Application Note

USER DEVICE DRIVER IMPLEMENTATION IN THE REAL TIME OPERATING SYSTEM

017–000019–00

Ordering No. 017-000019

Printed in the United States of America
Rev. 00, April 1975

This document is intended for ECLIPSE™* system users. It was derived from 017-000006-01, which is primarily for NOVA®** system users. There may still be references to NOVA equipment documents, and processes within this manual; if so, be assured that the references also apply to ECLIPSE systems.

Revision History:

017-000006

(RTOS 3.00) Original Release - March 1974
(RTOS 4.00) First Revision - March 1975

017-000019

Original Release - April 1975

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

TABLE OF CONTENTS

Chapter 3 - User Program Serviced Interrupts (Continued)

CHAPTER 1

RTOS 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 RTOS operating system or they can be implemented as part of the user application program and attached to the interrupt dispatch program.

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).

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 servicing 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).

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

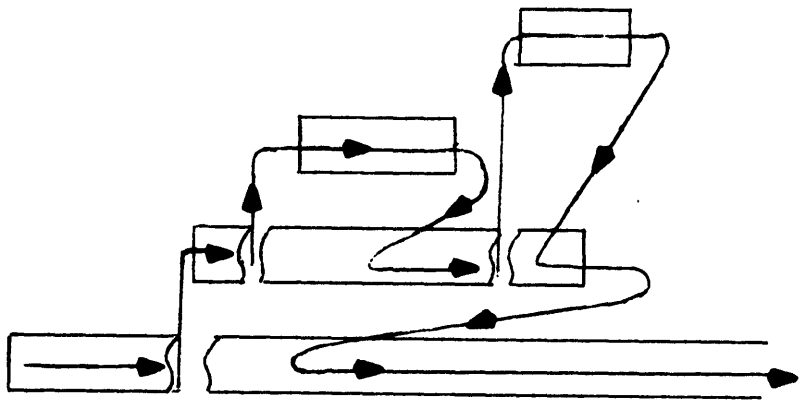
Mask	Assigned Devices
3	$TTO, $TTI, QTY (4060 multiplexor)
7	$PTP
10	$LPT
17	$PLT, MCA (4038 multiprocessor communications adapter transmitter/receiver).
617	moving head disk
717	fixed head disk
1737	$PTR
1777	$CDR, magnetic tape, cassette

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 cannot 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®*, line printer, and fixed head disk.

Program	Mask
Fixed Head Disk Interrupt	717
Plotter	17
Teletype Interrupt	3
Main Line User	0



INTERRUPT DISPATCH PROGRAM

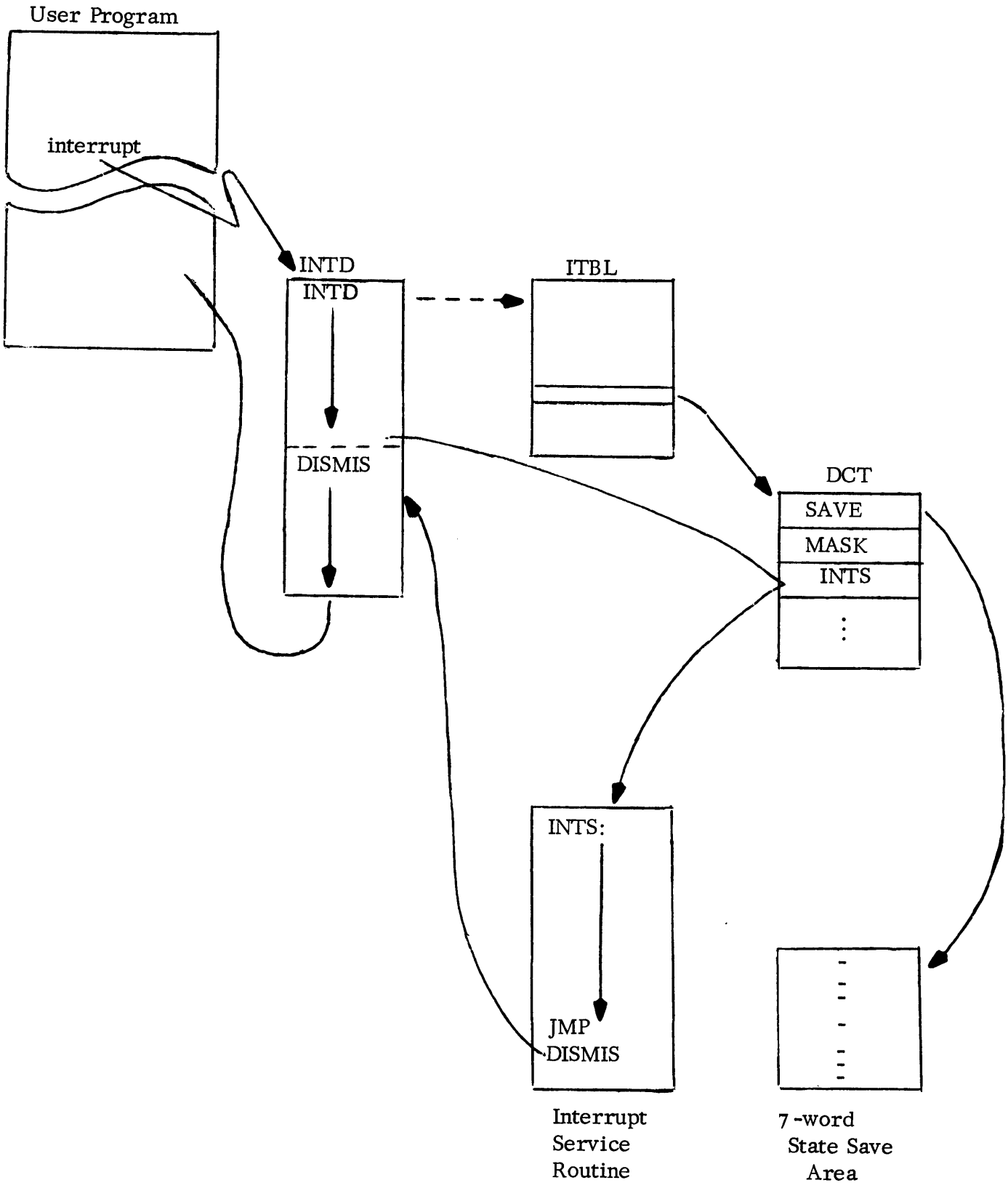
At the precise moment an unmasked interrupt has been detected at the hardware level, the Interrupt Dispatch program (INTD) receives control to service the interrupt. The INTD program is assembled as an integral portion of the RTOS system and resides in core at all times.

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

The INTD program directs control to the correct servicing routine by using the correct entry in the system Interrupt Branch Table (ITBL). The entry in this table is to a Device Control Table (DCT) associated with the servicing routine for the device.

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



Flow of Control During Interrupts

DEVICE CONTROL TABLE (DCT) STRUCTURE

Each device defined in the RTOS 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. For a thorough understanding of DCTs, a complete DCT as used in an RTOS interrupt servicing routine is described.

Word 0, DCTSV: Address of a seven-word interrupt state save area with the following structure:

Word 0, IPCC	PC and Carry
Word 1, IAC0	AC0
Word 2, IAC1	AC1
Word 3, IAC2	AC2
Word 4, IAC3	AC3
Word 5, ICMSK	Current Hardware Mask
Word 6, IRLOC	Contents of system page zero temporary

Word 1, DCTMS: Mask Word. Set a bit for every priority considered lower 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.* A complete list of system masks is provided 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 table below.

MNEMONIC	BIT	MEANING
DCCPO	1B15	Device requiring leader/trailer
DCCGN	1B14	Device requiring tab simulation
DCIDI	1B13	Buffered input device
DCCNF	1B12	Device requiring form feed simulation
DCTO	1B11	Teletype output device
DCKEY	1B10	Keyboard input device (uncontrollable)
DCNAF	1B09	Device requiring 16 nulls after form feeds
DCRAT	1B08	Device requiring rubouts after tabs
DCPCK	1B07	Device requiring even parity check on input, even parity computation on output

*All interrupts are disabled when the RTC is being serviced.

DEVICE CONTROL TABLE (DCT) STRUCTURE (Continued)

MNEMONIC	BIT	MEANING
DCLAC	1B06	Device requiring line feeds after carriage returns
DCPFR	1B05	Auto restart mode bit (internal to RTOS)
DCFWD	1B04	Full word device (any size greater than a byte)
DCFFO	1B03	Form feed sent on .OPEN
DCLTU	1B02	Convert lower to upper case ASCII
DCC80	1B01	80 column device (suppress line output after 80 columns)

Word 4, DCTCD: Device Code.

Word 5, DCTEX: Address of variable I/O instruction routine.

Word 6, DCTDT: Address of the device command dispatch table. One entry for every RTOS I/O function. The table order must correspond exactly to the order of the functions given below. Each entry is an address to the routine implementing the named RTOS function. If the device does not permit a command, a -1 should be entered in place of the address.

