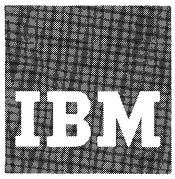
File No. S360-36
Form C27-6926-0

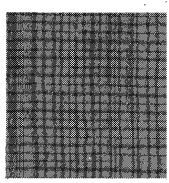
OS

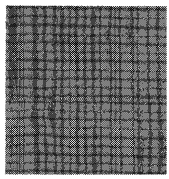


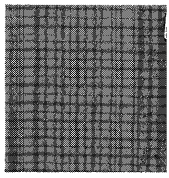
Systems Reference Library

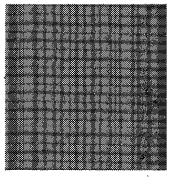
IBM System/360 Operating System Multiprogramming With a Fixed Number of Tasks (MFT) Concepts and Considerations

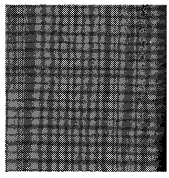
This publication describes the basic concepts of multiprogramming with a fixed number of tasks (MFT). It also includes aspects that must be considered to gain maximum effectiveness from the use of this operating system option.

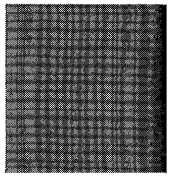


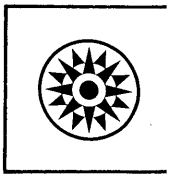


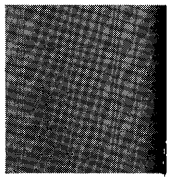












PREFACE

This publication provides the answers to three basic questions:

1. What is MFT ?
2. What does MFT offer the Operating System user?
3. What should the Operating System user do to prepare for effective use of MFT?

The "Concepts" section of this publication provides the information to answer the first two questions. The "Considerations" section includes specific information that must be taken into account during the planning stage of system installation.

The publication IBM System/360 Operating System Concepts and Facilities contains information that is considered prerequisite to an understanding of the contents of this publication.

First Edition (October 1966)

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

This publication was prepared for production using an IBM computer to update the text and to control the page and line format. Page impressions for photo-offset printing were obtained from an IBM 1403 Printer using a special print chain.

Copies of this and other IBM publications can be obtained through IBM Branch Offices.

A form for reader's comments appears at the back of this publication. Address any additional comments concerning the contents of this publication to: IBM Corporation, Programming Publications, Department 637, Neighborhood Road, Kingston, New York 12401

ILLUSTRATIONS

FIGURES

MFT provides the overall capability of multiprogramming with a fixed number of tasks. This means that as many as four jobs may be run concurrently within a single computing system having only one central processor.

MFT permits as many as four separately scheduled jobs to reside in their own predefined sections of main storage. While one job waits for the completion of an event (such as I/O operation or operator action at a graphics-device console), processing is switched to another job to take advantage of the temporary processing delay.

The jobs have a priority hierarchy, determined by the discrete area of main storage (partition) in which they reside. Figure 1 illustrates the main storage configuration in a two-partition system. The higher-address partition, P0, is always the higher-priority partition; P1, the lower-address partition, is the lower-priority partition. This means that the job in partition P0 will have preferential access to the central processor; the job in partition P1 will be given control of the central processor whenever processing in the higher-priority job is suspended (that is, when the job in the higher partition is in the wait condition). Upon completion of the event for which the job in high partition was waiting, processing is suspended in the lower-priority partition, control is returned to the higher-priority job, and processing continues in the high partition.

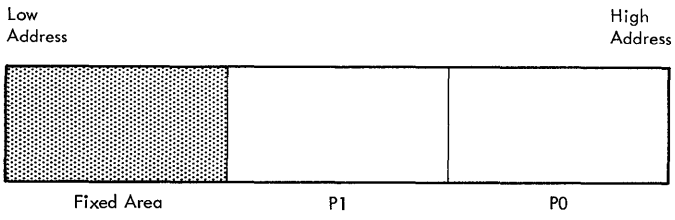


Figure 1. Main Storage Configuration, Two-Partition System

By dividing main storage into as many as four partitions (besides the fixed area), as many as four separate jobs can be executed concurrently. Although the number of partitions and the size of each is established during system generation, modifications may be made later (during nucleus initialization). Although the minimum partition size is 6K, the lowest priority partition must be large enough to contain the scheduler. For example, if the 18K scheduler were used in a system with 64K main storage, a four-partition system could be generated with a 22K nucleus, one 18K partition, two 6K partitions, and one 12K partition.

