File No. 1410-30
Form C28-0251-0

IBM Systems Reference Library

# IBM 1410 Input/Output Control System for 1301 Disk Storage

This publication contains the information necessary to understand and use the IBM 1410 Input/Output Control System for 1301 Disk Storage. Described in detail are the types of processing peculiar to disk storage (Single Reference, Random and Sequential), as well as the macro-instructions, DIOCS entries, DTF entries, and DA entries required for efficient use of 1301 Disk Storage

MAJOR REVISION (March 1963)

This publication supersedes the bulletin, IBM 1410
Input/Output Control System for 1301 Disk Storage,
Form J28-0251, and the associated Technical Newsletter,
Form N28-1023.

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

Address comments concerning the content of this publication to:

IBM Corporation, Programming Systems Publications, Dept. 637, Neighborhood Road, Kingston, N.Y. 12401.

CONTENTS

The purpose of this publication is to enable installations using IBM 1301 Disk Storage to avail themselves of the many advantages offered by the IBM 1410 Input/Output Control System for 1301 Disk Storage (hereafter referred to as the "1301 IOCS"). The bulletin explains the functions and use of the 1301 IOCS macro-instructions and describes the DIOCS, DTF, and DA entries needed to incorporate the 1301 IOCS into users' object programs. The publication augments the bulletin describing the 1410 IOCS for card and tape systems.

## Prerequisites

It is assumed that the reader of this publication has completed at least a basic course in programming the IBM 1410 and is familiar with the following IBM publications:

Reference Manual, "IBM 1410 Data Processing System," Form A24-1407
Reference Manual, "IBM 1410 Data Processing System, 1301 Disk Storage," Form A22-6670
IBM 1410 Data Processing System Bulletin, "Autocoder: Preliminary Specifications," Form J24-1433
IBM 1410 Data Processing System Bulletin, "IBM 1410 Input/Output Control System for Card and Tape Systems, Preliminary Specifications," Form J28-1432

## Machine Requirements

The 1301 IOCS can be incorporated into programs written for any IBM 1410 system that meets the following minimum configuration:

20,000 positions of core storage
1 IBM 1301 Disk Storage Unit, Model 1 (or Model 2)
Processing Overlap and Priority special features.

Programs incorporating the 1301 IOCS can be generated by the 1410 Autocoder processor which requires the following minimum machine configuration:

20,000 positions of core storage
4 IBM 729 II, 729 IV, or 7330 Magnetic Tape Units
1 IBM 1402 Card Read-Punch, Model 2, and
1 IBM 1403 Printer, Model 2.

## BASIC DISK PROCESSING CONSIDERATIONS

### Use of Disk Storage

The principal characteristics of disk storage are large storage capacity and speedy access to records regardless of their location in disk storage. A comparison of the basic characteristics of magnetic tape and 1301 disk storage clearly indicates the superiority of disk storage with respect to storage capacity and access time. See Figure 1.

| | 729 IV Magnetic Tape | 1301 Disk Storage |
|---|---|---|
| Maximum Storage Capacity | 8,000,000 characters per reel | 28,000,000 characters per module |
| Maximum Access Time | more than 2 minutes | 214 milliseconds |

Figure 1. Storage Capacity and Access Time of Magnetic Tape and 1301 Disk Storage

### File Maintenance Applications

It is evident from the above that disk storage is the ideal storage medium for applications that call for the speedy retrieval of single, non-sequential items from a large body of stored information. One such widely used application of disk storage is that of maintaining inventory files. In this type of application, records are obtained and adjusted as inventory changes occur, so that the file stored in disk storage always reflects the up-to-the-minute status of stocks on hand. This important feature of disk processing -- the ability to keep large files of information up to date by recording transactions as they occur -- is common to most disk-processing applications. Today, many different types of large business establishments, including brokerage firms and insurance companies, use IBM disk storage devices to furnish their executives with up-to-the-minute reports of all records and accounts.

### Types of Disk Processing

Depending on the arrangement of the data in disk storage and the nature of the application, users will find it advantageous to choose one of three methods of moving information to and from disk storage. The three methods are known as "Single-Reference Processing," "Random Processing," and "Sequential Processing," respectively. Each of these methods can be handled by the 1301 IOCS.

SINGLE-REFERENCE PROCESSING. Single-Reference Processing is the most flexible -- but also the most uneconomical -- method of moving data to and from disk storage. It enables the programmer to move data of any record format to and from any area in disk storage. The primary use of single-reference processing is in real-time applications where the timing, the precise nature and the location of the information to be moved to or from disk storage are not known at the time the program is written.

The 1301 IOCS provides the input/output and error routines needed for single-reference processing. See the description of the GETS, PUTS, and WAITS macro-instructions.

RANDOM PROCESSING. Random Processing is the most widely used method of reading and writing disk data. It is used to read and write disk data of uniform, predefined format stored within a predefined area of disk storage. The method is used whenever an application calls for data that belongs to a particular file of information and must be processed in an arbitrary, or "random," order.

Inventory maintenance and report writing are typical applications using this method of processing. When using this method, the programmer can use all of the 1301 IOCS macro-instructions, but he must define the file from or to which data is to be moved.

SEQUENTIAL PROCESSING. While the principal advantage of disk storage is its facility to process non-sequential information, there are applications for which disk information is best handled sequentially. Such applications include the loading and unloading of disk storage. The latter is of special significance when programs are stored in disk storage.

When information is transferred sequentially there is virtually no seek time -- provided files are properly arranged on disk cylinders. Moreover, many of the processing techniques that reduce machine time in tape processing also apply to sequential processing of disk data. Thus, at least two input/output areas can be used for the reading and writing of disk data, so that disk input/output operations can be overlapped with processing. All of the above factors make sequential processing the fastest method of moving information to and from disk storage. However, the method cannot be used for the many applications requiring arbitrary, or "random," retrieval of disk data.

The 1301 IOCS furnishes all the routines needed to handle sequential processing in the most efficient manner. This includes routines for blocking and deblocking of record blocks and for overlapping of disk input/output operations with processing.

This section describes how random-processing applications are handled by the 1301 IOCS.

## In-Line Processing

The most elementary approach to disk processing is known as "in-line processing." In this type of application, information is obtained from disk storage by SEEK and READ commands that are part of the main routine. Processing halts, therefore, each time a disk SEEK, READ or WRITE operation takes place, and processing does not resume until the desired disk information has been obtained.

Figure 2 illustrates this method of disk processing, which calls for:

(1) reading of a transaction record;
(2) retrieval of the corresponding information from disk storage;
(3) updating of this information, and
(4) return of the updated information to disk storage.

NOTE: A good example of this type of application is inventory control. This calls for the updating of part records (which are kept in disk storage) on the basis of receipts and disbursements of parts (i.e., "transactions"). Updating takes place as soon as transactions have occurred, and in this manner the inventory file is kept currently up to date.

## Information Retrieval Only

A special case of in-line processing calls only for the retrieval of information. Here information is obtained from disk storage, but it is not updated and it is not written back onto the disk. See Figure 3.

## Summary of In-Line Processing

In-line processing represents the simplest but also the most wasteful use of disk storage. It is wasteful because the Central Processing Unit stands idle during the disk SEEK, READ and WRITE operations. This halt in processing occurs because the instructions calling for these disk operations are part of the main routine.

## Overlapping of SEEK Operations

The most time-consuming disk operation is the SEEK, which requires from 0 to 180 milliseconds

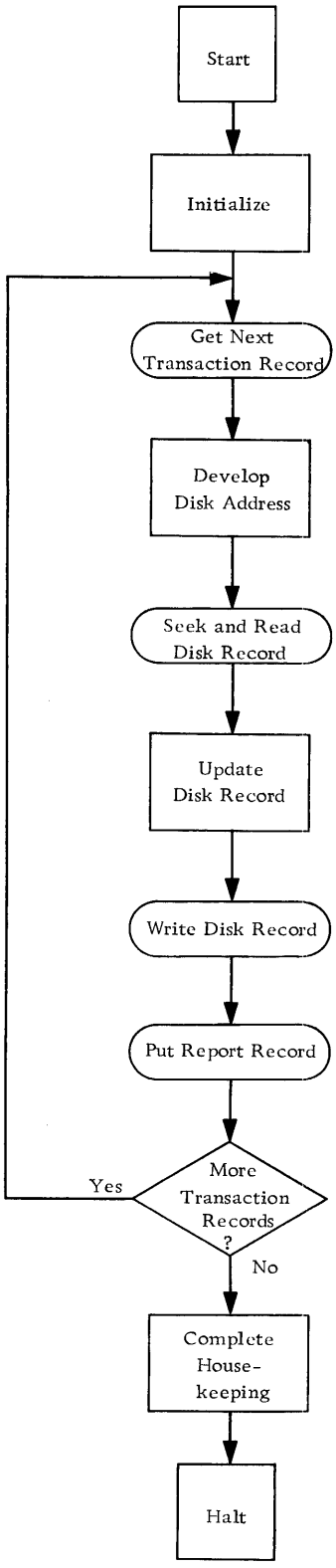



Figure 2. Random Processing - Type 1: In-Line Processing

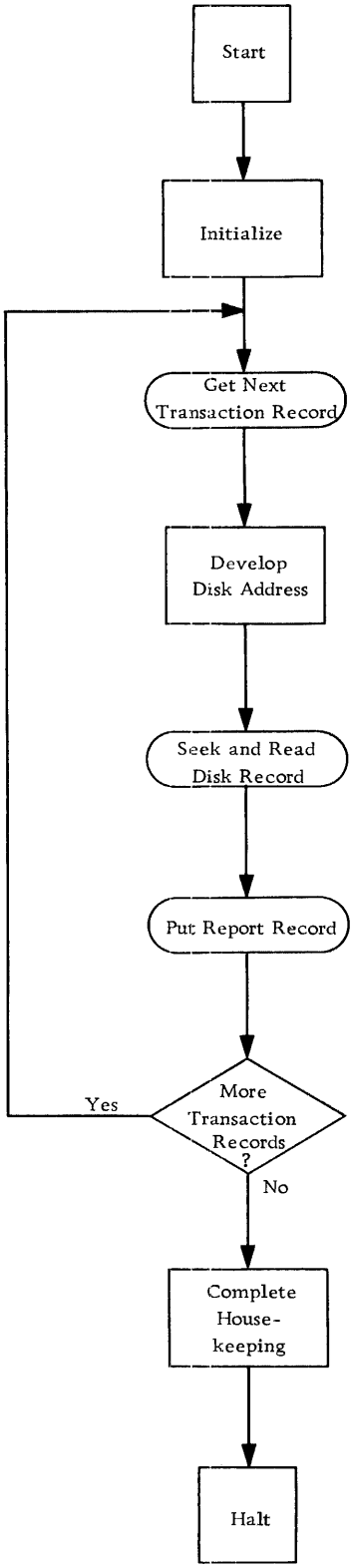



Figure 3. Random Processing - Type 1: Information Retrieval Only

(i.e., 180,000 microseconds).* The coding required to handle the simultaneous execution of several SEEK operations constitutes a considerable programming task. The main tasks that must be accomplished are:

(1) the retention, or "stacking," of successive transaction records until they can be processed;
(2) the holding of disk records obtained by the various SEEK and READ commands until these records can be processed;
(3) provisions insuring that disk records are updated with the correct set of transaction data, and
(4) the release, after processing, of areas used to retain transaction records and disk records.

The 1301 IOCS provides all the functions listed above, as will be explained in the next section.

## SEPARATION OF THE DISK ROUTINE FROM THE MAIN ROUTINE

The operating principles of the 1301 IOCS for random-processing applications are illustrated in Figure 4. Note that all instructions needed to obtain or process disk data (henceforth referred to collectively as the "Disk Routine") have been removed from the main routine.

Although the main routine initiates processing of the Disk Routine, the two routines are independent of one another: the main routine obtains and stores transaction records independently of any processing in the Disk Routine, and the Disk Routine obtains successive transaction records from the work area independently of processing in the main routine.

NOTE: Separation of the Disk Routine from the main routine is possible only if processing in the main routine does not depend on the information obtained by the Disk Routine. The most widely used applications of disk storage, such as inventory control and information retrieval, permit this approach.

The different program steps, as executed by the 1301 IOCS, are indicated in Table I.

*Since a module of IBM 1301 Disk Storage has only one access arm, multiple SEEKs can be used and overlapped only if the program uses more than one module of 1301 Disk Storage.

| | Main Routine | Disk Routine |
|---|---|---|
| | The main routine obtains a transaction record, stores it in Work Area I, and branches control to the Disk Routine. | |
| | | Processing continues in the Disk Routine until the SEEK and READ Disk Record command is encountered. The Disk Routine initiates the SEEK and then checks both work areas. |
| TEST OF WORK AREAS See also Figure 5. | If no disk record is ready for processing and a segment of Work Area I is available, control will be branched to the main routine: | If a previously read disk record is ready for updating, processing continues in the Disk Routine (Point B): |
| | Processing now continues at Point A (return address of the main routine). In Figure 4, this is the branch to the instruction that calls for the reading of the next transaction record. This transaction record is now moved to a free segment of Work Area I, and control is branched to the Disk Routine. (In this manner, the main routine initiates the reading and processing of each disk record required by a given transaction record.) Processing then continues as described above. | The waiting disk record is updated with the correct transaction data, and the WRITE operation that will write the updated disk record back into disk storage is initiated. (As indicated in Figure 4, the segment of Work Area II that contained the just-updated disk record will be released upon completion of the WRITE operation.) Processing in the Disk Routine now continues: any needed report is written, and the segment of the transaction stacking area that held the transaction record used to update the disk record is released. The test of the work areas indicated above is then made again, and control is branched to either (A) or (B), depending on the outcome of the test, as described above. |

Table I. Program Steps Executed by the 1301 IOCS

## Summary of Main IOCS Functions for Random Processing

As indicated in Figure 4 and the subsequent description, the IBM 1301 IOCS accomplishes each one of the tasks that were characterized above as necessary prerequisites for the simultaneous execution of several SEEK operations. The following summarizes these functions of the 1301 IOCS:

1. The retention, or "stacking," of successive transaction records until they can be processed.

   As indicated in Figure 4, successive transaction records are stored in a work area, which will henceforth be called the "Transaction Stacking Area." Whether a new transaction record is to be stored or a disk record is to be updated is determined immediately after the initiation of the SEEK operation and after processing of the transaction is completed.

   The updating of a disk record ready for processing takes precedence over the reading of another transaction record. Only if no disk record is ready for processing and a free segment of the Transaction Stacking Area is available, is control passed back to the main routine for the reading (and storing) of another transaction record. (If two transaction records request the same disk record, the 1301 IOCS does not process the second request until processing of the first has been completed.) See Figure 5.

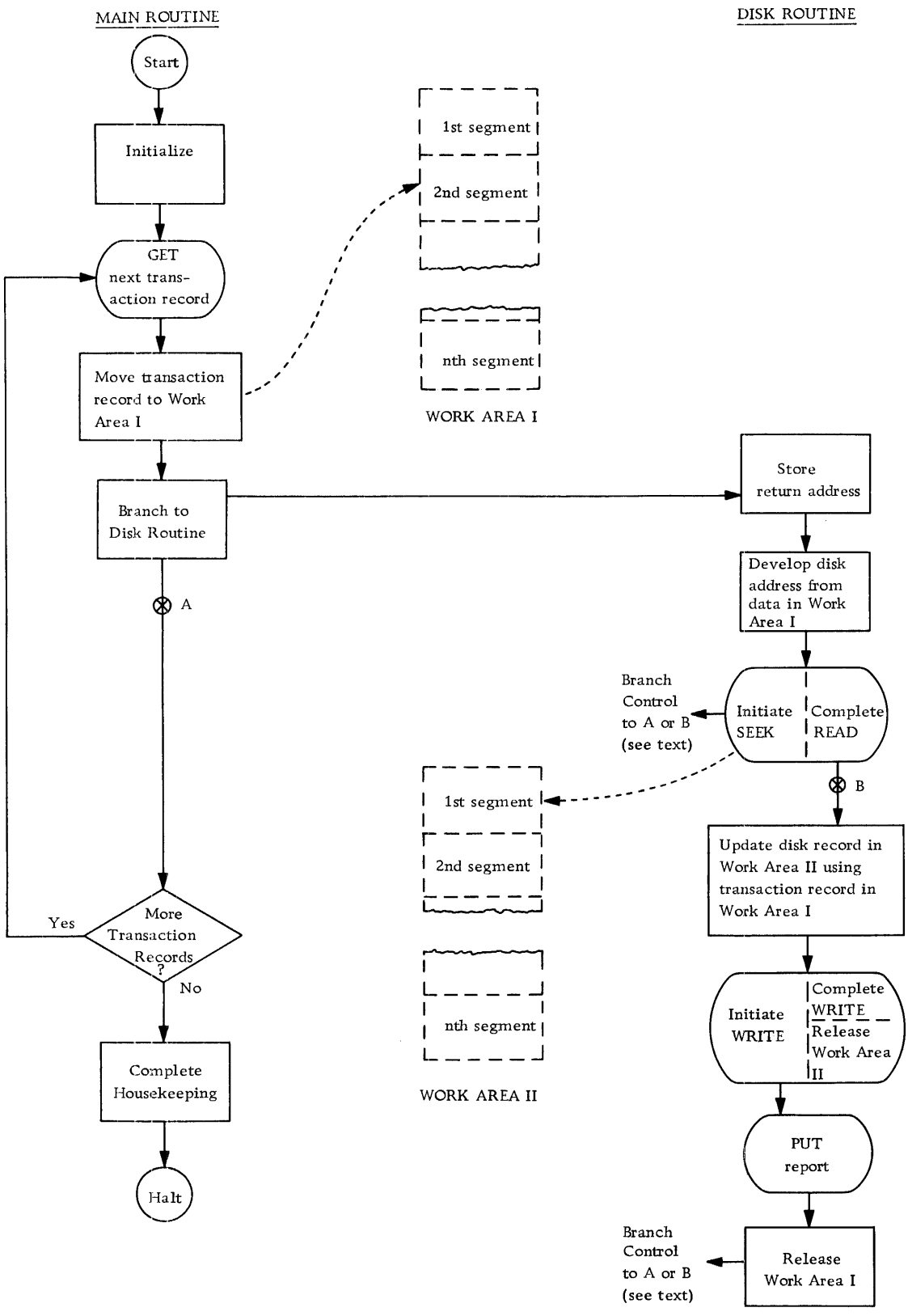



Figure 4. Separation of the Disk Routine from the Main Routine

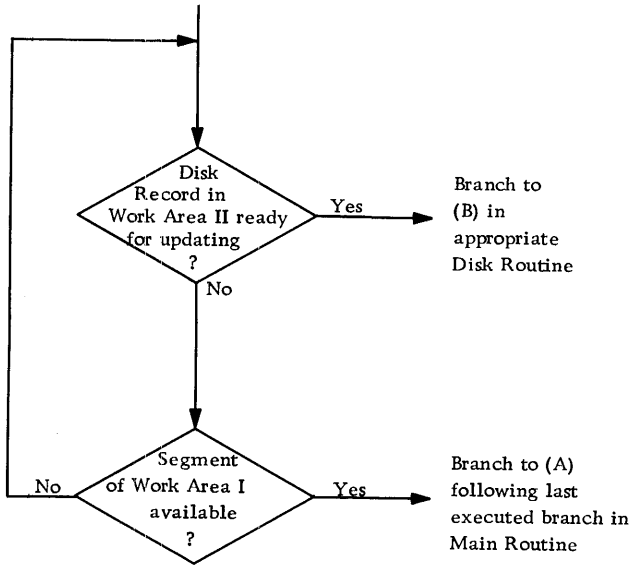



Figure 5. Control Switching

2. The holding of disk records obtained by the various SEEK and READ commands until these records can be processed.

   As indicated in Figure 4, records obtained from disk storage are held in another work area, which will henceforth be referred to as the "Disk Record Holding Area." The 1301 IOCS holds disk records in this area until they are no longer required. See the section on "Additional Functions of the FSEQP Macro-Instruction" for a detailed description of the release procedure of disk records.
3. Provisions insuring that disk records are updated with the correct set of transaction data.

   As indicated in Figure 4, the removal of the Disk Routine from the main routine permits the simultaneous execution of several SEEK and READ commands. As described, each time a disk record has been successfully sought and read, it is made ready for updating. The 1301 IOCS insures that whenever control is branched to the Disk Routine for the updating of a disk record, this disk record is updated with data from the correct transaction record stored in the Transaction Stacking Area.
4. Release of work areas after their contents are no longer required. Segments of the Transaction Stacking Area are released upon completion of processing in the Disk Routine. Segments of the Disk Record Holding Area are released at the completion of each WRITE operation, and as explained under "Additional Functions of the FSEQP Macro-Instruction."

Sequence of Disk Operations

Although transaction records are released to the Disk Routine in the order in which they were obtained by the main routine, updated disk records are not necessarily written back onto the disk in the same order. This is due to the different access time for information in disk storage. Information access time depends not solely on the order in which disk requests are given but also on the location of the requested information in disk storage.

For example, if one arm receives a request for information with an access time of 100 ms, and 10 ms later another arm receives a request with access time of 50 ms, the second request will be met before the first. In this case, the disk information requested last will be obtained before that requested just prior to it.

If disk records are to be processed in the same order as the incoming transaction records, processing of the data obtained by the second arm in the example must be delayed until the data obtained by the first arm has been processed.

Disk IOCS Macro-Instructions

The 1301 Disk IOCS permits the use of the following macro-instructions:

OPEN "Open Disk File(s)"
This macro-instruction may be used to open disk files used for random or sequential processing.

CLOSE "Close Disk File(s)"
This macro-instruction may be used to close disk files used for random or sequential processing.

MVRSA "Move Record to Stacking Area"
This macro-instruction may be used to move transaction records to the Transaction Stacking Area. It can be used only for random processing.

ENTDR "Enter Disk Routine"
This macro-instruction may be used to enter the Disk Routine and store the return address to the main routine. It can be used only for random processing.

GET "Get Disk Record"
This macro-instruction may be used to seek and read disk records. It can be used only for random and sequential processing.