MNEMONIC	DISPLACEMENT	MEANING
OF	0	Open a file
CF	1	Close a file
RS	2	Read Sequential
RL	3	Read Line
WS	4	Write Sequential
WL	5	Write Line
RB	6	Read Block
WB	7	Write Block
OA	10	Open a file for appending

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:	STOUT or address within the device handler; performs the equivalent of an interrupt service.

DEVICE CONTROL TABLE (DCT) STRUCTURE (Continued)

Word 10, DCTIN: Device initialization routine.

Word 11, DCTLK: Link to start of TCB queue competing for a device.

Word 12, DCTTO: Timeout constant (none if zero).

Word 13, DCTQL: Link in device request bead chain.

(Word 13, DCTD2: First displacement of DCT for mag tape or cassette unit; see discription at end of this section.)

Word 14, DCTDP: Device byte data pointer.

Word 15, DCTDC: Device byte count.

Word 16, DCTQS: Bead status word (described in GENERALIZED I/O ROUTINES).

Word 17, DCTBD: Bead address (i.e., the starting address of the bead, words 13-16).

Word 20, DCTQP: Request bead queue starting address (initially -1).

Word 21, DCTOC: Characteristics of opened device.

Word 22, DCTT1: First temporary for device control.

Word 23, DCTT2: Second temporary for device control.

Word 24, DCTCT: Current timeout count (input device).

(Word 24, DCTCC: Output device column counter.)

Word 25, DCTPR: Echo device DCT pair pointer, TTI only.

(Word 25, DCTLC: Output device line counter.)

Word 26, DCTSC: Saved device request byte counter. When special characters not in the buffer must be generated (e.g., LF after CR), this word is used to save the current device byte counter so that this character can be generated and transmitted.

Word 27, DCTGN: Character for generation (see word 26).

DEVICE CONTROL TABLE (DCT) STRUCTURE (Continued)

The following displacements, 13-28, are used with magnetic tape and cassette units using buffered I/O.

Word 13, UNUM: Device unit number.

Word 14, ADUCT: Base address of tape unit control table (UCT).

Word 15, DADR: Base address of data area.

Word 16, DCNT: Word or record count.

Word 17, IOFRP: Pointer to I/O function routine for .MTDIO

Word 18, CMDWD: Command word, ready for DOA.

Word 19, TEMP: Temporary word used by RTOS.

Word 20, OPRET: Temporary word used by RTOS.

Word 21, .UCTX: Base address of currently initialized unit's UCT.

Word 22, .ST: Address of "STATU" routine used to get magnetic tape unit status.

Word 23, .ISSU: Address of "ISSUE" in module MTSER.

Word 24, PFR: Bit 0 - flag set to 1 when unit is positioning for retry
Bits 1-15 - Counter used in attempting to erase for write retry.

Word 25, RTCTR: Write counter.

Word 26, MODE: I/O mode word; Line = 1, Sequential = 0.

*****

CHAPTER 2

OPERATING SYSTEM DEVICE DRIVER IMPLEMENTATION

GENERAL

There are two major approaches to implementing a device handler in RTOS. The first, and probably the most commonly selected choice, is to minimize the interface between RTOS and the new driver by providing only a three-word DCT header and issuing RTOS system call .IDEF. This defines the device to be a special user device, and information in the three- word DCT permits the user service routine to obtain control from the interrupt dispatcher at the right times.

The second choice is to integrate the new device handler into the main body of the operating system, thus taking advantage of the system's general purpose I/O facilities. This second choice requires a system level knowledge of RTOS. A driver implemented in this way will be able to use standard system calls, e.g., .OPEN, .CLOSE, .RDS, etc.

At key points in RTOS system call processing, the flow of control may be intercepted in order to make provision for new devices. In particular, the name of a file submitted as input to the .OPEN call must be interpreted by a routine which can derive a physical device location from the name of the device. Also a software channel to that device must be established for references in read/write calls. The system must be rescheduled at the initiation of an I/O transfer and again at its completion. The section which follows discusses the interactions between device handler and system at these times.

GENERALIZED I/O ROUTINES

The I/O module (GENIO) provides a number of useful, general purpose 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). GENIO also provides 3 routines used by RTOS to implement system calls .GCHAR, .PCHAR, and .RESET. These calls are discussed in the RTOS Reference Manual, 093-000056.

No system buffering of data is performed* under RTOS. Instead, data is transferred between a device and a user area, with pointers and counters maintained to indicate the amount of data in the area and the current word input or output. A four-word bead is assigned to maintain status information and pointers for each user area.

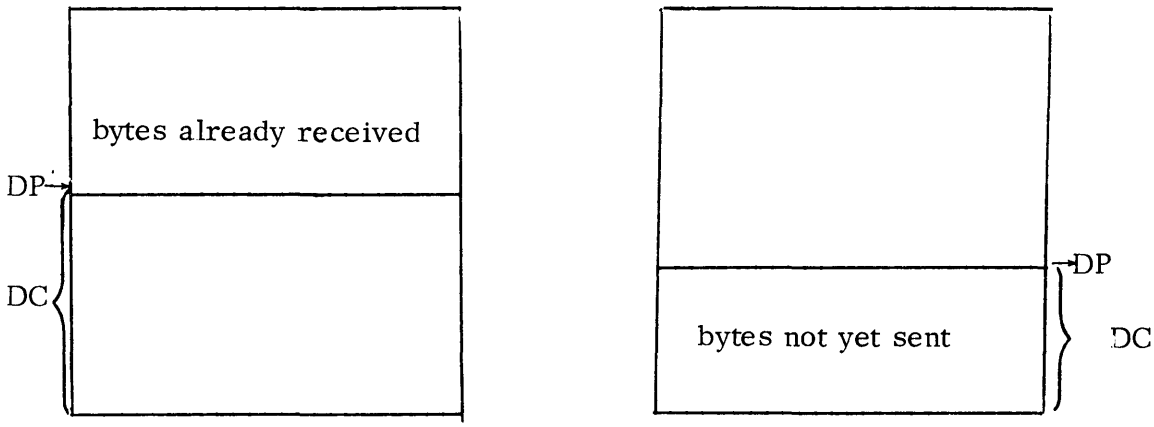
*Two exceptions to this rule are the $PTR and the $CDR. A 10-byte buffer is maintained for the $PTR to prevent chatter, and an 80-word buffer is maintained for the $CDR to permit the transfer of all 80 columns from a card.

GENERALIZED I/O ROUTINES (Continued)

While processing efficiency is gained by this direct transfer of data between program and device, the timing of device activity is more rigidly synchronized with the issuance of system calls.

The following illustration is similar to its counterpart in RDOS. The primary difference between this illustration and that given for bead structure in RDOS is that the data area governed by an RDOS bead is actually a system buffer which resides in the device handler. By contrast, RTOS deals directly with the user area.

An input device under RTOS inputs to the user area at interrupt time and an output device outputs from the user area at interrupt time.



Input Devices

Output Devices

Pointer DP indicates the current slot in the user area which is targeted for storage or retrieval of data by the device. Counter DC indicates the number of slots in the area which remain to be processed.

When DC becomes equal to zero, or when the completion of the request represented by a bead is otherwise indicated, the interrupt service routine checks to see if a user task needs to be readied. This is the case if a one is found in bit zero of the bead status word. Otherwise, bit 15 of the bead status word is set to one, indicating completion of I/O. In either case, the bead is removed from the queue.

A virtually unlimited number of user areas can be appended to the first area, providing the facility for unformatted stream I/O and substantially reducing system overhead. Nonetheless, current versions of RTOS employ no more than one bead outside the main DCT bead. The DCT bead itself is used for user task I/O. Additional I/O buffers are appended to the first buffer by linking or stringing

additional beads to the first bead. Each bead has a link to the next bead in the string; the terminating bead has a -1 link. IOSER provides routines to add beads to a string. Beads are removed from the string when data transfers are complete.

