NASA Technical Memorandum 78757

REFERENCE MANUAL FOR THE LANGLEY RESEARCH CENTER FLIGHT SIMULATION COMPUTING SYSTEM

JEFF I. CLEVELAND, II

DANIEL J. CRAWFORD

LAWRENCE F. ROWELL

JULY 1978

NASA

National Aeronautics and Space Administration

Langley Research Center
Hampton, Virginia 23665

REVISION RECORD

Revision	Description
- 1/76	Preliminary release.
A 7/78	First formal release. Revisions include technical and literary corrections. New features include subroutine RESUMEQ.

REVISION MARKINGS

New features, as well as changes, deletions, and additions to information in this manual are indicated by bars in the margins or by a dot near the page number if the entire page is affected. A bar by the page number indicates pagination rather than content has changed.

ACKNOWLEDGEMENT

The statement on the top of this page and Table 1, page 7 are reproduced with the permission of Control Data Corporation, Minneapolis, Minnesota.

TABLE OF CONTENTS

ENTRY POINT TABLE OF CONTENTS

INTRODUCTION

The real-time simulation system has been operating at Langley since 1967. It has been modified and improved many times since then. The latest modification includes an upgrading of the computers (6600s to Cyber 175s) and an upgrading to the Network Operating System (NOS). There are also several new innovations in the real-time hardware including a special real-time disk with its own dedicated peripheral processor. The NOS operating system has been modified to accommodate real-time simulation while simultaneously processing both interactive and batch jobs. These modifications have been made following the philosophy of minimum departure from the standard system so that the simulation software system can be more easily upgraded to future releases of NOS.

Real-Time Supervisor is a group of subprograms which are loaded with each simulation application program. The Supervisor provides the interface between the application program and the operating system and coordinates input and output to and from the simulation hardware. There are other simulation utilities which are not included in the Supervisor. The most notable of these are the graphics processors and the mode control subprogram which interfaces between the application program, the program control console, and the Supervisor. These processors are not discussed in this reference manual.

Since both the hardware and software environments of real-time simulation have changed since 1976, most of the utilities including Supervisor have been completely redesigned and reprogrammed.

This paper is a reference manual for the Real-Time Supervisor. As implementation progresses, this manual will be kept current.

The design of the Supervisor is modular and flexible and currently provides: communications with the Real-Time Monitor; interaction with the Digital/Analog Subsystem (DASS); communication to and from the real-time disk; interactive error processing; and interaction with the channel clock which provides time accounting and insures time schedule integrity. Future releases of Real-Time Supervisor will include more sophisticated error processing, a multi-frame rate per job capability, and other features as required by the user.

PROGRAM STATES

Persons familiar with the previous implementation of real-time simulation on the CDC 6600s may tend to confuse program states with the more familiar program modes. In the former system, there were only two program states -- synchronized real-time and HOLD. The program modes, Operate, Hold, Reset, and Idle, were executed in synchronized real-time; whereas such functions as copying the real-time disk file to a printer were done in the single HOLD state. With this new system, Real-Time Computing System (RTCS), four program states are supplied. The attributes of these states are described below and in Table 1.

HOLD3

When a job successfully executes a SKED** control statement to allocate real-time resources, it enters the HOLD3 program state. (A program may also enter HOLD3 from any other program state.) In this state, the job has high queue and CPU priority and occupies a real-time (high-numbered) control point which has immunity from automatic rollout. Rollout may occur only if (1) the job does time-sharing terminal input/output, (2) the job requests rollout, or (3) TELEX aborts or the terminal communication link is lost.

During the HOLD3 state, the job may communicate with the time-sharing terminal and the operating system in any manner. The job is not allowed to communicate with any real-time equipment (DASS, RTDISK, CRT, or ADAGE) during the HOLD3 state.

Synchronized Real-Time (SRT)

When a program is in this state, the strict synchronization with the

---

** See reference 2.

simulation real-time clock is maintained. At the beginning of each real-time frame, packed analog and discrete inputs are transferred to the Supervisor. At the end of the frame packed analog and discrete outputs, CRT outputs, RTDISK input/output and ADAGE output are transferred to the appropriate equipment as requested by the application program. In this state, normal communication with the operating system (such as file processing) is not allowed.

HOLD1

In this program state, the program may communicate with the real-time hardware (DASS, RTDISK, CRT, or ADAGE) and with the full NOS operating system except that input/output to the time-sharing terminal is not allowed and synchronization with the real-time clock is not maintained. The program runs asynchronously with the packed analog and discrete inputs being transferred each frame time, but the program is neither interrupted nor started; i.e., inputs from the DASS continue to be transferred each frame, but the program is essentially unaware of the beginning of every new frame. Output requests to real-time hardware may be done at any time. Note that input/output to local files, permanent files, and other equipment except the time-sharing terminal is allowed.

HOLD2

This program state is identical to HOLD1 except that analog and discrete inputs are not transferred, effectively "holding" these input values to the last time they were transferred. This state is intended to be used as a diagnostic aid.

Supervisor's functions include self-initialization, coordination of I/O to and from the special hardware, packing and unpacking analog channels, passing buffer addresses to tables used by the simulation processors, and communication with the Real-Time Monitor.

When a job is scheduled (via a SKED** control statement) into a real-time control point, it will be in the HOLD3 state. The simulation engineer can work interactively (such as edit, compile, etc.) with the computer in this state until such time that he is ready to execute his real-time program. When he begins execution, the program should immediately initialize itself by a call to RTINIT. The Supervisor initially sets all harware input/output arrays to zero. The program can then change states as is necessary by calls to RTHOLD1, RTHOLD2, HOLD3, or RTSRT. A call to any of the HOLD states is ignored if the program is already in that state; however, a call to RTSRT while the program is in SRT is an error condition.

The calls to RTCYCLE, RTIDLE, and RTCLEAR are essentially requests for I/O to the special real-time hardware. The difference between RTCYCLE and RTIDLE is that RTIDLE only affects discrete I/O; whereas, RTCYCLE initiates I/O for the DASS, RTDISK, and/or RTGRAPHICS subsystems. RTCLEAR is used for calibration of external equipment. It sends zero out on all assigned analog and discrete channels. In addition, if the program is in the SRT state, these calls cause the release of the central processor. Since these calls mark the end of a real-time cycle, one of them should be called once each frame before expiration of the required compute time (RCT).

A call to FINI terminates execution of the program and enables the interactive terminal so that the engineer can do any necessary processing such as program editing, nonreal-time graphics, compilation, and assemblies. At this point, he can either go back to real-time processing, or deallocate his resources by a call to SKED**. A brief summary is followed by a detailed description of the Supervisor program state calls.

CALL	FUNCTION
RTINIT	INITIALIZE FOR REAL TIME OPERATIONS
RTSRT	ENTER SYNCHRONIZED REAL TIME
RTHOLD1	ENTER HOLD1 STATE
RTHOLD2	ENTER HOLD2 STATE
HOLD3	ENTER HOLD3 STATE
RTCYCLE	END REAL TIME FRAME
RTIDLE	END REAL TIME FRAME
RTCLEAR	CLEAR REAL TIME OUTPUTS
FINI	ENTER TERMINATION PROCESS

** See reference 2.

Program Name / Characteristic	NON-REAL-TIME			REAL-TIME		
	BATCH	INTERACTIVE	HOLD3	HOLD2	HOLD1	SYNCHRONIZED REAL-TIME (SRT)
Q Priority	Standard NOS (low)	Standard NOS (medium)	Real-Time (high)	Real-Time (high)	Real-Time (high)	Real-Time (high)
CPU Priority	Standard NOS (low)	Standard NOS (low)	High Priority Batch	High Priority Batch	High Priority Batch	Real-Time (Synchronized)
Storage Moves	Yes	Yes	Yes, if not locked out by other RT jobs	No	No	No
Rollouts	Yes	Yes	Yes	No	No	No
Entry	Local or Remote card reader	TTY terminal	1. Call to SKED to allocate RT resources. 2. Program request from HOLD1, HOLD2, or SRT	1. Program request from HOLD3, HOLD1 or SRT	1. Program request from HOLD3, HOLD2, or SRT	1. Program request from HOLD3, HOLD2, or HOLD1
Exit (Assume total use of interrupt features)	Drop from control point	Log OFF	1. Call SKED to deallocate resources 2. Log off (interactive) 3. Drop from C. P. (Batch) 4. HOLD1, HOLD2, or SRT	1. Log off (Go to HOLD3) 2. End in RA+1 (HOLD3) 3. Program request for HOLD3, HOLD , or SRT	1. Log off (HOLD3) 2. End in RA+1 (HOLD3) 3. Program request for HOLD3, HOLD , or SRT	1. Log off (HOLD3) 2. End in RA+1 (HOLD3) 3. Program request for HOLD3, HOLD2, or HOLD1
Terminal I/O	No	Yes	Yes	No	No	No
RT Subsystem I/O except ADDIS	No	No	No	Yes	Yes	Yes
ADDIS inputs	No	No	No	No	Yes	Yes
Mini-CP Area and RT resource allocation	No	No	Yes	Yes	Yes	Yes
RTM Responds to Real-Time Starts	No	No	No	No	No	Yes

Table 1

PURPOSE: To initialize the Real-Time Supervisor and to establish ADC and DAC communication areas.

USE: Format 1 CALL RTINIT (ID, ADC, NADC, DAC, NDAC)

Format 2 CALL RTINIT (ID, ADC, NADC, DAC, NDAC), RETURNS ($ERROR)

ID one word Hollerith constant (10 characters maximum) containing the identification of the simulation program for messages to the mainframe console operator; e.g., "DMS-F11" or 10H TAIL SPIN.

ADC array at least NADC words long into which Supervisor will store the scaled floating point ADC values.

NADC number of ADC input channels to be unpacked and floated. May be 0.

DAC array at least NDAC words long from which Supervisor will take the scaled DAC values.

NDAC number of DAC output channels to be packed and unfloated. May be 0.

$ERROR statement number to which control is returned if an abnormal condition is detected. See error processing section for usage.

RESTRICTIONS:

1. RTINIT must be the first call to the Supervisor and should be the first executable statement in the program.
2. RTINIT may not be called in SRT.
3. The amount of packed analog data transferred to and from the DASS

equipment is determined by the SKED control statement sequence, whereas the number of channels unpacked (ADC's) or packed (DAC's) by the Supervisor is determined by RTINIT. The conversion is between a 60 bit floating point word in the CYBER mainframe and a 15 bit fixed point word in the analog subsystem.

DESCRIPTION: The first call to RTINIT causes the Supervisor to initialize itself. This includes establishing terminal I/O buffers, error recovery addresses, and initialization of the standard error recovery package. Colons (00B display) are removed from ID before it is placed in SIMID in common /SUPCOMM/. The ADC and DAC array addresses and lengths are retained for communication with the DASS hardware but DASS input/output is not performed. RTINIT may be called more than once if the array addresses or lengths are to change.

ERROR MESSAGES:

1. CALLED IN WRONG PROGRAM STATE (NERROR = 2)

   RTINIT may not be called in SRT.

2. NO. OF ADC OR DAC .LT. 0 (NERROR = 124)

   NADC or NDAC is less than 0.

3. BAD STATUS FROM URT - RTS FUNCTION (NERROR = 202)

   URT STATUS = _ _ _

   This error should never occur, and it indicates a failure of the Supervisor or the real-time system. An analyst should be informed of the problem.

PURPOSE: To enter synchronized real-time state from a nonsynchronized state.

USE: Format 1 CALL RTSRT

Format 2 CALL RTSRT, RETURNS ($ERROR)

$ERROR statement number to which control is returned if an abnormal condition is detected. See error processing section for usage.

RESTRICTIONS:

1. The program must be in a nonsynchronized real-time state before CALL RTSRT.
2. An RTQP request must have been included in the SKED resource requests.