FSEQP "Force Sequential Processing"
This macro-instruction can be used to insure that records are processed sequentially, i.e., written

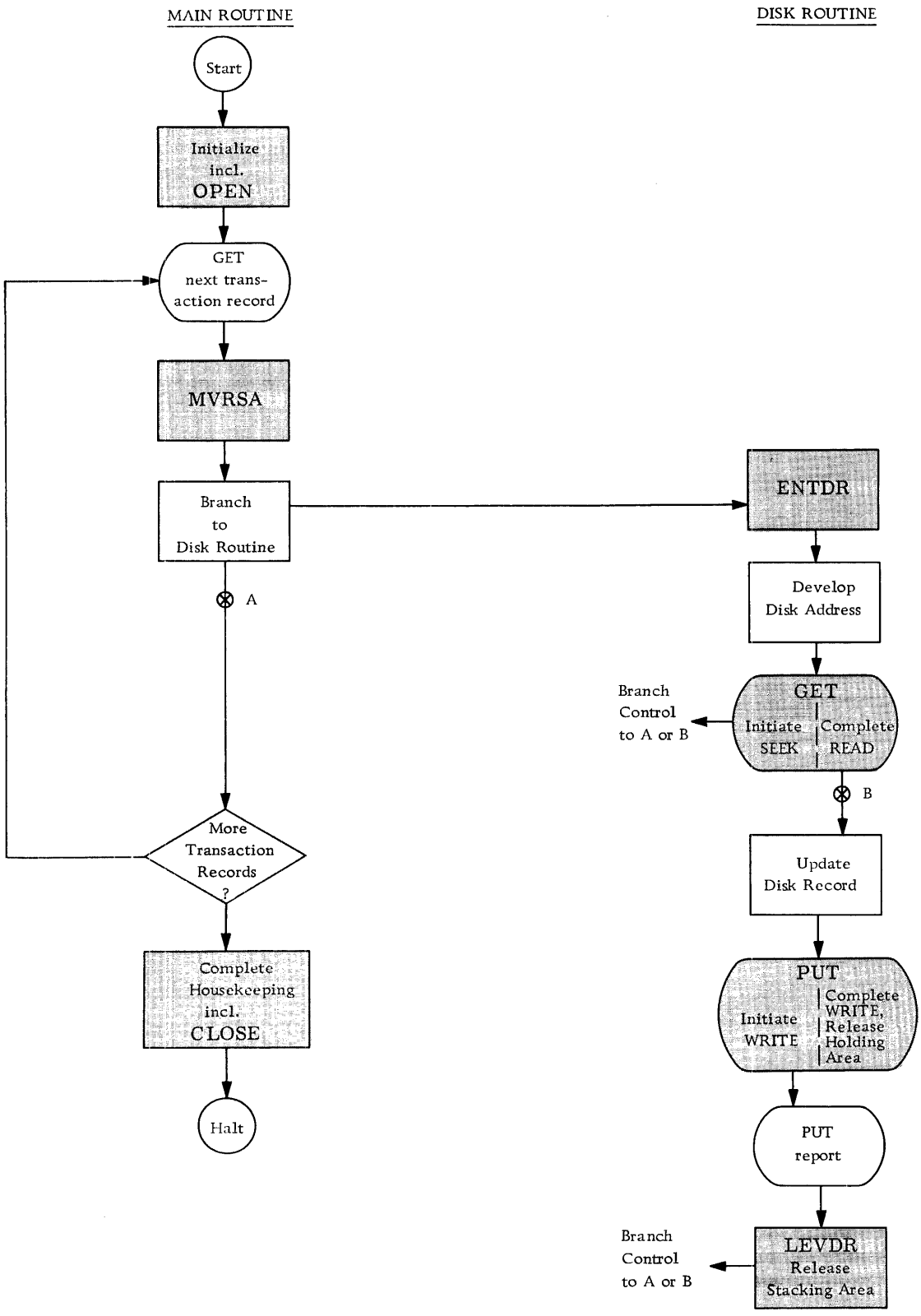



Figure 6. The 1301 IOCS Macro-Instructions for Random Processing

back onto disk storage in the same order in which their corresponding transaction records were obtained by the main routine. This macro-instruction can be used for random processing only.

PUT "Put Disk Record"
This macro-instruction may be used to write a single disk record. It can be used only for random and sequential processing.

LEVDR "Leave Disk Routine"
This macro-instruction may be used to release the segment of the Transaction Stacking Area just processed and to return control to the main routine. It can be used only for random processing.

GETS "Get Single-Reference"
This macro-instruction may be used to seek a specified disk cylinder or to seek and read a single disk record, a full track, or a full cylinder (optional feature).

PUTS "Put Single-Reference"
This macro-instruction may be used to seek a specified disk cylinder, to write a format track, or to seek and write a single disk record, a full track or a full cylinder (optional feature).

WAITS "Wait"
This macro-instruction may be used to develop the coding required to suspend processing until a specified disk record has been read into core storage or written onto disk storage.

Summary of Macro-Instructions

The use of the 1301 IOCS macro-instructions for random processing is summarized in Figure 6. A detailed description of each macro-instruction may be found in the section on "The Eleven 1301 IOCS Macro-Instructions."

## INDEPENDENT DISK ROUTINES

The applications discussed above use only one Disk Routine, and all information obtained by the disk arm(s) from the Transaction Stacking Area was always used by the same set of instructions.

Some applications require two (or more) independent Disk Routines. A typical application of this type is the updating of a job record and an employee record on the basis of one transaction record. Both routines use the same Transaction Stacking Area. See Figure 7.

As indicated in Figure 7, the same set of transaction data is required by more than one Disk Routine. The 1301 IOCS insures that no segment of the Transaction Stacking Area is released until all Disk Routines requiring data from this segment have been completed. The IBM 1301 IOCS can handle any number of independent Disk Routines.

## DEPENDENT DISK ROUTINES

Some applications use data from one transaction record to update two dependent disk records. Two records are considered dependent on one another if neither of them can be updated without data from the other. When dependent records are processed, both disk records must be obtained before either of them can be updated. This requires that disk records be retained in the Disk Record Holding Area until all Disk Routines requiring data from the same Holding-Area segment have been completed. The 1301 IOCS insures this.

The first FSEQP macro-instruction in Figure 8 is needed to retain the disk record obtained by Disk Routine A for use by Disk Routine B. The second FSEQP macro-instruction is used to synchronize the output of Disk Routine B with the incoming transaction records.

(For a detailed description of the functions of the FSEQP macro-instruction, see the section on "Additional Functions of the FSEQP Macro-Instruction.")

NOTE 1: The order of the PUTs takes place as indicated. In general, arms that have obtained information from disk storage remain in position until the updated information is returned to storage. This eliminates SEEK time for PUTs since the disk arms are already in position.

NOTE 2: The 1301 IOCS can handle any number of dependent Disk Routines.

It is possible to contain a random Disk Routine within a real-time routine. The following three points must be observed:

1) the MVRSA macro-instruction must be contained within the real-time routine, 2) care must be taken that real-time interrupts do not cause the Disk Routine to be executed before the disk file has been opened and after it has been closed and 3) when the Disk Routine requests exceed the number specified in the 'RNDMDEPTH' DIOCS entry, the real-time routine will control until more random disk requests may safely be accepted.

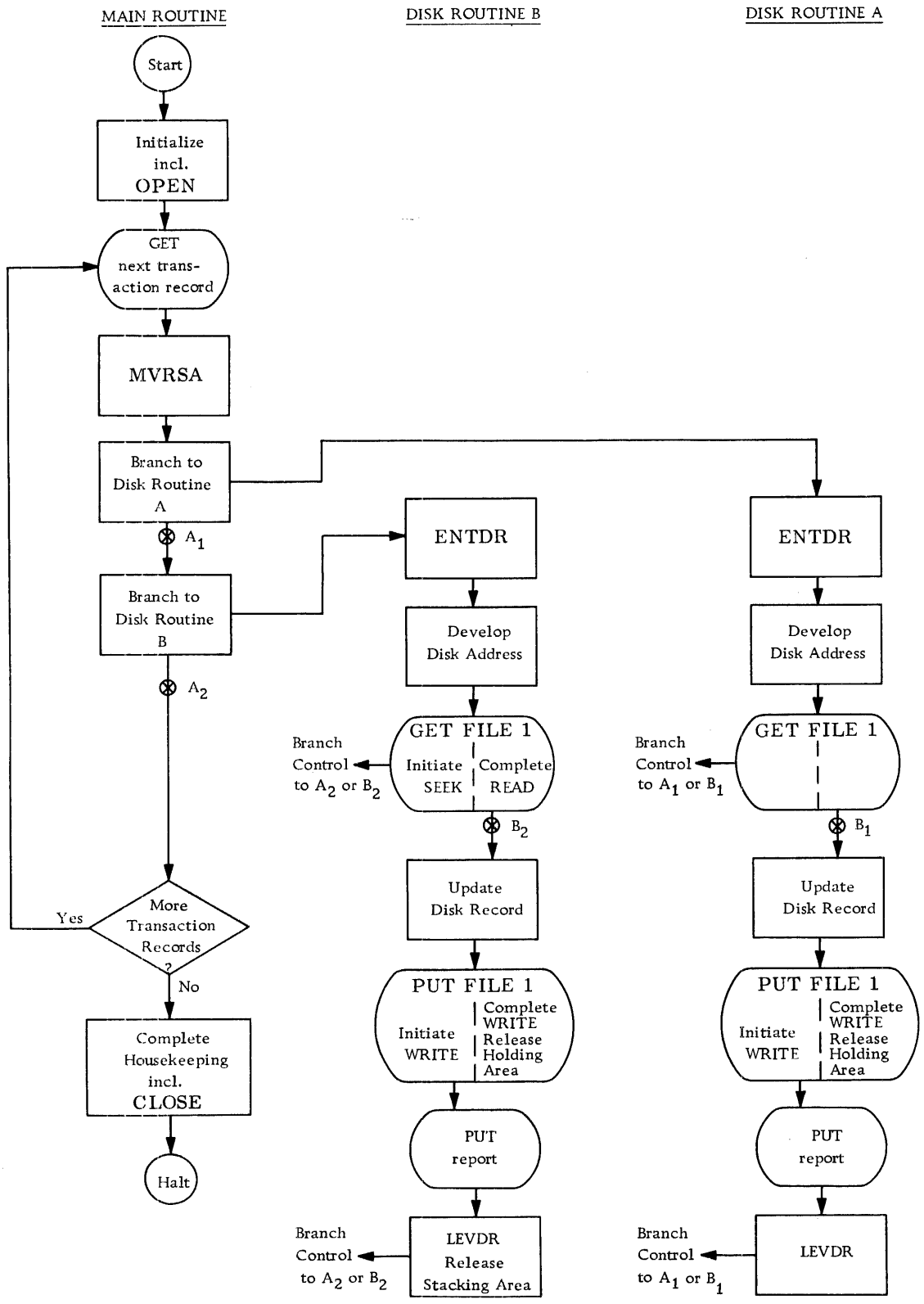



Figure 7. Random-Processing -- Two Independent Disk Routines Use Data Obtained by the Main Routine

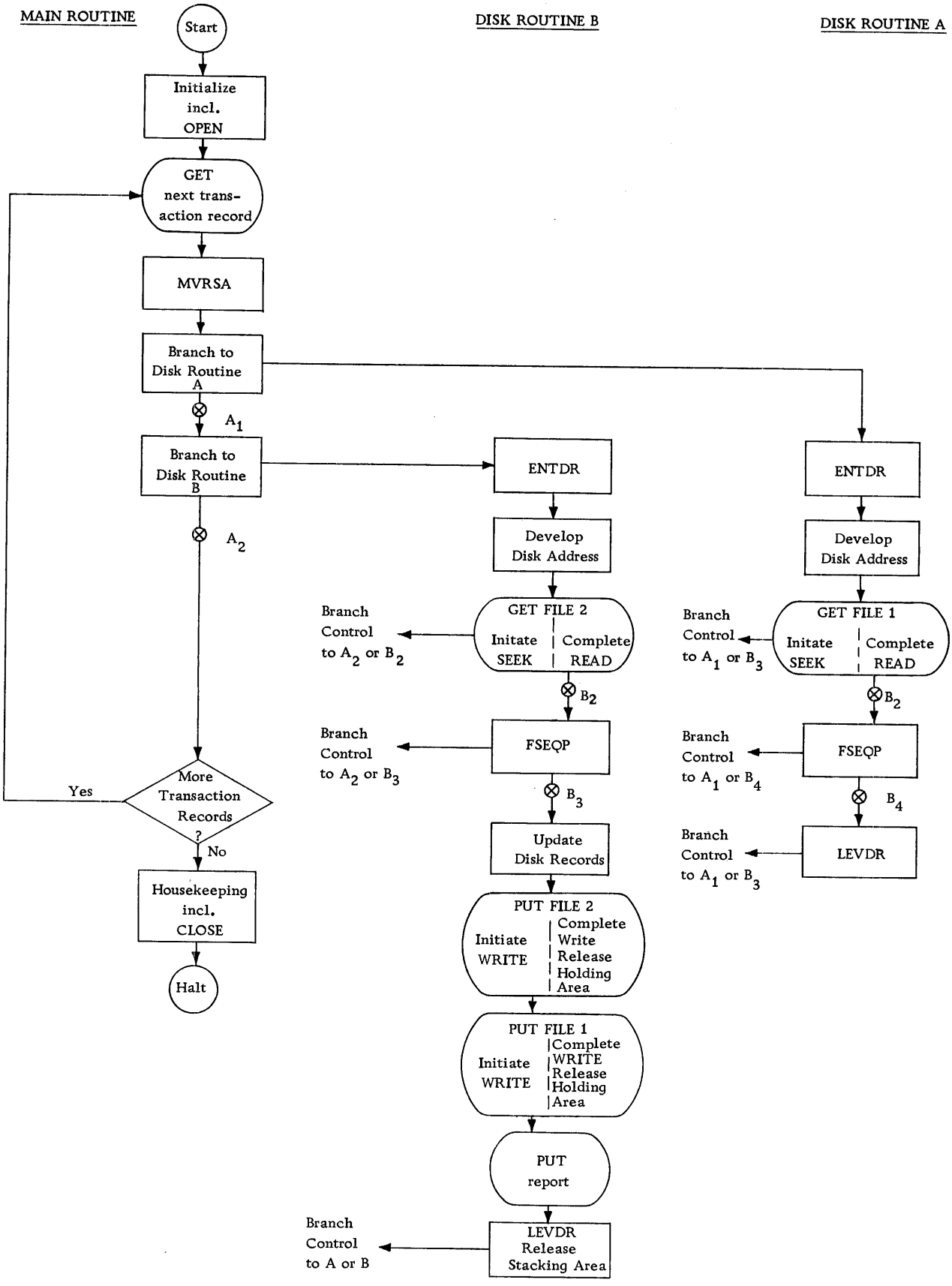



Figure 8. Random-Processing - Two Dependent Disk Routines Use Data Obtained from the Main Routine

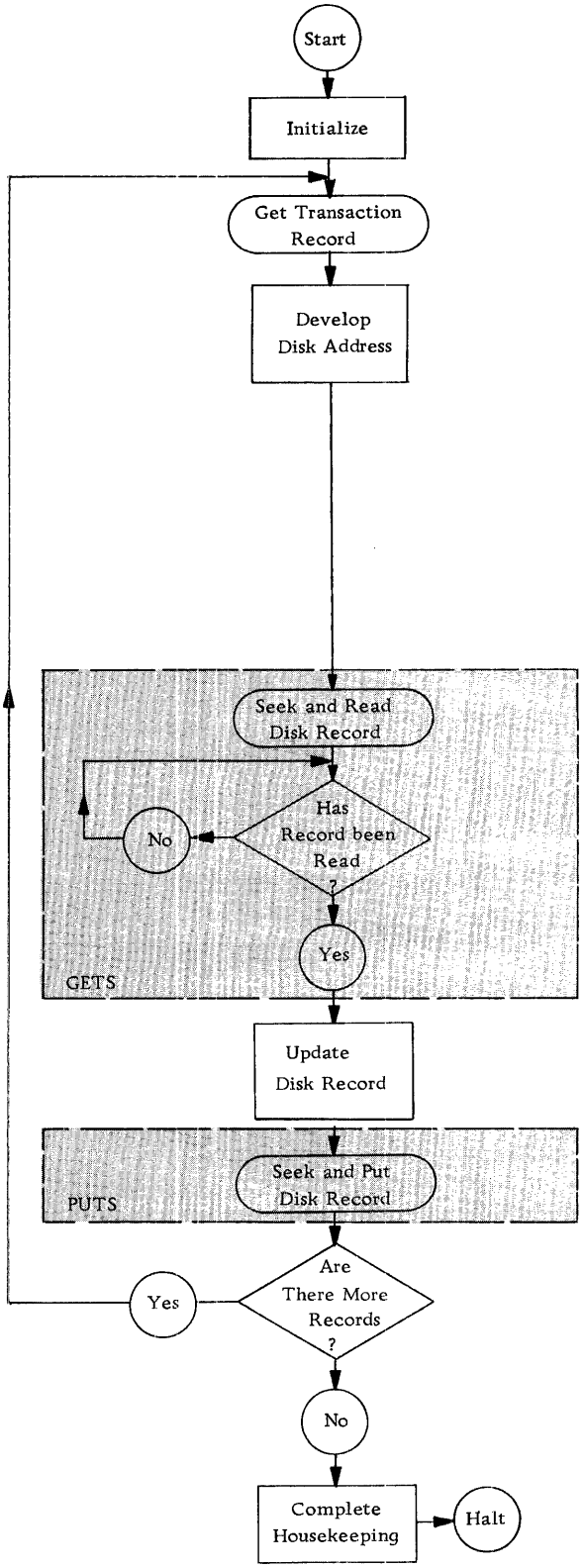



Figure 9. Method 1 of Single-Reference Processing

This section describes how single-reference processing can be handled by the 1301 IOCS.

METHOD 1

Figure 9 illustrates one method of single-reference processing available to users of the 1301 IOCS. It will be referred to as "Method 1."

As shown in Figure 9, Method 1 falls into the category of "in-line processing" described above because processing halts while a disk SEEK, READ, or WRITE operation is in process.

As indicated, the program reads a transaction record and develops the address of the disk information required to process the record. If a SEEK for the same record is not already in progress, a SEEK-and-READ command for the record is initiated, and processing halts until the record has been read into core storage. The record is then updated, put back into disk storage, and the next transaction record is read.

The shaded areas in Figure 9 indicate the functions that are performed by the GETS and PUTS macro-instructions when the programmer chooses this method of single-reference processing. See the sections on the GETS and PUTS macro-instructions for a detailed description of the functions of these macro-instructions.

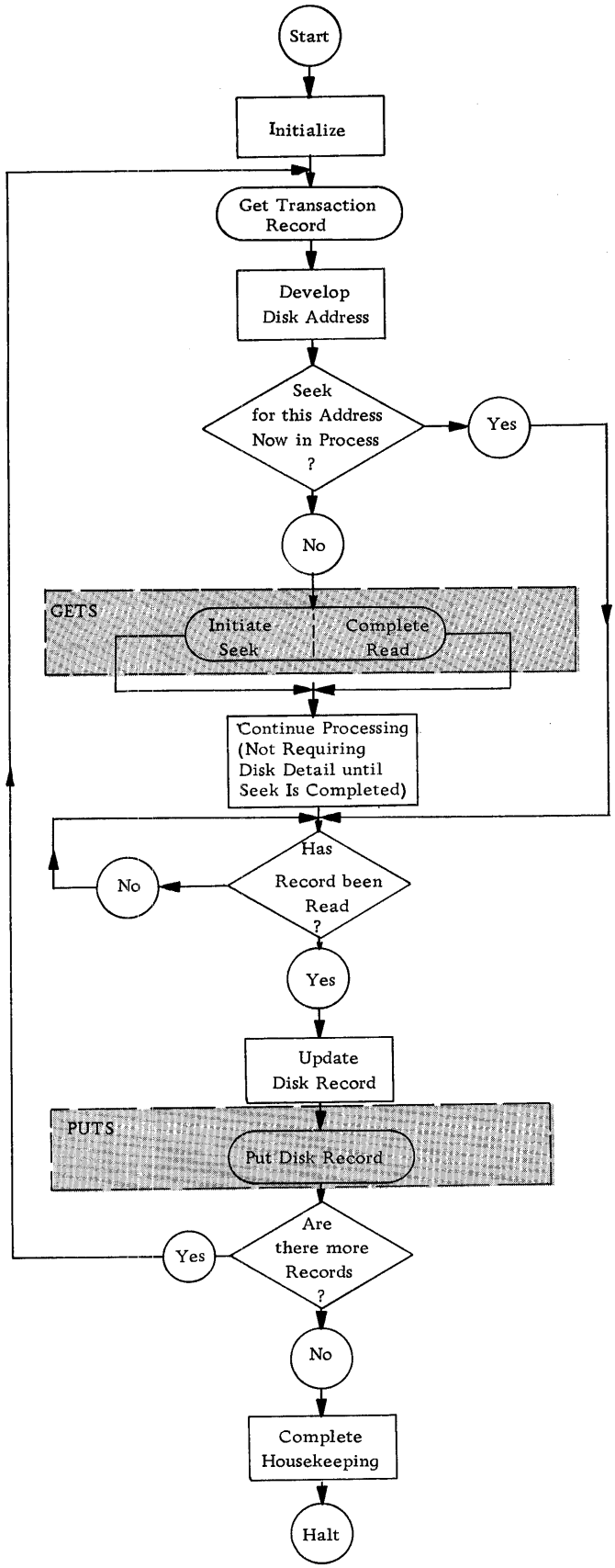



Figure 10. Method 2 of Single-Reference Processing

## METHOD 2

Figure 10 illustrates a second method of single-reference processing available to users of the 1301 IOCS. This method will be referred to as "Method 2."

As illustrated, Method 2 enables the user to overlap the SEEK and READ operations of the disk SEEK-READ operation with processing -- provided such processing does not require the disk record for which the SEEK was initiated.

As shown in Figure 10, the program reads a transaction record and develops the address of the disk information required to process the transaction record. If a SEEK for the same record is not already in progress, the SEEK is initiated. As soon as this has been done, processing resumes and continues until an interrupt signal provided by the 1301 IOCS indicates that the SEEK has been completed. The disk READ operation is then initiated, and processing resumes. Upon completion of the READ operation, the 1301 IOCS sets a switch indicating that the READ operation has been completed.

The test for completion of the disk READ operation is made just prior to the time the disk information is needed by the program. This permits overlapping of the SEEK and READ operations with processing, as described. If the disk record has not yet been obtained when it is needed by the program, the latter enters a waiting loop until the record has been obtained. At this point, Method 2 assumes the characteristics of in-line processing because processing halts until the required disk record has been obtained. See Figure 10.

NOTE: When Method 2 is used, the test for completion of the disk READ operation must be provided by the user unless he uses the WAITS macro-instruction. For details, see the description of the SWITCH operand of the GETS macro-instruction and the WAITS macro-instruction.

## Relationship of Card/Tape IOCS to 1301 IOCS

The 1301 IOCS augments the IBM 1410 Card/Tape IOCS by adding to the latter the routines required to include IBM 1301 Disk Storage in the resulting combined IBM 1410 Input/Output Control System. The 1301 IOCS furnishes the error routines and the disk arm and file schedulers for random, sequential, and single-reference processing. The Card/Tape IOCS provides the remaining routines required to include 1301 Disk Storage in the general IBM 1410 IOCS. See Figure 11.

