File Number GENL-34
Form C24-3185-2

IBM Systems Reference Library

Disk File Organization Routines Specifications IBM 1401, 1440, 1460 (1311 and 1301)

This publication contains the specifications of two separate but similar packages of disk-file organization routines. The two packages and the system configurations on which they are used are:

IBM 1401-1460 (1311 and 1301) Disk-File Organization Routines
IBM 1440 (1311 and 1301) Disk-File Organization Routines

The following subjects are discussed in this publication:

- An introductory section on general considerations of disk-file organization.
- Descriptions of each of the 13 individual object programs.
- Allowable sizes and formats of input-output files.
- Program exits and labels useful to user subroutines.
- The parameter cards used to generate the object programs and the RDLIN cards used for area definition at object time.
- Suggested retrieval routines (block diagrams).
- A complete example of a random file and the programs generated to load and maintain it.

Minor Revision November, 1964

This edition, C24-3185-2 is a reprint of the previous edition, C24-3185-1, and it includes the changes made in Technical Newsletter N24-0252.

Copies of this and other IBM publications can be obtained through IBM Branch Offices.
Address comments concerning the content of this publication to IBM Product Publications, Endicott, New York 13764.

Contents

Preface

This publication describes two separate but similar packages of disk-file organization routines. Each package consists of a prephase and a set of library routines from which two complete sets of object programs can be generated. One of the two sets of object programs is designed to load and maintain a *random* file using the *chaining* method, whereby a record is loaded into a disk location determined by the control data of the record. The other set of object programs loads and maintains a file of records in *sequential* order. In this case, a sequenced file is loaded into consecutive disk locations. A sequence link field is used to accommodate additions to and deletions from the file. Also, an index is built to facilitate random retrieval of records in the sequential file.

These file-organization routines are supplied by IBM in Autocoder library format. From these library routines, the macro generator portion of Autocoder generates object programs to be used in organizing and maintaining disk files. The object programs are generated in accordance with a series of parameter cards punched by the user and are tailored, by these parameters, to the user's particular application.

Disk File Organization Routines Specifications IBM 1401, 1440, 1460 (1311 and 1301)

Machine Requirements

IBM 1401-1460 Disk File Organization Routines

To Generate Object Programs. These routines are entered in the library portion of the IBM 1401-1440-1460 Disk Autocoder. This Autocoder can be operated from either 1311 or 1301 disk storage. The machine requirements to generate file-organization object programs are the same as for any 1401 or 1460 disk Autocoder run. See the publication *Autocoder (on Disk) Program: Specifications and Operating Procedures,* C24-3259.

To Run Object Programs.

IBM *1401 Processing Unit* with 4,000 core-storage positions or,

IBM *1441 Processing Unit* with 8,000 core-storage positions.

The high-low-equal compare feature is required. Additional core storage can be utilized for longer record and block lengths. The advanced-programming (1401) or indexing-and-store-address-register (1460) feature is not required but, if available, is utilized for faster, more efficient programming.

IBM *1402 Card Read Punch.*

IBM *1403 Printer, Model 1, 2, or 3.* If the console printer is available, the 1403 is used only for the optional audit trail printouts.

IBM *1407 Console Inquiry Station* or,

IBM *1447 Console* (without the buffer feature). The console printer is not required. If available, however, it is used to print all status and error messages, leaving the IBM 1403 Printer for only the (optional) audit trail printouts. A 1447 with the buffer feature cannot be used.

IBM *1301 Disk Storage, Model 11, 12, 21, or 22,* or

IBM *1311 Disk Storage Drive, Model 1 or 2.* These programs can be generated in either 1311 or 1301 disk storage for systems with either or both 1311 or 1301 disk storage. The 1311 direct-seek feature is not required but is utilized if available.

IBM *729 or 7330 Magnetic Tape Unit.* Required only if tape input or output is requested.

IBM 1440 Disk File Organization Routines

To Generate Object Programs. These routines are entered in the library portion of the IBM 1401-1440-1460 Disk Autocoder. This Autocoder can be operated from either 1311 or 1301 disk storage. The machine requirements to generate file-organization object programs are the same as for any 1440 disk Autocoder run. See the publication *Autocoder (on disk) Program Specifications and Operating Procedures,* C24-3259.

To Run Object Programs.

IBM *1441 Processing Unit.* When operating on IBM 1311 Disk Storage Drives, 4,000 core-storage positions are required. When operating on IBM 1301 Disk Storage, 8,000 positions are required. Additional core storage can be utilized for longer record and block lengths. The indexing-and-store-address-register feature is not required but is utilized if available.

IBM *Card Read Punch,* Model 1 or 2, or

IBM *1442 Card Reader,* Model 4. The selective-stacker feature (Model 1 only) is not required but can be used to separate-out mispunched cards when punching.

IBM *1444 Card Punch,* Model 1. Required only if punched output is requested on a system with only a Model 4 of the 1442.

IBM *1443 Printer,* Model 1 or 2. If the console printer is available, the 1443 is used only for the optional audit trail printouts.

IBM *1447 Console* (without the buffer feature). The console printer is not required. If available, however, it is used to print all status and error messages, leaving the IBM 1443 Printer for only the (optional) audit trail printouts. A 1447 with the buffer feature cannot be used.

IBM *1301 Disk Storage,* Model 11, 12, 21, or 22, or

IBM *1311 Disk Storage Drive,* Model 1 or 2. These programs can be generated in either 1311 or 1301 disk storage for systems with either or both 1311 or 1301 disk storage. The 1311 direct-seek feature is not required but is utilized if available.

IBM *7335 Magnetic Tape Unit.* Required only if tape input or output is requested.

General Considerations of Disk-File Organization

There are many advantages to random-access storage in a data processing system. The IBM 1311 Disk Storage Drive and the IBM 1301 Disk Storage have greatly increased these advantages. The flexibility of these units allows several kinds of processing within the same system. The advantages offered by one kind of processing are often quite different from those offered by another. For example, in a random-processing application, the advantages center around the ability to process only those records that have current transactions, rather than to process all of a file. In a second case, the advantage may be the ability to process one or more sequential files against a single sequenced input.

File organization is the key to effective use of disk storage. The objective of disk-file organization is the systematic storing of information in disk storage in such a manner that records can be retrieved in the quickest way possible, while still maintaining the over-all processing objectives of the system.

The method of organization best suited to a particular file of disk records depends upon many factors. These factors must be analyzed for each file in any one particular application. Often, more than one organization scheme can be used on the same file. In one application, records could be processed purely at random; in another, the same records could be processed in sequence by various control fields. For example, records within a file might be processed at *random* during an updating run and *sequentially* within certain groups, such as branch office or due date, when producing reports or billing. A file such as this would be analyzed to determine whether it should be organized:

1. Purely randomly, thus keeping process time at a minimum during one run but destroying the advantage of the sequential nature of the other.
2. Sequentially, thus minimizing the time required to produce reports but increasing updating time.
3. Randomly for updating and then sorted into sequence for reports.

The decision would depend on the nature of the file. Questions such as the following might be asked:

1. Can transactions be batched and sorted before processing, or must they be processed as they occur?
2. Is the activity distributed throughout the file in such a manner as to warrant passing the entire file when updating?
3. Would the processing time saved by sorting warrant the time and effort required?

Questions of this kind must be asked of each file in an installation. In choosing organization methods, the over-all processing objectives of the system must be kept in mind at all times.

Random Files

Direct Addressing

The simplest method of file organization is that in which a unique disk address is obtained from the control data of each record. This is referred to as the *direct addressing* method. Consider this case as an example:

A particular file consists of 10,000 records, each 100 characters long. No records are ever added to or deleted from the file. Each record has a unique 6-digit control field ranging from 000000 to 009999, inclusive. Further, this file is always processed in one of two ways:

1. Either purely randomly, with no predeterminable sequence, or
2. Purely sequentially by the 6-digit control fields.

In this hypothetical case there is one logical method of disk-storage organization. Record 000000 is stored in sector 000000; record 000001 in sector 000001; and so on, to record 009999 which is stored in sector 009999.

No elaborate file-organization program could possibly improve the organization of this file. When processing randomly, any record can be found with a single seek. When it is possible to process sequentially, only one seek is needed per cylinder. Processing time is thus at an absolute minimum. Also, the file area is packed 100%. No disk-storage space is lost. However, this was given as the optimum case, and it rarely occurs in actual practice.

In practice, the control data (item number, account number, etc.) of a file of records can seldom be used directly as disk addresses. Even when in the proper form, they do not always fall within the range of addresses desired. However, it may still be possible to use direct addressing. For example, suppose the records in the previously described file were 200 characters long. Thus each record would require two sectors. Unique disk addresses could be assigned to these records by multiplying the control data of each record by 2 and then adding a constant to arrive at addresses within the desired portion of the disk pack.

If these records were 25 characters long, and it were desired to use single-sector blocks with four records per block, each control field could be divided by four. The remainder of the division could be used to indicate the individual record position within the block.

If the file has no control fields that can be used directly as disk addresses, it is sometimes possible to pre-assign addresses. For example, the item number 4709GPX could become 4709GPX-023456, where 023-456 is the disk address of the record.

Indirect Addressing

Where none of the preceding methods are possible, it may be desirable to use a *conversion routine* to operate on the existing control data to produce disk addresses within the desired range. If the conversion routine used on a file produces a unique disk address for each record, direct addressing can be used. However, in most cases, using a conversion routine results in assigning some disk addresses to more than one record. Such duplicate disk addresses are called *synonyms.*

There are several ways of organizing files with synonyms. The method that has been determined most efficient in the majority of cases is that called the *chaining* method. This is the method used by the random disk-file organization routines described in this bulletin.

The objective sought by a conversion routine is to convert the control data of the records in a file to a randomly distributed series of disk addresses within a desired range, and to do this with a minimum number of synonyms. It is not possible to specify any one method of making this conversion. Each situation must be studied individually to determine the best method.

Chaining

The file organization technique used with a file employing indirect addressing must be able to accommodate the synonyms or duplicate addresses produced. The term *chaining* is applied to a technique that has been found to be most efficient in a majority of cases. Chaining is particularly suited for use in 1311 and 1301 disk storage because successive links of a chain of synonyms can be read without reseeking.

As each record is read in to be loaded in disk storage, its control data is converted to a disk address. These converted addresses are called *home addresses.* Because only one record can be stored in each location, the first synonym (home record) is placed in the original address developed (home address). The additional record or records (non-homes) are stored in *overflow* locations. The address of the first overflow location is stored in the home-address location. The address of the second overflow location is stored in the first

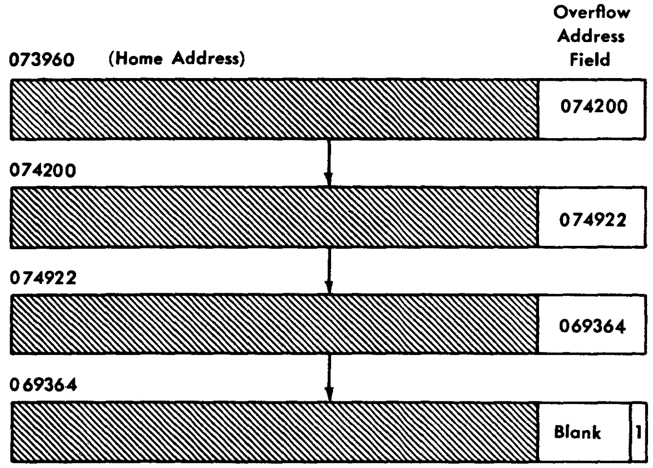


Figure 1. Disk-Storage Chaining

overflow location, etc. Chaining requires that, in the home address and all overflow locations, space be reserved for the address of the next location or link in the chain (Figure 1).

A chained file is normally loaded in two passes. During the first pass, only those records that can be placed in home locations are loaded. The second pass places all of the remaining records in overflow locations. Each overflow record is placed in an available location as close as possible to the previous link in the chain.

Retrieval of a record is accomplished by converting the control data to the home address. The record in the home-address location is read into core storage, and its control data is compared to that of the record being sought. If the control fields are not equal, the address of the first overflow record is extracted from the home record and another read command is issued using this address. The process is repeated until the desired record is found.

The time required to retrieve a particular record depends upon its position in a chain. All home records can be retrieved with a single seek and read. Subsequent links of a chain may require additional reads and, possibly, additional seeks.

File Packing

The average number of reads required to retrieve a record from a chained file depends upon the number of synonyms developed by the conversion routine. The number of synonyms produced by a randomizing conversion routine can be reduced by assigning more disk-storage space than actually required by the file. The percentage of the file area actually used for records is called the *packing factor*. For example, 11,000 disk locations might be used to store 10,000 records. Thus, the file would be said to be packed approximately 90%.

With a chained file, the average time required to retrieve records diminishes as the packing factor decreases. However, it is impossible to state any given packing factor as the best for all chained files. The packing of an efficiently organized file can vary from 65% to 95%. The time required to retrieve a record is not the only factor by which the efficiency of a file is measured. For example, a file might be packed close to 100% if by so doing the entire file can be kept within one disk pack. On the other hand, a file might be packed very loosely if it is desired to keep only one file or one group of related files on a single pack. Another point to consider is anticipated growth. Room must be left in a file area to accommodate records added at a later date.

Blocking Records in a Chained File

Records can be blocked easily in 1311 and 1301 disk storage. Because a single 100-character sector is the smallest unit of storage that can be read or written, the *block length* must be some multiple of 100 characters. The block length is limited only by the amount of core storage available for reading and writing the disk.

One reason for blocking records in a chained file is to save storage space. For example, four 125-character disk records can be stored in one 5-sector block with no wasted space. Another reason is that the average number of reads required to locate a record can usually be reduced by increasing the blocking factor (number of records per block). The greater the blocking factor, the greater the chance that overflow records will be in the same block as the home record. When they are, no additional read is required. It should be remembered, however, that the longer the block length, the longer the time required to accomplish a single read operation.

Frequency Loading

The total time required to process a chained file can often be reduced by loading the most active records first, thus assuring their being placed in a home location or a prime position in a chain. It has been found that in many applications approximately 80% of the transactions apply to only 20% of the file. For example, out of every 100 transactions, 80 will be for the 20% high-activity items.

If no activity count is available when a file is initially loaded, it may be worthwhile to set up a field in each record to accumulate such a count. At a later date, the file can be sorted on this count and reloaded.

Additions and Deletions

Continual additions to, and deletions from, a file affect the efficiency of any organization scheme. The effect on a chained file, however, is comparatively slight. Thus, chaining is particularly suited to files with a very large turnover.

Sequential Files

Two hundred sectors of 1311 disk storage or 800 sectors of 1301 disk storage can be read at a single access-arm setting. Because of this ability, many disk applications can be made more efficient by organizing files in a sequential order rather than in random order. The goal in a random file is to come as close as possible to an average of one seek and one read per record. The goal in a sequential file is one seek per cylinder and one read per block.

When a sequential file is being loaded, the records are previously sorted into sequence by control field. They are then read into the system and stored in consecutive disk locations. To process the file, a program requires only the upper and lower limits of the file area. It begins with the first record and processes each record in sequence.

Additions and Deletions

Additions to and deletions from a sequential file can be handled in several ways. In some cases it is possible to batch additions and deletions and merge them into the file during a regular updating run. Another method, the one used in the IBM control sequential file-organization routines, uses a *sequence link*. In this case, a field of blanks is appended to the end of each record as it is loaded into disk storage. Records to be added to the file are written into a separate area of disk storage. The address of the added record is written in the sequence-link field of the record that sequentially precedes the added record. Similarly, a record can be deleted and the sequence reestablished by placing the address of the following record in the sequence-link field of the preceding record. When the file is processed, a program always checks the sequence link of a record. If it is blank, it reads the next consecutive disk location. If it contains an address, it seeks and reads that address.

Additions and deletions quickly impair the efficiency of a sequential file. It may, therefore, be necessary to resequence the file often.

Blocking Records in a Sequential File

Records in a sequential file should be blocked in such a way as to reduce input/output time to a minimum. Because there need be but one seek per cylinder, the objective is to reduce rotational delay time.

As a general rule, the longer the block length, the shorter the *read, write, write-check* time per record. However, in some cases, the time required to process or update records is such that a rotation can be gained by using a shorter block length.

Random Processing of a Sequential File

In some applications it is desirable to process sequential files in a random order. Because there is no necessary relationship between the control data of a record in a sequential file and the disk address at which it is located, some other kind of addressing technique must be used.

The control sequential routines described in this bulletin use a *distribution index* for this purpose. The distribution index is a table set up during the initial loading of the file. It consists of the control data from certain records in the file and the disk addresses of those records. An entry can be made to the index once for each cylinder or once for any specified number of records. The distribution index is written into disk storage in an area specified by the user. When it is desired to locate a given record,-the index is brought to core storage and searched. The index does not necessarily give the exact disk address of the desired record, but it does establish the range within which it is located. For example, if an entry were made after every twentieth record of a file of 100-character records, the index would tell the track on which the record was located. The track could be easily searched or scanned for the desired record.

The average number of seeks and reads required to locate a single record in a sequential file depends upon:

1. The size of the file and the number of additions records.
2. The size of the distribution index.
3. The frequency of the entries in the index.

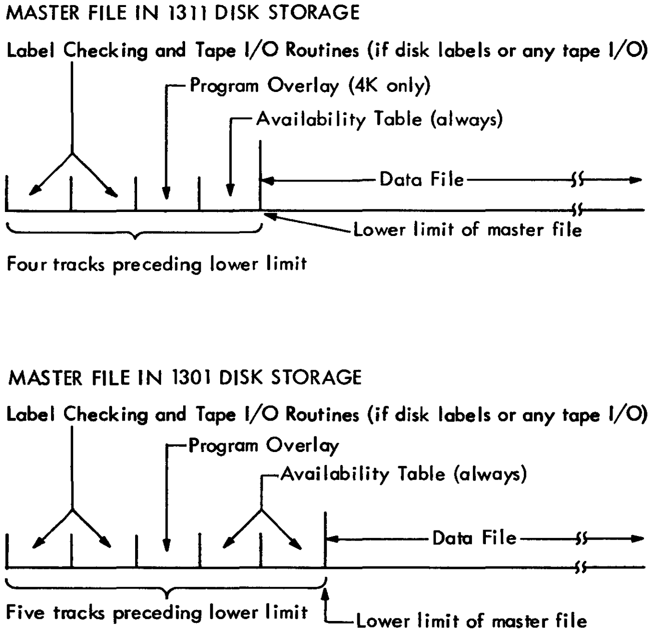
The size of the distribution index depends on the frequency of entries and the length of the control fields in the records. Thus at least one seek and possibly a number of reads are required to locate the. proper portion of the index. In some cases it is practical to set up a short table in core storage to index the distribution index. In this way the table in core storage indicates the desired sector from the distribution index.

Having found the range within which the desired record is located, another seek and additional reads are performed to locate the record.

Random Programs

This section describes eight programs used to load and maintain a random file. Each program is generated separately and is available from both of the file-organization packages. The file on which these programs are to operate can consist of both master and trailer records. Master records are the basic data records in a file. Trailer records are considered to be extensions of master records. A master record can have a number of trailer records, but there can be no trailers without a master.

Master and trailer records are loaded separately and into different areas of disk storage. The master and trailer areas must both be in the same type of disk storage unit. For example, if the master area is in 1301 storage, the trailer area must be also. However, input and output can be on different types of disk units. As each trailer is loaded, its disk address is linked to the master. In effect, the linked master and trailer records can be considered a variable-length disk record. Certain processing runs may consider only the master records; others, only the trailers. However, when the entire block of information is needed, a program can seek the master record and then follow a chain of any number of trailer records associated with that master.



MASTER FILE IN 1311 DISK STORAGE
Label Checking and Tape I/O Routines (if disk labels or any tape I/O)
Program Overlay (4K only)
Availability Table (always)
Data File
Lower limit of master file
Four tracks preceding lower limit
MASTER FILE IN 1301 DISK STORAGE
Label Checking and Tape I/O Routines (if disk labels or any tape I/O)
Program Overlay
Availability Table (always)
Data File
Five tracks preceding lower limit
Lower limit of master file

Figure 2. Work Tracks Preceding Random Master File

The random object programs that can be generated are:

1. PASS1	Loads home master records.
2. PASS2	Loads non-home master records.
3. RNADD	Adds master records to an organized file.
4. TRADD	Loads or adds trailer records.
5. RNDEL1	Deletes or tags master records and associated trailer records.
6. RNDEL2	Deletes previously tagged records.
7. TRDEL	Deletes trailer records.
8. RNUNLD	Unloads a random file for reorganization.

A complete set of random programs generated for a given file normally includes each of the above-mentioned programs and at least one additional version of:

1. PASS1	The first performs the initial load. The second reorganizes files unloaded by RNUNLD.
2. PASS2	Same purpose as the two versions of PASS1.
3. RNDEL1	The first deletes records. The second tags records.
4. RNUNLD	The first unloads master records for reorganization. The second unloads trailer records for reorganization. *Note:* Additional versions of this program are often generated to produce stripped files to be processed by other programs.

Work Tracks

An area immediately preceding the master-file area must be reserved for these programs to use as working storage. The area required may be from one to five tracks, depending on the amount of core storage in the system, the kind of disk unit being used, and whether disk or tape labels are used.

Figure 2 shows the work area for both 1311 and 1301 master files.

1. The first one (1311) or two (1301) tracks preceding the master file area are used to store the availability table. This table stays on the track from one run to the next and must not be destroyed.
2. The next track (second on 1311 or third on 1301) preceding the master area is always required if the file is in 1301 disk storage. If the file is in 1311 disk storage, this track is used only in a 4K system. It is used for temporary storage of program overlays.

Nothing is retained on the track from one run to the next; therefore, it can be used as a work area by other programs.

3. The third and fourth (1311) or fourth and fifth (1301) tracks preceding the master area are used to store disk-label checking routines and tape I/O routines. If the programs neither process disk labels nor process tape input-output, these tracks need not be reserved. Nothing is retained on these tracks from one run to the next; therefore, they can be used as a work area by other programs.

(PASS1) Pass-1 Random Load Program

PASS1 reads the entire input master file and loads all of those records whose control data is, or converts to, the address of an unused disk location. Such records are called *home* records. Records with control data that converts to disk locations that are already occupied are handled as *non-home* records; marked or set aside to be loaded during PASS2. If there are no duplicate disk addresses in the file, there are no non-home records. In this case, PASS1 is used to load all records in the file including any to be added during subsequent runs. Therefore, PASS2 and RNADD are not required.

Two PASS1 object programs are normally generated for a given file. The first is to perform the initial load of the file. The second is used to reorganize files unloaded by the random unload program, RNUNLD.

Input Records

Input to PASS1 can be from cards, tape, or disk storage. The formats allowed are discussed under *Input/Output Files.* The file should be sorted on activity, if such a count is available (see *Frequency Loading).*

Output Records

The format of the output master records depends on the number and length of the appended fields. When a master record is loaded into disk storage, the routines append the following fields:

Overflow Address Field: This field is appended to the end of each master record. If the master file is unblocked, this is a 6-digit field. If the file is blocked, this is a 7-digit field. If there are no duplicate disk addresses, a single position is appended. This position contains an availability indicator.

Trailer Address Field: If trailer records are used, each master record has a 7-digit field appended to the front of the record. After trailer records are loaded, this field contains the address of the first trailer record (if any) associated with the master record.

Figure 3 illustrates four kinds of disk-storage master records showing the user's data records plus the combinations of appended locations as required by the configuration of the file.

Conversion Routine

The user must supply a conversion routine to be assembled along with the following random programs:

PASS1	– except when reloading a file unloaded by a RNUNLD program generated with the operand CONADD in the TYPERNUNLD card.
PASS2	– only when the NONHOMES card has the operand PUNCHZONE.
RNADD	– always
TRADD	– always
RNDEL1	– always
RNDEL2	– except when there are no duplicate addresses.
TRDEL	– always
RNUNLD	– only when the TYPERNUNLD card has the operand CONADD.

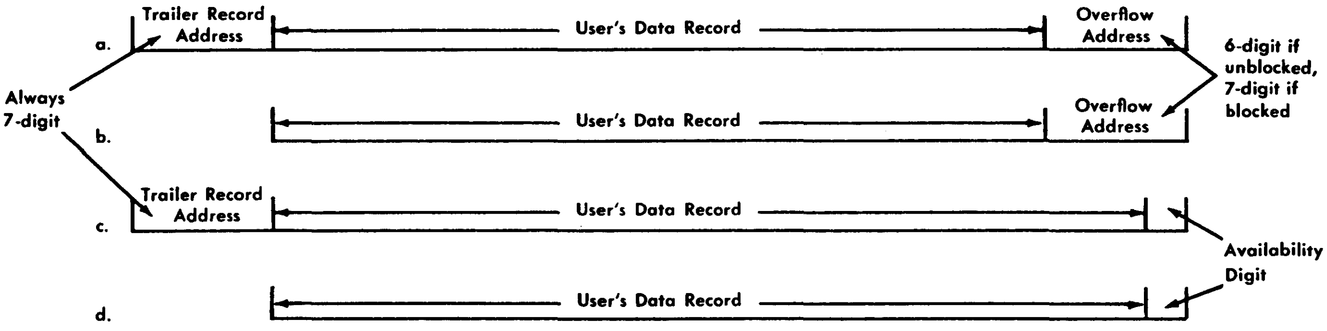


Figure 3. Four Kinds of Disk Storage Master Records

The object programs extract the control data of a record and place it in a field labeled CONTD. The source statements that define this field are taken from the file-organization library routines. The field is the exact length specified for the control data. If the control data is broken into subfields within the records, the subfields are arranged (from the left to right) in the order specified in the MASTCONTRL card. The field has a word mark in the high-order (left-most) position. This word mark must not be cleared by the user's routine, and no other word marks can be in the field when control is returned to the main program.

After placing the control data in the field labeled CONTD, the program branches to the first instruction of the user's conversion routine. This first instruction must be labeled CONROU. If index registers are used, their contents must be saved and restored before returning control to the main program. The conversion routine must take the control data from the field labeled CONTD, develop a valid address (see *Converted Record Addresses*), and place that address in a field labeled CONV1. The source statements that define this field are also taken from the file-organization library routines. If the file is unblocked, this is a 6-digit field; if blocked, a 7-digit field. It has a word mark in the high-order (left-most) position.

The last instruction of the conversion routine must be labeled RETU1. It returns control to the program by branching to START3.

A routine must be provided even if disk addresses are extracted directly from the records. In this case the routine simply moves the address to CONV1 and then branches to START3.

Note. If a file extends from one disk pack to another, it is up to the conversion routine to avoid those addresses that would cause a block to overflow from one pack to another. See *Cylinder Overflow* for special considerations of multi-pack files.

Converted Record Addresses

The random routines require that the control data of each record be, or be converted to, the address of the first sector of a block. If only one record is written per block, these must be 6-digit disk addresses. If more than one record is written per block, these must be 7-digit addresses in the form SSSSSSR. The first six digits are the address of the initial sector of a block within the file area.

The seventh digit (R) designates the position of the record within the block. This number can range from 0 to 9, allowing up to ten records per block. If only one record is written per block, this position is not used. If more than one record is written per block, the first is numbered 0; the second is numbered 1; the third, 2; and so on, to record 10 which is numbered 9. The maximum value that R is allowed to assume determines the blocking factor (records per block) of the file.