DESCRIPTION:

1. The program state is changed to synchronized real-time (SRT) from a nonsynchronized state (HOLD1, HOLD2, or HOLD3).
2. If an ADC array was specified in a call to RTINIT, then the ADC inputs are unpacked, converted to floating point format, and placed in the ADC array.
3. The discrete inputs are available.
4. Supervisor returns control to the calling program.

ERROR MESSAGES:

1. CALLED IN WRONG PROGRAM STATE (NERROR = 2)

   RTSRT was called in SRT (must be in a HOLD state).

2. REAL-TIME HAS NOT BEEN INITIALIZED (NERROR = 1)

   No previous call to RTINIT.

3. REAL-TIME HARDWARE FAILURE (NERROR = 3)

   _____ STATUS = _____

Hardware error detected in DASS equipment. Continued DASS operation may give unpredictable results.

4. PPU CALL ERROR IN _ _ _ _

   RA+1 = 2522150100...XXX PPU=URM

   The job did not have an RTQP resource request.

PURPOSE: To enter the HOLD1 state.

USE: Format 1 CALL RTHOLD1

Format 2 CALL RTHOLD1, RETURNS ($ERROR)

$ERROR statement number to which control is returned if an abnormal condition is detected. See error processing section for usage.

RESTRICTIONS: None.

DESCRIPTION: If the prgram is in SRT Supervisor issues a URM01** monitor request to complete the SRT frame. Supervisor issues a URM02** monitor request to change the state. The job is a high priority (CPU and QUEUE) nonsynchronous program. The job has access to RTDISK, CRT, ADAGE, and DASS hardware. (If a program is in the HOLD1 state and it makes this call, the call is treated as a no-op.) Discrete inputs are being refreshed by the hardware at the frame rate. See Table 1 for definition of HOLD1 state. Control is returned to the calling program.

ERROR MESSAGES:

1. REAL-TIME HAS NOT BEEN INITIALIZED (NERROR = 1)

   No previous call to RTINIT.

** See reference 2.

PURPOSE: To enter the HOLD2 state.

USE: Format 1 CALL RTHOLD2

Format 2 CALL RTHOLD2, RETURNS ($ERROR)

$ERROR statement number to which control is returned if an abnormal condition is detected. See error processing section for usage.

RESTRICTIONS: None.

DESCRIPTION: If the program is in SRT Supervisor issues a URM01** monitor request to the complete the frame. Supervisor issues a URM03** request to Real-Time Monitor. The job is a high priority (CPU and QUEUE) nonsynchronous program. The job has access to the real-time disk, CRT, ADAGE, and DASS (outputs only); however, the analog and discrete inputs are frozen to their values at the time that HOLD2 was entered. (If a program is in the HOLD2 state and it makes this call, the call is treated as a no-op.) See Table 1 for definition of HOLD2 state.

ERROR MESSAGES:

1. REAL-TIME HAS NOT BEEN INITIALIZED (NERROR = 1)

   No previous call to RTINIT.

** See reference 2.

PURPOSE: To enter the HOLD3 state.

USE: Format 1 CALL HOLD3

Format 2 CALL HOLD3, RETURNS ($ERROR)

$ERROR statement number to which control is returned if an abnormal condition is detected. See error processing section for usage.

RESTRICTIONS: None.

DESCRIPTION: Supervisor issues a URM04** monitor request to Real-Time Monitor. The job is a high priority (CPU and QUEUE) nonsynchronous program. In the HOLD3 state, the job is subject to being rolled in and out and storage moves can be affected. All TELEX interactive work must be done in this state. The RTDISK, CRT, ADAGE, and DASS hardware cannot be accessed in this state.

ERROR MESSAGES:

1. REAL-TIME HAS NOT BEEN INITIALIZED (NERROR = 1)

   No previous call to RTINIT.

** See reference 2.

PURPOSE: It signifies that all user processing for the frame is complete. RTCYCLE preserves the program state from which it was called (i.e., HOLD1, HOLD2, or SRT) and causes all real-time I/O to be performed.

USE: Format 1 CALL RTCYCLE

Format 2 CALL RTCYCLE, RETURNS ($ERROR)

$ERROR statement number to which control is returned if an abnormal condition is detected. See error processing section for usage.

RESTRICTIONS: This call is not allowed in HOLD3.

DESCRIPTION:

1. Control branches from the application program to Supervisor.
2. If either disk input or disk output is required (i.e., one or both of the file buffers need to be stored or refreshed), then the RTDISK flag will be set in the appropriate Real-Time Monitor request bit.
3. If any data is contained in the CRT buffers, then the RTCRT flag will be set in the appropriate Real-Time Monitor request bit.
4. The DASS flag will be set in the appropriate Real Time Monitor request bit. If a DAC buffer was specified in a call to RTINIT, then these words will be converted to DASS format and packed four to a central memory word in an output buffer.
5. A URM01** monitor request is issued and the central processor is returned to Real-Time Monitor.

** See reference 2.

6. Monitor will signal the necessary peripheral processors and they will accomplish the transfer of data to and from their respective hardware; RTDISK, CRT, and DASS, including analog and discrete input/output.
7. In due time, Real-Time Monitor returns the central processor to the Supervisor.
8. If an ADC buffer was specified in a call to RTINIT, then the input buffer which was filled by the DASS input peripheral processor will be unpacked, converted to floating point format, and placed in the ADC buffer (unless in HOLD2).
9. New discrete inputs are available.
10. Supervisor returns control to the applications program.

ERROR MESSAGES:

1. CALLED IN WRONG PROGRAM STATE (NERROR = 2)

   Called in HOLD3.

2. REAL-TIME HAS NOT BEEN INITIALIZED (NERROR = 1)

   No previous call to RTINIT.

3. REAL-TIME HARDWARE FAILURE (NERROR = 3)

   _____ STATUS = _____

   Hardware error detected in DASS equipment. Continued DASS operation may give unpredictable results.

PURPOSE: It signifies that all user processing for the frame is complete. RTIDLE preserves the program state from which it was called (i.e., HOLD1, HOLD2, or SRT). RTIDLE causes the discrete output data from the discrete buffer to be sent out to the DASS hardware and the discrete inputs to be read into the discrete input central buffer.

USE: Format 1 CALL RTIDLE

Format 2 CALL RTIDLE, RETURNS ($ERROR)

$ERROR statement number to which control is returned if an abnormal condition is detected. See error processing section for usage.

RESTRICTIONS: This call is not allowed in HOLD3.

DESCRIPTION:

1. Control branches from application program to Supervisor.
2. The DASS I/O bit will be set in the appropriate Real Time Monitor request word. However, the user's DAC buffer will not be packed, formatted, nor transferred to its associated output buffer.
3. A URM01** monitor request is issued and the central processor is returned to Real-Time Monitor.
4. Monitor will signal the DASS peripheral processor and data will be transferred to and from the analog hardware.
5. In due time, Real Time Monitor returns the central processor to the Supervisor.
6. Supervisor returns control to the application program.

** See reference 2.

ERROR MESSAGES:

1. CALLED IN WRONG PROGRAM STATE (NERROR = 2)

   Called in HOLD3.

2. REAL TIME HAS NOT BEEN INITIALIZED (NERROR = 1)

   No previous call to RTINIT.

3. REAL-TIME HARDWARE FAILURE (NERROR = 3)

   _____ STATUS = _____

   Hardware error detected in DASS equipment. Continued DASS operation may give unpredictable results.

PURPOSE: It signifies that all user processing for the frame is complete. It preserves the program state from which it was called (i.e., HOLD1, HOLD2, or SRT). It places all zeros in Supervisor's packed DAC buffer and transmits these to the Real-Time Subsystem. It also places all zeros in the user's packed discrete buffer and transmits the buffer to the discrete output hardware. If RTDISK or RTCRT I/O is required, it is done also. It causes the input discretes to be read into central memory along with the ADC inputs which are also unpacked and placed into the user's buffer. This differs from the previous Supervisor in that the user must reset his output discretes after coming out of RTCLEAR (i.e., discretes are not saved).

USE: Format 1 CALL RTCLEAR

Format 2 CALL RTCLEAR, RETURNS ($ERROR)

$ERROR statement number to which control is returned if an abnormal condition is detected. See error processing section for usage.

RESTRICTIONS: This call is not allowed in HOLD3.

DESCRIPTION:

1. Control branches from application program to Supervisor.
2. Supervisor's packed DAC buffer is set to zero.
3. User's packed discrete output buffer is set to zero.
4. The DASS flag (and possibly the RTDISK or RTCRT flags) will be set in the URM01** monitor request word and the request will be issued.

** See reference 2.

5. The zeroed outputs are transmitted to the DASS and the inputs are brought into central memory in time for the next frame.
6. In due time, Monitor returns the central processor to the Supervisor.
7. Supervisor returns control to the applications program.

ERROR MESSAGES:

1. CALLED IN WRONG PROGRAM STATE (NERROR = 2)

   Called in HOLD3.

2. REAL TIME HAS NOT BEEN INITIALIZED (NERROR = 1)

   No previous call to RTINIT.

3. REAL-TIME HARDWARE ERROR (NERROR = 3)

   _____ STATUS = _____

   Hardware error detected in DASS equipment. Continued DASS operation may give unpredictable results.

PURPOSE: To terminate a real-time simulation program.

USE: CALL FINI

RESTRICTIONS: None.

DESCRIPTION:

1. Nontime-sharing origin job.
   a. The program is placed in HOLD3 state.
   b. The program is terminated.
2. Time-sharing origin job.
   a. The program is placed in HOLD3 state.
   b. The standard error recovery package is called and various options are available for time-sharing terminal operation. See error processing section for usage.

ERROR MESSAGES:

1. REAL TIME HAS NOT BEEN INITIALIZED. (NERROR = 1)

   No previous call to RTINIT.

REAL TIME DATA MANAGEMENT

Each of the two CYBER 175s have the following disk file features:

1. access to a large permanent file system
2. access to a pool of disks which is used for local files and system files
3. access to a high performance real-time disk which is capable of storing or retrieving a total of 282 central memory words every 1/32 of a second.

It is this third feature that is discussed in this section. Each real-time job can have as many as two real-time files. The performance will be shared among all the real-time jobs on a single computer. If equally distributed, each of three jobs could store or retrieve as many as 94 words each 1/32 of a second.

Files on the real-time disk are scratch files and will disappear when a job deallocates its real-time resources. Simulation input data would likely be brought from the permanent file system to the local file pool and then be written on the RTDISK for later real-time access. Conversely, simulation output data would be written to the RTDISK in real-time and later copied to the local disk pool. From there, it might be routed to a printer, tape, punch, or to permanent file storage. One anticipated application of the RTDISK is to record a pilot's control inputs during a simulated flight and then to play back the flight by driving the simulation with the recorded inputs.

Since there is a limited amount of space on the disk, it must be allocated among the real-time jobs on the computer. The mechanism for doing this is contained in Static Scheduler (SKED) and the RTDM control card which has the following format:

RTDM (size 1, size 2)

The parameters of this card specify how much space (in tracks) the user wants for each of his files. If the space is available, Static Scheduler will assign it to his job. There are 1,616 tracks on the 844 disk, and each track can accommodate 6,848 central memory words. There is a possibility of upgrading to a double density 844 disk in which case available space will double.

The Supervisor manages the real-time data storage and retrieval capability. It creates and maintains the necessary file environment tables and maintains the other interfaces with the Real-Time Monitor and the RTDISK peripheral processor program. Supervisor does not rewind the real-time files without a specific directive (REWRTF) from the user. There are two types of data sections on the real-time files. The first contains the central memory addresses of the data on the file. These addresses are provided by the user's calls to FILEVAR. The second type is the data itself. This implementation allows the user to dynamically control which variables he is recording and to subsequently read the data back into the correct central memory addresses. The table of recorded variables can be increased dynamically by calls to FILEVAR or reinitialized by a call to NEWVAR.