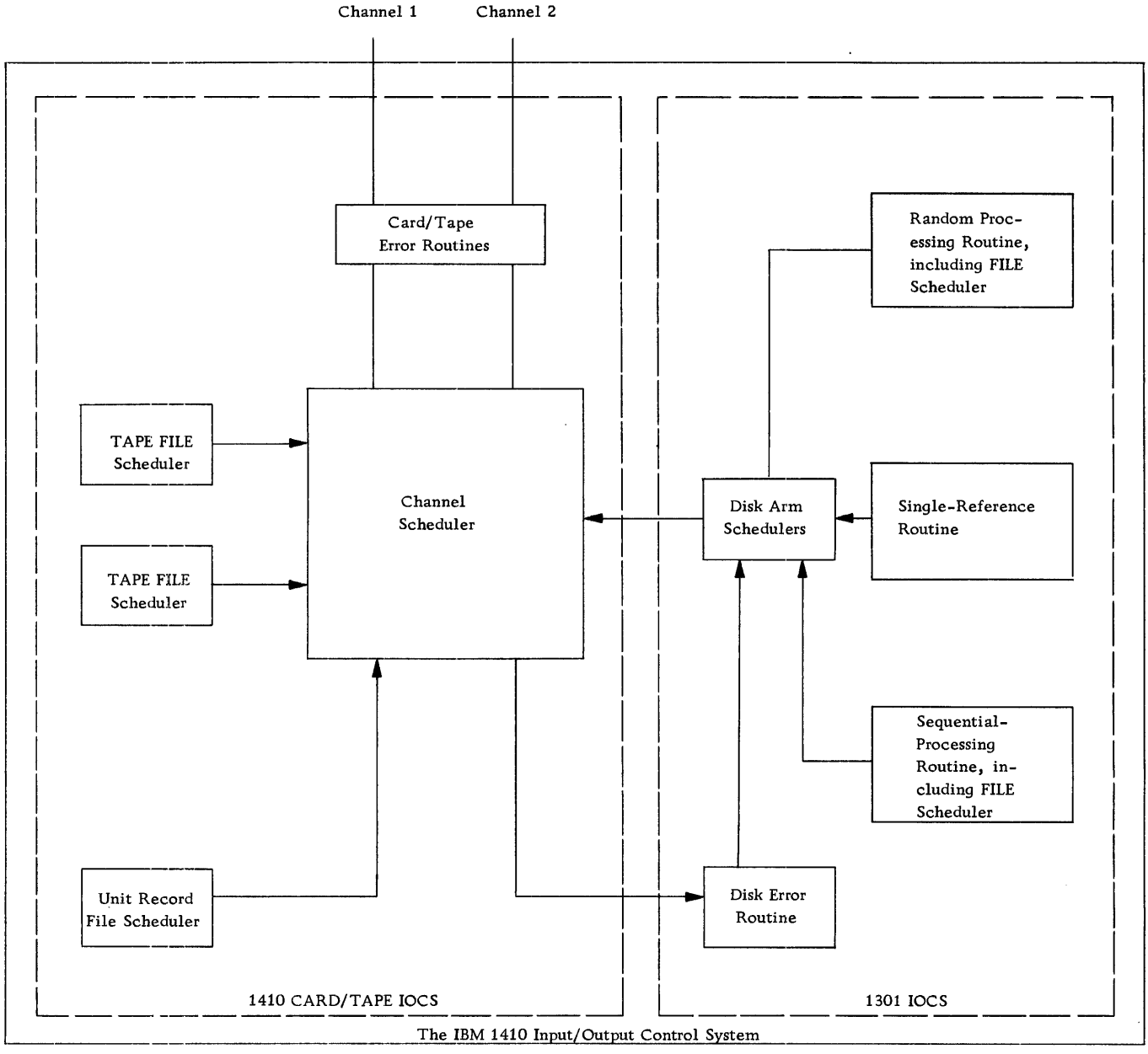



Figure 11. Relationship of the IBM 1410 C/T IOCS to the 1301 IOCS

Each time the 1410 program issues a disk SEEK, READ or WRITE request, the 1301 IOCS determines:

(1) which input or output area is to be used and
(2) whether the disk arm needed to handle the specified operation is available.

The disk arm schedulers then pass the request on to the channel scheduler furnished by the Card/Tape IOCS.

## Advantages of the 1301 IOCS

The IBM 1301 Disk Input/Output Control System consists of a set of tested routines that free the user from all coding of input and output routines for IBM 1301 Disk Storage.

### Random Processing

The 1301 IOCS provides the coding required for the simultaneous execution of any number of SEEK, READ, and WRITE operations and for the overlapping of all disk input and output operations with processing.

### Sequential Processing

The 1301 IOCS enables the programmer to handle logical records in sequential applications merely by using GET and PUT and related macro-instructions, thereby relieving him of all blocking and deblocking of disk records.

### Single-Reference Processing

The 1301 IOCS provides input/output and error-checking routines for single-reference processing. The IOCS also furnishes an interrupt signal upon completion of each SEEK, READ and WRITE operation. The user has the option of

(1) having the 1301 IOCS automatically halt processing until each SEEK, READ or WRITE operation is completed, or
(2) using the interrupt signal furnished by the 1301 IOCS to test whether READ operations have been completed.

NOTE: The coding required for this test can be furnished by the WAITS macro-instruction.

### Available IOCS Routines

The 1301 IOCS routines will automatically:

- schedule several SEEK-and-READ operations for simultaneous execution;
- overlap disk input/output operations with processing;
- schedule all available arms of the 1301 Disk Storage Units;
- seek, read and write disk records;
- check for read and write errors;
- correct (all correctable) read and write errors, and
- block and deblock sequential disk records.

### Substantial Savings

The design and coding of an efficient input/output control system permitting the overlapping of processing and disk operations, including the simultaneous scheduling of arms, is a difficult programming task. By providing tested routines that handle all of these functions, the IBM 1301 Input/Output Control System offers users substantial savings in program writing, testing and operating expenses.

## Using the 1301 IOCS

### For Random and Sequential Processing

For each such program that is to utilize the 1301 IOCS, the programmer must:

(1) use the appropriate 1301 IOCS macro-instructions in his program;
(2) write the required DIOCS entries;
(3) write the required DTF entries, and
(4) write DA (Define Area) statements for the Transaction Stacking and Disk Record Holding Areas used by his program.

### For Single-Reference Processing

For each such program that is to utilize the 1301 IOCS, the programmer must:

(1) use the GETS and PUTS macro-instructions in his program;
(2) write the required DIOCS entries; and
(3) write DA (Define Area) statements for the Holding Area Control Records used by his program.

## Assembly of Programs Using the 1301 IOCS

The DIOCS and DTF entries are punched into IBM cards and must precede the source program during Autocoder assembly. The DIOCS cards are entered following the DIOCS header card of the Card/Tape IOCS. The DIOCS cards of the two systems may be intermixed.

The required sets of disk DTF cards, each preceded by the appropriate disk DTF header card, are entered together with the sets of DTF entries for the Card/Tape IOCS. The two sets may be intermixed. See Figure 12.

The remainder of this publication consists primarily of a detailed explanation of the four programming steps required to utilize the IBM 1301 Input/Output Control System, namely, the writing of:

1. IOCS macro-instructions;
2. DIOCS entries;
3. DTF entries; and
4. DA entries.

The final section of this bulletin contains information regarding the size of the 1301 IOCS; a coding example, and a brief glossary.

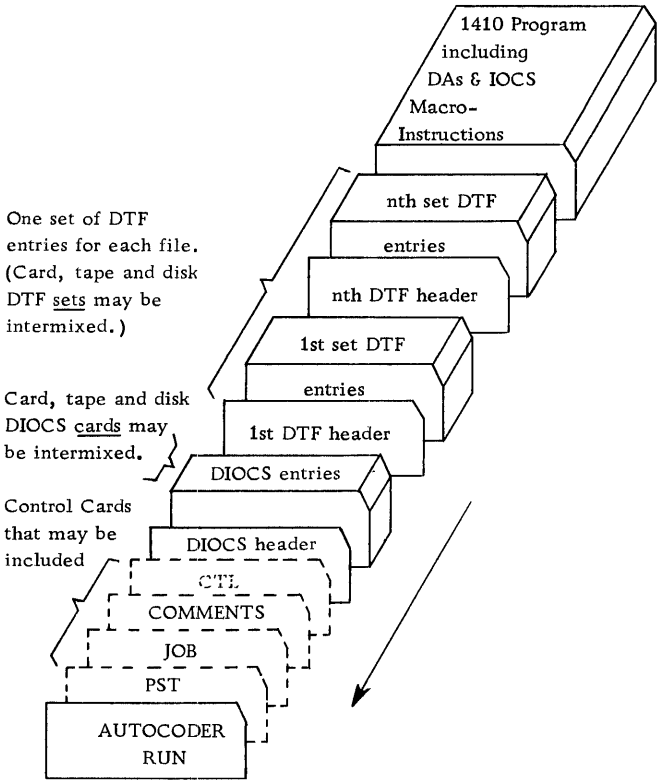



Figure 12. Assembly of Programs using the 1410 Card/Tape and 1301 Disk IOCS

FOR RANDOM PROCESSING: The eight macro-instructions for random processing are:

| | |
|---|---|
| OPEN | "Open File(s)" |
| CLOSE | "Close File(s)" |
| MVRSA | "Move Record to Stacking Area" |
| ENTDR | "Enter Disk Routine" |
| GET | "Get Logical Record" |
| FSEQP | "Force Sequential Processing" |
| PUT | "Put Logical Record" |
| LEVDR | "Leave Disk Routine" |

FOR SEQUENTIAL PROCESSING: The four macro-instructions for sequential processing are:

| | |
|---|---|
| OPEN | "Open File(s)" |
| CLOSE | "Close File(s)" |
| GET | "Get Logical Record" |
| PUT | "Put Logical Record" |

FOR SINGLE-REFERENCE PROCESSING: The three macro-instructions for single-reference processing are:

| | |
|---|---|
| GETS | "Get, Single Reference" |
| PUTS | "Put, Single Reference" |
| WAITS | "Wait, Single Reference" |

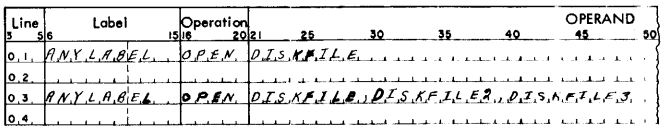
Each macro-instruction is described in detail below.

## OPEN

By using the OPEN macro-instruction, the programmer can let the 1301 IOCS handle the various initializing tasks which must be performed before data on a disk file can be used by the object program. These initializing functions differ for random and sequential processing and are explained separately below.

The OPEN macro-instruction must be given in the main routine and is written as indicated in Figure 13.

The operand in Figure 13 contains the name or names of the file(s) to be activated. The name(s)




must be those used to describe the file(s) in the DTF header line. If more than one file is named in the operand, the names must be separated by commas. Only one OPEN macro-instruction is needed to open all files used by the program, including any non-disk files.

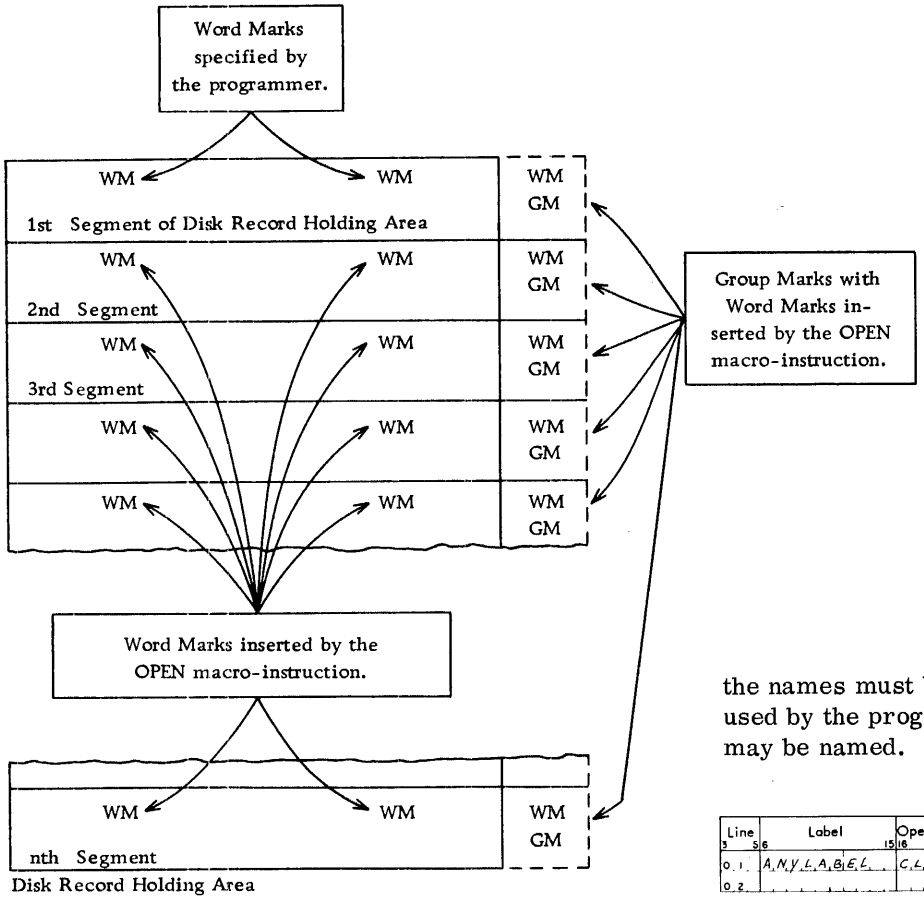
### What This Macro Will Do

RANDOM AND SEQUENTIAL PROCESSING. For each file named in the operand of the macro-instruction, the 1301 IOCS -- on the basis of the information contained in the DTF entries -- will:

(1) make the appropriate Disk Record Holding Area available;
(2) insert a Group Mark with Word Mark immediately to the right of each segment of the Disk Record Holding Area; and
(3) move the Word Marks specified in the first section of the Disk Record Holding Area into all other sections of the Disk Record Holding Area. See Figure 14.

RANDOM PROCESSING. The OPEN macro-instruction will cause the Transaction Stacking Area to be made available for incoming transaction records.

SEQUENTIAL PROCESSING. The OPEN macro-instruction will cause the track address of the first record of the file to be inserted into the Track Address Counter of the 1301 IOCS. The initial track address is obtained from the information supplied by the DTF 'FILESTART' entry.




This area must be specified by the programmer by means of an appropriate DA (Define Area) entry. See section describing "Define Area Entries Needed to Support the IOCS."

Figure 14. Word Marks and Group Marks inserted by the OPEN macro-instruction

## CLOSE

The programmer may use the CLOSE macro-instruction to have the 1301 IOCS develop all the coding required to close the disk file(s), i.e., to remove the disk file(s) from use by the object program. The CLOSE macro-instruction must be given in the main routine and is written as indicated in Figure 15.

| Line | Label | Operation | OPERAND |
|---|---|---|---|
| 01 | ANYLABEL | CLOSE | DISKFILE |
| 02 | | | |
| 03 | ANYLABEL | CLOSE | DISKFILE1,DISKFILE2,DISKFILE3 |
| 04 | | | |

Figure 15.

The operand in Figure 15 contains the name or names of the file(s) to be closed. The name(s) must be those used to describe the file(s) in the DTF header line. If more than one file is named in the operand, the names must be separated by commas. Any file used by the program, including any non-disk files, may be named. See Figure 16.

| Line | Label | Operation | OPERAND |
|---|---|---|---|
| 01 | ANYLABEL | CLOSE | DISKFILE1,DISKFILE2,TAPEINFILE |
| 02 | | | |

Figure 16.

### What This Macro Will Do

RANDOM AND SEQUENTIAL PROCESSING. The 1301 IOCS will develop all the coding required to:

(1) check whether all pending disk operations involving the files named in the operand have been completed, and
(2) close the file(s) named in the operand after all pending disk operations have been completed.

SEQUENTIAL PROCESSING. The 1301 IOCS will develop all the coding required to:

(1) check whether partially filled output blocks remain to be written on the output file(s) named in the operand of the macro-instruction;
(2) write out any partially filled output blocks;
(3) pad partially filled output blocks with the character specified by the DTF "PADDING" entry; and
(4) pad partially filled output blocks with blanks if the DTF "PADDING" entry was omitted.

NOTE 1: The following characters may not be used for padding: Asterisk, Tape Mark, Word Sepa-

rator Character, Record Mark, Cent Sign and Group Mark.

NOTE 2: When a sequential input file is closed by means of the 1301 IOCS 'CLOSE' macro-instruction, the last address written will be made available to the user in two ways: 1) a message containing the address will be typed on the console printer and 2) the address will be stored within the IOCS. The field within the IOCS that will contain this address is labeled 'IOCSCLFLD'. It is a 26-position field. The label refers to the high-order position of this field. The address may be used by the next program to specify file limits for other sequential files. In the case of single-record operation, however, the record address supplied in the message does not necessarily apply to the final address or any address used in the program.

## MVRSA (Move Record to Stacking Area)

All data developed by the main routine and required by the Disk Routine(s) must be placed into the Transaction Stacking Area. This insures that the main routine does not alter the data before it has been used by the Disk Routine(s). This transfer of data from the main routine to the Disk Routine(s) permits the separation of the Disk Routine(s) from the main routine discussed above.

The programmer may use the MVRSA macro-instruction to transfer data developed in the main routine to a segment of the Transaction Stacking Area specified by the DIOCS "STKAREA" entry. The data will be retained in the Transaction Stacking Area until all disk operations using the data have been completed.

The MVRSA macro-instruction must be given in the main routine before control is branched to the Disk Routine. See Figure 17.

The two formats of the MVRSA macro-instruction are written as indicated in Figures 18 and 21.

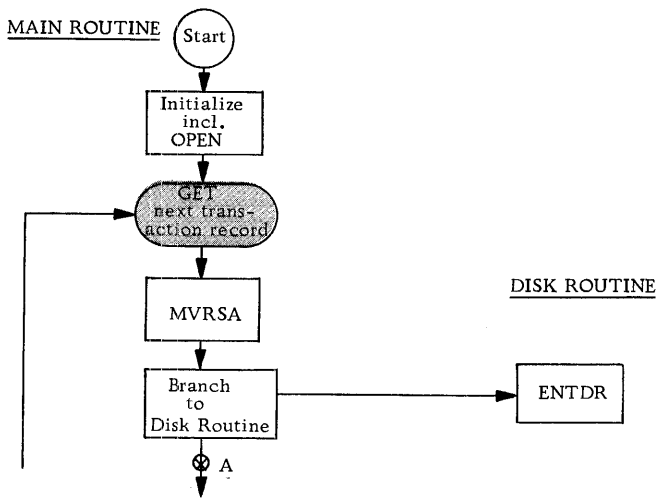



Figure 17. Use of the MVRSA Macro-Instruction

### Format A

Format A is written as indicated in Figure 18. The operand identifies the high-order position of the area from which information is to be moved to the Transaction Stacking Area. The areas from which information is to be moved must have a Record Mark or a Group Mark with Word Mark immediately to the right of the low-order position. See Figures 19 and 20.

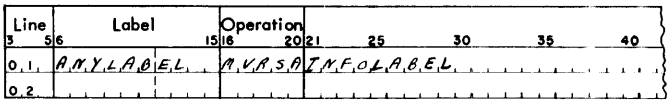



Figure 18.

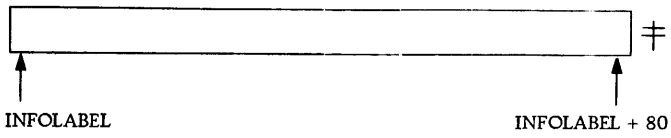



Figure 19. Area Referred to by the Operand of the MVRSA Macro-Instruction

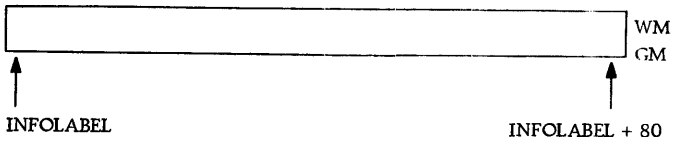



Figure 20. Area Referred to by the Operand of the MVRSA Macro-Instruction

### What This Format of the Macro Will Do

This format of the MVRSA macro-instruction will cause the 1301 IOCS to:

(1) select an available segment of the Transaction Stacking Area specified by the DIOCS "STKAREA" entry;
(2) have the program enter a waiting loop if no segment of the Transaction Stacking Area is available;
(3) insert the address of the selected segment into the indexing register specified by the DIOCS "STKINDEX" entry; and
(4) move the information and the Word Marks contained in the area specified by the operand into the segment of the Transaction Stacking Area selected by the 1301 IOCS.

Format B

Format B of the MVRSA macro-instruction has no operand and is written as indicated in Figure 21.

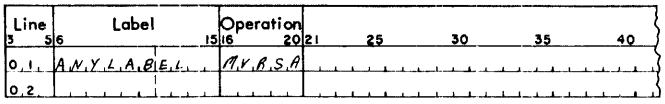



Figure 21.

This format of the MVRSA macro-instruction enables the programmer to move information into the Transaction Stacking Area by actual move commands.

NOTE: When doing so, the programmer must index the B-Address with the index register specified by the "STKINDEX" entry.

What This Format of the Macro Will Do

This format of the MVRSA macro-instruction will cause the 1301 IOCS to:

(1) select an available segment of the Transaction Stacking Area specified by the DIOCS "STKAREA" entry;
(2) cause the program to enter a waiting loop if no segment of the Transaction Stacking Area is available at the time of the request; and
(3) insert the address of the selected section of the Transaction Stacking Area into the indexing register specified by the DIOCS "STKINDEX" entry.

Thus, the programmer is free to move the desired data into the Transaction Stacking Area by means of actual move commands, using the specified indexing register.

## ENTDR (Enter Disk Routine)

The ENTDR macro-instruction must be the first instruction used in any Disk Routine of a program using the 1301 IOCS.

This macro-instruction develops the coding required to store the return address of the main routine. This is the address to which control will be branched by the 1301 IOCS to continue processing of the main routine. See Figure 22.

The ENTDR macro-instruction does not have an operand and is written as indicated in Figure 23.

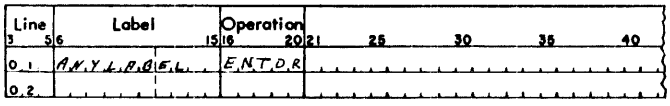



Figure 23.

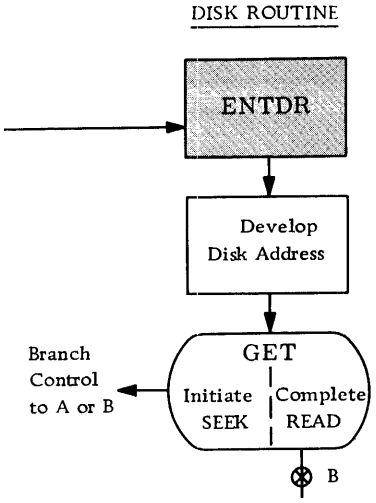



Figure 22. Use of the ENTDR Macro-Instruction

What This Macro Will Do

The coding provided by the ENTDR macro-instruction stores the return address of the main routine. This permits resumption of processing in the main routine as soon as the next disk operation has been initiated.

## GET

The programmer may use the GET macro-instruction to make a disk record available for processing.

The two formats of the GET macro-instruction are described below.

Format A

This format of the GET macro-instruction is written as indicated in Figure 24.

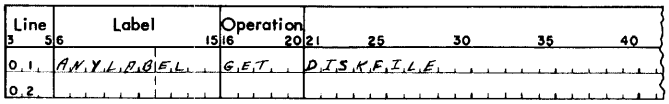



Figure 24.

The operand in Figure 24 is the name of the disk file from which records are to be obtained. The name must be that used to describe the file in the DTF header line.

What This Macro Will Do

The functions of this macro-instruction depend on whether it is used for random or sequential processing.

RANDOM PROCESSING. Before using this macro-instruction for random processing, the programmer must store the appropriate address in the eight-character field labeled IOCSDSKAD, as shown in Figure 25.

| Channel | Number of Disk Unit | Track or Record Address | | | | | |
|---|---|---|---|---|---|---|---|
| @ or * | | | | | | | |

IOCSDSKAD

Figure 25.

A corresponding coding example is shown in Figure 26.

| Line | Label | Operation | |
|---|---|---|---|
| 0.1 | | M.L.N.B. | A,D,D,R,F,I,E,L,D,,I,O,C,S,D,S,K,A,D, |
| 0.2 | | G.E.T. | D,T,S,K,F,I,L,E |
| 0.3 | | | |

Figure 26.

In either single-record or full-track random processing, the user must store in IOCSDSKAD the channel designation and the number of the disk unit, as indicated. The remainder of IOCSDSKAD is either HA1 and HA2, the track address, or the six-character record address of the desired record.

If the single-record mode is used in a random processing application, the user must place the four-digit track address into the field labeled IOCSSEKAD. (This field is located within the IOCS and is the four-character field immediately to the left of IOCSDSKAD.)

