2
(m/+)

Report No. 72-0012
Contract No. NAS8-27359

(NASA-CR-123559) MULTI-PROCESSING CONTROL SYSTEM FOR THE SEL 840MP (MPCS/1) USERS GUIDE. VOLUME 1: PROGRAMMING GUIDE (M&S Computing, Inc.) 27 Mar. 1972 91 p

N72-25216

Unclas
G3/08 15495

MULTI-PROCESSING CONTROL SYSTEM FOR THE SEL 840MP (MPCS/1)

USERS GUIDE

VOLUME I - PROGRAMMING GUIDE

March 27, 1972

Prepared for:

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
George C. Marshall Space Flight Center
Marshall Space Flight Center, Alabama 35812

M&S COMPUTING, INC.

PREFACE

This document provides the information required by an application programmer to fully utilize the SEL 840MP Multi-Processing Control System - Version I (MPCS/1). This system was developed by M&S Computing under Contract No. NAS8-27359 for NASA/MSFC.

Participating personnel were:

W. F. Anderson
J. R. Conway
L. C. Keller
M. L. Williams

Approved by:

[illegible]

T. T. Schansman

TABLE OF CONTENTS

TABLE OF CONTENTS
(continued)

TABLE OF CONTENTS
(continued)

TABLE OF CONTENTS
(continued)

TABLE OF CONTENTS
(continued)

TABLE OF CONTENTS
(continued)

LIST OF FIGURES

1. INTRODUCTION

1.1 General

Multi-Processor Control System (MPCS) is the operating system specifically designed for the SEL 840MP. In addition to the functions provided by CHANE (reference "CHANE SEL 840MP", Code Research Corporation, July 15, 1970), MPCS provides the functions summarized below. To enable users to gain a mastery of MPCS services and functions, explicit details are provided in subsequent sections of this user's guide.

1.2 Task Control

MPCS/1 is a "TASK-oriented" system. A TASK is a program module, whose scheduling can only be controlled through MPCS/1. This is in contrast to subroutines which are called directly by a TASK. One or more TASKS form a JOB or application program.

The user may declare a TASK to be PROCESSOR DEPENDENT. This means that the TASK must be executed by a particular processor. This would be necessary, for example, if a TASK is stored in a private memory. Conversely, the TASK may be declared to be PROCESSOR INDEPENDENT; i.e., it may be executed by any available processor.

The memory available and accessible by a particular processor is called a PROCESSOR DOMAIN, and includes a processor's private memory as well as all of shared memory.

TASK execution is scheduled through MPCS/1 based on PRIORITY and/or PERIODIC criteria. PRIORITY is assigned by the user to a TASK and reflects its execution priority relative to other TASKs. PERIODIC means that a TASK is to be scheduled at a specific time increment. The latter may, in addition, be specified to be SINGLE MODE - scheduled once for each execution request - or be specified to be ITERATIVE MODE - scheduled repetitively at a specified time interval.

MPCS/1 is also a "shared operating system". That is, each processor executes its own operating system functions either from shared memory or its own private memory as appropriate. This allows each processor to execute its own job, as well as execute tasks of jobs assigned to other processors with minimal problems of interprocessor coordination.

MPCS/1 does not force an application into a particular scheduling scheme. The application may use, for example, time/interrupt driven asynchronous sequences, major/minor loop, fully synchronous, or any other scheme suited to its requirements. Task Control does provide the basic control functions necessary to execute the scheduling specified over one or more processors.

The tasks communicate with Task Control, and each other, through a set of Task Control Commands. The detailed formats are described later in this document; the purpose of the individual commands is described below.

EXECUTE	-	This signals that a task should be dispatched for execution in accordance with scheduling criteria provided in the call parameters.
TERMINATE	-	A task signals that it has completed its execution.
DELETE	-	To signal that a task, previously requested to be EXECUTEd, should be set to a non-scheduled state.
SUSPEND	-	To signal that execution of a task should be temporarily halted and should be restarted as indicated in the call parameters provided. The parameters may indicate: o Restart based on time o Restart when requested by other task
CHECK TASK	-	To allow a task to interrogate the execution status of another task. This could be used for synchronization.
TRANSMIT/ RECEIVE	-	To allow transmission of data between tasks.

In addition a "NEXT TASK" may be associated with a task. The NEXT TASK information defines the task to be executed when a task completes its execution. NEXT TASK information may be modified during execution.

Each processor may accept a single job (i.e. set of tasks) at a time. No additional jobs can be assigned to a processor as long as the currently executing job has not been completed. However, if no jobs are assigned to a particular processor it will automatically execute processor-independent tasks that are part of a job assigned to another processor. Even if a job is assigned to a processor, but no tasks are ready for execution, the processor will also assist another processor in the execution of its tasks. In addition, a processor may be forced by another processor to execute the other processor's tasks, regardless of the status of its own tasks.

Figure 1-1 depicts the general process flow of the Task Control. The processor dispatcher is responsible for obtaining the highest priority and/or earliest requested task from the Priority Schedule Queue and to put it in active execution on the appropriate processor.

Tasks are added to the Priority Schedule Queue by the Timer Processor or the Priority Scheduler. The Timer Processor adds the tasks to the Priority Schedule Queue from the Periodic Queue as the specified time criteria are met. Tasks are added to the Periodic Queue by the Periodic Scheduler as requested by the application task. Tasks that do not use time as a scheduling criteria are handled through the Priority Scheduler. All other Task Control commands are also handled through the Priority Scheduler.

1.3 Input/Output Control

A detailed description of the SEL 840MP can be found in "Reference Manual, SEL 840MP General Purpose Computer System", Form No. 301-095098-001, System Engineering Laboratories. Of interest here is the functional systems configuration depicted in Figure 1-2.

Each processing element has access to its own private memory as well as all of shared memory. Simultaneous access of a shared memory module by the processors, is resolved by assigning fixed priorities to the processors. This particular access resolution scheme may result in the dominance of a particular processor, making true simultaneous execution from a shared memory module impractical.

Input/output devices may be attached to a processing element channel or to shared memory. When they are attached to a processing element channel, they can access a particular private memory and all of shared memory, but can only be commanded by the particular processing element. When the devices are attached to a shared memory channel, they can only access shared memory, but may be commanded by any processor.

To support the potential I/O configurations described above, it is necessary to provide "Remote Input/Output". This allows any task

TASK CONTROL

PROCESS FLOW

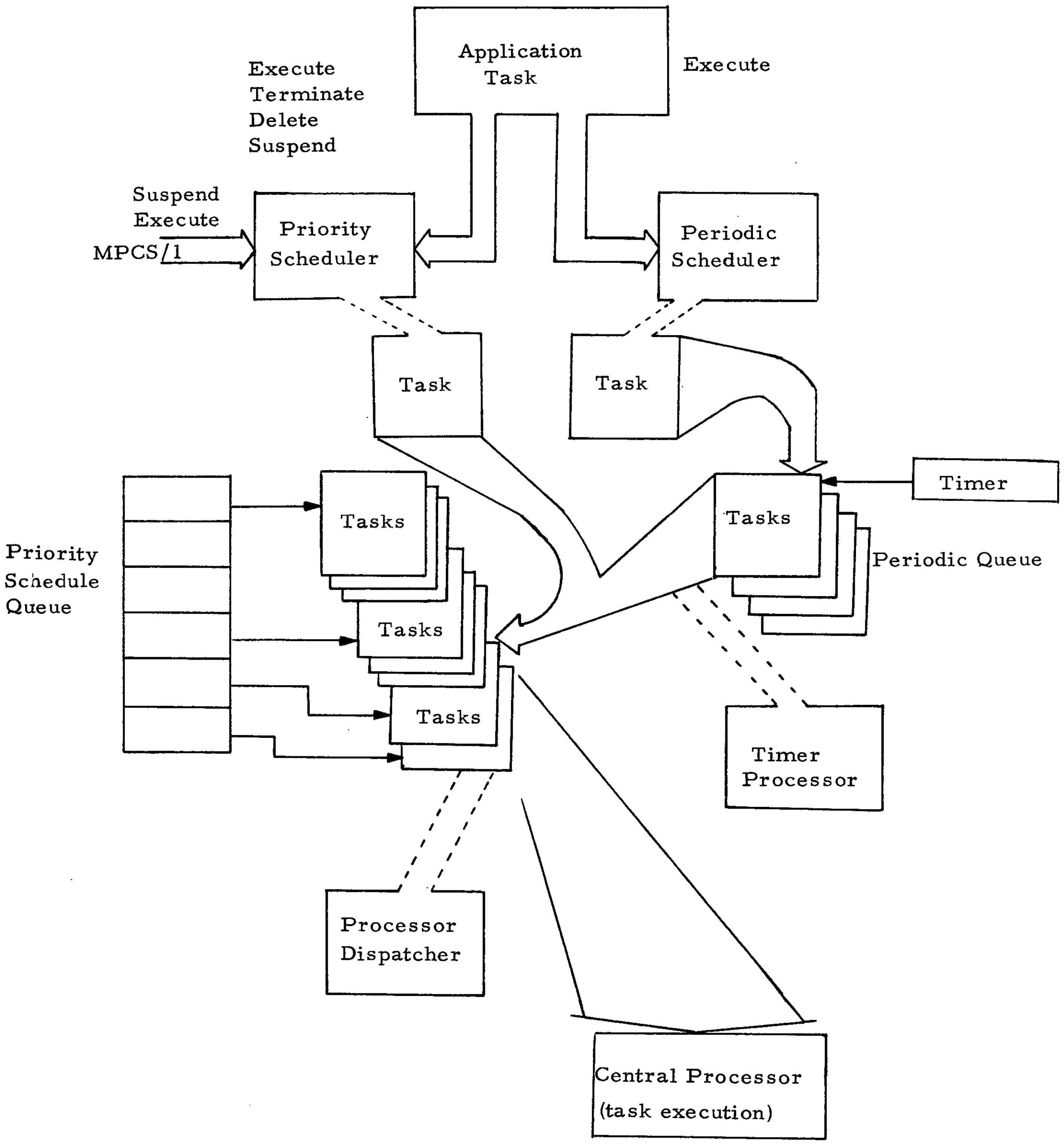


Figure 1-1

840MP HARDWARE CONFIGURATION

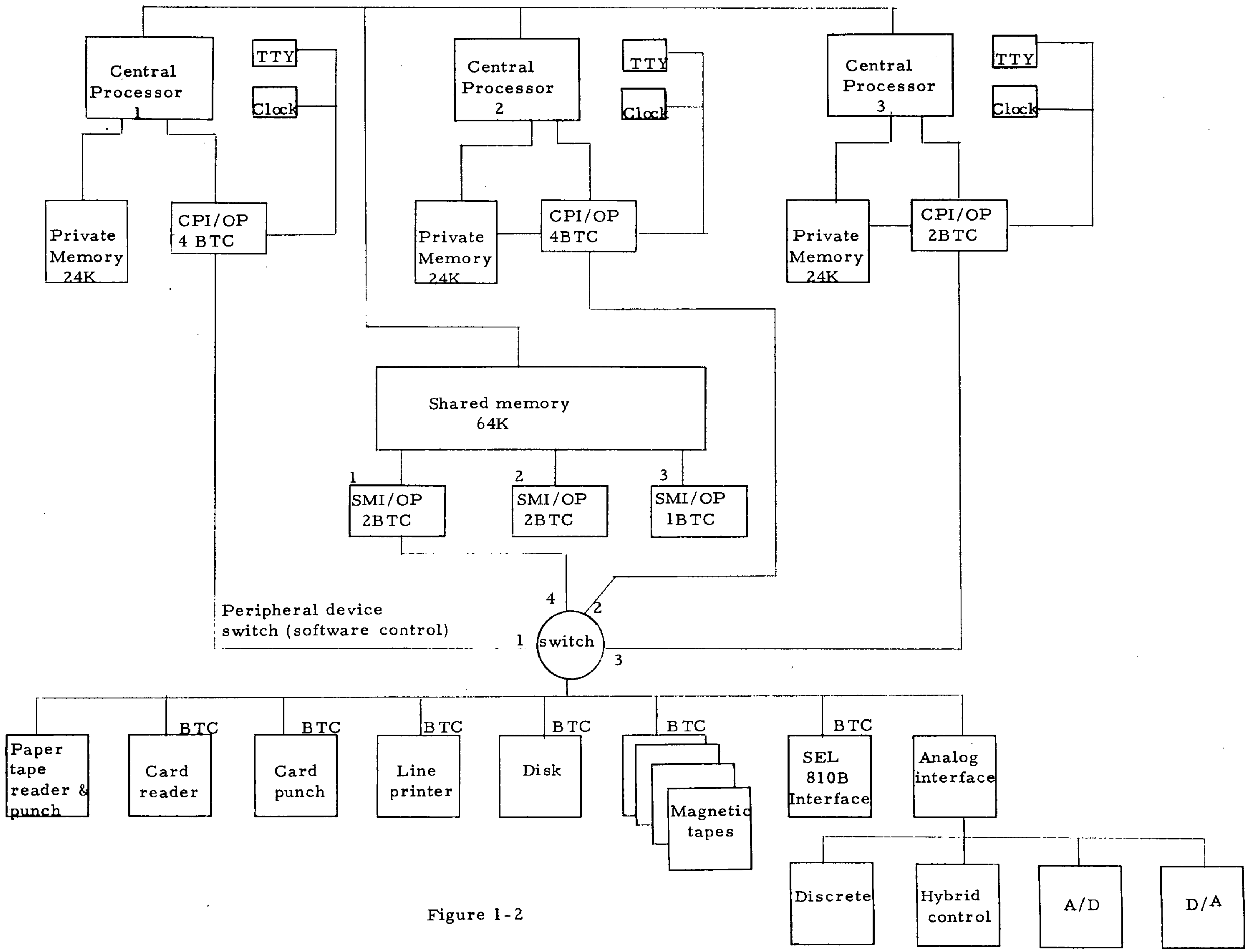


Figure 1-2

to request Input/Output, without regard to where the devices are actually attached. When all devices are attached to a shared memory channel, however, it has to be specified at systeminitialization time. The devices cannot be switched to a shared memory channel without reloading the operating system.

The actual I/O functions that can be performed are identical to those currently provided by CHANE. However, when a task requests an I/O function and specifies WAIT, the task is SUSPENDed by the system, such that other tasks can be executed. The suspended task is rescheduled by the system upon I/O completion.

1.4 Display Support

To enhance the man-machine interface of the SEL 840MP, MPCS provides a display support package. Section 3 provides details for generating application display libraries and Section 5 provides details that will enable users to employ these capabilities in their application programs.

The general procedural flow is depicted in Figure 1-3. The application programmer defines the fixed data of the Display Descriptions required for the application, and adds these off-line to the Display Library. The Display Library is accessed on request of either the application program or the console operator. The application program uses the Display Description to communicate with the console operator and provides variable data for displays whenever required during its execution. The console operator in turn uses the Display Description to communicate with the system through the actions predefined in the Display Description.

In the Display Description, the application programmer may specify any or all of the following.

Fixed Text Data - data to be presented each time the display is used.

Variable Text Data Locations - character positions on the display where variable data provided by the tasks should be displayed.

Compose Field Data Parameters - Compose Field Data is data to be entered from the keyboard. For each compose field used, the following parameters can be provided:

DISPLAY GENERATION
PROCEDURAL FLOW

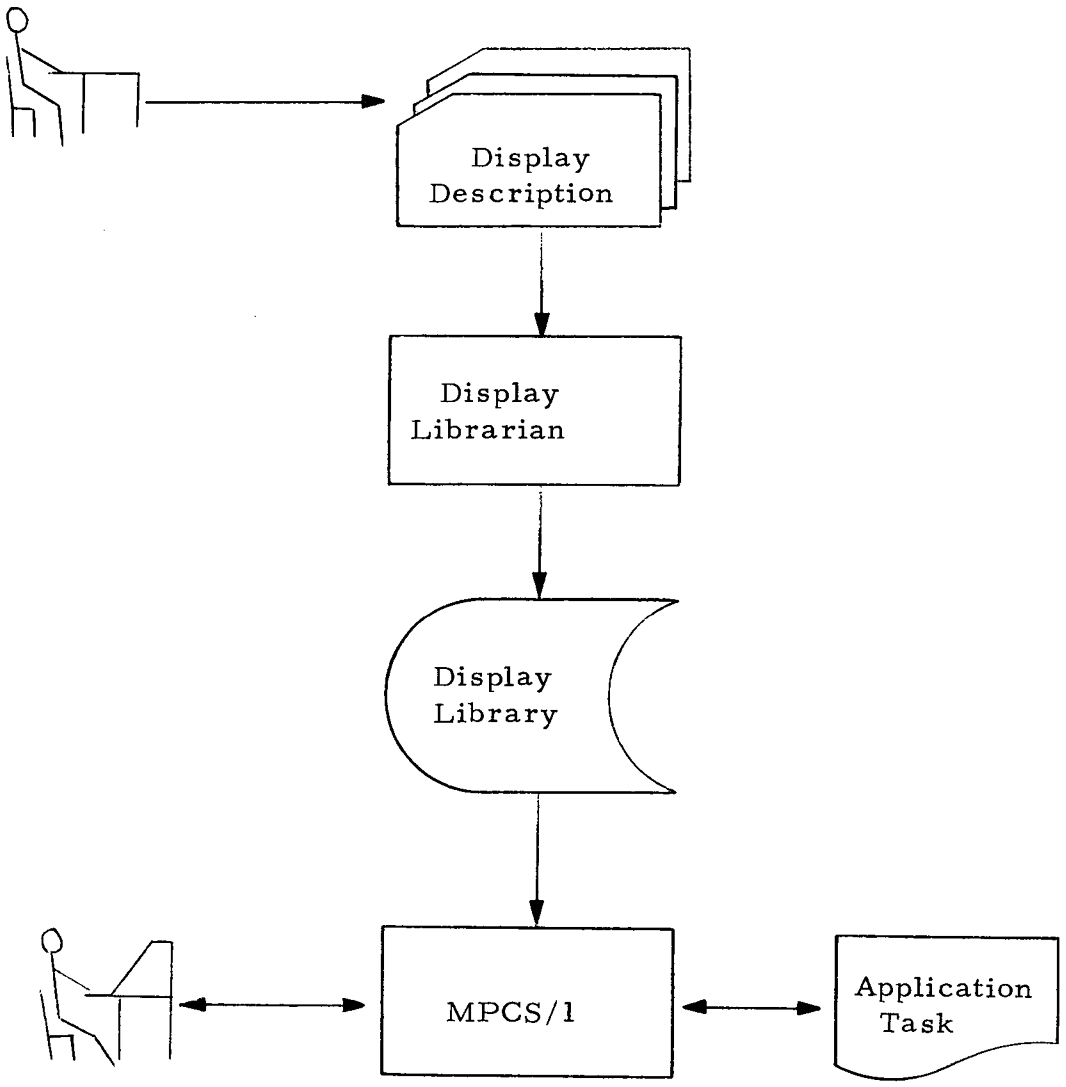


Figure 1-3

- o Position and length of field - to identify a particular compose field.

