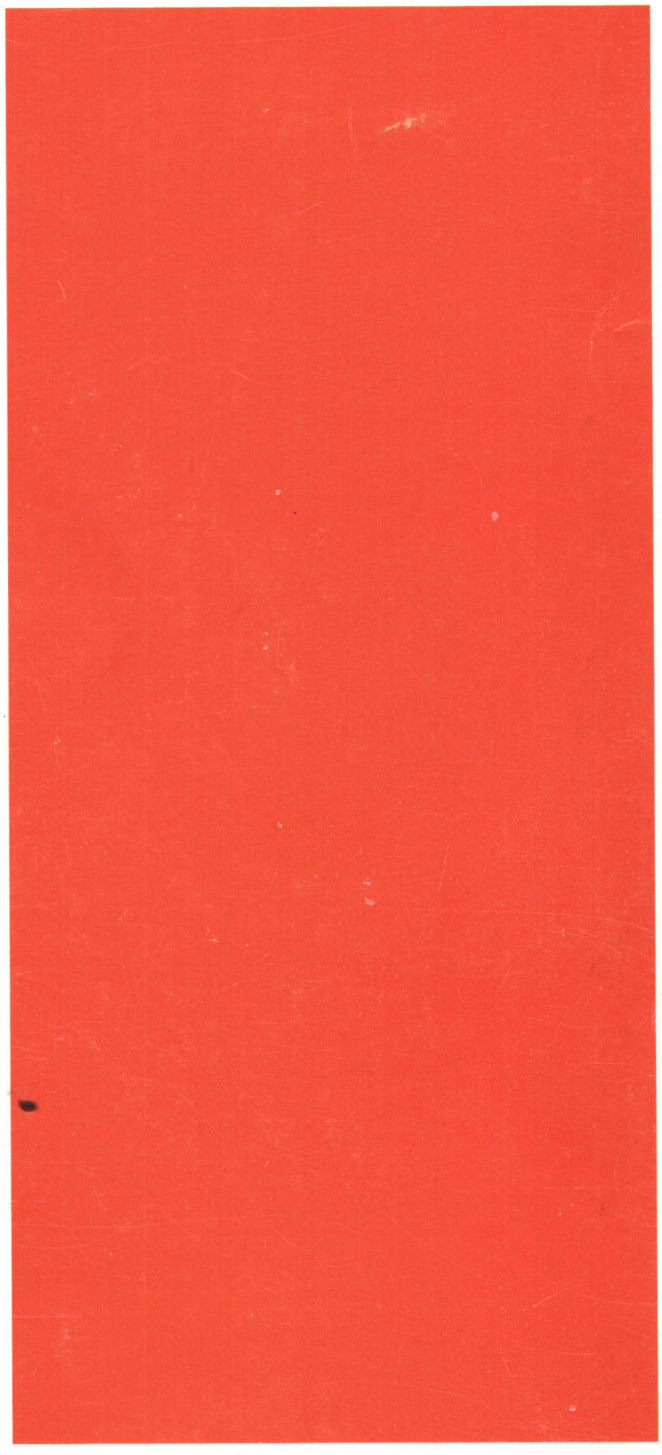
Honeywell

SYSTEMS MANUAL

SYSTEM 700

OS/700

SOFTWARE



Kd-Data AB
Streteredsvägen 6, 430 50 Kållered
Tel: 031-75 14 13, Postgiro 42 07 05 - 6
1973-10-13

Honeywell

SYSTEMS MANUAL

SYSTEM 700

OS/700

SUBJECT:

Overview, Basic Concepts, and System Macro Calls of the Operating System/700 (OS/700).

SPECIAL INSTRUCTIONS:

This manual supersedes the preliminary edition, dated February 1972. It has been extensively reorganized and rewritten. Therefore, no change bars or asterisks are used in this edition.

SOFTWARE SUPPORTED:

This publication supports Release 0100 of the Operating System/700.

DATE:

October 1972

ORDER NUMBER:

AG02, Rev. 1

DOCUMENT NUMBER:

70130072661B

PREFACE

This manual presents an overall view of OS/700 – the Operating System for System 700. The discussion proceeds from basic concepts to a detailed presentation of the system macros by which the user may access the various system functions.

The reader is expected to be familiar with assembly language programming and will find it helpful to consult two related mamuals: the System 700 Programmers' Reference Manual, Order Number AC72, and the DAP/700 Macro Assembler, Order Number AG17.

OS/700 is a coded system designed to extend the power of System 700 in the areas of program preparation and maintenance, data control, operations control, and utility functions. It is supported by comprehensive documentation and training; periodic program maintenance and, where feasible, improvements are furnished for the current version of the system, provided it is not modified by the user.

File No.: 1913

AG02

CONTENTS

CONTENTS (cont)

ILLUSTRATIONS

SECTION I
INTRODUCTION

System 700 models feature a software library that permits the configuration of both special and general application systems. With this library, System 700 systems can be configured for such diverse and different applications as:

- Remote message concentrators.
- Front-End processors.
- Multiprogramming operating systems.
- Real-Time application systems.

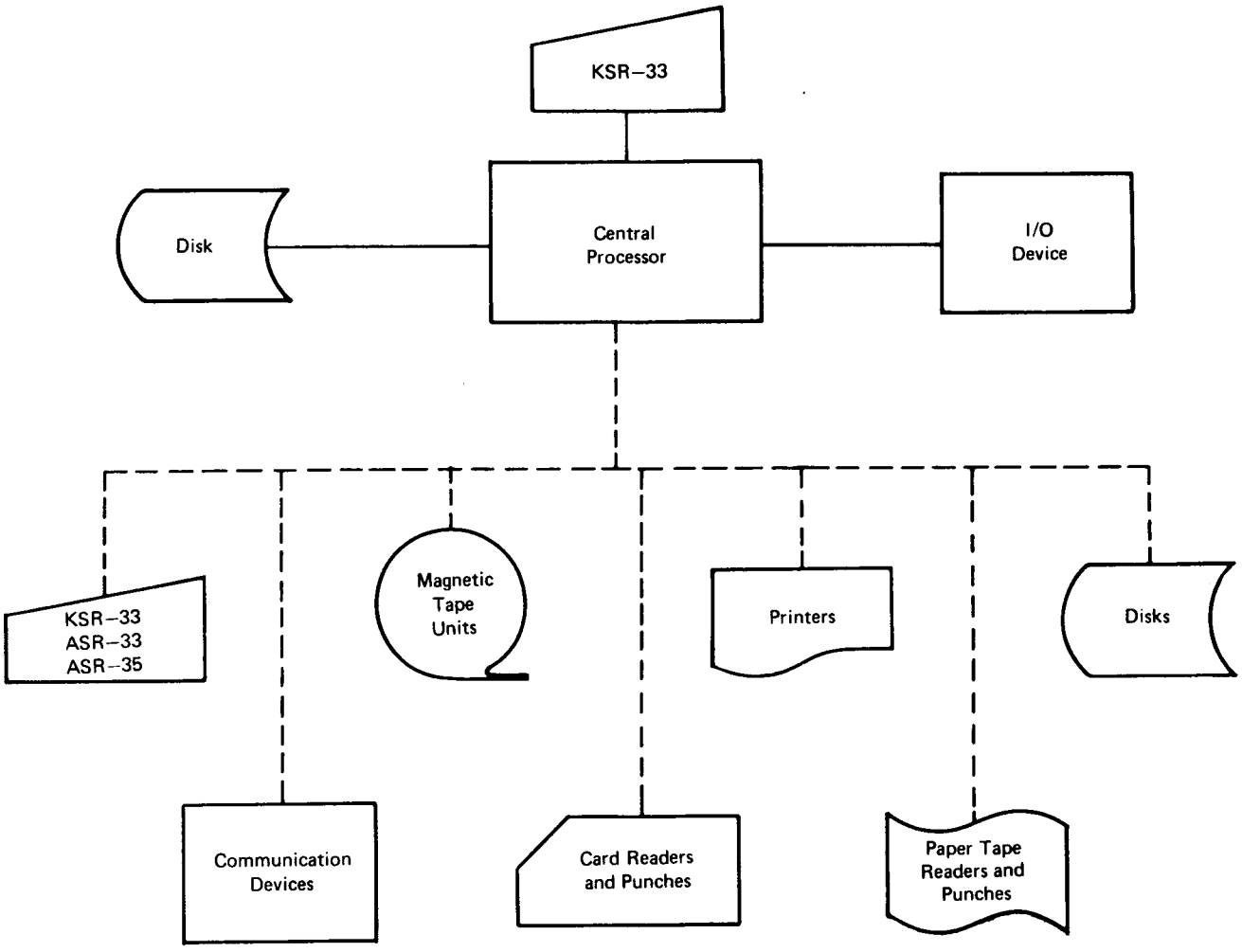
The library is used to configure the operating system and to tailor an application system. The operating system which supports System 700 is OS/700.

OS/700 includes the following components:

- Executive.
- Macro Assembler.
- FORTRAN Compiler (with real-time extensions).
- Linkage Editor.
- Text Editor.
- Systems Library.
- On-Line Utilities.

The hardware relating to OS/700 is illustrated in Figure 1-1. The specific peripheral devices that can be used on the System 700 include:

- Type 5310 KSR-33
- Type 5307 ASR-33, Type 5507 ASR-35
- Type 5010 High-Speed Paper Tape Reader
- Type 5210 High-Speed Paper Tape Punch
- Type 5140 Card Reader/Punch
- Type 5121 Card Reader
- System 700 Fixed Head Disk Subsystems
- System 700 Removable Disk Subsystems



NOTE: The basic system must consist of the processor, the disk, the KSR-33 and one I/O device. Depending on the System 700 model installed, the system can be expanded to include one or more of the peripherals shown in the diagram.

Figure 1-1. System 700 Equipment Supported by OS/700

- Type 5520 Line Printer
- Types 4021 7-Track Magnetic Tape Subsystem
- Type 4150 9-Track Magnetic Tape Subsystem
- Type 6312 Sync Single-Line Controller with Code Convention Option (Type 6313)
- Type 6321 Low Speed Multiline Controller
- Type 6322 Universal Multiline Controller
- Type 6333 Medium-Speed Multiline Controller

OS/700 OVERVIEW

OS/700 is a real-time, multiprogramming operating system with strong communications support and minimal main memory requirements. Each of the underlined items is defined below.

Real-Time: – guarantees immediate execution response and processing of an interrupt within the user's time requirements.

Multiprogramming: – simultaneously controls the execution of a number of activities, queuing requests for such activities and executing them according to their priority.

Communications Support: – supports a number of multiline controllers with line speeds ranging from 45 to 9600 baud and single-line controllers with line speeds up to 20,000 baud.

Minimal Main Memory Requirements (see Figure 1-2): – the Executive uses less than 5K of main memory (allowing 10K for user programs and 1K for free memory). It allows the user to include only those system modules that are needed, uses reentrant coding for many system operations (only one driver is required for each type of I/O device, regardless of how many are used) and uses overlays so that parts of the system can be stored on disk to be brought into memory when needed.

Features of OS/700 include:

Disk-resident – OS/700 is a Disk Operating System in which many of the system and user programs are stored on disk until required for execution.

Resource Management – OS/700 manages the allocation of time, memory, and I/O devices.

File Management Capability – OS/700 provides a file management capability in addition to physical I/O.

Priority Assignment – OS/700 permits the user to assign priority levels to the execution of activities and tasks.

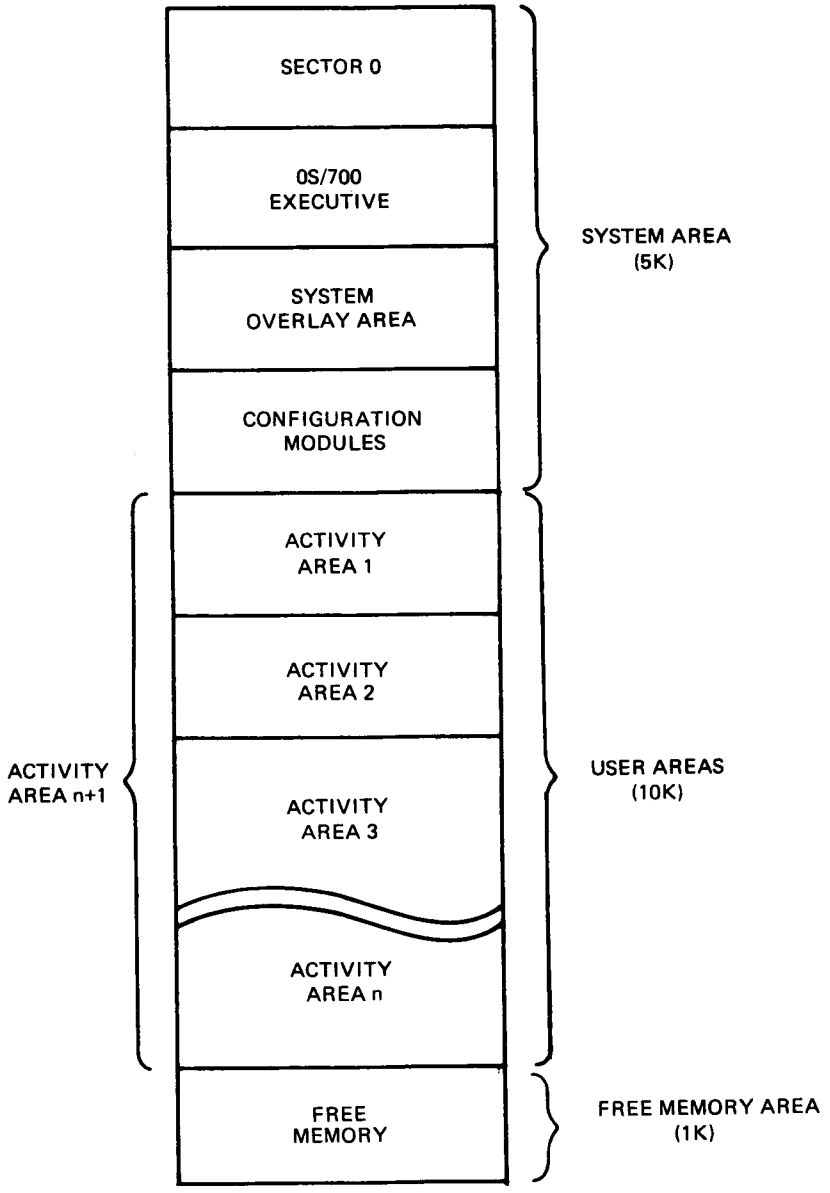


Figure 1-2. Memory Layout for Minimum System

Foreground-Background System Operation – Because of its priority assignment capability, OS/700 provides foreground-background operation. Normally, the foreground is used for high-priority operations and the background for program development and lower-priority operations.

Clock-Initiated Program Execution – OS/700 allows the user to execute programs at a particular time of day or on a periodic basis.

Overlay Capabilities – OS/700 provides overlay capabilities, allowing multiple use of memory areas by system and user programs.

Queuing Requests for Execution – OS/700 places requests for execution in separate queues according to their priority. It then executes those requests having the highest priority.

To summarize, OS/700 gives the user many of the important features of medium-scale computers in addition to real-time, modular, application-oriented capabilities. It allows the user to be extremely flexible, using System 700 for either dedicated applications and/or program development.

HARDWARE FEATURES

Some of System 700's more important hardware features – features which give OS/700 its many capabilities – include:

- Fast cycle times.
- Watchdog Timer, Clock Time-out, Power Failure, I/O, and other interrupt functions.
- Each type of interrupt automatically transfers execution to a separate dedicated memory location.
- Real-Time Crystal Clock.
- Push-down, pop-up stacks.
- Direct memory access for volume data transfers.

SECTION II

BASIC CONCEPTS

ACTIVITY AND TASK CONCEPTS

Since the operation of OS/700 centers around the concepts of activities and tasks, their definitions and relationships must be well understood before the features of OS/700 can be fully utilized.

Definitions

Programs run under OS/700 are termed activities. Activities are stored on disk and read into main memory as required (disk-resident) or are stored permanently in main memory (main memory resident). Disk resident activities can be stored on disk at any time; from the disk they are available for execution as needed. Main-memory-resident activities are specified at system configuration time and are loaded into main memory when the system is loaded.

Each activity consists of one or more tasks. A task is a logical function performed in an activity. More specifically a task is the execution of a set of instructions (task code) — not necessarily contiguous instructions — at a specified priority level.

Tasks may be interrupted by external interrupts. After the interrupt has been processed, the execution of the task code is resumed unless a request for a higher priority task has been made in the meantime. In addition the task may request suspension itself. A task can also voluntarily wait until the completion of an I/O operation, thus allowing other tasks to be executed.

A task can request another task, itself, another activity, or the activity in which it resides. In scheduling an activity or task, the user assigns a priority level which indicates the importance of its execution. The OS/700 Executive allows up to 16 separate priority levels to be specified and maintains a scheduling queue for each.

When an activity is scheduled, the Executive puts a request for the lead (first) task of the activity at the end of the appropriate queue. When a task is scheduled, the Executive puts the request at the end of the appropriate queue. When a task terminates, control is

returned to the Executive. The Executive then checks to determine which of the previously scheduled tasks should now be executed.

NOTE: The same task can be scheduled more than once by one or more tasks in the activity at the same or different priority levels. Therefore, there can be two or more requests at any time for a task in one or more of the scheduling queues.

A task is termed "dispatched" when the Executive starts execution of that task code. The Executive dispatches a previously scheduled task by going to the highest scheduling queue that contains one or more requests and selecting the top (first) request. Each time that control returns to the Executive after the termination of a task or after an interrupt has been processed, either a new task is dispatched or a previously interrupted task is resumed. If there are no requests in any of the scheduling queues, the Executive goes into an idle state, waiting for an interrupt.

Main memory has a specific area reserved for the user. This user area includes one or more activity areas as illustrated in Figure 1-2. After scheduling, A disk-resident activity is always loaded into the activity area for which it has been linked. More than one activity may be assigned to an activity area, but an activity is executed only in its own area.

The Executive maintains a queue for each activity area. If a scheduled disk-resident activity requires part or all of a main memory area in which another activity is currently being executed, the scheduled activity is placed in that area's queue. When the current activity has terminated execution, the Executive loads the next queued activity in that area.

If two disk-resident activities must be in main memory at the same time, the systems analyst must assign them separate, non-overlapping activity areas. Specifically, the analyst must configure main memory areas into which disk-resident activities are to be loaded so that there are no conflicts between activities which must be in memory at the same time. It must be remembered that the Executive always brings an entire activity into main memory.

Relationships

Because a task can be suspended, interrupted or can wait for I/O, it is possible under OS/700 to take advantage of the following situations:

- Two or more activities can be executed concurrently.
- Two or more tasks within an activity, having higher priority levels than a suspended task, can be executed concurrently.

- The same task, scheduled several times on two or more priority levels, can be be executed concurrently.

  NOTE: If a user schedules a particular task more than once, the task must be reentrant. An activity may be scheduled as many times as required without any restrictions.

Task code may be classified reentrant, reusable or non-reusable. A reentrant task can be interrupted, scheduled at a higher priority, dispatched and a return made to the execution of the interrupted task without loss of any data. A reusable task is not reentrant but can be executed serially with no need to reload itinto memory after each execution. A non-reusable task must be reloaded into memory each time it is to be used.

If a task is not reentrant and is currently scheduled, it may not be scheduled again until it has been terminated. A non-reusable task may only be scheduled once per loading of the activity in which it resides.

Activities can also be said to be reentrant, reusable or non-reusable. If all the tasks in an activity are reentrant, the activity is reentrant. If all the tasks in an activity are reusable (some can also be reentrant), the activity is reusable. If one or more of the tasks in an activity are non-reusable, the activity in non-reusable.

When an activity was initially made visible to the system, the user specified whether it was reentrant, reusable or non-reusable. (See OS/700 Utility Manual, Order No. AG14 for details.) This allows the Executive to perform as follows:

- If a reentrant activity, currently being executed, is scheduled again, the Executive will immediately put the request in the appropriate scheduling queue. Concurrent execution of the activity, or any task within it, is possible.
- If a reusable activity, currently being executed, is scheduled again, the Executive will delay scheduling of the lead task of that activity until after current execution of the activity is terminated. If an activity has been terminated and is still in main memory (has not been overlaid by another activity), the Executive will be able to dispatch the lead task of the activity at any time. If the activity is not in main memory, the Executive loads it in from the disk and then schedules it.
- If a non-reusable activity, currently being executed, is scheduled again, the Executive will wait until current execution is terminated, reload it from the disk and then schedule it.

Scheduling

When a user schedules an activity, the Executive performs one or more of the following basic operations:

- If the activity is already in main memory and is reentrant, it is immediately scheduled.
- If the activity is already in main memory, is terminated and is reusable, the Executive will reschedule it immediately.
- If the activity is already in main memory, is currently being executed and is reusable, the Executive will re-schedule it as soon as its current execution is terminated.
- If the activity in non-reusable or is not in memory and
- If part or all of the activity area into which it is to be loaded is not currently being used, the Executive will load it into memory and schedule it.
- If the activity must be loaded into memory and
- If part or all of the activity area into which it is to be loaded is currently being used by another activity, the Executive will keep checking until it finds the activity area is available; the Executive will then load the activity and shedule it.

The logical progression of these operations is illustrated in Figure 2-1.

When a task schedules another task, the following operations are performed by the Executive:

- A Task Control Block (TCB) is constructed, using the data in the TCB$ macro.
- A TCB is put in the scheduling queue whose priority is the same as the priority assigned to the task. When a task is dispatched the location of the TCB is passed to the task so that the programmer can obtain the task parameters and return the TCB when the task is terminated.

Summary

When utilizing acitivies and task concepts, the following major points whould be remembered:

- While the Executive requires that all of the code in an activity be in main memory before it is executed, execution of the activity is controlled at the task level.
- Priorities can be assigned to tasks and activities when they are scheduled.
- Scheduling is defined as placing the request for execution of an activity or task at the end of the scheduling queue specified.

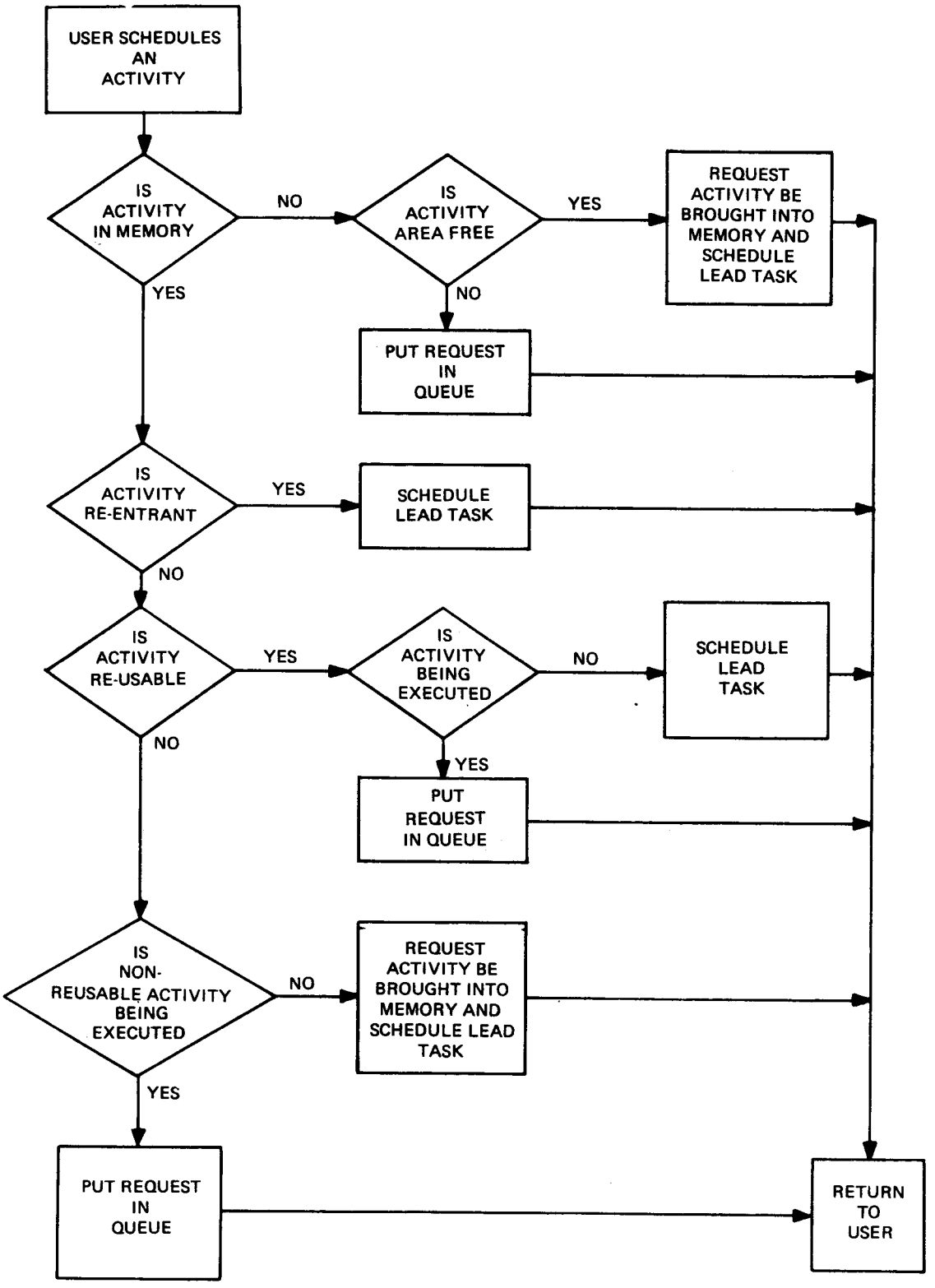


Figure 2-1. Logical Progression of Activity Scheduling

- Dispatching is defined as OS/700 initiating execution of the first scheduled task in the highest scheduling queue containing a request.
- Activities and tasks can be executed concurrently.
- Activities and tasks can be coded as reentrant, reusable or non-reusable.
- If an activity, currently being executed, is scheduled again, it can be immediately dispatched by the Executive if it is reentrant or can be dispatched after its current execution is terminated if it is reusable.
- However, non-reusable activities must be reloaded for each execution.

SUPPORT FUNCTION CONCEPTS

OS/700 provides these additional support functions:

- Clock-generated scheduling of activities and tasks.
- Management of interrupts and their processing.
- Dynamic allocation and deallocation of blocks of memory.
- Creation and use of queues.

Clock-Generated Activity and Tasks Scheduling

The Executive allows activities and tasks to be scheduled at any required absolute or relative time or at any specified interval. The scheduling may be specified in terms of the absolute time after midnight or the relative time in one of the following units: millisecond, half-second, second, and minute. The Executive maintains separate queues, one for the absolute time and one for each of the four relative units. Scheduling may occur once (non-cyclic) or at specified time intervals (cyclic).

The clock scheduling of an activity or task is identical to the normal scheduling of activities and tasks. However, the user also specifies:

- Whether absolute or relative time is required.
- The number of milliseconds, half-seconds, seconds or minutes.
- Whether the scheduling is to be non-cyclic (once) or cyclic (repeated).

Each time that a clock activity or clock task is requested, an entry is put in the appropriate clock timer queue. At regular intervals (as specified at system configuration time), a clock interrupt will occur[1], and the entries in each of the five queues examined.

[1] The clock interrupt can be configured to occur in multiples of 16.7 ms (at 60 cycles) or 20 ms (at 50 cycles) unless Fetaure 3000, the Real-Time Clock/Watchdog Timer, has been installed. Any activity or task whose time of scheduling falls between the current clock interrupt and the next one, will be scheduled at this interrupt time.

First, the entries in the millisecond queue are scanned. Any entries that have reached their specified time interval are scheduled. Cyclic entries are reset and left in the queue while non-cyclic entries are removed after they have reached their specified time interval.

Next, the Executive checks to see if a half-second interval has been reached. If it has, the entries in the half-second queue are scanned and processed as described above. If the second interval has been reached, the entries in the second queue are scanned and processed. When the minute interval is reached, the same operation is performed. After all entries whose time intervals have expired are scheduled, the Executive returns to normal execution.

The details of clock scheduling are described in Section III.

Interrupt Processing

There are a number of I/O - or processor-generated interrupts that may occur during the execution of a task. See the Programmer's Reference Manual, (Order No. AC72) for more detailed information on the effect of interrupts on task scheduling and dispatching.

The System 700 processor recognizes two basic types of interrupts: privileged and I/O. When either occur, the effect of the interrupts on the current task depends on the interrupt mode under which the processor is currently running. Interrupt modes include:

- Inhibited interrupt mode — in this mode, execution of the current task can only be interrupted by privileged interrupts.
- Enabled interrupt mode — in this mode, execution of the current task can be interrupted by any interrupt.

When an interrupt which will suspend current execution of a task occurs, the Executive performs the following operations:

- Saves the status of the interrupted task.
- Identifies the type of interrupt and transfers control to the appropriate response code which will process that interrupt.
- Resumes execution of the interrupted task when the processing of the interrupt is completed unless task that has a higher priority than the interrupted task has been scheduled in the meantime.

Dynamic Allocation and Deallocation of Free Memory

Under OS/700, the user may dynamically obtain one or more blocks of free memory. These blocks may be used for dynamic buffers, parameter lists, variable data of reentrant tasks, etc. The size of these blocks can be any power of 2 ranging from 2 to 512 words.

The size of free memory is defined at system configuration time. In addition, the number of blocks of various sizes that are to be reserved must be specified. The user may obtain blocks of memory ranging in size from 2 to 512 words although the system allocates blocks in powers of 2. If the block size required is not a power of 2 or is a power of 2 which is not configured, the Executive allocates a block of the next larger configured size.

If a request is made for a block of a certain size and there are none currently available, the Executive can obtain a block of the next larger size, split it up into two or more smaller blocks and allocate one of these blocks to the task requesting it. Once a block has been fragmented, it remains so. The system does not re-construct fragmented blocks.

All requested blocks of free memory must be deallocated by the task. When the task deallocates a block, the Executive returns the block to free memory.

Creating and Using Queues

OS/700 allows dynamic creation and manipulation of queues. Specifically, the user can: Create a queue, Add an item to either the beginning or the end of the queue, and Remove an item from the beginning of the queue.

> NOTE: The user can have either LIFO (Last In, First Out) or FIFO (First In, First Out), queues, depending on whether he adds new items to the beginning or end of the queue.

I/O CONCEPTS

The Executive allows the user to request a number of logical and physical I/O operations.

Logical I/O

OS/700 allows the following file management capabilities:

- Maintenance of files on the system disk.
- Fixed length records in a file.
- Sequential access of records in a file.

With these capabilities, the OS/700 user can create, read, or delete a file.

A file consists of one or more logical records (one or more words of data) all of which are the same size. There is no upper limit on the size of a logical record except the size of available memory and there is no upper limit on the size of a file except the amount of available space on the disk.

To create a file, the file must be opened in the output mode. Records are then stored sequentially in the file and the file is closed in the normal mode. To read a file, the file must be opened in the input mode. Records are then read sequentially from the file in the order that they were stored (they can not be read from the file in random order). The file is then closed in the normal mode. To delete a file which has been opened in either the input or output mode, the file must be closed in the delete mode.

All operations affecting a file are controlled by the data previously stored in a File Control Block (FCB). Before starting a logical I/O operation, the user must have previously created the file's FCB, supplying the address of a 3-word sequence that contains the file name and the address of the buffer which contains the record to be transferred (for output) or in which the record is to be placed (for input).

Physical I/O

In addition to logical I/O, OS/700 also provides physical I/O capabilities. These physical I/O operations include: controlling the operation of all I/O devices in the system, and transfer of data between main memory and any I/O device in the system. Physical I/O operations may be performed on any one of the following I/O devices:

- ASR-33, ASR-35, KSR-33
- Paper Tape Reader or Punch
- Card Reader/Punch or Card Reader
- Moving or Fixed Head Disk Subsystem
- 7- or 9-Track Magnetic Tape Equipment
- Line Printer

The physical I/O operations that can be performed on one or more of these I/O devices include transferring data to or from an I/O device, positioning a magnetic tape to another file or record, rewinding a magnetic tape, and putting an end-of-file mark on paper or magnetic tape.

Operations affecting I/O requests are controlled by the data previously stored in a Device Control Block (DCB). Before starting physical I/O operations, the user must have previously created a DCB, containing:

- Type of I/O device.
- Logical unit number.

- Data mode (ASCII, binary, etc.).
- User identification.
- Location of the status block.
- Location for return after I/O operation completion.

Once the DCB has been created, the desired device must be reserved. When the device becomes availalbe to the user, the user can then request physical I/O operations for the reserved device. All I/O requests for each device are placed in separate queues and are acted upon in the order of their priority.

At the end of the physical I/O operations for the reserved device, the user releases the I/O device so that is again available if one or more reserve requests have been placed in the reserve queue. In addition, the user is able to set up temporary work areas on the system disk. These temporary work areas can be used to store and retrieve data. These work areas are temporary because once the system has been shut down and re-started, the data they contain cannot be retrieved by the user.

OPERATOR INTERFACE CONCEPTS

The user has the capability of sending messages to the operator during the execution of a task code and, when required, of accepting keyboard inputs from the operator in response to the messages. Thus, the user can include on-line interaction with the operator as part of a task, both in sending messages to and accepting keyboard inputs from the operator. In the case of two or more messages to the operator, the system provides the operator the capability to identify the message to which he is responding.

COMMUNICATION CONCEPTS

A full range of capabilities are available for performing communication functions under control of the OS/700 Executive. Programs may send and receive data to communication terminals, receive and change the status of terminals and also receive information concerning the validity of data and integrity of the communication elements in the OS/700 system: communication functions, communication tasks, logical and physical terminals, and status and errors are discussed below. Then the communication macros are described.

Communication Functions

Program using the OS/700 communications functions will send and receive data to remote terminals as messages or message units. A message is a string of characters of

variable lengths with a beginning (start of message) and an end (end of message). Each message is independent of others and complete whereas a message unit is a subset of a message and has a fixed maximum length. One or more message units compose a message and these are the basic units of data transmitted and received from remote terminals.

Communication data is placed into communication buffers upon receipt or prior to transmission. These are composed of characters packed into linked strings of free memory blocks (message units) residing on input or output queues.

The specific characteristics of communication lines and terminals are established at configuration time independent of communication tasks. Thus the direct communication with and control of remote terminals is handled by the communication elements of OS/700. This includes polling and selection of terminals, code conversion, error detection and correction and maintaining the proper line procedure for the terminal. Polling is the procedure for requesting a terminal to send a message. Selection is the reverse process for and is the procedure for sending a message to a terminal. Code conversion is performed from the code set used by OS/700 to that required by the terminal on transmission and likewise from the terminal to that of the terminal on reception.

Communication Tasks

Programs utilizing the communicaton functions of OS/700 are run as communications tasks. These are scheduled and dispatched based on data reception from a terminal and the task connection to the terminal. This connection is via a routing task control block (RTCB) and is logical. A physical connection exists between the terminal and the communication controller via the communication line.

The connection or disconnection can be effected either dynamically at run time or when the system is configured. However, an implicit logical connection always exists with a communication task designated during system configuration called the default supervisor. Each terminal can only be connected to one task at a time. The task must disconnect to allow another task to communicate with the terminal. It is during the time that no task is connected to the terminal that the default supervisor will receive all the terminal data for transfer to the terminal.

Logical and Physical Terminals

Just as there exists logical and physical connections, there are logical and physical terminals. A logical terminal is called a station and is the identification used by communications tasks when logically connected. A physical terminal is identified by the communication controller (device), communication line(s) associated with that device and the terminal which may be physically connected to that communication line.

The system translates from station to terminal and from terminal to station during the sending and receiving of messages by communication tasks respectively. This association is established during system configuration.

Communications tasks can receive and send messages to stations only while connected. Thus data may be in the form of peripheral buffers or communications buffers. A peripheral buffer is compatible with that utilized by I/O devices other than communications devices. The system will automatically translate from one form to the other if desired.

Status and Errors

Besides handling all the transmission and reception of data to terminals, the system monitors each communication line and terminal for errors and changes of status. These conditions are reported either to the communication task or to a system related program called the communications supervisor.

Error in data, terminal malfunctions, successful and unsuccessful transmissions or reception of messages are reported to the connected communication task along with the received messages or as output status messages. The communication task may also request physical terminal status or change that status if desired.

All malfunctions in communication controllers, lines or terminals are reported directly to the communications supervisor. This program may in turn send the information to an error supervsior or to the system console via the operator interface. In turn, commands to change communication status can be entered into the system via the same device interfaced to the communication supervisor.

SYSTEM MACRO CONCEPTS

All the operations that have been described up to this point are performed under OS/700 and can be requested by the user. This section describes: the interface between the system and the user and the way in which the user can request system functions.

User-System Interface

The boundary between OS/700 and the user's activities must be clearly defined and observed. In addition, the method by which the user requests such system functions must be clear. Figure 2-2 illustrates that the only direct contact the user has with OS/700 is via macro calls. Otherwise, OS/700 and its operations are invisible to the user. OS/700, on the other hand, is able to interact with the user's activitities and tasks when requested by the user via a system macro call.

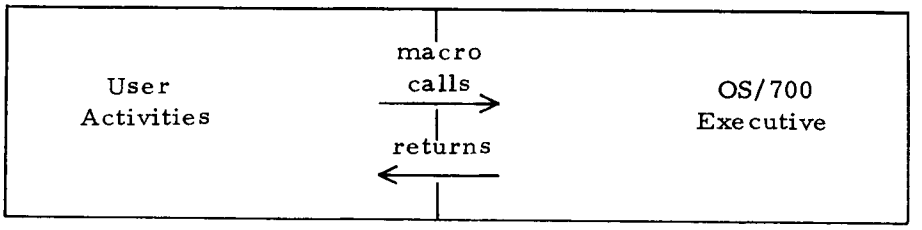


Figure 2-2. User, OS, 700 Interaction

In order to simplify the writing of reentrant coding, the system supplies the user with 6 pseudo-registers — ZCR1 through ZCR6. The registers are treated by the system in a manner similar to the hardware registers. These registers are to be referenced indirectly by the user.

When the user makes a system macro call from which a return to his task is to be made, the status of the various registers on entry to the Executive is as follows:

- Pseudo-registers ZCR1, ZCR2, and ZCR3 are saved.
- Pseudo-registers ZCR4, ZCR5 and ZCR6 are not saved.
- Hardware registers B and S are saved.
- The A-register will contain the macro call return address.
- The X-register will contain the address of the macro call parameter list.

There are 3 types of system macros:

- System macros that request system functions.
- System macros that define data for system functions.
- A system macro (LNK$) that provides the interface or linkage between the user activity and the OS/700 Executive.

Each independently-assembled segment of code having an END statement and containing one or more system macros that perform system functions must contain a LNK$ macro.

System Macro Calls

The user can request certain system functions by means of system macros. A system macro call consists of a macro name ending in a dollar sign ($) and a parameter set. The format of the system macro call is illustrated below:

Operation	Operand
MAC$	param 1, [param2],..,param n

where:

MAC$ is the system macro name

param 1, [param2],...., param n are the parameters, seprated by commas.

Param 1 is required; as indicated by the brackets, param 2 is optional

The parameters are positional (i.e., their positions must be preserved by commas if the parameters are omitted).