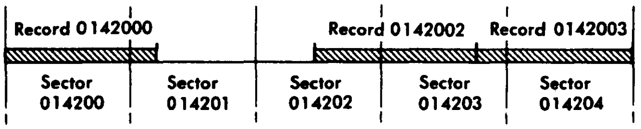


Figure 4. Four Records per Five-Sector Block

Figure 4 illustrates the addresses of records within a 5-sector block. In this example it is assumed that four 125-character disk records are stored in each block. The control data of three master records converts to the following addresses: 0142000, 0142002, and 0142003.

From this example note, first of all, that these records are not considered synonyms. Each has a different units digit, thus, they are not part of the same chain. The relative position of each record within the block is represented by the digits 0 to 9 in the units position of the address. The maximum value that this position can attain is equal to the number of records per block minus one. Obtaining the *relative* low-order core-storage address of individual records within a disk block in a random file is accomplished by multiplying the record length by R + 1.

General Operation of Program

Before loading the data records, the file area is cleared to blanks. The disk addresses originally in the area are retained, thus allowing blocks to be arranged in the most advantageous order. Because the original addresses are retained, the area must already be in the move mode. If the user requests that terminal record marks be written in the area, they are inserted at this time in the low-order position of every record location in the file area.

Note. When additions are being made to a file of records with no synonyms, the clearing routine is bypassed. See discussion of MASTR RDLIN card under *Disk RDLIN Cards.*

After each record is read and assembled for processing, the program branches to the user's conversion routine to develop a disk-storage address. The address is sought and the disk block in which the record is to be written is read into core storage. Once in core storage, the record location within the block is tested for availability by examining the *availability indicator.* This is the last position within the record except when terminal record marks are present. In this case, the next-to-last position is used as the availability indicator. If this position is blank, the record location is available. If it contains a character other than a blank, the location is occupied.

If the location is available, the input record is a home record. To identify all home records uniquely, A- and B-bits are placed in the third position of the overflow-address field (Figure 5). In addition to this, a digit 1 is placed in the availability-indicator position of the overflow-address field. The digit 1 is an availability code, indicating the presence of a record. Note that if an overflow address is subsequently placed in a home record, the digit 1 is replaced by the units position of the overflow address. However, the A- and B-bits in the third position are restored, retaining the home-record identification. The record is then written, with the block, back into disk storage.

If the availability indicator contains a character other than a blank, the input record is a non-home record.

Non-home records are processed according to the operand punched in the NONHOMES parameter card:

Operand	*Output*
RECINWORK	The entire non-home record, preceded by its converted address, is written in a disk work area.
RECONTAPE	The entire non-home record, preceded by its converted address, is written on magnetic tape.
RECINCARD	The entire non-home record, preceded by its converted address, is punched into cards. On a 1440 system, this operation can be performed only on an IBM 1444 Card Punch. It cannot be done on an IBM 1442 Card Read-Punch.
ADDINWORK	If the input is from disk storage, the input disk addresses of non-home records, preceded by the converted addresses, can be written in a disk work area.
1ADDINCARD or, 5ADDINCARD	If the input is from disk storage, the input disk addresses of non-home records, preceded by the converted addresses, can be punched into cards. The user can specify that either one or five sets of addresses be punched per card.
PUNCHZONE	On a 1440 system, if card input is read on an IBM 1442 Card Read-Punch, a zone can be punched in column 9 of the last card of each non-home input record. *Note.* This option cannot be taken if the user supplies a subroutine to assemble input records from cards (see *Card Files* under *Input/Output Files*).

Parameter Cards Required

ROUTINE
SYSTEMSPEC (optional*)
DISKDRIVES
EXITS (optional*)
TYPEINPUT
TYPERNUNLD
(if reorganized)
INPUTMEDIA
INPMASTREC
MASTCONTRL
ADDRESSING
MASTBLOCK
MASTLIMITS
NONHOMES (if duplicate addressing)
LOADAUDIT (optional*)
INLABELS (if disk input)
OUTLABELS

* See *Parameter Cards* to find out when this card can be omitted from the source deck.

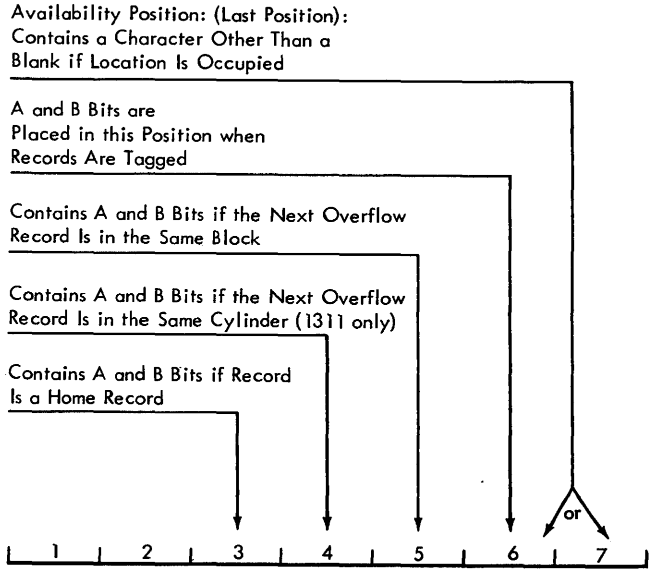

Availability Position: (Last Position): Contains a Character Other Than a Blank if Location Is Occupied
A and B Bits are Placed in this Position when Records Are Tagged
Contains A and B Bits if the Next Overflow Record Is in the Same Block
Contains A and B Bits if the Next Overflow Record Is in the Same Cylinder (1311 only)
Contains A and B Bits if Record Is a Home Record
or
1
2
3
4
5
6
7

Figure 5. Overflow-Address Field of Master Records

The user must supply his symbolic conversion routine, unless PASS1 is being used to load a reorganized file with the converted addresses appended to the records.

(PASS2) Pass-Two Random-Load Program

This program loads all non-home master records in a file. The input can be in any one of the four forms described for non-home output under *PASS1.*

If the input to PASS1 was from an IBM 1442 Card Read-Punch and if the PUNCHZONE option was taken in the NONHOMES card, all the original data cards are read, including those processed as home records during PASS1. Column 9 of the last card of each record is checked to see whether it contains a zone punch. If no zone is detected, the record was loaded as a *home* during PASS1 and is bypassed. If column 9 has a zone punch, the record must be chained. The program then branches to the user's conversion routine for the converted disk address.

If any of the other options were taken in the NONHOMES card, the input includes the converted address. The conversion routine is therefore not required in these cases.

The converted disk address is sought and read. During this pass, all home addresses are occupied. Each link of the existing chain is searched until the last link of the chain is reached. This last record is used as a *base record* in searching for an available location. The non-home record is placed in an available location as near as possible to the base record.

In both 1311 and 1301 disk storage, if the record is written in the same block as the base record, its actual disk address is written in the overflow-address field of the base record with A- and B-bits placed over the fourth and fifth digits.

In 1311 disk storage A- and B-bits are placed over the fourth digit only, if the record is written in the same cylinder but not in the same block as the base record (Figure 5).

If the entire cylinder containing the base record is filled, an available location must be found in another cylinder. The cylinder that is searched depends on the machine configuration.

The record is placed in a cylinder as close as possible to that containing the base record (the last link of the chain) if:

1. the master-file area is in 1301 disk storage, *or*
2. the master-file area is in 1311 disk storage and the system has *both:*
   a. the direct-seek feature, and
   b. more than 4,000 core-storage positions.

The record is placed in a cylinder as close as possible to cylinder 00 in the disk pack that contains the upper limit if:

1. the master-file area is in 1311 disk storage and the system has *either:*
   a. fewer than 8,000 core-storage positions, or
   b. no direct-seek feature.

Parameter Cards Required

With the exception of the ROUTINE card, the same cards used in generating PASS1 must be used for this program also. All the operands must be the same, with the exception, perhaps, of the INPUTMEDIA card. The INPUTMEDIA card must be changed if the non-home records were written on tape in PASS1. (See *INPUTMEDIA Card*).

The conversion routine is supplied only if the PUNCHZONE option was taken in the NONHOMES card.

(RNADD) Additions Program for Master Records

This program adds master records to an organized random file. Input can be from cards, disk, or tape. The specifications of the records added must be the same as those for the initial loading, but the input media need not be the same as for the initially loaded file. For example, the original file could have been loaded from disk, and the additions, from cards. Disk or tape input blocks need not be the same length nor have the same number of records per block as the original input file.

After a record to be added is assembled for processing, the program branches to the user's conversion routine for a disk-storage address. The address is sought and the disk block is read into core storage. The address to which the control data converted can be:

1. an unoccupied location
2. the home address of an existing chain
3. a location occupied by the second, third, etc., link of a chain.

When the control data of a record converts to an unoccupied location, the record is considered a home record and is processed as in PASS1.

When the control data of a record converts to the home address of an existing chain, RNADD adds the record to this chain by placing it in an available location near the last link of the chain. This is done as in PASS2.

If, however, the control data of a record converts to the address of a location occupied by the second, third, or higher link of an existing chain, the record contained in the link is removed and placed in another available location. The overflow-address field of the preceding link of the chain is modified to contain the *new* address of the displaced record. The new record is then written as a home record in the address to which its control data converted.

For example, in Figure 6, locations 0072010, 0072101, 0072620, 0072621, and 0072800 constitute the original chain. The record to be added converts to address 0072621, which is occupied by the fourth link of the chain. To place the new record in its proper location, the record in location 0072621 is moved to 0072740 and the overflow address in record 0072620 is changed to 0072740.

Parameter Cards Required

ROUTINE
SYSTEMSPEC (optional*)
DISKDRIVES
EXITS (optional*)
INPUTMEDIA
INPMASTREC
MASTCONTRL
ADDRESSING
MASTBLOCK
MASTLIMITS
LOADAUDIT (optional*)
INLABELS (if disk input)
OUTLABELS

* See *Parameter Cards* to find out when this card should be omitted from the source deck.

The user must supply the symbolic conversion routine when generating this program.

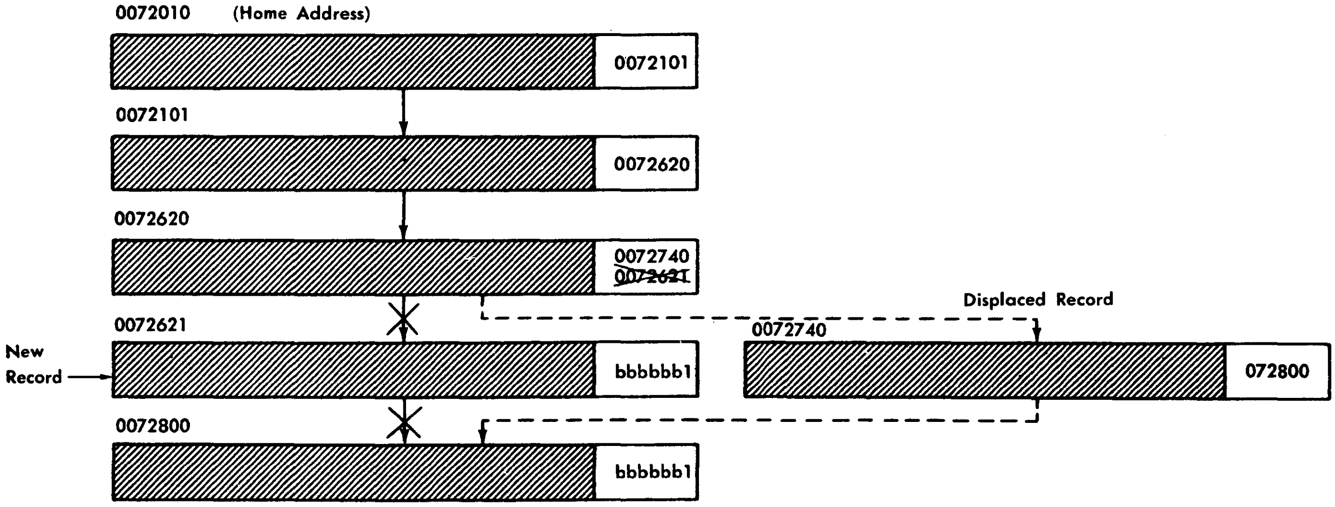


Figure 6. Additions to a Disk-Storage Chain

(TRADD) Loading and Additions Program for Trailer Records

This program loads or adds all trailer records in a file. Input can be from cards, disk or tape. Trailer records are loaded into a separate area of disk storage set aside by the user. This area must have contiguous addresses throughout, and the area must be distinct from the master-file area. A master record can have any number of trailer records, and there can be no trailer without a master.

A 7-digit address field is appended to the front of each trailer record as it is loaded. The low-order position of this field is used as an availability indicator. This position always contains a character if the location is occupied, and a blank if the location is vacant (Figure 7).

Trailer records can be written with up to ten records per block. Neither the block length nor the blocking factor need be the same as in the master file. Each record is written in the space immediately following the last-stored trailer record. The addresses of subsequent trailers belonging to the same master record are written in the address field of the preceding trailer record. These are always 7-digit addresses in the form discussed in the *Converted Record Addresses* section under PASS1. The seventh digit is retained even when the trailer file is unblocked; though in this case, it is always zero.

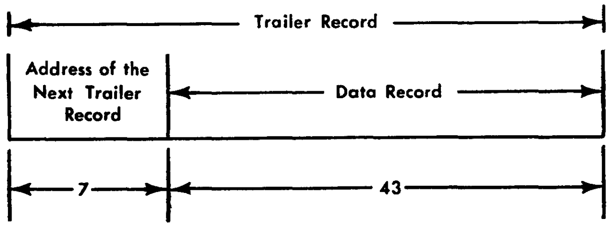


Figure 7. Layout of a 50-Character Trailer Record

The first record location in the trailer area contains a dummy record. Whenever an initial load of trailers is being performed, with either new or reorganized input, at least the dummy record location must be cleared to blanks in the move mode. At the completion of the initial loading run and each subsequent additions run, the address of the first available location in the trailer area is placed in positions 8-14 of this dummy record. Positions 15-20 contain a count of the number of trailer-record locations left in the area.

Parameter Cards Required

ROUTINE
SYSTEMSPEC (optional*)
DISKDRIVES
EXITS (optional*)
TYPEINPUT
TYPERNUNLD
(if reorganized)
INPUTMEDIA
INPMASTREC

MASTCONTRL
ADDRESSING
MASTBLOCK
MASTLIMITS
TRAILENGTH
TRAILCNTRL
TRAILBLOCK
INLABELS (if disk input)
OUTLABELS

* See *Parameter Cards* to find out when this card can be omitted from the source deck.

The conversion routine must be used with this program.

Note. This program must be generated to load either new trailers or a file unloaded by RNUNLD (REORGANIZED). If generated to load a reorganized file, the TYPERNUNLD card is included with one of the operands: ACTIVE, TAGGED, or BOTH. However, the program will load any of the three kinds of record. During the initial loading of a reorganized trailer file, when the dummy record is blank, all trailer address fields in the master area are blanked prior to loading the trailers. This prevents incorrect trailer address fields from remaining on master records after the trailer file has been reorganized.

Note that an entire reorganized trailer file does not have to be loaded during the initial reorganizing run. In subsequent runs of the reorganized TRADD program, if the dummy record is not blanked, the program will operate as a normal additions run.

(RNDEL1) Delete or Tag Program for Master and Trailer Records

This program is used either to delete or tag master records and their associated trailers. The tagged records can later be deleted by RNDEL2. The input to the program must be from cards and consists of the control data of the records to be deleted or tagged. The input cards can contain as many control fields as will fit on a card. The control data must be punched in contiguous card columns starting in column 1. If control fields in the records are broken in sections, the sections must be punched in the order of importance (as specified in the MASTCONTRL card, not necessarily in order of appearance in the records).

The control data of each record to be deleted or tagged is converted to the home address of the chain containing the record. The home address is sought and each link of the chain is examined until the record is found.

If the record is only to be tagged, A- and B-bits are placed in the sixth position of the overflow-address field of the master record and in the sixth position of the appended field of each trailer record.

If the record is to be physically deleted, the operation of the program depends on the location of the record. The record to be deleted can be:

1. a home record with no overflows
2. a home record with overflows
3. an overflow record in an existing chain.

If the record to be deleted is a home record and is the only link in a chain, the record is deleted by blanking the entire record.

If the record to be deleted is a home record and if there are one or more additional links in the chain, the home record is replaced by the entire contents (record and overflow address) of the first overflow location. The first overflow location is then blanked out.

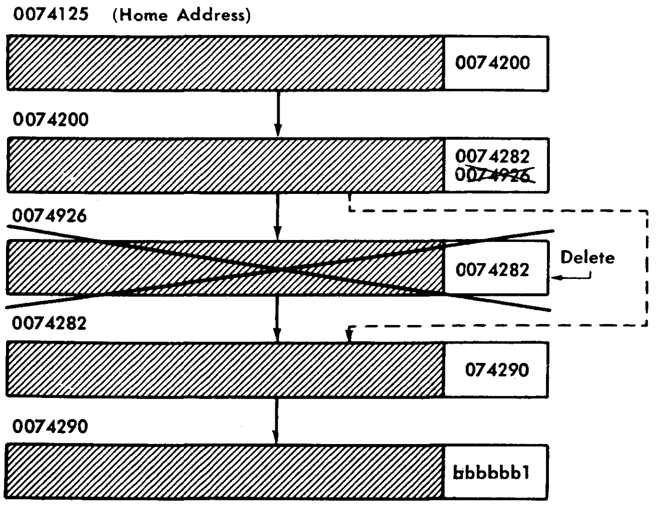


Figure 8. Deletions from a Disk-Storage Chain

If the record to be deleted is an overflow record in an existing chain, the record is deleted (again by blanking the entire record) and the overflow-address field of the deleted record is placed in the previous link of the chain. For example, in Figure 8 the record in location 0074926 is being deleted. To maintain a continuous chain, the address 0074282, which occupies the overflow-address field of location 0074926, must be placed in the overflow-address field of location 0074200.

Whenever a master record is deleted or tagged, all trailer records associated with it are also deleted or tagged. As with master records, trailer records are deleted by blanking the entire record.

If all record locations have terminal record-marks, these are not blanked.

Parameter Cards Required

ROUTINE
SYSTEMSPEC (optional*)
DISKDRIVES
EXITS (optional*)
INPMASTREC
MASTCONTRL
ADDRESSING

MASTBLOCK
MASTLIMITS
TRAILENGTH }
TRAILCNTRL } if trailers
TRAILBLOCK }
TYPEDELETE
LOADAUDIT (optional*)
INLABELS

[*] See *Parameter Cards* to find out when this card can be omitted from the source deck.

The conversion routine must be used with this program.

(RNDEL2) Delete Program for Tagged Master and Trailer Records

This program deletes the master and trailer records that were previously tagged by RNDEL1.

The program scans the disk-storage area between the master-file limits, deleting the previously tagged master records and their associated trailer records. Relinking the chains from which records are deleted is accomplished in this routine in the same manner as in the RNDEL1 routine.

Parameter Cards Required

The parameter cards required to generate RNDEL 2 are the same as those listed under RNDEL1. The conversion routine is required for this program if the file is chained.

(TRDEL) Delete Program for Single Trailer Records

This program deletes single trailer records. The input must be from cards and consists of the control data of the trailer records to be deleted. Each record is specified by two control fields: a major followed by a minor. The major control field is identical to the control data of the master record. The minor control field indicates the particular trailer record to be deleted. The control fields are punched in contiguous card columns, starting in column 1. If the major control fields are broken in sections, the sections must be punched in the order of importance, as specified in the MASTCONTRL card (not necessarily in order of appearance in the records). Each card can contain as many complete control fields (major and minor) as will fit on a card. No spaces are left between control fields.

The major control field is converted to the home address of the chain containing the master record. This chain is searched until the master record is found. From the trailer-address field in the master record, the program locates the first trailer record associated with that master. The trailer-record chain is then searched until the record identified by the minor control field is found. The trailer record is deleted by blanking it out. The trailer-address field of the deleted record is moved to the record preceding it in the chain. Terminal record marks are not deleted.

Parameter Cards Required

ROUTINE
SYSTEMSPEC (optional*)
DISKDRIVES
EXITS (optional*)
INPMASTREC
MASTCONTRL
ADDRESSING
MASTBLOCK
MASTLIMITS
TRAILENGTH
TRAILCNTRL
MINORCNTRL
TRAILBLOCK
TYPEDELETE
INLABELS

* See *Parameter Cards* to find out when this card can be omitted from the source deck.

The conversion routine is used by this program.

(RNUNLD) Unload Program for Reorganizing a Random File

This program is used to unload a random file of master or trailer records for subsequent reorganization and reloading. Master and trailer files must be unloaded separately. A file can be unloaded into cards, tape or disk storage. The program can unload:

1. only the active records of a file
2. only the tagged records of a file
3. both active and tagged records.

Several kinds of output are possible. The user's choice is specified in the TYPERNUNLD and UNLDMEDIA cards.

Card Output. If the file is punched into cards, the resulting card records will be in the format shown in Figure 9. The first eight positions contain an identification number. This number is the same for all of the cards for one record and is unique for each record.

Columns 9 and 10 contain an intrarecord sequence number, beginning with 01 for the first card of each record.

If desired, master records can be preceded by either the converted address of the record or the trailer-address field, or both. The overflow-address field is always stripped from master records, and the trailer-address field is always stripped from trailer records.

Non-home master records are indicated by a 12-punch over column 9 of the last card of the record.

Tagged records (both master and trailer) are indicated by a 12-punch over column 10 of the last card of the record.

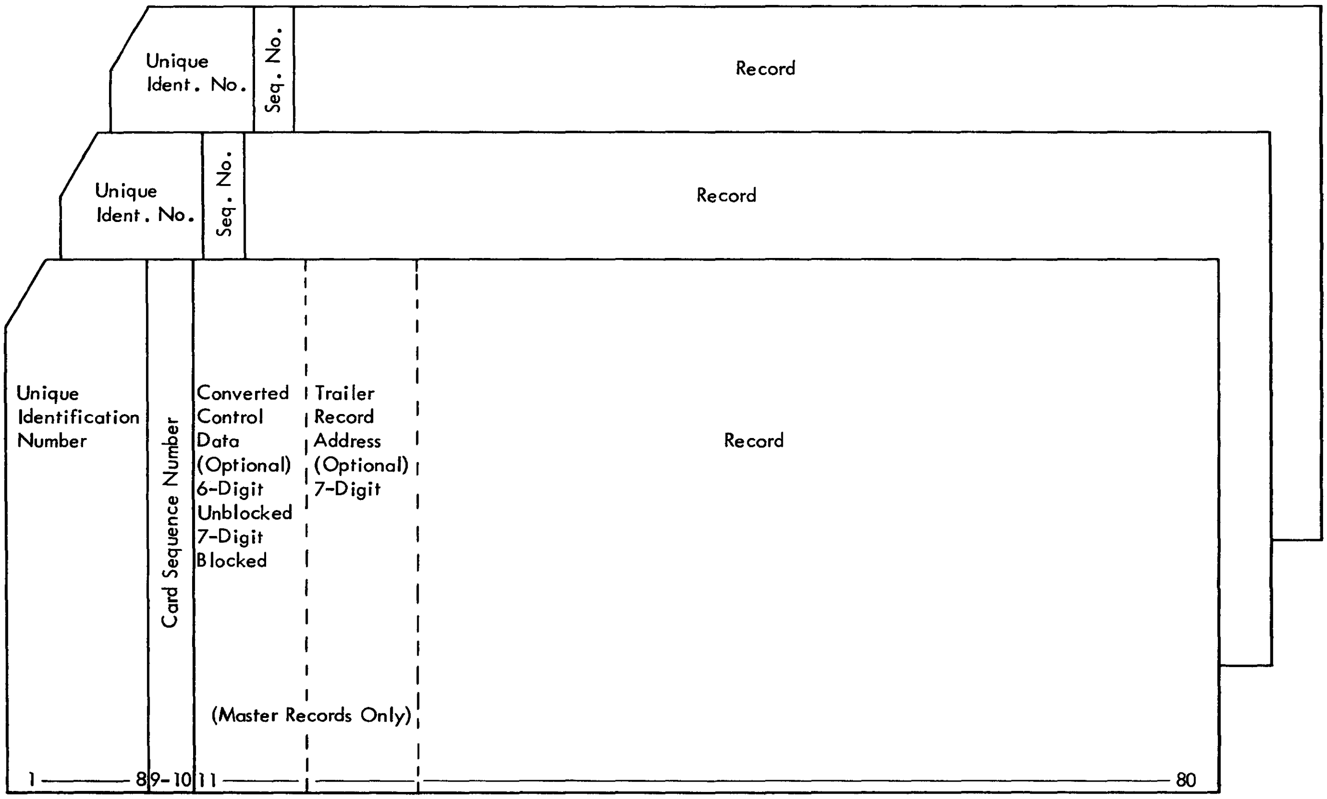

Unique Ident. No.
Seq. No.
Record
Unique Ident. No.
Seq. No.
Record
Unique Identification Number
Card Sequence Number
Converted Control Data (Optional) 6-Digit Unblocked 7-Digit Blocked
Trailer Record Address (Optional) 7-Digit
(Master Records Only)
Record
1 — 8
9-10
11
80

Figure 9. Card Ouput of RNUNLD

Disk or Tape Output. Output on disk or tape can be blocked with up to 70 records per block.

Unloaded master records can be in either of two formats, depending on the operands in the TYPERNUNLD card.

If CONADD is used, each master record is preceded by a single character position that contains:

1. A- and B-bits, if the record is tagged; otherwise, no zone bits.
2. A 1-bit, if the record is a non-home record; otherwise, no numeric bits.

This position is followed by a 7-digit field (6-digit if organized file is unblocked) that contains the converted disk address of the record.

If NOCONADD is used, the record is preceded by only the single position for tag and home-record indication.

The user also specifies whether the output master record is to include the 7-digit trailer-address field.

When trailer records are being unloaded, the trailer-address field is stripped from the record. A single character position is appended to the front of the record. If the record is tagged, this position contains A- and B-bits. If the record is not tagged, this position contains a blank.

Note. This tag-zone position is always appended, regardless of the operands: ACTIVE, TAGGED, or BOTH.

Parameter Cards Required

ROUTINE
SYSTEMSPEC (optional*)
DISKDRIVES
EXITS (optional*)
INPMASTREC
MASTCONTRL
ADDRESSING
MASTBLOCK

MASTLIMITS
TRAILENGTH
TRAILCNTRL
TRAILBLOCK } (if unloading trailer file)
TYPERNUNLD
UNLDMEDIA
INLABELS
OUTLABELS
(if disk output)

* See *Parameter Cards* to find out when this card can be omitted from the source deck.

The conversion routine is used if the TYPERNUNLD card contains the entry CONADD.

Control-Sequential Programs

The five Control-Sequential Disk-File Organization programs load and maintain a presequenced file. The programs construct and use a distribution index that makes it possible to locate records quickly within the file area. Linkage is established to records added and around records deleted.

The control-sequential programs that can be generated are:

1. CSLOAD — Loads a presequenced file and builds the distribution index.
2. CSADD — Adds records to a sequential file
3. CSDEL1 — Deletes or tags records
4. CSDEL2 — Deletes previously tagged records
5. CSUNLD — Merges the main sequential file with any records added by the CSADD program and either:
   a. unloads the file for use with other programs
   b. reorganizes the file and rebuilds the distribution index.

