NCR CENTURY FILE CONCEPTS

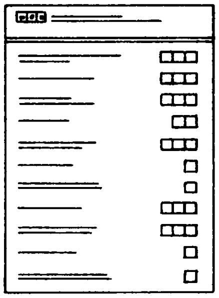
GENERAL CONCEPTS

The use of comprehensive and powerful system software eases the problem of data transfer to and from peripheral units. Each peripheral unit, i.e., printer, disc, magnetic tape, or punched paper equipment, is treated as a file storage device by the software. The programmer informs the software at compilation time of the specific file parameters and functions through the use of File Specification Sheets.

The File Specification Sheets require such information as the file reference name, the file type, the peripheral type, dates for protection, the record name, and other necessary file control information.

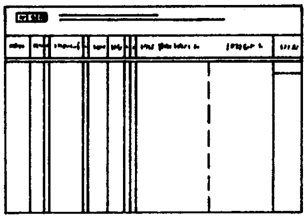
The records in the file are named and described on Data Layout Sheets which are input to the compiler immediately following the File Specification Sheets. Through this sequence of presentation, the compiler associates the file name and the record name so that instructional references to the file or record will affect the proper data.

FILE SPECIFICATION SHEETS AND DATA LAYOUT SHEETS



FILE SPECIFICATION

FILE NAME
FILE TYPE
BLOCK LENGTH
RECORD LENGTH
USER ROUTINE NAMES
FILE REFERENCE NAME
SYMBOLIC UNIT DESIGNATOR



DATA LAYOUT

RECORD NAME
RECORD SIZE
FIELD NAME
FIELD SIZE
FIELD TYPE
FIELD POSITION

- The file is described on the File Specification Sheets.
- The record is described on the Data Layout Sheets.
- The information on the Data Layout Sheets follows the information on the File Specification Sheets as input to the compiler.
- The compiler associates the record name on the Data Layout Sheets with the file name on the File Specification Sheets.

I/O INSTRUCTIONS

Simple input/output (I/O) instructions handle the normally complex operation of inputting or outputting data to a peripheral unit because the software carries the burden of the job.

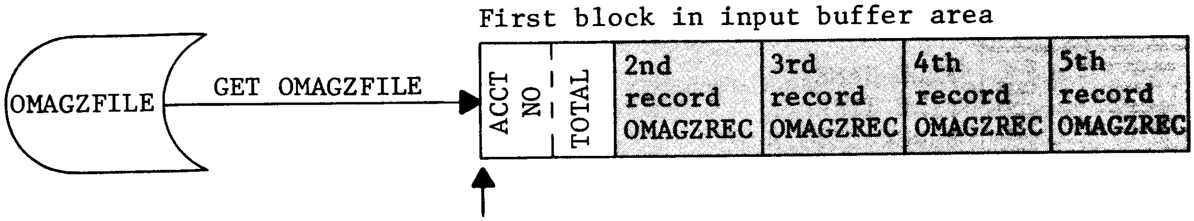
Generally, to sequentially input records from a file, the programmer need only specify GET File Reference. To sequentially output records, the programmer need only specify PUT File Reference. (File Reference is the actual file reference name - such as MAGZFILE - as it appears on the File Specification Sheets, that is, the name used by a particular program to refer to a particular file.)

Records are usually handled in groups called blocks. The block size may be some multiple of the record size or the same size as one record. The programmer indicates both the block length and the record length on the File Specification Sheets. Using this information, the software sets up reserved areas in memory called buffer areas which are normally the same size as a block. For output, the software uses these output buffer areas to form the various records into blocks. For input, the software uses these input buffer areas to break down the blocks and to present one record at a time to the program.

- Input

To input records from a file, the programmer specifies GET File Reference. Assuming the File Reference name is OMAGZFILE, the programmer specifies GET OMAGZFILE. The first time this instruction is executed, the first record in the file is presented to the program. (Assume this record is named OMAGZREC.)

FIRST EXECUTION OF GET

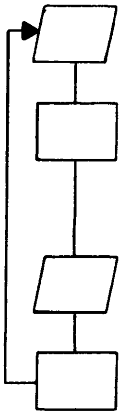


- The first GET instruction presents the first record in the input buffer to the program.

Although the instruction specifies GET OMAGZFILE, the record (OMAGZREC) is presented to the program. The GET instruction may therefore be interpreted as follows:

GET OMAGZFILE = GET an OMAGZREC from OMAGZFILE.

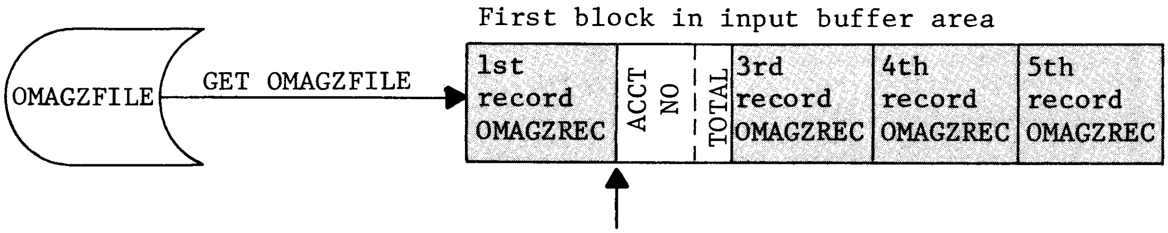
The record has the name and the format specified on the Data Layout Sheets. Any instructional reference to the record affects the data in the input buffer area.



GET OMAGZFILE	Makes the first OMAGZREC in the input buffer area available to the program.
ADD CONSTANT, TOTAL	Adds to the TOTAL field of OMAGZREC in the input buffer, the amount in an area called CONSTANT.
PUT PRINTFILE	Prints the result.
BRANCH	Branches to the beginning of this routine to GET another OMAGZREC.

Each additional encounter with the GET OMAGZFILE instruction presents another OMAGZREC to the program until the input buffer area is empty. Then, at the next execution of GET the software automatically reads the next sequential block of records from the file into the input buffer area.

SECOND EXECUTION OF GET

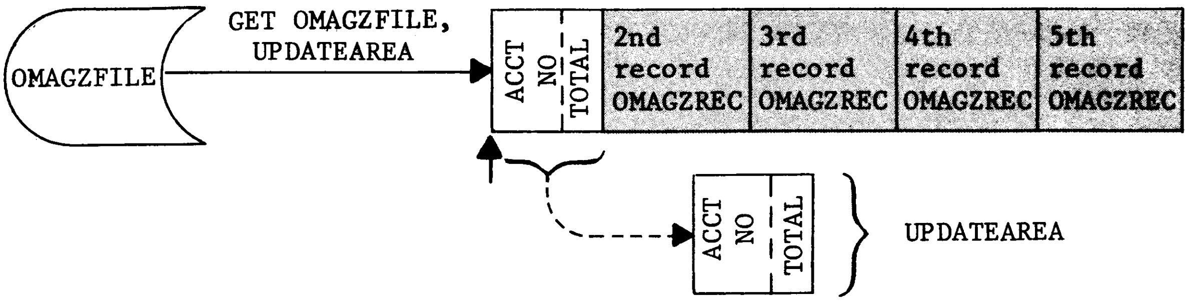


- The second GET instruction presents the second record in the input buffer to the program.

- Input with a Work Area

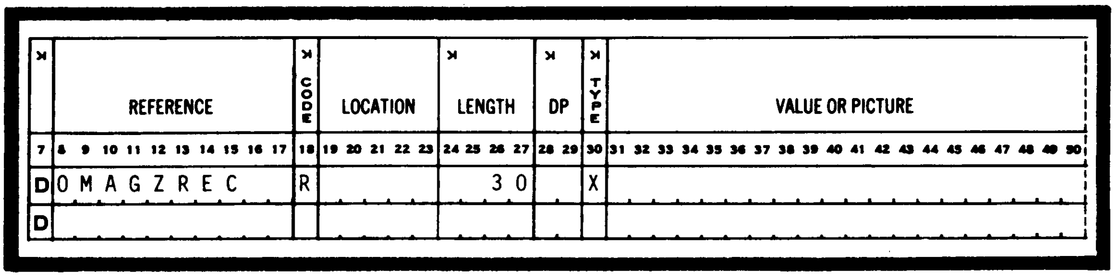
The programmer may name a work area as the second operand of the GET instruction, GET OMAGZFILE, UPDATEAREA. In this case, the record is made available in the input buffer area and is also moved to the named work area.

GET SPECIFYING A WORK AREA



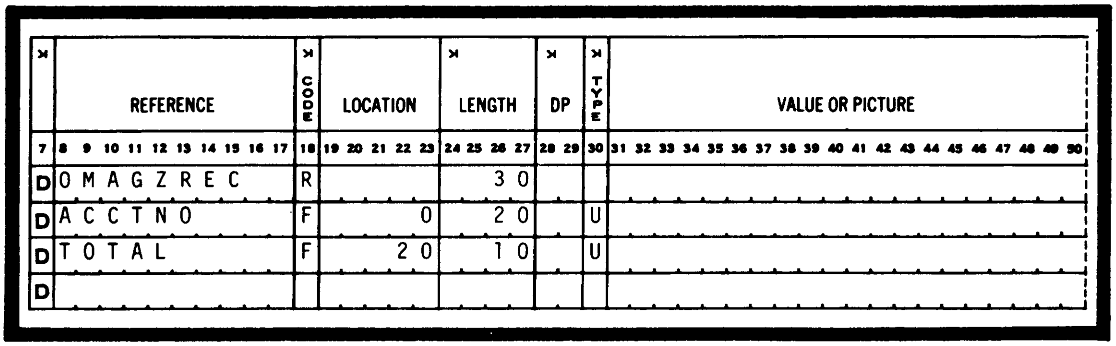
- GET presents a record in the input buffer to the program and also moves that record to the work area named in the second operand.

The record must be defined on the Data Layout Sheets which are input to the compiler immediately following the File Specification Sheets. The extent of detail in the record definition is dependent upon the number and type of accesses made to the record in the input buffer area. If individual fields within the record are not accessed in the input buffer area, no detail is necessary on the Data Layout Sheet.



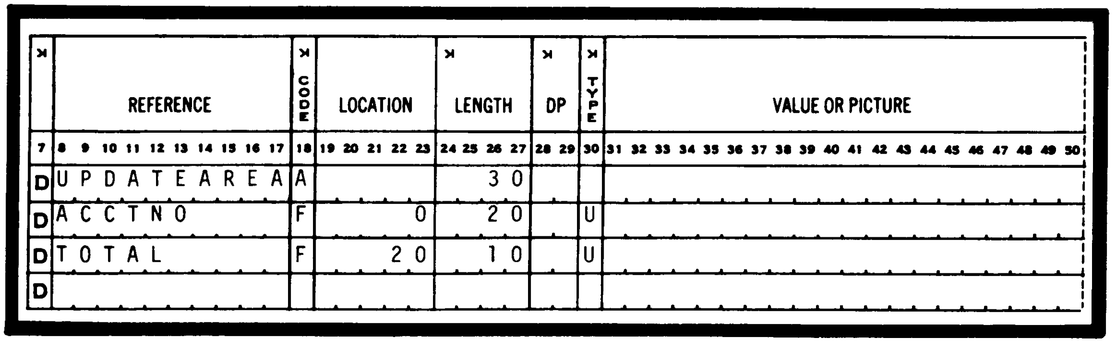
x	REFERENCE	x CODE	LOCATION	x LENGTH	x DP	x TYPE	VALUE OR PICTURE
7	8 9 10 11 12 13 14 15 16 17	18	19 20 21 22 23	24 25 26 27	28 29	30	31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
D	OMAGZREC	R		30		X	
D							

If individual fields are accessed in the input buffer area, those fields must be described in detail.



x	REFERENCE	x CODE	LOCATION	x LENGTH	x DP	x TYPE	VALUE OR PICTURE
7	8 9 10 11 12 13 14 15 16 17	18	19 20 21 22 23	24 25 26 27	28 29	30	31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
D	OMAGZREC	R		30			
D	ACCTNO	F	0	20		U	
D	TOTAL	F	20	10		U	
D							

The work area must also be defined on Data Layout Sheets. These sheets are input to the compiler after all file and record descriptions. The amount of detail in the definitions depends upon the number of fields to be accessed.



x	REFERENCE	x CODE	LOCATION	x LENGTH	x DP	x TYPE	VALUE OR PICTURE
7	8 9 10 11 12 13 14 15 16 17	18	19 20 21 22 23	24 25 26 27	28 29	30	31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
D	UPDATEAREA	A		30			
D	ACCTNO	F	0	20		U	
D	TOTAL	F	20	10		U	
D							

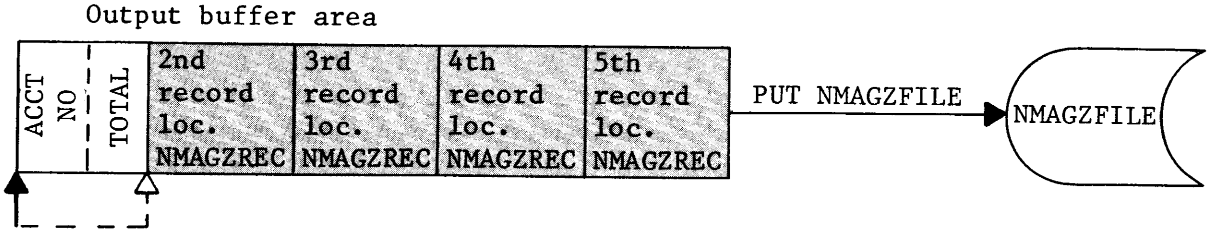
Both the record and the work area definitions may use the same field reference names; however, instructions referring to these fields must use a qualifier. The use of a qualifier insures that the proper field is affected by the instruction. Below are some examples of instructions using qualifiers:

● MOVE CONSTANT, OMAGZREC.TOTAL	This instruction moves the contents of an area called CONSTANT into the TOTAL field of OMAGZREC in the buffer area.
● MOVE CONSTANT, UPDATEAREA.TOTAL	This instruction moves the contents of an area called CONSTANT into the TOTAL field of the record in the work area called UPDATEAREA.