One bead consists of words 13 through 16 of the device's DCT. The structure of each bead is as follows:

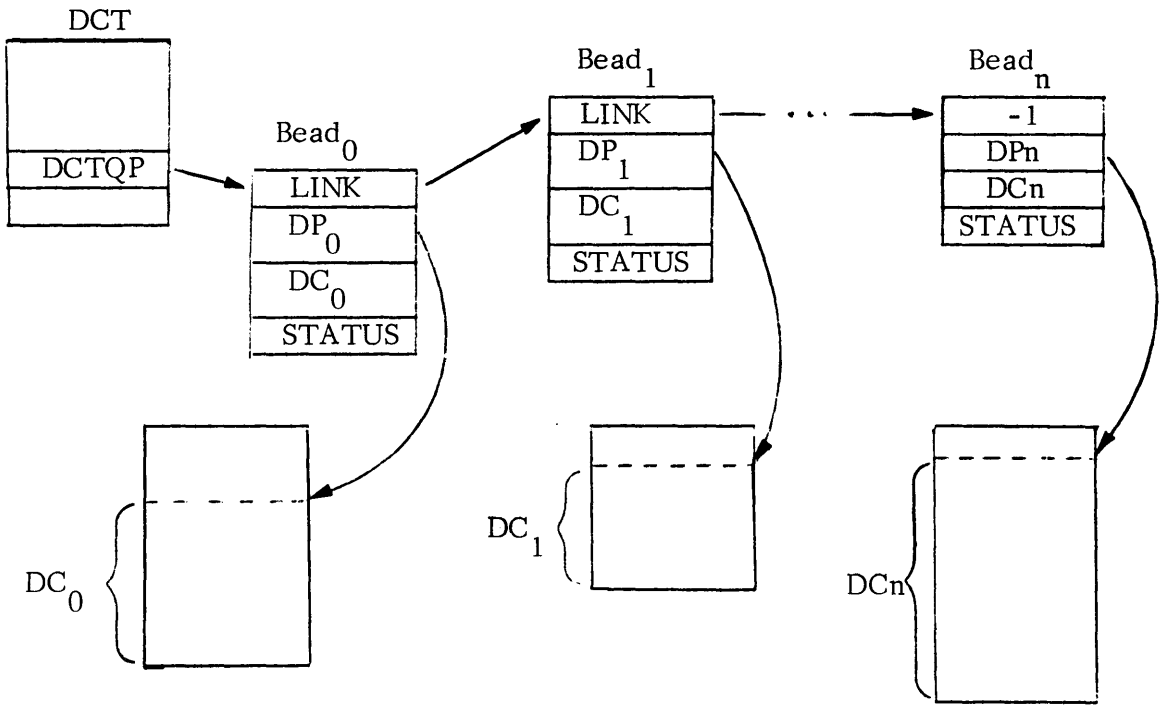
Displacement	Mnemonic	Contents
0	RQLK	Link Word
1	RQPTR	Byte Pointer (DP)
2	RQCNT	Byte Count (DC)
3	RQST	Bead Status Word

Successive beads used by the system for purposes like echoing are linked to by previous bead links. The first bead in the string is pointed to by DCTQP of the device's DCT.

Bit 0 of the bead status word is set to 1 if this bead is imbedded in the device's DCT (words 13 - 16). Bit 0 is reset if the bead is outside the device's DCT. The DCT bead is employed for user task I/O, and upon the successful emptying (or filling) of the data buffer, the task is readied and returned to the normal return of its system call. The non-DCT bead is used by RTOS for echoing data to the console. In the event that data is being written to $TTO and keyboard characters are also received, the keyboard characters are placed in a buffer whose bead is placed ahead of the $TTO DCT bead.

Bit 1 of the bead status word is reset to 0 if the I/O operation requires byte calculation to be transmitted in AC1 (as in .WRL, .RDS etc.). Bit 1 is set to 1 if no byte calculation is to be performed and the task's original AC1 is to be preserved. This occurs with such system calls as .OPEN (form feed or leader), .CLOSE (form feed or trailer), .PCHAR etc.

The meaning of bit 15 depends upon the setting of bit 1. If bit 1 is set to 1, then bit 15 set to 1 indicates a line mode I/O operation, and bit 15 set to 0 indicates sequential I/O. If bit 1 is cleared, bit 15 set to 0 indicates the I/O request is not done, while bit 15 set to 1 indicates the I/O is complete. (The ENQUE routine, discussed later, clears bit 15 for all non-DCT beads.)



A brief description of the major routines in GENIO follows. More detailed information can be obtained by reviewing the listing of GENIO. Before discussing specific I/O routines, the following observations must be made about all I/O routines whose entries are used in DCT device dispatch tables.

System I/O routines are not called in the conventional sense. Rather, the names of those routines which are required by a device driver are inserted into the appropriate displacements of that driver's dispatch table. This table is a nine-word table with the following definitions:

Displacement	Mnemonic	Command Function
0	OF	open a device
1	CF	close a device
2	RS	read sequential
3	RL	read line
4	WS	write sequential
5	WL	write line
6	RB	read block
7	WB	write block
10	OA	open a device for appending

Control is passed by the system to the routine pointed to in the dispatch table at a time when, after having waited in a queue for a device, the calling task has exclusive access to the device. Execution takes place at base level in the system. That is, all interrupt levels are enabled and their service has priority over these routines. As far as routines in the GENIO module are concerned, the device is idle and is awaiting the initiation of I/O. In actuality, the device may still be in operation, if it is one of those devices whose interrupt service routine provides its own buffering.

The actual function of the ..RDS, ..RDL, ..WRS and ..WRL routines is to initiate an I/O transfer in the appropriate mode by constructing a bead and then by placing it in a device queue. If a device's characteristics word specifies the need for special output (e.g., leader, trailer, etc.), the .CLSI and .CLSO routines initiate this I/O via beads. Read and write block entries are found in the disk DCTs for interrupt handling purposes and then issue the initial series of commands to the disk controller. If no I/O is initiated, the routines exit to the IOEND processor at base level; otherwise, control goes to the scheduler.

Command functions which are to be unavailable to a device are indicated by the placing of a -1 in the appropriate table offset location. An attempt to perform this forbidden command function would result in error code ERICD (illegal command for device) being issued. It is conceivable that a special I/O function might need to be written (e.g., a special read line, RDL1) and that the entry for this routine might be inserted into the table (e.g., at displacement 3). Special considerations for the writing of such customized routines are given at the end of this section. Along with a description of each of the standard GENIO routines there is given a list of input which is provided by the system when each of these routines receives control. In some cases, not all of these inputs are utilized by the routines; the complete list does indicate, however, what inputs are available should a customized routine in this command table displacement be written.

Open a Device (.OPNO, .OPNI)

.OPNO is used to open output devices, while .OPNI is used to open input devices. .OPNO provides a form feed and outputs leader if required. These routines presume that the DCT has provided space for all necessary I/O buffer information as well as the link to queued tasks awaiting I/O, the timeout constant, and temporaries for device control (words 11-20 and 22, 23 of DCT).

System inputs:
- AC1 - characteristic inhibit mask
- AC2 - DCT address
- AC3 - TCB address

Open a Device (.OPNO, .OPNI) (Continued)

Calling sequence: Insert .OPNO (.OPNI) mnemonic into the device dispatch table, displacement OF.

Close a Device (.CLSO, .CLSI)

.CLSO closes an output device, and punches trailer if required. .CLSI clears the device and initializes the handler, and then resets the bead queue (by setting the bead queue pointer to -1). A special check is made for keyboard devices, which are not cleared, etc., since they might be in the process of echoing information. Both .CLSO and .CLSI branch to the I/O end processor (IOEND) at base level; this module will be described at the end of this section.

System inputs:
AC2 - DCT address
AC3 - TCB address
(displacement TAC3 of TCB contains a pointer to the UFPT frame for the channel being closed)

Calling sequence: Insert .CLSO (.CLSI) mnemonic into the device dispatch table, displacement CF.

Read Sequential (..RDS)

This command function causes the device to 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.

System inputs:
AC0 - destination byte pointer (byte address of area to receive the data)
AC1 - byte count
AC2 - DCT address
AC3 - TCB address

Calling sequence: Insert ..RDS mnemonic into the dispatch table, displacement RS.

Read a Line (..RDL)

This routine is used to read ASCII data from a device and transmit this data to a user area. The transmission is terminated after detection of a carriage return, form feed, or null. All bytes are masked to seven bits. Line feeds are unconditionally ignored. Checks of the device characteristics are made to determine whether to perform parity checks, whether the device is an 80-column device, etc. This interpretation of ASCII data is done by the interrupt service routine.

Read a Line (..RDL) (Continued)

System inputs: AC0 - byte pointer to data area
AC2 - DCT address
AC3 - TCB address

Calling sequence: Insert mnemonic ..RDL into the device dispatch table, displacement RL.

Write Sequential (..WRS)

This routine outputs binary data in byte form from a user area to a device. The data is not altered in any manner. This mode is therefore the standard mode for binary output transfers.

System inputs: AC0 - byte pointer to data area
AC1 - byte count
AC2 - DCT address
AC3 - TCB address

Calling sequence: Insert mnemonic ..WRS into the device dispatch table, displacement WS.

Write a Line (..WRL)

This routine transmits ASCII data to an output device and terminates transmission after sending either a carriage return or form feed. Termination also occurs upon detection of a null, but the null is not written. Interrupt service routines make checks of the device characteristics to determine whether to perform parity computation, tab simulation, etc.

System inputs: AC0 - byte pointer
AC2 - DCT address
AC3 - TCB address

Calling sequence: Insert mnemonic ..WRL into the device dispatch table, displacement WL.