A complete set of control-sequential programs generated for a given file normally includes each of the above-mentioned programs and at least one additional version of:

1. CSDEL1 — The first deletes records
The second tags records.
2. CSUNLD — Various versions produce:
   a. 1-pass reorganization in a new file area
   b. 2-pass reorganization in the original area.
   c. Stripped, merged output for CSLOAD and other programs. If this type of output is used to reorganize a file, an additional version of CSLOAD is required.

Work Tracks

An area immediately preceding the main-file area may be required for these programs to use as working storage. One, two, or three tracks may be required, depending on the amount of core storage in the system, the kind of disk unit used, and whether disk or tape labels are used.

Figure 10 shows the work area for both 1301 and 1311 main-file areas.

1. The first track preceding the main-file area is always required if the file is in 1301 disk storage. If the file is in 1311 disk storage, this track is used only in a 4K system. It is used for temporary storage of program overlays. Nothing is retained on the track from one run to the next; therefore, it can be used as a work area by other programs.
2. The second and third tracks preceding the main-file area are used to store disk-label checking routines and tape I/O routines. If the programs neither process disk labels nor process tape input-output, these tracks need not be reserved. Nothing is retained on these tracks from one run to the next; therefore, they can be used as a work area by other programs.

Note. When the first track is not used for program overlay, the program still skips to the second and third tracks to store the label routines.

(CSLOAD) Load Program for Control-Sequential Files

This program loads a presequenced file into disk storage. The card, tape, or disk input file can be in either ascending or descending sequence. The load program bypasses and prints the control field of any record that has control data out of sequence. At least the first block of the organized file area must be cleared to blanks in the move mode before performing the initial load.

A maximum of ten records can be written in each block in a 4K system, and a maximum of 30 records per block, in an 8K or larger system. Unused positions at the end of disk blocks are padded with blanks.

A field of blanks is appended to the end of each record as it is loaded. If the file is unblocked, this is a 6-digit field; if blocked, a 7-digit field. These fields are used for any sequence links made necessary by additions to, or deletions from, the file.

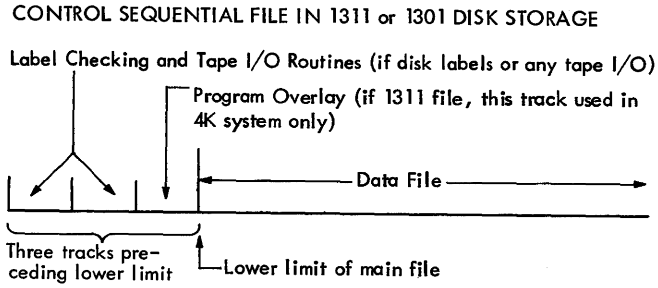

CONTROL SEQUENTIAL FILE IN 1311 or 1301 DISK STORAGE
Label Checking and Tape I/O Routines (if disk labels or any tape I/O)
Program Overlay (if 1311 file, this track used in 4K system only)
Data File
Three tracks preceding lower limit
Lower limit of main file

Figure 10. Work Tracks Preceding Main Sequential File Area

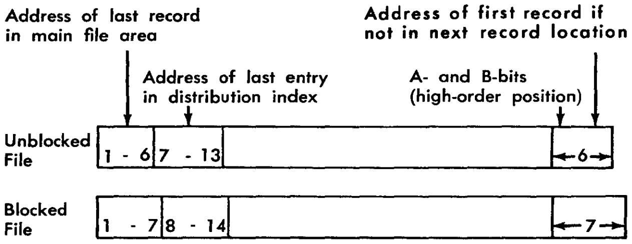

Address of last record in main file area
Address of first record if not in next record location
Address of last entry in distribution index
A- and B-bits (high-order position)
Unblocked File
1 - 6
7 - 13
6
Blocked File
1 - 7
8 - 14
7

Figure 11. Dummy Record Preceding Main Sequential File Area

A dummy record is written in the first record location in the file area (Figure 11). This dummy record contains:

1. the address of the last record in the main-file area.
2. the address of the last entry made to the distribution index.
3. the address of the first record in the file, if that record is not the one that immediately follows the dummy record. This situation arises when:
   a. a record is added in the additions area that sequentially precedes the first record in the main area or,
   b. the first record in the main-file area has been deleted.

The first position of the sequence-link field contains A- and B-bits to indicate a dummy record.

The first input record is written in the second record location, and each subsequent record is written in the next consecutive location.

Distribution Index

A distribution index is set up by CSLOAD as the file is loaded. This index is used to locate records within the sequential file. (See *Random Processing of a Sequential File.*) The distribution index is also used by CSADD and CSDEL1. The user specifies the frequency with which entries are to be made to the index. An entry consists of the control data of a record, followed by its disk address. An entry can be made for the first record written in each cylinder or for every *nth* record. An entry is always made for the first and last records loaded.

For example, if the user specifies that an entry is to be made for every 20 records, it is made for the first record, the twentieth, the fortieth, the sixtieth, and so on, until the last record has been loaded. An entry is then made for the last record unless it was already entered as a result of being an *nth* record.

If a subsequent run is made with CSLOAD to load additional records at the end of the main-file area, the count begins again at 001 when entries are being made for every *n* records. The last entry made during the last run is not removed, and an entry is not made for the first record of this run. Thus, in this case, the consistency of the interval of *n* records between entries might be broken. Additions to and deletions from the file can also break the consistency of the intervals. However, if an entry is made once per cylinder, the entry made for the last record of the previous run is replaced by that for the first record in the next cylinder, provided the last entry did not occur at the start of the cylinder.

The index is written in disk storage in an area specified by the user. The area must be within one disk pack or module. It is written one sector per block, with up to ten complete entries in one sector. The number of entries that can be written in a sector depends on the length of an entry. The length of an entry equals the length of the control fields in the record plus six if the file is unblocked, or seven if the file is blocked.

The address portion of the entry indicates the position of the record within the block. Standard 6-digit disk addresses are used in an unblocked file. A seventh digit (R) is added to the addresses used with blocked files. These addresses are in the form: SSSSSSR. The first six digits are the address of the first sector in the block. The seventh digit (R) designates the position of the record within the block. This number can range from 0 to 9 and has:

- No zone bits, for records 1-10
- B-bit, for records 11-20
- A- and B-bits, for records 21-30.

Parameter Cards Required

ROUTINE
SYSTEMSPEC (optional*)
DISKDRIVES
EXITS (optional*)
TYPEINPUT
INPUTMEDIA
INPMASTREC
MASTCONTRL
MASTBLOCK
COLLATE
DISTENTRY
LOADAUDIT (optional*)
TYPECSUNLD (if reorganized)
INLABELS (if disk input)
OUTLABELS

* See *Parameter Cards* to find out when this card can be omitted from the source deck.

(CSADD) Additions Program for Control-Sequential Files

This program is used to add records to a control-sequential file that has already been loaded. The records to be added must be in the same format as the original records and the additions-file area must follow the same file specifications as the original file area. The input media need not be the same as for the initially loaded file. For example, the original file could have been loaded from disk, and the additions, from cards. Disk or tape input blocks need not be the same length nor have the same number of records per block as the original input file. The card, tape, or disk input must be in the same sequence (ascending or descending) as it was for the load program. CSADD loads the additions records into a separately defined area of disk storage. The additions file area must be cleared to blanks in the move mode before performing the initial additions run. As each record is loaded, its disk address is written in the sequence-link field of the record that sequentially precedes it in the file. The address of the record that sequentially follows the added record is placed in the sequence-link field of the added record. See *Distribution Index* under *CSLOAD* for a description of these addresses.

The first record location in the additions area contains a dummy record (Figure 12). Whenever CSADD is used, the address of the last record added is placed in the first 6 or 7 positions of the dummy record. The next seven positions contain the number of records that can be added to the file. When the additions area is filled, this number is 0000000. The sequence-link field of this dummy record is blank except for A- and B-bits in the first position. These zone bits identify the dummy record.

CSADD makes use of the distribution index to locate records. It never adds *new* entries to the index. However, the program changes the last entry in the index when the record added sequentially follows the last record in the main-file area. In this case, the control data of the last record added replaces the control data of the last entry in the index. The disk address in the last entry remains the same and is still the address of the last record in the original file area.

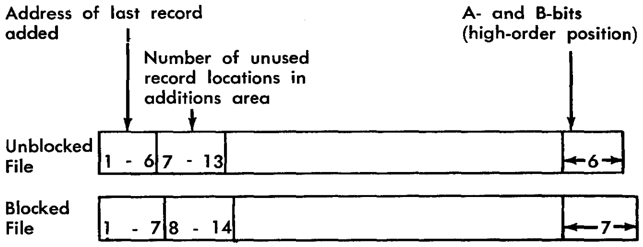

Address of last record added
Number of unused record locations in additions area
A- and B-bits (high-order position)
Unblocked File
1 - 6
7 - 13
←6→
Blocked File
1 - 7
8 - 14
←7→

Figure 12. Dummy Record Preceding Additions Area

If additions are to be made to the end of the file (that is, if all the records to be added contain control numbers beyond the last control numbers of the original input) the load program (CSLOAD) may be used to load the additions.

Parameter Cards Required

ROUTINE
SYSTEMSPEC (optional*)
DISKDRIVES
EXITS (optional*)
INPUTMEDIA
INPMASTREC
MASTCONTRL
MASTBLOCK
COLLATE
LOADAUDIT (optional*)
INLABELS (if disk input)
OUTLABELS

* See *Parameter Cards* to find out when this card can be omitted from the source deck.

(CSDEL1) Delete or Tag Program for Control-Sequential Files

This program is used either to delete or tag records in a control-sequential file. The input must be from cards and consists of the control data of the records to be deleted or tagged. The input cards can contain as many control fields as will fit on a card. The control fields are punched in contiguous card columns, starting in column 1. If control fields are broken in sections, the sections must be punched in the order of importance (as specified in the MASTCONTRL card, not necessarily in order of appearance in the records). Records can be deleted in any sequence. When a large number of records are deleted, however, the records should be deleted in sequence for fastest operation of the program.

Tag Records. A- and B-bits are placed in the sixth position of the sequence-link field of each record tagged.

Delete Records. Deleted records are replaced by blanks. Terminal record marks (if present) are rewritten. If the record being deleted contains an address in its sequence-link field, this address is moved to the sequence-link field in the record that sequentially precedes the deleted record. Otherwise, relinking is accomplished by moving the address of the record that follows the deleted record into the sequence-link field of the record preceding the deletion.

This program uses the distribution index to locate records but does not make any changes to the index. If desired, this program can print all records deleted. The entire record is printed.

Parameter Cards Required

ROUTINE
SYSTEMSPEC (optional*)
DISKDRIVES
EXITS (optional*)
INPMASTREC
MASTBLOCK
MASTCONTRL
COLLATE
TYPEDELETE
INLABELS

* See *Parameter Cards* to find out when this card can be omitted from the source deck.

(CSDEL2) Delete Program for Tagged Control-Sequential Records

This program deletes the records that were tagged by CSDEL1.

The program scans the entire file, following the sequence chain through the main-file area and the additions area, and deleting the previously tagged records. Relinking the sequence chain is accomplished in this program in the same manner as in CSDEL1.

This program does not use the distribution index. If desired, this program prints all records deleted. The entire record is printed. As in CSDEL1, terminal record marks, if present, are retained.

Parameter Cards Required

ROUTINE
SYSTEMSPEC (optional*)
DISKDRIVES
EXITS (optional*)
INPMASTREC
MASTBLOCK
TYPEDELETE
INLABELS

* See *Parameter Cards* to find out when this card can be omitted from the source deck.

(CSUNLD) Unload Program for Reorganizing a Control-Sequential File

This program is used to unload a control-sequential file. A file can be unloaded into cards, disk storage, or tape. The program can unload:

1. only the active records of a file, or
2. only the tagged records of a file, or
3. both active and tagged records.

The possibility of several types of output is offered. The user's choice is specified in the TYPECSUNLD and UNLDMEDIA cards. As the file is unloaded, the records in the additions area are merged with those in the main-file area.

Card Output. If the file is punched into cards, the sequence-link field is stripped from the records. The cards are punched in the format required for input to CSLOAD (see *Card Files,* Figure 14). Tagged records are indicated by a 12-punch over column 10 of the last card of the record.

Disk Output. Three kinds of disk output are possible:

1. *Stripped Merge.* If the TYPECSUNLD card has the operand NOLINK, the records in the main-file area and in the additions area are merged and written into a third disk area. The sequence-link field is stripped from the end of the records. A single character-position is appended to the front of each record. Tagged records are indicated by the presence of A- and B-bits in this position.

The ouput file can be reloaded by a version of CSLOAD generated to load a *reorganized file.* The tag bits are moved back to the sixth position of the sequence-link field when reloading, and the tag-bit position is dropped.

If this form of output is used, the output file can be blocked with up to 70 records per block.

No audit-trail print is available with this stripped merge. The distribution index is not rebuilt.

2. *Relocated Merge.* If the TYPECSUNLD card has the operand LINK, the file is merged and loaded into a *new* file area in one operation. In this case the sequence-link field is blanked except for tag-zone bits, and the field is retained. The output file is resequenced and corresponds to the usual output of CSLOAD.

Neither the block length (number of sectors per block) nor the blocking factor (number of records per block) need be the same in the new file as in the old.

The distribution index is rebuilt as the file is loaded and can be written in the same area as before or in a new area. The interval between entries in the index need not be the same as for the previous file.

The load audit option is available with this operation.

3. *2-Pass Merge.* If the TYPECSUNLD card contains the operands LINK, 2PASS, the file is merged and then written back into the original file area. The primary use of this operation is to reorganize files that occupy more than half of the on-line disk-storage space. The program requires a disk work area at least equal in size to the old additions area plus one disk block of that additions area.

The entire file is merged during the first pass. At the completion of this pass, the file is in sequence with the first sequential record in the second record location of the work area. The sequence continues through the entire work area and then continues from the first record location in the main-file area.

During the second pass, the two sections of the file are written in the proper order in the output area starting at the lower limit of the original main-file area.

Neither the block length (number of sectors per block), nor the blocking factor (number of records per block), need be the same in the new file as in the old. However, the ratio of the old blocking factor to the old block length must not be greater than the ratio of the new blocking factor to the new block length.

$$\frac{\text{old blocking factor}}{\text{old block length}} \leq \frac{\text{new blocking factor}}{\text{new block length}}$$

The distribution index is rebuilt as the file is loaded during the second pass and can be written in the same area as before, or in a new area. The interval between entries in the index need not be the same as for the previous file.

The load audit option is available with this operation.

Tape Output: This is the same as for stripped-merge disk output.

Parameter Cards Required

ROUTINE
SYSTEMSPEC (optional*)
DISKDRIVES
EXITS (optional*)
INPMASTREC
MASTCONTRL
MASTBLOCK

$\left.\begin{array}{l}\text{DISTENTRY}\\ \text{LOADAUDIT}\\ \quad\text{(optional*)}\end{array}\right\}$ (if sequence-link field is retained)
TYPECSUNLD
UNLDMEDIA
INLABELS
OUTLABELS (if disk output)

* See *Parameter Cards* to find out when this card can be omitted from the source deck.

Input/Output Files

Disk Files

Object programs can be generated to operate on files in either or both 1311 and 1301 disk storage. For example, an input file can be on 1311 disk storage and the organized output file on the 1301. Any given file or group of associated files, however, must be contained entirely in one or the other of the kinds of disk unit. This applies to random master and trailer files and the work area used for disk output of nonhome records or addresses. For control-sequential files, it applies to the main-file area, the additions area, the distribution-index area, and the temporary area used for a two-pass CSUNLD.

All disk input and output areas used by these file-organization programs are defined by lower- and upper-limit disk addresses. The limits of the master-file area used by the random programs are specified in the MASTLIMITS parameter card when generating the random programs. The limits of all other disk input/output areas are specified at object time in IOCS RDLIN macro information cards. These cards (referred to as RDLIN cards) are used for all files regardless of whether they have associated disk header labels. The RDLIN cards required for each file are described later in this section.

Single- and Multiple-Area Files

Certain disk input and output files can be defined as multiple areas by more than one set of limits. Others, depending on the particular program and the size of the system, must be contained within one area and defined as such within a single set of limits.

A group of disk records can be defined with a single set of limits if the entire area is on-line at one time and if consecutive disk addresses are used throughout the area. A single exception to the requirement of consecutive addresses is the case of a labeled 1311 file extending from the end of one pack to the beginning of another. These programs will skip the 20 addresses normally on the last track and continue on the first track on the next disk pack. For example, if the limits of a file area are given as 010000-030000, the program would read from 010000 to 019979 and then automatically bypass the header-label track and skip to another disk drive to read 020000 to 030000.

Note: The last record of the last complete block in a disk pack can be, but need not be, an EOR trailer label.

The limits of a single-area file are given in one RDLIN card, regardless of the number of disk packs or modules used. A RDLIN card must be supplied for each area of a multiple-area file.

Input Files

The organized-file areas are used as input to the delete and unload programs. See *Output Files* for a description of these areas. The specifications of the input files for each of the loading and additions programs are as follows:

PASS1, PASS2, and CSLOAD. These programs can accommodate multiple-area input files, regardless of core-storage capacity. The program reads the first RDLIN card and processes the area that it defines. If the end of a pack or module is reached (with or without an EOR trailer label) before the upper limit, the program automatically continues to the next disk pack or module.

When an EOR trailer label is read and the upper limit has been reached, the program halts. If an EOR trailer label is read and neither the upper limit nor the end of a pack or module has been reached, the program also halts. The operator can supply the RDLIN card for the next area and, if necessary, place a new disk-pack on-line. Pressing START causes the program to read and process the new RDLIN card.

When an EOF trailer label is read or when the upper limit of an input area is reached before finding an EOR or EOF trailer label, the program proceeds to the end-of-job.

RNADD, CSADD, and TRADD. The specifications of disk files allowable as input to the three additions programs depend on the core-storage capacity of the system on which the program is run.

In a 4K system, input to RNADD and CSADD must be contained within one area and that area must be contained in a single disk pack or module. TRADD can accommodate a multiple-pack or module input file but it must be defined by a single set of limits as one area.

In an 8K or larger system, input files can be broken into multiple areas on as many disk packs or modules as necessary. They are processed as described for *PASS1, PASS2,* and *CSLOAD.*

Output Files

The following output files can be broken into multiple areas:

1. Non-home records written by PASS1, if the entire record is written in the disk work area (RECINWORK option).
2. The output of RNUNLD.
3. The output of CSUNLD, if all of the appended fields are stripped from the output records (stripped merge).

In the three cases mentioned here, where output files can consist of multiple areas, an EOR label is written and the program halts when the upper limit of the area defined is reached before all records are processed. The user can supply a new RDLIN card and, if necessary, change output disk packs. Pressing START causes the program to read and process the new RDLIN card.

All other disk output must be contained within one contiguously addressed area. These areas are:

1. The master-file area of the random routines.
2. The non-home work area written by PASS1, if only the addresses of non-homes are written (ADDINWORK option).
3. The trailer-file area produced by TRADD.
4. The main-file area of the control-sequential programs, whether produced by CSLOAD or CSUNLD.
5. The distribution-index area produced by CSLOAD or CSUNLD.

   Note: This area must be contained in one disk pack or module.

6. The additions area produced by CSADD.

Disk RDLIN Cards

A RDLIN card must be supplied for each disk area (1311 or 1301) to be operated upon by a program. These cards supply the control number(s) of the disk drive(s) or module(s) being used and the limits of the file area. If 1311 disk header labels are to be processed, the RDLIN cards also contain the label information to be used. The cards are punched as follows:

Columns	*Contents*
1-5	Area identification
6-10	blank
11-15	Disk drive(s) or module(s) on which the file area is located.
16-20	RDLIN

Note: Columns 21-54 are used only for labeled files in 1311 disk storage.

21-24	File-Retention Period
25-29	Creation Date
30-39	File Identification
40-44	File Serial Number
45-49	Pack Serial Number
50	blank
51-54	File Sequence Number
55-60	Lower Limit
61-66	Upper Limit

Area Identification. An entry identifying the disk area defined by the card. (Not to be confused with the file-identification field in the label portion of the card.)

The area-identification field is used by these programs to ensure that the RDLIN cards are read in the proper order. The entires for the various areas are:

INPUT = disk area(s) used as input to the loading and additions programs:

1. PASS1 and CSLOAD — The file to be loaded.
2. PASS2 — The INPUT cards used in PASS1 are used in PASS2 also, unless the RECINWORK, RECINCARD, or RECONTAPE non-home option is taken. See *WORKF*.
3. RNADD, TRADD, and CSADD — The area containing the records to be added.

MASTR = the organized master-file area of the random routines and the main-file area of the control-sequential routines.

Note: Although there are no trailer records, thus no possibility of master records in the ordinary sense, the main-file area of a control-sequential file is often referred to as the *master file* to distinguish it from the additions area.

Note. PASS1 is used to make additions to a file with unique addresses. When making additions, punch an A in column 10 of the MASTR RDLIN card to bypass the routine that clears the file area.

INDEX = distribution-index area

TRAIL = organized trailer-file area

ADDIT = control-sequential file additions area

WORKF = output area for:

1. PASS1 — non-home records or addresses. If WORKF is used in PASS1, the same card is used in PASS2.
2. RNUNLD – all disk output
3. CSUNLD – all disk output, except temporary area used during 2-pass unload (see *TEMPF*).
   The final reorganized area of a reorganized unload and the output area of a stripped unload are defined in the WORKF card.

TEMPF = temporary file area during 2-pass CSUNLD.

Disk Drives or Modules

If any portion of the file area is in a 1311 disk pack with addresses:

000000-019999, punch the drive control number in column 11.
020000-039999, punch the drive control number in column 12.
040000-059999, punch the drive control number in column 13.
060000-079999, punch the drive control number in column 14.
080000-099999, punch the drive control number in column 15.

The numbers punched are:

0 for the first drive in the system
2 for the second drive
4 for the third
6 for the fourth
8 for the fifth.

If any portion of the file area is in a 1301 module with addresses:

000000-199999, punch the module identification in column 11.
200000-399999, punch the module identification in column 12.
400000-599999, punch the module identification in column 13.
600000-799999, punch the module identification in column 14.
800000-999999, punch the module identification in column 15.

The characters punched are:

? (plus zero) for the first module in the system
B for the second module
D for the third
F for the fourth
H for the fifth

Note. These identification characters are not the same as the *alternate codes* used in the disk-control field (+, S, U, W, Y), nor are they the numeric portion of these characters (0, 2, 4, 6, 8), as used in the RDLIN cards read by IOCS-produced routines in user programs.

RDLIN. The entry *RDLIN* is derived from the IOCS RDLIN (read label information) macro instruction. To simplify programming, the same card is used by these routines for labeled and unlabeled files.

Label Information. The information punched in columns 21-54 is discussed under *Disk Labels.*

Address Limits. The lower limit punched in the INPUT RDLIN card is the address of the first sector in the first block in the area, regardless of its position within a track. The upper limit punched is the address of the first sector in the last block in the area.

In all RDLIN cards other than the INPUT card, the lower limit must be the address of the zero sector of the first track in the area. The upper limit must be the zero sector of the last track in the area. If disk labels are written, this upper limit address is adjusted by the program and the actual upper limit address written in the label is the address of the last sector in the last track in the area.

Disk Labels (1311 files only)

Files organized in 1311 disk storage can have disk header labels. If labels are specified, routines to process them are taken from the IOCS library. Header labels are *not* processed for 1301 files. If the load programs are to write 1311 disk header labels, the Disk Label Program (part of the 1440 and 1401-1311 disk utility program packages) must have been used to set up the initial header-label track. The IOCS publications describe the standard 1311 disk header labels and the processing performed by the IOCS routines.

The entries in parameter cards INLABELS and OUTLABELS determine whether header labels are to be processed and, if so, the extent of checking to be performed. The standard IOCS options are available. See *Label Checking* for label operations performed by each of the programs.

Input Header Labels

The header-label checking performed on input files consists of a comparison between certain fields in the RDLIN card and the corresponding fields in the header labels on the designated disk pack(s). If an equal comparison is made (on each pack designated), the file is accepted. If an equal comparison is not made; that is, if the label requested is not found, the program halts, allowing the user to replace the disk pack.

The comparison is made using either the file-identification field only, or each of the first six fields in the RDLIN card. When a full check on the header labels of a multiple-pack file area is performed, the pack serial number is checked on only the first pack. A one (1) is added to the file-sequence number for each subsequent pack.

Output Header Labels

The user chooses whether to have header labels written on his 1311 output files. If they are written at all, they must be written for all associated disk output areas. Thus, if labels are written for the master-file area, they must also be written for the trailer file and for the disk work area used for non-home records during PASS1.

When writing header labels for output files, the user must insert a date card in the object deck. The date card is punched as follows:

Columns	*Contents*
1-3	082
4, 5	05
6	Word separator (0-5-8 punch)
7, 8	Year (for example, 63)
9-11	Day of year (001 represents Jan. 1, 365 represents Dec. 31)

The date is used by the programs to ensure that there are no unexpired files with limits overlapping those in the header label that is to be written.

If there are no unexpired files within the area, the label is written with each of the fields in the RDLIN card copied into the corresponding location in the label. In the event of a multiple-pack output area, the process is repeated for each pack, using the same RDLIN card. The existing pack serial number is retained in the labels on subsequent packs, and the file sequence number is increased by one (1) for each pack. The limits written in the header label of each pack are the correct limits for the section contained within the pack.

Cylinder Overflow

When disk files are processed, certain block lengths are such that a block *overflows* from one cylinder to the next. For example, if 3-sector blocks are processed in an area where the first block begins in sector 000000; the last complete block in the first cylinder ends in sector:

1311	1301
000197	000797

The next block begins in:

1311	1301
000198	000798

and overflows to:

1311	1301
000200	000800

Input to these programs can begin with any sector on a track. Therefore, depending on the starting address, cylinder overflow can occur in input files with any block length other than single-sector blocks.

Output files must begin at the zero sector of the first track in the area. Therefore, cylinder overflow can occur in output files, only if the block length is not evenly divisible into 20.

The programming necessary to process overflow blocks is generated in these programs if:

1. input files have more than one sector per block. In this case, overflow blocks are checked for blank record locations. If records are present, they are processed; if not, the block is skipped.
2. output files have a block length (sectors per block) that is not evenly divisible into 20.

Cylinder overflow must not occur (on input or output) from one disk pack or module to the next. The disk block that would have caused such overflow must begin at the zero sector of the first track on the next pack. In the case of the random programs, it is up to the user's conversion routine to ensure that pack overflow does not occur and that the first block in subsequent packs begins in the zero sector of the first track. The control-sequential programs automatically skip to the zero sector of the next pack or module.

Address Structure and Mode of Operation

These programs perform all operations in the move mode (7-bit coding). All input files, therefore, must be in the move mode. The random master-file area must be in the move mode before the initial run of PASS1 is performed. At least the first record location (dummy record) in the following output-file areas must be cleared to blanks in the move mode before performing the initial run of the program indicated:

1. main-file area written by CSLOAD
2. additions area written by CSADD
3. trailer area written by TRADD.

Disk addresses must not be repeated on any track used by these programs. All disk addresses normally associated with a track must be present. Addresses must be in sequence within a disk block, but the blocks can be arranged in any order found most suitable for the file. For example, in a file with 5-sector blocks, the sector addresses on the first track could be written in the order: 0-4, 10-14, 5-9, 15-19. This kind of sequencing can often greatly reduce rotational delay time.