- Output

To output records to a file, the programmer constructs the record in the output buffer area and then specifies PUT File Reference. Assuming the File Reference name is NMAGZFILE, the programmer specifies PUT NMAGZFILE. When this instruction is executed, the record constructed in the output buffer is no longer available to the program. The next record location is available for the construction of another record. The PUT instruction that fills the buffer area causes the software to output the entire block to the file. (Assume for the following example that the record name is NMAGZREC.)

FIRST EXECUTION OF PUT

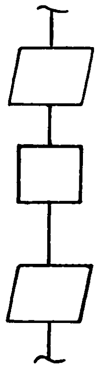


- The programmer constructs the record in the output buffer area.
- The programmer specifies PUT NMAGZFILE.
  - The previously constructed record is no longer available to the program.
  - The next record location in the output buffer area is available to the program.
  - When the output buffer area is full, the software automatically outputs the entire block to the file.

Although the instruction specifies PUT NMAGZFILE, the record (NMAGZREC) is put away. The PUT instruction may therefore be interpreted as follows:

PUT NMAGZFILE = PUT an NMAGZREC into NMAGZFILE.

The record has the name and the format specified on the Data Layout Sheets. Any instructional reference to the record affects the data in the output buffer area.

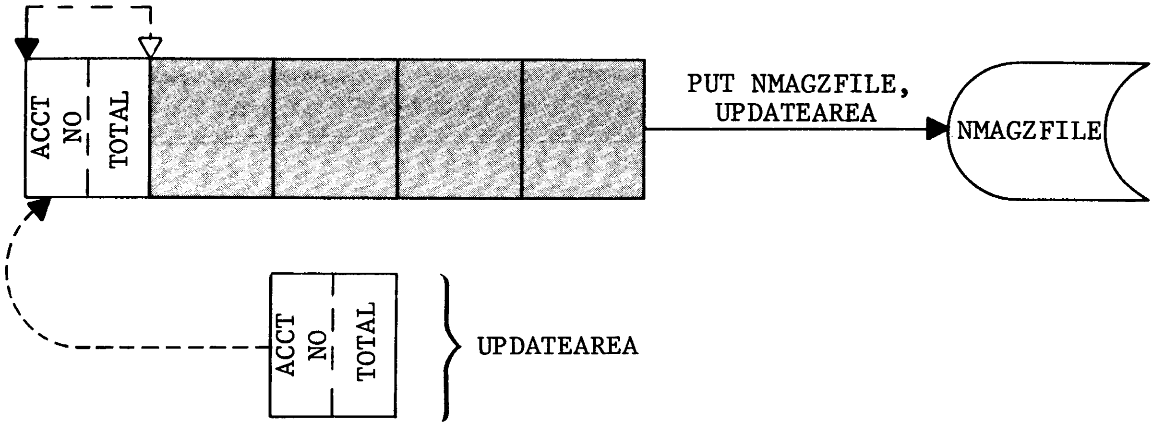


GET CARDFILE	GETs a CARDREC from CARDFILE.
ADD INAMT,TOTAL	ADDs the contents of the INAMT field of CARDREC into the TOTAL field of NMAGZREC.
PUT NMAGZFILE	PUTs the record away and makes the next record location in the output buffer area available to the program.

- Output with a Work Area

The programmer may name a work area in the second operand of the PUT instruction. In this case, the record is moved from the named work area into the current location of the output buffer area. Since the next record location always becomes available immediately after a PUT instruction is executed, the record in the output buffer is no longer available to the program. Since the move does not change the data in the source field, the record remains available in the work area.

PUT SPECIFYING A WORK AREA



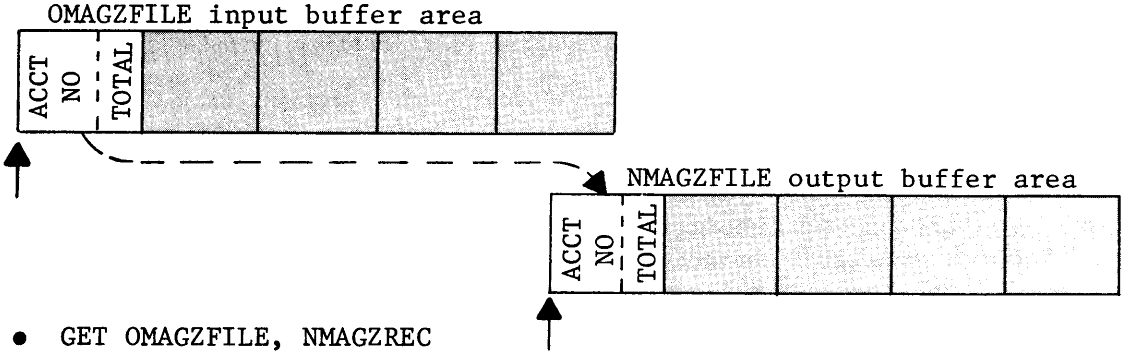
- PUT moves the record from the work area into the buffer and makes the next record location available to the program.

The record and work area must be defined on Data Layout Sheets as previously explained under Input with a Work Area.

- Further Use of the Work Area Operand

The work area operand may be used to move a record from the current record location in the input buffer area to the current record location in the output buffer area. This is accomplished by naming the record as the second operand in the instruction as illustrated in the following examples:

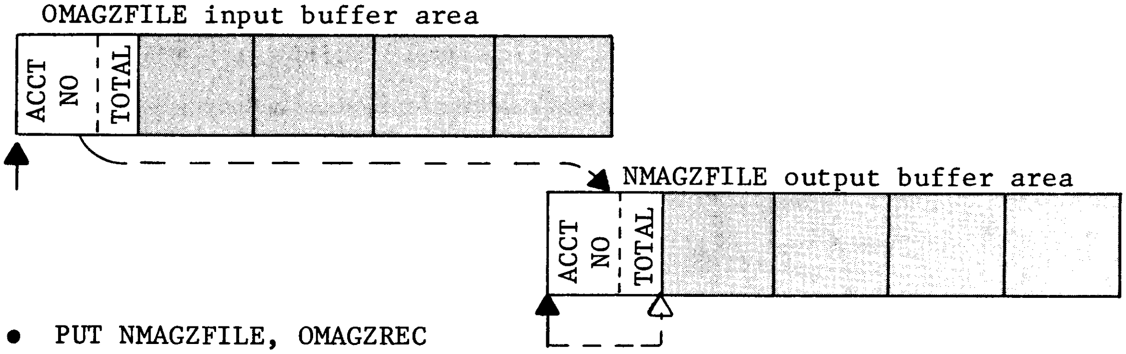
MOVING A RECORD FROM BUFFER TO BUFFER USING GET



- GET OMAGZFILE, NMAGZREC

GET a record from OMAGZFILE and place it in the current record position of the output buffer area for NMAGZFILE. (NMAGZFILE and NMAGZREC are associated by the software.) The record is available to the program in both the input and output buffer areas.

MOVING A RECORD FROM BUFFER TO BUFFER USING PUT



- PUT NMAGZFILE, OMAGZREC

PUT a record into the current record location of the output area for NMAGZFILE. The record to be placed there is OMAGZREC, the current record in the OMAGZFILE input buffer area. (OMAGZREC and OMAGZFILE are associated by the software.) The record is available to the program in the input buffer area but not in the output buffer area. The next record location in the output buffer area always becomes available immediately after a PUT instruction is executed.

RECORD TYPES

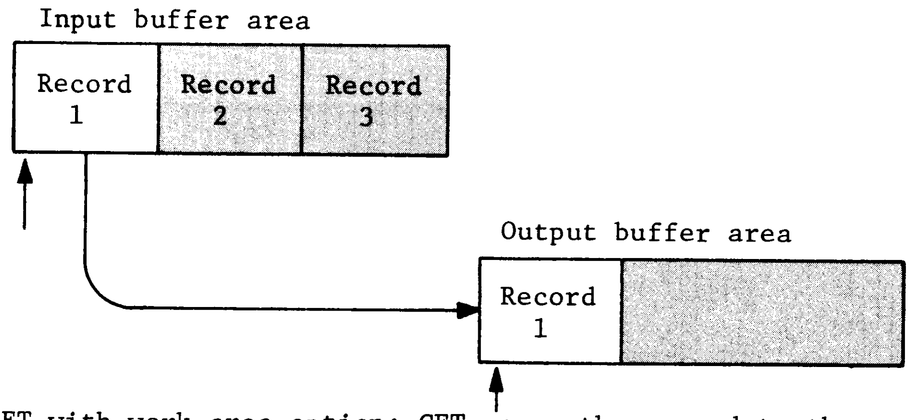
Files may contain either fixed-length or variable-length records, but not both. Records are considered fixed-length if all the records within a file are the same length and do not change in length during processing.

Records are considered variable-length if the records within a file change in length during processing, or if all the records in a file are not the same length, i.e. two or more fixed-length records of different lengths are contained in the same file. This manner of handling multiple-format records within a file allows the I/O instructions to automatically manipulate such records without special coding considerations.

Processing Fixed-Length Records

Fixed-length records may be processed in either the input buffer or the output buffer; however, a work area may be used if desired. If the record is processed in the input buffer, it may then be moved to the output buffer by using PUT with the work area option. If the record is processed in the output buffer, it may first be moved to that buffer by using the GET instruction with the work area option.

UPDATING FIXED — LENGTH RECORDS IN THE BUFFER



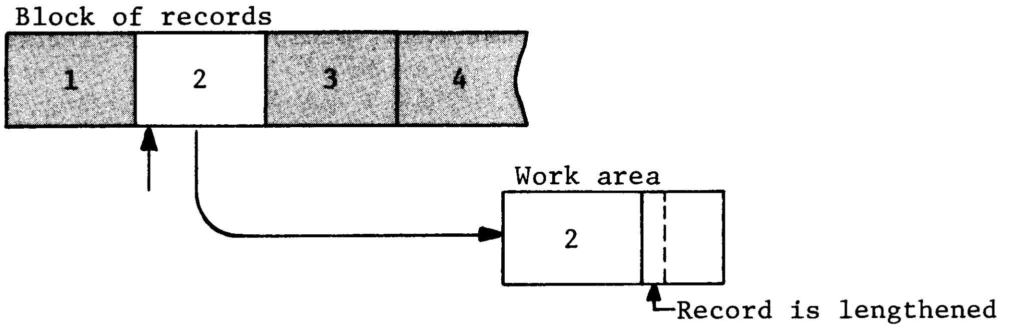
GET with work area option: GET moves the record to the output buffer for updating. After the record is updated, PUT must be executed.

PUT with work area option: PUT moves the updated record from the input to the output buffer.

Processing Variable-Length Records

Variable-length records should be processed in a work area. Changing the length in the input buffer could destroy the adjacent records; changing the length in the output buffer could cause inefficient use of magnetic media storage.

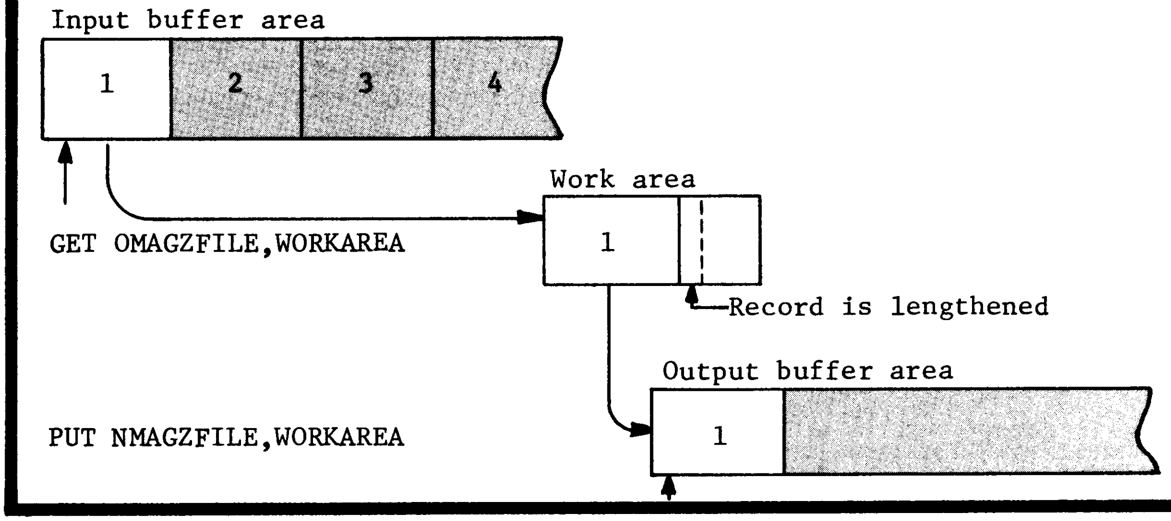
CHANGING THE RECORD LENGTH



Record 2 is moved to a work area and lengthened. If this were done in the buffer, part of record 3 would be destroyed.

Records can be moved to the work area by using the GET instruction with the work area option, updated, and then moved to the output buffer by using the PUT instruction with the work area option.

