVWD	1,2,3,4	Move a word string.
MBMVW	1,2,3,4	Move words with both addresses mapped.
MFMVW	1,2,3,4	Move words with a mapped <u>from</u> address.
MTMVW	1,2,3,4	Move words with a mapped <u>to</u> address.
CLEAR	1,2,3,4	Set a block of core to all zeroes.
MCLR	1,2,3,4	Clear a block of words with a mapped starting address.
CMPWD	1,2,3,4	Compare two word strings.
DIVI	1,2,3,4	Perform single precision integer divide.
DIVD	1,2,3,4	Perform double precision integer divide.
MPY	1,2,3,4	Perform single precision integer multiply.
MPYAD	1,2,3,4	Perform double precision integer multiply.
ROX	1,2	Perform an exclusive or.
SETFL	1,2	Set bits in a word to one.
RSTFL	1,2	Reset bits in a word to zero.
SRTN	1,2	Restore current state variables and do not return a frame to the system stack; return to the immediate caller (not to the address stored in RTLOC).
MBLK	2,4	Convert a user address to a mapped address.
SSOVF	3,4	Stack overflow handler.

List of General Subroutines

GENERAL SUBROUTINE PACKAGE (GSUB, MGSUB, BGSUB, AGSUB) (Continued)

Except for RSAVE, RETN and SRTN, each of these routines is called by means of an indirect call to a pointer within the calling routine containing the routine name, e.g., a call to ROX would be performed in the following manner:

```

        .EXTN    ROX
        JSR     @.ROX
        .
        .
        .
    .ROX:    ROX
    
```

RSAVE, RETN and SRTN are used frequently in the operating system routines, so their entry points have been stored in the RDOS page zero area at locations 3, 4, and 11 octal respectively. Calls to these routines are defined in the system parameters (PARS or MPARS) as follows:

RSAVE is a macro call defined in PARS for NOVA systems to the save subroutine whose address is stored in location 3:

```

    .MACRO  RSAVE
            STA     3,@CSP
            JSR     @.SAV
    %
    
```

For ECLIPSE systems, the RSAVE macro is defined as follows:

```

    .MACRO  RSAVE
            SAVE
            ↑ 1 + 1
    %

    RETN   = JSR @4
    SRTRN  = JSR @11
    
```

Use of these mnemonics is sufficient to invoke the routine.

The following summarizes the inputs required by each General Subroutine Package routine and describes the outputs obtained upon return from each call. These routines are used extensively by the system; for example, the card reader driver uses CMPWD and RING1 (discussed in a following section) uses LDBT. Each routine's description (except RSAVE, RETN, and SRTN) presumes the existence of a pointer containing the routine name as illustrated above.

Load/Store a Byte (Continued)

Output: they are not reentrant.

Exclusive Or

Input: AC1, First operand
AC0, Second operand

Calling Sequence: JSR @ .ROX

Output: The exclusive OR of the input operands is returned in AC1.
Both AC0 and AC2 are restored to their input values.

Clear a Block of Core

Input: AC0, Number of sequential words to be cleared to zeroes
AC2, Starting (lowest) address of the block.

Calling Sequence: JSR @ .CLEAR

Output: The specified block of core is cleared. AC0 through AC2
are restored upon exit.

Move a Word String

Input: AC0, Starting address of the string to be copied
AC1, Starting address to receive the string copy
AC2, Number of words to be copied

Calling Sequence: JSR @ .MVWD

Output: The specified word string is copied to the specified receiving
core area. AC0 through AC2 are restored to their input
values.

Compare Two Word Strings

Input: AC0, Starting address of the first string
AC1, Starting address of the second string
AC2, Number of Words to be compared

Calling Sequence: JSR @ .CMPWD
success return
failure return

Mapped Move a Byte String (Continued)

Calling
Sequence: JSR @ .MFMVB

- 2) Input: AC0, "From" byte pointer
AC1, Mapped "to" byte pointer
AC2, Byte count.

Calling
Sequence: JSR @ .MTMVB

Mapped Load and Store Bytes

There are two load/store byte routines for mapped systems; each routine accepts mapped byte pointer inputs only. The result accomplished by both routines is identical: the byte is loaded or stored where the mapped byte pointer specifies that it should be accomplished. Neither of these routines is reentrant.

The formats of the two routines are as follows:

- 1) Input: AC1, Mapped byte pointer
AC0, Byte to be stored (right adjusted)

Calling
Sequence: JSR @ .MSTBT

- 2) Input: AC1, Mapped byte pointer

Calling
Sequence: JSR @ .MLDBT

Output: AC0, Byte (right adjusted)

Mapped Move a Word String

There are three move word string routines for mapped systems. One routine accepts a mapped "to" address, another routine accepts a mapped "from" address, while the third accepts both a mapped "to" address and a mapped "from" address. The result accomplished by all three routines is identical: the specified word string is copied to the specified receiving core area. AC0 through AC2 are restored to their input values.

Mapped Clear a Block of Core (Continued)

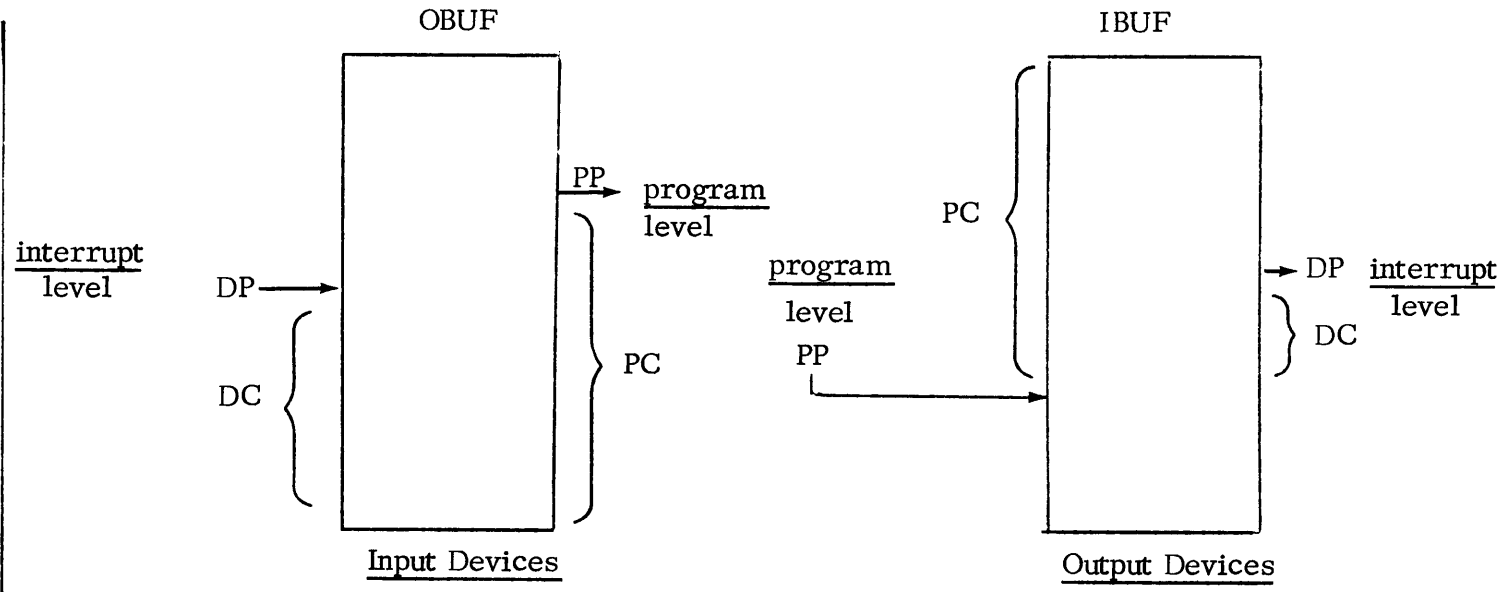
the following section.

GENERALIZED I/O ROUTINES

The I/O modules (RING1, RING2, and RING3) provide a number of useful, general-purpose reentrant routines for handling byte I/O from any device, on input or output, using the program interrupt facility. Entry names of the appropriate I/O routines are placed in each device driver dispatch table as required (see Device Control Table Structure, word 6). Each I/O module is a system overlay.

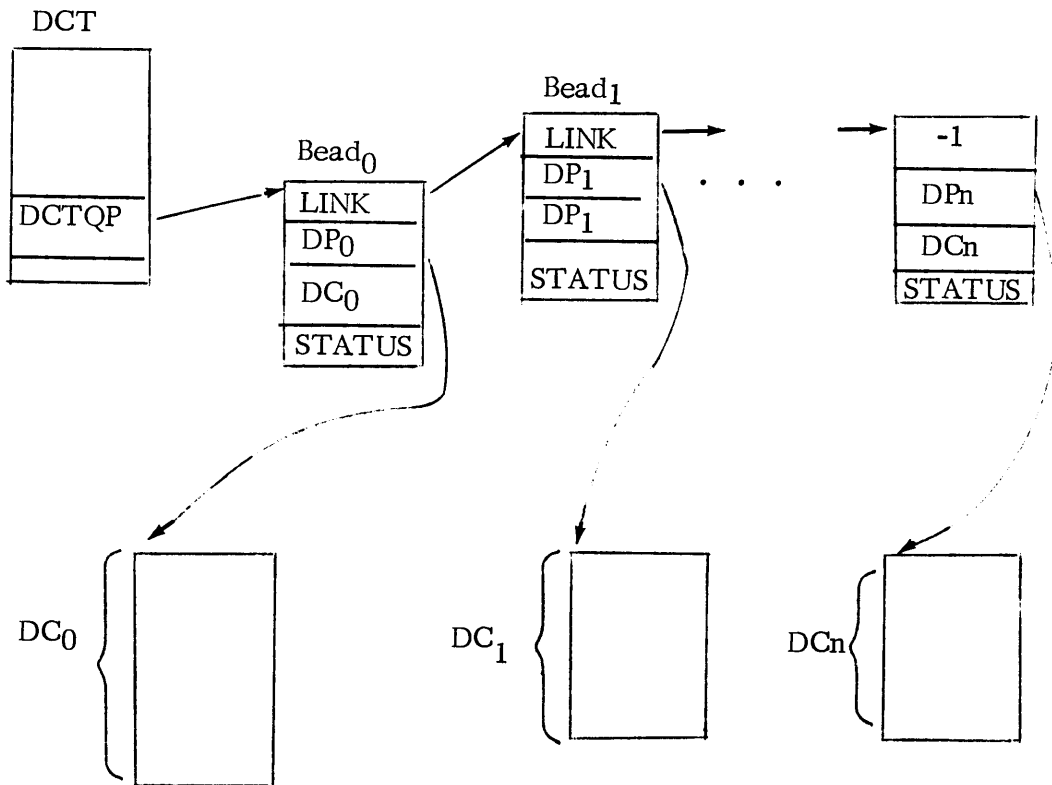
The basic buffer philosophy is to maintain one or more fixed length buffers, with pointers and counters maintained to indicate the amount of data in the buffers and the current word input or output. A virtually unlimited number of buffers can be appended to the first buffer, with a four-word bead assigned to maintain status information and pointers for each additional buffer.

An input device inputs to the buffer at interrupt time and outputs from the buffer at program base level. An output device inputs to the buffer at program base level and outputs from the buffer at interrupt time.



GENERALIZED I/O ROUTINES (Continued)

<u>Bit</u>	<u>Meaning</u>
1B7	Device opened to the background
1B8	Device opened to the foreground
1B9	Common service routines should not start this device (e.g., \$CDR)
1B12	Treat this buffer as a ring buffer when it becomes filled
1B13	Buffer is being treated as a ring buffer
1B14	Echo the character (TTI only)
1B15	Request is completed (cleared by ENQUE routine)



GENERALIZED I/O ROUTINES (Continued)

opened), then control would be dispatched through DCTDT and thence through displacement CF of the dispatch table. The name of the routine used by \$PTR to close this device is CLSI; this name is found in displacement CR of \$PTR's dispatch table.

If the routines provided by the I/O modules are not adequate for a user application, the user writing the device driver must write his own I/O routines (preferably making them core-resident), and insert their names into his driver's dispatch table at the appropriate displacements.

Whenever control is given to a routine whose name is in the dispatch table, the system provides the following well-defined input parameters:

- AC0 - As passed by user in .SYSTEM call
- AC1 - As passed by user in .SYSTEM call
- AC2 - Address of the UFT corresponding to the channel number supplied in the .SYSTEM call

The UFT is a table which, you will recall, is described in Chapter 1. The UFT is defined in PARU, and it is a table which describes each file or device that is currently open.