Two disk packs or modules containing the following related file areas must not have the same range of addresses:

1. random master and trailer file areas.
2. control-sequential main file, additions area, temporary area (for 2-pass unload), and distribution index area.

Disk areas used as input to the load and additions programs and output from the unload programs can have the same addresses as the organized-file areas.

When the work tracks preceding the main-file areas are used for program overlays, they are written in the load mode. These tracks are not cleared at end-of-job and therefore remain in load mode after completion of of the program run. The first one or two tracks preceding the random master file (used for the availability table) are written in the move mode. The only requirement of concern to the user when initially preparing his disk areas is that these tracks have the proper addresses. They can be in either the move or load mode and need not be cleared to blanks.

Block Length

When disk files are processed, it is generally true that the largest possible block length (sectors read and written in one operation) results in the fastest processing. In a chained random file, the longer the block, the better the chance of storing non-home records in the *home* block. In a control-sequential file, the main advantage is the fact that rotational delay time need be lost only once per block. There are exceptions to this rule:

1. In a random file with no synonyms, the optimum block length is that number of sectors required to hold a single record.
2. If the time required to process all of the records in a control-sequential block slightly exceeds a track revolution, it may be better to shorten the block, again minimizing the rotational delay.

The block length chosen for a given file depends on many factors other than minimum read/write time. For example, a maximum of ten records can be written per block in the random files organized by these programs. Therefore, the block length used would never exceed that number of sectors required to hold ten records. Storage utilization is another factor that must be weighed and balanced with time savings. A particular block length might be used that is less than the maximum possible if that block length results in a minimum of wasted disk-storage space.

The one overriding factor that always sets an upper limit on block length is the core storage available to read and write the block. The amount of core storage remaining for I/O areas depends on the amount required for the actual program instructions, constants, work areas, etc. Because the block length of a file is normally the same for all programs that operate on the file (this must be true for the file-organization programs), it can be no greater than that used in the largest of those programs.

Block-Length Limits Set by File-Organization Programs

The file-organization object programs are generated to fit the parameters of specific files. Only those operations actually called for by the user are included in the programs. There are no unnecessary routines included to take care of several possible formats or contingencies. During operation of the programs, certain routines are written in the work-area tracks preceding the master-file area. These routines are read and executed whenever needed. The amount of core storage required for the generated object programs can thus be kept at that minimum necessary to do the job. The remaining core storage can be used for input/output areas and the user's subroutines.

Some factors affecting the size of the object programs are:

1. the input media.
2. whether the number of sectors per block is such that cylinder overflow is possible and, therefore, whether the routines necessary to handle it must be generated.
3. whether there is one record per block in the organized file, or more than one.
4. the choice of available options, such as:
   a. kind of output for non-home records during PASS1 and the output requested for the unload programs.
   b. the load audit or print options.
5. the length of control fields in the records and the number of subfields comprising the control data.
6. whether chaining is required in a random file (chaining is required only if there are duplicate addresses).
7. whether trailer records are used.
8. the special features used.

   Note. The use of certain special features, such as the advanced programming feature, decreases the amount of programming needed and, therefore, leaves more room for data records.

The additions programs, RNADD and CSADD, are the largest of the file-organization programs generated from any given set of parameters. Except in special cases involving trailer records, these additions programs determine the maximum permissible block lengths. Because this is the case, it is usually true that, if a given input block and organized master- or main-file block fit the additions programs, they are permissible for the other programs also.

Figure 13 shows the sizes of RNADD and CSADD object programs that were generated with a choice of parameters chosen to give minimum and maximum size programs. The values shown in the figure can be used as a guide in determining whether a proposed block length is well within the allowable range or if trial programs should be generated to test the exact combination of parameters and subroutines. The increased core-storage requirements in the 8K or larger systems is due to the fact that additional functions are performed in the larger systems and portions of programs that are overlayed from disk storage in the 4K systems are kept in core storage in the 8K and larger systems. The largest single factor affecting the size of these programs is the room required to handle multiple-record blocks. The figure shows the maximum sizes of RNADD and CSADD when generated to process unblocked (input and output) files.

The core storage remaining must be used for all subroutines supplied by the user (including conversion routines, exit routines, and card-assembly routines in excess of 190 positions) and for the input/output areas shown in Figure 13.

Note. Card-assembly routines supplied by the user replace a 190-position routine normally generated from the library routines.

RNADD is not used if there are no synonyms in a random file. CSADD is not used if all additions are made to the end of a control-sequential file. In these cases, PASS1 and CSLOAD set the limits on block length.

The maximum block length of trailer files is set by TRADD. The block lengths possible for disk or tape output of the unload programs are set by RNUNLD and CSUNLD.

Example

Assume that the following file is to be organized as a random file with trailers in a 4K 1440-1311 system.

Disk Input.

131-character input record (130 character data record + terminal record mark) 400-character input block (3 records in 4 sectors)

Disk Output.

145-character master record (7-digit trailer address + 130-character data record +7-digit overflow address + record mark)

300-character master block (2 records in 3 sectors)

To determine whether these files will fit, refer to Figure 13:

Maximum RNADD object program	= 3290 positions
Input area	= 400 positions
Master-file block	= 300 positions
Master record	= 145 positions
Total core storage required	= 4135 positions

Thus, we see that this file would not fit in the *maximum* case. But it must be emphasized that the maximum case is an *unusual* case. Several factors can shorten the object program. For example, if the object machine has the indexing-and-store-address-register feature, 80 to 100 positions are saved. Note also that the maximum case assumes 30-character control fields, broken into three subfields. In this case it would be necessary to perform a trial generation of the RNADD program to see if the block will fit.

OBJECT PROGRAM	OBJECT SYSTEM												INPUT/OUTPUT AREAS USED
	1401 - 1311				1440 - 1311				1440 - 1301		1460 - 1301		
	4K		8-16K		4K		8-16K		8-16K		8-16K		
	Min.	Max.	Min.	Max.	Min.	Max.	Min.	Max.	Min.	Max.	Min.	Max.	
RNADD (blocked)	2510	3280	2420	5140	2530	3290	2450	5160	3980	5320	3960	5300	Input Area
RNADD (unblocked)		2880		4730		2890		4750		4880		4860	Master-File Block
													Master Record
PASS1 (blocked)		2450		2450		2480		2480		2580		2550	Input Area Master-File Block
RNUNLD (blocked)		2320		2320		2340		2340		2450		2430	Organized-File Block (master or trailer) Output Block (disk or tape)
CSADD (blocked)	2260	3270	2650	4270	2290	3280	2680	4280	2790	4440	2760	4430	Input Area
CSADD (unblocked)		2960		3120		2970		3130		3250		3240	Main-File Block
													Additions-File Block
CSLOAD (blocked)		2870		2930		2910		2970		3050		3010	Input Area Main-File Block
CSUNLD (blocked)													Main-File Block
Stripped Merge		2610		2590		2610		2590		2690		2690	Additions-File Block
Relocated Merge		2940		2910		2950		2920		3050		3040	Output-File Block (disk or tape)
Two-Pass Merge		2910		2880		2920		2890		3050		3040	
TRADD (blocked)		3210		4250		3210		4250		4350		4350	Input Area The larger of: Master-File Block Trailer-File Block

Figure 13. Significant Generated Program Sizes

Tape Files

Magnetic tape can be used as input to the load and additions programs and for output of the unload programs. Output of non-home records by PASS1 can also be on tape. All tape files must consist of fixed-length records. There can be as many as 70 records per block. As with disk input/output blocks, the maximum allowable block length depends on the core storage available (see *Block Length* under *Disk Files*). Incomplete output tape blocks are always padded with blanks. The padding character on tape input is specified in the INPUTMEDIA card.

When an end-of-reel condition is reached and an alternate tape unit is specified (by operand ALTTAPE in SYSTEMSPEC card), the change to the alternate unit is made automatically. If there is no alternate unit, a halt occurs to allow the operator to change the reel. These programs perform all tape operations on the following units:

1401 and 1460 Programs:

Input:	First unit – 2
	Alternate unit – 3
Output:	First unit – 4
	Alternate unit – 5

Dump Option (Output of Error Blocks) – 6

1440 Programs:

Input:	First unit – 1
	Alternate unit – 2
Output:	First unit – 1
	Alternate unit – 2

Dump Option (Output of Error Blocks) – 2

The following programs accept input from magnetic tape:

PASS1, PASS2, and CSLOAD. Input to these programs can be either a single- or multiple-reel file.

RNADD, TRADD, and CSADD. In a 4K system, input to these programs must be single-reel tape files. In an 8K or larger system, either a single- or multiple-reel file can be used.

The PASS1, RNUNLD, and CSUNLD programs can write single- or multiple-reel tape output. PASS1 is the only program that can have tape input and output.

Note. In 1440 programs, if tape input to PASS1 and the RECONTAPE nonhome option is specified, the nonhome records are written on unit 2. If card or disk input to PASS1, the nonhome records are written on unit 1. Do not make the mistake of calling for more tape operations than are possible in the object system. For example, in a single drive system, tape input to PASS1 would preclude the possibility of either the RECONTAPE nonhome option or the DUMP tape-error option.

Tape Labels

All tape processing in these programs is performed by IOCS routines. Files can be labeled or unlabeled and, if labeled, can have:

1. Standard 120-character labels (Type A)
2. Standard 80-character labels (Type B)
3. Standard 84-character labels (Type C)
4. Nonstandard labels

Either of the following publications may be referred to for complete details about the formats of the three types of standard tape label and the processing techniques used:

1. Input/Output Control System (on Disk) for IBM 1401/1640: Specifications, C24-1489.
2. Input/Output Control System for IBM 1440: Specifications, C24-3011.

The entries in the INPUTMEDIA and UNLDMEDIA cards determine whether header and trailer labels are to be processed and, if so, the extent of checking to be performed.

Exits are provided in IOCS for the user to insert his own routines for checking nonstandard labels.

Note. If non-homes are written on tape, they are in the following format:

If tape input to PASS1 has standard labels and the RECONTAPE option is chosen on the NON-HOMES card, the non-homes output tape has type B (80-character) header label(s) and corresponding trailer label(s). The header label(s) contains PASS2NONHM in the file-identification field and a retention period of one (1) day (-001b). The trailer label contains a label identifier (1EORb or 1EOFb) and the block count. A tape mark does not follow the header label.

If tape input to PASS1 has nonstandard labels or no labels, *or* if input was from cards or disk, the non-homes output tape has the same blocking factor as the master file, without labels. A tape mark is written for an end-of-reel or end-of-file condition.

Standard Input Header Labels

With the exception of PASS2, if standard labels (A, B, or C) are specified in the INPUTMEDIA card, a tape RDLIN card must be inserted in the program deck at object time. If standard label tape is input to PASS2, it is not necessary to provide a RDLIN card.

The standard header label checking performed on input tape files consists of a comparison between certain fields in the RDLIN card and the corresponding fields in the header label on the tape files. The comparison is made using either the file-identification field only or each of the fields specified in the tape RDLIN card. (In complete label checking of multi-reel files, a one (1) is added to the reel sequence number for each subsequent reel.)

PASS2 automatically supplies the RDLIN information for checking standard input tape header labels. The file identification field, only, is checked. If unequal comparison results in standard label checking, the program halts, allowing the user to replace the tape reel, if desired.

Standard Output Header Labels

The labeling of the tape output from PASS1 is determined by the input. For the CSUNLD and RNUNLD routines, an UNLDMEDIA card specifies the kind of labels to be written on the output file. If standard labels (A, B, or C) are specified, a tape RDLIN card must be inserted in the program deck at object time. Information from the RDLIN card is written on the header label on all reels of a tape file. For multi-reel files, the reel sequence number is increased by one (1) for each subsequent header label.

Before a standard header label on an output tape is written, the effective dates in the old header are checked to ensure that the data on the output tape is no longer active and may be destroyed. If the retention period has not elapsed, a programmed halt occurs so that the tape may be changed (if desired). When standard labels are processed, a date card must be loaded into storage at object time so that the retention cycle checking can be performed. This card is punched as for 1311 disk output checking (see *Disk Output Header Labels*).

Standard Input Trailer Labels

If standard labels (A, B, or C) are specified in the INPUTMEDIA card, IOCS checks and writes corresponding trailer labels. Two fields on the standard trailer label are checked. The labels are checked for an end-of-reel or end-of-file condition. Also, a count of the number of blocks processed, which was accumulated automatically by IOCS, is compared with the block count recorded on the trailer label. If an error is detected in checking, a programmed halt occurs.

Standard Output Trailer Labels

Trailer labels are written for PASS1 output if the INPUTMEDIA card indicated standard label input. If standard labels are specified in the UNLDMEDIA card, trailer labels are written for tape output of CSUNLD and RNUNLD. An end-of-reel (EOR) or end-of-file (EOF) indication is written. IOCS writes the accumulated block count in the block-count field of the trailer label. Blanks are written in the remaining fields.

Nonstandard Input Labels

If nonstandard labels are specified in the INPUTMEDIA card, the programmer must provide his own symbolic routines (which are placed in the source deck after the parameter cards), to read and check header and trailer labels. IOCS exits 6 and 7 are generated automatically in the programs for this purpose.

Exit 7 permits reading and checking of nonstandard input header labels. The user, in his routine, must provide his own area for entering the header information and must program to read and check the information. The first instruction of the user's routine must be labeled TEXIT7 and the last instruction must return control to the main program by branching to IOCSRE.

Exit 6 permits reading and checking nonstandard input trailer labels. The user must provide, in his routine, his own area for entering the trailer information, and he must program to read and check the information. The first instruction of the user's routine must be labeled TEXIT6. Control is returned to the main program by branching to IOCSRE on an end-of-reel condition and to TAPEND on end-of-file.

The block count is accumulated automatically and is available to the user whenever exit 6 is used. The count is in a 5-position field identified in the assembly listing by the following label:

1401/1460 Package	*1440 Package*
IOC2BK	IOC1BK

Nonstandard Output Labels

If nonstandard labels are specified in the UNLDMEDIA card, the programmer must provide his own symbolic routines (which are placed in the source deck after the parameter cards) to build and write header and trailer labels. IOCS exits 2 and 5 are automatically generated in the programs for this purpose.

Exit 5 is used to build and write nonstandard output header labels and to read and check the old header on an output tape. The user must program the checking procedure for the old header. His routine must provide its own area for building the new header information and the instructions to write the new header on tape. The first instruction of the user's routine must be labeled TEXIT5, and the last instruction must return control to the main program by branching to IOCSRE.

Exit 2 is used to build and write nonstandard output trailers. The user must provide his own area for building the trailer information and must program to write the trailer label on tape. IOCS provides the user with an indication of whether the trailer is to be written for an EOR (end-of-reel) or EOJ (end-of-job) condition. If EOR, there is no word mark in a location labeled TPCLSW. If EOJ, TPCLSW contains a word mark. The first instruction of the user's routine must be labeled TEXIT2, and the last instruction must return control to the main program by branching to IOCSRE.

The block count is accumulated automatically and is available to the user whenever exit 2 is used. The count is in a 5-position field identified in the assembly listing by the following label:

1401/1460 Package	*1440 Package*
IOC4BK	IOC1BK, or IOC2BK, for nonhome count if tape input *and* output.

The user's tape exit routines must follow immediately after the parameter cards in the source deck and must be followed by a LTORG card with an asterisk in the operand field.

Tape RDLIN Cards

A tape RDLIN card must be placed in the program deck at object time if a tape file with standard labels is to be processed by the program. The RDLIN card contains an identification field in columns 1-5 to indicate whether the tape file is an input or output file. Other data in the card must correspond to the fields of the particular standard label that will be checked or written on the tape file.

The reel sequence number is automatically increased by one (1) for every reel after the first in a multi-reel file. All other parts of the header label should be identical on all reels of an input tape file and will be written identically on an output file.

A RDLIN card does not have to be supplied for tape output of PASS1 or tape input to PASS2. These programs automatically supply RDLIN information for reading and checking standard labels.

The format of the RDLIN card associated with each of the three kinds of standard header labels is as follows:

RDLIN card for Type A (120-character) standard header label (fields 9-19 are *not* processed):

Columns	*Contents*	*Header-Label Field Number*
1-5	TAPEI – (for tape input to PASS1, RNADD, TRADD, CSLOAD and CSADD) TAPEO – (for tape output of CSUNLD and RNUNLD)	
16-20	RDLIN	
21-24	Retention Period	2
25-29	Creation Date	3
30-39	File Identification	4
40-44	File Serial Number	5
45-48	Reel Sequence Number	7

RDLIN card for Type B (80-character) standard header label:

Columns	*Contents*	*Header-Label Field Number*
1-5	TAPEI – (for tape input to PASS1, RNADD, TRADD, CSLOAD, and CSADD) TAPEO – (for tape output of CSUNLD and RNUNLD)	
16-20	RDLIN	
21-25	File Serial Number	3
26	– (minus sign)	4
27-29	Reel Sequence Number	4
30	blank	4
31-40	File Identification	5
41-45	Creation Date	6
46	– (minus sign)	7
47-49	Retention Period	7
50	blank	7

RDLIN card for Type C (84-character) standard header label:

Columns	*Contents*	*Header-Label Field Number*
1-5	TAPEI – (for tape input to PASS1, RNADD, TRADD, CSLOAD, and CSADD) TAPEO – (for tape output of CSUNLD and RNUNLD)	
16-20	RDLIN	
21	blank	3
22-26	File Serial Number	3
27	blank	4
28-31	Reel Sequence Number	4
32	blank	4
33-34	Creation Date: Year	5
35	blank	5
36-38	Creation Date: Day	5
39-41	blank	6
42-44	Retention Period	6
45-50	Label Information: blank Density Character Coding Checksum Block Sequence Checkpoint Record	7
51-68	File Identification	8

Label Checking

The following outlines the header-label operations performed by each of the programs if called for in the parameter cards. Where nonstandard labels are used, an exit is provided to the user's routine to perform the operation.

PASS1

1. Checks header labels of disk or tape input file.
2. Writes the header labels for the disk master-file area and the work tracks preceding it; or, checks the header labels if making additions to a uniquely addressed file.
3. Writes the header labels for the disk work areas or tape output file used for non-home records or addresses. A tape output file is labeled only if there was tape input with standard labels,

PASS2

1. Checks the header labels of the disk master-file area.
2. Checks the header labels of all disk input areas and/or work areas or tape input files used for non-home records or addresses. For tape input, PASS2 checks labels only if input to PASS1 was tape with standard labels. Only the file-identification field (PASS2NONHM) is checked.

TRADD

1. Checks header labels of disk input area or tape input file and the master-file area.
2. If performing initial loading of disk trailer records, a header label is written for the trailer area.
3. If adding trailer records, the header label of the disk trailer area is checked.

RNDEL1, RNDEL2, and TRDEL

1. Checks header labels of disk master file and trailer file (if trailers).

RNUNLD

1. Checks header labels on disk master or trailer area.
2. Writes header labels for disk output-file areas or tape output file.

CSLOAD

1. Checks header labels on disk input-file areas or input tape file.
2. If performing initial loading in area, writes header labels for the disk main-file area, for the distribution index area, and the three work tracks preceding the main-file area.
3. If adding records to the end of the disk main-file area, the header labels of the disk main-file area and the distribution index area are checked.

CSADD

1. Checks header labels on disk input-file areas or input tape file.
2. Checks header labels of disk main-file area and distribution index area.
3. If first additions to disk file, writes a header label for the additions area.
4. If adding records to an established additions area, the header label of the area is checked.

CSDEL1

1. Checks header labels of the disk main-file area and the additions area (if separate additions area).
2. Checks header label of distribution-index area.

CSDEL2

1. Checks header labels of the disk main-file area and the additions area (if separate additions area).

CSUNLD

1. Checks header labels of disk main-file area and additions area.
2. Writes header labels for output disk areas or output tape file.
3. Writes header labels for distribution index and work tracks, if reorganized output.

Card Files

Card Input to Load and Additions Programs

If input to the load and additions program is from cards, the user can either supply his own subroutine to assemble the disk records, or he can let the file-organization program assemble them.

Regardless of whether the user supplies a subroutine to assemble disk records, the last data card must be followed by a card with the entry, ENDCD, punched in columns 1-5.

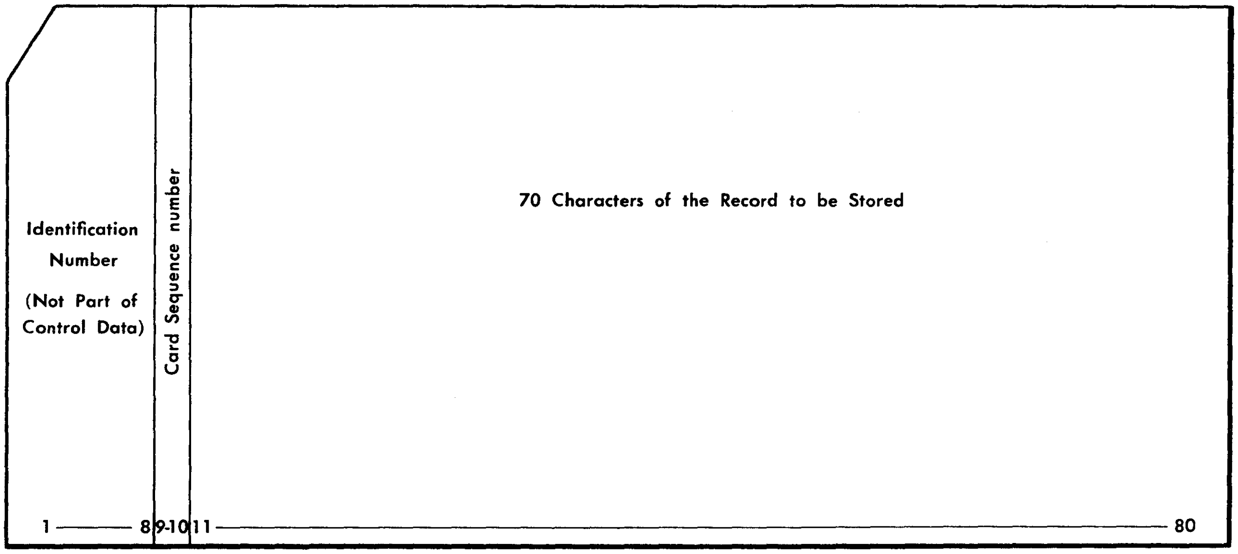

Identification Number (Not Part of Control Data)
Card Sequence number
70 Characters of the Record to be Stored
1 — 8
9-10
11 — 80

Figure 14. Format of the Input Card

Records Assembled by File-Organization Programs

If the file organization programs are to do the assembling, the records must be in the format shown in Figure 14. Each card within a record must contain the same identification number in columns 1-8. Note that this identification number is not considered to be the control data of the record. The control data is contained within the data portion of the record. Columns 1-8 can contain any value (including all blanks) as long as the same value is repeated in each card of a record. The number need not be unique for each record.

All cards composing one record must have sequence numbers beginning with 01 punched in columns 9 and 10.

The actual data that makes up the record is punched in columns 11-80. Thus each card can contain 70 characters of a record. A record can consist of up to 98 cards. All records within a particular file must be the same length.

Both the loading and additions programs examine the identification and sequence numbers to ensure that the correct record is written in the proper sequence. If a record contains an incorrect number of cards, an incorrect identification number, or if the cards of a record are out of sequence, the program prints an error message indicating the identification and sequence number of the erroneous card. The program then halts or bypasses the record, depending upon the user's choice of the operands HALT or BYPASS in the INPUTMEDIA card.

Records Assembled by User's Subroutine

If the user supplies a subroutine to assemble the disk records, a core-storage work area is generated by means of the following statements taken from the file-organization library routines:

```
RECBEG        DA       IxU
RECEND                 1,U
```

U equals the length of the user's input-data record and is taken from the value punched in the INPMASTREC card. The work area has no word marks other than the one in the high-order position.

The following card-read routine is generated in each of the load and additions programs. The routine reads each input card and checks for the last card, which must have ENDCD in columns 1-5.

```
REAERR           .7                             (Note 1)
INPUT        R                                  (Note 2)
             BIN        REAERR, ?
             SS                                 (Note 3)
             C          5, KENDCD
             C                                  (Note 4)
             C
             C
             C
             BE       END
             B      ASMBLE
             DCW        @E@
             DCW        @N@
             DCW        @D@
             DCW        @C@
KENDCD       DCW        @D@
PROCES       . . .      . . . .
             . . .      . . . .
             . . .      . . . .
```

Note 1. With the exception of those halts generated from IOCS, all halts in these object programs are 2-character halts, with the alphabetic character appearing in the A-register for halt identification.

Note 2. When reading from a 1442, this instruction includes the operand 1, 1.

Note 3. This instruction is not included if reading from a Model 1 of the 1442 that does not have the selective stacker feature. If two stackers are available on a 1442, the instruction is included with the operand 2. When reading from a 1402, the instruction is always included with the operand 1.

Note 4. This compare operation is chained to allow word marks in any position of the read-in area. The following explains how this compare works.

There is one latch used for equal-unequal compare. This latch being ON indicates an equal condition, while unequal is indicated by the latch being OFF. The equal-unequal latch is turned on at I-2 time of a compare operation and then left on if the characters compared are equal, or turned off if they are unequal. This compare is during the B-cycle.

The full-chained compare used here will turn the latch on during the first compare only, as this is the only time the machine will get to I-2. If any of the characters compared are unequal, the latch will be turned off and remain off. If all five characters are equal, the latch will remain on. Therefore, a complete compare is accomplished.

The program reads a card into core-storage positions 1-80. (Position 81 contains a group-mark with word-mark.) If no error occurs, the routine branches to ASMBLE, which must be the label of the first instruction in the user's assembly routine. The assembly routine can then select any fields from 1-80 and move them to the defined area RECBEG. To facilitate field selection, the assembly routine can set word marks in the read-in area. No additional word marks may be set in the area labeled RECBEG. The assembly routine then determines whether more cards are required for the record. On the basis of this decision, it either branches to INPUT or, if the entire record has been assembled, to PROCES, thus returning control to the file-organization program.

In the event of a read error (REAERR), the program halts. The user can correct the cause of the error and press START to continue.

The user's assembly routine, in symbolic form, is loaded with the parameter cards. It is suggested that all labels other than ASMBLE be started with ZZ to avoid multiply defined symbols.

Note 1. The user's assembly routine must not use the MOVE RECORD (MRCM) instruction to move data from the input area to RECBEG. This instruction moves the group mark along with the data, and this position was not defined in the DA statement shown above.

Note 2. When a record is displaced by RNADD, it is held temporarily in the area labeled RECBEG. The area is not cleared before branching to process the next record.

Card Output from the Unload Programs

The card output produced by RNUNLD and CSUNLD is discussed under the separate descriptions of those programs.

Program Exits

Exits are provided in each of the file-organization object programs. With one exception explained below (EXIT3), these exits are used to branch to the user's subroutines. If tape input or output files are used with nonstandard labels, exits are automatically generated by IOCS. These exits are branches to routines (referred to as TEXIT routines) supplied by the user to process the nonstandard labels. See *Tape Labels* for a discussion of the TEXIT routines.

In addition to the nonstandard-label routine exits, three exits can be generated from the file organization routines. These are optional exits, called for in the EXITS parameter card by the parameters: EXIT1, EXIT2, and EXIT3.

Exit 3 is available in all object programs. This exit is different from the others in that it does not cause a branch to a user subroutine. Instead, the program reads another card and branches to 001 at end-of-job. In this way, these programs can read in, and branch to, the first instruction of another loader program.