Another example of the macro format is shown below:

Operation	Operand
MAC$	buffer address, range

where:

buffer address is the address of the buffer

range is the integer giving the maximum number of words in the buffer

If the macro is used as defined in the example, it could be written as follows:

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6	7 8 14	15 16 31	32
	MAC$	BUFF,13	BUFF AND 13 ARE DEFINED PARAMETERS
	.		
	.		
	.		
	.		
BUFF	BSZ	13	

When an exclamation point (!) is placed before a parameter, it indicates that it is the address of a word containing the parameter defined in the macro description. If the maximum number of words in the buffer is to be computed during execution, will be no greater than 25 and will be stored in location MAXN0, the following statements could be written:

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6 7	8 14 15	16 31	32
	.		
	.		
	STA	MAXNO	STORE BUFFER SIZE
	.		
	.		
	.		
	MAC$	BUFF,!MAXNO	BUFF IS A DEFINED PARAMETER.
	.		MAXNO IS A WORD CONTAINING THE
	.		DEFINED PARAMETER
	.		
MAXNO	BSZ	1	MAXIMUM NUMBER OF WORDS IN BUFFER.
BUFF	BSZ	25	

When an asterisk (*) is placed before a parameter, it indicates that this is the address of a word containing the address of a parameter defined in the macro description. If a block is obtained from free memory and the address of this block is stored in T1 and if the maximum number of words in the buffer is stored in the 6th word of that block, the following statements could be written.

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6 7	8 14 15	16 31	32
	LDA	T1	SAVE BLOCK ADDRESS IN T1
	ADD	=5	DISPLACEMENT IN BLOCK
	STA	T2	ADDRESS OF MAXIMUM NUMBER OF WORDS
	.		
	.		

```
                .
        MAC$    BUFF,*T2        BUFF IS DEFINED PARAMETER.
                .               T2 IS A WORD CONTAINING THE ADDRESS
                .               OF THE WORD WHICH CONTAINS THE
                .               DEFINED PARAMETER
T1      BSZ     1
T2      BSZ     1
BUFF    BSZ     25
```

Returns From a System Macro

After a system macro call has been processed successfully and a normal return is taken, the next instruction after the system macro call is executed. If the system macro call is not successfully processed, an error return is taken to the address of a routine, specified in the system macro, which is to process the error. In such cases, the A-register will contain a code which indicates the type of error.

On return to a user task from a system macro call, the status of the various registers is as follows:

- Pseudo-registers ZCR1, ZCR2 and ZCR3 are restored.
- Pseudo-registers ZCR4, ZCR5 and ZCR6 are not restored.
- Hardware registers B and S are restored.
- The A-register will contain the error code if the error return is taken. Otherwise, the contents of the A-register are undefined.
- The X-register will contain a return parameter if any. Otherwise, the contents of the X-register are undefined.

NOTE: An exception to the above can occur if a Wait For I/O macro (WIO$) is used in which the I/O status block address pointer parameter is omitted. In this case, the contents of the pseudo and hardware registers is not guaranteed.

Out-Line Paramter Lists and Reentrant Coding

When a system parameter is used in a reentrant activity or task, the standard "in-line" macro parameters can not be put after the macro name. Instead, the parameters must be

stored in a list in the data section. In addition, the user will dynamically store his variable data in blocks obtained from free memory and may use the system pseudo-registers — ZCR1, ZCR2 and ZCR3 — to save the address of these blocks.

An example of an out-line parameter list is shown below:

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6	7 8 14	15 16 31	32
	LDX	LIST	ADDRESS OF PARAMETER LIST PUT IN X-REGISTER
	MAC$	(X)	SYSTEM MACRO
	---		NORMAL RETURN
	.		
	.		
	.		
BUFF	BSZ	13	THIRTEEN WORDS RESERVED FOR BUFFER
LIST	DAC	*+1	
	DAC	BUFF	
	DAC	13	

NOTE: The only way a macro call with an out-line parameter list can be written is by using an X enclosed in parentheses (X) as illustrated in the above example.

The description of each system macro will specify the format required for the out-line parameter list as well as giving an example of it.

User and System Domains

The execution of code under OS/700 may take place in either the system or the user domain. The system domain is defined as:

- no relocation of base sector
- extended addressing mode only
- single precision only
- stack interrupt disabled
- trace interrupt disabled

The user domain may be set to:

- relocation of base sector
- extended addressing mode only
- single or double precision
- stack interrupt enabled
- trace interrupt enabled

When a return is made from the system domain to the user domain as the result of a macro call, the user's state is restored, with the exception that, interrupts are enabled and the extended addressing mode is set.

SPECIAL SYSTEM MACROS

In this section, three special macros are described in detail. The Link macro (LNK$), as already mentioned, is required whenever system macros that perform system functions are used in a segment of coding that is independently assembled and contains an END statement. The Get System Parameters macro (GSP$), is used to obtain parameters that were specified at system configuration time. The Task Control Block macro (TCB$), allows the user to access data in the Task Control Block via symbolic names.

LINK MACRO (LNK$)

The LNK$ macro provides the proper user/system interface between all executable system macros and the Executive. A LNK$ macro call must be in each independently assembled block of code that contains executable macro instructions.

LNK$ Macro Action

LNK$ saves the address of the executable macro, enables extended addressing, and transfers control to the Executive for processing of the executable macro. The LNK$ must reside in user's data definition code area.

Macro Format

Location	Operation	Operand
	LNK$	

Normal Return: None.

Error Return: None.

Macro Action Details

This macro saves the address of the executable macro, enables extended addressing, and transfers control to the Executive for processing of the executable macro.

Out-Line Parameter List: Not applicable.

GET SYSTEM PARAMETERS (GSP$)

The GSP$ is used to obtain certain system parameters associated with the system disk and the operator console.

GSP$ Macro Action

GSP$ obtains a predefined set of system parameters and places these parameters in the user-supplied block.

Macro Format

Location	Operation	Operand
[symbol]	GSP$	parameter block address pointer, parameter set type error return address

where:

symbol — Optional

Is the symbolic label of the macro instruction

parameter block address pointer

Is the address of a word containing the address of a 1-, 2-, 3-, or 4-word block where the system parameters defined by the parameter set type are to be stored.

parameter set type

Is an integer specifying the particular set of system parameters defined below.

Parameter Set Type	Block size	Word No.	Items
1	4	0	system disk generic device type
		1	logical unit number for system disk
		2	segment size
		3	number of segments per work area
2	3	0	start segment number for activity directory
		1	index number of segments in activity directory index — (2's complement)
		2	start segment number for activity directory

Parameter Set Type	Block Size	Word No.	Items
3	3	0	start segment number for file directory index
		1	number of segments in file directory index
		2	start segment number for file directory
4	2	0	user area start address
		1	user area end address
5	1	0	operator console generic device type

error return address

The address to which control is returned if an error is found during the processing of the macro call.

Normal Return

Control is returned to the caller at the instruction following the macro.

Error Return

Control is returned to this address if the parameter set type number is out of the range 1 - 5.

Macro Action Details

The parameters associated with the parameter set number are placed in the block pointed to by the parameter block address pointer.

Out-Line Parameter List

Word No.	Operation	Operand
0	DAC	parameter block address pointer
1	DEC	parameter set type
2	DAC	error return address

where:

parameters in the out-line parameter list are the same as those described for the in-line parameter list.

Examples

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6	7 8 14	15 16 31	32
	GSP$	BLKPTR,1,ERX	
	.		
	.		
	.		
	.		
BLKPTR	DAC	BLOCK	BLOCK FOR SYSTEM PARAMETERS
	.		
	.		
	.		
BLOCK	BSZ	4	4 WORD BLOCK

An example of an out-line parameter list is shown below.

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6	7 8 14	15 16 31	32
	.		
	.		
	.		
	LDX	PLIST	
	GSP$	(X)	
	.		
	.		
	.		
	.		
PLIST	DAC	LIST	

```
        .
        .
        .
LIST    DAC     BLOCK       BLOCK FOR SYSTEM PARAMETERS
        DEC     1           PARAMETER SET TYPE
        DAC     ERX         ERROR RETURN ADDRESS
        .
BLOCK   BSZ     4           4 WORD BLOCK
```

TASK CONTROL BLOCK (TCB$)

The TCB$ macro provides the user with the capability of accessing specific words in the TCB with symbolic names instead of numeric displacements.

TCB$ Macro Action

TCB defines by EQU's the relative position of words in the TCB which must be user-visible.

Macro Format

Location	Operation	Operand
	TCB$	

Macro Action Details

The following relative locations within a TCB are defined:

ZTCBST	status
ZTCBP1	parameter 1
ZTCBP2	parameter 2
ZTCBCW	TCB control word
ZTCBPT	pointer to primary TCB
ZTCBAC	ACB address

Out-Line Parameter List

Not applicable.

Examples

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6	7 8 14	15 16 31	32
	.		
	.		
	TCB$		
	.		
	.		
	.		
	.		
* UPON	ENTRY	TO TASK, X REGISTER POINTS TO TCB	
	.		
	LDA	ZTCBP1,1	GET PARAMETER 1 FROM TCB
	.		
	.		
	.		
	.		
	LDA	ZTCBP2,1	GET PARAMETER 2 FROM TCB

SECTION III
SCHEDULING ACTIVITIES AND TASKS

The following system macros are used to initiate activity, task and clock operations: activity macros, task macros, and clock macros.

ACTIVITY MACROS

The two activity macros — Schedule Activity (SAC$) and Terminate Activity (TMA$) — are used to schedule activities at any point in a task and to indicate that the current activity is being terminated.

An activity comprises a collection of instructions and data which form one or more task codes. All of the activity resides in one contiguous main memory area, called the activity area, when it is executed. An activity, if disk-resident, is considered as one complete logical entity when brought into main memory from disk. The entrance to an activity resides in the task code defined as the lead task of the activity. It is this task which is scheduled when the activity is requested.

Activities may be classified as non-reusable, reusable, or reentrant. An activity is non-reusable if it must be reloaded prior to each execution; consequently, it must always be disk-resident. A reusable activity is one which must be reinitialized (by internal code) prior to each execution. It does not necessarily need to be reloaded. A reusable activity must run to completion before being executed again.

A reentrant activity may be interrupted and executed (at a higher priority level) on behalf of an entirely different input parameters without destroying the former state. Later, execution can be continued in the former state as though the activity had not been interrupted. The variable data areas required for a reentrant activities are obtained from free memory.

An activity must remain in main memory until all of its action is completed. All I/O must be finished; all clock tasks must be terminated (if non-cyclic) or disconnected (if cyclic); all blocks obtained from free memory must be returned unless they are being passed on to another activity. All activities must terminate via the Terminate Activity (TMA$) macro.

An activity is defined when it is made system-visible. This is accomplished by the utility function Load Activity (see OS/700 Utility Manual, Order No. AG14). At this time the activity is placed on the disk and an entry for it is made in the activity disk directory. This directory contains pertinent information concerning the activity, contains pertinent information concerning the activity, including the name and the default priority level.

The name of an activity comprises six or less alphanumeric characters, the first of which must be alphabetic. This name is used to identify an activity when it is scheduled (see SAC$ macro). The default priority level is the level at which the lead task of the activity will be scheduled if the level parameter in the SAC$ macro is omitted. If the level parameter is omitted in the STS$ macro when scheduling tasks in this activity, this default level is used.

Reentrant Activities

A reentrant activity is one which may be interrupted and executed again on behalf of a different set of input parameters without destroying the interrupted state. The interrupted state may then be resumed as though no interruption had taken place.

In order to achieve reentrancy all task code must observe the following rules:

- All task code must be pure procedure. Task code must not contain any words (instructions or data) which are changed.
- All variable information must be contained in variable data areas.

OS/700 provides the capability to obtain blocks of free memory (see GBL$ and RBL$ macros) which may be used as variable data areas.

Also, the Executive saves the state of a task when it is interrupted and restores this state when the task is resumed. The state which is preserved and restored includes:

The hardware registers: A, B, X, S, P, and keys.
Three pseudo-registers: ZCR1, ZCR2, ZCR3.

The three psuedo-registers are provided by the Executive and may be referenced from an activity indirectly through pointer words. The three pseudo-registers must be declared as externals in any source module using them unless the source module includes a LNK$ macro.

To make a task code reentrant, obtain a block of free memory for a variable data area, and keep the address of this block in one of the hardware registers or pseudo-registers.

Activity Interaction

Activity interaction may occur in two ways: scheduling and monitoring. An activity may schedule another activity using the SAC$ macro or one activity may monitor another activity. The monitoring activity, called the "master," controls the general action of another activity, called the "slave." In order to do this the secondary TCB parameter of the SAC$ macro is used.

The secondary TCB defines a secondary task within the master activity. When the slave activity is brought into main memory this secondary task is scheduled in place of the lead task of the slave activity.

When the secondary task is dispatched, the master activity has control with the slave activity being main memory resident, and monitoring may take place. When the secondary task is dispatched, the X-register contains the address of the secondary TCB. The address of the lead task TCB is in the word whose displacement is ZTCBPT with respect to the beginning of the secondary TCB. It is the responsibility of the master activity to insure that the slave activity is terminated. This may be accomplished in two ways:

- Schedule the lead task of the slave activity and allow the slave activity to execute to completion and terminate itself.
- Terminate the slave activity from within the master activity.

The latter method of terminating a slave activity involves use of special capabilities of the TMA$ macro. One of the parameters of this macro defines a TCB which is to be returned to free memory upon termination of the activity. The master activity, wishing to terminate the slave activity, can issue a TMA$ macro call, with the TCB of the lead task of the slave activity as a parameter. The TMA$ action terminates the slave activity, but returns control back to the master activity, rather than returning to the Executive.

NOTE: Regardless of the method used to terminate a slave activity, the master activity must terminate itself. In addition, master and slave activities must be main memory resident at the same time; therefore, they must occupy different activity areas.

SCHEDULE ACTIVITY MACRO (SAC$)

A request for the execution of an activity is made via the SAC$ macro.

SAC$ Macro Action

SAC$ creates a TCB for the activity lead task after obtaining a block from free memory. It then places the request to schedule the activity in the activity's request queue and returns to the caller. Later, when the activity's memory area becomes available, SAC$ brings the activity

into main memory and schedules the lead task of the activity (or the task defined by the secondary TCB).

Macro Format

Location	Operation	Operand
[symbol]	SAC$	activity name address, [level], [parameter 1], [parameter 2], error return address, [secondary tcb address]

where:

symbol — Optional

Is the symbolic label of the macro instruction.

activity name address

Is the address of the first word of the six-character name of the activity. The name is left-justified, right-filled with spaces.

level — Optional

Is the user priority level number of which the activity lead task is to be scheduled If a level number lower than the lowest user priority level is specified, the lowest user priority level will be used to schedule the task.

If this parameter is omitted, the task is scheduled on the default level specified in the ACB of the activity.

parameter 1 — Optional

Is a parameter word that can be used by the caller to pass data to the lead task. This parameter is in word ZTCBP1 of the lead task TCB.

If this parameter is omitted, a zero word is generated in the TCB for word ZTCBP1.

Note that if parameter 2 is not zero, parameter 1 must not be zero unless the free memory block parameter passing technique is being used.

parameter 2 — Optional

Is a parameter word that can be used by the caller to pass data to the lead task. This parameter is a word ZTCBP2 of the lead task TCB.

If this parameter is omitted, a zero word is generated in the TCB for word ZTCBP2.

Note that if parameter 1 is zero, parameter 2 must be zero unless the free memory block parameter passing technique is being used.

error return address

Is the address to which control is returned if an error is found during the processing of the macro call.

secondary TCB address — Optional

The address of a secondary TCB (defining a secondary task) which will be dispatched in place of the lead task of the activity.

If omitted, there is no secondary TCB, and the lead task will be dispatched.

Normal Return

Upon normal return, the request for the activity has been queued. The lead task of the activity (or secondary task) may have already been scheduled, if the activity is reentrant or reusable. If not already scheduled, the lead task will be scheduled at a later time.

Error Return

Control is returned to the error return address specified in the SAC$ macro with the error code in the A-register when any of the following errors are detected:

A-register Contents (Decimal)	Error Condition
14	Activity not on disk
18	Disk error while referencing the disk directory, stored on the disk.
22	No activity area for the activity, or the activity is too large to fit into its area.

Because the loading of the actitivy from the disk is done at a later time than at the SAC$ call, a load error cannot return to the error return specified in the macro call. Instead, an error message is output to the operator.

Operator messages			Error Condition
SE = 100030	000000	actnam	Activity area will be overrun.
SE = 100031	000000	actnam	Disk error while loading the activity.

where: actname is the name of the activity being loaded.

Macro Action Details

The activity is identified. If the activity is not already resident in main memory or on the disk (defined), the error return is taken. If it is defined, the request for the activity's execution is placed on the activity's request queue. Return is to the normal return.

At a later time, when the area for that activity becomes available, the activity is brought into main memory from the disk, and the lead task is scheduled. If a secondary TCB has been specified, the secondary TCB is scheduled.

Out-Line Parameter List

Word No.	Operation	Operand
0	DAC	activity name address
1	DAC	level
2	DAC	parameter 1
3	DAC	parameter 2
4	DAC	error return address
5	DAC	secondary tcb address

where:

Parameters in the out-line parameter list are the same as those described above for the in-line parameter list.

Notes

When the lead task or the secondary task is dispatched, the location of the lead or secondary TCB will be in the X-register. The words which are of interest to the user are:

ZTCBST — Status word.
ZTCBP1 — Parameter 1.
ZTCBP2 — Parameter 2.
ZTCBPT— Pointer to lead task TCB if secondary task has been dispatched.

The above symbols have definitions which are displacements relative to Word 0 of the TCB. Thus, for example, if upon start of the lead task parameter 2 must be examined, the coding would be:

```
LDA   ZTCBP2, 1   where the index register points
                  to the start of the TCB.
```

The status word appears in either a lead task TCB or the secondary task TCB. It has the following meaning:

Bits 1-2 — Task designation (binary)

00 — reserved.
01 — TCB is for lead task.
10 — TCB is for secondary task.
11 — reserved.

Bits 3-16 — Status (decimal)

0 — no errors.
2 — Activity area overrun error while loading the activity.
3 — Disk read error while load the activity

Status number 2 and 3 are errors returned to the secondary task only if the secondary task is not part of the same activity as the lead task.

The user is responsible for generating a secondary TCB. The Create TCB macro (CTC$) may be used to do this.

Examples:

In the following example, an activity, whose name is ACTNAM, is to be scheduled on its default level. Two parameters are passed to it, the address of T1 and the value 139. The error return is at ERR.

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6 7	8 14 15	16 31	32
	SAC$	NAMP,,T1,139,ERR	
	:		
	:		
NAMP	BCI	3,ACTNAM	ACTIVITY NAME
*			
T1	BSZ	1	ADDRESS OF T1 IS A PARAMETER
ERR	---		ERROR RETURN

If, in the above example, the level at which the activity is to be scheduled is found in word LEVEL, then the macro call would be written as:

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6 7	8 14 15	16 31	32
	SAC$	NAMP,,LEVEL,T1,139,ERR	

The following example is the same as the example above, except that the in-line parameter list is not used.

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6 7	8 14 15	16 31	32
	LDX	PLIST	
	SAC$	(X)	
	:		
	:		
PLIST	DAC	*+1	

```
        DAC     NAMP
        DCT     0
        PAC     T1
        DEC     139
        DAC     ERR
```

where:

NAMP, T1, and ERR have the same definition as in the first example.

TERMINATE ACTIVITY MACRO (TMA$)

The TMA$ macro is used to terminate either the activity in which it appears or another activity.

TMA$ Macro Action

TMA$ returns the TCB, if present, to free memory, if the TCB is to be released. TMA$ then terminates the activity and returns to the caller or exits to the Executive which dispatches the highest priority scheduled task.

Macro Format

Location	Operation	Operand
[symbol]	TMA$	[tcb address pointer], error return address

where:

symbol — Optional

Is the symbolic label of the macro instruction.

tcb address pointer — Optional

Is the address of a word containing the address of the Task Control Block associated with the activity to be terminated. The activity associated with the activity control block pointed to by the task control block is terminated. If this parameter is omitted, the current activity is terminated.

error return address

Is the address to which control is returned if an error is found during the processing of the macro call.

Normal Return

Control is returned to the caller, at the instruction following the TMA$ macro, if an activity is terminating another activity. If an activity is terminating itself, TMA$ exists to the Executive and control does not return to the caller.

Error Return

Currently, the error return is not taken, but is present for future use.

Macro Action Details

If the TCB address pointer parameter is present, the TCB is examined to determine if its block is to be returned to free memory. If so, the block is released. If the TCB address pointer parameter is omitted or if the parameter is the address of a word containing zero for the TCB address, no block is returned to free memory.

The activity associated with the ACB pointed to by the specified TCB is the activity to be terminated. If the parameter is omitted, the current activity is terminated. This termination may result in the lead task or secondary task being scheduled if the activity is reusable or non-reusable and there are queued requests. If there are no queued requests for the terminating activity or if this is the last of the dispatched requests for a reentrant activity, the activity area occupied by the activity is freed for use by other activities that are queued for the same activity area. If there are no queued requests for the same activity area, the area of main memory is freed for use by any intersecting activity area that has queued requests.

TMA$ then determines if it was called by an activity terminating itself or by one activity terminating another activity. If an activity is terminating itself, TMA$ exits to the Executive and does not return to the caller. If one activity is terminating another activity, TMA$ returns to the caller at the instruction following the TMA$ macro.

NOTE: Before using the TMA$ macro to terminate an activity, the user must:

1. Make sure all I/O initiated by the activity is completed.
2. Release all reserved I/O devices.
3. Close all opened files.
4. Return all blocks obtained from free memory.
5. Disconnect all clock tasks that reside in the activity.
6. Make sure all tasks in the activity have terminated.

Out-Line Parameter List

Word No.	Operation	Operand
0	DAC	[tcb address pointer]
1	DAC	error return address

where:

parameters in the out-line parameter list are the same as those described above for the in-line parameter list.

Examples

In the following example the task is terminated using a non-reentrant in-line parameter list.

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6	7 8 14	15 16 31	32
START	STX	PTTCB	SAVE TCB ADDRESS
	...		
	TMA$	PTTCB,ERA	TERMINATE ACTIVITY
	...		
ERA	HLT		ERROR HALT
	...		
PTTCB	DAC	**	TCB ADDRESS POINTER

In the following example the task is terminated using an in-line parameter list but it is reentrant because the parameter is in a pseudo register.

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6	7 8 14	15 16 31	32
START	STX	ZCR1	SAVE TCB ADDRESS
	...		
	TMA$	ZCR1,ERA	TERMINATE ACTIVITY
	...		
ERA	HLT		ERROR HALT

In the following example the task is terminated using a non-reentrant outline paramerer list.

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6	7 8 14	15 16 31	32
	...		
	LDX	PPLIST	PARAMETER LIST ADDRESS TO X
	TMA$	(X)	TERMINATE ACTIVITY
	...		
ERA	HLT		ERROR HALT

	...		
PPLIST	DAC	PLIST	PARAMETER LIST ADDRESS
PLIST	DAC	PTTCB	PARAMETER LIST, TCB ADDRESS PTR
	DAC	ERA	ERROR RETURN ADDRESS
	...		
PTTCB	DAC	TCB	TCB ADDRESS
TCB	BSZ	8	TCB

In the following example the task is terminated using a reentrant out-line parameter list.

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6	7 8 14	15 16 31	32
START	STX	ZCR1	SAVE TCB ADDRESS
	GBL$	2	GET BLOCK FOR PARAMETER LIST
	STX	ZCR2	SAVE PARAMETER BLOCK ADDRESS
	LDA	PZCR1	STORE POINTER TO TCB ADDRESS
	STA	0,1	IN WORD 0 OF PARAMETER LIST
	LDA	PERA	STORE ERROR RETURN ADDRESS
	STA	1,1	IN WORD 1 OF THE PARAMETER LIST
	...		
	LDX	ZCR2	PARAMETER LIST ADDRESS TO X
	TMA$	(X)	TERMINATE ACTIVITY
	...		
ERA	HLT		ERROR HALT
	...		
PZCR1	DAC	ZCR1	
PERA	DAC	ERA	

TASK MACROS

The five task macros — Schedule Task (STS$), Suspend Task (SUS$), Terminate Track (TMT$), Create a Task Control Block (CTC$), and Schedule a Task Control Block (STC$) — are used to:

- Schedule a task during execution of the current task.
- Suspend execution of the current task so that any other tasks with higher priorities can be dispatched.
- Indicate that a task is terminated.
- Create a Task Control Block (TCB) without scheduling the task.
- Schedule a task for which a TCB has already been created.

User and System Priority Levels

The OS/700 Executive dispatches tasks on a priority basis. The Executive provides for a maximum of 16 absolute priority levels with the number of levels in a given system being determined at system configuration time. The absolute priority levels are divided into two types: System Levels and User Levels. The system levels are the highest priority absolute levels and are reserved for use by Executive tasks. The user levels are below the system levels in priority and are the levels on which user tasks and activities are executed.

The absolute priority levels are assigned numbers from 0 to 15 with 0 being the highest priority level and 15 the lowest priority level. The system level numbers are the first S absolute level numbers (0 through S-1). The user levels are assigned under level numbers 1 through 16-S. It is these user level numbers that are used when scheduling tasks or activities by the STS$, STC$ and SAC$ macros. In these macros, the priority level number specified in the macro parameter is always biased by the number of system levels to determine the absolute level on which the task or activity will be scheduled. Because the user level number is biased, the user is prevented from running on any level reserved for the Executive.

In the macros that provide for a user level to be specified, a user level number of 0 will be the same as if the parameter was omitted and a default level will be used. Usually the default level is the level specified as the default level for the activity in which the task resides. If a user level number greater (lower priority) than the lowest priority user level in the system is specified, the lowest priority user level will be used.

Task Control Blocks

Associated with each task as it executes is a block of data, the Task Control Block (TCB). The TCB contains task-related data including: a status word, task entry address, task absolute priority level number, task parameters 1 and 2, and the address of the ACB associated with the activity in which the task resides. If the TCB is a secondary TCB, it has the address of the lead task's TCB. Normally, the user need not concern himself with the detail contents of the TCB or with the generation of TCB's. When tasks are scheduled via the STS$ macro and activities via the SAC$ macro, a TCB is automatically generated, using the macro call parameters, and scheduled. The user also has the

capability of generating a TCB via the CTC$ macro and subsequently scheduling the TCB via the STC$ macro.

Scheduling Tasks

When scheduling tasks, the user must remember that tasks lack two properties that activities have. First, tasks must be resident in main memory before they can be scheduled; this is not the case with activities. Second, the user must not schedule a reusable or non-reusable task before the previous execution has terminated. Further, a non-reusable task cannot be rescheduled without being reloaded by rescheduling the non-reusable activity in which it resides. These rescheduling restrictions do not apply to activities because the Executive automatically handles activity loading and dispatching. The user must make sure that these task restrictions are considered when issuing STS$ and STC$ macros to schedule tasks.

NOTE: When a task is scheduled it goes on a priority level schedule queue from which it will later be taken when it is dispatched. Tasks are dispatched by the Executive on return from processing interrupts or when the previous task terminates or suspends. Normally, there is at least one interrupt (the clock) occurring frequently in the system so that the Executive will be dispatching at other times than when a task terminates or suspends. When a a task schedules another task on a higher level, the scheduled task will normally not have been dispatched when the scheduling task regains control, unless the return to the scheduling task was interrupted. If it is desired to dispatch a task scheduled on a higher level immediately, the scheduling macro (STS$ or STC$) should be followed by a suspend macro (SUS$).

Task Entry

When a given task's TCB is at the beginning of a priority level scheduing queue that is the highest priority level with any tasks scheduled, the task will be dispatched by the Executive. A dispatched task gains control at the task entry address. When control is transferred to the task, the X-register contains the address (TCB address) of the first word in the task's TCB. It is the user's responsibility to save this TCB address so that the TCB may be given as a parameter to a TMT$ or TMA$ macro when the task or activity is terminated. The pseudo registers (ZCR1, ZCR2, or ZCR3) may be used to store the TCB address if desired.

The TCB address is also needed to give the task access to the two parameters ("parameter 1" and "parameter 2") specified when the task was scheduled. The user gains access to these parameter words by having the address of the TCB in the X-register and doing a LDA ZTCBP1, 1 and a LDA ZTCBP2, 1. ZTCBP1 and ZTCBP2 are defined by a TCB$ macro to be the relative displacement within the TCB of where "parameter 1" and "parameter 2" are found.

It is the task to which the parameters are passed that determines the significance of the parameters just as it is the subroutine and not the subroutine caller that determines the significance of the argument list. If more than two parameters are needed to be passed to a task, it is suggested that the Free Memory Parameter Passing Technique described in Appendix A be used. This parameter passing technique must be used by any lead task of an activity that is scheduled with parameters by a FORTRAN program or by the Utilities.

The secondary task of a master activity needs the TCB address so that it may determine if the slave activity was successfully loaded into main memory. It is also needed to schedule the lead task of the slave activity. The secondary task gets the status by having the secondary TCB address in the X-register and doing a LDA ZTCBST,1 (see the SAC$ macro for the significance of the status word). The TCB address of the lead task is obtained by having the address of the secondary TCB in the X-register and then doing a LDA ZTCBPT,1. The lead task TCB address can then be used in a subsequent STC$ macro to schedule the lead task of the slave activity.

Suspending Tasks

There are two ways in which a task may be suspended; by an interrupt and by a voluntary suspension. A user task has no control over the former. Voluntary suspension involves executing a SUS$ macro. Although the Executive handles both suspensions in the same manner, there is a distinction.

When an interrupt suspends a task, the Executive has determined that there is something of higher priority to which the system must be allocated. During this involuntary suspension, the system is executing higher priority programs.

When a task voluntarily suspends, the priority level on which it is running is blocked and any other tasks currently scheduled on the level will not be dispatched. Normally, this does not cause a problem because there may be other tasks scheduled on other levels (higher or lower). They will be dispatched and the system will not be idle. But if a task is in a loop that keeps suspending, waiting for some external event to occur, it can be wasteful of system resources if there are no other tasks scheduled on other levels and there are tasks scheduled on the suspend level.

The SUS$ also provides the means by which a task can assure that a task it schedules on a higher priority level will be immediately dispatched. This is done by following the STS$ or STC$ macro by a SUS$ macro.

Terminating Tasks

Unlike activities, tasks can only terminate themselves. When a task terminates via a TMT$ or TMA$ macro, it allows the Executive to dispatch any other scheduled task.

Because a task does not regain control after it terminates, it must replace all allocated system resources that have not been passed to other tasks in the system. This means that the task must, before terminating, make sure all I/O initiated by the task is completed, release all reserved I/O devices, close all opened files and return all blocks obtained from free memory. One block of free memory allocated the task is the TCB generated when the task was scheduled. This block may be released by giving the TCB address as a parameter in the TMT$ or TMA$ macro.

Normally, a task terminates by issuing a TMT$ macro but if it is the last task of an activity, the task and activity are terminated by a TMA$ macro. When an activity terminates, it must also make sure that any clock task is disconnected as well as release all allocated system resources. Clock tasks must be disconnected because the task code will be resident only as long as the activity is.

SCHEDULE TASK MACRO (STS$)

The STS$ macro is used to schedule a task that is currently resident in main memory for execution.

STS$ Macro Action

STS$ creates a TCB for the task after obtaining a block from free memory. Then schedules the task by attachning the TCB to the end of the priority level schedule queue, and returns to the caller.

Macro Format

Location	Operation	Operand
[symbol]	STS$	Task entry address, [level], [acb address pointer], [parameter 1], [parameter 2], error return address

where:

symbol — Optional

Is the symbolic label of the macro instruction.

task entry address

Is the address of the entry point of the task.

level — Optional

Is the user priority level number on which the task is to be scheduled. If a level number lower than the lowest user priority level is specified, the lowest user priority level will be used to schedule the task. If this parameter is omitted, the task is scheduled on the default level specified in the activity control block of the activity in which the task resides.

acb address pointer — Optional

Is the address of a word containing the address of the activity control block associated with the activity in which the task resides and under whose control the task is to be executed. If this parameter is omitted, the task is scheduled as part of the current activity. If this parameter is the address of a word containing zero (i.e., acb address = 0), the task is scheduled as part of the current activity.

parameter 1 — Optional

Is a parameter word that can be used by the caller to pass data to the task. This parameter is in word ZTCBP1 of the TCB. If this parameter is omitted, a zero word is generated in the TCB for word ZTCBP1. If parameter 2 is not zero, parameter 1 must not be zero unless the free memory block parameter passing technique is being used.

parameter 2 — Optional

Is a parameter word that can be used by the caller to pass data to the task. This parameter is word ZTCBP2 of the TCB. If this parameter is omitted, a zero word is generated in the TCB for word ZTCBP2. If parameter 1 is zero, parameter 2 must be zero unless the free memory block parameter passing technique is being used.

error return address

Is the address to which control is returned if an error is found during the processing of the macro call.

Normal Return

Control is returned to the caller, at the instruction following the STS$ macro, after the task has been scheduled.

Error Return

Currently, the error return is not taken, but is present for future use.

Macro Action Details

A block is obtained from free memory and used to contain the TCB generated using the specified STS$ parameters. If the level parameter is omitted, the task is scheduled on the default level of the activity in which it resides by using the default level in generating the TCB. The default level is found in the ACB associated with the activity. If the ACB address pointer parameter is omitted, the address of the ACB of the current activity is used in generating the TCB.

After the TCB is generated, the task is scheduled by placing the TCB at the end of the priority level schedule queue. If the priority level in the TCB is lower than the lowest user priority level, the task is scheduled on the lowest user priority level. STS$ exits by returning control to the caller at the instruction following the STS$ macro. When the caller regains control, the scheduled task may not have been dispatched.

Out-Line Parameter List

Word No.	Operation	Operand
0	DAC	task entry address
1	DAC	[level]
2	DAC	[acb address pointer]
3	DAC	[parameter 1]
4	DAC	[parameter 2]
5	DAC	error return address

where:

Parameters in the out-line parameter list are the same as those described above for the in-line parameter list.

Examples:

A task, starting at ATASK and residing in the current activity, is to be scheduled on level 2. The task requires two parameters: the location T1, and the contents of T2.

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6	7 8 14	15 16 31	32
	STS$	ATASK,2,,T1,!T2,	TERR
	:		
	:		
*			
ATASK	---		START OF TASK
	:		
	:		
T1	BSZ	1	THE ADDRESS OF T1 IS A PARAMETER
*			
T2	BSZ	1	THE CONTENTS OF T2 IS A PARAMETER
*			
TERR	HLT		ERROR RETURN, CURRENTLY NOT USED
*			

If the first example were to be written, using an out-line parameter list, the following would be written:

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6	7 8 14	15 16 31	32
	LDX	PLIST	
	STS$	(X)	
	:		
	:		
PLIST	DAC	*+1	
	DAC	ATASK	
	DAC	2	
	DAC	0	
	DAC	T1	
T2	BSZ	1	
	DAC	TERR	
*			
T1	BSZ	1	
*			
TERR	HLT		ERROR RETURN, CURRENTLY NOT USED
*			
ATASK	---		START OF TASK
	:		

SUSPEND TASK MACRO (SUS$)

The SUS$ macro is used to suspend the task in which it appears and allow the possible dispatching of any other currently scheduled task.

SUS$ Macro Action

SUS$ saves all hardware (except A-register) and pseudo register, exits to the Executive which dispatches the highest priority scheduled task other than the task being suspended, and returns to the caller when the task is resumed.

Macro Format

Location	Operation	Operand
[symbol]	SUS$	

where:

symbol – Optional

Is the symbolic label of the macro instruction and may be omitted.

There are no parameters for this macro.

Normal Return

Control is returned to the caller, at the instruction following the SUS$ macro, when the task is resumed. The suspended task will be resumed with all hardware (except the A-register) and pseudo registers restored.

When control is returned, as is the case after any Executive macro call, the addressing mode is set to extended mode, interrupts are enabled, and the J-base is set to the activity J-base value. This setting of the return state may result in a state being changed from that of when the SUS$ macro was issued.

Error Return

There is no error return for this macro.

Macro Action Details

The state of the suspending task is saved in the priority level suspend area. All hardware (except A-register) and pseudo registers are saved there. Then a test is made to determine if there is currently any task scheduled at a priority level higher than the suspending level. If so, the highest priority task is dispatched. If there are no tasks scheduled at a level higher than the suspending level, the highest priority scheduled task under the suspended level is dispatched. This may result in tasks of lower priority than the suspended task being dispatched and terminated before the suspended task is resumed.

The suspended task regains control with all hardware (with the exception of the A-register) and pseudo registers restored. The A-register is destroyed by the LNK$ macro instruction via which the SUS$ macro passed; it cannot be restored. Control is returned with extended addressing set, interrupts enabled, and the J-base set to the J-base value of the activity in which this task resides. This may result in a mode change from that of when the task was suspended.

While a task is suspended no other scheduled task on the same level will be dispatched.

Out-Line Parameter List

There is no parameter list for this macro.

Examples

A task is to be suspended to allow any currently scheduled tasks on different levels to be dispatched.

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6 7	8 14 15	16 31	32
	SUS$		
	---		RESUME FROM SUSPENSION

TERMINATE TASK MACRO (TMT$)

The TMT$ macro is used to terminate the task in which it appears. If the task being terminated is the last task in an activity, the TMA$ macro instruction must be used to terminate the activity as well as the task.

TMT$ Macro Action

TMT$ returns the TCB, if present, terminates the current task, and exits to the Executive which dispatches the highest priority scheduled task.

Macro Format

Location	Operation	Operand
[symbol]	TMT$	[tcb address pointer]

where:

symbol — Optional

Is the symbolic label of the macro instruction.

tcb address pointer — Optional

Is the address of a word containing the address of the task control block associated with the terminating task. The TCB is returned to free memory. If this parameter is omitted, no block is returned to free memory.

Normal Return

TMT$ exits to the Executive and control does not return to the caller.

Error Return

There is no error return for this macro.

Macro Action Details

If the TCB address pointer parameter is present, and the TCB address is non-zero, the block is returned to free memory. If the TCB address pointer parameter is omitted or if the parameter is present but the TCB address is zero, no block is returned to free memory.

TMT$ exits by giving control to the executive which dispatches the highest priority scheduled task. No return is made to the caller. Before using the TMT$ macro to terminate a task, the user must:

1. Make sure all I/O initiated by the task is completed.
2. Release all reserved I/O devices.
3. Close all opened files.
4. Return all blocks obtained from free memory.

Out-Line Parameter List

Word No.	Operation	Operand
0	DAC	[tcb address pointer]

where:

Parameters in the out-line parameter list are the same as those described above for the in-line parameter list.

Examples:

A task is to be terminated. Its TCB has been returned to free memory prior to terminateion.

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6 7	8 14 15	16 31	32
	TMT$		

A task is to be terminated. Its TCB address has been stored in word TCBA, and this TCB is to be returned to free memory at termination time.

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6 7	8 14 15	16 31	32
	TMT$	TCBA	
	:		
TCBA	BSZ	1	STORE TCB ADDRESS

The above example could be written with a parameter list as follows:

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6	7 8 14	15 16 31	32
	LDX	PLIST	
	TMT$	(X)	
	:		
	:		
PLIST	DAC	*+1	
	DAC	TCBA	
*			
TCBA	BSZ	1	STORE TCB ADDRESS

CREATE A TASK CONTROL BLOCK MACRO (CTC$)

The CTC$ macro is used to create a TCB without scheduling the task. The created TCB can be scheduled later using the STC$ macro instruction.

