File Number 1800-36
Form C26-3718-0

IBM Systems Reference Library

IBM 1800 Multiprogramming Executive System Introduction

The purpose of this publication is to provide under one cover a concise overview of the basic concepts, functions and general organization of the IBM 1800 Multiprogramming Executive (MPX) System. It is intended as a guide in understanding the system's capabilities and planning for their use.

The first section defines the MPX system, its operating environments and system requirements, and highlights the many advantages of this FORTRAN oriented, disk-based multiprogramming system. Subsequent sections discuss system concepts and system components. A summary of all MPX calling statements used is included in Appendix A, while Appendix B discusses MPX recommended interrupt level assignment practice.

Detailed specifications on the use of this system will be supplied in later publications.

First Edition

Specifications contained herein are subject to change from time to time. Any such change will be reported in subsequent revisions or Technical Newsletters.

Requests for copies of IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A form is provided at the back of this publication for reader's comments. If the form has been removed, comments may be addressed to IBM Corporation, Programming Publications, Department 232, San Jose, California 95114.

CONTENTS

The IBM 1800 Multiprogramming Executive (MPX) System is a 26-area FORTRAN oriented disk-based operating system designed to provide simultaneous disk operation, maximum throughput, efficiency and economy of operation. It is intended for process control and data acquisition applications which have high system usage requirements (see Figure 1).

Multiprogramming is regulated on a basis of I/O operation. When an input/output operation is initiated in one core storage area, that area can be placed in a suspended state until the I/O function is completed. Concurrently with this, a program loaded into a lower priority core storage area can also be executed. Since a core storage area can be assigned to an interrupt level, the programmed interrupt technique is used by the system to direct these levels of operation, thereby controlling the execution of any one of 24 possible core storage areas at any moment in time.

The MPX system also provides for unlimited queueing of I/O operations as well as the ability to achieve maximum overlap of I/O and computing. In addition, interrupt programs may be handled on a queueing basis, and executed on interrupt levels within a hierarchy of priority levels chosen by the user. Queueing and I/O overlap as used in the MPX system provide the ability to gain the full advantage of 9 cycle-stealing data channels, 24 interrupt levels, and, in general, the real potential of the IBM 1800 Data Acquisition and Control System.

MPX will divide core storage into several areas as follows:

- An area of permanent core storage for the resident MPX System Executive.
- A special core load area (SPAR) which enables the user to modify his high-priority interrupt subroutines on-line.
- Up to 23 partitioned core load areas which are transient in nature, and provide for user-written process core loads. Programs executed in these areas are executed on an interrupt level.
- A variable area of core storage in which mainline level and interrupt level programs are executed. This could contain a process or batch processing monitor program or an out-of-core interrupt program. The variable core area is time-shared between mainline process core loads and the Batch Processing Monitor. An interrupt will cause the mainline program to be saved on disk before out-of-core interrupt program is loaded.

Some of the many advantages of the MPX system are:

1. High throughput
2. Fast response
3. Optimum use of available processor-controller time
4. Automatic program scheduling -- low scheduling burdens
5. Ability to modify system on-line
6. Ability to execute user-written on-line hardware diagnostics
7. Time-sharing of foreground and background (Batch Processing Monitor) operations

MINIMUM SYSTEM REQUIREMENTS

To obtain the full capabilities of the IBM 1800 Multiprogramming Executive System, the machine configuration shall be at least:

1 IBM 1801/1802 Processor-Controller with a minimum of 24K words of core storage
1 IBM 1053 Printer or IBM 1816 Printer-Keyboard
1 IBM 1442 Card Read Punch
1 IBM 2310 Disk Storage Unit, Model A2 (standard — two disk drives) or C2 (fast access — two disk drives)

Machine Features Supported

In addition to the above, the following optional machine units and features are supported by the MPX system:

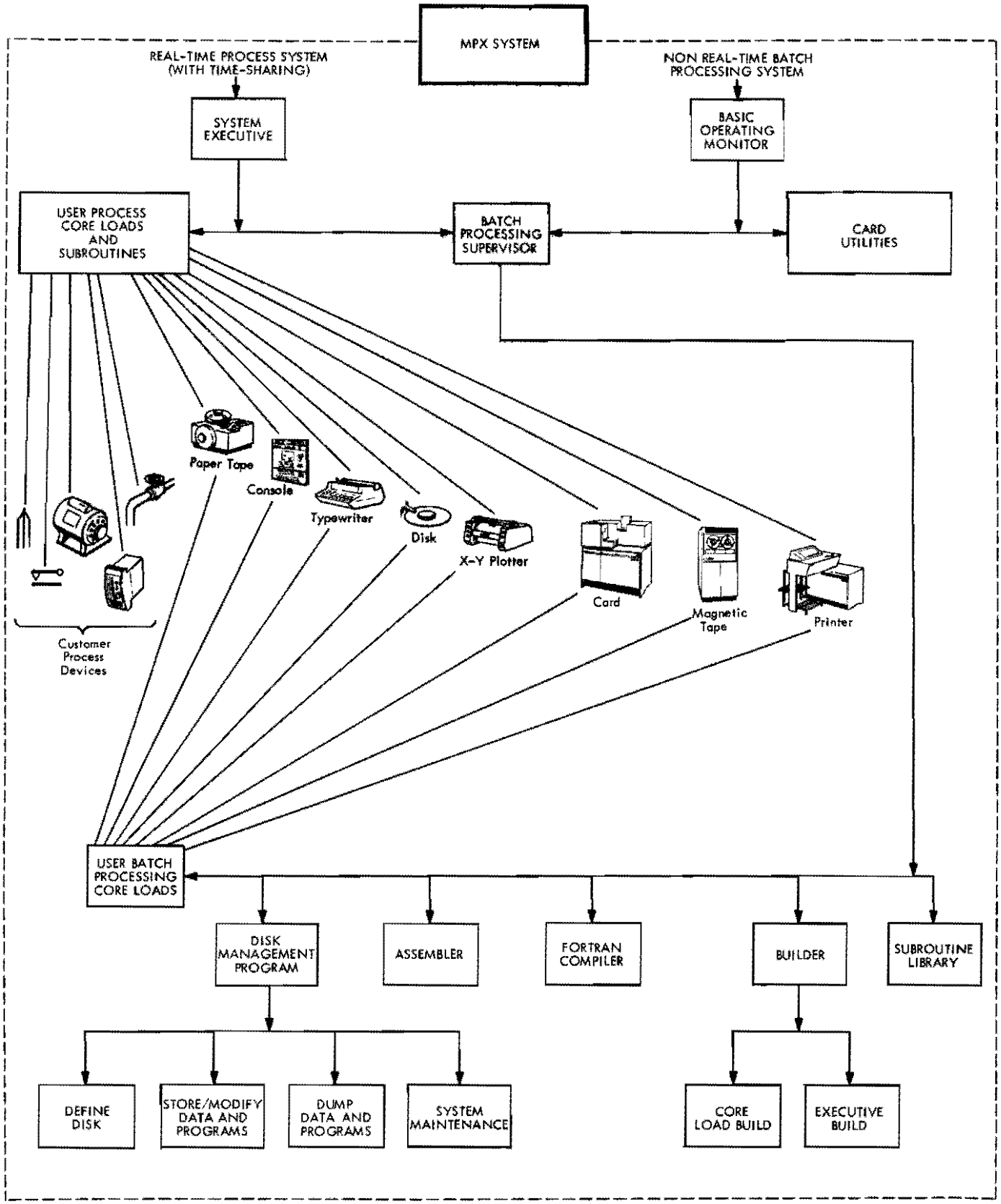

Figure 1. The IBM 1800 Multiprogramming Executive System

- Additional core storage (up to a maximum of 32,768 words)
- Additional disk drive for IBM 2310 Disk Storage Unit (up to a maximum of three disk drives)
- Additional IBM 1442 Card Read Punch Unit (up to a total of 2)
- Additional IBM 1816 Printer-Keyboard (up to a total of 2)
- Additional IBM 1053 Printer Units (up to a total of eight 1053s and 1816s)
- Additional Data Channels (up to a total of 9)
- Additional Interrupt Levels (up to a maximum of 24)
- Multiplexer Unit (Solid-state or Relay)
- Analog-Digital Converter (up to a total of 2)
- Digital-Analog Output
- Digital Input
- Comparator
- IBM 1443 Printer Unit
- IBM 2401 Magnetic Tape Units (maximum of 2)
- IBM 1627 Plotter Unit
- IBM 1054 Paper Tape Reader
- IBM 1055 Paper Tape Punch

OPERATING ENVIRONMENTS

The IBM 1800 Multiprogramming Executive System is composed, essentially, of two main parts: (1) A System Executive and (2) a Batch Processing Monitor. It is through the System Executive that process control and data acquisition applications are serviced in a real-time process mode, with the Batch Processing Monitor operative in the time-shared mode; or the Batch Processing Monitor can act as an independent programming system to provide data processing functions in a standard batch processing mode. Each of these two modes is brought into play by an appropriate and corresponding system generation procedure. The user elects the option of constructing a real-time process or batch processing system tailored to individual specifications.