The real-time files can only be accessed in the SRT, HOLD1, or HOLD2 states. File buffer sizes are left to the option of the user (FILEARR). Generally, increasing buffer size will increase performance at the cost of increased field length. Other calls provided by Supervisor are a skip logical record function (SKPRTR) and dispose to printer or punch function (DISPOSE). The DISPOSE call is the only one in this section which does not directly pertain to the real-time files.

A brief summary of the Supervisor data management calls is followed by a detailed list and some instructive examples.

CALL	FUNCTION
FILEVAR	ESTABLISH VARIABLE ADDRESSES
NEWVAR	REINITIALIZE VARIABLE TABLE
FILEARR	ESTABLISH BUFFER AND VARIABLE TABLE SIZE
RTWRITE	WRITE RECORD
RTREAD	READ RECORD
DISPOSE	SEND LOCAL FILE TO PRINTER OR PUNCH
SKPRTR	SKIP RECORDS ON RTDISK FILE

PURPOSE: To establish memory locations for both the file buffer and the variable address table.

USE: Format 1 CALL FILEARR (IFN, BUFFER, NBUFF, VARTAB, NTAB)

Format 2 CALL FILEARR (IFN, BUFFER, NBUFF, VARTAB, NTAB), RETURNS ($ERROR)

IFN integer file number (1 or 2) indicating for which file the arrays are being specified.

BUFFER array name of a block of memory to be used by Supervisor as a file buffer.

NBUFF number of words in buffer (should be the dimension of BUFFER) and should be a multiple of 64 plus one (i.e., 257, 1025, etc.).

VARTAB array name of a block of memory to be used by Supervisor as a table of variable addresses to control the transferring of data between the specified file and the program.

NTAB number of words in VARTAB (should be the dimension of VARTAB). NTAB must be at least the sum of the maximum number of variables plus the number of arrays specified in calls to FILEVAR.

$ERROR statement number to which control is returned if an abnormal condition is detected. See error processing section for usage.

RESTRICTIONS:

1. FILEARR must be called before the first activity on a real-time disk file.
2. For access to both files, FILEARR must be called twice with unique array names.

3. A real-time file must be rewound before a change in buffer array or variable table array. This assures flushing of the buffer.
4. FILEARR may not be called in SRT.

DESCRIPTION: By specifying different buffer sizes, the performance limit of the simulation disk can be affected. For this reason, the burden of specification of buffer size is given to the programmer. A program with a high recording frequency and a high number of recorded variables will require a large buffer and conversely a program with a low recording frequency and a low number of recorded variables will require a small buffer.

In order to not restrict the record size for a real-time file record, the programmer must supply the memory to hold the variable address table which controls the transfer of data between the program and the file buffer. Note that one location is required for each simple variable and only one location for each array specified.

ERROR MESSAGES:

1. BUFFER SIZE LESS THAN MINIMUM (NERROR = 101)

   NBUFF specification is too small to allow reasonable disk performance. Must be greater than 128 and greater than twice the size of VTAB.

2. FILE n NOT REWOUND (NERROR = 103, 104)

   File must be rewound before new specifications are made.

3. ILLEGAL FILE NUMBER (NERROR = 4)

   File number is not 1 or 2.

4. ILLEGAL TABLE SIZE (NERROR = 102)

   NTAB must be greater than 0.

5. REAL TIME HAS NOT BEEN INITIALIZED (NERROR = 1)

   No previous call to RTINIT.

6. CALLED IN WRONG PROGRAM STATE (NERROR = 2)

   May not be called in SRT.

7. NO RTDM FILES ALLOCATED (NERROR = 125)

   RTDM files not requested when SKED executed.

8. FILE 2 NOT ALLOCATED (NERROR = 126)

   IFN = 2 and RTDM SKED request did not request allocation for file 2.

9. BAD STATUS FROM URT - FET FUNCTION (NERROR = 203)

   STATUS = _ _ _

   This error should never occur and it indicates a failure of the Supervisor or the real-time system. An analyst should be informed of the problem.

PURPOSE: To establish a table of variable addresses for communication with simulation disk.

USE: Format 1 CALL FILEVAR (IFN, V1, V2, . . .)

Format 2 CALL FILEVAR (IFN, V1, V2, . . .), RETURNS ($ERROR)

IFN	integer file number (1 or 2) indicating for which file variables are being specified.
Vi	1 to 62 simple variable names. See Format 3 and Format 4 below for array referencing. (Note that FILEVAR may be called more than once to establish a variable table.)
$ERROR	statement number to which control is returned if an abnormal condition is detected. See error processing section for usage.

RESTRICTIONS:

1. If a specified variable is double precision, then only the more significant word is accessed.
2. If a specified variable is complex, then only the real part is accessed.
3. FILEVAR may be called only enough times to fill the variable table as specified by VARTAB and NTAB in FILEARR.
4. Must be called before any calls to RTWRITE.
5. Extreme caution must be used when calls related to a real-time disk file are used in multiple overlays. The variables must be in COMMON or in a common overlay.

DESCRIPTION: Each time FILEVAR is called, the specified set of variables is added to the variable address table for the specified file. This table is used to control the transfer of data records on subsequent calls to RTWRITE. FILEVAR may be called in any program state so that the size of a simulation disk record may vary as it is written. NEWVAR may be called to establish a new table (see SUBROUTINE NEWVAR) while data is being written.

ERROR MESSAGES:

1. ILLEGAL FILE NUMBER (NERROR = 4)

   File number is not 1 or 2.

2. TOO MANY VARIABLES SPECIFIED (NERROR = 120)

   Attempt to exceed variable table as specified from FILEARR.

3. REAL TIME HAS NOT BEEN INITIALIZED (NERROR = 1)

   No previous call to RTINIT

4. FILE n NOT INITIALIZED (NERROR = 5, 6)

   Failure to call FILEARR

5. BUFFER TOO SMALL (NERROR = 132)

   Buffer specified in FILEARR is not large enough to hold variable table and data.

USE: Format 3 CALL FILEVAR (IFN, ARRAY1, N1, ARRAY2, N2, . . .)

Format 4 CALL FILEVAR (IFN, ARRAY1, N1, ARRAY2, N2, . . .), RETURNS ($ERROR)

IFN negative file number (-1 or -2) indicating which file arrays are for.

ARRAYi name of array to be saved. (Maximum of 31 arrays per call.)

Ni number of words in ARRAYi (DIMENSION).

$ERROR statement number to which control is returned if an abnormal condition is detected. See error processing section for usage.

RESTRICTIONS:

1. If the array is multi-dimensional, Ni should be the product of the dimensions.
2. If an array is double precision or complex, the number of words specified (Ni) should be twice that specified in the dimensionality statement.
3. FILEVAR may be called only enough times to fill the variable table as specified by VARTAB and NTAB in FILEARR.
4. Must be called before any calls to RTWRITE.
5. Extreme caution must be used when calls related to a real-time disk file are used in multiple overlays. The variables must be in COMMON or in a common overlay.

DESCRIPTION: Each time FILEVAR is called, the specified set of arrays and lengths is added to the variable address table for the specified file. Note that each array reference requires only one word in the variable table. FILEVAR may be called in any program state so that the size of a simulation disk record may vary as it is written. NEWVAR may be called to establish a new table (see SUBROUTINE NEWVAR in the section) while data is being written.

ERROR MESSAGES:

1. ILLEGAL FILE NUMBER (NERROR = 4)

   File number is not -1 or -2.

2. TOO MANY VARIABLES SPECIFIED (NERROR = 120)

   Attempt to exceed variable table as specified from FILEARR.

3. REAL TIME HAS NOT BEEN INITIALIZED (NERROR = 1)

   No previous call to RTINIT.

4. FILE n NOT INITIALIZED (NERROR = 5, 6)

   Failure to call FILEARR.

5. ARRAY LENGTH NOT SPECIFIED (NERROR = 122)

   No corresponding Ni for ARRAYi.

6. BAD ARRAY LENGTH (NERROR = 123)

   Length is 0 or negative.

7. NEGATIVE UNUSED TABLE ENTRY COUNT (NERROR = 201)

   This error should never occur and indicates that either the Supervisor contains a logic flaw or that the data buffer has been inadvertantly overwritten. An analyst should be informed of this problem.

8. BUFFER TOO SMALL (NERROR = 132)

   Buffer specified in FILEARR is not large enough to hold variable table and data.

PURPOSE: To erase a file variable table as established by FILEVAR.

USE: Format 1 CALL NEWVAR (IFN)

Format 2 CALL NEWVAR (IFN), RETURNS ($ERROR)

IFN integer file number (1 or 2) indicating which file variable table is to be erased.

$ERROR statement number to which control is returned if an abnormal condition is detected. See error processing section for usage.

RESTRICTIONS:

1. FILEVAR must be called before any subsequent calls to RTWRITE.
2. NEWVAR should be called during a write sequence to reinitialize the table of recorded variables or after the file has been rewound. It should not be called during a read sequence.

DESCRIPTION: NEWVAR is called when it is desired to establish a new variable address table for a file. NEWVAR resets the table size to zero so that subsequent FILEVARs can establish a new variable table. Note that NEWVAR may be called in real-time so that dynamically changing read/write requirements may be accommodated. A variable table does not exist after a CALL NEWVAR.

ERROR MESSAGES:

1. ILLEGAL FILE NUMBER (NERROR = 4)

   File number is not 1 or 2.

2. REAL TIME HAS NOT BEEN INITIALIZED (NERROR = 1)

   No previous call to RTINIT.

3. FILE n NOT INITIALIZED (NERROR = 5, 6)

   Failure to call FILEARR.

4. CALLED DURING READ OPERATIONS - FILE n (NERROR = 129, 130)

NEWVAR was called when last operation on file was a read.

PURPOSE: To initially fill a buffer before the first transfer of data to the program in SRT by RTREAD.

USE: Format 1 CALL READBUF (IFN)

Format 2 CALL READBUF (IFN), RETURNS ($ERROR)

IFN integer file number of file to be read.

$ERROR statement number to which control is returned if an abnormal condition is detected. See error processing section for usage.

RESTRICTIONS:

1. READBUF may only be called in HOLD1 or HOLD2.
2. The file must have been previously written by RTWRITE.

DESCRIPTION: In order to read a file in real-time, the data must be in the buffer before it can be transferred to the simulation program. Therefore, READBUF or SKPRTR must be called prior to the first RTREAD to read the first buffer of data from the disk. Once RTREAD is called, RTREAD will continue to refresh the buffer as necessary.

ERROR MESSAGES:

1. ILLEGAL FILE NUMBER (NERROR = 4)

   File number is not 1 or 2.

2. FILE n NOT INITIALIZED (NERROR = 5, 6)

   Failure to call FILEARR.

3. CALLED IN WRONG PROGRAM STATE (NERROR = 2)

   State is not HOLD1 or HOLD2.

4. FILE n EMPTY (NERROR = 105, 106)

   No data written on file.

5. I/O SEQUENCE ERROR FOR FILE n (NERROR = 13, 14)

   Attempt to read a file when last request was a write. File must be rewound first.

6. REAL TIME HAS NOT BEEN INITIALIZED (NERROR = 1)

   No previous call to RTINIT.

7. VARIABLE TABLE TOO SMALL (NERROR = 121)

   File contains variable table that exceeds size of variable table specified in last call to FILEARR.

8. PARITY ERROR DETECTED ON FILE n (NERROR = 133, 134)

   Unrecoverable disk parity error. An analyst should be informed of the problem.

9. DISK HARDWARE ERROR (NERROR = 208)

   STATUS = _ _ _

   This error should never occur and it indicates that an RTDM response code was not recognized. An analyst should be informed of the problem.

PURPOSE: To read one set of program variables from a simulation disk in real time.

USE: Format 1 CALL RTREAD (IFN, LD), RETURNS ($END)

Format 2 CALL RTREAD (IFN, LD), RETURNS ($END, $ERROR)

IFN integer file number (1 or 2) indicating which file is to be read.