CTC$ Macro Action

CTC$ creates a TCB for the task after obtaining a block from free memory. It then places the address of the created TCB in the X-register, and returns to the caller without scheduling the task.

Macro Format

Location	Operation	Operand
[symbol]	CTC$	task entry address, [level], [acb address pointer], [parameter 1], [parameter 2], error return address

where:

symbol — Optional

Is the symbolic label of the macro instruction.

task entry address

Is the address of the entry point of the task.

level — Optional

Is the user priority level number on which the task is to be scheduled later. If a level number lower than the lowest user priority level is specified, the lowest user priority level will be used when the TCB is scheduled. If this

parameter is omitted, the TCB is generated using the default level specified in the ACB of the activity in which this task resides.

acb address pointer — Optional

Is the address of a word containing the address of the activity control block associated with the activity in which this task resides and under whose control the task is to be executed. If this paameter is omitted, the TCB is generated with the task as part of the current activity. If this parameter is the address of a word containing zero (i.e., ACB address = 0), the TCB is generated with the task as part of the current activity.

parameter 1 — Optional

Is a parameter word that can be used by the caller to pass data to the task. This parameter is in word ZTCBP1 of the TCB. If this parameter is omitted, a zero word is generated in the TCB for word ZTCBP1. If parameter 2 is non-zero, parameter 1 must not be zero unless the free memory block parameter passing technique is being used.

parameter 2 — Optional

Is a parameter word that can be used by the caller to pass data to the task. This parameter is word ZTCBP2 of the TCB. If this parameter is omitted, a zero word is generated in the TCB for word ZTCBP2. If parameter 1 is zero, parameter 2 must be zero unless the free memory block parameter passing technique is being used.

error return address

Is the address to which control is returned if an error is found during the processing of the macro call.

Normal Return

Control is returned to the caller, at the instruction following the CTC$ macro, after the TCB has been created. On return, the address of the created TCB is in the X-register.

Error Return

Currently, the error return is not taken, but is present for future use.

Macro Action Details

A block is obtained from free memory and is used to contain the TCB generated using the specified CTC$ parameters. If the level parameter is omitted, the task is scheduled on the default level of the activity in which it resides by using the default level in generating the TCB. The default level is found in the ACB associated with the activity. If the level is specified as lower than the lowest user priority level, the lowest user priority level will be used when the TCB is scheduled. If the ACB address pointer parameter is omitted, the address of the ACB of the current activity is used in generating the TCB.

CTC$ exits by returning control to the caller at the instruction following the CTC$ macro with the address of the created TCB in the X-register.

Out-Line Parameter List

Word No.	Operation	Operand
0	DAC	Task entry address
1	DAC	[level]
2	DAC	[acb address pointer]
3	DAC	[parameter 1]
4	DAC	[parameter 2]
5	DAC	error return address

where:

Parameters in the out-line parameter list are the same as those described above for the in-line parameter list.

Examples:

A TCB for a task, ATASK, is to be created. This task resides in the current activity and it is to be scheduled later on level 2, using the STC$ macro. The two parameters required by the task are to be filled in prior to its scheduling. The address of the TCB is stored in word T1.

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6 7	8 14 15	16 31	32
	CTC$	ATASK,2,,,0,0,TERR	
	STX	T1	SAVE TCB ADDRESS
	:		
	:		
	LDX	T1	
	STA	ZTCBP1,1	STORE PARAMETER 1 IN TCB
	LDA		(SECOND PARAMETER)
	STA	ZTCBP2,1	STORE PARAMETER 2 IN TCB
	:		
	:		
*			
ATASK	---		START OF TASK
	:		
	:		
T1	BSZ	1	STORE TCB ADDRESS
*			
TERR	HLT		ERROR RETURN, CURRENTLY NOT USED
*			

The above example could be written with an out-line parameter list in the following manner:

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6	7 8 14	15 16 31	32
	LDX	PLIST	
	CTC$	(X)	
	:		
	:		
	LDX	T1	
	STA	ZTCBP1,1	
	LDA		(SECOND PARAMETER)
	STA	ZTCBP2,1	
	:		
PLIST	DAC	*+1	
	DAC	ATASK	
	DAC	2	
	DAC	0	
	DAC	0	
	DAC	0	
	DAC	TERR	
*			
ATASK	---		START OF TASK
	:		
	:		
T1	BSZ	1	STORE TCB ADDRESS
*			
TERR	HLT		ERROR RETURN, CURRENTLY NOT USED
*			

SCHEDULE TASK CONTROL BLOCK MACRO (STC$)

The STC$ macro is used to schedule a task that is currently resident in main memory by using a previously defined TCB. The TCB may have been created by a previous CTC$ macro instruction.

STC$ Macro Action

STC$ schedules the task by attaching the TCB to the end of the priority level schedule queue and returns to the STC$ caller.

Macro Format

Location	Operation	Operand
[symbol]	STC$	TCB address pointer, error return address

where:

symbol — Optional

Is the symbolic label of the macro instruction.

tcb address pointer

Is the address of a word containing the address of a previously generated task control block associated with the task to be scheduled.

error return address

Is the address to which control is returned if an error is found during the processing of the STC$ macro call.

Normal Return

After the task has been scheduled, control is returned to the caller, at the instruction following the STC$ macro.

Error Return

Currently, the error return is not taken, but is present for future use.

Macro Action Details

The task is scheduled by placing the TCB at the end of the priority level schedule queue. If the priority level in the TCB is lower than the lowest user priority level, the task is scheduled on the lowest user priority level. STC$ exits by returning control to the caller at the instruction following the STC$ macro. When the caller regains control, the scheduled task may not have been dispatched.

Out-Line Parameter List

Word No.	Operation	Operand
0	DAC	tcb address pointer
1	DAC	error return address

where:

Parameters in the out-line parameter list are the same as those described above for the in-line parameter list.

Examples:

A task is to be scheduled. Its TCB has been created previously by the CTC$ macro. The address of the TCB has been stored in TCBA.

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6	7 8 14	15 16 31	32
	STC$	TCBA,TERR	
	:		
	:		
TCBA	BSZ	1	ADDRESS OF TCB STORED
*			
TERR	HLT		ERROR RETURN, CURRENTLY NOT USED
*			

The above example, using an out-line parameter list, could be written as:

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6	7 8 14	15 16 31	32
	LDX	PLIST	
	STC$	(X)	
	:		
	:		
PLIST	DAC	*+1	
	DAC	TCBA	
	DAC	TERR	

*			
TCBA	BSZ	1	ADDRESS OF TCB STORED
*			
TERR	HLT		ERROR RETURN, CURRENTLY NOT USED
*			

CLOCK MACROS

The five clock macros are used to schedule activities and tasks by means of the clock, to stop such scheduling and to specify or request the date and time. The macros are: Connect Clock Activity (CCA$), Disconnect Clock Acitivity (DCA$), Connect Clock Task (CCL$), Disconnect Clock Task (DCL$), and Get Date-Time (GDT$).

An activity which contains clock tasks must remain resident in main memory as long as those tasks are connected to the clock. Care should be taken to insure that such an activity does not monopolize an activity area to the detriment of the execution of other activities which use the same activity area.

CONNECT CLOCK ACTIVITY (CCA$)

A request for the execution of an activity on a timed basis (either cyclic or non-cyclic) is made via the CCA$ macro.

CCA$ Macro Action

CCA$ connects a special system clock task to the timer specified in the macro. A SAC$ macro parameter list is passed as a parameter to this special task.

CCA$ then returns control to the caller via the normal return. When the time expires, the activity is scheduled.

Macro Format

Location	Operation	Operand
[symbol]	CCA$	activity name address, [level], [no. of time units], [time unit type], [connect type], [parameter 1], [parameter 2], error return address

where:

symbol — Optional

Is the symbolic label of the macro instruction.

activity name address

Is the address of the name of the first word of the six-character name of the activity. The name is left-justified, right-filled with spaces.

level — Optional

Is the user priority level number on which the task is to be scheduled. If a level number lower than the lowest user priority level is specified, the lowest user priority level will be used to schedule the task.

If this parameter is omitted, the task is scheduled on the default level specified for the activity.

no. of time units — Optional

Is an integer expressing the number of time units from now until the activity is to be scheduled.

If omitted, zero is used.

The range of this parameter depends on the time unit type parameter.

time unit type — Optional

Is an integer specifying the units of time the no. of time units parameter is in.

0 = Absolute time of day in minutes from midnight.

1 = millisecond timer. The range of no. of time units is 0 through 1023.

2 = half-second timer. The range of no. of time units is 0 through 4095.

4 = second timer. The range of no. of time units is 0 through 4095.

8 = minute timer. The range of no. of time units is 0 through 4095.

If omitted, zero (absolute time) is used.

connect type — Optional

Is an integer specifying whether the activity is to be scheduled on a cyclic or non-cyclic basis.

0 = cyclic.

1 = non-cyclic

If omitted, zero (cyclic) is used.

parameter 1 — Optional

Is a parameter word that can be used by the caller to pass data to the lead task. This parameter is in word ZTCBP1 of the lead task TCB. If this parameter is omitted, a zero word is generated in the TCB for word ZTCBP1. If parameter 2 is non-zero, parameter 1 must not be zero unless the free memory block parameter passing technique is being used.

parameter 2 — Optional

Is a parameter word that can be used by the caller to pass data to the lead task. This parameter is in word ZTCBP2 of the lead task TCB. If this parameter is omitted, a zero word is generated in the TCB for word ZTCBP2. If parameter 1 is zero, parameter 2 must be zero unless the free memory block parameter passing technique is being used.

error return adddress

Is the address to which control is returned if an error is found during the processing of the macro call.

Normal Return

Upon normal return, the special system clock task has been connected to the proper timer. The parameter to this special clock task is the SAC$ parameter list needed to schedule the activity. Also upon normal return, the X-register contains the address of the TCB used to connect the special system task to the clock. This address must be saved if a subsequent DCA$ (Disconnect Clock Activity) macro is to be issued.

Error Return

Control is returned to the error return address specified in the CCA$ macro with the error code in the A-register when the following error is detected:

A-register Contents (Decimal)	Error Condition
1	Absolute timer not configured.

Because the activity is scheduled at a later time (when the special system task is dispatched), errors discovered by the Schedule Activity action are reported to the operator.

Operator Message			Error Condition
SE = 100116	000036	actnam	Activity not on disk.
SE = 100122	000036	actnam	Disk error.
SE = 100126	000036	actnam	No activity area for the activity, or activity is too large to fit into its area.
SE = 100030	000000	actnam	Activity area will be overrun.
SE = 100031	000000	actnam	Disk error while loading the activity.

where actnam is the name of the activity.

Macro Action Details

A clock TCB is generated for the special system clock task. A SAC$ parameter list is generated. The address of this list is placed on the clock TCB to be passed as a parameter to the special system clock task.

After the special system clock task is connected to the proper timer, return is made to the normal return.

If the absolute timer has been specified, and it is not configured, return is to the error return.

At the time specified, the special system clock task will be dispatched. This task issues a Schedule Activity request, using the SAC$ parameter list passed to it as a parameter. At this time, the regular SAC$ action (see SAC$ description) proceeds.

If the activity is to be cyclic, the special system task remains connected to the clock, so that the activity will be rescheduled when the time interval has elapsed once more. This will continue until the clock activity is disconnected via the DCA$ (Disconnect Clock Activity) macro. If the activity is to be non-cyclic, the special system, clock task is removed from the clock.

Out-Line Parameter List

Word No.	Operation	Operand
0	DAC	activity name address
1	DAC	[level]
2	DAC	[no. of time units]
3	DAC	[time unit type]
4	DAC	[connect type]
5	DAC	[parameter 1]
6	DAC	[parameter 2]
7	DAC	error return address

where:

Parameters in the out-line parameter list are the same as those described above for the in-line parameter list.

NOTE: The special system clock task is scheduled on the same level as that specified for the level parameter, unless this is omitted or zero. In this case, the special system clock task is scheduled on the highest user level.

Examples:

The example below illustrates the use of CCA$ to connect an activity to the clock for non-cyclic execution. The activity named ACTNAM, is to be connected to the clock for scheduling four hours (240 minutes) from now. The level at which the activity is scheduled is the default level. One parameter, the address of a 32-word block, is passed to the activity. The location of this block has been stored in location T1, prior to calling the CCA$ macro.

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6 7	8 14 15	16 31	32
	STA	T1	
	:		
	CCA$	ANAME,,240,8,1,!T1,,ERR	
	:		
	:		
ANAME	BCI	3,ACTNAM	ACTIVITY NAME
*			
T1	BSZ	1	CONTAINS ADDRESS OF BLOCK
*			
ERR	---		ERROR RETURN
	NOTE:	!T1 IN THE SIXTH PARAMETER POSITION INDICATES THAT	
		THE CONTENTS OF T1 CONTAINS THE DEFINED PARAMETER	

In the example below an out-line parameter list is used for the same conditions as the example above.

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6 7	8 14 15	16 31	32
	LDX	PLIST	
	CCA$	(X)	
	:		
	:		
PLIST	DAC	*+1	
	DAC	ANAME	
	DAC	0	
	DAC	240	240 MINUTES
	DAC	8	MINUTES
	DAC	1	NON-CYCLIC

```
        BSZ     1               CONTAINS ADDRESS OF BLOCK
        DAC     0
        DAC     ERR
*
ANAME   BCI     3,ACTNAM
*
ERR     ---                     ERROR RETURN
```

DISCONNECT CLOCK ACTIVITY (DCA$)

The Disconnect Clock Activity (DCA$) macro is used to disconnect a clock activity from the time queue on which it resides.

DCA$ Macro Action

DCA$ removes the clock TCB from the timer queue on which it resides, thus disconnecting the activity from the clock. It then returns all associated blocks to free memory and returns control to the caller.

Macro Format

Location	Operation	Operand
[symbol]	DCA$	tcb address pointer, error return address

where:

symbol — Optional

Is the symbolic label of the macro instruction.

tcb address pointer

Is the address of the word in which the address of the clock TCB has been stored. This TCB address has been returned to the caller in the X-register by a CCA$ macro.

error return address

Is the address to which control is returned if an error is found during the processing of the macro call.

Normal Return

Normally the clock activity has been disconnected from the clock.

Error Return

Control is returned to the error return address specified in the DCA$ macro with the error code in the A-register when the following error is detected:

A-register Contents (Decimal)	Error Condition
0	The specified clock TCB was not found in the timer queue.

Macro Action Details

The special system TCB is removed from the time queue on which it resides. If, in searching the timer queue, the given TCB cannot be found on the queue, the error return is taken. After the TCB is removed, all associated blocks are returned to free memory.

Out-Line Parameter List

Word No.	Operation	Operand
0	DAC	tcb store address
1	DAC	error return address

where:

Parameters in the out-line parameter list are the same as those described above for the in-line parameter list.

Examples:

In the example below an activity is to be disconnected from the clock. At some prior time, the activity has been connected to the clock and the address of the special clock task TCB has been stored in T1 (see CCA$). If this TCB cannot be found in the clock queue, then return is to be to NOTCB.

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6 7	8 14 15	16 31	32
	DCA$	T1,NOTCB	
	:		
	:		
T1	BSZ	1	CONTAINS TCB ADDRESS
*			
NOTCB	---		RETURN IF TCB NOT FOUND IN QUEUE

If an out-line parameter list were used with the first example the following would be written:

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6 7	8 14 15	16 31	32
	LDX	PLIST	
	DCA$	(X)	
	:		
	:		
PLIST	DAC	*+1	
	DAC	T1	
	DAC	NOTCB	
T1	BSZ	1	CONTAINS TCB ADDRESS
NOTCB	---		RETURN IF TCB NOT FOUND IN QUEUE

CONNECT CLOCK TASK (CCL$)

The Connect Clock Task (CCL$) macro is used to connect a clock task to a system timer.

CCL$ Macro Action

CCL$ generates a clock TCB and places it in the proper timer queue. Then, it returns control to the caller via the normal return. If the absolute timer has been specified, and it is not configured, return is to the error return.

Macro Format

Location	Operation	Operand
[symbol]	CCL$	task entry address [level], [no. of time units], [time unit type], [connect type], [parameter 1], [parameter 2], error return address

where:

symbol — Optional

Is the symbolic label of the macro instruction.

task entry address

Is the address of the entry point of the task.

level — Optional

Is the user priority level number on which the task is to be scheduled. If a level number lower than the lowest user priority level is specified, the lowest user priority level will be used to schedule the task. If this parameter is omitted, the task is scheduled on the default level specified in the ACB of the activity in which the task resides.

no. of time units — Optional

Is an integer expressing the number of time units from now until the task is to be scheduled.

If omitted, zero is used.

The range of this parameter depends on the time unit type parameter.

time unit type — Optional

Is an integer specifying the time unit in which the no. of time units parameter is:

0 = Absolute time of day in minutes from midnight.

1 = millisecond timer. The range of no. of time units is 0 through 1023.

2 = half-second timer. The range of no. of time units is 0 thorugh 4095.

4 = second timer. The range of no. of time units is 0 through 4095.

8 = minute timer. The range of no. of time units is 0 through 4095.

If omitted, zero (absolute time) is used.

connect type — Optional

Is an integer specifying whether the task is to be scheduled on a cyclic or non-cyclic basis.

0 = cyclic

1 = non-cyclic

If omitted, zero (cyclic) is used.

parameter 1 — Optional

Is a parameter word that can be used by the caller to pass data to the task. This parameter is in word ZTCBP1 of the TCB. If this parameter is omitted, a zero word is generated in the TCB for word ZTCBP1. If parameter 2 in non zero, parameter 1 must not be zero unless the free memory block parameter passing technique is being used.

parameter 2 — Optional

Is a parameter word that can be used by the caller to pass data to the task. This parameter is word ZTCBP2 of the TCB. If this parameter is omitted, a zero word is generated in the TCB for word ZTCBP2. If parameter 1 is zero parameter 2 must be zero unless the free memory block parameter passing technique is being used.

error return address

Is the address to which control is returned if an error is found during the processing of the macro call.

Normal Return

Upon normal return, the clock task has been connected to the proper clock timer. Also upon normal return, the X-register contains the address of the clock TCB used to connect

the task to the clock. This address must be saved if a subsequent DCL$ (Disconnect Clock Task) macro is to be issued.

Error Return

Control is returned to the error return address specified in the CCL$ macro with the error code in the A-register when the following error is detected:

A-register Contents (Decimal)	Error Condition
1	Absolute timer not configured.

Macro Action Details

A clock TCB, which defines the clock task, is generated. This TCB is placed in the proper timer queue. Return is then made to the normal return unless the absolute timer is specified and not configured. In this case, return is to the error return.

When the specified time interval has elapsed, the clock task is scheduled and dispatched. If the task is to be cyclic, its TCB remains on the timer queue until removed by a DCL$ (Disconnect Clock Task) macro. If the task is not cyclic, its TCB is removed from the timer queue when the task is scheduled.

Out-line Parameter List:

Word No.	Operation	Operand
0	DAC	task entry address
1	DAC	[level]
2	DAC	[no. of time units]
3	DAC	[time unit type]
4	DAC	[connect type]
5	DAC	[parameter 1]
6	DAC	[parameter 2]
7	DAC	error return address

where:

Parameters in the out-line parameter list are the same as those described above for the in-line parameter list.

Examples

The example below illustrates the use of CCL$ to connect a task to the clock for cyclic execution. The task CGOLT is to be connected to the clock for scheduling at 20-second intervals. The task will be scheduled on user priority level 4.

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6 7	8 14 15	16 31	32
*			CONNECT TASK "CGOLT" TO THE CLOCK FOR
*			CYCLIC EXECUTIONS
	CCL$	CGOLT,4,20,4,0,,	,ERRT
*			
*			NO STORAGE FOR A TCB
*			
ERRT	---		

The following example is the same as above, except that the in-line parameter list is not used.

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6 7	8 14 15	16 31	32
*			CONNECT TASK "CGOLT" TO THE CLOCK FOR
*			CYCLIC EXECUTION
	LDX	CLST	GET LIST ADDRESS
	CCL$	(X)	
*			
*			PARAMETER LIST FOR CCL$
*			
CLST	DAC	*+1	ADDRESS OF TASK
	DAC	CGOLT	ADDRESS OF TASK
	DEC	4	PRIORITY LEVEL = 4
	DEC	20	20 UNITS OF TIME
	DEC	4	UNITS = SECONDS
	OCT	0	CYCLIC CLOCK TASK
	OCT	0	USER PARAMETER 1
	OCT	0	USER PARAMETER 2
	DAC	ERRT	ERROR CODE ADDRESS

*			
*			ABSOLUTE TIMER NOT CONFIGURED
*			
ERRT	---		

DISCONNECT CLOCK TASK (DCL$)

The Disconnect Clock Task (DCL$) macro is used to disconnect a specified clock task from a system timer.

DCL$ Macro Action

DCL$ removes the clock task TCB from the timer queue on which it resides, thus disconnecting the task. It then returns all associated blocks to free memory and control to the caller.

Macro Format

Location	Operation	Operand
[symbol]	DCL$	tcb address pointer, error return address

where:

symbol — Optional

Is the symbolic label of the macro instruction.

tcb address pointer

Is the address of a word in which the address of the clock TCB has been stored. This TCB address has been returned to the caller in the X-register by the CCL$ macro.

error return address

Is the address to which control is transferred if an error is discovered during the processing of the macro.

Normal Return

The clock task has been disconnected from the clock.

Error Return

Control is returned to the error return address specified in the DCL$ macro with the error code in the A-register when the following error is detected:

A-register Contents (Decimal)	Error Condition
0	The specified clock TCB was not found in the timer queue.

Macro Action Details

The clock task TCB is removed from the timer queue. If, in searching the timer queue, the given TCB cannot be found on the queue, the error return is taken.

After the TCB is removed, all associated blocks are returned to free memory.

Return is to the normal return.

Out-Line Parameter List

Word No.	Operation	Operand
0	DAC	tcb address pointer
1	DAC	error return address

where:

Parameters in the out-line parameter list are the same as those described above for the in-line parameter list.

Example:

The example below illustrates the use of DCL$ to dissonnect a clock TCB from the clock. The address of the TCB to be disconnected will have been stored in word CTCB after the clock task was originally connected by CCL$.

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6 7	8 14 15	16 31	32
*			DISCONNECT A CLOCK TCB FROM CLOCK
*			
	DCL$	CTCB,ERRC	
*			
*			CLOCK TCB ADDRESS
*			
CTCB	BSZ	1	TO HOLD TCB ADDRESS
*			

*			TCB NOT FOUND IN THE CLOCK QUEUE
*			
ERRC	---		

GET DATE-TIME (GDT$)

The Get Date-Time (GDT$) macro allows the transfer of the ASCII date and time to a user-specified buffer.

GDT$ Macro Action

GDT$ rearranges the ASCII date-time into user format and stores it in the user's buffer. It then returns control to the caller via the normal return.

Macro Format

Location	Operation	Operand
symbol	GDT$	date-time block address, error return address

where:

symbol — Optional

Is the symbolic label of the macro instruction.

date-time block address

Is the address of the first word of a six-word block into which the macro action will place the date-time.

error return address

Is the address to which control is returned if an error is found during the processing of the macro call.

Normal Return

The ASCII date-time has been placed into the user-supplied buffer in the following format (two characters per word):

Word 0 — Hours (00-23)
Word 1 — Minutes (00-59)
Word 2 — Two space characters
Word 3 — Day (01-31)
Word 4 — Month (01-12)
Word 5 — Year (00-99)

Error Return

Currently, the error return is not taken, but is present for future use.

Macro Action Details

The ASCII date-time is obtained from the system, the data rearranged into the user format and stored in the user buffer.

Return is made to the normal return.

Out-Line Parameter List

Word No.	Operation	Operand
0	DAC	date-time block address
1	DAC	error return address

where:

Parameters in the out-line parameter list are the same as those described above for the in-line parameter list.

Example:

In the example below the buffer TBUF will be used to receive the date-time characters upon execution of GDT$.

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6 7	8 14 15	16 31	32
*			GET THE DATE AND TIME
*			
	GDT$	TBUF,ERR	
*			
*			THIS BUFFER WILL RECEIVE THE DATE-TIME CHARACTERS
*			
TBUF	BSZ	6	6 WORDS FOR DATE-TIME
ERR	---		ERROR RETURN

SECTION IV
MANIPULATING DATA

This chapter describes the system macros used to make available the records on an existing file, create and use a queue and obtain and release blocks of memory during the execution of a task. These macros are described as file and record macros, queue macros, and block macros.

FILE AND RECORD MACROS

The file and record macros are used to: open an existing file stored on the disk, read a record from that file or put a record on that file, and close the file when all record processing is complete.

They include:

1. Assign File Control Block (FCB$)
2. Open File (OPN$)
3. Close File (CLS$)
4. Get a Record (GET$)
5. Put a Record (PUT$)

Logical I/O may be used to create files, to retrieve data from files that have been created, and to delete files. These files exist on the system disk. Before any Logical I/O requests are made, the user must first create a File Control Block. Each user must create a File Control Block for each file referenced.

The File Control Block contains a pointer to the file name. The file name must be unique if the file is being created. The file name must be the name of a file that already exists on the system disk if the data from the file is to be retrieved. Every file in the system that is created must have a unique name. However, file names may be the same as activity names as long as this will not confuse the user as far as his processing is concerned.

Before creating a file, retrieving data from a file, or deleting a file, the file must be opened by issuing the OPN$ macro. If a file is opened with input mode specified, the file must

already exist on the system disk. If a file is opened with output mode specified, the file is to be created, and, therefore, must not already exist on the system disk.

Data is stored in a file that is being created by issuing a PUT$ macro. For each put request that is processed, one logical record, which is specified in the users record buffer, is added to the end of the file. Each record in a particular file is the same size. The size of the record is specified when the file is opened. There is no restriction on the size of the records in a file or the number of records in a file except the memory space available and the available space on the disk. While a file is being created, no user may retrieve data from this file. After all the data has been stored in a file, the file must be closed.

After a file has been opened with input mode specified, the data in the file may be retrieved by issuing the GET$ macro. The get request will cause the next sequential logical record of the file to be passed to the user. When there are no more records in the file to pass to the user, the end of file return is taken to the user. There may be any number of users retrieving data at the same time from a file that has been created. Each user who is accessing the file data must have created a File Control Block and each must have opened the file. When all of the desired data has been retrieved from a file, the file must be closed.

Every file that is opened successfully must be closed by issuing the CLS$ macro. This means that even if a disk error occurred when a get or put request was being processed and the user cannot continue, the file must be closed.

When issuing a close request, the file may be saved or deleted. If the file is closed with the close mode parameter specifying that the file is to be saved, the file information is saved in the disk directory for the file that is being created, or if the file was already created, any clean-up work that is required will be done at this time. If a file that is being created is saved, and a disk error occurs in the process of updating the disk directory, the file will be deleted. The file is deleted because there would be no way for the user to ever access this file nor would he be able to delete it at some later time.

If a file is closed with the close mode parameter specifying that the file is to be deleted, the specified file is deleted from the system disk. Since there are no checks for other users of a file that is being deleted, the user must make sure no other user is accessing a file that is being deleted.

Since there is no macro that allows the updating of a file that has been created, the user has to do the update himself. This is accomplished by first opening the file to be updated with input mode specified, and opening another file with output mode specified. This file will contain the updated file data. Each record from the old file is input by issuing a get request. As

each record is input and the desired changes made, the record is then added to the new file by issuing a put request. This procedure of getting and putting a record continues until the end of the file is reached at which time the files are closed. The old file may be closed with the close mode parameter indicating that the file is to be deleted, and the new file is closed with the close mode parameter indicating that the file is to be saved.

ASSIGN FILE CONTROL BLOCK MACRO (FCB$)

The FCB$ macro generates a list of parameters required for logical I/O requests which include Open A File (OPN$), Close A File (CLS$), Get A Record (GET$), and Put A Record (PUT$).

FCB$ Macro Action

FCB$ generates a list of parameters which contains no executable instructions.

Macro Format

Location	Operation	Operand
fcb name	FCB$	file name address, [record buffer address]

where:

fcb name

Is the symbolic label of the file control block created by FCB$ macro instruction.

file name address

Is the address of the first word of a three word block which contains the name of the file.

record buffer address — Optional

Is the address of the record buffer from which data is transferred for Put requests and into which data is stored for Get requests. The record buffer address must always be present in the FCB for Put requests. The record buffer address must also be present in the FCB for Get requests if the logical record length is greater than or equal to the length of a physical disk record. The only instance when the record buffer address may be omitted (equal to 0) is for a Get request when the logical record length is less than the length of a physical disk record. In this case, the address of the record is passed to the user in the A-register.

Normal Return

The FCB$ macro has no normal return since the FCB$ macro performs no action.

Error Return

There is no error return for this macro.

Macro Action Details

The FCB$ macro has no action since it is a list of parameters. Therefore, the FCB$ must be coded in a program as a data block or a buffer.

Out-Line Parameter List

Word No.	Operation	Operand
0	BSZ	1 (reserved for system use)
1	BSZ	1 (reserved for system use)
2	DAC	file name address
3	DAC	[record buffer address]

where:

the parameters in the out-line parameter list are the same as those described above for the in-line parameter list. However, note that words 0 and 1 must contain a BSZ 1. The user must also note that once a file is opened successfully, words 0, 1, and 2, are not to be altered until after the file is closed.

Examples:

The examples below illustrate the generation of File Control Blocks by using the FCB$ macro.

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6	7 8 14	16 31	32
*			
*			GENERATE AN FCB TO BE REFERENCED WHEN
*			MAKING LOGICAL I/O REQUESTS FOR
*			FILE ABCDEF
*			
*			
FCNM1	FCB$	FILE1,BUF1	
FILE1	BCI	3,ABCDEF	FILE NAME
BUF1	BSZ	60	RECORD BUFFER

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6	7 8 14	15 16 31	32
*			
*			GENERATE AN FCB TO BE REFERENCED WHEN
*			MAKING LOGICAL I/O REQUESTS FOR
*			FILE UVWXYZ
*			
*			
FCNM2	FCB$	FILE2,BUF2	
FILE2	BCI	3,UVWXYZ	FILE NAME
BUF2	BSZ	60	RECORD BUFFER
	LNK$		LINK TO SYSTEM

The examples below show how the File Control Blocks would be written without the use of the FCB$ macro.

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6	7 8 14	15 16 31	32
*			
*			GENERATE AN FCB TO BE REFERENCED WHEN
*			MAKING LOGICAL I/O REQUESTS FOR
*			FILE ABCDEF
*			
*			
FCNM1	BSZ	2	RESERVED FOR SYSTEM USE
	DAC	FILE1	FILE NAME ADDRESS
	DAC	BUF1	RECORD BUFFER ADDRESS
FILE1	BCI	3,ABCDEF	FILE NAME
BUF1	BSZ	60	RECORD BUFFER

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6	7 8 14	15 16 31	32
*			
*			GENERATE AN FCB TO BE REFERENCED WHEN
*			MAKING LOGICAL I/O REQUESTS FOR
*			FILE UVWXYZ
*			
*			
FCNM2	BSZ	2	RESERVED FOR SYSTEM USE
	DAC	FILE2	FILE NAME ADDRESS
	DAC	BUF2	RECORD BUFFER ADDRESS
FILE2	BCI	3,UVWXYZ	FILE NAME
BUF2	BSZ	60	RECORD BUFFER
	LNK$		LINK TO SYSTEM

OPEN FILE REQUEST MACRO (OPN$)

The OPN$ macro initializes parameters for processing other logical I/O functions which include Get A Record (GET$), Put A Record (PUT$), and Close A File (CLS$).

OPN$ Macro Action

OPN$ initializes parameters which will be used to contain the current state of the file, identifies the requested file, and transfers the record length to the user if input mode is specified, adds the specified file name to the disk directory, and retrieves the record length from the user if output mode is specified, and takes the normal return to the user.

Macro Format

Location	Operation	Operand
[symbol]	OPN$	fcb address, error return address, i/o mode, record length address.

where:

symbol — Optional

Is the symbolic label of the macro instruction.

fcb address

Is the address of the file control block which contains necessary data for the user to make logical I/O requests for a particular file.

error return address

Is the address to which control is returned if an error is found in the parameters specified in the file control block, in the parameters specified in the open request parameter list, or if an error occurs as a result of requesting I/O operations.

i/o Mode

Is a number which indicates either input or output. 0 indicates output and 1 indicates input. If output is specified, a file is to be created, and only put requests may be issued. If input is specified, a file is to be read, and only get requests may be issued.

record length address

Is the address of a word which contains the record length if a file is to be created or output mode is specified, or is the address of a word which contains 0 if a file is to be read or input mode is specified. Upon return from the open request, when input mode is specified, the record length address will contain the size of the logical records contained in the file.

Normal Return

Control is returned to the caller, at the instruction following the OPN$ macro, after the file has been opened successfully.

Error Return

Control is returned to the error return address specified in the OPN$ macro when any of the following errors are detected:

A-register Contents (Decimal)	Error Condition
14	The file name was not found in the disk directory. (Open in input mode)
15	The file name already exists in the disk directory. (Open in output mode)
16	The file name cannot be added to the disk directory because the disk directory is full. (Open in output mode)
17	The allocate request cannot be honored because all of the work areas on disk are being used. (Open in output mode)
18	A disk error occurred when trying to input or output the disk directory information. (Open in either input or output mode)
20	An open request was issued for a null file which is a file that is currently being created but has not been closed. (Open in input mode)

Also, the record length address was not specified in the parameter list of the open request.
(Open in either input or output mode)

Macro Action Details

A 64 word block in which parameters are initialized is fetched from free memory and the current state of the file is maintained for the user requesting the file.

If input mode is specified, the requested file is identified on the system disk, the size of the logical records in the file is passed to the user, and control is returned to the user at the normal return address.

If output mode is specified, the file name is added to the system disk directory, an area on the system disk is allocated where the initial records of the file to be created will be stored, the logical record length is retrieved from the user, and control is returned to the user at the normal return address.

Out-Line Parameter List

Word No.	Operation	Operand
0	DAC	fcb address
1	DAC	error return address
2	DEC	i/o mode
3	DAC	record length address

where:

the parameters in the out-line parameter list are the same as those described above for the in-line parameter list.

Examples:

The examples below illustrate the use of OPN$ to open files ABCDEF and UVWXYZ. The File Control Blocks, FCNM1 and FCNM2, which must be generated before the open requests are issued, are shown in the examples of Assign File Control Block.

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6 7 8	14 15 16	31 32	
*			
*			OPEN FILE ABCDEF
*			
	OPN$	FCNM1,ERT1,1,PRDR	OPEN FOR INPUT
	---		NORMAL RETURN,

```
*                                   FILE ABCDEF OPENED
*
*
*                                   ERROR RETURN - THE A-REGISTER CONTAINS
*                                   THE ERROR CODE
*
*
ERT1                                ERROR, ILLEGAL
*                                   PARAMETER IN THE FCB, FCNM1, OR IN THE
*                                   OPN$ MACRO LIST, OR THE FILE ABCDEF
*                                   DOES NOT EXIST.
*
*
*
PRDR    RSZ                         RECORD LENGTH RETURNED HERE
*                                   ON OPEN
*
```

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6 7	8 14 15	16 31	32
*			
*			OPEN FILE UVWXYZ
*			
	OPN$	FCNM2,ERT2,0,PRDW	OPEN FOR OUTPUT
	---		NORMAL RETURN,
*			FILE UVWXYZ OPENED
*			ERROR RETURN - THE A-REGISTER
*			CONTAINS THE ERROR CODE
*			
*			
*			
ERT2			ERROR, ILLEGAL
*			PARAMETER IN THE FCB, FCNM2, OR IN

*			THE OPN$ MACRO LIST, OR THE
*			FILE UVWXYZ ALREADY EXISTS.
*			
*			
PRDW	DEC	60	RECORD LENGTH

The following examples are the same as the examples above, except that out-line parameter lists are used.

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6	7 8 14	15 16 31	32
*			
*			OPEN FILE ABCDEF
*			
	LDX	PLST1	PARAMETER LIST
*			POINTER TO X
	OPN$	(X)	OPEN FILE FOR INPUT
	---		NORMAL RETURN,
*			FILE ABCDEF OPENED
*			
*			ERROR RETURN - THE A-REGISTER
*			CONTAINS THE ERROR CODE
*			
*			
ERT1	---		ERROR, ILLEGAL
*			PARAMETER IN THE FCB, FCNM1, OR IN
*			THE OPN$ MACRO LIST, OR THE FILE ABCDEF
*			DOES NOT EXIST.
*			
*			
*			

```
*                                   OPEN REQUEST
*                                   PARAMETER LIST FOR FILE ABCDEF
*
*
PLST1   DAC     *+1                 POINTER TO
*                                   PARAMETER LIST
        DAC     FCNM1               FCB ADDRESS
        DAC     ERT1                ERROR RETURN ADDRESS
        DEC     1                   INPUT MODE
        DAC     PRDR                RECORD LENGTH ADDRESS
*
PRDR    BSZ     1                   RECORD LENGTH RETURNED
*                                   HERE ON OPEN
*
```

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6	7 8 14	15 16 31	32
*			
*			OPEN FILE UVWXYZ
*			
	LDX	PLST2	PARAMETER LIST
*			POINTER TO X
	OPN$	(X)	OPEN FILE FOR OUTPUT
	---		NORMAL RETURN,
*			FILE UVWXYZ OPENED
*			
*			ERROR RETURN - THE A-REGISTER
*			CONTAINS THE ERROR CODE
*			
*			

```
ERT2    ---             ERROR, ILLEGAL
*                       PARAMETER IN THE FCB, FCNM2, OR IN
*                       THE OPN$ MACRO LIST, OR THE
*                       FILE UVWXYZ ALREADY EXISTS.
*
*
*
*
*                       OPEN REQUEST
*                       PARAMETER LIST FOR
*                       FILE UVWXYZ
*
PLST2   DAC     *+1     POINTER TO
*                       PARAMETER LIST
        DAC     FCNM2   FCB ADDRESS
        DAC     ERT2    ERROR RETURN ADDRESS
        DEC     0       OUTPUT MODE
        DAC     PRDW    RECORD LENGTH ADDRESS
PRDW    DEC     60      RECORD LENGTH
```

CLOSE FILE REQUEST MACRO (CLS$)

The CLS$ macro provides the capability of either saving a file that is being created or deleteing a file that already exists.

CLS$ Macro Action

CLS$ deletes a file if the close mode parameter indicates that the file is to be deleted, or saves a file that has been created if the close mode parameter indicates that the file is to be saved. It then returns blocks that were used to process requests for the specified file to free memory. Note that every file that was opened successfully must be closed.

Macro Format

Location	Operation	Operand
[symbol]	CLS$	fcb address, error return address, close mode

where:

symbol — Optional

is the symbolic label of the macro instruction.

fcb address

Is the address of the file control block which contains necessary data for the user to make logical I/O requests for a particular file.

error return address

Is the address to which control is returned if an error is found in the parameters specified in the file control block, or if an error occurs as a result of requesting I/O operations.

close mode

Is a number which indicates either normal or delete mode. Zero indicates normal mode and 1 indicates delete mode.

If a file was opened in input mode, and the file was closed with normal mode specified, the action taken is to return the blocks where used to process the user's requests for the specified file to free memory.

If a file was opened in output mode, and the file was closed with normal mode specified, the specified file is updated in the system disk directory to indicate that the file is now a permanent file, and the blocks which were used to process the user's requests for the specified file are returned to free memory.