UPDATING RECORDS IN A WORK AREA

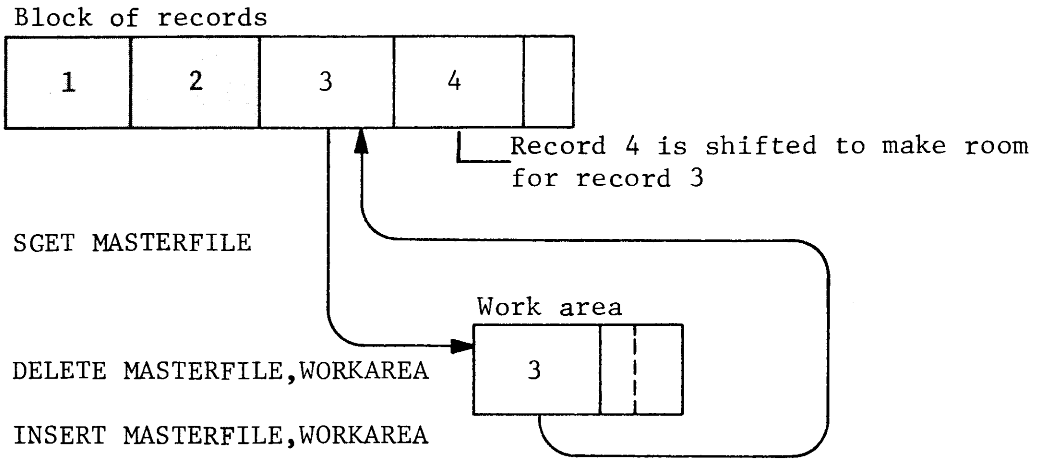


```
GET OMAGZFILE,WORKAREA
```

```
PUT NMAGZFILE,WORKAREA
```

Certain types of files permit records to be input, updated, and written back into the same position in the file. When the records in these files are variable-length, they must be deleted from the file and moved from the buffer to a work area, using the DELETE instruction. The updated record is moved from the work area back into the same buffer and reinserted into the file, using the INSERT instruction.

MOVING A RECORD USING INSERT AND DELETE



```
SGET MASTERFILE

DELETE MASTERFILE,WORKAREA

INSERT MASTERFILE,WORKAREA
```

SGET: presents the records until the desired one is located.

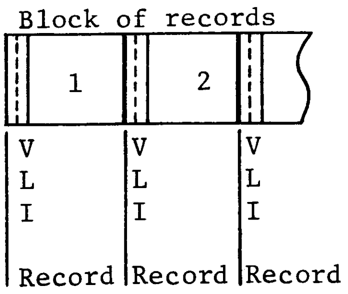
DELETE: removes the desired record from the file and automatically moves it to the work area.

INSERT: places the altered record from the work area into the buffer. Other records are automatically shifted as necessary to allow room for record 3 without destroying record 2 or record 4. A flag is set which informs the software to automatically write the block back into the file.

- Defining Variable-Length Records

All variable-length records must have a variable-length indicator (VLI) associated with them. This VLI, which must occupy the first field in each record, contains a number that indicates the total number of characters currently in the record (including the VLI itself). The GET, PUT, INSERT, DELETE, and other I/O instructions use the VLI to manipulate the proper number of characters. The programmer is responsible for updating the VLI when the record length is changed. The length and data type of the VLI field varies with the type of peripheral used; however, it is normally a 2-character binary field. (For further information on the VLI, see the sections of this manual that deal with specific peripherals.)

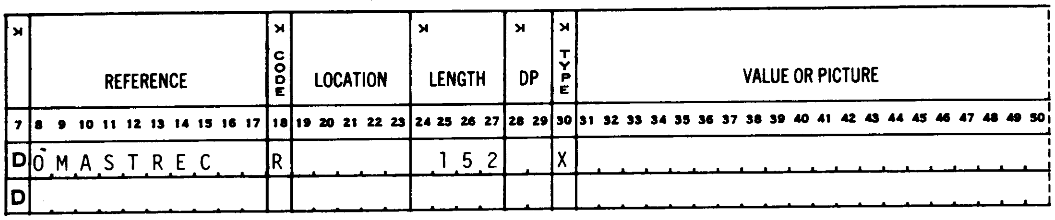
THE VARIABLE – LENGTH INDICATOR



The variable-length indicator must be defined as the first field in the record.

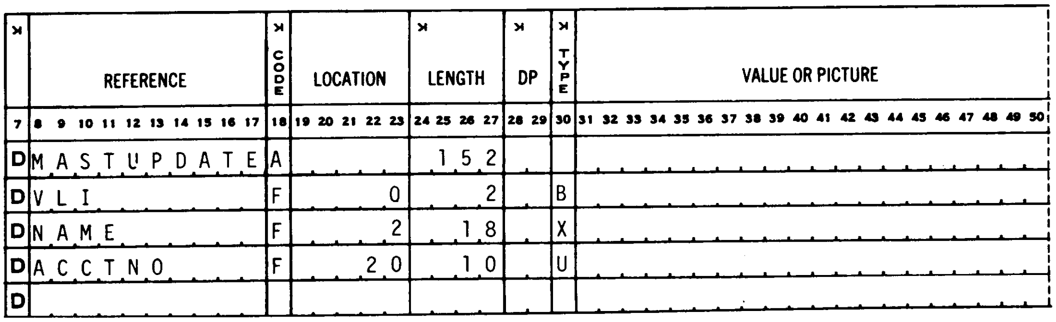
When defining a variable-length record for the buffer or a work area, the length column must contain the maximum length possible for that record. When variable-length records are used, and therefore updated in the work area, the record definition for the buffer generally defines only the record length. The work area definition must include the necessary field definition.

DEFINING THE BUFFER AND WORKAREA



	REFERENCE	CODE	LOCATION	LENGTH	DP	TYPE	VALUE OR PICTURE
7	8 9 10 11 12 13 14 15 16 17	18	19 20 21 22 23	24 25 26 27	28 29	30	31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
D	0MASTREC	R		152		X	
D							

The record definition for the buffer (input immediately following the File Specification Sheets) need only show the record name, maximum length, and type.

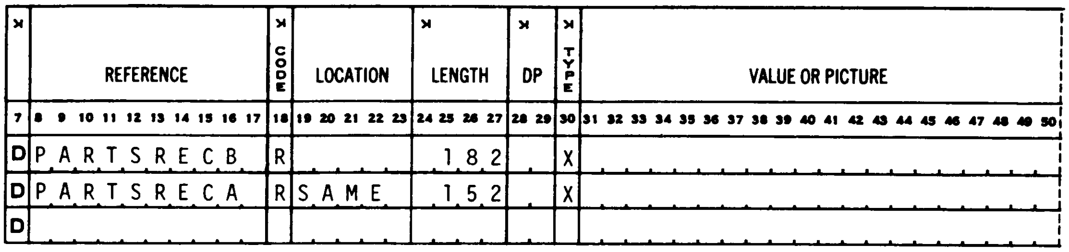


	REFERENCE	CODE	LOCATION	LENGTH	DP	TYPE	VALUE OR PICTURE
7	8 9 10 11 12 13 14 15 16 17	18	19 20 21 22 23	24 25 26 27	28 29	30	31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
D	MASTUPDATE	A		152			
D	VLI	F	0	2		B	
D	NAME	F	2	18		X	
D	ACCTNO	F	20	10		U	
D							

The record definition for the work area (input with constants and reserved area definitions) must show area name, maximum length, type and field definitions.

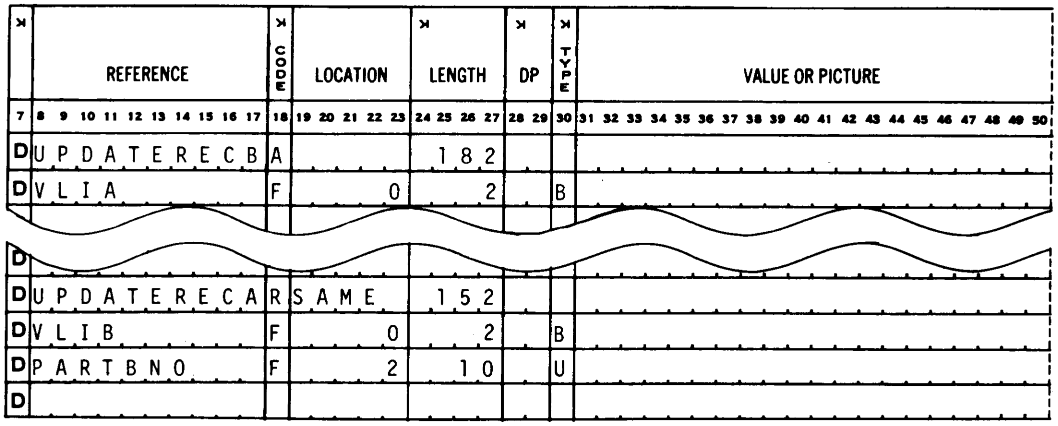
When defining records that are of different lengths but in the same file, the longest record must be defined first, and SAME must be used in the location column for the second and successive records. Since a work area should be used, the record definition for the buffer need not be detailed; however, the work area definition must include the necessary field definitions. (For further information on using SAME in the location column, see NEAT/3 REFERENCE MANUAL, INTRODUCTION AND DATA, tab 3, "Data Layout Sheets.")

DEFINING THE BUFFER – MULTIPLE FORMAT RECORDS



	REFERENCE	CODE	LOCATION	LENGTH	DP	TYPE	VALUE OR PICTURE
7	8 9 10 11 12 13 14 15 16 17	18	19 20 21 22 23	24 25 26 27	28 29	30	31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
D	PARTSRECB	R		182		X	
D	PARTSRECA	R	SAME	152		X	
D							

The record definition for the buffer need only show the record names, lengths, and types. The largest record must be defined first, and SAME must be entered in the location column of the second record.



	REFERENCE	CODE	LOCATION	LENGTH	DP	TYPE	VALUE OR PICTURE
7	8 9 10 11 12 13 14 15 16 17	18	19 20 21 22 23	24 25 26 27	28 29	30	31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
D	UPDATERECB	A		182			
D	VLIA	F	0	2		B	
D							
D	UPDATERECA	R	SAME	152			
D	VLIB	F	0	2		B	
D	PARTBNO	F	2	10		U	
D							

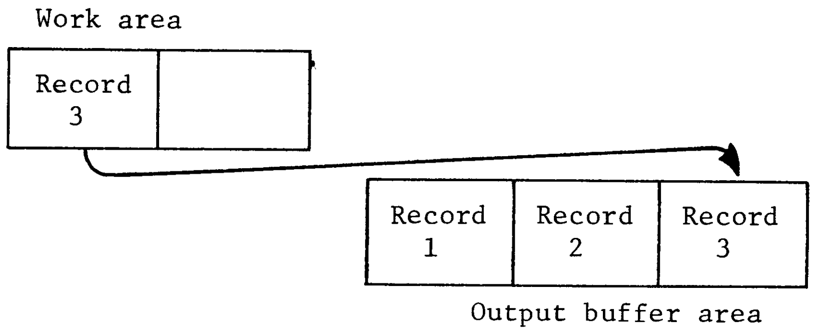
The record definition for the work area must show area names, lengths, types, and field definitions. The longest record must be defined first, and SAME must be entered in the location column of the second.

During processing, the program examines either the contents of the VLI or a key within the record to determine the proper record format. The program can then act upon the record in the appropriate manner.

- Maximum Packing of Records

To reduce the number of I/O operations and to make better use of magnetic media storage space, maximum packing of variable-length records should be requested on the File Specification Sheets. When maximum packing is requested, records must be updated in a work area. The PUT instruction compares the actual length of each updated record to the space remaining in the buffer. If the space is sufficient to receive the record, that record is placed in the buffer. If the space is not sufficient, the block is written out and the record is placed in the next block.

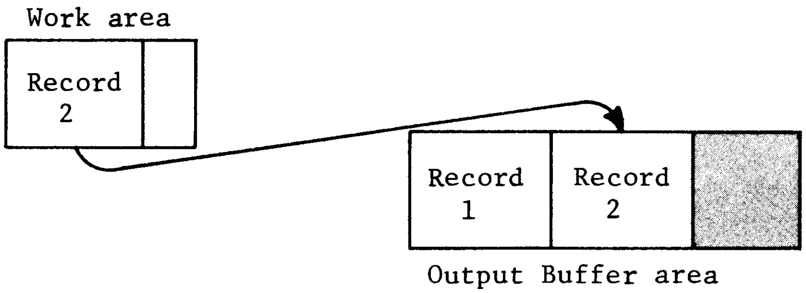
MAXIMUM PACKING



When maximum packing is requested, each record is compared to the remaining space in the buffer. If the record will fit, it is placed in the buffer.

If maximum packing is not requested, the space remaining in the buffer is compared to the maximum length of the largest variable-length record after each execution of the PUT instruction. If the space is insufficient to hold the largest defined record, the buffer is written out with that space empty.

NO MAXIMUM PACKING



When maximum packing is not requested, the space in the buffer is compared to the maximum length record. If the space is smaller, the block is written out with that space empty.

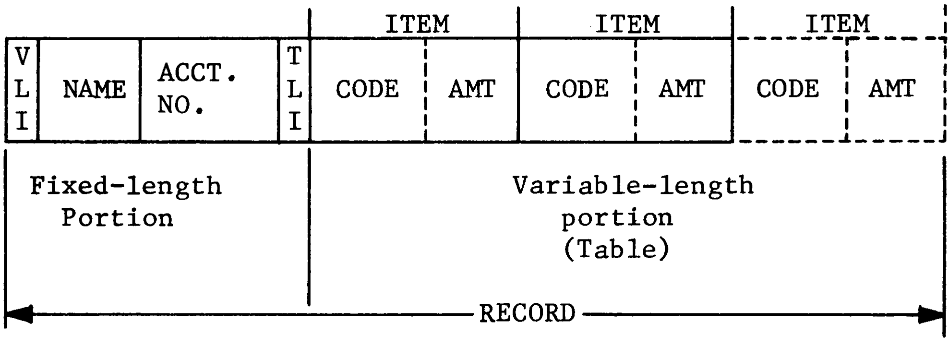
Maximum packing requires more memory space for the work area and extra coding. In a large program, maximum packing may not be desired if memory space is critical. However, maximum packing is normally used.