LD 0 returned indicates data transferred.

-1 returned indicates lost data gap (see RTWRITE).

$END statement number to which control is returned if end of file is encountered.

$ERROR statement number to which control is returned if an abnormal condition is detected. See error processing section for usage.

RESTRICTIONS:

1. RTREAD may not be called in HOLD3.
2. A buffer and variable address table space must have been previously established for this file number by a call to FILEARR.
3. A call to READBUF or SKPRTR must be called in HOLD1 or HOLD2 before the first call to RTREAD in SRT.
4. CAUTION: An optimizing compiler may optimize this call out of order since the variables are not specified in the call.

DESCRIPTION: Each time RTREAD is called one set of previously recorded variables is placed in the corresponding variable addresses. If the file being read contains gaps due to lost data conditions (from RTWRITE), then no

data will be transferred until the gap has been passed. (LD will be set to -1 during the gap.)

ERROR MESSAGES:

1. ILLEGAL FILE NUMBER (NERROR = 4)

   File number is not 1 or 2.

2. CALLED IN WRONG PROGRAM STATE (NERROR = 2)

   May not be called in HOLD3.

3. FILE n NOT INITIALIZED (NERROR = 5,6)

   Failure to call FILEARR.

4. LOST DATA DETECTED FOR FILE n (NERROR = 9,10)

   Exceeded data transfer capacity of simulation disk system (in SRT only).

5. BUFFER EMPTY (NERROR = 119)

   READBUF or SKPRTR must be called before the call to RTREAD.

6. REAL TIME HAS NOT BEEN INITIALIZED (NERROR = 1)

   No previous call to RTINIT.

7. VARIABLE TABLE TOO SMALL (NERROR = 121)

   File being read contains variable table that exceeds the size of the variable table specified in last call to FILEARR.

8. I/O SEQUENCE ERROR FOR FILE n (NERROR = 13,14)

   Attempt to read a file when last operation was a write. File must be rewound first.

9. UNRECOGNIZABLE FILE HEADER (NERROR = 205)

   This error should never occur and it indicates that Supervisor did not find a specially coded file header in a real-time file. An analyst should be informed of the problem.

10. PARITY ERROR DETECTED ON FILE n (NERROR = 133,134)

Unrecoverable disk parity occurred. An analyst should be informed of the problem.

11. DISK HARDWARE ERROR (NERROR = 208)

STATUS = _ _ _

This error should never occur and it indicates that a RTDM response code was not recognized. An analyst should be informed of the problem.

12. BUFFER TOO SMALL (NERROR = 132)

Buffer specified in FILEARR is not large enough to hold variable table and data.

PURPOSE: To write one set of program variables to the simulation disk in SRT.

USE: Format 1 CALL RTWRITE (IFN)

Format 2 CALL RTWRITE (IFN), RETURNS ($ERROR)

IFN integer file number (1 or 2) indicating which file is to be written.

$ERROR statement number to which control is returned if an abnormal condition is detected. See error processing section for usage.

RESTRICTIONS:

1. RTWRITE may not be called in HOLD3.
2. A buffer and variable address table space must have been previously established for this file number by a call to FILEARR.
3. A table of variables must have been previously established for this file number by one or more calls to FILEVAR.
4. CAUTION: An optimizing compiler may optimize this call out of order since the variables are not specified in the call.

DESCRIPTION: Each time RTWRITE is called, one set of program variables, as established by calls to FILEVAR for the file number, is written to the simulation disk file. Note that the variable address table, as established by calls to FILEVAR, may be changed while the file is being written. Additional calls to FILEVAR will lengthen the table, while a CALL NEWVAR, followed by calls to FILEVAR, will establish a new table. If the variable address table is altered, the new table is recorded before the data on the file so that when the file is read, the data is transferred to the same addresses from which it was recorded.

ERROR MESSAGES:

1. ILLEGAL FILE NUMBER (NERROR = 4)

   File number is not 1 or 2.

2. CALLED IN WRONG PROGRAM STATE (NERROR = 2)

   May not be called in HOLD3.

3. FILE n NOT INITIALIZED (NERROR = 5,6)

   Failure to call FILEARR.

4. NO VARIABLES FOR FILE n (NERROR = 7,8)

   FILEVAR had not been called to establish variable addresses for data transfer.

5. LOST DATA DETECTED FOR FILE n (NERROR = 9,10)

   Exceeded data transfer capacity of simulation disk system.

6. ATTEMPT TO EXCEED ALLOCATED FILE SPACE, FILE n (NERROR = 11,12)

   Attempt to use more disk space than allocated through SKED.

7. REAL TIME HAS NOT BEEN INITIALIZED (NERROR = 1)

   No previous call to RTINIT.

8. I/O SEQUENCE ERROR FOR FILE n (NERROR = 13,14)

   Attempt to write a file when last operation was a read. File must be rewound first.

9. PARITY ERROR DETECTED ON FILE n (NERROR = 133,134)

   Unrecoverable disk parity error occurred. An analyst should be informed of the problem.

10. DISK HARWARE ERROR (NERROR = 208)

    STATUS = _ _ _

    This error should never occur and it indicates that a RTDM response code was not recognized. An analyst should be informed of the problem.

PURPOSE: To flush the buffer and rewind a real-time disk file.

USE: Format 1 CALL REWRTF (IFN)

Format 2 CALL REWRTF (IFN), RETURNS ($ERROR)

IFN integer file number (1 or 2) indicating which file is to be rewound.

$ERROR statement number to which control is returned if an abnormal condition is detected. See error processing section for usage.

RESTRICTIONS:

1. No automatic file positioning is done by Supervisor. A file must be explicitly rewound between RTWRITE and RTREAD.
2. REWRTF may not be called in SRT.

DESCRIPTION: Each time REWRTF is called, the specified file is rewound to the beginning of information. If the file buffer is not empty and the last file activity was a write, the remaining data is written to the file. Rewinding a rewound file or an uninitialized file is a null operation. If REWRTF is called in HOLD3, Supervisor enters HOLD1, issues the rewind, and returns to HOLD3.

ERROR MESSAGES:

1. ILLEGAL FILE NUMBER (NERROR = 4)

   File number is not 1 or 2.

2. CALLED IN WRONG PROGRAM STATE (NERROR = 2)

   REWRTF may not be called in SRT.

3. ATTEMPT TO EXCEED ALLOCATED FILE SPACE, FILE n (NERROR = 11, 12)

   Attempted to use more disk than allocated through SKED.

4. PARITY ERROR DETECTED ON FILE n (NERROR = 133, 134)

Unrecoverable disk parity error occurred. An analyst should be informed of the problem.

5. DISK HARDWARE ERROR (NERROR = 208)

STATUS = _ _ _

This error should never occur and it indicates that a RTDM response code was not recognized. An analyst should be informed of the problem.

PURPOSE: To skip forward records on a real-time disk file.

USE: Format 1 CALL SKPRTR (IFN, N)

Format 2 CALL SKTRTR (IFN, N), RETURNS ($END, $ERROR)

IFN integer file number (1 or 2) indicating on which file records are to be skipped.

N number of records to be skipped.

$END statement number to which control is returned if end of file is encountered.

$ERROR statement number to which control is returned if an abnormal condition is detected. See error processing section for usage.

RESTRICTIONS:

1. Skipping may be done only on a file that is being read.
2. Skipping may not be done in HOLD3.

DESCRIPTION: Each time that SKPRTR is called, the specified number of records is skipped on the file. If there are not enough records left on the file to skip, the file is positioned at end of information and control is returned to the $END statement number.

ERROR MESSAGES:

1. ILLEGAL FILE NUMBER (NERROR = 4)

   File number is not 1 or 2.

2. NEGATIVE RECORD COUNT (NERROR = 107)

   Number of records may only be 0 or positive.

3. I/O SEQUENCE ERROR FOR FILE n (NERROR = 13, 14)

   Skipping may only be done on a file that is being read.

4. REAL TIME HAS NOT BEEN INITIALIZED (NERROR = 1)

   No previous call to RTINIT.

5. FILE n NOT INITIALIZED (NERROR = 5, 6)

   Failure to call FILEARR.

6. VARIABLE TABLE TOO SMALL (NERROR = 121)

   File being skipped contains a variable table that exceeds the size of the variable table specified in last call to FILEARR.

7. BUFFER EMPTY (NERROR = 119)

   SKPRTR was called in SRT before READBUF or SKPRTR had been called to initialize the buffer in HOLD1 or HOLD2.

8. LOST DATA DETECTED FOR FILE n (NERROR = 9, 10)

   Exceeded data transfer capacity of simulation disk system (SRT only).

9. UNRECOGNIZABLE FILE HEADER (NERROR = 205)

   This error should never occur and it indicates that Supervisor did not find a specially coded file header in a real-time file. An analyst should be informed of the problem.

10. PARITY ERROR DETECTED ON FILE n (NERROR = 133, 134)

    Unrecoverable disk parity error occurred. An analyst should be informed of the problem.

11. DISK HARDWARE ERROR (NERROR = 208)

    STATUS = _ _ _

    This error should never occur and it indicates that a RTDM response was not recognized. An analyst should be informed of the problem.

PURPOSE: To dispose a file to an output queue.

USE: Format 1 CALL DISPOSE (LFN, QUEUE, OT)

Format 2 CALL DISPOSE (LFN, QUEUE, OT), RETURNS ($ERROR)

LFN logical file name, left justified, either blank filled or zero filled (e.g., 2LMF or 2HMF or "MF").

QUEUE (optional) queue to which file is to be disposed. Valid values are:

"PR" - print queue (DEFAULT)

"PH" - punch queue coded 026

"P9" - punch queue coded 029

"PB" - punch binary

"P8" - punch 80 column binary

OT (optional) origin type queue to which the file is to be disposed. Valid values are:

"BC" - batch queues (DEFAULT)

"EI" - remote batch queues

$ERROR statement number to which control is returned if an abnormal condition is detected. See error processing section for usage.

RESTRICTIONS:

1. The file must be a local mass storage file.
2. The file must exist and be nonempty.
3. The file should be rewound to assure that all information has been written.
4. May not be called in SRT.

DESCRIPTION: The file is released to the specified queue. If the origin type is "EI" and the computer being used is R or T, then the file is released to the simulation batch terminal on that machine. If the origin type is "EI" and some other computer is used, the file is released to the user's user number.

ERROR MESSAGES:

1. ILLEGAL FILE TYPE (NERROR = 108)

   File is not a local mass storage file.

2. FILE EMPTY (NERROR = 109)

   No information on file or file does not exist.

3. ILLEGAL FILE NAME (NERROR = 110)

   LFN is not a file name.

4. ILLEGAL QUEUE TYPE (NERROR = 111)

   Queue type is unrecognizable.

5. ILLEGAL ORIGIN TYPE (NERROR = 112)

   Origin type is unrecognizable.

6. REAL TIME HAS NOT BEEN INITIALIZED (NERROR = 1)

   No previous call to RTINIT.

7. CALLED IN WRONG PROGRAM STATE (NERROR = 2)

   May not be called in SRT.

8. ERROR RETURN FROM LFM (NERROR = 206)

   STATUS = _ _ _

   This error should never occur and it indicates that LFM (local file manager) returned an error code that was not recognized by Supervisor. An analyst should be informed of the problem.

EXAMPLE 1

PROBLEM STATEMENT:

Write variables T, X, Y, and Z and arrays P, D, and Q to RTDISK during a synchronized simulation run. After the run is completed, play back the written data, print the data, and dispose the print file to the batch line printer.