- o Legality check data - to insure that major errors in data input are detected.

- o Actions - the functions to be performed when the compose field is used by the console operator.

  - Initiate new task and transfer entered data to it.

  - Present a new display.

Photopen Sensitive Areas - positions on the display that signal a specific area if indicated by the photopen.

Resulting actions may be:

- Initiation of a task

- Presentation of new display

In addition to the data and functions defined by the application programmer, each display has a standard two-line "Message Area." One line is used to present an option to return to the previous display. One line is used for error and other messages from the system or from the application program.

An additional feature specified in the package is the support of the Function Switches located on each side of the CRT. Facilities are provided which permit tasks to be attached to, or detached from, the switches such that a task attached to a given switch is scheduled each time the switch is set by the console operator.

2. TASK CONTROL

2.1 General Description

Task control functions define the operational environment provided by MPCS/1. This environment features a design that exhibits an extremely powerful control, and yet is highly adaptable. Tasks specifically designed for the MPCS/1 environment perform most efficiently in that environment. However, performance of the system may also be enhanced for those tasks or functions that have been designed for other operating environments.

Task control functions have been designed for a JOB-TASK hierarchy. Jobs consist of one or more tasks, where a task is defined as a program module consisting of one or more subroutines that may be initiated by MPCS/1 in response to programmer defined specifications. Before loading, a job is not identified or associated with any particular processing unit or other system resource. After being loaded, however, a job is identified to the system by the processor domain in which it resides. This identity is valid for the period of time that the job is resident in that processor. Task division, priority, and definition within a job are the responsibility of the programmer.

When currently implemented programs and functions are visualized as single jobs containing one task, they are immediately and directly compatible with MPCS/1. For these programs, certain restrictions and limitations will exist, both for the program and for MPCS/1. An example of such a restriction would be in the use of the interval timer. It is not possible for both MPCS/1 and a job to directly control this device. Current programs and future tasks requiring direct control of the timer will prohibit MPCS/1 from providing control for that device during those periods when this requirement exists. It will be possible for MPCS/1 to perform during these periods with priority scheduling only. Conflicts will also arise when two separate jobs resident at the same time require the same resource. This or any similar conflict must be resolved by the programmer.

Each task of a particular job has an entry in a central table which has been defined as the Job Task Table (JTT). Unique entries in the JTT are defined as Task Queue Items (TQI). Figure 2-1 illustrates the general configuration and relationship of a job, task, JTT, and TQI. After a job is loaded into a particular processor domain, every element of the job is resident in that domain and is, therefore, accessible by MPCS/1 or any task within the job.

JOB TASK TABLE (JTT)

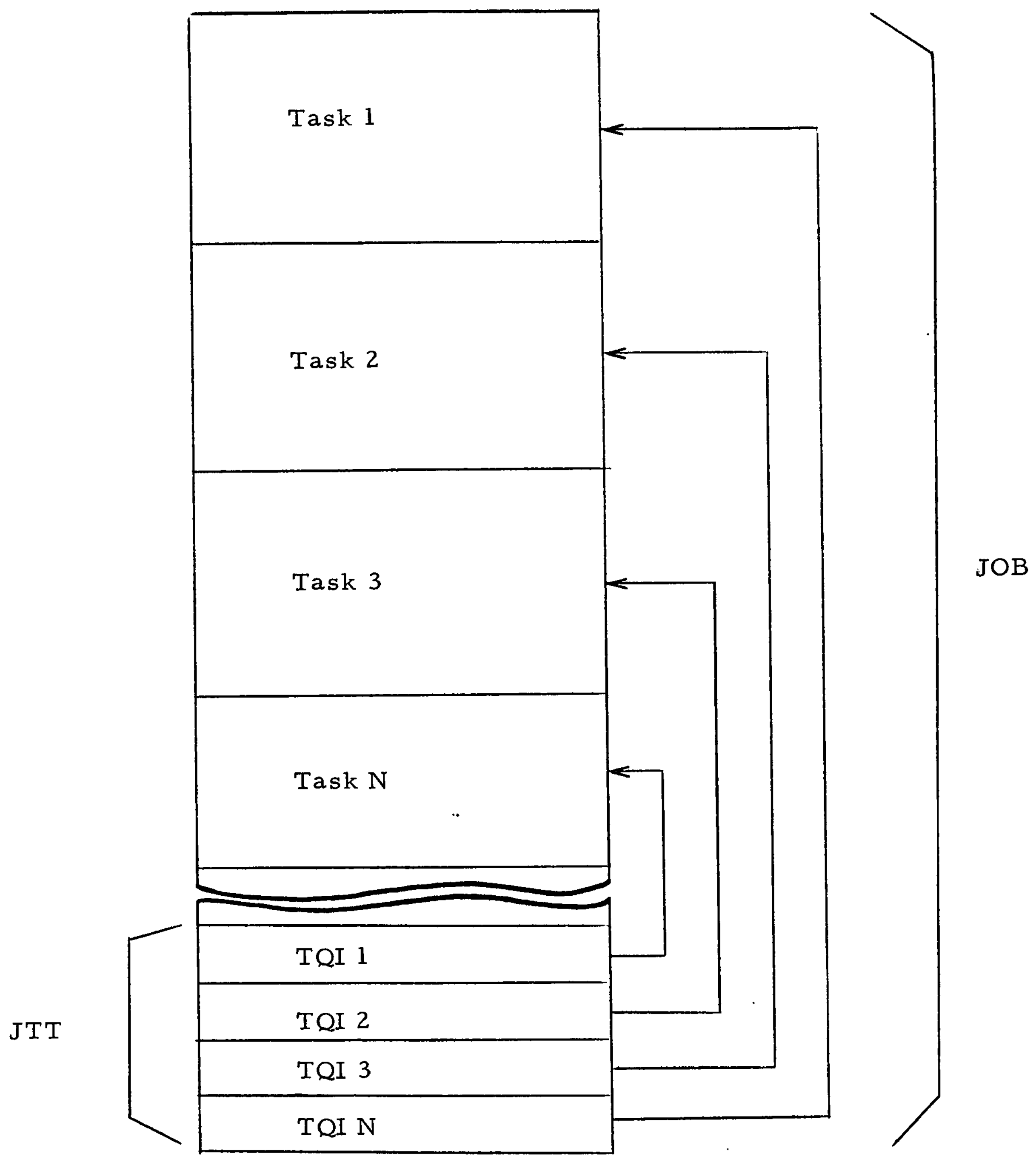


Figure 2-1

Tasks within a job may be designated processor-dependent or processor-independent. Tasks in the processor dependent category specify that, because of certain conditions, they must be executed by the processor in whose domain they reside. Processor-independent tasks, though loaded and identified with a particular job in a particular processor domain, may be executed by any processor. Tasks of this type <u>must</u> be loaded in shared memory.

It should be noted here that the resources required by a job are, in fact, a summation of the resources required by its tasks. Thus, a job containing one processor-dependent task becomes a processor-dependent job. Job definitions should be carefully formulated to insure that when jobs are loaded and executed, system resources will be efficiently utilized.

Task control functions have been designed to automatically distribute processor-independent tasks over available processors. Distribution is accomplished in the following manner. Processors, which for this discussion can be designated 1, 2, and 3, first search for tasks waiting for execution from a list within their own domain. Finding no tasks waiting, the processor moves to a list for another domain. From this list, a processor must first find tasks waiting and then determine if the task type is processor-independent. Finding a processor-independent task waiting, the processor executes that task. Finding no task waiting, the processor moves to the next list. In this manner, a job containing processor-independent tasks loaded into processor 1 domain could have its tasks executed by processor 1, 2, or 3 as they became available.

Figure 2-2 illustrates the three possible mixtures of processor-dependent and processor-independent tasks within a job. In processor domain 1 (which consists of private memory 1 and all of shared memory), a job containing both types of tasks has been loaded. Tasks 6-10 of the job could be executed by processor 1, 2, or 3. Tasks 1-4 of the job residing in processor 2 domain could also be executed by 1, 2, or 3. All tasks of the job residing in processor domain 3 must be executed by that processor since each task resides in private memory. It should be noted that tasks in job 1 and 2 may be restricted to the loaded processor domain simply by designating that each task be of the processor-dependent type. It should also be noted that JTT's are accessible by any processor since they reside in shared memory.

If requirements for a job exceed the capacity of a single processor, some of the tasks within the job may be specifically assigned to another

JOB TASK MIXTURE

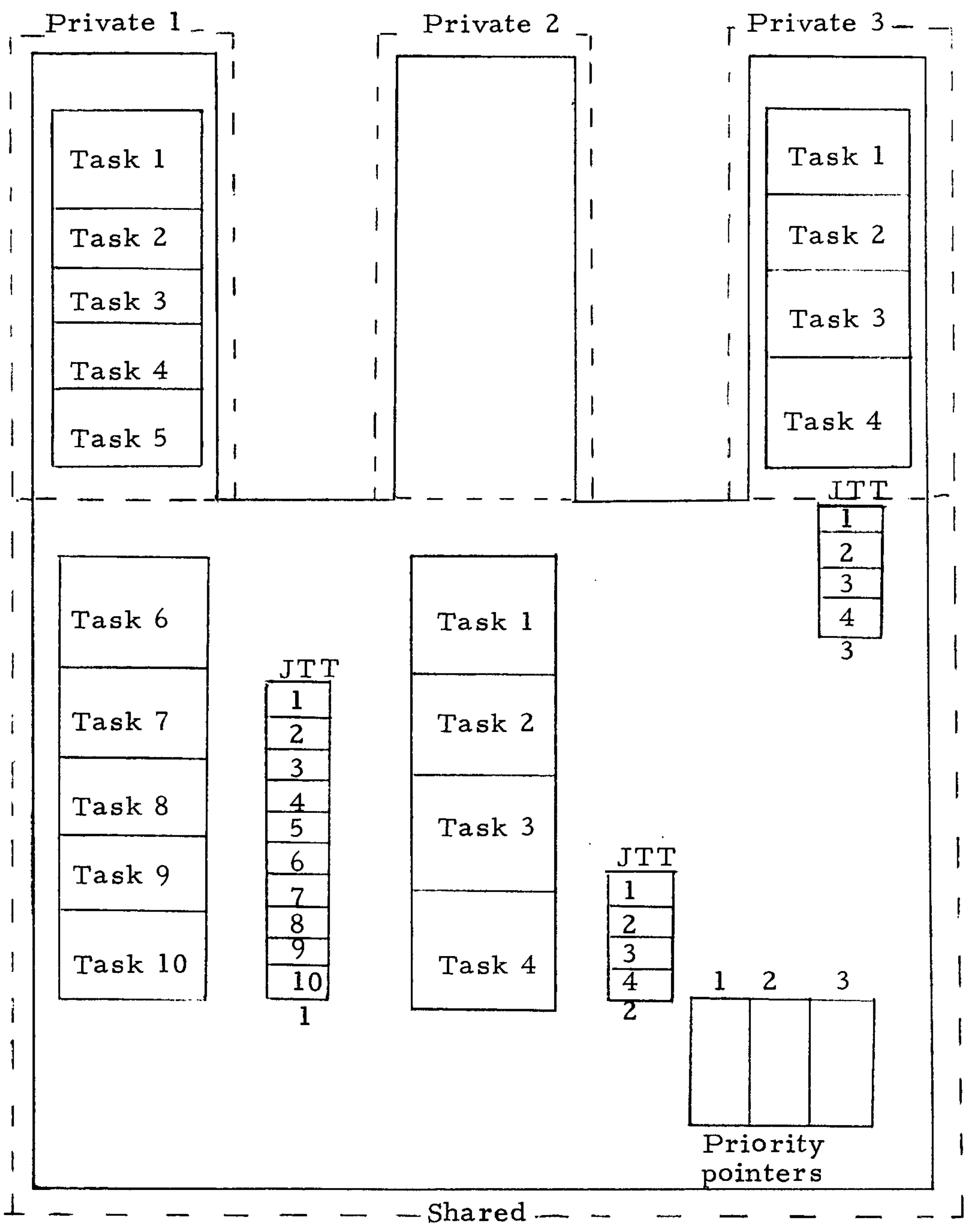


Figure 2-2

processor. An assignment of priority level "16" to a task indicates that it is to be executed by another processor. As tasks of this type are scheduled, cross processor interrupts are issued to the other processors of the system. When the interrupt is received by another processor, any task being executed by that processor is suspended and execution of the priority "16" task is initiated. When all level "16" tasks are completed, the requested processor is released.

2.2 Task Scheduling

To be eligible for execution, a task must be entered in the Priority Schedule Queue. A task is entered in the queue by one of the scheduling functions and remains there until it is either deleted or terminated. Details of all scheduling functions will be discussed in following paragraphs and in Section 4. When a task is entered in a schedule queue, it is linked to those tasks having the same priority. A task is thus eligible for execution based on its priority and its relative position within its priority chain. Execution of a task is initiated when it becomes the highest priority waiting in the Priority Schedule Queue. Tasks of the same priority are executed on a first in, first out basis.

The process of initializing task execution generally proceeds sequentially, starting with higher priority tasks. Once initiated, execution of a task will normally continue until it completes or terminates. Execution of a task will be suspended, however, if tasks of higher priorities are scheduled during its execution. This is possible when a task schedules a higher priority task, when a timer interrupt causes a periodic task to be scheduled, or when an I/O completion is received for a higher priority task.

It is also possible for one processor to force execution of its tasks on another processor. When this occurs, the processor being forced suspends any task it is executing and responds to the schedule queue of the requesting processor. Such forcing occurs whenever a task of priority "16" is scheduled.

Except for forced execution, all tasks scheduled in one processor domain are completed before a task in another processor domain can be executed. The effect of this is that any task scheduled within its own domain is higher in priority than <u>any</u> task in any other domain.

A status for each task scheduled is maintained by MPCS/1. Tasks registered in the Priority Schedule Queue may be in one of the following conditions:

- Executing
- Scheduled, Waiting
- Suspended
- Scheduled, Waiting after Suspension
- Not Scheduled
- Deletion Requested

Executing - This status indicates the task is currently being executed.

Scheduled, Waiting - This status indicates the task is waiting to be executed. Execution of a task is started when it becomes the highest priority waiting in the task schedule queue.

Suspended - This status indicates the task has been voluntarily or involuntarily suspended from execution. Executive calls to MPCS/1 are provided to enable a task to voluntarily suspend itself indefinitely or for a specified length of time. A task may be voluntarily suspended to wait for the completion of an I/O operation. Execution of one task may be involuntarily suspended to execute another task of higher priority. This type of suspension may occur when a higher priority task is scheduled, when an I/O completion is received for a higher priority task, or when forced execution is requested.

Scheduled, Waiting after Suspension - This status indicates the task has ended its voluntary or involuntary suspension. When execution is resumed, it will be at the point of suspension.

Not Scheduled - This status indicates the task is no longer registered in the schedule queue.

Deletion Requested - This status indicates that a delete request has been received while the specified task was either Executing or Suspended. Since a task may be deleted only when it is Scheduled/Waiting, this status enables a task to be deleted automatically whenever it terminates.

MPCS scheduling functions permit complete control of task execution. These enable a task to schedule, suspend, delete, and terminate execution; communicate information; or check status of any task resident in a processor domain. Details of these functions will be presented in following paragraphs.

Scheduling functions provide two basic methods for adding task items to the Priority Schedule Queue. These are Priority Scheduling and Periodic Scheduling.

> Priority Scheduled - Priority Scheduled items are added directly to the Priority Schedule Queue. Each task to be executed must appear in this control queue.
>
> Periodic Scheduled - Periodic Scheduled tasks are scheduled in the Priority Schedule Queue after a specified time increment. Either single or iterative mode may be specified. Periodic Scheduled, single mode tasks are scheduled once for each entry in the periodic task queue. Periodic Scheduled, iterative mode tasks are repetitively scheduled at the specified time increment.

Access to scheduling functions are provided by Executive Calls to MPCS/1. These calls enable the following functions to be performed.

- Schedule Task, Priority - adds designated task to Priority Schedule Queue.

- Schedule Task, Periodic - adds designated task to Periodic Schedule Queue to await the specified time increment. At the end of the increment, the item is added to the Priority Schedule Queue.

- Delete Task - enables a task to be removed from either or both the Periodic or Priority Schedule Queues. A task may be deleted only when scheduled/waiting.

- Terminate Task - indicates to MPCS/1 that a task has completed its execution and it will be removed from the Priority Schedule Queue. If it is a Periodic, Iterative Mode Task, it will remain entered in the Periodic Schedule Queue.

- Schedule Multiple Entry - enables a specific entry from a list to be designated. This entry will receive control when the task is executed. Both Priority and Periodic Scheduling may be requested.

- Suspend Task, Indefinite Time - enables a task to suspend its execution. The suspension must be removed by another task. When execution resumes, it is at the point of suspension.

- Suspend Task, Specific Time - enables a task to be suspended for a specified increment of time.

- Sequential Scheduling - permits one task to be scheduled at the termination of another task.

Details of scheduling functions as well as other task control functions are discussed in detail in Section 4.

2.3 Job Task Table

Every job designed to utilize MPCS task control functions is required to have a Job Task Table (JTT) which defines the job's task structure for the operating system. The JTT contains an entry called a Task Queue Item (TQI) for each task in the job.

A TQI is a table that contains both required and optional parameters, pointers, and flags that enable items to be referenced and maintained by the operating system. Items required by each TQI include:

- Task Name
- Task Type (processor dependent or independent)
- Priority
- Optional Parameter Flags
- Initial Task Entry Point
- Status
- Task Save Area
- Pointer to next TQI

Currently defined optional parameters include:

- Periodic Task Item (required for periodic scheduling)
- Cross-Processor Communication
- Task Communication (required for communication of data from task to task and for tasks that support display functions)
- Identity of next task to be scheduled
- Multiple entry list

MPCS provides facilities for generating the JTT (reference Volume II, Section 3) from programmer supplied TQI definition cards discussed in the following sub-section. However, before TQI definitions are elaborated upon, the physical nature of the JTT and its associated tasks should be thoroughly understood by the programmer.

