DATA GENERAL
CORPORATION

Southboro,
Massachusetts 01772
(617) 485-9100

PROGRAM

REAL TIME OPERATING SYSTEM

REFERENCE MANUAL

ABSTRACT

The Real Time Operating System (RTOS) for the DGC family computers consists primarily of a small, core resident, general-purpose multi-task monitor designed to control a wide variety of real time input/output devices. User programs are relieved from the details of I/O timing, data buffering, priority handling, and task scheduling. In addition, tasks are provided with a parallel processing capability plus inter-task communication and synchronization facilities.

Communication with the monitor takes place through a small set of system and task commands. Calling sequences and mnemonics are identical to those in Data General's Real Time Disk Operating System (RDOS). This allows software development and debugging to be carried out on an RDOS system for later use in a core-only RTOS system. Moreover, since RTOS is highly modular, additional device handlers can be added to an RTOS system with relative ease.

093-000056-05

(RTOS 00)	Original Release	-	June, 1971
(RTOS 01 and 02)	First Revision	-	August, 1971
(RTOS 03)	Second Revision	-	January, 1972
(RTOS 04)	Third Revision	-	August, 1972
(RTOS 05)	Fourth Revision	-	August, 1973
(RTOS 3.00)	Fifth Revision	-	March, 1974

This revision of the Real Time Operating System Reference Manual, 093-000056-05, is a major revision. A vertical bar on the outer margin of each page indicates substantially new, changed, or deleted information.

TABLE OF CONTENTS

Chapter 1 - Introduction to RTOS

RTOS Task Concepts 1-1
- Task States 1-2
- Task Identification Numbers 1-3
- Task Synchronization and Communication 1-3
- System and Task Call Formats 1-3

Device Support Under RTOS 1-5

Magnetic Tape and Cassette Units 1-6
- Initializing and Releasing a Tape Drive 1-6
- Free Format I/O 1-7

Asynchronous Data Communications Multiplexor (QTY) 1-7

Disk Files 1-8
- Disk File Organization 1-8

RTOS/RDOS Compatibility 1-9
- Tape and Disk File Structures 1-9
- Task and System Calls 1-9
- Channel/Task Specification 1-10
- RTOS Program Development Under RDOS 1-10
- Avoiding Global Symbol Conflicts under RTOS 1-11
- Generating Console Interrupts 1-11
- Inter-revision Source Level Incompatibilities 1-11

Chapter 2 - System Calls

File and Input/Output Commands 2-1
- System Call List 2-2
- Open a File or Device (.OPEN) 2-4
- Get the Number of a Free Channel (.GCHN) 2-6
- Open a Device for Appending (.APPEND) 2-6
- Close a File or Device (.CLOSE) 2-7
- Close all Files and Devices (.RESET) 2-7
- Initialize a Magnetic Tape Unit or Cassette (.INIT) . . . 2-8
- Open a Cassette or Magnetic Tape Unit for Free Format I/O (.MTOPD) 2-8
- Release a Magnetic Tape Unit or Cassette (.RLSE) . . . 2-9
- Read a Series of Disk Blocks (.RDB) 2-10
- Read a Line (.RDL) 2-11
- Read Sequential (.RDS) 2-12
- Use of the Card Reader in .RDL and .RDS Commands 2-12

Chapter 2 - System Calls (Continued)

Hollerith - ASCII Translation Table 2-14
Write a Series of Disk Blocks (.WRB). 2-16
Write a Line (.WRL) 2-16
Write Sequential (.WRS) 2-17
Free Format Tape I/O (.MTDIO). 2-18
Teletypewriter and Video Display Commands 2-22
Get a Character (.GCHAR). 2-22
Put a Character (.PCHAR) 2-22
Wait for a Keyboard Character (.WCHAR). 2-23
Memory Size Commands 2-23
Determine Available Memory (.MEM) 2-24
Allocate Memory by Changing NMAX (.MEMI). 2-24
System Return Commands 2-25
System Return (.RTN). 2-25
System Error Return (.ERTN) 2-26
Clock and Calendar Commands 2-26
Delay the Execution of a Task (.DELAY) 2-26
Get Today's Date (.GDAY). 2-27
Set Today's Date (.SDAY) 2-27
Get the Time of Day (.GTOD). 2-28
Set the Time of Day (.STOD) 2-28
Examine the System Real Time Clock (.GHRZ) 2-28
Define a User Clock (.DUCLK). 2-29
Exit from a User Clock Routine (.UCEX) 2-30
Remove a User Clock (.RUCLK) 2-30

Chapter 3 - Task Calls

Abort a Task (.ABORT). 3-1
Kill All Tasks of a Specified Priority (.AKILL) 3-2
Ready All Tasks of a Specified Priority (.ARDY). 3-3
Suspend All Tasks of a Given Priority (.ASUSP) 3-3
Get a Task's Status (.IDST) 3-3
Transmit a Message from a User Interrupt Service Routine (.IXMT). 3-4
Delete a Calling Task (.KILL) 3-5
Change the Priority of a Task (.PRI) 3-5
Receive a Message (.REC). 3-5
Suspend a Task (.SUSP). 3-6
Create a Task (.TASK) 3-6

Chapter 3 - Task Calls (Continued)

Kill a Task Specified by I.D. Number (.TIDK) 3-7
Change the Priority of a Task Specified by I.D. Number (.TIDP) 3-7
Ready a Task Specified by I.D. Number (.TIDR) 3-8
Suspend a Task Specified by I.D. Number (.TIDS) 3-8
Transmit a Message (.XMT), and Wait (.XMTW). 3-9

Chapter 4 - User Interrupts and Power Fail/Auto Restart Procedures

Servicing User Interrupts 4-1
Identifying User Interrupt Devices (.IDEF). 4-2
Exit from a User Interrupt Routine (.UIEX) 4-2
Modifying the Current Interrupt Mask (.SMSK) 4-3
Remove User Interrupt Servicing Program (.IRMV) . . . 4-3
Power Fail/Auto Restart Procedures 4-4
Exit from a Power Fail Service Routine (.UPEX) 4-5
High Priority User Interrupt Service 4-5

Chapter 5 - Multiple Processor Systems

Multiple Processor Programming 5-1
Data Transmissions 5-1
Get the Current CPU's MCA Number (.GMCA). 5-2
Multiprocessor System Illustration 5-3

Chapter 6 - System Organization

RTOS Page Zero. 6-1
RTOS Core Map. 6-2
User Status Table (UST) 6-4
Task Control Block (TCB) Pool 6-5
User File Pointers Table (.UFPT). 6-6
Device File Tables. 6-6
High Priority Interrupt Table (.HINT) 6-8
Interrupt Table (.ITBL). 6-9
Standard Device Name Table (.CHTB) 6-10

Appendix A - RTOS Command Summary. A-1

Error Message Summary. A-7

Appendix B - Generating and Loading an RTOS System B-1

Definition of Terms B-1
Preparation for System Generation B-1
System Generation B-1
List of Tapes for System Generation and System Loading B-2
Loading RTOSGEN in a SOS or Stand-alone Environment B-3
Loading RTOSGEN in an RDOS Environment B-3
Producing the RTOS Module B-3
Sample RTOSGEN Dialogue B-11
Loading and Running a Program in a Stand-Alone Environment B-12
Performing a Stand-alone or SOS Relocatable Load. . . . B-12
Executing a Stand-alone Program B-13
Loading and Running a Program in an RDOS Environment . . . B-14
Loading an RTOS Program under RDOS B-14
Executing an RTOS Program with HIPBOOT B-15
Executing an RTOS Program with TBOOT, MCABOOT or CBOOT B-18
Executing an RTOS Program via Paper Tape Produced Under RDOS B-19
Executing an RTOS Program with the SOS Core Image Loader/Writer B-20
RTOS Execution Procedures B-22

Appendix C - RTOS Parameters C-1

Appendix D - RTOS Assembly Language and FORTRAN IV Programming . D-1
Assembly Language Illustration D-1
Sample Assembly Language Program D-2
Load Dialogue and Program Output D-5
Real Time FORTRAN IV Programming. D-6

Appendix E - RTOS Source Level Incompatibilities E-1

CHAPTER 1

INTRODUCTION TO RTOS

The Real Time Operating System (RTOS) for the DGC family computers consists primarily of a small, general purpose multi-task monitor designed to control a wide variety of real time input/output devices. RTOS is entirely core-resident, highly modular and largely reentrant, and allows for the straightforward addition of special device handlers. Moreover, RTOS, revision 3.00 is a compatible subset of RDOS revision 3.00, Data General's Real Time Disk Operating System.

User programs are relieved from the details of I/O timing, data buffering, priority handling, and task scheduling. In addition, users are provided with a parallel processing capability plus inter-task communication and synchronization facilities. Communication with the RTOS monitor takes place through a small set of RTOS system and task calls.

A task is the basic logical unit controlled by RTOS. Tasks are created by means of one of the RTOS task calls, and having been created, a task may be terminated at any time. A large number of common processing situations lend themselves admirably to this sort of operational control philosophy. Examples of these processing situations include the reading or writing of a block of data while simultaneously performing arithmetic computations, listening for input from several devices at the same time, shared device use by multiple tasks, sophisticated communications problems, etc.

RTOS TASK CONCEPTS

A task is a logically complete execution path through user address space which demands the use of system resources. Many tasks may be assigned to operate asynchronously in a single reentrant sequence of instructions, and each task may be assigned a unique priority and identification number.

Due to the serial nature of a computer, tasks which appear to be executing their operations in parallel are in actuality executing these operations in short, serial segments. It is necessary then for RTOS to maintain certain status information (primarily active registers) concerning all tasks which are not currently in control of the Central Processing Unit (CPU).

This information is retained in an information structure called the Task Control Block (TCB). The maximum number of TCBs is defined at the time of system generation.

Task States

In a multitask environment, tasks may exist in either of two states. Tasks are either ready for execution or they are suspended. The highest priority ready task will be given control of the CPU and the other ready tasks await their turn in a queue organized by priority.

Suspended tasks are tasks which were once ready. A task may become suspended for one or more of the following reasons:

1. It has been suspended by .SUSP, .ASUSP, or .TIDS.
2. It has suspended itself for a specified period by .DELAY.
3. It is waiting for a message from another task, .REC.
4. It has issued a message-and-wait call, .XMTW.
5. It is awaiting the completion of a .SYSTM call.

Just as a number of different events may suspend a ready task, several events can cause a suspended task to be readied:

1. The completion of a .SYSTM call (such as a request for I/O or the expiration of a time delay).
2. The posting of a message for a suspended task awaiting its receipt, or the awaited receipt of a transmitted message.
3. The readying of a task by .ARDY or .TIDR task calls.

If a task is suspended by both a task suspend call and by some other event, the call must be readied both by an .ARDY (or .TIDR) call and by whatever other event is required to ready the task.

Suspended and ready tasks are each connected in queues. Tasks may be deleted from either the ready or the suspended queues, either separately (.ABORT or .TIDK) or as a priority class (.AKILL). Tasks which have been deleted add their empty TCBs to an inactive chain of free TCBs. When a task is initiated (.TASK), a TCB is taken from the free chain, the state of the calling task is saved in its own TCB, and both tasks are entered into the ready queue as ready tasks. The .TASK command must be used to initiate a multitask environment.

If all tasks are killed, the effect is to place the entire system in the idle state and to close all channels, with control passing to the task scheduler. The system remains capable of servicing interrupts.

Task Identification Numbers

When a task is created, it may be created both with a unique identification number (I. D.) from 1 to 377_8 and at a specified priority level (from 0 to 377_8). The identification number allows tasks to be readied, suspended, or killed on a selective basis. If unique I. D. 's are not desired, tasks may all be created with I. D. 's of 0. Tasks may exist at priority levels of 0 (the highest) through 377_8 (the lowest priority). Moreover, several or all tasks may exist at the same priority level. The task scheduler always allocates CPU control to the highest priority ready task; ready tasks within the same priority level receive CPU control on a round-robin basis.

Task Synchronization and Communication

RTOS permits tasks to communicate with one another by sending and receiving one-word non-zero messages. A one-word message is sent to a task in an agreed-upon location in user address space. User address space is understood to include all locations from address 16 through NMAX-1 inclusively.

The task sending a message may either return to the Task Scheduler immediately (.XMT) or it may wait (.XMTW) and place itself in the suspended state until the receiving task has issued a receive request (.REC) and has received the message. Receipt of the message includes the resetting of the contents of the message address to all zeroes. Upon receipt of the message, the recipient reverts to the ready state.

System and Task Call Formats

Calls to the RTOS monitor can be separated into two categories: system calls and task calls. System calls generally perform system I/O. Task calls perform user task management functions.

System command words and the mnemonic .SYSTM that must precede each command word are recognized as legal mnemonics by both the RDOS and stand-alone extended assemblers. Appearance of the mnemonic .SYSTM in a program results in the assembling of a JSR @ 17 instruction. The specific system command word is assembled as the word following the mnemonic .SYSTM.

Once system action is complete and the task receives CPU control in priority fashion, normal return is made to the second instruction after the system command word. If an exceptional condition is detected, return is made to the first instruction following the system command word.

System and Task Call Formats (Continued)

The general form of a system call description is:

AC_n - required input to the call

```
.SYSTM
command
error return (error code in AC2)
normal return ( all AC's except AC3 are restored unless output
                is returned via accumulator)
```

AC_n - output from the call

There are 2 basic command word formats:

command n and command

where n is a number from 0 to 76_8 representing an I/O channel number. The maximum number of channels, like the maximum number of tasks, is defined at the time of system generation. Any system command requiring a channel number n need not specify this number in the command word. Instead, by specifying n to be octal 77, the system will use the number passed in AC2 as the channel number.

When no I/O is needed in command execution, the command word appears alone in the instruction. If the command requires arguments, these are passed in the accumulators.

Status of the accumulators upon return from the system call is as follows. If the system returns no information as a result of the call, the carry bit and all accumulators except AC3 will be preserved.

AC2 is used when an exceptional return is made to return a numeric error code. Error codes are listed by number in the RTOS parameter listing, and the applicable codes are listed for each command.

System and Task Call Formats (Continued)

AC3 is destroyed by both .SYSTM and task calls, since they are each equivalent to JSR instructions. On return from the system however, AC3 is loaded with the contents of memory location 016. This location is defined as a permanent symbol by the assembler and has the name User Stack Pointer (USP). A convenient method of saving AC3 is to store it in location 016 before issuing either .SYSTM or task calls.

The general form of a task call in a program is:

AC_n - required input to the call

command
error return (error code in AC2)
normal return (all AC's except AC3 are restored unless output is returned via accumulators)

AC_n - output from the call

Users of task calls are cautioned to reference all task call commands whose operations are required within a program by their call names in an .EXTN statement in that program. Only those calls which are so referenced will have the appropriate task call processing modules loaded by the relocatable loader.

The significant differences between a .SYSTM call and a task call are as follows:

1. Task calls are not preceded by the .SYSTM mnemonic.

2. Not all task calls have error returns. Those which do not have an error return do not reserve an error return location. All system calls reserve error return locations even if there is no error return possible.

DEVICE SUPPORT UNDER RTOS

I/O devices are given special reserved names which often begin with the character $. The following list gives the names of devices supported under RTOS and their reserved names:

$CDR	-	Card reader.
$CDR1	-	Second card reader.
CTn	-	Data General cassette unit n (n can be from 0 to 7).
DK0	-	Data General fixed head NOVADISC.

DEVICE SUPPORT UNDER RTOS (Continued)

DPn	-	Moving head disk, unit n (n can be from 0 to 3).
$LPT	-	80- or 132-column line printer.
$LPT1	-	Second line printer, 80 or 132 columns.
MCAR	-	Multiprocessor Communications Adapter Receiver.
MCAT	-	Multiprocessor Communications Adapter Transmitter.
MTn	-	7- or 9-track magnetic tape transport (n can be from 0 to 7).
$PLT	-	Incremental plotter.
$PLT1	-	Second incremental plotter.
$PTP	-	Paper tape punch.
$PTP1	-	Second paper tape punch.
$PTR	-	High-speed paper tape reader.
$PTR1	-	Second paper tape reader.
QTY	-	4060 asynchronous data communications multiplexor.
$TTI	-	Teletype* or video display terminal keyboard. **
$TTI1	-	Second teletype or display terminal keyboard. **
$TTI2	-	Third teletype or display terminal keyboard. **
$TTO	-	Teletype printer or display terminal screen.
$TTO1	-	Second printer or display terminal screen.
$TTO2	-	Third printer or display terminal screen.

MAGNETIC TAPE AND CASSETTE UNITS

RTOS accesses data on magnetic tape and cassettes in free format. A single system may contain up to eight magnetic and eight cassette tape drives. Magnetic tape units can be in any combination of 7- and 9-track units at high or low desnity.

Initializing and Releasing a Tape Drive

Before any tape files on a tape drive can be accessed, the drive must be initialized via the .INIT system call. Initializing a tape drive causes the tape on that drive to be rewound. Full initialization causes the tape to be rewound and two EOF's to be written (effectively erasing all files from the tape). In both cases the tape file pointer maintained by RTOS is reset to 0.

The system call .RLSE is issued to remove the transport from the system, reset the tape file pointer, and rewind the tape.

*Teletype® is a registered trademark of Teletype Corporation, Skokie, Illinois. All references to teletypes in this manual shall apply to this mark.

**If the teletype reader is turned on for line reads, data read will be echoed on the teletype printer.

Free Format I/O

Data is read and written on magnetic and cassette tape in free format. Data records may be of varying length, containing from 2 to 4096 16-bit words each, and with 1 or more records per file. Each tape reel can contain as many files as the reel size will permit, although only the first 100_{10} files can be positioned directly when a file is opened.

Before any free format I/O can occur on a device, that device must first be initialized and then opened for this type of I/O. The system call .MTOPD is issued to open either mag tape or cassettes for free format I/O. When a tape unit is opened, it is positioned to a specified file, and the unit is associated with an RTOS channel. Thus, even though the unit has positioned a tape reel to a specific file, all files on the tape can then be accessed via space forward/space backward commands, and all records within each file can be similarly accessed.

ASYNCHRONOUS DATA COMMUNICATIONS MULTIPLEXOR (QTY)

The type 4060 asynchronous data communications multiplexor is another device supported by RTOS. RTOS assigns the system mnemonic QTY to this device. The QTY can accommodate from 1 to 64 full or half duplex lines, in either half or full duplex operation.

Each multiplexed line of the QTY corresponds to a file name of the form

QTY:xx

where xx is a multiplexor line number in the range 0-64 decimal. Input/output operations are performed on each line by RTOS line or sequential read and write commands.

Each single QTY line may be opened on a single RTOS channel only. No more than one read or one write request can be outstanding on any one line.

RTOS provides a facility for monitoring line activity on all unopened QTY lines. If line number 64, QTY:64, is opened and either a read line or read sequential operation is attempted, the task issuing this call will be suspended until such time as an unopened line receives an interrupt request. When this occurs, the normal return of the read sequential or read line call will be taken, and AC2 will contain the following data word:

1	0	Line #	Char.
0	1	2 7	8 15

Thus this data word describes the line providing the character, and contains the character itself in its right byte.

If two unopened lines receive an interrupt, only the first one to receive the interrupt will be reported. There will be no report of other unopened interrupting lines occurring before the next read line or read sequential on QTY:64 is issued.

DISK FILES

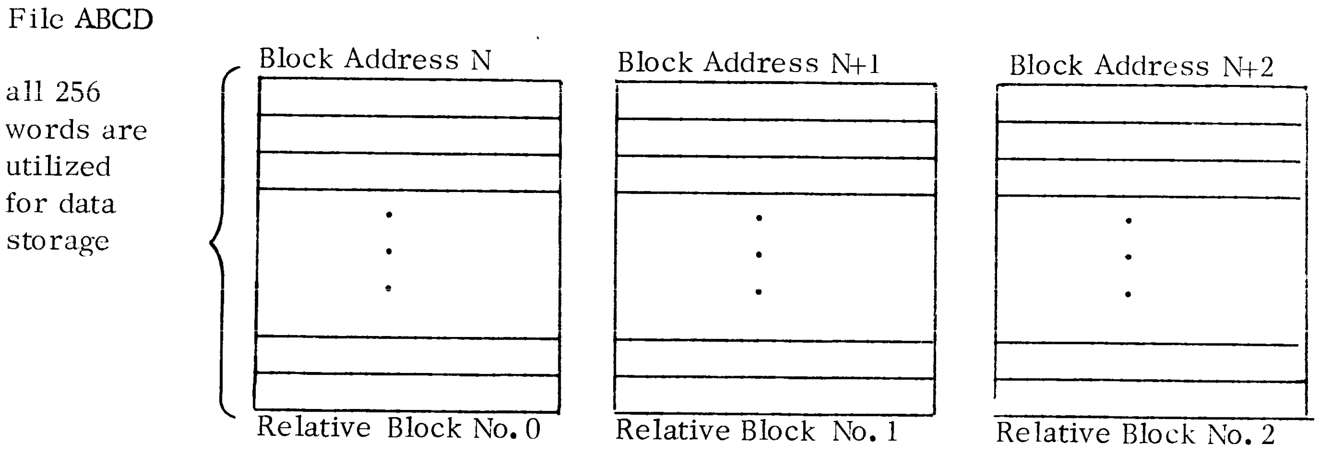
RTOS extends support to both fixed and moving head disk units. RTOS supports a fixed head NOVADISC controller with up to eight logical units of 128K, 256K, 512K or 756K storage words; total fixed head storage per controller is from 131 thousand to 2 million words. Up to four moving head disk devices (disk pack or cartridge type) can also be included in any system, with from 2 to 20 surfaces per unit; maximum total moving head disk storage is 49.2 million words.

Disk files are defined at the time of system generation. At this time, file sizes are specified and names are assigned to each file. File names consist of from 4 to 6 ASCII characters followed by a trailing null. Allowable ASCII characters in the file name are all upper case alphabetic characters and numerals 0 through 9. A file must be opened (i.e., associated with an RTOS channel) before it can be accessed.

Disk File Organization

Disk files in RTOS may be organized contiguously only. Contiguously organized files consist of a fixed number of one or more disk blocks located at an unbroken series of disk block addresses. These files can be neither expanded nor reduced in size. Since the data blocks are at sequential logical block addresses, all that is needed to access a block within a contiguous file is the address of the first block and the relative block number within the file.

Disk File Organization (Continued)



Contiguously organized files have the advantage of allowing quick access to their data blocks, with all disk storage being used for data.

RTOS/RDOS COMPATIBILITY

The design intent of RTOS is to make this operating system a compatible subset of RDOS, the DGC Real Time Disk Operating System. Accordingly, all file structures, task concepts, system and task calls, and other features discussed for RTOS may be tested and run on an RDOS system. RDOS features omitted from RTOS were omitted because they could not be applied in RTOS, given the constraints of a quick responding core-resident-only operating system.

Tape and Disk File Structures

Cassette and magnetic tape file structures under RTOS are identical to those provided by free format I/O under RDOS.