EXPLANATION: (numbers refer to program listing below)

1. The PROGRAM statement establishes the standard input and output files. In addition, it establishes a special file which will be written on and disposed to the batch printer.
2. This DIMENSION establishes arrays which will be used for the real-time file buffer and the variable address table.
3. This DIMENSION establishes the simulation recording array variables.
4. This DATA statement establishes a file name for subsequent access to the file PRTFILE.
5. This CALL defines the data buffer and variable address buffer for real-time file 1.
6. These two CALLS establish the program variables and arrays for real-time file 1.
7. This CALL executed in synchronized real-time causes the variables and arrays established in 6 to be written to real-time file 1.
8. This CALL, executed in HOLD1 or HOLD2, causes real-time file 1 to be rewound.

9. This loop causes one record of real-time data to be read from the real-time disk and written to file PRTFILE each time through the loop. After the last record has been processed, control is transferred to statement 90032.

10. PRTFILE is rewound to ensure that all data is flushed from the buffer; then the file is disposed to the batch print queue for printing.

11. Real-time file 1 is rewound before making another run. Subsequent CALLS to RTWRITE will cause the same set of variables and arrays to be written as before.

EXAMPLE 1

```
-1-             PROGRAM A (INPUT,OUTPUT,PRTFILE)
                    .
                    .
-2-             DIMENSION BUF1(2049),VTAB(7)
-3-             DIMENSION P(5),D(5,3),Q(10)
                INTEGER PRTFILE
                    .
                    .
          C     ESTABLISH FILE NAME FOR PRTFILE
-4-             DATA PRTFILE /7LPRTFILE/
                    .
                    .
          C     ESTABLISH FILE BUFFER AND VARIABLE ADDRESSES
-5-             CALL FILEARR (1,BUF1,2049,VTAB,7)
-6-             CALL FILEVAR (1,T,X,Y,Z)
-6-             CALL FILEVAR (-1,P,5,D,15,Q,10)
                    .
                    .
          C     ENTER SRT FRAME LOOP
          C     THE FOLLOWING SECTION IS THE REAL TIME LOOP
                    .
                    .
          C     WRITE OUT THE PROGRAM VARIABLES EACH FRAME TIME
-7-             CALL RTWRITE (1)
                    .
                    .
          C     END OF SRT LOOP
                    .
                    .
          C     ENTER HOLD1 OR HOLD2 STATE
                    .
                    .
          C     READ BACK THE PROGRAM VARIABLES AND WRITE TO PRTFILE
-8-             CALL REWRTF(1)
          90030 CALL RTREAD (1,LD), RETURNS (90032)
                WRITE (PRTFILE,20000) T,X,Y,Z,P,D,Q
-9-       20000 FORMAT (      )
                GO TO 90030
          C     ALL VARIABLES HAVE BEEN WRITTEN, DISPOSE TO PRINTER
-10-      90032 REWIND PRTFILE
-10-            CALL DISPOSE (PRTFILE,"PR")
                    .
                    .
          C     REWIND FILE 1 FOR NEXT RUN
-11-            CALL REWRTF (1)
          C     RETURN TO SRT
                    .
                    .
                END
```

EXAMPLE 2

PROBLEM STATEMENT:

Copy a local file containing data to be read in real-time to a real-time disk file. Read the data (3 variables) from the real-time disk file in synchronized real time and write the results of the simulation run (7 variables) to the other real-time file. After the simulation run is complete, print the results and prepare for another run using the same real-time data.

EXPLANATION: (numbers refer to program listing below)

1. The PROGRAM statement establishes the standard input and output files. In addition, it is used to establish both the local file RTDATA which contains the real-time data and the local file PRTFILE which will be written on and disposed to the batch line printer.
2. The DIMENSION establishes arrays for the real-time file buffers and variable address tables.
3. This DATA statement establishes file names for subsequent access to files RTDATA and PRTFILE.
4. This CALL defines the data buffer and variable table buffer for real-time file 2.
5. This CALL establishes the program variables that will be associated with real-time file 2.
6. This loop causes sets of the variables VW, DELE, and DELA to be read from the local file RTDATA and then recorded on real-time file 2 until the end of file on RTDATA is reached. Each set of the variables becomes a logical record on the real-time file.
7. Real-time file 2 is rewound and the buffer is loaded initially for subsequent RTREADs.

8. A file buffer and associated variables are defined for real-time file 1 which will be used to record the results of the simulation run.
9. File 2 is read in real-time. Each time RTREAD is called, one set of the variables VW, DELE, and DELA are placed in memory. The program continues at statement 90100 whether or not end of information on real-time file 2 is reached.
10. A set of results is written to real-time file 1.
11. Real-time file 1 is rewound before being played back.
12. This loop causes one record of real-time results to be read from the real-time disk and written to file PRTFILE each time through the loop. After the last record has been processed, control is transferred to statement 90032.
13. PRTFILE is rewound to ensure that all data is flushed from the buffer; then the file is disposed to the batch print queue for printing.
14. Both files are rewound prior to the next run.
15. The real-time input file buffer is initialized for the next run.

EXAMPLE 2

```
-1-             PROGRAM B (INPUT,OUTPUT,RTDATA,PRTFILE)
                    .
                    .
-2-             DIMENSION BUF1(2049),BUF2(513),VTAB1(7),VTAB2(3)
                INTEGER RTDATA,PRTFILE
                    .
                    .
              C ESTABLISH FILE NAMES
-3-             DATA RTDATA /6LRTDATA/, PRTFILE /7LPRTFILE/
                    .
                    .
              C INITIALIZE REAL TIME INPUT FILE AND COPY DATA FROM RTDATA
-4-             CALL FILEARR (2,BUF2,513,VTAB2,3)
-5-             CALL FILEVAR (2,VW,DELE,DELA)
              { 100 READ (RTDATA,10000) VW,DELE,DELA
              { 10000 FORMAT (     )
-6-           {     IF (EOF(RTDATA)) 120,110
              {  110 CALL RTWRITE (2)
              {     GO TO 100
-7-            120 CALL REWRTF (2)
-7-             CALL READBUF (2)
                    .
                    .
              C INITIALIZE REAL TIME OUTPUT FILE
-8-             CALL FILEARR (1,BUF1,2049,VTAB1,7)
-8-             CALL FILEVAR (1,T,U,V,W,X,Y,Z)
                    .
                    .
              C ENTER SRT LOOP
              C THE FOLLOWING SECTION IS THE REAL TIME LOOP
                    .
                    .
              C READ THE REAL TIME DATA FILE, IGNORE END OF FILE
-9-             CALL RTREAD (2), RETURNS (90100)
-9-           90100 CONTINUE
                    .
                    .
              C WRITE OUT THE PROGRAM VARIABLES EACH FRAME TIME
-10-            CALL RTWRITE (1)
                    .
                    .
              C END OF SRT LOOP
                    .
                    .
              C ENTER HOLD1 OR HOLD2 STATE
                    .
                    .
                    .
```

```
      C  READ BACK THE PROGRAM VARIABLES AND WRITE TO PRTFILE
-11-       CALL REWRTF(1)
      90030 CALL RTREAD (1,LD), RETURNS (90032)
            WRITE (PRTFILE,20000) T,U,V,W,X,Y,Z
-12-  20000 FORMAT (      )
            GO TO 90030
      C  ALL VARIABLES HAVE BEEN WRITTEN, DISPOSE TO PRINTER
-13-  90032 REWIND PRTFILE
-13-       CALL DISPOSE (PRTFILE,"PR")
                 .
                 .
      C  INTIALIZE FILES FOR NEXT RUN
-14-       CALL REWRTF (1)
-14-       CALL REWRTF (2)
-15-       CALL READBUF (2)
      C  RETURN TO SRT
                 .
                 .
           END
```

EXAMPLE 3

PROBLEM STATEMENT:

During an aircraft landing simulation, write the variables T, H, X, Y, Z, U, V, and W to a real-time file until the altitude (H) gets below 8000 feet. At that time, begin writing additional variables RANGE, AZIMUTH, and ELEVAT. After the simulation run print the time history.

EXPLANATION: (numbers refer to program listing below)

1. The PROGRAM statement establishes the standard input and output files. In addition, it establishes a special file which will be written on and disposed to the batch printer.
2. This DIMENSION establishes the arrays which will be used for the real-time file buffer and variable table.
3. This DATA statement establishes a file name for subsequent access to the file PRTFILE.
4. This CALL defines the buffer and variable table array for real-time file 1.
5. This CALL establishes the initial set of program variables that will be associated with real-time file 1.
6. This initializes the control variable INIT.
7. This section of code causes the variables RANGE, AXIMUTH, and ELEVAT to be added to the set of variables associated with real-time file 1. This is done the first time that H gets below 8000 feet.
8. This CALL executed in SRT causes the variables established by FILEVAR to be written to real-time file 1.
9. This CALL causes real-time file 1 to be rewound.

10. This loop reads back the data on real-time file 1 and writes the time history on PRTFILE. The loop exits to 90032 when all the data has been processed.
11. PRTFILE is rewound to ensure that all data is flushed from the buffer; then the file is disposed to the batch print queue for printing.
12. Real-time file 1 is rewound before making another run.
13. This CALL erases the previous file 1 variable table so that RANGE, AZIMUTH, and ELEVAT will not be recorded initially.
14. FILEVAR is called to establish the initial file 1 variables.
15. Control variable INIT is reinitialized.

EXAMPLE 3

```
-1-             PROGRAM C (INPUT,OUTPUT,PRTFILE)
                    .
-2-             DIMENSION BUF1(1025),VTAB1(11)
                INTEGER PRTFILE
                    .
                    .
              C  ESTABLISH FILE NAME FOR PRTFILE
-3-             DATA PRTFILE /7LPRTFILE/
                    .
                    .
              C  ESTABLISH FILE BUFFER AND VARIABLE ADDRESSES
-4-             CALL FILEARR (1,BUF1,1025,VTAB1,11)
-5-             CALL FILEVAR (1,T,H,X,Y,Z,U,V,W)
-6-             INIT = 0
                    .
                    .
              C  ENTER SRT FRAME LOOP
              C  THE FOLLOWING SECTION IS THE REAL TIME LOOP
                    .
                    .
              C  WHEN H PASSES THRU 8000 START WRITING RANGE, AZIMUTH, AND ELEVATION
                    IF (H.GT.8000.0) GO TO 91000
                    IF (INIT.LT.0) GO TO 91000
-7-                 CALL FILEVAR (1,RANGE,AZIMUTH,ELEVAT)
                    INIT = -1
              91000 CONTINUE
                    .
                    .
              C  WRITE OUT THE PROGRAM VARIABLES EACH FRAME TIME
-8-                 CALL RTWRITE (1)
                    .
                    .
              C  END OF SRT LOOP
                    .
                    .
              C  ENTER HOLD1 OR HOLD2 STATE
                    .
                    .
              C  READ BACK THE PROGRAM VARIABLES AND WRITE TO PRTFILE
-9-                 CALL REWRTF(1)
              90030 CALL RTREAD (1,LD), RETURNS (90032)
                    WRITE (PRTFILE,20000) T,H,X,Y,Z,U,V,W
-10-                IF (H.LT.8000.0) WRITE (PRTFILE,20001) RANGE,AZIMUTH,ELEVAT
              20000 FORMAT (       )
              20001 FORMAT (       )
                    GO TO 90030
              C  ALL VARIABLES HAVE BEEN WRITTEN, DISPOSE TO PRINTER
-11-          90032 REWIND PRTFILE
-11-                CALL DISPOSE (PRTFILE,"PR")
                    .
                    .
```

```
      C  INITIALIZE FILE 1 FOR NEXT RUN
-12-        CALL REWRTF (1)
-13-        CALL NEWVAR (1)
-14-        CALL FILEVAR (1,T,H,X,Y,Z,U,V,W)
-15-        INIT = 0
      C  RETURN TO SRT
                 .
                 .
                 .
           END
```

LOST TIME SYNCHRONIZATION

A condition known as lost time synchronization occurs when a program in synchronized real-time attempts to use more central processor time in a single frame than was allotted to the program through SKED. When this occurs, Real-Time Monitor interrupts the program, saves its operating registers, and restarts the program on the next frame at a previously established interrupt address in Supervisor. At this point, the Supervisor has two paths that can be processed. If no previous CALL LOSTIME has been executed or if the argument NFRAME is negative, the Supervisor transfers control to the standard error package. If LOSTIME recovery is enabled, control is transferred to the recovery statement. The program is still in SRT and at this recovery point the application program can issue a CALL RESUME or RESUMEQ to resume execution at the point of interruption or the program may elect to do some other processing.