MFT is upward compatible with other operating system configurations. Further, no loss in device independence is suffered in its use. The general design of MFT is essentially identical to that of the Primary Control Program with the added capability of one or more jobs in storage at the same time.

FEATURES

MFT allows the user to specify at system generation the number and size of partitions, and the scheduler to be used. At nucleus initialization, the number of partitions may be reduced and their sizes changed without repeating the system generation process. Other features include the ability to specify the inclusion of main storage protection, and the ability to specify separately the desired number of write and reply buffers for console communication. In addition, serially reusable system resources may be enqueued upon to assure non-conflicting, sequential availability to all jobs operating at the same time, and to assure integrity of direct-access device space management.

STORAGE PROTECTION

The main storage protection feature may be specified at system generation. Use of main storage protection with MFT not only protects the system area (nucleus) from problem programs but also protects problem programs from each other, since each partition has a different storage key.

BUFFERS FOR CONSOLE COMMUNICATION

At system generation, the MFT user may specify separately the number of write and reply buffers for use with WTO and WTOR macro instructions. These buffers assure that concurrent operations may continue in another partition even though one task is waiting for operator action. Also, task processing is interrupted only minimally by console message output, even if several

operator messages are issued in close sequence.

CHARACTERISTICS

The Primary Control Program without MFT functions as a sequential scheduling system. One job at a time is brought from the input stream into main storage and that job may use all of main storage beyond a fixed area set aside for the control program. In such a system, no other job can be brought into storage until the first job is terminated.

The MFT system schedules jobs sequentially also, but main storage beyond the fixed area is separated into discrete areas called partitions. Each partition is treated as though it were the total amount of main storage available; one job is read and scheduled into each partition, and only that job uses that partition until the job is terminated. Examination of a typical sequence of operation proves helpful in understanding the characteristics of MFT.

TYPICAL SEQUENCE OF OPERATION

The normal sequence of operation in an MFT system differs from a sequential scheduling system. Figures 2 through 7 illustrate a typical sequence, including system initialization, job scheduling, and partition activation, in a four-partition system. The following narrative is keyed to those illustrations.

Loading and Initializing the System

The system is loaded by the normal initial program loading procedure (Figure 2) and initializes itself by use of the nucleus initialization program (Figure 3). In a system with MFT , nucleus initialization includes additional functions associated with the partitioned environment. These include establishing both the size of partitions and the number of partitions as defined during system generation, and insuring that enough main storage is available to satisfy these values. During initialization, the operator may use the system console to reduce the number of partitions and increase or decrease their sizes. After partition definition is complete, system initialization is identical to the sequential scheduling system.

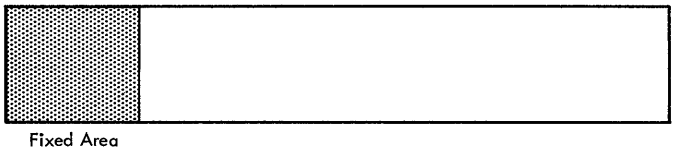


Figure 2. Contents of Main Storage After IPL

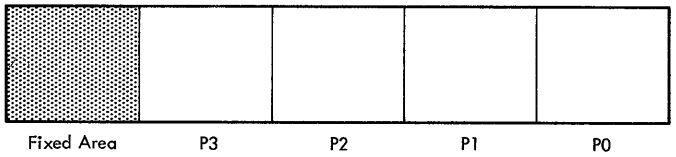


Figure 3. Contents of Main Storage After Nucleus Initialization

The First Job

After system initialization is complete the first job in the input stream is scheduled and read into the highest priority partition. This is done by the scheduler (Figure 4) which, after being brought into the high partition during initialization, causes itself to be overlayed by the first job (Figure 5). The scheduler remains logically connected to P0, however, until the first WAITR macro-instruction is encountered during execution of the first job. This WAITR need not necessarily cause an actual wait.