- Tables in Variable-Length Records

Since all fields are fixed in length, records expand and contract in fixed-length segments (Items). Tables provide a convenient method of handling fixed-length data.

Generally when tables are used in a variable-length record, the record is divided into two parts: a fixed-length part and a variable-length part (the table).

TABLE IN A RECORD



Each table has a table-length indicator (TLI) associated with it. This indicator, which is similar to the VLI, is updated automatically each time a fixed-length item (CODE AMT) is added to or removed from the table portion of the record. The table instructions also update the VLI automatically at this time, thereby relieving the programmer of the task. For complete details on using tables in records, see NEAT/3 REFERENCE MANUAL, INSTRUCTIONS, tab 4, "Table Concepts."

* * * *

FILE TYPES

Source, destination, source-destination, or piggyback files may be used with the NCR Century Systems. A description of each type of file and a list of the applicable peripherals follow:

- Source File

  A source file is one from which data may be input only. It is a previously-created file used to supply information to a program. This information may be used as reference material in solving a problem or may be updated and written out to another file. Card readers and paper tape readers are input devices that deal with source files only. When files on magnetic peripherals (such as disc or magnetic tape) are being read only, they are designated source files.

- Destination File

  A destination file is one to which data is output only. Most files are originally created as destination files. Card punches, paper tape punches, and printers are output devices that deal with destination files. When files on magnetic peripherals (such as disc or magnetic tape) are being written only, they are designated destination files.

- Source-Destination Files

  A source-destination file is one from which data may be input and to which data may be output. On the NCR Century Systems, the disc unit, the CRAM unit, and the optional card reader/punch unit may use source-destination files.

  Records are generally read from the disc source-destination file, updated, and then written back in the same location. Since the old record is replaced by the new record, the records in the old file are destroyed.

  Records are read from a punched card source-destination file as the card moves through the read station to the punch station. The card stops at the punch station where more information may be punched into the same card.

- Piggyback Files

  A piggyback file is a type of destination file. Normally, once a destination file is created, it cannot be extended unless the existing portion is first copied. Declaring the file a piggyback file instructs the software to find the end of the existing file and perform the necessary steps to allow for extension of that file.

The following is a table showing the file types and the peripheral units to which each file type applies:

PERIPHERAL – FILE TYPE TABLE [illegible]

		FILE TYPE			
		SOURCE	DESTINATION	SOURCE-DESTINATION	PIGGYBACK
PERIPHERAL TYPE	Punched card	reading a card file	punching a card file	reading and punching in same card file	NA*
	Punched tape	reading a tape file	punching a tape file	NA*	NA*
	Magnetic tape	reading a tape file	writing a tape file	NA*	extending an existing file
	Magnetic disc	reading a disc file	writing a disc file	reading and writing on the same file	extending an existing file
	CRAM	reading a CRAM file	writing a CRAM file	reading and writing on the same file	extending an existing file
	Printer	NA*	printing	NA*	NA*

*NA = Not Applicable

FILE ORGANIZATION

Two basic file organizations are provided: standard and chained. Standard file organization, used for Father-Son processing, Selective-Serial processing, and reading or creating files in sequential order, has limited provision for random access. Chained file organization handles sequential or random processing and provides for overflow and insertion or deletion of records.

- Standard File Organization

  In standard file organization, blocks of records are presented to the program in the sequence in which they were recorded. The records within the blocks may or may not be in numeric or alphabetic order. For example, the input records to a sort run are unsorted. The output of the run is in either numeric or alphabetic order for input to another run.

  Standard file organization does not incorporate a specified overflow area. To handle file expansion, the programmer must leave sufficient room in a section and, upon filling this section, open a new one. A limited degree of random access may be employed when standard files are on the disc.

Standard File Father-Son Processing

In Father-Son processing, a new master file, known as the Son, is created and the old master file, known as the Father, is retained. Each master record from the old master file (Father) is read into memory where its key is compared to the key of a transaction record. If the keys do not compare equally, the master record is not altered but is written out to the new master file (Son). If the keys compare equally, the master record is first updated by the data in the transaction record and is then written to the new master file. The GET instruction is used to read in the two source files, the transaction file and the old master file; the PUT instruction is used to write out the destination file, the new master file.

Initially the Father file is created as a destination file through the use of a specially written one-time conversion program. Assume for the following explanation that this file has been created and is called MAGZFILE. Also assume that another source file which contains the necessary transaction records exists.

During the update run, all records from the MAGZFILE are read into the processor in blocks. Each record in each block is presented to the program sequentially, updated if necessary, placed into blocks again, and written out as a new version of MAGZFILE. Since a new version of MAGZFILE is created, the original version becomes the old master file, and the new version is the new master file; however, both versions carry the file name MAGZFILE.

To ease the coding of I/O instructions, two names are associated with each file: File Reference and File Name. The File Name is the actual name of the file, and the File Reference is the name that appears in the GET and PUT I/O instructions.

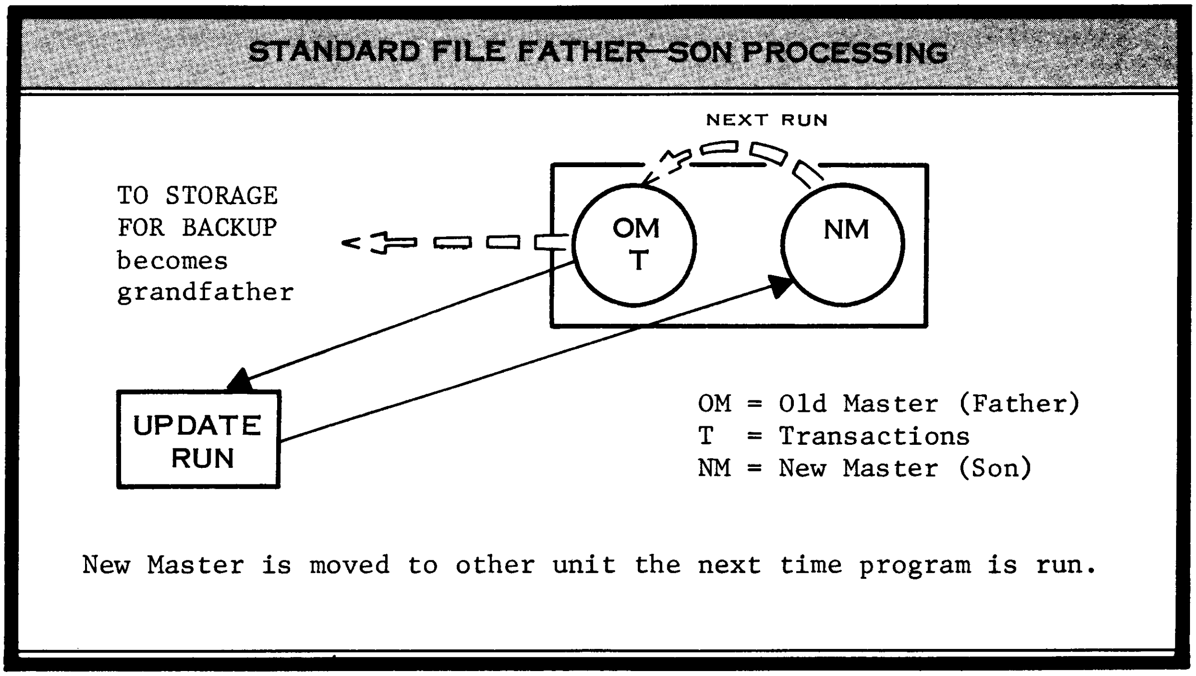
Both of these names are requested on the File Specification Sheets along with other pertinent file information. The programmer must specify, for example, that the source file is on a certain unit and has the File Name MAGZFILE and the File Reference OMAGZFILE (O for old). Here, the programmer is really stating that MAGZFILE on a certain unit should be accessed when OMAGZFILE is specified.

The programmer also specifies that the destination file being created is on a different unit, its File Name is also MAGZFILE, but its File Reference is NMAGZFILE (N for new). Using the PUT instruction, the programmer can now specify PUT NMAGZFILE. By using information from File Specification Sheets for this file, the software knows that the desired file is actually MAGZFILE on a different unit with a different data, etc., associated with it. The name MAGZFILE is actually recorded in a label for each file.

The operator must mount the proper version of MAGZFILE on the proper unit. If the wrong version is mounted accidentally, date and version protection prevent the file from being destroyed. File protection is covered later in this publication.

Each time a new Son file is created, the old Son becomes the Father. All previous versions are forced back one generation: the previous Father becomes the Grandfather; the previous Grandfather becomes the Great Grandfather, etc. The Father and Grandfather are generally kept in storage for backup, but as many versions as desired may be kept.

STANDARD FILE FATHER—SON PROCESSING



OM = Old Master (Father)
T = Transactions
NM = New Master (Son)

New Master is moved to other unit the next time program is run.

- Chained File Organization

Chained file organization, as the name implies, is a technique of automatically linking blocks of records together. A block is an organizational unit within a file that may contain a single record or multiple records. Chained files, which can be used with either selective serial processing or random processing, provide for both the insertion and deletion of records. When specified, chained files also provide for an overflow area. The use of an overflow area is discussed later. Chained file organization is optional when working with magnetic disc files, but is the only technique available with CRAM files. (It is never used with magnetic tape files, which use only standard file. Father-Son processing.) For the discussion in this section, magnetic disc files are assumed for all examples.

Chained file organization is generally confined to source-destination files, but may be associated with source or destination files; i.e., a chained source-destination file may be created as a chained destination file or used as a source file in a different run. Chained source-destination files may be processed either sequentially or randomly.

A chained file consists of blocks of records automatically connected in sequence by the software. The first 10 characters in each block are used by the software to link that block to both the next and the previous blocks in the file. Therefore, the maximum number of program-usable characters in a block for a chained file is reduced by 10. For example, the programmer has 502 characters available in a 512-character block. Block size and length are discussed later.

A chained file is generally divided into two areas: a main file area and an overflow area.

- Main File Area

  The main file area, which is always present in a chained file, may consist of 1 or more sections, with each section containing a specific number of sectors as determined by the programmer.

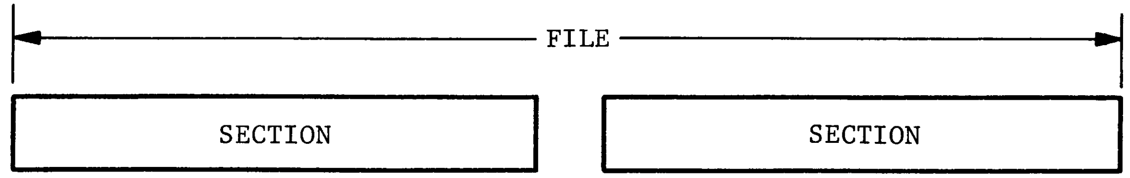
  In addition, the main file area is further divided into smaller areas called buckets. As a logical division of the main file area, the bucket concept saves program execution time by confining record search and record pushdown (when inserting new records) to the limits of the bucket area, rather than searching or pushing down through the entire file or file section. The size and number of buckets in a given file is determined from information entered on the file specifications worksheet by the programmer. The programmer can decide to include the entire area of the main file in a single bucket, or to divide it into several buckets.

  The file specifications worksheet also includes the definitions of several other units of organization within the file, which are related (and in effect restrictive) to bucket organization. Before bucket organization is discussed further, these additional file structures are outlined briefly and illustrated on the next several pages:

  1. Section

     A file section is a logical division of a file and may be a physical division as well, such as multi-section files contained on more than one disc pack. In addition, a multi-section file can exist on a 4-zone pack, one section per zone.

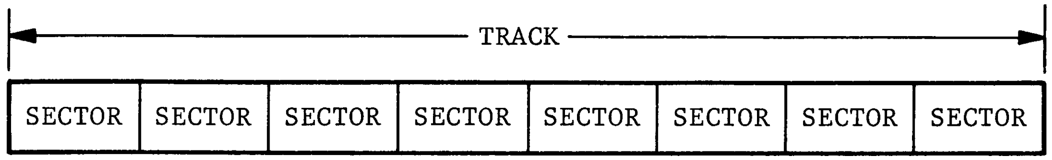
Each section of a file contains a specific number of sectors which are allocated by the programmer.



- Sections of a file may be physically separate, as on two disc packs, or separated by zones on a 4-zone pack, one section per zone.

2. Sectors

On magnetic disc, a track is the area covered by one read/write head during one revolution of the disc. Each track is divided into eight sectors. Each sector can contain up to 512 characters of data.

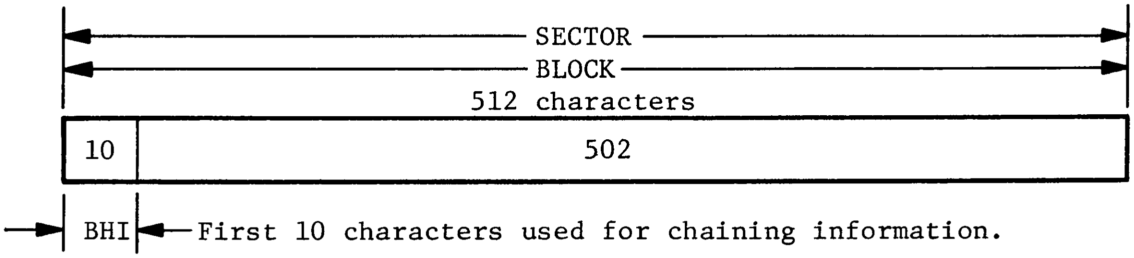


- Each sector of a track may contain up to 512 characters.