Read and Write Disk Blocks

Displacements RB and WB in each device dispatch table are reserved for entries to disk read/write block routines. These routines are found within the disk drivers proper, not in GENIO. Input parameters provided by the system upon entry to these routines are as follows:

Read and Write Disk Blocks (Continued)

AC0 - starting core address for the transfer
AC1 - starting relative disk block number
AC2 - DCT address
AC3 - TCB address

Open a Device for Appending

Since opening a device for appending is equivalent to simply opening the device, displacement OA in the device dispatch table is set with routine .OPNO for those devices which permit appending.

Writing a Custom I/O Routine

If the I/O routines described in this section are inadequate, special routines may be written and their entries can be inserted appropriately into a device dispatch table. Before this project is undertaken, detailed study of RTOS listings, especially SYSTEM, IOSER, and GENIO should be made. The following discussion is presented as an introduction to such a study.

The above discussions of GENIO routines indicate what inputs are provided by the system for each type of call. It remains to be seen how control exits from these routines, so that similar exits could be made from custom I/O routines.

Exit from generalized routines occurs in one of two ways, depending upon whether or not the routines expect further interrupts to be generated. If further interrupts are anticipated, the following sequence occurs. First, the bead status word is set up, as are the byte pointer (DCTDP) and the byte count (DCTDC); line mode operations set DCTDC to 077777. The bead address, DCTBD, is then loaded into AC1; the scheduler address SCHED is loaded into AC3 and interrupts are disabled. Control is then dispatched to ENQUE to enqueue a bead to the DCT, after which control is dispatched to the task scheduler.

If further interrupts are not anticipated as the immediate consequence of the call, AC2 is set to the address of the DCT, and an unconditional transfer to .IOEND is made.

If no task is awaiting the use of this device, the current task is readied and return is made to the scheduler (SCHED) if .IOEND was entered at base level. Otherwise, control goes to the interrupt dismissal routine (DISMISS). Rescheduling then occurs only if the system switch, .SYS., indicates that RTOS is not in the system mode.

If another task is awaiting the use of this device, provision is made to start the device again and to ready the current task; control then returns either to the scheduler or to the dismissal routine as described above.

DISMISS, location 11, contains the address of an entry, DISN, in the interrupt dispatch module INTD. This routine is entered with the device DCT address in AC2. The original hardware mask is restored. If dismissing to another interrupt servicing routine, interrupts are re-enabled and return is made to the interrupted routine. If dismissing to a user task, a rescheduling check is made. If no rescheduling is to occur, interrupts are re-enabled and return to the task is made. Otherwise, the state of the interrupted task is saved in its TCB and the scheduler is entered.

Opening a User File

The system call .OPEN uses the file name supplied as an argument to identify the device to which subsequent I/O operations will be directed. If channel-oriented I/O calls are to be usable with a new device, there must be a way for the .OPEN routine to trace each file name to the DCT for that device, e.g., $PTR for the paper tape reader. A more complex file name construction is necessary if the device consists of multiple communication lines or circuits, channels or units, or if the device is a storage medium with compartmentalized data.

During the processing of a .OPEN call, a series of attempts is made to match the file name with entries in tables built at RTOSGEN time. Without modifying either the RTOSGEN program or the resident RTOS .OPEN logic, the user may add a routine to recognize file names for new devices. After the RTOS routine has tried and failed to recognize a file name, it will pass control to a user routine with the entry label .CKUS, provided that such an entry was defined at relocatable load time.

The .CKUS routine may then check for other forms of device specification and, if successful, locate the corresponding control blocks for opening the channel. Having opened the channel, the user returns control to entry .FOPN which finishes the .OPEN process. This includes calling the routine found in the dispatch table pointed to by displacement DCTDT of the DCT.

When control arrives at .CKUS, AC3 contains a return address which is to be used only if the file name is not recognized. The system ensures that page zero location CTCB contains the address of the caller's TCB. Displacement TAC3 of the caller's TCB contains the address of a two-word block which must be set up to open the software channel. The .CKUS routine must place the appropriate DCT address in the first word of this pair. The second word of the pair is optionally available for a

Opening a User File (Continued)

pointer to a control block for a unit, circuit, etc., within the device. This word pair is called a user file pointer table (UFPT) entry. On completion, if the file name was recognized, the .CKUS routine must branch to entry .FOPN in RTOS with the DCT address in AC3. If the file name was not recognized, control must be returned to the address input in AC3 upon entry to .CKUS. This will cause rejection of the file name and an error return for the caller (error code ERDLE: non-existent file).

I/O SERVICE MODULE (IOSER)

The IOSER module is used by system drivers generally for byte-oriented devices. One use of IOSER 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 IOSER is to provide common input/output character device interrupt processing. IOSER also provides routines to place a bead at the end of a bead string, and to priority enqueue a bead at the head of the queue.

The following list summarizes the entries in IOSER:

Entry	Use
.CLOSE	Common output interrupt service.
.STOU	Start an output device.
.ENQB	Add a bead to the end of a string.
.PENQ	Priority enqueue a bead at the head of a string.
.FINP	Common input interrupt service.
..IAC	DIAC instruction processor.
.DIAS	DIAS instruction processor.
.DOAS	DOAS instruction processor.
.NIOC	NIOC instruction processor.
.NIOS	NIOS instruction processor.
.SKPB	SKPBZ instruction processor.

Calls to any of the instruction processor entries 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.

Common Output Device Interrupt Service (.COSE, .STOU)

.COSE clears a character device and then provides interrupt service for the device. .STOU provides the equivalent of interrupt service for the purpose of starting a device. 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
AC3 - return address

Calling sequence:

```
        .EXTN .COSE   (.STOU)
        JSR @ COSE
        return to DISMISS

COSE:   .COSE   (.STOU)
```

Function: Line mode editing is performed in accordance with the device's characteristic word, DCTCH, of its device control table. Specifically, the following functions can be performed: lowercase ASCII is converted to uppercase (DCLTU), tabs are simulated (DCCGN), a rubout is inserted after each tab (DCNAF), the line output is truncated by suppressing output after 80 columns (DCC80), and parity computation is performed (DCPCK). AC2 is preserved upon exit.

Add a Bead to the String (.ENQB)

.ENQB attaches a bead to a device. This bead becomes the last bead in the string, and is given a LINK of -1. .ENQB is called by means of the following calling sequence:

Input:
AC1 - Bead address
AC2 - DCT address

Calling sequence:

```
        .EXTN .ENQB
        JSR @ ENQ
        return

ENQ:    .ENQB
```

Function: The bead is attached to the device. The device is started if it was formerly idle. AC2 is restored upon exit.

Insert a Bead at the Start of a String (.PENQ)

.PENQ attaches a bead to the front of the bead queue. This routine is used principally for echoing characters to an output device. .PENQ is called by means of the following calling sequence:

Input:
AC1 - Bead address
AC2 - DCT address

Calling sequence:

```
.EXTN .PENQ
JSR @ PENQ
return

PENQ: .PENQ
```

Function: The DCT bead (with its associated buffer) is placed before all other beads in the string. AC2 remains unchanged upon exit.

Common Input Device Interrupt Service (.FINP)

.FINP provides interrupt service to input character devices. .FINP is called by means of the following calling sequence:

Input:
AC0 - input character
AC2 - DCT address
AC3 - return address (DISMISS or IOEND)

Calling sequence:

```
.EXTN .FINP
JSR @ FINP
return to DISMISS or IOEND

FINP: .FINP
```

Output: For line mode transfers, a parity check is performed and the input character is examined to see if it is an end-of-file (CTRL Z); if so, the line is terminated without storing the CTRL Z. Exit character checks are then made, and if a rubout or \ is detected, either the last input character is deleted or the entire line is deleted. Line feeds are ignored. A check is made for carriage return, form feed, or null terminators. Non-keyboard devices cause rubout characters simply to be ignored. For sequential mode, the byte is stored in the device buffer, and the number of bytes transmitted is calculated and returned to the task AC1.

Declaring the Device Name and DCT Address

The names of user devices and the addresses of their DCTs are declared when the RTOS module, output by RTOSGEN, is created. Declaration of the DCT names of user devices is accomplished by responding in the affirmative to question 19 of RTOSGEN: "USER SUPPLIED DRIVERS?". By striking "1" on the console keyboard, you trigger the further pair of questions:

```
DEVICE CODE?
NAME?
```

Respond with the octal device code for the first user device, and follow this with a carriage return. Then input a 3 alphanumeric character name for the device, and follow this with a carriage return. The pair of questions is repeated for more devices. Respond with a simple carriage return to the device code query to terminate this interrogation. As described in Appendix B of the RTOS Reference Manual, 093-000056, RTOS then outputs a complete list of system and user device codes, DCT names (which are simply the device names with a DC suffix), and device names. RTOS will place table entries for all user devices into .ITBL; each table entry will contain the address of that device's DCT.