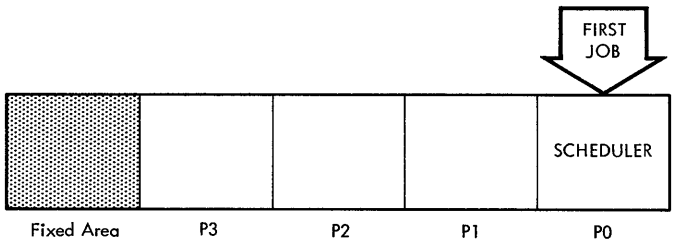


Figure 4. Contents of Main Storage After START Command

Execution of the first WAITR macro instruction in P0 causes the scheduler to be dissociated from P0, and to be logically associated with P1 (Figure 6). Partition P1 competes with P0 for the central processor. When the first actual wait occurs in the first job (such as for completion of an input/output operation), P1 is given control. At that time, the next job in the input stream is immediately

scheduled into partition P1 (Figure 7). (The WAITR macro-instruction is described in the publication IBM System/360 Operating System: Control Program Services, Form C28-6541.)

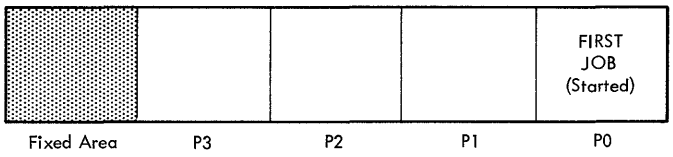


Figure 5. Contents of Main Storage After First Job is Scheduled

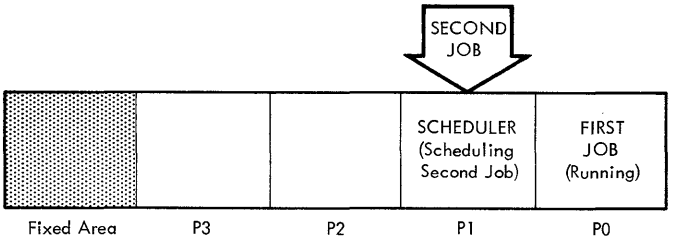


Figure 6. Contents of Main Storage After First Job Issues WAITR

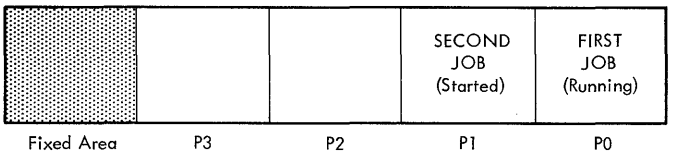


Figure 7. Contents of Main Storage After Second Job is Scheduled

Subsequent Jobs

Subsequent jobs in the input stream are scheduled into consecutively lower partitions (P2 and P3). After P3 has been activated, all succeeding jobs are scheduled into that partition unless the operator intervenes to reassign the scheduler to a higher-priority partition. Thus, while higher priority jobs continue to run, the lowest partition is used for "batch" processing. In effect, then, P3 is the main storage for a sequential scheduling system, and long-duration jobs (typically with a high ratio of input/output activity to processing activity) are run in the higher priority partitions.

The scheduler can be reassigned to a higher-priority partition by operator action. This can be accomplished when the jobs in lower priority partitions have been terminated.

Job Termination

If the higher-priority partitions contain jobs of indefinite duration, job termination normally occurs only in the lowest partition. At termination, operations proceed as in sequential scheduling systems. The completed job is terminated, the next job in the input stream is brought into the partition, and the processing of the new job then begins.

When a job in a higher-priority partition must be terminated, the operator is informed of the requirement. He may then reassign the scheduler to the partition containing the job to be terminated by entering the SHIFT command. The SHIFT command prevents the scheduling of subsequent jobs into lower priority partitions. When jobs in lower-priority partitions have been terminated, the scheduler terminates the job that requested termination, and schedules the next job in the input stream into that partition.

As stated previously, when a job is terminated, the job scheduler continues to be associated with that partition until a subsequent job scheduled into that partition issues a WAITR macro-instruction. (In the lowest-priority partition and, after the first WAITR macro-instruction has been encountered in a given job, the system treats the WAITR macro-instruction as a WAIT macro-instruction.)

Job Cancellation

The operator may cancel any job in any partition by use of the CANCEL command. If the job to be cancelled is not in the partition with which the scheduler is currently associated, the job is not terminated (following the cancellation) until the operator reassigns the scheduler to that partition.