If the file was opened in either input or output mode, and the file is closed with delete more specified, the specified file is deleted from the system disk, and all blocks which were used to process the user's requests for the specified file are returned to free memory.

It should be noted that when a file is deleted, there are no checks for other users of the file that is being deleted.

Normal Return

Control is returned to the caller, at the instruction following the CLS$ macro, after the file has been closed successfully.

Error Return

Control is returned to the error return address specified in the CLS$ macro when any of the following errors are detected:

A-register Contents (Decimal)	Error Condition
18	A disk error occurred when trying to input or output the disk directory information (close in either input or output mode).
19	A close request was issued for a file that was not previously opened (close in either input or output mode).

NOTE: If an error occurs when closing a file in normal mode, the file will be deleted, and the error return will be taken to the user with the error code in the A-register.

Macro Action Details

If a file is to be deleted (close mode parameter equal to 1), the file name is removed from the system disk directory, the work areas on the system disk which were allocated for the file are deallocated, and control is returned to the user at the normal return address.

If a file is to be saved (close mode parameter equal to 0), the last physical disk record (which may contain one or more logical records or only a part of a logical record) is stored on the system disk, the system disk directory information is updated to indicate that the file is a permanent file (no longer a null file, a file that is being created but has not been closed and saved), and control is returned to the user at the normal return address.

Out-Line Parameter List

Word No.	Operation	Operand
0	DAC	fcb address
1	DAC	error return address
2	DEC	close mode

where:

the parameters in the out-line parameter list are the same as those described above for the in-line parameter list.

Examples:

The examples below illustrate the use of CLS$ to close the files ABCDEF and UVWXYZ. Example 1 under In-line Parameter List and example 1 under Out-line Parameter List show how to close and delete a file. Example 2 under In-line Parameter List and example 2 under Out-Line Parameter List show how to close and save a file. The File Control Blocks, FCNM1 and FCNM2, which must be generated before the files are closed, are illustrated in the examples under Assign File Control Block, and the open requests, which must be processed before the files are closed, are illustrated in the examples under Open File.

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6	7 8 14	15 16 31	32
*			
*			CLOSE FILE ABCDEF
*			
	CLS$	FCNM1,ERT5,1	CLOSE AND DELETE
*			FILE ABCDEF
	---		NORMAL RETURN,
*			FILE ABCDEF DELETED

```
                                        
*                                       
*                                       ERROR RETURN - THE A-REGISTER
*                                       CONTAINS THE ERROR CODE
*                                       
*                                       
ERT5    ---                             ERROR, ILLEGAL
*                                       PARAMETER IN THE FCB, FCNM1, OR
*                                       DISK ERROR ON INPUT OR OUTPUT
*                                       OF DISK DIRECTORY
*                                       
```

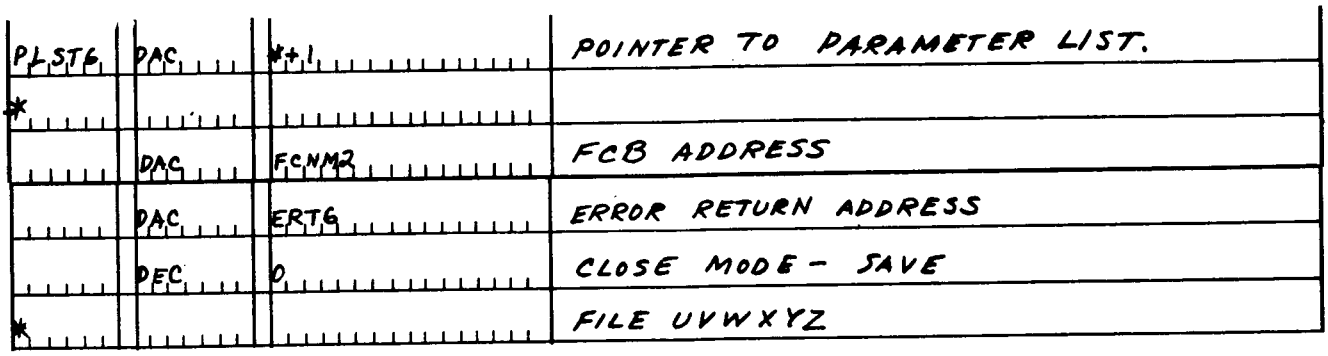
LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6	7 8 14	15 16 31	32
*			
*			CLOSE FILE UVWXYZ
*			
	CLS$	FCNM2,ERT6,0	CLOSE AND SAVE
*			FILE UVWXYZ.
	---		NORMAL RETURN,
*			FILE UVWXYZ SAVED.
*			
*			ERROR RETURN - THE A-REGISTER
*			CONTAINS THE ERROR CODE.
*			
*			
ERT6	---		ERROR, ILLEGAL
*			PARAMETER IN THE FCB, FCNM2, OR
*			DISK ERROR ON INPUT OR OUTPUT
*			OF DISK DIRECTORY.
*			

The following examples are the same as the examples above, except that out-line parameter lists are used.

```
LOCATION  OPERATION  OPERAND FIELD      COMMENTS
1       6 7 8    14 15 16         31 32
*
*                                     CLOSE FILE ABCDEF.
*
          LDX        PLST5            PARAMETER LIST
*                                     POINTER TO X.
          CLS$       (X)              CLOSE AND DELETE
*                                     FILE ABCDEF
          ---                         NORMAL RETURN,
*                                     FILE ABCDEF DELETED.
*
*
*                                     ERROR RETURN- THE A-REGISTER
*                                     CONTAINS THE ERROR CODE.
*
*
ERT5      ---                         ERROR, ILLEGAL
*                                     PARAMETER IN THE FCB, FCNM1, OR
*                                     DISK ERROR ON INPUT OR OUTPUT
*                                     OF DISK DIRECTORY.
*
*
*
*                                     CLOSE REQUEST
*                                     PARAMETER LIST FOR FILE ABCDEF.
*
*
PLST5     DAC        *+1              POINTER TO PARAMETER LIST
*
          DAC        FCNM1            FCB ADDRESS
```

	DAC	ERT5	ERROR RETURN ADDRESS
	DEC	1	CLOSE MODE - DELETE
*			FILE ABCDEF

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6	7 8 14	15 16 31	32
*			
*			CLOSE FILE UVWXYZ
*			
	LDX	PLST6	PARAMETER LIST
*			POINTER TO X.
	CLS$	(X)	CLOSE AND SAVE
*			FILE UVWXYZ.
	---		NORMAL RETURN,
*			FILE UVWXYZ SAVED.
*			
*			ERROR RETURN - THE A-REGISTER
*			CONTAINS THE ERROR CODE.
*			
*			
ERT6	---		ERROR, ILLEGAL
*			PARAMETER IN THE FCB, FCNM2, OR
*			DISK ERROR ON INPUT OR OUTPUT
*			OF DISK DIRECTORY.
*			
*			
*			CLOSE REQUEST
*			PARAMETER LIST FOR FILE UVWXYZ.
*			
*			



GET RECORD REQUEST MACRO (GET$)

The GET$ macro causes the next sequential logical record of the requested file to be passed to the user.

GET$ Macro Action

GET$ reads one or more physical disk records from the system disk, if required, passes the next sequential logical record of the file to the user, and takes the normal return to the user.

Macro Format

Location	Operation	Operand
[symbol]	GET$	fcb address, error return address, eof return address, [record buffer address]

where:

symbol — Optional

Is the symbolic label of the macro instruction.

fcb address

Is the address of the file control block which contains necessary data for the user to make logical I/O requests for a particular file.

error return address

Is the address where control is to be returned if an error is found in the parameters specified in the file control block, or if an error occurs as a result of requesting I/O operations.

eof return address

Is the address where control is transferred when there are no more records in the requested file to be passed to the user.

record buffer address — Optional

Is the address of the record buffer into which the data of a logical record is stored. If this parameter is omitted, the record buffer address specified in the file control block will be used. If this parameter is present, it will be used to replace the record buffer address in the file control block, and any subsequent requests using record buffer address. Get requests using the same file control block will use the new record buffer address.

NOTE: This updating of the record buffer address in the FCB is performed by the execution of the in-line macro expansion and not by the macro action routine.

Normal Return

Control is returned to the caller, at the instruction following the GET$ macro, after the Get request has been processed successfully.

Error Return

Control is returned to the error return address specified in the GET$ macro when any of the following errors are detected:

A-register Contents (Decimal)	Error Condition
18	A disk error occurred when trying to input a physical disk record from the system disk.
19	A Get request was issued for a file that was not previously opened, or a Get request was issued for a file that was opened in output mode.
21	The record buffer address is not specified in the file control block and the record length is greater than or equal to the physical disk record.

Macro Action Details

A test is made to determine if the end of the file has been reached. If the end of the file has been reached (which means there are no more logical records in the requested file to pass to the user) control is transferred to the user at the EOF return address. If the end of the requested file has not been reached, the next sequential logical record of the requested file is read from the system disk, if required, and the logical record is passed to the user. The logical record will be transferred to the buffer specified in the file control block by the record buffer address parameter if the record buffer address was specified (is not equal to 0). The pointer to the logical record will be passed to the user in the A-register if the record buffer address is not specified (is equal to 0) in the file control block, and the logical record length is less than the physical disk record length. Control is then returned to the user at the normal return address.

Out-Line Parameter List

Word No.	Operation	Operand
0	DAC	fcb address
1	DAC	error return address
2	DAC	eof return address

where:

the parameters in the out-line parameter list are the same as those described above for the in-line parameter list.

NOTE: If an out-line parameter list is used and it is desired to update the FCB record buffer address, the user must precede the GET$ macro call with instructions to store the new record buffer address in the forth word (word 3) of the FCB.

Examples:

The examples below illustrate the use of GET$ to get a record from the file ABCDEF. The File Control Block, FCNM1, which must be generated before the GET request is issued, is shown above in the examples under Assign File Control Block, and the open request, which must be processed before the GET request is issued, is shown above in the examples under Open File. In example 1 under In-line Parameter List and example 1 under Out-line Parameter List, the address of the Record Buffer, BUF1, which is in the File Control Block, FCNM1, is overwritten and becomes the address of the Record Buffer, BUF3. In example 2 under In-line Parameter List and example 2 under Out-line Parameter List, the address of the record buffer in the File Control Block, FCNM1, becomes 0. In example 1, the record is stored in BUF3 while in example 2, the pointer to the first word of the record is passed in the A-Register.

Example 1

```
LOCATION  OPERATION  OPERAND FIELD          COMMENTS
1      6  7 8    14  16              31  32
*
*                                         GET A RECORD FROM FILE ABCDEF
*
*
          GET$       FCNM1,ERT3,EOF1,BUF3
                     ---                  NORMAL RETURN, RECORD FETCHED
*                                         AND STORED IN BUF3
*
*
*                                         ERROR RETURN- THE A-REGISTER
*                                         CONTAINS THE ERROR CODE
*
*
ERT3                 ---                  ERROR, ILLEGAL
*                                         PARAMETER IN THE FCB, FCNM1, OR
*                                         DISK ERROR ON INPUT OF RECORD
*
```

*			
*			END OF FILE RETURN
*			
EOF1	---		EOF, ALL RECORDS FROM FILE ABCDEF
*			HAVE BEEN READ
*			
BUF3	BSZ	60	RECORD BUFFER

Example 2

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6	7 8 14	15 16 31	32
*			
*			GET A RECORD FROM FILE ABCDEF
*			
*			
	CRA		0 TO RECORD BUFFER
	STA	FCNM1+3	ADDRESS IN THE FCB
	GET$	FCNM,ERT3,EOF1	
	---		NORMAL RETURN, RECORD FETCHED AND
*			POINTER TO FIRST WORD OF RECORD IN
*			THE A-REGISTER
*			
*			

The following examples are the same as the examples above, except that out-line parameter lists are used.

Example 1

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6	7 8 14	15 16 31	32
*			
*			GET A RECORD FROM FILE ABCDEF
*			
*			
	LDA	PBUF3	FETCH POINTER TO RECORD BUFFER
*			
	STA	FCNM1+3	POINTER TO FCB
	LDX	PLST3	PARAMETER LIST
*			POINTER TO X
	GET$	(X)	GET A RECORD
	---		NORMAL RETURN, RECORD FETCHED
*			AND STORED IN BUF3
*			
*			
*			ERROR RETURN - THE A-REGISTER
*			CONTAINS THE ERROR CODE
*			
*			
ERT3	---		ERROR, ILLEGAL
*			PARAMETER IN THE FCB, FCNM1, OR
*			DISC ERROR ON INPUT OF RECORD
*			
*			
*			END OF FILE RETURN
*			
EOF1	---		EOF, ALL RECORDS FROM
*			FILE ABCDEF HAVE BEEN
*			READ

```
*
*                                   GET REQUEST
*                                   PARAMETER LIST
*
PLST3   DAC     *+1                 POINTER TO PARAMETER LIST
*
        DAC     FCNM1               FCB ADDRESS
        DAC     ERT3                ERROR RETURN ADDRESS
        DAC     EOF1                EOF RETURN ADDRESS
*
PBUF3   DAC     BUF3                RECORD BUFFER ADDRESS
BUF3    BSZ     60                  RECORD BUFFER
```

Example 2

```
*
*                                   GET A RECORD FROM FILE ABCDEF
*
*
        CRA                         0 TO RECORD BUFFER ADDRESS IN
        STA     FCNM1+3             THE FCB PARAMETER LIST
        LDX     PLST3               PARAMETER LIST
*                                   POINTER TO X
        GET$    (X)                 GET A RECORD
        ---                         NORMAL RETURN, RECORD FETCHED
*                                   AND POINTER TO FIRST WORD OF
*                                   RECORD IN THE A-REGISTER
*
*
```

PUT RECORD REQUEST MACRO (PUT$)

The PUT$ macro causes a logical record to be added to the end of the specified file being created.

PUT$ Macro Action

PUT$ transfers the logical record specified in the user's record buffer to a system buffer in a block of free memory, writes the system buffer onto the system disk when the system buffer has been filled with the data from one or more logical records (or part of a logical record if the logical record length is greater than or equal to the physical disk record), and takes the normal return to the user.

Macro Format

Location	Operation	Operand
[symbol]	PUT$	fcb address, error return address, [record buffer address]

where:

symbol — optional

Is the symbolic label of the macro instruction.

fcb address —

Is the address of the file control block which contains data the user needs to make logical I/O requests for a particular file.

error return address —

Is the address where control is to be returned if an error is found in the parameters specified in the file control block or if an error occurs as a result of requesting I/O operations.

record buffer address — Optional

Is the address of the record buffer which contains the data of the logical record to be transferred to the specified file on the system disk. If this parameter is omitted, the record buffer address specified in the file control block will be used to replace the record buffer address in the file control block and any subsequent Put requests using the same file control block will use the new record buffer address.

NOTE: This updating of the record buffer address in the FCB is performed by the execution of the in-line macro expansion and not by the macro action routine.

Normal Return

Control is returned to the caller, at the instruction following the PUT$ macro, after the Put request has been processed successfully.

Error Return

Control is returned to the error return address specified in the PUT$ macro when any of the following errors are detected:

A-register Contents (Decimal)	Error Condition
17	The allocate request cannot be honored because all of the work areas on disk are being used.
18	A disk error occurred when trying to write a physical disk record onto the system disk.
19	A Put request was issued for a file that was not previously opened or a Put request was issued for a file that was opened in input mode.
21	The record buffer address is not specified in the file control block.

Macro Action Details

The data of the logical record specified in the user's record buffer is transferred to a block of free memory. When the block of free memory is filled with one or more logical records or a part of a logical record (a part of a logical record when the logical record length is greater than or equal to the physical disk record), the block of free memory is written onto the system disk for the specified file. Control is then returned to the user at the normal return address.

NOTE: Work areas on the system disk are allocated as required so there is no limit to the size of the file except the amount of disk space that is available.

Out-Line Parameter List

Word No.	Operation	Operand
0	DAC	fcb address
1	DAC	error return address

where:

The parameters in the out-line parameter list are the same as those described above for the in-line parameter list.

NOTE: If an out-line parameter list is used and it is desired to update the FCB record buffer address, the user must precede the PUT$ macro call with instructions to store the new record buffer address in the forth word (word 3) of the FCB.

Examples:

The examples below illustrate the use of PUT$ to add a record to the end of file UVWXYZ. The File Control Block, FCNM2, which must be generated before the Put request is issued, is illustrated in the examples under Assign File Control Block, and the open request, which must be processed before the Put request is issued, is illustrated in the examples under Open File.

```
LOCATION  OPERATION  OPERAND FIELD          COMMENTS
1      6  7 8    14 15 16          31 32
*                                        
*                                        PUT A RECORD AT THE END OF
*                                        FILE UVWXYZ
*                                        
*                                        
          PUT$       FCNM2,ERT4,BUF4     
          ---                            NORMAL RETURN, RECORD TRANSFERRED
*                                        FROM BUF 4 AND ADDED TO THE
*                                        END OF FILE UVWXYZ
*                                        
*                                        
*                                        
*                                        ERROR RETURN - THE A-REGISTER
*                                        CONTAINS THE ERROR CODE
*                                        
*                                        
ERT4      ---                            ERROR, ILLEGAL
*                                        PARAMETER IN THE FCB, FCNM2,
*                                        DISK ERROR ON OUTPUT OF RECORD,
*                                        OR ALLOCATION ERROR
*                                        
*                                        
BUF4      BSZ        60                  RECORD BUFFER
```

The following example is the same as the example above, except the out-line parameter list is used.

```
LOCATION  OPERATION  OPERAND FIELD          COMMENTS
1      6  7 8    14 15 16          31 32
*                                        
*                                        PUT A RECORD AT THE END OF
*                                        FILE UVWXYZ
*                                        
*                                        
          LDA        PBUF4               FETCH POINTER TO RECORD BUFFER
```

```
*
        STA     FCNM2+3         POINTER TO FCB
        LDX     PLST4           PARAMETER LIST
*                               POINTER TO X
        PUT$    (X)             PUT A RECORD
        ---                     NORMAL RETURN, RECORD TRANSFERRED
*                               FROM BUF 4 AND ADDED TO THE
*                               END OF FILE UVWXYZ
*
*
*
*                               ERROR RETURN - THE A-REGISTER
*                               CONTAINS THE ERROR CODE
*
*
ERT4    ---                     ERROR, ILLEGAL
*                               PARAMETER IN THE FCB, FCNM2,
*                               DISK ERROR ON OUTPUT OF RECORD,
*                               OR ALLOCATION ERROR
*
*
*                               PUT REQUEST
*                               PARAMETER LIST
*
PLST4   DAC     *+1             POINTER TO PARAMETER LIST
*
        DAC     FCNM2           FCB ADDRESS
        DAC     ERT4            ERROR RETURN ADDRESS
PBUF4   DAC     BUF4            RECORD BUFFER ADDRESS
BUF4    BSZ     80              RECORD BUFFER
```

QUEUE MACROS

The queue macros allow the user to create a queue, add items to the beginning (top) or end (bottom) of the queue and remove items from the beginning of the queue. In actual operation, the Create Queue macro (CRQ$) sets up a 2-word queue header. As items are added to the beginning or end of the queue, the first word of the queue header points to the first item in the list and the second word of the queue header points to the last item in the queue.

The first word of each item in the queue points to the next item in the queue. (The first word of the last item in the queue will therefore always be zero.) Pointer operation is illustrated in Figure 4-1.

The queue macros are: Create Queue (CRQ$), Attach Entry to Queue (ATQ$), and Get Beginning Entry From Queue (GTQ$).

A queue must first be created using the CRQ$ macro. Thereafter, any reference to this queue, for the purpose of queue management, must be made via the queue header address given to the CRQ$ macro. The ATQ$ macro permits an entry to be made either to the beginning or the end of a queue. Thus LIFO (last-in, first-out), and FIFO (first-in, first-out) queues can be generated. A LIFO queue is generated by always attaching new items to the beginning of the queue. A FIFO queue is generated by always attaching new items to the end of the queue.

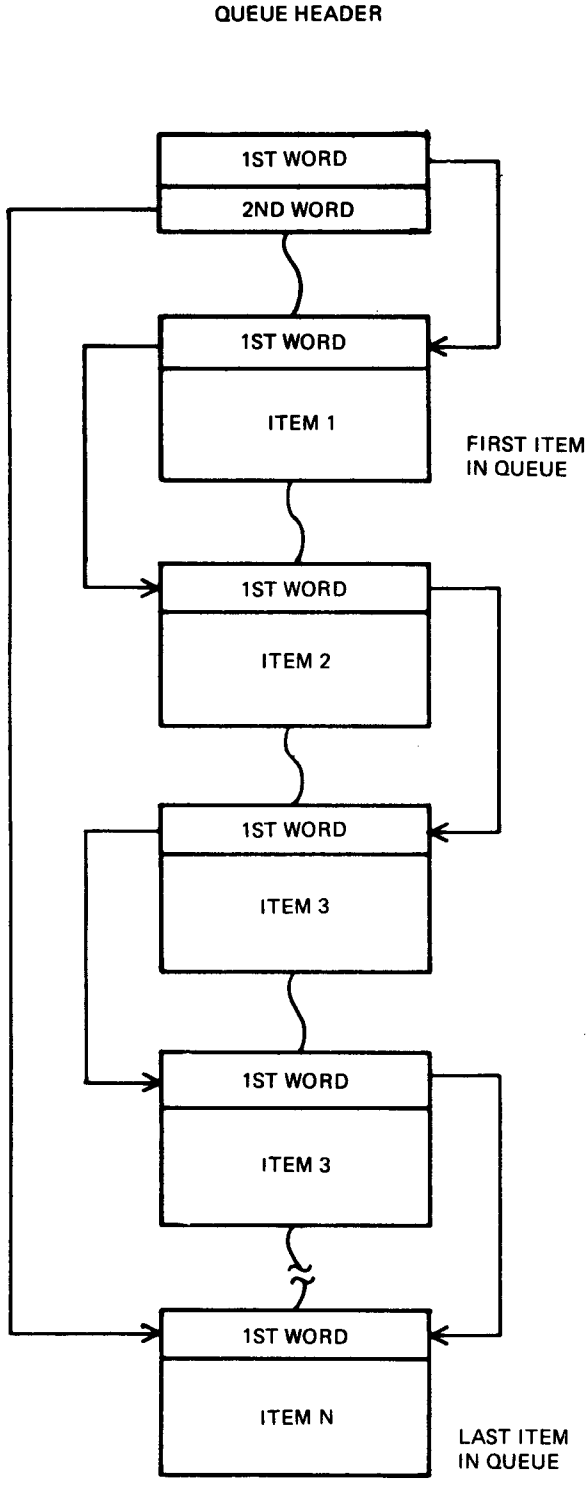


Figure 4-1. Queue Operation

CREATE QUEUE (CRQ$)

A queue may be initialized via the CRQ$ macro.

CRQ$ Macro Action

CRQ$ initializes the queue as being empty and returns to the caller via the normal return.

Macro Format

Location	Operation	Operand
[symbol]	CRQ$	queue header address, error return address

where:

symbol — Optional

Is the symbolic label of the macro instruction.

queue header address

Is the address of the first word of the queue header.

error return address

Is the address to which control is returned if an error is found during the processing of the macro call.

Normal Return

Upon normal return, the named queue header has been initialized.

ERROR Return

Currently, the error return is not taken, but is present for future use.

Macro Action Details

The two words of the queue header, identified by the queue header address parameter, are cleared to zero, initializing the queue as an empty queue. Return is to the caller via the normal return.

Parameter List

Word No.	Operation	Operand
0	DAC	queue header address
1	DAC	error return address

where:

Parameters in the out-line parameter list are the same as those described above for the in-line parameter list.

Example

The following example illustrates the use of CRQ$ to initialize the queue header specified by RTRQ.

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6	7 8 14	15 16 31	32
*			CREATE A QUEUE HEADER.
*			
	CRQ$	RTRQ,ERR	
*			
*			THE FOLLOWING WILL BE A QUEUE HEADER.
*			
RTRQ	BSZ	1	START OF QUEUE.
	BSZ	1	END OF QUEUE
ERR	---		ERROR RETURN

ATTACH ENTRY TO QUEUE (ATQ$)

The Attach Entry to Queue (ATQ$) macro permits an entry to be attached either to the begining or the end of a queue.

ATQ$ Macro Action

ATQ$ links the entry either to the beginning or the end of the queue and returns to the caller via the normal return.

Macro Format

Location	Operation	Operand
[symbol]	ATQ$	queue header address, [queue attach mode], queue entry block address, error return address

where:

symbol — Optional

Is the symbolic label of the macro instruction.

queue header address

Is the address of the first word of the queue header. This address is the same as used in the CRQ$ macro.

queue attach mode — Optional

Is an integer specifying the place in the queue to attach the entry.

0 = attach to end of queue.

1 = attach to beginning of queue.

If omitted, zero is used.

queue entry block address

Is the address of the first word of the entry block to be linked to the queue. The first word becomes the link word.

error return address

Is the address to which control is returned if an error is found during the processing of the macro call.

Normal Return

Upon normal return, the entry has been linked either to the beginning or the end of the specified queue.

Error Return

Currently, the error return is not taken, but is present for future use.

Macro Action Details

The specified entry is linked to either the beginning or end of the queue as indicated by the queue attach mode parameter. Return is made to the caller via the normal return.

Parameter List

Word No.	Operation	Operand
0	DAC	queue header address
1	DAC	[queue attach mode]
2	DAC	queue entry block address
3	DAC	error return address

where:

Parameters in the out-line parameter list are the same as those described above for the in-line parameter list.

Examples

The example below illustrates the use of ATQ$ to attach an entry to a queue. The entry specified ENT will be attached to the end of the queue RTRQ. An in-line parameter is used in this example.

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6 7	8 14 15	16 31	32
*			ATTACH AN ENTRY TO BOTTOM OF QUEUE.
*			
	ATQ$	RTRQ,,ENT,ERR	
*			
*			THE FOLLOWING IS THE ENTRY TO BE
*			ATTACHED.
ENT	BSZ	8	ENTRY = 8 WORDS.
ERR	---		ERROR RETURN

In the example below, ATQ$ is used with an out-line parameter list to attach an entry to the beginning of the queue RTRQ. The address of the first word of the entry will have been placed in word ENTA (the third word of the parameter list) before ATQ$ is executed.

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6	7 8 14	16 31	32
*			ATTACH AN ENTRY TO TOP OF QUEUE.
*			
	LDX	LIST	GET LIST ADDRESS.
	ATQ$	(X)	ATTACH ENTRY TO QUEUE.
*			
*			PARAMETER LIST FOR ATQ$.
*			
LIST	DAC	*+1	ADDRESS OF LIST.
	DAC	RTRQ	ADDRESS OF QUEUE HEADER.
	DAC	1	ATTACH TO BEGINNING OF QUEUE.
ENTA	BSZ	1	FOR ENTRY ADDRESS.
	DAC	ERR	ERROR RETURN ADDRESS.

GET BEGINNING ENTRY FROM QUEUE (GTQ$)

The Get Beginning Entry from Queue (GTQ$) macro is used to access the entry currently at the beginning of the queue, and to make the next entry in the queue the beginning entry.

GTQ$ Macro Action

GTQ$ unlinks the entry from the beginning of the queue, returns to the caller via the normal return with the address of the unlinked entry, if the queue is not empty, or returns to the caller via the error return if the queue is empty.

Macro Format

Location	Operation	Operand
[symbol]	GTQ$	queue header address, error return address

where:

symbol — Optional

Is the symbolic label of the macro instruction.

queue header address —

Is the address of the first word of the queue header. This address is the same as used in the CRQ$ macro.

error return address —

Is the address to which control is returned if an error is found during the processing of the macro call.

Normal Return

Upon normal return, the entry currently at the beginning of the queue has been unlinked. The X-register contains the address of the first word of this entry.

Error Return

Control is returned to the error return address specified in the GTQ$ macro with the error code in the A-register when the following error is detected:

A-register Contents (Decimal)	Error Condition
0	No entry in the queue. The queue is empty.

Macro Action Details

If the queue is empty, return is made to the error return. If the queue is not empty, the entry at the beginning of the queue is unlinked. Its location is placed in the X-register, and return is to the normal return.

Out-Line Parameter List

Word No.	Operation	Operand
0	DAC	queue header address
1	DAC	error return address

where:

Parameters in the out-line parameter list are the same as those described above for the in-line parameter list.

Examples:

In the following example, GTQ$ is used with an in-line parameter list to fetch the first entry from the RTRQ queue. If the queue is empty when GTQ$ is executed, then control will be transferred to location NENT so that the error can be processed.

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6	7 8 14	15 16 31	32
*			GET BEGINNING ENTRY FROM QUEUE.
*			
	GTQ$	RTRQ,NENT	
*			
*			ERROR RETURN IF QUEUE IS EMPTY.
*			
NENT	---		

The example below is the same as above, except that GTQ$ is used with an out-line parameter list.

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6	7 8 14	16 31	32
*			GET BEGINNING ENTRY FROM QUEUE.
*			
	LDX	PLST	GET LIST ADDRESS.
	GTQ$	(X)	GET ENTRY.
*			
*			PARAMETER LIST FOR GTQS.
*			
PLST	DAC	*+1	ADDRESS OF LIST.
	DAC	RTRQ	ADDRESS OF QUEUE HEADER.
	DAC	NENT	ERROR RETURN ADDRESS
*			
*			ERROR RETURN IF QUEUE IS EMPTY.
*			
NENT	---		

BLOCK MACROS

Block macros are used during a task to obtain on release a block of free memory. The block macros are: Get Storage Block (GBL$) and Return Storage Block (RBL$).

Free memory is divided into pools of different size blocks. The block sizes in the system are configurable and may range from a minimum of two words to the maximum configured length. The system provides only blocks that are 2^n words long, where n is an integer. However, the user may request blocks of any size; if the requested block size is not configured, then the next larger configured block will be given to him.

Blocks may be requested on a conditional or unconditional basis. When a block is requested conditionally and the pool for that block size is empty, the error status is placed in the A-register and a return is made to the user via the specified error address. If an unconditional request is made for a block and the pool for that block size is empty, the Executive will check the pools for larger blocks. If there is a larger block, then fragmentation will be scheduled to divide that block into smaller blocks so as to satisfy the user's request.

In this case, the user will receive the "suspend" return. If there is no larger block to be fragmented, then the error status is put in the A-register and a return is made to the user via the specified error address. If the suspend return is taken in the GBL$ macro, the user must then suspend his task, via the SUS$ macro (see the first example of the GBL$ macro). Upon resuming, the user may then issue the GBL$ call again. At this time, a block may have become available. In returning a block to free memory via the RBL$ macro, the block must have been obtained via the GBL$ macro.

GET STORAGE BLOCK (GBL$)

The Get Storage Block (GBL$) macro is used to allocate a block of free memory to the task in which the macro is used.

GBL$ Macro Action

GBL$ obtains a block of the size requested from free memory, and returns to the normal return with the address of this block in the X-register, if the block is available. GBL$ returns to the suspend return if the caller must wait until a block is available, or returns to the error return if there are no blocks or the size requested is illegal.

Macro Format

Location	Operation	Operand
[symbol]	GBL$	block-size, [get type]. [suspend return address]. error return address

where:

symbol — Optional

Is the symbolic label of the macro instruction.

block size —

A positive integer specifying the number of words required. If it is not a power of two, the next larger integer which is a power of two will be used.

get type — Optional

An integer specifying the method for obtaining the block.

0 = get block conditionally.

1 = get block unconditionally.

If omitted, zero is assumed.

suspend return address — Optional

The address to which control is returned if the caller must wait until a block of free memory is available.

This parameter is ignored if the get type parameter is zero.

error return address

Is the address to which control is returned if an error is found during the processing of the macro call.

Normal Return

Upn normal return, the address of the first word of the block obtained from free memory is in the X-register.

Suspend Return

Upon suspend return, the user must suspend to wait for a block of free memory. (See the first example.)

Error Return

Control is returned to the error return address specified in the GBL$ macro with the error code in the A-register when any of the following errors are detected:

A-register Contents (Decimal)	Error Condition
0	No blocks available.
1	Block size was zero, less than zero, or larger than the largest configured block.

Macro Action Details

The block size parameter is examined to determine if it is greater than zero. If not, return is made via error return address. Otherwise, if block size is not a power of two, the size is increased to the next higher power of two.

An attempt is made to obtain the block from free memory. If a block is available, return is made via the normal return. If a block of the size requested is not available and the get parameter is conditional, return is made via the error return. If a block of the size requested is not available, the get type parameter is unconditional, and a block of larger size than requested is available, then return is made via the suspend return parameter. If a block of the size requested is not available, the get type parameter is unconditional, and no block of any larger size is available, then return is made via the error return address.

Out-Line Parameter List

Word No.	Operation	Operand
0	DEC	block size
1	DEC	[get type]
2	DAC	[suspend return address]
3	DAC	error return address

where:

Parameters in the out-line parameter list are the same as those described above for the in-line parameter list.

Examples

In the example below, a block of 32 words is to be fetched unconditionally. If the 32-word block pool is empty and there is a larger block available for fragmentation, then control will be transferred to location SUS, where the user must suspend himself in order to allow fragmentation to run. If no 32-word block is available for fragmentation, then control will be transferred to location BER, with the "no-block" status in the A-register. If the 32-word size is larger than the largest configured block size, then control will be transferred to location BER, with the "illegal-size" status in the A-register.

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6 7	8 14 15	16 31	32
*			GET A BLOCK OF FREE MEMORY.
*			
TRY	GBL$	32,1,SUS,BER	
*			
*			FRAGMENTATION SCHEDULED, SUSPEND.
*			
SUS	SUS$		SUSPEND FOR FRAGMENTATION.
	JMP	TRY	NOW TRY AGAIN.
*			
*			ERROR RETURN, DETERMINE CAUSE.
*			
BER	SZE		WAS SIZE TOO LARGE?
	JMP	BER1	YES.
*			NO, NO BLOCK AVAILABLE.

The example below illustrates the use of GBL$ without an in-line parameter list. The action that results is the same as in the above example.

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6	7 8 14	15 16 31	32
*			GET A BLOCK OF FREE MEMORY.
*			
TRY	LDX	BLST	GET ADDRESS OF LIST.
	GBL$	(X)	GET STORAGE BLOCK.
*			
*			FRAGMENTATION SCHEDULED, SUSPEND.
*			
SUS	SUS$		SUSPEND FOR FRAGMENTATION.
	JMP	TRY	NOW TRY AGAIN.
*			
*			ERROR RETURN, DETERMINE CAUSE.
*			
BER	SZE		WAS SIZE TOO LARGE?
	JMP	BER1	YES.
*			
*			PARAMETER LIST FOR GBL$.
*			
BLST	DAC	*+1	ADDRESS OF LIST.
	DEC	32	SIZE = 32 WORDS.
	OCT	1	UNCONDITIONAL
	DAC	SUS	SUSPEND RETURN ADDRESS.
	DAC	BER	ERROR RETURN ADDRESS.

RETURN STORAGE BLOCK (RBL$)

The Return Storage Block (RBL$) macro deallocates a block and returns it to free memory.

RBL$ Macro Action

RBL$ returns the specified block to free memory, and returns to the caller via the normal return or returns to the caller via the error return if the block size is larger than that of any block size configured.

Macro Format

Location	Operation	Operand
[symbol]	RBL$	block address pointer, block size, error return address

where:

symbol — Optional

Is the symbolic label of the macro instruction.

block address pointer

the location of a word containing the address of the first word of the block to be obtained. This block address must have been obtained via the GBL$ macro

block size

An integer specifying the size of the block (in words) to be returned. This integer is the same as the block size in the GBL$ macro used to obtain the the block.

error return address

Is the address to which control is returned if an error is found during the processing of the macro call.

Normal Return

The normal return is made to the caller when the block is returned to free memory.

Error Return

Control is returned to the error return address specified in the RBL$ macro with the error code in the A-register when the following error is detected:

A-register Contents (Decimal)	Error Condition
0	Block size was zero or less, or block size was too large.

Macro Action Details

The block size parameter is examined. If it is not greater than zero, return is made via the error return. If a block size is not a power of two, the size is increased to the next higher power of two. If the result is larger than any size configured, return is made to the caller via the error return. If not, the block is returned to free memory, and return is made to the caller via the normal return.

Out-Line Parameter List

Word No.	Operation	Operand
0	DAC	block address pointer
1	DAC	block size
2	DAC	error return address

where:

Parameters in the out-line parameter list are the same as those described above for the in-line parameter list.

Examples

In the example below, the address of the 32-word block that is to be returned to free memory will have been stored in word ADDB before RBL$ is executed. If the 32-word size is larger than the largest configured block size, then control will be transferred to instruction SERR so that the error may be processed.

```
LOCATION  OPERATION  OPERAND FIELD          COMMENTS
1      6  7 8    14 15 16           31 32
*                                      RETURN STORAGE BLOCK.
*
          RBL$       ADDB,32,SERR
*
*                                      BLOCK SIZE IS IN ERROR.
*
SERR      ---
*
*                                      THE FOLLOWING LOCATION WILL CONTAIN THE
*                                      ADDRESS OF THE BLOCK TO BE RETURNED.
ADDB      BSZ        1
```

The following example is the same as the above, except that RBL$ is used with an out-line parameter list.

```
LOCATION  OPERATION  OPERAND FIELD          COMMENTS
1      6  7 8    14 15 16           31 32
*                                      RETURN STORAGE BLOCK.
*
          LDX        PLST                GET ADDRESS OF LIST.
          RBL$                           RETURN BLOCK.
*
*                                      BLOCK SIZE ERROR.
*
```

```
SERR    ---
*
*                                   PARAMETER LIST.
*
*
PLST    DAC     *+1                 ADDRESS OF LIST.
ADDB    BSZ     1                   BLOCK ADDRESS POINTER.
        DEC     32                  SIZE = 32 WORDS.
        DAC     SERR                ERROR RETURN ADDRESS.
```

SECTION V
CONTROLLING PHYSICAL I/O OPERATIONS

This chapter describes the system macros used for physical I/O operations on the disk and all other peripherals except communication equipment (Communication macros are described in Section 7). These macros are termed: physical I/O macros and disk macros.

PHYSICAL I/O MACROS

Physical I/O macros are used to initiate physical I/O operations on all peripherals that can be used under System 700. They include:

1. Assign Device Control Block (DCB$)
2. Reserve a Device (RSV$)
3. Release a Device (REL$)
4. Input (INP$)
5. Output (OTP$)
6. Write End-of-File (EOF$)
7. Space File (SPF$)
8. Space Record (SPR$)
9. Rewind (RWD$)
10. Unload (ULD$)
11. Wait for I/O (WIO$)

Physical I/O may be requested on input and output devices which have been configured into the system. Before any physical I/O request is issued, a Device Control Block must be generated, and the desired device must be reserved by issuing the RSV$ macro. Not until the desired device is reserved for the user may physical I/O requests be issued. If the device is configured as a non-sharable device, the requested device is reserved for only one user at a time. If the device is configured as a sharable device, the requested device may be reserved by any number of users at a time. If more than one user attempts to reserve a non-sharable device, the device will be assigned to the first user requesting the reservation and subsequent user's reserve requests will be queued if queuing was specified in the reserve request. The physical I/O requests include input (INP$), Output (OTP$), space file (SPF$), space record (SPR$), end-of-file (EOF$), rewind (RWD$), and unload (ULD$).