PURPOSE: To provide a lost time synchronization recovery capability.

USE: Format 1 CALL LOSTIME (NFRAME), RETURNS ($RECSTAT)

Format 2 CALL LOSTIME (NFRAME), RETURNS ($RECSTAT,$ERROR)

NFRAME maximum number of frames the program is allowed to be in continuous lost synchronization recovery. If NFRAME is negative, no recovery processing will be done and Supervisor will transfer control to the standard error package.

$RECSTAT statement number to which control is transferred each time a lost synchronization interrupt occurs.

$ERROR statement number to which control is returned if an abnormal condition is detected. See error processing section for usage.

RESTRICTIONS: None.

DESCRIPTION: If this call is not made at least once, control is transferred to the standard error recovery package when a lost-synchronization interrupt occurs. Subsequent calls to this entry point may reset the paramenter NFRAME and return addresses $RECSTAT and $ERROR. A counter is maintained for adjacent lost time synchronization interrupts. Until this counter exceeds NFRAME, control is transferred to $RECSTAT. If the count exceeds NFRAME, control is transferred to $ERROR if specified, or if not, to the standard error recovery package. This counter will be reset if a normal frame end occurs (RTCYCLE, RTIDLE, or RTCLEAR) or a change of program state is affected.

ERROR MESSAGES:

1. REAL TIME HAS NOT BEEN INITIALIZED (NERROR = 1)

   No previous call to RTINIT.

2. LOST TIME SYNCHRONIZATION INTERRUPT HAS OCCURRED (NERROR = 113)

   LOSTIME was not called or recovery was disabled and a single lost time interrupt occurred. (Note that this error is not generated by LOSTIME, but by the lost synchronization detection section of RTINIT. The message is included here for clarity.)

3. TOO MANY LOST SYNCHRONIZATION INTERRUPTS (NERROR = 114)

   Number of continuous lost synchronization interrupts exceeds NFRAME. (Note that this message is not generated by LOSTIME, but by the lost synchronization detection section of RTINIT. The message is included here for clarity.)

4. NO LOST TIME RECOVERY ADDRESS (NERROR = 127)

   $RECSTAT was not specified for NFRAME positive.

PURPOSE: To resume processing at the point where lost time synchronization interrupt occurred.

USE: Format 1 CALL RESUME

Format 2 CALL RESUME, RETURNS (\$ERROR)

\$ERROR statement number to which control is returned if an abnormal condition is detected. See error processing section for usage.

RESTRICTIONS:

1. A lost time synchronization interrupt must have occurred.
2. The program must be in SRT state.
3. No state changes are allowed between the interrupt and the CALL RESUME.

DESCRIPTION: When a lost time synchronization interrupt occurs, the complete program state (exchange package) is saved. By calling RESUME, the program is restarted at the point that the interrupt occurred.

ERROR MESSAGES:

1. CALLED IN WRONG PROGRAM STATE (NERROR = 2)

   RESUME must be called in SRT only.

2. NO LOST SYNCHRONIZATION, RESUME CALLED (NERROR = 115)

   RESUME was called before lost time synchronization interrupt occurred or the program state had been changed.

3. REAL TIME HAS NOT BEEN INITIALIZED (NERROR = 1)

   No previous call to RTINIT.

4. RESUME CALLED BY NONREAL-TIME JOB (NERROR = 128)

   RESUME may not be called by a nonreal-time job.

PURPOSE: To resume processing at the point where lost time synchronization interrupt occurred.

USE: Format 1 CALL RESUMEQ

Format 2 CALL RESUMEQ, RETURNS ($ERROR)

$ERROR statement number to which control is returned if an abnormal condition is detected. See error processing section for usage.

RESTRICTIONS:

1. A lost time synchronization interrupt must have occurred.
2. The program must be in SRT state.

DESCRIPTION: When a lost time synchronization interrupt occurs, the complete program state (exchange package) is saved. By calling RESUMEQ, the program is restarted at the point that the interrupt occurred. The only difference between RESUMEQ and RESUME is that RESUMEQ will allow changing of states between the lost time synchronization interrupt and the resuming call.

ERROR MESSAGES:

1. CALLED IN WRONG PROGRAM STATE (NERROR = 2)

   RESUMEQ must be called in SRT only.

2. REAL TIME HAS NOT BEEN INITIALIZED (NERROR = 1)

   No previous call to RTINIT.

3. RESUMEQ CALLED BY NONREAL-TIME JOB (NERROR = 128)

   RESUMEQ may not be called by a nonreal-time job.

4. NO LOST SYNCHRONIZATION, RESUMEQ CALLED (NERROR = 115)

   RESUMEQ was called before lost time synchronization interrupt occurred.

EXAMPLE 4

PROBLEM STATEMENT:

Set up a simulation program such that one frame of lost synchronization is allowed before the program "synchs" up again. If more than one adjacent lost synchronization interrupt occurs, go into HOLD3 state, issue messages, and do cleanup processing.

EXPLANATION: (numbers refer to program listing below)

1. The PROGRAM statement establishes access to the standard input and output files and PRTFILE.
2. This CALL initializes the program for real-time processing.
3. This CALL establishes the maximum number of frames the program is allowed to be in continuous lost synchronization recovery (1), the lost synchronization recovery statement (90100), and the error recovery address (90200).
4. The program enters synchronized real-time.
5. The program recovers to this point on the first of a series of lost synchronization interrupts. It calls RESUME to resume the program at the point of interruption.
6. The program recovers to this point when more than one contiguous lost synchronization interrupt occurred. The program enters HOLD3 to communicate with the terminal and to do cleanup processing.

EXAMPLE 4

```
-1-             PROGRAM A (INPUT,OUTPUT,PRTFILE)


          C  INITIALIZE FOR REAL TIME
-2-             CALL RTINIT ("TEST LTIME",ADC,4,DAC,30)


          C  INITIALIZE LOST TIME PROCESSING
-3-             CALL LOSTIME (1), RETURNS (90100,90200)
                .
                .
          C  ENTER SRT
-4-             CALL RTSRT
          C  BEGIN SRT LOOP


          C  END OF SRT LOOP

                .
          C  LOST TIME INTERRUPT POINT
-5-       90100 CALL RESUME


          C  LOST TIME SYNCHRONIZATION OCCURRED TWICE IN SUCCESSION
-6-       90200 CALL RTHOLD3
          C  ISSUE MESSAGE TO TERMINAL AND DO CLEANUP PROCESSING


               END
```

NONREAL-TIME USAGE

The Supervisor consists of two subroutine sets, each on a different library. One set is used for the normal processing of a simulation with connection to the real-time hardware. The other set is used for debugging and analysis where connection to the real-time hardware is not required and no communication with Real-Time Monitor, the DASS hardware, the real-time disk, or the CRTs is done.

An entry point (RTCLASS) is provided that returns a condition flag indicating which Supervisor is being used. If the nonreal-time Supervisor is being used, the Real-Time Monitor and the DASS system are simulated. Initially, all discrete inputs are off and all ADC inputs are 0. These values are not changed by the Supervisor. Real-time disk files 1 and 2 become local files RTDM1 and RTDM2. If an error is detected and if a corresponding $ERROR statement number has been specified, then transfer to $ERROR is done. If $ERROR is not specified and the job is time-sharing origin, the program may recover to the global recovery address (see RECADD), if specified; otherwise, all errors are fatal.

PURPOSE: To determine which Supervisor is being used.

USE: CALL RTCLASS (N)

N 0 if nonreal-time Supervisor is being used.

-1 if real-time Supervisor is being used.

RESTRICTIONS: None.

DESCRIPTION: Supervisor sets the value of N depending upon which library is used. This allows a real-time program and a nonreal-time program to be identical and the programmer need only to use the correct library for loading.

ERROR MESSAGES: None.

COMMON BLOCK COMMUNICATION AND DISCRETE INPUT/OUTPUT

Four labeled common blocks (Q9RTIO, SUPCOMM, ERROR, DSCRETE) are maintained by Supervisor. Q9RTIO should not be declared in the user's program because it is used for communication between various utility routines in the real-time library. SUPCOMM contains certain information that Supervisor knows about the job. The user would declare this common if he needs any of this data. The applications program should never write into this common because it may adversely affect Supervisor. ERROR returns information from Supervisor to the user on error conditions. This common is referred to in the error processing section. Finally, DSCRETE contains the packed discretes and two arrays of masks which are used in testing and writing discretes. Discretes are not unpacked by Supervisor because of the unnecessary and excessive time used in the process. This common should appear in all user programs that overtly use any discretes.

A detailed description of the commons and some examples of testing and writing discretes follows.

```
COMMON /Q9RTIO/ DASS,RTCRT,RTDM,ADAGE,R(6)
```

This common is supplied for hardware processor interface and should not be declared by the user. It is included here for processor development. If a processor is required to perform real-time input/output, the corresponding word of Q9RTIO is set to all ones. Supervisor will then set the corresponding bit in the RTIDLE or RTRECALL** request and clear the Q9RTIO flag. The R(6) is reserved for future expansion.

** See reference 2.

```
COMMON /SUPCOMM/ RCT,FT,NADDCS,NDACS,NIDISS,NODISS,ADSTAT,DASTAT,HTIME,
       ADBUF(20),DABUF(64),TIMUSED,TIMREM,LSEXP(16),SRTF,HLD1F,HLD2F,
       HLD3F,LSF,OT,FL,USERNUM,SIMID,MACHID,RSVD(8)
```

RCT	Integer containing the requested compute time in units of 128 microseconds.
FT	Integer containing the frame time in units of 1/1024 seconds.
NADCS	Number of ADCs allocated through SKED.
NDACS	Number of DACs allocated through SKED.
NIDISS	Number of discrete inputs allocated through SKED.
NODISS	Number of discrete outputs allocated through SKED.
ADSTAT	ADCON and RTC-IT status as defined on page 4-6 of RTCS Reference Manual.
DASTAT	DACON status as defined on page 4-6 of RTCS Reference Manual.
HTIME	Time of day and ADC overload condition as defined on page 4-6 of RTCS Reference Manual.
ADBUF	Buffer for packed ADC inputs with four ADCs per word.
DABUF	Buffer for packed DAC outputs with four DACs per word.
TIMUSED	Integer CPU time in microseconds that was used in the previous SRT frame. If lost synchronization occurred on previous frames, TIMUSED contains the cumulative continuous CPU time used.
TIMREM	Integer CPU time in microseconds that was not used in the previous SRT frame.
LSEXP	Lost time synchronization exchange package at time of interruption. See NOS Operating System Reference Manual for detailed definition.

SRTF	Flag for SRT, -1 if SRT, 0 if not.
HLD1F	Flag for HOLD1, -1 if HOLD1, 0 if not.
HLD2F	Flag for HOLD2, -1 if HOLD2, 0 if not.
HLD3F	Flag for HOLD3, -1 if HOLD3, 0 if not.
LSF	Flag for lost time synchronization, -1 if in continuous lost synchronization recovery, 0 if not.
OT	Job origin type "BCOT" if batch origin. "TXOT" if time sharing origin "EIOT" if remote batch origin "SYOT" if system origin "MTOT" if multiterminal origin
FL	Integer containing field length at last state change i.e. CALL HOLD3, CALL RTSRT, etc.
USERNUM	User number in display code, left-justified, zero-filled.
SIMID	The simulation identification as specified in RTINIT with colons (00B display) changed to blanks (55B display).
MACHID	One character, left justified zero filled word containing the identification of the computer in use; e.g., 1LR, 1LT, 1LY, etc.
RSVD	Reserved for future expansion.

```
COMMON /ERROR/ NERROR,NAME1,NAME2,ABSP,RELP
```

NERROR	Error number of last detected error.
NAME1	If nonzero, contains the name of the routine causing the error.
NAME2	If nonzero, contains the name of the routine which detected the error.
ABSP	Integer, absolute program address where error occurred.
RELP	Integer, if nonzero, approximate program address relative to the beginning of the offending routine.

```
COMMON /DSCRETE/ IDIS(16),ODIS(16),TMASK(60),FMASK(60)
```

IDIS Packed discrete inputs - 60 per word

ODIS Packed discrete outputs - 60 per word

TMASK "True" mask array. The nth bit (starting on the right of the word as bit 1) of TMASK(n) is set. All other bits in TMASK(n) are off.