User-written routines may either do some preprocessing of inputs received at the dispatch table and then transfer control to one of the I/O routines, or the user-written routine may perform all the processing of these inputs. Preprocessing of inputs provided by system calls to OPEN or CLOSE a file/device are not allowed.

Open (OPNO, OPNI)

OPNO is used to open output devices, while OPNI is used to open input devices. OPNI issues an operator intervention message (if required), while OPNO issues the message, provides a form feed, and outputs leader if required. Both routines clear a device and initialize its DCT. This implies that the DCT has provided all necessary I/O buffer information as well as the seven words of variable storage (words 12-16 and 22-23 of the DCT.)

Input: AC2 - UFT address

Calling Sequences: 1) Insert OPNO (OPNI) mnemonic into device dispatch table at displacement 0, OF. (Input is provided by the system.)

Open (OPNO, OPNI) (Continued)

Calling Sequences: 2) .EXTN OPNO (OPNI), OVLAY
(Continued) JSR @ .OVLY
OPNO (OPNI)
error return (never taken)
normal return
:
.OVLY:OVLAY

Users may do no preprocessing of inputs to OPNO (OPNI). If a user wishes to provide a special open routine for a device, the system OPNO (OPNI) routine cannot also be called for that device.

Close (CLSO, CLSI)

CLSO should be used only to close output devices. It waits until all output has settled, clears the column counter, clears the device, initializes the DCT, and provides trailer if required. Alternatively, CLSI should be used only to close input devices. It merely clears the device and initializes the DCT.

Input: AC2 - UFT address

Calling Sequences: 1) Insert either mnemonic CLSO or CLSI into the device
dispatch table at displacement 1 (CF). (Input is provided
by system.)
2) .EXTN CLSO (CLSI), OVLAY
JSR @ .OVLY
CLSO (CLSI)
error return (never taken)
normal return
:
.OVLY:OVLAY

Users may do no preprocessing of inputs to CLSO (CLSI). If a user wishes to provide a special close routine for a device, the system close CLSO (or CLSI) routine may not also be called for that device.

Read Sequential (RDS)

The device will be read, one byte at a time, until the byte count requested is satisfied. The data is not modified in any manner; this command is used for binary transfers.

Input: AC2 - UFT address
AC1 - byte count for RDS
AC0 - destination byte pointer

Read Sequential (RDS) (Continued)

Calling Sequences: 1) Insert RDS mnemonic into device dispatch table at displacement 2 (RS). (Input is provided by system.)

2) .EXTN RDS, OVLAY
JSR @ .OVLY
RDS
error return
normal return
:
:
:
.OVLY: OVLAY

The error return is taken if an end of file or device timeout occurs, at which time error code EREOF is returned in AC2 and the partial byte count is returned at AC1. If the normal return is taken, the full byte count read is returned in AC1.

Read Line (RDL)

This routine is used to transmit ASCII data and terminate after transmission of a carriage return, form feed, or null. All bytes transmitted are masked to seven bits. Line feeds and rubouts are unconditionally ignored.

Input: AC2 - UFT address
AC0 - destination byte pointer

Calling Sequence: 1) Insert RDL mnemonic into device dispatch table at displacement 3 (RL). (Input is provided by system.)

2) .EXTN RDL, OVLAY
JSR@ .OVLY
RDL
error return
normal return
:
:
:
.OVLY: OVLAY

The read byte count is returned in AC1, whether the error or normal returns are taken. The error return is taken and a system error code returned in AC2 for the following conditions:

Read Line (RDL) (Continued)

<u>AC2</u>	<u>Mnemonic</u>	<u>Meaning</u>
6	EREOF	End of file or device timeout.
22	ERLLI	Line length exceeded 132 characters without a valid terminator.
24	ERPAR	Parity error in the last character transmitted.

Write Sequential (WRS)

Data is output in byte form to a device until the byte count has expired. The data is not altered in any manner. This mode is therefore the standard mode for binary output transfers.

Input: AC2 - UFT address
 AC1 - byte count for WRS
 AC0 - source byte pointer

Calling Sequences: 1) Insert mnemonic WRS into the device dispatch table at displacement 5 (WS). (Output is provided by the system.)

```

2)      .EXTN WRS, OVLAY
        JSR@   .OVLAY
        WRS
        error return
        normal return
        :
        :
        .OVLAY: OVLAY
    
```

The byte count written is returned in AC1. The error return is never taken.

Write a Line (WRL)

This routine transmits ASCII data to the appropriate device and terminates after transmitting either a carriage return or a form feed. Termination also occurs upon detection of a null, but the null is not written. Checks are made of the device characteristics to determine whether to perform:

1. Parity on output
2. Nulls after form feeds
3. Line feeds after carriage returns
4. Tab simulation (every 8 columns)
5. Rubouts after tabs

Write a Line (WRL) (Continued)

Input: AC2 - UFT address
 AC0 - source byte pointer

Calling Sequences: 1) Insert WRL mnemonic into device dispatch table at displacement 6, WL. (Output is provided by the system.)

```
2)            .EXTN  WRL, OVLAY
              JSR@   .OVLY
              WRL
              error return
              normal return
              :
              :
              .OVLY: OVLAY
```

The write byte count is return in AC1 upon exit. The error return is taken after 132 bytes have been written without detection of a valid terminator, and AC2 contains the following error code:

<u>AC2</u>	<u>Mnemonic</u>	<u>Meaning</u>
22	ERLLI	Line limit exceeded.

The RING I/O modules do not provide for random record reading or writing.

Get Master or Default Directory Name (GMDIR/GDIRS)

These routines pass the name of the master directory device (GMDIR) or the name of the current directory (GDIRS). The master directory device is the primary or secondary partition which becomes the current directory after a full system initialization or a bootstrap; this directory also contains the system overlays. Since several versions of the operating system may be available for bootstrapping, different devices may become the master device. If the name of the current directory is requested, only the current directory name, not the colon delimiter and not the names of superior directories, is returned. The format of these calls is:

Input: AC0 - Byte pointer to 13₈ byte user area

Get Master or Default Directory Name (GMDIR/GDIRS) (Continued)

```

Calling Sequences: .EXTN OVLAY, GMDIR, GDIRS
                   JSR @ .OVLAY
                   GMDIR           ;GET THE MASTER DIRECTORY NAME
                   error return    ;ATTEMPT TO OVERWRITE THE SYSTEM
                   normal return
                   :
                   JSR @ .OVLAY
                   GDIRS           ;GET THE CURRENT DIRECTORY NAME
                   error return    ;ATTEMPT TO OVERWRITE THE SYSTEM
                   normal return
                   :
                   .OVLAY:OVLAY
    
```

Two error codes may be returned:

<u>AC2</u>	<u>Mnemonic</u>	<u>Meaning</u>
33	ERRD	Attempt to overwrite system (unmapped only).
74	ERMPR	Address outside address space (mapped only).

I/O BUFFER MODULE (IOBUF)

IOBUF is a core-resident module used by nearly all system drivers. One use of IOBUF is to execute certain types of I/O instructions; this permits the writing of reentrant drivers used by multiple devices (e.g., a multiple TTY system requiring only one TTY driver). Another use of IOBUF is to provide common input/output character device interrupt processing. IOBUF also provides routines to place a bead at the end of a bead string and to remove any bead from the string. Finally, IOBUF provides routines which perform common input and output character device interrupt servicing.

I/O BUFFER MODULE (IOBUF) (Continued)

The following list summarizes the entries in IOBUF. Before any of these calls can be issued, interrupts must be disabled unless they are masked due to the call's being issued at interrupt level.

<u>Entry</u>	<u>Use</u>
CISER	Common input interrupt service.
COSER	Common output interrupt service.
DEQUE	Remove any bead from the string.
ENQUE	Add a bead at the end of the string.
FINP	Partial input interrupt service.
IBUF	Input a character to the I/O buffer.
OBUF	Output a character from the I/O buffer.
PENQU	Priority enqueue a bead.
STOUT	Initiate an output device, then provide interrupt service.
XDIAC	DIAC instruction processor.
XDOAS	DOAS instruction processor.
XNIOC	NIOC instruction processor.
XNIOS	NIOS instruction processor.
XSKPB	SKPBZ instruction processor.
XDIAP	DIAP instruction processor.
XIBUF	Priority enqueue a bead and input to buffer.

Calls to any of the instruction processor entries (XDIAC, XDOAS, etc.) cause that instruction to be built (for the specific device in question), stored in the driver's "execute I/O instruction" area (pointed to by DCTEX of the device's DCT), and executed. Control is then returned to the caller. Calls to the instruction processors are issued solely by the system; specific call types depend upon the entries selected in the device dispatch table or in displacement DCTST of the Device Control Table. Upon return from any of these calls, the contents of AC3 are undefined.

Common Input Device Interrupt Service (CISER)

CISER provides interrupt service for character input devices. CISER is called either by placing the mnemonic CISER in word 2 of the input device's DCT or by means of the following calling sequence:

Input: AC2 - DCT address

Calling Sequence: .EXTN CISER
JSR @ .CISE
return to DISMIS

.CISE: CISER

Common Input Device Interrupt Service (CISER) (Continued)

Output: If the instruction execution returns a character, this character is returned in the device buffer. The input contents of AC1 are lost; the input contents of AC2 are preserved.

Common Output Device Interrupt Service (COSER, STOUT)

COSER provides interrupt service for character output devices. STOUT initiates an output device, starts the device, then branches to COSER. Each routine can be called either by placing its mnemonic in word 2 (DCTIS) of the DCT or by means of the following calling sequence:

Input: AC2 - DCT address

Calling Sequence: .EXTN COSER (STOUT)
JSR @ .COSE
return to DISMISS

.COSE: COSER (STOUT)

Output: If a character is available for output in the device buffer, it is output. AC2 is preserved upon exit.

Add a Bead to the String (ENQUE)

ENQUE attaches a bead and I/O buffer to the end of the string of beads and buffers. This new bead becomes the last bead in the string, and is given a LINK of -1. ENQUE is called by means of the following calling sequence:

Input: AC1 - Bead address
AC2 - DCT address

Calling Sequence: .EXTN ENQUE
JSR @ .ENQU
return

.ENQ: ENQUE

Output: The bead and its associated buffer are attached to the end of the bead string. If the device is not busy, it is started. AC2 is unchanged upon exit.

Terminate Bead from Head of List (DEQRQ)

Like FINP, DEQRQ shares common code with CISER. DEQRQ's operations, however, are limited to the following: Sets "Request Done" in the bead status word, readies a system task if necessary, and determines if any other beads are enqueued

Terminate Bead from Head of List (DEQRQ) (Continued)

for the device. If other beads are enqueued, updates the bead pointer in DCTQP and restarts the device; otherwise clears the device.

Input: AC2 - DCT address

Calling .EXTN DEQRQ

Sequence: JSR @ .DEQRQ

normal return

.DEQRQ:DEQRQ

Remove a Bead from the String (DEQUE)

DEQUE removes a bead from the bead string by setting the bead status word to "inactive" and adjusting the adjoining bead links, and by clearing the device associated with this string if the device is no longer active. The DEQUE calling sequence is:

Input: AC1 - Address of the bead to be removed

AC2 - DCT address

Calling .EXTN DEQUE

Sequence: JSR @ .DEQU

return

.DEQU: DEQUE

Output: The bead is inactivated and removed from the string. AC2 is unchanged upon exit.

Partial Input Interrupt Service (FINP)

FINP performs all functions accomplished by CISER without initially issuing a call to XDIAC followed by a restart of the device; FINP is thus an entry in CISER code which omits those two functions. FINP presumes that these functions have been performed elsewhere (as in the TTY and CDR drivers).

Input: AC2 - DCT address

Calling .EXTN FINP

Sequence: JSR @ FINP

normal return

.FINP: FINP

Input a Character to the I/O Buffer (IBUF)

IBUF places an eight-bit character in the currently available buffer slot. All book-keeping in the DCT is maintained.

Input: AC0 - character (left byte ignored)
 AC2 - DCT address

Calling Sequence: .EXTN IBUF
 JSR @ .IBUF
 return - buffer full
 return - buffer not full

 .IBUF: IBUF

Priority Enqueue a Bead (PENQU)

PENQU places a bead and its buffer at the head of a bead string. If the device is not busy, it is started.

Input: AC1 - Bead address
 AC2 - DCT address

Calling Sequence: .EXTN PENQU
 JSR @ .PENQU
 normal return

 .PENQU: PENQU

Output: The bead and its associated buffer are attached to the beginning of the bead string. If the device is not busy, it is started. AC2 is unchanged upon exit.

Priority Enqueue a Bead and Input to an I/O Buffer (XIBUF)

XIBUF places a bead and its buffer at the head of a bead queue, and places an eight-bit character in the currently available buffer slot. All bookkeeping in the device's DCT is maintained.