A wait I/O (WIO$) request must be issued following each of these physical I/O requests except unload. If sequential I/O is desired, the wait I/O request must be issued at the normal return which is the instruction immediately following the physical I/O request. If non-sequential I/O is desired, the user's processing may continue until the completion of the physical I/O is required; then the wait I/O request must be issued. When the physical I/O operation is completed, the user's I/O completion return is scheduled. When the user receives control at the I/O completion return, he must assume the responsibility for checking if the physical I/O operation was completed successfully. See Appendix D for a general description of the status information which is returned, and Appendix E for the specific information returned by each device driver. When all of the physical I/O requests for the device which was reserved have been completed, the device must be released by issuing a REL$ macro.

ASSIGN DEVICE CONTROL BLOCK MACRO (DCB$)

The DCB$ macro generates a list of parameters required for reserve (RSV$), release (REL$), input (INP$), output (OTP$), space file (SPF$), space record (SPR$), end-of-file (EOF$), rewind (RWD$), and unload (ULD$) requests.

DCB$ Macro Action

DCB$ generates a list of parameters which contain no executable code.

Macro Format

Location	Operation	Operand
dcb name	DCB$	generic device type, logical unit no., data mode, user id, i/o status block address, i/o completion return address

where:

dcb name

Is the symbolic label of the device control block created by the DCB$ macro instruction.

generic device type

Is a number indicating the generic type of device which is to perform the I/O operation. Refer to Appendix B for the assignments of the generic device types.

logical unit no.

Is a number that specifies the logical unit of a generic device type which is to perform the I/O operation. For example, assume that the system is configured for 2 magnetic tape controls and that each controls 4 units. The physical unit numbers for the units on each of the controls range from 0 to 3. The logical unit numbers for the units on both of the controls range from 0 to 7.

NOTE: If it is desirable to perform physical I/O to the system disk, a work area on the system disk must be allocated (see Disk Macro Description in this section). To specify the system disk as the I/O device in the DCB$ macro, omit the generic device type and the logical unit number parameters from the in-line parameter list of the DCB$ macro. To specify the system disk in the out-line parameter list, the Get System Parameter macro (see GSP$ macro description in Section II) may be used, or a "-1" may be specified as the generic device type and the logical unit number in the parameter list. When an I/O request is made with a "-1" as the generic device type, the generic device type and logical unit number for the system disk will be stored in the user's DCB$ parameter list.

data mode

Is a number which specifies the kind of data transfer that is to be performed. Refer to Appendix C for the assignment of the types of data modes.

user id

Is one word (16 bits) of identification.

i/o status block address

Is the address of an I/O status block. For I/O requests, the status block is used by the device driver to schedule the I/O completion return to the user and to store the status information as described in Appendix D. The status block is also used to schedule the queued RSV$ request return address when processing a RSV$ request that has been queued. The I/O status block address will be in the X-register when control is returned to the user at the I/O completion return address or the queued RSV$ request return address.

i/o completion return address

Is the address where control is to be returned after I/O completion.

Normal Return

There is no normal return for the DCB$ macro since the DCB$ macro performs no action.

Error Return

There is no error return for the DCB$ macro.

Macro Action Details

The DCB$ macro has no action since it is a list of parameters. The DCB$ macro must therefore be coded in a program as a data block or buffer.

Out-Line Parameter List

Word No.	Operation	Operand
0	BSZ	1 (Reserved for system use)
1	DEC	generic device type
2	DEC	logical unit no.
3	DEC	data mode
4	DEC	user ID
5	DAC	I/O status block address
6	DAC	I/O completion return address

Where:

The parameters in the out-line parameter list are the same as those described above for the in-line parameter list. However, note that the first word of the list must be a BSZ 1.

Examples

The example shown below illustrates the generation of a Device Control Block by using the DCB$ macro.

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6 7	8 14 15	16 31	32
*			GENERATE A DCB TO BE REFERENCED WHEN MAKING I/O REQUESTS
*			FOR MAGNETIC TAPE, LOGICAL UNIT NO. 1
DCNM	DCB$	1,0,1,0,,146724,T	BAD,RTAD
*			
TBAD	BSZ	8	I/O STATUS BLOCK, TBAD+1 WILL CONTAIN 0 IF NO
*			ERROR OCCURRED ON I/O COMPLETION
	LNK$		LINK TO SYSTEM

The example below shows how the Device Control Block would be written without the use of the DCB$ macro.

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6 7	8 14 15	16 31	32
*			GENERATE A DCB TO BE REFERENCED WHEN MAKING
*			I/O REQUESTS FOR MAGNETIC TAPE, LOGICAL UNIT NO. 1
DCNM	BSZ	1	RESERVED FOR SYSTEM USE.
	DEC	10	GENERIC DEVICE TYPE FOR MAGNETIC TAPE
	DEC	1	LOGICAL UNIT NUMBER 1
	DEC	0	DATA MODE (ASCII).
	BCI	1,MT	USER ID.
	DAC	TBAD	I/O STATUS BLOCK ADDRESS
	DAC	RTAD	I/O COMPLETION RETURN ADDRESS
*			
TBAD	BSZ	8	I/O STATUS BLOCK, TBAD+1 WILL CONTAIN 0 IF
*			NO ERROR OCCURRED ON I/O COMPLETION
	LNK$		LINK TO SYSTEM

RESERVE REQUEST MACRO (RSV$)

The RSV$ macro associates a particular device with a particular user ID specified in the device control block so that the requestor may be the solitary user of a device.

RSV$ Macro Action

RSV$ checks the legality of the parameters specified in the RSV$ macro and the device control block, and takes the error return if any one of the parameters is illegal. Then, a check is made to determine if the requested device has been reserved by another user; if it has, it queues the reserve request (if queueing was specified) and takes the error return. If the device has not been reserved, it reserves the requested device and returns to the RSV$ caller at the normal return.

Macro Format

Location	Operation	Operand
[symbol]	RSV$	dcb address, error return address, [queued rsv$ request return address], [control p tcb address pointer]

where:

symbol — Optional

Is the symbolic label of the macro instruction.

dcb address

Is the name of the device control block associated with the device to be reserved.

error return address

Is the address to which control is returned if an error is found in the RSV$ macro call parameters, the specified device control block parameters, or if the specified device is currently reserved for another user.

queued rsv$ request return address — Optional

Is the address to which control is transferred after a queued reserve request is processed. If this address is present in the RSV$ macro parameter list, and if the requested device was already reserved for another user when the reserve request was issued, the reserve request is queued according to the priority level of the task that issued the RSV$ macro. Control is then transferred to the error return address with zero in the A-register, which indicates that the requested device is already reserved for another user. The user who issued the reserve request must at this time issue a WIO$ macro instruction to wait for the requested device to be reserved for him. When the user's reserve request is removed from the reserve request queue and processed, control will be transferred to the user at the queued RSV$ request return address with the I/O status block address in the X-register.

If the queued RSV$ request return address is null (equal to zero) in the RSV$ macro parameter list, and the requested device was already reserved for another user when the reserve request was issued, control is transferred to the error return address with zero in the A-register. The reserve request is not queued in this case.

control p tcb address pointer – Optional

Is a pointer to a word which contains the address of a TCB. This TCB must have been created by the user before the reserve request was issued. The TCB may be created by the use of the create TCB macro (CTC$) or by getting an 8 word block from free memory using the Get Block Macro (GBL$). The TCB must contain the address of a special action routine which will be scheduled by the KSR or ASR driver when a Control P character is typed on the KSR or ASR keyboard. The special action routine must be within the same activity which contains the reserve request. This parameter is mandatory only when the requested device is either the KSR (generic device type 0) or the ASR (generic device type 12).

Normal Return

Control is returned to the caller, at the instruction following the RSV$ macro, after the requested device has been reserved, and only when the requested device was not reserved by another user at the time the reserve request was issued.

Queued RSV$ Request Return

After the WIO$ macro instruction, control will be returned to the queued RSV$ request return address when the queued reserve request is processed.

Error Return

Control is returned to the error return address specified in the RSV$ macro with the error code in the A-register when any of the following errors are detected.

A-Register Contents (Decimal)	Error Condition
0	The requested device was already reserved when another reserve request was issued.
1	The generic device type specified in the device control block is not configured for this system.
3	The logical unit number specified in the device control block is not configured for this system.
12	A reserve request was issued for a disabled device.
24	The device driver for the requested device is disk resident, and there was an activity area overrun error while loading the device driver.
25	The device driver for the requested device is disk resident, and there was a disk read error while loading the device driver.

Macro Action Details

The following checks are made to determine if the reserve request is legal:

1. A test is made to determine if the generic device type and the logical unit number specified in the device control block have been configured into the system.
2. A test is made to determine if the requested device is enabled.
3. A test is made to determine if the requested device is either a KSR or an ASR, and if it is, a test is made to determine if the control P TCB address pointer has been specified in the reserve parameter list.

If the reserve request is not legal, control is returned to the error return address specified in the RSV$ macro with the error code in the A-register. If the reserve request is legal, a test is made to determine if the device driver for the requested device is disk resident, and if it is, the device driver is brought into main memory. Then a test is made to determine if the requested device is reserved for another user. If the device is not reserved, the device is reserved for the user who issued the reserve request, and control is returned to the user at the normal return.

If the device is already reserved for another user, a test is made to determine if the queued RSV$ request return address specified in the reserve request parameter list is omitted (equal to zero). If omitted, control is returned to the error return address specified in the reserve request parameter list with zero in the A-register. If the queued RSV$ request return address is specified, the reserve request is queued according to the priority level of the task which issued the RSV$ macro, and control is returned to the error return address specified in the reserve request parameter list with zero in the A-register.

When control is returned to the user at the error return address with a zero in the A-register, and the queued RSV$ request return address was specified in the reserve parameter list, the user must issue a WIO$ macro instruction to wait for the device to be reserved for him. When the user's reserve request is processed and the device is reserved for him, control will be returned to the user at the queued RSV$ request return address.

Out-Line Parameter List

Word No.	Operation	Operand
0	DAC	dcb address
1	DAC	error return address
2	DAC	[queued rsv$ request return address]
3	DAC	[control p tcb address pointer]

where:

The parameters in the out-line parameter list are the same as those described above for the in-line parameter list. Note that if it is not desirable to have the reserve request queued when the requested device is already reserved for another user, word 2 must contain a BSZ 1.

Examples

The examples below illustrate the use of RSV$ to reserve a device so that I/O requests for Logical Unit No. 1 of the magnetic tape can be processed. The Device Control Block is designated by the name DCNM (see the examples under Assign Device Control Block, above). The Queued Reserve Request Return, QUED, specifies that the Reserve Request is to be added to the Reserve Request Queue if the Magnetic Tape, Logical Unit Number 1, has already been reserved by another user.

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6 7	8 14 15	16 31	32
*			RESERVE DEVICE
*			
	RSV$	DCNM,ERAD,QUED	RESERVE THE MAGNETIC TAPE, LOGICAL UNIT NO. 1
	---		NORMAL RETURN, RESERVE PROCESSED, I/O MAY BE ISSUED
*			
*			ERROR RETURN- A-REGISTER CONTAINS ERROR CODE.
*			
ERAD	SZE		WAS CALL QUEUED?
	JMP	ERR	NO, PARAMETER IN DCB ILLEGAL.
	JST	SETP	YES, SET UP FOR I/O CALL.
	WIO$		WAIT FOR RESERVE TO BE PROCESSED.
*			
*			QUEUED RESERVE REQUEST RETURN AFTER RESERVE
*			IS PROCESSED.
QUED	---		RESERVE PROCESSED, I/O MAY BE ISSUED.

The following example is the same as the example above, except that the out-line parameter list is used.

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6 7	8 14 15	16 31	32
*			RESERVE DEVICE
*			

```
        LDX     PLT1            PARAMETER LIST POINTER TO X.
        RSV$    (X)             RESERVE THE MAGNETIC TAPE, LOGICAL UNIT NO. 1
        ---                     NORMAL RETURN, RESERVE PROCESSED, I/O MAY BE ISSUED
*
*                               ERROR RETURN- A-REGISTER CONTAINS ERROR CODE.
*
ERAD    SZE                     WAS CALL QUEUED?
        JMP     ERR             NO, PARAMETER IN DCB ILLEGAL.
        JST     SETP            YES, SET UP FOR I/O CALL.
        WIO$                    WAIT FOR RESERVE TO BE PROCESSED
*
*                               QUEUED RESERVE REQUEST RETURN AFTER RESERVE
*                               IS PROCESSED
QUED    ---                     RESERVE PROCESSED, I/O MAY BE ISSUED
*
*                               PARAMETER LIST
*
PLT1    DAC     *+1             POINTER TO PARAMETER LIST
        DAC     DCNM            DCB ADDRESS
        DAC     ERAD            ERROR RETURN ADDRESS
        DAC     QUED            QUEUED RESERVE REQUEST RETURN ADDRESS
```

RELEASE REQUEST MACRO (REL$)

The REL$ macro releases a device previously reserved by a RSV$ macro instruction.

The REL$ Macro Action

REL$ checks to determine the legality of the parameters specified in the device control block and takes the error return if any one of the parameters is illegal. Otherwise, it releases the specified device. REL$ also checks the reserve request queue for users waiting to reserve the released device, and reserves the device for the user of the highest priority. Then, it returns to the REL$ caller at the normal return.

Macro Format

Location	Operation	Operand
[symbol]	REL$	dcb address, error return address

where:

symbol — Optional

Is the symbolic label of the macro instruction.

dcb address

Is the name of the device control block associated with the device to be released.

error return address

Is the address to which control is returned if an error is found in the specified device control block parameters.

Normal Return

After the specified device has been released, control is returned to the caller, at the instruction following the REL$ macro.

Error Return

Control is returned to the error return address specified in the REL$ macro with the error code in the A-register when any of the following errors are detected:

A-register Contents (Decimal)	Error Condition
1	The generic device type parameter specified in the device control block is not configured for this system.
3	The logical unit number specified in the device control block is not configured for this system.
8	The device has not been previously reserved under the user ID specified in the device control block.

Macro Action Details

The following checks are made to determine if the release request is legal:

1. A test is made to determine if the generic device type and the logical unit number specified in the device control block have been configured into the system.
2. A test is made to determine if the user ID specified in the device control block matches the ID for which the device was reserved.

If the release request is not legal, control is returned to the error return address specified in the REL$ macro with the error code in the A-register. If the release request is legal, the specified device is released. A check is then made to determine if there are entries in

the reserve request queue. If there are no users waiting to reserve the device, control is returned to the user at the normal return. If there are one or more entries in the reserve request queue, the device is reserved for the user of highest priority, the queued RSV$ request return address is scheduled and control is returned to the user who issued the release request at the normal return.

Out-Line Parameter List

Word No.	Operation	Operand
0	DAC	dcb address
1	DAC	error return address

where:

The parameters in the out-line parameter list are the same as those described above for the in-line parameter list.

Examples

The examples shown below illustrate the use of REL$ to release the Magnetic Tape, Logical Unit Number 1, specified in the Device Control Block, DCNM (see the examples under Assign Device Control Block, above).

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6 7 8	14 15 16	31 32	
*			RELEASE DEVICE
*			
	REL$	DCNM,ERAD	RELEASE LOGICAL UNIT NO. 1 OF THE MAGNETIC TAPE
	---		NORMAL RETURN, DEVICE RELEASED
*			
*			ERROR RETURN-A- REGISTER CONTAINS THE ERROR CODE
*			
ERAD	---		ERROR, ILLEGAL PARAMETER IN THE DCB, DCNM

The following example is the same as the example above, except that the out-line parameter list is used.

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6	7 8 14	15 16 31	32
*			RELEASE DEVICE
*			
	LDX	PLS3	PARAMETER LIST POINTER TO X
	REL$	(X)	RELEASE LOGICAL UNIT NO. 1 OF THE MAGNETIC TAPE
	---		NORMAL RETURN, DEVICE RELEASED.
*			
*			ERROR RETURN- A-REGISTER CONTAINS THE ERROR CODE
*			
ERAP	---		ERROR, ILLEGAL PARAMETER IN THE DCB, DCNM
*			
*			PARAMETER LIST
*			
PLS3	DAC	*+1	POINTER TO PARAMETER LIST
	DAC	DCNM	DCB ADDRESS
	DAC	ERAP	ERROR RETURN ADDRESS

INPUT REQUEST MACRO (INP$)

The INP$ macro initiates the transfer of a record from a specified I/O device into a specified area of main memory.

INP$ Macro Action

INP$ checks to determine if the specified device has been previously reserved by a RSV$ macro instruction; if not, it takes the error return. Then, a check is made to determine the legality of the parameters specified in the INP$ macro list and the device control block. It takes the error return if any one of the parameters is illegal. INP$ then queues the input request according to the priority level of the task which issued the INP$ macro, initiates the input request if the specified device is not currently busy processing another request, and returns to the INP$ caller at the normal return at which time the WIO$ macro instruction must be issued.

Macro Format

Location	Operation	Operand
symbol	IMP$	dcb address, error return address, buffer address, range, [mass memory segment no. address], [i/o status block address], [i/o completion return address]

where:

symbol — Optional

Is the symbolic label of the macro instruction.

dcb address

Is the name of the device control block associated with the input device.

error return address

Is the address to which control is returned if an error is found in the INP$ macro call parameters, the specified device control block parameters, or if the input device was not previously reserved by a RSV$ macro instruction.

NOTE: Control does not return to the error return address if an error occurs during the retrieval from the input device. A user tests for retrieval error by checking the I/O status block after control is returned to the I/O completion return address.

buffer address

Is the address of the first word of the buffer into which the record is to be placed.

range

Is the number of words to be read, from a minimum of 1 to a maximum of 4095. The number of words read into the buffer will be determined by the expiration of the range count or the end-of-record, whichever occurs first.

mass memory segment no. address — Optional

Is the address of a word containing the segment number from which the record is to be read if inputting from a mass memory device (disk). This parameter is omitted for all but mass memory devices.

i/o status block address — Optional

Is the address of an 8 word I/O status block used by the device driver to schedule the I/O completion return to the user and to store the status information as described in Appendix D. The I/O status block address will be in the X-register when control is returned to the user at the I/O completion return address. If this parameter is omitted, the I/O status block address specified in the device control block will be used. If this parameter is present, it will be used to replace the I/O status block address in the device control block and any subsequent I/O call using the same device control block will use the new I/O status block address.

NOTE: Updating of the I/O status block address in the DCB is performed by the execution of the in-line macro expansion and not by the macro action routine.

i/o completion return address — Optional

Is the address where control is to be returned after I/O completion. If this parameter is omitted, the I/O completion return address specified in the device control block will be used. If this parameter is present, it will be used to replace the I/O completion return address in the device control block and any subsequent I/O call using the same device control block will use the new I/O completion return address.

NOTE: Updating of the I/O completion return address in the DCB is performed by the execution of the in-line macro expansion and not by the macro action routine.

Normal Return

Control is returned to the caller, at the instruction following the INP$ macro, after the record input has been initiated (which may be before it has been retrieved). The caller must wait for input completion by issuing a WIO$ macro instruction.

I/O Completion Return

After a WIO$ macro instruction, control will be returned to the I/O completion return address whether the record has been retrieved successfully or unsuccessfully. The user must then test for input errors by checking the I/O status block (see I/O status block error code in Appendix D).

Error Return

Control is returned to the error return address specified in the INP$ macro with the error code in the A-register when any of the following errors are detected:

A-Register Contents (Decimal)	Error Condition
1	The generic device type specified in the device control block is not configured for this system.
3	The logical unit number specified in the device control block is not configured for this system.
4	The data mode specified in the device control block is not in the range 0 to 4, or is not a valid mode for the specified input device.
5	The number of words specified in the range parameter is less than 1 or more than 4095.
8	The device has not been previously reserved under the user ID specified in the device control block.
9	An input request was issued for an output only device.
12	An input request was issued for a disabled device.
26	The input device has a disk-resident driver and the device was not previously reserved by a RSV$ macro instruction.

Macro Action Details

The following checks are made to determine if the input request is legal:

1. A test is made to determine if the generic device type and the logical unit number specified in the device control block have been configured into the system.
2. A test is made to determine if the requested device is enabled.

3. A test is made to determine if the requested device has been previously reserved by a RSV$ macro instruction.

4. A test is made to determine if the user ID specified in the device control block matches the ID for which the device was reserved.

5. A test is made to determine if the data mode specified in the device control block is within the range from 0 to 4, and is a valid mode for the specified device.

6. A test is made to determine if the range value specified in the INP$ macro is within the limits from 1 to 4095.

7. A test is made to determine if the specified device is an input device.

If the input request is not legal, control is returned to the error return address specified in the INP$ macro with the error code in the A-register. If the input request is legal, the input request is queued according to the priority level of the task which issued the INP$ macro. If the requested device is currently busy processing another request, control is returned to the caller at the normal return address. If the requested device is not currently busy, the input request is initiated, and control is returned to the caller at the normal return. When the caller receives control at the normal return, a WIO$ macro must be issued immediately if the caller desires sequential input. If non-sequential input is desired, the caller continues processing to the point that the input completion is required. At this point, a WIO$ macro must be issued. Upon completion of the requested input, the I/O completion return address will be scheduled. When the caller receives control at the I/O completion return address, the caller must assume the responsibility of interrogating the I/O status block to determine if the input request was successful. Word 1 of the status block will be zero if no error occurred. (See Appendix D for the status information).

Out-Line Parameter List

Word No.	Operation	Operand
0	DAC	dcb address
1	DAC	error return address
2	DAC	buffer address
3	DEC	range
4	DAC	[mass memory segment no. address]

where:

The parameters in the out-line parameter list are the same as those described above for the in-line parameter list.

NOTE: If an out-line parameter list is used and it is desired to update the DCB I/O status block address or the I/O completion return address, the user must precede the INP$ macro call with the instructions to store the new I/O status block address in the sixth word (word 5) of the DCB or the new I/O completion return address in the seventh word (word 6) of the DCB.

Examples

The examples below illustrate the use of INP$ to input sequentially from magnetic tape. The call is sequential because WIO$ is issued immediately following the input request. The Device Control Block, DCNM, is shown in the example given under Assign Device Control Block, and the Reserve request is shown in the example given under Reserve. The buffer address is BFAD. The I/O completion return address, RTAD, is overwritten and

becomes NMAD, but the I/O status block address, TBAD, remains the same as in the DCB, since a new I/O status block address is not specified in the input request.

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6	7 8 14	16 31	32
*			INPUT A 60-WORD RECORD FROM MAGNETIC TAPE,
*			LOGICAL UNIT NO. 1
	INP$	DCNM,ERAD,BFAD,,60,,,NMAD	INPUT A RECORD
	WIO,$	PTBAD	SEQUENTIAL, WAIT FOR COMPLETION OF INPUT
*			I/O COMPLETION RETURN
NMAD	LDA	TBAD+1	FETCH THE STATUS
	SZE		DID AN ERROR OCCUR?
	JMP	ERR	YES, PROCESS ERROR
	---		NO ERROR, PROCESS DATA
*			
*			ERROR RETURN-A-REGISTER CONTAINS THE ERROR CODE
*			
ERAD	---		ERROR, ILLEGAL PARAMETER IN THE INP$ MACRO LIST OR
*			IN THE DCB, DCNM.
PTBAD	DAC	TBAD	I/O STATUS BLOCK ADDRESS

The following example is the same as the example above, except that the out-line parameter list is used.

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6	7 8 14	16 31	32
*			INPUT A 60-WORD RECORD FROM MAGNETIC TAPE,
*			LOGICAL UNIT NO. 1
	LDA	PNAD	A-REGISTER SET TO DAC NMAD
	STA	DCNM+6	STORE I/O COMPLETION RETURN ADDRESS IN THE DCB
	LDX	PLST	POINTER TO PARAMETER LIST TO X

```
        INP$    (X)             INPUT
        WIO$    PTBAD           SEQUENTIAL, WAIT FOR COMPLETION OF INPUT
*                               I/O COMPLETION RETURN
NMAD    LDA     TBAD+1          FETCH THE STATUS
        SZE                     DID AN ERROR OCCUR?
        JMP     ERR             YES, PROCESS ERROR
        ---                     NO ERROR, PROCESS DATA
*
*                               ERROR RETURN - A-REGISTER CONTAINS THE ERROR CODE
*
ERAD    ---                     ERROR, ILLEGAL PARAMETER IN THE INP$ MACRO LIST
*                               OR IN THE DCB, DCNM
*                               PARAMETER LIST
*
PLST    DAC     *+1             POINTER TO PARAMETER LIST
        DAC     DCNM            DCB ADDRESS
        DAC     ERAD            ERROR RETURN ADDRESS
        DAC     BFAD            BUFFER ADDRESS
        DEC     60              RANGE

PTBAD   DAC     TBAD            I/O STATUS BLOCK ADDRESS
PNAD    DAC     NMAD            I/O COMPLETION RETURN ADDRESS
```

OUTPUT REQUEST MACRO (OTP$)

The OTP$ macro initiates the transfer of a record from a specified area of main memory to a specified I/O device.

OTP$ Macro Action

OTP$ checks to determine if the specified device has been previously reserved by a RSV$ macro instruction, and, if not, takes the error return. Next, the macro checks the legality of the parameters specified in the OTP$ macro and the device control block and takes the error return if any one of the parameters is illegal. Then, the macro queues the

output request according to the priority level of the task which issued the OTP$ macro, initiates the output request if the specified device is not currently busy processing another request, and returns to the OTP$ caller at the normal return at which time the WIO$ macro instruction must be issued.

Macro Format

Location	Operation	Operand
symbol	OTP$	dcb address, error return address, buffer address, range, [mass memory segment no. address], [i/o status block address], [i/o completion return address]

where:

symbol — Optional

Is the symbolic label of the macro instruction.

dcb address

Is the name of the device control block associated with the output device.

error return address

Is the address to which control is returned if an error is found in the macro call parameters or the specified device control block parameters, or if the output device was not previously reserved by a RSV$ macro instruction.

NOTE: Control does not return to the error return address if an error occurs during the transfer from the output device. A user must test for a transfer error by checking the I/O status block after control is returned to the I/O completion return address.

buffer address

Is the address of the buffer from which the record is to be transferred.

range

Is the number of words to be transferred, from a minimum of 1 to a maximum of 4095. The number of words transferred from the buffer will be determined by the expiration of the range count or the end-of-record, whichever occurs first.

mass memory segment no. address — Optional

Is the address of a word containing the segment number into which the record is to be placed if outputting to a mass memory device (disk).

This parameter is omitted for all but mass memory devices.

i/o status block address — Optional

Is the address of an 8-word I/O status block used by the device driver to schedule the I/O completion return to the user and to store the status information as described in Appendix D. The I/O status block address will be in the X-register when control is returned to the user at the I/O completion return address. If this parameter is omitted, the I/O status block address specified in the device control block will be used. If this parameter is present, it will be used to replace the I/O status block address in the device control block and any subsequent I/O call using the same device control block will use the new I/O status block address.

NOTE: Updating of the I/O status block address in the DCB is performed by the execution of the in-line macro expansion and not by the macro action routine.

i/o completion return address — Optional

Is the address where control is to be returned after I/O completion.

If this parameter is omitted, the I/O completion return address specified in the device control block will be used. If this parameter is present, it will be used to replace the I/O completion return address in the device control block and any subsequent I/O call using the same device control block will use the new I/O completion return address.

NOTE: Updating of the I/O completion return address in the DCB is performed by the execution of the in-line macro expansion and not by the macro action routine.

Normal Return

Control is returned to the caller, at the instruction following the OTP$ macro, after the output request has been initiated. This may occur before the record to be output has been transferred. The caller must wait for output completion by issuing a WIO$ macro instruction.

I/O Completion Return

After a WIO$ macro instruction, control will be returned to the I/O completion return address whether the record has been transferred successfully or unsuccessfully. The user must then test for output errors by checking the I/O status block (see I/O status block error code in Appendix D).

Error Return

Control is returned to the error return address specified in the OTP$ macro with the error code in the A-register when any of the following errors are detected:

A-register Contents (Decimal)	Error Condition
1	The generic device type specified in the device control block is not configured for this system.
3	The logical unit number specified in the device control block is not configured for this system.
4	The data mode specified in the device control block is not in the range 0 to 4, or is not a valid mode for the specified output device.
5	The number of words specified in the range parameter is less than 1 or more than 4095.
8	The device has not been previously reserved under the user ID specified in the device control block.

A-register Contents (Decimal)	Error Condition
10	An output request was issued for an input only device.
12	An output request was issued for a disabled device.
26	The output device has a disk-resident driver and the device was not previously reserved by a RSV$ macro instruction.

Macro Action Details

The following checks are made to determine if the output request is legal:

1. A test is made to determine if the generic device type and the logical unit number specified in the device control block have been configured into the system.
2. A test is made to determine if the requested device is enabled.
3. A test is made to determine if the requested device has been previously reserved by a RSV$ macro instruction.
4. A test is made to determine if the user ID specified in the device control block matches the ID for which the device was reserved.
5. A test is made to determine if the data mode specified in the device control block is within the range from 0 to 4, and is a valid mode for the specified device.
6. A test is made to determine if the range value specified in the OTP$ macro is within the limits from 1 to 4095.
7. A test is made to determine if the specified device is an output device.

If the output request is not legal, control is returned to the error return address specified in the OTP$ macro with the error code in the A-register. If the output request is legal, the output request is queued according to the priority level of the task which issued the OTP$ macro. If the requested device is currently busy processing another request, control is returned to the caller at the normal return. If the requested device is not currently busy, the output request is initiated, and control is returned to the caller at the normal return. When the caller receives control at the normal return, a WIO$ macro must be issued immediately if the caller desires sequential output. If nonsequential output is desired, the caller continues processing to the point that the output completion is required. At this point, a WIO$ macro must be issued.

Upon completion of the requested output, the I/O completion return address will be scheduled. When the caller receives control at the I/O completion return address, the caller must assume the responsibility of interrogating the I/O status block to determine if the output request was successful. Word 1 of the status block will be zero if no error occurred. (See Appendix D for the status information.)

Out-Line Parameter List

Word No.	Operation	Operand
0	DAC	dcb address
1	DAC	error return address
2	DAC	buffer address
3	DEC	range
4	DAC	[mass memory segment no. address]

where:

Parameters in the out-line parameter list are the same as those described above for the in-line parameter list.

NOTE: If an out-line parameter list is used and it is desired to update the DCB I/O status block address or the I/O completion return address, the user must precede the OTP$ macro call with the instructions to store the new I/O status block address in the sixth word (word 5) of the DCB or the new I/O completion return address in the seventh word (word 6) of the DCB.

Examples

The examples below illustrate the use of OTP$ to output multiple files to the magnetic tape nonsequentially. The calls are nonsequential because processing of data is done after the output request is issued. The Device Control Block, DCNM, and the Reserve request are shown in the examples given under Assign Device Control Block and Reserve, respectively. The I/O status block address and the I/O completion return address are changed in the DCB for each output request.

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6	7 8 14	15 16 31	32
*			OUTPUT MULTIPLE 60-WORD RECORDS TO MAGNETIC
*			TAPE, LOGICAL UNIT NO. 1, NONSEQUENTIALLY
*			
	JST	STB1	SET UP BUF1 FOR OUTPUT
RETT	OTP$	DCNM,ERA,BUF1,60	,,STA1,RET1 OUTPUT BUF1
	JST	STB2	SET UP BUF2 FOR OUTPUT
	WIO$	PST1	WAIT FOR OUTPUT COMPLETION OF BUF1
RET1	LDA	1,1	OUTPUT OF BUF1 COMPLETE, FETCH STATUS, X CONTAINS POINTER TO STA1
	SZE		DID AN ERROR OCCUR?
	JMP	ERR	YES, PROCESS ERROR
	IRS	CNTR	HAS ALL DATA BEEN OUTPUT?
	SKP		NO
	JMP	FNSH	YES, ALL DATA OUTPUT

```
          OTP$    DCNM,ERA,BUF2,6,0,,STA2,RET2     OUTPUT BUF2
          JST     STB1                             SET UP BUF1 FOR OUTPUT
          WIO$    PST2                             WAIT FOR OUTPUT COMPLETION OF BUF2
RET2      LDA     1,3,1                            OUTPUT OF BUF2 COMPLETE, FETCH STATUS, X CONTAINS POINTER TO STA2
          SZE                                      DID AN ERROR OCCUR?
          JMP     ERR                              YES, PROCESS ERROR
          IRS     CNTR                             HAS ALL DATA BEEN OUTPUT?
          JMP     RETT                             NO, GO TO OUTPUT BUF1
*
*
*         ERROR RETURN-A-REGISTER CONTAINS ERROR CODE
*
ERA       ---                                      ERROR, ILLEGAL PARAMETER IN THE OTP$ MACRO LIST OR IN THE DCB, DCNM
PST1      DAC     STA1                             I/O STATUS BLOCK ADDRESS
PST2      DAC     STA2                             I/O STATUS BLOCK ADDRESS
```

The following example is the same as the example above, except that the out-line parameter list is used.

```
LOCATION  OPERATION  OPERAND FIELD                  COMMENTS
1      6  7 8    14 15 16                   31 32
*                                                 OUTPUT MULTIPLE 60-WORD RECORDS TO MAGNETIC
*                                                 TAPE, LOGICAL UNIT NO. 1, NONSEQUENTIALLY.
*
          JST     STB1                            SET UP BUF1 FOR OUTPUT
RETT      LDA     PST1                            A-REGISTER SET TO DAC STA1
          STA     DCNM+5                          STORE I/O STATUS BLOCK ADDRESS IN THE DCB
          LDA     PRT1                            A-REGISTER SET TO DAC RET1
          STA     DCNM+6                          STORE I/O COMPLETION RETURN ADDRESS IN THE DCB
          LDX     PLS8                            POINTER TO PARAMETER LIST TO X
          OTP$    (X)                             OUTPUT BUF1
          JST     STB2                            SET UP BUF2 FOR OUTPUT
          WIO$    PST1                            WAIT FOR OUTPUT COMPLETION OF BUF1
```

```
RET1    LDA     1,1             OUTPUT OF BUF1 COMPLETE, FETCH STATUS, X CONTAINS A POINTER TO STA1
        SZE                     DID AN ERROR OCCUR?
        JMP     ERR             YES, PROCESS ERROR
        IRS     CNTR            HAS ALL DATA BEEN OUTPUT?
        SKP                     NO
        JMP     FNSH            YES, ALL DATA OUTPUT
        LDA     PST2            A-REGISTER SET TO DAC STA2
        STA     DCNM+5          STORE I/O STATUS BLOCK ADDRESS IN THE DCB
        LDA     PRT2            A-REGISTER SET TO DAC RET2
        STA     DCNM+6          STORE I/O COMPLETION RETURN ADDRESS IN THE DCB
        LDX     PLS9            POINTER TO PARAMETER LIST TO X
        OTP$    (X)             OUTPUT BUF2
        JST     ST&1            SET UP BUF1 FOR OUTPUT
        WIO$    PST2            WAIT FOR COMPLETION OF BUF2
RET2    LDA     1,1             OUTPUT OF BUF2 COMPLETE, FETCH STATUS, X CONTAINS A POINTER TO STA2
        SZE                     DID AN ERROR OCCUR?
        JMP     ERR             YES, PROCESS ERROR
        IRS     CNTR            HAS ALL DATA BEEN OUTPUT?
        JMP     RETT            NO, GO TO OUTPUT BUF1
*
*       ERROR RETURN-A-REGISTER CONTAINS ERROR CODE
*
ERA     ---     ERROR, ILLEGAL PARAMETER IN THE OTP$ MACRO LIST OR IN THE DCB, DCNM
*
*       PARAMETER LIST FOR BUF1 OUTPUT
*
PLS8    DAC     *+1             POINTER TO PARAMETER LIST
        DAC     DCNM            DCB ADDRESS
        DAC     ERA             ERROR RETURN ADDRESS
        DAC     BUF1            BUFFER ADDRESS
        DEC     60              RANGE
*
*       PARAMETER LIST FOR BUF2 OUTPUT
```

```
*
PLS9    DAC     *+1             POINTER TO PARAMETER LIST
        DAC     DCNM            DCB ADDRESS
        DAC     ERA             ERROR RETURN ADDRESS
        DAC     BUF2            BUFFER ADDRESS
        DEC     60              RANGE

PST1    DAC     STA1            I/O STATUS BLOCK ADDRESS
PRT1    DAC     RET1            I/O COMPLETION RETURN ADDRESS

PST2    DAC     STA2            I/O STATUS BLOCK ADDRESS
PRT2    DAC     RET2            I/O COMPLETION RETURN ADDRESS
```

END OF FILE REQUEST MACRO (EOF$)

The EOF$ macro initiates the writing of an end of file on the specified device.

The EOF$ Macro Action

EOF$ checks to determine if the specified device has been previously reserved by a RSV$ macro instruction; if the device has not, the macro takes the error return. Next, the macro checks to determine the legality of the parameters specified in the device control block and takes the error return if any one of the parameters is illegal. Then, the macro queues the end-of-file request according to the priority level of the task which issued the EOF$ macro, initiates the end-of-file request if the specified device is not currently busy processing another request, and returns to the EOF$ caller at the normal return, at which time the WIO$ macro instruction must be issued.

Macro Format

Location	Operation	Operand
[symbol]	EOF$	dcb address, error return address, [i/o status block address], [i/o completion return address]

where:

symbol — Optional

Is the symbolic label of the macro instruction.

dcb address

Is the name of the device control block associated with the end-of-file device.

error return address

Is the address to which control is returned if an error is found in the specified device control block parameters or if the specified device was not previously reserved by a RSV$ macro instruction.

NOTE: Control does not return to the error return address if an error occurs during the writing of the end-of-file. The user must test for such an error by checking the I/O status block after control is returned to the I/O completion return address.

i/o status block address — Optional

Is the address of an 8 word I/O status block used by the device driver to schedule the I/O completion return to the user and to store the status information as described in Appendix D. The I/O status block address will be in the X-register when control is returned to the user at the I/O completion return address. If this parameter is omitted, the I/O status block address specified in the device control block will be used. If this parameter is present, it will be used to replace the I/O status block address in the device control block and any subsequent I/O call using the same device control block will use the new I/O status block address.

NOTE: Updating of the I/O status block address in the DCB is performed by the execution of the in-line macro expansion and not by the macro action routine.

i/o completion return address — Optional

Is the address where control is to be returned after I/O completion. If this parameter is omitted, the I/O completion return address specified in the device control block will be used. If this parameter is present, it will be used to replace the I/O completion return address in the device control block and any subsequent I/O call using the same device control block will use the new I/O completion return address.

NOTE: Updating of the I/O completion return address in the DCB is performed by the execution of the in-line macro expansion and not by the macro action routine.

Normal Return

Control is returned to the caller, at the instruction following the EOF$ macro, after the writing of the end-of-file has been initiated. This may occur before the writing is complete. The caller must wait for the completion of the writing of the end-of-file by issuing a WIO$ macro instruction.

I/O Completion Return

After a WIO$ macro instruction, control will be returned to the I/O completion return address whether the end-of-file has been written successfully or unsuccessfully. The user must then test for an error which occurred during the writing of the end-of-file by checking the I/O status block (see I/O Status Block Error Code in Appendix D).