Each of the other two exits is generated as a branch to a user subroutine. The user supplies these subroutines in symbolic source language and they are assembled along with the file-organization object program. The user's routines are identified by the label (EXIT1 or EXIT2) of the first instruction in each. The EXIT1 routine returns control to the main program by branching to ENTRY1; the EXIT2 routine returns to ENTRY2.

The following points must be considered when writing all user subroutines:

1. During operation of these programs, portions of the programs are written in the reserved tracks preceding the master-file area and are overlaid by other portions. Therefore, any word marks associated with group marks in the user's routines must be set and cleared by the user's routine.
2. When TEXIT routines are used, they must be placed immediately after the parameter cards and before any other subroutines. They must be followed by a literal origin (LTORG) card with an asterisk in column 21. With that single exception, the user's routines must not include any of the following statements:

   EX — Execute
   INCLD — Include
   LTORG — Literal Origin
   XFR — Transfer
   END — End of Assembly (except for the END START card after the last card in the source deck)
3. If a subroutine uses index registers, it must first save the contents of the registers and restore them before returning control to the main program.

4. The first card in the first symbolic subroutine supplied by the user must not be a comments card.

The following explains the points at which exits 1 and 2 occur in the object programs. Figure 15 lists certain labels that may be useful in the user's subroutines. See *Block Length* under *Input/Output* Files for a discussion of the core-storage space available for subroutines.

PASS1

Exit 1 occurs after assembling the input record and before extracting the control data for conversion. This exit is not available if the user supplies an assembly routine for card input.

Exit 2 occurs after writing the output record and before performing the audit print. The exit does not occur after processing non-home records.

PASS2

Exit 1 occurs after assembling the input record and before extracting the control data for conversion.

Exit 2 occurs after writing the output record and before performing the audit print.

RNADD

Exit 1 occurs after assembling the input record and before extracting the control data for conversion.

Exit 2 occurs after writing the output record and before performing the audit print.

TRADD

Exit 1 occurs after assembling the input record and before extracting the control data for conversion. This exit is not available if the user supplies an assembly routine for card input.

Exit 2 — not available.

RNDEL1

Exit 1 — not available.

Exit 2 occurs after reading the master record to be deleted or tagged and before performing the audit print. The record has not been deleted or tagged but has been verified as the correct record.

RNDEL2

Exit 1 — not available.

Exit 2 occurs each time a tagged master record is found. The record has not been deleted but has been verified as a tagged record. The program does not recheck for the tag zone after the exit. There is no exit after reading the tagged trailer records.

TRDEL

Exit 1 — not available.

Exit 2 — occurs after reading the trailer record to be deleted and before performing the audit print. The record has not been deleted but has been verified as the correct record.

RNUNLD

Exit 1 occurs after reading each input record (master or trailer) and before processing the record.

Exit 2 — not available.

CSLOAD

Exit 1 occurs after assembling the input record and before checking the record for sequence.

Exit 2 occurs after moving the record to the output area and before performing the audit print. The output record has not been written.

CSADD

Exit 1 occurs after assembling the input record and before checking the record for sequence.

Exit 2 occurs after writing the new record and after establishing the sequence linkage, but before the audit print.

CSDEL1

Exit 1 — not available.

Exit 2 occurs after reading the record that is to be deleted or tagged and before performing the audit print. The record has not been deleted or tagged but has been verified as the correct record.

CSDEL2

Exit 1 — not available.

Exit 2 occurs each time a tagged record is found. The record has not been deleted but has been verified as a tagged record. The program does not recheck for the tag zone after the exit.

CSUNLD

Exit 1 – available only in a 2-pass unload. The exit occurs during the first pass, after moving the input record to the output area, but before writing the output block in the work area.

Exit 2 – available with all three kinds of output. The exit occurs after moving the record to the output area and before the audit print. The output block has not been written. In a 2-pass program, this occurs during the second pass, before writing the record from the merged file back into the original file area.

Note. The DMPADR labels in Figure 15 are used to modify the record in the output area. This can be done in Exit 1 of a 2-pass program or in Exit 2 of a relocate or strip program. No label is provided to reach the output record during the second pass of a 2-pass program.

Label	Programs and Conditions for Presence of Label	Core-Storage Field or Area Referenced
RECEND	All load and additions programs and RNUNLD.	In the load and additions programs, this label refers to the last (low-order) position of the input record. If a terminal record mark is present on input, it has been stripped off. No overflow-address, sequence-link, availability indicator, or output record mark has been appended. In RNUNLD, this is the last position of the input disk record. Thus, it is the terminal record mark, if RMIN. If NORMIN, it is the availability-indicator position of a master record or the last data character of a trailer record.
RECBEG	All load and additions programs, if card input.	These three labels are used to refer to the first (high-order) position of the input record. If new input, this is the first position in the data record. No sequence-link or trailer-address fields have been appended. If reorganized disk or tape input, the label refers to the tag-zone position appended to the front of the input record.
INAREA	All load and additions programs, if disk input.	
TAPEIN	All load and additions programs, if tape input.	
PRINT	Defined in all programs.	In all programs, a DA statement is used to define the print area. This label refers to the first (high-order) position of the area. The print area always begins in position 201.
CONTD	Defined in all random programs except RNUNLD with no converted-address output.	The control data of each record is extracted (before exit 2) in the order defined in the MASTCONTRL card and placed in this field. The label refers to the last (low-order) position of the field.
CONV1	All random programs except RNUNLD with no converted-address output.	This field contains the converted disk address of the record being processed. The address is available during exit 2.
WORKN	PASS2 and RNADD	This is the label of a 6- or 7-digit field that contains the actual disk address used to write the record. The address is available during exit 2.
FILLOC	CSDEL1, if blocked file and no index registers.	This is the label of a three-position field that contains the core-storage address of the last (low-order) position of the sequence-link field of the record being deleted or tagged.
CHEK1+6	CSDEL2, if blocked file and no index registers.	This is the character-adjusted label of the B-address of an instruction. The B-address is the core-storage address of the last (low-order) position of the sequence-link field of the record being deleted.
DATA	CSDEL1 and CSDEL2, if blocked file and index registers, or if unblocked file.	This is the label of the last (low-order) position of the data record being deleted. Note: This is the last data character, not the sequence-link field.
CL2+6	RNDEL1, if delete. RNDEL2, always.	This is the character-adjusted label of the B-address of an instruction. The B-address is the core-storage address of the availability-indicator position of the master record being deleted.
TAG+6	RNDEL1, if tag.	This is the character-adjusted label of the B-address of an instruction. The B-address is the core-storage address of the availability-indicator position of the master record being tagged.
CLEART+6	TRDEL	This is the character-adjusted label of the B-address of an instruction. The B-address is the core-storage address of the last (low-order) position in the trailer record being deleted.
DMPADR+3	CSUNLD- Stripped disk or tape output only, with NORMOUT.	This is the character-adjusted label of the B-address of an instruction. The B-address is the core-storage address of the last character in the record being unloaded.
DMPADR-4	CSUNLD- Two-pass program, if NORMOUT. Relocate program, if NORMOUT. Strip program, if RMOUT (not for card output).	This is the character-adjusted label of the B-address of an instruction. In either a two-pass or relocating reorganization program, the B-address is the core-storage address of the last position in the sequence-link field of the record being processed. In a program that produces stripped output, the address is that of the record mark.
DMPADR-5	CSUNLD Two-pass program or relocate program, only if RMOUT.	This is the character-adjusted label of the B-address of an instruction. The B-address is the core-storage address of the terminal record mark.

Figure 15. Labels Used by Exit Routines

The disk file-organization routines packages as supplied by IBM include special prephases for the corresponding Autocoder processor programs and 17 library routines. The object programs are generated from these library routines by means of an Autocoder assembly run. The file-organization prephase of Autocoder reads a series of parameter cards punched by the user and, from these, builds a parameter table. The macro generator of Autocoder then uses this table to select and tailor a symbolic program from the file-organization routines and from certain IOCS library routines. Any symbolic subroutines supplied by the user are then read and assembled along with the file-organization program. The output is an assembled object deck, ready to be used to operate on the file for which it was generated.

Although these programs are produced by the macro generator portion of the Autocoder, it must be understood that they are not macros in the conventional sense. They cannot be used as subroutines with other programs, nor can more than one program be generated and used concurrently during any particular run.

The parameter cards required for these programs are shown in Figure 17. The cards are to be punched in the regular Autocoder format, with the label in columns 6-15 and the parameters, separated by commas, starting in column 21.

The following checks are performed on the operand portion on each parameter card:

1. The first four characters of alphabetic parameters are checked for misspelling.
2. The @ symbols must not appear in any operand.
3. Operands may not be greater than 11 characters.
4. There can be no zone bits in numeric operands.
5. The operands are checked for too many parameters.
6. All parameters must appear in the prescribed order.
7. When a parameter is indicated as necessary (in Figure 17 under *Condition for Including Parameter),* the parameter must not be left out unless it, and all following parameters, are *NO* parameters. When the condition for including the parameter does not apply (for example, the parameter specifying type of disk unit if card or tape input in INPUTMEDIA card), simply leave the parameter out. Double commas are never used to show the commission of a parameter.

Example: The following two sets of parameters in the operand of the EXITS card are considered to be the same:

EXIT1, EXIT2, NOEXIT3 or
EXIT1, EXIT2

However, an error is detected if the card is punched:

EXIT1, ,EXIT3

Note: The SYSTEMSPEC, EXITS, and LOADAUDIT parameter cards may be omitted from the source desk if *all* parameters are to be *NO.* All other parameter cards required must be supplied, regardless of parameters. This includes the INLABELS and OUTLABELS cards, even when using 1301 disk storage.

It is recommended that the *NO* parameters be included to provide more easily understood documentation of the programs.

Numeric parameters are processed from right to left. Zeros are inserted in missing digit positions, and excess numbers are dropped from the left (high-order) end of entry. For example, if a *2* were punched for a parameter that required a 2-digit number, the prephase would insert the high-order zero, resulting in 02. If *1234* were punched for the same 2-digit parameter, the prephase would drop the two high-order positions, resulting in 34.

The parameter cards are preceded by a card with the entry FILE in the operation-code portion of the card (columns 16-19). When Autocoder reads this card, it calls in the file-organization prephase, which reads the parameter cards.

The first parameter card must be the ROUTINE card. The other cards required for a program can be entered in any order.

The user's symbolic deck must be followed by a card with the entry END in columns 16-18 and START in columns 21-25. This card follows any subroutines being used with the file-organization programs (conversion, card assembly, and exit routines). Unless batched assemblies are being performed, the END card is followed by a card with HALT in columns 16-19. Figure 16 shows the makeup of the source deck.

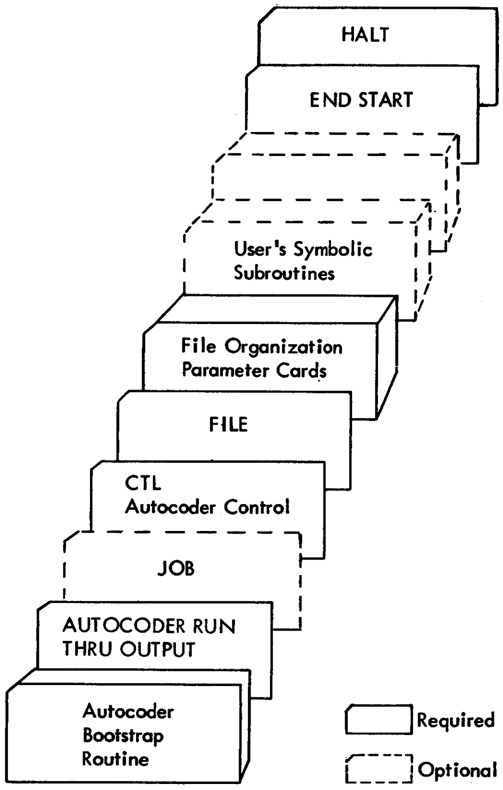

HALT
END START
User's Symbolic Subroutines
File Organization Parameter Cards
FILE
CTL Autocoder Control
JOB
AUTOCODER RUN THRU OUTPUT
Autocoder Bootstrap Routine
Required
Optional

Figure 16. Source Deck to Generate Object Programs

Summary of Parameter Cards Required

The following summary lists the parameter cards used in generating each of the 13 types of object programs.

Parameter Cards for PASS1 and PASS2

ROUTINE
SYSTEMSPEC (optional)
DISKDRIVES
EXITS (optional)
TYPEINPUT
INPUTMEDIA
INPMASTREC
MASTCONTRL
MASTBLOCK
MASTLIMITS
ADDRESSING
LOADAUDIT (optional)
NONHOMES
(if duplicate addressing)
TYPERNUNLD
(if reorganized)
INLABELS (if disk input)
OUTLABELS

Parameter Cards for RNADD

ROUTINE
SYSTEMSPEC (optional)
DISKDRIVES
EXITS (optional)
INPUTMEDIA
INPMASTREC
MASTCONTRL
MASTBLOCK
MASTLIMITS
ADDRESSING
LOADAUDIT (optional)
INLABELS (if disk input)
OUTLABELS

Parameter Cards for TRADD

ROUTINE
SYSTEMSPEC (optional)
DISKDRIVES
EXITS (optional)
TYPEINPUT
INPUTMEDIA
INPMASTREC
MASTCONTRL
MASTBLOCK
MASTLIMITS
ADDRESSING
TRAILENGTH
TRAILCNTRL
TRAILBLOCK
TYPERNUNLD
(if reorganized)
INLABELS (if disk input)
OUTLABELS

Parameter Cards for RNDEL1 and RNDEL2

ROUTINE
SYSTEMSPEC (optional)
DISKDRIVES
EXITS (optional)
INPMASTREC
MASTCONTRL
MASTBLOCK
MASTLIMITS
ADDRESSING
TRAILENGTH }
TRAILCNTRL } if trailers
TRAILBLOCK }
LOADAUDIT (optional)
TYPEDELETE
INLABELS

Parameter Cards for TRDEL

ROUTINE
SYSTEMSPEC (optional)
DISKDRIVES
EXITS (optional)
INPMASTREC
MASTCONTROL
MASTBLOCK
MASTLIMITS
ADDRESSING
TRAILENGTH
TRAILCNTRL
MINORCNTRL
TRAILBLOCK
TYPEDELETE
INLABELS

Parameter Cards for RNUNLD

ROUTINE
SYSTEMSPEC (optional)
DISKDRIVES
EXITS (optional)
INPMASTREC
MASTCONTRL
MASTBLOCK
MASTLIMITS
ADDRESSING
TRAILENGTH }
TRAILCNTRL } (if unloading trailer file)
TRAILBLOCK }
TYPERNUNLD
UNLDMEDIA
INLABELS
OUTLABELS (if disk output)

Parameter Cards for CSLOAD

ROUTINE
SYSTEMSPEC (optional)
DISKDRIVES
EXITS (optional)
TYPEINPUT
INPUTMEDIA
INPMASTREC
MASTCONTRL
MASTBLOCK
COLLATE
DISTENTRY
LOADAUDIT (optional)
TYPECSUNLD (if reorganized)
INLABELS (if disk input)
OUTLABELS

Parameter Cards for CSADD

ROUTINE
SYSTEMSPEC (optional)
DISKDRIVES
EXITS (optional)
INPUTMEDIA
INPMASTREC
MASTCONTRL
MASTBLOCK
COLLATE
LOADAUDIT (optional)
INLABELS (if disk input)
OUTLABELS

Parameter Cards for CSDEL1

ROUTINE
SYSTEMSPEC (optional)
DISKDRIVES
EXITS (optional)
INPMASTREC
MASTBLOCK
MASTCONTRL
COLLATE
TYPEDELETE
INLABELS

Parameter Cards for CSDEL2

ROUTINE
SYSTEMSPEC (optional)
DISKDRIVES
EXITS (optional)
INPMASTREC
MASTBLOCK
TYPEDELETE
INLABELS

Parameter Cards for CSUNLD

ROUTINE
SYSTEMSPEC (optional)
DISKDRIVES
EXITS (optional)
INPMASTREC
MASTCONTRL
MASTBLOCK
DISTENTRY }
LOADAUDIT } (if sequence-link field is retained)
(optional) }
TYPECSUNLD
UNLDMEDIA
INLABELS
OUTLABELS (if disk output)

Label	Parameters and Possible Operands	Condition for Including Parameter	Valid Range	Operand is Used in Generating													Explanation
				PASS1	PASS2	RNADD	TRADD	RNDEL1	RNDEL2	TRDEL	RNUNLD	CSLOAD	CSADD	CSDEL1	CSDEL2	CSUNLD	
ROUTINE	1. Object Program to be generated (one of the following): PASS1, PASS2, RNADD, TRADD, RNDEL1, RNDEL2, TRDEL, RNUNLD, CSLOAD, CSADD, CSDEL1, CSDEL2, CSUNLD.	Always.		X	X	X	X	X	X	X	X	X	X	X	X	X	This must be the first parameter card and immediately follows the FILE card.
SYSTEMSPEC	1. Direct seek feature: DIRECT or NODIRECT	Always.		X	X	X	X	X	X	X	X	X	X	X	X	X	If using 1311's with the direct seek feature, punch DIRECT. If using 1311's without the direct seek feature, punch NODIRECT. If using only 1301's, punch NODIRECT.
	2. Console printer: CONPRINT or NOCONPRINT	Always.															If the object machine has a console printer, punch CONPRINT; if not, punch NOCONPRINT
	3. Selective stacker feature: STACK or NOSTACK	Always.															If using a 1442 Model 1 without the selective stacker feature, punch NOSTACK. Otherwise, punch STACK.
	4. Alternate tape drive: ALTTAPE or NOALTTAPE	Parameters 4 and 5 are checked only when the program being generated is to process tape input or output. The parameters can be punched for all programs, however, allowing the same card to be used for each program generated.															If multiple-reel tape files are to be read or written on alternate tape drives,punch ALTTAPE. If only one tape drive is to be used, punch NOALTTAPE.
	5. Tape read error routine: ERRORSCAN or DUMP or blank																The procedure followed in the event of a permanent tape read error (failed to read after 100 tries) depends on this parameter. If the operand is: a. Blank, the erroneously read tape block is bypassed. b. ERRORSCAN, the program halts, allowing the operator to scan core storage for the invalid character. c. DUMP, the erroneously read tape block is written on another tape drive for later investigation.
DISKDRIVES	1. All disk units on line in object system: 0, 2, 4, 6, 8, +0, B, D, F, H	Always.		X	X	X	X	X	X	X	X	X	X	X	X	X	Punch 0, 2, 4, 6, 8 for 1311 disk drives on line. Punch +0, B, D, F, H for 1301 modules on line.
EXITS	1. Exit after assembling input: EXIT1 or NOEXIT1	Always.		X	X	X	X	X	X	X	X	X	X	X	X	X	This card is optional in all programs. See Program Exits.
	2. Exit after processing output: EXIT2 or NOEXIT2																
	3. Read a card and branch to 001 at EOJ. EXIT3 or NOEXIT3																
TYPEINPUT	1. Type of input file: NEW or REORGANIZED	all load programs		X	X		X					X					If the input file was produced by RNUNLD or CSUNLD, punch REORGANIZED; otherwise, punch NEW.
	2. Home-record-indicator check: CHECKHOME or blank	PASS1 and PASS2 if reorganized.		X	X												If the nonhome-record zone-bit indicator is to be checked by PASS1, punch CHECKHOME; otherwise, leave blank.
INPUTMEDIA (cont.)	1. Input media and blocking. DISK, X, Y or	If disk input.	X= 1-98 Y= 1-70	X	X	X	X					X	X				If disk input, punch DISK, X, Y where X = sectors per block Y = records per block.
	CARD, X or	If card input.	X= 0-98	X	X	X	X					X	X				If card input, punch CARD, X where X= cards per input record. Note. If input records are assembled by user's subroutine, X = 0.
	TAPE, X	If tape input.	X= 1-70	X	X	X	X					X	X				If tape input, punch TAPE, X where X= records per block. Note. If tape input to PASS2, X = 1.
	2. Type of disk unit. 11 or 01	If disk input.		X	X	X	X					X	X				If disk input from 1311 disk storage, punch 11. If disk input from 1301 disk storage, punch 01.

Figure 17. Parameter Cards (Part 1 of 5)

Figure 17. Parameter Cards (Part 2 of 5)

				pass1	pass2	madd	tradd	mdel1	mdel2	trdel	munld	csload	csadd	csdel1	csdel2	csunld	
INPUTMEDIA	3. Card read-error procedure. HALT or BYPASS	Only if card input. Not used if records are assembled by user's subroutine.		X	X	X	X					X	X				If a record contains an incorrect number of cards, an incorrect identification number, or if the cards of a record are out of sequence, the program prints an error message indicating the identification and sequence number of the erroneous card. The program then halts or bypasses the record, depending on this parameter.
	4. Tape padding character. Z or BLANK	Only if tape input.		X	X	X	X					X	X				If blocked tape input, punch the padding character. If unblocked tape input or if the padding character is blank, punch the word BLANK. Note. If tape input to PASS2, punch BLANK.
	5. Tape labels. NOLABEL, Y or NONSTANDARD or STANDARD,A STANDARD,B STANDARD,C	Only if tape input.	Y= 1-98	X	X	X	X					X	X				If the input tape has: a. No labels, punch NOLABEL, Y where Y = the number of reels to be read. b. Nonstandard labels, punch NONSTANDARD. c. Standard 120-character labels, punch STANDARD,A. d. Standard 80-character labels, punch STANDARD,B. e. Standard 84-character labels, punch STANDARD,C.
	6. Tape mark following header labels. TPMRK or NOTPMRK	Only if standard labels. Not used for PASS2.		X		X	X					X	X				If a tape mark follows standard tape header labels, punch TPMRK; if not, punch NOTPMRK Note. Type-A labels must be followed by tape marks.
	7. Extent of tape-label checking. ALL or IDENT	Only if standard labels. Not used for PASS2.		X		X	X					X	X				The extent of header label checking performed depends on this parameter. ALL - All fields in tape RDLIN card are checked. (See Tape RDLIN Cards.) IDENT - Only the file-identification field is checked.
INPMASTREC	1. Input master record length. X	Always.	Random: 1-9799 (unblocked) 1-4899 (blocked) Control Sequential 13-9794 (unblocked) 14-4893 (blocked)	X	X	X	X	X	X	X	X	X	X	X	X	X	When initially loading a file with PASS1, PASS2, or CSLOAD, and when making additions with RNADD or CSADD, X equals the actual length of the input record, including input terminal record marks. This count does not include fields added by the program such as trailer-address field, sequence-link field, or terminal record marks. When loading a reorganized file with PASS1, PASS2, or CSLOAD, X equals the actual length of the records produced by the unload program. This includes all appended fields such as trailer-address and converted-disk-address fields, the tag-zone position, and terminal record marks. (This value appears in the assembly listing of RNUNLD and CSUNLD in a DCW field labeled OUTL.) When generating TRADD and each of the delete and unload programs, X equals the length of the disk record in the organized master or main file, including all appended fields and terminal record marks. (This value appears in the assembly listing of PASS1 and CSLOAD in a field labeled DISKL.)
	2. Terminal record mark on input master record. RMIN or NORMIN	Always.		X	X	X	X	X	X	X	X	X	X	X	X	X	If input master data records have terminal record marks, punch RMIN; if not, punch NORMIN. For TRADD and all delete and unload programs, this parameter applies to the organized master or main file.
	3. Terminal record mark on output master and trailer records. RMOUT or NORMOUT	Always		X	X	X	X	X	X	X	X	X	X	X	X	X	If terminal record marks are to be written at the end of all output records, punch RMOUT; if not, punch NORMOUT. **To generate TRADD and all delete programs, these parameters must be either:** RMIN,RMOUT or NORMIN,NORMOUT.
MASTCONTRL	1. Position and length of first segment of control field. X, Y	Always.		X	X	X	X	X	X	X	X	X	X	X		X	The control data in each master record can be a single field of up to 30 characters or it can be broken into two or three subfields that total up to 30 characters. Regardless of the position of these fields in the data record, they are extracted and used as a single field. When two or three segments are defined, the second defined segment is placed to the right of the first, and the third to the right of the second. Each segment of the control data is defined by a parameter in the form: X, Y where: X = the low order (right hand) position of the segment within the data record, disregarding all appended fields. Y = the length of the segment.
	2. Position and length of second segment of control field. X, Y	If two or three segments of control data.															
	3. Position and length of third segment of control field. X, Y	If three segments of control data.															
MASTBLOCK (cont.)	1. Sectors per block in master file: X	Always	X= 1-98	X	X	X	X	X	X	X	X	X	X	X	X	X	These entries always refer to the organized master or main file.
	2. Records per block in master file: Y	Always	Random: Y= 1-10 Cont. Seq. Y= 1-10 (4K) Y= 1-30 (8K)	X	X	X	X	X	X	X	X	X	X	X	X	X	

Label	Parameters and Possible Operands	Condition for Including Parameter	Valid Range	Operand is Used in Generating													Explanation
				PASS1	PASS2	RNADD	TRADD	RNDEL1	RNDEL2	TRDEL	RNUNLD	CSLOAD	CSADD	CSDEL1	CSDEL2	CSUNLD	
MASTBLOCK	3. Type of disk unit: 11 or 01	Always		X	X	X	X	X	X	X	X	X	X	X	X	X	If the organized file area is in 1311 disk storage, punch 11. If the organized file area is in 1301 disk storage, punch 01.
	4. Separate additions area: CSADAREA or NOCSADAREA	Control sequential programs only. (Not checked by CSLOAD, but can be punched).											X	X	X	X	If a separate additions area is used, punch CSADAREA. If no additions are made, or if all additions are higher in sequence and are made to the end of the file area, punch NOCSADAREA.
MASTLIMITS	1. Lower limit of master file area. SSSSK0	Always included for random programs.		X	X	X	X	X	X	X	X						This must be the address of the first sector in the first track in the area. (K must be an even number, 0 must be zero.)
	2. Upper limit of master file area. SSSSK0																This must be the address of the first sector in the last track in the area. (K must be an even number, 0 must be zero.)
ADDRESSING	1. Converted addresses. DUPLICATE or UNIQUE	Always included for random programs.		X	X	X	X	X	X	X	X						If chained file is used to accomodate duplicate converted addresses, punch DUPLICATE. If there are no duplicate addresses, punch UNIQUE.
	2. Trailer files TRAILERS NOTRAILERS																If trailer records are used, punch TRAILERS; if not, punch NOTRAILERS.
TRAILENGTH	1. Length of input trailer records. X	Included for all random programs that process trailer records.	X= 20-9793				X	X	X	X	X						X equals the actual length of the input trailer records as read by the program being generated. It includes all appended fields present in the input but not those added by the program. When loading a reorganized file, it equals the length of the output trailer records produced by RNUNLD. (This value appears in the assembly listing of RNUNLD in a field labeled OUTL.) The delete programs and RNUNLD use the organized trailer file as input. Therefore, X for these programs is the length of the trailer records in the organized file. (This value appears in the assembly listing of TRADD in a field labeled TRLLTH.
	2. Terminal record marks on input trailer record. RMIN or NORMIN						X	X	X	X	X						If input trailer records have terminal record marks, punch RMIN; if not, punch NORMIN.
TRAILCNTRL	1. Position of first segment of major control field in trailers. X	Always.					X	X	X	X	X						All trailer records must contain the same control data as their associated master records. The control data need not be in the same positions within the record, but it must be defined in the same sequence as it was in the MASTCONTRL card. The operands in the TRAILCNTRL card give only the low-order position of each master-control-data segment within the trailer record.
	2. Position of second segment of major control field in trailers. X	If two or three segments of control data.	X= 1-9793				X	X	X	X	X						
	3. Position of third segment of major control field in trailers. X	If three segments of control data.					X	X	X	X	X						
MINORCNTRL	1. Position and length of minor control field in trailers. X, Y	Used for TRDEL only.	X= 1-9793 Y= 1-30							X							X = the low-order (right-hand) position of the minor control field within the trailer record, disregarding the trailer address field. Y = the length of the minor control field.
TRAILBLOCK	1. Sectors and records per block in trailer file. X, Y	Included for all programs that process trailer records.	X= 1-98 Y= 1-10				X	X	X	X	X						These entries apply to the organized trailer file. Neither the block length nor the blocking factor need be the same as in the master file.
COLLATE	1. Collating sequence ASCENDING or DECENDING	CSLOAD, CSADD, CSDEL1										X	X	X			Collating sequence of the control sequential file.