Corresponding to the RTOSGEN declaration of user device names, each user device driver must have an entry with the same DCT name as that specified to RTOSGEN. That is, if the device name was specified to be ABC, the driver for this device must contain an entry to the device's DCT: .ENT ABCDC.

High Priority Interrupt Devices

High priority interrupt devices are devices which will receive interrupt service control before standard devices. System high priority devices are the real time clock and the power fail/auto restart option. Question 18 of RTOSGEN permits special user devices to also receive high priority interrupt service.

In essence, the operation of the high priority interrupt dispatcher, .HINT, is as follows. Each high priority interrupt device is examined to see if it generated the interrupt. The power fail monitor is tested first, then the real time clock, and then each of the other user devices specified at RTOSGEN time in the order that they were specified during system generation. If the source of the interrupt is found, control is dispatched to its interrupt service routine; otherwise, control is given to the ordinary interrupt dispatcher.

High Priority Interrupt Devices (Continued)

The name given in response to question 18 for each user high priority device with the suffix IS added becomes the name of the entry point to the service routine for each such device. Thus if the name given for one high priority user device was GHK, the entry point to this device's interrupt service routine would be GHKIS, and this address would have been entered in the service routine: .ENT GHKIS. For information describing the writing of high priority user interrupt service routines, see Chapter 3, HIGH PRIORITY INTERRUPT SERVICE.

Loading a User Driver

After specifying all user drivers at RTOSGEN time and assembling these drivers, they may be loaded with user relocatable binaries and the RTOS libraries. Depending upon the loader used, one of several relocatable load procedures can be followed. These procedures are described in Appendix B of the RTOS Reference Manual. In essence, RTOS.RB, user driver relocatable binaries, and the user program binaries must be loaded before the RTOS libraries. Consult the RTOS Reference Manual for more details.

RTOS System Library List

The following list names and describes each of the library modules contained in the RTOS libraries for revision 4.00 of RTOS, run on both the NOVA and ECLIPSE computers.

Relocatable Binary Title	Function(s)
BFPKG	Core Buffer I/O package (see 017-000003).
TXMT	.XMT, .REC, .XMTW, .IXMT logic.
TACALL	.AKILL, .SUSP, .ASUSP, .ARDY logic.
TABOR	.ABORT logic.
TSKID	.IDST, .TIDS, .TIDR, .TIDK, .TIDP logic.
TMIN	System call set up logic for single task programs. Single task environment task scheduler.
TCBMON	System call set up logic for multitask programs. Multitask environment task scheduler.
TUMOD	.SMSK, .UCEX, .UPEX, .UIEX logic.
TQTAS	.QTSK, .DQTSK logic.
TIDSR	Additional support for .QTSK/.DQTSK.

RTOS System Library List (Continued)

Relocatable Binary Title	Function(s)
SYSTEM	System call processor and error handler. End of I/O processor.
INTD	Interrupt processor. Interrupt dispatching and dismissal.
RTIN	Initializes system device DCTs. Resets system state variables. Initializes TCB pool and chains. Branches to user starting address.
IDEB	RTOS debugger (see 093-000044).
TTY1D	Second Teletype driver.
TTY2D	Third Teletype driver.
TTYDR	Teletype driver.
PTR1D	Second paper tape reader driver.
PTRDR	Paper tape reader driver.
PTP1D	Second paper tape punch driver.
PTPDR	Paper tape punch driver.
RTCDR	Time of day clock calls. Get RTC frequency. Set up user clock. Remove user clock. Clock delay system call.
CDR1D	Second card reader driver.
CDRDR	Card reader driver.
LPT1D	Second line printer driver.
LPTDR	Line printer driver.
PLT1D	Second plotter driver.

RTOS System Library List (Continued)

Relocatable Binary Title	Function(s)
PLTDR	Plotter driver.
QTYDR	4060 data communications multiplexor.
MCADR	4038 multiprocessor communications adapter.
PWRDR	Power monitor/automatic restart.
GENIO	Generalized routines corresponding to system I/O calls (except .RDB/.WRB).
IOSER	I/O service routines, mostly interrupt level.
CNVRT	Hollerith card code to 7-bit ASCII converter.
DSKDR	Fixed head disk drive for controller 0.
DKPDR	Moving head disk drive for controller 0.
MTU0 through MTU7	Unit control tables (UCTs) and buffers for line and sequential I/O on magnetic tape units 0 through 7.
MTBUF	Read/write line/sequential I/O instruction support for magnetic tape and cassette units.
MTADR	Seven or nine-track magnetic tape device driver for controller 0.
MTSER	Magnetic tape/cassette interrupt service module.
SMTU	Truncated magnetic tape unit control tables for units MT0 through MT7 employing non-buffered tape I/O.
CTU0 through CTU7	Unit control tables (UCTs) and buffers for line and sequential I/O on cassette units 0 through 7.
CASDR	Cassette device driver for controller 0.

PRACTICAL HINTS FOR SYSTEM DEVICE DRIVER IMPLEMENTATION

Elements in I/O Routines

The following illustration lists the RDL routine, found in GENIO, as an example of the required elements in an I/O routine.

```
        .EXTN  .ENQB
        .ENT  ..RDL
        ;READ LINE SUBROUTINE
        ;INPUT:  AC0:  BYTE POINTER
        ;        AC2:  DCT ADDRESS
        ;        AC3:  TCB ADDRESS


RDL:    LDA  1  LLIM          ; PICK UP MAX BYTE COUNT
        MOVO  1  1            ; SET CARRY FOR ASCII LINE MODE
        JMP  RDS0             ; GO TO COMMON RDL/RDS CODE
LLIM:   132

RDS0:   SUBCL  3  3           ; LOAD CARRY
        ADDOR  3  3           ; SET BIT 0 FOR MAIN BEAD
                              ; SET UP BEAD FOR REQUEST
SBEAD:  STA  0  DCTDP         ; SET REQUEST BYTE POINTER
        STA  1  DCTDC 2       ; SET REQUEST BYTE COUNT
        STA  3  DCTQS 2       ; SET BEAD STATUS/MODE WORD
        LDA  1  DCTBD 2       ; LOAD BEAD ADDRESS FOR
        LDA  3  SCHED         ; RETURN TO SCHEDULER
        INTDS
        JMP  @ .ENQU          ; ENQUE THIS BEAD, STARTING
.ENQU:  .ENQB                 ; DEVICE IF NECESSARY.  ENQUE
                              ; WILL TRANSFER TO SCHEDULER
                              ; WHICH WILL REENABLE INTERRUPT.
```

Examination of a System Device Driver

This section is devoted to a line-by-line examination of an actual RTOS driver, the high speed paper tape punch driver (PTPDR). Also, to highlight their similarities, the second paper tape punch driver is also illustrated.

Following are the first 10 lines of the RTOS paper tape punch driver:

```
 0001 PTPDR MACRO REV 02                    15:02:53 01/16/74
01
02              ;REAL TIME OPERATING SYSTEM (RTOS)
03
                        .TITL   PTPDR   ; RTOS PAPER TAPE PUNCH DRIVER
05
06                      .ENT PTPDC .PPDT .PPSV .PPEX .PPIN
07
08                      .EXTN .OPNO .CLSO ..WRS ..WRL   ;COMMAND ENABLE
09                      .EXTN .STOU .NTOC .COSE
10
```

Entry PTPDC on line 6 defines the paper tape punch Device Control Table (DCT) starting address. .PPDT, .PPSV, .PPEX, and .PPIN define the punch dispatch table address, state save area, execute-I/O instruction routine, and punch initialization routine respectively. .PPDT, .PPSV, .PPEX and .PPIN are all ENTered so that the second paper tape punch driver, PTP1D, may reference these entries. .OPNO, .CLSO, ..WRS, and ..WRL are externally referenced since they define the system I/O command types which can be issued to the punch. .STOU and .COSE, entries in IOSER, will be required to start the punch and provide output device interrupt service.