Disk files in RTOS must be organized contiguously; sequentially and randomly organized disk files are available only under RDOS. Names are assigned to RTOS disk files at RTOSGEN time. Allowable ASCII characters in RTOS file names are all upper case alphabetics and numerals 0 through 9. Unlike RDOS, however, disk file names must be 4 to 6 characters in length, may not include the $ character, and can have no file name extensions; file names defined at RTOSGEN time cannot be changed. Disk files must be opened before they may be accessed.

Task and System Calls

Unless otherwise specified, all RTOS system and task calls operate as they do under RDOS. Any attempt to reference a task call under RTOS which is not

Task and System Calls (Continued)

found in its library will cause an unresolved external to be reported by the loader.

In most cases, an attempt to execute an RDOS system call which is not implemented in RTOS results in an error return being taken with error code 2, ERICM (illegal system command), reported in AC2. However, some calls are treated as no-ops by RTOS to achieve downward compatibility (from RDOS to RTOS). The following list names all of these calls which cause control to go directly to the normal return with no further action:

.CCONT	Create a contiguous file.
.CRAND	Create a random file.
.DELET	Delete a file.
.RENAM	Rename a file.
.SYSI	SOS-compatible call.
.SPKL	Kill spooling.
.SPDA	Disable spooling.
.SPEA	Enable spooling.

Note that these calls will not operate correctly under RTOS in the presence of error conditions. Thus, for example, an attempt to delete a non-existent file would take the error return for .DELET under RDOS, but would take this call's normal return under RTOS.

Channel/Task Specification

The number of channels and tasks in a program run under RTOS is defined at RTOSGEN time. RTOS programs being tested under RDOS may specify channels and tasks by means of the RLDR local switches /C and /K, or by the .COMM TASK (or CHANTASK) statements.

RTOS Program Development Under RDOS

Bearing in mind the above restrictions, RTOS program development under an RDOS system is convenient. RTOS programs are edited, assembled, loaded and debugged under RDOS, and when they are considered to be error free they are reloaded with an RTOS module (produced by RTOS SYSGEN) and the RTOS library, using the RDOS RLDR command with the /C global switch. This load procedure creates a version of the program (called a save file) which can be run under RTOS.

Having created this save file, one of several bootstrap procedures are followed to execute the program, depending upon whether the RTOS program is to be run on the system which is currently executing under RDOS or whether it is to be run under another system, with or without disk, which is not running under RDOS.

RTOS Program Development Under RDOS (Continued)

Complete details describing the use of the RDOS RLDR command, operation of TBOOT, CBOOT, HIPBOOT, and procedures for relocatable loading of an RTOS program module on a stand-alone system are outlined in Appendix B of this manual.

Avoiding Global Symbol Conflicts under RTOS

In order to minimize the incidence of global symbol conflicts between user programs and RTOS subprograms, RTOS generally follows the convention of using a dot as the first character in each symbol. Thus user-defined global symbols should always avoid the use of a dot as the first character in a symbol.

This rule has several exceptions which users should be aware of, i.e., several global symbols in RTOS are not preceded by dots. These undotted global symbols are as follows: PWRIS, RTCIS, DCT names (summarized at the end of system generations), entries in the buffer package (BFPKG), entries in Fortran libraries FORT. LB and RTOS FMT. LB, and user high priority interrupt handlers.

Generating Console Interrupts

RTOS does not permit the generation of console interrupts (CTRL A, CTRL C, or CTRL F) which are available under RDOS. Nonetheless, RTOS does provide a facility which resembles an RDOS keyboard interrupt. This facility is the keyboard character wait command, .WCHAR . This command, discussed fully in Chapter 2. activates logic within RTOS such that when a user-specified character is received from any console keyboard, control will branch to a user-specified routine for appropriate processing.

Inter-revision Source Level Incompatibilities

Users of RTOS 05 who wish to be upgraded to RTOS 3.00, the current revision, should be aware of certain source level incompatibilities between the two revisions. A summary of these considerations is given in Appendix E.

***

CHAPTER 2

SYSTEM CALLS

The following page contains an alphabetized list of all RTOS system command word mnemonics.

All system calls except user interrupt calls will be discussed in this chapter, with calls being grouped in the following sections:

File and Input/Output Commands
Teletypewriter/Video Display Commands
Memory Size Commands
System Return Commands
Clock and Calendar Commands

FILE AND INPUT/OUTPUT COMMANDS

All I/O is handled by system I/O commands. These commands require a channel number from 0 to 77 to be given in the argument field of the command word as discussed in Chapter 1. The number of channels available is determined by the user when the RTOS module is generated by the user (see Appendix B).

The user may also define a number of fixed length files when the RTOS module is generated, and may assign alphanumeric names to these files. Such files are organized contiguously, and are composed of a fixed number of disk blocks which are located at an unbroken series of physical block addresses. These files can neither be expanded nor reduced in size. Since the data blocks are in sequence, all that is needed to access a block within a contiguous file is the name of the file (indicating to RTOS the address of the first block) and the relative block number within the file. Since no time is required for reading a file index, disk files under RTOS can be accessed rapidly.

System Call List

.APPEND	Append to a device.
.CLOSE	Close a file or device.
.DELAY	Suspend a task for a specific interval of time.
.DUCLK	Define a user clock.
.ERTN	Idle the system abnormally.
.GCHAR	Get a character.
.GCHN	Get a free channel number.
.GDAY	Get today's date.
.GHRZ	Get the real time clock frequency.
.GMCA	Get the current CPU's MCA number.
.GTOD	Get the time of day.
.IDEF	Identify a user interrupt device.
.INIT	Initialize a magnetic tape or cassette.
.IRMV	Remove a user interrupt device.
.MEM	Determine available memory.
.MEMI	Change NMAX.
.MTDIO	Perform free format I/O on magnetic tape or cassette.
.MTOPD	Open a magnetic tape or cassette for free format I/O.
.OPEN	Open a file or device other than the magnetic tape or cassette.
.PCHAR	Output a character to the teletypewriter.
.RDB	Read a disk block.
.RDL	Read a line.
.RDS	Read sequential bytes.
.RESET	Close all devices and files.
.RLSE	Release a magnetic tape or cassette unit.
.RTN	Idle the system normally.
.RUCLK	Remove a user clock.
.SDAY	Set today's date.
.STOD	Set the time of day.
.WCHAR	Wait for a character on a teletype.
.WRB	Write a disk block.
.WRL	Write a line.
.WRS	Write sequential bytes.

A channel is initially linked to a particular file or device by means of the .OPEN (or .APPEND) command. A channel is linked to a magnetic tape transport or cassette unit by means of the .MTOPD command only. The association between file or device and channel number is broken by using the .CLOSE command. All currently open files and devices can be closed, and their associated channels freed, by means of the system .RESET command.

RTOS provides four different I/O modes for reading and writing. These modes are:

- direct block
- line
- sequential
- free format tape I/O

In direct block mode, the user effects a transfer of a continuous group of disk blocks. Core locations used in the transfer are also in sequence. The direct block mode commands are .RDB and .WRB, read a block series and write a block series. This mode is used only with disk I/O, and no other I/O mode is used for disk I/O under RTOS.

Line mode data transfers assume that the data read or written consists of ASCII character strings terminated by either carriage returns, form feeds, or nulls. Position within a file is implicit from the last call. That is, file data is processed line by line in sequence from the beginning of the file to its end. In this mode the system handles all device dependent editing at the device driver level. For example, line feeds are ignored on paper tape and teletype input devices and are supplied after carriage returns to all paper tape and teletype output devices. Moreover, reading and writing do not require byte counts since reading continues until a terminator is read and writing proceeds until a terminator is written. The line mode commands are .RDL and .WRL, read and write a line.

The third mode is sequential mode. In this mode data is transmitted exactly as read from the device or from core memory. No assumption is made by the system as to the nature of this information. Thus this mode would always be used for processing binary data. This mode requires the caller to specify a byte count for each read or write request. The sequential mode commands are .RDS and .WRS, read and write sequential.

Free format I/O permits the reading or writing of data on a word by word basis to cassette or magnetic tape. This mode provides users with the means of accessing data in variable size records within tape files. Free format I/O permits the reading or writing of data records containing from 2 to 4096 words each. Free format I/O commands also permit a tape reel to be spaced forward or backward from 1 to 4095 records or to the start of a new data file, and these commands permit the transport status word to be read.

Before free format I/O operations can be performed, the cassette or magnetic tape unit must be initialized (.INIT) and opened (.MTOPD). To prevent further I/O access from occurring, the device is released (.RLSE) and its channel is closed (.CLOSE).

Open a File or Device (.OPEN)

Before other I/O commands can be used, files and devices must be linked to channels. Two parameters must be passed to .OPEN : a byte pointer to the device or file name string, and a characteristic inhibit mask.

For every bit set in the characteristic inhibit mask word a corresponding device characteristic is inhibited. (This mask is ignored when .OPEN is opening a file.) Furthermore, these characteristics will be inhibited for as long as the device remains open. The following lists the bit assignments, characteristic mnemonics, and the characteristics in the inhibit mask:

Bit	Mnemonic	Meaning
1	DCC80	80-column device.
2	DCLTU	Device changing lower case ASCII to upper case.
3	DCFFO	Device requiring a form feed on open.
4	DCFWD	Full word device (reads or writes more than a byte).
6	DCLAC	Device requiring line feeds after carriage returns.
7	DCPCK	Input device requiring a parity check; output device requiring parity computation.
8	DCRAT	Output device requiring a rubout after every tab.
9	DCNAF	Output device requiring nulls after every form feed.
10	DCKEY	A keyboard input device.
11	DCTO	A teletype output device.
12	DCCNF	Output device without form feed hardware.
14	DCCGN	Output device without tabbing hardware.
15	DCCPO	Output device requiring leader/trailer.

If an MCA line is being opened, AC1 cannot contain a characteristic inhibit mask. Instead, for receiver lines, AC1 must be cleared to all zeroes. If a transmitter line is to be opened and the default number of retries (specified at RTOSGEN time) is to be used, AC1 must be cleared to all zeroes. However, if a different timeout value is to be specified, bit 15 of AC1 must be set to one (and all other bits in AC1 must be cleared). The actual specification of a retry count will then be deferred to the time the write sequential I/O command (.WRS) is issued.

Open a File or Device (.OPEN) (Continued)

Having opened a file via .OPEN, the user is not guaranteed of being the exclusive user of the file; others may also have opened the file via .OPEN and may have modified its contents. This command cannot be used to open a cassette or magnetic tape file.

The format of the .OPEN command is:

AC0 - Byte pointer to file or device name terminated by a null. The pointer must be even, i.e., the string must begin on a full word boundary.

AC1 - Characteristic inhibit mask.

```
        .SYSTM
        .OPEN n             ; OPEN CHANNEL n
        error return
        normal return
```

In general, the user will wish to preserve all device characteristics as defined by the system. This can be accomplished by preceding the .OPEN call with a SUB 1,1 instruction, passing an all-zero mask in AC1.

As an example, if the user wishes to read an ASCII tape without parity from the high speed reader, he may inhibit parity checking by the following command sequence:

```
        LDA     0, READR
        LDA     1, MASK
        .SYSTM
        .OPEN   3

READR:  .+1*2
        .TXT    *PTR*
MASK:   DCPCK                  ;PARITY CHARACTERISTIC
```

Possible errors resulting from the .OPEN command are:

AC2	Mnemonic	Meaning
0	ERFNO	Illegal channel number.
1	ERFNM	Illegal file name.
12	ERDLE	File does not exist.
21	ERUFT	Attempt to use channel already in use.
31	ERSEL	Unit improperly selected.
60	ERFIU	Attempt to open a busy MCA unit.

Get the Number of a Free Channel (.GCHN)

This call enables the user to obtain the number of a channel that is currently unused, if any, so that a file may be opened on this channel via one of the file open calls. .GCHN does not open a file on a free channel; it merely indicates a channel that is free at the moment. Thus if .GCHN is issued in a multitask environment where it is followed by a call to one of the file open commands, the channel may not be free by the time the file open call is issued. To solve this problem, the user should process error return ERUFT ("Channel already in use") given by the file open command by reissuing the call to .GCHN; this will ensure that a truly free channel is discovered.

The format of this call is:

```
.SYSTM
.GCHN
error return
normal return
```

Upon a normal return, the channel number is returned in AC2:

AC2 - Free channel number

One possible error return may occur.

AC2	Mnemonic	Meaning
21	ERUFT	No channels are free.

Open a Device for Appending (.APPEND)

An alternate system call for opening a device is implemented that is identical to .OPEN in every respect except that it may not be used with disk files.

This routine requires the same two input parameters as does .OPEN, viz., a byte pointer to the device name string, and a characteristic inhibit mask. The mask bit definitions are as described earlier for .OPEN .

The format of the .APPEND command is:

AC0 - Byte pointer to device name. The pointer must be even, i.e., the string must begin on a full word boundary. The string must be terminated by a null.

AC1 - Characteristic inhibit mask.

Open a Device for Appending (.APPEND) (Continued)

```
.SYSTM
.APPEND n
error return
normal return
```

Possible errors resulting from the .APPEND command are:

AC2	Mnemonic	Meaning
0	ERFNO	Illegal channel number.
21	ERUFT	Attempt to use channel already in use.

Close a File or Device (.CLOSE)

This command closes a device (performing any housekeeping required like trailer output) and frees the device channel. If a file is closed, the file's channel is released. The format of the .CLOSE command is:

```
.SYSTM
.CLOSE n                    ;CLOSE CHANNEL n
error return
normal return
```

Possible errors resulting from a .CLOSE command are:

AC2	Mnemonic	Meaning
0	ERFNO	Illegal channel number.
15	ERFOP	Attempt to reference a channel not in use.

Close all Files and Devices (.RESET)

This command causes all currently open files and devices to be closed. The format of the .RESET command is:

```
.SYSTM
.RESET
error return
normal return
```

The error return is never taken.

Initialize a Magnetic Tape Unit or Cassette (.INIT)

Before free format tape I/O can occur, the magnetic tape unit or cassette must be initialized. Initialization for a magnetic tape unit or cassette consists of making the device known to the system, rewinding the tape to BOT, and resetting the tape file pointer to zero.

A full initialization causes the tape to be rewound to BOT and two end-of-file marks to be written. This effectively erases any information which may have been on the tape. A partial initialization causes the tape to be rewound to BOT and resets the system tape file pointer to zero; no end of file mark is written.

Input parameters to this call are as follows:

- AC0 - Byte pointer to cassette or magnetic tape name (e.g., MT0, CT2, etc.). The byte pointer must be even, i.e., the string must begin on a full word boundary.
- AC1 - -1 for full initialization; other values indicate a partial initialization.

The format of the .INIT command is:

```
.SYSTM
.INIT
error return
normal return
```

Possible errors resulting from an .INIT command are:

AC2	Mnemonic	Meaning
2	ERICM	Illegal command for device.
31	ERSEL	Unit improperly selected.
36	ERDNM	Device not in system.
45	ERIBS	Device already initialized.

Open a Cassette or Magnetic Tape Unit for Free Format I/O (.MTOPD)

Before free format reading or writing can be performed on either an initialized magnetic tape or cassette unit, the device must be opened and be linked to an RTOS channel. The RTOS command to open files or devices (.OPEN) cannot be used to open a magnetic or cassette tape unit for free format I/O; only .MTOPD can be used to open these devices.

Open a Cassette cr Magnetic Tape Unit for Free Format I/O (.MTOPD) (Continued)

Input parameters to this call are the same as for the .OPEN command. .MTOPD positions a free format tape to a desired file, since the file name passed to .MTOPD will be of the form MTn:m or CTn:m . The input parameter to .MTOPD is as follows:

AC0 - Byte pointer to cassette or magnetic tape file name terminated by a null. The pointer must be even, i.e., the string must begin on a full word boundary.

The format of the MTOPD command is:

```
.SYSTM
.MTOPD n                    ;n IS THE CHANNEL NUMBER
error return
normal return
```

Possible errors resulting from a .MTOPD command are:

AC2	Mnemonic	Meaning
0	ERFNO	Illegal channel number.
1	ERFNM	Illegal file name.
12	ERDLE	File does not exist.
21	ERUFT	Attempt to use channel already in use.
31	ERSEL	Unit improperly selected.

Release a Magnetic Tape Unit or Cassette (.RLSE)

To prevent further file access to either a magnetic tape or cassette unit, the system command .RLSE must be issued. This command prevents further file access until the device is initialized (see the .INIT command), and causes the tape to be rewound to BOT.

One input parameter is required for this call:

AC0 - Byte pointer to device name. The byte pointer must be even, i.e., the string must begin on a full word boundary.

The format of this call is:

```
.SYSTM
.RLSE
error return
normal return
```

Possible errors resulting from an .RLSE command are:

AC2	Mnemonic	Meaning
2	ERICM	Illegal command for device.
36	ERDNM	Device not in system.

Read a Series of Disk Blocks (.RDB)

This command causes a series of disk blocks to be read into a user-specified area in core memory. This routine requires four input parameters including the number of the channel upon which the disk file was previously opened. These parameters are: the starting disk block number within the disk file, the number of disk blocks to be read, and the starting (i.e., lowest) core address to receive the data. In the case where the channel number n in the command argument is set to 77, the right byte of AC2 contains the channel number. The left byte of AC2 contains the number of blocks to be transferred. Each block that is read contains 256 16-bit words of disk storage.

The format of the .RDB command is:

AC0 - Starting core address to receive the data.
AC1 - Starting relative disk block number.
AC2, left byte - Number of blocks to be read.
AC2, right byte - Optional channel number.

```
.SYSTM
.RDB n                     ;n IS THE CHANNEL NUMBER.
error return
normal return
```

Possible error codes resulting from an .RDB command are:

AC2	Mnemonic	Meaning
0	ERFNO	Illegal channel number.
3	ERICD	Illegal command for device.
6	EREOF	End of file.
15	ERFOP	File is not opened.

Upon detection of error EREOF the error code is returned in the right byte of AC2; the left byte of AC2 contains the partial read count.

Read a Line (.RDL)

This command causes an ASCII line to be read. Required input to this command is a byte pointer in AC0 to the starting byte address within the user area into which the line will be read. This area should be 133 bytes long.

Reading will be terminated normally after a carriage return, form feed, or null is detected. Reading will be terminated abnormally after reading 132 (decimal) characters without detecting a carriage return, form feed, or null; upon detecting a parity error; or upon an end of file. In all cases the read byte count, including the carriage return, form feed, or null, will be returned in AC1.

If the read is terminated because of a parity error, the character having the incorrect parity will be stored (with its parity bit cleared) as the last character read. The byte pointer to the character in error can always be computed as:

(AC0) + (AC1)-1 *

The format of the .RDL command is:

AC0 - Starting byte address.

```
.SYSTM
.RDL n                    ;READ FROM CHANNEL n
error return
normal return
```

AC1 - Byte count.

Possible errors resulting from a .RDL command are:

AC2	Mnemonic	Meaning
0	ERFNO	Illegal channel number.
3	ERICD	Illegal command for device.
6	EREOF	End of file.
15	ERFOP	Attempt to read an unopened file.
22	ERLLI	Line limit (132 characters) exceeded.
24	ERPAR	Parity error.
47	ERSIM	Simultaneous reads on same QTY line.
106	ERCLO	QTY input terminated by channel close.

* (ACn) means "contents of ACn"

Read Sequential (.RDS)

This command causes a sequential mode data transfer, causing data to be read exactly as is. Required input parameters to this command are as follows: a byte pointer to the starting byte address within the user area into which data will be read, and the number of bytes (2^{15}-1 maximum) to be read.

The format of the .RDS command is:

AC0 - Starting byte address.
AC1 - Number of bytes to be read.

```
.SYSTM
.RDS n          ;READ SEQUENTIAL FROM CHANNEL n
error return
normal return
```

Possible errors resulting from an .RDS command are:

AC2	Mnemonic	Meaning
0	ERFNO	Illegal channel number.
3	ERICD	Illegal command for device.
15	ERFOP	Attempt to read an unopened device or file.
47	ERSIM	Simultaneous reads on same QTY line.
106	ERCLO	QTY/MCA input terminated by channel close.

Use of the Card Reader in .RDL and .RDS Commands

When using $CDR or $CDR1, the end of file condition on a .RDL will occur only if a special end of file code is detected in column 1 of a card. This code is "all rows punched." It can be punched on a 029 keypunch by multipunching "+", "-", and "0" through "9."

Note also that a Hollerith to ASCII translation only occurs if a .RDL has been requested. The translation assumes 029 keypunch codes. A table of Hollerith-ASCII translation is given in this section.

A .RDL is terminated upon the first trailing blank (which is translated as a carriage return). If all 80 columns are data, a carriage return is appended as the eighty-first character. If an illegal character is detected, a back slash is substituted for the illegal character.

Use of the Card Reader in .RDL and .RDS Commands (Continued)

In an .RDS command, the card is read in image binary. Each two bytes will be used to store a single column. The packing is done as follows:

Byte	←				1			→	←				2			→
Column Number					12	11	0	1	2	3	4	5	6	7	8	9
Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
	0	0	0	0	d	d	d	d	d	d	d	d	d	d	d	d

The "d's" will be 1 for every column punched.

The byte count and byte pointer input to an .RDS command must both be even.

CHAR.	CARD CODE	ASCII CODE
NUL	12-0-9-8-1	000
SOH	12-9-1	001
STX	12-9-2	002
ETX	12-9-3	003
EOT	9-7	004
ENQ	0-9-8-5	005
ACK	0-9-8-6	006
BEL	0-9-8-7	007
BS	11-9-6	010
HT	12-9-5	011
LF	0-9-5	012
VT	12-9-8-3	013
FF	12-9-8-4	014
CR	12-9-8-5	015
SO	12-9-8-6	016
SI	12-9-8-7	017
DLE	12-11-9-8-1	020
DC1	11-9-1	021
DC2	11-9-2	022
DC3	11-9-3	023
DC4	4-8-9	024
NAK	9-8-5	025
SYN	9-2	026
ETB	0-9-6	027
CAN	11-9-8	030
EM	11-9-8-1	031
SUB	9-8-7	032
ESC	0-9-7	033
FS	11-9-8-4	034
GS	11-9-8-5	035
RS	11-9-8-6	036
US	11-9-8-7	037

CHAR.	CARD CODE	ASCII CODE
SPACE	NO PUNCHES	040
!	12-8-7	041
"	8-7	042
#	8-3	043
$	11-8-3	044
%	0-8-4	045
&	12	046
' or ´	8-5	047
(	12-8-5	050
)	11-8-5	051
*	11-8-4	052
+	12-8-6	053
,	0-8-3	054
-	11	055
.	12-8-3	056
/	0-1	057
0	0	060
1	1	061
2	2	062
3	3	063
4	4	064
5	5	065
6	6	066
7	7	067
8	8	070
9	9	071
:	8-2	072
;	11-8-6	073
<	12-8-4	074
=	8-6	075
>	0-8-6	076
?	0-8-7	077

Hollerith - ASCII Translation Table

CHAR.	CARD CODE	ASCII CODE
@	8-4	100
A	12-1	101
B	12-2	102
C	12-3	103
D	12-4	104
E	12-5	105
F	12-6	106
G	12-7	107
H	12-8	110
I	12-9	111
J	11-1	112
K	11-2	113
L	11-3	114
M	11-4	115
N	11-5	116
O	11-6	117
P	11-7	120
Q	11-8	121
R	11-9	122
S	0-2	123
T	0-3	124
U	0-4	125
V	0-5	126
W	0-6	127
X	0-7	130
Y	0-8	131
Z	0-9	132
[	12-8-2	133
\	0-8-2	134
]	11-8-2	135
¬ or ↑	11-8-7	136
— or ←	0-8-5	137

CHAR.	CARD CODE	ASCII CODE
`	8-1	140
a	12-0-1	141
b	12-0-2	142
c	12-0-3	143
d	12-0-4	144
e	12-0-5	145
f	12-0-6	146
g	12-0-7	147
h	12-0-8	150
i	12-0-9	151
j	12-11-1	152
k	12-11-2	153
l	12-11-3	154
m	12-11-4	155
n	12-11-5	156
o	12-11-6	157
p	12-11-7	160
q	12-11-8	161
r	12-11-9	162
s	11-0-2	163
t	11-0-3	164
u	11-0-4	165
v	11-0-5	166
w	11-0-6	167
x	11-0-7	170
y	11-0-8	171
z	11-0-9	172
{	12-0	173
		12-11	174
}	11-0	175
~	11-0-1	176
DEL	12-9-7	177

Hollerith - ASCII Translation Table (Continued)

Write a Series of Disk Blocks (.WRB)

This command causes a series of data blocks, each 256 16-bit words in size, to be written onto disk from a user-specified area in core memory. This routine requires four input parameters including the number of the channel upon which the disk or disk file was previously opened. These parameters are: the starting relative disk block number within the file, the number of disk blocks to be written, and the starting (i.e., lowest) core address to transmit the data. In the case where the channel number n in the command argument field is set to 77, the right byte of AC2 contains the channel number. The left byte of AC2 contains the number of blocks to be written.