The JTT, in actuality, is the MAIN program for a job. As such, it must provide for the allocation of FORTRAN COMMON storage in addition to supplying TQI entries for the tasks. Each task is a subroutine or group of subroutines coded in either FORTRAN or Assembly Language. Task entry points are defined in FORTRAN through use of the SUBROUTINE statement and in Assembly Language via a NAME statement which references the task entry point. Note that MPCS will activate the task with an SPB instruction to the task entry point. The task name may or may not be the same as the initial entry point for the task.

When a job is loaded for execution, MPCS automatically schedules the task associated with the first TQI in the JTT. This task can then perform any additional scheduling required by the application.

2.4 JTT Generation Control Cards

There are three basic types of control cards which can be used to define the JTT:

- COMMON allocation cards,
- TQI definition cards, and
- TQI option specification cards.

These cards must be provided by the programmer using the formats illustrated below. They are processed as discussed in Volume II, Section 3 to produce a properly formatted JTT.

2.4.1 COMMON Allocation Control Cards

All COMMON storage requirements for a job must be specified using JTT COM control cards in a manner analogous to that used in a FORTRAN MAIN program. COM control card formats are shown below.

1 4	5 8	9 13	14 80
COM	COMMON Name	COMMON Size	Not Used

Field 1 - COM defines the card type.

Field 2 - Specifies the name of the COMMON block (left-adjusted). Unnamed COMMON is specified using four blanks.

Field 3 - Specifies the size of the COMMON block (right-adjusted with leading zeroes).

All COM cards must be positioned before any of the TQI control cards in the input deck. The COMMON blocks are allocated in the order in which the cards appear.

2.4.2 TQI Definition Control Cards

One TQI definition control card must exist for each task in the job. Each must be immediately followed by the option specification control cards required by the associated task.

1 4	5 8	9 12	13	14 15	16 80
TQI	Task Name	Initial Entry Name	Task Type	Priority	Not Used

Field 1 - TQI defines the card type.

Field 2 - Specifies the task name (left-adjusted).

Field 3 - Specifies the initial entry point for the task (left-adjusted).

Field 4 - Specifies task type:

0 = processor-dependent
1 = processor-independent

Field 5 - Task priority ranging from 01-16. Tasks with a priority of 16 must be processor independent.

2.4.3 TQI Periodic Option Control Card

Periodic tasks require this TQI option for specification of timing parameters.

1 4	5	6	7 11	12 80
TQIP	Mode	Time Units	Execution Time	Not Used

Field 1 - TQIP defines the card type.

Field 2 - Specifies whether the task is to be executed once or repetitively:

0 = single execution
1 = repetitive

Field 3 - Specifies the units for the execution time:

1 = milliseconds
2 = seconds
3 = minutes
4 = hours

Field 4 - Specifies the time at which the task is to be executed (right-adjusted, leading zeroes). For repetitive tasks, it represents the repetition rate. For single-shot tasks, it represents a delay time referenced from the current time. Depending on the time units specified in Field 3, this field is restricted to the following limits:

32767 milliseconds
32767 seconds
2640 minutes
44 hours

2.4.4 TQI Cross-Processor Option Control Card

Tasks which utilize cross-processor communication services require this TQI option for specification of communication buffers.

1 4	5 8	9 12	13 80
TQIC	Buffer Name	Number of Words	Not Used

Field 1 - TQIC defines the card type.

Field 2 - Specifies the name of the communication area (left-adjusted).

Field 3 - Specifies the number of words in the communication area (right-adjusted, leading zeroes).

2.4.5 TQI Inter-Task Communication Option Control Card

Tasks utilizing cross-task communication services require this TQI option for specification of communication buffers. This includes tasks which are scheduled in response to display console operator actions, since the MPCS Display Controller uses cross-task communication to pass display input to the task.

1 4	5 8	9 12	13	14 80
TQII	Buffer Name	Number of Words	Format Type	Not Used

Field 1 - TQII defines the card type.

Field 2 - Specifies the name of the communication area (left-adjusted).

Field 3 - Specifies the number of words in the communication area (right-adjusted, leading zeroes) and is limited to 4095.

Field 4 - Specifies the format of the information to be transmitted or received:

1 = character (6 bits)
2 = half-word (12 bits)
3 = word (24 bits)
4 = double word (48 bits)

2.4.6 TQI Next Task Option Control Card

This TQI option is required by tasks which utilize the sequential scheduling services.

1 4	5 8	9	10 80
TQIN	Next Task Name	Task Type	Not Used

Field 1 - TQIN defines the card type.

Field 2 - Specifies the name of the task to be scheduled for execution following termination of the current task (left-adjusted).

Field 3 - Specifies the type of the task to be scheduled:

0 = priority
1 = periodic

2.4.7 TQI Multiple Entry Option Control Card

Tasks which utilize multiple entry point scheduling services require this control card for each of the entry points in the task. Care should be taken to sequence these control cards properly so that the correct entry point is referenced when it is scheduled by number. The number of entry points for a given task is limited to 64.

1 4	5 8	9 80
TQIE	Entry Point Name	Not Used

Field 1 - TQIE defines the card type.

Field 2 - Specifies the entry point name (left-adjusted).

2.4.8 END Control Card

An END control card must follow the last JTT control card to notify the JTT processor that all cards have been processed. The format is shown below:

1 4	5 80
END	Not Used

3. DEFINING DISPLAYS

Applications utilizing MPCS display support services will require development of Display Descriptions for those applications. In the system, Display Descriptions provide the man-machine interface with the application program. They enable the display console operator to communicate with the application program, and with the system, and they enable the application program to display fixed and variable task data.

The Display Description is a specifically formatted display that is designed and coded by the application programmer. The first step in its development is definition of the data content. The data content for a display is a function of the application and will be determined by the application programmer as he defines his program design. That is, the application programmer will include displays strategically in his program design to enable input of data and control parameters to his program from the display console, and to output program computational results (task data) to the display.

The data which can be included in a display is summarized below. Additional details of this data, such as limitations and restrictions, are given in Section 3. 2.

The data which can be included in a display may be any or all of the following:

- o Text Data - This is the data that will appear on the display whenever the display is used by the application program. It constructs the content of the display and is comprised of character data. The characters which may be used as text characters are any legal 840 ASC II characters except [ ] ↑ ← ! : ; @ $ % & \ . These ASCII characters are used by the 816 hardware to perform certain hardware functions and are not available for use in constructing the display.

- o Task Fill-In Data Locations - Character positions in the display where variable task data output by the application program is to be displayed. These character positions are specified by use of # characters in the Text Data.

- o Compose Field Data Locations - Character positions in the display reserved for data to be entered on-line from the display keyboard. These character positions are specified by use of / characters in the Text Data.

For each compose field specified in the Text Data, the following parameters are also included in the display data: number of compose subfields, length of the compose field, next display, next task, and legality checking data.

When keyboard data is entered into a compose field of a display, one or more of the following actions will be taken by MPCS/1:

- Check the data entered for errors.
- Initiate a new task and transfer the data to it.
- Present a new display.

o Photopen Sensitive Areas - Character positions in the display that signal a specific action when selected on-line by use of the Light Pen. These character positions are specified by use of the ⟨ and ⟩ characters in the Text Data. Characters sensitive to a light pen detect are contained within these two characters. The resulting action taken by the system when a photopen area in the display is selected on-line by use of the Light Pen will be one or more of the following:

  - Continue presentation of the current display.
  - Present a new display.
  - Initiate a new task.

o Line Vectors - Lines specified in the display presented along with the Text Data.

The following subsections give the necessary display data formatting details required to enable the application programmer to design his displays.

3.1 Display Design

First the display data must be formulated using the Display Layout example shown in Figure 3-1. Character positions 1 through 75 and lines 1 through 35 can be used to design the display. Line 36 is reserved for the 'RETURN TO PRIOR LEVEL' option and line 37 is reserved for the application program one-line message option.

An example of a typical display is shown in Figure 3-2 to further illustrate display design. Note in this illustration that each compose field specified in the Text Data is addressed by a Compose card containing the number of subfields, the length of the compose field, next display, and next task data parameters. Likewise, each photopen area specified is addressed by a Pen card containing the next display and next task data parameters.

The number of compose fields which may be included in a display is limited only by the physical space contained therein. That is, as many compose fields as can be contained within the thirty-five lines provided for display construction may be included in a display. A compose field may consist of more than one line of Text Data. In Figure 3-2, the second compose field is an example of this.

The second compose field in Figure 3-2 is also an example of the use of compose subfields. Compose subfields are defined as parts of a compose field separated in the Text Data by a character or characters other than the / character. In this particular example, the compose subfields are contained in different text lines.

The number of photopen sensitive areas (or pen fields) which may be included in a display is limited only by the size of the thirty-five lines provided for display construction. A photopen sensitive area must be contained within one Text Data line.

3.2 Display Coding

Once preliminary design of the display is completed, coding of the display contents may begin. A language which defines data types and associated options of any display is provided to enable easily generated displays. This coding language is the Display Librarian Language (DLL). DLL is an easy to use language consisting of eleven operators.

These operators are:

DISPLAY LAYOUT

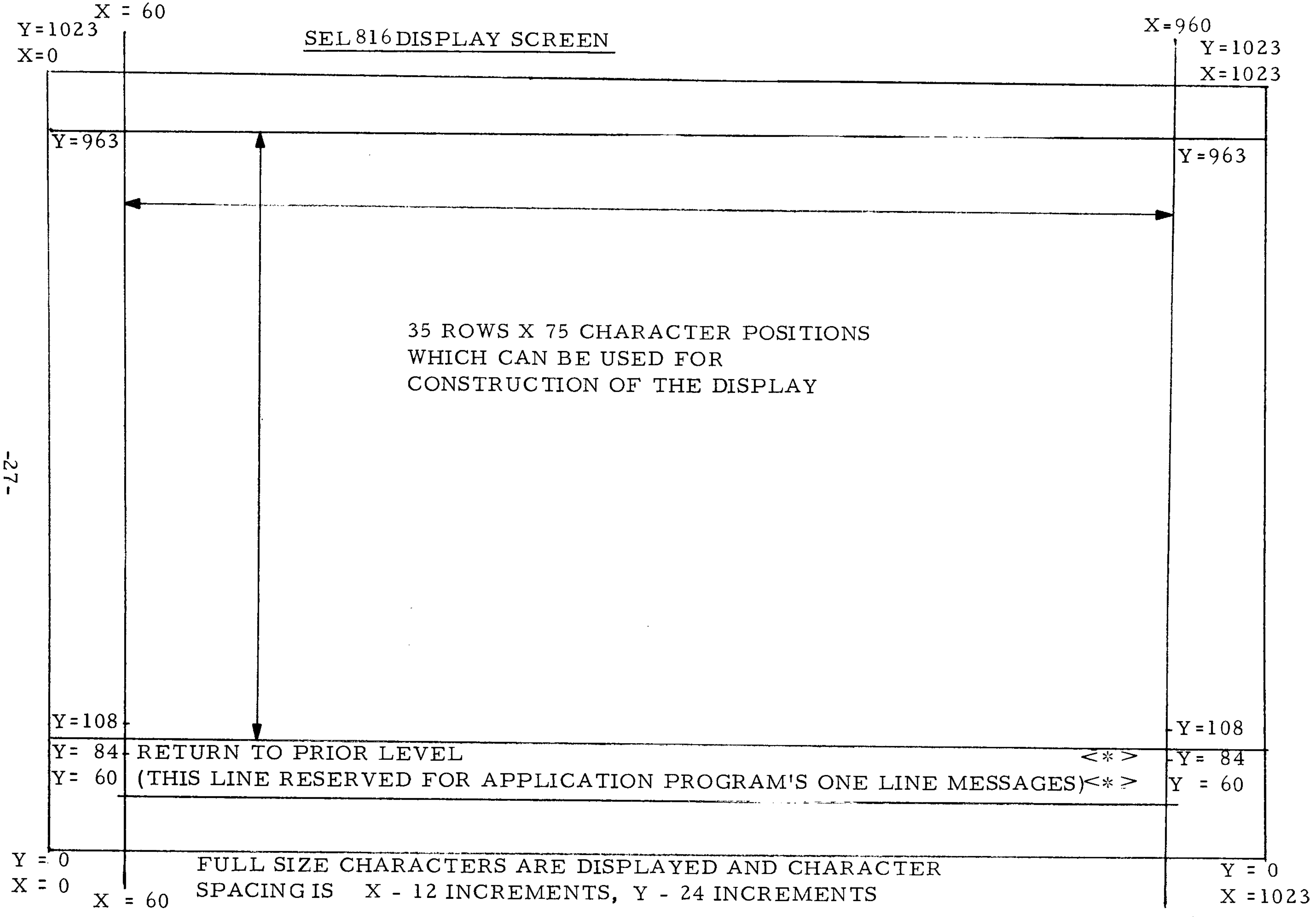

SEL 816 DISPLAY SCREEN
Y=1023
X=0
X = 60
X=960
Y=1023
X=1023
Y=963
Y=963
35 ROWS X 75 CHARACTER POSITIONS
WHICH CAN BE USED FOR
CONSTRUCTION OF THE DISPLAY
Y=108
Y=108
Y= 84
RETURN TO PRIOR LEVEL <*>
Y= 84
Y= 60
(THIS LINE RESERVED FOR APPLICATION PROGRAM'S ONE LINE MESSAGES)<*>
Y = 60
Y = 0
X = 0
X = 60
FULL SIZE CHARACTERS ARE DISPLAYED AND CHARACTER
SPACING IS X - 12 INCREMENTS, Y - 24 INCREMENTS
Y = 0
X =1023
X = 960

Figure 3-1

TYPICAL DISPLAY

```
N,0040
C,1,8,SAME,SIM1
X(3D)(1S.)(4D)
C,2,16,SIM1,PRTD
X(3D)(1S.)(4D)(3D)(1S.)(4D)
P,SAME,INIT
P,0050,SAME
*T
T,DISPLAY 0040            AS510 SI-C SIMULATIØN
T,
T,    TIME REFERENCE ////////
T,
T,  REQUEST DETAIL PRINT   FRØM ////////
T,                           TØ ////////
T,
T,  PITCH
T,
T, DELTA THETA ########  THETA DØT ########  THETA DØUBLE DØT ########
T, BETA C ########
T, BETA 1 ########  BETA 2 ########  BETA 3 ########  BETA 4 ########
T,
T,  YAW
T,
T, DELTA THETA ########  THETA DØT ########  THETA DØUBLE DØT ########
T, BETA C ########
T, BETA 1 ########  BETA 2 ########  BETA 3 ########  BETA 4 ########
T,
T,  RØLL
T,
T, DELTA THETA ########  THETA DØT ########  BETA C ########
T,
T,
T, BETA T ########  ALPHA T ########
T,   VW Y ########     VW Z ########    MACH ########     ALT ########
T,   MASS ########     FT Z ########    FT Y ########       Q ########
T,
T,
T, INITIALIZE DISPLAY FØR NEW TIME SELECT  <INIT>
T,
T, SELECT SIM2 DISPLAY  <0050>
END
```

Figure 3-2

TYPICAL DISPLAY

(continued)

```
                              DISPLAY   40

*******************************************************************--------NEXT DISPLAY NAMES---------
*                                                                 *
*    DISPLAY 0040            AS510 SI-C SIMULATIØN                *
*                                                                 *
*       TIME REFERENCE ////////                                   *SAME
*                                                                 *
*      REQUEST DETAIL PRINT   FRØM ////////                       *
*                                 TØ ////////                     *SAME
*                                                                 *
*      PITCH                                                      *
*                                                                 *
*     DELTA THETA ########  THETA DØT ########  THETA DØUBLE DØT ######## *
*     BETA C ########                                             *
*     BETA 1 ########  BETA 2 ########  BETA 3 ########  BETA 4 ######## *
*                                                                 *
*      YAW                                                        *
*                                                                 *
*     DELTA THETA ########  THETA DØT ########  THETA DØUBLE DØT ######## *
*     BETA C ########                                             *
*     BETA 1 ########  BETA 2 ########  BETA 3 ########  BETA 4 ######## *
*                                                                 *
*      RØLL                                                       *
*                                                                 *
*     DELTA THETA ########  THETA DØT ########  BETA C ########   *
*                                                                 *
*                                                                 *
*     BETA T ########  ALPHA T ########                           *
*       VW Y ########     VW Z ########    MACH ########     ALT ######## *
*       MASS ########     FT Z ########    FT Y ########       Q ######## *
*                                                                 *
*     INITIALIZE DISPLAY FØR NEW TIME SELECT  <INIT>              *SAME
*                                                                 *
*     SELECT SIM2 DISPLAY  <0050>                                 *0050
*                                                                 *
*                                                                 *
*    RETURN TØ PRIØR LEVEL                                   <*>  *PREV
*    (THIS LINE RESERVED FØR APPLICATIØN PRØGRAM'S ØNE LINE MESSAGES)  <*>  *SAME
*                                                                 *
*******************************************************************
```

Figure 3-2
(continued)

- o INPUT OPTION
- o OUTPUT OPTION
- o NAME
- o PEN
- o COMPOSE
- o LEGALITY
- o LINE
- o TEST CONTROL
- o TEXT
- o END
- o DELETE

These DLL source records are input to the Display Librarian Program on 80-column punched cards. Columns 1 through 72 will contain the input data and each record will contain an indentifier character in column 1.

3.2.1 INPUT OPTION Card (I)

I, T	- Input normally is from cards. If, however, card image magnetic tape input is desired, this option card is used. Column 1 contains the identifier "I", column 2 contains a comma, and column 3 contains the character "T". The input tape is mounted on magnetic tape drive 2.

3.2.2 OUTPUT OPTION Card (O)

O, S, P, L	- Column 1 contains the identifier "O" followed by a comma in column 2. The output options are punched beginning in column 3. The output options are: "S"-list the source cards; "P"-print the formatted display(s) on the line printer; and "L"-write (add) the display(s) to the display library.

If the OUTPUT OPTION card is not included in the input deck, the Display Librarian Program will assume the "P" option was selected.

When the "L" option is selected, that display or displays will be written to the display library only if no errors are detected in the source card input. If an error is detected, that display will be printed, and if the "S" option was selected, the source records will be listed.

All errors detected in the source records by the Display Librarian Program will result in error messages being printed, regardless of which output options were selected.

3.2.3 NAME Card (N)

N, XXXX - Column 1 contains the identifier "N" followed by a comma in column 2. The Display Name Tag is contained in columns 3 through 6. Display Name Tags can be any four digit numbers in the range of 0001 through 9999.