Figure 17. Parameter Cards (Part 3 of 5)

Figure 17. Parameter Cards (Part 4 of 5)

DISTENTRY	1. Frequency of distribution index entries. X or CYLINDER	Required for CSLOAD. Used for CSUNLD only if reorganized output.	X= 1-999998									X				X	An entry is made to the distribution index once every X records or once every cylinder.
LOADAUDIT	1. Audit trail print-out. PRINT or NOPRINT	This card is optional		X	X	X		X	X			X	X			X	Specify whether the loading and additions programs are to print the control data of each record as it is loaded, along with the actual disk address where loaded. Use also with CSUNLD if reorganizing. If this card is used with RNDEL1 and RNDEL2, the control data and new disk address are printed for each record displaced by a deletion. RNADD also prints the control data and new disk address of any records displaced by an addition.
NONHOMES	1. Nonhome record output. RECINWORK ADDINWORK RECONTAPE RECINCARD 1ADDINCARD 5ADDINCARD PUNCHZONE	PASS1 and PASS2 only.		X	X												See PASS1 for a description of the nonhome output options. Note. Use PUNCHZONE only if punch unit is a 1442 card read-punch, Model 1 or 2. Use RECINCARD only if punching on a 1402 or 1444.
	2. Type of punch unit. PARALLEL or blank	Use only with 1440 package.		X	X												If using 1444 card punch for RECINCARD or xADDINCARD, punch PARALLEL. Otherwise, leave blank.
TYPEDELETE	1. Type of delete program. DELETE or TAG	All delete programs.						X	X	X				X	X		If the generated program is to delete records, punch DELETE. If the program is to tag records, punch TAG. When generating RNDEL2, TRDEL, and CSDEL2, always punch DELETE.
	2. Audit of deleted or tagged records. PRINT or NOPRINT							X	X	X				X	X		If the entry is PRINT, the entire deleted or tagged record is printed out. If no printing is wanted, enter NOPRINT.
TYPERNUNLD	1. File to be unloaded. MASTER or TRAILER	RNUNLD always. PASS1, TRADD and PASS2 if reorganizing.		X	X		X				X						If the program is generated to unload a master file, punch MASTER; if generated to unload a trailer file, punch TRAILER.
	2. Records to be unloaded. ACTIVE or TAGGED or BOTH			X	X		X				X						This parameter tells whether only the active, only the tagged, or both active and tagged records are to be unloaded. If there are no tagged records in the file, punch BOTH.
	3. Converted address with output. CONADD or NOCONADD	Use only when unloading master records.		X	X						X						If the converted disk addresses are to be appended to the front of output master records, punch CONADD; if not, punch NOCONADD.
	4. Trailer addresses withoutput. TRAIL or NOTRAIL			X	X						X						If the trailer-address fields are ro be retained on the front of output master records, punch TRAIL; if not, punch NOTRAIL.
TYPECSUNLD	1. Records to be unloaded. ACTIVE or TAGGED or BOTH	Always used for CSUNLD. Used for CSLOAD only if reloading output of CSUNLD.										X				X	This parameter tells whether only the active, only the tagged, or both active and tagged records are to be unloaded. If there are no tagged records in the file, punch BOTH.
	2. Sequence link with output. LINK or NOLINK											X				X	If the file is to be reorganized with the sequence-link field blanked but retained, and the distribution index rebuilt, punch LINK. If the file is to merged and written with the sequence-link field stripped, punch NOLINK
	3. Type of unload program. 2PASS or blank	Used for two-pass unload only.														X	If a two-pass program is to be generated, punch 2PASS. (If this entry is used, LINK must be used also.) If generating a program to produce a relocated merge or a stripped merge, leave blank.
UNLDMEDIA (cont.)	1. Unload media. CARD or DISK or TAPE	Always used for RNUNLD and CSUNLD.									X					X	
	2. Disk output blocking. X, Y	Disk output only.	X= 1-98 Y= 1-70	pass1	pass2	rnadd	tradd	rndel1	rndel2	trdel	rnunld	csload	csadd	csdel1	csdel2	csunld	X = sectors per output block. Y = records per output block.

Label	Parameters and Possible Operands	Condition for Including Parameter	Valid Range	Operand is Used in Generating													Explanation
				PASS1	PASS2	RNADD	TRADD	RNDEL1	RNDEL2	TRDEL	RNUNLD	CSLOAD	CSADD	CSDEL1	CSDEL2	CSUNLD	
UNLDMEDIA	3. Tape output blocking. Y	Tape output only.	Y= 1-70								X					X	Y = records per output block.
	4. Type of punch unit. PARALLEL or blank	Used only with 1440 package.									X					X	If using a 1444 card punch for card output, punch PARALLEL; otherwise, leave blank.
	5. Type of disk unit. 11 or 01	Disk output only.									X					X	If disk output on 1311 disk storage, punch 11. If disk output on 1301 disk storage, punch01.
	6. Tape labels. NOLABELS NONSTANDARD STANDARD,A STANDARD,B STANDARD,C	Tape output only.									X					X	If the output tape is to be written with: a. No labels, punch NOLABEL. b. Nonstandard labels, punch NONSTANDARD. c. Standard, 120-character labels, punch STANDARD,A. d. Standard, 80-character labels, punch STANDARD,B. e. Standard, 84-character labels, punch STANDARD,C.
INLABELS	1. Extent of label checking on disk input. NOHEADER or PRTHEADER or CHKHEADER	If disk input (1311 and 1301). Note: Organized file area is considered input to delete and unload programs.		X	X	X	X	X	X	X	X	X	X	X	X	X	If 1311 disk input has no header labels, or if 1301 input,punch NOHEADER If input disk header labels are to be checked: a. On file identification field only, punch PRTHEADER. b. In their entirety, punch CHKHEADER.
OUTLABELS	1. Output disk header labels. NOHEADER or HEADER	If disk output (1311 and 1301),		X	X	X	X				X	X	X			X	If header labels are to be written for 1311 disk output files, punch HEADER. If 1301 output, or if no labels on 1311 output, punch NOHEADER.
	2. Extent of label checking on subsequent runs against organized file. IDENT or ALL	All load and additions programs, only if HEADER punched as first parameter.		X	X	X	X					X	X				If header labels are written for the organized-file areas, this parameter specifies the extent of label checking to be performed during subsequent runs. The parameter is not required for RNUNLD or CSUNLD but can be included.

Figure 17. Parameter Cards (Part 5 of 5)

Random File Example

The format of the file used throughout this example is shown in Figure 18. It is a hypothetical file, with many features included for the purpose of example. In order to make the example more meaningful, let us discuss the purpose of the file and the operations that are to be performed.

The file is conceived as an accounts-receivable file. It consists of a master file of customer accounts and a trailer file containing all transactions. The transactions are loaded as they occur, and the master records are updated semimonthly during a billing run. Again, it should be emphasized that this file has features included for the purpose of example and that it is certainly not recommended as the best way to organize typical accounts-receivable files. In the first place, it is more likely that the master records be updated daily, rather than loading the transactions separately as trailer records. However, the master-trailer approach can be useful in applications where the trailer file alone is to be used as input to programs that process information about the sales without regard to the specific customer accounts. For example, a program might process the trailer file (without looking at the master file) to compute sales commissions. If the trailer records included details about the specific items purchased, the trailer file could be processed by programs that assist in merchandise management.

In the sample file, there is a master record for each customer account. This master record contains the account number, the customer name and address (used in billing), and such current information as:

1. beginning balance
2. ending balance
3. credit limit
4. average number of purchases for a given period of time
5. total purchases for the life of the account
6. standard discount rate

The beginning and ending balances are aged, showing the amounts of the balance that is over 30 days old, over 60, and over 90.

All transactions against the file are entered as trailer records. These transactions include:

1. purchases
2. return of goods
3. payments

The three kinds of transactions can be entered from the same form. In fact, a *single* trailer record can include all three kinds.

Periodically, the file is processed by a billing program. This program processes those records called for in a processing-order file (POR). The POR contains the control fields of all of those records that are to be billed at a certain time of the month. The program reads the master record and then follows the trailer chain (if any) and updates all applicable fields in the master record. The bill is then printed out. As each trailer record is processed, it is tagged in the same way as RNDEL1 tags records (A- and B-bits over the sixth position of the trailer-address field appended to the trailer record). In order to repack the trailer area, doing away with those records that have been processed, RNUNLD is used to unload all trailer records that are not tagged. These trailers are then reloaded with a version of TRADD that is generated to load a reorganized file.

Note. The SYSTEMSPEC and DISKDRIVES cards describe the system on which the programs are to be run and the procedure to follow in the event of a tape-read error. These are factors that normally do not change from one program to the next. These cards can be the same for all programs even though a particular parameter is sometimes not needed. In Figure 19 for example, the ALTTAPE and ERRORSCAN operands are punched although there is no tape input or output in PASS1. These parameters are required for the tape output and input assumed for the reorganizing programs, however, and the programmer may wish to use the same card in all programs.

The same is true for the DISKDRIVES card: it is suggested that this card contain the control numbers of *all* disk units in the system regardless of the specific units used by a particular program. Once a program is generated, the specific units used are specified at object time in RDLIN cards.

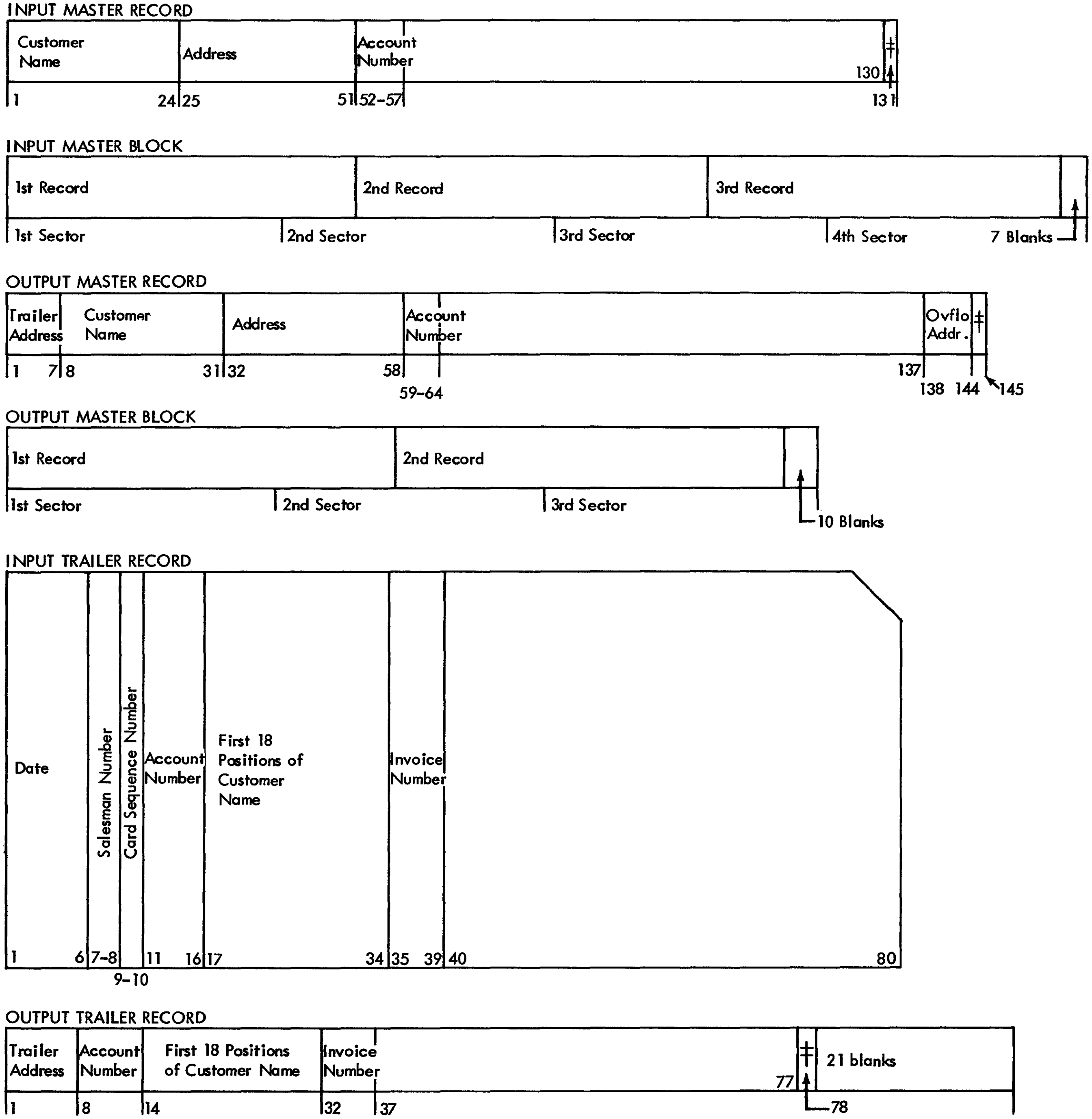
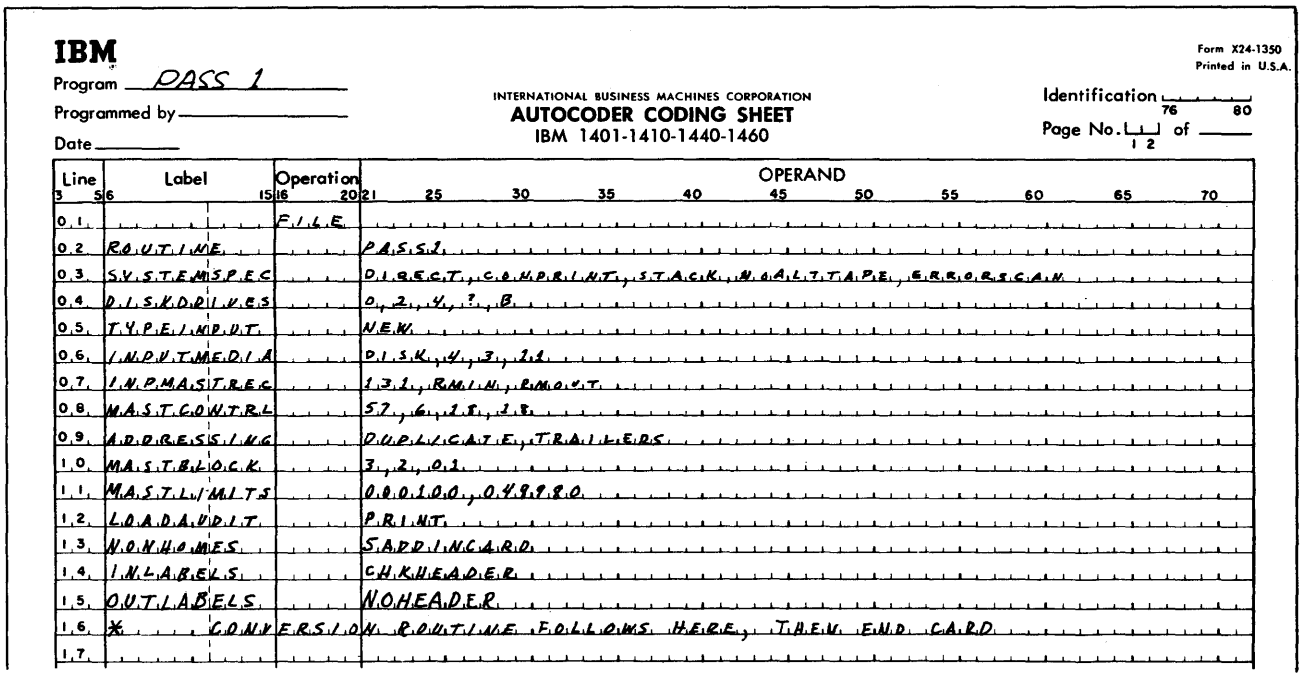


Figure 18. Record Formats in Sample File



IBM

Program PASS 1

Programmed by

Date

INTERNATIONAL BUSINESS MACHINES CORPORATION

AUTOCODER CODING SHEET

IBM 1401-1410-1440-1460

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

Identification 76 80

Page No. 1 2 of

```
Line  Label      Operation  OPERAND
3  5  6        15 16    20 21   25   30   35   40   45   50   55   60   65   70
0 1              FILE
0 2   ROUTINE              PASS1
0 3   SYSTEMSPEC           DIRECT,CONPRINT,STACK,NOALTTAPE,ERRORSCAN
0 4   DISKDRIVES           0,2,4,?,8
0 5   TYPEINPUT            NEW
0 6   INPUTMEDIA           DISK,4,3,11
0 7   INPMASTREC           131,RMIN,RMOUT
0 8   MASTCONTRL           57,6,18,18
0 9   ADDRESSING           DUPLICATE,TRAILERS
1 0   MASTBLOCK            3,2,01
1 1   MASTLIMITS           000100,049980
1 2   LOADAUDIT            PRINT
1 3   NONHOMES             SADDINCARD
1 4   INLABELS             CHKHEADER
1 5   OUTLABELS            NOHEADER
1 6   *    CONVERSION ROUTINE FOLLOWS HERE, THEN END CARD
1 7
```

Figure 19. Parameter Cards for Sample PASS1 Program

Load and Additions Programs

PASS1

Figure 19 shows the parameter cards used to generate the PASS1 object program for the sample file shown in Figure 18.

Input File

TYPEINPUT – This program is to perform the initial loading of the file. If a program is generated to reorganize the file after using RNUNLD, all of the cards that define the input would be changed except the MASTCONTRL and ADDRESSING cards.

INPUTMEDIA – Input is from 1311 disk storage. There are three data records per 4-sector input block.

Note: Assume that the disk input file was produced as the output of a sort program. As was mentioned, the master records include a field that shows the average number of purchases in a given period of time. The file could be sorted on this field, thus placing the most active records first in the file. This assures their being loaded in a home-record location or in a prime position in a chain.

INPMASTREC – The input records are 131 characters long, *including* a terminal record mark after each record. This card also specifies that the output records are to be followed by terminal record marks. These output record marks are placed *after* the overflow-address field.

Note: If it were desired to leave the record mark at the end of the output data record and *before* the overflow-address field, the operands would be: NORMIN, NORMOUT.

These programs would then treat the record mark as simply the last character of the data record.

MASTCONTRL – In a file of this kind, a good distribution of disk addresses can usually be obtained from the account number alone. In this example, however, the first 18 positions of the customer name are used along with the account number. This might be desirable if it were found that certain groupings or patterns in the account numbers show up in the disk addresses. Note that the positions of these two subfields are always defined relative to the first position of the data record, disregarding fields appended to the front of the disk record by the load or unload programs.

ADDRESSING – The control fields and conversion routine used can produce duplicate disk addresses. Trailer records are to be linked to the masters.

Output File

MASTBLOCK – The output file is to be written in 1301 disk storage. Two master records can be written in each 3-sector output block.

MASTLIMITS – One fourth of the first 1301 module is to be used for the master file. The first five tracks are reserved as work tracks.

LOADAUDIT – The control data of each record is to be printed as it is loaded, along with the actual disk address of the record.

NONHOMES – When a non-home record is encountered, its converted disk address and its input disk address are to be punched. Five sets of these addresses are punched in each card.

Disk Labels

INLABELS – The disk input file has header labels. This program is to check the file-identification field in these labels.

OUTLABELS – Because the output file is in 1301 disk storage, there is no possibility of output header labels. However, the OUTLABELS card is required with the entry NO HEADER.

Conversion Routine

Figure 20 shows a conversion routine for the sample file. This symbolic routine is placed after the parameter cards for all of the programs except PASS2.

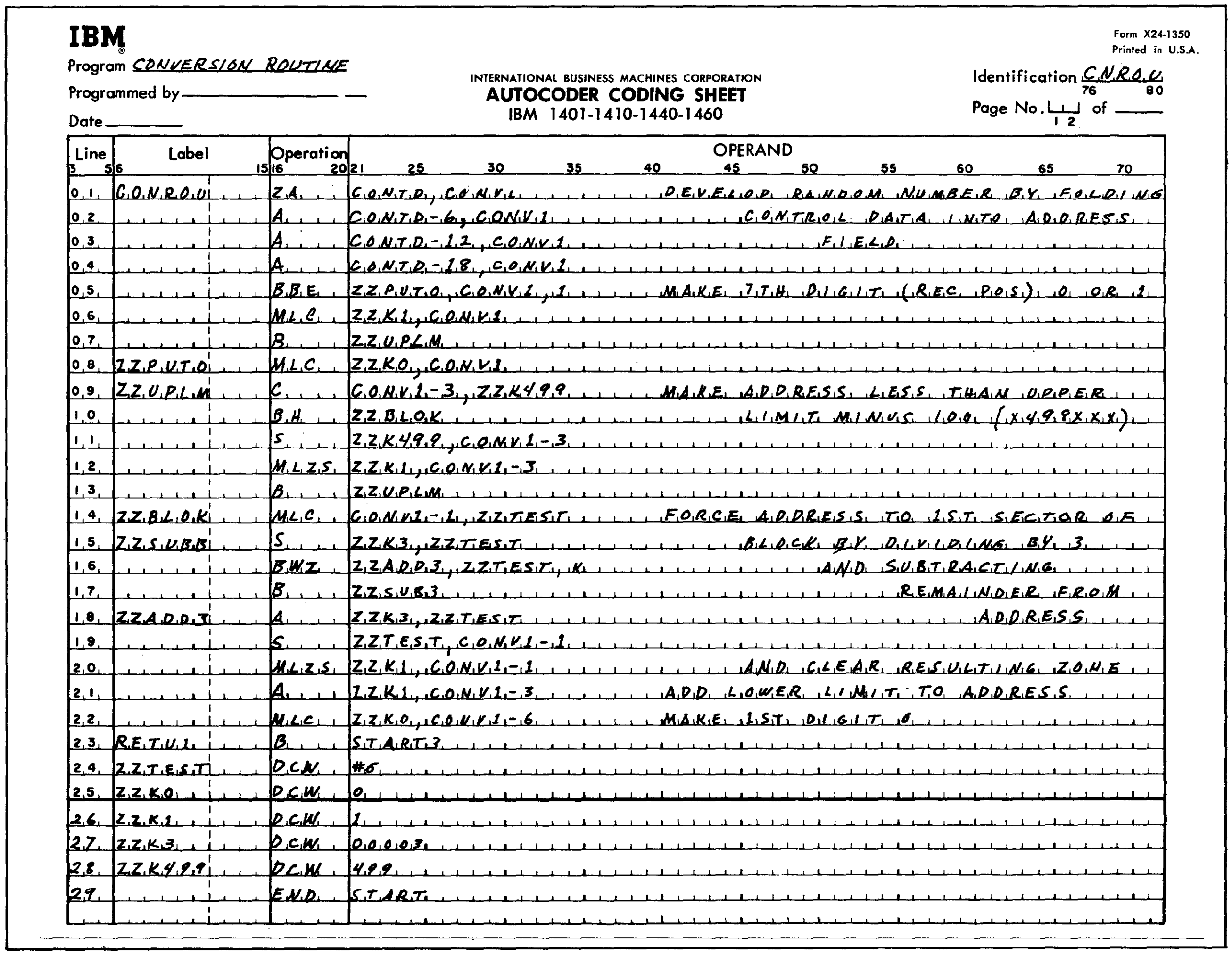
Note. The routine would be used for PASS2 also, if the NONHOMES card had the operand PUNCHZONE. It would not be used for RNUNLD if the TYPERNUNLD card did not include the operand CONADD.

The routine converts the customer name and customer number fields in the master *and* trailer records to valid disk addresses.

PASS2

The same cards used for PASS1 are used to generate the PASS2 load program. The operand of the ROUTINE card is changed to *PASS2,* and all of the other cards remain the same.

Note. If the RECONTAPE option had been used in the NONHOMES card for PASS1, the INPUTMEDIA card would have to be changed for PASS2. (See Figure 16.)