Input: AC0 - Character
 AC2 - Base address of a dummy DCT. The only portion of this DCT which is used by the routine is the bead portion of the DCT: displacements DCTQL through DCTQS. It is this bead which is placed at the head of the device queue.

Priority Enqueue a Bead and Input to an I/O Buffer (XIBUF) (Continued)

```
Calling Sequence:  .EXTN XIBUF
                  JSR @ .XIBUF
                  error return          ;BUFFER FULL
                  normal return
                  :
                  :
.XIBUF:          XIBUF
```

Output a Character from the I/O Buffer (OBUF)

OBUF retrieves an eight-bit character from position PP of the input buffer if one is available. All bookkeeping in the DCT is maintained.

Input: AC2 - DCT address

```
Calling Sequence:  .EXTN OBUF
                  JSR @ .OBUF
                  return - buffer empty
                  return - buffer not empty
.OBUF:           OBUF
```

Output: If the buffer was not empty, a character is returned in AC0, bits 8-15.

I/O BUFFER MANAGEMENT

As mentioned earlier, there is an I/O buffer for each device in the system, and its size may be increased or decreased by the addition or removal of buffer beads. At program base level (as opposed to interrupt level), users wishing to fetch a character from a buffer issue either an RDL or RDS call which, in turn, calls another routine, RCHR. RCHR, RDL, and RDS are disk resident I/O routines in RING I/O. The purpose of RCHR is to perform a generalized character fetch using the Device Control Table address to read a character from the physical device. Having received a character from the buffer, RCHR passes this character to the calling read routine.

Each time RCHR is called, interrupts are disabled and OBUF is called. A character is fetched from the buffer, if one is available. (OBUF is a core resident routine in the IOBUF module, described earlier.) If a character is not available in the buffer, the caller is suspended for a period of time less than or equal to the device timeout constant, DCTTO. If a timeout occurs, RTRN is executed; otherwise a loop back to the beginning of the RCHR sequence is made.

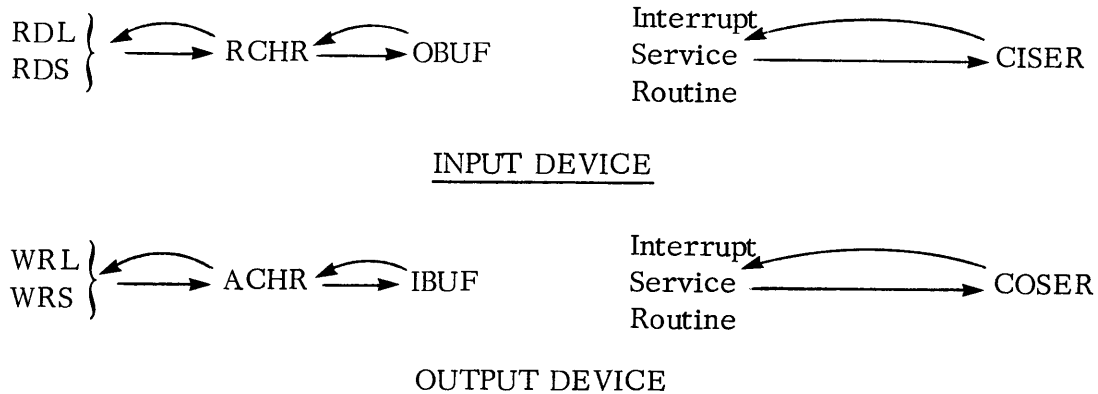
I/O BUFFER MANAGEMENT (Continued)

At interrupt level, input devices input characters to a buffer by means of a call to CISER, also described on the preceding page.

A similar sequence of subroutine calls occurs for output devices. At program base level, a call to WRL or WRS results in a call to ACHR which adds a character to the I/O buffer. ACHR is a disk resident I/O routine which calls IBUF to add a character to the buffer.

ACHR makes an attempt to place the character in the I/O buffer by means of a call to IBUF. IBUF, described earlier, performs functions complementary to those of OBUF. If the buffer was not full, then the character is added to the buffer and thence is output to the device (spooled, if possible). If the buffer was full, a spool for the data is started (if spooling is possible). Otherwise the system task is suspended for a period less than or equal to the device timeout constant.

At the output interrupt level, COSER is used to output a character to the device.



Retrieve a Character from the I/O Buffer (RCHR)

Retrieve a character from the buffer by means of a call to OBUF.

Input: AC2 - DCT address

Calling Sequence: .EXTN RCHR OVLAY
 JSR @ .OVLY
 RCHR
 timeout return
 normal return
 .OVLY: OVLAY

Output: AC0 -(if successful) character in bits 8-15.

Add a Character to the I/O Buffer (ACHR)

Insert a character in the I/O buffer by means of a call to IBUF.

Input: AC2 - DCT address
 AC0 - character to be inserted

Calling Sequence: .EXTN ACHR, OVLAY
 JSR @ .OVLY
 ACHR
 error return
 normal return
 .OVLY: OVLAY

The error return is taken in the event of a fatal disk error.

Declaring the DCT Address

The last relocatable binary in system library RDOSC.LB is TABLE. TABLE contains the interrupt vector, named ITBL, discussed earlier. This vector is a linear array of DCT addresses which the system interrupt handler indexes by device code. The form of each ITBL entry is:

.dvdDCT: dvdDCT ; OCTAL DEVICE CODE

where dvd represents the DGC device mnemonic for each device, and dvdDCT is the DCT address of each device. Each DCT address must be declared as a normal external. When adding an entry, the user can select any three-letter device mnemonic not used by a DGC device.

Creating a Peripheral Device Entry in a Directory

All peripheral device entries are created (and deleted) as required on every initialization of a disk device. All information necessary to accomplish this is in the system overlay SFTAB. The information is as follows:

Word 1:	Byte pointer (overlay relative) to entry name, packed left to right.
Word 2:	File attributes of the required entry.
Word 3:	Logical device code. This is the same as the physical device code except in cases where two or more logical devices share the same device code, such as \$TTI/\$TTR.

The latter part of the overlay contains the directory entry text strings. The text string for the paper tape reader, for example, is as follows:

```

PTRP          ;NAME POINTER
ATPER+ATWP+ATCHA ;ATTRIBUTES
PTR           ;LOGICAL DEVICE CODE

PTRP:        .TXTM 1
             .TXT  /$PTR/
```

Updating the System Libraries

The RDOS 04 system libraries (RDOSA/MRDOSA/ARDOSA/BRDOSA etc. for system types 1-4) are arranged as shown in the following table. For a listing of the modules in any other libraries, perform an analyze via the library file editor (LFE). To add a driver, insert it into any of the libraries (A,B, or C) provided it is inserted after SYSTE (BSYSTE/MSYSTE/ASYSTE) and before TABLE (BTABLE/MTABLE/ATABLE).

The column labeled "CPU" describes the type of computer system which utilizes a specific relocatable binary module. CPU types and their keys are as follows:

- 1 - Unmapped NOVA computer
- 2 - Mapped NOVA computer
- 3 - Unmapped ECLIPSE computer
- 4 - Mapped ECLIPSE computer

SYSTEM LIBRARY LIST

	<u>Relocatable Binary Title</u>	<u>CPU</u>	<u>Primary Function or Contents</u>	
LIBRARY A:	INIT1, INIT2, INIT3	1	Full and partial system initializations	
	BINIT1, BINIT2, BINIT3	3	Full and partial system initializations	
	MPWRFL	2, 4	Mapped power fail handler	
	MAPZ/AMAPZ	2, 4	Mapped system page zero	
	SCHED/MSCHED/BSCHED/ASCHED	1, 2, 3, 4	System task scheduler	
	SYMOD/MSYMOD/BSYMOD/ASYMOD	1, 2, 3, 4	.SYSTEM call processor	
	SYSTE/MSYSTE/BSYSTE/ASYSTE	1, 2, 3, 4	.SYSTEM call table and processor	
	OVLAY/OVLAY/BOVLAY/BOVLAY	1, 2, 3, 4	System overlay handler	
	OPPRO/MOPPRO/BOPPRO/AOPPRO	1, 2, 3, 4	Operator-Foreground/Background communications	
	TTY1D	1, 2, 3, 4	Second Teletype driver	
	TTYDR/MTTYDR/TTYDR/MTTYDR	1, 2, 3, 4	Teletype/video display driver	
	DPMOD/DPMOD/BDPMOD/BDPMOD	1, 2, 3, 4	Dual processor module	
	RWBLK/MRWBLK/BRWBLK/ARWBLK	1, 2, 3, 4	.RDB/.WRB and .EXEC/.RTN handler	
	BLKIO/B LKIO/BBLKIO/BBLKIO	1, 2, 3, 4	Block I/O Management	
	FILIO/MFILIO/BFILIO/AFILIO	1, 2, 3, 4	Disk file I/O	
	GSUB/MGSUB/BGSUB/AGSUB	1, 2, 3, 4	General purpose subroutines and linkage	
	PLT1D	1, 2, 3, 4	Second plotter driver	
	PLTDR	1, 2, 3, 4	Incremental plotter driver	
	CDR1D	1, 2, 3, 4	Second card reader driver	
	CDRDR/CDRDR/BCDRD/BCDRDR	1, 2, 3, 4	Card reader driver	
	PTP1D	1, 2, 3, 4	Second high speed punch driver	
	PTPDR	1, 2, 3, 4	High speed punch driver	
	PTR1D	1, 2, 3, 4	Second high speed reader driver	
	PTRDR	1, 2, 3, 4	High speed reader driver	
	LP132	1, 2, 3, 4	132 column line printer characteristics word for \$LPT	
	LP180	1, 2, 3, 4	80 column line printer characteristics word for \$LPT	
	LP232	1, 2, 3, 4	132 column line printer characteristics word for \$LPT1	
	LP280	1, 2, 3, 4	80 column line printer characteristics word for \$LPT1	
	LPT1D	1, 2, 3, 4	Second line printer driver	
	LPTDR	1, 2, 3, 4	Line printer driver	
	IOBUF/IOBUF/BIOBUF/BIOBUF	1, 2, 3, 4	I/O buffer handlers	
	LIBRARY B:	M17DB through M00DB	1, 2, 3, 4	Magnetic tape device control blocks
		C17DB through C00DB	1, 2, 3, 4	Cassette device control blocks
		MTADR/MMTADR/BMTADR/AMTADR	1, 2, 3, 4	Magnetic tape/cassette driver
		MTA1D	1, 2, 3, 4	Second magnetic tape device control table
MTADC		1, 2, 3, 4	Magnetic tape device control table	
CTA1D		1, 2, 3, 4	Second cassette device control table	
CTADC		1, 2, 3, 4	Cassette device control table	
LIBRARY C:	STACKS/STACKS/BSTACKS/BSTACKS	1, 2, 3, 4	System stacks	
	CELLS	1, 2, 3, 4	System cells	
	DKDCB	1, 2, 3, 4	Fixed head disk DCB, first controller	
	DK1DB	1, 2, 3, 4	Fixed head disk DCB, second controller	
	DSKDR/MDSKDR/BDSKDR/ADSKDR	1, 2, 3, 4	Fixed head disk driver	
	DSKDC/DSKDC/BDSKDC/BDSKDC	1, 2, 3, 4	First fixed head disk device control table	
	DSK1D/DSK1D/BDSK1D/BDSK1D	1, 2, 3, 4	Second fixed head disk device control table	
	DP7DB through DPODB	1, 2, 3, 4	Moving head disk DCBs for units 7 through 0	
	DKPDR/MDKPDR/BDKPDR/ADKPDR	1, 2, 3, 4	Moving head disk driver	

SYSTEM LIBRARY LIST (Continued)

	<u>Relocatable Binary Title</u>	<u>CPU</u>	<u>Primary Function or Contents</u>
LIBRARY C:	DKPDC, DKPDC, BDKPDC, BDKPDC	1, 2, 3, 4	First moving head disk device control table
	DKP1D, DKP1D, BDKP1D, BDKP1D	1, 2, 3, 4	Second moving head disk device control table
	QTYDR, MQTYDR, BQTYDR, AQTYDR	1, 2, 3, 4	Asynchronous multiplexer driver
	MCT1D, MMCT1D, MCT1D, MMCT1D	1, 2, 3, 4	Second MCA device control table
	MCTDC, MMCTDC, MCTDC, MMCTDC	1, 2, 3, 4	First MCA device control table
	MCADR, MMCADR, BMCADR, AMCADR	1, 2, 3, 4	MCA driver
	P31DB through P00DB	1, 2, 3, 4	Partition/subdirectory DCBs 31 through 0
	PWRFL	1	Unmapped NOVA power fail handler
	BPWRFL	3	Unmapped ECLIPSE power fail handler
	INTD, MINTD, BINTD, AINTD	1, 2, 3, 4	Interrupt determinater
	PANIC	1, 2, 3, 4	System PANIC handler
	TABLE, MTABLE, BTABLE, ATABLE	1, 2, 3, 4	Tables section

Updating the System Libraries (Continued)

For more information about the contents of the RDOS 04 libraries, perform an analyze.