Real-time Process Mode. In real-time processing, inputs arrive randomly from a process being monitored to the processor-controller. The computer rapidly responds to each input, usually by conveying an output back to the process. This is in contrast with conventional batch processing where groups of input data are processed by passes through the computer. The notion of real-time usually implies that a processor-controller is responding to inputs as they occur in the physical world.

MPX operates in this mode under the control of the System Executive. In a real-time environment, user-written programs are continuously monitoring or controlling a process. The machine is also permitted to be time-shared by process and batch process work: that is, batch processing jobs can be interleaved simultaneously with the real-time process being monitored. Whenever variable core is not required for a process program, the Batch Processing Monitor may be brought into service. All programs executed are fetched from the system resident disk cartridge(s).

Batch Processing Mode. The batch processing MPX system operates in this nonreal-time mode under the control of the Basic Operating Monitor (BOM) as a dedicated Batch Processing Monitor System. Typical batch processing operations are assemblies, compilations, disk management functions and the execution of batch processing programs.

A batch processing system can be used to test problem programs before they are permanently catalogued on the system resident disk cartridge; to execute problem programs that require the full capacity of available disk drives for data files, or to process problem programs that are used so infrequently that their real-time storage is not justified. It is also initially used to construct a real-time process disk resident system.

MULTIPROGRAMMING

In most current computer installations, process and data processing functions are performed sequentially; that is, a new function is not begun until the current function is completed. The average function performed by a computing system requires at any given moment only a fraction of the total available resources of the system. Many parts of the system are, therefore, often idle for significant periods of time. For example, a data conversion and printing function requires only a fraction of the storage space and input-output devices available in the system, and intermittent use of the processor-controller.

To increase throughput, the IBM 1800 Multiprogramming Executive System enables programs, core storage space, input-output facilities and control of the processor-controller to be allocated and concurrently shared among several process functions. These facilities permit multiprogramming; that is, they permit several process functions to be performed concurrently and to share the basic resources of the computing system. The MPX operating system helps to ensure that as much of the total system as possible is kept busy performing productive work as much of the time as possible. This is accomplished by efficiently allocating the available resources of the system to more than one function, and switching control from one function to another as a delay is encountered while awaiting an event, such as the completion of an input-output operation, or the end of a timing interval. Among the services provided by MPX to allow concurrent operation are:

- Loading programs and routines into main storage
- Scheduling the use of programs and routines in main storage
- Switching control of the processor-controller from one function to another, based on I/O operation
- Controlling the execution of the various functions in accordance with a defined hierarchy of priority

Method of Operation

MPX provides the overall capability of multiprogramming with a fixed number of jobs. This means that a maximum of twenty-four programs can be run concurrently within a single computing system having only one processor-controller.

The system permits from one to twenty-four separately scheduled programs to reside in their own predefined sections of core storage. A program occupies a discrete and contiguous area of core storage; the number of core areas used as well as the size of each area is determined when the system is generated.

Two types of programs are used in multiprogramming: foreground and background. Background programs are normally initiated by the Batch Processing Monitor from the batched-job input stream. Foreground programs do not execute from a stack; they are brought into operation by the occurrence of random or periodic real-time events. Background and foreground programs initiate and terminate independently of one another. The background job is time-shared with the lowest priority foreground job; that is, when there is no foreground job available to be run in the lowest priority foreground area (variable core), the background job is brought into core and executed. If a foreground job later requires that area, the background job is saved on bulk storage.

MPX is capable of concurrently operating one background and one or more (up to twenty-four) foreground programs. Priority for processor-controller processing is based on a multi-level interrupt level hierarchy controlled by the Executive Director, with foreground programs having priority over background programs. All programs operate with interruptions enabled. When an interrupt occurs, the Director gains control, processes the interrupt and gives control to the highest priority program linked to a specific discrete area or partition of core storage in which it is to reside at execution time. Control is taken away from a high priority program when that program encounters a condition that prevents continuation of processing until a specified event has oc-

curred. For example, this condition would occur when a READ operation is issued to a disk storage unit. Control is taken away from a lower priority program when an event on which a higher priority program was waiting has been completed. In the above example, control would return to the high priority program when the READ I/O operation has been completed.

Multiple Core Load Areas

By dividing core storage into as many as twenty-three partitioned core load areas (in addition to a fixed area for the resident executive, a SPAR area and a variable area), twenty-four separate programs can be executed concurrently. This permits I/O overlap during program loading, and improved throughput, since core exchanges would not be required, except in variable core. The concept of core load areas provides for efficient use of the equipment, while allowing the system to handle multiple jobs in rapid succession.

Program execution in a core load area can be based on interval timer rundown, process interrupts, programmed interrupts, or the queueing of a program by another program. During the loading and execution phase of a program assigned to a particular interrupt level, programs linked to that level and competing for processor-controller time will be queued. There is one queue table for each interrupt level; programs placed in the queue can be assigned with a priority within a level.

Execution of process programs will be initially triggered by a programmed interrupt which is activated when a disk I/O operation is completed. Linking facilities between core loads within each core load area are available. Each core load area will have its own COMMON area and LOCAL (load-on-call) capability.

Figure 2 illustrates the position of core load areas within a general core organization layout of an MPX system.

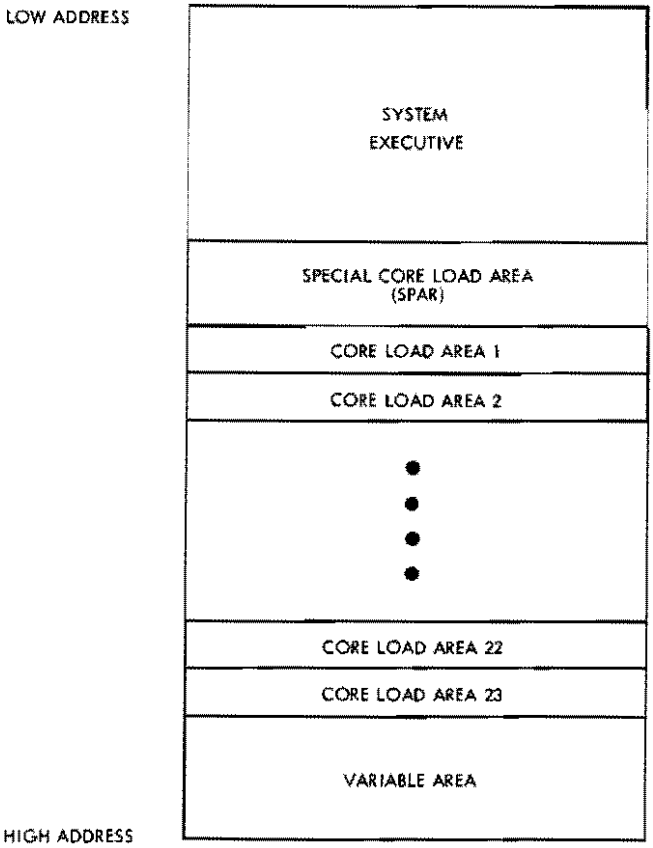

Figure 2. General Core Organization of MPX System

TIME-SHARING

In many industrial control installations, the real-time computer control system will have spare capacity to perform background jobs. Time-sharing techniques can thus be employed whenever idle or free processor-controller time is available in a given system environment to offer the user the kind of service he requires.