If the disk address is a track address, IOCSDSKAD must contain the four-digit HA1 followed by the two-digit HA2. In full-track operations, there is no record address and the field IOCSSEKAD is not used. Each time the programmer uses the GET macro-instruction in full-track mode, the 1301 IOCS will develop the coding required to:

(1) check whether another disk operation is using the disk track specified by the disk address in the IOCSDSKAD location;
(2) let the program enter a waiting loop if the required disk track is being used by another disk operation;
(3) assign a disk arm to read the information;
(4) assign the segment of the Disk Record Holding Area into which the disk record is to be read;
(5) seek the track specified by the disk address;
(6) read the disk record into the assigned segment of the Disk Record Holding Area;
(7) check whether another disk record is ready for processing in the Disk Record Holding Area;
(8) branch control to the disk routine if another disk record is waiting to be processed;
(9) check whether a segment of the Transaction Stacking Area is available;
(10) branch control to the main routine if a Stacking-Area segment is available;
(11) check for disk read errors;
(12) correct correctable read errors*, and
(13) release the segment of the Disk Record Holding Area used by a GET immediately preceding the present GET macro-instruction. (See section describing the FSEQP macro-instruction.)

* On DATA CHECK indications, eight additional attempts to execute the command are made before an error message is printed on the console printer.

SEQUENTIAL PROCESSING. Format A of the GET macro-instruction causes the 1301 IOCS to develop the coding required to:

(1) make the next logical record available for processing;
(2) take the next logical record from the next track, if a block of records has been exhausted;
(3) check for read errors;
(4) correct correctable read errors*;
(5) increase the Track Address Counter whenever records of the current track have been exhausted;
(6) check whether the address contained in the Track Counter is a valid disk address for the system defined by the DIOCS entries for this program, and
(7) branch the program to the location specified in the DTF "EOFADDR" entry if the track address exceeds the address specified in the DTF "FILEND" entry.

The area into which records are placed by this format of the GET macro-instruction depends on record type and the number of input/output areas, as follows:

1. For blocked files using only one input/output area and for all files using two input/output areas:
   a. If indexing is used, this macro-instruction leaves the logical record in the input area and places the address of the record's high-order position into the specified index register.
   b. If indexing is not used, this macro-instruction places the next logical record into the work area specified by the DTF "WORKAREA" entry.

2. For unblocked files using only one input/output area:
   This macro-instruction leaves the next logical record in the input areas.

Format B

This format of the GET macro-instruction is used only for sequential processing and is written as indicated in Figure 27.

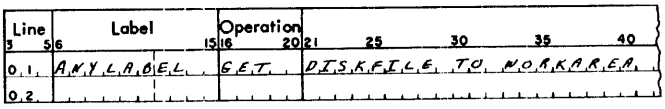



Figure 27.

The first entry in the operand of Figure 27 is the name of the disk file from which records are to be obtained. The name must be that used to describe the file in the DTF header line. The second entry is the name (label) given to the work area to which the record is to be moved. This format of the GET macro-instruction may be used for all record formats except unblocked records that use only one input area.

## PUT

The programmer may use the various formats of the PUT macro-instruction to develop the coding required to include processed or unprocessed records in a disk output file.

What the PUT Macros Will Do

In addition to the functions listed under each format, all PUT macro-instructions will cause the 1301 IOCS to develop the coding required to:

(1) check that the disk arm required to write the information is available;
(2) check for disk write errors;
(3) correct correctable write errors*, and
(4) release the segment of the Disk Record Holding Area that contained the information placed into disk storage.

For the purpose of discussion, PUT macro-instructions will be divided into three types: that used to return updated records to disk storage, that used to load information into sequential locations in disk storage, and that used to load information into non-sequential locations in disk storage.

* On DATA CHECK indications, eight additional attempts to execute the command are made before an error message is printed on the console printer.

RETURNING UPDATED RECORDS TO DISK STORAGE. This type of PUT macro-instruction is used to return updated disk records to the locations in disk storage in which they were originally contained. It may be used for both random and sequential processing and is written as indicated in Figure 28.

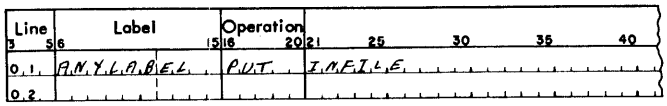



Figure 28.

The operand in Figure 28 is the name of the disk file from which information was taken for updating. The name must be that used to describe the file in the DTF header line.

What This Macro Will Do

This type of PUT macro-instruction will develop the coding required to return a record (which contains the updated information) from the Disk Record Holding Area to the disk file named in the operand of the macro-instruction.

NOTE: When using this type of PUT macro-instruction, the programmer must process the disk records in the Disk Record Holding Area.

LOADING RECORDS INTO SEQUENTIAL DISK-STORAGE LOCATIONS. This type of PUT macro-instruction has two formats. Both are used to load records into sequential disk-storage locations as described below.

Format A

This format is written as indicated in Figure 29.

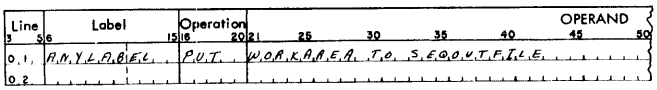



Figure 29.

The first entry in the operand field in Figure 29 is the name of a work area defined by a DA statement. The name of the file must be that used to describe the file in the DTF header line.

What This Format of the Macro Will Do

This macro-instruction will develop the coding required to write successive logical records from a work area into sequential locations in disk storage.

Format B

This format may be used to load records contained in a tape file into sequential disk-storage locations and is written as indicated in Figure 30.

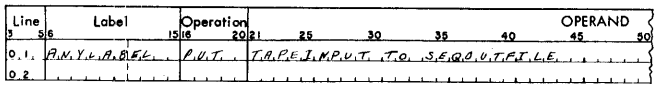


Figure 30.

The first entry in the operand in Figure 30 is the name of the tape file from which records are to be taken. The last entry in the operand is the name of the disk file into which the tape records are to be loaded. The names of both files must be those used to define the files in their respective DTF header lines.

What This Format of the Macro Will Do

This macro-instruction will develop the coding required to write successive records from the specified tape file into the specified disk file. The tape records are placed into the Disk Record Holding Area.

LOADING RECORDS INTO NON-SEQUENTIAL DISK-STORAGE LOCATIONS. This type of PUT macro-instruction may be used to load records into nonsequential locations in disk storage. (An example of this kind of application is the loading of new part records into an existing inventory file.) Before using this type of PUT macro-instruction, the programmer must:

(1) place the address of the disk location (into which the information is to be placed) into the location labeled IOCSDSKAD, and
(2) place the information to be written into disk storage into a segment of the Disk Record Holding Area (which is addressed by the indexing register that was specified by the DTF "INDEXREG" entry of the file).

This type of PUT macro-instruction is written as indicated in Figure 31.

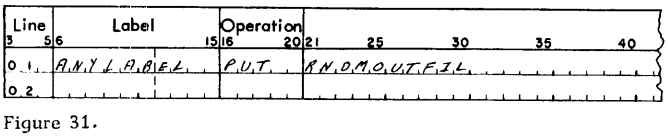


Figure 31.

The operand in Figure 31 is the name of the disk file into which records are to be loaded. The name must be that used to describe the file in the DTF header line.

What this Macro Will Do

This PUT macro-instruction will develop the coding required to load records contained in the Disk Record Holding Area into specified locations in disk storage.

NOTE 1: This type of PUT macro-instruction cannot be used to return updated disk records to disk storage.
NOTE 2: This macro-instruction causes the replacement of the entire contents of a disk track, depending on the record format used. The programmer is cautioned against inadvertently destroying disk data when using this macro-instruction.

## FSEQP (Force Sequential Processing)

The programmer may use the FSEQP macro-instruction to insure that disk records obtained by the disk routine(s) will be processed and returned to disk storage in the same order in which the corresponding transaction records were obtained by the main routine. See Figure 32.

Such synchronization of the disk routine(s) with the main routine is important whenever reports are written by the disk routine(s).

For example: Assume that Module 1 receives a request for information with an access time of 100 ms, and that 10 ms later Module 2 receives a request with access time of 50 ms. In this case, the arm of Module 2 will obtain the specified information before the arm of Module 1.

If the Disk Routine does not use the FSEQP macro-instruction, the information obtained by the arm of Module 2 will be processed and the result returned to disk storage before the information sought by the arm of Module 1 can be read. In this case, the updated information will not be returned to disk storage in the order in which the corresponding transaction records were read by the main routine.

If the Disk Routine uses the FSEQP macro-instruction, the information obtained by the arm of Module 2 will not be processed until the information obtained by the arm of Module 1 has been processed and written back into disk storage.

The FSEQP macro-instruction has no operand and is written as indicated in Figure 33. It may be used anywhere in the program.

What This Macro Will Do

Each time the programmer inserts a FSEQP macro-

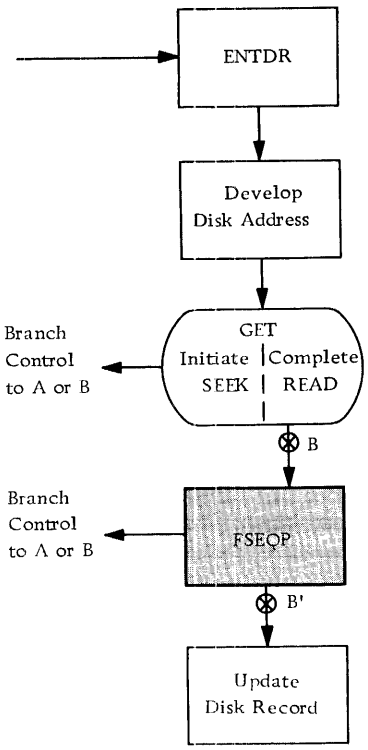



Figure 32. Use of the FSEQP Macro-Instruction

| Line | Label | Operation | |
|---|---|---|---|
| 0 1 | ANYLABEL | FSEQP | |
| 0 2 | | | |

Figure 33.

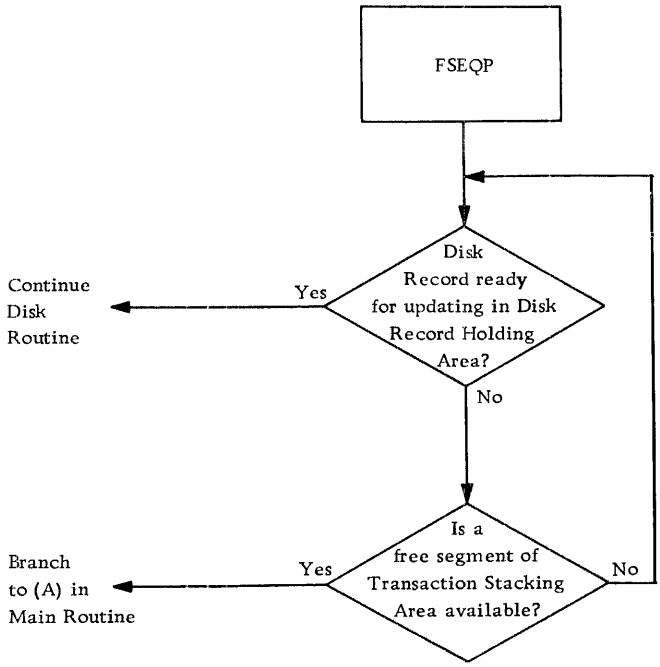



Figure 34. Control Functions of the FSEQP Macro-Instruction

instruction in his program, the 1301 IOCS will develop the coding required to:

(1) halt the processing of all transaction data until all data from previous transactions have been processed;
(2) check whether another disk record is ready for processing in the Disk Record Holding Area;
(3) branch control to the disk routine if another disk record is waiting to be processed;
(4) check whether a segment of the Transaction Stacking Area is available;
(5) branch control to the main routine if a Stacking Area segment is available, and
(6) branch control to a waiting loop if neither a disk record nor a transaction record can be processed. See Figure 34.

NOTE: If the FSEQP macro-instruction is given in the main routine, processing in the main routine will not continue until all disk-storage data has been processed.

## Additional Functions of the FSEQP Macro-Instruction

The FSEQP macro-instruction has two important additional functions. It can be used to prevent:

(1) the release of a segment of the Disk Record Holding Area by the second of two successive GET macro-instructions, or
(2) the release of a segment of the Disk Record Holding Area by an LEVDR macro-instruction.

Both functions are described below.

RETENTION OF DISK DATA AFTER SECOND GET MACRO. Each time two disk GET macro-instructions follow one another in a program, the second GET causes the release of the information obtained by the first GET macro-instruction. Thus, the coding sequence:

```
GET DISKFILE1
GET DISKFILE2
PROCESS
PUT DISKFILE2
```

will cause the release of the information obtained by the first GET; and only the information obtained by the second GET can be moved to PUT. The FSEQP macro-instruction may be used to prevent the release of the information obtained by the first GET macro-instruction, as indicated by the following coding sequence:

```
GET DISKFILE1
FSEQP
GET DISKFILE2
PROCESS
PUT DISKFILE2
PUT DISKFILE1
```

In this case, the FSEQP macro-instruction prevents the release of the information obtained by the first GET.

NOTE 1: The order of the PUTs takes place as indicated. In general, arms that have obtained information from disk storage remain in position until the updated information is returned to storage. This eliminates SEEK time for PUTs, since the disk arm is already in position.

NOTE 2: The FSEQP macro-instruction cannot be used to hold information obtained by the first of two GETs involving the same file. Thus, in the coding sequence:

```
GET DISKFILE1
FSEQP
GET DISKFILE1
PROCESS
```

the FSEQP macro-instruction cannot prevent the release of the information obtained by the first GET.

RETENTION OF DISK DATA AFTER THE 'LEVDR' MACRO. The LEVDR macro-instruction, too, releases the information obtained by the last GET, as described in the discussion of the LEVDR macro-instruction. Thus, the coding sequence:

```
GET DISKFILE1
PROCESS
LEVDR
```

will cause the release of the information obtained by the GET macro-instruction.

The FSEQP macro-instruction may be used to prevent the release of disk-record information by the LEVDR macro-instruction. Thus, in the coding sequence:

```
GET DISKFILE1
FSEQP
LEVDR
```

the information obtained by the GET will be retained in the Disk Record Holding Area for processing by a subsequent disk routine.

NOTE 1: A FSEQP macro-instruction can cause erasure of information in the Disk Record Holding Area followed by rereading of the disk record. For this reason, the contents of a segment of the Disk Record Holding Area must not be changed between the GET and the FSEQP macro-instructions. Thus, the coding sequence:

```
GET DISKFILE1
UPDATE
FSEQP
```

might result in the retention of the non-processed information contained in the Disk Record Holding Area.

NOTE 2: The FSEQP macro-instruction may follow only those GET macro-instructions that refer to disk storage data.

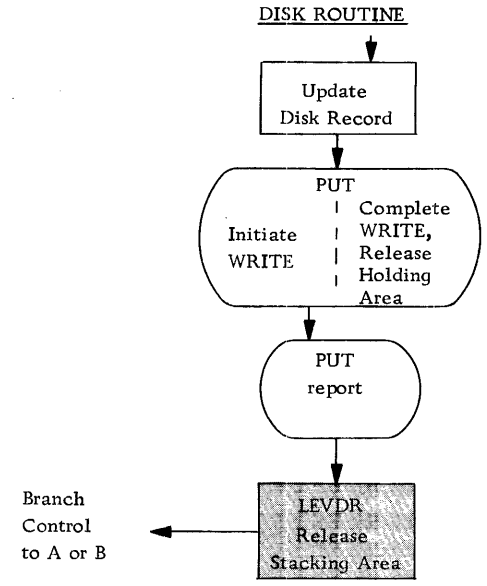



Figure 35. Use of the LEVDR Macro-Instruction

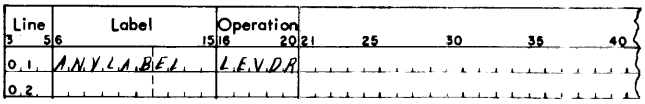



Figure 36.

## LEVDR (Leave Disk Routine)

Each time processing of a set of transaction data has been completed by the disk routine, the work areas used by the data must be released and control must be returned to the main routine. The LEVDR macro-instruction, when used by the programmer as the last instruction in a disk routine, will cause the 1301 IOCS to develop all the coding required to handle these functions. See Figure 35.

The LEVDR macro-instruction does not have an operand and is written as indicated in Figure 36.

### What This Macro Will Do

Each time the programmer uses the LEVDR macro-instruction, the 1301 IOCS will develop the coding required to:

(1) check whether the segment of the Transaction Stacking Area is required by another disk routine;

(2) free the segment of the Transaction Stacking Area used by a completely processed transaction;
(3) check whether another disk record is ready for processing in the Disk Record Holding Area;
(4) branch control to the Disk Routine if another disk record is waiting to be processed;
(5) check whether a segment of the Transaction Stacking Area is available;
(6) branch control to the main routine if a Stacking Area segment is available, and
(7) release the segment of the Disk Record Holding Area used by a GET immediately preceding the present LEVDR macro-instruction. (See the section describing the FSEQP macro-instruction.)

## GETS

The programmer may use the GETS macro-instruction to perform any of the following 1301 disk operations:

1. Seek a specified disk cylinder;
2. Seek and Read a single disk record;
3. Seek and Read a full track with or without addresses, and
4. Seek and Read a full cylinder (optional feature).

When a 'GETS' macro-instruction is used within a Disk Routine, single-reference logic takes precedence over random logic. Therefore, no return to the main-line routine takes place as in the random processing use of the 'GET' macro-instruction. Any fields that are constructed or referenced in the user's program prior to the 'GETS' macro-instruction remain unchanged and may be used after resumption of the user's program.

NOTE: The GETS macro-instruction thus can be used to perform the operations identified by the mnemonic operation codes SD, RD, RHA, RFT, RDT and RCY.

What This Macro Will Do

Each time the programmer uses a GETS macro-instruction, the 1301 IOCS will develop the coding required to:

(1) check whether the disk arm required to perform the operation is available;
(2) "stack" the request for the required disk arm if that arm is not available;
(3) issue a Seek Disk command to position the access arm on the cylinder containing the required information;
(4) set a switch to indicate that the disk READ operation has been completed.

Additional functions of the GETS macro-instruction are described below.

HOLDING AREA CONTROL FIELD. Before using the GETS macro-instruction, the programmer must furnish the 1301 IOCS with information describing the precise nature of the operation to be performed. The programmer must enter this information in a 30-character area known as the "Holding Area Control Field," or, simply, "Control Field." The programmer must reserve this control field in core storage by means of a DA entry, as indicated in Figure 37.

| Line | Label | Operation | |
|---|---|---|---|
| 3 5 | 6 15 | 16 20 | 21 25 30 35 40 |
| 0.1 | HLDAREACTL | DA | 1,X,2,9,,G |
| 0.2 | AREANAME | | 1,,5 |
| 0.3 | CODE | | 6,,6 |
| 0.4 | | | 7,,7 |
| 0.5 | HAI | | 8,,1,1 |
| 0.6 | DISKINSTR | | 1,2,,1,5 |
| 0.7 | OPMODE | | 1,2 |
| 0.8 | CHANNEL | | 1,3 |
| 0.9 | F | | 1,4 |
| 1.0 | OPERATION | | 1,5 |
| 1.1 | DMODIFIER | | 1,6,,1,6 |
| 1.2 | SWITCHADR | | 1,7,,2,1 |
| 1.3 | DISKARM | | 2,2,,2,2 |
| 1.4 | DISKADR | | 2,3,,2,9 |
| 1.5 | | DA | RXL |
| 1.6 | IOAREA | | |
| 1.7 | | | |

Figure 37.

The control field must immediately precede the input area to be used. See Figure 38.

The following information must be placed into the Holding Area Control Field before the GETS macro-instruction is encountered by the program:

AREANAME (Positions 1-5). This field is reserved for the 1301 IOCS.

CODE (Position 6). This field contains one BCD character whose bit configuration is determined as follows:

B = ON if wrong-length-record checking is to be performed by the 1301 IOCS; otherwise OFF.
A = OFF
8 = OFF
4 = ON if Write Disk Check operation is to be performed; otherwise OFF.
2 = ON if only a SEEK operation is to be performed; otherwise OFF. Modification of SEEK only operations while the SEEK is in

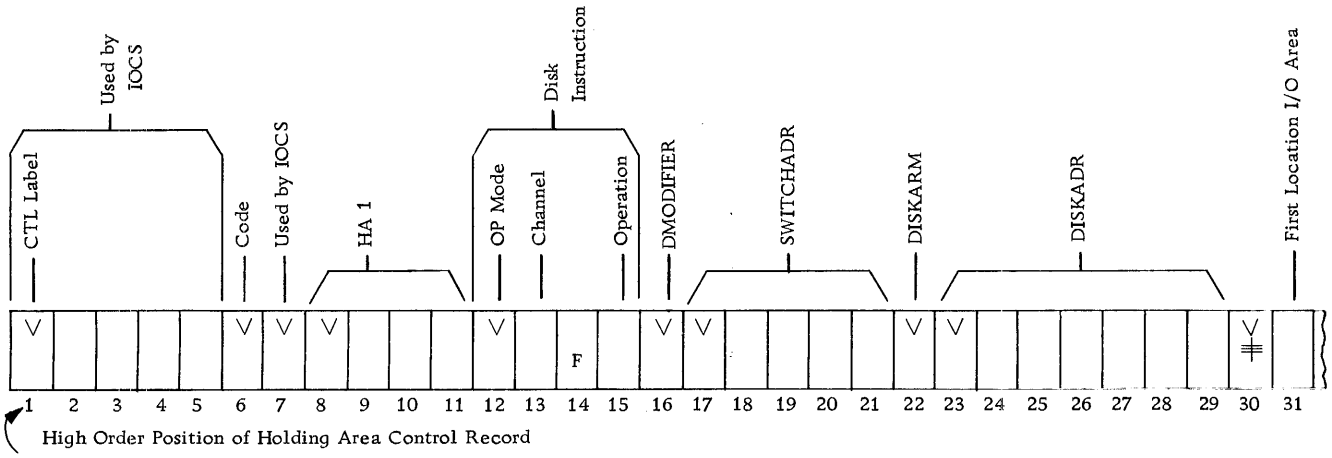



Figure 38. The Holding Area Control Record

progress is described in the section "Additional Information for Programmers."

1 = OFF

The BCD characters formed by these various bit configurations are listed in Figure 91. See section, "Additional Information for Programmers."

None (Position 7). Reserved for use by IOCS.

HA1 (Positions 8-11). Contains Home Address 1.

NOTE: Needed only if single records are to be read or written.

DISKINSTR (Positions 12-15). Contains the machine-language operation code of the operation to be performed. This is of the form ABFn, where:

A = M if the operation is to be performed in the Move-mode;
L if the operation is to be performed in the Load mode.

B = @ if the disk unit is attached to Channel 1;
* if the disk unit is attached to Channel 2.

F = F

n = 1 if a single record is to be read;
2 if a full track without addresses is to be read;
5 if a full track with home address is to be read;
6 if a full track with addresses is to be read;
@ if a full cylinder is to be read.

DMODIFIER (Position 16). The d-character of the disk instruction. This is

R

unless a "to-end-of-core" read operation is to be performed, in which case the entry is

$

SWITCHADR (Positions 17-21). Contains the address of the core-storage location of the character used as the Read-Operation-Complete Switch (see description of Formats C and D of the GETS macro-instruction).

NOTE: This field is left blank if Format A or B of the GETS macro-instruction is used.