3. Blocks

Records are usually handled in groups called blocks for ease of processing. For example, to sequentially input a block of records from a file, the programmer need only specify one GET instruction.

A block may contain a single record or multiple records. The maximum block size on the integrated disc is 512 characters, or one sector. The first 10 characters of each block are used by the software for chaining information and are called the Block Header Indicator (BHI). These 10 characters must be included when the programmer defines the maximum block length.



First 10 characters used for chaining information.

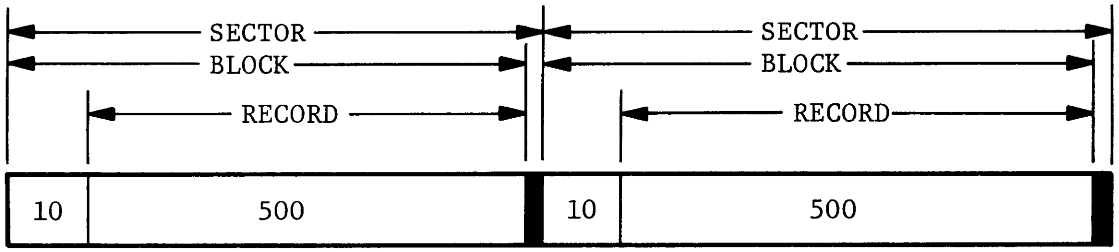
The maximum number of program-usable characters in a block for a disc chained file is 502 characters.

Optional common trunk disc units on the NCR Century 100 and all disc units on the NCR Century 200 have maximum block lengths exceeding 512 characters.

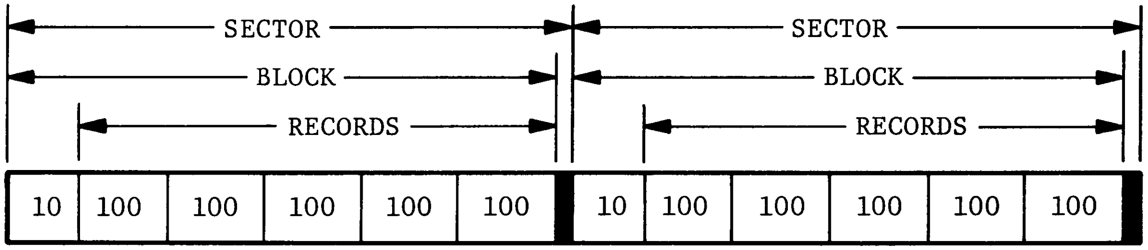
4. Records

Records may be fixed or variable in length. The maximum size record on the integrated disc is 512 characters, or one sector. When chained files are used, however, this number is reduced by 10 characters for the BHI.

Therefore, a block can contain one record of 502 characters, or any number of records, if the record length is divisible into 502. Any characters remaining in the sector are not used.



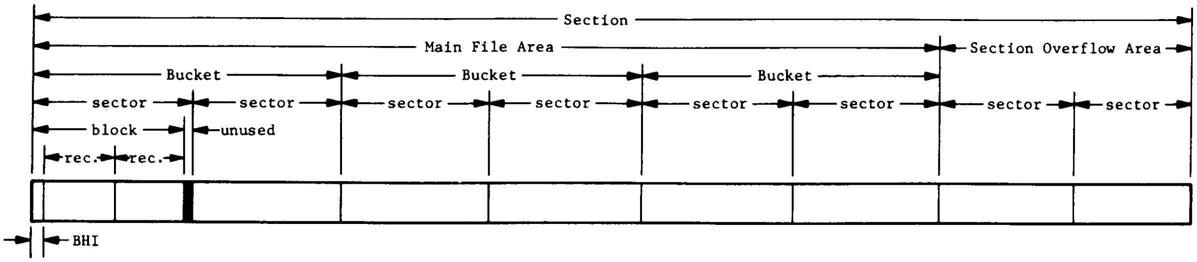
- The first 10 characters contain information that chains the two records together.
- The record length is 500 characters.
- The last two characters (of each sector) are not used.



- The first 10 characters contain information that chains the two blocks together.
- The block length is 500 characters and contains 5 records, each 100 characters in length.
- The last two characters are not used.

5. Buckets

A bucket is a logical division of a section of a file, with each bucket containing a predetermined number of sectors. The following illustration shows the relative position of buckets in chained file organization.



- The section length contains 8 sectors, or 4096 characters.
- The main file area contains 6 sectors, or 3072 characters.
- The overflow area contains 2 sectors or 1024 characters.
- The main file area is divided into 3 buckets, each containing two sectors, or 1024 characters.
- The first sector of the first bucket includes a sample block.
  - The block length includes the Block Head Indicator (BHI) of 10 characters and two records, each 250 characters long, for a total of 510 characters.
  - 2 characters at the end of the sector are not used.
  - This format would be repeated in other sections.

The bucket concept of file organization offers several advantages in terms of program execution time by giving the program an indication that an error or exception exists.

- When a record search is made, a branch is taken if the record is not found by the time the end of the bucket is encountered. The program does not have to search the entire file.

- When a new record is inserted, record pushdown is limited to the bucket, rather than requiring pushdown within the entire file or file section.

- When a bucket becomes full, records overflowing from the bucket are placed in the overflow area and are chained back to the last record in the bucket.

The RGET instruction may be used to randomly access a bucket, and the GET or SGET instruction to sequentially step through the bucket. SGET causes a branch to the routine referenced in the second operand when any of the following occur:

- End-of-bucket
- End-of-section
- End-of-file
- Empty (null) block is read.

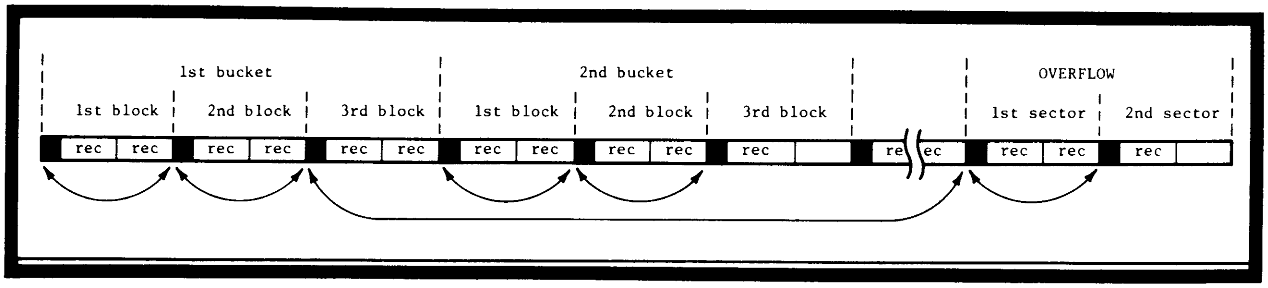
- Overflow Area

The overflow area accepts records overflowing from buckets in the main file area. The need and size of the overflow area is determined by the programmer after considering file size, expected insertion/deletion rate and expected record growth. When the size of the overflow area is determined, it is assigned a specific number of sectors. Consider the following examples:

For the purpose of these illustrations, assume the following information:

- Each bucket contains 3 sectors.
- Each sector contains blocking, as follows:
  - Block Header Indicator (BHI) for chaining information, 10 characters.
  - Two records, each containing 251 characters.
- Chaining is indicated by double-headed arrows, since each block of records is chained to both previous and following blocks of records.

- Example 1

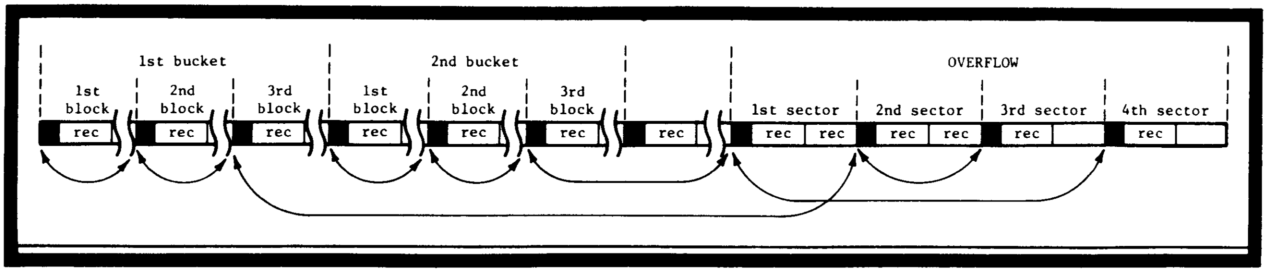


Chaining in the first bucket is as follows:
- Blocks 1 and 2 are chained together.
- Blocks 2 and 3 are chained together.
- Block 3 is chained to the first sector in the overflow area.
- Sectors 1 and 2 in the overflow area are chained together.

Chaining in the second bucket is as follows:
- Blocks 1 and 2 are chained together.
- Blocks 2 and 3 are chained together.
- Block 3 contains only one record. Since the bucket is not full, chaining to the overflow area has not occurred.

Overflow occurred as follows:
- Block 3 in the first bucket is filled. Software locates the first available sector (1) in the overflow area.
- Sector 1 in the overflow area is filled. Software locates the next available sector (2) in the overflow area.

- Example 2



Chaining in the first bucket is as follows:

- Blocks 1 and 2 are chained together.
- Blocks 2 and 3 are chained together.
- Block 3 is chained to the second sector in the overflow area.
- Sectors 2 and 3 in the overflow area are chained together.

Chaining in the second bucket is as follows:

- Blocks 1 and 2 are chained together.
- Blocks 2 and 3 are chained together.
- Block 3 is chained to the first sector in the overflow area.
- Sectors 1 and 4 in the overflow area are chained together.

Overflow occurred as follows:

- Block 3 in the second bucket is filled. Software locates the first available sector (1) in the overflow area.
- Block 3 in the first bucket is filled. Software locates the next available sector (2) in the overflow area.
- Sector 2 in the overflow area is filled. Software locates the next available sector (3) in the overflow area.
- Sector 1 in the overflow area is filled. Software locates the next available sector (4) in the overflow area.

- Selective Serial Processing

  This type of file processing is only used with units that handle source-destination files: for example, the disc. A source-destination file is one that can be read from and written into during the same run. Each time the file is updated, the old master itself becomes the new master file. The source-destination master file used in selective serial processing may be either a chained file or a standard file.

  Transaction records are presorted in a separate file and stored in the same sequence as the master file. The GET instruction is used to sequentially read in both the master file and the transaction file. The WRITSP instruction is used to write updated blocks back into the same location in the master file from which they were read.

  Generally the master file is created as a destination file through the use of a one-time conversion program. Assume for the following explanation that a file, called ACCTFILE, has been created. It is the source-destination file used in the update run. Also assume that another source file containing the necessary transaction records exists. During the update run, all records from ACCTFILE are read into the processor in blocks. Each record in each block is presented to the program sequentially. If updating occurs, the block is written back into the file from which it was read.

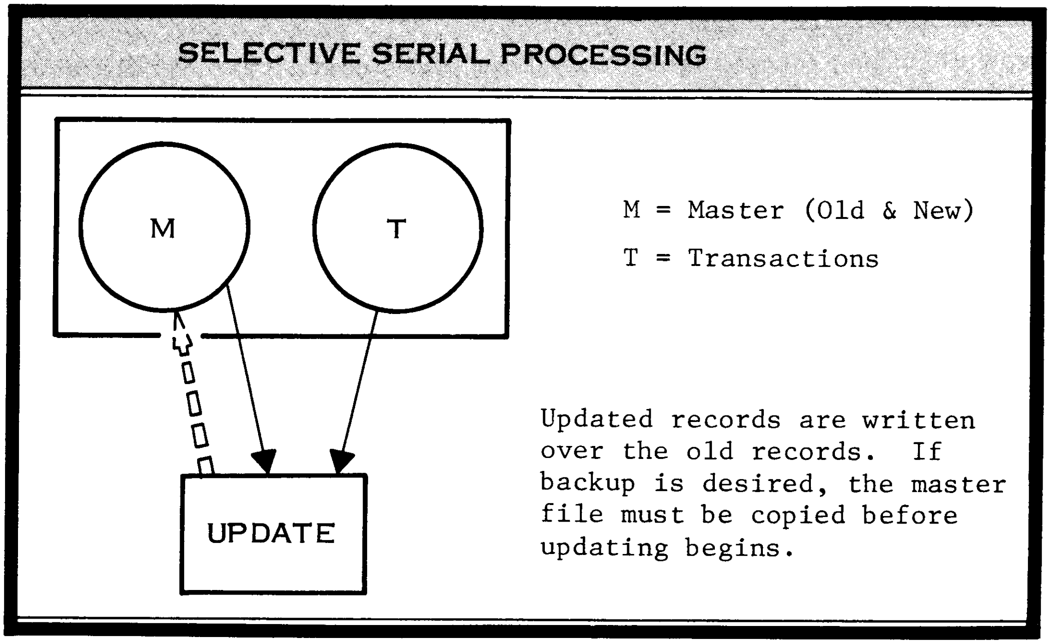
Since reading from the disc does not destroy the information on the disc, only the updated blocks need be written back. The WRITSP instruction is used to write back only those blocks which have been updated. The programmer places WRITSP at the end of that portion of the program responsible for updating. WRITSP is then executed only when a record in a block has been updated. Under these conditions, WRITSP guarantees that the updated blocks are automatically written back after the completion of all the processing required on all records in the block.

Selective serial processing is more flexible with a chained source-destination file than with a standard source-destination file since the chained file software has the ability to chain blocks to an overflow area and to find these blocks on demand. When using a chained file, the size of the record and the size of the block can be altered during update. Also, new accounts may be inserted into the file or obsolete accounts may be deleted from the file during the update run.

Because the structure of the file changes during update the old master file is destroyed. For backup, the master file must be copied periodically. This copy can be done as a straight copy of the file or during a Father-Son run used to perform some maintenance on the file.