CAPABILITIES

A typical requirement that exploits the capability for four partitions would be that where it was desired to have telecommunications, graphics, and batch processing running with concurrent peripheral operations (CPO). In such a case, the recommended arrangement of jobs is illustrated in Figure 8. The factors on which this arrangement is based are explained under "Considerations."

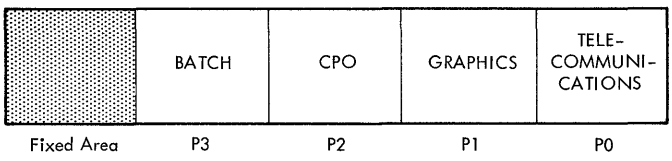


Figure 8. Relative Job Priority in a Typical Four-Partition System

A more common case would be that of concurrent telecommunication and batch processing, or batch processing with concurrent peripheral operations.

Capabilities afforded the user of Option 2 include improved machine utilization, early experience with multiprogramming, and increased installation flexibility because of the many combinations of jobs that can be run.

Information in this section, intended to aid the user in planning for the use of Option 2 by his installation, is presented in two principal categories:

Considerations Related to System Characteristics and Architecture, which include

- Evaluating job requirements with respect to system resources
- Determining the number and size of partitions needed at the installation
- Using a WAITR utility program as an alternative to coding a WAITR macro-instruction within a problem program
- Using magnetic tape units for systems input and systems output
- Evaluating existing programs to insure that they are optimized for execution in the partitioned environment
- Planning the sequence in which jobs are presented to the system

Considerations Related to Specific Types of Applications, which include

- Telecommunications
- Graphics
- Concurrent Peripheral Operation

GENERAL CONSIDERATIONS RELATED TO SYSTEM CHARACTERISTICS

The characteristics of the partitioned environment necessarily influence the choice of jobs to be run under control of the system, and to some extent, the coding techniques used within these jobs. Information below describes those effects of system characteristics that the user must consider.

EVALUATING JOB REQUIREMENTS

When a job is run under the Primary Control Program (single-task environment), it has unrestricted access to system resources not required by the PCP itself. Because only one job occupies main storage, there is no contention for system resources, and the problem programmer need not concern himself with requirements of other jobs. With the inclusion of Option 2, it becomes imperative that all jobs be evaluated in terms of their possible interference with other jobs that may run concurrently.

Input/Output Device Requirements

Jobs that require the allocation of more input/output devices than are available cannot be scheduled. In a system with Option 2 this can occur if the devices required have been allocated exclusively to jobs already scheduled.

To avoid the problem of conflicting device requirements, the user should specify device types rather than actual device addresses in the UNIT parameter of the data definition statement. Conversely, the user can insure that a device is available exclusively to a given job by specifying that the device not be shared. This could be done, for example, to assure that a particular direct access device is available without interference to a telecommunications task that uses it for maintaining input and output queues.

Main Storage Requirements

Because partition sizes are fixed after nucleus initialization, the user must not attempt to run a job whose main storage requirements exceed the size of the partition into which it is to be scheduled. The job is scheduled but is terminated abnormally before it can be started. Therefore, when planning runs, the user should always ascertain the storage requirements of those jobs that will be in the system concurrently. Apparent conflicts can be resolved by redefining partition sizes.

Use of the Console Device

Jobs that make frequent use of the console device will probably exhibit improved performance when running under Option 2 because of the improved WTO/WTOR facility it includes. Before running these jobs, the user should be sure they do not depend on the absence of overlap (rather than an explicit test) to inhibit processing after issuing a WTOR macro-instruction. Such programs will issue the WTOR and continue processing, perhaps erroneously. The buffered WTO/WTOR facility aids in

achieving greater performance, since the time formerly spent in waiting for the console device to become available can now be used for processing either by the program issuing the message, or by one of higher priority.

Use of Wait Loops

The use of loops within a problem program that prevent its continuing until some event occurs (wait loops) can cause serious degradation of overall performance. Wait loops must be executed, and thus require use of the central processor. Use of the WAIT and POST macro-instructions proves considerably more efficient, since the central processor can then be assigned to a lower priority task that is ready for processing.