The format of the .WRB command is:

AC0 - Starting core address to transmit the data.
AC1 - Starting relative disk block number.
AC2, left byte - Number of blocks to be read.
AC2, right byte - Optional channel number.

```
.SYSTM
.WRB     n          ;n IS THE CHANNEL NUMBER
error return
normal return
```

Possible error codes resulting from an .WRB command are:

AC2	Mnemonic	Meaning
0	ERFNO	Illegal channel number.
3	ERICD	Illegal command for device.
15	ERFOP	File is not opened.
27	ERSPC	Disk space is exhausted.

Upon detection of error ERSPC, the error code is returned in the right byte of AC2; the left byte of AC2 contains the partial write count.

Write a Line (.WRL)

This command writes a line to a user ASCII file. Required input to this command is a byte pointer in AC0 defining the user core area from which writing will occur.

Characters are written with even parity and writing will be terminated normally upon encountering a null, carriage return, or form feed. Writing will be terminated abnormally after the transmission of 132 (decimal) characters if the 133rd character is not a carriage return, form feed, or null. In all cases, AC1 will contain, upon termination of the write, the number of bytes written from the user area. The termination of a write line on a null allows formatting of output without forcing a carriage return.

The format of the .WRL command is:

AC0 - Starting byte address.

```
.SYSTM
.WRL n              ;WRITE FROM CHANNEL n
error return
normal return
```

AC1 - Byte count.

Possible errors resulting from the .WRL command are:

AC2	Mnemonic	Meaning
0	ERFNO	Illegal channel number.
3	ERICD	Illegal command for device.
15	ERFOP	Attempt to write an unopened file.
22	ERLLI	Line limit (132 characters) exceeded.
47	ERSIM	Simultaneous writes on same QTY line.
106	ERCLO	QTY output terminated by channel close.

Write Sequential (.WRS)

This command writes binary data from a user core area. In addition to the channel number, two parameters are input: a byte pointer to the starting address of the user area, and the number of bytes (2^{15}-1 maximum) to be written. The byte pointer must be even if this call is issued for an MCA transmission. To transmit an end-of-file in an MCA transmission, a byte count of zero is used in AC1 and the contents of AC0 are disregarded. Also, if the MCA transmit line was opened with a timeout to be specified, the left byte of AC2 input to .WRS specifies the length of the timeout period. Each unit period is approximately 200 milliseconds, and acceptable multiples input in AC2 are 1 to 255 decimal. A zero value yields the default timeout period specified at RTOSGEN time.

Write Sequential (.WRS) (Continued)

The format of the .WRS command is:

AC0 - Starting address of the data within the user area.
AC1 - Byte count.
AC2, left byte - Optional timeout constant.
AC2, right byte - Optional channel number.

```
.SYSTM
.WRS n          ;WRITE TO CHANNEL n
error return
normal return
```

Possible errors are:

AC2	Mnemonic	Meaning
0	ERFNO	Illegal channel number.
3	ERICD	Illegal command for device.
15	ERFOP	Attempt to reference an unopened file or device.
47	ERSIM	Simultaneous writes on same QTY line.
101	ERDTO	Timeout has occurred.
103	ERMCA	No complementary MCA request.
104	ERSRR	Incomplete MCA transmission due to short receiver request.
106	ERCLO	MCA/QTY output terminated by channel close.
113	ERNMC	No MCA receiver request.

Free Format Tape I/O (.MTDIO)

This command permits the operation of magnetic tape and cassette units on a machine level: To read and write words in variable length records of from 2 to 4096 words within a data record, to space forward or backward from 1 to 4095 data records or to the start of a new data file, and to perform other similar machine level operations. Before free format I/O can be performed on a tape unit, that unit must first have been opened for free format I/O by means of the .MTOPD system command.

The input parameters to .MTDIO are as follows:

AC0 - Core data address, if data is to be transferred.
AC1 - Command word, subdivided into the following fields:

bit 0: set to 1 for even parity, 0 for odd parity.
bits 1-3: set to one of the seven command codes which follow.

- 0 - read (words)*
- 1 - rewind the tape
- 3 - space forward (over records or over a file of any size)
- 4 - space backward (over records or over a file of any size)
- 5 - write (words)
- 6 - write end of file
- 7 - read device status word

bits 4-15: word or record count. If 0 on a space forward (or space backward) command, the tape is positioned to the beginning of the next (or previous) file on the tape. If 0 on a read or write command 4096 words are read (or written) unless an end of record is detected.

AC2 - channel number if n equals 77_8. (Note that when a file is opened for free format I/O, it is opened globally. That is, all files on the specified device can be accessed.)

The format of the .MTDIO command is:

```
.SYSTM:
.MTDIO  n               ;n IS THE CHANNEL NUMBER
error return
normal return
```

Upon a rewind or read status command, if no system error is detected, AC2 returns containing a status word with one or more of the following bits set:

- bit 0 - error (bit 1, 3, 5, 6, 7, 8, 10 or 14 is set to 1)
- bit 1 - data late
- bit 2 - tape is rewinding
- bit 3 - illegal command
- bit 4 - high density if set to 1; otherwise, low density (always 1 for cassettes)
- bit 5 - parity error
- bit 6 - end of tape
- bit 7 - end of file

* When attempting to read a 7-track tape with odd parity (i.e., a tape not written on an RTOS system), the end-of-file is not detected by the controller; the first word in the next record is read as 007417. Thus the first record of each file (after the first file) has appended to it the end-of-file of the previous file.

bit 8 - tape is at load point
bit 9 - 9-track if set to 1; otherwise, 7-track (always set to 1 for casettes)
bit 10 - bad tape or write failure
bit 11 - send clock (always set to zero for cassettes)*
bit 12 - first character (always set to zero for cassettes)*
bit 13 - write-protected or write-locked
bit 14 - odd character (always set to zero for cassettes)
bit 15 - unit ready.

For more information about each of these status bits, see "Nova Cassette" or "Magnetic Tape" in How To Use the Nova Computers.

Upon a read, write, space forward, or space backward command, if no system error is detected, AC1 contains the number of words written (or read) or the number of records spaced. A word or record count is returned in AC1 upon a premature end of file.

Upon detection of a system error, the error return is taken and AC2 is set to contain one of the following:

AC2	Mnemonic	Meaning
0	ERFNO	Illegal channel number.
3	ERICD	Illegal command for device (i.e., improper open).
15	ERFOP	Attempt to reference an unopened file.
30	ERFIL	File read error.

If no system error is detected but a hardware error occurs (i.e., bit 0 of the status word is set), the error return is taken and AC2 is set to the transport status word. If no system error and no hardware error occurs, the normal return is taken and the transport status word is returned in AC2.

The following table summarizes the return taken and contents of AC1 and AC2 with different MTDIO command selections.

* Bits used for maintenance only.

Free Format Tape I/O (.MTDIO) (Continued)

COMMAND	RETURN	AC1	AC2
Any MTDIO command with a system error detected	Error	Same as input	System error code
Rewind (TSW bit 0 = 0)	Normal	Original input is lost	Transport status word (TSW)
Rewind (TSW bit 0 = 1)	Error		
Read Status (TSW bit 0 = 0)	Normal	Original input is lost	TSW (status bit 0 is 0)
Read Status (TSW bit 0 = 1)	Error	Original input is lost	TSW (status bit 0 is 1)
Read, Write, Space Forward, Space Backward; bit 0 in TSW is set to 0	Normal	Word or Record count	TSW
Read, Write, Space Forward, Space Backward; bit 0 in TSW is set to 1	Error (only after 10 retries in read/write)		
Write end-of-file (TSW bit 0 = 1)	Error	Original input is lost	TSW

The system will perform 10 read retries before taking the error return. For write errors, the following sequence will be performed 10 times before taking the error return: backspace and write 9 times, then backspace, erase a 2-1/2" length of tape, and write. Thus the system will perform 100 write retries before signaling an error.

TELETYPEWRITER AND VIDEO DISPLAY COMMANDS

Transfer of single ASCII characters between $TTI/$TTO and AC0 is handled by the system commands .GCHAR and .PCHAR . No channel is required for the transfers, and $TTI/$TTO is always available without requiring a prior .OPEN command.

An additional system call exists which allows a task to be readied upon detection of a user-defined character input via a teletype or display keyboard.

Get a Character (.GCHAR)

This command returns a character input via $TTI to AC0. The character is right adjusted in AC0 with bits 0-8 cleared. No channel is required; the $TTI is always used as input for this command. The format of the .GCHAR command is:

```
.SYSTM
.GCHAR
error return
normal return
```

AC0, bits 9-15 - ASCII input character

No error return is possible from this command.

Put a Character (.PCHAR)

This command transmits a character in AC0, right adjusted, to $TTO. No channel is required, and the $TTO is always used with this command. The format of the .PCHAR command is:

AC0, bits 9-15 - ASCII output character

```
.SYSTM
.PCHAR
error return
normal return
```

No error return is possible from this command.

Wait for a Keyboard Character (.WCHAR)

This command suspends the caller until either a specified character is typed onto any console keyboard or the call is reissued by another task to terminate this keyboard wait. Only one task may be suspended for a keyboard character wait at any one moment.

The required input to this call is either the keyboard character which will ready the task or -1, terminating the keyboard character wait and readying the suspended task. These parameters are input via AC0. Since the calling task may be readied by either the transmission of a specified keyboard character or by the keyboard wait call being terminated, an appropriate code is returned in AC1 when the normal return is taken. This code will be either the device code of the console keyboard which issued the requested wait character, or -1; -1 indicates that the previous wait was terminated.

The format of this call is:

AC0 - Wait character in right byte, or -1 to terminate another task's wait request.

```
.SYSTM
.WCHAR
error return
normal return
```

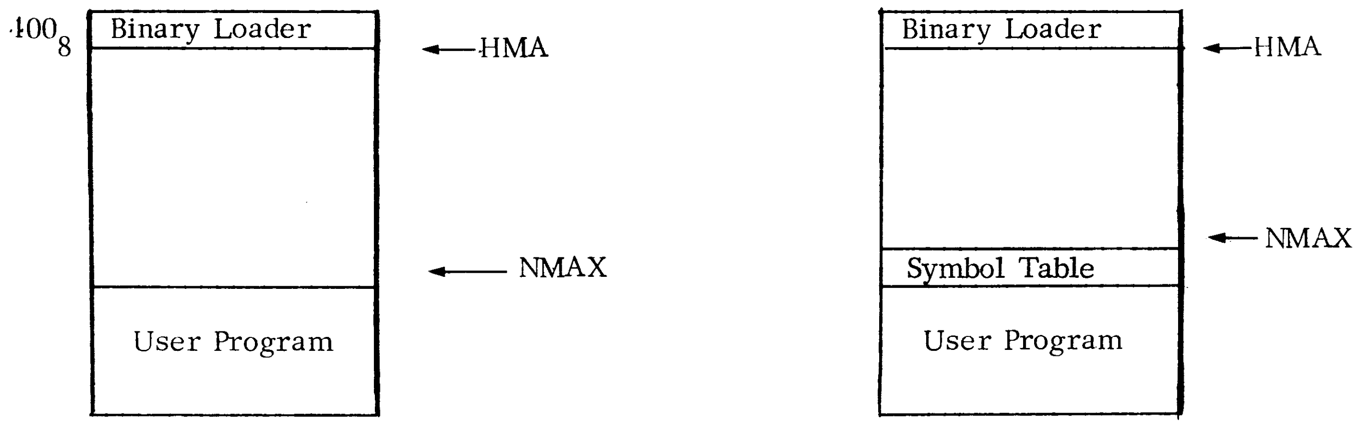
AC1 - Device code of the keyboard transmitting the wait character, or -1. Minus 1 indicates that the previous wait request was terminated.

The error return is taken if a second task tries to suspend itself for a keyboard character while another task is still suspended for a wait character. In this case, AC2 is set to the following:

AC2	Mnemonic	Meaning
47	ERSIM	A previous wait-character request is outstanding.

MEMORY SIZE COMMANDS

RTOS provides a pair of system commands used to determine the first location available above the loaded user program (NMAX), the highest memory address available in user address space (HMA), and a means to increase or decrease the address space allocated to the user program by varying NMAX. NMAX is a pointer contained in the User Status Table at displacement USTNM; this table is discussed in Chapter 5, SYSTEM ORGANIZATION. (Note that the /S switch must not be used in the RLDR command.)



Memory Maps with and without Symbol Table

Determine Available Memory (.MEM)

This command returns the current value of NMAX and the value of HMA. HMA represents the location immediately below the bottom of the binary loader (or the top of the symbol table). A SUB 1, 0 instruction after the.MEM call determines the additional amount of memory available to the user program by putting its value in AC0.
The format of the .MEM command is:

```
.SYSTM
.MEM
error return
normal return
```

AC1 - NMAX.
AC0 - HMA.

There are no error returns from this command.

Allocate Memory by Changing NMAX (.MEMI)

This command permits the user to increase or decrease the value of NMAX. The increment or decrement is passed (in two's complement) in AC0. This command causes the value of NMAX to be updated in the User Status Table, and the new value of NMAX is returned in AC1.

Allocate Memory by Changing NMAX (.MEMI) (Continued)

NMAX will not be changed if the new value would be equal to or higher than the lowest address of the binary loader. No check is made as to whether or not the user decreases NMAX below its original value (as determined at relocatable load time) nor, if a symbol table resided in upper memory, whether NMAX is increased beyond the bottom of the symbol table.

The format of the .MEMI command is:

AC0 - NMAX increment or decrement:

```
.SYSTM
.MEMI
error return
normal return
```

AC1 - New NMAX.

There is one error which may result from a .MEMI command:

AC2	Mnemonic	Meaning
26	ERMEM	Attempt to overwrite the binary loader.

SYSTEM RETURN COMMANDS

These calls return control to their error returns unconditionally.

System Return (.RTN)

This command stops all tasks (except the caller), closes all channels, and returns control unconditionally to the error return; no normal return is reserved. The format of the .RTN command is:

```
.SYSTM
.RTN
error return
```

The following error code is returned:

AC2	Mnemonic	Meaning
23	ERRTN	Attempt to restore a nonexistent image (as in RDOS).

System Error Return (.ERTN)

This command stops all tasks (except the caller), closes all channels, and returns control unconditionally to the error return; no normal return is reserved. The format of the .ERTN command is as follows:

```
.SYSTM
.ERTN
error return
```

The following error code is returned in AC2:

AC2	Mnemonic	Meaning
23	ERRTN	Attempt to restore a nonexistent image (as in RDOS).

CLOCK AND CALENDAR COMMANDS

RTOS clock and calendar commands may be used in any system which includes a real time clock. RTOS clock and calendar commands permit ready tasks to be suspended for a period of time, permit the system real time clock to be examined, and allow the creation of a user clock to specify a recurring interruption of multitask activity.

Delay the Execution of a Task (.DELAY)

This command suspends the caller for a specifiable number of pulses of the system real time clock. Thus this command permits the creation of a time slicing facility within RTOS. The system real time clock frequency is set when a system is generated, and no system call can alter this frequency.

The format of the .DELAY command is:

AC1 - Number of RTC pulses in the delay period.

```
.SYSTM
.DELAY
error return
normal return
```

Contents of AC1 are lost upon return. One possible error may occur:

Delay the Execution of a Task (.DELAY) (Continued)

AC2	Mnemonic	Meaning
2	ERICM	Illegal system command (i.e., no RTC in system).

Get Today's Date (.GDAY)

This command requests the system to return the number of the current day, month, and year. The day is returned in AC0, the month in AC1, and the year in AC2. The The format of the .GDAY command is:

```
.SYSTM
.GDAY
error return
normal return
```

AC0 - Day of the month.
AC1 - Month of the year.
AC2 - Year.

The error return is never taken.

Set Today's Date (.SDAY)

This command permits the system calendar to be set to a specific date. The system will increment the date when the time of day passes 23 hours, 59 minutes, and 59 seconds. The caller passes the number of the day within the month in AC0, the number of the month in AC1 (January is month number 1), and the current year in AC2. The format of the .SDAY command is:

AC0 - Day of the month.
AC1 - Month of the year.
AC2 - Year.

```
.SYSTM
.SDAY
error return
normal return
```

One possible error return is:

AC2	Mnemonic	Meaning
41	ERTIM	Illegal day, month or year.

Get the Time of Day (.GTOD)

This command requests the system to pass the current time in hours, minutes and seconds to the caller. The format of the .GTOD command is:

```
.SYSTM
.GTOD
error return
normal return
```

AC0 - Second.
AC1 - Minute.
AC2 - Hour (using a 24-hour clock).

No error return is possible.

Set the Time of Day (.STOD)

This command permits the caller to set the system clock to a specific hour, minute, and second. The format of the .STOD command is:

AC0 - Second.
AC1 - Minute.
AC2 - Hour (using a 24-hour clock).

```
.SYSTM
.STOD
error return
normal return
```

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

AC2	Mnemonic	Meaning
41	ERTIM	Illegal time of day.

Examine the System Real Time Clock (.GHRZ)

This system call permits the caller to examine the frequency of the real time clock (the frequency was set when the RTOS system was generated). One of five frequency codes is returned in AC0, as described below. The format of the .GHRZ command is:

Examine the System Real Time Clock (.GHRZ) (Continued)

```
.SYSTM
.GHRZ
error return
normal return
```

AC0: 0 - there is no real time clock in the system.
1 - the frequency is 10HZ.
2 - the frequency is 100HZ.
3 - the frequency is 1000HZ.
4 - line frequency is 60HZ.
5 - line frequency is 50HZ.

The error return is never taken.

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 outside the task environment. No task calls (other than .UCEX and .IXMT) may be issued from this interrupt routine, since multitask activity is in suspension. Only one user clock may be defined at any one moment.

The format of the .DUCLK command is:

AC0 - Number of RTC pulses during each user clock interval.
AC1 - Address of user interrupt routine.

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

Only one error condition is possible:

AC2	Mnemonic	Meaning
45	ERIBS	A user clock already exists.

Exit from a User Clock Routine (.UCEX)

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, AC3 must be loaded with the return address that it contained upon entry to the routine, and task call .UCEX must be issued.

The format of this call is:

AC3 - Return address upon entry to 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. 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 is never taken.

****

CHAPTER 3

TASK CALLS

The following is a list of all RTOS task call mnemonics:

.ABORT	Terminate a task immediately.
.AKILL	Kill all tasks of a specified priority.
.ARDY	Ready all tasks of a specified priority.
.ASUSP	Suspend all tasks of a specified priority.
.IDST	Get a task's status by I.D. number.
.IXMT	Transmit an interrupt message.
.KILL	Kill the calling task.
.PRI	Change the calling task's priority.
.REC	Receive a task message.
.SUSP	Suspend the calling task.
.SMSK	Modify the current interrupt mask.
.TASK	Create a task.
.TIDK	Kill a task specified by I.D. number.
.TIDP	Change the priority of a task specified by I.D. number.
.TIDR	Ready a task specified by I.D. number.
.TIDS	Suspend a task specified by I.D. number.
.UCEX	Exit from a user clock routine.
.UIEX	Exit from a user interrupt routine.
.UPEX	Exit from a user power fail routine.
.XMT	Transmit a task message.
.XMTW	Transmit a task message and wait for its receipt.

All RTOS task calls except .SMSK, .UCEX, .UIEX and .UPEX are described in this chapter in alphabetical order. .UCEX is found in Chapter 2, and the remaining task calls are described in Chapter 4.

Abort a Task (.ABORT)

The .ABORT task call causes a specified task to be readied immediately and to execute the equivalent of a .KILL task call as soon as it gains control of the CPU. The exact time of completion of the .KILL is dependent on the priority of the aborted task relative to other ready tasks. For example, a task attempting to perform a write sequential of 500 bytes might be aborted after writing any number of bytes. The task which is to be aborted is specified by I.D. number. Thus the caller may abort either itself or some other ready or suspended task.

Abort a Task (.ABORT) (Continued)

Outstanding operations performed by the task, like waiting for a message transmission/reception (.XMTW/.REC), are terminated. Likewise, all system calls are aborted with the exception of calls performing QTY or MCA I/O. QTY and MCA I/O can be aborted by closing their channel(s) with a .CLOSE or .RESET system command.

The format of this call is as follows:

AC1 - I.D. of the task to be aborted.

```
.ABORT
error return
normal return
```

The contents of AC0 is lost upon return.

The error return is taken under one of two conditions:

AC2	Mnemonic	Meaning
61	ERTID	An I.D. of zero was specified, or no such task I.D. was found.
110	ERABT	The specified task was performing QTY or MCA I/O.

Kill All Tasks of a Specified Priority (.AKILL)

This command deletes all tasks with a priority given in AC0. The calling task itself may be deleted by this command. All tasks that are killed have their TCB's placed in the free element TCB chain. If an attempt is made to kill a task which is suspended due to an outstanding .SYSTM call, that task will be killed at the completion of the .SYSTM call. Tasks suspended by .XMTW, .REC, .TIDS, or .SUSP will also be killed.

The format of this call is:

AC0 - Task priority.

```
.AKILL
normal return
```

There is no error from this call. If no tasks exist with the priority specified in AC0, no action is taken. If all tasks become deleted, the effect is to close all the channels and to idle the system, since control returns to the task scheduler.

Ready All Tasks of a Specified Priority (.ARDY)

This call readies all tasks which were previously suspended by .ASUSP (.SUSP or .TIDS) whose priority is specified in AC0. That is, bit 1 in word TPRST of each TCB (see Chapter 5) that was set by a previous call to .ASUSP, .SUSP, or .TIDS is now reset. Tasks suspended for other reasons too (e.g., outstanding system calls) will only be readied when the task is fully ready, i.e., bits 0 and 2 are also reset. The format of this call is:

AC0 - Task priority.

```
.ARDY
normal return
```

There is no error return from this call. If there are no tasks with the priority specified in AC0, no action is taken and control goes to the normal return.