IBM

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

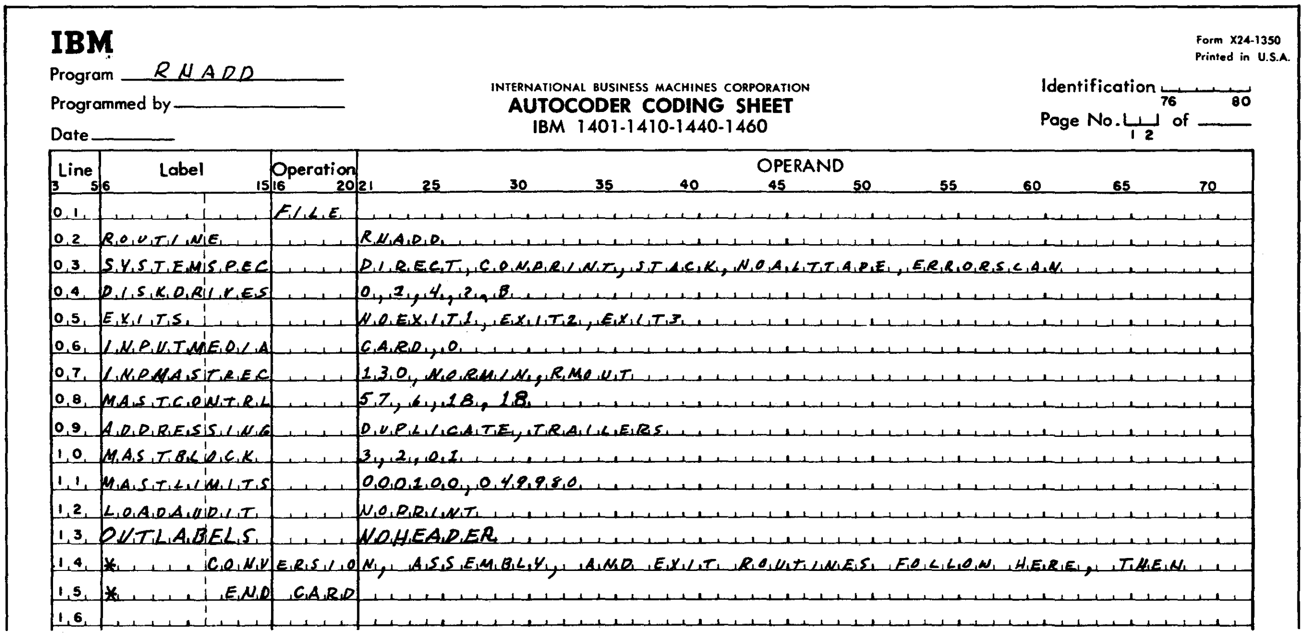
Program CONVERSION ROUTINE
Programmed by
Date

INTERNATIONAL BUSINESS MACHINES CORPORATION
AUTOCODER CODING SHEET
IBM 1401-1410-1440-1460

Identification CNROU
Page No. of

```
Line Label     Operation OPERAND
01 CONROU    ZA   CONTD,CONV1         DEVELOP RANDOM NUMBER BY FOLDING
02           A    CONTD-6,CONV1            CONTROL DATA INTO ADDRESS
03           A    CONTD-12,CONV1                FIELD
04           A    CONTD-18,CONV1
05           BBE  ZZPUTO,CONV1,1      MAKE 7TH DIGIT (REC POS) 0 OR 1
06           MLC  ZZK1,CONV1
07           B    ZZUPLM
08 ZZPUTO    MLC  ZZK0,CONV1
09 ZZUPLM    C    CONV1-3,ZZK499      MAKE ADDRESS LESS THAN UPPER
10           BH   ZZBLOK                   LIMIT MINUS 100 (X498XXX)
11           S    ZZK499,CONV1-3
12           MLZS ZZK1,CONV1-3
13           B    ZZUPLM
14 ZZBLOK    MLC  CONV1-1,ZZTEST      FORCE ADDRESS TO 1ST SECTOR OF
15 ZZSUBB    S    ZZK3,ZZTEST              BLOCK BY DIVIDING BY 3
16           BWZ  ZZADD3,ZZTEST,K               AND SUBTRACTING
17           B    ZZSUB3                             REMAINDER FROM
18 ZZADD3    A    ZZK3,ZZTEST                             ADDRESS
19           S    ZZTEST,CONV1-1
20           MLZS ZZK1,CONV1-1             AND CLEAR RESULTING ZONE
21           A    ZZK1,CONV1-3        ADD LOWER LIMIT TO ADDRESS
22           MLC  ZZK0,CONV1-6        MAKE 1ST DIGIT 0
23 RETU1     B    START3
24 ZZTEST    DCW  #5
25 ZZK0      DCW  0
26 ZZK1      DCW  1
27 ZZK3      DCW  00003
28 ZZK499    DCW  499
29           END  START
```

Figure 20. Conversion Routine for Sample File



IBM

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

Program RNADD

Programmed by

Date

INTERNATIONAL BUSINESS MACHINES CORPORATION
AUTOCODER CODING SHEET
IBM 1401-1410-1440-1460

Identification ___ 76 80

Page No. ___ of ___ 1 2

Line 3 5	Label 6 15	Operation 16 20	OPERAND 21 25 30 35 40 45 50 55 60 65 70
01		FILE	
02	ROUTINE		RNADD
03	SYSTEMSPEC		DIRECT, CONPRINT, STACK, NOALTTAPE, ERRORSCAN
04	DISKDRIVES		0, 2, 4, 2, 8
05	EXITS		NOEXIT1, EXIT2, EXIT3
06	INPUTMEDIA		CARD, 0
07	INPMASTREC		130, NORMIN, RMOUT
08	MASTCONTRL		57, 6, 18, 18
09	ADDRESSING		DUPLICATE, TRAILERS
10	MASTBLOCK		3, 2, 01
11	MASTLIMITS		000100, 049980
12	LOADAUDIT		NOPRINT
13	OUTLABELS		NOHEADER
14	* CONV	ERSIO	N, ASSEMBLY, AND EXIT ROUTINES FOLLOW HERE, THEN
15	* END	CARD	
16			

Figure 21. Parameter Cards for Sample RNADD Program

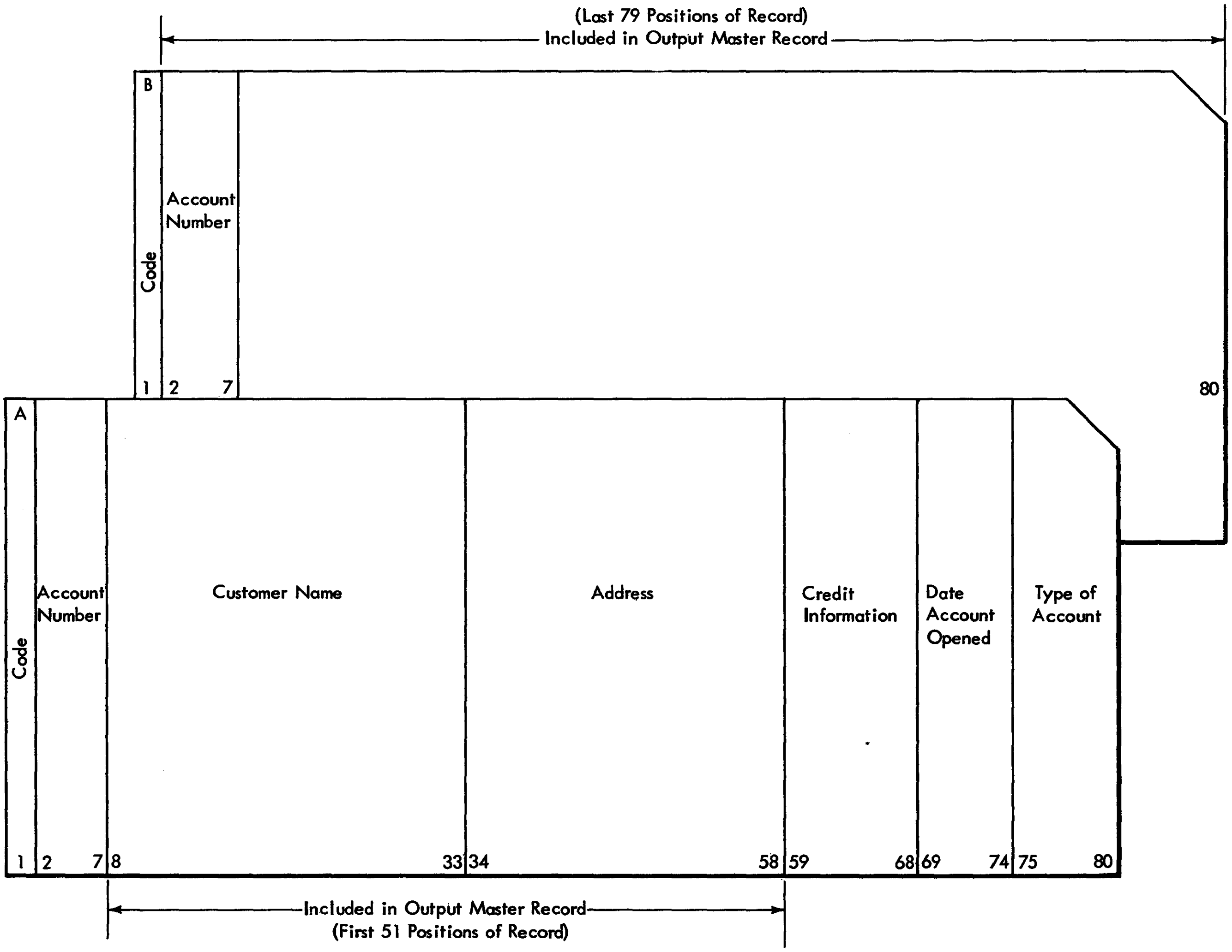


Figure 22. Input to Sample RNADD Program

RNADD

Figure 21 shows the parameter cards used to generate the RNADD additions program. The following cards are exactly the same as for PASS1.

SYSTEMSPEC	MASTCONTRL
DISKDRIVES	MASTBLOCK
ADDRESSING	MASTLIMITS

Input File

INPUTMEDIA – Input to the load programs was from disk storage. Card input is assumed for the additions program. The format of this input is shown in Figure 22. Because the format is not that required for automatic entry of the input records, the operand 0 indicates that a symbolic record-assembly routine is furnished with the source deck.

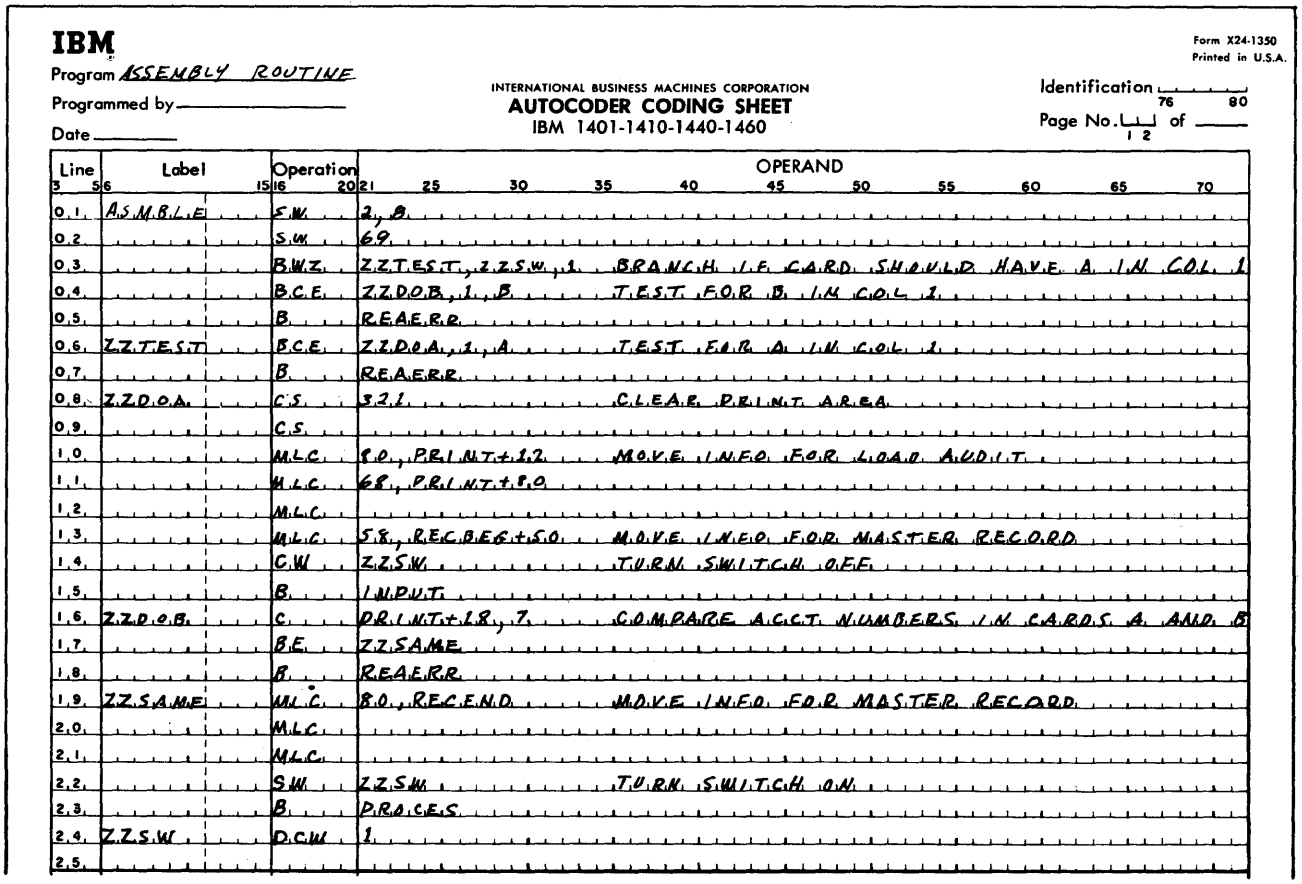
INPMASTREC – The assembled input data records are 130 characters long. There are no terminal record marks on input, but they must be appended to the output records, because they were during the initial loading of the file.

User's Subroutines

The assembly routine (Figure 23) extracts certain fields from the input cards and places these in the input record area. Two of these fields are also moved to the print area, along with other information that is present in the input but that is not included in the output master records. The assembly routine then branches back to the main program.

After processing the records, the program branches to an EXIT2 routine (Figure 24). This routine checks to see whether the record was written in a home or non-home position. If written as a home record, the disk address produced by the conversion routine is the actual address used to write the record. This address is moved to the print area. If the record is written as a non-home, the routine moves both the converted disk address (CONV1) and the actual disk address (WORKN) to the print area. The audit trail is then printed.

At end of job, the RNADD program prints a message giving the total number of records added. However, no separate count is given for home and non-home records. This distinction is automatically made by PASS1.



IBM

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

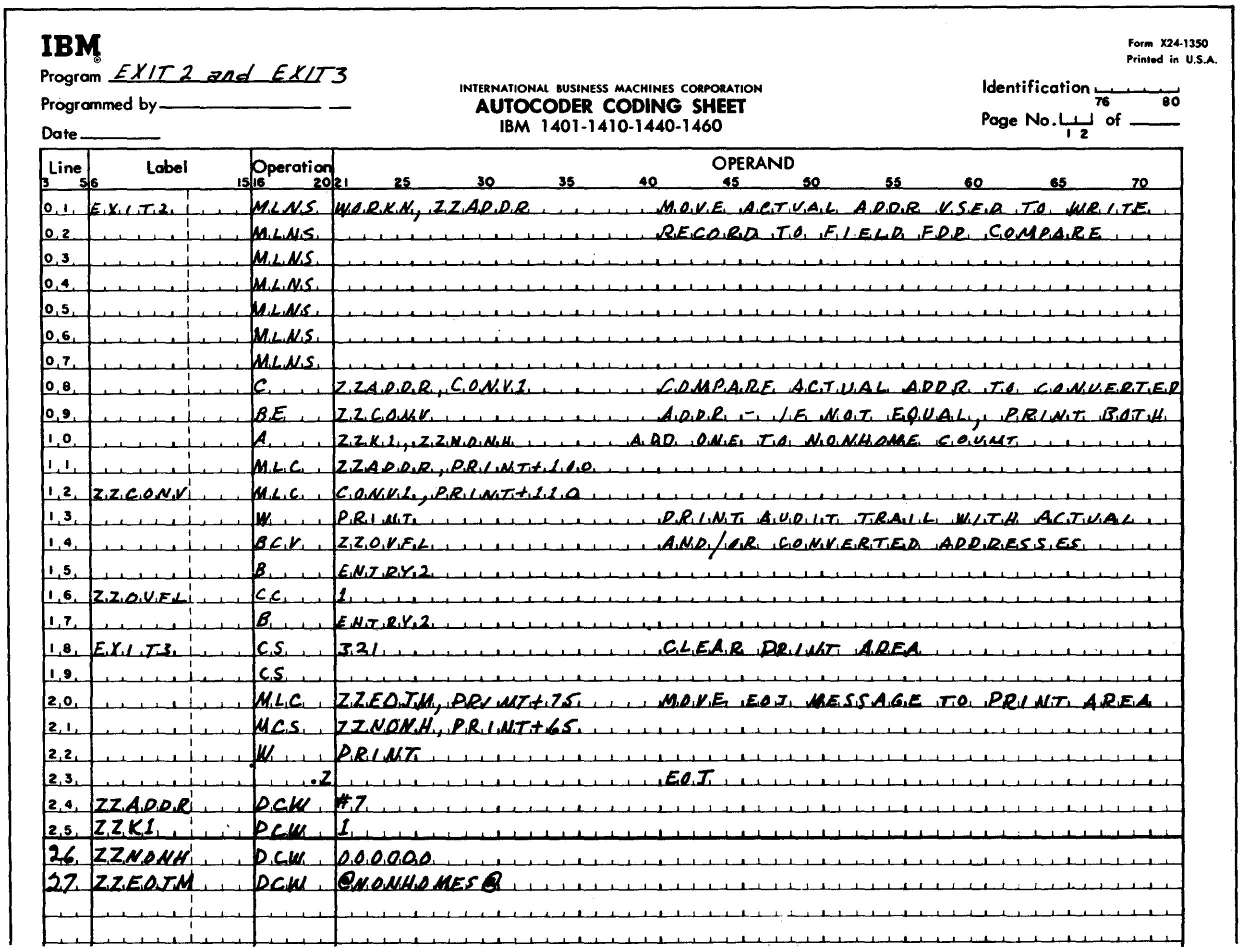
Program ASSEMBLY ROUTINE
Programmed by
Date

INTERNATIONAL BUSINESS MACHINES CORPORATION
AUTOCODER CODING SHEET
IBM 1401-1410-1440-1460

Identification
Page No. of

Line	Label	Operation	OPERAND
01	ASMBLE	SW	2,8
02		SW	69
03		BWZ	ZZTEST,ZZSW,1 BRANCH IF CARD SHOULD HAVE A IN COL 1
04		BCE	ZZDOB,1,B TEST FOR B IN COL 1
05		B	REAERR
06	ZZTEST	BCE	ZZDOA,1,A TEST FAR A IN COL 1
07		B	REAERR
08	ZZDOA	CS	321 CLEAR PRINT AREA
09		CS	
10		MLC	80,PRINT+12 MOVE INFO FOR LOAD AUDIT
11		MLC	68,PRINT+80
12		MLC	
13		MLC	58,RECBEG+50 MOVE INFO FOR MASTER RECORD
14		CW	ZZSW TURN SWITCH OFF
15		B	INPUT
16	ZZDOB	C	PRINT+18,7 COMPARE ACCT NUMBERS IN CARDS A AND B
17		BE	ZZSAME
18		B	REAERR
19	ZZSAME	MLC	80,RECEND MOVE INFO FOR MASTER RECORD
20		MLC	
21		MLC	
22		SW	ZZSW TURN SWITCH ON
23		B	PROCES
24	ZZSW	DCW	1
25			

Figure 23. Assembly Routine for Sample File



IBM

Program EXIT 2 and EXIT3

Programmed by

Date

INTERNATIONAL BUSINESS MACHINES CORPORATION
AUTOCODER CODING SHEET
IBM 1401-1410-1440-1460

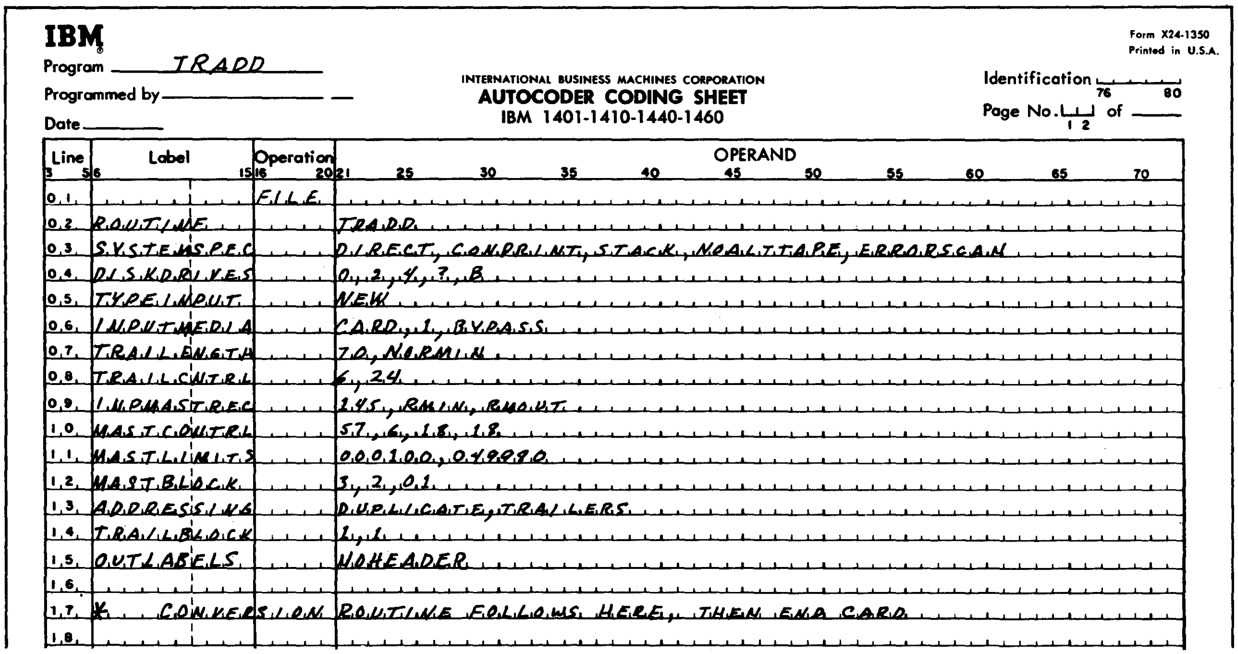
Form X24-1350
Printed in U.S.A.

Identification (76–80)

Page No. (1 2) of

Line (3–5)	Label (6–15)	Operation (16–20)	OPERAND (21–70)
01	EXIT2	MLNS	WORKN, ZZADDR MOVE ACTUAL ADDR USED TO WRITE
02		MLNS	RECORD TO FIELD FOR COMPARE
03		MLNS	
04		MLNS	
05		MLNS	
06		MLNS	
07		MLNS	
08		C	ZZADDR, CONV1 COMPARE ACTUAL ADDR TO CONVERTED
09		BE	ZZCONV ADDR - IF NOT EQUAL, PRINT BOTH
10		A	ZZK1, ZZNONH ADD ONE TO NONHOME COUNT
11		MLC	ZZADDR, PRINT+110
12	ZZCONV	MLC	CONV1, PRINT+110
13		W	PRINT PRINT AUDIT TRAIL WITH ACTUAL
14		BCV	ZZOVFL AND/OR CONVERTED ADDRESSES
15		B	ENTRY2
16	ZZOVFL	CC	1
17		B	ENTRY2
18	EXIT3	CS	321 CLEAR PRINT AREA
19		CS	
20		MLC	ZZEOJM, PRINT+75 MOVE EOJ MESSAGE TO PRINT AREA
21		MCS	ZZNONH, PRINT+65
22		W	PRINT
23		.Z	EOT
24	ZZADDR	DCW	#7
25	ZZK1	DCW	1
26	ZZNONH	DCW	000000
27	ZZEOJM	DCW	@NONHOMES@

Figure 24. Exit Routines for Sample File



IBM

Program TRADD

Programmed by

Date

INTERNATIONAL BUSINESS MACHINES CORPORATION
AUTOCODER CODING SHEET
IBM 1401-1410-1440-1460

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

Identification (76–80)

Page No. (1 2) of

Line (3–5)	Label (6–15)	Operation (16–20)	OPERAND (21–70)
01		FILE	
02	ROUTINE		TRADD
03	SYSTEMSPEC		DIRECT, CONPRINT, STACK, NOALTTAPE, ERRORSCAN
04	DISKDRIVES		0, 2, 4, 7, 8
05	TYPEINPUT		NEW
06	INPUTMEDIA		CARD, 1, BYPASS
07	TRAILENGTH		70, NORMIN
08	TRAILCNTRL		6, 24
09	INPMASTREC		145, RMIN, RMOUT
10	MASTCONTRL		57, 6, 18, 18
11	MASTLIMITS		000100, 049990
12	MASTBLOCK		3, 2, 01
13	ADDRESSING		DUPLICATE, TRAILERS
14	TRAILBLOCK		1, 1
15	OUTLABELS		NOHEADER
16			
17	* CONVE	RSION	ROUTINE FOLLOWS HERE, THEN END CARD
18			

Figure 25. Parameter Cards for Sample TRADD Program

In order to get this count, the EXIT2 routine adds 1 to a counter each time a record is processed as a nonhome. In order to print this total, EXIT3 is called for. The input file is followed by a card with ENDCD in columns 1-5. This card is then followed by one with BXXX in columns 1-4, where XXX is the actual machine address of the first instruction in the EXIT3 routine. This address can be found by looking up the label *EXIT3* in the assembly listing. The EXIT3 routine prints the total and halts.

TRADD

The transactions, or trailer records, are loaded into disk storage from cards. The input cards and the resulting output trailer records are shown in Figure 18. The parameter cards used to generate the TRADD program that loads the trailer records are shown in Figure 25. This TRADD program is used each time new trailer records are to be loaded. In a file such as the one discussed in this example, the transactions would probably be entered in the file daily. Another version of the TRADD program is generated to reload trailers unloaded by RNUNLD (see *Reorganization Programs*).

The following parameter cards in Figure 25 are exactly the same as for PASS1.

SYSTEMSPEC
DISKDRIVES
TYPEINPUT
ADDRESSING
MASTCONTRL
MASTBLOCK
MASTLIMITS
LOADAUDIT

Input Trailer File

INPUTMEDIA – Each input trailer record is contained in a single card. In the event of an error, the program is to print the control field of the record and bypass it.

Note. Because these are single-card records, the only error in the input record format that could be detected would be a sequence number (columns 9-10) other than 01.

TRAILENGTH – The input trailer records are 70 positions long and do not include terminal record marks.

TRAILCNTRL – The major control field used in the trailers is also used in the master records. However, the two subfields comprising it have been rearranged in the record and now appear in the order in which they are extracted for the conversion routine.

Master File

INPMASTREC – The organized master records are 145 positions long, including terminal record marks.

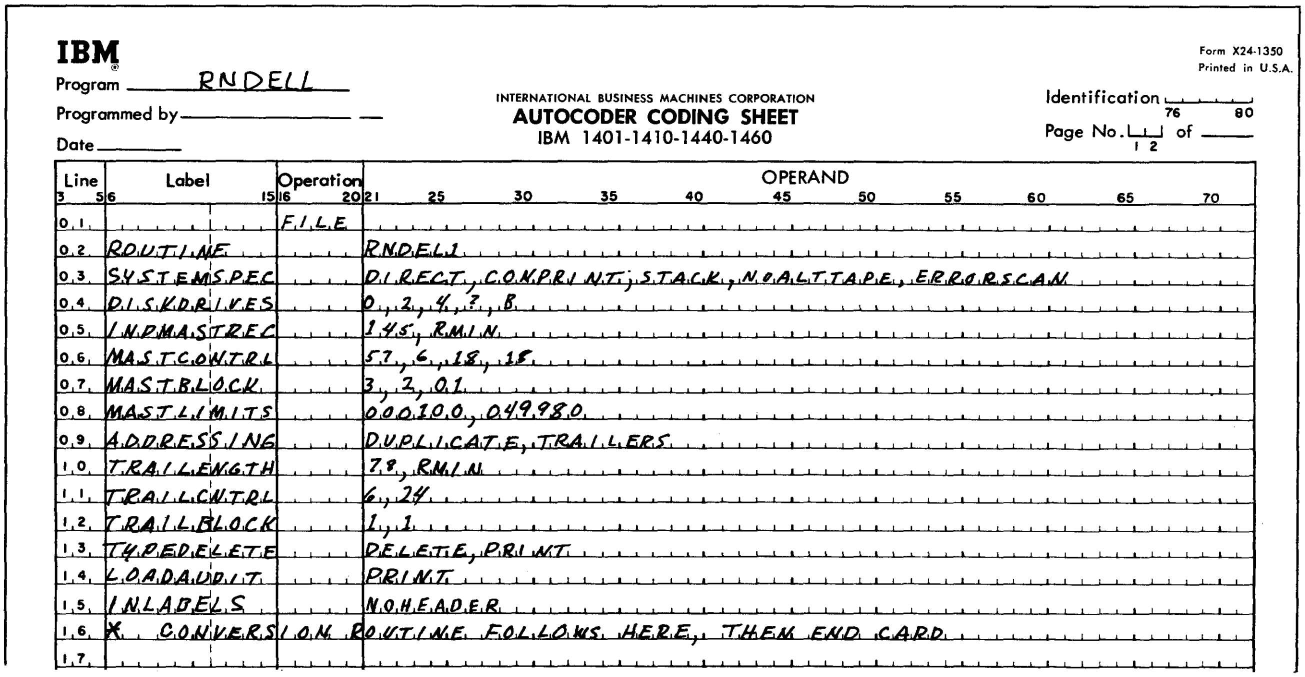
Output Trailer File

TRAILBLOCK – The output trailer file is written one sector at a time, with one 78-character transaction in each sector. The remaining 22 positions are padded with blanks.