Creating a System Queue Entry

A system queue must be created for the device within the device driver. The structure of this entry is as follows:

```
   dvdQ:      .ENT      dvdQ
              .BLK 2          ;USED TO POINT TO DCT AND UFT
              100000         ;SYSTEM STACK IS REQUIRED
              dvdDCT         ;DCT ASSOCIATED WITH QUEUE
              .BLK  SQLN-. +dvdQ ;SYSTEM STORAGE AREA
```

The address of this queue entry must be defined in the system queue table (SQ) in the TABLE mode.

An example of this queue entry for the paper tape reader is as follows:

```
   PTRQ:      .BLK 2
              100000
              PTRDCT
              .BLK  SQLN-. +PTRQ
```

System Generation

Invoke the SYSGEN save file and answer all queries. Determine the additional space necessary to load the new user driver plus any additional words added to the system. Before creating the new operating system save file, the value contained in the file named NREL should be adjusted down by the additional amount of space required by the new device driver.

To insure that the driver will get loaded from the library at system generation time, a small program should be written and loaded at this time ahead of the RDOSA.LB (MRDOSA.LB) library. The program could take the following form:

```
   .EXTN dvdDC
   .END
```

and the relocatable binary called DUMMY.RB.

System Generation (Continued)

File SYSGEN.CM must be modified so that DUMMY appears as an entry after @NREL@.

Invoke SYSGEN.CM to build the new RDOS system

PRACTICAL HINTS FOR SYSTEM DEVICE DRIVER IMPLEMENTATION

This section is devoted to a line-by-line examination of an actual RDOS driver, the high speed paper tape reader.

Elements Required in User-Written I/O Routines

User device drivers may perform I/O in three ways:

1. By using system routines, placing the routine names in the user device dispatch table.
2. By appending some pre-processing instructions to an existing system routine.
3. By substituting system I/O routines with user written routines, and either placing these routines in-line or by placing the routine names in the dispatch table.

The first case, placing a system I/O routine name in the device dispatch table, is the easiest to implement.

In the second case, where the user wishes to do some pre-processing of input parameters,* then transfer control to a system routine, the following steps must be followed. The steps in this example illustrate the case where RDL is the system call which will have instructions appended to it by the user:

1. Perform a SAVE before calling RDL.
2. Ensure that RDL is passed its requisite parameters. These are the UFT addresses in AC2 and the destination byte pointer in AC0. Use the RDL calling sequence defined earlier.
3. Perform an RTRN upon return from the call to RDL.

In the last case, a user wishes to write an I/O routine which will be used instead of a system routine (perhaps placing this routine in his driver module).

* except upon opening or closing a device.

Elements Required in User-Written I/O Routines (Continued)

The following example sketches the essential elements of such a routine (leaving blank the I/O code sequence proper):

```

RDL1:      .ENT      RDL1
           STA@     3, CSP
           RSAVE
           LDA      0, OAC0, 3      ;SAVE CALLER'S AC'S
           .
           MOV#    2, 2, SNR      ;FETCH INPUT AC0
           JMP     TIMEOUT
           .
           ISZ     ORTN, 3        ;TEST FOR END OF FILE ERROR
           RTRN
TIMEOUT:   LDA      2, ERROR      ;GO TO NORMAL RETURN
           STA     2, OAC2        ;GET ERROR CODE
           RTRN
ERROR:     EREOF                ;RETURN ERROR CODE IN AC2
    
```

Examination of a System Device Driver

This section is devoted to a line-by-line examination of an actual RDOS driver, the high-speed paper tape reader driver (PTRDR).

Following are the first 11 lines of the RDOS paper tape reader driver:

```

10002 PTRDR
01
03          .TITLE  PTRDRV  ; PAPER TAPE READER DRIVER
04          .NREL
05
06          .ENT  PTRDC, PTRQ, PTRDT, PTREX
07
08          .EXTN RDS, OPNI, CLSI, RDL ;COMMAND ENABLE
09          .EXTN XNIOS      ;EXECUTE I/O INSTRS
10          .EXTN CISER      ;INTERRUPT SERVICE
11
    
```

Examination of a System Device Driver (Continued)

The entries PTRDC and PTRQ on line 6 define the PTR Device Control Table address and the system queue entry respectively. PTRDT and PTREX define the dispatch table address and the execute - I/O instruction routine respectively. PTRDT and PTREX are entered so that they can be referenced by PTR1D, the second paper tape reader driver.

```

12          ; PAPER TAPE READER DEVICE CONTROL TABLE
13
14 00000'000000 PTRDCT: 0
15 00001'001737          MKPTR          ; MASK
16 00002'177777          CISER          ;INT SERVICE
17 00003'000404          DCIDI+DCPCK          ; CHARACTERISTICS
18 00004'000012          PTR          ;DEVICE CODE
19 00005'000065'        PTREX          ;EXECUTE IO INSTRUCTION
20 00006'000110'        PTRDT          ; DISPATCH TABLE ADDRESS
21 00007'177777          XNIOS          ; READER START ROUTINE ADDRESS
22 00010'000100          PTRSZ*2          ; BUFFER SIZE
23 00011'000052"        PTRBF*2          ; BUFFER FIRST BYTE ADDRESS
24 00012'000001          .BLK 1          ;PROGRAM BYTE COUNT
25 00013'000001          .BLK 1          ;PROGRAM BYTE POINTER
26 00014'000001          .BLK 1          ;DEVICE BEAD LINK
27 00015'000001          .BLK 1          ;DEVICE DATA BYTE POINTER
28 00016'000001          .BLK 1          ;DEVICE DATA COUNT
29 00017'040011          BSUPC+BSDON+BSLPA ; BEAD STATUS
30 00020'000014'        .-4          ;BEAD ADDRESS
31 00021'177777          -1          ;REQUEST QUEUE POINTER
32 00022'000002          .BLK 2          ;DEVICE TEMPS
33 00024'000002          2          ;TIME OUT IN SECONDS

```

Locations 0 through 24 (lines 14-33) comprise the paper tape reader Device Control Table (DCT). Word 1 (line 15) defines the interrupt mask. The paper tape reader mask word, 1737, prevents all devices with mask bit assignments 6-9 and 11-15 inclusive from interrupting the reader (thus preventing chatter); see PARS, Device Interrupt Masks. Word 2 specifies that reader interrupt service is performed by CISER, the common interrupt service routine found in the IOBUF module. Words 4 and 5 define the device characteristics and device code; DCIDI indicates that the reader requires operator intervention, while DCPCK indicates that the reader is a device requiring an even parity check on input.

PTREX, word 5, is a pointer to the reader I/O execute instruction area. This area will be seen later. PTRDT points to the reader dispatch table; this table will be seen on page 2 of the reader driver listing. XNIOS, line 21, is an entry in the IOBUF module and will be resolved by the loader. XNIOS starts the reader.

Examination of a System Device Driver (Continued)

Finally, on page 3 of the driver listing we find the paper tape reader dispatch table, PTRDT. This table provides entries to open and close the reader and to perform read line and read sequential operations:

```

10003 PTRDR
01
02
03           ; DEFINE THE PAPER TAPE READER DISPATCH TABLE
04
05 00110'177777 PTRDT:  OPNI           ; PTR OPEN
06 00111'177777       CLSI           ; PTR CLOSE
07 00112'177777       RDS            ; PTR READ SEQUENTIAL
08 00113'177777       RDL            ; PTR READ LINE
09 00114'177777       -1             ; PTR READ RANDOM
10 00115'177777       -1             ; PTR WRITE SEQ
11 00116'177777       -1             ; PTR WRITE LINE
12 00117'177777       -1             ; PTR WRITE RANDOM
13 00120'177777       -1             ; PTR OPEN FOR APPENDING
14 00121'000110'      OPNI           ; PTR READ ONLY OPEN
15 00122'000121'      OPNI           ; PTR EXCLUSIVE OPEN
16 00123'177777       -1             ; PTR TRANSPARENT OPEN
17
18
19           .END
    
```

The read random record and all write commands are illegal for the paper tape reader. This is indicated by the placement of -1 in these positions of the dispatch table.

Following is the cross-referenced symbol listing for the paper tape reader driver:

```

0004 PTRDR

CISER 000002' XN    2/10    2/16
CLSI  000111' XN    2/08    3/06
OPNI  000122' XN    2/08    3/05    3/14    3/15
PTRBT 000025'      2/23    2/37
PTRDC 000000' EN    2/06    2/14    2/46
PTRDT 000110' EN    2/06    2/20    3/05
PTREX 000065' EN    2/06    2/19    2/39
PTRQ  000070' EN    2/06    2/44    2/47
PTRSZ 000040      2/22    2/35    2/37
RDL   000113' XN    2/08    3/08
RDS   000112' XN    2/08    3/07
XNIOS 000007' XN    2/09    2/21
    
```

CHAPTER 3

USER PROGRAM SERVICED INTERRUPTS

SERVICING USER INTERRUPTS

Special user devices may be identified either at the time an RDOS system is loaded or at run time. This chapter describes the procedure for identifying a user device at run time and for creating a user clock driven by the system real time clock. The considerations given for identifying a user device are common to both single and multitask environments; the user clock facility may also be used in both task environments. Only ten (decimal) user devices--excluding the user clock--can be identified to the system at any moment.

Upon detection of an interrupt request, the system will be dispatched through the device interrupt vector table, .ITBL (see Chapter 1). In this table are pointers to Device Control Tables (DCTs) for devices established at system initialization time, whether system or user devices. Chapter 1 describes the structure of system DCTs.

User Device Driver Implementation at Run Time

In order to identify a user device to the system at run time, the user must provide a three-word DCT as an interface between the system interrupt dispatch routine and the user-interrupt servicing routine. The structure and mnemonic assignments of this three-word table are as follows:

<u>Displacement</u>	<u>Mnemonic</u>	<u>Purpose</u>
0	DCTBS	Pointer to an 8-word state save area.
1	DCTMS	Interrupt service mask.
2	DCTIS	Interrupt service routine address.

DCTBS is a pointer to an eight word state variable save area reserved for compatibility with RTOS. In RDOS, state information is reserved on the interrupt stack. DCTIS is a pointer to the routine which services this particular device interrupt. DCTMS is the interrupt mask* that the user wants to be ORed with the current interrupt mask while in the user interrupt service routine. This mask establishes which devices--if any--will be able to interrupt the currently interrupting device.

Upon transferring control to the user interrupt service routine, the system will ensure that AC3 contains the return address required for exit from the routine, and that AC2 contains the address of the DCT upon exit from the routine.

* See "How to Use the NOVA computers," Section 2.4.

User Device Driver Implementation at Run Time (Continued)

Exit is accomplished by issuing task call .UIEX; rescheduling may occur as an option.

All multitask environment activity ceases at the moment that a user device interrupt is detected. Nonetheless, it is possible for a user to communicate a message to a task from a service routine. If the task in question has been expecting such a message through issuance of a .REC and is now in the suspended state, issuance of the message via .IXMT will cause that task to be readied even though multitask activity is in abeyance. If no task has issued a .REC for such a message, .IXMT simply posts the message and takes no further action. For more information on communicating to tasks from a user interrupt service routine, see Chapter 5 of the RDOS User's Manual, 093-000075-04.

In addition to .IXMT, certain other task calls can be issued from a user interrupt routine or user power fail routine. A complete list of these task calls follows: .IXMT, .SMSK, .UIEX and .UPEX.

All user devices are removed from the system when either a program swap or chain occurs. Receipt of a user interrupt on a new program level (which has not identified the user device) will cause the system to clear the device's done and busy flags and then return to normal program execution.

Identify a User Interrupt Device (.IDEF)

In order to introduce to the system those devices (not identified at SYSGEN time) whose interrupts the system is to recognize, the system call .IDEF must be issued. This places an entry in the interrupt vector table. AC0 contains the device code of the new device. AC1 contains the address of the new device's DCT. In unmapped systems, this address must exceed 400g; in mapped systems, bit 0 of this address is set if the device is a data channel device. In mapped systems with a device using the data channel, AC2 contains the number of 1K core memory blocks which will be needed by the data channel map. This number will be one larger than the integer number of 1024_{10} word blocks used for data channel core buffers. The format of this command is:

Identify a User Interrupt Device (. IDEF) (Continued)

.SYSTM
. IDEF
error return
normal return

Possible error messages are:

<u>AC2</u>	<u>Mnemonic</u>	<u>Meaning</u>
36	ERDNM	Illegal device code (> 778). Device code 778 is reserved for the power monitor/auto restart option.
45	ERIBS	Interrupt device code in use.
65	ERDCH	Insufficient room in data channel map.
74	ERMPR	Address outside address space (mapped systems only).