DISKARM (Position 22). The digit "9" must be placed into this field before the program encounters the first GETS macro-instruction. The "9" can be assembled in this field as a constant.

NOTE: The contents of this field must not be changed by the programmer.

DISKADR (Positions 23-29). This field must contain the B-field of the disk instruction to be performed.

None (Position 30). This field must contain a group mark with word mark.

Position 31. This is the first location of the input area into which the disk information is to be read.

NOTE 1: The programmer may use any labels he wishes, but he must insure that they are unique if more than one control field is defined.

The four formats of the GETS macro-instruction are described below.

Format A

This format of the GETS macro-instruction has no operand and is written as indicated in Figure 39.

| Line | Label | Operation | |
|---|---|---|---|
| 3 5 | 6 15 | 16 20 | 21 25 30 35 40 |
| 0.1 | ANYLABEL | GETS | |
| 0.2 | | | |

Figure 39.

NOTE: Before using this format of the GETS macro-instruction, the programmer must place

the high-order address of the Holding Area Control Record (see Figure 38) into Index Register 14.

## What This Format of the GETS Macro Will Do

In addition to the functions listed at the beginning of this section, this format of the GETS macro-instruction will cause the 1301 IOCS to develop the coding required to:

(1) branch control to a waiting loop while the disk seek operation is in progress;
(2) interrupt waiting loop upon completion of the seek operation:
(3) initiate the specified disk read operation;
(4) branch control to a waiting loop while the disk read operation is in progress;
(5) interrupt waiting loop upon completion of the disk read operation:
(6) check for disk read errors, and
(7) correct correctable disk read errors.*

As indicated in Figure 40, the 1301 IOCS causes the program to enter a waiting loop until the disk record defined by the contents of the Holding Area Control Record has been read into core storage.

Program execution, therefore, proceeds as follows: Upon encountering a GETS macro-instruction, the program initiates the SEEK and then immediately branches control to the Operation-Complete Test. (See Figure 40.) When the SEEK is completed, an

* On DATA CHECK indications, eight additional attempts to execute the command are made before an error message is printed on the console printer.

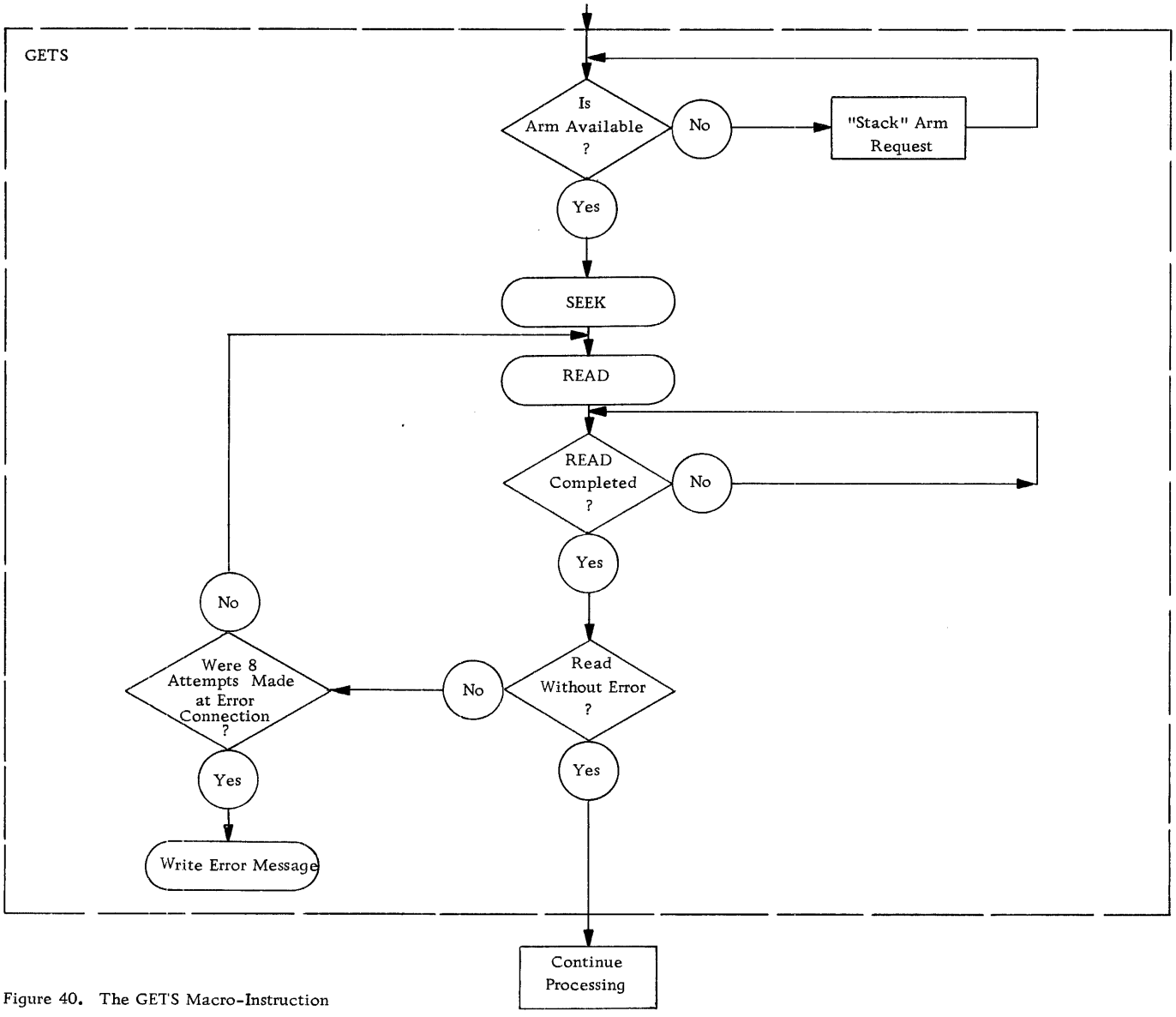



Figure 40. The GETS Macro-Instruction

IOCS-provided interrupt causes processing to initiate the READ operation. The program then again branches control to the Operation-Complete Test, and program execution does not continue until this test indicates that the READ operation has been completed. Processing then continues.

Operation-Complete Switch - The test for completion of seek and read operations is made by the 1301 IOCS as follows:

When the program encounters a GETS macro-instruction, the IOCS sets a word mark in the core-storage position designated by the contents of the location labeled SWITCHADR. See Figure 38.

The IOCS indicates completion of the SEEK and READ operations by clearing this word mark. The program, consequently, enters a waiting loop until the SEEK and READ operations are completed (i.e., until the word mark is cleared), whereupon the IOCS causes program execution to continue.

## Format B

Format B of the GETS macro-instruction is written as indicated in Figure 41.

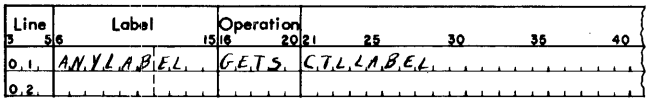


Figure 41.

The operand is the label of the high-order position of the Holding Area Control Record.

## What This Format of the GETS Macro Will Do

This format of the GETS macro-instruction performs the same functions (and develops the same coding) described above for Format A. However, when using Format B of the GETS macro-instruction, the programmer need not place the high-order address of the Holding Area Control Record into Index Register 14 prior to issuing the GETS macro-instruction.

NOTE: The functions provided by Formats A and B of the GETS macro-instruction correspond to Method 1 described in the section on "Single-Reference Processing."

## Format C

This format of the GETS macro-instruction is written as indicated in Figure 42. The operand is "SWITCH".

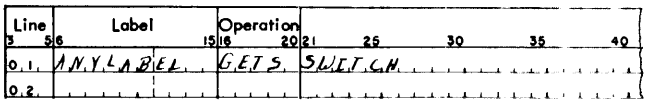


Figure 42.

NOTE: Before using this format of the GETS macro-instruction, the programmer must place the high-order address of the Holding Area Control Record into Index Register 14.

## What This Format of the GETS Macro Will Do

In addition to the functions listed at the beginning of this section, this format of the GETS macro-instruction will cause the 1301 IOCS to develop the coding required to:

(1) branch control to the main routine to continue processing while the disk seek operation is in progress;
(2) interrupt processing upon completion of the seek operation;
(3) initiate the specified disk read operation;
(4) branch control to the main routine to continue processing while the disk read operation is in progress;
(5) interrupt processing upon completion of the disk read operation;
(6) check for disk read errors, and
(7) correct correctable disk read errors.*

Operation-Complete Switch. In contrast with Formats A and B of the GETS macro-instruction, the programmer may place the Operation-Complete Switch anywhere in his program.

NOTE: This means that the programmer can use Method 2 described in the section on "Single-Reference Processing." See Figures 10 and 43.

* On Data Check indications, eight additional attempts to execute the command are made before an error message is printed on the console printer.

As indicated in Figure 43, the test for completion of the disk SEEK and READ operations is made just prior to the time the disk information is needed by the program.

This permits overlapping of the SEEK and READ operations with processing until such time as the disk record is needed for further processing. At this point, the program enters a waiting loop, and processing halts until the disk record has been obtained.

NOTE: The 1301 IOCS will set and clear a word-mark switch, in the manner described under Format A of the GETS macro-instruction, to indicate completion of SEEK and READ operations. The location of this word-mark switch is defined by the programmer, who must place the address of the core-storage location of the desired word-mark switch into the SWITCHADR entry of the Holding Area Control Record. This must be done before the program encounters the GETS macro-instruction. The programmer must provide the coding required to test the (word-mark) switch unless he uses the WAITS macro-instruction.

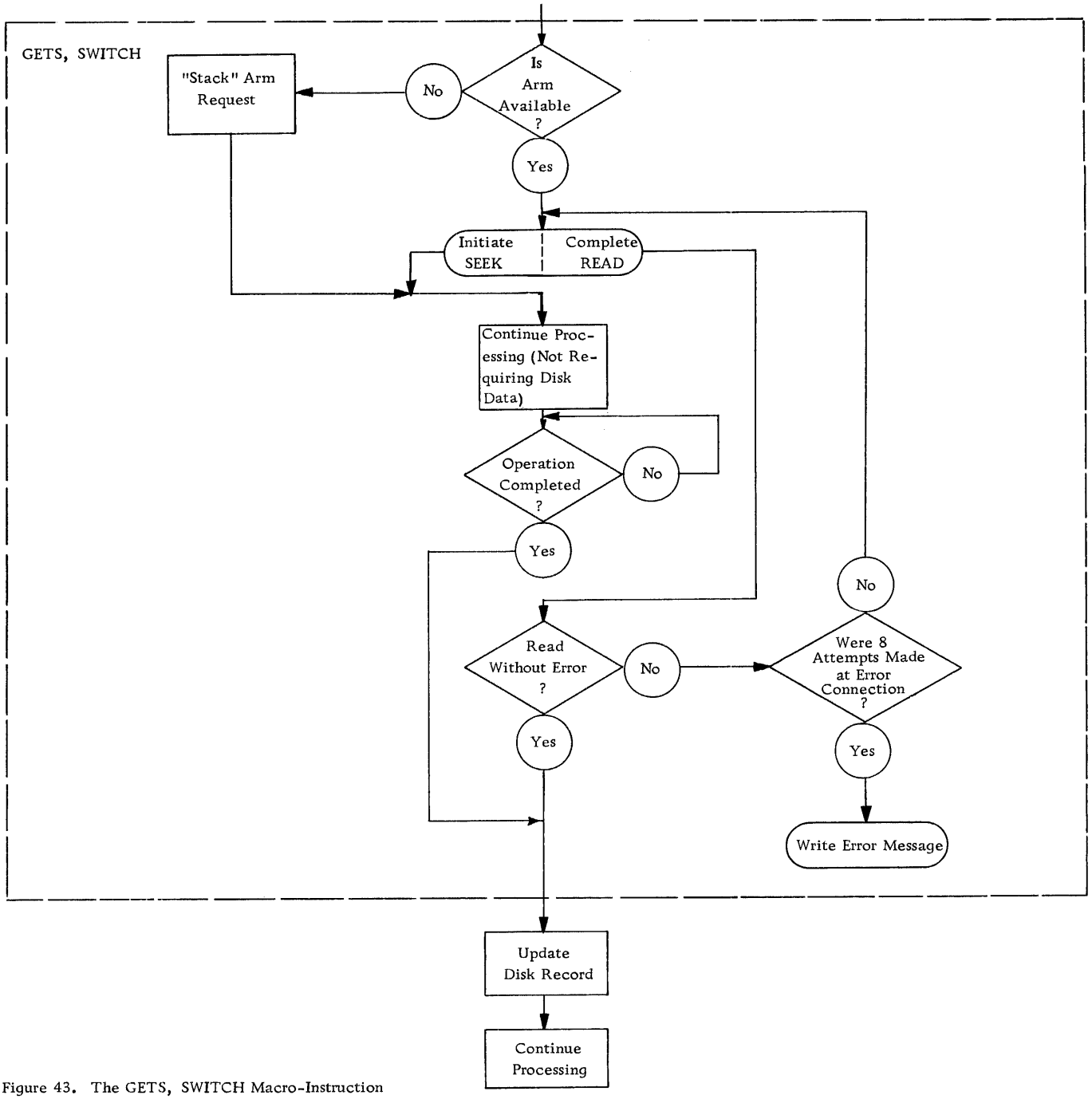



Figure 43. The GETS, SWITCH Macro-Instruction

Format D

This format of the GETS macro-instruction is written as indicated in Figure 44.

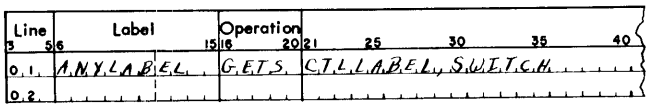


Figure 44.

The first operand shown in Figure 44 is the label of the high-order position of the Holding Area Control Record. The second operand is "SWITCH".

NOTE: The operands may be listed in any order.

What This Format of the Macro Will Do

This format of the GETS macro-instruction performs the same functions (and develops the same coding) described above for Format C. However, when using Format D of the GETS macro-instruction, the programmer need not place the high-order address of the Holding Area Control Record into Index Register 14 prior to issuing the GETS macro-instruction.

## PUTS

The programmer may use the PUTS macro-instruction to perform any of the following 1301 disk operations:

1. Seek a specified disk cylinder;
2. Seek and write a single disk record;
3. Seek and write a full track with home address;
4. Seek and write a full cylinder (optional feature), and
5. Write a format track.

When a 'PUTS' macro-instruction is used within a Disk Routine, single-reference logic takes precedence over random logic. Therefore, no return to the main-line routine takes place as in the random processing use of the 'PUT' macro-instruction. Any fields that are constructed or referenced in the user's program prior to the 'GETS' macro-instruction remain unchanged and may be used after the resumption of the user's program.

NOTE: The PUTS macro-instruction thus can be used to perform any of the operations identified by the mnemonic operation codes SD, WD, WHA, WCY and WFO.

What This Macro Will Do

Each time the programmer uses a PUTS macro-instruction, the 1301 IOCS will develop the coding required to:

(1) check whether a disk arm required to perform the operation is available;
(2) "stack" the request for the required disk arm if that arm is not available;
(3) issue a Seek Disk command to position the access arm on the cylinder on which the information is to be written, and
(4) set a switch to indicate that the disk WRITE operation has been completed.

Additional functions of the PUTS macro-instruction are described below.

Before using the PUTS macro-instruction, the programmer must furnish the 1301 IOCS with information describing the precise nature of the operation to be performed. He does this by means of the same Holding Area Control Field described in the section on the "GETS" macro-instruction.

The Holding Area Control Field is reserved in the same manner and contains the same information as the corresponding area described in the section on the "GETS" macro-instruction, except as follows:

DISKINSTR (Positions 12-15). This may contain the following (see description of the Holding Area Control Record):

1 if a single record is to be written;
2 if a full track without addresses is to be written;
5 if a full track with home addresses is to be written;
6 if a full track with addresses is to be written;
7 if a format track is to be written, and
@ if a full cylinder is to be written.

DMODIFIER (Position 16). This contains the d-character of the disk instruction. This is

W

unless a "to-end-of-core" write operation is to be performed, in which case the entry is

X

SWITCHADR (Positions 17-21). This contains the address of the core-storage location of the character used as the Operation-Complete Switch (see description of Formats C and D of the PUTS macro-instructions).

NOTE: This field is left blank if Format A or B of the PUTS macro-instruction is used.

DISKARM (Position 22). The digit "9" must be placed into this field before the program encounters the PUTS macro-instruction.

NOTE: The contents of this field must not be changed at any other time.

ANYLABEL (Position 31). This is the first location of the output area from which information is to be written into disk storage.

The four formats of the PUTS macro-instruction are described below.

Format A

This format of the PUTS macro-instruction has no operand and is written as indicated in Figure 45.

| Line | Label | Operation |
|---|---|---|
| 0 1 | ANYLABEL | PUTS |
| 0 2 | | |

Figure 45 .

NOTE: Before using this format of the PUTS macro-instruction, the programmer must place the high-order address of the Holding Area Control Record (see Figure 38) into Index Register 14.

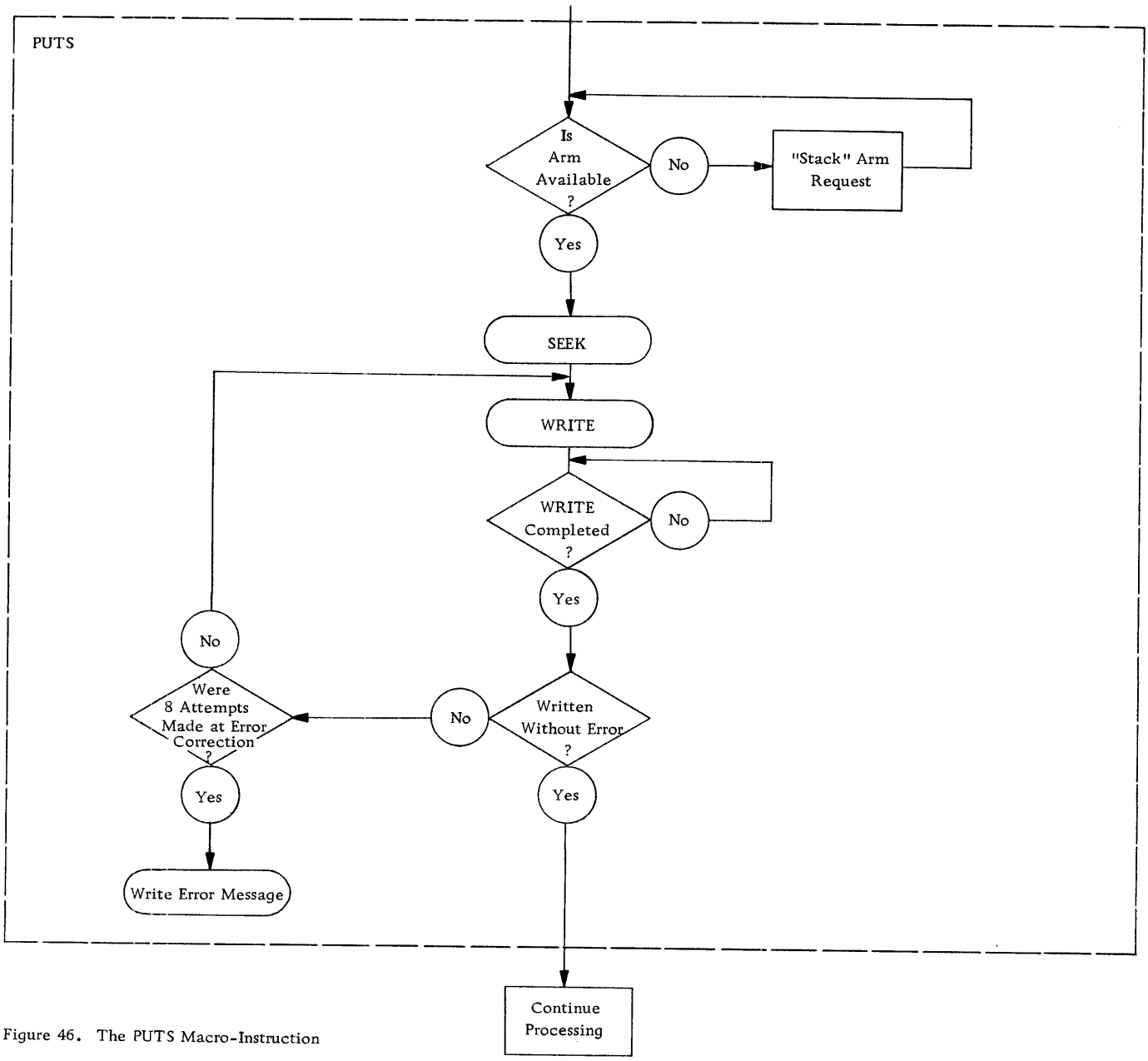



Figure 46. The PUTS Macro-Instruction

What This Format of the Macro Will Do

In addition to the functions listed at the beginning of this section, this format of the PUTS macro-instruction will cause the 1301 IOCS to develop the coding required to:

(1) branch control to a waiting loop while the disk seek operation is in progress;
(2) interrupt waiting loop upon completion of the seek operation;
(3) initiate the specified disk write operation;
(4) branch control to a waiting loop while the disk write operation is in progress;
(5) interrupt waiting loop upon completion of the disk write operation;
(6) check for disk write errors, and
(7) correct correctable disk write errors.*

As indicated in Figure 46, the 1301 IOCS causes the program to enter a waiting loop until the disk record defined by the contents of the Holding Area Control Record has been written into disk storage.

Program execution, therefore, proceeds as follows: Upon encountering the PUTS macro-instruction, the program initiates the SEEK and then immediately branches control to the Operation-Complete Test. (See Figure 46.) When the SEEK is completed, an interrupt causes IOCS to initiate the WRITE operation. The program then again branches control to the Operation-Complete Test, and program execution does not continue until this test indicates that the WRITE operation has been completed. Processing then continues.

Operation-Complete Switch. The test for completion of the SEEK and WRITE operations is the same as that described above for completion of read operations.

When the program encounters a PUTS macro-instruction, the IOCS sets a word mark in the core-storage position to which control will be branched upon initiation of the SEEK and WRITE operations. See Figure 46.

The IOCS indicates completion of the SEEK and WRITE operations by clearing this word mark. The program, consequently, enters a waiting loop until the SEEK and WRITE operations are completed (i.e., until the word mark is cleared), whereupon the IOCS causes program execution to continue.

Format B

Format B of the PUTS macro-instruction is written as indicated in Figure 47.

* On DATA CHECK indications, eight additional attempts to execute the command are made before an error message is printed on the console printer.

Figure 47.

The operand shown in Figure 47 is the label of the high-order position of the Holding Area Control Record.

What This Format of the Macro Will Do

This format of the PUTS macro-instruction performs the same functions (and develops the same coding) described above for Format A. However, when using Format B of the PUTS macro-instruction, the programmer need not place the high-order address of the Holding Area Control Record into Index Register 14 prior to issuing the PUTS macro-instruction.

NOTE: The functions provided by Formats A and B of the PUTS macro-instruction correspond to Method 1 described in the section on "Single-Reference Processing."

Format C

This format of the PUTS macro-instruction is written as indicated in Figure 48. The operand is "SWITCH".

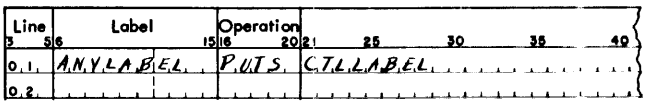



Figure 48.