Error Return

Control is returned to the error return address specified in the EOF$ macro with the error code in the A-register when any of the following errors are detected:

A-Register Contents (Decimal)	Error Condition
1	The generic device type specified in the device control block is not configured for this system.
3	The logical unit number specified in the device control block is not configured for this system.
7	The end-of-file request was issued for a device other than magnetic tape, the high speed paper tape punch, or the ASR paper tape punch.
8	The device has not been previously reserved under the user ID specified in the device control block.
12	An end-of-file request was issued for a disabled device.
26	The specified device has a disk-resident driver and the device was not previously reserved by a RSV$ macro instruction.

Macro Action Details

The following checks are made to determine if the end-of-file request is legal:

1. A test is made to determine if the generic device type and the logical unit number specified in the device control block have been configured into the system.
2. A test is made to determine if the specified device is enabled.
3. A test is made to determine if the specified device has been previously reserved by a RSV$ macro instruction.
4. A test is made to determine if the user ID specified in the device control block matches the ID for which the device was reserved.
5. A test is made to determine if the requested device is magnetic tape, the high speed paper tape punch, or the ASR paper tape punch.

If the end-of-file request is not legal, control is returned to the error return address specified in the EOF$ macro with the error code in the A-register. If the end-of-file request is legal, the end-of-file request is queued according to the priority level of the task which issued the EOF$ macro. If the specified device is currently busy processing another request, control is returned to the caller at the normal return. If the specified device is not currently busy, the end-of-file request is initiated, and control is returned to the caller at the normal return. When the caller receives control at the normal return, a WIO$ macro must be issued immediately if the caller desires sequential I/O. If non-sequential I/O is desired, the caller continues processing to the point that the end-of-file completion is required. At this point, a WIO$ macro must be issued. Upon completion of the end-of-file, the I/O completion return address will be scheduled. When the caller receives control at the I/O completion return address, the caller must assume the responsibility of interrogating the I/O status block to determine if the end-of-file request was successful.

Word 1 of the status block will be zero if no error occurred. (See Appendix D for the status information.)

Out-Line Parameter List

Word No.	Operation	Operand
0	DAC	dcb address
1	DAC	error return address

where:

Parameters in the out-line parameter list are the same as those described above for the in-line parameter list.

NOTE: If an out-line parameter list is used and it is desired to update the DCB I/O status block address or the I/O completion return address, the user must precede the macro call with the instructions to store the new I/O status block address in the sixth word (word 5) of the DCB or the new I/O completion return address in the seventh word (word 6) of the DCB.

Examples

The following examples illustrate the use of EOF$ to write an end-of-file mark on a magnetic tape. The Device Control Block, DCNM, and the Reserve Request, which must be executed prior to the request to write an end-of-file, have been illustrated previously. The end-of-file request is sequential, and the I/O status block address and the I/O completion return address remain as shown in DCNM, since they are not specified in the end-of-file request.

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6	7 8 14	15 16 31	32
*			WRITE END-OF-FILE ON MAGNETIC TAPE
*			
	EOF$	DCNM,EFER	WRITE EOF
	WIO$		SEQUENTIAL, WAIT FOR WRITING OF EOF TO BE COMPLETED
RTAD	LDA	TBAD+1	EOF WRITTEN, FETCH STATUS
	SZE		DID AN ERROR OCCUR?
	JMP	ERR	YES, PROCESS ERROR
	---		NO ERROR, CONTINUE
*			
*			ERROR RETURN-A-REGISTER CONTAINS THE ERROR CODE
*			
EFER	---		ERRORS, ILLEGAL PARAMETER IN THE DCB, DCNM

The following example is the same as the example above, except that the out-line parameter list is used.

```
LOCATION  OPERATION  OPERAND FIELD      COMMENTS
1      6  7 8    14 15 16        31 32
*                                    WRITE END-OF-FILE ON MAGNETIC TAPE
*
          LDX       PL,3             POINTER TO PARAMETER LIST TO X
          EOF$      (X)              WRITE EOF
          WIO$                       SEQUENTIAL, WAIT FOR WRITING OF EOF TO BE COMPLETED
RTAD      LDA       TBAD+1           I/O COMPLETION RETURN, FETCH STATUS
          SZE                        DID AN ERROR OCCUR?
          JMP       ERR              YES, PROCESS ERROR
          ---                        NO ERROR, CONTINUE
*
*                                    ERROR RETURN-A-REGISTER CONTAINS THE ERROR CODE
*
EFER      ---                        ERROR, ILLEGAL PARAMETER IN THE DCB, DCNM
*
*                                    PARAMETER LIST
*
PL,3      DAC       *+1              POINTER TO PARAMETER LIST
          DAC       DCNM             DCB ADDRESS
          DAC       EFER             ERROR RETURN ADDRESS
```

SPACE FILE REQUEST MACRO (SPF$)

The SPF$ macro initiates the spacing of one or more files on a magnetic tape.

SPF$ Macro Action

SPF$ checks to determine if the specified device has been previously reserved by a RSV$ macro instruction; if the device has not been reserved, the macro takes the error return. Next, the macro checks the legality of the parameters specified in the SPF$ macro and the device control block and takes the error return if any one of the parameters is illegal. If legal, the macro queues the space file request according to the priority level

of the task which issued the SPF$ macro, initiates the space file request if the specified device is not currently busy processing another request, and returns to the SPF$ caller at the normal return at which time the WIO$ macro instruction must be issued.

Macro Format

Location	Operation	Operand
[symbol]	SPF$	dcb address, error return address, no. of files address, [i/o status block address], [i/o completion return address]

where:

symbol — Optional

Is the symbolic label of the macro instruction.

dcb address

Is the name of the device control block associated with the space file device.

error return address

Is the address to which control is returned if an error is found in the SPF$ macro call parameters, the specified device control block parameters, or if the specified device was not previously reserved by a RSV$ macro instruction.

NOTE: Control does not return to the error return address if an error occurs during the spacing of a file. The user must test for such an error by checking the I/O status block after control is returned to the I/C completion return address.

no. of files address

Is the address of a word containing the number of files to be spaced. A positive number indicates forward spacing, and a negative number indicates backward spacing.

i/o status block address — Optional

Is the address of an 8-word I/O status block used by the device driver to schedule the I/O completion return to the user and to store the status information as described in Appendix D. The I/O status block address will be in the X-register when control is returned to the user at the I/O completion return address. If this parameter is omitted, the I/O status block address specified in the device control block will be used. If this parameter is present, it will be used to replace the I/O status block address in the device control block and any subsequent I/O call using the same device control block will use the new I/O status block address.

NOTE: Updating of the I/O status block address in the DCB is performed by the execution of the in-line macro expansion and not by the macro action routine.

i/o completion return address — Optional

Is the address to which control is returned after I/O completion.

If this parameter is omitted, the I/O completion return address specified in the device control block will be used. If this parameter is present, it will be used to replace the I/O completion return address in the device control block and any subsequent I/O call using the same device control block will use the new I/O completion return address.

NOTE: Updating of the I/O completion return address in the DCB is performed by the execution of the in-line macro expansion and not by the macro action routine.

Normal Return

Control is returned to the caller, at the instruction following the SPF$ macro, after the spacing of the file has been initiated. This may occur before the spacing is complete. The caller must wait for space file completion by issuing a WIO$ macro instruction.

I/O Completion Return

After a WIO$ macro instruction, control will be returned to the I/O completion return address whether the file(s) has been spaced successfully or unsuccessfully. The user must then test for space file errors by checking the I/O status block (see I/O status block error code in Appendix D).

Error Return

Control is returned to the error return address specified in the SPF$ macro with the error code in the A-register when any of the following errors are detected:

A-Register Contents (Decimal)	Error Condition
1	The generic device type specified in the device control block is not configured for this system.
3	The logical unit number specified in the device control block is not configured for this system.
6	The number of files to be spaced was zero.
7	The space file request was issued for a device other than magnetic tape.
8	The device has not been previously reserved under the user ID specified in the device control block.
12	A space file request was issued for a disabled device.
26	The specified device has a disk-resident driver and the device was not previously reserved by a RSV$ macro instruction.

Macro Action Details

The following checks are made to determine if the space file request is legal:

1. A test is made to determine if the generic device type and the logical unit number specified in the device control block have been configured into the system.
2. A test is made to determine if the specified device is enabled.

3. A test is made to determine if the specified device has been previously reserved by a RSV$ macro instruction.
4. A test is made to determine if the user ID specified in the device control block matches the ID for which the device was reserved.
5. A test is made to determine if the number of files to be spaced is ≠ 0.
6. A test is made to determine if the specified device is magnetic tape.

If the space file request is not legal, control is returned to the error return address specified in the SPF$ macro with the error code in the A-register. If the space file request is legal, the space file request is queued according to the priority level of the task which issued the SPF$ macro. If the specified device is currently busy processing another request, control is returned to the caller at the normal return. If the specified device is not currently busy, the space file request is initiated, and control is returned to the caller at the normal return. When the caller receives control at the normal return, a WIO$ macro must be issued immediately if the caller desires sequential I/O. If nonsequential I/O is desired, the caller continues processing to the point that the space file completion is required. At this point, a WIO$ macro must be issued. Upon completion of the space file, the I/O completion return address will be scheduled. When the caller receives control at the I/O completion return address, the caller must assume the responsibility of interrogating the I/O status block to determine if the space file request was successful.

Word 1 of the status block will be zero if no error occurred. (See Appendix D for the status information.)

Out-Line Parameter List

Word No.	Operation	Operand
0	DAC	dcb address
1	DAC	error return address
2	DAC	no. of files address

where:

Parameters in the out-line parameter list are the same as those described above for the in-line parameter list.

NOTE: If an out-line parameter list is used and it is desired to update the DCB I/O status block address or the I/O completion return address, the user must precede the SPF$ macro call with the instructions to store the new I/O status block address in the sixth word (word 5) of the DCB or the new I/O completion return address in the seventh word (word 6) of the DCB.

Examples

The following examples illustrate the use of SPF$ to forward space five files on a magnetic tape. The Device Control Block, DCNM, and the Reserve request, which must

be executed prior to the forward space, have been illustrated previously. The I/O status block address and the I/O completion return address are not referenced in the forward space call, so they remain as shown in DCNM.

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6 7	8 14 15	16 31	32
*			FORWARD SPACE 5 FILES
*			
	SPF$	DCNM,ERSP,P5	FORWARD SPACE
	WIO$		SEQUENTIAL, WAIT FOR SPACE FILE COMPLETION
RTAD	LDA	TBAD+1	FORWARD SPACE COMPLETED, FETCH STATUS
	CAS	EOF	DID AN ERROR OCCUR?
	JMP	ERR	YES, PROCESS ERROR
	SKP		NO ERROR, CONTINUE
	JMP	ERR	YES, PROCESS ERROR
*			ERROR RETURN-A-REGISTER CONTAINS THE ERROR CODE
*			
ERSP	---		ERROR, ILLEGAL PARAMETER IN THE SPF$ MACRO LIST OR
*			IN THE DCB, DCNM
EOF	OCT	10	I/O STATUS BLOCK INDICATOR FOR END OF FILE
P5	DEC	5	NUMBER OF FILES TO BE SPACED FORWARD

The following example is the same as the example above, except that the out-line parameter list is used.

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6 7	8 14 15	16 31	32
*			FORWARD SPACE 5 FILES
*			
	LDX	PL!1	POINTER TO PARAMETER LIST TO X
	SPF$	(X)	FORWARD SPACE FILES
	WIO$		SEQUENTIAL, WAIT FOR SPACE FILE COMPLETION
RTAD	LDA	TBAD+1	FORWARD SPACE COMPLETED, FETCH STATUS

```
      CAS   EOF          DID AN ERROR OCCUR?
      JMP   ERR          YES, PROCESS ERROR
      SKP                NO ERROR, CONTINUE
      JMP   ERR          YES, PROCESS ERROR
*
*                        ERROR RETURN-A-REGISTER CONTAINS THE ERROR CODE
*
ERSP  ---                ERROR, ILLEGAL PARAMETER IN THE SPF$ MACRO LIST
*                        OR IN THE DCB, DCNM
*                        PARAMETER LIST
*
PLII  DAC   *+1          POINTER TO PARAMETER LIST
      DAC   FCNM         DCB ADDRESS
      DAC   ERSP         ERROR RETURN ADDRESS
      DAC   PS           NUMBER OF FILES ADDRESS
*
EOF   OCT   10           I/O STATUS BLOCK INDICATOR FOR END OF FILE
PS    DEC   5            NUMBER OF FILES TO BE SPACED FORWARD
```

SPACE RECORD REQUEST MACRO (SPR$)

The SPR$ macro initiates the spacing of one or more records on a magnetic tape.

SPR$ Macro Action

SPR$ checks to determine if the specified device has been previously reserved by a RSV$ macro instruction; if the device has been reserved, the macro takes the error return. Otherwise, the macro checks the legality of the parameters specified in the SPR$ macro and the device control block and takes the error return if any one of the parameters is illegal. If legal, the macro queues the space record request according to the priority level of the task which issued the SPR$ macro, initiates the space record request if the specified device is not currently busy processing another request, and returns to the SPR$ caller at the normal return. At that time the WIO$ macro instruction must be issued.

Macro Format

Location	Operation	Operand
[symbol]	SPR$	dcb address, error return address, no. of records address, [i/o status block address], [i/o completion return address]

where:

symbol — Optional

Is the symbolic label of the macro instruction.

dcb address

Is the name of the device control block associated with the space record device.

error return address

Is the address to which control is returned if an error is found in the SPR$ macro call parameters, the specified device control block parameters, or if the specified device was not previously reserved by a RSV$ macro instruction.

NOTE: Control does not return to the error return address if an error occurs during the spacing of a record. A user must test for such an error by checking the I/O status block after control is returned to the I/O completion return address.

no. of records address

Is the address of a word containing the number of records to be spaced. A positive number indicates forward spacing, and a negative number indicates backward spacing.

i/o status block address — Optional

Is the address of an 8 word I/O status block used by the device driver to schedule the I/O completion return to the user and to store the status information as described in Appendix D. The I/O status block address will be in the X-register when control is returned to the user at the I/O completion return address. If this parameter is omitted, the I/O status block address specified in the device control block will be used. If this parameter is present, it will be used to replace the I/O status block address in the device control block and any subsequent I/O call using the same device control block will use the new I/O status block address.

NOTE: Updating of the I/O status block address in the DCB is performed by the execution of the in-line macro expansion and not by the macro action routine.

i/o completion return address — Optional

Is the address where control is to be returned after I/O completion.

If this parameter is omitted, the I/O completion return address specified in the device control block will be used. If this parameter is present, it will be used to replace the I/O completion return address in the device control block and any subsequent I/O call using the same device control block will use the new I/O completion return address.

NOTE: Updating of the I/O completion return address in the DCB is performed by the execution of the in-line macro expansion and not by the macro action routine.

Normal Return

Control is returned to the caller, at the instruction following the SPR$ macro, after the spacing of the record has been initiated. This may occur before the spacing is complete. The caller must wait for space record completion by issuing a WIO$ macro instruction.

I/O Completion Return

After a WIO$ macro instruction, control will be returned to the I/O completion return address whether the record(s) has (have) been spaced successfully or unsuccessfully. The user must then test for space record errors by checking the I/O status block (see I/O status block error code in Appendix D).

Error Return

Control is returned to the error return address specified in the SPR$ macro with the error code in the A-register when any of the following errors are detected:

A-register Contents (Decimal)	Error Condition
1	The generic device type specified in the device control block is not configured for this system.
3	The logical unit number specified in the device control block is not configured for this system.
6	The number of records to be spaced was zero.
7	The space record request was issued for a device other than magnetic tape.
8	The device has not been previously reserved under the user ID specified in the device control block.
12	A space record request was issued for a disabled device.
26	The specified device has a disk-resident driver and the device was not previously reserved by a RSV$ macro instruction.

Macro Action Details

The following checks are made to determine if the space record request is legal:

1. A test is made to determine if the generic device type and the logical unit number specified in the device control block have been configured into the system.
2. A test is made to determine if the specified device is enabled.
3. A test is made to determine if the specified device has been previously reserved by a RSV$ macro instruction.
4. A test is made to determine if the user ID specified in the device control block matches the ID for which the device was reserved.

5. A test is made to determine if the number of records to be spaced is ≠ 0.
6. A test is made to determine if the specified device is magnetic tape.

If the space record request is not legal, control is returned to the error return address specified in the SPR$ macro with the error code in the A-register. If the space record request is legal, the space record request is queued according to the priority level of the task which issued the SPR$ macro. If the specified device is currently busy processing another request, control is returned to the caller at the normal return. If the specified device is not currently busy, the space record request is initiated, and control is returned to the caller at the normal return. When the caller receives control at the normal return, a WIO$ macro must be issued immediately if the caller desires sequential I/O. If non-sequential I/O is desired, the caller continues processing to the point that the space record completion is required. At this point, a WIO$ macro must be issued. Upon completion of the space record, the I/O completion return address will be scheduled. When the caller receives control at the I/O completion return address, the caller must assume the responsibility of interrogating the I/O status block to determine if the space record request was successful.

Word 1 of the status block will be 0 if no error occurred. (See Appendix D for the status information).

Out-Line Parameter List

Word No.	Operation	Operand
0	DAC	dcb address
1	DAC	error return address
2	DAC	no. of records address

where:

The parameters in the out-line parameter list are the same as those described above for the in-line parameter list.

NOTE: If an out-line parameter list is used and it is desired to update the DCB I/O status block address or the I/O completion return address, the user must precede the SPR$ macro call with the instructions to store the new I/O status block address in the sixth word (word 5) of the DCB or the new I/O completion return address in the seventh word (word 6) of the DCB.

Examples

The following examples illustrate the use of SPR$ to backward space two records on a magnetic tape. The Device Control Block, DCNM, and the Reserve Request, which must be executed prior to the backward space, have been illustrated previously. As in the examples under Space File, the I/O status block address and the I/O completion return address remain as shown in DCNM.

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6	7 8 14	16 31	32
*			BACKWARD SPACE 2 RECORDS
*			
	SPR$	DCNM,ERSR,M2	BACKWARD SPACE RECORDS
	WIO$		SEQUENTIAL, WAIT FOR SPACE RECORD COMPLETION
RTAD	LDA	TBAD+1	BACKWARD SPACE COMPLETED, FETCH STATUS
	SZE		DID AN ERROR OCCUR?
	JMP	ERR	YES, PROCESS ERROR
	---		NO ERROR, CONTINUE
*			
*			ERROR RETURN-A-REGISTER CONTAINS THE ERROR CODE
*			
ERSR	---		ERROR, ILLEGAL PARAMETER IN THE SPR$ MACRO LIST OR
			IN THE DCB, DCNM
M2	DEC	-2	NUMBER OF RECORDS TO BE SPACED BACKWARD

The following example is the same as the example above, except that the out-line parameter list is used.

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6	7 8 14	16 31	32
*			BACKWARD SPACE 2 RECORDS
*			
	LDX	PL12	POINTER TO PARAMETER LIST TO X
	SPR$	(X)	BACKWARD SPACE RECORDS
	WIO$		SEQUENTIAL, WAIT FOR SPACE RECORD COMPLETION
RTAD	LDA	TBAD+1	BACKWARD SPACE COMPLETED, FETCH STATUS
	SZE		DID AN ERROR OCCUR?
	JMP	ERR	YES, PROCESS ERROR
	---		NO ERROR, CONTINUE
*			
*			ERROR RETURN-A-REGISTER CONTAINS THE ERROR CODE
*			

```
ERSR    ---     ----------------    ERROR, ILLEGAL PARAMETER IN THE SPR$ MACRO LIST
*       ------  ----------------    OR IN THE DCB, DCNM
*       ------  ----------------    PARAMETER LIST
*       ------  ----------------
PL1R    DAC     *+1 ------------    POINTER TO PARAMETER LIST
        DAC     DCNM ------------   DCB ADDRESS
        DAC     ERSR ------------   ERROR RETURN ADDRESS
        DAC     M2 --------------   NUMBER OF RECORDS ADDRESS
M2      DEC     -2 --------------   NUMBER OF RECORDS TO BE SPACED BACKWARD
```

REWIND REQUEST MACRO (RWD$)

The RWD$ macro initiates the rewinding of a magnetic tape.

RWD$ Macro Action

RWD$ checks to determine if the specified device has been previously reserved by a RSV$ macro instruction; if it has not been reserved, the macro takes the error return. If it has been reserved, the macro checks the legality of the parameters specified in the device control block and takes the error return if any one of the parameters is illegal. If legal, the macro inserts the rewind request at the beginning of the queue for the requested device, initiates the rewind request if the specified device is not currently busy processing another request, and returns to the RWD$ caller at the normal return. At that time the WIO$ macro instruction must be issued.

Macro Format

Location	Operation	Operand
[symbol]	RWD$	dcb address, error return address, [i/o status block address], [i/o completion return address]

where:

symbol — Optional

Is the symbolic label of the macro instruction.

dcb address

Is the name of the device control block associated with the rewind device.

error return address

Is the address to which control is returned if an error is found in the specified device control block parameters, or if the specified device was not previously reserved by a RSV$ macro instruction.

NOTE: Control does not return to the error return address if an error occurs during the rewinding of a magnetic tape. An error during the rewind must be tested for by checking the I/O status block after control is returned to the I/O completion return address.

i/o status block address — Optional

Is the address of an 8 word I/O status block used by the device driver to schedule the I/O completion return to the user and to store the status information as described in Appendix D. The I/O status block address will be in the X-register when control is returned to the user at the I/O completion return address. If this parameter is omitted, the I/O status block address specified in the device control block will be used. If this parameter is present, it will be used to replace the I/O status block address in the device control block and any subsequent I/O call using the same device control block will use the new I/O status block address.

NOTE: Updating of the I/O status block address in the DCB is performed by the execution of the in-line macro expansion and not by the macro action routine.

i/o completion return address — Optional

Is the address where control is to be returned after I/O completion.

If this parameter is omitted, the I/O completion return address specified in the device control block will be used. If this parameter is present, it will be used to replace the I/O completion return address in the device control block and any subsequent I/O call using the same device control block will use the new I/O completion return address.

NOTE: Updating of the I/O completion return address in the DCB is performed by the execution of the in-line macro expansion and not by the macro action routine.

Normal Returns

Control is returned to the caller, at the instruction following the RWD$ macro, after the rewind has been initiated. This may occur before the rewind is complete. The caller must wait for rewind completion by issuing a WIO$ macro instruction.

I/O Completion Return

After a WIO$ macro instruction, control will be returned to the I/O completion return address whether the magnetic tape has been rewound successfully or unsuccessfully. The user must then test for errors which may have occurred during the rewind by checking the I/O status block (see I/O Status Block Error Code in Appendix D).

Error Return

Control is returned to the error return address specified in the RWD$ macro with the error code in the A-register when any of the following errors are detected:

A-Register Contents (Decimal)	Error Condition
1	The generic device type specified in the device control block is not configured for this system.
3	The logical unit number specified in the device control block is not configured for this system.
7	The rewind request was issued for a device other than magnetic tape.
8	The device has not been previously reserved under the user ID specified in the device control block.
12	A rewind request was issued for a disabled device.
26	The specified device has a disk-resident driver and the device was not previously reserved by a RSV$ macro instruction.

Macro Action Details

The following checks are made to determine if the rewind request is legal:

1. A test is made to determine if the generic device type and the logical unit number specified in the device control block have been configured into the system.
2. A test is made to determine if the specified device is enabled.
3. A test is made to determine if the specified device has been previously reserved by a RSV$ macro instruction.
4. A test is made to determine if the user ID specified in the device control block matches the ID for which the device was reserved.
5. A test is made to determine if the specified device is magnetic tape.

If the rewind request is not legal, control is returned to the error return address specified in the RWD$ macro with the error code in the A-register. If the rewind request is legal, the rewind request is placed at the beginning of the queue for the specified device. If the requested device is currently busy processing another request, control is returned to the caller at the normal return. If the specified device is not currently busy, the rewind request is initiated, and control is returned to the caller at the normal return. When the caller receives control at the normal return, a WIO$ macro must be issued immediately if the caller desires sequential I/O. If non-sequential I/O is desired, the caller continues processing to the point that the rewind completion is required. At this point, a WIO$ macro must be issued. Upon completion of the rewind, the I/O completion return address will be scheduled. When the caller receives control at the I/O completion return address, the caller must assume the responsibility of interrogating the I/O status block to determine if the rewind request was successful. Word 1 of the status block will be zero if no error occurred. (See Appendix D for the status information). Note that before the rewind request is issued, all I/O requests for the device specified in the device control block of the rewind request must be completed.

Out-Line Parameter List

Word No.	Operation	Operand
0	DAC	dcb address
1	DAC	error return address

where:

The parameters in the out-line parameter list are the same as those described above for the in-line parameter list.

NOTE: If an out-line parameter list is used and it is desired to update the DCB I/O status block address or the I/O completion return address, the user must precede the RWD$ macro call with the instructions to store the new I/O status block address in the sixth word (word 5) of the DCB or the new I/O completion return address in the seventh word (word 6) of the DCB.

Examples

The following examples illustrate the use of RWD$ to rewind a magnetic tape. The Device Control Block, DCNM, and the Reserve Request, which must be executed prior to the rewind, have been shown previously.

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6	7 8 14	16 31	32
*			REWIND MAGNETIC TAPE
*			
	RWD$	DCNM,ERRR	REWIND
	WIO$		SEQUENTIAL, WAIT FOR REWIND COMPLETION
RTAD	LDA	TBAD+1	REWIND COMPLETION, FETCH STATUS
	SUB	BOT	CHECK FOR BEGINNING OF TAPE
	SZE		DID AN ERROR OCCUR?
	JMP	ERR	YES, PROCESS ERROR
	---		NO ERROR, CONTINUE
*			
*			
*			ERROR RETURN- A-REGISTER CONTAINS THE ERROR CODE
ERRR	---		ERROR, ILLEGAL PARAMETER IN THE DCB, DCNM
BOT	OCT	2	I/O STATUS BLOCK INDICATOR FOR BEGINNING OF TAPE

The following example is the same as the example above, except that the out-line parameter list is used.

```
LOCATION  OPERATION  OPERAND FIELD          COMMENTS
1      6  7 8    14 16              31 32
*                                        REWIND MAGNETIC TAPE
*
          LDX       PLIO                 PARAMETER LIST POINTER TO X
          RWD$      (X)                  REWIND
          WIO,$                          SEQUENTIAL, WAIT FOR REWIND COMPLETION
RTAP      LDA       TBAD+1               REWIND COMPLETION, FETCH STATUS
          SUB       BOT                  CHECK FOR BEGINNING OF TAPE
          SZE                            DID AN ERROR OCCUR?
          JMP       ERR                  YES, PROCESS ERROR
          ---                            NO ERROR, CONTINUE
*
*                                        ERROR RETURN- A-REGISTER CONTAINS THE ERROR CODE
*
ERRR      ---                            ERROR, ILLEGAL PARAMETER IN THE DCB, DCNM
*
*                                        PARAMETER LIST
*
PLIO      DAC       *+1                  POINTER TO PARAMETER LIST
          DAC       DCNM                 DCB ADDRESS
          DAC       ERRR                 ERROR RETURN ADDRESS
BOT       OCT       2                    I/O STATUS BLOCK INDICATOR FOR BEGINNING OF TAPE
```

UNLOAD REQUEST MACRO (ULD$)

The ULD$ macro initiates the rewinding of a magnetic tape, and, upon the completion of the rewind, releases the requested device without the caller issuing a REL$ macro.

ULD$ Macro Action

ULD$ checks to determine if the specified device has been previously reserved by a RSV$ macro instruction; if it has not been reserved, the macro takes the error return. If it has, the macro checks the legality of the parameters specified in the device control block and takes the error return if any one of the parameters is illegal. If all are legal, the macro inserts the unload request at the beginning of the queue for the requested device, initiates the unload request if the device is not currently busy processing another request, and returns to the ULD$ caller at the normal return.

Macro Format

Location	Operation	Operand
[symbol]	ULD$	dcb address, error return address, [i/c status block address]

where:

symbol — Optional

Is the symbolic label of the macro instruction.

dcb address

Is the name of the device control block associated with the unload device.

error return address

Is the address to which control is returned if an error is found in the specified device control block parameters, or if the specified device was not previously reserved by a RSV$ macro instruction.

i/o status block address — Optional

Is the address of an 8 word I/O status block used to insert the unload request in the queue for the specified device.

If this parameter is omitted, the I/O status block address specified in the device control block will be used. If this parameter is present, it will be used to replace the I/O status block address in the device control block and any subsequent I/O call using the same device control block will use the new I/O status block address.

NOTE: Updating of the I/O status block address in the DCB is performed by the execution of the in-line macro expansion and not by the macro action routine.

Normal Returns

Control is returned to the caller, at the instruction following the ULD$ macro, after the unload request has been initiated. This may occur before the unload is complete.

I/O Completion Return

There is no I/O completion return for unload requests.

Error Return

Control is returned to the error return address specified in the ULD$ macro with the error code in the A-register when any of the following errors are detected:

A-Register Contents (Decimal)	Error Condition
1	The generic device type specified in the device control block is not configured for this system.
3	The logical unit number specified in the device control block is not configured for this system.
7	The unload request was issued for a device other than magnetic tape.
8	The device has not been previously reserved under the user ID specified in the device control block.
12	An unload request was issued for a disabled device.
26	The specified device has a disk-resident driver and the device was not previously reserved by a RSV$ macro instruction.

Macro Action Details

The following checks are made to determine if the unload request is legal:

1. A test is made to determine if the generic device type and the logical unit number specified in the device control block have been configured into the system.
2. A test is made to determine if the specified device is enabled.
3. A test is made to determine if the specified device has been previously reserved by a RSV$ macro instruction.
4. A test is made to determine if the user ID specified in the device control block matches the ID for which the device was reserved.
5. A test is made to determine if the specified device is magnetic tape.

If the unload request is not legal, control is returned to the error return address specified in the ULD$ macro with the error code in the A-register. If the unload request is legal, the unload request is placed at the beginning of the queue for the requested device. If the requested device is currently busy processing another request, control is returned to the caller at the normal return. If the requested device is not currently busy, the unload request is initiated, and control is returned to the caller at the normal return. When the caller receives control at the normal return, the task or activity may be terminated.

NOTE: Before the unload request is issued, all I/O requests for the device specified in the device control block of the unload request must be completed. Also, after the unload request is issued, no other I/O requests for the device specified in the device control block of the unload request may be issued until the device is again reserved by issuing a RSV$ macro.

Out-Line Parameter List

Word No.	Operation	Operand
0	DAC	dcb address
1	DAC	error return address

wnere:

Parameters in the out-line parameter list are the same as those described above for the in-line parameter list.

NOTE: If an out-line parameter list is used and it is desired to update the DCB I/O status block address, the user must precede the ULD$ macro call with the instructions to store the new I/O status block address in the sixth word (word 5) of the DCB.

Examples

The following examples illustrate the use of ULD$ to rewind a magnetic tape and automatically release the device specified in the DCB. This allows the user to terminate without waiting for the magnetic tape to finish rewinding. The Device Control Block is DCNM, which is shown in a previous example, and the Reserve Request, which must be executed prior to the unload, is also shown in a previous example.

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6 7	8 14 15	16 31	32
*			UNLOAD - REWIND MAGNETIC TAPE WITH AUTOMATIC
*			RELEASE
	ULD$	DCNM,ERUL	REWIND AND RELEASE THE DEVICE
	---		RETURN BLOCKS TO FREE CORE STORAGE AND TERMINATE
*			
*			ERROR RETURN - A-REGISTER CONTAINS THE ERROR CODE
*			
ERUL			ERROR, ILLEGAL PARAMETER IN THE DCB, DCNM

The following example is the same as the example above, except that the out-line parameter list is used.

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6	7 8 14	16 31	32
*			UNLOAD-REWIND MAGNETIC TAPE WITH AUTOMATIC RELEASE
*			
	LDX	PL20	PARAMETER LIST POINTER TO X
	ULD$	(X)	REWIND AND RELEASE THE DEVICE
	---		TERMINATE
*			
*			ERROR RETURN-A-REGISTER CONTAINS THE ERROR CODE
*			
ERUL	---		ERRORS, ILLEGAL PARAMETER IN THE DCB, DCNM
*			
*			PARAMETER LIST
*			
PL20	DAC	*+1	POINTER TO PARAMETER LIST
	DAC	DCNM	DCB ADDRESS
	DAC	ERUL	ERROR RETURN ADDRESS

WAIT I/O REQUEST MACRO (WIO$)

The WIO$ macro causes a wait state to exist until some remote action takes place.

WIO$ Macro Action

WIO$ checks to determine if the I/O status block address pointer has been specified in the WIO$ macro, and if it has not, returns to the Executive. If it has, the macro saves the user pseudo registers ZCR1, ZCR2, and ZCR3, the user's state (the keys, the B and S hardware registers), and returns to the Executive.

Before returning to the user at the I/O completion return address or the queued RSV$ request return address, the macro restores the user's pseudo registers ZCR1, ZCR2, and ZCR3, and the user's state (the keys, the B and S hardware registers).

Macro Format

Location	Operation	Operand
[symbol]	WIO$	[i/o status block address pointer]

where:

symbol — Optional

Is the symbolic label of the macro instruction.

i/o status block address pointer — Optional

Is the address of a word which contains the address of the I/O status block. The I/O status block must be the I/O status block which was specified in the device control block for an Input (INP$), Output (OTP$), Space File (SPF$), Space Record (SPR$), End-Of-File (EOF$), Rewind (RWD$), or Reserve (RSV$) request for which the wait I/O request is being issued. If the I/O status block address pointer is specified in the WIO$ macro, the keys, the B and S hardware registers, and the pseudo-registers ZCR1, ZCR2, and ZCR3 will be saved and then restored before returning to the user at the I/O completion return address or the queued RSV$ request return address. If the I/O status block address pointer is not specified in the WIO$ macro, the registers specified above are not saved and restored. If it is essential that the above specified registers be saved and restored, only one I/O request may be issued prior to the WIO$ macro.

NOTE: If the users program is re-entrant and it is not desirable to specify the I/O status block address pointer in the WIO$ macro, the I/O status block may be a block which is larger than 8 words in length, and the area following or prior to the 8 words required for the I/O status block may be used to contain pointers to other blocks fetched from free memory.

Normal Return

There is no return to the user from the WIO$ macro. Control is returned to the I/O completion return address or queued RSV$ request return address. There is no error return for this macro.

Macro Action Details

A check is made to determine if the I/O status block address pointer has been specified (is not = 0) in the WIO$ macro; if it has not, return is made to the Executive. If the I/O status block address pointer is specified in the WIO$ macro, the pseudo-registers ZCR1, ZCR2, and ZCR3, the user's keys, and the user's B and S hardware registers are saved, and return is made to the Executive.

Before returning to the user at the I/O completion return address or the queued RSV$ request return address, the pseudo-registers ZCR1, ZCR2, and ZCR3, the user's keys, and the user's B and S hardware registers are restored.

Out-Line Parameter List

Word No.	Operation	Operand
0	DAC or	[I/O status block address pointer]
0	BSZ	1

where:

The parameters in the out-line parameter list are the same as those described above for the in-line parameter list. Note that if it is not desirable to have the pseudo-registers and the user's state saved and restored, word 0 must contain a BSZ 1.

Examples

See the examples of the WIO$ macro which are included in the examples of reserve (RSV$), input (INP$), and output (OTP$).

DISK MACROS

Disk macros are used to allocate and deallocate work areas on disk for use with physical I/O INP$ and OTP$ macros.

The disk macros in this section are the Allocate Disk Work Area (ALC$) and Deallocate Disk Work Area (DLC$) macros.

Before physical I/O requests are made to the system disk a work area on the system disk must be allocated by issuing the ALC$ macro. A work area is a configurable number of tracks on the disk. The number of tracks in a work area is determined at system configuration time. The user must issue an allocate request for each work area that is needed. When the work areas that were allocated are no longer needed, each work area must be deallocated by issuing the DLC$ macro.

ALLOCATE WORK AREA (ALC$)

The ALC$ macro is used to allocate a work area on the disk.

ALC$ Macro Action

ALC$ allocates a work area and returns to the caller with the starting segment number of the allocated work area.

Macro Format

Location	Operation	Operand
[symbol]	ALC$	[generic device type], [logical unit no.], starting segment no. address, no. of work areas, error return address

where:

symbol — Optional

Is the symbolic label of the macro instruction.

generic device type — Optional

Is a value specifying the generic device type of the disk (moving head or fixed head) on which the work area is requested. If this parameter is omitted, the generic device type of the system disk is used.

logical unit no. — Optional

Is a value specifying the logical unit number of the disk. If this parameter is omitted, the logical unit number of the system disk is used.

starting segment no. address

Is the address of a word which will contain the starting segment number of the work area after it is allocated.

no. of work areas

Is a value indicating the number of work areas to be allocated. OS/700 Executive currently allocates only one work area per request.

error return address

Is the address to which control is returned if an error is found during the processing of the macro call.

Normal Return

Control is transferred to the instruction following the ALC$ macro after a work area is successfully allocated. Upon a normal return the segment number of the first segment in the work is contained in the word specified by the starting segment number address.

Error Return

Control is returned to the error return address specified in the ALC$ macro with the error code in the A-register when the following error is detected:

A-Register Contents (Decimal)	Error Condition
17	No work areas available for allocation

Macro Action Details

OS/700 Executive maintains a table of all the work areas configured for every disk in the system. As the work areas are allocated, the table is updated to reflect the availability of various work areas. Upon a ALC$ macro call, a test is made to see if a work area is available for allocation. If there is, its starting segment number is placed in the starting segment number address parameter specified in the macro call. The normal return is taken to the caller.

Out-Line Parameter List

Word No.	Operation	Operand
0	DAC	[generic device type]
1	DAC	[logical unit no.]
2	DAC	starting segment no. address
3	DAC	no. of work areas
4	DAC	error return address

where:

Parameters in the out-line parameter list are the same as those described above for the in-line parameter list.

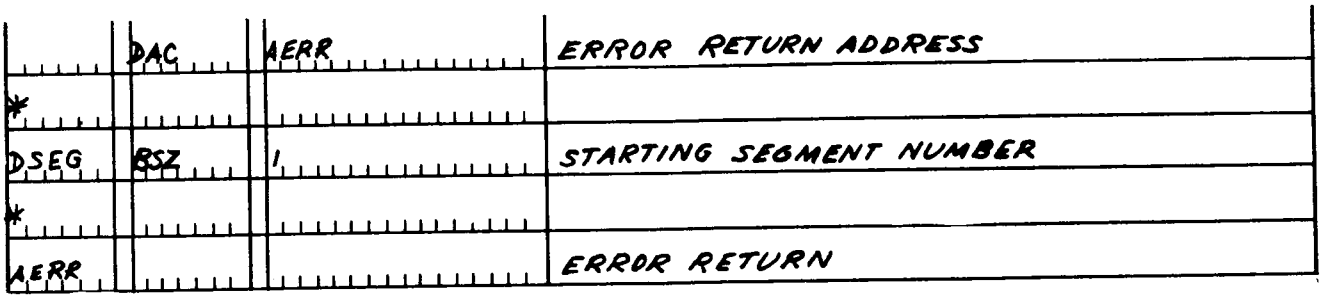
Examples

It is desired to allocate a work area on the system disk. The first segment number of this work area is to be stored in word DSEG.