Exit from a User Interrupt Routine (.UIEX)

Upon a user device interrupt, AC3 will contain the return address upon entry to the user routine.

In unmapped systems, before issuing task call .UIEX AC3 must be loaded with the return address that it contained upon entry to the user routine. In mapped systems, the contents of AC3 are ignored when .UIEX is issued. In both mapped and unmapped systems, rescheduling of both the task and program environment (if a foreground/background system) will occur upon exit only if AC1 contains some non-zero value. If compatibility with RTOS is to be maintained, AC2 must be restored to the value it had upon entry to the user interrupt routine.

The format of this call is:

- AC1 - Zero only if rescheduling is to be suppressed.
- AC3 - Return address upon entry to routine (unmapped systems only).

.UIEX

Control returns to the point outside the user routine which was interrupted by the user device. No errors are possible from this call. This call can be issued in a single task environment.

Remove User Interrupt Servicing Program (.IRMV)

To prevent the system's recognition of user interrupts which have been previously identified by the .IDEF command, the .IRMV command is issued. Required input to this call is the user device code corresponding to the device which is to be removed.

The format of this call is:

AC0 - Device code.

.SYSTEM

.IRMV

error return

normal return

Remove User Interrupt Servicing Program (.IRMV) (Continued)

One possible error message may be given.

<u>AC2</u>	<u>Mnemonic</u>	<u>Meaning</u>
36	ERDNM	Illegal device code (>778) or attempt to remove a system device (i.e., one established at SYSGEN time).

Set the Data Channel Map (.STMAP)

User devices employing the data channel in mapped systems must issue a system call to set up the data channel map. (A separate map is maintained by the mapping hardware for data channel usage.) This call sets up the data channel map for the user device and returns in AC1 the logical address which should be sent to the device. Required inputs to this call are the user device code in AC0, and in AC1 the starting address of the device buffer in user address space.

The format of this call is:

```
.SYSTM  
.STMAP  
error return  
normal return
```

This call is a no-op when issued in unmapped systems. In mapped systems, two possible error conditions may occur.

<u>AC2</u>	<u>Mnemonic</u>	<u>Meaning</u>
36	ERDNM	Device code not previously identified as a data channel device.
74	ERMPR	Address outside address space (mapped systems only).

Modifying the Current Interrupt Mask (.SMSK)

Whenever a user interrupt occurs, the interrupt mask is ORed with the mask contained in DCTMS of the user DCT to produce the current interrupt mask. Nonetheless, it is possible in the service routine to produce a current mask which ignores the contents of DCTMS, producing a new mask which is the logical OR of the old mask (upon entry to the service routine) and a new value. If compatibility with RTOS is to be maintained, AC2 must be restored to the value it had upon entry to the user interrupt routine. This is done by task call .SMSK, whose format is as follows:

AC0 - New value to be ORed with old mask.

.SMSK
normal return

There is no error return possible from this call. This call may be issued in a single task environment.

WRITING USER POWER FAIL SERVICE

RTOS provides software support for the power fail/automatic restart option. Upon detection of a power loss, the system transfers control to a power fail routine which saves the status of accumulators 0 through 3, the PC and Carry.

When power is restored, if the console key is in the LOCK position, the message

****POWER FAIL****

is output on the system console and the state variables are restored before control resumes operation at the point where it was interrupted. If the console key was in the ON position when input power failed, the user must set the console switches to all zeroes (down) and START must be pressed when power is restored. This causes the console message to be output and state variables to be restored as when the key is in the LOCK position.

The following system devices are given power-up restart service by RDOS:

WRITING USER POWER FAIL SERVICE (Continued)

paper tape readers/punches
Teletypes
quad multiplexors
card readers
line printers
disks

Character devices may lose one or more characters during power up. Each card reader may lose up to 80 columns of information on a single card. Line printers may lose up to a single line of information. Since power up service for disks includes a complete re-read or re-write of the current disk block, no disk information is lost, although moving head disk units will require 30 to 40 seconds before disk operations can continue. Devices requiring operator intervention (like line printer, card readers, etc.) must receive such action if power was lost for an extended period of time. No power up service is provided for magnetic tape or cassette units.

Power up service for special user devices (or for magnetic tape or cassette units) must be provided by the user via the system call `.IDEF`. The format of this call when used to identify user power up service is as follows:

AC0	-	778
AC1	-	Starting address of user power up service routine.

`.SYSTEM`
`.IDEF`
error return
normal return

The error return is never taken.

Exit from a user power-up service routine in both mapped and unmapped systems forces rescheduling and is accomplished by task call `.UPEX`. The same restrictions applying to the use of system and task calls in a user interrupt service routine apply to a user power fail routine.

Exit from a Power Fail Service Routine (.UPEX)

Upon entering a user power fail service routine, AC3 will contain the address required for exit from the routine. To return from the user power fail routine in an unmapped environment, AC3 must be loaded with this return address and task call .UPEX must be issued. In mapped systems, the value input in AC3 when this call is issued is ignored.

The format of this call is:

AC3 - Return address upon entry to the routine (unmapped systems only).

.UPEX

Control returns to the location which was interrupted by a power failure. No error return or normal return need be reserved. .UPEX can be issued in a single task environment. Note that this call can be issued only in revision 03 and higher revisions of RDOS.

USER PROGRAMMED CLOCK

Two system commands, .DUCLK and .RUCLK, are available to permit the definition and removal of a user clock driven by the system's real time clock (RTC). A user clock routine gains control at user-definable intervals. When one of these intervals expires, control goes to a user-specified routine outside the current single or multitask environment. No task calls (other than .IXMT, .SMSK, and .UCEX) may be issued from this interrupt servicing routine.

Define a User Clock (.DUCLK)

This command permits the definition of a user clock. When the user-defined interval expires, the task scheduler and multitask environment--if any-- are placed in suspension, and control goes to a user-specified routine. No system or task call (except .UCEX, .SMSK, .IXMT, and .UCEX) may be issued from this user routine. Moreover, in NOVA systems assembly instruction INTEN cannot be issued.

Define a User Clock (.DUCLK) (Continued)

The format of this call is:

- AC0 - Integer number of RTC cycles to elapse between each user clock interruption.
- AC1 - Address of user routine to receive control when each interval expires.

.SYSTEM
.DUCLK
error return
normal return

If the error return is taken, the following error code is issued:

<u>AC2</u>	<u>Mnemonic</u>	<u>Meaning</u>
45	ERIBS	A user clock already exists

Upon a user clock interrupt, AC3 will contain the address of the return upon entry to the routine specified in .DUCLK. To return from the user clock routine in an unmapped environment, AC3 must be loaded with the return address that it contained upon entry to the routine and task call .UCEX is issued. In mapped systems, the value input in AC3 when this call is issued is ignored. In both mapped and unmapped systems, rescheduling of both the task environment and the program environment (if a foreground/background system) will occur upon exit only if AC1 contains some non-zero value.

The format of this call is:

- AC1 - Zero only if rescheduling is to be suppressed.
- AC3 - Return address upon entry to routine (unmapped systems only).

.UCEX

Control returns to the point outside the user routine which was interrupted by the user clock. No errors are possible from this call. This call can be issued in a single task environment.

Remove a User Clock (.RUCLK)

This system command removes a previously defined user clock from the system.

Remove a User Clock (.RUCLK) (Continued)

The format of this call is:

```
.SYSTEM  
.RUCLK  
error return  
normal return
```

The error return must be reserved, but it is never taken.

EXAMPLES OF USER SERVICED INTERRUPTS

This section illustrates two implementations of user-written interrupt servicing programs for devices not incorporated into the system at SYSGEN time.

Analog to Digital Converter

The first of the two illustration programs is an analog to digital converter driver, found on page 3-12. This driver consists of four subroutines which can be called from a user program, the interrupt servicing subroutine, and a Device Control Table. The following is a line by line analysis of the A/D driver.

Lines 6 - 10 show that the title of the driver is AIDDEV, that four entry points exist for user access of this driver, and that the driver program is normal relocatable (i.e., that it does not use any page zero locations). Line 15 gives the address of the Device Control Table which is defined in lines 17-19. This is an abbreviated DCT as discussed earlier.

The first location in the DCT (line 17) points to an 8-word state save area defined on line 30. The second entry in the table is the hardware mask that is set while the A/D interrupt is serviced. 1B8 indicates that all other devices can interrupt the A/D converter. The third and final entry in the DCT is the address of the interrupt servicing program, AINTS. AINTS is found on page 4 of the driver listing, and will be discussed later.

Lines 25 - 29 contain additional variables and constants which are required while servicing requests to this handler. These values are, in order, the device code of the A/D converter, a busy/done status flag, pointers to the address and data tables, and the number of points to be read for this call.

Page 2 of the driver listing illustrates two subroutines called by the user to attach (detach) the A/D converter interrupt servicing routine to (or from) the RDOS dispatch table, ITBL. The first routine, AIDDEF, clears the ACTIV flag to indicate that there are no active requests in the handler for data. AIDDEF then issues the system call .IDEF to define this handler and introduce it to RDOS. An error return

Analog to Digital Converter (Continued)

could result if the A/D converter device code (218) already was assigned to some other device. This would happen if a second call to AIDEF were issued without any intervening call to AICLR. AICLR removes the A/D routine from the RDOS dispatch table.

Page 3 of the driver listing contains two subroutines, the analog read random routine (AIRDR) and the read request routine (AICLK). The first of these routines, AIRDR, is called with AC2 containing the address of a control table. This table is described on page 1 of the listing (lines 34 - 50). This subroutine first checks whether the device is currently active on another request by testing the ACTIV flag defined earlier (page 1, line 26). If the handler is already active, an error return is made to the user. If the handler is not active, the ACTIV flag is set and the request is initiated. The address of the control table is saved in the active switch and the number of points to be read is moved into the handler from the table. Since this routine performs reads of analog inputs, two tables must be provided by the user. One table must contain the input point addresses to be read, and the other table (of equal or greater size) is used to store the analog input readings. The addresses of these two tables is given in the control table. Before returning control back to the caller, the address of the first point is fetched, sent to the converter, and the converter is started.

The fourth routine, AICLK, is used to check the status of a call made to the converter. There are two returns to the caller: busy and call completed. Page 4 of the driver listing contains the interrupt service routine, AINTS. This routine is entered from the RDOS interrupt dispatch program with AC2 containing the address of the A/D converter DCT and AC3 containing the address of the RDOS interrupt dismissal program. AINTS reads the new converted value, saves it in the user-supplied data buffer, and then checks to see if there are additional points to be read. If there are, it initiates the next conversion. If this was the last point to be read, AINTS sets the status of the request to indicate that the call is completed and the handler is available for the next user request.