NOTE: Before using this format of the PUTS macro-instruction, the programmer must place the high-order address of the Holding Area Control Record into Index Register 14.

What This Format of the PUTS Will Do

In addition to the functions listed at the beginning of this section, this format of the PUTS macro-instruction will cause the 1301 IOCS to develop the coding required to:

(1) branch control to the main routine to continue processing while the disk seek operation is in progress;
(2) interrupt processing upon completion of the seek operation;
(3) initiate the specified disk write operation;
(4) branch control to the main routine to continue processing while the disk write operation is in progress;

(5) interrupt processing upon completion of the disk write operation;
(6) check for disk write errors, and
(7) correct correctable disk write errors.*

Operation-Complete Switch. In contrast with Formats A and B of the PUTS macro-instruction, the programmer may place the Operation-Complete Switch anywhere in his program.

NOTE: This means that the programmer can use Method 2 described in the section on "Single-Reference Processing." See Figures 11 and 49.

As indicated in Figure 49, the test for completion of the disk SEEK and WRITE operations is made just prior to the time the updated disk record is replaced into disk storage. This permits overlapping of the SEEK and WRITE operations with processing.

* On DATA CHECK indications, eight additional attempts to execute the command are made before an error message is printed on the console printer.

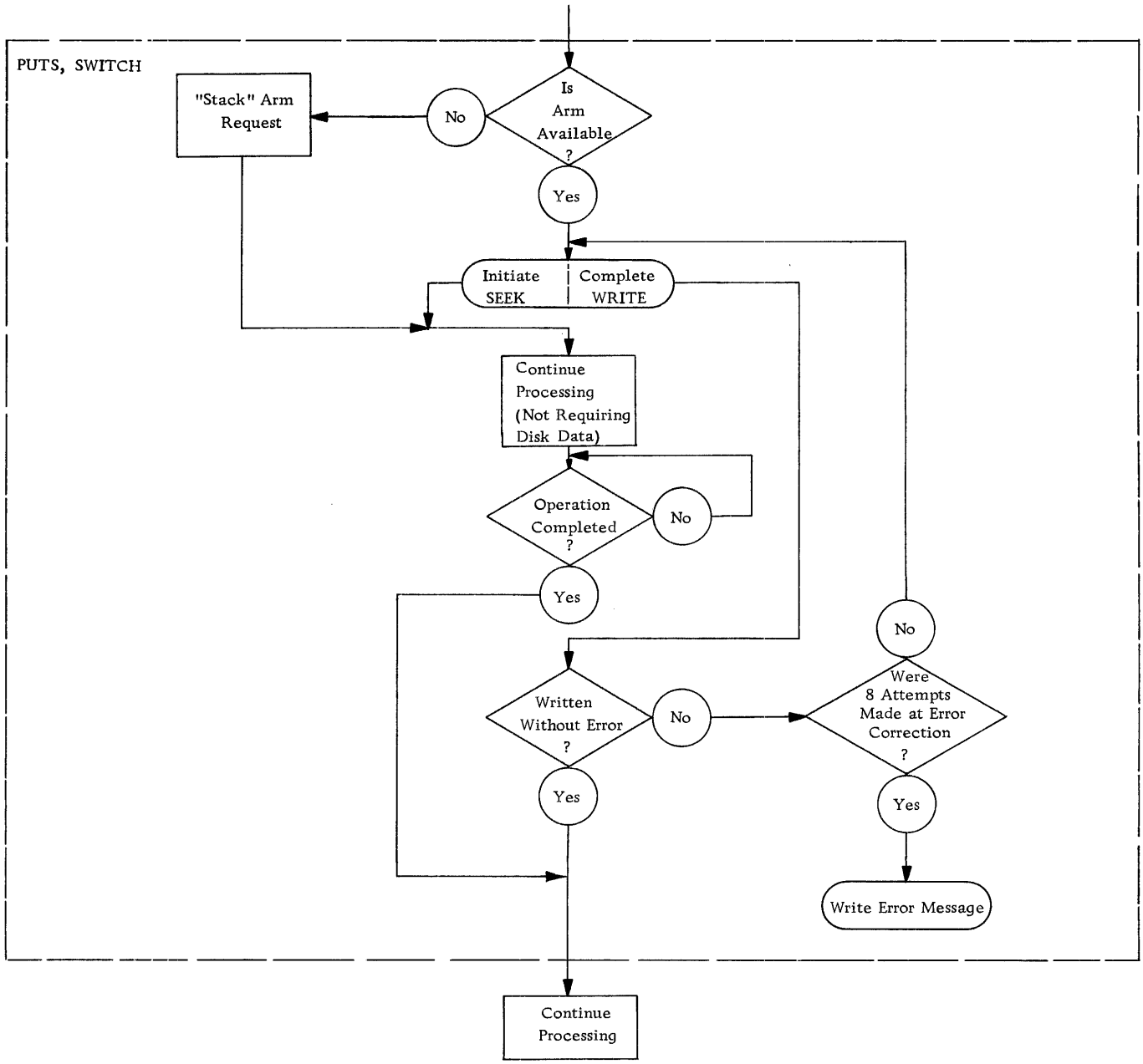



Figure 49. THE PUTS, SWITCH Macro-Instruction

NOTE: The 1301 IOCS will set and clear a word-mark switch, in the manner described under Format A of the PUTS macro-instruction, to indicate completion of the SEEK and WRITE operations. The location of this word-mark switch is defined by the programmer, who must place the address of the core-storage location of the desired word-mark switch into the SWITCHADR entry of the Holding Area Control Record. This must be done before the program encounters the PUTS macro-instruction. The programmer must provide the coding required to test the (word-mark) switch, unless he uses the WAITS macro-instruction.

Format D

This format of the PUTS macro-instruction is written as indicated in Figure 50.

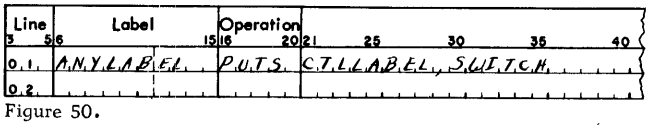


Figure 50.

The first operand in Figure 50 is the label of the high-order position of the Holding Area Control Record. The second operand is "SWITCH".

NOTE: The operands may be listed in any order.

What This Format of the Macro Will Do

This format of the PUTS macro-instruction performs the same functions (and develops the same coding) described above for Format C. However, when using Format D of the PUTS macro-instruction, the programmer need not place the high-order address of the Holding Area Control Record into Index Register 14 prior to issuing the PUTS macro-instruction.

## WAITS ("Wait Single-Reference")

The programmer may use the WAITS macro-instruction to develop the coding required to test the word-mark switch that is set and cleared by the IOCS to indicate the completion of disk input/output operations. See the description of Formats C and D of the GETS and PUTS macro-instructions.

The WAITS macro-instruction is written as shown in Figure 51. The operand of this macro-instruction ('SWITCHTAG' in Figure 51) can be any label assigned by the user, but must not be defined by the user in his source program. The label will be generated by the macro-instruction.

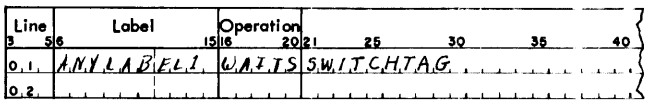


Figure 51.

What This Macro Will Do

This macro-instruction will cause the program to enter a waiting loop until the input/output operation initiated by the preceding GETS or PUTS macro-instruction has been completed. When the operation is complete, the loop will be interrupted, the word mark cleared, and control returned to the user at the point beyond this macro-instruction.

"NOTE: The programmer must place the address of the label specified in the operand of the WAITS macro-instruction (and generated by IOCS) in the Holding Area Control Record, in the field labeled 'SWITCHADR.' (See Figure 38)

## Purpose

Before the programmer can use the 1301 IOCS, he must supply the 1410 Autocoder processor with the information needed to determine which of the 1301 IOCS routines are required for the object program. This information consists of several card entries listed individually on the IBM 1401/1410 Autocoder Coding Sheet. These entries specify the sections of the 1301 Input/Output Control System to be included in the object program, and are known collectively as the DIOCS ("Define Input/Output Control System") entries. Each entry is described in detail below.

## General Format

The first DIOCS entry is mandatory and consists of the mnemonic code DIOCS in the operation field. It is known as the "DIOCS header line." This card must be the first card (except for special control cards) to enter the system during Autocoder assembly.

NOTE: Only one DIOCS header card is permitted. It is normally supplied by the DIOCS header line written for Card/Tape IOCS. See Figure 12.

Each subsequent 1301 IOCS entry has a blank operation field and must have one of the labels listed below.* All DIOCS entry cards may contain comments. These must be separated from the DIOCS entry by at least two adjacent blanks. The DIOCS entries may be listed in any order and may be intermixed with the Card/Tape DIOCS cards. See Figure 12.

## List of DIOCS Entries

This section describes the purpose of each of the following DIOCS entries:

| | |
|---|---|
| FEATURES | STKINDEX |
| CHANx | SGMTLENGTH |
| PROCESTYPE | DISKARMS |
| RNDMDEPTH | DISKOPTION |
| STKAREA | NORCDEXIT |

* Labels used in programs to be assembled by the 1410 Autocoder processor must not have more than 10 characters.

### FEATURES

This DIOCS entry is mandatory. Its operands are OVERLAP, PRIORITY. See Figure 52.

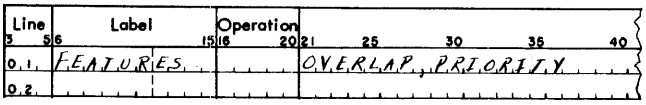



Figure 52.

### CHANx

This DIOCS entry is mandatory. It is used to indicate the channel to which the 1301 Disk Storage unit used by the program is attached. The "x" in the CHANx label is

1 if the 1301 Disk Storage unit is attached to Channel 1, and

2 if the 1301 Disk Storage unit is attached to Channel 2.

The operand of the CHANx entry is "1301".

NOTE: If there is a 1301 Disk Storage unit attached to each channel, a CHAN1 and a CHAN2 entry must be made.

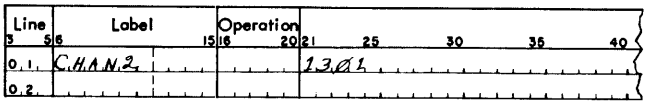



Figure 53.

The entry in Figure 53 indicates that the 1301 Disk Storage unit used by the program is attached to Channel 2.

### PROCESTYPE

This entry is not needed for single-reference processing.

This DIOCS entry causes the inclusion in the object program of the file schedulers required for sequential and/or random processing. Its operands are:

RANDOM -- if the program calls for random processing of disk files, and

SEQUENTIAL -- if the program calls for sequential processing of disk files.

If both operands are used (i.e., if the program calls for both random and sequential disk processing), the operands must be separated by a comma. They may be listed in any order.

| Line | Label | Operation | |
|---|---|---|---|
| 01 | PROCESSTYPE | | RANDOM |
| 02 | | | |

Figure 54.

The operand in Figure 54 indicates that the program calls for random processing of one or more disk file(s).

| Line | Label | Operation | |
|---|---|---|---|
| 01 | PROCESSTYPE | | SEQUENTIAL,RANDOM |
| 02 | | | |

Figure 55.

The operand in Figure 55 indicates that the program calls for both random and sequential disk processing operations.

RNDMDEPTH ("Random Depth")

This entry is required only for random files that are used for input operations. The operand indicates the maximum number of pending operations that are to be stacked. This number is the greater of two amounts: the number of arms used by a random file, or the number of random files serviced by a single arm.

NOTE: For optimum arm scheduling, it is suggested that the programmer specify the total number of available arms (not exceeding ten).

| Line | Label | Operation | |
|---|---|---|---|
| 01 | RNDMDEPTH | | 3 |
| 02 | | | |

Figure 56.

The operand in Figure 56 indicates that up to three pending operations can be stacked at one time.

STKAREA ("Stacking Area")

This entry is required only for random files used for input operations. The operand of the STKAREA entry is the label of the DA (Define Area) statement that defines the Transaction Stacking Area. See Figures 57 and 58.

The operand in Figure 58 indicates that the label of the DA statement that defines the Transaction Stacking Area of the program is labeled STACKAREA. See Figure 57.

| Line | Label | Operation | |
|---|---|---|---|
| 01 | STKAREA | | STACKAREA |
| 02 | | | |

Figure 58.

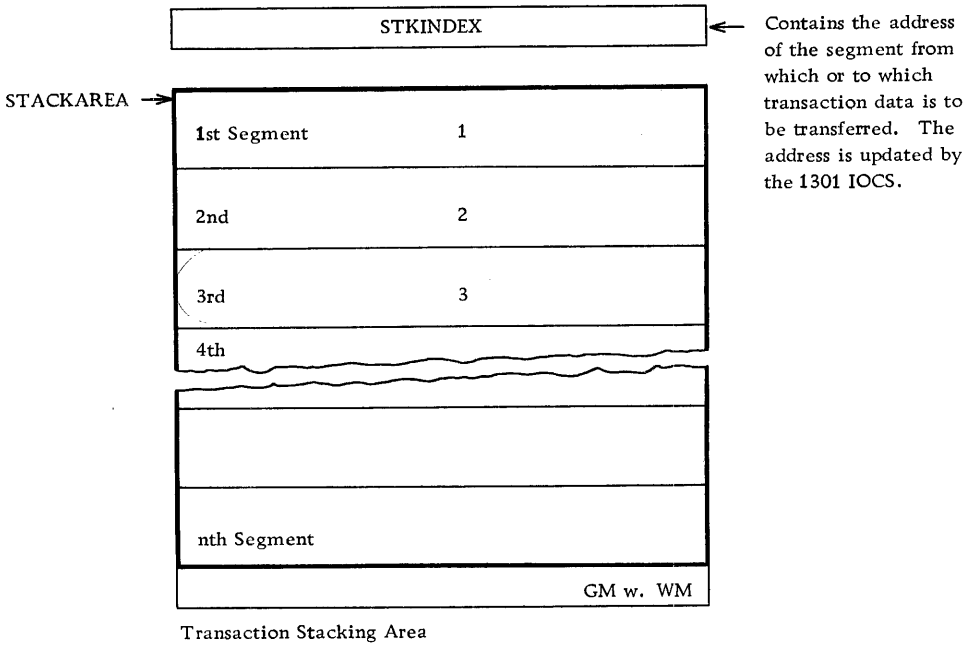



Figure 57. The Transaction Stacking Area

STKINDEX ("Stacking Index")

This entry is required only for random files used for input operations.

The operand of the STKINDEX entry is:

y where "y" is X1, X2,..., X12 and identifies the index register assigned to the Transaction Stacking Area. This index register contains the address of the segment of the Transaction Stacking Area from which successive sets of transaction data are to be taken. See Figures 4 and 6.

NOTE: Index Registers 13-15 may not be used for this purpose.

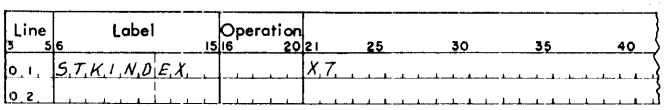



Figure 59.

The operand in Figure 59 indicates that Index Register 7 has been assigned to the Transaction Stacking Area of the program.

SGMTLENGTH ("Segment Length")

This entry is required only for random files used for input operations.

The operand of the SGMTLENGTH entry is:

x where "x" is an integer indicating the number of positions of each segment of the Transaction Stacking Area.

NOTE 1: If variable-length transaction records are to be moved to the Transaction Stacking Area, the segments must be large enough to hold the maximum-size record.

NOTE 2: Each segment must contain a location for the Record Mark or Group Mark with Word Mark that terminates the move operation.

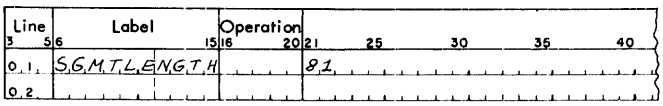



Figure 60.

The operand in Figure 60 indicates that each segment of the Transaction Stacking Area has 81 positions. See also Figure 57.

DISKARMS

The operand of the DISKARMS entry is the maximum number of modules of 1301 Disk Storage used by the program.

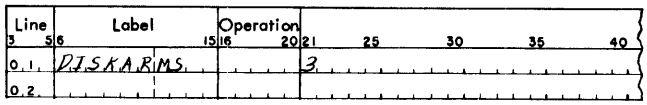



Figure 61.

The operand in Figure 61 indicates that the program uses three modules of 1301 Disk Storage.

DISKOPTION

This entry is needed only

(1) if the program reads or writes single disk records, or
(2) if the program calls for one or more write disk check operations, or
(3) if the 1301 Disk Storage unit addressed by the program is shared with a 7000-series computer.

The entry is used to indicate whether the program calls for any of the above disk operations.

The operands of the DISKOPTION entry are:

SINGLERCD -- if the program reads and/or writes single records.

WRITECHECK -- if the program calls for at least one Write Disk Check operation.

SHARED -- if the 1301 Disk Storage is shared by a 7000-series computer.

The operands may be listed in any order.

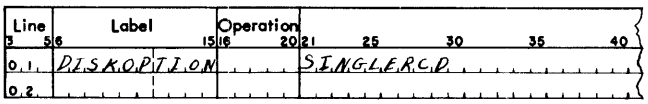



Figure 62.

The operand in Figure 62 indicates that the program is to write and/or read at least one single disk record.

NOTE 1: If the SHARED operand is used, the IOCS will provide the coding required to enable the sharing system to have access to disk storage and to prevent interrupts from operations issued by the sharing system.

At the completion of all disk operations and immediately prior to returning control to the main program, IOCS will execute a Prevent Seek Complete (PSC) operation to inhibit interrupts from the completion of seek operations and a Release (REL) operation to permit the sharing system to have access to the 1301 disk storage unit.

NOTE 2: The first system to issue a disk instruction or a command to the 1301 gains control of the disk storage unit and retains it until the execution of a Release instruction. A PSC (Prevent Seek Complete ) instruction should be executed first by any program on a computer using a shared 1301; otherwise, interrupts caused by seeks issued by the sharing system will have to be serviced.

NORCDEXIT

This DIOCS entry is needed only if the program calls for the use of single record mode of the disk operation. This entry enables the programmer to have control branched to his own routine in the event of a No-Record-Found Condition.

The use of the 'NORCDEXIT' entry will cause control to be branched to the routine specified by the operand of the entry each time a No-Record-Found Condition is encountered. The first instruction in the user's No-Record-Found Routine must be a Store B-Register (SBR) that stores the return address to the IOCS.

NOTE: Under no circumstances may the user's No-Record-Found Routine perform any I/O function, with or without the IOCS. In effect, the user's No-Record-Found Routine is operating within the structure of the IOCS and any I/O operation will destroy the IOCS control.

Before branching control to the user's routine, the 1301 IOCS places into Index Register 14 the high-order address of the B-field of the disk instruction that led to the No-Record-Found Condition (see 'DISKADR', Figure 38). Upon completion of the user's No-Record-Found Routine, the IOCS checks whether the user has changed the B-field, HA 1 or the channel specified in the Holding Area Control Record. If a change was made, the IOCS will attempt to locate the record in an alternate address developed by the user's No-Record-Found Routine. This is normal in random processing.

When no record can be produced from disk, the transaction may be ignored by the IOCS or retained by the IOCS for the user. Placing an $\check{S}$ in the address portion (2+X14) of the 'DISKADR' in the Holding Area Control Record while in the No-Record-Found Routine causes the IOCS to ignore the transaction and skip the remainder of the disk routine. In this case, the transaction is not retrievable. By placing a $\check{P}$ in the same position the transaction will be retained by the IOCS for the user. The use of the $\check{P}$ allows the user to test for the $\check{P}$ after the IOCS returns to GET +1 or PUT +1. For form 1, 2 or 3 records, the $\check{P}$ may be located by subtracting 7 from the index register associated with that file. Form 4 records require that 11 be subtracted from this index register because of the block character-count field, which is four positions long (see Figure 75). After return to GET +1 or PUT +1, the transaction may then be processed as the user desires. This means of communicating with the IOCS (by using the $\check{P}$ or $\check{S}$) is necessary because at the time the users no-record-found routine is being executed neither the address of the transaction that caused the no-record-found condition is known, nor may any I/O operation be performed to dispose of the transaction, even if the address were known.

If nothing is changed in the Holding Area Control Record after four attempts to locate the record on disk, the IOCS will withdraw the access mechanism completely, reposition it at track number 0000, reissue the SEEK command for the desired track and try again to execute the disk instruction. If the no-record-found condition then recurs, the IOCS will enter a waiting loop and an appropriate message will be typed on the console printer.

## THE 'DTF' ENTRIES

### Purpose

In addition to the DIOCS entries, the programmer who wishes to use the 1301 Input/Output Control System must write one set of DTF (Define The File) entries for each disk file used by his program. This information consists of up to 14 entries listed individually on the IBM Autocoder 1401/1410 Coding Sheet.

Each set of DTF entries describes the characteristics of the file for which it was written and indicates the methods to be used by the 1301 IOCS in handling the file. Using the information supplied in the DTF entries, the Autocoder processor develops the File Scheduler and the coding required for the proper handling of each file.

NOTE: DTF entries are not required for single-reference processing because files need not be defined for this mode of processing.

### General Format

The first DTF entry is the "DTF header line." It consists of the mnemonic code "DTF" in the operation field followed by the name of the file in the operand field. All subsequent DTF entries have blank operation fields and must have the labels listed below. All disk DTF entries may be followed by comments. These must be separated from the DTF entries by at least two adjacent blanks. The entries following the header line may be listed in any order.

All operands of disk DTF entries may use address modification provided that the operand consists of no more than 13 characters. Thus, 'LABEL + 110' is a valid operand if LABEL consists of no more than nine characters. All symbolic operands of DTF entries, except those of input/output areas, may be indexed. (The number of characters used to designate the index must be included in the count of 13.)

The sets of Disk DTF cards may be intermixed with the sets of the Card/Tape DTF cards and enter the system immediately after the DIOCS cards during Autocoder assembly. See Figure 12. Each Disk DTF entry is described below under a subheading indicating the label of the entry.

NOTE: DTF cards without operands are not permitted.

### List of DTF Entries

This section describes the function and use of each of the DTF entries listed below.

The following entries apply to both random and sequential processing:

DTF header line
FILETYPE
SIZEREC
HOLDAREA
INDEXREG
FILEFORM
BLOCKSIZE
DISKCHECK
WLRADDR

The following entries apply only to sequential processing:

RECFORM
SCRAMBLE
NRECORDS
PADDING
WORKAREA
FILESTART
FILEND
EOFADDR

## THE 'DTF' HEADER LINE

The first DTF entry is mandatory and consists of the mnemonic code DTF in the operation field, followed in the operand field by the name of the file defined by this DTF.

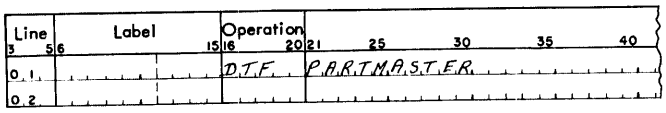



Figure 63.

The operand in Figure 63 indicates that the set of DTF entries following this header line defines a file called PARTMASTER.

### FILETYPE

The FILETYPE entry indicates that the file described by this DTF is a disk file and specifies whether it is a random or a sequential file, and whether it is used for input or for output operations.

The operands of the FILETYPE entry are:

DISK -- This entry is mandatory and indicates that the file described by the DTF is a disk file. (The DISK operand is needed because the "FILETYPE" DTF entry is also used by the Card/Tape IOCS.)

* Note that data from a disk input file may be returned to that file after updating.