Examination of a System Device Driver (Continued)

```
11                          .NREL
12
13                  ; PAPER TAPE PUNCH DEVICE CONTROL TABLE
14
15          100000'PTPDC=.
16  000000'000034'          .PPSV                       ; SAVE MACHINE STATE
17  000001'000007           MKPTP                       ;MASK
18  000002'177777           .COSE                       ;COMMON OUTPUT INTERRUPT SERV
19  000003'001701           DCRAT+DCCPO+DCPCK+DCLAC+DCNAF   ; CHARACTERISTICS
20  000004'000013           PTP                         ;DEVICE CODE
21  000005'000030'          .PPEX                       ;EXECUTE
22  000006'000043'          .PPDT                       ; DISPATCH TABLE ADDRESS
23  000007'177777           .STOU                       ; PUNCH START ROUTINE
24  000010'000053'          .PPIN                       ;DEVICE INIT
25  000011'000000           0                       ;TCB LINK
26  000012'000000           0                       ;TIMEOUT CONSTANT
27  000013'000001           .BLK 1                  ;READ: LINK
28  000014'000001           .BLK 1                  ;DEVICE DATA BYTE POINTER
29  000015'000001           .BLK 1                  ;DEVICE DATA COUNT
30  000016'100000           100                     ;STATUS WORD - PRIMARY READ
31  000017'000013'          .-4                     ;READ ADDRESS
32  000020'177777           -1                      ;DEVICE REQUEST CHAIN
33  000021'000001           .BLK 1                  ;OPENED DEVICE CHARACTERISTICS
34  000022'000002           .BLK 2                  ;DEVICE TEMPS
35  000024'000004           .BLK 4                  ;MISC.
```

Relocatable locations 0 through 24 (lines 16-35) comprise the paper tape punch DCT. The indirect bit is set in the starting address of the DCT (line 15) to indicate that the punch is a system device having a more or less standard DCT. .PPSV is the start of the state save area (reserved later on). Word 1 (line 17) defines the paper tape interrupt mask. This mask, 7, prevents all devices with mask bit assignments 13 through 15 from interrupting the punch. Word 2 specifies that punch interrupt service is performed by .COSE, the common output interrupt routine. Word 3, the device characteristics word, specifies that the punch has the following characteristics: rubouts must be sent after tabs, the device requires leader and trailer, a parity computation is required, line feeds must be issued after carriage returns, and nulls must be sent after form feeds. PTP, word 4 (line 20), defines the punch device code to be 13.

Word 5 (line 21) defines the start of the punch I/O execution area. Thus when such instructions as "NIOS device"(see IOSER) are built for the punch, they are deposited in relative location 30 and control is then transferred to them. .PPDT, line 22, defines the start of the punch dispatch table. .STOU, line 23, is the entry in IOSER used to start the punch. .PPIN, line 24, is the address of the punch initialization routine (we will see that no initialization is really performed for the punch).

Word 11 is reserved for the link to the queue of tasks awaiting use of the punch. The zero in word 12 prevents monitoring of the device for timeouts.

Words 13 through 16 reserve the first bead of the punch driver; bit zero of the status word is set to indicate that this is the primary bead. Word 17, line 31, is a pointer to the primary bead. Word 20 (line 32) points to the current bead in the queue. Word 21 contains the current device characteristics word (which will differ from word 3 if some device characteristics are suppressed upon being opened). Words 22 and 23 are used by the interrupt service routine. The last block of miscellaneous temporaries is used for storing such variables as the output line count.

```
36
37 000030'000000 .PPEX:  0
38 000031'001400         JMP 0,3
39 000032'001401         JMP 1,3
40
41 000033'001400 .PPIN:  JMP 0 3
```

Location .PPEX receives I/O instructions built for the punch during the course of its operation. After receiving an instruction, .PPEX is called, the instruction is executed, and then control returns to the caller via word 31 or 32 (depending upon whether or not the instruction performed a skip).

.PPIN, the initialization routine, merely returns control to the caller. (There is no initialization necessary for the punch.)

Examination of a System Device Driver (Continued)

```
10002 PTPDR
01
02 00034'000007 .PPSV:  .BLK 7              ;INTERRUPT SERVICE SAVE STATE
03
04              ; DEFINE THE PAPER TAPE PUNCH DISPATCH TABLE
05
06 00043'177777 .PPDT:  .OPNO                       ; PTP OPEN
07 00044'177777         .CLSO                       ; PTP CLOSE
08 00045'177777         -1
09 00046'177777         -1
10 00047'177777         ..WRS                       ; PTP WRITE SEQUENTIAL
11 00050'177777         ..WRL                       ; PTP WRITE LINE
12 00051'177777         -1
13 00052'177777         -1
14 00053'000043'        .OPNO                       ; PTP OPEN FOR APPENDING
15
16                      .END
```

.PPSV defines the interrupt service state save area. .PPDT, the punch dispatch table, contains entries enabling the punch to be opened, closed, appended to and written to in both line and sequential modes.

The cross index for PTPDR is given below:

```
   0003 PTPDR

PTPDC 100000' EN     1/06     1/15
.CLSO 000044' XN     1/08     2/07
.COSE 000002' XN     1/09     1/18
.NIOC 177777  XN     1/09
.OPNO 000053' XN     1/08     2/06     2/14
.PPDT 000043' EN     1/06     1/22     2/06
.PPEX 000030' EN     1/06     1/21     1/37
.PPIN 000033' EN     1/06     1/24     1/41
.PPSV 000034' EN     1/06     1/16     2/02
.STOU 000007' XN     1/09     1/23
..WRL 000050' XN     1/08     2/11
..WRS 000047' XN     1/08     2/10
```

Examination of a System Device Driver (Continued)

As can be seen by an examination of the DCT for a second paper tape punch, this DCT uses entries .PPSV, .PPEX, .PPDT, and .PPIN defined earlier for the first punch. The device code for the second punch, word 4, is the only other difference between the first and second punch DCTs. Since the masks established for both punches are identical, one punch cannot interrupt the other and thus there is no conflict in their use of such portions of PTPDR as the state save area.

```
 0001 PTP1D MACRO REV 02                    15:03:29 01/16/74
01
02                ;REAL TIME OPERATING SYSTEM (RTOS)
03
                          .TITL   PTP1D    ; RTOS SECOND PAPER TAPE PUNCH DRIVER
05
06                        .ENT PTP1D
07
08                        .EXTN .PPSV .PPEX .PPDT .PPIN
09                        .EXTN .OPNO .CLSO ..WRS ..WRL    ;COMMAND ENABLE
10                        .EXTN .STOU .NIOC .COSE
11
12                        .NREL
13
14                ; PAPER TAPE PUNCH DEVICE CONTROL TABLE
15
16        000000'PTP1D=.
17  000000'177777         .PPSV                    ; SAVE MACHINE STATE
18  000001'000007         MKPTP                    ;MASK
19  000002'177777         .COSE                    ;COMMON OUTPUT INTERRUPT SERVIC
20  000003'001701         DCRAT+DCCPO+DCPCK+DCLAC+DCNAF   ; CHARACTERISTICS
21  000004'000053         PTP1                     ;DEVICE CODE
22  000005'177777         .PPEX                    ;EXECUTE
23  000006'177777         .PPDT                    ; DISPATCH TABLE ADDRESS
24  000007'177777         .STOU                    ; PUNCH START ROUTINE
25  000010'177777         .PPIN                    ;DEVICE INIT
26  000011'000000         0                  ;TCB LINK
27  000012'000000         0                  ;TIMEOUT CONSTANT
28  000013'000001         .BLK 1             ;BEAD: LINK
29  000014'000001         .BLK 1             ;DEVICE DATA BYTE POINTER
30  000015'000001         .BLK 1             ;DEVICE DATA COUNT
31  000016'100000         100                ;STATUS WORD - PRIMARY BEAD
32  000017'000013'        .-4                ;BEAD ADDRESS
33  000020'177777         -1                 ;DEVICE REQUEST CHAIN
34  000021'000001         .BLK 1             ;OPENED DEVICE CHARACTERISTICS
35  000022'000002         .BLK 2             ;DEVICE TEMPS
36  000024'000004         .BLK 4             ;MISC.
37
38                        .END
```

*****

CHAPTER 3

USER PROGRAM SERVICED INTERRUPTS

SERVICING USER INTERRUPTS

Special user devices may be identified either at the time an RTOS system is generated (RTOSGEN time) 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. Considerations applying to high priority interrupt routines are given at the end of this chapter. 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.

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 RTOSGEN 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:

Displacement	Mnemonic	Purpose
0	DCTSV	Pointer to a 7-word state save area.
1	DCTMS	Interrupt service mask.
2	DCTIS	Interrupt service routine address.

DCTSV is a pointer to a seven-word state variable save area used by the system to save the PC, accumulators, carry, etc. of the interrupted task. 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.

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

User Device Driver Implementation at Run Time (Continued)

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. In revision 05 of RTOS, exit was accomplished by a jump to the return address specified by AC3 upon entry. In subsequent releases of RTOS, task call .UIEX is issued. Rescheduling will occur immediately after the completion of interrupt dismissal.

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 3 of the RTOS Reference Manual, 093-000056.

Identifying User Interrupt Devices (.IDEF)

In order to introduce to the system those devices (not identified at RTOSGEN time) whose interrupts the system is to recognize, the system call .IDEF must be issued. This call places an entry in the device interrupt vector table, .ITBL. Required inputs to this call are the device code of the user device and the starting address of this device's DCT. The format of this call is:

AC0	-	Device code of the user device.
AC1	-	Starting address of the user device's DCT.

```
.SYSTM
.IDEF
error return
normal return
```

Possible error messages are:

AC2	Mnemonic	Meaning
36	ERDNM	Illegal device code (>77_8). Device code 77_8 is reserved for the power monitor/auto restart option.
45	ERIBS	Interrupt device code in use.