LICENSED MATERIAL - PROPERTY OF DATA GENERAL CORPORATION

0001 AIDEV MACRO REV 02

01:10:26 12/08/73

```

01
02      ; ANALOG TO DIGITAL CONVERTER DEVICE DRIVER
03      ; -----
04      ; -----
05
          .TITL AIDEV
07
          .ENT AIDEF,AICLR,AIRDR,AICLK
08
          .NREL
09
10
11
12      ; DEVICE CONTROL TABLE LAYOUT
13      ; -----
14
15 00000'000001'ADDCT:  USDCT      ; ADDRESS OF ADCV DCT.
16
17 00001'000011'USDCT:  AISAVE     ; INTERRUPT STATE SAVE AREA
18 00002'000200          1B8       ; MASK WORD
19 00003'000063'        AINTS     ; INTERRUPT ROUTINE ADDRESS
20
21      ; ADDITIONAL VARIABLES FOR ADCV HANDLER
22      ; -----
23
24 00004'000021 DCODE:  ADCV       ; DEVICE CODE OF A/D CONVERTER
25 00005'000000 ACTIV:  0         ; STATUS FLAG
26 00006'000000 DAPTR:  0         ; DATA ADDRESS ARRAY POINTER
27 00007'000000 DVPTR:  0         ; DATA VALUE ARRAY POINTER
28 00010'000000 DTCNT:  0         ; DATA COUNT STORAGE
29
30 00011'000010 AISAVE:  .BLK 10   ; STATE SAVE AREA
31
32
33      ; USER CONTROL TABLE LAYOUT
34      ; -----
35
36      ; WORD 0      STATUS FLAG
37      ;              0 = CALL COMPLETED SUCCESSFULLY
38      ;              -VE = REQUEST BEING PROCESSED
39
40      ; WORD 1      ADDRESS TABLE POINTER
41
42      ; WORD 2      DATA STORAGE TABLE POINTER
43
44      ; WORD 3      NUMBER OF POINTS TO BE READ
45
46      000000 .DUSR   STAT=0
47      000001 .DUSR   ATPTR=1
48      000002 .DUSR   VTPTR=2
49      000003 .DUSR   NPT=3
50

```


LICENSED MATERIAL - PROPERTY OF DATA GENERAL CORPORATION

10002 AIDEV

```

01
02           ; INITIALIZE THE A/D HANDLER
03           ; -----
04
05           ; CALLING SEQUENCE:
06           ;       JSR AIDEF
07           ;       <ERROR RETURN> ; DEVICE CODE IN USE ALREADY
08           ;       <NORMAL RETURN>
09
10 00021'054016 AIDEF: STA 3 USP           ; SAVE RETURN ADDRESS
11 00022'102400          SUB 0 0           ; SET ROUTINE NOT BUSY
12 00023'040762          STA 0 ACTIV
13 00024'020760          LDA 0 DCODE       ; GET DEVICE CODE
14 00025'024753          LDA 1 ADDCT      ; GET DCT ADDRESS
15 00026'006017          .SYSTEM         ; ATTACH TO RDOS INTERRUPT SYSTEM
16 00027'021007          .IDEF
17 00030'001400          JMP 0 3           ; ERROR RETURN
18 00031'001401          JMP 1 3           ; NORMAL RETURN
19
20
21           ; DETACH THE INTERRUPT SERVICING ROUTINE
22           ; -----
23
24           ; CALLING SEQUENCE:
25           ;       JSR AICLR
26           ;       <NORMAL RETURN>
27
28
29 00032'054016 AICLR: STA 3 USP           ; SAVE RETURN ADDRESS
30 00033'020751          LDA 0 DCODE
31 00034'006017          .SYSTEM         ; DETACH FROM RDOS INTERRUPT SYSTEM
32 00035'021010          .IRMV
33 00036'001400          JMP 0 3
34 00037'001400          JMP 0 3           ; NORMAL RETURN

```

LICENSED MATERIAL - PROPERTY OF DATA GENERAL CORPORATION

10003 AIDEV

```

01
02
03      ; ANALOG READ RANDOM SUBROUTINE
04      ; -----
05
06      ; INPUT:
07      ;       AC2=CONTROL TABLE ADDRESS
08
09      ; CALLING SEQUENCE:
10      ;       JSR AIRDR
11      ;       <ERROR RETURN>  ; HANDLER ALREADY ACTIVE
12      ;       <NORMAL RETURN>
13
14
15 00040'020745 AIRDR:  LDA 0 ACTIV      ; PICKUP STATUS FLAG
16 00041'101004      MOV 0 0 SZR      ; CHECK IT
17 00042'001400      JMP 0 3        ; ALREADY ACTIVE--ERROR RETURN
18 00043'050742      STA 2 ACTIV      ; SAVE CONTROL BLOCK ADDRESS
19 00044'021003      LDA 0 NPT 2     ; GET # POINTS TO BE READ
20 00045'040743      STA 0 DTCNT     ; SAVE AS DATA COUNT
21 00046'021001      LDA 0 ATPTR 2   ; GET STARTING ADDRESS OF POINT ADDRESSE
22 00047'040737      STA 0 DAPTR     ; SAVE POINT ADDRESS TABLE
23 00050'021002      LDA 0 VTPTR 2   ; GET VALUE STORAGE TABLE ADDRESS
24 00051'040736      STA 0 DVPTR     ; SAVE FOR STORING VALUES
25 00052'102000      ADC 0 0        ; SET REQUEST ACTIVE STATUS
26 00053'041000      STA 0 STAT 2
27 00054'022732      LDA 0 0DAPTR    ; GET FIRST POINT ADDRESS
28 00055'061121      DOAS 0 ADCV     ; START UP CONVERSION
29 00056'001401      JMP 1 3        ; TAKE NORMAL RETURN
30
31
32      ; ANALOG INPUT READ REQUEST CHECK SUBROUTINE
33      ; -----
34
35      ; INPUT:
36      ;       AC2=CONTROL TABLE ADDRESS
37
38      ; CALLING SEQUENCE:
39      ;       JSR AICLK
40      ;       <BUSY RETURN>
41      ;       <COMPLETED RETURN>
42
43
44 00057'021000 AICLK:  LDA 0 STAT 2     ; GET CALL STATUS
45 00060'101004      MOV 0 0 SZR
46 00061'001400      JMP 0 3        ; REQUEST BEING PROCESSED
47 00062'001401      JMP 1 3        ; CALL COMPLETED

```

LICENSED MATERIAL - PROPERTY OF DATA GENERAL CORPORATION

0004 AIDEV

```

01
02
03      / ANALOG TO DIGITAL CONVERTER INTERRUPT ROUTINE
04      / -----
05
06      / INPUT:
07      /      AC2=DCT ADDRESS
08      /      AC3=INTERRUPT DISMISSAL ROUTINE ADDRESS
09
10 00063'062621 AINTS:  DICC 0 ADCV      / READ VALUE--CLEAR CONVERTER DONE FLAG
11 00064'042723      STA 0 #DVPTR    / SAVE NEWLY READ DATA VALUE
12 00065'014723      DSZ DTCNT     / DECREMENT DATA COUNTER
13 00066'000405      JMP AIMORE    / MORE TO COME
14 00067'102400      SUB 0 0        / REQUEST COMPLETE
15 00070'042715      STA 0 #ACTIV   / SET USER DONE FLAG
16 00071'040714      STA 0 ACTIV    / SET CONVERTER NON-BUSY
17 00072'001400      JMP 0 3        / DISMISS THE INTERRUPT
18
19 00073'010713 .AIMORE: ISZ DAPTR     / INCREMENT ADDRESS POINTER
20 00074'010713      ISZ DVPTR     /      "      DATA TABLE POINTER
21 00075'022711      LDA 0 #DAPTR   / SETUP NEXT CONVERSION
22 00076'061121      DOAS 0 ADCV    / START IT
23 00077'001400      JMP 0 3        / DISMISS THE INTERRUPT
24
25      .END
    
```

LICENSED MATERIAL - PROPERTY OF DATA GENERAL CORPORATION

0005 AIDEV

ACTIV	000005'		1/25	2/12	3/15	3/18	4/15	4/16
ADDOCT	000000'		1/15	2/14				
AICHK	000057'	EN	1/08	3/44				
AICLR	000032'	EN	1/08	2/29				
AIDEF	000021'	EN	1/08	2/10				
AIMOR	000073'		4/13	4/19				
AINTS	000063'		1/19	4/10				
AIRDR	000040'	EN	1/08	3/15				
AISAV	000011'		1/17	1/30				
DAPTR	000006'		1/26	3/22	3/27	4/19	4/21	
DCODE	000004'		1/24	2/13	2/30			
DTCNT	000010'		1/28	3/20	4/12			
DVPTR	000007'		1/27	3/24	4/11	4/20		
USDCT	000001'		1/15	1/17				

External Interrupt Recognition

This driver program illustrates an interrupt servicing routine which readies a user task as a result of an external interrupt. This program illustrates a type 4066 digital interface device driver.

Page 1 of the listing is almost identical in form to the first page of the A/D driver listing discussed earlier. The only difference is in the variables and storage required by the digital interface. Here, the user must supply the address of a communications core location. When a call is made to attach the interrupt routine, address to the RDOS dispatch table (page 2 of the listing), AC0 must contain the communications core address. After attaching the interrupt to the RDOS dispatch vector table, the initialization routine arms the digital interface so that it can cause an interrupt.

When the external interrupt occurs, control is dispatched to the digital interface servicing routine, starting on line 38 of page 2. After the service routine gains control, it reads a sixteen bit digital register provided by the interface. If the register's contents are non-zero, the contents are transmitted to a user task using the task call .IXMT; after this, the interrupt is dismissed. If the register's contents is zero, the interrupt is simply dismissed. Upon dismissal of the interrupt, the system re-arms the digital interface interrupts.

A practical example of the use of such a scheme would be the servicing of an operator's console. Such a console would generate an external interrupt indicating that the operator desires some form of action by the computer; the sixteen bit register contents (selected by the console operator) would indicate exactly the type of action desired.

LICENSED MATERIAL - PROPERTY OF DATA GENERAL CORPORATION

0001 DIDEV MACRO REV 02

0111102 12/08/73

```

01
02
03 ; DIGITAL INTERFACE (TYPE 4066) DEVICE DRIVER
04 ; -----
05 ; -----
06
      .TITL DIDEV
08
09      .ENT DIDEF,DICLR
10
11      .EXTN ,IXMT
12
13      .NREL
14
15      000042 .DUSR   DIO=42           ; DEVICE CODE DEFINITION
16
17      ; DEVICE CONTROL TABLE LAYOUT
18      ; -----
19
20 00000'000001'DIDCT:  USDCT           ; ADDRESS OF DIO DCT
21
22 00001'000004'USDCT:  DISAVE          ; INTERRUPT STATE SAVE AREA
23 00002'000400        1B7             ; INTERRUPT MASK
24 00003'000036'      DINTS           ; INTERRUPT ROUTINE ADDRESS
25
26      ; ADDITIONAL VARIABLES AND STORAGE USED BY HANDLER
27      ; -----
28
29 00004'000010 DISAVE:  .BLK 10        ; STATE SAVE AREA
30 00014'000042 DCODE:  DIO           ; DEVICE CODE
31 00015'000000 DICOM:  0             ; COMMUNICATIONS ADDRESS

```

LICENSED MATERIAL - PROPERTY OF DATA GENERAL CORPORATION

10002 DIDEV

```

01
02      ; INITIALIZE THE DIGITAL INTERFACE
03      ; -----
04
05      ; INPUT: AC0=MESSAGE ADDRESS
06
07      ; CALLING SEQUENCE:
08      ;      JSR DIDEF
09      ;      <ERROR RETURN> ; DEVICE CODE (DIO) ALREADY IN USE
10      ;      <NORMAL RETURN>
11
12 00016'054016 DIDEF: STA 3 USP
13 00017'040776      STA 0 DICOM      ; SAVE COMMUNICATIONS ADDRESS
14 00020'020774      LDA 0 DCODE
15 00021'024757      LDA 1 DIDCT
16 00022'006017      .SYSTEM          ; ATTACH TO RDOS INTERRUPT SYSTEM
17 00023'021007      .IDEF
18 00024'001400      JMP 0 3          ; ERROR RETURN
19 00025'060142      NIOS DIO         ; ARM THE EXTERNAL INTERRUPT
20 00026'001401      JMP 1 3          ; NORMAL RETURN
21
22      ; DETACH THE INTERRUPT SERVICING ROUTINE & CLEAR DEVICE
23      ; -----
24
25      ; CALLING SEQUENCE:
26      ;      JSR DICLR
27      ;      <NORMAL RETURN>
28
29 00027'054016 DICLR: STA 3 USP
30 00030'060242      NIOC DIO         ; CLEAR DIO DEVICE
31 00031'020763      LDA 0 DCODE
32 00032'006017      .SYSTEM          ; DETACH HANDLER
33 00033'021010      .IRMV
34 00034'001400      JMP 0 3          ; NORMAL RETURN
35 00035'001400      JMP 0 3          ;      "      "
36
37

```

LICENSED MATERIAL - PROPERTY OF DATA GENERAL CORPORATION

```

38           ; DIGITAL INTERFACE EXTERNAL INTERRUPT HANDLER
39           ; -----
40
41           ; INPUT: AC2=DCT ADDRESS
42           ;           AC3=INTERRUPT DISMISSAL ADDRESS
43
44 00036!064542 DINTS: DIAS 1 DIO           ; READ 16 BIT REGISTER
45 00037!125005          MOV 1 1 SNR       ; VALUE ZERO ?
46 00040!001400          JMP 0 3         ; YES--DISMISS INTERRUPT
47
48 00041!054412          STA 3 DISMISS    ; NO--SAVE DISMISSAL ROUTINE ADDRESS
49 00042!050410          STA 2 SAVE2     ; SAVE AC2 ALSO
50 00043!102400          SUB 0 0         ; CLEAR SIGNAL ADDRESS
51 00044!042751          STA 0 @DICOM    ;
52 00045!020750          LDA 0 DICOM     ; GET SIGNAL ADDRESS
53 00046!177777          .IXMT          ; WAKE UP USER TASK
54 00047!000401          JMP .+1       ;
55 00050!030402          LDA 2 SAVE2     ; RESTORE AC2
56 00051!002402          JMP @DISMISS   ; DISMISS THE INTERRUPT
57
58 00052!000000 SAVE2: 0           ; TEMPORARY STORAGE FOR AC2
59 00053!000000 DISMISS: 0        ; INTERRUPT DISMISSAL ADDRESS

                .END

```

LICENSED MATERIAL - PROPERTY OF DATA GENERAL CORPORATION
0003 DIDEV

DCODE	000014!		1/30	2/14	2/31	
DICLR	000027!	EN	1/09	2/29		
DICOM	000015!		1/31	2/13	2/51	2/52
DIDCT	000000!		1/20	2/15		
DIDEF	000016!	EN	1/09	2/12		
DINTS	000036!		1/24	2/44		
DISAV	000004!		1/22	1/29		
DISMI	000053!		2/48	2/56	2/59	
SAVE2	000052!		2/49	2/55	2/58	
USDCT	000001!		1/20	1/22		
.IXMT	000046!	XN	1/11	2/53		

Multiprocessor Communications Adapter

RDOS revisions 03 and higher extend system support to option 4038, the multi-processor communications adapter (MCA). Nonetheless, the MCA can be treated as a user-defined device. The following program illustrates such an MCA driver and illustrates the .STMAP call which is required when running data channel devices in the user area on a mapped system.