INPUT -- if the file described by the DTF is used for input operations.*

OUTPUT -- if the file described by the DTF is used only for output operations.

SEQUENTIAL -- if the file described by the DTF is a sequential file.

RANDOM -- if the file described by the DTF is a random file.

NOTE 1: The operands INPUT and RANDOM may be listed but are not required.

The operands of the FILETYPE entry may be listed in any order and must be separated by commas.

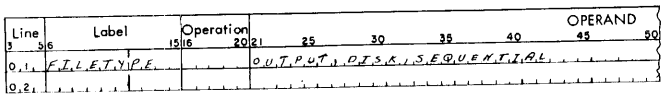



Figure 64.

The operands in Figure 64 indicate that the disk file described by this DTF is a sequential output file.

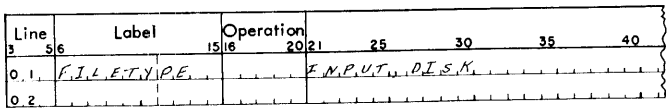



Figure 65.

The operands in Figure 65 indicate that the file described by this DTF is a random input file.

SIZEREC

The SIZEREC entry is mandatory. Its operand depends on record size as follows:

Variable-Length Records (Sequential Files Only)

The operand of the SIZEREC entry is:

n where "n" indicates that the low-order position of each record's character-count field is the "nth" character of each record. See the example below.

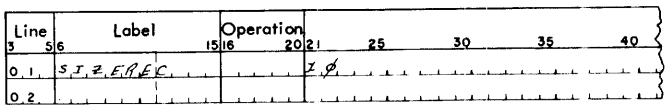



Figure 66.

The entry in Figure 66 indicates that the low-order position of the character-count field in each record of this disk file is the 10th position of the record. See also Figure 67.

Fixed-Length Records

The operand of the SIZEREC entry is:

m where "m" is the number of characters in the record, including the record mark. (Thus, the operand is "80" for eighty-character records.) See Figure 68.

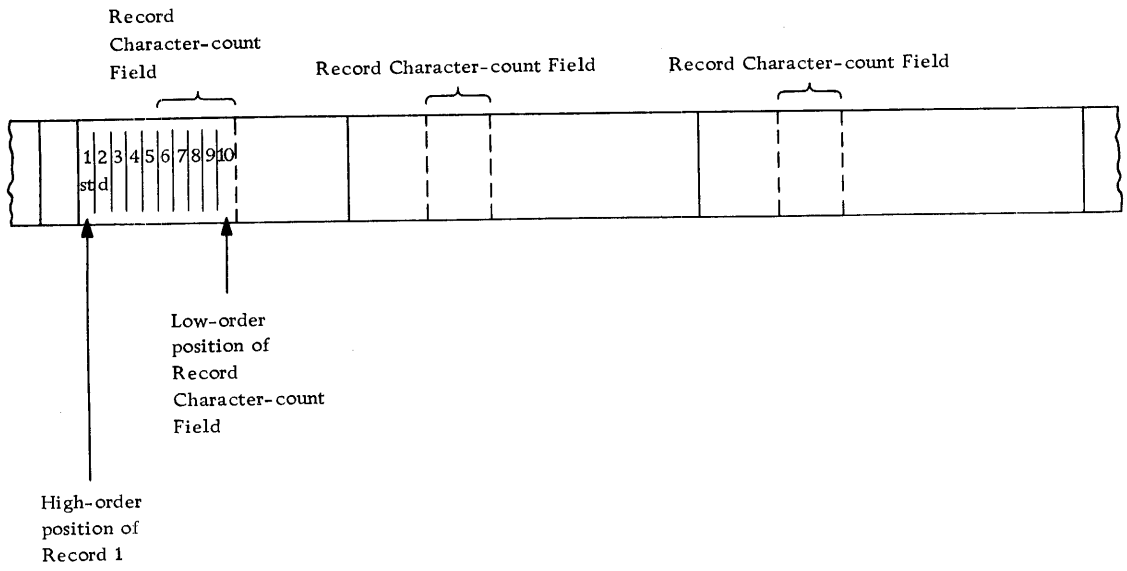



Figure 67. The Record Character-count Field

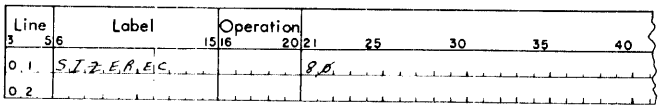



Figure 68 .

NOTE: In random processing and in single-reference mode, variable-length records are treated by IOCS as fixed length records. (See DTF 'RECFORM' entry.)

HOLDAREA

The operand of the HOLDAREA entry is the label of the DA (Define Area) statement that defines the Disk Record Holding Area assigned to the file. The location represented by this label is the high-order position of the Disk Record Holding Area.

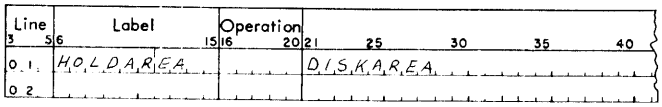



Figure 69.

The operand in Figure 69 indicates that the label of the Disk Record Holding Area of the file is DISKAREA. (See Figures 14 and 86.)

The holding area consists of n sets of two fields. The fields in each set consist of a 30-character Holding Area Control Field immediately followed by an input/output segment. (See the description of the GETS macro-instruction for a complete explanation of the Holding Area Control Field.) In both random and sequential processing applications, the user need only define the areas for these fields by means of DA statements. IOCS will place the necessary information in these fields as required.

The length of a Holding-Area segment is specified in the DTF 'BLOCKSIZE' entry, and depends on the record format and the mode in which the information is recorded. (See Figure 87.) If more than one segment is used, the IOCS will set up the word marks in all segments in the same manner as defined by the programmer in his DA entry for the first segment.

The size of the Holding Area is determined as shown in Figure 86 and explained in the accompanying description.

The DA statements required for a Holding Area which contains six segments for 120-character records are shown in figure 69A.

Number of Segments

The number of segments is determined as follows:

Sequential Processing. Either one or two segments may be used. The total output per unit of time is usually greater if two Holding-Area segments are used.

Random Processing. The number of segments depends on

(1) the number of arms available to service a file, and
(2) the number of transaction records which may be stacked and request disk records from the file.

For optimum arm scheduling, the number of segments should not be less than the number of arms available to the file.

Size of Segments

The size of each segment of the Disk Record Holding Area depends on the record format and the mode in which the information is to be read or written. See Figure 87. For a detailed description of the Disk Record Holding Area see Figure 86 and the description of the DTF 'BLOCKSIZE' entry.

INDEXREG

This entry is not needed for sequential files if an index register is not to be assigned to the files

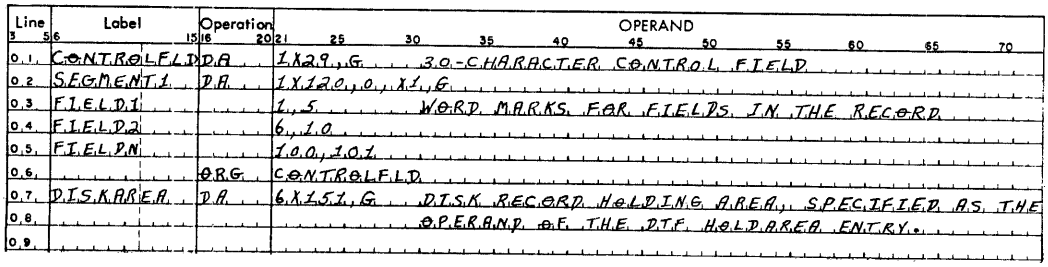



Figure 69A.

described by the DTF. The entry is mandatory for all random disk files.

The operand of the INDEXREG entry is:

X1, X2,..., X12,

indicating the index register assigned to the file.

NOTE 1: Index Registers 13, 14 and 15 may not be assigned to a disk file.

NOTE 2: The programmer must refer to disk data stored in the Disk Record Holding Area by means of the index register specified by the DTF "INDEXREG" entry for that file. This index register will contain the address of the high-order position of the disk record. Therefore, if the programmer designates the index register specified by the DTF "INDEXREG" entry in the DA entry for the Disk Record Holding Area, the fields in that area will be automatically addressed by the index register.

FILEFORM

This entry is mandatory.

The operands of the FILEFORM entry are:

MOVE -- if the file described by the DTF is read or written in the MOVE mode;

LOAD -- if the file described by the DTF is read or written in the LOAD mode;

SINGLE -- if the file described by the DTF consists of single records. NOTE: if this operand is used, the SCRAMBLE entry must be given;

CYLINDER -- if the file described by the DTF is contained on an entire cylinder. (The cylinder optional feature must be attached.)

TRACK -- if the file described by the DTF consists of full-track records.

NOADDRESS -- if full tracks without record address are to be read or written.

Move Mode vs. Load Mode

On magnetic tape, word marks are represented by word separator characters in disk storage. Word marks are represented by actual bits. Hence, whenever the programmer desires to place information into disk storage with word marks, each character in disk storage must be associated with an additional disk-storage bit for the word-mark bit. The character storage-capacity of a given track of disk storage, therefore, varies with the mode in which the information is read or written.

"MOVE" MODE. When written in the MOVE mode (i.e., word marks are not required), each character is represented by seven bits: six BCD-bits plus one space bit. Each track can hold up to 2,800 characters written in the MOVE (or "6-bit") mode.

"LOAD" MODE. When written in the LOAD mode (i.e., word marks are required) each character is represented by nine bits: six BCD-bits, one word-mark bit, and two space bits. Each track can hold up to 2,165 characters written in the LOAD (or "8-bit") mode. See Figures 70 and 71.

SCRAMBLE

This entry is used only for sequential files, with single-reference processing. The user must save the B-address register to effect the return to IOCS. IOCS will take the SCRAMBLE exit whenever it needs a new record address.

Before branching control to the user's routine specified in the operand of this macro-instruction, IOCS places into Index Register 14 the high-order position of the Holding Area Control Record (see Figure 38). The track address is located in the field labeled 'HA1.' (See the DTF 'NRECORDS' entry for a description of how sequential track addresses are updated.)

The record address developed by the routine must be placed in the B-field of the Holding Area Control Record.

DISKCHECK

This entry is only required if a Write Disk Check operation is to be performed. The operand of this DTF entry is YES.

RECFORM

This entry is needed only for sequential disk files containing other than fixed-length, unblocked records. *

The operands of the RECFORM are:

VARIABLE -- if the file described by the DTF consists of variable-length records, and

BLOCKED -- if the records described by the DTF are blocked.

The operands must be separated by a comma and may be listed in any order.

NOTE: The operands FIXED and UNBLOCKED, referring to fixed-length and unblocked records, respectively, may be used but are not required.

A description of the record formats that can be handled by the 1301 IOCS follows.

* Random input files are treated as fixed-length, blocked files; random output files are treated as fixed-length, unblocked files.

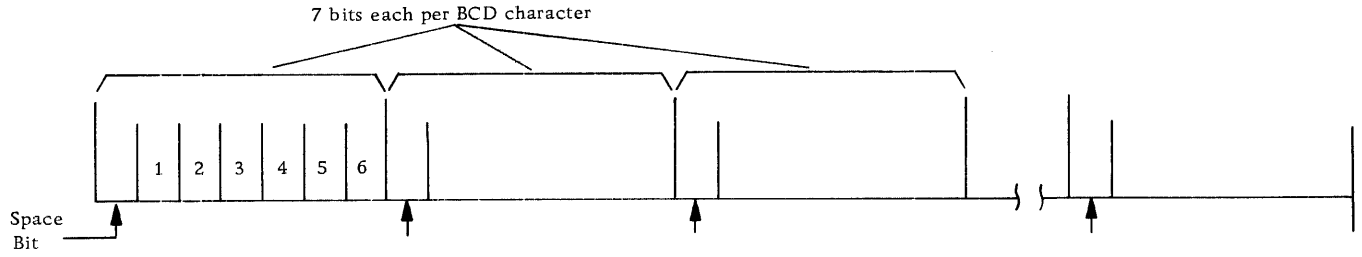



Figure 70. MOVE Mode: Seven Disk Bits per BCD Character, 2800 Characters per Track

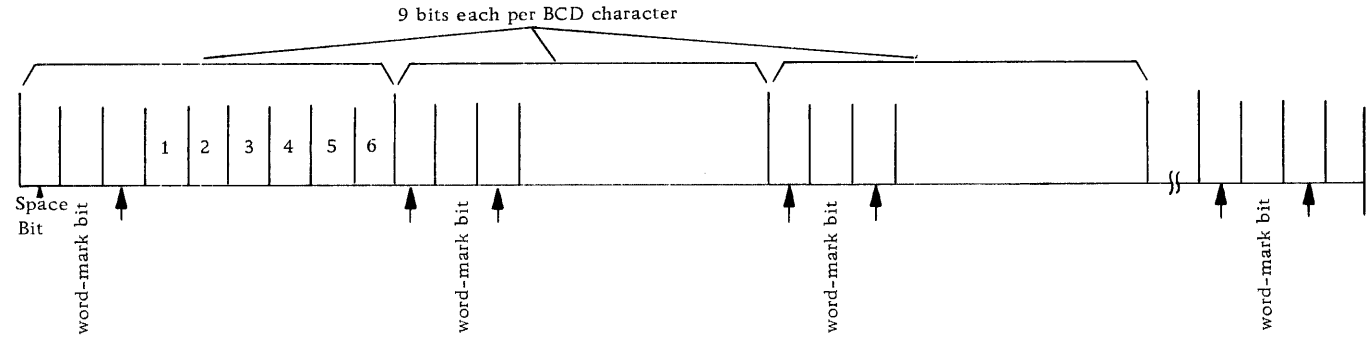



Figure 71. LOAD Mode: Nine Disk Bits per BCD Character, 2,165 Characters per Track

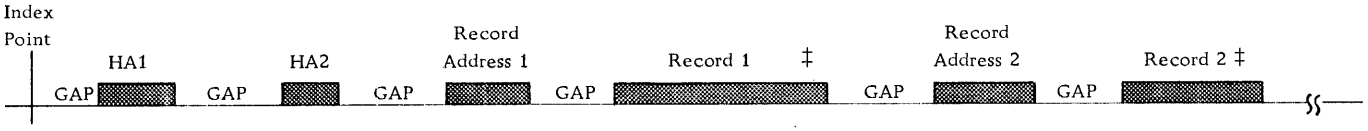



Figure 72. Disk Track with Form-1 Records, with Record Marks

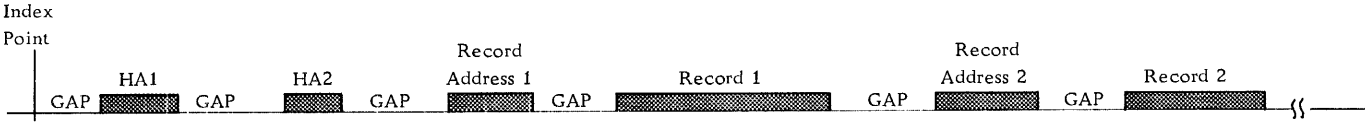



Figure 73. Disk Track with Form-1 Records, without Record Marks

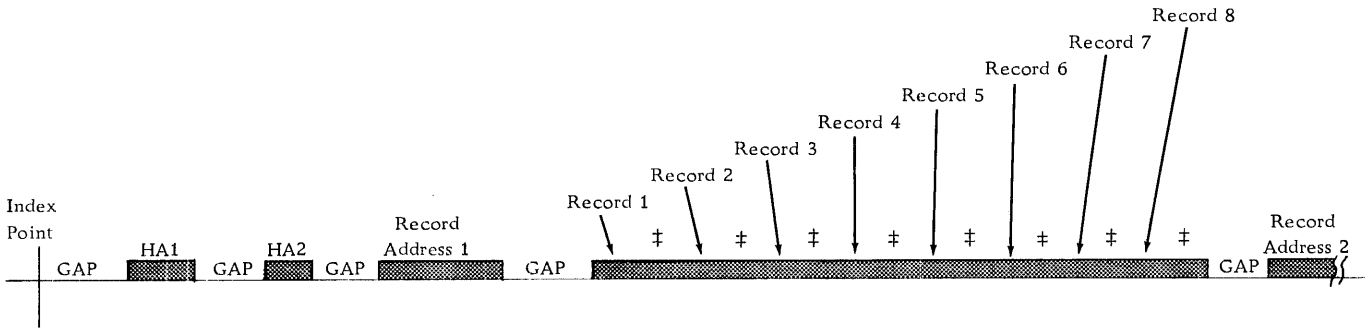



Figure 74. Disk Track with Form-2 Records

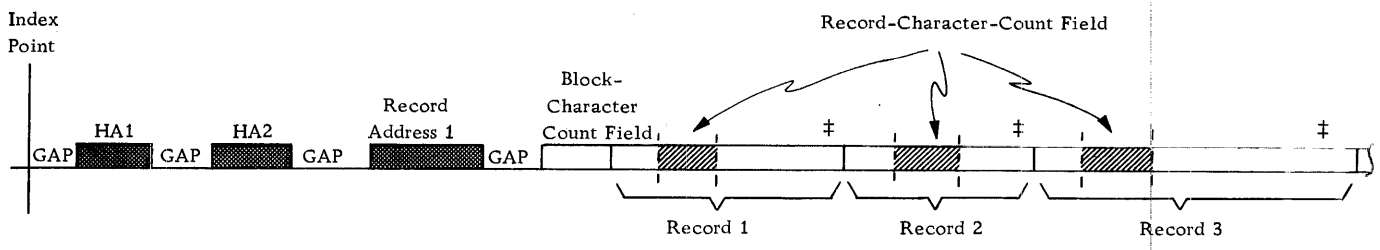



Figure 75. Disk Track with Form-4 Records
Each block has a variable number of variable-length records

| | Un-blocked | Blocked | BLOCK | | RECORD | | May use Record Marks | May Omit Record Marks | May use indexing registers | May use work Areas |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | Fixed-Length | Variable-Length | Fixed-Length | Variable-Length | | | | |
| Form 1 | X | | - | - | X | | Yes[1] | Yes | Needed only if 2 I/O areas are used. | |
| Form 2 | | X | X | | X | | Yes | No | Yes | Yes |
| Form 3 | X | | - | - | | X | Yes[1] | Yes | Needed only if 2 I/O areas are used. | |
| Form 4 | | X | | X | | X[2] | Yes | No | Yes | Yes |

[1]Record marks are required only if the output files are to be blocked.
[2]Record Character-count is contained in Record Character-count Field of each record.

Figure 76. Summary of Record Formats for Sequential Processing Using 1301 IOCS

## Record Formats That Can Be Handled by the 1301 IOCS

Form-1 Records. These are fixed-length, unblocked records -- with or without Record Marks. See Figures 72 and 73.

Form-2 Records. These are fixed-length, blocked records -- with Record Marks -- with padding of short-length blocks.* See Figure 74.

NOTE 1: Fixed-length, blocked records that are only partially filled are padded -- either with the character specified in the DTF "PADDING" entry or with blanks if the PADDING entry was omitted.

NOTE 2: Form-3 Records (i.e., variable-length, unblocked records) will be handled by the 1301 IOCS as fixed-length records because of format-track requirements.

Form-4 Records. These are variable-length, blocked records, with Record Marks and Block Character-Count Field, and Record Character-Count Field in each record. See Figure 75.

NOTE: In random processing and in single-reference operations, Form-2 and Form-4 records (blocked records) are logically treated as unblocked records by IOCS. The disk records are read or written as requested, but unblocking cannot be performed by IOCS.

BLOCK CHARACTER-COUNT FIELD. A four-character Block Character-Count Field at the beginning of each block contains a count of the total number of characters in the block including the four-character Block Character-Count Field, itself. The Block Character-Count Field has AB zone bits over the units position. The count is used for checking and correcting wrong-length-record conditions. See Figure 76.

* Blocked records must always have a record mark in the low-order position. Unblocked records may or may not contain record marks. However, unblocked records which are to be moved by "GET DISKFILE TO WORKAREA" or "PUT RNDMOUTFIL" macro-instructions must have either a record mark or a group mark with word mark immediately to the right of the low-order position.

RECORD CHARACTER-COUNT FIELD. A Record Character-Count Field of up to four characters in each record contains a count of the number of characters in that record, including itself and the record mark.

The principal characteristics of permissible record formats are summarized in Figure 76.

Examples of RECFORM entries are given in Figures 77 and 78.

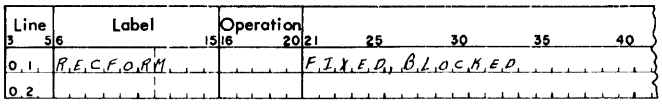



Figure 77.

The operand in Figure 77 indicates that the file described by the DTF contains Form-2 (i.e., fixed-length, blocked) Records. The same DTF statement could have been written as indicated below:

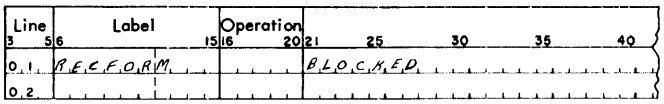



Figure 78.

The entry indicated in Figure 78 is equivalent to that shown in Figure 77 because the operand FIXED need not be written.

BLOCKSIZE

This entry is required. The operand specifies the number of positions occupied by each segment of the holding area reserved for an input or output disk record.

NOTE: The 30-character Holding Area Control Record and the terminating group mark with word mark are not included in this count. (These must be included in the DA entries for the Disk Record Holding Area.)

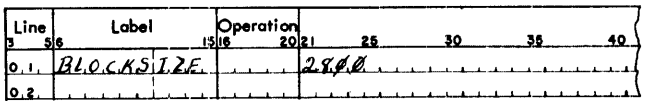



Figure 79.

The entry in Figure 79 indicates that disk-record segments of the holding area for the file defined by this DTF are each 2800 characters in length.

NRECORDS

This entry is required for all sequential disk files from or onto which single records are read or written.

The operand of the NRECORDS entry is

n where "n" is the number of records per track.

NOTE 1: The operand 'n' can be any number from 1 to 99. For every 'n' requests for a record, the track address is stepped by one. The user is provided with the track address in HA1 of the Holding Area Control Record when IOCS exits to the routine specified in the user's DTF 'SCRAMBLE' entry.

NOTE 2: Files of this type always require the DTF "SCRAMBLE" entry.

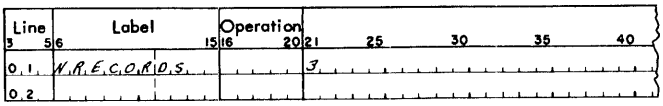



Figure 80.

The operand in Figure 80 indicates that each disk track of this file contains three records.

PADDING

This entry is needed only for sequential disk output files containing fixed length, blocked records.*

The operand of the PADDING entry is:

x where "x" is the character with which the block is to be padded.

The following characters may not be used for padding: asterisk, tape mark, word separator character, record mark, cent sign and group mark.

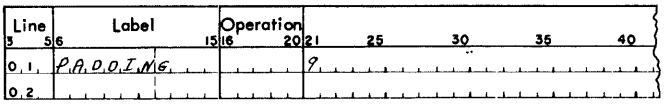



Figure 81.

The entry in Figure 81 indicates that partially filled blocks are to be padded with the digit 9.

* If the PADDING entry is omitted, partially filled blocks of this record type will be padded with blanks.