The Display Name Tag of 0001 must be specified by the user as the first display in the display library. The Display Librarian Program will not begin compilation of the display library until a Display Name Tag of 0001 is detected. Display Name Tags of 0002 through 9999 are then used for all subsequent displays to be included in or added to the display library.

Display Name Tags of 0001 through 8999 are used for normal displays and 9000 through 9999 are used for "special message displays" (see Section 3.2.9 for explanation of special message display).

3.2.4 PEN Card (P)

P, XXXX, YYYY - Column 1 contains the identifier "P" followed by a comma in column 2. Columns 3 through 6 contain the Next Display Name. The Next Display Name may be any legal Display Name Tag (see Section 3.2.3) or "SAME" or "PREV". This defines all of the required data for the PEN card.

There is, however, an optional data field. This data field contains the Next Task Name, where the Next Task Name can be any legal

MPCS/1 Task Name (reference Sections 2 and 4). The Next Task Name is one to four alphabetic characters beginning in column 8 of the card.

PEN cards are required only if there are pen fields defined in the text data (Section 3.2.9) for the display. The Next Display Name and Next Task Name designate the next display and next MPCS/1 task to be displayed and executed, respectively, when the associated pen field of the display is selected by the light pen. "SAME" or "PREV" in the Next Display Name data field indicates that the same or the previous display is to be presented when the associated pen field in the display is selected.

The PEN cards must appear in the input deck in the same order as the associated pen fields appear in the text data input.

3.2.5 COMPOSE Card (C)

C, VV, WW, XXXX, YYYY - Column 1 contains the identifier "C" followed by a comma in column 2. The four required data fields begin in column 3, and they are: number of compose subfields - one or two digit number; number of compose characters - one or two digit number; Next Display Name; and Next Task Name.

COMPOSE cards are required to be input only if there are compose fields defined in the text data (Section 3.2.9) for display. The Next Display Name and Next Task Name specify the next display and next MPCS/1 task to be displayed and executed, respectively, when the associated compose field in the display is filled by keyboard data. "SAME" or "PREV" can be used for Next Display Name which specifies that the same or the previous display is to be presented in response to the compose field data.

The COMPOSE cards must appear in the input deck in the same order as the associated compose fields appear in the text data.

3.2.6 LEGALITY Card (X)

X(L1T1R1)(L2T2R2)---(LNTNRN) - Column 1 contains the identifier "X" followed by the legality data beginning in column 2. "L" is the length of the legality subfield, where a legality subfield is that data enclosed by parentheses in the format example shown above. "T" is the type of the restriction data. "T" can be any of the following:

"O" - octal,
"B" - binary,
"D" - decimal,
"A" - alphabetic,
"X" - no checking,
"S" - special characters.

The special characters allowed for keyboard entry to a compose field character position when "S" checking is specified for that character position are: ƀ " # ' ( ) * = - +. , ? / < and >. "R" is the restriction data and can be explicit magnitudes, magnitude ranges, or combinations of both.

An additional example of the LEGALITY card format is shown below to show how the legality data for a typical compose field might be specified:

```
Compose field:   / / / / / / b̄b̄b̄b̄/ / / / /b̄b̄b̄/ / /
                 A A A A B B      C C C C C   D D D
```

where:

A = Decimal Display Name Tag
B = Binary ID Code
C = Octal Unit Address Code
D = Alphabetic End Key

LEGALITY Card: X(4D0001-0005,0010)(2B)(5O10077)(3AEND)

The first subfield of this LEGALITY Card specifies that when four characters of data are entered via the keyboard to the first four character positions of the associated compose field, they must be decimal characters in the range of 0001 through 0005 or the explicit decimal character sequence 0010. Likewise, the second subfield must be two characters of valid binary data of any possible combination 00, 01, 10, or 11. The third subfield will accept only the five character octal number 10077. The fourth subfield will only accept the three alphabetic characters END.

LEGALITY cards are optional. If included in the input deck, a LEGALITY card must immediately follow the COMPOSE card to which it applies. If no LEGALITY card is input with a COMPOSE card, no legality checking (validation) will be performed by the Display Processor for keyboard data input to this compose field.

LEGALITY cards can require more than 71 card columns for data. To accommodate this, a continuation card is allowed and is indicated by a non-blank character in column 72 of the LEGALITY card. The column 72 character is not interpreted as a data character by the Display Librarian Program, and is used only to indicate that there is a continuation card for this record. The LEGALITY continuation card contains the identifier "X" in column 1 followed by the continued legality data beginning in column 2.

3.2.7 LINE Card (L)

There are two types of line format specifications allowed. The first specifies a Line Vector to be included in the display and is specified in terms of display screen coordinates:

L, C, IIII, JJJJ, KKKK, LLLL - Column 1 contains the identifier "L" followed by a comma in column 2. Column 3 contains the character "C" followed by a comma in column 4. The four line specification data fields begin in column 5 and each can be a one to four digit number in the range of 0 through 1023. They are:"From" X ordinate, "From" Y ordinate, "To" X ordinate, and "To" Y ordinate.

The second line format specification specifies a Text Underline to be included in the display and is specified in terms of text character and line numbers:

L, L, MM, NN, OO, PP - Column 1 contains the identifier "L" followed by a comma in column 2. Column 3 contains the character "L" followed by a comma in column 4. The four line specification data fields begin in column 5 and each can be a one or two data number in the range of 1 through 75 for the character numbers, and 1 through 35 for the line numbers. They are: "From" Text character number, "From" Text line number, "To" Text character number, and "To" Text line number.

3.2.8 TEXT CONTROL Card (*)

T - Column 1 contains the identifier "" followed by the character "T" in column 2.

This control card is used by the Display Librarian Program to initiate TEXT card processing and validation processing of previous control cards. It appears in the input deck immediately following all input card types thus far discussed, and immediately preceding the first TEXT card.

3.2.9 TEXT Card (T)

T, JJJJJ. . . . . JJJ - Column 1 contains the identifier "T" followed by a comma in column 2. The text characters are punched in columns 3 through 71.

TEXT cards contain the data that is to appear on the display. They also contain data areas on the display defined as pen fields contained within < and > characters; compose fields and subfields which are defined by / characters, and task fill-in fields which are defined by # characters. All TEXT cards must follow the TEXT CONTROL card in the input deck and must be in the order that the text is to appear on the display. A TEXT record can contain a maximum of seventy-five text characters. To accommodate this, a TEXT continuation card is allowed and is indicated by a non-blank character in column 72 of the TEXT card. The TEXT continuation card contains the identifier "T" in column 1 followed by a comma in column 2, and followed by the continued text data beginning in column 3.

Text characters may be any legal 840 ASCII characters except [ ] ↑ ← ! : ; @ $ % &\. These ASC II characters are used by the 816 hardware to perform certain hardware control functions and will be detected by the Display Librarian Program as an Illegal Text Data Character when they are included in the text data.

A maximum of thirty-five TEXT records are allowed for a normal display. A maximum of twelve TEXT records are allowed for a "special message display". The special message display is used only when the application program needs to output a message that is longer than seventy-five characters. It may contain no pen, compose, or task fill-in fields. It cannot be updated in on-line execution as a normal display.

3.2.10 END Card (E)

END -	Contains the identifier "E" in column 1 followed by the characters N and D in columns 2 and 3. This operator indicates end of the input data for a display.

3.2.11 DELETE Card (D)

D, XXXX -	Contains the identifier "D" in column 1 followed by a comma in column 2. The Display Name Tag of the display to be deleted from the display library is contained in columns 3 through 6.

3.3 Display Data Input Deck Setup

The input deck setup shown in Figure 3-3, details the input data set required for one display. Input data sets for any number of displays can be stacked contiguously and processed in a single run. When multiple input data sets are stacked, the EOJ card is included only after the END card for the last display in the run.

The DELETE card is a special case and the input data set required to delete a display from the display library is comprised of only the DELETE card itself.

DISPLAY DATA INPUT DECK SETUP

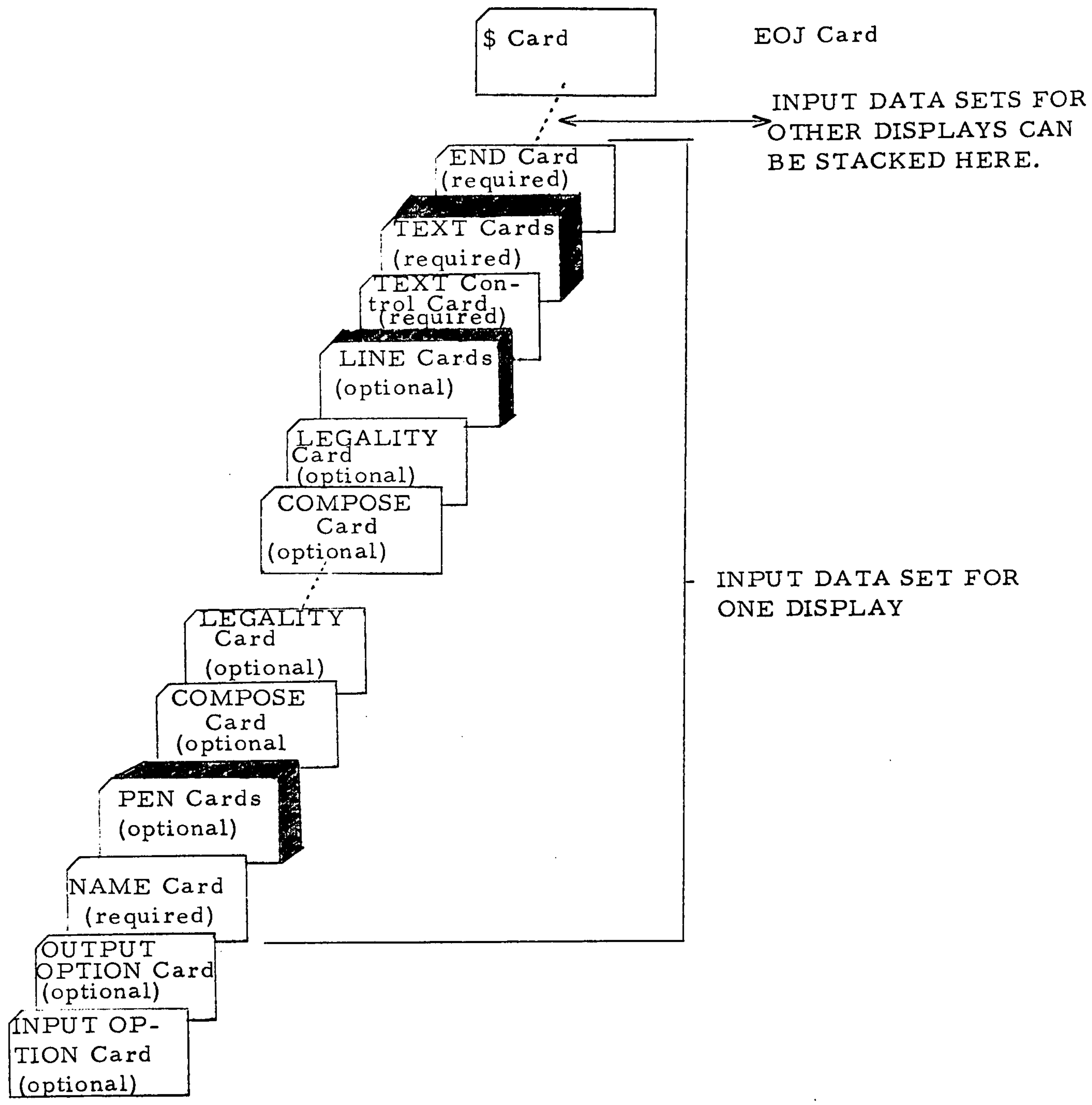


Figure 3-3

PRECEDING PAGE BLANK NOT FILMED

4. MPCS CALLS FOR TASK CONTROL FUNCTIONS

MPCS calls for task control functions provide the capability to schedule, suspend, delete, terminate or check status of tasks designed to be controlled by MPCS/1. The call sequences for the task control functions have been designed so that they may be easily implemented for tasks written in either FORTRAN or assembly language. Each MPCS task control function has been assigned a unique system name. The general format of the name is:

M$NN

where:

M$ are the two characters reserved by the system for exclusive use by MPCS calls. Users should not use these two characters as the first two characters of a FORTRAN subroutine name or or of a six character symbolic external name.

NN is a two character number that specifies the particular task control function.

Figure 4-1 provides details of task control functions and their associated symbolic assignments.

FORTRAN CALLS

The syntax of the FORTRAN call sequence for MPCS task control functions is as follows:

```
CALL  M$NN  (PARAMETER 1, PARAMETER 2, ...PARAMETER N)
```

where:

M$NN (M$01-M$15) specifies a particular task control function and PARAMETER 1 - PARAMETER N specify arguments required by the particular function.

Generally, the first argument in a task control function call is the name of the task for which the particular service is to be performed. The last two arguments are generally the error code address and the error return statement number. Figure 4-2 provides a summary of all assigned error codes. Error return statement numbers may be coded as follows:

PRECEDING PAGE BLANK NOT FILMED

TASK CONTROL FUNCTION SYMBOLIC NAMES

EXTERNAL NAME	TASK CONTROL FUNCTION	PAGE
M$01	Schedule task, periodic	45
M$02	Schedule task, priority	46
M$03	Set pointer to task TQI	47
M$04	Initialize Task Communications pointers	48
M$05	Delete Task	49
M$06	Terminate Task	50
M$07	Check Status of Task	50
M$08	Schedule Priority Task, Multiple Entry	51
M$09	Schedule Periodic Task, Multiple Entry	53
M$10	Suspend Task, Indefinite	55
M$11	Specify Next Task Name	56
M$12	Suspend Task, Specified Time	57
M$13	Task Communications	58
M$14	Connect Task to Interrupt	64
M$15	Cross Processor Communications	65

Figure 4-1

TASK CONTROL ERROR SUMMARY

FUNCTION	CODE	EXPLANATION
SCHEDULE PERIODIC TASK	1	Task not in JTT
	2	Task is scheduled in periodic queue
	3	Periodic parameter not present in TQI
	6	Increment type error
	7	Task is active in priority queue (no error return)
SCHEDULE PRIORITY TASK	1	Task not in JTT
	2	Task is scheduled in priority queue
	3	Invalid priority
SET POINTER TO TASK TQI	1	Task not in JTT
	3	Optional parameter not present
INITIALIZE TASK COMMUNICATIONS POINTERS	1	Task not in JTT
	3	Optional parameter not present in TQI
DELETE TASK	1	Task not in JTT
TERMINATE TASK		NONE
CHECK STATUS OF TASK	1	Task name not in JTT
SCHEDULE PRIORITY TASK, MULTIPLE ENTRY	1	Task name not in JTT
	2	Task is scheduled in priority queue
	3	Parameter not present in TQI
	4	Specified entry not valid

Figure 4-2

TASK CONTROL ERROR SUMMARY
(continued)

FUNCTION	CODE	EXPLANATION
SCHEDULE PERIODIC TASK, MULTIPLE ENTRY	1	Task name not in JTT
	2	Task is scheduled in periodic queue
	3	Periodic parameter not present in TQI
	4	Specified entry not valid
	6	Increment type error
	7	Task is active in priority queue (no error return)
SUSPEND TASK, INDEFINITE		NONE
SPECIFY NEXT TASK NAME	1	Task name not in JTT
	3	Parameter not present in TQI
SUSPEND TASK, SPECIFIED TIME		NONE
TASK COMMUNICATIONS	1	Task name not found in JTT
	2	Task is active (executing)
	3	Parameter not present in TQI
	4	Requested task has not executed since last communication
	5	Limit of information has been exceeded

Figure 4-2
(continued)

TASK CONTROL ERROR SUMMARY
(continued)

FUNCTION	CODE	EXPLANATION
CONNECT TASK TO INTERRUPT	1	Task name not found in JTT
	3	Periodic parameter notpresent
	5	Incompatible task type
	6	Invalid interrupt level
CROSS PROCESSOR COMMUNICATIONS		NONE

Figure 4-2
(continued)

```
$NNN
```

where:

the statement number (NNN) is the statement that is to be executed in response to an error condition and is precedure by a $ character.

Call sequences for those functions that require no arguments have the following syntax:

```
CALL  M$NN
```

ASSEMBLY LANGUAGE CALLS

Assembly language call sequences have the following syntax:

```
SPB    $M$NN
DATA  PARAMETER 1
DATA  PARAMETER 2
   .
   .
   .
DATA  PARAMETER N
```

where:

$M$NN specifies the external name of the particular task control function.

PARAMETER 1 - PARAMETER N are the addresses of the arguments required by the particular function.

Call sequences requiring no arguments take the form:

```
SPB  $M$NN
```

The following subsections provide details for all task control functions. Each subsection provides call sequence requirements, including function names and parameters required, and details of error codes and error code definitions. Examples illustrate typical FORTRAN and Assembly Language call sequences.

4.1 Schedule Task, Periodic (M$01)

This function provides the capability to schedule a periodic task. Six parameters are required for this MPCS call:

Parameter 1 - the address of the name of the task to be scheduled.

Parameter 2 - Address of a location containing an integer that designates the unit of time, where the integers have been assigned as follows:

1 - designates milliseconds
2 - designates seconds
3 - designates minutes
4 - designates hours

Parameter 3 - Address of a location containing an integer that specifies the increment of time in the units specified by Parameter 2 above.

Parameter 4 - Address of a location containing an integer which may be either 1 or 0. 1 designates an iterative periodic task (one that is executed repetitively at the specified period) and 0 designates a single execution at the specified period of time where the time of scheduling assumes a time reference of zero. A task scheduled in the iterative mode (Parameter 4 equals 1) having a period of 40 milliseconds would be executed every 40 milliseconds. If the same task were scheduled in the single mode, it would be executed once 40 milliseconds from the time of scheduling.

Parameter 5 - Address that will receive any error codes returned. Error codes have been assigned as follows:

1 - Requested task is not resident
2 - Requested task is currently scheduled
3 - Periodic parameter is not present for requested task
6 - Error in specifying increment of time
7 - Task is active in priority scheduling queue

Error codes 1, 2, 3, and 6 cause an error return; error code 7 does not.

Parameter 6 - Statement number or address that receives control for any error return.

Examples:

FORTRAN

```
CALL  M$01(4HTS02,2,10,1,IRC,$200)
```

ASSEMBLY LANGUAGE

```
SPB     $M$01
DATA    ="TS02"
DATA    =2
DATA    =10
DATA    =1
DAC     IRC
DAC     ERR
```

The above examples schedule task TS02 in the iterative mode for a period of execution of 10 seconds. Symbolic address IRC specifies the error code address and statement 200 or symbolic location ERR will receive control if error conditions are detected.