Exit from a User Interrupt Routine (.UIEX)

Upon entering a user interrupt routine, AC3 will contain the address required for exit from the routine. In all versions of RTOS, exit may be accomplished by a jump to the return address specified by AC3 upon entry to the user routine. In revision 3.00 and later revisions, task call .UIEX can be issued to return from a user interrupt routine.

The format of this call is:

```
AC3     -     Return address.

.UIEX
```

Control returns to the location which was interrupted by the user device. No error return need be reserved. .UIEX 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 control table which is to be removed.

The format of this call is:

```
AC0     -     Device code.

.SYSTM
.IRMV
error return
normal return
```

One possible error message may be given.

AC2	Mnemonic	Meaning
36	ERDNM	Illegal device code (>77_8) or attempt to remove a system device (i.e., one established at RTOSGEN time).

Modifying the Current Interrupt Mask (.SMSK)

RTOS contains a task call which permits the current interrupt mask to be modified. When a user interrupt occurs, the interrupt service mask contained in DCTMS of the user device DCT is ORed with the current hardware mask. Nonetheless, it is possible to modify the current mask further by issuing task call .SMSK. The format of this call is as follows:

AC1	-	New hardware mask to be ORed with the mask in effect before the current interrupt.
AC2	-	DCT address

```
.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 RTOS:

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 printers, 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	-	77_8
AC1	-	Starting address of user power up service routine.

```
.SYSTM
.IDEF
error return
normal return
```

The error return is never taken.

Task call .UPEX is issued to provide exit from the user power-up routine.

Exit from a Power Fail Service Routine (.UPEX)

Task call .UPEX is issued to return from a user power-up service routine. The format of this call is:

AC3	-	Return address specified in AC3 upon entry to routine.

```
.UPEX
```

Exit for a Power Fail Service Routine (.UPEX) (Continued)

Control returns to the location which was interrupted by a power failure. No error return nor normal return need be reserved. .UPEX can be issued in a single task environment.

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). This user clock interrupts multitask activity at user-definable intervals. When one of these interruptions occurs, control goes to a user-specified routine outside the current single or multitask environment. No task calls (other than .IXMT) may be issued from this interrupt servicing routine.

Define a User Clock (.DUCLK)

This command permits the definition of a user clock. When an interrupt is generated by this clock, 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 .IXMT) may be issued from this user routine. The format of this call is:

AC0	-	Integer number of RTC cycles during each clock interval.
AC1	-	Address of user routine to receive control.

```
.SYSTM
.DUCLK
error return
normal return
```

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

AC2	Mnemonic	Meaning
45	ERIBS	A user clock already exists.

Upon a user clock interrupt, AC3 will contain the address of the return upon entry to the user routine. Exit from the user clock routine is performed by issuing task call .UCEX. Rescheduling always occurs.

Exit from a User Clock Routine (.UCEX)

The format of the .UCEX call is:

AC1	-	Zero only if rescheduling is to be suppressed.
AC3	-	Return address upon entry to the clock routine.

```
.UCEX
```

Control returns to the point outside the user routine which was interrupted by the user clock. No errors are possible from this call. No error or normal return need be reserved. 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. The format of this call is:

```
.SYSTM
.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 RTOSGEN time.

Analog to Digital Converter

The first of the two illustration programs is an analog to digital converter driver, found on page 3-10. 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 - 11 show that the title of the driver is AIDEV, 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 16 gives the address of the Device Control Table which is defined in lines 18 - 20. This is an abbreviated DCT as discussed earlier.

Analog to Digital Converter (Continued)

The first location in the DCT (line 18) 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 RTOS dispatch table, ITBL. The first routine, AIDEF, clears the ACTIV flag to indicate that there are no active requests in the handler for data. AIDEF then issues the system call .IDEF to define this handler and introduce it to RTOS. An error return could result if the A/D converter device code (21$_8$) 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 RTOS dispatch vector table.

Page 3 of the driver listing contains two subroutines, the analog read random routine (AIRDR) and the read request routine (AICHK). 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 subroutine, AICHK, is used to check the status of a call made to the converter. There are two returns to the caller: busy and call completed.

Analog to Digital Converter (Continued)

Page 4 of the driver listing contains the interrupt service routine, AINTS. This routine is entered from the RTOS interrupt dispatch program with AC2 containing the address of the A/D converter DCT and AC3 containing the address of the RTOS 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.

```
 0001 AIDEV MACRO REV 02.6.21          08:44:12 08/01/73
01
02                ; ANALOG TO DIGITAL CONVERTER DEVICE DRIVER
03                ; ------------------------------------------
04                ; ------------------------------------------
05
                          .TITL AIDEV
07
08                        .ENT AIDEF,AICLR,AIRDR,AICHK
09                        .EXTN .UTEX
10
11                        .NREL
12
13                ; DEVICE CONTROL TABLE LAYOUT
14                ; --------------------------
15
16 00000'000001'ADDCT:  USDCT            ; ADDRESS OF ADCV DCT
17
18 00001'000011'USDCT:  AISAVE           ; INTERRUPT STATE SAVE AREA
19 00002'000200         1B8              ; MASK WORD
20 00003'000063'        AINTS            ; INTERRUPT ROUTINE ADDRESS
21
22                ; ADDITIONAL VARIABLES FOR ADCV HANDLER
23                ; -------------------------------------
24
25 00004'000021 DCODE:  ADCV             ; DEVICE CODE OF A/D CONVERTER
26 00005'000000 ACTIV:  0                ; STATUS FLAG
27 00006'000000 DAPTR:  0                ; DATA ADDRESS ARRAY POINTER
28 00007'000000 DVPTR:  0                ; DATA VALUE ARRAY POINTER
29 00010'000000 DTCNT:  0                ; DATA COUNT STORAGE
30
31 00011'000010 AISAVE: .BLK 10          ; STATE SAVE AREA
32
33
34                ;  USER CONTROL TABLE LAYOUT
35                ; --------------------------
36
37                ;  WORD 0        STATUS FLAG
38                ;                  0 = CALL COMPLETED SUCCESSFULLY
39                ;                -VE = REQUEST BEING PROCESSED
40
41                ;  WORD 1         ADDRESS TABLE POINTER
42
43                ;  WORD 2         DATA STORAGE TABLE POINTER
44
45                ;  WORD 3         NUMBER OF POINTS TO BE READ
46
47        000000 .DUSR    STAT=0
48        000001 .DUSR   ATPTR=1
49        000002 .DUSR   VTPTR=2
50        000003 .DUSR     NPT=3
51
```

```
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         .SYSTM            ; ATTACH TO RTOS 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         .SYSTM            ; DETACH FROM RDOS INTERRUPR SYSTEM
32 00035'021010         .IRMV
33 00036'001400         JMP 0 3
34 00037'001400         JMP 0 3           ; NORMAL RETURN
```

```
;0003 AIDEV
01
02
03                ; ANALOG READ RANDOM SUBROUTINE
04                ; ------------------------------
05
06                ; INPUT:
07                ;       AC2=CONTROL TABLE ADDRESS
08
09                ; CALLING SEUENCE:
10                ;       JSR AIRDR ,
11                ;       <ERROR RETURN>  ; HANDLER ALREADY ACTIVE
12                ;       <NORMAL RETURN>
13
14
15 000400'020745 AIRDR:  LDA 0 ACTIV      ; PICKUP STATUS FLAG
16 000401'101004         MOV 0 0 SZR      ; CHECK IT
17 000402'001400         JMP 0 3          ; ALREADY ACTIVE--ERROR RETURN
18 000403'050742         STA 2 ACTIV      ; SAVE CONTROL BLOCK ADDRESS
19 000404'021003         LDA 0 NPT 2      ; GET # POINTS TO BE READ
20 000405'040743         STA 0 DTCNT      ; SAVE AS DATA COUNT
21 000406'021001         LDA 0 ATPTR 2    ; GET STARTING ADDRESS OF POINT ADDRESSE
22 000407'040737         STA 0 DAPTR      ; SAVE POINT ADDRESS TABLE
23 000500'021002         LDA 0 VTPTR 2    ; GET VALUE STORAGE TABLE ADDRESS
24 000501'040736         STA 0 DVPTR      ; SAVE FOR STORING VALUES
25 000502'102000         ADC 0 0          ; SET REQUEST ACTIVE STATUS
26 000503'041000         STA 0 STAT 2
27 000504'022732         LDA 0 @DAPTR     ; GET FIRST POINT ADDRESS
28 000505'061121         DOAS 0 ADCV      ; START UP CONVERSION
29 000506'001401         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 AICHK
40                ;       <BUSY RETURN>
41                ;       <COMPLETED RETURN>
42
43
44 000507'021000 AICHK:  LDA 0 STAT 2     ; GET CALL STATUS
45 000510'101004         MOV 0 0 SZR
46 000511'001400         JMP 0 3          ; REQUEST BEING PROCESSED
47 000512'001401         JMP 1 3          ; CALL COMPLETED
```

```
[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  000063'062621 AINTS:  DICC 0 ADCV      ; READ VALUE--CLEAR CONVERTER DONE
11  000064'042723         STA 0 @DVPTR      ; SAVE NEWLY READ DATA VALUE
12  000065'014723         DSZ DTCNT         ; DECREMENT DATA COUNTER
13  000066'000405         JMP AIMORE        ; MORE TO COME
14  000067'102400         SUB 0 0           ; REQUEST COMPLETE
15  000070'042715         STA 0 @ACTIV      ; SET USER DONE FLAG
16  000071'040714         STA 0 ACTIV       ; SET CONVERTER NON-BUSY
17  000072'177777         .UIEX             ; DISMISS THE INTERRUPT
18
19  000073'010713 AIMORE: ISZ DAPTR         ; INCREMENT ADDRESS POINTER
20  000074'010713         ISZ DVPTR         ;      "     DATA TABLE POINTER
21  000075'022711         LDA 0 @DAPTR      ; SETUP NEXT CONVERSION
22  000076'061121         DOAS 0 ADCV       ; START IT
23  000077'000072'        .UIEX             ; DISMISS THE INTERRUPT
24
25                        .END
```

```
  0005 AIDEV

ACTIV 000005'  .      1/26    2/12    3/15    3/18    4/15    4/16
ADDCT 000000'         1/16    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'         1/20    4/10
AIRDR 000040'  EN     1/08    3/15
AISAV 000011'         1/18    1/31
DAPTR 000006'         1/27    3/22    3/27    4/19    4/21
DCODE 000004'         1/25    2/13    2/30
DTCNT 000010'         1/29    3/20    4/12
DVPTR 000007'         1/28    3/24    4/11    4/20
USDCT 000001'         1/16    1/18
.UIEX 000077'  XN     1/09    4/17    4/23
```