Standard file selective serial processing has greater limitations. (1) Since the block is written back into the exact area from which it was read, the size of the record and the size of the block cannot be altered. (2) No insertions of new records or deletions of old records are allowed during the update run.

Selective serial processing with a standard file is best used if the records are fixed length and insertions and deletions are handled in a separate Father-Son run.



SELECTIVE SERIAL PROCESSING

M = Master (Old & New)

T = Transactions

Updated records are written over the old records. If backup is desired, the master file must be copied before updating begins.

* OPENING FILES

Before data can be output to or input from a file, that file must be conditioned; that is, opened. For ease in file handling and ease in program coding, the programmer ususally allows files to be opened by the software when the program is loaded into memory. Any file that does not have an OPEN instruction associated with it is opened when the program is loaded.

If the programmer wishes to delay opening a file or wishes to take advantage of certain options, he may use an OPEN instruction. See NEAT/3 REFERENCE MANUAL, INSTRUCTIONS, tab 1, "OPEN Instructions."

- Common Opening Procedures for Source Files

  Whenever a source file that is to be processed sequentially is opened, all the input buffer areas for that file are filled except one. When the first GET instruction is issued, the last buffer is filled while the first record is presented to the program. If that file was on a magnetic media peripheral, the file is first checked for the proper name and version number. If the proper file name or version cannot be located on the named peripheral, the software displays a message to the operator. The operator may mount a new disc pack or reel; or, in certain cases, he may declare that a new file is to be created. The action taken depends on the type of file and type of peripheral. If the file is on a punched paper peripheral, the open procedure is confined to a unit check.

- Common Opening Procedures for Destination Files

  Whenever a destination file is opened on a magnetic media peripheral, the software checks to insure that room exists on the disc pack or reel of tape. For a printer file, an operational check of the peripheral is made. For a punched paper peripheral file, the open is confined to a unit check.

  NOTE: Specific opening procedures for each peripheral are covered in the section dealing with that peripheral.

CLOSING a FILE

Before a program terminates, all files associated with that program must be closed. The programmer may use an instruction to close a file, or he may let the software close it for him. When the program reaches the FINISH instruction, the software automatically closes all files that are still open.

For ease in file handling and ease in program coding, the programmer usually allows his files to be closed for him. However, he may use the CLOSE instruction if he desires control over the time of closing or if he desires to take advantage of one of the CLOSE instruction options. See NEAT/3 REFERENCE MANUAL, INSTRUCTIONS, tab 1, "CLOSE Instructions."

- Common Closing Procedures for Source Files

  When a source file is closed, little or no action occurs.

- Common Closing Procedures for Destination Files

  When a destination file is closed, all output buffers not yet empty are written out into their respective files. If the file is on disc, the software places the end-of-data sector number in the disc directory. If it is on magnetic tape, the software constructs end-of-file labels. If the destination file is on a printer, the paper is advanced to a standard position. If it is on a card punch, a card encoded END$ is punched to signal end-of-file.

NOTE

> Specific closing procedures for each peripheral are covered in the section dealing with that peripheral.

** RESCUE DUMPS

A rescue dump saves information concerning the state of the processor (the condition of internal flags, etc.) and a picture of memory. This feature provides the user with the ability to recover from unavoidable processing interruption without having to restart the run from the beginning. The programmer designates on the File Specification Sheets when a rescue dump is to occur (rescue point) as well as where the rescue dump data is to be stored. The software does the rest.

These automatic rescue dumps, provided by the software at programmer direction, may not be used in conjunction with runs that incorporate source-destination files.

There is a software routine called RESTART available to handle restarting a program from a rescue point.

- Rescue Point

  The rescue point (time at which a rescue dump is taken) is designated by the programmer. It must be in accordance with software regulations and good processing techniques. The software demands that rescue points occur only at the end of a file section, for example, at the end of a reel of magnetic tape. Since long punched card files are generally not divided into sections, the programmer may schedule special cards (coded RES$) to be placed in desired locations within the file. The software treats these cards in the same manner as an end-of-section.

  The programmer specifies the file which is to initiate rescue dumps for a run. Only one file should be assignea this option (generally the file that reaches end-of-section most often). If a punched paper tape file exists in a program it must initiate the rescue dump at end-of-reel. In this way a new reel may be used to correctly reposition the tape when restarting. If a source card file exists, the special RES$ cards should be used to simulate end-of-section and thereby initiate rescue dumps.

The following table illustrates the time at which a rescue may occur for certain peripherals:

PERIPHERAL - RESCUE POINT TABLE

PERIPHERAL	RESCUE TAKEN
Disc	At the end of each section of the file.
Magnetic Tape	At the end of each reel (section).
Punched Tape	At the end of each reel (section).
Punched Cards	At a RES$ card.

Storage of Rescue Dumps

The software stores the rescue dump either within the file that triggered the dump or in a standard rescue file, depending upon the type of peripheral that initiated the dump.

If a destination file on a magnetic media peripheral (disc, magnetic tape, etc.) reaches end-of-section and if that file was designated by the programmer to initiate rescue dumps, the dump is placed at the beginning of the next section of that file.

If a source file on a magnetic media peripheral or a source or destination file on punched paper peripherals reaches end-of-section and if that file was designated to initiate rescue dumps, the dump is placed in a standard rescue file.

The standard rescue file is a programmer-defined magnetic media destination file. It is used for the storage of all rescue dumps not initiated by the end-of-section for some other magnetic media destination file. If this file is on disc it may contain rescue dumps only; however, if the file is on magnetic tape, it may also contain normal file data. That is, a normal data file on magnetic tape may also serve as the standard rescue file.

Standard rescue file definition is covered in the sections of this manual that pertain to specific peripherals.

CRATCH FILES

scratch file is a magnetic media file that is used for the temporary storage f data. Generally such a file is obsolete at the end of the program that reated it or at the end of a series of related runs. This type of file is enerally date-protected in a manner which makes it obsolete the day after it s created; that is, the area occupied by the file may be reused the next day. he proper dating used in creating or reading a scratch file is covered in the ections of this manual that pertain to magnetic media peripherals.

scratch file may also be made obsolete after closing through the use of the LOSEO instruction. See NEAT/3 REFERENCE MANUAL, FILES, tab 1, "CLOSE Instruction."

* * * *

The following table is a partial list of I/O instructions that are applicable to various types of files discussed in this section.

I/O INSTRUCTIONS AND APPLICABLE PERIPHERAL FILES

I/O INSTRUCTIONS	CARDS	PAPER TAPE	PRINTER	DISC	MAGTAPE	CRAM
OPEN	X	X	X	X	X	X
OPENC				X		X
OPENS				X	X	X
OPENT				X		X
ROPENS				X	X	X
ROPEND				X	X	X
ROPENR				X		X
ROPENP				X	X	X
CLOSE	X	X	X	X	X	X
CLOSEO				X	X	X
CLOSES				X	X	X
CLOSET				X		X
GET	X	X		X	X	X
LGET					X	
RGET				X		X
SGET				X		X
SGETC				X		X
SGETL				X		X
PUT	X	X	X	X	X	X
LPUT					X	
WRITSP				X		X
WRITBI				X		X
INSERT				X		X
RFILE				X		X
DELETE				X		X
MARK				X		X
RESET				X		X
BLKCHK				X	X	X
BLKOUT				X	X	X
DEFALT				X		X
READ				X	X	X
WRITE				X	X	X

MAGNETIC DISC FILES

INTRODUCTION

Each NCR Century 100 System incorporates one integrated dual disc unit; however, two integrated units are permitted. Each spindle of this two-spindle unit is treated as a separate device by both the selection logic of the I/O software and by the machine hardware. Since each spindle may contain one removable disc pack with a storage capacity of approximately 4.2 million characters, the total storage capacity of the unit is over 8 million characters.

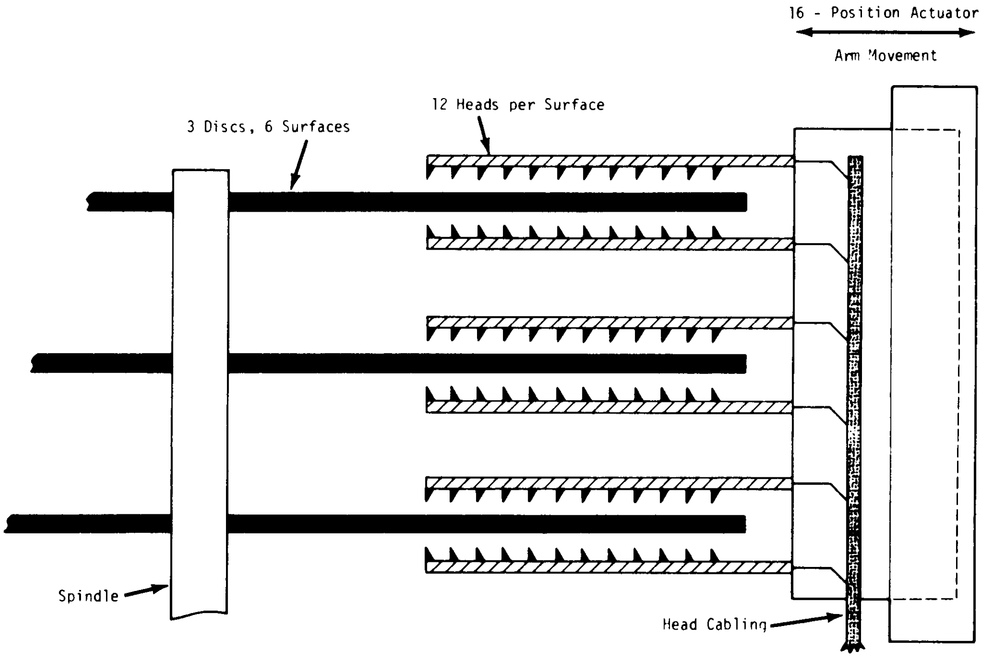
The disc unit is supported by highly sophisticated software which makes it flexible, yet easy to use. The disc software automatically handles such things as the location of source files, the allocation of room for destination files, and the end-of-section alternation.

Additional common trunk disc units and disc controllers are also available on an optional basis. These units permit larger block sizes than the integrated dual disc.

DISC PACK

A single disc pack is composed of three discs that are coated on both sides to provide six recording surfaces. Since each of the six surfaces is serviced by 12 read/write heads, each pack is serviced by a total of 72 read/write heads. Of these 72 heads, 64 are available to the user and 8 are used by the hardware for selection of the desired read/write location. All 72 heads which are connected to a single arm, are moved as a unit to any one of 16 positions by an actuator.

READ / WRITE HEADS AND 16 – POSITION ACTUATOR

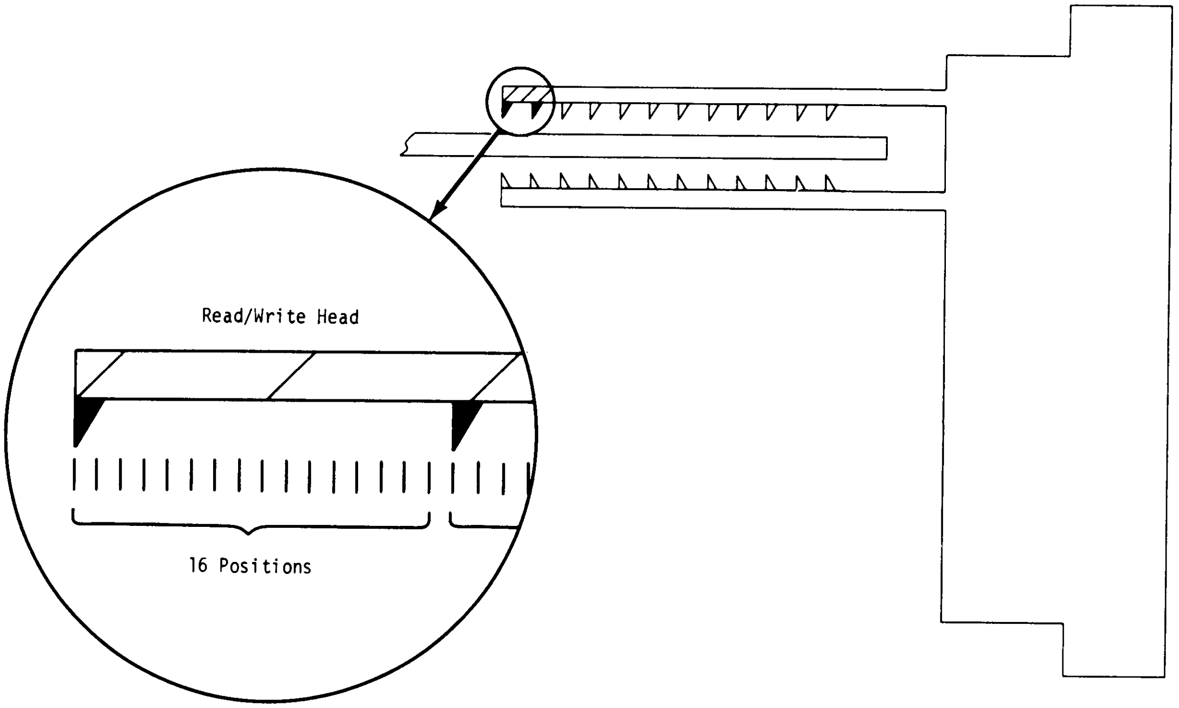


- Each pack is composed of 3 discs which provide 6 recording surfaces.
- Each surface is serviced by 12 read/write heads; therefore, each pack is serviced by 72 read/write heads.
- Each pack has 64 heads available to the user and 8 heads reserved for the hardware.
- All heads are shifted simultaneously into one of 16 positions by the 16-position actuator.