Delete Programs

RNDEL1

The parameter cards required to generate a RNDEL1 program for the sample file are shown in Figure 26. The following cards are exactly the same as for TRADD:



IBM
Program RNDELL
Programmed by
Date

INTERNATIONAL BUSINESS MACHINES CORPORATION
AUTOCODER CODING SHEET
IBM 1401-1410-1440-1460

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

Identification 76 80
Page No. 1 2 of

Line 3 5	Label 6 15	Operation 16 20	OPERAND 21 25 30 35 40 45 50 55 60 65 70
01		FILE	
02	ROUTINE		RNDEL1
03	SYSTEMSPEC		DIRECT,CONPRINT,STACK,NOALTTAPE,ERRORSCAN
04	DISKDRIVES		0,2,4,?,8
05	INPMASTREC		145,RMIN
06	MASTCONTRL		57,6,18,18
07	MASTBLOCK		3,2,01
08	MASTLIMITS		000100,049980
09	ADDRESSING		DUPLICATE,TRAILERS
10	TRAILENGTH		78,RMIN
11	TRAILCNTRL		6,24
12	TRAILBLOCK		1,1
13	TYPEDELETE		DELETE,PRINT
14	LOADAUDIT		PRINT
15	INLABELS		NOHEADER
16	* CONVERSION ROUTINE FOLLOWS HERE, THEN END CARD		
17			

Figure 26. Parameter Cards for Sample RNDEL1 Program

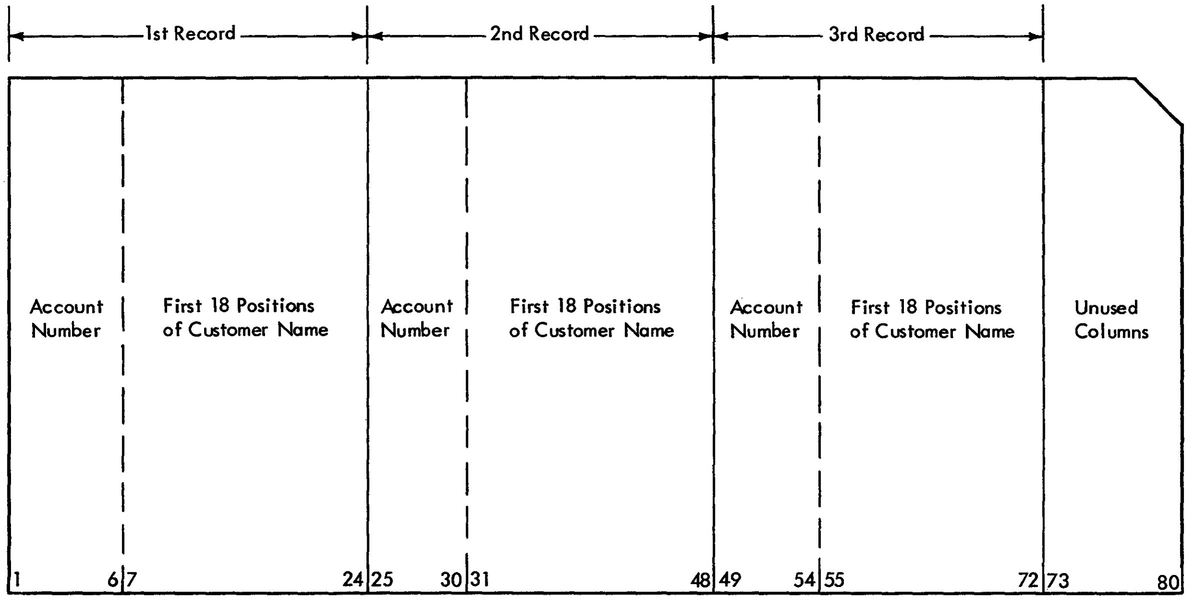


Figure 27. Input to RNDEL1

SYSTEMSPEC
DISKDRIVES
INPMASTREC
MASTCONTRL
MASTLIMITS
MASTBLOCK
ADDRESSING
TRAILCNTRL
TRAILBLOCK
LOADAUDIT

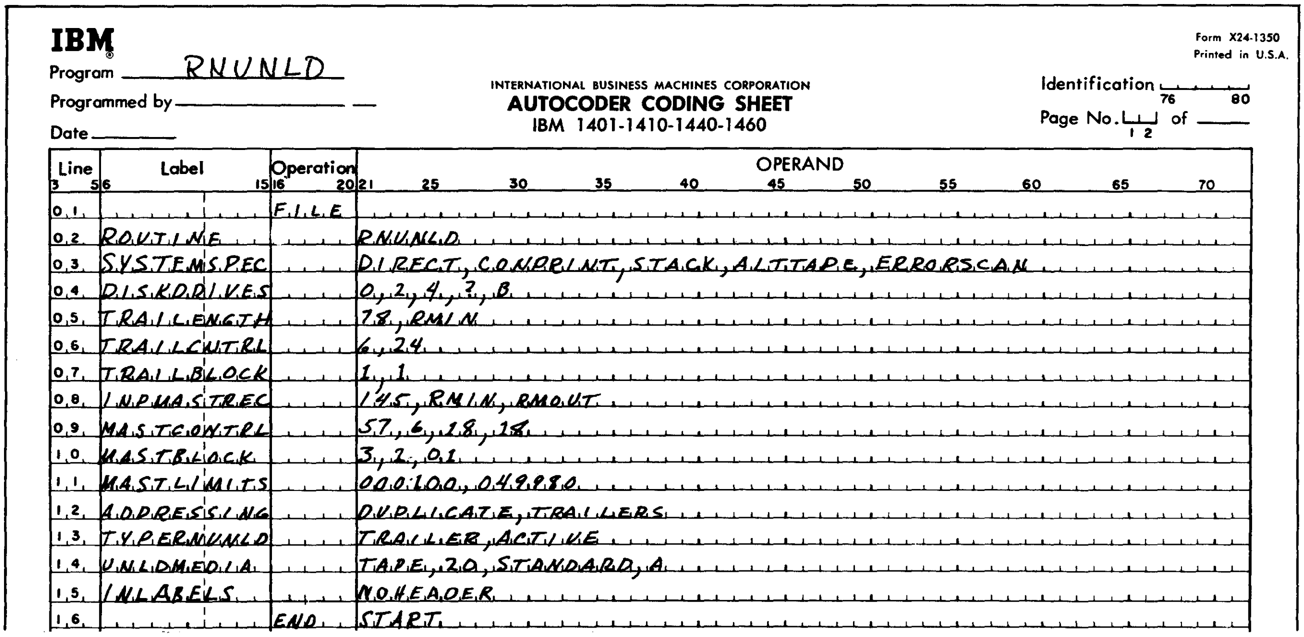
The TRAILENGTH card gives the length of the trailer records as they appear in the organized file. The TYPEDELETE card specifies that records (master and trailers) are to be deleted (instead of tagged) and that each record deleted is to be printed. The LOADAUDIT card is included and specifies that the control data and *new* disk address be printed for each record displaced by a deletion.

Note. The LOADAUDIT card would not be used if TAG were punched in the TYPEDELETE card.

The input to RNDEL1 is shown in Figure 27. Each card can contain the control data of one, two, or three records.

RNDEL2

If RNDEL1 were generated to tag records, a RNDEL2 program would be needed to delete all tagged records. The parameter cards would be exactly the same as shown for RNDEL1, except for the ROUTINE card.



IBM
Program RNUNLD
Programmed by
Date

INTERNATIONAL BUSINESS MACHINES CORPORATION
AUTOCODER CODING SHEET
IBM 1401-1410-1440-1460

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

Identification 76 80
Page No. 1 2 of

Line 3-5	Label 6-15	Operation 16-20	OPERAND 21-72
01		FILE	
02	ROUTINE		RNUNLD
03	SYSTEMSPEC		DIRECT, CONPRINT, STACK, ALTTAPE, ERRORSCAN
04	DISKDRIVES		0, 2, 4, 6, 8
05	TRAILENGTH		78, RMIN
06	TRAILCNTRL		6, 24
07	TRAILBLOCK		1, 1
08	INPMASTREC		145, RMIN, RMOUT
09	MASTCONTRL		57, 6, 18, 18
10	MASTBLOCK		3, 2, 01
11	MASTLIMITS		000100, 049980
12	ADDRESSING		DUPLICATE, TRAILERS
13	TYPERNUNLD		TRAILER, ACTIVE
14	UNLDMEDIA		TAPE, 20, STANDARD, A
15	INLABELS		NOHEADER
16		END	START

Figure 28. Parameter Cards for Sample RNUNLD Program

TRDEL

To generate a TRDEL program for the sample file, we would change the ROUTINE card shown in Figure 26, supply a MINORCNTRL card, leave out the LOADAUDIT card, and use all of the other cards exactly as they appear in Figure 26. The trailer records shown in Figure 18 have a field containing the invoice number in positions 35-39. This could be used as the minor control field. The MINORCNTRL card would be punched with the operands 39, 5.

Reorganization Programs

RNUNLD

The assumption was made in the description of the sample file that a billing program tags trailer records when they are processed and are no longer required. The parameter cards shown in Figure 28 are used to generate a RNUNLD program that is used to unload the active, or not-tagged, trailers. The following cards are exactly the same as for RNDEL1:

SYSTEMSPEC
DISKDRIVES
TRAILENGTH
TRAILCNTRL
TRAILBLOCK
MASTCONTRL
MASTBLOCK
MASTLIMITS
ADDRESSING

The INPMASTREC card includes the operand RMOUT, which specifies in this case that the *output trailer records* are to have terminal record marks.

The TYPERNUNLD card calls for active trailers to be unloaded.

The UNLDMEDIA card specifies tape output, with 20 records per tape block. The output tape is to be written with standard 120-character tape labels.

TRADD (Reorganized)

The unloaded, active trailer records can be reloaded with a TRADD program generated with the parameter cards shown in Figure 29. The following cards are the same as for the initial-loading version of TRADD:

ROUTINE
SYSTEMSPEC
DISKDRIVES
TRAILCNTRL
MASTLIMITS
INPMASTREC
MASTBLOCK
MASTCONTRL
ADDRESSING
TRAILBLOCK
LOADAUDIT

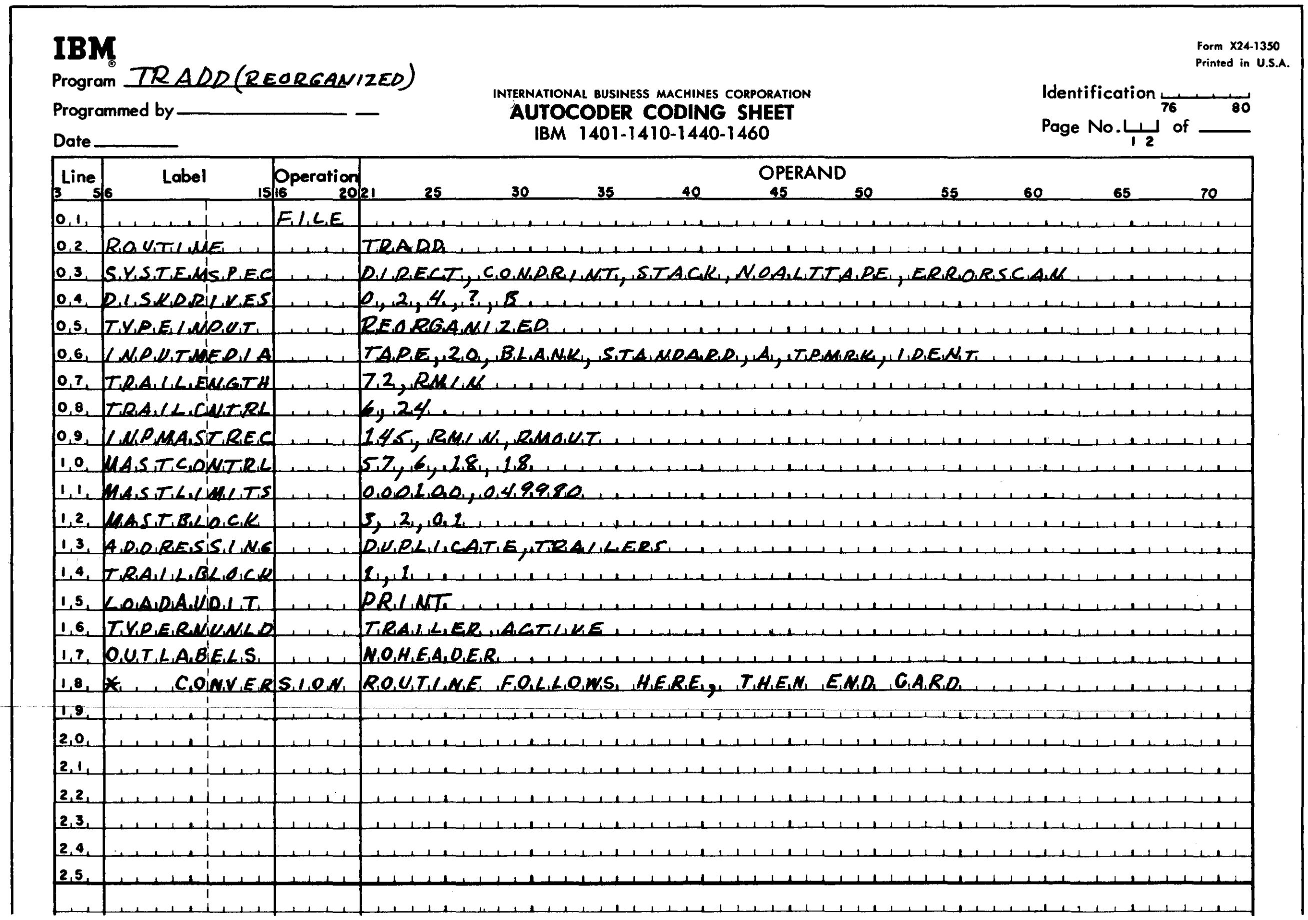
The following cards are the same as for the RNUNLD program:

TYPERNUNLD
UNLDMEDIA

The TYPEINPUT card specifies:

1. tape input
2. 20 records per tape block
3. blanks padding out any unused record locations
4. standard 120-character tape labels followed by tape marks
5. that the tape header labels are to be checked on file identification field only.

The TRAILENGTH card gives the length of the input trailer records. These records were unloaded by RNUNLD with record marks at the end of the records and a blank position in front of each record.



IBM

Program TRADD (REORGANIZED)

Programmed by ______

Date ______

INTERNATIONAL BUSINESS MACHINES CORPORATION

AUTOCODER CODING SHEET

IBM 1401-1410-1440-1460

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

Identification ______ 76 80

Page No. __ of ______ 1 2

Line 3 5	Label 6 15	Operation 16 20	OPERAND 21 25 30 35 40 45 50 55 60 65 70
0 1		FILE	
0 2	ROUTINE		TRADD
0 3	SYSTEMSPEC		DIRECT, CONDPRINT, STACK, NOALTTAPE, ERRORSCAN
0 4	DISKDRIVES		0, 2, 4, 7, 8
0 5	TYPEINPUT		REORGANIZED
0 6	INPUTMEDIA		TAPE, 20, BLANK, STANDARD, A, TPMRK, IDENT
0 7	TRAILENGTH		72, RMIN
0 8	TRAILCNTRL		6, 24
0 9	INPMASTREC		145, RMIN, RMOUT
1 0	MASTCONTRL		57, 6, 18, 18
1 1	MASTLIMITS		000100, 049980
1 2	MASTBLOCK		3, 2, 01
1 3	ADDRESSING		DUPLICATE, TRAILERS
1 4	TRAILBLOCK		1, 1
1 5	LOADAUDIT		PRINT
1 6	TYPERNUNLD		TRAILER, ACTIVE
1 7	OUTLABELS		NOHEADER
1 8	* CONVER	SION	ROUTINE FOLLOWS HERE, THEN END CARD
1 9			
2 0			
2 1			
2 2			
2 3			
2 4			
2 5			

Figure 29. Parameter Cards for Sample TRADD Reorganize Program

Retrieval Routines

The IBM disk-file organization routines do not include any retrieval routines. The Input/Output Control System and the Report Program Generator can often be used to advantage on files organized by these routines. Both of these programming systems accommodate files organized by the random and the control-sequential routines.

The block diagram shown in Figure 30 shows a method of retrieving records from a random file.

Control-sequential files are normally processed sequentially: a program simply reads the next consecutive disk location unless directed to an additions area by an address in the sequence-link field.

When it is desired to retrieve certain records from a control-sequential file without passing the entire file, a method such as that shown in Figure 31 can be used. The block diagram in Figure 31 uses a search method that is especially efficient in a large file if the entries in the distribution index represent a large range of disk storage. For example, if an entry were made to the index once for each cylinder, this method is probably quicker than any other. However, if entries are made to the distribution index more often, it may be quicker to scan the records sequentially within the range until the desired record is found. Linkage need not be followed to the additions area unless the desired record is actually in that area.

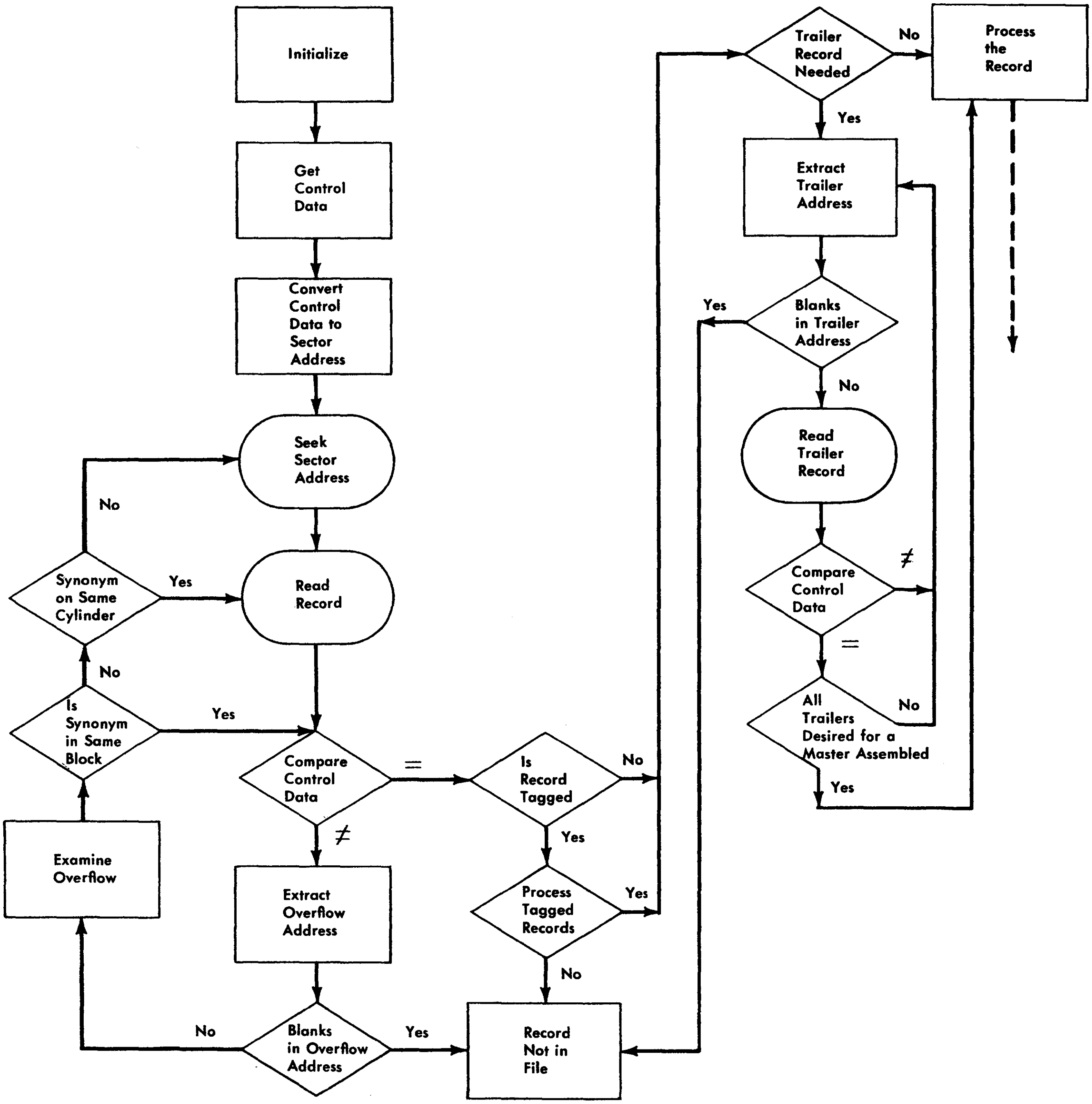


Figure 30. Retrieval from a Random File

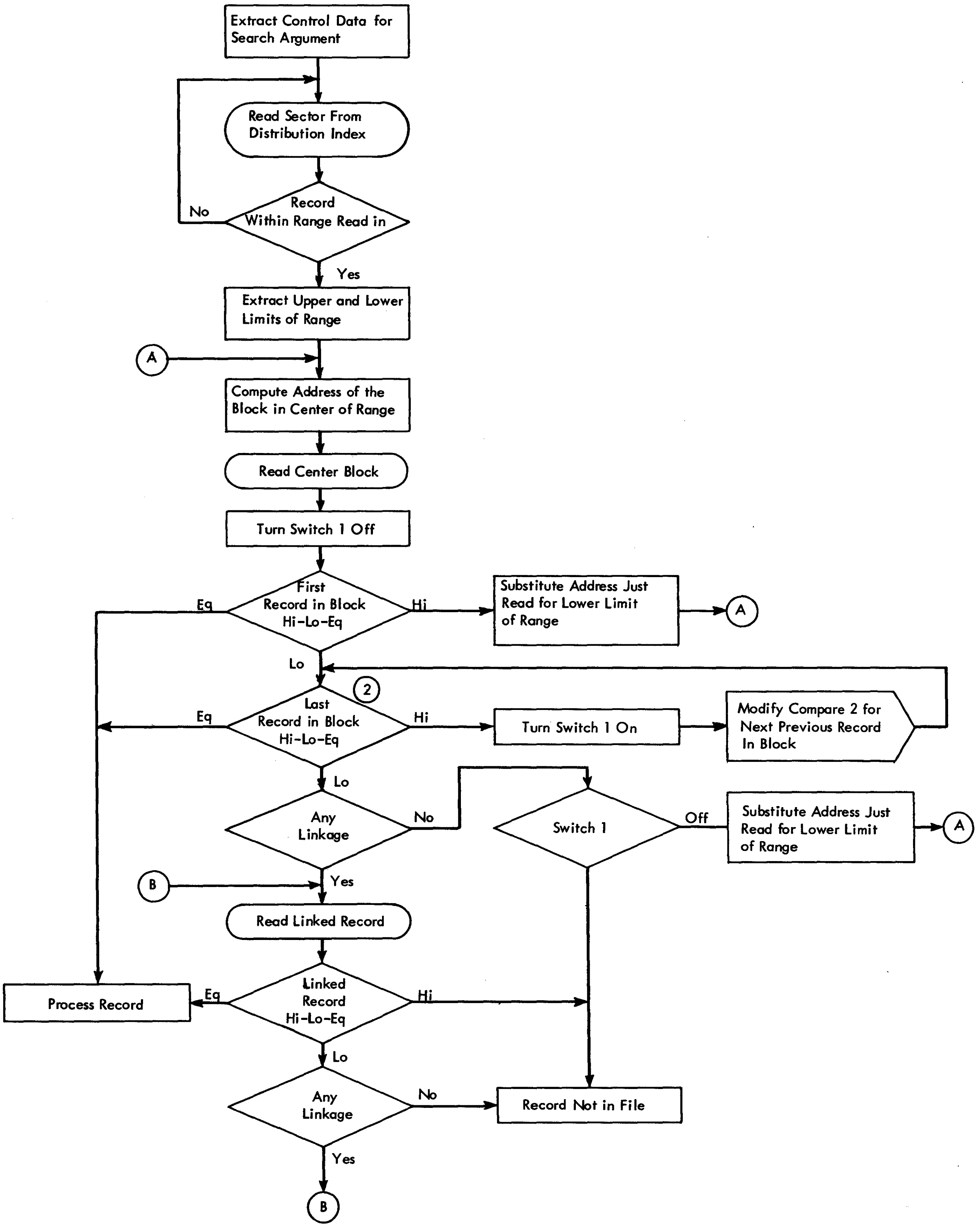

Extract Control Data for Search Argument
Read Sector From Distribution Index
Record Within Range Read in
No
Yes
Extract Upper and Lower Limits of Range
A
Compute Address of the Block in Center of Range
Read Center Block
Turn Switch 1 Off
First Record in Block Hi-Lo-Eq
Eq
Hi
Substitute Address Just Read for Lower Limit of Range
A
Lo
2
Last Record in Block Hi-Lo-Eq
Eq
Hi
Turn Switch 1 On
Modify Compare 2 for Next Previous Record In Block
Lo
Any Linkage
No
Switch 1
Off
Substitute Address Just Read for Lower Limit of Range
A
B
Yes
Read Linked Record
Process Record
Eq
Linked Record Hi-Lo-Eq
Hi
Lo
Any Linkage
No
Record Not in File
Yes
B

Figure 31. Random Retrieval from a Control-Sequential File

Index

READER'S SURVEY FORM

Disk File Organization Routines Specifications
IBM 1401, 1440, 1460 (1311 and 1301) Form C24-3185-2

- Is the material:

	Yes	*Satisfactory*	*No*
Easy to read?	□	□	□
Well organized?	□	□	□
Fully covered?	□	□	□
Clearly explained?	□	□	□
Well illustrated?	□	□	□

- How did you use this publication?
  As an introduction to the subject □
  For additional knowledge of the subject □

- Which of the following terms best describes your job?

  Customer Personnel
  Manager □
  Systems Analyst □
  Operator □
  Programmer □
  Trainee □
  Other ____________

  IBM Personnel
  Customer Engineer □
  Instructor □
  Sales Representative □
  Systems Engineer □
  Trainee □
  Other ____________

- Check specific comment (if any) and explain in the space below:
  (*Give page number*)
  □ Suggested Change (Page) □ Suggested Addition (Page)
  □ Error (Page) □ Suggested Deletion (Page)

Explanation:

Space is available on the other side of this page for additional comments.
Thank you for your cooperation.

Staple

Fold

Fold

FIRST CLASS
PERMIT NO. 170
ENDICOTT, N. Y.

BUSINESS REPLY MAIL
NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

POSTAGE WILL BE PAID BY . . .

IBM Corporation
General Products Division
Development Laboratory
Endicott, N. Y. 13764

Attention: Product Publications, Dept. 171

Fold

Fold

Cut Along Line

IBM 1401, 1440, 1460 (1311 and 1301) Printed in U.S.A. C24-3185-2

IBM®
International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N. Y. 10601

Additional Comments:

IBM 1401, 1440, 1460 (1311 and 1301)   Printed in U.S.A.   C24-3185-2

IBM®
International Business Machines Corporation
Data Processing Division
112 East Post Road, White Plains, N. Y. 10601