The notion of time-sharing also implies the sharing of computer resources, since not only time but primary and secondary storage as well as input-output facilities are also shared.

When idle time is available in the IBM 1800 Multiprogramming Executive System, control is automatically transferred to an independent Batch Processing Monitor System which is identical to any stack-job monitor system. All assembling, compiling and system activities can now be executed under the control of the Batch Processing Monitor. Performing such jobs "time-shared" has a distinct advantage in that any time not required for process control functions can be used for batch processing functions. Also, since process control problem programs and strategies often tend to change, the time-sharing mode makes it possible to modify these programs and strategies at the process on-line installation without taking the computer off-line. Time-sharing is another feature of the MPX system which helps to optimize the utilization of the IBM 1800 Data Acquisition and Control System.

ROLE OF THE SYSTEM EXECUTIVE

The System Executive constitutes the main framework of a process real-time MPX system, and must be resident in permanent core storage before any continuous and coordinated real-time processing can take place. Other portions of the system are brought into core from bulk storage as they are required to perform specific functions.

The Executive is extremely flexible. By the system generation process, a user can build his own unique system tailored to individual requirements. In this way, core will not be wasted for the user who does not desire certain optional features. Each operating system consists of a choice of control routines and programming facilities that are closely integrated with a selection of processing, storage, and input-output options to form a balanced system for a particular range of applications. The process real-time system is generated using standard language and editing facilities.

The user may include frequently-called subroutines, high-priority interrupt servicing routines and other user-written programs in the System Executive to make the most effective use of his control system.

A typical Executive may consist of the following constituent parts as illustrated in Figure 3. The function of each individual part will now be described.

Executive I/O. This is a set of input-output subroutines and tables which provides a rapid and easy method for the user to reference the various data processing I/O devices for input and output of data. It includes:

- BULKN (Bulk Storage Subroutine: performs all reading from and writing to the IBM 2310 Disk Storage Unit)

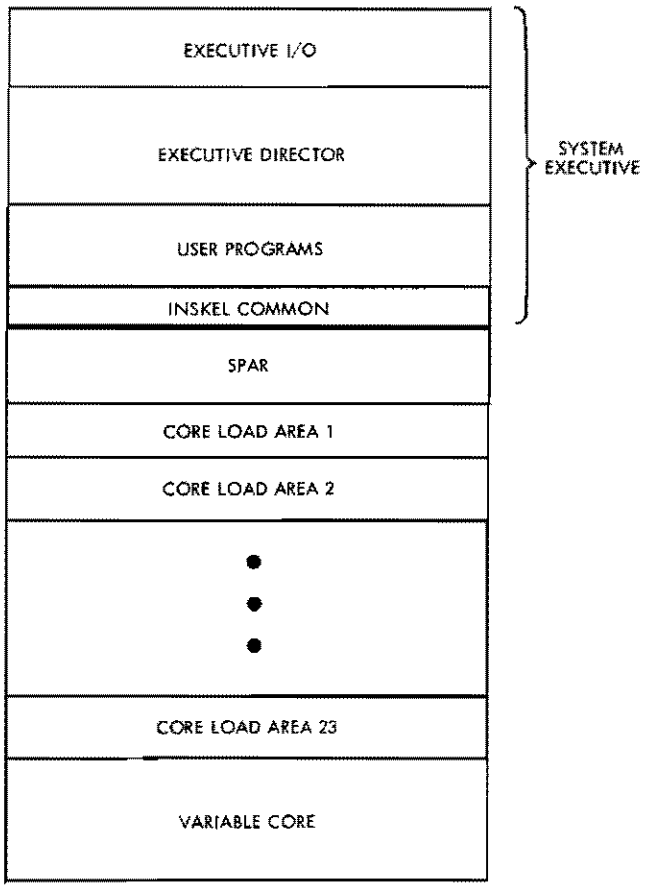

Figure 3. An MPX Real-time System - Illustrating the Four Component Parts of the System Executive

- TYPEN/WRTYN (Printer-Keyboard Subroutine: transfers data to and from the IBM 1053 and IBM 1816 Printer-Keyboard)
- PRNTN (Printer Subroutine: handles all print and carriage control functions relative to the IBM 1443 Printer)
- I/O Common Routines
- Error Alert Control (EAC) program package: provides standard error detection and recovery procedures

These and other basic system routines make up the Executive I/O package which corresponds to an identical set of input-output subroutines used by BOM.

Executive Director. The Director is the control center of the multiprogramming operating system. Its primary function is to perform a variety of services for other parts of the system, including problem programs. It coordinates and controls the performance of these services in such a way as to ensure efficient and coordinated use of the physical and programming facilities or resources of the system.

To perform its functions, the Director receives control of the processor-controller by way of an interruption or a call. The interrupt may result from a specific request for services from another part of the operating system; from a problem program; an automatic interrupt that occurs at the completion of an input-output operation; or from a random or periodic event that occurs outside the system. Automatic interruptions associated with a hierarchy of priorities, in general, enable the Director to coordinate its control over the physical and programming resources of the system.

A service performed by the Director may be specifically requested by a program, such as a request to load another program, or it may be a service that is automatically provided when a contingency occurs, such as attempting to recover from an error condition. Among the services that may be provided by the Executive director are the following:

- Loading programs into main storage
- Sharing areas of main storage among routines that need not be in main storage at the same time
- Controlling the concurrent execution of programs and routines
- Providing the time of day and other timing services, such as keeping track of the time at which a particular operation is to be performed
- Making the system available to the Batch Proceessing Monitor

User-written Programs. The user has the option to include as many programs and subroutines as possible in the System Executive for reasons of frequent usage, rapid response and optimum utilization of bulk storage. These may be:

- Interrupt Servicing Subroutines
- Time-related Subroutines
- Special trace subroutine
- IBM-supplied arithmetic, I/O and other subroutines
- Other user-written subroutines

Such routines are first assembled/compiled in relocatable format and stored. At Executive build time, they are bodily incorporated into the System Executive.

INSKEL COMMON. A unique labelled common area can be set aside for communications between the various types of core loads used in the system. It can be referenced by any process or data processing program. The size of INSKEL COMMON is user-defined when the system is built.

On-line Modification of In-core Interrupt Subroutines

A special core load area called SPAR is provided, at the user's option, which can be considered as an extension to the System Executive. This area can contain frequently-employed interrupt servicing routines or other routines that may be subject to change in a process control environment. By loading these routines into SPAR (rather than having them assembled with the System Executive), the user gains the ability to change or switch any high priority subroutine there to meet the demands of the system being

controlled. This important feature of MPX facilitates modification of in-core interrupt subroutines without the necessity of a system regeneration.

SPAR is fixed in size by the user when the system is initially generated. A core load containing any user-selected servicing subroutines and routines referenced by these subroutines is loaded during the cold start procedure.

It will henceforth always remain in core, unless a specific call is made to the Executive Director to load a new SPAR core load. Subroutines loaded into the special core load area will not be used by any other area core load.

PROGRAM SEGMENTATION--CORE LOADS

When the core storage size of a multiprogramming system is not sufficient to hold all the necessary process control programs at one time, some form of program read-in scheme is normally employed whereby programs, upon demand, are brought in from bulk storage. Very often, such a scheme will reserve part of core storage for program read-in, and divide it into fixed-length partitioned core storage areas which form a repository for programs of a limited size. This forces the multiprogrammer to make a segmentation of his program into one or more parts which will fit into the fixed-size core areas at execution time. The program segments can be of any size as long as they do not exceed the core storage area size.