A wait loop can be especially injurious to total performance if it is part of a job in one of the higher-priority partitions. In this case, none of the jobs in the lower priority partitions are enabled for processing, and the system's multiprogramming capabilities are defeated.

CHOOSING THE NUMBER AND SIZE OF PARTITIONS NEEDED

The number of partitions needed at an installation is dictated primarily by the number of jobs expected to be run concurrently. In practice, the user generally should define at system generation the maximum number of partitions for which he has main storage. If fewer partitions are needed for a particular run, the number of partitions can be reduced by the operator during nucleus initialization. Within the limits of the system, the maximum number of partitions that can be specified depends on the size of the selected scheduler in relation to the amount of main storage available. Each partition must be at least large enough to accommodate the scheduler. Thus, if a job to be run is known to exceed the size of its intended partition, a partition can be eliminated, and its storage reassigned to one or more of the other partitions at nucleus initialization.

PROGRAMS THAT READ DATA FROM THE INPUT STREAM

Programs that read data from the input stream reduce total performance unless they are executed in the lowest partition. Because such programs rely on finding data in the input stream, the WAITR macro-intruction to initiate scheduler reassignment cannot precede the closing of the input data set. Therefore, scheduling of the next job in the input stream is delayed. If the WAITR does precede closing of the input data set, the scheduler causes remaining data cards to be run out as it searches the input stream for the next JOB card. The conflict does not exist for programs run in the lowest partition, because the scheduler need not be reassigned. To avoid the delays in job initiation caused by data in the input stream (as well as the possible loss of data), the user should consider reading the data directly from a device other than the systems input device.

PROGRAMS THAT WRITE DATA IN THE OUTPUT STREAM

Programs that write data in the output stream (SYSOUT=A) should not be run concurrently. Otherwise, because access to the systems output device is not restricted, output data from different jobs is interspersed and/or overwritten. To avoid this, the user should consider writing the output data directly to a device other than the systems output device.

ALLOCATION OF SERIALLY-REUSABLE RESOURCES

Jobs that require access to the same serially-reusable resource can be run concurrently with a minimum of inter-job interference. The enqueueing/dequeueing facilities available with Option 2 permit jobs (tasks) to enqueue (ENQ) on, and dequeue (DEQ) from serially reusable resources such as system data sets. These facilities allow tasks to request exclusive or shared access to a resource. Exclusive access prevents other tasks from gaining access to the resource. Shared access permits other tasks access to the resource.

With these enqueuing and dequeuing capabilities, all tasks in the system can read from a common data set. For example, updating of catalogs and directories can be performed as a background operation in systems dedicated primarily to telecommunications or graphics. Existing jobs should be evaluated in terms of these capabilities so that the user can take full advantage of them.

USE OF A WAITR UTILITY PROGRAM

Existing programs that do not include a WAITR macro-instruction should be run only in the lowest partition of a system with

Option 2 If they are run in higher-priority partitions, scheduler reassignment is not initiated, and lower-priority partitions are not activated. To run such programs in higher-priority partitions and still utilize the system's full capabilities, the user can modify the programs to include a WAITR macro-instruction, or precede them with a utility program that issues a WAITR and transfers control to the program to be run. Either method is subject to the considerations mentioned previously under "Programs That Read Data From The Input Stream."

A WAITR utility program must perform the following

- Issue a dummy WAITR macro-instruction; i.e., one that specifies an event control block (ECB) whose completion bit is preset on
- Restore the PARM field of the EXEC statement that initiated execution of the utility program, deleting only the name of the program to be invoked. This allows the associated parameters to be passed to the program
- Dynamically invoke the desired program

CONSIDERATIONS RELATED TO APPLICATION TYPES

The requirements of applications such as telecommunications, graphics, and concurrent peripheral operations (CPO), must be considered by the user in relation to the partitioned environment in which they will reside. Information that follows describes requirements that are of primary importance to the user.

TELECOMMUNICATIONS USING THE BASIC TELECOMMUNICATIONS ACCESS METHOD

To avoid delays in servicing lines, a telecommunications task should have unrestricted access to the resources of the central processor. For this reason telecommunications jobs should be run in the highest-priority partition. Because the telecommunications job is not alone in the system, its activities should cause minimum interference with jobs in other partitions, and it should not be susceptible to interference from these other jobs. In particular, a telecommunications job that is to be run in a system with Option 2 should issue a WAITR macro-instruction during program initialization so that scheduling of the next job is not delayed.