4.2 Schedule Task, Priority (M$02)

This function enables a priority task to be scheduled for execution. Three parameters are required for this function call:

Parameter 1 - the address of the name of the task to be scheduled.

Parameter 2 - the address that is to receive any error codes. Error codes have been assigned as follows:

1 - Requested task is not resident
2 - Task is currently scheduled
3 - Requested task has been assigned an invalid priority. Priorities may range from 1 to 16.

Parameter 3 - Statement number or location that is to receive control for any error return.

Examples:

FORTRAN

```
CALL  M$02(4HTS03,IRC, $210)
```

ASSEMBLY LANGUAGE

```
SPB     $M$02
DATA    ="TS03"
DAC     IRC
DAC     ERR
```

The above examples schedule task TS03 in the priority queue. Location IRC will receive any error codes if errors are detected. Statement 210 and location ERR will receive control if error conditions are detected.

4.3 Set Pointer to Task TQI (M$03)

This function enables an application task to locate its Task Queue Item (TQI). Any of the parameters contained within the TQI may be examined by the task but may not be altered in any way since the TQI's reside in protected memory. Four parameters are required for this MPCS function:

Parameter 1 - Location containing the address of the name of the requested task.

Parameter 2 - Location containing an integer that designates the control or initial entry within the TQI or one of the five optional parameters which may be contained within the TQI. The integers have the following designation:

0 - initial TQI pointer (start of the TQI)
1 - optional parameter one (Periodic Task Item)
2 - optional parameter two (Task Data Save Area)
3 - optional parameter three (Task Communication Option)
4 - optional parameter four (Name of Next Task to be Scheduled)
5 - optional parameter five (TQI Entry List)

Parameter 3 - Location that is to receive the address or pointer to the TQI requested. If errors are detected, this location receives the error code. Assigned error codes are as follows:

1 - Requested task is not resident
3 - Requested optional parameter is not contained in the TQI of the specified task.

Parameter 4 - Statement number or address that receives control for any error return.

Examples:

FORTRAN

```
CALL  M$03 (4HTEST, 3, IRC, $201)
```

ASSEMBLY LANGUAGE

```
SPB     $M$03
DATA    ="TEST"
DATA    =3
DAC     IRC
DAC     ERR
```

The above examples request the location of Parameter 3 of the TQI of task TEST. Location IRC will receive either the TQI pointer or error codes if error conditions are detected. Statement 201 and location ERR will receive control if error conditions are detected.

4.4 Initialize Task Communications Pointers (M$04)

This function enables a task containing a communications area to initialize the data pointers and counters contained within the optional TQI parameter for the area. Three parameters are required for this MPCS function:

Parameter 1 - Location containing the address of the name of the requested task.

Parameter 2 - Location that is to receive any error code. Error codes have been assigned as follows:

1 - Requested task is not resident
3 - Task communications parameter is not contained within the requested task.

Parameter 3 - Statement number or address that receives control for any error return.

Examples:

FORTRAN

```
CALL  M$04 (4HTASK, IRC, $220)
```

ASSEMBLY LANGUAGE

```
SPB     $M$04
DATA    ="TASK"
DAC     IRC
DAC     ERR
```

The above examples illustrate that the task named TASK will have its communications area parameters initialized. Location IRC will receive error codes for any detected errors and statement 220 and location ERR will receive control if error conditions exist.

4.5 Delete Task (M$05)

This function enables a task to delete itself or another task from the schedule queues. In particular, this function is designed to remove iterative mode periodic tasks from the periodic schedule queue. The necessity for this function exists because as an iterative mode task executes to termination, it remains registered in the periodic queue, and unless deleted, will execute at the specified scheduling period indefinitely. Three parameters are required for this function:

Parameter 1 - Location containing address of the requested task name.

Parameter 2 - Location that is to receive any error code, where the only assigned error code is:

1 - Requested task is not resident

Parameter 3 - Statement number or address that receives control for any error return.

Examples:

FORTRAN

```
CALL M$05 (4HTST1,IRC,$230)
```

ASSEMBLY LANGUAGE

```
SPB     $M$05
DATA    ="TST1"
DAC     IRC
DAC     ERR
```

These examples request that task TST1 is to be deleted. Location IRC receives any detected error codes. Statement 230 and location ERR receive control when error conditions are detected.

4.6 Terminate Task (M$06)

This function enables a task to terminate execution. This is the normal task exit as a task completes execution. No parameters are required by this function.

Examples:

FORTRAN

```
CALL  M$06
```

ASSEMBLY LANGUAGE

```
SPB   $M$06
```

These examples illustrate the MPCS calls to terminate execution of a task.

4.7 Check Status of Task (M$07)

This function enables one task to check the status of another task. Three parameters are required for this function:

Parameter 1 - Location containing address of the name of the requested task.

Parameter 2 - Location that will receive the status code or, if the requested task is not resident, will receive the error code. The error return will be taken if an error is detected. Status codes for each task are maintained by MPCS and are required parameters in each task TQI. Status codes are maintained in bits 0 - 5 of the second word of each TQI. Status codes are returned to the user as right-adjusted integers (bits 18-23) in the specified location. Assigned status codes are illustrated in Figure 4-3. There is one possible error code, 1, which denotes that the requested task is not resident. If present, the error code is set in the designated location.

Parameter 3 - Statement number or location that receives control when an error condition is detected.

Examples:

FORTRAN

```
CALL M$07 (4HTEST, IRC, $210)
```

ASSEMBLY LANGUAGE

```
SPB     $M$07
DATA    ="TEST"
DAC     IRC
DAC     ERR
```

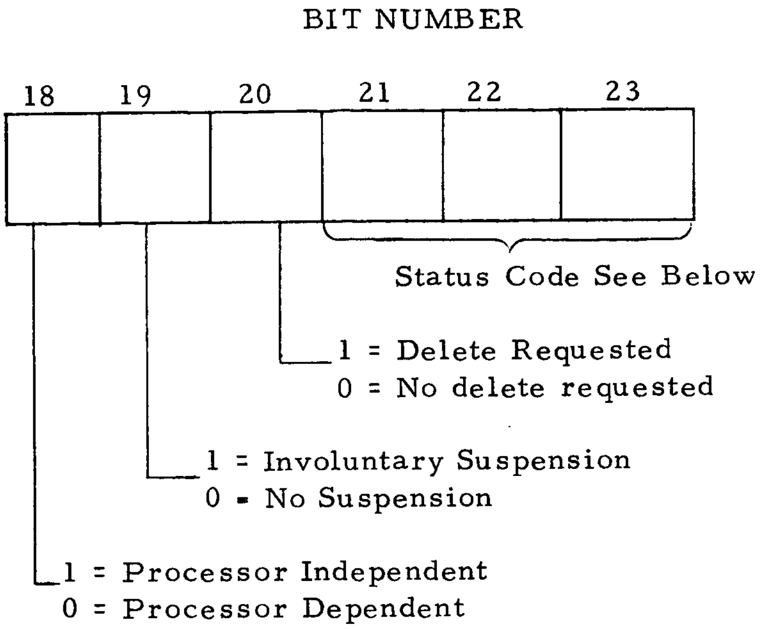
These examples illustrate the method of checking the current status of task TEST. Location IRC receives any detected error codes. Statement 210 and location ERR receive control if error conditions are detected.

4.8 Schedule Priority Task, Multiple Entry (M$08)

This function enables the specification of a particular entry from a list of entry points contained in the TQI of the task. The specific entry point will be valid at the next and subsequent executions of the task. Lists of entry points are specified by optional parameter five for the JTT Generator Program. (See Section 2). This function requires four parameters:

Parameter 1 - Location containing the address of the name of the requested task.

MPCS TASK STATUS CODES



STATUS CODE BITS 21, 22, 23 ABOVE

BITS			STATUS
21	22	23	
0	0	0	Not Scheduled
0	0	1	Scheduled, waiting initial entry
0	1	0	Active (Executing)
0	1	1	Scheduled, waiting entry after suspension
1	0	0	Voluntary Suspension
1	0	1	Suspended in MPCS mode

Figure 4-3

Parameter 2 - Location containing an integer that specifies a particular entry point from the list contained in the task TQI.

Parameter 3 - Location that will receive any error codes returned. Error codes have been assigned as follows:

1 - Requested task is not resident.
2 - Requested task is presently scheduled in priority schedule queue.
3 - Optional parameter five which is a requirement for multiple entry tasks is not present in the task TQI.
4 - Requested entry point is not valid. Number of entry points exceeds the size of the list contained in the task TQI.

Parameter 4 - Statement number or location that is to receive control when error conditions are detected.

Examples:

FORTRAN

```
CALL  M$08 (4HTASK, 7, IRC, $700)
```

ASSEMBLY LANGUAGE

```
SPB     $M$08
DATA    ="TASK"
DATA    =7
DAC     IRC
DAC     ERR
```

These examples illustrate the method of scheduling task TASK designating entry point 7 as the initial entry. Location IRC has been designated as the location to receive any detected error codes. Statement 700 and location ERR have been designated to receive control if error conditions are detected.

4.9 Schedule Periodic Task, Multiple Entry (M$09)

This function enables the specification of a particular entry from a list of entry points contained in the TQI of the requested task.

C.2

The specified entry point becomes valid at the next and subsequent executions of the task. Lists of entry points are specified by optional parameter five for the JTT Generator Program (See Section 2). This MPCS function requires seven parameters:

Parameter 1 - Location containing the address of the name of the requested task.

Parameter 2 - Address of a location containing an integer that specifies the particular entry point requested from a list contained in the task TQI.

Parameter 3 - Address of a location containing an integer that designates the unit of time, where the integers have been assigned as follows:

1 - designates milliseconds
2 - designates seconds
3 - designates minutes
4 - designates hours

Parameter 4 - Address of a location containing an integer that specifies the increment of time in the units specified by Parameter 3 above.

Parameter 5 - Address of a location containing an integer which may be either 1 or 0:

1 - designates iterative mode task or one that is executed repetitively at the specified period.
0 - designates a single execution at the specified time of scheduling.

Parameter 6 - Address that will receive any error codes returned. Error codes have been assigned as follows:

1 - Requested task is not resident.
2 - Task is currently scheduled in periodic queue.
3 - Optional Parameter 1 (Periodic Task Item) which is required for periodic scheduling is not present in the TQI of the requested task.
4 - Requested entry point is not valid. Number of entry point exceeds the size of the list contained in the task TQI.

6 - Error in specifying increment of time.
7 - Requested task is active in the priority queue. (This is a possible error condition; therefore, no error return is taken).

Parameter 7 - Statement number or address that receives control in the event error conditions 1, 2, 3, 4, or 6 are detected.

Examples:

FORTRAN

```
CALL  M$09 (4HSCAN, 63,1,60,1,IRC,$900)
```

ASSEMBLY LANGUAGE

```
SPB     $M$09
DATA    ="SCAN"
DATA    =63
DATA    =1
DATA    =60
DATA    =1
DAC     IRC
DAC     ERR
```

These examples illustrate the method of scheduling task SCAN designating entry point 63 as the initial entry. The task will be executed 60 milliseconds from the time of scheduling and, since the iterative mode has been specified, every 60 milliseconds thereafter. Location IRC has been designated as the location to receive any detected error codes. Statement 900 and location ERR have been designated as error return locations.

4.10 Suspend Task, Indefinite (M$10)

This function enables a task to suspend itself indefinitely. The suspension can only be removed by a scheduling function. When execution resumes, control is transferred to the first statement or location following the suspension call. No parameters are required for this function and no error conditions are detected.

Examples:

FORTRAN

```
CALL  M$10
```

ASSEMBLY LANGUAGE

```
SPB   $M$10
```

These examples illustrate MPCS calls to suspend tasks.

4.11 Specify Next Task Name (M$11)

This function enables a task to specify the name of a second task that is to be scheduled when the first task terminates. The task to be scheduled next may be either a periodic or priority type. The type is designated when specifying optional Parameter 4 for the JTT Generator Program (see Section 2). This MPCS function requires three parameters:

Parameter 1 - Location containing address of the name of the requested task.

Parameter 2 - Location that will receive any error codes returned. Error codes are as follows:

1 - Requested task is not resident.
3 - Optional Parameter 4 is not present in the TQI of the task requesting this MPCS function.

Parameter 3 - Statement number or address that receives control in the event error conditions are detected.

Examples:

FORTRAN

```
CALL  M$11 (4HALOG,IRC, $456)
```

ASSEMBLY LANGUAGE

```
SPB     $M$11
DATA    ="ALOG"
DAC     IRC
DAC     ERR
```

These examples illustrate how task ALOG is designated as the next task to be scheduled when the current task terminates. Location IRC has been designated as the location to receive any error codes. Statement 456 and location ERR have been designated to receive control if any errors are detected.

4.12 Suspend Task, Specified Time (M$12)

This function enables a task to suspend itself for a specified length of time. When the suspension period has elapsed, the execution of the task resumes at the first statement or location following the suspension call. A suspension may be removed by a scheduling function request from another task. Tasks requesting this function must contain optional Parameter 1 (Periodic Task Item) in their TQI. This function requires two parameters:

Parameter 1 - Address of a location containing an integer that designates the unit of time where the integers have the following designations:

1 - designates milliseconds
2 - designates seconds
3 - designates minutes
4 - designates hours

Parameter 2 - Address of a location containing an integer that specifies the increment of time for the suspension in the units specified by Parameter 1 above.

No error codes are returned. However, if the requesting task does not contain the required optional Parameter 1 in its TQI, an indefinite suspension is assumed.

Examples:

FORTRAN

```
CALL  M$12 (4HSCAN, 2, 1)
```

ASSEMBLY LANGUAGE

```
SPB     $M$12
DATA    ="SCAN"
DATA    =2
DATA    =1
```

These examples illustrate how task SCAN is suspended for a period of 1 second.

4.13 Task Communications (M$13)

This function enables one task to communicate with another task. Communicated information may consist of virtually any data, parameters, constants, or flags. Although this function has been designed as a general service available to any task, it will be of a particular value for those tasks that support display functions. Display data can be transmitted or received efficiently by these tasks. Information to be transmitted is limited only by the number of words which may be contained within a task communication area. This limit is set at a maximum of 4095 words. The task that initiates or requests a transfer of data specifies in the call sequence the array or buffer that is to receive or provide the data. The second task, which is indicated in the call sequence, must contain optional parameter P3 in its TQI.

One condition must be met before this executive function can be employed. If the task that initiates the transfer of information is processor independent, shared memory must be designated as the area to receive or send the information.

Six parameters are required for this MPCS function:

Parameter 1 - Location containing name of the task that is to receive or transmit the requested data.

Parameter 2 - Address of a location containing an integer that is a six bit function code. Figure 4-4 provides details of the function code assignment. It will be noted from the figure that data to be transmitted or received may be one of four types:

(1) Byte (6 bits)
(2) Half-word (12 bits)
(3) Word (24 bits)
(4) Double word (48 bits)

It will be noted from the figure that the least significant bits designate the data type as follows:

1 - denotes byte
2 - half-word
3 - word
4 - double word

FORTRAN calls for this MPCS service must specify the decimal integer for function codes.

TASK COMMUNICATION FUNCTION CODE ASSIGNMENT

T/R	Mode	Format				
		0	0	0	1	Byte
		0	0	1	0	Half-word
		0	0	1	1	Word
		0	1	0	0	Double word

T/R:	1 = transmit
	0 = receive
Mode:	0 = initialize
	1 = sequential

Figure 4-4

PRECEDING PAGE BLANK NOT FILMED

Data is always maintained in the task communication area in a packed format. Upon receipt, data is packed and stored into the area. Upon transmit, data is unpacked and stored in the requested receiving area in an unpacked format, right-adjusted within the word. Figure 4-5 illustrates the format of the types of data.

Two modes of information transfer may be specified - initialize mode and sequential mode. The initialize mode specifies that the information transfer operation should start at the beginning of the data. The sequential mode specifies that the information transfer should begin at the point where the previous operation ended. When this parameter is reset (0), it specifies initialize mode; set (1), sequential mode.

Parameter 3 - Address of location containing pointer to area that is to receive or transmit data to the requested task.

Parameter 4 - Address of location containing number of increments to be transmitted or received.

Parameter 5 - Location containing address that is to receive error codes, where error codes have been assigned as follows:

1 - Requested task is not resident.
2 - Requested task is active (executing).
3 - Optional Parameter 3 (Task Communications) which is required for a requested task is not present in the TQI.
4 - Requested task has not executed since the last communications function request. This implies that the previous data has not been processed.
5 - Requested data exceeds limits of the communications area.

Parameter 6 - Statement or location that is to receive control if error conditions are detected.