This technique is employed in the MPX system where programs are formed into smaller units termed core loads. A core load is, by definition, an executable program or portion of a program which performs some user function. It is not necessarily a program in its entirety because the program may be too large to fit into a core load area in one piece for execution. The core load is unique in that it is stored on disk in core image format to facilitate rapid loading for execution.

Figure 4 illustrates a typical type of core load commonly used in MPX. A core load may contain other subroutines that are not associated with the main program--that is, subroutines not otherwise available in core (either included in the System Executive or employed in the form of LOCAL subprograms). A typical core load may consist of a mainline or interrupt program, and required subroutines that are not included with the System Executive.

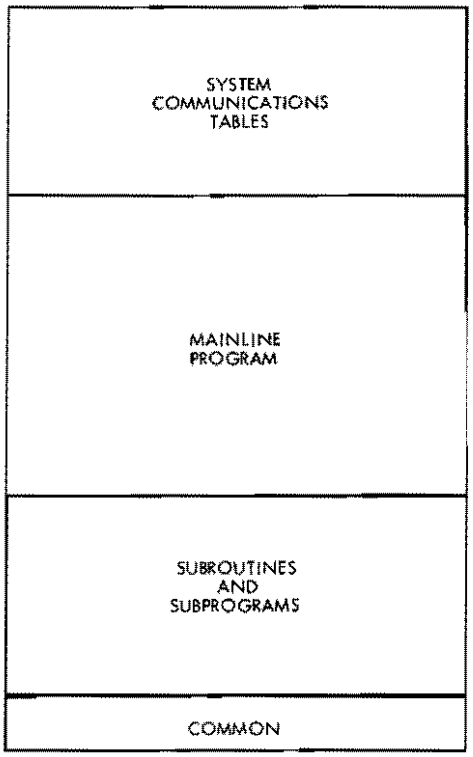

Figure 4. A Typical Core Load Used in MPX

Core loads are important in real-time systems for the following reasons:

- They are permanently built and stored on bulk storage
- It is possible to reference the program by name
- No compiling/assembling is needed at execution time
- Rapid execution

REENTRANT CODING

One of the basic problems that arises in multi-level programming is that different levels of operation require the use of the same subroutine. Suppose a block of coding, servicing interrupt level 3, calls subroutine ALPHA. Before completing subroutine ALPHA, a higher priority interrupt at level 1 occurs, which also calls the same subroutine ALPHA. If

ALPHA is coded in the conventional manner, the argument pointers, temporary core storage locations, computed instructions, and the return point for the first call to ALPHA are destroyed by the second call or reentry of ALPHA. Reentry, then, is defined as the use of a subroutine before completion of a previous call to that subroutine. A similar situation occurs when a program calls itself, which is known as recursion.

A sophisticated method of reentrant coding using level work areas is devised for MPX to allow one subroutine to be entered at any time and from any interrupt level without loss of results. All system subroutines that are required on multiple levels in MPX are fully reentrant. That is, they can be called repeatedly by different interrupt subroutines at different levels; they are automatically reenterable, and automatically keep guard of the partial results acquired at the time they were interrupted. Within the MPX philosophy, a single subroutine can be used simultaneously at all 26 levels, while it is servicing any other level. The automatic accounting of the partial results of subroutines is a very significant step forward which is made possible by the programming structure of MPX.

Many of the subroutines in the MPX Subroutine Library are supplied in both reentrant and non-reentrant form. The non-reentrant versions have been added to give the user greater flexibility in the choice of execution speeds.

The choice of selecting a reentrant or non-reentrant subroutine for use with a specific program should be accomplished with care. In general, the reentrant version of a subroutine should be used if the subroutine is to be called from different levels, included in the System Executive, or included in the Special Core Load Area (SPAR). Any type of subroutine can be included within a core load.

MPX also provides the user with the capability to tailor each core load to the specific uses of that core load. For example, non-reentrant subroutines may be specified for inclusion in core loads where speed is more significant than core size, even though corresponding reentrant subroutines may be resident in the System Executive.

Some of the advantages of MPX reentrant coding may be summarized as follows:

- All levels of operation may execute any given subroutine
- The size of the overall system in core and on bulk storage is reduced
- All subroutines can share the same temporary data storage area on a given interrupt level
- The system overhead may be reduced

The ability to cope with program reentrance is one of the most important attributes of the MPX system.

LOCAL (LOAD-ON-CALL) SUBPROGRAMS

MPX supplies a facility for loading subroutines at the time they are called for in the execution of a program. Such a subroutine is known as a LOCAL (load-on-call). All LOCALs associated with the same program use the same area of core storage by overlaying one another as they are called. A copy of each LOCAL subprogram used with a core load is kept on disk in core-image format together with that core load.

LOCALs allow the user to have, effectively, a larger program executed in core by having certain individual or groups of subroutines specified as load-on-calls. There is no theoretical limit to the number of LOCALs which the user can implement. This means a virtual extension of a core load area. Other advantages of this feature are (a) the ability to read in subroutines, (b) the breakdown of core loads to the subroutine level.

COMMON AREAS

Four unique areas of core storage are used for FORTRAN COMMON storage within MPX. These are

1. INSKEL COMMON
2. Normal COMMON
3. Interrupt COMMON
4. Core Load Area COMMON

INSKEL COMMON has already been defined (see Role of the System Executive). To assign a variable to this area, a special FORTRAN statement, COMMON/INSKEL/, must be used. All other attributes of the COMMON statement remain the same.

The normal COMMON area at the high-address end of variable core can be referenced only by mainline or batch processing core loads executable in that area. The normal COMMON statement in a mainline, special or batch processing core load is used to refer

to this area. This COMMON may be used to communicate between LINKed mainline and batch processing core loads. Communications between mainline core loads which call other mainline core loads via a CALL SPECL or a CALL EXIT must take place through INSKEL COMMON.

The third COMMON area (Interrupt COMMON) is used only for inter-program communication for programs that form an interrupt core load. The normal COMMON statement in an interrupt core load is used to refer to this area. The highest addressed location of this area must be assigned by the user when the system is assembled. This specified location is the high-address boundary of the variable core storage area that is saved when an interrupt core load is loaded for execution. Thus, it is necessary to save only the area specified by the user for interrupt core loads (not the entire variable area).

The fourth type of COMMON (Core Load Area COMMON) will reside at the end of each core load area. It will be used to communicate between programs that are linked or queued to that area.

FLEXIBILITY IN SYSTEM CONFIGURATION

A modern real-time operating system must be geared to change and diversity. The MPX system itself can exist in an almost unlimited variety of machine configurations: different installations will typically have different configurations as well as different applications. Moreover, the configuration and control strategy at a given installation may frequently change. If we look at application and configuration of an operating system, we see that the operating system must cope with an unprecedented number of environments. All of this puts a premium on system modularity and flexibility.

MPX gives the user the ability to define his configuration according to his needs: he is therefore never bound to a fixed system. Furthermore, after having specified and generated a particular system, he is still free to move process and/or data processing portions of that system from one disk cartridge to another.

In general, the input-output capability of the IBM 1800 Data Acquisition and Control System can be backed-up. For example, under program control, a 1053 printer can have its messages automatically switched to a back-up 1053 printer; disk storage drives can be logically switched or removed from the system; and any device may be removed from service if it continues to fail. This dual capacity indicates that an installation may suffer from the failure of one or more input-output devices and remain "on the air" under the most stringent usage conditions. Hand-in-hand with this back-up capability, a history of hardware device failures can be examined at any time for maintenance purposes.

MPX components can be considered under two separate group-headings: 1) control programs and 2) processing programs. In general, control programs govern the order in which the processing programs are executed, and provide services that are required in common by the processing programs during their execution. A key control program is the Executive Director which is loaded into main storage (as part of the resident System Executive) and remains there indefinitely to ensure continuous coordinated operation of the system. Other parts of the system are brought into main storage from secondary storage as they are required to perform specific functions. Processing programs consist of language translators and service programs that are provided by IBM to assist the user, as well as problem programs that are user-written and incorporated as part of the MPX system. Both IBM and user programs have the same functional relationship to the control programs.