16–Position Actuator

The 16-position actuator moves the heads across the disc pack to access data from each of 16 positions. The distance between the read/write heads is such that throughout the movement of the arm one head never enters an area previously occupied by another head; that is, there are 16 positions between each head.

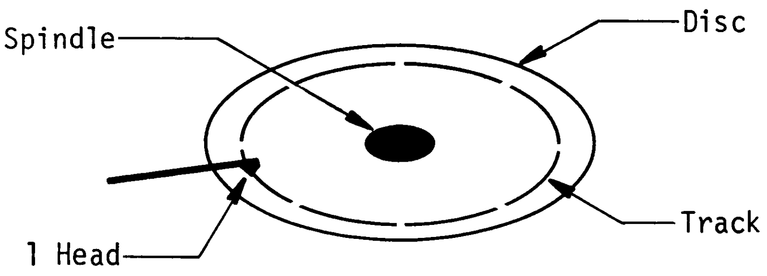
16 POSITIONS BETWEEN HEADS



Tracks

The area covered by one read/write head during one revolution of the disc is called a track. Since there are 64 heads available, there are 64 tracks available for reading or recording data in each of the 16 positions. Therefore, over the entire 16 positions, a total of 1024 tracks are available for data.

TRACKS

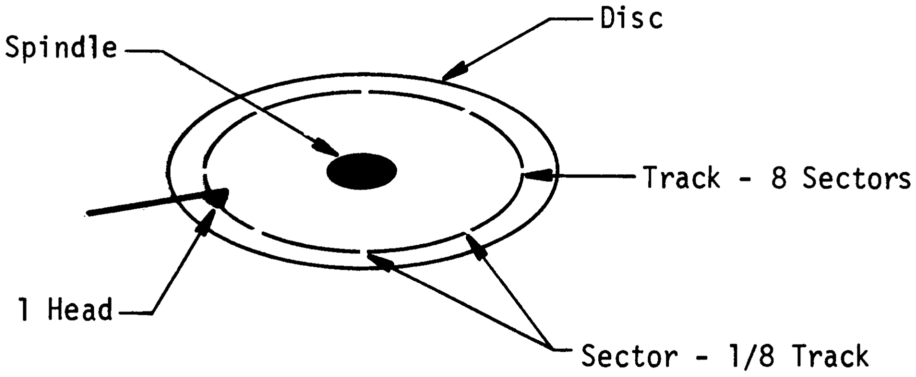


- The area of the disc covered by one read/write head during a revolution is called a track.
- A total of 64 tracks is available in any one of the 16-actuator positions.
- A total of 1024 tracks is available over the entire 16 positions.

Sectors

Each track of the disc is divided into eight addressable units called sectors. Since there are 64 tracks in each of the 16 positions, 512 sectors are available in each position of the actuator. Therefore, over the entire 16 positions, 8192 sectors are available for data storage. Each sector may contain up to 512 characters of data, and the entire pack may contain 4,194,304 data characters.

SECTORS



- Each track is divided into 8 sectors.
- Each of 16 positions has 512 sectors.
- Each pack has 8192 sectors.
- Each sector may contain 512 characters.
- Each pack may contain 4,194,304 characters.

Timing

Data is transferred between the processor and most dual disc units at a rate of 108,000 characters per second; however, one optional common trunk unit has a transfer rate of 180,000 characters per second. The average latency time, which is dependent upon the revolution time of the disc, is 21 milliseconds for disc units that have a transfer rate of 108,000 characters per second; the latency time is 12.5 milliseconds for disc units that have a transfer rate of 180,000 characters per second.

The average access time, which is dependent upon the actuator movement, is 55 milliseconds for all disc units. The minimum access time is 30 milliseconds and the maximum access time is 70 milliseconds.

FILE DESCRIPTION

Disc Pack Formats

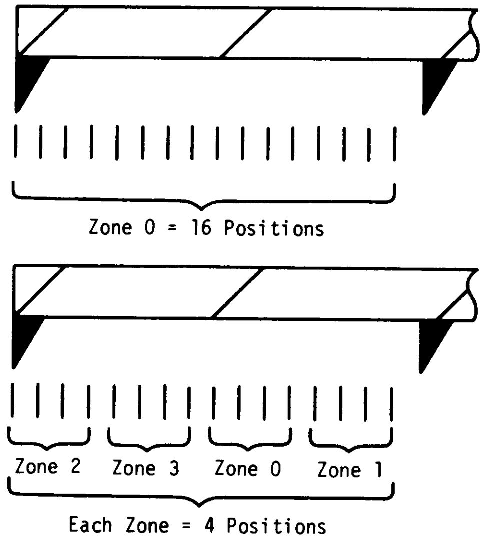
Prior to first use, the disc initializer utility routine (DINIT) must be used to set up (initialize) disc packs in one of two formats: Format 1 or Format 9. Any pack initialized as Format 1 contains the operating system software; any pack initialized as Format 9 does not contain the system software, thereby allowing additional room for files. Since Format 9 packs do not contain system software, they may not be placed on either of the system discs. For this reason, NCR Century 100 Systems containing only one integrated dual disc unit may not use Format 9 packs.

Zones

Disc packs may be treated as an entity (one-zone packs) or may be divided into four parts (four-zone packs). On one-zone packs all 16 positions are treated as a single zone (zone 0), while on a four-zone pack a group of four consecutive positions is treated as a zone (zone 0, 1, 2, and 3).

The programmer uses the file specification sheets to map his files and thereby indicate that the pack is to contain one zone or four zones. One zone packs are most commonly used; however, four-zone packs may be used if the programmer wishes to divide his file into sections on a single disc pack.

ZONES



One-Zone Pack
On a one-zone pack all 16 positions are considered to be in zone 0.

Four Zone Pack
On a four-zone pack a group of four consecutive positions is treated as one zone.

- One-Zone Packs

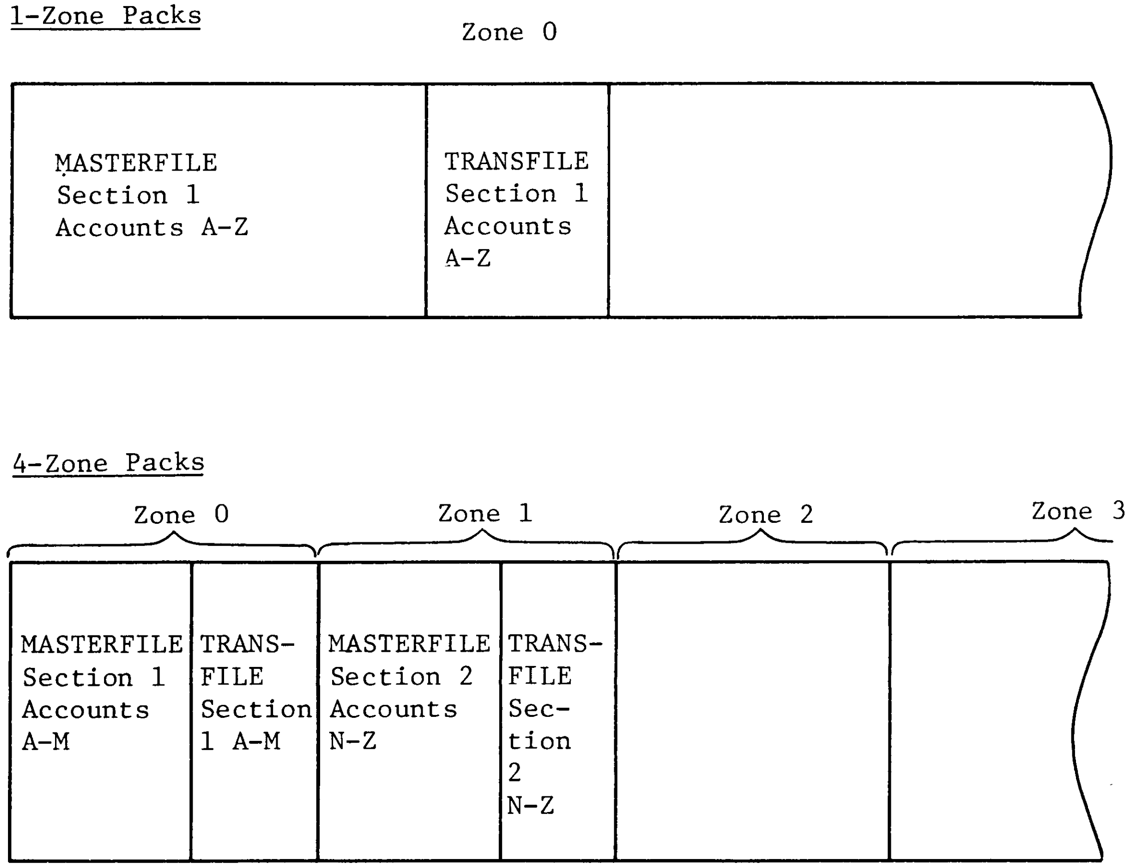
  Format 1 packs with one zone are used most often; however, Format 9 packs with one zone may be used if the system configuration permits. The programmer has approximately 7350 sectors available for file storage on a Format 1 pack with one zone and has approximately 8100 sectors available on a Format 9 pack with one zone.

- Four-Zone Packs

This type of pack is divided into four separate zones (0, 1, 2, and 3) with each zone containing four positions. The programmer has approximately 1200 sectors available in zone 0 of a Format 1 pack and approximately 1950 sectors available in zone 0 of a Format 9 pack. Zones 1, 2, and 3 of either format pack contain 2048 sectors each.

Each zone may contain a section from more than one file, as in the example below, allowing related sections of different files to be placed in a single zone. Large files may be extended over several packs if necessary.

1—ZONE AND 4—ZONE PACKS

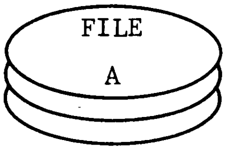

1-Zone Packs
Zone 0
MASTERFILE Section 1 Accounts A-Z
TRANSFILE Section 1 Accounts A-Z
4-Zone Packs
Zone 0
Zone 1
Zone 2
Zone 3
MASTERFILE Section 1 Accounts A-M
TRANS-FILE Section 1 A-M
MASTERFILE Section 2 Accounts N-Z
TRANS-FILE Sec-tion 2 N-Z

4-zone packs allow related portions of two or more files to be placed in the same zone.

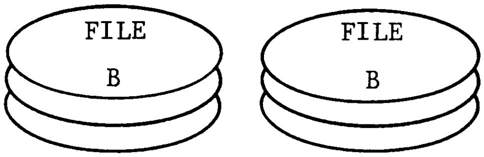
Disc/File Relationship

One or more files may be placed on a disc pack, or one file may be placed on several disc packs. When small files are involved, several files may occupy one pack or one zone. Programs associated with the files on a given pack should appear on that pack.

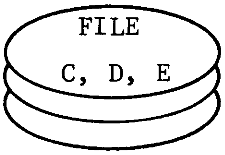
FILE/PACK RELATIONSHIP



Single-Pack File: One file is contained on one pack. The file may be either a single section or multiple sections. The disc pack may be a 1-zone or a 4-zone pack.



Multi-Pack File: One file is contained on two or more packs. This is a multi-section file with either a 1-zone or 4-zone pack.



Multi-File Pack: More than one file is contained on a pack. Each file may be either a single section or multiple sections. The disc pack may be either a 1-zone or a 4-zone pack.

Labels/Disc Directory

All labels for all files on a disc pack are confined to an area of the pack called the Disc Directory. These labels contain information the software needs for the protection, identification, and location of files. Some of the directory entries include:

- File name - Contains the file name as indicated on the file specification sheets.
- Creation date - Contains the virtual date on which the file was created. (Virtual dates are covered in the description of file specification sheets under this tab.)
- Expiration date - Contains the date the file expires.
- Generation number - Contains a number which is updated each time a new version is created. (Generation numbers are sometimes used in place of expiration dates.)
- File start sector - Contains the sector number at which the file begins.
- File end sector - Contains number of the sector following the last sector in the file.
- File type - Contains a character to indicate if this is a chained file or a standard file.

When a destination file is opened, the disc directory is scanned to locate free sectors on a pack. The number of sectors requested on the file specification sheets is allocated if available. Sectors are available if they are not occupied by a file or if they are occupied by a file which has expired. When space is allocated to a file, a new entry is built and inserted into its proper place in the directory -- according to the placement of the file.

When a source file is opened, the Disc Directory is scanned to locate the requested file as named on the file specification sheets. Once the proper entry has been located, the starting sector address of the file is extracted, and processing can begin. This starting sector is used by the software to compute the actual sector address from the relative address specified by the programmer.

The programmer need not concern himself with the actual location of the file on the disc. When the programmer must specify a particular sector, as with the RGET instruction, he has only to specify its relative position in a file section. The first relative sector address in any section of the file is zero.

- Creating the Disc Directory/Initializing the Disc Pack

  Since the Disc Directory is initially created by the disc initializer utility routine, all packs must be initialized before use in the system. Once the directory is created, the software automatically maintains it.

  When a disc pack copy is done, all areas between active files may be eliminated in each zone for a four-zone pack. All areas between files in an entire pack may be eliminated for a one-zone pack. Files, as well as entries in the Disc Directory, may be consolidated on the new pack.

Data Blocks

Data blocks consist of one or more records. Records within these blocks or the blocks themselves may be fixed or variable in length.

Minimum and maximum block lengths are dependent upon several factors. One of these factors is the type of disc unit used; another factor is the type of file organization used.

- Disc Unit Limitations on the NCR Century 100

  The integrated disc unit on the NCR Century 100 allows a minimum block length of 1 character and a maximum block length of 512 characters. Optional common trunk disc units on the NCR Century 100 allow a minimum block length of 1 character; however, blocks may be either 1, 2, or 4 sectors long, that is 512, 1024, or 2048 characters long.