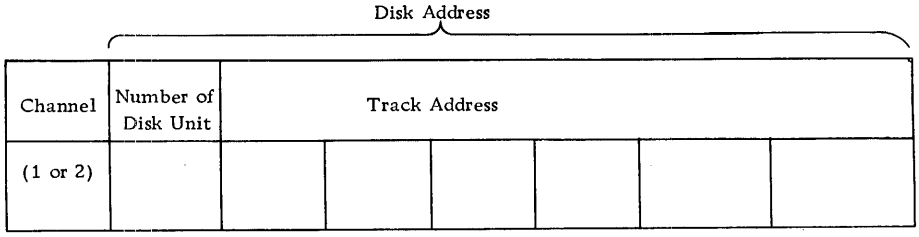



Figure 82. Operand of the DTF "FILESTART" and "FILEND" entries

WORKAREA

This entry is needed only for sequential files that use a work area and do not use the DTF "INDEXREG" entry.

The operand of the WORKAREA entry is the label of the work area used by the input or work file.

FILESTART

This entry is needed only for sequential files. The operand of the FILESTART entry is:

xxxxxxxx where "xxxxxxxx" are eight digits defining the first track of the (sequential) file, as indicated in Figure 82.

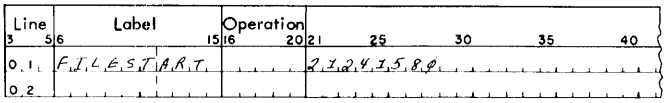



Figure 83.

The operand in Figure 83 indicates that the first track of the file defined by this DTF is track 2415 with HA2 of 80 in Disk Unit 1, which is attached to Channel 2 of the system.

FILEND

This entry is needed only for sequential files. The operand of the FILEND entry is:

xxxxxxxx where "xxxxxxxx" are eight digits defining the last track of the (sequential) file, as indicated in Figure 84.

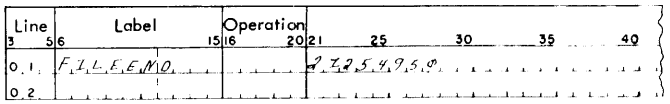



Figure 84.

The operand in Figure 84 indicates that the last track of the file defined by this DTF is track 2549 stored in Disk Unit 1, which is attached to Channel 2 of the system.

EOFADDR

This entry is required for sequential files only.

The operand of the EOFADDR entry is the label of the routine (written by the programmer) to which control is to be branched if the program develops an address exceeding that defined by the DTF "FILEND" entry.

WLRADDR

This DTF entry, when used for a disk file, eliminates the normal wrong-length record checking performed by the IOCS for that disk file. It is useful if the programmer is expecting a wrong-length record indication and wishes the IOCS to ignore it. The operand in the disk 'WLRADDR' DTF entry is 'NO', to indicate that no checking is to be done.

NOTE: The use of the 'WLRADDR' DTF entry for files assigned to disk storage is different from the use of this entry for any other type of device.

## DA (DEFINE AREA) ENTRIES NEEDED TO SUPPORT THE 1301 IOCS

All areas required by the 1301 IOCS (i.e., Transaction Stacking Areas, Disk Record Holding Areas, and Holding Area Control Records) must be reserved by the programmer by means of appropriate DA entries. (A general discussion of how DAs are written may be found in the IBM Data Processing Systems bulletin, "Basic Autocoder for the IBM 1410: Preliminary Specifications," Form J24-1413.) All such areas must be terminated by a group mark with word mark immediately to the right of the low-order position of the area.

The following describes how DA entries are written for the Transaction Stacking Area for Disk Record Holding Areas.

### DA ENTRY FOR THE TRANSACTION STACKING AREA

Each 1410 program written for random processing that is to utilize the IBM IOCS requires one Transaction Stacking Area. The Transaction Stacking Area permits the "stacking" of incoming transaction records for use by the disk routine(s).

NOTE: The segment of the Transaction Stacking Area assigned to a given transaction may also be used for storage during the processing of the transaction. Information so stored will be protected until the segment of the Transaction Stacking Area is released. Information stored by the disk routine in any other field in storage might be destroyed during the processing of another transaction record. See Figures 4 and 5.

The DA entry for a Transaction Stacking Area is of the form shown in Figure 85.

| Line | Label | Operation | |
|---|---|---|---|
| 3 5 | 6 15 | 16 20 | 21 25 30 35 40 |
| 0 1 | ANYLABEL | DA | N,M,G,,,INSTACKX |
| 0 2 | FIELD1 | | 9 |
| 0 3 | FIELD2 | | 99 |
| 0 4 | FIELD3 | | 99 |
| 0 5 | | | |

Figure 85 .

In Figure 85,

- N indicates the number of identical segments to be reserved;
- M indicates the number of positions to be reserved for each segment;
- G specifies that a group mark with word mark (needed by the 1301 IOCS) is to follow immediately to the right of the area reserved by this DA, and
- INSTACKX is the label (name) of the index register assigned to the Transaction Stacking Area. See the 1301 IOCS "STKINDEX" entry, and Figure 39.

NOTE 1: For applications requiring more than one disk routine, the number of segments N should be calculated by the following formula:

$$N = \frac{Op}{S}$$, where:

- Op = Operand of the DIOCS "RNDMDEPTH" entry, and
- S = Number of SEEKS per transaction that can be executed simultaneously.

NOTE 2: If variable-length transaction records are to be moved to the Transaction Stacking Area, each segment must be large enough to hold the maximum-size record.

NOTE 3: Each segment must contain a location for the record mark or the group mark with word mark that terminates the move operation and is moved together with the input information.

NOTE 4: The word marks specified for Fields 1, 2 and 3 are needed only if the programmer desires to move data to the Transaction Stacking Area by actual move commands.

NOTE 5: The label of the DA statement is used to describe the area in the DIOCS "STKAREA" entry.

## DA ENTRIES FOR DISK RECORD HOLDING AREAS

Each program utilizing the IBM 1301 IOCS requires one Disk Record Holding Area for each disk file used by the program. The Disk Record Holding Areas are used for the storing and processing of information obtained from disk storage. Each Disk Record Holding Area must be reserved by the programmer by an appropriate DA entry.

The size of each Disk Record Holding Area (i.e., the number of segments of the area and the size of each segment) depends on the record format and the type of application. The number of segments required has been described in the discussion of the DTF "HOLDAREA" entry.

Figure 86 indicates the general format of a Disk Record Holding Area. As indicated in Figure 86, each segment of the Disk Record Holding Area must be preceded by a 30-position area required by the 1301 IOCS. Each segment must be followed by a location for a group mark with word mark to be inserted by the 1301 IOCS. (See OPEN macro-instruction and Figure 14.) The entire Disk Record Holding Area must be followed by one location for a final group mark with word mark that must be specified by the programmer.

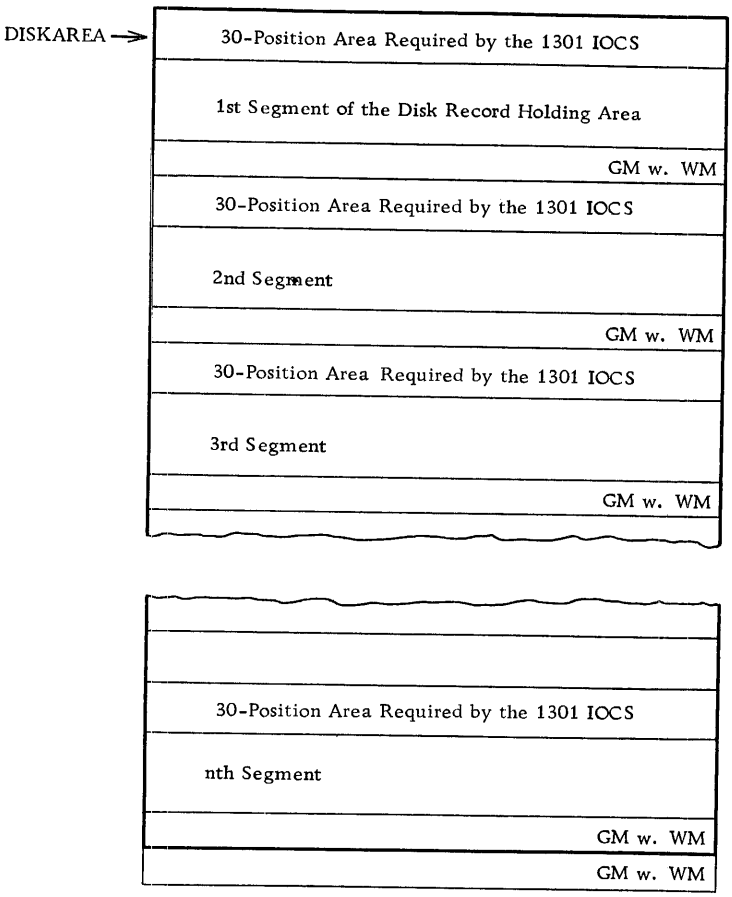



Figure 86. Format of Disk Record Holding Area

The size of each segment of the Disk Record Holding Area depends on the record format and the mode in which the information is to be read or written. See Figure 87.

Figure 88 shows an example of the type of DA entries that must be written for a sequential disk file containing three 65-character records per block that are read and written in the MOVE mode. See also section on "Move Mode vs. Load Mode."

NOTE 1: The group mark with word mark for each block and the word marks for the second and third blocks are provided by the OPEN macro-instruction. See description of the OPEN macro-instruction, including Figure 14.

NOTE 2: All random files must be indexed (relative to zero) with the indexing register specified by the DTF "INDEXREG" entry.

NOTE 3: Information in sequential files may be either indexed or moved. See description of the GET macro-instructions.

## DA ENTRIES FOR HOLDING AREA CONTROL RECORDS

See description of the GETS macro-instruction.

| Record Format and Mode | Total number of positions which must be reserved by that programmer for each segment. | Use of the Reserved Area | | |
|---|---|---|---|---|
| | | For Disk Data | For Disk IOCS Control Information | For GM w. WM |
| Full TRACK, LOAD | 2195 | 2165 | 29 | 1 |
| Full TRACK, MOVE | 2830 | 2800 | 29 | 1 |

Figure 87. SIZE and USE of the segments of Disk Record Holding Areas

| Line | Label | Operation | OPERAND |
|---|---|---|---|
| 0.1 | FILEAREA | DA | 1,1,2,9,,G DEFINE CONTROL AREA |
| 0.2 | | DA | 3,1,6,5,,,4,, TMDEIWORD |
| 0.3 | FIELD1 | | 1,,3 AREA NEEDED |
| 0.4 | FIELD2 | | FOR TIME |
| 0.5 | FIELD3 | | FIRST BLOCK |
| 0.6 | | ORG | FILEAREA |
| 0.7 | | DA | 2,1,2,2,5,,G PAD FIRST BLOCK, PROVIDE SECOND BLOCK AND |
| 0.8 | | | 6 FINAL GM/WM |
| 0.9 | | | |

Figure 88.

NOTE 1 The group mark with word mark for each block & the word marks for all blocks after the 1st are provided by the OPEN macro instruction. However, the user must take care to include the count for those group marks with word marks in the final DA statement which reserves the total area required for the file. See description of the OPEN macro instruction, including Figure 14.

## THE SIZE OF THE 1301 IOCS ROUTINES

The approximate number of storage locations required by the 1301 IOCS Routines are as follows:

| | |
|---|---|
| Single Reference Routine: | 2,500 locations |
| Random-Processing Routine: | 3,000 locations |
| Sequential-Processing Routine: | 1,000 locations |
| Error Routines: | 3,500 locations |

See Figure 11.

## USE OF INDEX REGISTERS

Index Registers X13-X15 are used by the 1301 IOCS, and their contents may change each time a macro-instruction is encountered. The programmer may use these index registers, but he should consider the effect of the above on his program.

**IBM** Form X24-1350-1 Printed in U.S.A.

Program ____ Programmed by ____ Date ____

INTERNATIONAL BUSINESS MACHINES CORPORATION
**IBM 1401 AND 1410 DATA PROCESSING SYSTEMS**
AUTOCODER CODING SHEET

Identification SMPLE 76 80 Page No. 01 of 04

| Line | Label | Operation | OPERAND |
|---|---|---|---|
| 0.1 | *CODING EXAMPLE. | 1410 IOCS FOR 1301 DISK STORAGE. | |
| 0.2 | | IOCS | |
| 0.3 | FEATURES | | OVERLAP,PRIORITY REQUIRED |
| 0.4 | CHAN1 | | READER,TAPE RDR,TP ON CH1 |
| 0.5 | CHAN2 | | 1301 DISK ON CH2 |
| 0.6 | PROCESTYPE | | RNDM RNDM PROCESSING |
| 0.7 | RNDMDEPTH | | 6 SGMNTS IN HOLDAREA |
| 0.8 | STKAREA | | TRANSACTN TRANSACTIONS STACKED HERE |
| 0.9 | STKINDEX | | X1 INDEX REG FOR TRANS RCDS |
| 1.0 | SGMTLENGTH | | 81 CARD RCDS ARE 80, 1 FOR # |
| 1.1 | DISKARMS | | 3 3 MODULES |
| 1.2 | | DTF | READER |
| 1.3 | FILETYPE | | READER TRANSACTION INPUT FROM CARDS |
| 1.4 | CARDPOC | | 1 SELECT READ POCKET 1 |
| 1.5 | IOAREAS | | INPUT TRANSACTION READ HERE |
| 1.6 | EOFADDR | | END EOF ROUTINE |
| 1.7 | | DTF | MASTERFILE |
| 1.8 | FILETYPE | | DISK,RANDOM,INPUT DISK FILE |
| 1.9 | SIZEREC | | 120 RCD LENGTH |
| 2.0 | FILEFORM | | MOVE,SINGLE SINGLE RCD, MOVE MODE |
| 2.1 | HOLDAREA | | DISKINPUT HOLDING AREA |
| 2.2 | INDEXREG | | X2 INDEX REG FOR DISK RCDS |
| 2.3 | | DTF | REPORT |
| 2.4 | FILETYPE | | TAPE,OUTPUT UPDATED RCDS FOR OFF-LINE 1401 |
| 2.5 | CHANTYPE | | 10 PRINTING |
| 2.6 | IOAREAS | | RPTREC1 1 I/O AREA |

**IBM** Form X24-1350-1 Printed in U.S.A.

Program ____ Programmed by ____ Date ____

INTERNATIONAL BUSINESS MACHINES CORPORATION
**IBM 1401 AND 1410 DATA PROCESSING SYSTEMS**
AUTOCODER CODING SHEET

Identification SMPLE 76 80 Page No. 02 of 04

| Line | Label | Operation | OPERAND |
|---|---|---|---|
| 0.1 | *I/O AREAS | | |
| 0.2 | TRANSACTN | DA | 6X81,0,X1,G TRANSACTION STACKING AREA |
| 0.3 | LOCATION | | 1,4 EQUALS TRACK ADDRESS |
| 0.4 | PARTNO | | 5,12 CHANNEL(2),MODULE,6 RECORD ADDRESS |
| 0.5 | TYPE | | 13 |
| 0.6 | ACTIVITY | | 14,20 |
| 0.7 | INPUT | DA | 1X80,G TRANS. CARD INPUT |
| 0.8 | DISKINPUT | DA | 1X29,G HOLDING AREA CONTROL RECORD |
| 0.9 | | DA | 1X120,0,X2,G LENGTH OF SEGMENT WITH # |
| 1.0 | BALANCE | | 1,7 FIELD DEFINITIONS. THESE WORD-MARKS |
| 1.1 | DESCRIPTN | | 10,50 WILL BE PLACED IN EACH SEGMENT OF THE |
| 1.2 | UNITCOST | | 51,56 HOLDING AREA. THE 30-CHAR CTL RCD |
| 1.3 | LOCATION | | 61,80 WILL BE SET UP PRECEDING EACH |
| 1.4 | | ORG | DISKINPUT SEGMENT. |
| 1.5 | | DA | 6X151,0,X2,G DEFINITION OF HOLDING AREA |
| 1.6 | RPTREC1 | DA | 1X120,G REPORT RECORD OUTPUT AREA |
| 1.7 | MSG1 | DC | C'THAT'S ALL@ |
| 1.8 | | DCW | @#@ |

Figure 89.

IBM

Program

Programmed by

Date

INTERNATIONAL BUSINESS MACHINES CORPORATION
IBM 1401 AND 1410 DATA PROCESSING SYSTEMS
AUTOCODER CODING SHEET

Form X24-1350-1
Printed in U.S.A.

Identification SMPLE (76-80)
Page No. 03 (1-2) of 04

| Line (3-5) | Label (6-15) | Operation (16-20) | OPERAND (21-70) |
|---|---|---|---|
| 0.1 | *SAMPLE PROGRAM. | READ A CARD., GET RANDOM DISK RECORD AND ADD | |
| 0.2 | * ACTIVITY | FIELD, PUT DISK RECORD BACK AND PUT DISK RECORD | |
| 0.3 | * ON TAPE FOR OFF-LINE 1401 REPORT. | | |
| 0.4 | | | |
| 0.5 | START | OPEN | READER,MASTERFILE,REPORT OPEN ALL FILES |
| 0.6 | *MAIN ROUTINE. | | |
| 0.7 | READCARD | GET | READER READ TRANSACTION RECORD |
| 0.8 | | MVRSA | INPUT MOVE RCD TO STACKING AREA |
| 0.9 | | B | DISKROUTIN TO DISK ROUTINE |
| 1.0 | | B | READCARD RETURN FROM DR,, GET NXT TRANS. |
| 1.1 | *END OF FILE PROCEDURE. | | |
| 1.2 | END | CLOSE | READER,MASTERFILE,REPORT CLOSE ALL FILES |
| 1.3 | | CONSL | WCP,MSG1 |
| 1.4 | | H | *-5 DEAD END |
| 1.5 | | DCW | @+@ WORD MARK AFTER LAST INSTR. |

IBM

Program

Programmed by

Date

INTERNATIONAL BUSINESS MACHINES CORPORATION
IBM 1401 AND 1410 DATA PROCESSING SYSTEMS
AUTOCODER CODING SHEET

Form X24-1350-1
Printed in U.S.A.

Identification SMPLE (76-80)
Page No. 04 (1-2) of 04

| Line (3-5) | Label (6-15) | Operation (16-20) | OPERAND (21-70) |
|---|---|---|---|
| 0.1 | *DISK ROUTINE | - RANDOM PROCESSING | |
| 0.2 | DISKROUTIN ENTDR | START OF DISK ROUTINE | |
| 0.3 | | MLCB | PARTNO,IOCSDSKAD SET UP DSKAD AND CHAIN TRACK ADR |
| 0.4 | | MLCB | TO IOCSSEKAD |
| 0.5 | | GET | MASTERFILE |
| 0.6 | *PROCESS | RECORD | |
| 0.7 | | A | ACTIVITY,BALANCE UPDATE DISK RECORD |
| 0.8 | | MRCM | 0+X2,RPTREC,1, MOVE RCD TO REPORT |
| 0.9 | | PUT | MASTERFILE PUT DISK RCD BACK |
| 1.0 | | PUT | REPORT WRITE REPORT |
| 1.1 | | LEVDR | RETURN TO MAIN LINE |

Figure 90.

## CODING EXAMPLE

The following coding example illustrates the incorporation of the 1301 IOCS into a 1410 program. It illustrates a parts-inventory application. The inventory file is updated on the basis of transaction records showing receipts and disbursements. The program also calls for the writing of a report showing the warehouse location and retail price of each part listed under disbursements. See Figures 89 and 90.

This type of application has been described above under "Random Processing."

## MODIFICATION OF 'SEEK-ONLY' OPERATIONS WHILE SEEK IS IN PROGRESS

The programmer can modify an initiated SEEK operation by taking three steps before the completion of the SEEK. He must:

1. mask priority interrupts by means of the Branch to Exit Priority Alert (mnemonic operation code BXPA) instruction;
2. check the specified switch for completion of the SEEK operation (See formats C and D of the GETS macro-instruction), and
3. change bit position 2 from ON to OFF. (ON is indicated by the presence of a bit; OFF is indicated by the absence of a bit.)

This will cause completion of the SEEK operation to be followed by the operation specified by the D-modifier.

This technique enables the programmer to initiate the SEEK operation and while the SEEK is in progress determine by processing whether the SEEK is to be followed by a READ or WRITE operation.

In each case, the turning OFF of bit position 2 will involve a change of the character entered in CODE (position 6) of the Holding Area Control Record. See Figure 91.

Let it be assumed, for example, that the programmer placed a W into DMODIFIER, and a K into CODE. In this case the presence of the 2 bit (BCD representation for K = B2) indicates that only a SEEK operation is to be performed. (See explanation of the CODE entry in the Holding Area Control Record.)

If the programmer determines by processing (while the SEEK is in progress) that the SEEK operation should be followed by a disk READ operation with wrong-length-record checking by IOCS, he need only change the contents of CODE from K to - (minus sign). The elimination of the 2-bit in

CODE now causes the SEEK to be followed by the operation specified by the contents of the DMODIFIER entry (in this case a READ operation) followed by a wrong-length-record check provided by IOCS because of the presence of the B-bit. (The BCD representation of the minus sign is a B-bit.)

## IOCS LABELS THAT MAY BE USEFUL

IOCSxxBFLD - A seven-position field containing the module number and six-position disk address for a sequential track file. It may be altered to control changes in the "sequential" order.

IOCSxxDKAD - A four-position field containing a track address for single-record files.

IOCSxxINCR and IOCSxxCONS - These two fields are used to insure accessing the first record of a new track address in single-record files the maximum number of times after a change to 'IOCSxxDKAD'.

A ZA IOCSxxINCR, IOCSxxCONS instruction must be given when accessing records from other than the next sequential track. The maximum number of times data may be accessed from a track is equal to the number of data records on that track, as defined in the 'NRECORDS' DTF entry.

'IOCSxxSTRT' - An eight-position field that contains the information supplied to the IOCS by the user's 'FILESTART' DTF entry. The field may be modified at any time; however, the change will not be in effect until an 'OPEN' macro-instruction has been executed for that file.

'IOCSxxEND' - An eight-position field that contains the information supplied to the IOCS by means of the user's 'FILEND' DTF entry. It may be modified at anytime; however, it must be greater than the current address as given in 'IOCSxxBFLD' or 'IOCSxxDKAD'.

The 'PREFIX' macro-instruction may be used to define xx in the above labels.

## GLOSSARY

The following Glossary is restricted to basic terms used or introduced in this bulletin.

Disk Record Holding Area. A work area, used by the 1301 IOCS, in which records obtained from disk storage are temporarily retained for processing and/or subsequent return to disk storage.

Disk Routine. A series of object-program instructions needed to obtain or process disk-storage data.

Holding Area Control Record. A body of control information defining the specific operation to be performed by a GETS or PUTS macro-instruction. This information must be entered into core storage -- in an area reserved by the programmer for this purpose -- before the associated macro-instruction is encountered by the program.

Random File. A file of information contained on disk storage for use in a RANDOM PROCESSING application.

Random Processing. Processing of disk records of uniform format and belonging to specific files in any (arbitrary) order of addresses.

Sequential Processing. Processing of disk records in the order of ascending addresses.

Single-Reference Processing. Processing of disk records of any format (and located anywhere in disk storage) in any (arbitrary) order of addresses.

Transaction Stacking Area. A work area used by the 1301 IOCS to store transaction records for subsequent processing and updating by the Disk Routine(s).

| Disk Operation to be performed | CODE-Field Entry |
|---|---|
| SEEK only | 2 |
| WRITE Disk Check | 4 |
| Wrong-Length-Record Check | - |
| SEEK + Write Disk Check | 6 |
| SEEK + Length Check | K |
| Write Disk Check + Length Check | M |
| SEEK + Write Disk Check + Length Check | O |

Figure 91. BCD Characters entered in CODE field of Holding Area Control Record

# INDEX

Printed in U. S. A. C28-0251-0

IBM®

International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, New York