CONTROL PROGRAMS

There are four control programs within the MPX system, as follows:

- Executive Director
- Basic Operating Monitor (BOM)
- Batch Processing Supervisor (SUP)
- Input-Output Control

The Executive Director

This forms the heart of the MPX system and controls all facets of process monitoring. It resides in core storage at all times as part of the executive in a real-time MPX system where permanent areas are storage-protected to ensure that they are not inadvertently violated or altered.

The Executive Director directs the handling of process and data processing input-output interrupts in a priority fashion determined by the user; provides timer control over the process; supervises the execution of any number of mainline core loads or programs dictated by the process; and makes the system available to the Batch Processing Monitor. Basically, it is made up of four control routines whose functions can be summarized as follows:

Program Sequence Control (PSC) -- Sequences and initiates the loading and execution of user-specified core loads.

Master Interrupt Control (MIC) -- Automatically determines the type of each interrupt as it is recognized, and transfers control to the appropriate servicing routine.

Interval Timer Control (ITC) -- Services all interrupts involving three machine interval timers, nine programmed timers, and a programmed real-time clock. Provision is also made for a real-time diagnostic timer.

Time-Sharing Control (TSC) -- Controls the time allocation of variable core between real-time and batch processing core loads such that batch processing programs may be executed when variable core is not required by the process.

Primary entry to the Executive Director is from (1) internal and external hardware interrupts, and (2) MPX calls in user's programs. The control program is read from disk only during a cold start or reload procedure.

Basic Operating Monitor (BOM)

BOM is a stand-alone disk-oriented monitor program from which a real-time or batch processing MPX system is constructed. It performs three distinct functions:

1. Supervises the generation of a disk-resident MPX operating system according to user specifications.
2. Supports a full monitor capability so that MPX can be used as a batch processing monitor system.
3. Allows the user to load absolute programs into core for execution or to store them on disk.

Since real-time process control installation requirements vary from installation to installation, it follows that each installation must be defined or tailored to the specific system function requirements and input-output configuration of that installation. The tailoring function, defined as system generation, is carried out by BOM, which provides the facilities for the creation and maintenance of a monitor system composed of IBM and user-written programs. In a real-time MPX system, BOM control ceases at cold start time when the System Executive has been loaded into core storage. In a batch processing MPX system, BOM itself functions in much the same fashion as a System Executive with permanent time-sharing.

Figure 5 illustrates BOM organization in simplified form.

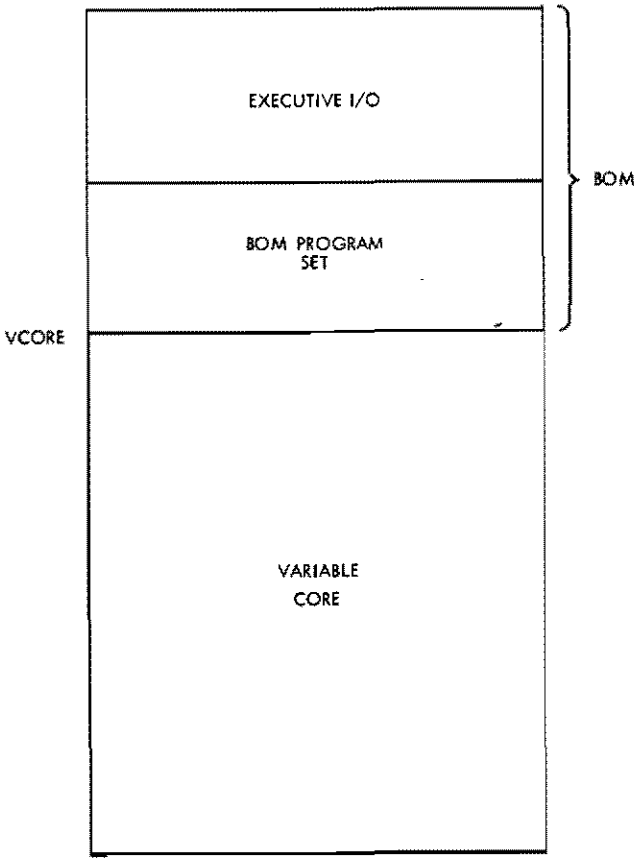

Figure 5. Basic Operating Monitor (BOM) Organization

The Executive I/O is a collection of input-output and general supporting subroutines that the MPX system requires to be in core at all times. It is that portion of a user-configurated BOM which corresponds exactly to the Executive I/O in a real-time system. The BOM Program Set is that integral part of the Basic Operating Monitor which functions in an analogous manner to the Executive Director.

Like the Executive Director, BOM can be assembled (at system generation time) with extreme flexibility so that no core is wasted by selecting only those options desired. For example, if the user elects to include the complete trace and utility package, BOM will assist him in debugging his programs before he loads them into the System Executive. Furthermore, portions of the BOM system can be deleted. The user thus selects a configuration that best matches the functions required.

Batch Processing Supervisor

The Batch Processing Supervisor (SUP) directs the execution of all batch processing core loads which may be either user-written or IBM-supplied as part of the MPX package, and provides continuous inter-job linkage within the Batch Processing Monitor. It normally operates in the time-sharing mode under the control of the Executive Director, but it may also be run as a dedicated nonreal-time monitor system under BOM.

Its main function is to recognize certain system control records and transfer control to the processing program specified. It also initializes the batch processing system when a JOB control record is identified.

Input-Output Control

The 1800 Processor-Controller communicates with I/O devices by means of direct program control or through a data channel and by the multi-level interrupt facility. Data channel or direct program control allows the user to initiate I/O operations, or makes it possible for him to check on a specific device status by analyzing the 16-bit device status word for that device, while the completion of an I/O operation causes an interrupt to occur on a particular level

(specified at system generation time) which can be recognized through programming.

In the MPX system, I/O subroutines are designed to accomplish the above by handling all of the details peculiar to each device, including the complex interrupt functions; they are also capable of controlling multiple input-output devices simultaneously and asynchronously. To provide this control, each input-output subroutine is written in three distinct parts (see Figure 6).