FMASK "False" mask array. The nth bit (starting on the right of the word as bit 1) of FMASK(n) is off. All other bits in FMASK(n) are set.

RESTRICTIONS:

1. Always DIMENSION precisely as shown above regardless of how many discretes are used.
2. Supervisor initializes the masks in RTINIT. The masks are used internally and must not be changed. If other masks are needed, they should be created independently.

EXAMPLES:

1. Test input discrete 71 (bit 11, word 2). If it is on (true) branch to statement 1000.

   ```
   IF ((IDIS(2).AND.TMASK(11)).NE.0) GO TO 1000
   ```

   or (slightly faster)

   ```
   IF (SHIFT(IDIS(2),60-11).LT.0) GO TO 1000
   ```

2. Set output discrete 83 (bit 23, word 2) to true.

   ```
   ODIS(2) = ODIS(2).OR.TMASK(23)
   ```

3. Set output discrete 44 (bit 44, word 1) to false

   ```
   ODIS(1) = ODIS(1).AND.FMASK(44)
   ```

4. Test input discrete 37 (bit 37, word 1). If it is off (false) set H = HMIN

```
IF ((IDIS(1).AND.TMASK(37)).EQ.0) H = HMIN
```

or (slightly faster)

```
IF (SHIFT(IDIS(1),60-37).GE.0) H = HMIN
```

NOTE: The change from using unpacked to using packed discretes will necessitate a change (see example) in how one references (reads or writes) discretes. In addition, the relative numbering of the discrete inputs within the user's program will be incremented by one. The first discrete input in the user's array is wired to the abort button on the console. This is the only discrete that will be monitored by Supervisor, and it will transfer control to the error processing module at the interactive console when the button is depressed.

Because of these changes, the discrete assignment tables from the Simulation Manual (Section 5100.1, pages 27-30) have been altered and are included here for easy reference.

DASS DISCRETE INPUT ASSIGNMENTS

	Quantity	Bit Position		Patchboard Wiring	
S/C 1		Word #	Bit #	Discrete #	Destination
ABORT	1	1	1	1	CLO (AJC 1)
DATA ENTRY	16	1	2-17	2-17	RMS 17-32
MODE CONTROL	16	1	18-33	18-33	RMS 1-16
FSS	16	1	34-49	34-49	FSS 1-16 (AJC 1)
INPUT FROM SITE	60	1	50-60	50-109	RMS 49-108
		2	1-49		
S/C 2					
ABORT	1	1	1	301	CLO (AJC 2)
DATA ENTRY	16	1	2-17	302-317	RMS 305-320
MODE CONTROL	16	1	18-33	318-333	RMS 289-304
FSS	16	1	34-49	334-349	FSS 1-16 (AJC 2)
INPUT FROM SITE	60	1	50-60	350-409	RMS 337-396
		2	1-49		
S/C 3					
ABORT	1	1	1	481	CLO (AJC 5)
DATA ENTRY	16	1	2-17	482-497	RMS 929-944
MODE CONTROL	16	1	18-33	498-513	RMS 913-928
FSS	16	1	34-49	514-529	FSS 1-16 (AJC 5)
INPUT FROM SITE	60	1	50-60	530-589	RMS 529-588
		2	1-49		

NOTE: Bit numbering starts with bit 1 on the right and progresses through bit 60 on the left. The bit number is one more than the conventional bit numbering in the computer.

DASS DISCRETE OUTPUT ASSIGNMENTS

SIMULATION CONSOLE 1

	Quantity	Bit Position		Patchboard Wiring	
		Word #	Bit #	Discrete #	Destination
OUTPUT TO SITE	45	1	1-45	1-45	RML 1-45
EVENT MARKERS	9	1	46-48	46-48	REC 2, EV 3-5
			49-51	49-51	3, 2-4
			52-54	52-54	4, 2-4
XY PEN LIFT	1	1	55-56	55-56	PLOTTER 1, PL (AJC 1)
					PLOTTER 2, PL
RECORDER					
START/STOP	4	1	57-60	57-60	
WHITE LIGHTS	39	2	1-39	61-99	WI 1-39 (AJC 1)
AUDIBLE ALARM	1	2	40	100	OVA (AJC 1)
EVENT MARKERS	9	2	41-49	101-109	REC 1, EV 1-9 (AJC 1)
EVENT MARKERS	1	2	50	110	REC 2, EV 2 (AJC 1)
RED LIGHTS	8	2	51-58	111-118	RI 1-8 (AJC 1)
MODE LIGHTS	16	2	59-60	119-134	RMS 33-48
		3	1-14		
DECIMAL DISPLAY	46	3	15-60	135-180	RML 145-190

NOTE: Bit numbering starts with bit 1 on the right and progresses through bit 60 on the left. The bit number is one more than the conventional bit number in the computer.

DASS DISCRETE OUTPUT ASSIGNMENTS

SIMULATION CONSOLE 2

	Quantity	Bit Position		Patchboard Wiring	
		Word #	Bit #	Discrete #	Destination
OUTPUT TO SITE	45	1	1-45	301-345	RML 313-342
EVENT MARKERS	9	1	46-48	346-348	REC 2, EV 3-5
			49-51	349-351	3, 2-4
			52-54	352-354	4, 2-4
XY PEN LIFT	2	1	55-56	355-356	PLOTTER 1, PL (AJC 2)
					PLOTTER 2, PL
RECORDER					
START/STOP	4	1	57-60	357-360	
WHITE LIGHTS	39	2	1-39	361-399	WI 1-39 (AJC 2)
AUDIBLE ALARM	1	2	40	400	OVA (AJC 2)
EVENT MARKERS	9	2	41-49	401-409	REC 1, EV 1-9 (AJC 2)
EVENT MARKERS	1	2	50	410	REC 2, EV 2 (AJC 2)
RED LIGHTS	8	2	51-58	411-418	RI 1-8 (AJC 2)
MODE LIGHTS	16	2	59-60	419-434	RMS 321-336
		3	1-14		
DECIMAL DISPLAY	46	3	15-60	435-480	RML 265-310

NOTE: Bit numbering starts with bit 1 on the right and progresses through bit 60 on the left. The bit number is one more than the conventional bit numbering in the computer.

DASS DISCRETE OUTPUT ASSIGNMENTS

SIMULATION CONSOLE 3

	Quantity	Bit Position		Patchboard Wiring	
		Word #	Bit #	Discrete #	Destination
OUTPUT TO SITE	45	1	1-45	481-525	RML 481-525
EVENT MARKERS	9	1	46-48	526-534	REC 2, EV 3-5
			49-51		3, 2-4
			52-54		4, 2-4
XY PEN LIFT	1	1	55-56	535-536	PLOTTER 1, PL (AJC 5)
					PLOTTER 2, PL
RECORDER					
START/STOP	4	1	57-60	537-540	
WHITE LIGHTS	39	2	1-39	541-579	WI 1-39 (AJC 5)
AUDIBLE ALARM	1	2	40	580	OVA (AJC 5)
EVENT MARKERS	9	2	41-49	581-589	REC 1, EV 1-9 (AJC 5)
EVENT MARKER	1	2	50	590	REC 2, EV 2 (AJC 5)
RED LIGHTS	8	2	51-58	591-598	RI 1-8 (AJC 5)
MODE LIGHTS	16	2	59-60	599-614	RMS 945-960
		3	1-14		
DECIMAL DISPLAY	46	3	15-60	615-660	RML 769-814

NOTE: Bit numbering starts with bit 1 on the right and progresses through bit 60 on the left. The bit number is one more than the conventional bit numbering in the computer.

PURPOSE: To change the arrays that are used for discrete input and output buffers.

USE: Format 1 CALL DISADD (IARR, OARR)

Format 2 CALL DISADD (IARR, OARR), RETURNS ($ERROR)

IARR array name of a 16 word array to contain input discretes.

OARR array name of a 16 word array to contain output discretes.

$ERROR statement number to which control is returned if an abnormal condition is detected. See error processing section for usage.

RESTRICTIONS: DISADD may only be called in HOLD3 state.

DESCRIPTION: This is intended for diagnostic purposes. Discrete input/output arrays and associated masks are initially assigned to common block DSCRETE thusly:

```
COMMON /DSCRETE/ IDIS(16), ODIS(16), TMASK(60), FMASK(60).
```

For some applications, it may be necessary to change the discrete array assignment. DISADD will accomplish this function and the change will be effective until another CALL DISADD is executed.

ERROR MESSAGES:

1. CALLED IN WRONG PROGRAM STATE (NERROR = 2)

   DISADD may only be called in HOLD3.

2. REAL TIME HAS NOT BEEN INITIALIZED (NERROR = 1)

   No previous call to RTINIT.

3. BAD STATUS FROM URT-RTS FUNCTION (NERROR = 202)

   This error should never occur and it indicates a failure of the Supervisor or the real-time system. An analyst should be informed of the problem.

ERROR PROCESSING AND TERMINAL INTERACTION

This section describes the action taken by Supervisor when an error is detected and describes the interaction with the time-sharing terminal.

ERROR PROCESSING: The following stepchart illustrates Supervisor processing when an error is detected:

1. Update the values stored in COMMON /ERROR/.
2. If a $ERROR statement is specified, exit from Supervisor error processor to $ERROR.

   If no $ERROR is specified

3. Force HOLD3 state.
4. Issue messages to job dayfile.
5. If job is not time sharing origin, terminate the job abnormally.

   If job is time-sharing origin

6. Issue messages to interactive terminal.
7. Make options available to user through terminal interaction. These options include unformatted memory dump, display exchange package, and analyze program memory image.
8. Through terminal interaction, the user may elect to:
   a. end the program through normal termination and continue with next control statement in sequence.
   b. end the program through abnormal termination and continue with first control statement following the next EXIT statment.
   c. branch to a predetermined recovery point to restart the program if the point has been specified. (See SUBROUTINE RECADD following.)

The following illustrates the format of error messages. Consider that RTREAD was called with a file number of 17; then Supervisor would issue:

```
ERROR NUMBER 4 IN DMS-F11
ILLEGAL FILE NUMBER
RTREAD CALLED BY PROGF11 AT LINE 179
AT ABS  25347 APPROX.   1347 REL
```

The error number indicates the number in Table 2 which contains a listing of the particular error. The third line tells which Supervisor entry point was called and which user routine (PROGF11) called it. The fourth line gives the absolute location of the CALL statement and the approximate address relative to the beginning of the routine.

The simulation program itself may use the error processing section of Supervisor. This use is described in a later section on the subroutine RTERROR.

TERMINAL INTERACTION: The Supervisor determines if a job is of time-sharing origin. If it is, the Supervisor connects the file TT to the time-sharing terminal. All Supervisor terminal interaction is done through the file TT so that files INPUT and OUTPUT need not necessarily be assigned to the terminal. The user may use the file TT through the subroutines TTINPUT, TTMESS, and TTYDAYF or through FORTRAN I/O.

When RTINIT is called, Supervisor issues the message REAL TIME INITIALIZED. When the program terminates normally, the message REAL TIME NORMAL TERMINATION is displayed. If Supervisor aborts the program, the message REAL TIME ABORTED is displayed.