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6	7 8 14	15 16 31	32
	ALC$	,,DSEG,1,AERR	ALLOCATE A WORK AREA ON THE SYSTEM DISK
	:		
	:		
DSEG	BSZ		STARTING SEGMENT NUMBER
*			
AERR	---		ERROR RETURN

If an out-line parameter list were used in the first example, it would be written as:

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6	7 8 14	15 16 31	32
	LDX	PLIST	PARAMETER LIST POINTER TO X
	ALC$	(X)	ALLOCATE A WORK AREA ON THE SYSTEM DISK
	:		
	:		
PLIST	DAC	*+1	POINTER TO PARAMETER LIST
	DAC	0	DEFAULT TO GENERIC DEVICE TYPE FOR SYSTEM DISK
	DAC	0	DEFAULT TO LOGICAL UNIT NO. FOR SYSTEM DISK
	DAC	DSEG	STARTING SEGMENT NUMBER ADDRESS
	DAC	1	NO. OF WORK AREAS



DEALLOCATE WORK AREA (DLC$)

The DLC$ macro is used to deallocate a work area on the disk.

DLC$ Macro Action

DLC$ deallocates a work area whose starting segment number is specified and returns to the caller.

Macro Format

Location	Operation	Operand
[symbol]	DLC$	[generic device type], [logical unit no.], starting segment no. address, no. of work areas, error return address

where:

symbol — Optional

Is the symbolic label of the macro instruction.

generic device type — Optional

Is a value specifying the generic device type of the disk (Removable or Fixed Head) on which the work area is requested. If this parameter is omitted, the generic device type of the system disk is used.

logical unit number — Optional

Is a value specifying the logical unit number of the disk. If this parameter is omitted, the logical unit number of the system disk is used.

starting segment no. address

Is the address of a word containing the starting segment number of the work area being deallocated.

no. of work areas

Is a value indicating the number of work areas to be deallocated. OS/700 currently deallocates only one work area per request.

error return address

Is the address to which control is returned if an error is found during the processing of the macro call.

Normal Return

Control is transferred to the instruction following the DLC$ macro after a work area is successfully deallocated.

Error Return

Control is returned to the error return address specified in the DLC$ macro with the error code in the A-register when the following error is detected:

A-Register Contents (Decimal)	Error Condition
17	The segment number of the work area being deallocated is not the first segment of the work area.

Macro Action Details

A test is made to determine if the starting segment number of the work area is a valid number (on a work area boundary) and if so the work area allocation table is modified to reflect the deallocation of the specified work area. A normal return is then taken to the caller.

If the test indicates that the starting segment number of the work area is not a valid number, control is transferred to the error return address with 17 in the A-register.

Out-Line Parameter List

Word No.	Operation	Operand
0	DAC	[generic device type]
1	DAC	[logical unit number]
2	DAC	starting segment no. address
3	DAC	no. of work areas
4	DAC	error return address

where:

Parameters in the out-line parameter list are the same as those described above for the in-line parameter list.

Examples

It is desired to deallocate a work area on the system disk. The first segment number of this work area is stored in word DSEG, and is the value 300.

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6	7 8 14	15 16 31	32
	DLC$	,,DSEG,1,DERR	DEALLOCATE A WORK AREA ON THE SYSTEM DISK
	:		
	:		
DSEG	DEC	300	STARTING SEGMENT NO.
*			
DERR	---		ERROR RETURN

If an out-line parameter list were used in the first example, it would be written as:

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6	7 8 14	15 16 31	32
	LDX	PLIST	PARAMETER LIST POINTER TO X
	DLC$	(X)	DEALLOCATE A WORK AREA ON THE SYSTEM DISK
	:		
	:		
PLIST	DAC	*+1	POINTER TO PARAMETER LIST
	DAC	0	DEFAULT TO GENERIC DEVICE TYPE FOR SYSTEM DISK
	DAC	0	DEFAULT TO LOGICAL UNIT NO. FOR SYSTEM DISK
	DAC	DSEG	STARTING SEGMENT NUMBER ADDRESS
	DAC	1	NUMBER OF WORK AREAS
	DAC	DERR	ERROR RETURN ADDRESS
*			
DSEG	DEC	300	STARTING SEGMENT NO.
*			
DERR	---		ERROR RETURN

SECTION VI
OPERATOR INTERFACE OPERATIONS

This chapter describes the system macros used to interface activities and tasks with the operator.

An activity may interface with the operator console via the operator interface processor. An activity may type a message on the operator console (KSR or ASR) via a type message (TYP$) macro or it may type a message and request a response from the operator via a type message and input a response (TPR$) macro.

The format of the message output on the operator console is as follows:

(TYP$ macro) ACTVTY MESSAGE
(TPR$ macro) MN ACTVTY MESSAGE

where

ACTIVITY denotes the six character name of the associated activity

MN denotes a 2 character message number

The blanks, the activity name, and the message number are inserted by the OS/700 Executive. The message number is used to associate a response to a particular message.

Where the TYP$ or the TPR$ macro are utilized and the message cannot be typed because message table is full or because an OS/700 utility has control of the operator console, the user may try again at a later time. While using TYP$ or TPR$ macro, the user must be aware that control is not transferred to him until the whole message is output (in case of TYP$ macro) or the response received (in case of TPR$ macro). While specifying the input range for the response, the range may not be larger than 55 words (110 characters) including a terminating carriage return.

TYPE A MESSAGE (TYP$)

The TYP$ macro is used to type a message on the operator console.

TYP$ Macro Action

TYP$ types the message on the operator console, and returns to the caller after the message is completely typed.

Macro Format

Location	Operation	Operand
[symbol]	TYP$	buffer address, range, error return address

where:

symbol – Optional

Is the symbolic label of the macro instruction.

buffer address

Is the address of the first word of the buffer containing the message to be typed. The first seven words of the buffer must not be used for storing the message. These words must be left for system use and the message to be printed should start on the eighth word. The message may consist of ASCII characters only.

range

Is the number of words in the buffer. This range must include the seven words reserved in the buffer for system use.

error return address

Is the address to which control is returned if an error is found during the processing of the macro call.

Normal Return

Control is returned to the caller, at the instruction following the macro, after the message is output, provided no errors were detected during the processing of the TYP$ macro.

Error Return

Control is returned to the error return address specified in the TYP$ macro with the error code in the A-register when any of the following errors are detected:

A-register Contents (Decimal)	Error Condition
2	An OS/700 utility has control of the operator console and no messages can be typed on the operator console.
4	An error was encountered before or during the message output.

Macro Action Details

A test is made to determine if an OS/700 utility has control of the OIP console and if so, control is transferred to the error return address with 2_{10} in the A-register. Otherwise, the name of the activity making the request is inserted in front of the message and a request is made on the I/O scheduler to output the specified number of words on the operator console.

After I/O completion, the I/O status block is checked for any errors and if any, control is transferred to the error return address with a 4_{10} in the A-register. If no errors are indicated in the I/O status block, control is transferred to the normal return (instruction following the macro call).

Out-Line Parameter List

WORD NO.	OPERATION	OPERAND
0	DAC	buffer address
1	DAC	range
2	DAC	error return address

where:

Parameters in the out-line parameter list are the same as those described above for the in-line parameter list.

Examples

It is desired to print out on the operator console a message consisting of five words. The seven words reserved for the TYP$ action makes a total range of 12 words. The 12 words begin at word OBUF.

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6 7	8 14 15	16 31	32
	TYP$	OBUF,12,ERTYP	
	:		
	:		
OBUF	BSZ	7	7 WORDS FOR TYP$ ACTION
	BCI	5,	MESSAGE
*			
ERTYP	---		ERROR RETURN

Assume the conditions of the first example exist, except that the address of the message buffer is known only at execution time. Prior to the call of TYP$, the address of the message buffer is stored in word T1. The following would be written:

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6 7	8 14 15	16 31	32
	STA	T1	STORE BUFFER ADDRESS
	:		
	TYP$	!T1,,12,ERTYP	
	:		
	:		
T1	BSZ	1	T1 CONTAINS THE DEFINED PARAMETER
*			
*			
ERTYP	---		ERROR RETURN

If an out-line parameter list is used for the conditions of the second example, the following would be written:

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6 7	8 14 15	16 31	32
	STA	T1	STORE BUFFER ADDRESS
	:		
	LDX	PLIST	
	TYP$	(X)	
	:		
	:		
PLIST	DAC	*+1	
T1	BSZ	1	BUFFER ADDRESS
	DAC	12	
	DAC	ERTYP	
*			
ERTYP	---		ERROR RETURN

TYPE A MESSAGE AND INPUT A RESPONSE (TPR$)

TPR$ Macro Action

TPR$ types the message, specified in a macro parameter, on the operator console, and waits for the operator to input the response. The response is passed to the user in a buffer.

Macro Format

Location	Operation	Operand
[symbol]	TPR$	output buffer address, output range, error return address, input buffer address, input range.

where:

symbol - Optional

Is the symbolic label of the macro instruction.

output buffer address

Is the address of the buffer containing the message to be typed. The first seven words of the buffer must not be used for storing the message. These words must be left for system use and the message to be printed starts on the eighth word. The message may contain ASCII characters only.

output range

Is the number of words contained in the message. The output range must include the seven words reserved in the output buffer for system use.

error return address

Is the address to which control is returned if an error is found during the processing of the macro call.

input buffer address

Is the address of the first word of the buffer in which the response is stored.

input range

Is the number of words expected in the response message including the terminating carriage return character.

Normal Return

Control is returned to the caller, at the instruction following the macro, after the message output and response input, provided no errors were detected during the processing of the TPR$ macro.

Error Return

Control is returned to the error return address specified in the TPR$ macro with the error code in the A-register when any of the following errors are detected:

A-register Contents (Decimal)	Error Condition
2	An OS/700 utility has control of the operator console and no messages may be typed on the operator console.
3	The message table (containing the identity of users waiting for response to their messages) is full and no more TPR$ requests can be acknowledged at this time.
4	An error was encountered before or during the message output.

Macro Action Details

A test is made to determine if an OS/700 utility has control of the operator console and if so, return is made to the error return address with 2_{10} in the A-register. Another test is made to see if the message table is full and if so, control is transferred to the error return address with 3_{10} in the A-register. If there are no errors, a message number and the activity name are inserted in front of the message and a request made on the I/O scheduler to output the requested message. An entry is made in the message table to indicate an activity waiting for a response.

Upon the I/O completion return, a check is made to see if there were any errors and if so, control is transferred to the error return address with 4_{10} in the A-register. If there are no errors and after a response is keyed in by the operator, a check is made for a valid message number and control is transferred to the normal return of the associated activity. If an invalid message number or an illegal number of characters are keyed by the operator, a message is typed on the operator console indicating the error and allowing the operator to key in the proper response.

On-Line Parameter List

WORD NO.	OPERATION	OPERAND
0	DAC	output buffer address
1	DAC	output range
2	DAC	error return address
3	DAC	input buffer address
4	DAC	input range

where:

Parameters in the out-line parameter list are the same as those described above for the in-line parameter list.

Examples

A 16-word message is to be printed on the operator console, and the operator must type in a response of not more than 63 characters plus a carriage return. The output message is contained in OBUF, and the operator response will be collected in IBUF.

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6	7 8 14	15 16 31	32
	TPR$	OBUF,16,ERTYP,IBUF,32	
	:		
	:		
OBUF	BSZ	7	RESERVED FOR TPR$ ACTION
	BCI	9,	MESSAGE
*			
IBUF	BSZ	32	
*			
ERTYP	---		ERROR RETURN

If an out-line parameter list is used, the first example would appear as:

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6	7 8 14	15 16 31	32
	LDX	PLIST	
	TPR$	(X)	
	:		
	:		
PLIST	DAC	*+1	
	DAC	OBUF	
	DAC	16	
	DAC	ERTYP	
	DAC	IBUF	
	DAC	32	
*			
OBUF	BSZ	7	
	BCI	9,	MESSAGE
*			
IBUF	BSZ	32	
*			
ERTYP	---		ERROR RETURN

SECTION VII
COMMUNICATION OPERATIONS

Communication macros are used to communicate with terminals that operate under System 700. The macros are:

1. Connect Station Macro (CCST$)
2. Disconnect Station Macro (CDST$)
3. Receive Macro (CREC$)
4. Receive and Reformat Macro (CRAR$)
5. Send Macro (CSND$)
6. Reformat and Send Macro (CRAS$)
7. Get Station Status Macro (CGSS$)
8. Change Station Status Macro (CCSS$)
9. Terminate Communication Task Macro (CTMC$)

CONNECT STATION MACRO (CCST$)

This macro allows a user to connect a communications task to a station.

CCST$ Macro Action

CCST$ checks for configured station number, for a station available for connection, performs the connection and returns to the caller.

Macro Format

Location	Operation	Operand
[symbol]	CCST$	station address pointer, error return address, routing tcb address pointer

where:

symbol — Optional

Is the symbolic label of the macro instruction station address pointer.

station address pointer

Is the address of a word that contains the station number.

error return address

Is the address to which control is returned if an error is found during the processing of the macro call.

routing tcb address pointer

Is the address of a word that contains the address of the routing TCB that marks the connection of the user's communications task with the specified station.

Normal Return

Control is returned to the caller, at the instruction following the CCST$ macro, after the user's communication task has been connected to the specified station.

Error Returns

Control is returned to the error return address specified in the CCST$ macro with the error code in the A-register when any of the following errors are detected:

A-register Contents (Decimal)	Error Condition
1	The station is not configured
2	The station is already connected to a different user's communications task
3	The station is already connected to the same task.

Macro Action

A test is made to see if the station is configured. If it is not, the A-register is set to 1_{10} and control is returned to the caller at the error return address. If the station is configured, a test is made to see if the station can be connected; if not, the A-register is set up appropriately and control is returned to the caller at the error return address.

If the station is available for connection, the connection is made and control is returned to the caller at the normal return.

Out-Line Parameter List

Word No.	Operation	Operand
0	DAC	station address pointer
1	DAC	error return address
2	DAC	routing tcb address pointer

where:

Parameters in the out-line parameter list are the same as those described above for the in-line parameter list.

Examples:

The example below illustrates the use of an in-line parameter list.

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6	7 8 14	16 31	32
	CCST$	S,ERAD,RTCA	
	----		NORMAL RETURN
ERAD			ERROR RETURN
	-		
	-		
	-		
			POINTERS
S	DEC	1	STATION 1
RTCA	DAC	RTCB	ROUTING TCB ADDRESS

The following example illustrates the use of an out-line parameter list.

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6	7 8 14	15 16 31	32
	LDX	PLT+1	
	CCST$	(X)	
PLT+1	DAC	*+1	POINTER TO STATION ADDRESS
	DAC	S	POINTER TO STATION ADDRESS
	DAC	ERAD	ERROR RETURN ADDRESS
	DAC	RTCA	ROUTING TCB ADDRESS
.			
.			
.			
ERAD	-		ERROR RETURN
.	-		
.	-		
S	DEC	1	STATION 1
RTCA	DAC	RTCB	ROUTING TCB ADDRESS

DISCONNECT STATION MACRO (CDST$)

This macro allows a user to disconnect a communications task from a station.

CDST$ Macro Action

CDST$ checks for configured station number and the legality of the disconnect request. If they both check out, the macro performs the disconnect and returns to the caller.

Macro Format

Location	Operation	Operand
[symbol]	CDST$	station address pointer, error return address, routing tcb address pointer

where:

symbol — Optional

Is the symbolic label of the macro instruction.

station address pointer

Is the address of a word that contains the station number.

error return address

Is the address to which control is returned if an error is found during the processing of the macro call.

routing tcb address pointer

Is the address of a word that contains the address of the Routing TCB that presently marks the connection of the user's communications task with the specified station.

Normal Return

Control is returned to the caller, at the instruction following the CDST$ macro, after the user's communications task has been disconnected from the specified station.

Error Return

Control is returned to the error return address specified in the CDST$ macro with the error code in the A-register when any of the following errors are detected:

A-register Contents (Decimal)	Error Condition
1	The station is not configured
2	The station is not connected to this communications task
3	Input pending

Macro Action

A test is made to see if the station is configured. If it is not, the A-register is set to 1_{10} and control is returned to the caller at the error return address. If the station is configured, a test is made to see if the station can be disconnected. If not, the A-register is set up appropriately and control is returned to the caller at the error return address.

If the disconnect request is allowable, the disconnect is made, and control is returned to the caller at the normal return.

Out-Line Parameter List

Word No.	Operation	Operand
0	DAC	station address pointer
1	DAC	error return address
2	DAC	routing tcb address pointer

where:

Parameter in the out-line parameter list are the same as those described above for the in-line parameter list.

Examples:

The example below illustrates the use of an in-line parameter list.

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6 7	8 14 15	16 31	32
	CDST$	S,ERAD,RTCA	
	-----		NORMAL RETURN
.			
.			
.			
ERAD	-		ERROR RETURN
	-		
	-		
.			
.			
.			
S	DEC	1	
RTCA	DAC	RTCB	

The following example illustrates the use of an out-line parameter list.

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6	7 8 14	16 31	32
	LDX	PLT1	
	CDST$	(X)	
	----		NORMAL RETURN
.			
.			
.			
ERAD	----		ERROR RETURN
.			
.			
PLT1	DAC	*+1	
	DAC	S	POINTER TO STATION ADDRESS
	DAC	RTCA	ROUTING TCB ADDRESS
.			
.			
S	DEC	1	STATION 1
RTCA	DAC	RTCB	ROUTING TCB ADDRESS

RECEIVE MACRO (CREC$)

This macro allows a user's communications task to receive information from a station in communication buffer format or receive output status concerning a station.

CREC$ Macro Action

CREC$ checks the legality of the connection. If legal, it determines whether input data or output status is next to be processed, and stores the appropriate information in the user specified locations. Then, the macro returns to the caller at the normal return, the error return, or the special return.

Macro Format

Location	Operation	Operand
[symbol]	CREC$	station address pointer, error return address, buffer address pointer, range address pointer, status address pointer, routing tcb address pointer, special return address

where:

symbol — Optional

Is the symbolic label of the macro instruction.

station address pointers

Is the address of a word where the system stores the station number.

error return address

Is the address to which control is returned if an error is found during the processing of the macro call.

buffer address pointer

Is the address of a word where the system stores the address of the first word of data input.

range address pointer

Is the address of a word where the system stores the number of bytes of data that have been input.

status address pointer

Is the address of a word where the system stores the status of the input or output data. The status that is returned to the user's communications task when input data is detected has the following format:

bit 1 = 1 forced termination of this message

bit 2 = 1 inter-character time-out has occurred

bit 3 = 1 missing SOM sequence

bit 4 = 1 missing EOA sequence

bit 5 = 1 duplicate SOM sequence detected

bit 6 = 1 duplicate EOM sequence detected

bit 7 = 1 characters are missing in this message unit

bit 8 = 1 overrun was detected

bit 9 = 1 parity errors in this message unit

bit 10= 1 a stuck tape condition was detected

bit 11-14 spare (not used set to zero)

bit 15 = 1 EOM sequence detected

bit 16 = 1 open character(s) in the message unit

The status that is returned to the user's communications task when output status is detected has the following format:

bit 1 = 0 message unit identified by the sequence number was successfully delivered to the specified station.

= 1 message unit identified by the sequence number could not be delivered to the specified station.

bits 2-13 Spare

bits 14-16 sequence number given by the user's communication task in a Send or Reformat and Send macro call

routing tcb address pointer

Is the address of a word that contains the address of the routing TCB that marks the connection of the user's communications task with the sending station.

special return address

Is the address to which control is returned if special conditions are found during the processing of the macro call.

Normal Return

Control is returned to the caller, at the instruction following the CREC$ macro, when input data has been detected in the processing of the macro call. Thus the status parameter reflects input status.

Error Return

Control is returned to the error return address specified in the CREC$ macro with the error code in the A-register when the following error is detected:

A-register Contents (Decimal)	Error Condition
1	The Routing TCB address does not match the Routing TCB that marks the connection of the station that is currently being processed.

Special Return

Control is returned to the special return address specified in the CREC$ macro with the error code in the A-register when any of the following special conditions are detected.

A-register Content (Decimal)	Special Condition
1	No more data or output status is currently available.
2	Output status has been stored in the status parameter.

Macro Action

A check is made to determine if the station to be processed next is connected to the user's communications task. If not, the A-register is set to 1_{10} and control is returned to the caller at the error return address. If there are no more entries to process, the A-register is set to 1_{10} and control is returned to the caller at the special return address. If the next entry is output status, the status and station number are stored, the A-register is set to 2_{10}, and control is returned to the caller at the special return address.

If the next entry is input data, the status buffer address, range and station number are stored. Control is then returned to the caller at the normal return.

Out-Line Parameter List

Word No.	Operation	Operand
0	DAC	station address pointer
1	DAC	error return address
2	DAC	buffer address pointer
3	DAC	range address pointer
4	DAC	status address pointer
5	DAC	routing tcb address pointer
6	DAC	special return address

where:

Parameters in the out-line parameter list are the same as those described above for the in-line parameter list.

Examples:

The example below illustrates the use of an in-line parameter list.

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6	7 8 14	16 31	32
	CREQ$	S,ERAD,B,R,ST,RT	CA,SRAD
	----		NORMAL RETURN
ERAD	----		ERROR RETURN

SRAD	----		SPECIAL RETURN

S	DAC	0	STATION STORED HERE
B	DAC	0	BUFFER START HERE
ST	DAC	0	STATUS STORED HERE
RTCA	DAC	RTCB	ROUTING TCB ADDRESS
R	DAC	0	RANGE STORED HERE

The following example illustrates the use of an out-line parameter list.

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6	7 8 14	15 16 31	32
	LDX	PLT1	
	CREC$	(X)	
	---		NORMAL RETURN
.			
.			
.			
ERAD	----		ERROR RETURN

.			
.			
.			
SRAD	----		SPECIAL RETURN

PLT1	DAC	*+1	
	DAC	S	STATION
	DAC	ERAD	ERROR RETURN ADDRESS
	DAC	B	BUFFER
	DAC	R	RANGE
	DAC	ST	STATUS
	DAC	RTCA	ROUTING TCB POINTER
	DAC	SRAD	SPECIAL RETURN

.			
S	DAC	..	STATION STORED HERE
B	DAC	..	BUFFER START HERE
ST	DAC	..	STATUS STORED HERE
RTCA	DAC	RTCB	ROUTING TCB ADDRESS
R	DAC	..	RANGE STORED HERE

RECEIVE AND REFORMAT MACRO (CRAR$)

This macro allows a user's communications task to receive information from a station in peripheral buffer format or receive output status concerning a station.

CRAR$ Macro Action

CRAR$ checks the legality of the connection. If legal, it determines whether input data or output status is next to be processed. If input, the macro checks to determine if the peripheral buffer is large enough to accommodate the input data. Whether input or output, the macro stores the appropriate information in the user specified locations, and returns to the caller at the normal return, the error return, or the special return.

Macro Format

Location	Operation	Operand
[symbol]	CRAR$	station address pointer, error return address, buffer address pointer, buffer size address pointer. status address pointer, routing tcb address pointer, special return address

where:

symbol — Optional

Is the symbolic label of the macro instruction.

station address pointer

Is the address of a word where the system stores the station number.

error return address

Is the address to which control is returned if an error is found during the processing of the macro call.

buffer address pointer

Is the address of a word that contains the address of a peripheral buffer where the system stores input data. The size in words of the peripheral buffer is assumed to be in the word preceding the address given for the buffer.

buffer size address pointer

Is the address of a word that contains the size of the buffer (in words).

status address pointer

Is the address of a 2 word area where the system stores the status of the input or the input or output data.

The status that is returned to the user's communications task when input data is detected has the following format:

word 1

bit 1 = 1 forced termination of this message

bit 2 = 1 inter-character time out has occurred

bit 3 = 1 missing SOM sequence

bit 4 = 1 missing EOA sequence

bit 5 = 1 duplicate SOM sequence detected

bit 6 = 1 duplicate EOM sequence detected

bit 7 = 1 characters are missing

bit 8 = 1 overrun detected

bit 9 = 1 parity errors detected

bit 10 = 1 stuck tape condition detected

bit 11 - 14 Spare (not used)

bit 15 = 1 EOM sequence detected

bit 16= 1 open characters detected

word 2 The number of words actually input.

The status that is returned to the user's communications task when output status is detected uses only 1 word and has the following format.

bit 1 = 0 the data identified by the sequence number was successfully delivered to the specified station.

bit 1 = 1 the data identified by the sequence number could not be delivered to the specified station.

bits 2-13 Spare

bits 14-16 sequence number given by the user's communications task in a CSND$ or CRAS$ macro call.

routing tcb address pointer

Is the address of a word that contains the address of the Routing TCB that marks the connection of the user's communications task with the sending station.

special return address

Is the address to which control is returned if special conditions are found during the processing of the macro call.

Normal Return

Control is returned to the caller, at the instruction following the CRAR$ macro, when input data has been detected in the processing of the macro call. Thus the status parameter reflects input status.

Error Return

Control is returned to the error return address specified in the CRAR$ macro with the error code in the A-register when the following error is detected:

A-register Contents (Decimal)	Error Condition
1	The Routing TCB address does not match the Routing TCB that marks the connection of the station that is currently being processed.

Special Return

Control is returned to the special return address specified in the CRAR$ macro with the error code in the A-register when any of the following special conditions are detected:

A-register Contents (Decimal)	Special Condition
1	No more data or output status is currently available.
2	Output status has been stored in the status parameter.
3	The peripheral buffer was not large enough to store the input data in. The X-register contains the difference (in words) between the input data size and the given size.

Macro Action

A check is made to determine if the station to be processed next is connected to the user's communications task. If not, the A-register is set to 1_{10} and control is returned to the caller at the error return address. If there are no more entries to process, the A-register is set to 1_{10} and control is returned to the caller at the special return address. If the next entry is output status, the status and station number are stored, the A-register is set to 2_{10}, and control is returned to the caller at the special return address.

If the next entry is input data, a check is made to see that the peripheral buffer is large enough to accommodate the input data. If not, the A-register is set to 3_{10} and the X-register is set to the difference in words between the input data size and the peripheral buffer size and control is returned to the caller at the special return address.

If the peripheral buffer is large enough, the range and status are stored, the input data is transferred to the peripheral buffer and control is returned to the caller at the normal return.

Out-Line Parameter List

Word No.	Operation	Operand
0	DAC	station address pointer
1	DAC	error return address
2	DAC	buffer address pointer
3	DAC	buffer size address pointer
4	DAC	status address pointer
5	DAC	routing tcb address pointer
6	DAC	special return address

where:

Parameters in the out-line parameter list are the same as those described above for the in-line parameter list.

Examples

The example below illustrates the use of an in-line parameter list.

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6	7 8 14	15 16 31	32
	CRAR$	S,ERAD,B,BSIZR,ST,RTCA,SRAD	
	----		NORMAL RETURN
.			
ERAD	----		ERROR RETURN

.			
ERAD			
.			
SRAD	----		SPECIAL RETURN

S	DAC	..	STATION STORED HERE
BSIZ	DEC	20	SIZE OF BUFFER
ST	BSIZ	2	STATUS STORED HERE
RTCA	DAC	RTCB	ROUTING TCB ADDRESS

```
B       DAC     *+1             POINTER TO BUFFER
                                
        BSZ     20              BUFFER
```

The following example illustrates the use of an out-line parameter list.

```
LOCATION  OPERATION  OPERAND FIELD        COMMENTS
1      6  7 8    14 15 16          31 32
          LDX       PLTI
          CRAR$     (X)
          ----                         NORMAL RETURN
.
ERAD      ----                         ERROR RETURN
          ----
.
.
SRAD      ----                         SPECIAL RETURN
          ----
PLTI      DAC       *+1
          DAC       S                  STATION POINTER
          DAC       ERAD               ERROR RETURN ADDRESS
          DAC       B                  BUFFER POINTER
          DAC       BSIZ               BUFFER SIZE
          DAC       ST                 STATUS POINTER
          DAC       RTCA               ROUTING TCB POINTER
          DAC       SRAD               SPECIAL RETURN ADDRESS
.
.
S         DAC       ..                 STATION STORED HERE
BSIZ      DEC       20                 BUFFER SIZE STORED HERE
```

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6	7 8 14	16 31	32
ST	BSZ	2	STATUS STORED HERE
RTCA	DAC	RTCB	ROUTING TCB ADDRESS
B	DAC	*+1	POINTER TO BUFFER
	BSZ	20	BUFFER

SEND MACRO (CSND$)

This macro allows a user's communications task to data in communications buffer format to a station.

CSND$ Macro Action

CSND$ checks for configured station number and connected station number. It then deposits the data on the appropriate output queue and returns to the caller.

Macro Format

Location	Operation	Operand
[symbol]	CSND$	station address pointer, error return address, buffer address pointer, send parameter address pointer, routing tcb address pointer

where:

symbol — Optional

Is the symbolic label of the macro instruction.

station address pointer

Is the address of a word that contains the station number to which the user's communications task wants to output a message unit.

error return address

Is the address to which control is returned if an error is found during the processing of the macro call.

buffer address pointer

Is the address of a word that contains the address of the first word of the message unit to output.

send parameter address pointer

Is the address of a word that contains the various options available to the user's communications task concerning output of data to a station. The parameter has the following format:

bit 1 = 1 Send ETX

= 0 Send ETB

bit 2 = 1 Return output status when the message unit has been successfully transmitted or not.

The status is returned in a Receive or a Receive and Reformat macro call.

bit 3 = 1 Requeue the message unit on a terminal error.

= 0 Release the message unit to free storage on a terminal error.

bit 4 = 1 The message unit is a test message. A test message is given top priority and is released if not successfully transmitted.

bit 5 - 11 Spare

bit 12 = 1 Physically disconnected the terminal after the message unit has been successfully transmitted. This bit is effective only if the terminal is attached to a switched line.

bit 13 = 1 De-select the terminal after the message unit has been successfully transmitted.

bit 14 - 16 Sequence number.

routing tcb address pointer

Is the address of a word that contains the address of the Routing TCB which marks the connection of the user's communications task with the specified station.

Normal Return

Control is returned to the caller, at the instruction following the CSND$ macro, when the message unit has been successfully delivered to the specified station's output queue.

Error Return

Control is returned to the error return address specified in the CSND$ macro with the error code in the A-register when any of the following errors are detected:

A-register Contents (Decimal)	Error Condition
1	The station is not configured
2	The station is not operational but the message is queued
3	The station is not connected to this communications task
4	The size of the message unit is too large. The X-register contains the difference in bytes
5	The format of the message unit is illegal.

Macro Action

A check is made to determine if the station specified is configured. If not, the A-register is set to 1_{10} and control is returned to the caller at the error return address. If the station is configured, a check is made to see if the station is connected to the specified user's communications task. If not, the A-register is set to 3_{10} and control is returned to the caller at the error return address. If the station is connected, a check is made to see if the message unit is too large. If it is too large the A-register is set to 4_{10}, the X-register is set to the difference (bytes), and control is returned to the caller at the error return address.

If the message unit size is legal, an attempt is made to deliver the message unit to the specified station. If the station is not operational, the A-register is set to 2_{10} and control is returned to the caller at the error return address. If the format of the message unit is illegal, the A-register is set to 5_{10}, and control is returned to the caller at the error return address.

If the station is operational, control is returned to the caller at the normal return.

Out-Line Parameter List

Word No.	Operation	Operand
0	DAC	station address pointer
1	DAC	error return address
2	DAC	buffer address pointer
3	DAC	send parameter address pointer
4	DAC	routing tcb address pointer

were:

Parameters in the out-line parameter list are the same as those described above for the in-line parameter list.

Examples

The example below illustrates the use of an in-line parameter list.

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6 7	8 14 15	16 31	32
	CSND,$	S,ERAD,B,SD,RTCA	
	----		NORMAL RETURN
.			
.			
ERAD	----		ERROR RETURN

S	DEC	1	STATION 1

```
B       DAC     BUFF            POINTER TO BUFFER
SD      OCT     140007          SEND ETX, RETURN STATUS SEQ. NO. 7
RTCA    DAC     RTCB            ROUTING TCB
```

The following example illustrates the use of an out-line parameter list.

```
LOCATION  OPERATION  OPERAND FIELD        COMMENTS
1      6  7 8    14  15 16          31  32
          LDX       PLT1
          CSND$     (X)
          ----                           NORMAL RETURN
.
ERAD      ----                           ERROR RETURN
          ----
PLT1      DAC       0+1                  STATION
          DAC       ERAD                 ERROR RETURN ADDRESS
          DAC       B                    BUFFER POINTER
          DAC       SD                   STATUS POINTER
          DAC       RTCA                 ROUTING TCB POINTER
.
.
.
S         DEC       1                    STATION 1
B         DAC       BUFF                 POINTER TO BUFFER
SD        OCT       140007               SEND ETX, RETURN STATUS SEQ. NO. 7
RTCA      DAC       RTCB                 ROUTING TCB
```

REFORMAT AND SEND MACRO (CRAS$)

This macro allows a user's communications task to send data to a station in peripheral buffer format.

CRAS$ Macro Action

CRAS$ checks for configured station number, a connected station number, and an oversize buffer. If there are no problems, it reformats the peripheral buffer, deposits the reformatted data on the appropriate output queue, and returns to the caller.

Macro Format

Location	Operation	Operand
[symbol]	CRAS$	station address pointer, error return address buffer address pointer, buffer size address pointer, send parameter address pointer, routing tcb address pointer

where:

symbol — Optional

Is the symbolic label of the macro instruction.

station address pointer

Is the address of a word that contains the station number to which the user's communications task wants to output a peripheral buffer.

error return address

Is the address to which control is returned if an error is found during the processing of the macro call.

buffer address pointer

Is the address of a word that contains the address of the first word of data in the peripheral buffer.

buffer size address pointer

Is the address of a word that contains the size of the peripheral buffer. The size given is assumed to be in words (not bytes).

send parameter address pointer

Is the address of a word that contains the various options available to the user's communications task concerning output of data to a station. The parameter has the following format:

bit 1 = 1 Send ETX

= 0 Send ETB

bit 2 = 1 Return output status when the message unit has been successfully transmitted or not, the status is returned in a Receive and Reformat macro call.

bit 3 = 1 Requeue the message unit on a terminal error.

= 0 Release the message unit to free storage on a terminal error.

bit 4 = 1 The message unit is a test message. A test message is given top priority and is released if not successfully transmitted.

bit 5 = 1 Queue the message unit even if the station is not operational.

bits 6-11 Spare

bit 12 = 1 Physically disconnected the terminal after the message unit has been successfully transmitted. This bit is effective only if the terminal is attached to a switched line.

bit 13 = 1 De-select the terminal after the message unit has been successfully transmitted.

bit 14-16 Sequence Number.

routing tcb address pointer

Is the address of a word that contains the address of the Routing TCB that marks the connection of the user's communications task with the specified station.

Normal Return

Control is returned to the caller, at the instruction following the CRAS$ macro, when the peripheral buffer has been successfully delivered to the specified station's output queue.

Error Return

Control is returned to the error return address specified in the CRA$ macro with the error code in the A-register when any of the following errors are detected:

A-register Contents (Decimal)	Error Condition
1	The station is not configured.
2	The station is not operational.
3	The station is not connected to this communications task.
4	The size of the peripheral buffer is too large. The X-register contains the difference in words.

Macro Action

A check is made to see if the station specified is configured. If not, the A-register is set to 1_{10} and control is returned to the caller at the error return address. If the station is configured, a check is made to see if the station is connected to the specified user's communications task. If not, the A-register is set to 3_{10} and control is returned to the caller at the error return address.

If the station is connected, a check is made to see if the peripheral buffer is too large. If it is too large, the A-register is set to 4_{10}, the X-register is set to the difference (words) and control is returned to the caller at the error return address. If the peripheral

buffer size is legal, the buffer is reformatted into communications buffer(s) and an attempt is made to deliver the communications buffer(s) to the specified station. If the station is not operational, the A-register is set to 2_{10} and control is returned to the caller at the error return address. If the station is operational, and control is returned to the caller at the normal return.

Out-Line Parameter List

Word No.	Operation	Operand
0	DAC	station address pointer
1	DAC	error return address
2	DAC	buffer address pointer
3	DAC	buffer size address pointer
4	DAC	send parameter address pointer
5	DAC	routing tcb address pointer

where:

Parameters in the out-line parameter list are the same as those described above for the in-line parameter list.

Examples

The example below illustrates the use of an in-line parameter list.

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6 7	8 14 15	16 31	32
	CRAS$	S,ERAD,B,BSIZ,SD,SRAD	
	----		NORMAL RETURN
.			
ERAD	----		ERROR RETURN
.			
.			
S	DEC	1	STATION 1
SD	OCT	140007	ETX, RETURN STATUS, SEQ. NO. 7
RTCA	DAC	RTCB	ADDRESS OF ROUTING TCB
B	DAC	*+2	POINTER TO BUFFER
BSIZ	DEC	3	COUNT IN WORDS
	-		WORD 0
	-		WORD 1
	-		WORD 2

The following example illustrates the use of an out-line parameter list.

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6 7	8 14 15	16 31	32
	LDX	PLT1	
	CRAS$	(X)	
	----		NORMAL RETURN
.			
ERAD	----		ERROR RETURN

PLT1	DAC	*+1	
	DAC	S	STATION
	DAC	ERAD	ERROR RETURN ADDRESS
	DAC	B	BUFFER
	DAC	BSIZ	BUFFER SIZE
	DAC	RTCA	ROUTING TCB
.			
.			
S	DEC	1	STATION 1
SD	UCT	140007	EXT, RETURN STATUS, SEQ. NO. 7
RTCA	DAC	RTCB	ROUTING TCB ADDRESS
B	DAC	*+2	POINTER TO BUFFER
BSIZ	DEC	3	COUNT IN WORDS
	-		WORD 0
	-		WORD 1
	-		WORD 2

GET STATION STATUS MACRO (CGSS$)

This macro allows a user's communications task to obtain the status of a device, line, or terminal associated with a station.

CGSS$ Macro Action

CGSS$ checks for configured station number, obtains the specified status, and returns to the caller.

Macro Format

Location	Operation	Operand
[symbol]	CGSS$	station address pointer, error return address, association parameter address pointer, status address pointer

where:

symbol — Optional

Is the symbolic label of the macro instruction.

station address pointer

Is the address of a word that contains the station number.

error return address

Is the address to which control is returned if an error is found during the processing of the macro call.

association parameter address pointer

Is the address of a word that contains the value of the association parameter. This parameter tells the system what kind of status is desired. The kinds of status that can be requested are:

0 - Terminal status is desired

1 - Line status is desired

2 - Device status is desired

status address pointer

Is the address of a word that the system stores the desired status in. The status returned depends upon the setting of the Association Parameter.

If terminal status is deisred

bits	1-13	not used
bit	14	terminal on output hold (=1)
bit	15	terminal on skip poll (=1)
bit	16	terminal not operational (=1)

If line status is desired

bits	1-7	not used
bits	8	line active (=1)
bit	9	line open (=1)
bit	10	line disabled (=1)
bit	11	line looped (=1)

bit 12 line output hold (=1)

bit 13 line polling disabled (=1)

bit 14 line physically disconnected (=1)

bit 15 line output queue not empty (=1)

bit 16 line stuck (=1)

If device status is desired

bits 1-12 not used

bit 13 Device looped (=1)

bit 14 Device on output hold (=1)

bit 15 Device on skip poll (=1)

bit 16 Device not operational (=1)

Normal Return

Control is returned to the caller, at the instruction following the CGSS$ macro, after the status has been fetched and stored by the system.

Error Return

Control is returned to the error return address specified in the CGSS$ macro with the error code in the A-register when any of the following errors are detected:

A-register Contents (Decimal)	Error Condition
1	The station is not configured.
2	The value of the association parameter is illegal.