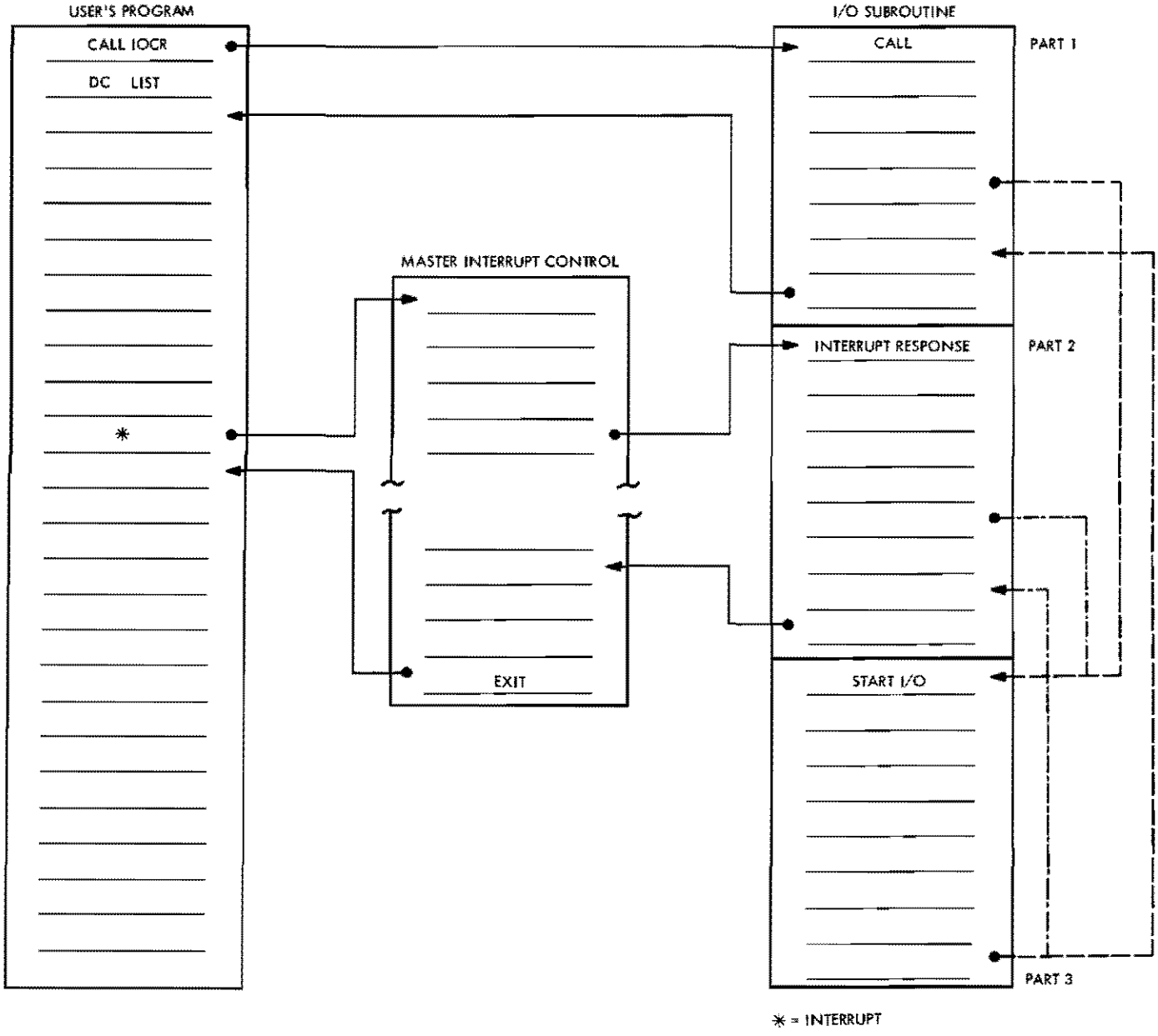

Figure 6. MPX I/O Control

Part 1. Call Routine (analyzes the I/O call for validity and queues the list for execution).
Part 2. Interrupt Response Routine (services the operation-complete interrupt).
Part 3. Start I/O Routine (initiates the function specified by a list when instructed to do so by Part 1 or Part 2).

The Call Routine is entered when a user's calling sequence is executed; the Interrupt Response Routine is entered as a result of an I/O interrupt; the Start I/O Routine is entered from either the Call Routine or the Interrupt Response Routine whenever it is time to initiate a new I/O operation.

All the requests received by I/O Control are inserted into a queue and threaded in priority sequence according to the priority of the calls that request the use of a device. Requests of the same priority will be serviced in the order received. The queue thus represents at any moment in time the complete picture of the input-output operations requested.

The user initiates an I/O operation by calling Part 1 of a subroutine written for a specific type of device; the subroutine may be resident in the System Executive or contained in a mainline core load from which the call is issued. The I/O parameters in the call's associated list are now analyzed for correctness, and an entry placed in the queue. If this is the only list in the queue, and the device is idle at the time, Part 3 of the subroutine is entered and initializes the particular I/O function. No immediate action will occur if the device should be busy at the time, while incorrect parameters cause an error indication. The user's program may then resume operation following the I/O calling sequence (see Exit from I/O Calls).

Upon completion of an I/O transfer (which may be one character/word in the case of direct program control or multiple words in the case of data channel operation) an operation-complete interrupt will cause a branch to Part 2 of the subroutine which will check for errors, initiate retry operations and manipulate data. If there are any lists in the queue at this time, Part 3 is reentered to start a new operation, such as for the next block of data to be read/written in the case of the disk routine, or for another character to be typed in the case of the 1053 printer routine. The Interrupt Response Routine will reset the device busy status and return to the user program where it had been interrupted.

Figure 6 illustrates the general flow of command used by MPX I/O subroutines. A call to IOCR enters the I/O request into the queue, and may then return to the sequence of statements following the calling sequence. Dotted lines show entry to and exit from the Start I/O Routine in the two situations discussed above. The Interrupt Response Routine (Part 2) itself is entered by a hardware interrupt. The MPX Master Interrupt Control (MIC) program is responsible for routing the interrupt through a branch table to the required entry point. In order to reset the interrupt priority and to restore the machine status, it is necessary to pass through the exit portion of MIC. Part 1 of the I/O subroutine is always executed on the same priority level as the program from which the call is given, while Part 2 is executed on the device priority level (which must be higher).

Exit from I/O Calls

MPX provides three types of exits in FORTRAN and Assembler language for all I/O devices that are initiated from a CALL statement.

Type 1—This is the normal exit associated with MPX which is used either in the variable area of core, in SPAR, or in the resident Executive. The calling section will transfer control to the I/O servicing routine and return to the first statement following the call as soon as the call's list has been entered into the I/O queue. At operation-complete time, the I/O servicing routine will return control at the point of interruption. It is the user's responsibility to determine when the I/O operation has been completed by testing when the list has been removed from the I/O queue.
Type 2—This exit is assumed to specify an operation-complete parameter which points to a subroutine. The I/O call proceeds in the manner explained under Type 1 except that at operation-complete time, control will be transferred to a subroutine referenced in the CALL statement. The user may code his operation-complete subroutine in FORTRAN or Assembler language. One example of the use of this type of exit is to free an area for the execution of other programs. If the user has placed the operation-complete subroutine outside of the executing core load area (in the System Ex-

ecutive) and the I/O buffer and I/O List in INSKEL COMMON, he may call EXIT from the mainline core load. This will free the current core load area, so that the next program in the queue may be loaded. The operation-complete subroutine could, in turn, queue another program for the area.

Type 3—The Type 3 exit will be used only in the core load areas. After the call has been initiated, the core load area will be placed in a suspended state, and the execution of a program on a lower priority level continued until the I/O operation is completed. At this time the calling core load will be reactivated and program execution continued at the first statement following the I/O call.

PROCESSING PROGRAMS

Processing programs consist of service programs and language translators broken down as follows:

Service Programs

- System Loader
- MPX Builder
- Cold Start
- IBM MPX Subroutine Library
- Disk Management Program (DMP)

Language Translators

- Assembler
- FORTRAN Compiler

Service Programs

Service Programs include a group of loaders and builders which serve as system generation aids, as well as a disk management program and a comprehensive IBM MPX Subroutine Package.

System Loader

The System Loader is the means by which the initial IBM MPX system is loaded onto the disk, an interrupt-assignment table is built from user-supplied control records, and the disk layout prepared for system generation. System assignment cards are specified by the user to define the interrupt structure; that is, to inform the loader of individual interrupt level assignments of digital, analog and data processing input-output devices, interval timers and external (process) interrupts. As each program is loaded to disk, an entry is made in a directory called the Location Equivalence Table (LET) for each component part of the MPX source system. LET thus serves as a disk map of all system programs, subroutines and relocatable programs.

The MPX Builder

This is a disk-resident composite program which performs two distinct functions:

- Core Load Build
- Executive Build

It operates under the control of the Batch Processing Monitor Supervisor.

Core Load Build. The Core Load Build function is provided for use in combining program segments that were individually compiled or assembled into a single executable core load that is ready to be loaded into main storage for execution. It also enables changes to be made in a program without recompiling or reassembling the complete program; only those sections that are modified need be recompiled.

Input to the MPX Builder is supplied by the user in the form of control records which contain the name of the relocatable mainline, restart mainline name, data files used, interrupt routines included as part of the core load and LOCAL (load-on-call) subprograms. This information will enable the builder to distinguish between real-time and batch processing programs, and to take the appropriate action for these two types of programs.