External Interrupt Recognition

The driver program given at the end of this section 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 RTOS dispatch table (page 2 of the listing), AC0 must contain the communications core location address. After attaching the interrupt to the RTOS 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 is 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 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.

```
 0001 DIDEV MACRO REV 02.6.21              16:27:35 08/03/73
01
02
03                ; DIGITAL INTERFACE (TYPE 4066) DEVICE DRIVER
04                ; ---------------------------------------------
05                ; ---------------------------------------------
06
                          .TITL DIDEV
08
09                        .ENT DIDEF,DICLR
10
11                        .EXTN .IXMT .DIEX
12
13                        .NREL
14
15        000042 .DUSR    DIO=42            ; DEVICE CODE DEFINITION
16
17                ; DEVICE CONTROL TABLE LAYOUT
18                ; ----------------------------
19
20 000000'000021'DIDCT:   USDCT             ; ADDRESS OF DIO DCT
21
22 000001'000004'USDCT:   DISAVE            ; INTERRUPT STATE SAVE AREA
23 000002'000400          1B7               ; INTERRUPT MASK
24 000003'000036'         DINTS             ; INTERRUPT ROUTINE ADDRESS
25
26                ; ADDITIONAL VARIABLES AND STORAGE USED BY HANDLER
27                ; ------------------------------------------------
28
29 000004'000010 DISAVE: .BLK 10            ; STATE SAVE AREA
30 000014'000042 DCODE:   DIO               ; DEVICE CODE
31 000015'000000 DICOM:   0                 ; COMMUNICATIONS ADDRESS
```

```
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 (DTO) ALREADY IN USE
10                ;       <NORMAL RETURN>
11
12 000016'054016 DIDEF:  STA 3 USP
13 000017'040776         STA 0 DICOM      ; SAVE COMMUNICATIONS ADDRESS
14 000020'020774         LDA 0 DCODE
15 000021'024757         LDA 1 DIDCT
16 000022'006017         .SYSTM           ; ATTACH TO RTOS INTERRUPT SYSTEM
17 000023'021007         .IDEF
18 000024'001400         JMP 0 3          ; ERROR RETURN
19 000025'060142         NIOS DIO         ; ARM THE EXTERNAL INTERRUPT
20 000026'001401         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 000027'054016 DICLR:  STA 3 USP
30 000030'060242         NIOC DIO         ; CLEAR DIO DEVICE
31 000031'020763         LDA 0 DCODE
32 000032'006017         .SYSTM           ; DETACH HANDLER
33 000033'021010         .IRMV
34 000034'001400         JMP 0 3          ; NORMAL RETURN
35 000035'001400         JMP 0 3          ;    "       "
36
37
38                ; DIGITAL INTERFACE EXTERNAL INTERRUPT HANDLER
39                ; --------------------------------------------
40
41                ; INPUT: AC2=DCT ADDRESS
42                ;        AC3=INTERRUPT DISMISSAL ADDRESS
43
44 000036'064542 DINTS:  DIAS 1 DIO       ; READ 16 BIT REGISTER
45 000037'125005         MOV 1 1 SNR      ; VALUE ZERO ?
46 000040'001400         JMP 0 3          ; YES--DISMISS INTERRUPT
47
48 000041'054410         STA 3 DISMISS    ; NO--SAVE DISMISSAL ROUTINE ADDRESS
49 000042'102400         SUB 0 0          ; CLEAR SIGNAL ADDRESS
50 000043'042752         STA 0 @DICOM
51 000044'020751         LDA 0 DICOM      ; GET SIGNAL ADDRESS
52 000045'177777         .IXMT            ; WAKE UP USER TASK
53 000046'000401         JMP .+1
54 000047'034432         LDA 3 DISMISS    ; RESTORE AC3 FOR EXIT
55 000050'177777         .DIEX            ; DISMISS THE INTERRUPT
56
57 000051'000000 DISMISS: 0               ; INTERRUPT DISMISSAL ADDRESS
58
59                       .END
```

```
0003 DIDEV

DCODE 000014'          1/3V    2/14    2/31
DICLP 000027' EN       1/09    2/29
DICOM 000015'          1/31    2/13    2/50    2/51
DIDCT 000000'          1/2V    2/15
DIDEF 000016' EN       1/09    2/12
DINTS 000036'          1/24    2/44
DISAV 000004'          1/22    1/29
DISMT 000051'          2/48    2/54    2/57
USDCT 000001'          1/2V    1/22
.IXMT 000045' XN       1/11    2/52
.UIEX 000059' XN       1/11    2/55
```

HIGH PRIORITY INTERRUPT SERVICE

As described in the RTOS Reference Manual, special high priority interrupt devices may be incorporated into an RTOS system at RTOSGEN time. The real time clock and power fail/auto restart device are two such high priority interrupt devices; users may also specify custom high priority interrupt service routines.

All high priority devices have entries in a high priority interrupt dispatch table, .HINT. Entries in this table are scanned whenever an interrupt occurs, and only if no high priority device has caused an interrupt will control branch through the normal interrupt table, .ITBL. All other system devices, and user devices announced at run time (via system call .IDEF), have entries in .ITBL.

Interrupts are disabled whenever high priority interrupt service is being performed. Users writing high priority interrupt service routines must also conform to several programming conventions. In general, these conventions are as follows:

1) Issue no task or system calls.
2) Save and restore accumulators and Carry.
3) Save the contents of location 0, and place a HALT instruction in location 0 (optional).

The state of Carry and the contents of accumulators AC0 through AC2 must be saved within the routine, and these state variables must be restored before leaving the routine. AC3 is saved by the system in location .SAC3 (a location within module INTD), and AC3 too must be restored before exit is accomplished even if the routine didn't use AC3. The contents of location 0 will contain the return address

needed for exit; this address should be stored in a user-provided location, e.g., RET, and a HALT instruction should be stored in location 0. This practice is adhered to by RTOS to capture program control in the event of system failure.

The final two instructions which must be executed when leaving a high priority interrupt service routine are to enable interrupts and then to perform an indirect JMP to the location containing the original return PC, e.g., RET. Control will then pass to the next instruction which would have been executed had no high priority interrupt occurred. The following instruction sequence accomplishes these operations.

```
            .EXTN  .SAC3
            .
            .                       ; RESTORE AC0, AC1, AC2, CARRY
            .
            LDA 3 @ SAC3
            INTEN
            JMP @ RET
SAC3:       .SAC3
RET:        .BLK 1
            .END
```

*****

DataGeneral

SOFTWARE DOCUMENTATION
REMARKS FORM

Document Title	Document No.	Tape No.

SPECIFIC COMMENTS: List specific comments. Reference page numbers when applicable. Label each comment as an addition, deletion, change or error if applicable.

GENERAL COMMENTS: Also, suggestions for improvement of the Publication.

FROM:

Name	Title	Date

Company Name

Address (No. & Street)	City	State	Zip Code

Form No. 10-24-004 Rev. 7-75

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

STAPLE