Suspend All Tasks of a Given Priority (.ASUSP)

This command suspends all tasks with the priority given in AC0. The calling task itself may be suspended by this call. All tasks suspended by .ASUSP (including those additionally suspended for other reasons such as an outstanding system call or waiting for a task message transmission) will remain suspended until readied by an .ARDY or .TIDR command.

The format of this call is:

AC0 - Task priority.

```
.ASUSP
normal return
```

There is no error return from this call. If no tasks exist with the given priority, no action is taken and control goes to the normal return.

Get a Task's Status (.IDST)

This command obtains a code describing a task's status. The task whose status is to be obtained is specified by inputting its identification number in AC1. The format of this call is:

AC1 - Task I. D. number

```
.IDST
normal return
```

Get a Task's Status (.IDST) (Continued)

The code describing the task's status is returned in AC0:

0 - Ready.
1 - Suspended by a .SYSTM call.
2 - Suspended by a .SUSP, .ASUSP, or .TIDS.
3 - Suspended by .XMTW or .REC.
4 - Not used.
5 - Suspended by .ASUSP, .SUSP, or .TIDS and by .SYSTM.
6 - Suspended by .XMTW or .REC and by .SUSP, .ASUSP, or .TIDS.
7 - Not used.
10- No task exists with the specified I.D. number.

There is no error return from this call.

Transmit a Message from a User Interrupt Service Routine (.IXMT)

Whenever a device requiring special user service generates an interrupt request, (see Chapter 4), the entire task environment becomes frozen until servicing of the special user interrupt is completed. All tasks will resume their former states when the environment is restarted unless the user transmits a message to one of them by means of the .IXMT call from the interrupt service routine. In the latter case, rescheduling occurs when the task environment is restarted.

If the task for which the non-zero message is intended has issued a .REC for the message, the task state is changed from suspended to ready even though task activity is in suspension. If more than one task is awaiting a message at this location, only one will receive the message and be readied. Contents of all accumulators are destroyed upon return from .IXMT, so the user is cautioned to restore AC2 and AC3 before attempting an exit from the service routine.

As with .XMT (described later in this chapter), .IXMT causes a non-zero message to be deposited in a location. The contents of this location must be zero at the time .IXMT is invoked, or else the location will be deemed to be already in use.

The format of this call is:

AC0 - Message location.
AC1 - Message.

```
.IXMT
error return
normal return
```

Transmit a Message from a User Interrupt Service Routine (.IXMT) (Continued)

The error return is taken if the message address is already in use (i.e., if its contents are non-zero).

AC2	Mnemonic	Meaning
43	ERXMT	Message location is in use.

Delete a Calling Task (.KILL)

This command deletes the calling task's TCB from the ready queue and places it in the free element TCB chain. The calling task is the only task that may be deleted via this command.

The format of this call is:

```
        .KILL
```

There is no return from this call. Control goes to the task scheduler.

Change the Priority of a Task (.PRI)

This command changes the priority of the calling task. The calling task will be assigned the lowest position in the new priority class. That is, equal priority tasks receive control on a round-robin basis, and this task will be the last task in this priority class to be allocated CPU control by the scheduler.

The format of this command is:

```
    AC0 - New task priority.

        .PRI
        normal return
```

There is no error return from this command. If a priority greater than 255_8 is requested, only the value in bits 8 through 15 will be accepted.

Receive a Message (.REC)

This command returns a message in AC1 that another task or interrupt service routine has posted by means of a transmit command, and restores the contents of the message location to all zeroes. Only one task at a time can receive a message from a given location.

Receive a Message (.REC) (Continued)

If the transmitter has not yet posted a message for the receiving task, the receiver becomes suspended until the message is issued. If the message has already been issued and if the task has not also been suspended by some other event, control returns to the task scheduler.

The format of this command is:

AC0 - Message address.

.REC
normal return.

AC1 - Message.

There is no error return from this command.

Suspend a Task (.SUSP)

This command places the calling task in the suspended state by setting bit 1 of the task's status and priority word. The format of this call is:

.SUSP
normal return

There is no error return. The suspended task remains suspended until it is readied by an .ARDY or .TIDR command.

Create a Task (.TASK)

This command creates a new task in the user environment, and assigns to it a TCB from the TCB pool allocated during system generation. This command creates a task at a specified priority and assigns a unique identification number (I. D.) to the task, if desired. When the user program is started, only one task (the default task) exists. Thus this command is used to start up a multitask environment.

The new task can be assigned any priority from 0 through 255 decimal, and any task I. D. in the same range. If the priority input to this command is 0, the priority of the caller will be assigned to the new task. More than one task with an I. D. of zero can exist. This call will pass to the new task the contents of the caller's AC2; thus this accumulator can be used for relaying an initial one-word message to the newly created task.

Create a Task (.TASK) (Continued)

The format of this command is:

ACO, left byte - New task's I.D.
ACO, right byte - New task's priority.
AC1 - New task's starting address.
AC2 - Caller's AC2 passed to the new task.

```
.TASK
error return
normal return
```

If the error return is taken, AC2 will contain one of the following error codes:

AC2	Mnemonic	Meaning
42	ERNOT	No TCB's available.
61	ERTID	Same I.D. number (except 0) already assigned.

Kill a Task Specified by I.D. Number (.TIDK)

This call kills only that task whose identification number is specified in AC1. The format of this command is:

AC1 - I.D. of task to be killed.

```
.TIDK
error return
normal return
```

If the error return is taken, AC2 will contain the following error code:

AC2	Mnemonic	Meaning
61	ERTID	No task exists with this I.D.

Change the Priority of a Task Specified by I.D. Number (.TIDP)

This command changes the priority of that task whose identification is specified by the contents of AC1. The new priority to be assigned to the task is given in ACO, bits 8 to 15. Thus the format of this task call is as follows:

ACO, right byte - New priority.
AC1 - I.D. of task whose priority is to be changed.

Change the Priority of a Task Specified by I.D. Number (.TIDP) (Continued)

```
.TIDP
error return
normal return
```

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

AC2	Mnemonic	Meaning
61	ERTID	No task exists with this I.D.

Ready a Task Specified by I.D. Number (.TIDR)

This command readies only that task whose identification number is input in AC1. That is, this command resets bit 1 in TPRST of this task's TCB which was set by a previous call to .ASUSP, .SUSP, or .TIDS. The format of this call is:

AC1 - I.D. of task to be readied.

```
.TIDR
error return
normal return
```

If the error return is taken, AC2 will contain the following error code:

AC2	Mnemonic	Meaning
61	ERTID	No task exists with this I.D.

Suspend a Task Specified by I.D. Number (.TIDS)

This command suspends only that task whose identification number is input in AC1. That is, this call sets bit 1 in word TPRST of the specified task's TCB. The format of this command is:

```
.TIDS
error return
normal return
```

If the error return is taken, AC2 will contain the following error code:

AC2	Mnemonic	Meaning
61	ERTID	No task exists with this I.D.

Transmit a Message (.XMT), and Wait (.XMTW)

These two calls permit the sending of a one-word non-zero message by one task to an empty (all-zero) message location for another task. The difference between these commands is .XMT simply causes the message to be deposited, while .XMTW deposits the message and suspends the caller until the message is received. .XMTW will not cause the caller to be suspended if a .REC has already been issued for this message.

The format of this call is:

AC0 - Message address.
AC1 - Message.

```
.XMT (or .XMTW)
error return
normal return
```

The error return is taken if the message address is already in use (i.e., if the contents are non-zero). AC2 will then contain the following error code:

AC2	Mnemonic	Meaning
43	ERXMT	The message address is already in use.

*****

CHAPTER 4

USER INTERRUPTS AND POWER FAIL / AUTO RESTART PROCEDURES

SERVICING USER INTERRUPTS

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 describes the special considerations applying to special, high priority user interrupt devices.

Upon detection of an interrupt request, the system will be dispatched through the device interrupt vector table, .ITBL . In this table are pointers to Device Control Tables (DCTs) for devices established at RTOSGEN time, whether system or user devices. Procedures for writing a device driver for insertion in the system at RTOSGEN time are given in DGC application note 017-000006, User Device Driver Implementation in the Real Time Operating System.

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 the state save area (an 8-word area).
1	DCTMS	Interrupt service mask.
2	DCTIS	Interrupt service routine address.

DCTSV is a pointer to an eight-word state variable save area used by the system to store the PC, accumulators, carry, etc. 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.

SERVICING USER INTERRUPTS (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. (Caution: RDOS 03 does not place the DCT address in AC2.) Exit is accomplished by issuing task call .UCEX; this call may be issued in both single and multitask environments.

All multitask environment activity ceases at the moment 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.

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 a user device interrupt, AC3 will contain the return address upon entry to the user routine and AC2 will contain the DCT address. To return from the user

Exit from a User Interrupt Routine (.UIEX) (Continued)

interrupt routine, AC3 and AC2 must be restored to the values they contained upon entry to the routine, and task call.UIEX must be issued. (Caution: RDOS 03 will execute this call correctly without the DCT address in AC2.)

The format of this call is:

AC2 - DCT address.
AC3 - Return address upon entry to routine.

.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. Rescheduling occurs only if a task state change occurred.

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. This is done by task call .SMSK, whose format is as follows:

AC0 - New value to be ORed with old mask.
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. (Caution: RDOS 03 will execute this call correctly without the DCT address in AC2.)

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).

POWER FAIL/AUTO RESTART PROCEDURES

RTOS provides software support for the power fail/automatic restart option. Upon detection of a loss of power, 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:

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 FAIL/AUTO RESTART PROCEDURES (Continued)

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 . To use .IDEF for this purpose, AC0 must be input with 77_8 and AC1 must input the starting address of the user power up routine. Exit from this power up routine is accomplished by task call .UPEX, described below.

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 service routine, AC3 must be loaded with this return address and task call .UPEX must be issued.

The format of this call is:

AC3 - Return address upon entry to the routine.

.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.

HIGH PRIORITY USER INTERRUPT SERVICE

As described in Chapter 6 and in Appendix B, 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 if used by this routine.
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 this routine if they are altered in this routine, and these state variables must be restored before leaving the routine. AC3 is saved by the system in location .SAC 3 (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
```

*****

CHAPTER 5

MULTIPLE PROCESSOR SYSTEMS

MULTIPLE PROCESSOR PROGRAMMING

All features of RTOS discussed in previous chapters are available to systems with two or more processors. Additional hardware support extended to multiple processor systems by RTOS is in the form of a Multiprocessor Communications Adapter (MCA), option 4038.

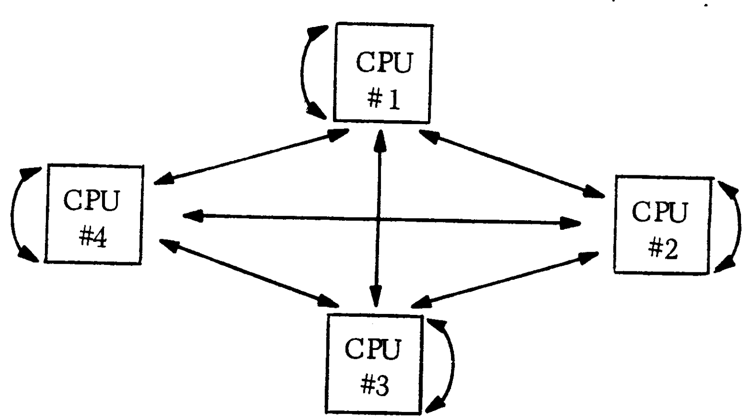
Data Transmissions

The type 4038 Multiprocessor Communications Adapter receiver/transmitter (MCAR/MCAT) makes it possible for full duplex interprogram communications to take place, in blocks of up to 8192 bytes, via the data channel. MCA support can be given to a single processor in a multiple processor system so that an RTOS program running in one processor can communicate with an RTOS program running in another processor or with either a foreground or background program running under RDOS in another processor. Each CPU may communicate with any of up to 14 other CPUs.

Each MCA line corresponds to a file name of the following form:

MCAT:mm   or   MCAR:nn

where mm represents a receiver unit number from 1 - 15 inclusive and where nn represents a transmitter unit number in the range 0 - 15. Thus a four-CPU, RTOS-only system would be logically configured with 10 separate lines if every possible communications link were to be used.



Data Transmissions (Continued)

If CPU #1 wanted to read from CPU #3, each unit would have to issue the following sets of instructions:

Unit #1	Unit #3
.OPEN n ;OPEN MCAR:3	.OPEN n ;OPEN MCAT:1
.RDS n	.WRS n

Note that units #1 and #3 are operating under distinct operating systems. Thus, in the illustration on the previous page there is no correspondence between channel n for unit #1 and channel n for unit #3.

If, in a receive request, unit number zero is specified to be the transmitter (e.g., MCAR:0), the receive request becomes generalized to indicate that any unit may transmit to this receiver. Thus if unit #1 had three outstanding receive requests, MCAR:4, MCAR:3, and MCAR:0, it could receive concurrent transmissions from three sources: a transmission unit from unit #4, a transmission from unit #3, and a message from any other unit that chose to transmit to it.

A timeout can occur only in the MCA transmitter; the receiver can wait indefinitely. The timeout period ranges from approximately 10 milliseconds to approximately 655 seconds. The default timeout, specified at RTOSGEN time, may be superseded by specifying a different timeout both when the transmitter is opened and when the write sequential is issued.

Get the Current CPU's MCA Number (.GMCA)

It is possible to get the MCA unit number in the CPU which is currently executing a user program. To obtain the unit number, system call .GMCA is issued. The format of this command is as follows:

```
.SYSTM
.GMCA
error return
normal return
```

If the normal return is taken, the MCA unit number is returned in AC1. If the error return is taken, the following error code will be input in AC2:

AC2	Mnemonic	Meaning
36	ERDNM	Device not in system (i.e., no MCA was specified at SYSGEN time in this operating system).

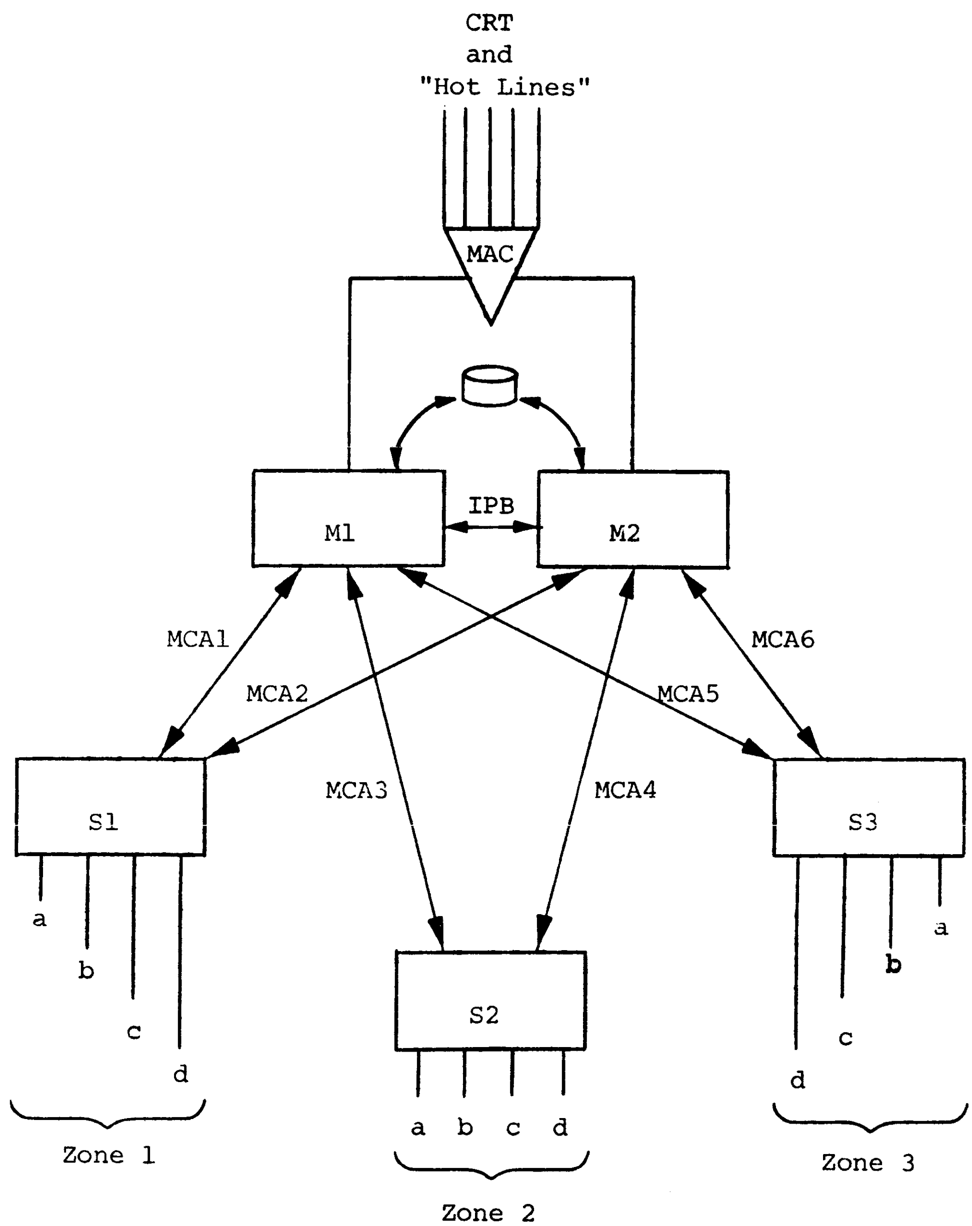
MULTIPROCESSOR SYSTEM ILLUSTRATION

Consider the following application for a multiprocessor system. A large laboratory complex needs an automated system to control the environmental conditions within the complex, to keep track of the number of personnel at locations throughout the complex, and to monitor the complex for alarm conditions (alerting key personnel when a condition cannot be corrected by the system itself). Moreover, the system must be fail-safe, and can allow down-time for no longer than a few seconds.

Such a system might well be configured along the lines suggested by the illustration at the end of this section. Two master CPUs, a Dual Nova system running under RDOS, are connected in redundant fashion so that if one fails, the other detects this failure and gains immediate control. The masters access a common data base which contains, among other information, alarm messages and destinations to which they should be sent in the event of an alert. Also contained in this file space is a log of the current master's activity, so that if it should experience failure, the alternate master CPU would have a record of recent events. An IPB connects the masters so that they may access common disk files and so that one may act as a watchdog on the other's behavior.

There are three slave CPUs, each of which monitors and controls various parameters within one zone of the building complex. Each slave is capable of monitoring and adjusting both the humidity and the temperature of each zone. Additionally, each slave keeps track of the positions of personnel within each zone. Finally, each slave monitors its zone for the occurrence of alarm conditions, and it can perform limited response to emergencies, e.g., it can activate a sprinkler system if fire is detected. The functions performed by each slave are relatively simple and could be performed by Nova 2's running under RTOS.

Each slave has a high speed data channel line, MCA1 through MCA6, to each of the two master computers, so that continuous status reports can be generated by them for transmission to CRT monitors via the Multiline Asynchronous Controller, MAC. The MAC also has direct "hot lines" to key personnel (security guards, fire station personnel, etc.) who should be alerted in the event of an emergency.



a - temperature sensing and control
b - census taking
c - humidity sensing and control
d - intrusion/fire/smoke alarm and control

M1 - Master CPU
M2 - Back-up Master CPU
S1,S2,S3 - Slave CPUs

MULTIPROCESSOR SYSTEM ILLUSTRATION

CHAPTER 6

SYSTEM ORGANIZATION

This chapter describes the collection of tables and fixed locations which are used by programs running under the real time operating system. As described in Appendix C, each user program supported by RTOS is loaded with an RTOS module generated by the RTOSGEN program, followed by user-supplied RTOS drivers and the RTOS system library. Loading itself is accomplished by means of the stand-alone or SOS extended relocatable loader or the RDOS loader. At the completion of loading, pointers and tables are found as illustrated in the RTOS Core Map shown on the following page.

RTOS Page Zero

Locations 0 through 15 are reserved for use by the system and cannot be taken by user programs. Location 0 receives the current PC upon each interrupt, and location 1 contains a pointer to the system interrupt dispatch logic. This is usually either .HINT, the high priority interrupt dispatcher, or .INTP, the regular interrupt processor. These modules are produced by RTOSGEN.

When a task in RTOS is in the executing state, the CPU is said to be in User Mode. Otherwise (as when RTOS is engaged in some system function like task scheduling) the CPU is said to be in System Mode. Location 5, .SYS. , is a flag which is set to zero when the system is in User Mode, and is set to non-zero in System Mode. Interrupts in a real-time environment occur randomly. Since it is inappropriate for certain interrupt-triggered system functions to be reentrant (e.g., task scheduling) .SYS. serves as an interlock to prevent undesired entries into these routines.

Location 2, BEGIN, contains a pointer to the RTOS initialization program. This program is used to initialize stacks and clear switches in RTOS. The first section of this routine zeroes system switches and intializes all device handlers. The starting task is initially set to a priority of zero, the hardware interrupt mask is made zero, the system is set in User Mode (.SYS. = 0), the real time clock is started, and TCB chain pointers are initialized. The wait character logic, activated by .WCHAR, is also reset.

The last operations performed by the initialization program are to enable the interrupt facility by the INTEN instruction, and to jump to the start of the user program (which must have been specified in an .END statement).

RTOS Core Map

Page Zero	Contents
1	Interrupt service routine address.
2 BEGIN	Starting address for system initializer.
3 CTCB	Address of currently executing task's TCB.
4 SCHED	Entry point to task scheduler.
5 .SYS.	System mode indicator.
6 RLOC	Page zero temporary.
7 IOEND	Entry point to I/O end processor.
10 .CMSK	Current system interrupt mask.
11 DISMISS	Interrupt dismissal address.
12 USTP	Reserved for RDOS/RTOS compatibility; set to 400.
13 TLINK	Entry point to routine linking ready TCB to TCB chain.
14 RSCHD	Entry point to reschedule the system.
15 .TSAVE	Address of TCB state save routine.
16 USP	User Stack Pointer (USP).
17	Entry point to system call processor.
20	First page zero location available to user program.

Page One	
400 UST	Start of User Status Table.
426	End of User Status Table.
440	Beginning of NREL area into which RTOS and user modules may be loaded. The following global symbols may be present in the RTOS modules:
.TCBP	Start of TCB pool (total length = no. of TCBs *12_{10}).
.UFPT	Start of User File Pointers Table, UFPT (total length = no. of channels *2).
.DTBL	Start of fixed head disk table, DTBL (total length = 1 + 5*number of disk files defined at RTOSGEN time).
.PTBL	Start of moving head disk table, PTBL (total length = 1 + 6*number of disk files defined at RTOSGEN time).
.QTBL	Start of 4060 asynchronous multiplexor (QTY) table, QTBL (total length = 10_{10}*number of QTY lines).
.MCTB	Start of Multiprocessor Communication Adapter line table, (total length = 7 + number of adapters in the network *14_{10}).
.HINT	Start of high priority dispatcher (length = 11_{10}+ 3*number of high priority interrupt devices).
.ITBL	Start of interrupt table, ITBL (64_{10} words long).
.ETBL	End (last location) of ITBL (user power fail handler, if present).
.CHTB	Start of device name table, CHTB (length = 1+4*number of devices in system).
	User program with user device drivers, if any.
	RTOS device drivers.
	RTOS system library modules.

RTOS Page Zero (Continued)

Location 3, CTCB, contains the TCB address of the currently executing task. Location 4, SCHED, points to the main entry in the task scheduler. This entry causes the highest priority ready task (if any) to be executed. Location 14, RSCHD, points to an alternate entry in the scheduler. This entry links the currently executing task's TCB to the ready chain, then transfers control to the entry pointed to by SCHED .

Location 6, RLOC, is a temporary location used by RTOS; location 7, IOEND, is the entry point to the I/O end processor module. This routine is used to handle the end of I/O operations for a device handler at either the interrupt or system level. If while in this routine it is determined that another request is pending, the routine will cause the next I/O operation on the device to be started. .CMSK, location 10, contains the current system interrupt mask.

DISMISS, location 11, contains the interrupt dismissal routine address. USTP, location 12, is reserved for use by RTOS FORTRAN, and points to the start of the User Status Table (location 400). The value "0400" is also equivalent to a "JMP . " instruction, so RTOS transfers control to this location whenever a system panic is detected. System panics result from unknown system errors and are generally unrecoverable; restarting at location 376 will restart the system provided the RTOS program has not been damaged.

Location 13, TLINK, points to a routine which links a ready TCB to the ready chain. This routine is entered by the system whenever a task is raised to the ready state. RSCHD, location 14, contains the entry point to the task scheduler which links the current TCB to the ready chain, then executes the highest priority ready task. Location 15, .TSAVE, contains the address of the TCB state save routine.

Location 16 contains the User Stack Pointer, USP . This location is used by FORTRAN IV and is loaded into AC3 by RTOS on return from system or task calls. AC3 is destroyed whenever any such call is issued, since it is equivalent to a JSR instruction. On return from a call however, AC3 is loaded with the contents of USP. Thus a convenient method of saving AC3 before issuing a call is to first save AC3 in USP. AC3 will then be restored by the system upon returning from the call. Furthermore, since USP is saved in the TCB as part of the state of a task in execution, it may be used as the equivalent of an extra register by tasks without stacks.

User Status Table (UST)

The User Status Table (UST) is found at the start of NREL memory. This table is 27_8 words long, and contains information pertaining to the status of the user program. Unused words within the UST are set to -1. The structure of the UST is as follows:

Displacement	Mnemonic	Contents
0	USTPC	Maintained for RDOS compatibility only.
1	USTZM	ZMAX, the lowest unused ZREL memory location.
2	USTSS	The start of the symbol table.
3	USTES	The end of the symbol table (i.e., its lowest core address).
4	USTNM	NMAX, the lowest unused NREL memory location.
5	USTSA	The user program starting address.
6	USTDA	The address of the debugger.
7	USTHU	The highest address occupied by the user program and RTOS upon completion of loading.
10	USTCS	FORTRAN common area size.
11	USTIT	Maintained for RDOS compatibility only.
12	USTBR	Maintained for RDOS compatibility only.
13	USTCH	Number of tasks (left half), number of channels (right half).
14	USTCT	TCB address of the first user task to execute after RTOS initialization.
15	USTAC	Start of ready TCB queue.
16	USTFC	Start of the free (dormant) TCB chain.
17	USTIN	Start of user NREL program (set by the loader).
20	USTOD	Reserved only for compatibility with RDOS.
21	USTSV	FORTRAN state-save routine.
22	USTSQ	Start of suspended TCB queue.
23	USTXQ	Start of the .XMTW/.REC queue.
24	USTPQ	Start of internal system queue of tasks being serviced (i.e., TCBs returning to base level after I/O).
25	USTOS	Scheduler idle counter (used in checking $PTR timeouts).
26	USTNA	Number of active tasks (TCBs in use).

The last word in the UST is also assigned the mnemonic USTEN.

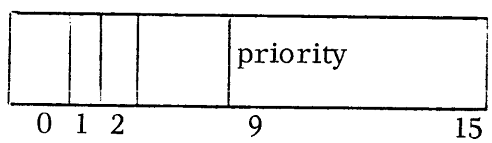
Task Control Block (TCB) Pool

Following the User Status Table is found the TCB pool. This is a collection of TCBs which will be available to the user program. Each TCB is a 14_8-word frame defined as follows:

Displacement	Mnemonic	Contents
0	TPC	Carry in bit 0, PC in bits 1-15.
1	TAC0	AC0
2	TAC1	AC1
3	TAC2	AC2
4	TAC3	AC3
5	TPRST	Task status and priority.
6	TSYS	System call word.
7	TLNK	TCB link word (-1 if last TCB in the queue).
10	TUSP	Task USP
11	TELN	FORTRAN IV variables save area address.
12	TID	Task identifier.
13	TTMP	Used to service .ABORT calls.

Word 5, the task status and priority, contains information describing the state of the task, its priority, and whether it has any outstanding message transmit or receive request:

TPRST:



Bit 0 is set to a 1 if the task is suspended due to an outstanding system call. Bit 1 is set to 1 if the task is suspended due to task calls .SUSP, .ASUSP, or .TIDS. Bit 2 is set to a 1 only if the task is suspended due to outstanding task calls .XMTW or .REC . The task priority is contained in bits 9-15.

Word 6, TSYS, is used by RTOS in executing system calls. Word 7 contains the starting address of the next TCB in the queue (the last TCB in a queue has a link of -1). Word 12, TID, contains the task's 8-bit identifier in bits 9-15.

User File Pointers Table (.UFPT)

Following the TCB pool is a table called the User File Pointers Table (.UFPT). The purpose of this table is to indicate what device or file is open on which RTOS channels. The .UFPT consists of a series of two-word frames, one for each channel defined for the RTOS system at RTOSGEN time. The first frame represents the device (or file) opened on channel zero, etc.

The two-word frame will contain one of two sets of information, depending upon whether it is a single- or multi-file device which is opened on the designated channel. Frames for all devices contain the device DCT address in displacement 0. Displacement 1 for single-file device frames is unspecified. Displacement 1 of multi-file device frames, however, contains a pointer to a frame within a device file table or within a device driver itself. Device file tables are discussed in the following section. Frames within device file tables describe disk files, QTY lines, or MCA units. Frames for cassettes and magnetic tape units point to unit control tables within these devices' drivers.

Device File Tables

One or more device file tables may follow the .UFPT. The following devices have special device tables and have no entries in the standard device name table (.CHTB, to be discussed later):

Device	Table Name
fixed head disk	.DTBL
moving head disk	.PTBL
asynchronous mux	.QTBL
Multiprocessor Communications Adapter	.MCTB

The device file table established for fixed head disks is composed of a series of five-word frames, one for each file defined at RTOSGEN time. Each frame contains the name of the file and its contiguous disk block address boundaries. This approach allows four-, five-, or six-character file names to be given to each disk file. Disk files in RTOS will be comprised of sets of contiguous blocks on disk, fully compatible with contiguous disk files as defined in the real time disk operating system, RDOS. Each frame in .DTBL has the following structure:

Device File Tables (Continued)

	Displacement	Contents
.DTBL frame	0-2	File name, left justified, trailing null bytes.
	3	Starting disk block address.
	4	Ending disk block address.

The device file table for moving head disks (cartridge or pack) is similar to .DTBL, but has one additional entry to describe the drive unit number:

	Displacement	Contents
.PTBL frame	0	Drive unit number.
	1-3	File name, left justified, trailing null bytes.
	4	Starting disk block address.
	5	Ending disk block address.

The 4060 asynchronous multiplexor driver (QTY) device table is named .QTBL. This table consists of a series of ten-word frames with one frame reserved for each QTY line; the first frame corresponds to line number 0, the second frame corresponds to line number 1, etc. Each frame in .QTBL has the following structure:

	Displacement	Contents
.QTBL frame	0	Bit zero set if not opened; line number if open.
	1	Receive byte pointer.
	2	Transmit byte pointer.
	3	Read request TCB address.
	4	Write request TCB address.
	5	Read sequential limit; 0 if read line.
	6	Write sequential limit; -1 if finished writing; 0 if write line.
	7	Character hold for echo during read line.
	10	Line feed insertion flag (non-zero for insertions).
	11	Device characteristics:

DCKEY	echo characters
DCPCK	parity check/generation
DCLAC	line feeds after carriage returns.

The device file table for MCA lines is named .MCTB . This table consists of a series of seven-word frames, each frame reserved for an MCA unit number; each MCA line connects two MCA units (a transmitter and a receiver). The total number of frames equals two times the number of lines defined at RTOSGEN time plus one (for unit number zero reception requests). The first frame corresponds to unit number zero, etc. Each frame in .MCTB has the following structure:

	Displacement	Contents
.MCTB frame	0	List link.
	1	Word count.
	2	Current address.
	3	Device code of the adapter at the other end of the line; code is positioned as in the MCA status words.
	4	Error retry count.
	5	TCB address of task issuing the read or write request; -1 if no task is issuing such a request. This word is set to zero if the line is closed.
	6	Device retry specification input in AC1 to .OPEN .

Line table entries are linked via displacement 0 of each frame. Entries in the chain correspond to MCA units which have MCA transmit or receive requests outstanding.

High Priority Interrupt Table (.HINT)

If any high priority interrupt devices were defined during system generation, one of two high priority interrupt dispatch tables will be loaded. If only the power fail/ auto restart option was selected in the RTOS system, a truncated high priority interrupt dispatcher will be placed after the last device file table; otherwise, the full high priority interrupt dispatcher, .HINT, will be placed there. These dispatch routines are illustrated on the following page. In essence, the operation of .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 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 Table (.HINT) (Continued)

The format of the high priority interrupt dispatcher is as follows:

```
        SKPDZ CPU        ; CHECK FOR POWER FAIL INTERRUPT
        JMP @  A         ; YES. GO TO POWER FAIL INTERRUPT SERVICE.
        STA  3@ B        ; OTHERWISE, SAVE 3
        INTA  3          ; AND GET INTERRUPT DEVICE CODE.
        SKPDZ  RTC       ; WAS IT THE REAL TIME CLOCK?
        JMP @ C          ; YES. GO TO RTC INTERRUPT SERVICE
        SKPDZ device1    ; WAS IT DEVICE 1?
        JMP @ D          ; YES. GO TO DEVICE 1 INTERRUPT SERVICE
           .
           :             ; ETC.
        SKPDZ devicen

        JMP @ N          ; GO TO DEVICE N INTERRUPT SERVICE, BUT IF
        JMP @ .+1        ; NO HIGH PRIORITY DEVICE INTERRUPT
        .INTD            ; GO TO ORDINARY INTERRUPT DISPATCH ROUTINE
B:      .SAC3
A:      PWRIS            ; POWER FAIL INTERRUPT SERVICE ADDRESS
C:      RTCIS            ; RTC INTERRUPT SERVICE ADDRESS
D:      DV1IS            ; DEVICE 1 INTERRUPT SERVICE
           .
           :
N:      DVNIS            ; DEVICE N INTERRUPT SERVICE.
```

The power fail - only interrupt dispatcher looks like the following:

```
        SKPDZ  CPU       ; POWER FAIL INTERRUPT?
        JMP @ .+3        ; YES. GO TO POWER FAIL SERVICE
        JMP @ .+1        ; NO, GO TO ORDINARY INTERRUPT DISPATCHER
        .INTP
        PWRIS            ; POWER FAIL INTERRUPT SERVICE ROUTINE
```

Interrupt Table (.ITBL)

One table which is always loaded is the interrupt table, .ITBL . .ITBL is a 64_{10}-word table which has 64 one-word frames, one for every possible device code. The first entry in the table corresponds to device code zero, the second entry corresponds to device code 1, etc. Table entries corresponding to devices in the system will contain the address of that device's Device Control Table (DCT). Moveover, if the device is a system device, bit zero of this entry will be set to one; if the device is a user device, bit zero will be reset to zero.

Interrupt Table (.ITBL) (Continued)

The last entry in this table, the entry for device code 77_8, is named .ETBL. This entry is reserved for a user power fail/auto restart handler address.

System devices will be initialized by the RTOS initialization program .RTOS. No such initialization is performed for user devices. User device drivers must perform their own initialization.

Standard Device Name Table (.CHTB)

The last RTOS table which may be loaded is the name table, .CHTB. This is a table containing entries for single-file devices like the teletypewriter, paper tape reader/punch, card reader, line printer, and plotter. This table is built at system generation time, and consists of a series of four-word frames.

Device entries are listed in this table in the order that the devices are found in .ITBL (i.e., order is by ascending device code). The first three words of each frame contain the system name for the device, left justified and with trailing nulls. The fourth word in each frame contains the base address of the device's DCT. .CHTB is terminated with a -1.

Thus a typical .CHTB would have the following structure:

$	T
T	I
null	null
TTIDC	
$	T
T	O
null	null
TTODC	
...	...
$	L
P	T
1	null
LPT1DC	
-1	

*****

APPENDIX A

RTOS COMMAND SUMMARY

Call	AC0	AC1	AC2
.ABORT		I.D. of task to be aborted.	
.AKILL[(1)]	Priority of tasks to be killed.		
.SYSTM .APPEND n	Byte pointer to device name.	Device characteristics mask (see .OPEN).	Channel number (if n = 77).
.ARDY[(1)]	Priority of tasks to be readied.		
.ASUSP[(1)]	Priority of tasks to be suspended.		
.SYSTM .CLOSE n			Channel number (if n = 77).
.SYSTM .DELAY		Number of RTC ticks.	
.SYSTM .DUCLK	Number of RTC ticks.	Address of user interrupt routine.	
.SYSTM[(2)] .ERTN			Data word to be placed in AC2.
.SYSTM .GCHAR	bits 9-15: character. bits 0-8: clear.		
.SYSTM .GCHN			(returned) Free channel number

(1) No error return.
(2) No normal return.

Call	AC0	AC1	AC2
.SYSTM .GDAY	(returned) Day.	(returned) Month.	(returned) Year.
.SYSTM .GHRZ	0: no RTC. 1: 10 HZ. 2: 100 HZ. 3: 1000 HZ. 4: 60 HZ (line frequency) 5: 50 HZ (line frequency)		
.SYSTM .GMCA		(returned) Unit number.	
.SYSTM .GTOD	(returned) Second.	(returned) Minute.	(returned) Hour (using a 24-hr clock).
.SYSTM .IDEF	Device code.	DCT.	
.IDST	(returned) 0: ready. 1: suspended by .SYSTM call. 2: suspended by .SUSP, .TIDS, .ASUSP. 3: waiting for .XMTW/.REC . 4: not used. 5: suspended by .SUSP, ASUSP, or .TIDS and .SYSTM call. 6: suspended by .XMTW/.REC and .SUSP, ASUSP, or .TIDS 7: not used. 10: killed.	bits 8-15: task I.D. number.	
.SYSTM .INIT	Byte pointer to device name.	-1: full init. 0: partial init.	
.SYSTM .IRMV	Device code.		
.IXMT	Message address.	Message (non-zero).	

Call	AC0	AC1	AC2
.KILL[1] [2]			
.SYSTM .MEM	HMA	NMAX	
.SYSTM .MEMI	NMAX increment or decrement (2's complement).	(returned) New NMAX (after change).	
.SYSTM .MTDIO n	Core data address, if a data transfer.	bit 0: 1, even parity; 0, odd parity. bits 1-3: 0 read (words) 1 rewind tape. 3 space forward. 4 space backward. 5 write (words). 6 write EOF. 7 read device status word. bits 4-15: word or record count. If 0 on space command, position tape to new file.	Channel number (if n=77). (Status word or system error code if error return; status word if read status normal return.) 1B0: error. 1B1: data late. 1B2: tape rewinding. 1B3: illegal command. 1B4: high density or cassette if 1; low density if 0. 1B5: parity error. 1B6: end of tape. 1B7: end of file. 1B8: tape at load point. 1B9: 9-track or cassette if 1; 7-track if 0. 1B10: bad tape; write failure. 1B11: send clock (0 if cassette) 1B12: first character (0 if cassette). 1B13: write-protected or write-locked 1B14: odd character (0 if cassette). 1B15: unit ready.
.SYSTM .MTOPD n	Byte pointer to tape file specifier.	Characteristic inhibit mask (see .OPEN).	Channel number (if n = 77).
.SYSTM .OPEN n	Byte pointer to file name	1B1: 80-column device 1B2: lower-to-upper case ASCII. 1B3: form feed on open. 1B4: full word device. (continued on next page)	Channel number (if n = 77).

(1) no error return
(2) no normal return

Call	AC0	AC1	AC2
.SYSTM .OPEN n (Continued)		1B6: LF after CR. 1B7: parity check/ generation. 1B8: rubout after tab. 1B9: null after FF. 1B10: keyboard input. 1B11: TTY output device. 1B12: no FF hardware. 1B14: no TAB hardware. 1B15: leader/trailer. ("01" if user-specified MCAT timeout).	
.SYSTM .PCHAR	bits 9-15: character		
.PRI(1)	bits 8-15: new task priority.		
.SYSTM .RDB n	Starting core address to receive data.	Starting disk relative block number.	bits 0-7: number of blocks to be read. (2) bits 8-15: channel number (if n = 77). (2)
.SYSTM .RDL n	Byte pointer to user core area.	Read byte count (including terminator) at end of read.	Channel number (if n = 77).
.SYSTM .RDS n	Byte pointer to core buffer. (even for MCA)	Number of bytes to be read (if EOF detected, partial byte count returned).	Channel number (if n = 77).
.REC(1)	Message address.	Message.	
.SYSTM .RESET			
.SYSTM .RLSE	Byte pointer to device name.		
.SYSTM(3) .RTN			

(1) no error return

(2) if error EREOF, error code in bits 8-15, partial read count in bits 0-7.

(3) no normal return

Call	AC0	AC1	AC2
.SYSTM .RUCLK			
.SYSTM .SDAY	Day.	Month.	Year.
.SMSK	New interrupt mask to be ORed with old mask.		DCT address
.SYSTM .STOD	Second.	Minute.	Hour.
.SUSP(1)			
.TASK	bits 0-7: task I.D. bits 8-15: task priority.	New task entry point address	Message to new task.
.TIDK		bits 8-15: task I.D. number.	
.TIDP		bits 8-15: task I.D. number.	
.TIDR		bits 8-15: task I.D. number.	
.TIDS		bits 8-15: task I.D. number.	
.UCEX(1)(2)(3)		Any non-zero value if rescheduling is to occur.	
.UIEX(1)(2)(3)		Any non-zero value if rescheduling is to occur.	DCT address
.UPEX(1)(2)(3)		Any non-zero value if rescheduling is to occur.	

(1) no error return.
(2) no normal return.
(3) return address is in AC3.

Call	AC0	AC1	AC2
.SYSTM .WCHAR	-1, terminate wait request. bits 9-15: wait character.	Device code of keyboard transmitting the wait character or -1 if wait request terminated.	
.SYSTM .WRB n	Starting core address.	Starting relative block number.	bits 0-7: number of disk blocks. bits 8-18: channel number (if n = 77).
.SYSTM .WRL n	Byte pointer to core buffer.	Write byte count, including terminator, returned at end of write.	Channel number (if n = 77).
.SYSTM .WRS n	Byte pointer to core buffer.	Number of bytes to be written.	right byte: channel number (if n = 77). left byte: # of MCA retries (each retry takes 200 milliseconds).
.XMT	Message address.	Message (non-zero).	
.XMTW	Message address.	Message (non-zero).	

ERROR MESSAGE SUMMARY

Code	Mnemonic	Meaning	Applicable Commands		
0	ERFNO	Illegal channel number.	.APPEND .MTOPD .RDL .WRB	.CLOSE .MTDIO .RDR .WRL	.OPEN .RDB .RDS .WRS
1	ERFNM	Illegal file name.	.OPEN	.MTOPD	.INIT
2	ERICM	Illegal system command.	.RLSE	.INIT	.DELAY
3	ERICD	Illegal command for device.	.RDS .WRL .MTDIO	.WRS .RDB .GMCA	.RDL .WRB
6	EREOF	End of file.	.RDB	.RDL	
12	ERDLE	Attempt to refer to a non-existent file or device.	.OPEN	.MTOPD	
15	ERFOP	Attempt to reference an unopened file or device.	.RDS .WRS .RDB	.WRL .CLOSE .WRB	.RDL .MTDIO
21	ERUFT	Attempt to use a channel which is already in use.	.APPEND .MTOPD	.GCHN	.OPEN
22	ERLLI	Line limit exceeded.	.RDL	.WRL	
23	ERRTN	Attempt to return (.RTN/.ERTN) from current program.	.RTN	.ERTN	
24	ERPAR	Parity on read line.	.RDL		
26	ERMEM	Not enough memory available.	.MEMI	.RDL	.RDS
27	ERSPC	Out of disk space.	.WRB		
30	ERFIL	File read error.	.RDS		.MTDIO
31	ERSEL	Unit improperly selected.	.INIT	.OPEN	.MTOPD

Code	Mnemonic	Meaning	Applicable Commands		
36	ERDNM	Device not in system, or illegal device code.	.INIT .IDEF	.RLSE .GMCA	.IRMV .STMAP
41	ERTIM	Attempt to set illegal time or date.	.STOD	.SDAY	
42	ERNOT	Out of TCBs	.TASK		
43	ERXMT	Message address already in use.	.IXMT	.XMT	.XMTW
45	ERIBS	Device code already in system.	.DUCLK	.IDEF	.INIT
47	ERSIM	Simultaneous read or write attempt on same QTY line.	.RDL .WRS	.RDS .WCHAR	.WRL
60	ERFIU	Attempt to open a busy MCA unit.	.OPEN		
61	ERTID	Task I.D. error.	.TASK .TIDS	.TIDK .TIDP	.TIDR .ABORT
101	ERDTO	Device timeout	.WRS		
103	ERMCA	No complementary MCA request.	.RDS		.WRS
104	ERSRR	Short MCA receive request.	.RDS	.WRL	.WRS
106	ERCLO	Channel closed by another task.	.RDL .WRS	.RDS	.WRL
110	ERABT	Task not abortable.	.ABORT		
113	ERNMC	No MCA receive request.	.WRS		

*****

APPENDIX B

GENERATING AND LOADING AN RTOS SYSTEM

DEFINITION OF TERMS

This appendix details the steps to be followed when creating an RTOS system tailored to a specific device/core configuration and to the channel/task environment of the real time program which will be supported by the system and user program, both in a disk and in a non-disk environment.

System generation is the procedure followed to produce a relocatable binary which will trigger the loading of appropriate device drivers from the RTOS library, allocate tables and control blocks used by the system, and allocate a fixed number of channels and task control blocks. The system generation program, RTOSGEN, produces a relocatable binary with the default name RTOS.RB, by issuing a series of questions about the hardware configuration and the task/channel requirements of the user program.

System loading is the procedure followed to load the system generation relocatable binary, user drivers if any, user relocatable binaries, and the RTOS library. System loading is accomplished by using a relocatable loader.

PREPARATION FOR SYSTEM GENERATION

If the system is configured with either a type 4048 or type 4057 moving head disk pack drive, it is necessary to format the disk pack using the appropriate formatter program before system generation is attempted. In general, all disk packs that are to be used in the system must be formatted prior to their use. Note that it is not necessary to format the disk cartridges used in a type 4047 disk drive. The disk pack formatter programs are stand-alone programs. The appropriate formatter programs and their associated manuals are listed below:

Disk Pack Drive	Formatter Program	Manual
Type 4048	095-000072	096-000039
Type 4057	095-000071	096-000038

SYSTEM GENERATION

The following page contains lists of tapes required to generate and load a system in stand-alone and RDOS environments. To generate an RTOS program under the Stand-alone Operating System (SOS), the RTOSGEN program itself must be configured with appropriate device support before it can be used. Consult the Stand-alone Operating System User's Manual, 093-000062, for the procedures to be followed to configure and load SOS programs.

Stand-Alone Systems:

RTOS TASK MONITOR LIBRARY RTOS1. LB	099-000060
RTOS HANDLER LIBRARY RTOS2. LB	099-000061
RTOS CASSETTE HANDLER LIBRARY (CASDR. LB)	099-000062
RTOS MAGNETIC TAPE HANDLER LIBRARY (MTADR. LB)	099-000063
RTOS FIXED HEAD DISK HANDLER LIBRARY (DSKDR. LB)	099-000064
RTOS MOVING HEAD DISK HANDLER LIBRARY (DKPDR. LB)	099-000065
Stand-alone Extended Relocatable Loader, or	091-000038
SOS Relocatable Loader	089-000120
RTOS SYSTEM GENERATION PROGRAM RTOSGEN	091-000081
RTOSGEN. RB (to produce RTOSGEN under SOS)	089-000163
GSUBR. RB (to produce RTOSGEN under SOS)	089-000164

RDOS Systems:

RDOS/MRDOS DUMP: RTOS SYSGEN PROGRAM RTOSGEN. SV	088-000082
RDOS/MRDOS DUMP: RELOCATABLE LOADER RLDR. SV	088-000049
RTOS TASK MONITOR LIBRARY RTOS1. LB	099-000060
RTOS HANDLER LIBRARY RTOS2. LB	099-000061
RTOS CASSETTE HANDLER LIBRARY (CASDR. LB)	099-000062
RTOS MAGNETIC TAPE HANDLER LIBRARY (MTADR. LB)	099-000063
RTOS FIXED HEAD DISK HANDLER LIBRARY (DSKDR. LB)	099-000064
RTOS MOVING HEAD DISK HANDLER LIBRARY (DKPDR. LB)	099-000065

List of Tapes for System Generation and System Loading

Loading RTOSGEN in a SOS or Stand-alone Environment

RTOSGEN is provided as a stand-alone program on paper tape for users wishing to perform RTOS system generations without the support of RDOS. Standard binary load procedures, described in section 2.8 of How to Use the Nova Computers, must be followed to load either the stand-alone RTOSGEN program or RTOSGEN run under SOS.

Loading RTOSGEN in an RDOS Environment

RTOSGEN is provided as a save file on paper tape for users wishing to perform a system generation on an RDOS system. To load this save file, mount tape number 088-000082 in the reader, and type the following command to the CLI:

LOAD/V $\left\{\begin{matrix}\$PTR \\ \$TTR\end{matrix}\right\}$)

The system will respond:

LOAD $\left\{\begin{matrix}\$TTR \\ \$PTR\end{matrix}\right\}$, STRIKE ANY KEY.

Load the reader with the dump tape and strike any console key. The tape will be read, and the teletype will respond:

RTOSGEN.SV

Producing the RTOS Module

You are now ready to begin executing RTOSGEN. This program configures the system by interrogating you as to the hardware characteristics and channel/task requirements of your program. If RTOSGEN is loaded using binary load procedures, it will self-start. In an RDOS system, type the command

RTOSGEN)

to invoke the program.

The system generation program now outputs the message:

RTOS SYSTEM GENERATION

and proceeds to issue a series of questions requiring operator keyboard responses. An improper reply to an RTOSGEN question causes the question to be repeated. The questions are given and responded to in the following order unless stated otherwise.

1. CORE STORAGE (IN K WORDS)

   Respond with any number from 4 (4K) to 32 (32K) in increments of 2 (2K), and follow this and all other responses with a carriage return.

2. RTC FREQ (0=NONE, 1=10HZ, 2=100HZ, 3=1000HZ, 4=LINE)

   Respond with 0, 1, 2, 3, or 4 as appropriate. If you give a non-zero response, the system will maintain the system clock and calendar. You are cautioned to select the lowest acceptable frequency, since higher clock frequencies increase system overhead. If the line frequency is requested, RTOS asks question 3; otherwise, it steps to question 4.

3. LINE FREQ (0=50HZ, 1=60HZ)

   Select 0 or 1 as appropriate. RTOSGEN now goes to question 4.

4. TASKS (1-255) ?

   Respond with a decimal integer from 1 through 255 corresponding to the number of tasks required by your program which will be loaded with the RTOS module. If you select one task, the minimum task scheduler, TMIN, will be loaded from the RTOS library; otherwise the multitask scheduler, TCBMON, will be loaded.

5. CHANNELS (1-63) ?

   Respond with a decimal integer from 1 through 63, corresponding to the number of simultaneously open channels required by your program.

   After you have answered questions 1 through 5, RTOSGEN responds:

   RESPOND WITH NUMBER OF UNITS

   RTOSGEN now continues with a series of questions concerning peripheral support given to your program.

6. DSK (0-1) ? (fixed head disks)

   If you respond "0", the program steps to the next question; a response of "1" prompts the following series of questions:

   DISK STORAGE (IN K WORDS)

   Respond with any decimal integer from 128 (K equals 1024_{10}) through 2048 (2 million words) in increments of 128 (128K). The program then queries you about the file subdivisions and file names you may wish to assign to the disk space for the fixed head disk. These queries are made in a series of questions with three parts each:

```
1ST BLOCK?
   END BLOCK?
       NAME?
```

You respond with the first logical block address in each disk file; the first available block will be 0 unless you plan to use the disk bootstrap program, HIPBOOT, to load and execute RTOS programs. If disk space for HIPBOOT is to be reserved, the first available block address is block number 6.

You must assign a file name to all disk space which you want to be program accessible. The file names you define will be the names by which the disk files are opened via the system .OPEN command. File names consist of from 4 to 6 upper-case alphabetic and numeric characters. Each file name must uniquely identify its associated file; the same block cannot be assigned more than one file name, since one disk block cannot be included within the file space of more than one file. Files defined during system generation cannot be expanded or reduced in size. For a discussion of contiguous disk files, see Chapter 1, "Disk File Organization."

After defining all file space, respond with a carriage return to the "1ST BLOCK" question. RTOSGEN will now proceed to question number 7.

7. DKP (0-4) ? (moving head disks)

Respond with the number of moving head disk units in your system (the 4047B is considered to be two units). If there are no moving head units, respond 0; the system will then proceed to question 8.

After you answer the unit number question affirmatively, the program requests the number of sectors per surface in each unit, and the number of disk surfaces per unit (see How to Use the Nova Computers for a discussion of disk terms):

```
#SECTORS ?
# SURFACES/UNIT ?
```

Specify 6 sectors for the 4048 unit, or 12 sectors for either the 4047 or the 4057 units. Specify 2 surfaces for the 4047 unit, 10 for the 4048 unit, or 20 for the 4057 unit.

The program then queries you about the file subdivisions and file names you wish to assign to the disk space for each moving head disk unit. These queries are made in a series of questions with three parts each:

Producing the RTOS Module (Continued)

1ST BLOCK?
END BLOCK?
NAME?

You respond with the first logical block address in each disk file; the first available block in each unit will be 0 unless you plan to load the disk bootstrap program, HIPBOOT, for use in loading and executing RTOS programs. If disk space for HIPBOOT is to be reserved, the first available block address is block number 6.

You must assign a file name to all disk space which you want to be program accessible. The file names you define will be the names by which the disk files are opened via the system .OPEN command. File names consist of from 4 to 6 upper-case alphabetic and numeric characters. Each file name must uniquely identify its associated file; the same block cannot be assigned more than one file name, since a single disk block cannot be included within the file space of more than one file. Files defined during system generation cannot be expanded or reduced in size. For a discussion of contiguous disk files, see Chapter 1, "Disk File Organization."

After defining all file space within each moving head unit, respond with a carriage return to the "1ST BLOCK" question. RTOSGEN will now proceed to question number 8.

8. MTA (0-8) ? (magnetic tape transports)

   Respond with the appropriate integer indicating the number of 7- or 9-track magnetic tape transports in your system.

9. CAS (0-8) ? (cassette units)

   Respond with the appropriate integer indicating the number of cassettes in your system.

10. PTR (0-2) ? (high-speed paper tape readers)

   Respond with the appropriate integer indicating the number of high-speed paper tape readers in your system.

11. PTP (0-2) ? (high-speed paper tape punches)

   Respond with the appropriate integer indicating the number of high-speed paper tape punches in your system.

12. LPT (0-2) ? (line printers)

    Respond with the appropriate integer indicating the number of line printers in your system. If your response is 1 or 2, the program asks you for the column size(s) of your printer(s) with the query:

    COLUMN SIZE (80, 132)

    The query is repeated if you have specified 2 line printers in your system.

13. CDR (0-2) ? (punched or mark sense card readers)

    Respond with the appropriate integer indicating the number of card readers in your system.

14. PLT (0-2) ? (incremental plotters)

    Respond with the appropriate integer indicating the number of digital plotters in your system.

15. QTY (0-64) ? (asynchronous data communications multiplexor lines)

    Respond with the appropriate integer indicating the number of full duplex lines in your system.

16. TTYS (0-3) ? (teletypewriters or video displays)

    Respond with the appropriate integer indicating the number of teletypewriters or video display units in your system.

17. MCA (0-15) ?

    Respond with the appropriate integer indicating the number of MCA lines in your system (each line is capable of both transmitting and receiving). If your response is non-zero, the program asks you for the default number of transmission retries:

    #TIMEOUT RETRIES (0-65535) ?

    Each hardware timeout consumes approximately 10 milliseconds. After you respond to this question, the program outputs a query signaling the approach of the last block of RTOSGEN questions:

    RESPOND WITH 0 FOR NO, 1 FOR YES

18. AUTO RESTART?

Respond "1" if the power fail/auto restart is included in your system; otherwise, respond to "0".

19. HIGH PRIORITY INTERRUPTS?

Respond "1" if you have any user-written drivers whose interrupts you want to be serviced before all system devices (but after the power fail monitor and real time clock). Respond "0" if you do not have any high priority interrupt devices. If you respond "1", the program will ask you for the name and device code of each high priority interrupt device (the name consists of 3 alphanumeric characters):

DEVICE CODE?
NAME?

The system appends "IS" to the device name and inserts the name into the high priority interrupt table. After listing all high priority interrupt devices, respond with a carriage return to the DEVICE CODE query, and the program will then step to question 20.

20. USER SUPPLIED DRIVERS?

Respond "1" if you have any user-written drivers you want included in the RTOS module output by RTOSGEN. (This is an alternative to introducing drivers at run time by the system call .IDEF .) If you have no such drivers, respond "0". If you respond in the affirmative, the program requests the device code and name of each device (the name consists of 3 alphanumeric characters):

DEVICE CODE?
NAME?

The system appends "DC" to the device name to create the DCT name for the device. After listing all user device drivers, respond with a carriage return to the DEVICE CODE query, and the program will then step to the RTOS system generation summary.

Having responded to the above questions, RTOS outputs a list of all device codes, DCT names, and device names for all system devices and user devices specified during the system generation process. The power fail/auto restart option and high priority user interrupt devices are not included in this list, since there is no DCT associated with these devices.

Producing the RTOS Module (Continued)

After the list has been finished, the program asks whether the system generation procedure has been followed satisfactorily:

SYSGEN OKAY?

At this point you must decide whether or not there are any errors in your selections of devices; if there are no errors, type "1"; otherwise type "0". If you type "0" the entire system generation dialogue will be repeated.

21. If you respond "1", RTOSGEN will then ask the name of the file or device which is to be used for outputting the RTOS module:

    OUTPUT FILENAME?

    Respond with the name of the appropriate file or device. Respond with a carriage return if the default name RTOS.RB is desired.

After the RTOS module has been output under RDOS, the system will output an R prompt and return to the CLI. RTOSGEN will restart itself in a stand-alone program.

The following information is given as a guide for estimating the size in words of any tailored RTOS system. Sizes are given in decimal and exclude page zero requirements:

Basic System: (1)	1393
Options:	
Multitask Programming (2)	662
Each additional TCB	12
Each additional channel	2
Power Fail/Auto Restart (PWRDR)	103
High Priority Interrupts (n = number of high priority devices excluding RTC and PWR)	11 + 3 n
Real Time Clock (RTCDR)	296
Peripherals:	
Fixed Head Disk (DSKDR)	147
Moving Head Disk (DKPDR)	309
Each disk file	5 for fixed head, 6 for moving head
Magnetic Tape Controller (MTADR)	49
Cassette Controller (CASDR)	49
Tape service routine (MTSER); can be shared by MTA and CAS	444
Teletype Driver (TTYDR)	233
Each additional Teletype (TTY1D, TTY2D)	65
Paper Tape Reader (PTRDR)	141
Second Paper Tape Reader (PTR1D))	37
Paper Tape Punch (PTPDR)	44
Second Paper Tape Punch (PTP1D)	24
Card Reader (CDRDR)	289
Second Card Reader (CDR1D)	120
Line Printer (LPTDR)	44
Second Line Printer (LPT1D)	24
Plotter (PLTDR)	43
Second Plotter (PLT1D)	24
Device Name Table, .CHTB	4* no. of single file devices
Type 4060 Multiplexor (QTYDR)	502
Each QTY line	10
Multiprocessor Communications Adapter (MCADR)	495
MCA Device File Table (.MCTB)	20* (no. of lines + 1)

(1) TMIN, SYSTE, INTD, RTIN, GENIO, IOSER

(2) TCBMON (less the size of TMIN), TXMT, TACAL, TSKID, TUMOD, TABOR

This page illustrates a sample output listing of the RTOSGEN dialogue.

```
    RTOS SYSTEM GENERATION

    CORE STORAGE (IN K WORDS) 12

    RTC FREQ (Ø=NONE,1=1ØHZ,2=1ØØHZ,3=1ØØØHZ,4=LINE) 4
    LINE FREQ (Ø=5ØHZ,1=6ØHZ) 1

    TASKS(1-255) ? 1Ø
    CHANNELS(1-63) ? 8

  RESPOND WITH NUMBER OF UNITS

    DSK(Ø-1) ? 1
      DISK STORAGE (IN K WORDS) 128

      DISK FILE STRUCTURE
      1ST BLOCK? 5
        END BLOCK? 2ØØ
          NAME? FILEA
      1ST BLOCK? 2Ø1
        END BLOCK? 5Ø6
          NAME? FILEB
      1ST BLOCK? 5Ø7
        END BLOCK? 5Ø9
          NAME? FILEC
      1ST BLOCK? 51Ø
        END BLOCK? 511
          NAME? FILED
   DKP(Ø-4) ? Ø
   MTA(Ø-8) ? Ø
   CAS(Ø-8) ? Ø
   PTR(Ø-2) ? 1
   PTP(Ø-2) ? 1
   LPT(Ø-2) ? 1
     COLUMN SIZE (8Ø,132) 8Ø
   CDR(Ø-2) ? Ø
   PLT(Ø-2) ? Ø
   QTY(Ø-64) ? Ø
   TTYS(Ø-3) ? 1
   MCA(Ø-15) ? 2
     # TIMEOUT RETRIES (Ø-65535) ? 2ØØ

   RESPOND WITH Ø FOR NO, 1 FOR YES

   AUTO RESTART ? 1

   HIGH PRIORITY INTERRUPTS? Ø

   USER SUPPLIED DRIVERS? Ø

      SUMMARY OF RTOS SYSGEN

      CODE      DCT NAME    NAME

       Ø6        MCTDC
       Ø7        MCRDC
       1Ø        TTIDC       $TTI
       11        TTODC       $TTO
       12        PTRDC       $PTR
       13        PTPDC       $PTP
       14        RTCDC
       17        LPTDC       $LPT
       2Ø        DSKDC

  SYSGEN OKAY? 1
  OUTPUT FILENAME ? SYS1
R
```

LOADING AND RUNNING A PROGRAM IN A STAND-ALONE ENVIRONMENT

Having produced the RTOS module, you are now ready to load the operating system with program relocatable binaries and execute it as an RTOS program or run it under RDOS. If you wish to load and execute it in a stand-alone environment you must first perform a stand-alone or SOS relocatable load.

Performing a Stand-alone or SOS Relocatable Load

For a complete description of operating procedures using the SOS and stand-alone extended relocatable loaders, refer to the Extended Relocatable Loaders Manual, 093-000080. The following summarizes stand-alone procedures required for loading system and user programs.

The stand-alone relocatable loader is in absolute binary format, and thus it is loaded by means of the binary loader. Once loaded, the relocatable loader self-starts and types on the console:

SAFE=

You respond with a carriage return to reserve the upper 200 words of memory, preserving both the bootstrap and binary loaders. The relocatable loader now outputs the prompt:

*

You now proceed to load a series of paper tapes, following tape loading procedures which will be described. The order in which the first three categories of tapes is loaded is not critical; the RTOS libraries must be the last items loaded:

1. The RTOS module
2. User drivers (if any)
3. User relocatable binaries (the user program proper)
4. The RTOS libraries

To load each of the above paper tapes, mount each tape in turn in either the teletype reader or the paper tape reader, and type either 1 or 2 indicating to the loader whether the teletype reader (1) or the high-speed reader (2) is to be used. After loading each tape, the loader outputs the star prompt (*).

Performing a Stand-alone or SOS Relocatable Load (Continued)

After the last paper tape has been loaded, you may request a loader map by typing the number 6. Then, to terminate the load process and prepare the program for execution, type the number 8. This causes the previously loaded program to be moved so that it resides at the absolute addresses indicated by the loader map. After shuffling the program downward to its indicated positions in core memory, the loader halts.

If the SOS Magnetic Tape/Cassette relocatable loader (SOS loader) is to be used, the core image loader/writer should first be loaded into main memory; the same series of files must then be loaded in sequence. After the prompt

RLDR

is received, a command line must be input via the console. Suppose that cassette files are used and three cassette transports are available. Moreover, the user RTOS program binaries exist on file zero of one cassette, and the RTOSGEN module, user drivers, and RTOS libraries exist on files 0, 1, and 2 of another cassette. One possible command line would be as follows:

$TTO/L CT0:1/S CT1:0 CT1:1 CT2:0 CT1:2)

This command line would cause a numeric symbol table listing to be output on $TTO and the RTOS save file to go to file 1 of transport 0 (CBOOT, the cassette bootstrap, exists on file 0). The RTOS module, user drivers user program binaries, and RTOS libraries are then loaded in order. Upon the successful completion of the relocatable load, the message "OK" is output on the console and the system halts with the loaded program in core memory ready to be executed.

Executing a Stand-alone Program

The first operation performed by any program run under RTOS is a system initialization. Since the address of the RTOS initialization routine always resides in location 2, control must be dispatched to the address in location 2 in order to start an RTOS program. RTOS has a JMP @2 instruction at location 376. Thus to start your program, place "376" in the data switches, press RESET, then START.

RTOS will initialize the system and transfer control to your program if you defined a starting address in your program in the .END statement. If you defined no such starting address, you must now place your program starting address in the data switches and press START. Your program now runs until a HALT or JMP . instruction is encountered, or all tasks are killed (in which case control is returned to the task scheduler).

LOADING AND RUNNING A PROGRAM IN AN RDOS ENVIRONMENT

After writing your program, you may test it as an RDOS save file, you may run it on an RDOS system as a stand-alone program (disabling RDOS temporarily), or you may output it as an absolute program on some external medium for execution on another system.

Running the program under RDOS for test purposes may save you debugging time. This procedure is equivalent to testing and running an RDOS save file, since the RTOS command set is a compatible subset of the RDOS command repertoire. If you wish to test your program under RDOS, consult the Extended Relocatable Loaders Manual, 93-000080, and the Real Time Disk Operating System User's Manual, 93-000075. Note that you may specify task/channel information either via RLDR local switches /C and /K, or in the .COMM TASK statement.

Alternatively, you want to load your RTOS program using the RDOS loader but with the RTOSGEN module and RTOS libraries, and execute your program on this RDOS system or on another RDOS system. The following sections describe these procedures.

Loading an RTOS Program under RDOS

For a complete description of operating procedures using the RDOS relocatable loader, refer to the Extended Relocatable Loaders Manual, 93-000080, and to the Real Time Disk Operating System User's Manual, 93-000075. The following summary information gives procedures for loading a user program with the RTOS module under RDOS.

The following files are required for the relocatable loading process:

1. RTOS.RB (produced during RTOS system generation).
2. User drivers, if any.
3. User relocatable binaries (i.e., the user program proper).
4. The RTOS libraries.

These four files must be loaded onto disk by mounting each on the teletype reader or high-speed reader and by issuing the following command:

$$\text{XFER} \begin{Bmatrix} \$PTR \\ \$TTR \end{Bmatrix} \underline{\text{filename}}\ \rangle$$

Having loaded all the necessary tapes, you now issue the CLI relocatable load command:

Loading an RTOS Program under RDOS (Continued)

RLDR/C [/D] user binaries [user drivers] RTOSGEN module RTOS libraries ↑

[{$TTO/L / $LPT/L}])

This command will cause the complete save file to be constructed, starting at location zero. The load map will be output on the listing device, if one is specified.

If in the above illustration you have loaded the debugger (global /D), you must do one of two things to transfer control to the debugger. Your choice depends upon the means selected to start the RTOS program. These means are described fully in the following section. If location 2 will receive control upon the execution of the save file, then the debugger address (found in the load map, DEBUG) must be placed on location 2 via the Octal Editor. Alternatively, you may get the debugger address from location 406 of the program's User Status Table and start at that address. In either case, after starting the debugger, the command " .RTOS$R" will transfer control to the RTOS initializer, starting the RTOS program.

Executing an RTOS Program with HIPBOOT

Having performed a relocatable load of your program with the RTOS module and RTOS libraries, you may now choose from many different methods to execute the program; your choice of method will be made largely on the basis of the type of system which will be used to run the program.

If you intend to execute the program on an RDOS system, you may use the disk bootstrap program, HIPBOOT, to transfer control to your program. Otherwise, you must produce an absolute binary paper tape, or a magnetic tape/cassette version of the program and load it on another system by means of an appropriate loader. If HIPBOOT is used, the computer halts at the termination of loading. This provides an opportunity to activate the debugger via the front panel switches, if desired.

If your program is to be executed on the present RDOS system or on another RDOS system you may invoke the disk bootstrap program, HIPBOOT, to execute your program. If the RTOS program is on a removable pack or cartridge you may simply transfer the pack or cartridge to the new system; otherwise you must dump the RTOS save file and reload it on the new system:

DUMP outputdevicename program-name.SV)

LOAD inputdevicename program-name.SV)

Having obtained a copy of the RTOS program on the RDOS system where it is to be run, you now invoke HIPBOOT, following ordinary disk bootstrap procedures. These procedures are described fully in the RDOS User's Manual, Appendix E; a summary of these procedures follows.

Disk bootstrap procedures vary with the type of computer used and the presence or absence of the program load feature. The following three procedures are given.

1. Nova 2/Nova 1200/Nova 800 series without the Program Load Option:

   a. Enter in location 376: 601nn
      where nn=20 when bootstrapping from the fixed head disk and
      nn=33 when bootstrapping from the moving head disk.

   b. Enter in location 377: 377.

   c. Press RESET, then start at location 376; go to step 3c.

2. Nova 2/Nova 1200/Nova 800 series with the Program Load Option:

   a. Set bit 0 of the data switches up.

   b. Enter the proper disk device code (20 or 33 as described in 1a.) into the data switches, bits 10-15.

   c. Press RESET, then PROGRAM LOAD; go to step 3c.

3. Supernova:

   a. Enter the proper disk device code (20 or 33 as described in 1a).

   b. Press RESET, then CHANNEL START.

   c. HIPBOOT now requests the name of your program.

FILENAME?

You must now respond with the name of your program in one of two ways:

1. name/A)
2. name)

If you select the first method, your program will be loaded into memory and the computer will halt. You then place a starting address (your program's or the debugger's) into the address switches, press RESET, and then START. If you select the second response, the program will self-start.

Having been loaded, if the RTOS program self-starts it will initialize the system and transfer control to your program if you defined a starting address via your program's .END statement. If the program does not self-start, you may start at either the contents of location 2 or, to activate the debugger, at the contents of location 406.

Since this procedure overwrites portions of RDOS, RTOS gains control; RDOS can only be restored via disk bootstrap procedures similar to those given on the previous page.

Executing an RTOS Program with TBOOT, CBOOT or MCABOOT

If you wish to execute your program on a system which has a magnetic tape transport or cassette unit, you must first transfer the magnetic tape bootstrap (TBOOT) or cassette bootstrap (CBOOT) to file 0 of the tape on unit 0.