Examples:

FORTRAN

```
CALL  M$13 (4HTST1, 35, INFO, 100, IRC, $500)
```

COMMUNICATION DATA FORMAT

RECEIVE MODE

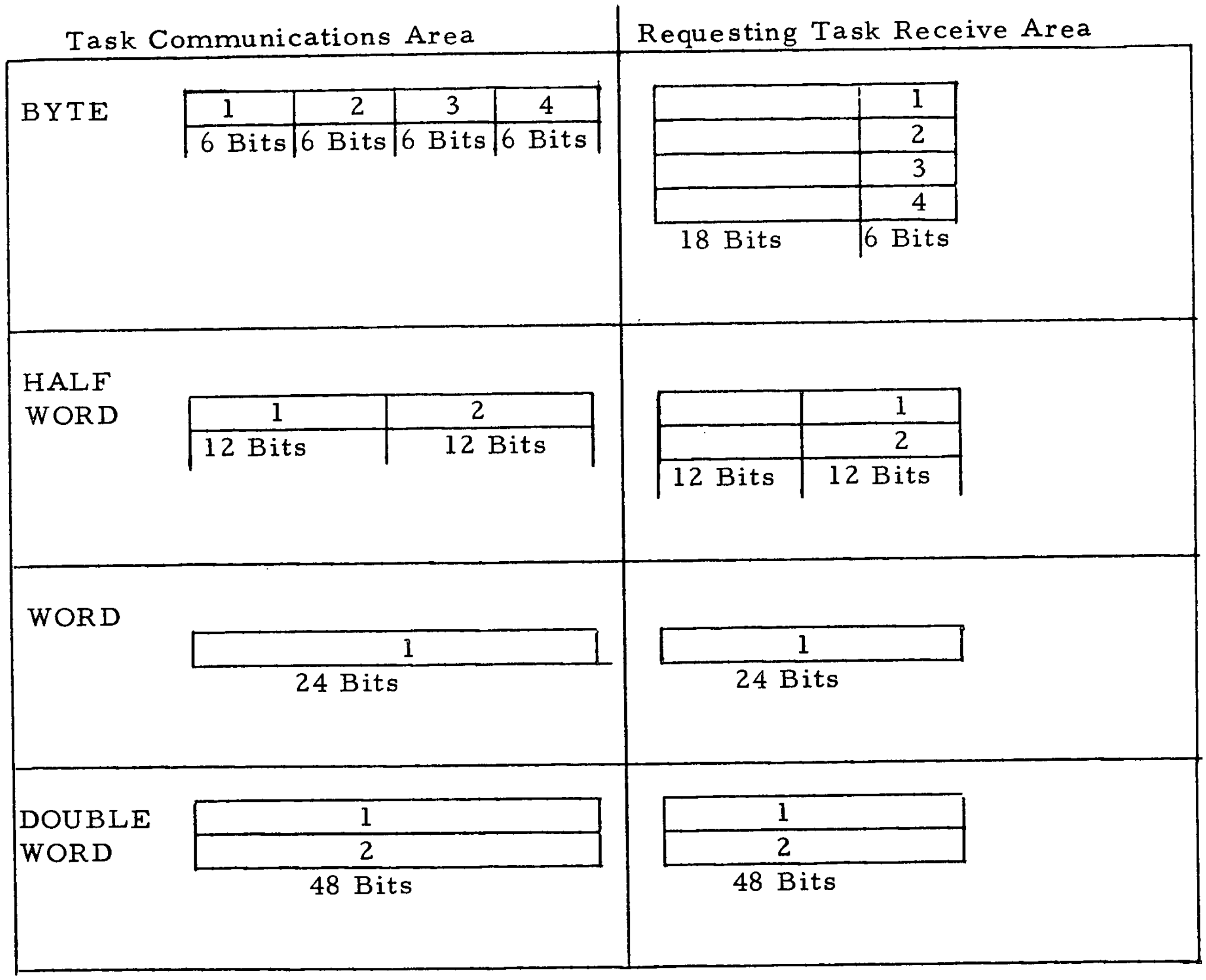


Figure 4-5

COMMUNICATION DATA FORMAT
(continued)

TRANSMIT MODE

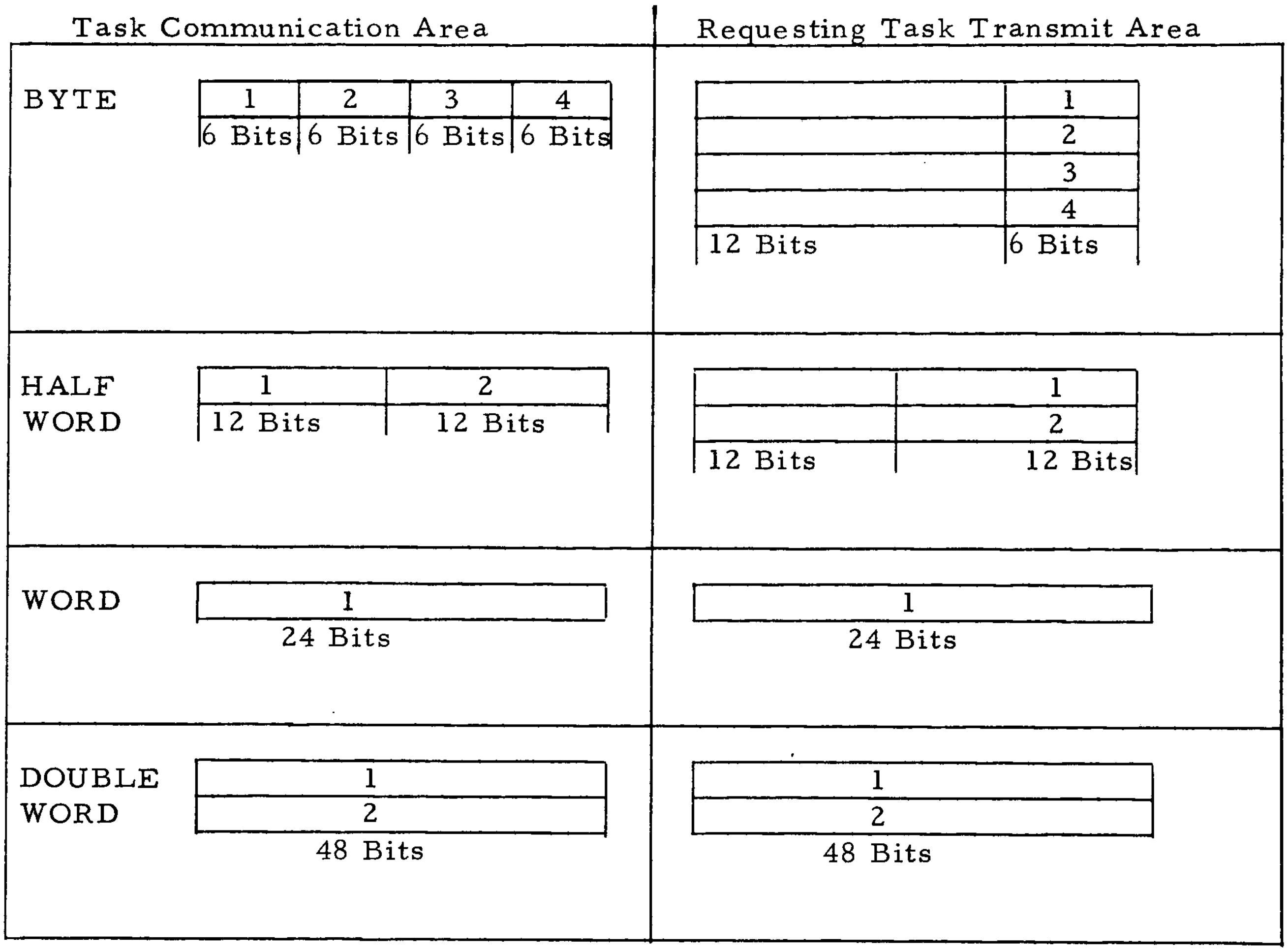


Figure 4-5
(continued)

ASSEMBLY LANGUAGE

```
SPB     $M$13
DATA    ="TST1"
DATA    ='43
DAC     INFO
DATA    =100
DAC     IRC
DAC     ERR
```

These examples illustrate task communications functions. Task TST1 has been designated to receive 100 words from a block named INFO. The function code (Parameter 2) specifies the transmit word request in the initialize mode. Location IRC has been designated as the location to receive any detected error codes. Statement 500 and location ERR have been designated to receive control if error conditions are detected.

4.14 Connect Task to Interrupt (M$14)

This function enables a task to be connected to an external interrupt. The connection is indirect in the sense that when the specified interrupt occurs, the connected task is scheduled for execution rather than executed directly. Facilities for connecting a subroutine directly to an interrupt are discussed in Section 6. It should be noted, however, that a directly connected subroutine is not executed as a task and therefore should not attempt to use MPCS facilities provided solely for tasks.

Care should be taken to insure compatibility between the type of the connecting task and the type of the connected task. If the connecting task is independent, the connected task must also be independent. Also note that if the task to be connected is periodic, the timing parameters used will be the ones in the TQI at the time the interrupt occurs.

Parameter 1 - Location containing the address of the task name.

Parameter 2 - Location containing the address of an integer which specifies the interrupt level and whether priority or periodic scheduling is required by the task. The absolute value of the level must range from 1 - 15. A positive value signifies priority scheduling. A negative value specifies periodic scheduling.

Parameter 3 - Location containing the address of a variable which will receive any error codes. Error codes are defined as follows:

1 - Requested task does not have a TQI.
3 - Periodic option is not specified in TQI.
5 - Requestor is an independent task but specified task is dependent.
6 - Interrupt level is not in range of 1-15.

Examples:

FORTRAN

```
CALL M$14 ( 4HTET1, 12, IRC, $670)
```

ASSEMBLY LANGUAGE

```
SPB    $M$14
DATA   ="TET1"
DATA   = 12
DATA   IRC
DATA   ERR
```

These examples illustrate that task TET1 is to be connected to interrupt level 12. The task will be scheduled in the priority queue when the interrupt is received. Location IRC will receive any detected error codes. Statement 670 and location ERR will receive control if error conditions are detected.

4.15 Cross Processor Communications (M$15)

This function enables a task resident is one processor to communicate with a task resident in another processor. This function requires six parameters:

Parameter 1 - Location containing the address of the task name that is to receive or transmit the requested data.

Parameter 2 - Location containing the address of an integer which specifies the communications function code. Function codes have been assigned as follows:

CODE	DEFINITION
1	Request for I/O
2	I/O Completion
3	Processor request for priority level 16 task
4-63	Not assigned. Available for future assignment and use by application task.

Function codes 1-3 above may not be used by application tasks. Codes 4-63 are not presently assigned.

Parameter 3 - Location containing the address of a word count specifying the number of data words to be received or transmitted. The word count may not exceed 512.

Parameter 4 - Location containing the address of the receive or transmit area.

Parameter 5 - Location containing the address that is to receive error codes, where error codes have been assigned as follows:

1 - Requested task is not resident
3 - Optional Parameter 2 (Cross Processor Communication Parameter) is not present in the TQI of the requested task.
4 - Invalid parameter
5 - Requested data exceeds limits of the cross processor communications area.

Parameter 6 - Location or statement number that is to receive control if any error condition is detected.

NOTE:

Currently used only for remote I/O and forced execution request.

5. MPCS CALLS AND SYSTEM INTERFACES FOR DISPLAY FUNCTIONS

Applications utilizing MPCS display support services must be concerned with system interfaces for both receiving display console requests and issuing display update requests to the console. Both are discussed in the following subsections. A summary of program display requests is presented in Figure 5-1 and a summary of error return codes is presented in Figure 5-2.

5.1 Display Input Data Formats

Input data from the display console are generated as a result of one of the following actions by the console operator:

(1) Depressing the keyboard "Return" key.
(2) Using the light pen to select an option which was defined to have an associated task.
(3) Raising a console function switch to which a task is attached.

The format of the data presented to a task in response to each of these actions is described in the following subsections.

Cross-task communication services are used by the MPCS Display Controller to pass the input data to the receiving task. Therefore, each application task designed to receive data from the display console must have the cross-task communication option specified in its TQI (reference Section 3, Volume II) with a communications area (buffer) large enough to hold the data.

Another factor to consider is the priority of the receiving task. Since each input request received from the display console is stored into the cross-task communication area of the receiving task immediately, data from a previous request will be lost if it has not yet been processed. Such tasks should be assigned a relatively high priority (above 7) to help prevent data loss. A high priority, however, is not always sufficient. For example, if the task is suspended for any reason, data may be lost during the suspension period. The overall task structure of an application should be designed to prevent data loss or, at least to minimize its effects.

5.1.1 Compose Field Input Format

The data received by a task from a display compose field entered by the console operator varies in length according to field size as speci-

DISPLAY REQUEST SUMMARY

EXTERNAL NAME	DISPLAY REQUEST FUNCTION	PAGE
M$20	Tabular Data Display	73
M$21	Position Data Cursor	76
M$22	Display One-Line Message	78
M$23	Request New Display	81
M$24	Function Switch Attach/Detach	83

Figure 5-1

DISPLAY RETURN CODE SUMMARY

Request / Return Code	Tabular Data M$20	Cursor Move M$21	One-Line Msg. M$22	New Display M$23	Function Switch Attach/Detach M$24
-2	Wrong processor	Wrong processor	Wrong processor	Wrong processor	
-1	Requester has no TQI	Requester has no TQI	Requester has no TQI	Requester has no TQI	
0	Successful completion	Successful completion	Successful completion	Successful completion	
1	Display name does not match	Display name does not match	Display name does not match	Display name does not match	
2	840-810 error in transmission	840-810 error in transmission	840-810 error in transmission	840-810 error in transmission	
3	Character count too large	Invalid compose field no.	Character count is invalid	Invalid previous display request	
4	No fill-in fields in display	No compose fields in display		Display not found in library	
5				Requested task has no TQI	
6				Task is active	
7				Task has no input buffer	
8				Task has not processed previous request	
9				Task input buffer is too small	

Figure 5-2

fied at display definition time. The format of the data is shown below:

Word 0	Display No.
1	Compose Field No.
2	No. of Data Words
3	Data Word 1
4	Data Word 2
.	.
.	.
.	.
N+2	Data Word N

Word 0 is the first word of the cross-task communication area specified by the TQI option for the receiving task. It contains the display number which is an integer value representing the number (tag) of the display from which the compose field request was issued by the operator. The compose field number is an integer specifying which compose field was entered. Compose fields are numbered from left-to-right, top-to-bottom as they appear on the CRT. The number of data words is also an integer and specifies the number of data words to follow.

The remaining data words contain the actual data entered by the operator. Data are packed,6-bit characters (4/word) and are stored in the order in which the characters appeared on the CRT. Compose field character positions for which no data were entered will contain blanks. For compose fields which contain subfields, each subfield is started on a word boundary with the preceding subfield padded on the right with blanks.

Example:

Consider a display defined with the following Display Librarian Language statements:

```
N, 0257
C, 3, 6, 0300, TSKX
C, 3, 9, 0302, TSKY
.
.
.
```

```
*T
'
'
'
T,      COMPOSE FIELD 1  //     //     //
'
'
'
T,      COMPOSE FIELD 2  ///    /      /////
'
'
'
```

If the operator enters ABC=12345 into compose field 2, task TSKY will receive the following data:

00000401	Display 257
00000002	Compose Field 2
00000004	4 Data Words
01020340	Subfield 1 (ABC)
75404040	Subfield 2 (=)
61626364	Subfield 3 (1234)
65404040	Subfield 3 (5)

5.1.2 Light Pen Option Input Format

When the operator initiates a task through use of the light pen, the three words of data shown below are stored in the first three words of the cross-task communication area specified in the task TQI. Note that the task name is an optional parameter in the specification of light pen options at display definition time. Pen options which are defined without a task name will simply result in the loading of a new display when selected.

Word 0	Display No.
1	Option No.
2	00000000

The display number is an integer specifying the number (tag) of the display from which the console operator made the pen selection. Option number is an integer value which specifies the number of the option selected. The options are numbered sequentially from one to N, left-to-right, top-to-bottom. The last word containing zeroes is provided to enable the receiving task to differentiate between an option number and a compose field number originating from the same display.

5.1.3 Function Switch Input Format

Each time the operator raises a function switch to which a task has been attached, the following two words are stored into the first two words of the task cross-task communication area.

Word 0	Display No.
1	Function Switch No.

The display number is an integer specifying the number (tag) of the display on the CRT at the time the switch was raised by the operator. The second word is a negative integer (ranging from -1 to -16) specifying the number of the function switch raised. It is presented as a negative value to enable the receiving task to distinguish function switch requests from compose field and pen option requests.

5.2 Task Display Requests Formats

MPCS provides five basic display services which can be requested by application tasks:

(1) BCD data can be displayed in predefined tabular data fill-in fields in a given display.

(2) The data cursor can be positioned to any compose field in a given display.

(3) A message can be displayed via the standard one-line message area which appears at the bottom of each normal display. Additionally, a new display and task can be assigned to the one-line message pen option.

(4) A new display can be loaded in either a delayed mode (requiring selection of a pen option by the CRT console operator) or an immediate mode. The request can also

specify the name of a task to be scheduled at the time the display is loaded.

(5) An application task can be attached to or detached from a display console function switch.

Statement formats for requesting these services are discussed in the following subsections.

For each of the first four services listed above, the requesting application task must be suspended while the request is transmitted to the 810B for processing. When a return code is received from the 810B, the task is reactivated. Since a task must have a TQI by definition, display services are available only to MPCS tasks. Ordinary programs executing under MPCS without a TQI will receive an abnormal return from any display request.

5.2.1 M$20 - Display Request for Tabular Data

The caller specifies the number of the display to receive the data, the number of characters to be processed, the character string itself, the name of a variable to receive a return code, and an option selection indicator. Depending on the option specified one of the following will be performed:

(1) Add-On - The character string will be inserted into the predefined tabular data character fields of the display on a one-for-one basis from left-to-right, top-to-bottom. If a previous call has already caused characters to be displayed, the new character string will be concatenated to the previous characters, assuming that enough predefined character fields are available to receive the new data. The character fields need not be adjacent to each other in the display. For example, a single request with a ten-bit character string can be used to display 10 one-bit character fields in the display.

(2) Reset - All predefined tabular data character fields will be reset to their initial status (their contents replaced with #). A reset request can also include a character string to be displayed as discussed in the preceding paragraph. In such a request the new characters are added to the display after the reset is performed.

(3) Update - Beginning at a specified relative character position, the character string will be inserted as in paragraph 1 above. This enables one or more tabular data fields in a display to be updated independently of other fields. The relative position for the update is determined by considering all N predefined tabular data characters in a given display as a character string with characters numbered 1 through N.

After performing the requested service, the system returns to the caller with either a normal or abnormal return and a return code.

Call: FORTRAN CALL M$20 (IRC, IDSP, ICOD, NC, CHAR, $SN)

ASSEMBLY LANGUAGE

```
CALL  M$20
DATA  IRC
DATA  IDSP
DATA  ICOD
DATA  NC
DATA  CHAR
DATA  ERR
```

where:

IRC - is the name of the variable to receive the integer return code.

IDSP - is the integer variable or constant specifying the number (tag) of the display to receive the character string.

ICOD - is the integer variable or constant code specifying the desired option as follows:

+1 - add-on
0 - reset
-N - update beginning at character N

NC - is the integer variable or constant specifying the number of characters of data to be displayed. The number of characters that can be processed in a single request is limited to 72 regardless of the number of predefined character fields in the display.

CHAR - is the variable or constant BCD character string to be displayed.

$SN - represents the number of the FORTRAN statement to which the system returns control in the event of an error (non-zero return code).

ERR - is the name of the assembly language statement (instruction) to which the system returns control in the event of an error (non-zero return code).

Return Codes:

-2	Display processing is being performed in another processor.
-1	Indicates the caller does not have a TQI and, consequently, cannot have his request serviced.
0	The request was serviced successfully, and the system will perform a normal return.
1	The display specified by IDSP was not the current display when the request was processed. The requested service was not performed.
2	I/O errors in 840-810 transmissions made it impossible to service the request or to determine whether or not it was serviced properly.
3	The character count was too large. Characters were stored in the remaining available predefined fields of the display until all available fields, if any, were filled.
4	There were no predefined tabular data fields in the specified display.