Macro Action

A check is made to see if the station is configured. If it is not, the A-register is set to 1_{10} and control is returned to the caller at the error return address. If the station is configured, a check is made to see if the value of the association parameter is legal. If it is not, the A-register is set to 2_{10} and a control is returned to the caller at the error return address.

If the value of the association parameter is legal, the desired status is fetched and stored in the specified status location. Control is returned to the caller at the normal return.

Out-Line Parameter List

Word No.	Operation	Operand
0	DAC	station address pointer
1	DAC	error return address
2	DAC	association parameter address pointer
3	DAC	status address pointer

where:

Parameters in the out-line parameter list are the same as those described above for the in-line parameter list.

Examples

This example below illustrates the use of an in-line parameter list.

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6 7	8 14 15	16 31	32
	CGSS$	S,ERAD,AP,ST	
	----		NORMAL RETURN
.			
.			
ERAD	----		ERROR RETURN

.			
.			
S	DEC	1	STATION 1
AP	DEC	0	TERMINAL STATUS
ST	DEC	..	STATUS STORED HERE

The following example illustrates the use of an out-line parameter list.

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6 7	8 14 15	16 31	32
	LDX	PLT1	
	CGSS$	(X)	
	----		NORMAL RETURN
.			

```
ERAD    ----                    ERROR RETURN
        ----
PLT1    DAC     *+1
        DAC     S               STATION
        DAC     ERAD            ERROR RETURN
        DAC     AP              ASSOCIATION PARAMETER
        DAC     ST              STATUS
.
S       DEC     1               STATION 1
AP      DEC     0               TERMINAL STATUS
ST      DEC     ..              STATUS STORED HERE
```

CHANGE STATION STATUS MACRO (CCSS$)

This macro allows a user's communications task to change the status of a device, line, or terminal associated with a station. The change is allowed only if the user's communications task is connected to the station at the proper level.

CCSS$ Macro Action

CCSS$ checks for configured station, valid association parameter, valid change parameter, and connection of proper level. If they check out, the macro starts the processing of the change request and returns to the caller.

Macro Format

Location	Operation	Operand
[symbol]	CCSS$	station address pointer, error return address, association parameter address pointer, change parameter address pointer, routing tcb address pointer

where:

symbol — Optional

Is the symbolic label of the macro instruction.

station address pointer

Is the address of a word that contains the station number.

error return address

Is the address to which control is returned if an error is found during the processing of the macro call.

association parameter address pointer

Is the address of a word that contains the value of the association parameter. This parameter tells the system what kind of status change is desired. The meaning of the association parameter is as follows:

0 - terminal status change is desired

1 - line status change is desired.

2 - device status change is desired.

change parameter address pointer

Is the address of a word that contains the value of the change parameter. The value of the change parameter depends upon the value of the association parameter.

If the association parameter is zero (terminal) the change parameter can then equal:

0 - Poll Enable

1 - Poll Disable

2 - Select Enable

3 - Select Disable

If the association parameter is one (line), then the change parameter can equal:

0 - Line Enable

1 - Line Disable

2 - Auto-Answer Enable } Switched Lines

3 - Auto-Answer Disable }

4 - Line Loop

5 - Line Unloop

6 - Poll Enable } Controller Line

7 - Poll Disable }

8 - Line Output Enable

9 - Line Output Hold

10 - Link Enable } Binary Synchronous Communications Line Only

11 - Link Disable }

If the association paramter is 2 (device), then the change parameter can equal:

0 - Device Enable

1 - Device Disable

2 - Line Enable } Binary Synchronous Device Only

3 - Link Disable }

4 - Device Loop } Certain Devices Only

5 - Device Unloop }

6 - Poll Enable

7 - Poll Disable

8 - Output Enable

9 - Output Disable

Routing tcb address pointer

Is the address of a word that contains the address of the Routing TCB that marks the connection of the user's communications task with the specified station.

Normal Return

Control is returned to the caller, at the instruction following the CCSS$ macro, after the processing of the change request has been started.

Error Return

Control is returned to the error return address specified in the CCSS$ macro with the error code in the A-register when any of the following errors are detected.

A-register Contents (Decimal)	Error Condition
1	The station is not configured.
2	The value of the association parameter is illegal.
3	The station is not connected to this communications task.
4	The station is connected to this communications task on a level lower than the change level desired.
5	The value of the change parameter is illegal.

Macro Action

A check is made to see if the specified station is configured. If not, the A-register is set to 1_{10} and control is returned to the caller at the error return address. If the station is configured, a check is made to see if the values of the association and change parameters are valid. If not, the A-register is set to 2_{10} or 5_{10} and control is returned to the caller at the error return address. If the values of the association and change parameters are valid, a check is made to see if the user's communications task is connected on a level consistent in the change request. If not, the A-register is set to 4_{10} and control is returned to the caller at the error return address. If connected, a check is made to see if the station is connected to this communication task. If not, the A-register is set to 3_{10} and control is returned to the caller at the error return address.

If the station is connected at a level consistent with the change request, the change request processing is started and return is made to the caller at the normal return.

Out-Line Parameter List

Word No.	Operation	Operand
0	DAC	station address pointer
1	DAC	error return address
2	DAC	association parameter address pointer
3	DAC	change parameter address pointer
4	DAC	routing tcb address pointer

where:

Parameters in the out-line parameter list are the same as those described above for the in-line parameter list.

Examples

The example below illustrates the use of an in-line parameter list.

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6 7	8 14 15	16 31	32
	CCSS$	S,ERAD,AP,CP,RTCA	
	----		NORMAL RETURN
ERAD	----		ERROR RETURN

S	DEC	1	STATION 1
AP	DEC	1	LINE STATUS CHANGE
CP	DEC	0	LINE ENABLE
RTCA	DAC	RTCB	ROUTING TCB ADDRESS

The following example illustrates the use of an out-line parameter list.

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6 7	8 14 15	16 31	32
	LDX	PLTI	
	CCSS$	(X)	
	----		NORMAL RETURN
.			

```
ERAD    ----                    ERROR RETURN
        ----
PLT1    DAC     *+1
        DAC     S               STATION
        DAC     ERAD            ERROR RETURN
        DAC     AP              ASSOCIATION PARAMETER
        DAC     CP              CHANGE PARAMETER
        DAC     RTCA            ROUTING TCB
.
.
S       DEC     1               STATION 1
AP      DEC     1               LINE STATUS CHANGE
CP      DEC     0               LINE ENABLE
RTCA    DEC     RTCB            ROUTING TCB ADDRESS
```

TERMINATE COMMUNLCATIONS TASK MACRO (CTMC$)

This macro terminates a user's communicatons task further input data or output status is received from a connected station.

CTMC$ Macro Action

The CTMC$ macro checks for configured station and valid connection. If they check out, the macro resets a "communications task is running" flag and exits to the Executive.

Macro Format

Location	Operation	Operand
[symbol]	CTMC$	station address pointer, error return address, routing tcb address pointer

where:

symbol — Optional

Is the symbolic label of the macro instruction.

station address pointer

Is the address of a word that contains any station number that to which this communications task is connected.

error return address

Is the address to which control is returned if an error is found during the processing of the macro call.

routing tcb address pointer

Is the address of a word that contains the address of a Routing TCB that marks the connection of the user's communications task with the specified station.

Normal Return

CTMC$ exits to the Executive and control does not return to the caller.

Error Return

Control is returned to the error return address specified in the CTMC$ macro with the error code in the A-register when any of the following errors are detected:

A-register Contents (Decimal)	Error Condition
1	The station is not configured.
2	The station is not connected to this communications task.

Macro Action:

A check is made to see if the station is configured. If not, the A-register is set to 1_{10} and control is returned to the caller at the error return address. If the station is configured, a check is made to see if the station is connected to the user's communications task. If not, the A-register is set to 2_{10} and control is returned to the user at the error return address.

If the station is connected, exit is made to the Executive.

Out-Line Parameter List:

Word No.	Operation	Operand
0	DAC	station address pointer
1	DAC	error return address
2	DAC	routing tcb address pointer

where:

Parameters in the out-line parameter list are the same as those described above for the in-line parameter list.

Examples

The examples below illustrates the use of an in-line parameter list.

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6 7	8 14 15	16 31	32
	CTMC$	S,ERAD,RTCA	
ERAD	----		ERROR RETURN
.			
.			
S	DEC	0	STATION 0
RTCA	DAC	RTCB	ROUTING TCB ADDRESS

The following example illustrates the use of an out-line parameter list.

LOCATION	OPERATION	OPERAND FIELD	COMMENTS
1 6 7	8 14 15	16 31	32
	LDX	PLT1	
	CTMC$	(X)	
ERAD	----		ERROR RETURN
PLT1	DAC	*+1	
	DAC	S	STATION
	DAC	ERAD	ERROR RETURN
	DAC	RTCA	
S	DEC	0	STATION 0
RTCA	DAC	RTCB	ROUTING TCB ADDRESS

APPENDIX A
FREE MEMORY BLOCK PARAMETER PASSING TECHNIQUE

The format of the parameter passing technique used by FORTRAN under OS/700 when using free memory blocks involves the TCB format, Parameter Block format, Parameter format, and data structure.

TCB Format

Use of the free memory block parameter passing technique is designated whenever the TCB has the following format:

Word Relative Displacement Symbol	Contents
ZTCBP1	Must be zero. This is parameter 1 in the SAC$, STS$, CTC$, CCA$, or CCL$ call.
ZTCBP2	Points to the first word of the block containing the actual parameters. This is parameter 2 in the SAC$, STS$, CTC$, CCA$, or CCL$ call.

Parameter Block Format

The block containing the actual parameters has the following format:

Word No.	Contents
0	Reserved
1	Reserved
2	Parameter block size descriptor.

Bit	Contents
1	Block release control. 0 = Release block to free memory when done. 1 = Don't release block to free memory when done.
2-8	Must be zero.

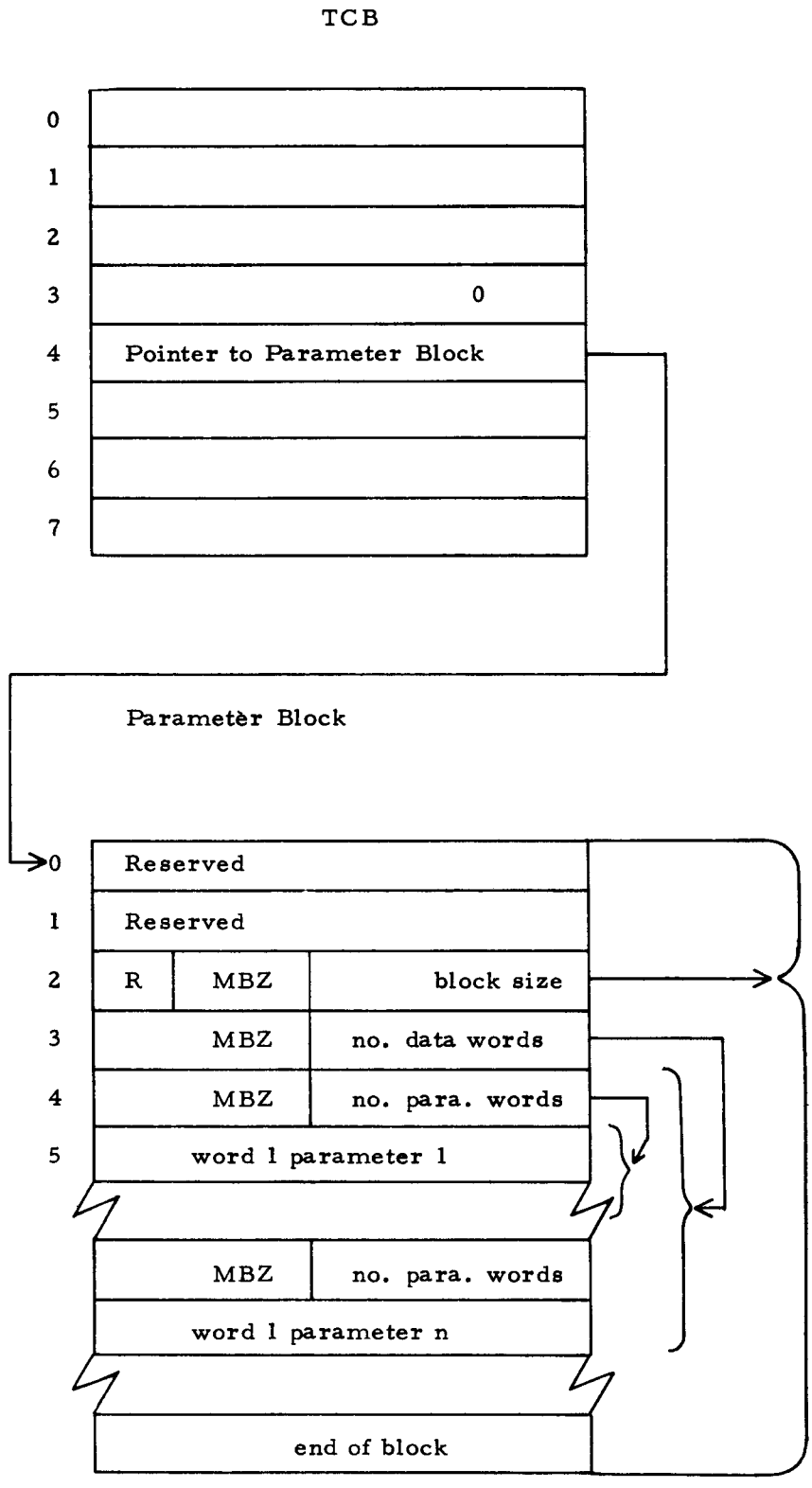
Word No.	Contents
	Bit Contents 9-16 Parameter block size in number of words (note, it is not expressed as a power of 2).
3	Parameter block descriptor. Bit Contents 1-8 Reserved for parameter passing technique type. Must be zero (this is technique 0). 9-16 Number of data words in the parameter block. This is a count of the number of following words which actually contain parameter data.
4-N	Parameter data.

Parameter Format

Each parameter in the parameter block has the following format:

Word No.	Contents
0	Parameter descriptor. Bits Contents 1-8 Reserved for parameter type. Must be zero (this is type 0). 9-16 Number of data words in the parameter. This count does not include this word itself.
1	First word of actual parameter.
2	Second word of actual parameter (if more than 1 word).
3	Third word of actual parameter (if more than 2 words).

Data Structure Diagram



MBZ = Must be zero.

Parameter Block Example

As an example of the use of the free memory block parameter passing technique, the following parameter block would be passed by the Start Activity Utility if the operator types in /SA ABC,,,10,'123,UVWXYZ (CR).

Word No.	Octal Contents	Meaning
0	0	Reserved
1	0	Reserved
2	20	16 Word Block, to be returned
3	10	8 words of data
4	1	1 word for Parameter 1
5	12	Decimal 10
6	1	1 word for Parameter 2
7	123	Octal 123
8	3	3 words for Parameter 3
9	152726	UV
10	153730	WX
11	154732	YZ
12		Unused
13		Unused
14		Unused
15		Unused

APPENDIX B
PHYSICAL I/O GENERIC DEVICE TYPE ASSIGNMENTS

Type No.	Generic Device Type
0	KSR
1	Reserved
2	High-Speed Paper Tape Reader
3	High-Speed Paper Tape Punch
4	Reserved
5	Card Punch
6	Card Reader
7	Fixed Head Disk Subsystem
8	Removable Disk Subsystem
9	Line Printer
10	Magnetic Tape Subsystem
11	Reserved
12	ASR - Keyboard and Printer
13	ASR - Reader and Punch

APPENDIX C

PHYSICAL I/O DATA MODE ASSIGNMENTS

Mode No.	Data Mode
0	ASCII without Checksum
1	Binary without Checksum
2	Verbatim
3	ASCII with Checksum for High-Speed Paper Tape Reader/Punch and ASR Paper Tape Reader/Punch Only
4	Binary with Checksum for High-Speed Paper Tape Reader/Punch and ASR Paper Tape Reader/Punch Only

APPENDIX D
PHYSICAL I/O STATUS BLOCK FORMAT

The I/O status block (Words 0-7) specified for a physical I/O request has Words 1, 3, 4 and 6 containing the following information upon I/O completion.

Word 1

Bit	Interpretation
1	Word 4 contains the hardware status word which indicates the error. If the hardware has two status words, the second is returned in Word 6.
2	Reserved
3	Missed Interrupt
4	Data Not Ready
5	Device Disabled
6	Missed Data
7	Checksum Error
8	Parity Error
9	Format Error or Mode Error
10	Recovery Error or Control P
11	Protect Error
12	Reserved
13	End-of-File
14	End-of-Media or Range Error
15	Beginning-of-Tape or No Block
16	Device Busy

The above described states are indicated if the appropriate bit is set.

Word 3

For input and output requests, this word contains the actual number of words transferred or received. For space file and space record requests, this word contains the number of files or records spaced.

Word 4

The interpretation of this word is indicated by bit 1 of Word 1.

Word 6

This word is used only for devices which have two hardware status words. The second hardware status word is returned here if bit 1 in Word 1 is set.

APPENDIX E
PHYSICAL I/O DEVICE INFORMATION

KSR

The only legal physical I/O requests for the KSR are input (INP$) and output (OTP$). The only legal data mode for the KSR is 0 (ASCII).

When input is requested from the KSR, the characters that are input from the keyboard are packed two characters per word and stored in the user's buffer. Characters are input until the user's buffer is full or until a carriage return character is input from the keyboard. The carriage return character is stored in the user's buffer. The following control characters are checked on input and the action described is taken:

@ - Ignore last line input and start inputting a new line.

← - Ignore last character typed in.

Control P - Whenever a control P (CTRL-P) is received on a reserved device, either of two actions may be taken:

- Current I/O in process will be terminated with the appropriate bit set in the status word.
- If there is no I/O in process, then the special action TCB will be scheduled. The device will not respond to another CTRL-P until an I/O request is made on that device.

Control-Shift-M — Terminate input request and return end-of-file status word.

When output to the KSR is requested, all of the data specified in the buffer must be packed two characters per word and is output to the printer until the range count is exhausted. Before starting the physical output, the first word of the buffer is examined for line spacing control. If required, the necessary carriage control characters are set up. The second byte of the first word of the caller's buffer is treated as follows:

Blank	=	Advance one line (insert CR, LF)
0	=	Advance two lines (insert CR, LF, LF)
+	=	Do not advance (no insert)
Other	=	Ignore character and advance one line (insert CR, LF)

All other output carriage control is the user's responsibility and must be included in his buffer.

The following is a description of the status information returned to the user when control is transferred to the user's I/O completion return:

Word 1 (Second Word) of I/O Status Block

Bit	Interpretation
1-3	Reserved
4	Data not ready
5	Missed interrupt
6-8	Reserved
9	Wrong mode
10	Control-P received during output
11-13	Reserved
14	Range error
15-16	Reserved

The above described states are indicated if the appropriate bit is set.

Word 3 (fourth word) of I/O Status Block

See Appendix D.

High Speed Paper Tape Reader

The legal physical I/O request for the high speed paper tape reader is input (INP$). The legal data modes for the high speed paper tape reader are all of those listed in Appendix C. When input is requested from the high speed paper tape reader, the data on the paper tape is read until an X OFF character is reached. The data read is then stored in the user's input buffer. The word count in Word 3 (fourth word) of the I/O status block will indicate the number of words stored in the user's input buffer.

The following is a description of the status information returned to the user when control is transferred to the user's I/O completion return:

Word 1 (second word) of I/O Status Block

Bit	Interpretation
1-2	Reserved
3	Missed interrupt
4	Data not ready
5	Device Disabled
6	Reserved
7	Checksum error
8	Parity error
9	Format error
10	Recovery error
11-12	Reserved
13	End-of-File
14	Range error
15	No free memory available
16	Reserved

The above described states are indicated if the appropriate bit is set.

Word 3 (fourth word) of I/O Status Block

See Appendix D.

High Speed Paper Tape Punch

The legal physical I/O requests for the high speed paper tape punch are output (OTP$) and end of file (EOF$). The legal data modes for the high speed paper tape punch are all of those listed in Appendix C. When output is requested to the high speed paper tape punch, the data in the user's output buffer is punched on the paper tape until the range count is exhausted. The punching of the end of file characters is initiated only by an EOF$ request.

The following is a description of the status information returned to the user when control is transferred to the user's I/O completion return:

Word 1 (second word) of I/O Status Block

Bit	Interpretation
1-2	Reserved
3	Missed interrupt
4	Data not ready

Bit	Interpretation
5	Device disabled
6-9	Reserved
10	Recovery error
11-14	Reserved
15	No free memory available
16	Reserved

The above described states are indicated if the appropriate bit is set.

Word 3 (fourth word) of I/O Status Block

See Appendix D.

Card Reader and Card Punch

The legal physical I/O request for the card reader is input (INP$). The legal physical I/O request for the card punch is output (OTP$). The legal data modes for the card reader and card punch are 0 (ASCII) and 1 (Binary).

When input is requested from the card reader, the data is input from 1 card and converted, if the configurable conversion routines have been loaded in the system. The number of words stored in the user's input buffer will equal the range value unless the range value exceeds the number of words read on a single card. In this case only the number of words on the card are stored in the user's input buffer.

When output is requested to the card punch, the data in the user's output buffer is converted if the configurable conversion routines have been loaded in the system. If the range of the words to be output is greater than the number of words that can be punched on a single card, only one card will be punched, and the remaining words in the user's output buffer will be ignored.

The following is a description of the status information returned to the user when control is transferred to the user's I/O completion return:

Word 1 (second word) of I/O Status Block

Bit	Interpretation
1	An error has occurred, the hardware status is in Word 4
2-4	Reserved

Bit	Interpretation
5	Device disabled
6-9	Reserved
10	Recovery error - an attempt to retry has resulted in an error
11-12	Reserved
13	End-of-file - EOF detected, not an error condition
14-16	Reserved

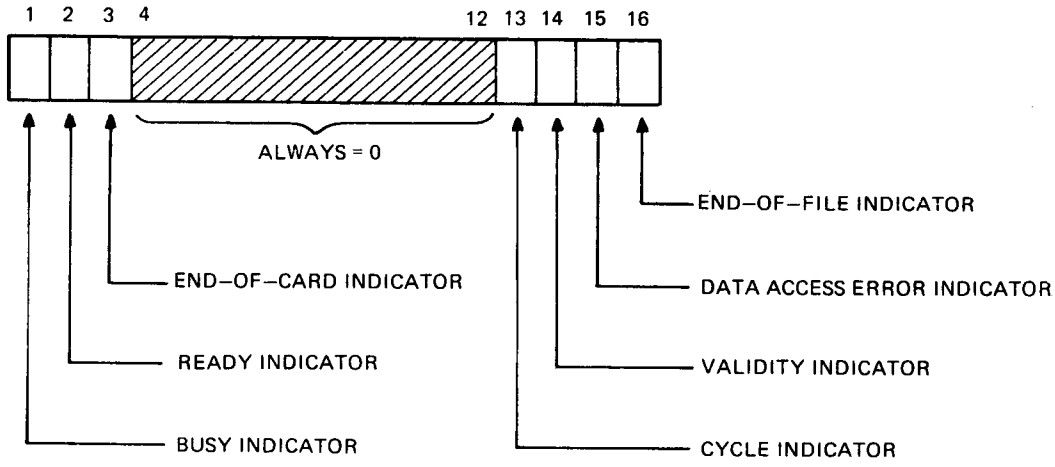
The above described states are indicated if the appropriate bit is set.

Word 3 (fourth word) of I/O Status Block

See Appendix D.

Word 4 (fifth word) of I/O Status Block for Input Requests for the Card Reader Device.

In addition to the status information returned in Word 1 of the I/O status block, the following hardware status is always returned in Word 4 of the I/O status block.



Word 4 (fifth word) of I/O Status Block for the Card Reader/Punch Device

In addition to the status information returned in Word 1 of the I/O status block, the following hardware status is always returned in Word 4 of the I/O status block.

Bit	Condition Tested	Bit	Condition Tested
1	Controller Busy	9	Unused
2	Ready	10	
3	End of Card	11	
4	Unused	12	Punch Check Error
5		13	Read Check Error
6		14	Validity Error
7		15	Data Access Error
8		16	Read End of File

Fixed Head Disk Subsystem

The only legal Physical I/O requests for the fixed head disk are input (INP$) and output (OTP$). The data mode (see Appendix C) does not apply to the fixed head disk, but the data mode specified in the DCB for I/O requests for the disk must be in the range from 0 to 4.

When input is requested from the fixed head disk, the data input from the disk and stored in the user's input buffer will be determined by the range value or the length of the physical disk record. If the range value is less than or equal to the physical disk record, the number of words stored in the user's input buffer is determined by the range value. If the range value is greater than the physical disk record length, the number of words stored in the user's input buffer is determined by the length of the physical disk record.

When output is requested to the fixed head disk, the range value must not exceed the physical disk record length. The data output to the disk from the user's output buffer is determined by the range value.

The following is a description of the status information returned to the user when control is transferred to the user's I/O completion return.

Word 1 (second word) of I/O Status Block

Bit	Interpretation
1	An error has occurred, the hardware status is in Word 4
2-16	Reserved

The above described states are true if the appropriate bit is set.

Word 3 (fourth word) of I/O Status Block

See Appendix D.

Word 4 (fifth word) of the I/O Status Block

The hardware status is always returned in Word 4 of the I/O status block.

Bit	Condition Tested
1	Operational
2	Busy
3	Active
4	Check byte error
5	Time out error
6	Device not active error
7	Write protect error
8	Access error
9	Bus parity error
10	Device zero active
11	Device one active
12	Device two active
13	Device three active
14	Not used
15	Device going active interrupt
16	Busy reset interrupt

Removable Disk Subsystem

The only legal physical I/O requests for the removable disk are input (INP$) and output (OTP$). The data mode (see Appendix C) does not apply to the removable disk but the data mode specified in the DCB for I/O requests for the disk must be in the range from 0 to 4.

When input is requested from the removable disk, the data input from the disk and stored in the user's input buffer will be determined by the range value or the length of the physical disk record. If the range value is less than or equal to the physical disk record, the number of words stored in the user's input buffer is determined by the range value. If the range value is greater than the physical disk record length, the number of words stored in the user's input buffer is determined by the length of the physical disk record.

When output is requested to the removable disk, the range value must not exceed the physical disk record length. The data output to the disk from the user's output buffer is determined by the range value.

The following is a description of the status information returned to the user when control is transferred to the user's I/O completion return:

Word 1 (second word) of I/O Status Block

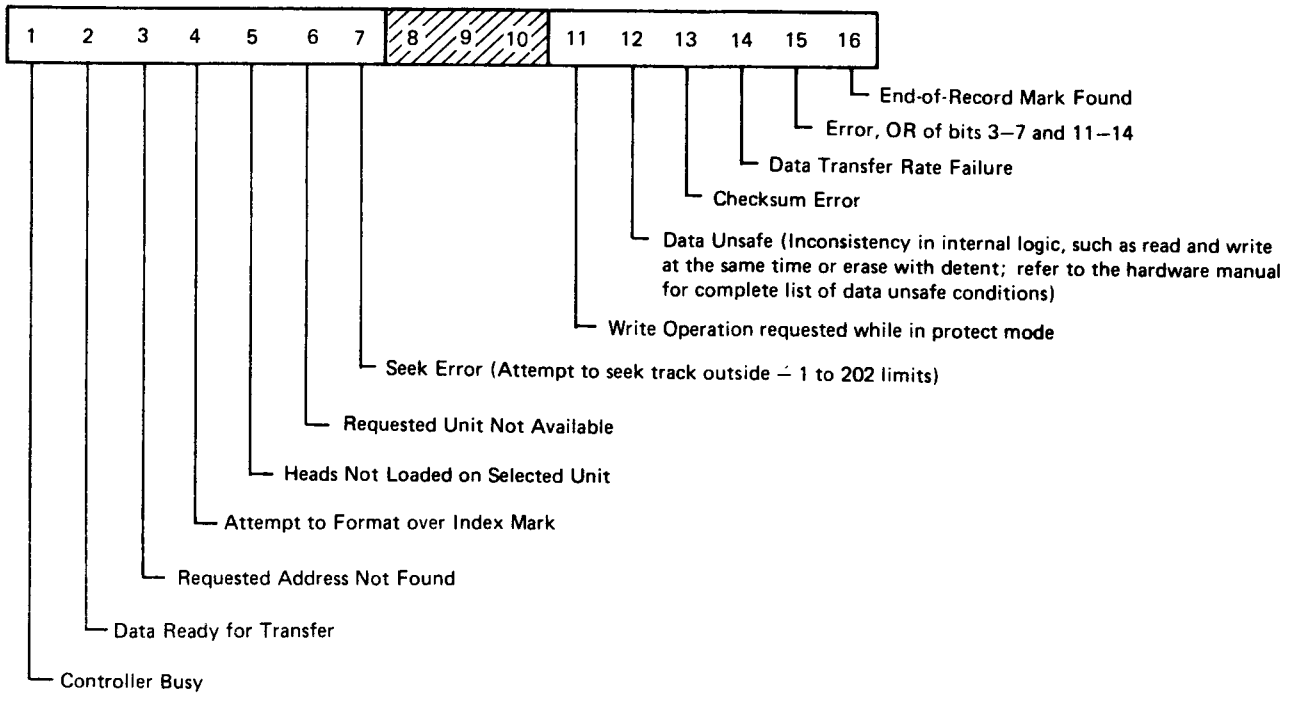
Bit	Interpretation
1	Word 4 contains the hardware status word which indicates the error
2	Reserved
3	Missed interrupt
4-15	Reserved
16	Device busy

The above described states are indicated if the appropriate bit is set.

Word 3 (fourth word) of I/O Status Block

See Appendix D.

Word 4 (fifth word) of I/O Status Block



In addition to the software status returned in Word 1 of the I/O status block, the following hardware status is returned in Word 4 of the I/O status block whenever bit 1 of Word 1 is set and also whenever Word 1 is 0.

Line Printer

The only legal physical I/O request for the line printer is output (OTP$). The only legal data mode for the line printer is 0 (ASCII). When output is requested to the line printer, the characters that are to be output must be packed two characters per word, and the first word must contain a control character which is right justified. If the range of the words to be output is greater than one line on the line printer, only one line will be output, and the remaining words in the user's output buffer will be ignored.

The first word of the user's output buffer must contain an ASCII forms control character right justified as follows:

SPACE	240	Advance one line
+ (PLUS)	253	No line advance
0 (ZERO)	260	Advance two lines
1	261	Advance to Top of Form
3	263	Advance according to Channel 3
4	264	Advance according to Channel 4
5	265	Advance according to Channel 5
6	266	Advance according to Channel 6
7	267	Advance according to Channel 7
H	310	Advance to Top of Form

If none of the above, advance one line is forced.

The following is a description of the status information returned to the user when control is transferred to the user's I/O completion return:

Word 1 (Second Word) of I/O Status Block

Bit	Interpretation
1	Word 4 contains the hardware status word which indicates the error. If no error, this Bit = 0.
2-4	Reserved
5	Disabled
6-9	Reserved
10	Recovery error
11-16	Reserved

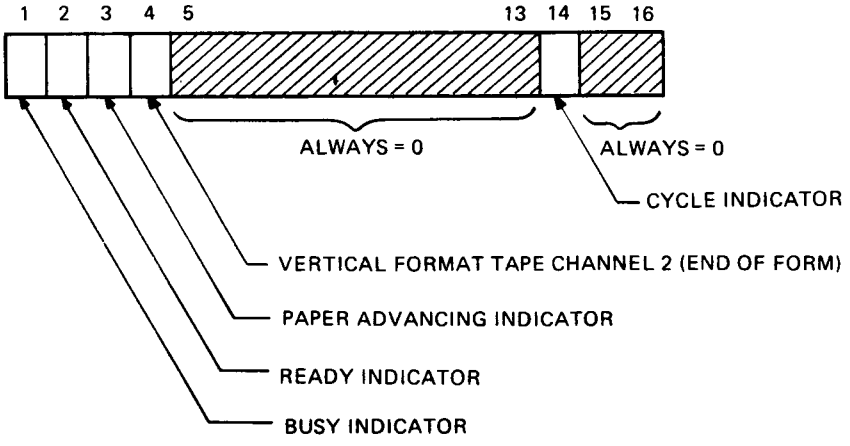
The above described states are indicated if the appropriate bit is set.

Word 3 (fourth word) of I/O Status Block

See Appendix D

Word 4 (fifth word) of I/O Status Block

In addition to the software status returned in Word 1 of the I/O status block, the following hardware status is always returned in Word 4 of the I/O status block:



Magnetic Tape

The Physical I/O requests for the magnetic tape are input (INP$), output (OTP$), end of file (EOF$), space file (SPF$), space record (SPR$), rewind (RWD$), and unload (ULD$). The legal data modes for the magnetic tape are 0 (ASCII), 1 (Binary), and 2 (Verbatim). Binary mode specifies that 16 bits of each word are to be transferred. Verbatim mode specifies that the high order 12 bits of each word are to be transferred.

When input is requested from the magnetic tape, one record is read from the magnetic tape and the data is stored in the input buffer. The range value must be greater than or equal to the record that is to be read from the magnetic tape. If the range value is less than the

physical magnetic tape record, the user will get the parity error indicator set in the I/O status block. If the data mode specified was ASCII, the data read from the magnetic tape is converted to ASCII and stored in the user's buffer.

When output is requested to the magnetic tape, all of the data in the user's output buffer will be output to the record on the magnetic tape. If the data mode specified was ASCII, the data in the user's output buffer is converted to BCD before being output to the magnetic tape.

If the end of the tape or the beginning of the tape is detected before the specified number of files to be spaced has been reached when processing a SPF$ request, the spacing of the files is not continued. The number of files actually spaced is return in Word 3 of the I/O status block.

If the end of the file is detected before the specified number of records to be spaced has been reached when processing a SPR$ request, the spacing of the records is not continued. The number of records actually spaced is returned in Word 3 of the I/O status block.

The following is a description of the status information returned to the user when control is transferred to the user's I/O completion return:

Word 1 (second word) of I/O Status Block

Bit	Interpretation
1-2	Reserved
3	Missed interrupt
4	Reserved
5	Device disabled
6-7	Reserved
8	Parity error
9-10	Reserved
11	Protect error
12	Reserved
13	End-of-file
14	End-of-tape
15	Beginning of tape
16	Device busy

The above described states are indicated if the appropriate bit is set.

Word 3 (fourth word) of I/O Status Block

See Appendix D

ASR Keyboard and Printer

The physical I/O requests, the mode, and the control characters, etc., for the ASR keyboard and printer are the same as described above for the KSR. The only difference is that bit 15 in Word 1 (second word) of the I/O status block will be set if there are no free memory blocks available for the ASR device driver to use.

ASR Reader and Punch

The legal physical I/O request for the ASR Reader is input (INP$). The legal Physical I/O requests for the the ASR punch are output (OTP$) and end of file (EOF$).

The legal data modes for the ASR reader and punch are all of those listed in Appendix C.

When input is requested from the ASR reader, the data on the paper tape is read until an X OFF character is reached. The data read is then stored in the user's input buffer. The word count in Word 3 (fourth word) of the I/O status block will indicate the number of words stored in the user's input buffer.

When output is requested to the ASR punch, the data in the user's output buffer is punched on the paper tape until the range count is exhausted. The punching of the end-of file character is initiated only by an EOF$ request.

The following is a description of the status information returned to the user when control is transferred to the user's I/O completion return.

Word 1 (second word) of I/O Status Block

Bit	Interpretation
1-2	Reserved
3	Missed interrupt
4	Device not ready
5-6	Reserved
7	Checksum error
8	Parity error
9	Mode error or format error

10	Control-P received during printer output
11-12	Reserved
13	End-of-file
14	Range error
15	Free memory not available
16	Reserved

Word 3 (fourth word) of I/O Status Block

See Appendix D.

APPENDIX F

EXECUTIVE MACRO CALL ERROR RETURN CODE TABLE[1]

Contents of A-Register	Error Conditions
0	The requested device was already RESERVed when another RESERVE request was issued.
0	There were no blocks of free memory when a GBL$ request was made.
0	The block size parameter specified in the RBL$ request was illegal.
1	The "<Generic Device Type>" specified in the Device Control Block is not configured for this system.
1	The block size parameter specified in the GBL$ request was illegal.
1	A CCA$ or CCL$ request was issued to connect the Absolute timer, but the Absolute Timer was not configured into the system.
2	The requested device was already enabled when an Enable request was issued.
2	A TYP$ or TPR$ request was issued but no messages could be typed on the operator console because an OS/700 utility reserved the operator console.
3	The "<Logical Unit Number>" specified in the Device Control Block is not configured for this system.
3	A TPR$ request was issued but could not be acknowledge because the message table is full.
4	The "<Data Mode>" specified in the Device Control Block is not in the range 0 to 4, or is not a valid mode for the specified device.
4	A TYP$ or TPR$ request was issued and an error was encountered before or during the message output.
5	The number of words specified in the "<Range>" parameter is less than 1 or more than 4095.
6	The number of records or files to be spaced is = 0.

[1] Excluding Communication Macro Error Return Codes.

Contents of A-Register	Error Conditions
7	The function requested is illegal for the device specified in the Device Control Block.
8	The requested device has not been previously reserved under the "<User ID>" specified in the Device Control Block.
9	An Input request was issued for an Output only device.
10	An Output request was issued for an Input only device.
11	Not Used
12	A request was issued for a disabled device
13	A Reserve request was issued for the KSR (Generic Device Type 0) or the ASR (Generic Device Type 12) and the "<Control P TCB Address Pointer>" was not specified in the parameter list of the Reserve request.
14	The activity or file name was not found in the disc directory.
15	The activity or file name already exists in the disc directory.
16	The activity or file name cannot be added to the disc directory because the disc directory is full.
17	The allocate request cannot be honored because all of the work areas on disc are being used, or the disc segment number specified in the Deallocate request was not the segment number of the first segment of the work area to be deallocated.
18	A Disc Error occurred when transferring a file or activity to or from the disc.
19	A Logical I/O request was issued for a file that was not previously Opened, a Put request was issued for a file that was Opened in Input mode, or a Get request was issued for a file that was Opened in Output mode.
20	An Open request was issued for a null file (a null file is a file that is currently being created but has not been Closed), or the "<Record Length Address>" was not specified in the parameter list of the Open request.
21	The "<Record Buffer Address>" is not specified in the parameter list of the Put request, or the "<Record Buffer Address>" is not specified in the parameter list of the Get request when the logical record is greater than or equal to the physical disc record.
22	There is no activity area for the named activity (the main-memory starting address given in the disc directory is not equal to the beginning of any activity area), or the activity is too large to fit into the allocated activity area (i.e., the activity ending address exceeds the activity area ending address).
23	Unused
24	The device driver for the requested device is disc resident, and the there was an activity area overrun error while loading the device driver.

Contents of A-Register	Error Conditions
25	The device driver for the requested device is disc resident, and there was a disc read error while loading the device driver.
26	The request device has a disc resident driver, and the device was not previously Reserved by a RSV$ macro instruction.
no significance	The parameter set type in the GSP$ request was out of range.
no significance	A DCA$ or DCL$ request was issued, but the specified activity or task was not in the clock queue.
no significance	One of the parameters specified in the SDT$ request was out of range.

Honeywell

HONEYWELL INFORMATION SYSTEMS

AG02, Rev. 1