```
INIT   {MT0}  )
       {CT0}

XFER   {TBOOT.SV}  {MT0:0}  )
       {CBOOT.SV}  {CT0:0}
```

After transferring the appropriate tape bootstrap to file 0, transfer the RTOS program to file 1 of the same tape:

```
XFER   program name.SV   {MT0:1}  )
                         {CT0:1}
```

If several RTOS programs are to be stored on the tape reel, they can be transferred to sequential file numbers following file 1.

Having produced the tape reel, release the reel via the RDOS command

$$\text{RELEASE} \left\{ \begin{matrix} \text{MT0} \\ \\ \text{CT0} \end{matrix} \right\} \downarrow$$

Dismount the reel, and mount it on the unit zero transport or cassette unit of the system where the RTOS program(s) is to be executed.

After mounting the reel, perform one of the following operation sequences:

1. On machines having the Program Load feature (Nova 2/800/1200 families), set the console switches to 100022 for a magnetic tape transport, or 100034 for a cassette unit, and press PROGRAM LOAD.

2. On Supernovas with the Channel Start option, set the console switches to octal 22 (for a magnetic tape unit) or octal 34 (for a cassette unit), and press RESET, then CHANNEL START.

3. On machines without Channel Start or Program Load options, deposit NIOS MTA (60122) or NIOS CTA (60134) in location 376, deposit 377 in location 377, press RESET, and START at location 376.

The appropriate tape bootstrap program will be loaded into memory, and the following initialization message will be output on the console:

FULL (0) OR PARTIAL (1) ?

Type "1" in response to this query, and the tape bootstrap will then request the number of the file containing the program which is to be executed:

$$\text{FROM} \left\{ \begin{matrix} \text{MT0:} \\ \\ \text{CT0:} \end{matrix} \right\}$$

Respond with the number of the file containing the program, and follow this with a carriage return. The tape file will be read, and control will be given to your program.

To transmit an RTOS program from one CPU to another via the multiprocessor communications adapter, two actions must be performed: one at the transmitting RDOS CPU, and another at a receiving CPU. At the transmitting CPU, the following CLI command must be issued.

MCABOOT   MCAT:n   RTOS program name/S

where n is a number from 1 to 15 decimal denoting the receiving MCA unit number. At the receiving CPU, an operator must have requested the sender's transmission by first placing "100007" in the receiver's data switches, then by depressing RESET, followed by PROGRAM LOAD. The transmitting unit will wait for the receiver to request the transmission, up to the default timeout period. After the RTOS program has been received, the receiving operator must place the value "376" in the data switches, and then press RESET followed by START.

EXECUTING AN RTOS PROGRAM VIA PAPER TAPE WHICH WAS PRODUCED UNDER RDOS

If you have loaded your RTOS program via the RDOS relocatable loader, but now wish to run this program in a stand-alone environment, you may output this program onto paper tape. If you do not wish to preserve a disk file containing this RTOS program, simply issue the command:

MKABS/Z   program name.SV [376/S] {$TTP | $PTP} ↵

Alternatively, if you wish to preserve a disk copy of this program before outputting it onto paper tape, issue the commands:

MKABS/Z   program name.SV   program name.AB [376/S] ↵

XFER   program name.AB $PTP ↵

The paper tape version of your program can now be loaded onto a non-RDOS system following conventional binary load procedures. After loading the paper tape, if you supplied the intializer entry with the /S switch, control will be transferred to the RTOS initializer; if you did not specify an initializer entry address, you must place the value "376" into the data switches, press RESET, then START.

In either case, after RTOS performs its initialization it will transfer control to the starting address of your program if you specified a starting address as an argument in the main program's .END statement. If you specified no such starting address, the machine will stop after initialization and you must place your program's starting address in the data switches, press RESET, then START. Your program will now run until a HALT or JMP . instruction is encountered.

EXECUTING AN RTOS PROGRAM WITH THE SOS CORE IMAGE LOADER/WRITER

If the SOS core image loader/writer is to be used to execute an RTOS program, one of two relocatable loaders must have been used and one of two corresponding means must have been used to place the RTOS program onto cassette or magnetic tape:

1. SOS relocatable loader (the core image loader/writer must have been resident in main memory before the relocatable load).

2. RDOS relocatable loader.

If the first method is selected, the core resident RTOS program must be written onto either a reel of magnetic tape or a cassette cartridge onto which the loader/writer has been written as file zero. Core image loader/writer operation is discussed in detail in the SOS User's Manual, 093-000062. In summary, to transfer the SOS relocatably loaded program to tape, start the computer at the next to last address in main memory. This will activate the loader/writer, which outputs a prompt, "#". After outputting the prompt, the loader/writer waits for you to input a device number and file number, separated by a colon, to which the core resident program is to be written. After the file has been specified, the core image writer will request specification of the upper core address (NMAX) to be written onto tape. It does this by typing

NMAX:

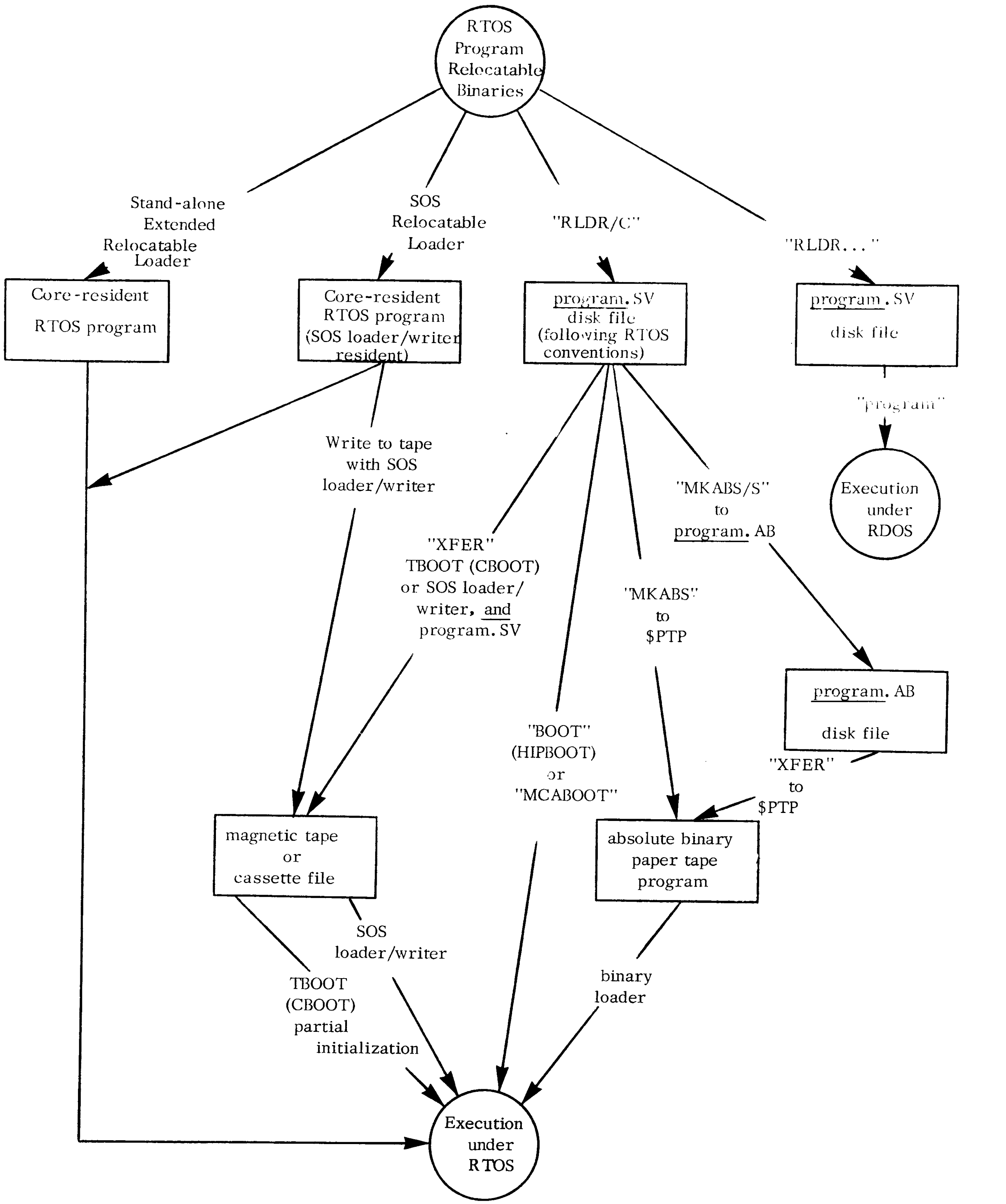
on the console. You must then respond with the highest core address (in octal) which is to be written out.

EXECUTING AN RTOS PROGRAM WITH THE SOS CORE IMAGE LOADER/WRITER
(Continued)

Having written the RTOS program onto tape, it then may be read in from tape and be executed by means of the core image loader. Set the console data switch 0 up, and activate the core image loader by starting the computer at the highest available memory address. The loader issues the "#" prompt on the console, and then awaits your response of a device number (0-7),colon, and a file number (0-99) followed by a carriage return. The indicated tape file is then loaded into memory starting at address zero. The loader will halt after loading is complete.

If instead of the SOS loader the RDOS relocatable loader is used to create the RTOS program, this save file can be XFERed by the CLI to a file on cassette or magnetic tape. This file can then be executed by means of the core image loader as described above.

The illustration on the following page summarizes the various methods which may be used to execute an RTOS program. The illustration starts with the relocatable binary modules comprising the program; these binaries are produced either by an RDOS assembler or by the SOS assembler.



RTOS Execution Procedures

APPENDIX C

RTOS PARAMETERS

Supplied with RTOS is a set of parameters which must be assembled with programs using RTOS and using system-defined mnemonics. A listing of these parameters begins on page C-2.

```
                ;*******************************************************;
                ;                                                       ;
                ;       RRRRR      TTTTTTT      OOOO        SSSS        ;
                ;       R    R        T        O    O      S    S       ;
                ;       R    R        T        O    O      S            ;
                ;       RRRRR         T        O    O       SSSS        ;
                ;       R   R         T        O    O           S       ;
                ;       R    R        T        O    O      S    S       ;
                ;       R    R        T         OOOO        SSSS        ;
                ;                                                       ;
                ;       THE OPERATING SYSTEM YOU CAN UNDERSTAND         ;
                ;*******************************************************;



                ;RTOS - - REAL TIME OPERATING SYSTEM
                ;     - - FOR THE NOVA FAMILY OF COMPUTERS

                ;  **** RTOS PARAMETER TAPE ****

                ;FORTRAN 4 CONDITIONAL ASSEMBLY SWITCH

        000000 F4SW=0           ;SUPERSEDED BY FORTRAN IV PARAMETERS IF PRESENT


                ; DEFINE THE USER STACK DISPLACEMENTS

        177771 .DUSR SSLGT=     -7      ; VARIABLE LENGTH OF CALLING'S FRAME
        177772 .DUSR SSOSP=     -6      ; PREVIOUS STACK POINTER
        177773 .DUSR SSRTN=     -5      ; RETURN ADDRESS OF CALLING PROGRAM
        177774 .DUSR SSEAD=     -4      ; ENTRY ADDRESS OF CALLED ROUTINE
        177775 .DUSR SSCRY=     -3      ; CARRY
        177776 .DUSR SSAC0=     -2      ; SAVE STORAGE FOR CALLER'S ACCUMULATORS
        177777 .DUSR SSAC1=     -1
        000000 .DUSR SSAC2=     0       ; (DON'T MODIFY THIS DISPLACEMENT!!)


                        ; DEVICE CHARACTERISTICS

                ;.DUSR   ...      1B15   ; IF SUPPRESSED ON OPEN:
                                         ;; USER SPECIFIED TIMEOUT (MCA).
        000001 .DUSR    DCCPO=    1B15   ; REQUIRES LEADER/TRAILER
        000002 .DUSR    DCCGN=    1B14   ; REQUIRES TAB SIMULATION
        000004 .DUSR    DCIDI=    1B13   ; BUF'D INPUT DEV. (RDOS: OP. INTERV.)
        000010 .DUSR    DCCNF=    1B12   ; OUTPUT DEV. WITHOUT FORM FEED HARDWARE
        000020 .DUSR    DCTO=     1B11   ; TELETYPE OUTPUT DEVICE
        000040 .DUSR    DCKEY=    1B10   ; KEYBOARD DEVICE
        000100 .DUSR    DCNAF=    1B09   ; REQUIRES NULLS AFTER FORM FEEDS
        000200 .DUSR    DCRAT=    1B08   ; REQUIRES RUBOUTS AFTER TABS
        000400 .DUSR    DCPCK=    1B07   ; REQUIRES PARITY CHECK
        001000 .DUSR    DCLAC=    1B06   ; REQUIRES LINE FEEDS AFTER C.R.'S
        002000 .DUSR    DCPFR=    1B05   ; INTERNAL TO RTOS: AUTORESTART MODE BIT
        004000 .DUSR    DCFWD=    1B04   ; INTERRUPT PER FULL WORD I/O DEVICE
        010000 .DUSR    DCFFO=    1B03   ; REQUIRES FORM FEED ON OPEN
        020000 .DUSR    DCLTU=    1B02   ; CHANGE LOWER CASE ASCII TO UPPER
        040000 .DUSR    DCC80=    1B01   ; WRITE 80 COLUMNS
                ;.DUSR   ...      1B0    ; DEVICE WITH BEAD QUEUE
                                         ; (INTERNAL TO RTOS, NON-PARAMETRIC)
```

```
0002 .MAIN

                ; USER STATUS TABLE (UST) TEMPLATE

     000400 .DUSR    UST=     400      ; START OF USER STATUS AREA
     000000 .DUSR    USTPC=   0        ; RESERVED FOR RDOS COMPATIBILITY
     000001 .DUSR    USTZM=   1        ; ZMAX
     000002 .DUSR    USTSS=   2        ; START OF SYMBOL TABLE
     000003 .DUSR    USTES=   3        ; END OF SYMBOL TABLE
     000004 .DUSR    USTNM=   4        ; NMAX
     000005 .DUSR    USTSA=   5        ; STARTING ADDRESS
     000006 .DUSR    USTDA=   6        ; DEBUGGER ADDRESS
     000007 .DUSR    USTHU=   7        ; HIGHEST ADDRESS USED
     000010 .DUSR    USTCS=   10       ; FORTRAN COMMON AREA SIZE
     000011 .DUSR    USTIT=   11       ; INTERRUPT ADDRESS (NOT USED)
     000012 .DUSR    USTBR=   12       ; BREAK ADDRESS (NOT USED)
     000013 .DUSR    USTCH=   13       ; # TASKS (LEFT), # CHANS (RIGHT)
     000014 .DUSR    USTCT=   14       ; INITIAL TCB ADDRESS
     000015 .DUSR    USTAC=   15       ; START OF   R E A D Y   TCB QUEUE
     000016 .DUSR    USTFC=   16       ; START OF FREE TCB CHAIN
     000017 .DUSR    USTIN=   17       ; INITIAL START OF NREL CODE
     000020 .DUSR    USTOD=   20       ; RESERVED FOR RDOS COMPATIBILITY
     000021 .DUSR    USTSV=   21       ; FORTRAN STATE SAVE ROUTINE (OR 0)
            ;                            --RDOS UST ENDS HERE--
     000022 .DUSR    USTSQ=   22       ; START OF SUSPEND QUEUE
     000023 .DUSR    USTXQ=   23       ; START OF .XMT/.REC QUEUE
     000024 .DUSR    USTPQ=   24       ; START OF PSEUDO TASK QUEUE
     000025 .DUSR    USTOS=   25       ; 1-SECOND CLOCK, SCHEDULER IDLE COUNTER
     000026 .DUSR    USTNA=   26       ; NUMBER OF ACTIVE TASKS (TCB'S IN USE)
     000026 .DUSR    USTEN=   26       ; LAST ENTRY


                ; LAYOUT OF AN RTOS TASK CONTROL BLOCK (TCB)

     000000 .DUSR TPC=0       ; USER PC + CARRY
     000001 .DUSR TAC0=1      ; AC0
     000002 .DUSR TAC1=2      ; AC1
     000003 .DUSR TAC2=3      ; AC2
     000004 .DUSR TAC3=4      ; AC3
     000005 .DUSR TPRST=5     ; STATUS BITS + PRIORITY
     000006 .DUSR TSYS=6      ; SYSTEM CALL WORD
     000007 .DUSR TLNK=7      ; LINK WORD
     000010 .DUSR TUSP= 10    ; USP
     000011 .DUSR TELN=11     ; TCB EXTENSION ADDR(USED BY FORTRAN SCHEDULERS)
     000012 .DUSR TID=12      ; TASK ID ENTRY
     000013 .DUSR TTMP=13     ; RDOS REV 3'S CONTRIBUTION
     000014 .DUSR TLN=TTMP-TPC+1
```

```
10003 .MAIN

              ; INTERRUPTED MACHINE STATUS STORAGE BLOCK LAYOUT

      000000 .DUSR   IPCC=     0         ; PROGRAM COUNTER AND CARRY
      000001 .DUSR   IAC0=     1         ; ACCUMULATOR STORAGE
      000002 .DUSR   IAC1=     2
      000003 .DUSR   IAC2=     3
      000004 .DUSR   IAC3=     4
      000005 .DUSR   ICMSK=    5         ; CURRENT HARDWARE MASK
      000006 .DUSR   IRLOC=    6         ; RLOC
      000007 .DUSR   ISVLN=    IRLOC+1   ; SAVE AREA LENGTH

              ; PARAMETERS DESCRIBING LAYOUT LOCATIONS 0-17

              ;      ...       0         ; INTERRUPT PC STORAGE
              ;      ...       1         ; INTERRUPT SERVICE ROUTINE ADDRESS
      000002 .DUSR   BEGIN=    2         ; OVERALL STARTING ADDRESS (FIXED AT 2)
      000003 .DUSR   CTCB=     3         ; CURRENT TCB ADDRESS--INITIALLY HAS
                                         ;; STARTING ADDR FOR TBOOT USE (FIXED
                                         ;; AT 3).
      000004 .DUSR   SCHED=    4         ; ENTRY POINT TO SCHEDULER
      000005 .DUSR   .SYS.=    5         ; SYSTEM MODE INDICATOR
                                         ;; (INITIALLY 0 FOR HIPBOOT)
      000006 .DUSR   RLOC=     6         ; PAGE ZERO TEMP (=6 FOR COMPATIBILITY)
      000007 .DUSR   IOEND=    7         ; ENTRY POINT TO I/O END PROCESSOR
      000010 .DUSR   .CMSK=    10        ; CURRENT SYSTEM INTERRUPT MASK
      000011 .DUSR   DISMISS=11          ; INTERRUPT DISMISSAL ROUTINE ADDRESS
      000012 .DUSR   USTP=     12        ; DEFINED FOR COMPATIBILITY, SET TO 400
      000012 .DUSR   PANIC=    USTP      ; SYSTEM PANIC (CONTAINS JMP .)
      000013 .DUSR   TLINK=    13        ; ENTRY POINT TO ENQUEUE READY TCB
      000014 .DUSR   RSCHD=    14        ; ENTRY POINT TO READY (CTCB), SCHEDULE
      000015 .DUSR   .TSAVE=   15        ; ADDRESS OF TCB STATE SAVE ROUTINE
              ;.DUSR  USP=      16        ; USER STACK POINTER
              ;      ...       17        ; ENTRY POINT TO SERVICE SYSTEM REQUEST
```

0004 .MAIN

```
                ; DEVICE CONTROL TABLE (DCT) LAYOUT

     000000 .DUSR DCTSV=   0        ; INT STATE SAVE ADDR
     000001 .DUSR DCTMS=   1        ; MASK OF LOWER PRIORITY DEVICES
     000002 .DUSR DCTIS=   2        ; INTERRUPT SERVICE ADDR
     000003 .DUSR DCTCH=   3        ; DEVICE CHARACTERISTICS

     000004 .DUSR DCTCD=   4        ; DEVICE CODE
     000005 .DUSR DCTEX=   5        ; WHERE TO EXECUTE I/O INSTRUCTION
     000006 .DUSR DCTDT=   6        ; COMMAND DISPATCH TABLE ADDRESS

                        ; DEFINE THE COMMAND OFFSETS
     000000             .DUSR   OF=     0       ; OPEN A FILE
     000001             .DUSR   CF=     1       ; CLOSE A FILE
     000002             .DUSR   RS=     2       ; READ SEQUENTIAL
     000003             .DUSR   RL=     3       ; READ LINE
     000004             .DUSR   WS=     4       ; WRITE SEQUENTIAL
     000005             .DUSR   WL=     5       ; WRITE LINE
     000006             .DUSR   RB=     6       ; READ BLOCK
     000007             .DUSR   WB=     7       ; WRITE BLOCK
     000010             .DUSR   OA=     10      ; OPEN FILE FOR APPENDING

     000007 .DUSR DCTST=   7        ; DEVICE START ROUTINE
     000010 .DUSR DCTIN=   10       ; DEVICE INITIALIZATION ROUTINE
     000011 .DUSR DCTLK=   11       ; FORWARD TCB LINKAGE
     000012 .DUSR DCTTO=   12       ; TIMEOUT CONSTANT OR ZERO

                ; THE REMAINING DEFINITIONS ARE FOR BEAD DEVICES ONLY.