Page 1 of the listing illustrates the MCA receiver and transmitter DCT's, RUSDCT and TUSDCT. These DCT's are standard user DCT's; note that bit zero is set to one. This is required when the devices are identified to the system in subroutine MIDEF, since they are data channel devices used in a mapped system.

The MCA handler must be initialized before use, and this is done by a user call to MIDEF illustrated on page 5 of the listing. This routine identifies both the transmitter and receiver to the system and defines the number of 1024₁₀ word blocks which will be required for data channel transfers (two blocks each for transmissions and receptions). Additionally, this routine converts the logical buffer addresses, input to the initialization call, to physical addresses required for data channel transfers in mapped machines (lines 34-43). That is, whenever a user program sets aside a series of logical locations as a data channel buffer, .STMAP returns the actual address assignments made by the mapping unit. All addresses in mapped user programs are logical, since the user program is unaware of the actual locations assigned by the hardware; the actual address assignments, however, must be sent to data channel devices. Beyond sending these addresses to the data channel devices, the user program need not be concerned with the actual address assignments.

Since MIDEF identifies the MCA units to the system only once, MIDEF can be invoked by the user whenever different read/write buffers are to be used.

The read and write subroutines proper are found on listing pages 2 and 3. Each of these routines first checks to see if the device is busy; if so, the error return is taken. Otherwise, each routine outputs the word count and logical starting address to its device, and the transmit routine also outputs the receiver number to the transmitter. Then each routine waits, via a call to .REC, until its associated interrupt processor (page 4 of the listing) readies it via an interrupt message call, .IXMT, indicating that the process is complete. The read/write routine then resets its busy flag and takes the call's normal return.

LICENSED MATERIAL - PROPERTY OF DATA GENERAL CORPORATION

0001 MCAUR MACRO REV 02

13:12:24 12/18/73

```

01
02
03
04      ; MULTIPROCESSOR COMMUNICATIONS ADAPTER (TYPE 4038)
05      ; -----
06      ; -----
07
          .TITLE MCAUR      ; MCA DRIVER
09
          .ENT MCAUR MCAWT MIDEF
10
          .EXTN .IXMT, .REC
11
          .NREL
12
13
14
15
16      ; DEVICE CONTROL TABLE LAYOUT
17      ; -----
18
19 00000'100002'MRDCT: 1B0+RUSDCT      ; ADDRESS OF MCA RECEIVER DCT
20 00001'100005'MTDCT: 1B0+TUSDCT      ; ADDRESS OF MCA TRANSMITTER D
21
22 00002'000012'RUSDCT: MSAVE          ; INTERRUPT STATE SAVE AREA
23 00003'000010          1B12          ; MASK WORD
24 00004'000101'          RINTS        ; INTERRUPT ROUTINE ADDRESS
25
26 00005'000012'TUSDCT: MSAVE          ; INTERRUPT STATE SAVE AREA
27 00006'000010          1B12          ; MASK WORD
28 00007'000106'          TINTS        ; INTERRUPT ROUTINE ADDRESS
29
30      ; ADDITIONAL VARIABLES FOR MCA DRIVER
31      ; -----
32
33 00010'000007 RCODE: MCAUR          ; RECEIVER DEVICE CODE
34 00011'000006 TCODE: MCAUR          ; TRANSMITTER DEVICE CODE
35 00012'000010 MSAVE: .BLK 10        ; STATE SAVE AREA

```

LICENSED MATERIAL - PROPERTY OF DATA GENERAL CORPORATION

10002 MCADR

```

01
02           ; MCA READ SUBROUTINE
03           ; -----
04
05           ; INPUT:-
06           ;     AC0=WORD COUNT
07
08           ; CALLING SEQUENCE:-
09           ;     JSR MCARD
10           ;     <ERROR RETURN> ; HANDLER ALREADY ACTIVE
11           ;     <NORMAL RETURN>
12
13           ; OUTPUT:-
14           ;     AC1=MCA STATUS WORD
15
16 00022'030420 MCARD:  LDA 2 RBUSY           ; PICKUP STATUS FLAG
17 00023'151004          MOV 2 2 SZR           ; CHECK IT
18 00024'000422          JMP RERTN           ; DEVICE BUSY
19 00025'010415          ISZ RBUSY           ; SET ROUTINE BUSY
20 00026'050416          STA 2 RSIGL+1       ; RESET DONE SIGNAL
21 00027'054412          STA 3 RRTN           ; SAVE RETURN ADDRESS
22 00030'062007          DOB 0 MCA          ; SEND WORD COUNT TO RECEIVER
23 00031'024414          LDA 1 RADDR        ; PICK UP LOGICAL STARTING ADDRESS
24 00032'065107          DOAS 1 MCA        ; SEND STARTING ADDRESS
25 00033'020410          LDA 0 RSIGL        ; WAIT FOR OPERATION COMPLETE FLAG
26 00034'177777          .REC
27 00035'102400          SUB 0 0
28 00036'040404          STA 0 RBUSY        ; RESET RECEIVER BUSY FLAG
29 00037'010402          ISZ RRTN
30 00040'002401          JMP @RRTN         ; TAKE NORMAL RETURN
31
32 00041'000000 RRTN:   0
33 00042'000001 RBUSY: 1
34
35 00043'000044'RSIGL: ,+1
36 00044'000001          .BLK 1
37
38 00045'000000 RADDR: 0           ; RECEIVER BUFFER LOGICAL ADDRESS
39                               ; AS DEFINED BY .STMAP SYSTEM CALL
40                               ; DURING DEVICE INITIALIZATION .
41
42 00046'030402 RERTN:  LDA 2 RDVBSY        ; DEVICE ALREADY BUSY
43 00047'001400          JMP 0 3           ; PROCESSING A PREVIOUS CALL
44
45 00050'000047 RDVBSY: ERSIM       ; SIMULTANEOUS READS OR WRITES NOT ALLOW

```

LICENSED MATERIAL - PROPERTY OF DATA GENERAL CORPORATION

10003 MCADR

```

01
02           ; MCA WRITE SUBROUTINE
03           ; -----
04
05           ; INPUT:-
06           ;       AC0=WORD COUNT
07           ;       AC1=RECEIVER NUMBER IN BITS 0-3 (1-17(8))
08
09           ; CALLING SEQUENCE:-
10           ;       JSR MCAWT
11           ;       <ERROR RETURN>
12           ;       <NORMAL RETURN>
13
14           ; OUTPUT:-
15           ;       AC1=MCA STATUS WORD FOR TRANSMITTER
16
17 00051'030423 MCAWT: LDA 2 TBUSY           ; PICKUP STATUS FLAG
18 00052'151004      MOV 2 2 SZR           ; CHECK IT
19 00053'000773      JMP TERTN            ; ALREADY ACTIVE - ERROR RETURN
20 00054'010420      ISZ TBUSY           ; INDICATE DEVICE IS BUSY
21 00055'044423      STA 1 RECDV         ; SAVE RECEIVER NUMBER
22 00056'050420      STA 2 TSIGL+1      ; RESET TRANSMITTER DONE SIGNAL
23 00057'054414      STA 3 TRTN         ; SAVE RETURN ADDRESS
24 00060'062006      DOB 0 MCAI
25 00061'024416      LDA 1 TADDR         ; PICKUP LOGICAL STARTING ADDRESS
26 00062'065006      DOA 1 MCAI
27 00063'030415      LDA 2 RECDV
28 00064'073106      DOCS 2 MCAI
29 00065'020410      LDA 0 TSIGL         ; WAIT FOR OPERATION COMPLETE
30 00066'000034'     ,REC
31 00067'102400      SUB 0 0
32 00070'040404      STA 0 TBUSY         ; RESET TRANSMITTER BUSY FLAG
33 00071'010402      ISZ TRTN
34 00072'002401      JMP 0 TRTN         ; TAKE NORMAL RETURN TO USER
35
36 00073'000000 TRTN: 0
37 00074'000001 TBUSY: 1
38
39 00075'000076' TSIGL: ,+1
40 00076'000001      ,BLK 1
41
42 00077'000000 TADDR: 0                 ; TRANSMITTER BUFFER LOGICAL ADDRESS
43                                     ; AS DEFINED BY ,STMAP SYSTEM CALL
44                                     ; DURING DEVICE INITIALIZATION .
45
46           000046' TERTN= RERTN         ; TRANSMITTER BUSY ERROR RETURN
47
48
49 00100'000000 RECDV: 00                 ; RECEIVER DEVICE NUMBER

```

LICENSED MATERIAL - PROPERTY OF DATA GENERAL CORPORATION

10004 MCA DR

01

02 ; MCA RECEIVER INTERRUPT PROCESSOR

03 ; -----

04

05 00101'066607 RINTS: DICC 1 MCA R ; CLEAR RECEIVER/READ STATUS

06 00102'020741 LDA 0 R SIGL ; GET SIGNAL ADDRESS

07 00103'177777 .IXMT

08 00104'000406 JMP DISMISS

09 00105'000405 JMP DISMISS

10

11 ; MCA TRANSMITTER INTERRUPT PROCESSOR

12 ; -----

13

14 00106'066606 TINTS: DICC 1 MCA T ; CLEAR TRANSMITTER/READ STATUS

15 00107'020766 LDA 0 T SIGL ; GET SIGNAL ADDRESS

16 00110'000103' .IXMT

17 00111'000401 JMP .+1

18 00112'030402 DISMISS: LDA 2 DSCDE ; GET DISMISS CODE

19 00113'060202 NIOC MAP ; TRIGGER THE MPU

20

21 00114'000003 DSCDE: 03 ; CODE FOR DISMISSING THE INTERRUPT

22 ; UNDER A NOVA 840 SYSTEM

LICENSED MATERIAL - PROPERTY OF DATA GENERAL CORPORATION

```

10005 MCAOR
01
02
03      ; INITIALIZE THE MCA HANDLER
04      ; -----
05
06      ; INPUT:-
07      ;      AC0=STARTING CORE ADDRESS OF RECEIVER BUFFER
08      ;      AC1=STARTING CORE ADDRESS OF TRANSMITTER BUFFER
09
10      ; CALLING SEQUENCE:-
11      ;      JSR MIDEF
12      ;      <ERROR RETURN>
13      ;      <NORMAL RETURN>
14
15 00115'054437 MIDEF:  STA 3 MIRTN      ; SAVE RETURN ADDRESS
16 00116'040724      STA 0 RBUSY      ; SAVE USER BUFFER ADDRESSES
17 00117'044755      STA 1 TBUSY
18 00120'010435      ISZ MCA TV      ; DEVICE ALREADY INTRODUCED TO SYSTEM?
19 00121'000414      JMP MACTV      ; YES, JUST GET LOGICAL ADDRESSES
20 00122'020666      LDA 0 RCODE      ; RECEIVER DEVICE CODE
21 00123'024655      LDA 1 MRDCT      ; RECEIVER DEVICE CONTROL TABLE
22 00124'030432      LDA 2 C2        ; NUMBER OF 1K BLOCKS
23 00125'006017      .SYSTEM        ; DEFINE DEVICE TO SYSTEM
24 00126'021007      .IDEF
25 00127'002425      JMP 0MIRTN      ; TAKE ERROR RETURN
26 00130'020661      LDA 0 TCODE      ; TRANSMITTER DEVICE CODE
27 00131'024650      LDA 1 MTDCT      ; TRANSMITTER DEVICE CONTROL TABLE
28 00132'006017      .SYSTEM        ; DEFINE DEVICE TO SYSTEM
29 00133'021007      .IDEF
30 00134'002420      JMP 0MIRTN      ; TAKE ERROR RETURN
31 00135'054420 MACTV:  STA 3 MCA TV      ; SET ACTIVE FLAG
32 00136'020652      LDA 0 RCODE      ; GET DEVICE CODE
33 00137'024703      LDA 1 RBUSY      ; USER BUFFER ADDRESS
34 00140'006017      .SYSTEM        ; GET LOGICAL ADDRESS OF BUFFER
35 00141'021035      .STMAP
36 00142'002412      JMP 0MIRTN      ; TAKE ERROR RETURN
37 00143'044702      STA 1 RADDR      ; SAVE LOGICAL ADDRESS
38 00144'020645      LDA 0 TCODE      ; DEVICE CODE
39 00145'024727      LDA 1 TBUSY      ; GET BUFFER ADDRESS
40 00146'006017      .SYSTEM        ; GET LOGICAL ADDRESS OF USER BUFFER
41 00147'021035      .STMAP
42 00150'002404      JMP 0MIRTN      ; TAKE ERROR RETURN
43 00151'044726      STA 1 TADDR      ; SAVE LOGICAL ADDRESS OF TRANSMITTER B
44 00152'010402      ISZ MIRTN
45 00153'002401      JMP 0MIRTN      ; NORMAL RETURN
46
47 00154'000000 MIRTN:  0              ; RETURN ADDRESS
48
49
50 00155'177777 MCA TV:  -1            ; MCA DEVICE ACTIVE FLAG
51 00156'000002 C2:    2              ; NUMBER OF 1024 WORD BLOCKS FOR MCA
52
53      .END