Using the information supplied by the System Loader (and the Executive Build function) as well as information from programs and subroutines, the builder establishes all subroutine linkages, hardware interrupt servicing connections and the creation of certain communications areas which are integrated with instructions to make up an independent core load. Core loads may be of several types: process mainline, interrupt, batch processing or special. Core loads are stored on bulk storage in core-image format to facilitate rapid loading whenever the core load is called for execution. The storing function is carried out by the Disk Management Program. By definition, all process core loads must be built and stored on disk prior to execution under control of an on-line MPX system.

Executive Build. To complete an on-line system, the System Executive and user programs for execution with that executive must be built. The Executive Build function operates under Batch Processing Monitor control to construct this in-core executive core load and store it on disk for MPX system operation within the limits prescribed by the user.

Input to the builder is obtained from user-assigned control records, programs, subroutines and information from the System Loader. The executive can be considered as the permanent portion of all executable core loads. It is read into core for execution by a cold start operation.

Cold Start

This program loads the System Executive into main storage, storage protects it, starts the real-time clock and calls the user's initial process core load from disk for execution. The design of the initial core load by the user will greatly determine the manner of execution of existing core loads. The cold start procedure places the executive in direct control of the real-time system.

IBM MPX Subroutine Library

This comprises a comprehensive set of reentrant subroutines as well as a selected set of non-reentrant subroutines designed to aid the user in making efficient use of the IBM 1800 Data Acquisition and Control System. The library consists of the following groups of subroutines:

1. Data Processing and Process input-output subroutines
2. Conversion subroutines
3. Arithmetic and Functional subroutines
4. FORTRAN input-output subroutines
5. Miscellaneous subroutines

Data Processing and Process I/O Subroutines. Data processing (printers, punches, etc.) and process input-output (P I/O) subroutines provide a quick and direct method for the programmer to reference the various data processing, digital and analog I/O devices for input or output of data. All I/O routines may be called directly from FORTRAN; D P I/O subroutines may be called indirectly by the use of FORTRAN I/O.

Conversion Subroutines. The design and operation of the numerous input-output devices is such that many of them impose a unique correspondence between character representations in the external medium and the associated bit configurations within the computer. Conversion subroutines convert inputs from these devices into a form on which the computer can operate and prepare computed results for output.

Arithmetic and Functional Subroutines. The arithmetic and functional group of subroutines includes a selection of twenty-seven basic routines which are most frequently required because of their general applicability. The arithmetic library contains both the routines that are visible to the FORTRAN programmer, as well as the many routines that are used by the FORTRAN compiler-generated object code, which may also be used by the Assembler programmer. A useful feature permits the testing of error indicators set by the functional routines through a FORTRAN call.

FORTRAN I/O Subroutines. FORTRAN I/O subroutines provide a link between the FORTRAN object program and the I/O devices. They support both standard and extended precision.

Miscellaneous Subroutines. The miscellaneous group provides the user with the ability to perform certain machine operations using the FORTRAN language.

These include real-time, selective dump, trace and overlay routines.

Real-time subroutines are operational control routines which service the System Executive in a real-time environment. Examples are DEFER (specify one of two hardware interval timers or one of nine programmed interval timers for some periodic activity), LEVEL (set one of twenty-four levels for programmed interrupt use), QAREA (place a core load into the mainline core load queue table according to name, execution priority, level number, and core load area), and MASK (inhibit selectively one or more levels of interrupt).

Selective dump subroutines allow the user to print chosen areas of core storage during the execution of an object program. For example, DUMP will output on the list printer, in hexadecimal or decimal form, a certain portion of core storage; DUMPS will print the status of the 1800 (that is, status indicators, contents of registers, and work areas).

The user can exercise the option of writing his own mainline trace interrupt routine which can be included in a core load to process a trace interrupt. He might, for example, design such a routine to monitor a number of conditions. The TRPRT subroutine is available for use in tracing routines which print a specified number of characters on the 1053/1816 keyboard printer or 1443 printer.

The MPX Subroutine Library also contains an overlay routine called FLIP which serves to call LOCAL (load-on-call) subprograms into core storage. All LOCALs in a given core load are executed from the same core storage locations; each LOCAL group overlays the previous group. In order to permit entry from multiple programs and interrupt levels before completing computations from a previous call, the arithmetic and functional subroutines, and most of the I/O subroutines are designed to be reentrant. That is, they can be entered from a different level of machine operation despite the fact that they may not have completed operation on a previous level. Non-reentrant versions of the arithmetic, functional, and conversion subroutines are also supplied.

Disk Management Program

The Disk Management Program (DMP) is a comprehensive group of generalized utility and maintenance routines designed to aid the user in the day-to-day operation of the MPX system. By this means, the most frequently required services of disk and data maintenance can be performed with a minimum of effort. The MPX DMP philosophy is to provide as much assistance as possible to the user. DMP is a component part of the Batch Processing Monitor. DMP is called into service by the Batch Processing Monitor Supervisor (SUP) whenever it recognizes a DMP monitor control card. It is also automatically summoned after the successful completion of an assembly or FORTRAN compilation. DMP functions can be summarized as follows.

1. Permits the user to store, modify and refer to programs and data using the compact and economical direct-access disk storage facilities of the system without regard to specific input-output configurations.
2. Allows the free interchange and use of programs and data among programmers.
3. Provides a systematic method for identifying and locating programs and data, and systematic methods to reference data after it is located.

All of these functions can be carried out while the MPX system is in real-time, as well as in the batch processing mode. Examples of DMP facilities include the following.

- Define the disk system configuration for a real-time or batch processing MPX system.
- Pack a disk file to eliminate unused areas, thus minimizing disk storage requirements.
- Remove one or more system programs from the disk-resident system.
- Update a master disk cartridge.
- Reserve a data file area on disk without actually moving any data.
- Copy one disk to another disk on-line.
- Dump an entire disk to cards on-line.
- Segment available core storage into a number of partitioned core load areas during system generation.
- Build up to four types of core loads, in conjunction with the Core Load Builder.
- Modify core loads on line.
- Change the sequence of execution of a series of

core loads.

- Print out a map of all interrupt core loads contained in the System Executive.
- Delete a program, core load or data file from the disk.
- Dump data/programs from one medium to another.

Language Translators

Language translators assist a programmer by enabling him to define a problem or an application in a language form that can be readily learned and understood. In the MPX system, the user may define his problem solution or application

- In a flexible easy-to-use symbolic language (Assembler language) and/or
- In a form of mathematical notation (FORTRAN)

Assembler

The Assembler program is a one-for-one disk oriented symbolic-type translator which assembles object programs in machine language from source programs written in symbolic language. It normally resides on disk. The Assembler accepts control records and source programs in card form only. Upon a successful assembly, the object program in relocatable format is moved to disk where it is permanently stored or, alternatively, where it can be called for execution. The Assembler language is fully described in the publication IBM 1800 Assembler Language (Form C26-5882).

FORTRAN Compiler

The FORTRAN language is a widely accepted and used language that closely resembles the language of mathematics and enables engineers and scientists to to define problem solutions in a familiar, easy-to-use notation.

The MPX FORTRAN Compiler is a disk-resident version of the 1800 Card Compiler which accepts source program statements written in the IBM 1800 FORTRAN language as input from cards, and produces, as output, an executable machine language program. At object time, the system utilizes advanced techniques, such as relocatable subroutines, highly compressed formats, and flexible input and output command structures which facilitate data conversion operations. The FORTRAN language optimizes redundant subscript calculations to produce an efficient object program.

The MPX FORTRAN Compiler permits the user to make the most of the digital and analog I/O features using a higher level language--while at the same time allowing background jobs to be executed. The FORTRAN language is described in the publication IBM 1130/1800 Basic FORTRAN IV Language (Form C26-3715).

APPENDIX A. SUMMARY OF MPX CALLING STATEMENTS

MPX calling statements can be conveniently classified, according to the type of function performed, into five groups as follows:

Sequencing or Linking Statements

CALL LINK	Specify the next program to be executed.
CALL SPECL	Suspend current program, save it in bulk storage, and execute another program.
CALL BACK	Return to a program that was only partly executed.