Examples:

```
        CALL  M$20 (IRET, 256, 0, 8, 8HOVERFLOW, $210)
```

or

```
        CALL  M$20
        DATA  IRET
        DATA  =256
```

```
DATA =0
DATA =8
DATA ="OVERFLOW"
DATA ERR1
```

This request will cause all tabular data fields of display 256 to be reset with their initial contents and subsequently have the characters OVERFLOW inserted into the first 8 predefined character positions of the display. If an error occurs, control will be returned to the FORTRAN statement 210 or the assembly language statement ERR1 with the appropriate return code stored in IRET.

```
CALL M$20 (IRC, 8002, -5, 6, DATA, $900)
```

This request will cause 6 characters to be displayed from DATA. It will be inserted into character positions 5 through 10 of display 8002.

Note that the contents of DATA must be in BCD format.

5.2.2 M$21 - Display Request to Position the Data Cursor

The caller specifies the number of the display for which the cursor is to be positioned, the number of the compose field to which the cursor is to be positioned, and the name of the variable to receive a return code. After positioning the data cursor to the first character of the specified compose field, the system returns control to the caller with either a normal or abnormal return and a return code.

Call: <u>FORTRAN</u> CALL M$21 (IRC, IDSP, NCF, $SN)

<u>ASSEMBLY LANGUAGE</u>

```
CALL M$21
DATA IRC
DATA IDSP
DATA NCF
DATA ERR
```

where:

IRC - is the name of the variable to receive the integer return code.

IDSP - is the integer variable or constant specifying the number (tag) of the display for which the cursor is to be positioned.

NCF - is the integer variable or constant specifying the number of the compose field to which the cursor is to be positioned. Must range from 1 to N where N is the number of compose fields in the display.

$SN - represents the number of the FORTRAN statement to which the system returns control in the event of an error (non-zero return code).

ERR - is the name of the assembly language statement (instruction) to which the system returns control in the event of an error (non-zero return code).

Return Codes:

-2	Display processing is being performed in another processor.
-1	Indicates the caller does not have a TQI and, consequently, cannot have his request serviced.
0	The request was serviced successfully and the system will perform a normal return.
1	The display specified by IDSP was not the current display when the request was processed. The requested service was not performed.
2	I/O errors in 840-810 transmissions made it impossible to service the request or to determine whether or not it was serviced properly.
3	The specified compose field number is invalid. It is either negative or larger than the number of compose fields in the display. A value of zero will be accepted and processed as if it were a one.
4	The specified display has no compose fields.

Examples:

```
CALL  M$21 (IRC, 16, 7, $600)
```

or

```
CALL  M$21
DATA  IRC
DATA  =16
DATA  =7
DATA  ERR2
```

The data cursor will be moved to the first character of compose field 7 of display 16. The return code will be stored in IRC and, in the event an error is found, control will return to FORTRAN statement 600 or assembly language statement ERR2.

5.2.3 M$22 - Display Request for a One-Line Message

The caller specifies the number of the display to receive the message, the number of characters in the message, the BCD character string message itself, the number (tag) of the display and the name of the task to be assigned to the standard one-line message pen option, and the name of the variable to receive the return code. After inserting the character string in the one-line message display buffer and assigning the display number and task name to the pen option associated with the one-line message, the system returns control to the caller with either a normal or abnormal return and a return code.

Call: FORTRAN

```
CALL  M$22 (IRC, IDSP, NDSP, TASK, NC, CHAR, $SN)
```

ASSEMBLY LANGUAGE

```
CALL  M$22
DATA  IRC
DATA  IDSP
DATA  NDSP
DATA  TASK
DATA  NC
DATA  CHAR
DATA  ERR
```

where:

IRC - is the name of the variable to receive the integer return code.

IDSP - is the integer variable or constant specifying the number (tag) of the display to receive the message. If it is desired to issue the message to any display without regard for the display number, this parameter should have a value of -1.

NDSP - is the integer variable or constant specifying the number (tag) of the display to be assigned to the one-line message pen option. If no display is to be assigned, this parameter should have a value of -1. If the preceding display is to be assigned, this parameter should have a value of -2. The latter value will cause the same display to be invoked as would the "RETURN TO PRIOR LEVEL" pen option.

TASK - is the variable or constant BCD character string name of the task to be assigned to the one-line message pen option. If no task is to be assigned, this parameter should have an integer value of zero.

NC - is the integer variable or constant specifying the number of characters in the message. Maximum message size is 72 characters. Messages exceeding 72 characters will be truncated.

CHAR - is the variable or constant BCD character string message.

$SN - represents the number of the FORTRAN statement to which the system returns control in the event of an error (non-zero return code).

ERR - is the name of the assembly language statement (instruction) to which the system returns control in the event of an error (non-zero return code).

Return Codes:

-2	Display processing is being performed in another processor.
-1	Indicates the caller does not have a TQI and, consequently, cannot have his request serviced.
0	The request was serviced successfully and the system will perform a normal return.
1	The display specified by IDSP was not the current display when the request was processed. The requested service was not performed.

2 I/O errors in 840-810 transmissions made it impossible to service the request or to determine whether or not it was serviced properly.

3 The message size (character count) is invalid. A negative value or zero will not be accepted. A message larger than 72 characters will be accepted but will be truncated such that only the first 72 characters will be displayed.

Examples:

```
        CALL  M$22 (IRET, CDSP, NDSP, NTSK, 22, MSG1, $200)
```

or

```
        CALL  M$22
        DATA  IRET
        DATA  CDSP
        DATA  NDSP
        DATA  NTSK
        DATA  =22
        DATA  MSG1
        DATA  ERRX
```

The 22 character message stored in MSG1 (in BCD format) will be displayed in the one-line message area of the display specified by CDSP. The contents of NDSP and NTSK specify the display and task to be assigned to the one-line message pen option. Error returns will be to the FORTRAN statement 200 or assembly language statement ERRX with an appropriate return code in IRET.

```
CALL M$22 (IRC, -1, -1, 0, 40, 40HOUTBOARD ENGINE CUTOFF HAS
        BEEN DETECTED, $300)
```

This request will cause the 40 character message to be inserted into the one-line message buffer of whatever display is currently on the screen. The new display and task names (-1 and 0 respectively) assigned to the one-line message pen option will result in no action if the option is selected (-1 specifies SAME display and 0 specifies no task). Note that if a previous request had assigned values to the one-line message pen option, the values would be reset by this request.

5.2.4 M$23 - Display Request for a New Display

The caller specifies the number of the display through which the request is to be made, the number of the new display, the name of the task to be scheduled when the new display is loaded, and the name of the variable to receive the return code.

In the delayed mode, the system displays a standard display request message in the one-line message display area and assigns the new display number and task name to the one-line message pen option. Control is then returned to the caller.

In the immediate mode, the system loads the new display and schedules the task automatically before returning control to the caller.

Control may be returned either normally or abnormally and will provide an appropriate return code.

Call: <u>FORTRAN</u>

```
CALL  M$23 (IRC,IDSP,NDSP, TASK, $SN)
```

<u>ASSEMBLY LANGUAGE</u>

```
CALL  M$23
DATA  IRC
DATA  IDSP
DATA  NDSP
DATA  TASK
DATA  ERR
```

where:

IRC - is the name of the variable to receive the integer return code.

IDSP - in the delayed mode, is the integer variable or constant specifying the number (tag) of the display through which the request is to be made. If it is desired to make the request through any display without regard for display number, this parameter should have a value of -1. In the immediate mode, this parameter must have a value of -2.

NDSP - is the integer variable or constant specifying the number of the new display. If it is desired to retain the same display, this parameter should have a value of -1. If it is desired to re-load the previous display (as in "RETURN TO PRIOR

LEVEL"), this parameter should have a value of -2.

TASK - is the variable or constant BCD character string name of the task to be scheduled when the new display is loaded. If no task is to be assigned, this parameter should have an integer value of zero.

$SN - represents the number of the FORTRAN statement to which the system returns control in the event of an error (non-zero return code).

ERR - is the name of the assembly language statement (instruction) to which the system returns control in the event of an error (non-zero return code).

Return Codes:

-2	Display processing is being performed in another processor.
-1	Indicates the caller does not have a TQI and, consequently, cannot have his request serviced.
0	The request was serviced successfully and the system will perform a normal return.
1	The display specified by IDSP was not the current display when the request was processed. The requested service was not performed.
2	I/O errors in 840-810 transmissions made it impossible to service the request or to determine whether or not it was serviced properly.
3	The automatic request for a previous display was invalid due to the fact that the current display is the first entry in the display history table.
4	The requested display could not be found in the display library.
5	The requested task has no TQI.
6	The requested task is already scheduled.

7 The requested task has no Task Communication Option in its TQI.

8 The requested task had not yet processed a previous request. The data for the new request was stored on top of the previous request.

9 The task input buffer is too small.

NOTE:

Return codes 3-9 apply only to automatic (immediate mode) requests.

Examples:

```
CALL  M$23 (IRC, 706, 710, 4HT12A, $120)
```

or

```
CALL  M$23
DATA  IRC
DATA  =706
DATA  =710
DATA  ="T12A"
DATA  ERR5
```

This delayed mode request will cause display 710 and task T12A to be assigned to the one-line message pen option of display 706 if it is currently being displayed. The standard display request message will also appear on the screen. A return code will be stored in IRC and abnormal returns will be to FORTRAN statement 120 or assembly language statement ERR5.

```
CALL  M$23 (IRET, -2, 9005, 0, $75)
```

This is an immediate mode request for display 9005. The new display will be loaded immediately into the special display buffer.

5.2.5 M$24 - Display Request Function Switch Attach/Detach

The caller specifies a task name and the number of the function switch to which the task is to be attached or from which the task is to be detached. A positive switch number indicates the attach option, a negative switch number indicates detach.

After performing the attach or detach, the system returns to the caller with either a normal or an abnormal return.

Call: FORTRAN CALL M$24 (TASK, NFS, $SN)

ASSEMBLY LANGUAGE

```
CALL  M$24
DATA  TASK
DATA  NFS
DATA  ERR
```

where:

TASK- is the variable or constant BCD character string name of the task to be attached to or detached from the specified function switch.

NFS - is the integer variable or constant specifying the number of the function switch to be affected. It must have values ranging from +1 through +16 for an attach and from -1 through -16 for a detach.

$SN - represents the number of the FORTRAN statement to which the system returns control in the event of an error.

ERR - is the name of the assembly language statement (instruction) to which the system returns control in the event of an error.

Return Codes:

There are no return codes for this service but error returns are possible.

1. An error return will be made for an attach if the specified function switch was not in the range 1 through 16.

2. An error return will be made for a detach if the specified task name does not match the name of the task currently assigned to the specified function switch.

Examples:

```
        CALL  M$24 (4HTAX1,14,$117)
```

or

```
        CALL  M$24
        DATA  ="TAX1"
        DATA  =14
        DATA  ERRB
```

The Task TAX1 will be attached to function switch 14. In the event of an abnormal return, control will be passed to FORTRAN statement 117 or assembly language statement ERRB.

```
        CALL  M$24 (TSK1,-2,$506)
```

The task named by the contents of TSK1 will be detached from function switch 2.

6. MPCS CALLS FOR I/O FUNCTIONS AND SERVICES

6.1 General I/O Services

Input and output functions of the system peripheral devices are provided by a central MPCS service routine. Application tasks and programs written in assembly language request this service with an explicit MPCS call, whereas tasks and programs written in FORTRAN are linked to this central MPCS routine via subroutines. The descriptions and definitions contained within this section pertain directly to the assembly language user, but may benefit the FORTRAN user as a source of general information.

The assembly language call sequence for this MPCS service is:

```
LAA   FCT
SPB   1
```

where:

FCT - is a pointer contained in the A accumulator which specifies the start address of the user Function Control Table (FCT).

SPB 1 - is the specific MPCS entry for the general I/O services.

The Function Control Table (FCT) consists of from 1 to 5 words and contains all pertinent parameters required by the general I/O service routine. Parameters contained within an FCT are as follows:

WORD 1

A	B	C	D	E	F	G	H
0	1	2	3 4 5 6 7 8	9	10	11	12 13 14 15 16 17 18 19 20 21 22 23

WORD 2 (C1)

J	K	L	M
0	1 2	3 4 5 6 7 8	9 10 11 12 13 14 15 16 17 18 19 20 21 22 23

WORD 3 (C2)

N	P
0 1 2 3 4 5 6 7 8 9 10 11	12 13 14 15 16 17 18 19 20 21 22 23

WORD 4
(C3)

Q
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23

WORD 5
(C4)

R
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23

where:

A - Initialized to 0 by user and set to 1 by MPCS when device is busy and reset by MPCS when operation is completed.

B - Set by user to 0 if MPCS is to wait for I/O completion before returning to user. If set to 1, return to user is immediately after I/O is started and user must test bit 0 to determine if his buffer is available for alteration. No error recovery is accomplished by MPCS in the no wait mode.

C - Set by user to 0 for forward direction and to 1 for reverse direction. This parameter is used only for disk and magnetic tape control operations.

D - Function Code for I/O Operation. Function codes are defined in the following table.

CODE	FUNCTION
1 (LAA)	Read record
2 (LBA)	Write Record
3 (STA)	Write end of file
4 (STB)	Rewind, page eject and punch leader
5 (AMA)	Space record, upspace printer
6 (SMA)	Space file
9 (BRU)	Read (Special Mode) 12 bit mode for card device 16 bit mode for scope device

CODE	FUNCTION
10 (SPB)	Write (Special Mode) 12 bit mode for card device 16 bit mode for scope device

E - If set to 1, optional words C1 and C2 as a pair are present. If set to 0, C1 and C2 are not present.

F - If set to 1, optional word C3 is present and MPCS will return to the user specified abnormal address when an end of file is sensed or if an error condition is encountered.

G - If set to 1, optional word C4 is present. C4 must be present if the user desires to assign an LDN to the disk.

H - Logical Device Number

LDN	MNEMONIC	DEVICE
0	NO	No Device (No I/O is performed)
1	KB	ASR33 Teletype Keyboard
2	TP/TR	ASR33 Teletype Paper Tape
3	PP/PR	High Speed Paper Tape
4	CP/CR	Card Reader/Punch
5	LP	Line Printer
6	M0	Magnetic Tape Drive Unit 0
7	M1	Magnetic Tape Drive Unit 1
10	SY	Disk System Area
11	DC	Disk Scratch Area
12	BO	Disk Binary Object Area
13	M2	Magnetic Tape Drive Unit 2

LDN	MNEMONIC	DEVICE
14	M3	Magnetic Tape Drive Unit 3
15	SC	CRT Display
16	DO	Line Printer
17	LI	Disk Library Area

J - Set to 1 for binary mode or set to 0 for TASCII (truncated ASCII or 6 bit ASCII)

K - Not Used

L - Error or abnormal condition code

where:

CODE	ERROR CONDITION
1	Mag tape parity error (read)
2	EOF during read
3	Mag tape parity error (write)
4	End of tape
5	EOF error from skip file reverse or skip file forward
6	Line printer paper low
7	Disk check sum error
8	Disk full
9	FCT specification error
10	Card reader error
11	Card punch error
12	Data overflow or underflow

M - Address of user buffer

N - Number of data words transmitted

P - Number of words to transmit. A maximum word count of 4096 may be specified and is indicated by setting this parameter to zero.

Q - Address of user routine to be entered if abnormal and/or error conditions are detected.

R - Record Number. This word is set by the user initially to zero and is updated by MPCS to reflect the number of sectors read or written during an I/O request. The user may perform random disk operations by altering Parameter R to reflect the starting record to be read or written. The record number is calculated by dividing the word count by the sector capacity (64 words) and rounding to the next whole number.

If it is the intent of the user to assign his non-disk device to the disk in the future, Parameter R must be present in the FCT. Parameter R in an FCT does not interfere with I/O operations to non-disk devices.

If Parameter R is present in an FCT used to position a device (function code 5 or 6), it will be used to determine the number of records, sectors, or files to position the device. If Parameter R is absent in the FCT, the default is one record, sector, or file to be advanced or backspaced.

6.2 Hybrid Interrupt Services

6.2.1 Interrupt Enable/Disable

This function enables a user to request that a specific interrupt level be enabled or disabled. The FORTRAN call statement is:

CALL INTSER (CODE, INTNUM)

where:

CODE = Argument indicating the interrupt service to be performed. If this argument is a positive one (+1), the indicated level will be disabled and if it is a positive two (+2), it will be enabled.

INTNUM = The external interrupt level to be serviced. If INTNUM is 0, interrupt level 1, group 2 is designated.

The Assembly Language call is:

```
SPB     '17
```

where:

Caller's A accumulator must be set to positive to enable or negative to disable the interrupt level and the B accumulator set to define the level as follows:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	INTERRUPT LEVEL
9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	

BIT NUMBER

where each bit on (1) will set or reset the specific interrupt level.

6. 2. 2 Connect Interrupt Direct

This function enables an application program or subroutine to be connected to one of the external interrupts. The FORTRAN call statement is:

```
CALL  CONINT (INTNUM, INTSUB)
```

where:

INTNUM = The external interrupt level to be connected.

INTSUB = The user's interrupt subroutine name.

The Assembly Language call is:

```
SPB     '16
```

where:

Caller's A accumulator must be set to the interrupt number and the B accumulator set to the address of the first instruction to be executed when the particular interrupt is received.

6.3 Hybrid I/O Services

6.3.1 Read Real Time Entry (RTE)

Sequential Mode - This service will enable a user to read a number of consecutive multiplexer channel addresses from the RTE subsystem into a user specified memory buffer. The data will be in the form of single precision floating point numbers.

The FORTRAN call statement is:

```
CALL  RTES (ICHAN, NO, BUFF, SF)
```

where:

ICHAN - Address of the first multiplexer channel to be read.

NO - Number of consecutive multiplexer channels to be read.

BUFF - Name of array in the calling program into which data will be read. The data will be single precision floating point and will be normalized such that 100 volts is equal to unity in the digital computer. The values stored in BUFF will be the result of converting from fixed to floating point and multiplying by the appropriate scale factor.

SF - Name of array in the calling program containing scale factors to be applied to the multiplexer channel readings.

Random Mode - This service enables the user to read one multiplexer channel into a specified memory location. The data will be a single precision floating point number.

The FORTRAN call statement is:

```
CALL  RTER (CHANUM, USRLOC)
```

where:

CHANUM - the multiplexer channel to be read.