```

LICENSED MATERIAL - PROPERTY OF DATA GENERAL CORPORATION
 0006 MCAOR

C2	000156!		5/22	5/51					
DISMI	000112!		4/08	4/09	4/18				
DSCDE	000114!		4/18	4/21					
MACTV	000135!		5/19	5/31					
MCARD	000022!	EN	1/10	2/16					
MCATV	000155!		5/18	5/31	5/50				
MCAWT	000051!	EN	1/10	3/17					
MIDEF	000115!	EN	1/10	5/15					
MIRTN	000154!		5/15	5/25	5/30	5/36	5/42	5/44	5/45
			5/47						
MRDCT	000000!		1/19	5/21					
MSAVE	000012!		1/22	1/26	1/35				
MTDCT	000001!		1/20	5/27					
RADDR	000045!		2/23	2/38	5/37				
RBUSY	000042!		2/16	2/19	2/28	2/33	5/16	5/33	
RCODE	000010!		1/33	5/20	5/32				
RDVBS	000050!		2/42	2/45					
RECDV	000100!		3/21	3/27	3/49				
RERTN	000046!		2/18	2/42	3/46				
RINTS	000101!		1/24	4/05					
RRTN	000041!		2/21	2/29	2/30	2/32			
RSIGL	000043!		2/20	2/25	2/35	4/06			
RUSDC	000002!		1/19	1/22					
TADDR	000077!		3/25	3/42	5/43				
TBUSY	000074!		3/17	3/20	3/32	3/37	5/17	5/39	
TCODE	000011!		1/34	5/26	5/38				
TERTN	000046!		3/19	3/46					
TINTS	000106!		1/28	4/14					
TRTN	000073!		3/23	3/33	3/34	3/36			
TSIGL	000075!		3/22	3/29	3/39	4/15			
TUSDC	000005!		1/20	1/26					
.IXMT	000110!	XN	1/12	4/07	4/16				
.REC	000066!	XN	1/12	2/26	3/30				

INDEX

<p>AC0 2-4f *</p> <p>AC1 2-4f</p> <p>AC2 2-4f</p> <p>ACHR 2-31f</p> <p>AGSUB 2-5, 2-7ff</p> <p>analog to digital converter 3-10ff</p> <p>beads 2-16ff</p> <p>BGSUB 2-5, 2-7ff</p> <p>cassette 2-1</p> <p>\$CDR/\$CDR1 1-3,2-1,2-7,2-18f,3-7</p> <p>CLEAR 2-9,2-12</p> <p>CLSI 2-21</p> <p>CLSO 2-21</p> <p>CMPWD 2-9,2-11ff</p> <p>COSER 2-27,2-31</p> <p>CSP 2-5ff</p> <p>DC 2-16ff</p> <p>DCHMP 1-11</p> <p>DCHNM 1-11</p> <p>DCCRQ 1-10</p> <p>DCDST 1-10</p> <p>DCNBK 1-10</p> <p>DCNB1 1-10</p> <p>DCSTR 1-10</p> <p>DCTBC 1-8</p> <p>DCTBD 1-9</p> <p>DCTBP 1-8</p> <p>DCTBS 1-7,3-1</p> <p>DCTCC 1-9</p> <p>DCTCD 1-8</p> <p>DCTCH 1-7</p> <p>DCTDC 1-9</p> <p>DCTDP 1-9</p> <p>DCTDT 1-8, 2-19f</p> <p>DCTEX 1-8</p> <p>DCTIN 1-10</p> <p>DCTIS 1-7,3-1</p> <p>DCTLK 1-9</p> <p>DCTLK 1-9</p>	<p>DCTMN 1-10</p> <p>DCTMS 1-7,3-1</p> <p>DCTNH 1-10</p> <p>DCTNS 1-10</p> <p>DCTOF 1-9</p> <p>DCTON 1-9</p> <p>DCTOP 1-9</p> <p>DCTPC 1-8</p> <p>DCTPP 1-9</p> <p>DCTPR 1-9</p> <p>DCTQL 1-9</p> <p>DCTQP 1-9,2-17</p> <p>DCTQS 1-9</p> <p>DCTRD 1-10</p> <p>DCTRL 1-10</p> <p>DCTRN 1-10</p> <p>DCTRS 1-10</p> <p>DCTSL 1-9</p> <p>DCTST 1-8</p> <p>DCTSZ 1-10</p> <p>DCTTO 1-9,2-30</p> <p>DCTT1 1-9</p> <p>DCTT2 1-9</p> <p>DCTT3 1-9</p> <p>DEQRQ 2-27f</p> <p>DEQUE 2-28</p> <p>device codes 2-1f</p> <p>Device Control Table (DCT) 1-5f,2-3, 2-32,3-15,3-17,3-20f</p> <p>digital interface driver (4066) 3-16ff</p> <p>disk 1-3f,1-10f,2-1,3-7</p> <p>DISMIS 1-6</p> <p>dispatch table 2-20ff</p> <p>DIVD 2-9,2-13</p> <p>DIVI 2-9,2-13</p> <p>DP 2-16ff</p> <p>\$DPO/\$DPI 1-3,2-1</p> <p>.DUCLK 3-8f</p> <p>ENQUE 2-18,2-27</p>
--	---

*Note: 'f' means ' and the following page'; 'ff' means ' and the following pages.'

frame pointer 2-6
 frame sizes 1-10

 GDIRS 2-24f
 GMDIR 2-24f
 GSUB 2-5, 2-7ff

 IBUF 2-16ff, 2-29, 2-31
 .IDEF 2-1, 3-2f, 3-7, 3-10, 3-13, 3-18, 3-25

 INTD 1-4, 1-6
 INTS 1-6
 IOBUF 2-25ff, 2-30
 IPB 1-3
 .IRMV 3-4, 3-13, 3-18
 ITBL 1-5ff, 3-1
 .IXMT 3-2, 3-16f, 3-19f, 3-24

 LDBT 2-9
 LDCHR 2-9
 \$LPT/\$LPT1 2-1, 3-7

 magnetic tape 2-2
 mask 1-3, 1-6
 MBLK 2-9
 MBMVW 2-9, 2-14f
 MCA 1-3, 2-1f, 3-20ff
 MCLR 2-9, 2-12, 2-15
 MGSUB 2-5, 2-7ff
 MFMVB 2-9, 2-13f
 MFMVW 2-9, 2-14f
 MLDBT 2-9, 2-14
 MPY 2-9
 MPYAD 2-9
 MSTBT 2-9, 2-11, 2-14
 MTMVB 2-9, 2-13f
 MTMVW 2-9, 2-14f
 MVBYT 2-9, 2-11
 MVMFW 2-7
 MVWD 2-7, 2-9, 2-13f
 MXTMP 2-4f

 NMAX 1-11

 OACn 2-5ff, 2-39
 OBUF 2-16ff, 2-30
 OPNI 2-20
 OPNO 2-20
 ORTN 2-5ff
 OSP 2-6f
 OVRTN 2-6f

 PARS 2-5, 2-10
 PARU 1-11, 2-30
 PC 2-16ff
 PENQU 2-29
 \$PLT/\$PLT1 1-3f, 2-2
 power fail service 3-6ff
 PP 2-16ff
 \$PTP/\$PTP1 1-3, 2-2, 3-7
 \$PTR/\$PTR1 1-3, 2-2, 2-19, 2-39ff, 3-7

 QTY 1-3, 2-2, 3-7
 quad multiplexer (see QTY)

 RCHR 2-30ff
 RDL 2-20, 2-30f, 2-38
 RDS 2-21f, 2-30f
 .REC 3-20, 3-22f
 RETN 2-8ff
 RING1 2-7, 2-15ff
 RING2 2-7, 2-15ff
 RING3 2-15ff
 ROX 2-9f, 2-12
 RSAVE 2-6, 2-9ff
 RSTFL 2-9, 2-13
 RTC 2-2
 RTLOC 2-4f
 RTN 2-5f
 RTRN 2-6, 2-11, 2-30, 2-38f
 .RUCLK 3-9f

 SAV 2-5
 SAVE 2-6, 2-38
 SETFL 2-9, 2-13
 SFTAB 2-33
 .SMSK 3-2, 3-6
 SP 2-4
 SRTN 2-5f, 2-9f

SRFRN 2-6, 2-11
 stack frames 2-4
 STBT 2-9, 2-11
 STCHR 2-9, 2-11
 .STMAP 3-5, 3-20, 3-25
 STOUT 2-27
 SYS.DR 2-2f
 system generation (SYSGEN) 2-3, 2-37f
 system library 2-3, 2-35ff
 system queue 2-3, 2-37
 system stacks 2-3ff

 Teletype or console 1-3f, 2-2, 2-18
 TMP 2-4, 2-6f

 .UCEX 3-9f
 UFCA1 1-12
 UFD 1-11
 UFEA1 1-12
 UFFA1 1-12
 UFLAN 1-12
 UFLA1 1-12
 UFNA1 1-12
 UFTAC 1-12
 UFTAD 1-12
 UFTAT 1-12
 UFTBC 1-12
 UFTBK 1-12
 UFTCA 1-12
 UFTCB 1-12
 UFTDC 1-12
 UFTDL 1-12
 UFTDR 1-12
 UFTEA 1-12
 UFTEX 1-12
 UFTFA 1-12
 UFTFN 1-12
 UFTHM 1-12
 UFTLA 1-12
 UFTLK 1-12
 UFTNA 1-12
 UFTP1 1-12
 UFTP2 1-12
 UFTST 1-12
 UFTUC 1-12

 UFTUN 1-12
 UFTYD 1-12
 .UIEX 3-2, 3-4
 .UPEX 3-2, 3-8
 User File Table (UFT) 1-11f, 2-20ff, 2-38
 user interrupts Chapter 3

 VCT 1-4
 VRTN 2-4f

 WRL 2-23f
 WRS 2-23

 XIBUF 2-29f



How Do You Like This Manual?

Title _____ No. _____

We wrote the book for you, and naturally we had to make certain assumptions about who you are and how you would use it. Your comments will help us correct our assumptions and improve our manuals. Please take a few minutes to respond.

If you have any comments on the software itself, please contact your Data General representative. If you wish to order manuals, consult the Publications Catalog (012-330).

Who Are You?

- EDP Manager
- Senior System Analyst
- Analyst/Programmer
- Operator
- Other _____

What programming language(s) do you use? _____

How Do You Use This Manual?

(List in order: 1 = Primary use)

- _____ Introduction to the product
- _____ Reference
- _____ Tutorial Text
- _____ Operating Guide
- _____ _____

Do You Like The Manual?

Yes	Somewhat	No	
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Is the manual easy to read?
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Is it easy to understand?
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Is the topic order easy to follow?
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Is the technical information accurate?
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Can you easily find what you want?
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Do the illustrations help you?
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	Does the manual tell you everything you need to know?

Comments?

(Please note page number and paragraph where applicable.)

Large empty box for handwritten comments.

From:

Name _____ Title _____ Company _____
 Address _____ Date _____

FOLD DOWN

FIRST

FOLD DOWN

FIRST
CLASS
PERMIT
No. 26
Southboro
Mass. 01772

BUSINESS REPLY MAIL

No Postage Necessary if Mailed in the United States

Postage will be paid by:

Data General Corporation

Southboro, Massachusetts 01772

ATTENTION: Software Documentation

FOLD UP

SECOND

FOLD UP