ERROR MESSAGES

NERROR	MESSAGE
1	REAL TIME HAS NOT BEEN INITIALIZED
2	CALLED IN WRONG PROGRAM STATE
3	REAL-TIME HARDWARE FAILURE
	STATUS = _ _ _
4	ILLEGAL FILE NUMBER
5	FILE 1 NOT INITIALIZED
6	FILE 2 NOT INITIALIZED
7	NO VARIABLES FOR FILE 1
8	NO VARIABLES FOR FILE 2
9	LOST DATA DETECTED FOR FILE 1
10	LOST DATA DETECTED FOR FILE 2
11	ATTEMPT TO EXCEED ALLOCATED FILE SPACE, FILE 1
12	ATTEMPT TO EXCEED ALLOCATED FILE SPACE, FILE 2
13	I/O SEQUENCE ERROR FOR FILE 1
14	I/O SEQUENCE ERROR FOR FILE 2
15-100	Reserved for Expansion

TABLE 2

NERROR	MESSAGE
101	BUFFER SIZE LESS THAN MINIMUM
102	ILLEGAL TABLE SIZE
103	FILE 1 NOT REWOUND
104	FILE 2 NOT REWOUND
105	FILE 1 EMPTY
106	FILE 2 EMPTY
107	NEGATIVE RECORD COUNT
108	ILLEGAL FILE TYPE
109	FILE EMPTY
110	ILLEGAL FILE NAME
111	ILLEGAL QUEUE TYPE
112	ILLEGAL ORIGIN TYPE
113	LOST TIME SYNCHRONIZATION INTERRUPT HAS OCCURRED
114	TOO MANY LOST TIME INTERRUPTS
115	NO LOST SYNCHRONIZATION, RESUME CALLED
116	NOT TIME SHARING ORIGIN
117	TOO MANY CHARACTERS
118	CONTROL WORD OUT OF RANGE
119	BUFFER EMPTY
120	TOO MANY VARIABLES SPECIFIED
121	VARIABLE TABLE TOO SMALL
122	ARRAY LENGTH NOT SPECIFIED
123	BAD ARRAY LENGTH
124	NO. OF ADC OR DAC .LT. 0

TABLE 2 (Cont'd)

NERROR	MESSAGE
125	NO RTDM FILES ALLOCATED
126	FILE 2 NOT ALLOCATED
127	NO LOST TIME RECOVERY ADDRESS
128	RESUME CALLED BY NONREAL-TIME JOB
129	CALLED DURING READ OPERATIONS - FILE 1
130	CALLED DURING READ OPERATIONS - FILE 2
131	NO RECOVERY ADDRESS SPECIFIED
132	BUFFER TOO SMALL
133	PARITY ERROR DETECTED FOR FILE 1
134	PARITY ERROR DETECTED FOR FILE 2
135	FORTRAN END PROCESSING ATTEMPTED
136-200	Reserved for Expansion

TABLE 2 (Cont'd)

NERROR	MESSAGE
201	NEGATIVE UNUSED TABLE ENTRY COUNT
202	BAD STATUS FROM URT-RTS FUNCTION
	STATUS = _ _ _
203	BAD STATUS FROM URT-FET FUNCTION
	STATUS = _ _ _
204	Q8LSREC CALLED IN NONREAL TIME
205	UNRECOGNIZABLE FILE HEADER
206	ERROR RETURN FROM LFM
	STATUS = _ _ _
208	DISK HARDWARE ERROR
	STATUS = _ _ _
209-299	Reserved for Expansion
300-399	Reserved for Real-Time CRT
400-499	Reserved for ADAGE

TABLE 2 (Cont'd)

PURPOSE: To issue error messages and control optional Supervisor processing after a detected error.

USE: CALL RTERROR (NERROR, MESSAGE, OPTIONS, LN, RPOS)

NERR — error number of the message. For Supervisor detected errors, see Table 2. For user detected errors, see appropriate documentation. Error numbers 1-499 are reserved for the Supervisor and real-time processors.

MESSAGE — message to be displayed as second line. Must be terminated by trailing byte of zeros.

OPTIONS — bits in this word control the options selected for optional Supervisor processing. The bit assignment will be defined during implementation.

LN — level number of routine which produced the error (i.e., how far to traceback for error routine). If 0, no traceback is done.

RPOS — integer indicating which return in the returns list in the calling subroutine is the recovery statement number. If 0, there is no recovery return.

RESTRICTIONS:

1. RTINIT must have been called; otherwise, all errors are fatal and the job is aborted with appropriate messages.
2. If a user level subroutine wishes to use the $ERROR returns feature, the call to RTERROR must be followed by a RETURN DOLERR, where DOLERR is the error return statement parameter.

DESCRIPTION: A detailed flow chart of error processing will be provided as implementation proceeds.

ERROR MESSAGES: ALL

PURPOSE: To define an error recovery address for time-sharing programs.

USE: CALL RECADD (N), RETURNS ($RECPT)

N 0, disable recovery

1, new recovery address

$RECPT statement number to which control is returned after Supervisor error processing.

RESTRICTIONS: None.

DESCRIPTION: For a time-sharing origin job when an error is detected by Supervisor for which there is no corresponding $ERROR statement number, messages are issued to the terminal, certain optional processing is allowed, and then, if directed by the terminal operator, transfer is made to $RECPT. Use of this subroutine in batch origin jobs has no effect.

ERROR MESSAGES:

1. REAL TIME HAS NOT BEEN INITIALIZED (NERROR = 1).
   No previous call to RTINIT.
2. NO RECOVERY ADDRESS SPECIFIED (NERROR = 131)
   $RECPT was not specified for N=1.

PURPOSE: To read a line from the interactive terminal.

USE: Format 1 CALL TTINPUT (MESS, N)

Format 2 CALL TTINPUT (MESS, N), RETURNS ($ERROR)

MESS array, which should be dimensioned to 16, that will receive the characters entered at the interactive terminal.

N length of line in words including the end of line character (see page F-2 of NOS 1.2 Reference Manual) received from terminal. If line entered from terminal consists of a carriage return only, then N will be 0.

$ERROR statement number to which control is returned if an abnormal condition is detected. See error processing section for usage.

RESTRICTIONS:

1. Must be a time-sharing origin job.
2. Must be in HOLD3 state.

DESCRIPTION: The Supervisor establishes a connection to the TTY terminal through the file TT. The file is read and the line and length are returned to the program. This is somewhat analogous to:

```
READ 100,MESS
100 FORMAT (16A10)
```

except that a 12 bit byte of binary 0's terminates the line after the last nonblank character.

ERROR MESSAGES:

1. NOT TIME-SHARING ORIGIN (NERROR = 116)

   Called by wrong origin job.

2. CALLED IN WRONG PROGRAM STATE (NERROR = 2)

   Program not in HOLD3.

3. REAL TIME HAS NOT BEEN INITIALIZED (NERROR = 1)

   No previous call to RTINIT.

PURPOSE: To write a line to the interactive terminal.

USE: Format 1 CALL TTMESS (MESS, NC)

Format 2 CALL TTMESS (MESS, NC), RETURNS ($ERROR)

MESS line of characters to be displayed on interactive terminal. Must be terminated by an end of line sequence, i.e. any word with four octal zeroes in the twelve rightmost bits. (See page F-2, Volume 1 of the NOS 1.2 Reference Manual.)

NC control word

0 for single message or last line of multiline message

1 for intermediate lines of multiline message

$ERROR statement number to which control is returned if an abnormal condition is detected. See error processing section for usage.

RESTRICTIONS:

1. A line is limited to 150 characters.
2. Must be in HOLD3 state.

DESCRIPTION: The Supervisor establishes a connection to the interactive terminal through the file TT. The line is written to file buffer and the buffer is written if full or NC is zero. If the job is not time-sharing origin type, local file TT is written.

ERROR MESSAGES:

1. CONTROL WORD OUT OF RANGE (NERROR = 118)

   NC is not 0 or 1.

2. TOO MANY CHARACTERS (NERROR = 117)

   Line length limited to 150 characters.

3. CALLED IN WRONG PROGRAM STATE (NERROR = 2)

Program not in HOLD3.

C-2

TTYDAYF

PURPOSE: To issue a message to the interactive terminal and to the job dayfile.

USE: Format 1 CALL TTYDAYF (MESS)

Format 2 CALL TTYDAYF (MESS), RETURNS ($ERROR)

MESS line of characters to be sent to terminal and job dayfile. Line must be terminated by an end of line sequence, i.e. any word with four octal zeroes in the twelve rightmost bits. (See page F-2, Volume 1 of the NOS 1.2 Reference Manual.)

$ERROR statement number to which control is returned if an abnormal condition is detected. See error processing section for usage.

RESTRICTIONS:

1. A line is limited to 80 characters.
2. Must be in HOLD3 state.

DESCRIPTION: The Supervisor establishes a connection to the interactive terminal through the file TT. The message is written to the terminal and the job dayfile. If the job is not time-sharing origin type, local file TT is written.

ERROR MESSAGES:

1. CALLED IN WRONG PROGRAM STATE (NERROR = 2)

   Program must be in HOLD3 state.

UTILITY SUBROUTINES

This section contains the documentation of utility subroutines that will be useful to the simulation programmer.

PURPOSE: To obtain CPU time used and CPU time remaining in an SRT frame.

USE: Format 1 CALL RTIME (USED, REM)

Format 2 CALL RTIME (USED, REM), RETURNS ($ERROR)

USED integer microseconds of CPU time used this frame.

REM integer microseconds of CPU time remaining in this frame.

$ERROR statement number to which control is returned if an abnormal condition is detected. See error processing section for usage.

RESTRICTIONS: Program must be in SRT state.

DESCRIPTION: Supervisor issues a URM10** request to Real-Time Monitor to obtain the CPU time used and then calculates the time remaining. The clock resolution is about 64 microseconds.

ERROR MESSAGES:

1. CALLED IN WRONG PROGRAM STATE (NERROR = 2)

   Must be in SRT.

2. REAL TIME HAS NOT BEEN INITIALIZED (NERROR = 1)

   No previous call to RTINIT.

** See reference 2.

REFERENCES

1. Control Data Corporation: NOS Version 1 Reference Manual, Volume 1 and Volume 2, Publication Numbers 60435400 and 60445300, Nov. 1977.

2. Control Data Corporation: Real Time Computing Subsystem Reference Manual, Publication Number 60454770, July 1977.

INDEX OF ENTRY POINTS

1. Report No.	2 Government Accession No	3 Recipient's Catalog No.
NASA TM 78757		

4. Title and Subtitle	5. Report Date
Reference Manual for the Langley Research Center Flight Simulation Computing System	July 1978
	6 Performing Organization Code

7. Author(s)	8 Performing Organization Report No.
Jeff I. Cleveland, II, Daniel J. Crawford, and Lawrence F. Rowell	
	10. Work Unit No.

9. Performing Organization Name and Address	11 Contract or Grant No
NASA-Langley Research Center Hampton, VA 23665	

12 Sponsoring Agency Name and Address	13. Type of Report and Period Covered
National Aeronautics & Space Administration Washington, D.C. 20546	Technical Memorandum
	14 Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

The Langley Research Center Flight Simulation Computing System provides a researcher with an advanced real-time digital simulation capability. This capability is controlled at the user interface level by the Real Time Simulation Supervisor. The Supervisor is a group of subprograms loaded with a simulation application program. The Supervisor provides the interface between the application program and the operating system, and coordinates input and output to and from the simulation hardware. The Supervisor also performs various utility functions as required by a simulation application program.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Real Time Simulation Digital Simulation Flight Simulation	Unclassified - Unlimited Subject Category 61

19 Security Classif. (of this report)	20. Security Classif. (of this page)	21. No of Pages	22. Price*
Unclassified	Unclassified	102	$6.50

* For sale by the National Technical Information Service, Springfield, Virginia 22161