Minimizing Interference to Other Jobs

A major source of interference to other jobs in the system is the use of a system console device by the telecommunications job for line status information. When this technique is used, the wait time of the telecommunications job is used for writing the line status information, and processing time available to lower priority jobs is severely limited. This limitation is avoided by using a "local terminal" for operator communication with the telecommunications job, and reserving the console device for communication with other jobs that do not have access to such facilities.

GRAPHICS

In general, graphics jobs are subject to the same considerations as telecommunications jobs. They should be run in a high-priority partition, and they should cause minimum interference with other jobs. When graphic jobs are run in the same system as a telecommunications job, the telecommunications job should be run in the highest-priority partition, and the graphics job(s) in the next-highest priority partition(s). A graphics job associated with an unbuffered IBM 2250 Display Unit may operate with reduced performance if high telecommunications activity interferes with its access to the central processor for regenerating the display. In this case the user must determine the relative importance of the graphics and telecommunications jobs, and decide which to run in the highest-priority partition.

CONCURRENT PERIPHERAL OPERATION

Concurrent peripheral operation (CPO) is the capability of performing utility functions such as card-to-tape, tape-to-print, or tape-to-punch while other jobs in the system continue processing. Another name for this capability is simultaneous peripheral operation on-line (SPOOL). A CPO job should usually be run in a higher-priority partition. In a system that includes both telecommunications and graphics, the CPO job should usually be run in the third-highest partition, with the graphic job in the second-highest, and the telecommunications job in the highest. The CPO job should issue a WAITR, so that the initial sequential processing ("batch") job can be scheduled. Because the CPO job would appear (to the system) as an unending job, the user should insure that all required devices are requested when the job is initially scheduled.

Once the batch job is started in the lowest partition, the scheduler remains

assigned to that partition. For this reason, the CPO job should contain a control program that appears to the system as an unending job. Such a control program could communicate with the operator by means of the WTOR macro-instruction to determine which utility function is to be executed, and then -- by use of the LINK macro-instruction or a LOAD and LINK sequence -- execute the function. The utility program might then use the RETURN macro-instruction to return to the CPO control program, which would remain resident.

C27-6926-0

Printed in U.S.A. C27-6926-0

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601
[USA Only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]

READER'S COMMENTS

IBM SYSTEM/360 OPERATING SYSTEM: MULTIPROGRAMMING WITH
A FIXED NUMBER OF TASKS (MFT); CONCEPTS AND CONSIDERATIONS

C27-6926-0

Your comments will help us produce better publications for your use. Please check or fill in the items below, adding explanations and other comments in the space provided.

All comments and suggestions become the property of IBM.

Which of the following terms best describes your job?

☐ Programmer	☐ Systems Analyst	☐ Customer Engineer
☐ Manager	☐ Engineer	☐ Systems Engineer
☐ Operator	☐ Mathematician	☐ Sales Representative
☐ Instructor	☐ Student/Trainee	☐ Other (explain) _____

Does your installation subscribe to the SRL Revision Service? ☐ Yes ☐ No

How did you use this publication?

☐ As an introduction
☐ As a reference manual
☐ As a text (student)
☐ As a text (instructor)
☐ For another purpose (explain) _____

Did you find the material easy to read and understand? ☐ Yes ☐ No (explain below)

Did you find the material organized for convenient use? ☐ Yes ☐ No (explain below)

Specific criticisms (explain below)

Clarifications on pages _____
Additions on pages _____
Deletions on pages _____
Errors on pages _____

Explanations and other comments:

No postage necessary if mailed in U.S.A.

FOLD FOLD

FIRST CLASS
PERMIT NO. 116
KINGSTON, N. Y.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN U. S. A.

POSTAGE WILL BE PAID BY

IBM CORPORATION
NEIGHBORHOOD ROAD
KINGSTON, N. Y. 12401

ATTN: PROGRAMMING PUBLICATIONS
DEPARTMENT 637

FOLD FOLD

Printed in U.S.A. C27-6926-0

IBM

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N.Y. 10601
[USA Only]

IBM World Trade Corporation
821 United Nations Plaza, New York, New York 10017
[International]