Queueing Statements

CALL QUEUE	Place a program into the queue table according to program name, execution priority, level number and core load area.
CALL DEQUE	Delete a program from the queue table according to program name, execution priority and level number.
CALL EXIT	Terminate the current program and execute the highest priority program named in the queue table on present level. If the queue is empty, time-share.

Timing Statements

CALL DEFER	Specify one of eleven hardware or programmed interval timers for some periodic activity.
CALL DELAY	Specify a program by level number and bit position for execution on one of eleven hardware or programmed interval timers.
CALL REPET	Cause a subroutine to be executed repeatedly, after each user-specified interval of time.
CALL CYCLE	Set-up a programmed interrupt by level and bit position repeatedly, after each user-specified interval of time.
CALL SUSPN	Suspend a program execution, but allow lower priority interrupts to occur.
CALL CANCL	Terminate a specified hardware or programmed interval timer.
CALL SETCL	Set-up the programmed real-time clock to some desired value.
CALL CLOCK	Read the programmed real-time clock in hours and thousandths of hours.
CALL TIME	Read the programmed real-time clock in hours, minutes, and seconds.

Masking/Unmasking Statements

CALL MASK	Inhibit selectively one or more levels of interrupt.
CALL UNMSK	Enable selectively one or more levels of interrupt.
CALL SAVMK	Save current masked status.
CALL RESMK	Restore masked status.

Miscellaneous Statements

CALL LEVEL	Program an interrupt on a specified level and bit.
CALL OPMON	Reset Operations Monitor.
CALL LSPAR	Change the program in SPAR (special core load area).
CALL DUMP	Print out on the list printer a selected portion of core storage.

APPENDIX B. MPX RECOMMENDED INTERRUPT LEVEL ASSIGNMENTS

Priority assignments are necessary so that an order of precedence (that is, a level) can be established among the several interrupt conditions. In configurating a multi-level interrupt system, the user should remember that certain I/O devices such as the disk, magnetic tape and timers require high response capabilities. Other I/O devices such as the list printer, typewriter and card read punch do not demand such a critical response.

In general, process interrupts (PISWs) are assigned lower priority levels than data processing and process I/O devices, except for process interrupts that do not require I/O and demand immediate response, or initiate extended operations at lower levels through the programmed interrupt feature. The reason process interrupts are assigned lower priorities than I/O devices is because user-written subroutines for the servicing of these process interrupts can then utilize all I/O devices. I/O devices must, however, receive an operations-complete interrupt which cannot occur if it is located on a lower priority level than the level from which the I/O device is called.

The amount of computer time required to service a particular interrupt can influence its priority assignment. If, for example, its servicing is relatively short, an interrupt can be accorded higher priority than one which entails more elaborate servicing procedures. Those basic I/O devices that demand fast response include the disk, magnetic tape, and timers. Because the 1053 Printer uses the disk when it buffers messages, the analog interrupts should be at a higher level than the assignment of the 1053 Printers, due to a possible loss of comparator interrupts. It should be pointed out that, although fast response is not normally required by the 1053 Printer, this device should be assigned to a high enough interrupt level to allow it to run continuously at a maximum rate. Thus, typewriter messages will be serviced without overloading of the message buffer.

It is recommended that the Analog Input Comparator feature be assigned to a higher priority level than the Analog Input. The remaining I/O devices do not possess any special characteristics for assignment at a high level, except that they should be at a level higher than the highest level from which they are called, and at a higher level than any assigned core load.

Figure 7 illustrates how a multi-level interrupt system configuration might look in the IBM 1800 Data Acquisition and Control System for a typical process control application. The example serves to convey some of the principles noted above: it should not be taken as a model. The machine configuration chosen for this example includes:

1 IBM 1802 Processor-Controller
32K words of core storage
1 IBM 2310 Disk Storage Unit with three disk drives
1 IBM 2401 Magnetic Tape Unit
4 IBM 1053 Printer Units
1 IBM 1443 Printer Unit
1 IBM 1442 Card Read Punch
1 IBM 1627 Plotter Unit
1 Analog Input Basic with Comparator
1 Analog Input Extended with Comparator
1 Digital Input
1 Digital and Analog Output
12 Interrupt Levels

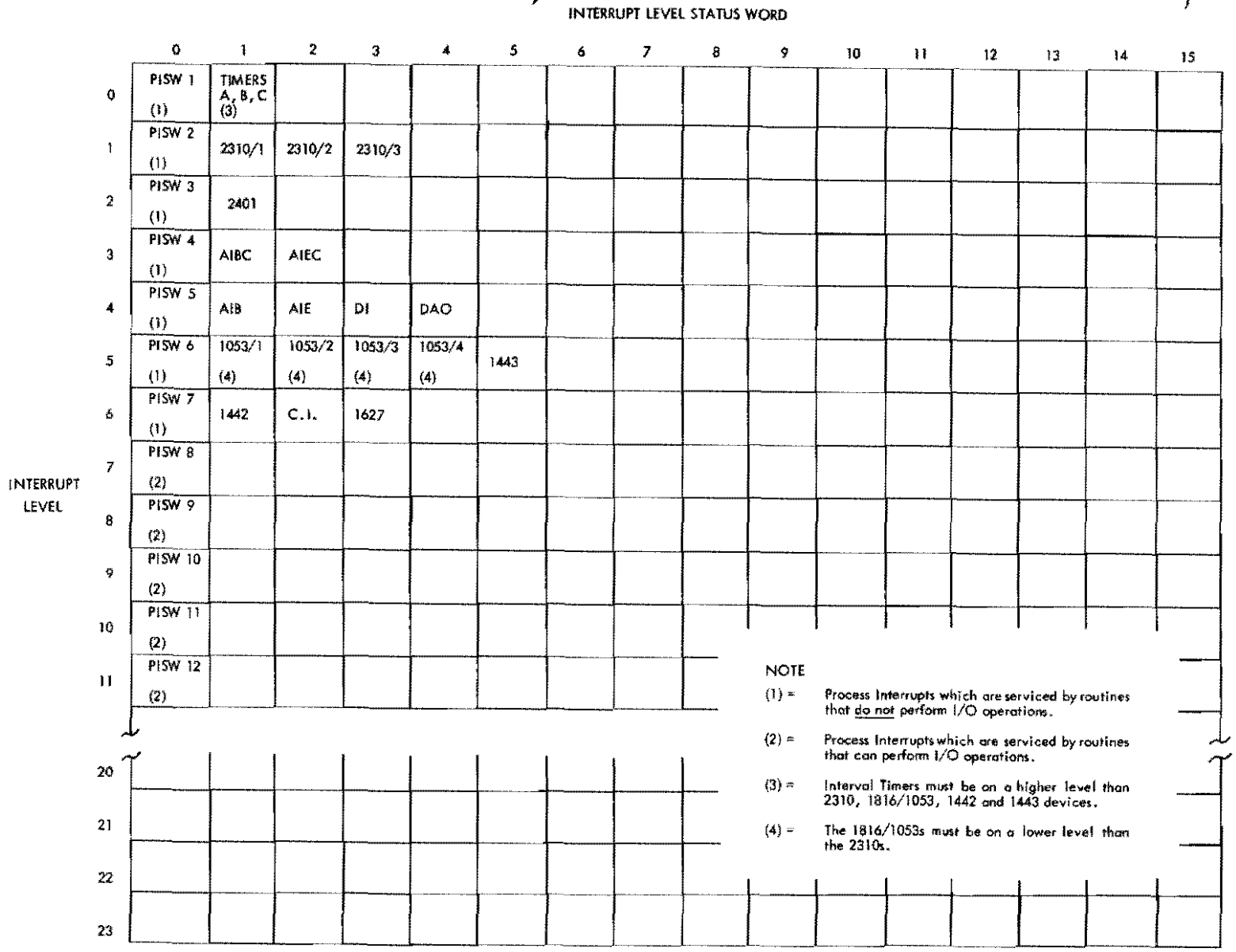

Figure 7. Example of Interrupt Level Assignment