USRLOC - variable into which the RTE data is to be stored. The data will be normalized such that 100 volts is represented by unity in the digital computer.

The Assembly Language call to read the RTE is:

```
SPB     '35
```

where:

The A accumulator is set to the RTE address prior to calling the monitor. If bit 0 of the A accumulator is set, one RTE channel is read in the random mode and its fixed point value is returned in the A accumulator. If bit 0 of the A accumulator is not set, RTE channels are read in the sequential mode with the RTE address in the A accumulator being the first read. Index register 1 must be set with the negative count of the number of RTE channels to read and the B accumulator must be set to the buffer address where the fixed point values will be stored.

6.3.2 Write Digital to Analog Converter (DAC)

Sequential Mode - This service enables the user to set a number of consecutive digital-analog converter (DAC) channels to values defined in a user specified memory buffer. The data presented to the service routine must be in the form of single precision floating point numbers.

The FORTRAN call statement is:

```
CALL  WDACS (IDAC, NO, BUFF, SF)
```

where:

IDAC - address of the first DAC to be written.

NO - number of consecutive DAC's to be written.

BUFF - name of array in the calling program containing the data to be written to the DAC's.

SF - name of an array of scale factors in the calling program. The data values from BUFF will be divided by the corresponding scale factor before being converted to fixed point.

The Assembly Language call is:

```
SPB     '37
```

where:

The A accumulator is set to the address of the caller's buffer and the B accumulator is set to the number of DAC's. The caller's buffer must be in the following format:

Bits 0-13 define the value to be output
Bits 14-23 define the address of the DAC channel

It should be noted that random or sequential addresses may be employed in the contiguous buffer.

Random Mode - This service enables the user to write one DAC with the data from a user specified memory location. The data must be a single precision floating point number.

The FORTRAN call statement is:

```
CALL  WDACR (DACADD, USRLOC)
```

where:

DACADD - the address of the DAC to be written

USRLOC - the variable containing the data to be written.

The Assembly Language call is:

```
SPB     '36
```

where:

The A accumulator is set to the following value prior to calling the monitor.

Bits 0-13 define the value to be output
Bits 14-23 define the address of the DAC channel

The service routine outputs a fixed point value to the DAC channel whose address is specified by the application program.

6.3.3 Write Clock Word and Select Mode

This service enables application programs to set the real time clock initial count register. One-shot or iterative modes for that device may also be selected. It should be noted that application jobs that require control of this clock may not contain periodic scheduled tasks (See Section 4 for Task Control Functions).

The FORTRAN call statement is:

```
CALL  WCLK (ICOUNT, MODE)
```

where:

ICOUNT - The initial count (right-justified integer). This count will be the time in microseconds between clock interrupts.

MODE - Mode selection argument (0 = iterative, 1 = one-shot).

The Assembly Language call is:

```
SPB  '33
```

where:

The A accumulator is set to the value to be written to the clock, and the B accumulator is set to 0 to specify the iterative mode or to 1 to specify the one-shot mode.

6.3.4 Read Clock Word

This service enables application programs to read the current count of the continuous running Real Time Clock (RTC). The continuous running RTC counts down from positive full scale to negative full scale and automatically recycles.

The FORTRAN call statement is:

```
CALL  RCLK (COUNT)
```

where:

COUNT - The variable in which the right-adjusted integer representing the current RTC count is returned.

The Assembly Language call is:

```
SPB     '32
```

where:

The A accumulator is set to the current clock value and is returned to the calling application program.

6.3.5 Clock Mode Control

This service enables application programs to control the modes of the RTC.

The FORTRAN call statement is:

```
CALL  CLKMOD (I)
```

where:

I - Integer selecting the mode as indicated below:

I	Mode
1	Run
2	Hold

The Assembly Language call is:

```
SPB     '34
```

where:

The caller's A accumulator is set to 1 to specify the Run mode or 2 to specify the Hold mode.

NOTE:

The previous value sent to MPCS for a Write Digital Clock service is used to reset the clock prior to placing it in the Run mode. If no previous call was made to set a clock value, the control function for a Run mode is ignored.

6.3.6 Read Digital Word

This service enables a user to read 24 sense lines in parallel and stores the result as a 24 bit integer in a user specified memory location.

The FORTRAN call statement is:

CALL RDW (IW)

where:

IW - The variable into which the result is to be stored.

The Assembly Language call is:

```
SPB     '26
```

where:

The A accumulator is set to the value read and is returned to the caller.

6.3.7 Write Digital Word

This service enables the user to set the 24 control lines in parallel to a configuration defined by a 24 bit integer specified by an application program.

The FORTRAN call statement is:

CALL WDW (IVAL)

where:

IVAL - An integer that corresponds to setting of control lines 0 to 23.

The Assembly Language call is:

```
SPB     '30
```

where:

The caller's A accumulator must be set to the 24 bit value prior to requesting this MPCS service.

6.3.8 Write Control Lines

This service enables the user to selectively set a control line whose address is in the range 0 to 23. These control lines may be reset by service routine WDW (see Section 6.3.7).

The FORTRAN call statement is:

```
CALL  WCL (ICL)
```

where:

ICL - An integer that corresponds to the address of the line that is to be set; the remaining control lines retain their original setting.

The Assembly Language call is:

```
SPB     '31
```

where:

The caller's A accumulator must designate the number of the control line (0-23) that is to be set.

6.3.9 Read Sense Lines

This service enables the user to determine the state of sense lines whose addresses are in the range of 0 to 23. A user memory location is set to 0 if the specified sense line is false and set to 1 if the sense line is true.

The FORTRAN call statement is:

```
CALL  RSL (IVAL, ISL)
```

where:

IVAL - The integer into which the result of the test is to be stored.

ISL - An integer corresponding to the address of the sense line to be read.

The Assembly Language call is:

```
SPB     '27
```

where:

The caller sets the A accumulator to an octal integer in the range of 0 to 23 that corresponds to the sense line to be tested. Upon returning to the caller, the A accumulator will contain 0 if the sense line was not set or 1 if the sense line was set.

6.4 Analog Setup and Control Services

6.4.1 Read Digital Ratiometer

This service enables the user to read the digital ratiometer and have the reading stored in a specified memory location. The reading will be returned to the user in floating point format with a 100 volt reading being normalized to unity.

The FORTRAN call statement is:

```
CALL  RDRM (USRLOC) or
CALL  RDVM (USRLOC)
```

where:

USRLOC - The real variable in which the DVM reading will be returned to the user.

6.4.2 Read Analog Component

This service enables a user to read a number of sequential AD-4 component addresses within a class and have them stored in a specified memory buffer. The data will be stored in floating point format with 100 volts being equal to unity in the digital computer.

The FORTRAN call statement is:

```
CALL  READ (CXXX, BUF, NO)
```

where:

CXXX - The address of the first AD-4 component to read. C is the class designator followed by XXX which are three integers required to complete the AD-4 addresses exactly as they exist on the AD-4 patchboard. Listed below are the AD-4 class designators.

C	Class
A	+Amplifier
B	- Amplifier
D	Digital Coefficient
P	Potentiometer Coefficient
S	Summing Junction
F	Function Generator Summing Junction
T	Trunk (TR. A = D)
R	Resolver

BUF - The name of the array into which the floating point readings will be read.

NO - The number of consecutive AD-4 components to be read within the designated class.

6.4.3 Read AD-4 Status

This service enables a user to test the current mode status of the AD-4.

The FORTRAN call statement is:

```
CALL  RDANST (MODE, RESULT)
```

where:

MODE - The integer indicating which of the modes is to be tested as indicated in the table below:

MODE	Tested Status
1	Digital Control Possible

MODE	Tested Status
2	Null Error
3	Servo
4	Invalid Address in AD-4
5	Control of AD-4 Clock System via Logic Exec
6	Analog Mode Operate
7	Analog Mode Hold
8	Analog Mode Initial Condition
9	Analog Mode Problem Verify
10	Logic Mode Run
11	Logic Mode Stop
12	Logic Mode Load
13	Time Scale X1
14	Time Scale X10
15	Time Scale X100
16	Time Scale X1000

RESULT - Variable into which the result of the test will be returned. 0 will be returned if the tested mode is not ON and 1 will be returned if it is.

The Assembly Language call is:

```
SPB     '21
```

where:

Caller's accumulator must be set to the desired AD-4 mode. Data from the AD-4 is returned in the A accumulator.

6.4.4 Set AD-4 Mode

This service enables a user to command the AD-4 to one of the four computational modes, one of the three logic modes, or one of the four available time scales.

The FORTRAN call statement is:

```
CALL  MODE (I)
```

where:

I - Integer corresponding to the desired mode as indicated below:

I	Mode
0	Analog Initial Condition
1	Analog Hold
2	Analog Operate
3	Problem Verify and Initial Condition
4	Logic Load
8	Logic Stop
12	Logic Run
16	Time Scale X1
32	Time Scale X10
48	Time Scale X100
64	Time Scale X1000

Modes are additive: thus, I = 78 = 2+12+64 would yield analog operate, logic run, and time scale X1000. The analog mode is set with each call, but the logic mode and time scale remain unchanged unless a change is specifically requested.

The Assembly Language call is:

```
        SPB     '20
```

where:

Caller's A accumulator must be set to the desired AD-4 mode and the B accumulator bits 0-15 set to the output data.

6.4.5 Set Potentiometer

This service enables a user to set a sequential number of potentiometers on the AD-4 to values obtained from a table supplied by the user. The data in the table will be floating point numbers normalized to unity.

The FORTRAN call statement is:

```
        CALL  SETPOT (PXXX, BUF, I)
```

where:

PXXX - The address (in ASCII code) of the first potentiometer to be set by this call.

BUF - Name of the array in the calling program containing the desired potentiometer settings.

I - The number of consecutive potentiometers to be set by this call.

7. MPCS CALLS FOR FILE MANAGEMENT

This section will be added at a later date.

8. MPCS CALLS FOR UTILITY SERVICES

8.1 Normal End of Job

The termination of a user program for a normal end of job is accomplished by a MPCS service. Upon request of the service, MPCS resets all external interrupt levels (if any) and enters the Control Command Processor (CCP) for execution. This service does not reset any of the system Logical Device Numbers to their standard assignments.

The call for this service is:

```
SPB     2
```

8.2 Abnormal End of Job

This service performs the normal end of job functions with the addition of a complete register dump of the caller's program at the time of the request. If any external interrupt level service routines were active (entered and not completed) prior to the call, the contents of the individual registers are also dumped in the sequence in which each interrupt became active. Output is made to the console teletype of the processor in which the job resides.

The call for this service is:

```
SPB     '3
```

The register dump is in the form of:

```
INST CTR XXXXXXXX INSTR.   XXXXXXXX  ZIP      XXXXXXXX
REGS     XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX XXXXXXXX
```

where:

REGS are in the following order: A accumulator, B accumulator, Index one, Index two, and Index three.

ZIP is a special memory cell whose address is absolute '120, and may be used as a pseudo register. It is saved and restored by MPCS in processing interrupts.

8.3 File Assignment

This service enables a user to accomplish the assignment or re-assignment of standard or user Logical Device Numbers to any of the physical devices associated with the system.

The call for this service is:

```
SPB     '4
```

where:

The A accumulator contains the LDN currently assigned to the physical device and the B accumulator contains the system or user LDN that is associated with the application program FCT.

If the user LDN assignment table is full, MPCS will set the A accumulator to a negative value and return to the caller. If the contents of a user's A accumulator does not represent a valid system LDN, the caller's program is aborted.

8.4 Magnetic Tape Mode Control

This service enables a user during the execution of his program to alter the various modes for reading and writing magnetic tapes. The mode that is set by the user remains in effect until the command to reset standard system assignment is given to the Control Command Processor.

The call for this service is:

```
SPB     '5
```

where:

The A accumulator must be set to the value of the CEU second word format for magnetic tape as follows:

0 ———————— 0	**	0 ———— 0	**
0 — 15	16 — 17	18 — 21	22 — 23

** Magnetic tape density and character per word

BITS 16 & 17

00 = 200BPI
*01 = 556BPI
10 = 800BPI

BITS 22 & 23

01 = 1 character per word
10 = 2 character per word
11 = 3 character per word
*00 = 4 character per word

* STANDARD SYSTEM MODE

8.5 Set BARs

This service enables a user to change Bank Address Register (BAR) 1, 2, or 3.

The call sequence for this service is:

```
SPB     6
```

where:

The A accumulator is set prior to the call with the required BAR configuration. Application programs may not alter BAR 0.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
		BAR 0						BAR 1						BAR 2						BAR 3			

8.6 Get Current Sector of Disk Scratch Area

This service enables application programs and system processors to obtain the next available sector number for the disk scratch area.

The call for this service is:

```
SPB'7
```

where:

The next available sector number is returned to the caller in the A accumulator.

8.7 Snapshot Dump

The snapshot dump service causes MPCS to produce a listing of the A and B accumulators, the three index registers, the pseudo register, ZIP, and the program counter. The format of the output is similar to the information printed for an Abnormal End of Job request. The listing device is the console teletype of the processor in which the job resides.

The call for this service is:

```
SPB     '13
```

8.8 Convert Binary to TASCII

This service converts a 24 bit binary word into 8 TASCII characters. The A accumulator is loaded with the word to be converted prior to calling MPCS. The 8 characters are returned to the caller in the A and B accumulators.

The call for this service is:

```
SPB     '14
```

8.9 ASCII to AD-4 Address Conversion

These services provide users a means of converting the format required by the AD-4, or of converting an address read from the AD-4 to ASCII code. These services must be called from an assembly language routine.

The calling sequences are described below:

ASCII to AD-4

```
LAA     ASC             ASCII address in A accumulator
SPB     $ASAD4          Convert to AD-4 address
STA     AD4             AD-4 address in A accumulator
```

AD-4 to ASCII

```
LAA     AD4             AD-4 address in A accumulator
SPB     $AD4AS          Convert to ASCII
STA     ASC             ASCII address in A accumulator
```

8.10 Floating Point to AD-4 Integer Conversion

These services provide users a means of converting between single precision floating point data and the integer format required by the AD-4.

These services must be called from an assembly language routine. The calling sequences are described below.

For floating point to AD-4 integer conversion:

```
ELO     VAL         Load floating point value
SPB     $FPAD4      Convert to AD-4 integer
```

On return, the result is in the A accumulator.

For AD-4 integer to floating point conversion:

```
LAA     VAL            Load AD-4 integer
SPB     $AD4FP         Convert to floating point
```

On return, the result is in the EAU A accumulator.

8.11 Digital Delay

This service enables a user to invoke a digital delay in milliseconds. Timing is based on the SEL840 MP cycle time.

The call statement is:

```
CALL  DELAY (ICOUNT)
```

where:

ICOUNT - The number of milliseconds of delay.

PRECEDING PAGE BLANK NOT FILMED

9. MPCS ASSEMBLY LANGUAGE CALL SEQUENCE SUMMARY

CALL	FUNCTION	EXPLANATION
SPB'0	Unassigned	Caller is aborted
SPB'1	I/O Requests	Caller's A accumulator must be set to the address of the Function Control Table (FCT).
SPB'2	Normal end of job	Control Command Processor is entered for execution.
SPB'3	Abnormal end of job	Outputs a dump of caller's registers and enters the Control Command Processor for execution.
SPB'4	File assignment	Caller's A accumulator must be set to object system LDN and the B accumulator set to the user/system LDN that is to be re-assigned.
SPB'5	Magnetic tape mode control	Caller's A accumulator must be set to the proper bit pattern of the CEU second word format.
SPB'6	Change BARs	Caller's A accumulator must be loaded with the new BAR addresses. BAR 0 cannot be changed.
SPB'7	Get current scratch disk sector number	Caller's A accumulator is returned set to the next available sector number.
SPB'10 through SPB'12	Unassigned	Caller is aborted.
SPB'13	Snapshot	Performs a dump of the caller's registers similar to an abort dump but returns to the caller's next sequential instruction.

CALL	FUNCTION	EXPLANATION
SPB'14	Binary to TASCII	Caller's A accumulator is set to a 24 bit word to be converted to printable octal equivalence. The 8 characters are returned in the A & B accumulators.
SPB'15	External Interrupt Return	MPCS restores the status of the interrupted program and returns to its execution.
SPB'16	External Interrupt Connect	Caller's A accumulator must be set to the interrupt number and the B accumulator set to the address of the first instruction to execute at interrupt time.
SPB'17	Service External Interrupts	Caller's A accumulator must be set to positive to enable or negative to disable the interrupt level and the B accumulator set to levels bit pattern.
SPB'20	Write to AD-4	Caller's A accumulator must be set to the desired AD-4 op code and the B accumulator set to the output data.
SPB'21	Read from AD-4	Caller's A accumulator must be set to the desired AD-4 op code. Data from the AD-4 is returned in the A accumulator.
SPB'22	Unassigned	Caller is aborted
SPB'23	Unassigned	Caller is aborted
SPB'24	Unassigned	Caller is aborted
SPB'25	Unassigned	Caller is aborted
SPB'26	Read Digital Word	MPCS sets the caller's A accumulator to the 24 bit value read.
SPB'27	Read Sense Line	Caller's A accumulator must be set to the number of the line to be sensed (0-23). MPCS sets the caller's A accumulator to 0 if not set or 1 if set.

CALL	FUNCTION	EXPLANATION
SPB'30	Write Digital Word	Caller's A accumulator must be set to 24 bit value prior to calling the monitor.
SPB'31	Write Control Line	Caller's A accumulator must be set to the number of the control line (0-23) to be acted upon.
SPB'32	Read Digital Clock	MPCS sets the caller's A accumulator to the value read.
SPB'33	Write Digital Clock	Caller's A accumulator must be set to the value to output to the clock and the B accumulator set to 0 for iterative or 1 for one-shot mode.
SPB'34	Set Digital Clock Mode	Caller's A accumulator must be set to 1 for run or 2 for hold prior to calling MPCS.
SPB'35	Read RTE Random	The A accumulator contains the RTE channel to read with bit 0 set. The fixed point value read is returned by MPCS in the A accumulator.
	Read RTE Sequential	The A accumulator contains the starting RTE channel with bit 0 reset. The B accumulator contains the address of the buffer where the fixed point values will be stored. Index 1 contains negative number of channels to read.
SPB'36	Write DAC Random	Caller's A accumulator contains the value in bits 0-13, bits 14-23 contain the address of the DAC channel to write.
SPB'37	Write DAC Sequential	The caller's A accumulator contains the address of the buffer. Each word in the buffer is set up like the SPB' 36 A accumulator. The B accumulator contains the number of DAC's to write.