     000013 .DUSR DCTQL=   13       ; LINK IN DEVICE REQUEST BEAD CHAIN
     000014 .DUSR DCTDP=   14       ; DEVICE BYTE DATA POINTER
     000015 .DUSR DCTDC=   15       ; DEVICE DATA COUNT
     000016 .DUSR DCTQS=   16       ; BEAD STATUS WORD

     000017 .DUSR DCTBD=   17       ; BEAD ADDRESS (.-4)
     000020 .DUSR DCTQP=   20       ; REQUEST BEAD QUEUE STARTING ADDR
     000021 .DUSR DCTOC=   21       ; OPENED DEVICE CHARACTERISTICS
     000022 .DUSR DCTT1=   22       ; TEMP 1 FOR DEVICE CONTROL
     000023 .DUSR DCTT2=   23       ; TEMP 2 FOR DEVICE CONTROL
     000024 .DUSR DCTCT=   24       ; CURRENT TIMEOUT COUNT (INPUT DEVICE)
     000024 .DUSR DCTCC=   24       ; COLUMN COUNTER (OUTPUT DEVICE)
     000025 .DUSR DCTPR=   25       ; ECHO DEVICE PAIR POINTER (TTI ONLY)
     000025 .DUSR DCTLC=   25       ; LINE COUNTER (OUTPUT DEVICE)
                                    ; -FOR SPECIAL OUTPUT MODE:
     000026 .DUSR DCTSC=   26       ;SAVED DEVICE REQUEST BYTE COUNTER
     000027 .DUSR DCTGN=   27       ;CHARACTER FOR GENERATION
```

```
10005 .MAIN


        ; BEAD COMPONENTS

000000 .DUSR RQLK=      0         ; REQUEST LINK
000001 .DUSR RQPTR=     1         ; REQUEST BYTE POINTER
000002 .DUSR RQCNT=     2         ; REQUEST BYTE COUNT
000003 .DUSR RQST=      3         ; REQUEST STATUS (MODE)

        ; STATUS/MODE BITS ARE DEFINED AS FOLLOWS:
        ;
        ; 1B0   BEAD IN DCT (MAIN BEAD), ASSOCIATED WITH TCB @DCTLK
        ; 1B1   PRESERVE TASK'S AC1, DON'T CALCULATE BYTES TRANSMITTED
        ; 1B15  MEANING DEPENDS ON BIT 0 -
        ;       MAIN BEAD: LINE MODE
        ;       OTHER BEAD: REQUEST DONE (CLEARED IN ENQUE ROUTINE)
        ;
        ;; OFFSETS FOR USER TASK QUEUE TABLE
        ;
        ;.DUSR QPC=      0         ; STARTING PC
        ;.DUSR QNUM=     1         ; NUMBER OF TIMES TO EXECUTE
        ;.DUSR QTOV=     2         ; OVERLAY NUMBER (RDOS)
        ;.DUSR QSH=      3         ; STARTING HOUR
        ;.DUSR QSMS=     4         ; STARTING SEC IN HOUR
        ;.DUSR QPRI=     TPRST     ; MUST BE SAME
        ;.DUSR QRR=      6         ; RERUN TIME IN SECONDS
        ;.DUSR QTLNK=    TLNK      ; MUST BE SAME
        ;.DUSR QOCH=     10        ; CHAN OVERLAYS OPEN ON (RDOS)
        ;.DUSR QCOND=    11        ; TYPE OF LOAD
        ;.DUSR QTLN=     QCOND-QPC+1
```

```



                       ; DEFINE THE DEVICE MASK BITS
                       ;         RDOS REV 3 VERSION

        000003 .DUSR    MKTTO=   1B14+1B15        ; TTO
        000003 .DUSR    MKTTI=   MKTTO            ; TTI
        000003 .DUSR    MKQTY=   MKTTI            ; QTY
        000007 .DUSR    MKPTP=   1B13+MKTTI       ; PTP
        000017 .DUSR    MKMCA=   1B12+MKPTP       ; MCA
        000017 .DUSR    MKPLT=   MKMCA            ; PLT
        000017 .DUSR    MKLPT=   MKPLT            ; LPT
        000217 .DUSR    MKDPO=   1B08+MKMCA       ; DPO
        000217 .DUSR    MKDPI=   MKDPO            ; DPI
        000617 .DUSR    MKDKP=   1B07+MKDPO       ; MHD
        000717 .DUSR    MKDSK=   1B09+MKDKP       ; FHD
        001717 .DUSR    MKIPB=   1B06+MKDSK       ; IPB
        001737 .DUSR    MKPTR=   1B11+MKIPB       ; PTR
        001777 .DUSR    MKCDR=   1B10+MKPTR       ; CDR
        001777 .DUSR    MKMTA=   MKCDR            ; MTA
        001777 .DUSR    MKCAS=   MKMTA            ; CAS
```

```
10007 .MAIN



                  ; DEFINE THE EXCEPTIONAL STATUS CODES

        000000 .DUSR ERFNO=     0           ; ILLEGAL CHANNEL NUMBER
        000001 .DUSR ERFNM=     1           ; ILLEGAL FILE NAME
        000002 .DUSR ERICM=     2           ; ILLEGAL SYSTEM COMMAND
        000003 .DUSR ERICD=     3           ; ILLEGAL COMMAND FOR DEVICE
        000006 .DUSR EREOF=     6           ; END OF FILE
        000012 .DUSR ERDLE=     12          ; A NON-EXISTENT FILE
        000015 .DUSR ERFOP=     15          ; FILE NOT OPENED
        000021 .DUSR ERUFT=     21          ; ATTEMPT TO USE A UFT ALREADY IN USE
        000022 .DUSR ERLLI=     22          ; LINE LIMIT EXCEEDED
        000023 .DUSR ERRTN=     23          ; .RTN/.ERTN WITH NOWHERE TO GO
        000024 .DUSR ERPAR=     24          ; PARITY ERROR ON READ LINE
        000026 .DUSR ERMEM=     26          ; NOT ENUF MEMORY AVAILABLE
        000027 .DUSR ERSPC=     27          ; OUT OF FILE SPACE
        000030 .DUSR ERFIL=     30          ; FILE READ ERROR
        000031 .DUSR ERSEL=     31          ; UNIT NOT PROPERLY SELECTED
        000036 .DUSR ERDNM=     36          ; ILLEGAL DEVICE NAME
        000041 .DUSR ERTIM=     41          ; USER SET TIME ERROR
        000042 .DUSR ERNOT=     42          ; OUT OF TCB'S
        000043 .DUSR ERXMT=     43          ; SIGNAL ADDRESS ALREADY BUSY
        000045 .DUSR ERIBS=     45          ; DEVICE ALREADY IN SYSTEM
        000046 .DUSR ERICB=     46          ; INSUFFICIENT CONTIGUOUS BLOCKS
        000047 .DUSR ERSIM=     47          ; QTY ERROR
               ;.DUSR ERQTS=    50          ; ERROR IN USER TASK QUEUE TABLE
        000060 .DUSR ERFIU=     60          ; FILE IN USE
        000061 .DUSR ERTID=     61          ; TASK ID ERROR
        000101 .DUSR ERDTO=     101         ; DEVICE TIMEOUT
        000103 .DUSR ERMCA=     103         ; MCA ERROR
        000104 .DUSR ERSRR=     104         ; SHORTER RECEIVE REQUEST
        000106 .DUSR ERCLO=     106         ; I/O TERMINATED BY CLOSE
        000110 .DUSR ERABT=     110         ; TASK NOT ABORTABLE
        000113 .DUSR ERNMC=     113         ; NO MCA RECEIVE REQUEST



                          .EOT              ; END OF RTOS PARAMETER TAPE
```

APPENDIX D

RTOS ASSEMBLY LANGUAGE AND FORTRAN IV PROGRAMMING

This appendix illustrates a sample real time assembly language program and describes the procedures which must be followed in order to load and execute a FORTRAN IV program under the Real Time Operating System.

Assembly Language Illustration

The sample assembly language program, illustrated on the following pages, causes four user tasks to compete for the use of the system console, $TTO. Each task types a unique message on channel zero. The sequence of events in the user program is as follows.

First, all of the task calls which will be issued within the program are referenced externally by an .EXTN statement on line 6. Thus the following two task calls will be issued: .PRI and .TASK. If any task call were issued which was not externally referenced, the assembler would report an undefined symbol and the program would not be executable. System calls, on the other hand, must not be externally referenced; the SOS and RDOS assemblers recognize each system call mnemonic and assemble the appropriate value for each mnemonic. The .TXTM 1 statement, line 7, packs all text strings from left to right; this is always required under RTOS and RDOS.

Upon entry to TOT, the start of the program and only entry point declared by the .ENT statement (line 5), the teletype is opened on channel zero.
Next, the program creates three tasks and passes to each task a different displacement into the series of message byte pointers which will be used when the tasks output to the teletype. Note that each of the three tasks is created at priority 10_8. When rescheduling occurs after each task call, the default task will continue to receive control since it is created at priority zero when the program is first started. After creating the three tasks, the default task adjusts its priority to that of the other three tasks so that it can compete for the teletype on an equal priority basis (line 30).

Each of the four tasks now executes the code beginning on line 33, and outputs its message via system call .WRS . As each task issues the system call, it becomes suspended and control goes to the task scheduler, which raises the highest priority ready task to the executing state. Each task remains suspended until its system call is completed. If the error return from any system or task call is taken, the program issues a "JMP ." call, and the task is idled.

The program .END statement has the argument "TOT". This will enable the RTOS initializer to transfer control directly to the program.

```
0001 TOT    MACRO REV 02                          14:00:32 02/25/74

                    ;TYPEOUT TEST WITH MULTITASKING

                              .TITL TOT
                              .ENT TOT
                              .EXTN .PRI .TASK
         000001               .TXTM 1
                              .NREL

   00000'020436 TOT:          LDA 0 .TTO
   00001'126400               SUB 1 1
   00002'000017               .SYST
   00003'014000               .OPEN 0
   00004'004431               JSR ERR

   00005'020425               LDA 0 PRIOR
   00006'024425               LDA 1 NEWTASK
   00007'152400               SUB 2 2
   00010'177777               .TASK
   00011'004424               JSR ERR

   00012'151400               INC 2 2
   00013'000010'              .TASK
   00014'004421               JSR ERR

   00015'151400               INC 2 2
   00016'000013'              .TASK
   00017'004416               JSR ERR

   00020'177777               .PRI
   00021'151400               INC 2 2

   00022'034420 L:            LDA 3 .MESS
   00023'157000               ADD 2 3
   00024'021400               LDA 0 0 3
   00025'024407               LDA 1 COUNT
   00026'006017               .SYST
   00027'016400               .WRS 0
   00030'004405               JSR ERR
   00031'000771               JMP L

   00032'000010 PRIOR:        10
   00033'000022'NEWTASK:            L
   00034'000010 COUNT:        8.

   00035'000400 ERR:          JMP .

   00036'000076".TTO:         .+1*2
   00037'022124               .TXT /$TTO/
         052117
         000000

   00042'000043'.MESS:        .+1
   00043'000116"              MESS0*2
   00044'000130"              MESS1*2
   00045'000142"              MESS2*2
   00046'000154"              MESS3*2
   00047'052101 MESS0:        .TXT /TASK 1<15><12>/
         051513
         020061
```

Sample Assembly Language Program

```
 0002 TOT
              006412
              000000
   00054'052101 MESS1:   .TXT /TASK 2<15><12>/
              051513
              020062
              006412
              000000
   00061'052101 MESS2:   .TXT /TASK 3<15><12>/
              051513
              020063
              006412
              000000
   00066'052101 MESS3:   .TXT /TASK 4<15><12>/
              051513
              020064
              006412
              000000

                         .END TOT
```

Sample Assembly Language Program

```
 0003  TUT

COUNT  000034'         1/36     1/44
ERR    000035'         1/14     1/20     1/24     1/28     1/39     1/46
L      000022'         1/33     1/40     1/43
MESS0  000047'         1/54     1/58
MESS1  000054'         1/55     2/03
MESS2  000061'         1/56     2/08
MESS3  000066'         1/57     2/13
NEWTA  000033'         1/17     1/43
PRIOR  000032'         1/16     1/42
TUT    000000'  EN     1/04     1/05     1/10     2/19
.MESS  000042'         1/33     1/53
.PRI   000020'  XN     1/06     1/30
.TASK  000016'  XN     1/06     1/19     1/23     1/27
.TTO   000036'         1/10     1/48
```

Sample Assembly Language Program (Continued)

The relocatable binary produced by assembling this program can be loaded by either the stand-alone extended relocatable loader, the SOS loader, or the RDOS loader. In this illustration we show the dialogue that ensues when the program is loaded by the stand-alone loader.

After the loader is loaded, it self-starts and outputs the message "SAFE = ." A carriage return response causes the top 200 locations to be saved, preserving the binary loader. After this, the star prompt is output. The program relocatable binary (TOT), RTOSGEN module, and two RTOS libraries are then loaded. The RTOS module loaded with this program specifies 5 user tasks. Four user tasks are needed, and a system task is also required since use of the peripheral device is simultaneously requested by more than one user task. At the termination of loading, (*8), the initializer starting address, 376, is placed in the data switches. RESET, is then pressed, followed by START. The program is initialized and begins at entry TOT, outputting the task messages until STOP is pressed.

```
SAFE =
*2 TOT
*2 RTOS
*2
*2
*8TASK 1
TASK 2
TASK 3
TASK 4
TASK 1
TASK 2
TASK 3
TASK 4
TASK 1
TASK 2
TASK 3
TASK 4
TASK 1
TASK 2
TASK 3
TASK 4
TASK 1
TASK 2
TASK 3
TASK 4
TASK 1
TASK 2
TASK 3
```

<u>Load Dialogue and Program Output</u>

Real Time FORTRAN IV Programming

Since RTOS is a compatible subset of RDOS, RTOS will support a subset of DGC Real Time FORTRAN IV. To write a Real Time (RT) FORTRAN IV program for use with RTOS, you may use either the RDOS FORTRAN IV compiler or the 12K SOS FORTRAN IV compiler. Operating procedures for using these compilers are documented in the FORTRAN IV User's Manual, 093-000053, Appendix D. The only restriction on use of DGC RT FORTRAN IV under RTOS is that only those real time calls may be used which have corresponding system and task calls implemented in RTOS. Thus you must exclude use of the OVERLAY statement, and you may use all RT FORTRAN IV calls except the following: FCHAN, FOVLD, FOVRL, FSWAP, OVEXT, OVEXX, OVKIL, OVKIX, OVLOD, OVOPN, CFILW, DFILW, DIR, FSTAT, and WRITR. Only the following disk and tape I/O-related call are available: WRBLK and RDBLK for disk I/O, and MTOPD and MTDIO for magnetic tape.

Having produced one or more FORTRAN IV relocatable binaries, you use relocatable load procedures which are similar to those documented in this manual, Appendix B, for assembly language program binaries. The only addition to these procedures is the loading of the FORTRAN run time libraries, RTOSFMT. LB, FORT1. LB, FORT2. LB, FORT3. LB and the appropriate integer multiply/divide library. Thus the relocatable load sequence using either the stand-alone extended relocatable loader, the SOS relocatable loader, or the RDOS relocatable loader is as follows:

1. FORTRAN relocatable binaries (the program proper)
2. RTOS module produced by RTOSGEN.
3. RTOSFMT. LB (the RTOS real time FORTRAN IV run time library).
4. FORT1. LB
5. FORT2. LB
6. FORT3. LB
7. An integer multiply/divide library
8. RTOS1. LB
9. RTOS2. LB
10. Other RTOS libraries as required (RTOS MTA. LB, RTOS CAS. LB, RTOS DSK. LB, or RTOS DKP. LB)

After relocatable loading is complete, the FORTRAN program is started just as any other RTOS program is started. Details for executing an RTOS program are given in Appendix B.

*****

APPENDIX E

RTOS Source Level Incompatibilities

Certain features of RTOS revision 3 operate differently from the way they did in the previous revision. A summary of these differences follows.

1) An attempt to initialize or release a tape unit which has already been initialized will result in error ERIBS being reported, "device already initialized". Under revision 05, no error was reported, the normal return was taken, and the call performed no operation if the device was initialized.

2) Under revision 05, system calls .RTN and .ERTN idled the system by executing a "JMP . " in locations USTRL/USTEL of the User Status Table. In revision 3 these displacements of the User Status Table have been removed; execution of either of these calls causes the error return to be taken unconditionally.

3) The label of the entry point to the RTOS initialization routine has been changed from "INIT" to ".RTOS" .

*****

INDEX

.ABORT 3-1f, 6-5
.AKILL 1-2, 3-1f
.APPEND 2-2f, 2-6f
.ARDY 1-2, 3-1, 3-3
ASCII codes 2-14f
assembly language illustration D-1ff
.ASUSP 1-2, 3-1, 3-3
asynchronous data communications mux (see QTY)

BEGIN 6-1f

cassette tape
- data format 1-6f
- initialization (see .INIT) 1-6
- I/O (see .MTOPD, .MTDIO)

CBOOT 1-11, B-17f, B-22
channel 1-4 (see .GCHN)
characteristics inhibit mask 2-4f
.CHTB 6-2, 6-10
clock and calendar commands 2-26ff
close a file or device (see .CLOSE, .RLSE, or .RESET)
.CLOSE 2-2f, 2-7, 3-2
.CMSK 6-2f
.COMM TASK 1-10
command summary A-1ff
compatibility with RDOS 1-9ff
console interrupts (see .WCHAR)
CSP 6-2f
CTCB 6-2f

.DELAY 1-2, 2-2, 2-26
device control table (DCT) 4-1f, B-8, C-5
device file tables (see .DTBL, .PTBL, .QTBL, .MCTB)
device support under RTOS 1-5f
direct block I/O 2-3, 2-10, 2-16
disk file structure 1-8f, 2-1
DISMISS 6-2f
.DTBL 6-2, 6-6f
.DUCLK 2-2, 2-29

error message summary A-7f
.ERTN 2-2, 2-6

file and I/O system commands 2-1ff
formatting a disk B-1
FORTRAN IV D-6
free format I/O 2-3, 2-8f, 2-18ff

.GCHAR 2-2, 2-22
.GCHN 2-2, 2-6
.GDAY 2-2, 2-27
generating an RTOS system Appendix B
.GHRZ 2-2, 2-28f
.GMCA 2-2, 5-2f
.GTOD 2-2, 2-28

high priority interrupt devices 4-1, 4-5f, 6-8f, B-8
.HINT 6-2, 6-8f
HIPBOOT 1-11, B-15f
HMA 2-24f
Hollerith - ASCII translation 2-14f

.IDEF 2-1f, 4-2f, 4-5, B-8
idle the system 1-2, 2-25
.IDST 3-1, 3-3f
incompatibilities, source level E-1
.INIT 1-6, 2-2f, 2-8
interrupt table (.ITBL) 4-1, 4-5f, 6-2, 6-9
I/O modes (see direct block, line, sequential, free format)
.INTP 6-1
IOEND 6-2f
.IRMV 2-1f, 4-3f
.IXMT 2-29, 3-1, 3-4f, 4-2, 4-6

.KILL 3-1, 3-5

line I/O 2-3
loading an RTOS program B-12ff, D-5

magnetic tape
- data format 1-7, 1-9
- initializing (see .INIT) 1-6
- I/O (see .MTOPD, .MTDIO)

MCABOOT B-19
.MCTB 6-6
.MEM 2-2, 2-24
.MEMI 2-2, 2-24f

memory size commands 2-23ff
.MTDIO 2-2, 2-8, 2-18ff
.MTOPD 1-7, 2-2f, 2-8f
Multiprocessor Communications Adapter (MCA)
1-6, 2-4f, 2-17,
Chapter 5, 6-6, B-7, B-10

NMAX 1-3, 2-23f
no-ops 1-10

.OPEN 2-2f, 2-4f

panics 6-3
.PCHAR 2-2, 2-22
power fail 4-4f, B-8
.PRI 3-1, 3-5, D-1f
.PTBL 6-2, 6-6f

.QTBL 6-2, 6-6f
QTY 1-7f, 6-2, 6-6f

.RDB 1-4, 2-2f, 2-10
.RDL 2-2f, 2-11ff
.RDS 2-2f, 2-12f
.REC 3-1, 3-5f, 4-2
.RESET 2-2f, 2-7
RLOC 6-2f
.RLSE 1-6, 2-2f, 2-9f
RSCHED 6-2f
.RTN 2-2, 2-25
RTOS
    device support 1-5f
    organization Chapter 6
    parameters 1-4, Appendix C
RTOSGEN 2-1, 4-1f, 6-1, Appendix B, D-5
.RUCLK 2-2, 2-30

SCHED 6-2f
.SDAY 2-2, 2-27
sequential I/O 2-3
.SMSK 3-1, 4-3
standard device table (see .CHTB)
.STOD 2-2, 2-28
.SUSP 1-2, 3-1, 3-6
.SYS. 6-1f

System call
    descriptions Chapter 2, Chapter 4
    format 1-3ff, 1-9f
    equivalence 1-3, 1-5
    list 2-2
system generation (see RTOSGEN)

Task
    Concepts 1-1, Chapter 3
    format 1-3ff, 1-9f
    identification (see .TIDR, .TIDK, .TIDP, .TIDS, .TASK)
    scheduler 1-2f
    states 1-2
    status (see .IDST)
    synchronization 1-3
.TASK 1-2, 3-1, 3-6f, D-1
TBOOT 1-11, B-17f
TCB 1-1ff, Chapter 3, 6-1f, 6-5
Teletype commands 2-22f
.TIDK 1-2, 3-7
.TIDP 3-1, 3-7f
.TIDR 1-2, 3-1f, 3-8
.TIDS 1-2, 3-1f, 3-8
TLINK 6-2f
.TSAVE 6-2f

.UCEX 2-30, 3-1
.UIEX 3-1, 4-2f
.UPEX 3-1, 4-5
User File Pointers Table (.UFPT) 6-2, 6-6
User interrupts Chapter 4
User Status Table (UST) 6-2, 6-4
USP 6-2f
USTP 6-2

.WCHAR 2-2, 2-23, 6-1
.WRB 1-4, 2-2f, 2-16
.WRL 2-2f, 2-16f
.WRS 2-2f, 2-17f

.XMT 1-3, 3-1, 3-9
.XMTW 1-3, 3-1, 3-4, 3-9

cut along dotted line

DATA GENERAL CORPORATION
PROGRAMMING DOCUMENTATION
REMARKS FORM

DOCUMENT TITLE ________________________________

DOCUMENT NUMBER (lower righthand corner of title page) ______________

TAPE NUMBER (if applicable) ______________________

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

General Comments and Suggestions for Improvement of the Publication.

FROM: Name: ______________________ Date: __________
Title: ______________________
Company: ______________________
Address: ______________________
____

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: Programming Documentation

FOLD UP SECOND FOLD UP

STAPLE