- Disc Unit Limitations on the NCR Century 200

  All disc units on the NCR Century 200 allow a minimum block length of 1 character; however, blocks may be a maximum of either 1, 2, 4, or 8 sectors long, that is 512, 1024, 2048, or 4096 characters long.

NOTE

> The programmer should use caution is assigning his block length when multiple sectors are involved. The software examines the specified block length and assigns the next highest number of sectors permitted. For example, specifying a block length of 1050 characters results in the software assigning four sectors to each block. Almost two whole sectors would be blank for each block output.

- File Organization Limitations

If standard file organization is used with a disc unit, the minimum block lengths and maximum block lengths remain the same. If chained file organization is used, the minimum block length is 10 characters; the maximum block length remains the same. The software uses these 10 characters for chaining and flags. Chained files are discussed in detail in this publication under File Organization.

Records in Blocks

Records within magnetic disc blocks may be fixed or variable in length. Record sizes for the disc units on the NCR Century 100 range from a minimum of 1 character to a maximum of 512 characters on the integrated disc unit and either 512, 1024, or 2048 characters on the optional common trunk disc units. Record sizes for the disc units on the NCR Century 200 range from a minimum of 1 character to a maximum of either 512, 1024, 2048, or 4096 characters.

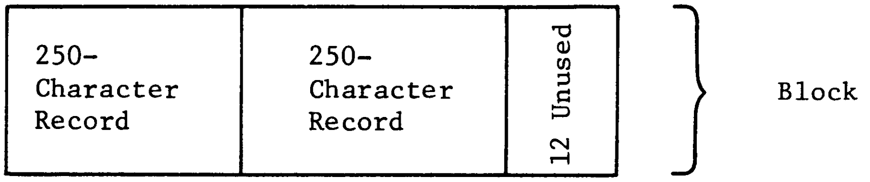
NOTE

> For clarity, all examples pertain only to the integrated disc unit on the NCR Century 100.

- Fixed-Length Records

If the records are fixed-length, all blocks are normally full blocks. The programmer assigns a block length that is a multiple of his record length. For example, if each record is 100 characters long, the block length could be 100, 200, or any other multiple of 100 up to 500 characters. To best use disc space, the programmer should specify a block length that is as close to 512 characters as possible.

FIXED—LENGTH RECORDS IN BLOCKS



Maximum block length = 512 characters.
Each record = 250 characters.
Each block may contain 2 records.
Each block leaves 12 character positions unused.

** ● Variable-Length Records

When variable-length records are used, a variable length indicator (VLI) must be defined on the data layout sheet as the first field in the record. This indicator contains a number that specifies the number of characters in a record and is used by the software to present complete records to the program.

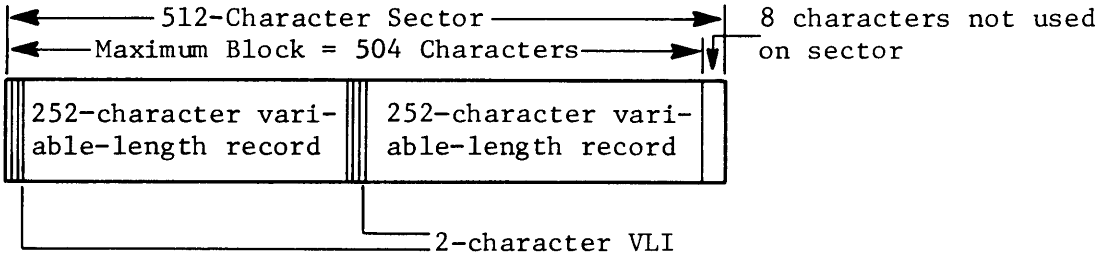
The VLI field must be two characters long and must contain a binary number which indicates the number of characters in the record (including the VLI itself). The programmer must place the correct number in the VLI. If translation is requested with the GET and PUT instructions, the VLI is not translated.

VARIABLE LENGTH RECORDS IN BLOCKS

PAGE	LINE		REFERENCE	CODE	LOCATION	LENGTH	DP	TYPE	VALUE OR PICTURE
1 2 3	4 5 6	7	8 9 10 11 12 13 14 15 16 17	18	19 20 21 22 23	24 25 26 27	28 29	30	31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
		D	MAGZREC	R		252			
		D	VLI	F	0	2		B	
		D	ACCTNO	F	2	10		U	

Record Length: The programmer specifies the maximum length of the variable record as indicated on the file specification sheet. This length must include the 2-character field containing the VLI.

VLI Field Length: The programmer must reserve a 2-character binary field beginning at relative position 0.


512-Character Sector
Maximum Block = 504 Characters
8 characters not used on sector
252-character variable-length record
252-character variable-length record
2-character VLI

FILE ORGANIZATION

Standard file organization or chained file organization may be used with the disc. Standard file organization is used for father-son processing, selective serial processing, and with limitations for random processing. Chained file organization, which provides both for overflow and for insertion and deletion of records, is used for random processing and selective serial processing.

Standard File Organization

When standard file organization is used, blocks of records are generally presented to the program in the sequence they were recorded; however, the records within the block may or may not be in numeric or alphabetic order. For example the input records to a sort run are unsorted, but the output is sorted in either numeric or alphabetic order and becomes input to another run.

Standard file organization does not provide for a specified overflow area. To handle file expansion, the programmer must leave sufficient room in a section; after filling that section, he must allocate a new one.

Random accesses are permitted to a limited degree; however, records should not be deleted or inserted unless the entire file is copied.

* - Father-Son Processing and Selective Serial Processing

  For a detailed explanation of father-son processing or selective serial processing, see the NEAT/3 REFERENCE MANUAL, FILES, "NCR Century File Concepts."

- Random Processing

  Limited random access is available when standard files are used. The programmer uses the RGET instruction to randomly access a specific sector and to make the first record in the block on that sector available to the program. To step sequentially through the remaining records in that block, he may use the GET or SGET instruction. After all records in the block have been presented, another execution of the GET or SGET instruction reads in the next block of records in sequence. This limited random access provides no simultaneity in reading from a standard file.

  Generally the random processing of standard files is confined to randomly updating a source-destination file. In this instance, the records in the master file are generally in numeric or alphabetic order, but the transaction records may be posted to the master in sorted or unsorted order. The program examines the transaction record and, from the information contained in it, determines the sector on which the corresponding master record is located. This location may be determined by a table lookup, by computation on the account number, etc. Once determined, the relative sector address is placed in a special area in memory to be accessed by the RGET instruction. The address may point to the exact block in which that record is stored or to some block closely preceding the desired block.

NOTE

All instructions names are discussed in detail in the NEAT/3 REFERENCE MANUAL, INSTRUCTIONS, tab 1.

COMPUTING SECTOR NUMBER FOR RANDOM PROCESSING— STANDARD FILE ORGANIZATION

Transaction Record

Acct No. 0030	Name	Data

A transaction record is read.

Acct No. 0030

The account number is extracted.

ACCTABLE

Key	Rel. Sec.
0010	0000
0020	0005
0030	0010
0040	0015
0050	0020
0060	0025
0070	0030

The table is constructed so that a reference to any account number between 21 and 30 indicates relative sector 10. The program must search through each sector beginning at 10 until the desired master record is encountered. (For further information on tables, see, INSTRUCTIONS, tab 4, "Table Concepts."

If the block addressed is the actual desired block, the RGET instruction presents the first record in that block to the program. If, by comparison, it is found that this is not the desired record, the GET or SGET instruction presents the next record in the block for comparison. Succeeding executions of the GET instruction will sequentially present the other records in that block until the desired record is encountered.

If the RGET instruction presents a block closely preceding the one desired, successive executions of the GET or SGET instruction step through that block and sequentially access new blocks until the desired record is found.

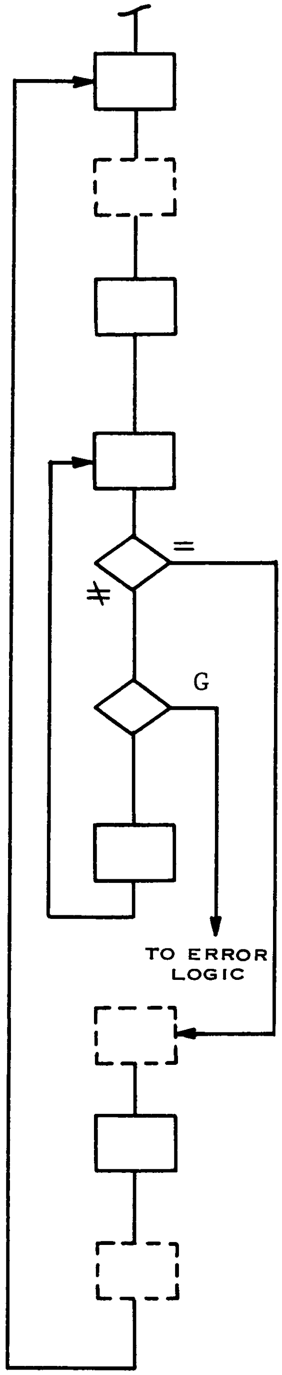
Since reading from the disc does not destroy the information on the disc, only the updated blocks need be written back. The WRITSP instruction is used to write back only those blocks which have been updated. The programmer places WRITSP at the end of that portion of the program responsible for updating. The WRITSP instruction is then encountered only when a block has been updated. Under these circumstances, WRITSP guarantees that the software will automatically write updated blocks back into the file.

The updated record is placed exactly over the old record in the block. Because of this, the record length and block length must remain the same. No new records should be inserted or deleted during the update run; instead, records should be inserted and deleted while the entire file is copied as in a father-son run. The old master file is destroyed since it is written over.

Standard file organization is used with random processing if the file incorporates fixed-length records or variable-length records that do not change in size during update. If the variable-length records are to change in size or records are to be inserted or deleted during the update run, chained file organization should be used.

Since RGET uses relative addressing to randomly access records within a section, all sectors within one section of a randomly-accessed file must be physically adjacent to each other. To keep the relative addressing consistent, the file must be in one piece within each section. Therefore, the question on the file specification sheets, "Are disjoint pieces acceptable if the number of sectors cannot be found in one piece," must be answered "NO."

USING WRITSP



GET Read in a transaction record.

Calculate the location of the master record and place the address in an area for accessing by RGET.

RGET Read in the block containing the master record or a block closely preceding the desired one. Present the first record in the block.

COMP Compare the presented master record to the transaction record.

BRE If they compare equal, branch to that leg of the program responsible for updating. If not, go to next instruction.

BRG If the master is greater than the transaction, go to error coding. Otherwise go to the next instruction.

GET Present the next record in the block to the program; or if no records are left in this block, read the next block.

Update the master record.

WRITSP Set up to write the block containing the updated master record back into the file. (Writing takes place automatically.)

Set up for next transaction.

Chained File Organization

Chained file organization, used for selective serial processing or random processing, provides both for overflow and for insertion and deletion of records.

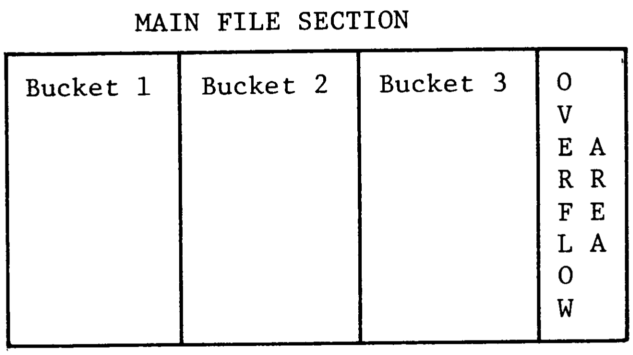
A chained file consists of blocks of records automatically connected in sequence by the software. The first 10 characters in each block are used by the software to link a block to the next sequential block in the file. Therefore, the maximum number of program-usable characters in a block for a chained file is reduced by 10. For example, the programmer has 502 characters available in a 512 character block.

Chained file organization is generally confined to source-destination files but may be associated with source or destination files, e.g. a chained source-destination file may be created as a chained destination file or used as a source file in a different run. These source-destination files may be processed sequentially or randomly.

The chained file is generally divided into two areas: a main file and an overflow area. The main file area is further divided into buckets. The size of the bucket is determined by the programmer. The entire file may be one bucket, or the file may contain many buckets.

A bucket is a group of sectors from which overflow can occur. It is a subdivision of a file section and a logical break point in the file that aids the programmer in searching for a record. The bucket concept also saves program execution time when the INSERT instruction is used, because records overflowing from a bucket are placed in the overflow area and are chained back to the previous record in the bucket. The GET and SGET instructions are capable of following this chaining.

BUCKETS AND OVERFLOW AREA



The programmer specifies the following on the file specification sheets:

- The number of sectors in a section.
- The number of sectors in a bucket.
- The number of sectors in the overflow area.

* * * *

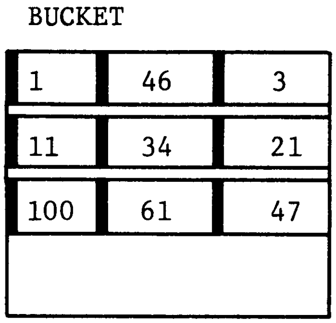
When chained files are used on the disc, the programmer is concerned with how many sectors to allocate to a bucket. In arriving at a conclusion, the programmer should consider the following questions:

- Is the file sorted or unsorted?
- How large is the file?
- How many insertions can be expected?
- How many deletions can be expected?
- What is the expected record growth?

- Unsorted Files

If the file is unsorted, a new record need not be placed in a specific location in the bucket. In this case the programmer uses the RFILE instruction to place the new record in the first available location in the bucket. This location may be in a partially full block, in a block left empty for expansion in the bucket, or in a block made empty by the previous deletion of a record. If the record will not fit in the bucket, it is placed in the overflow area. Large buckets may be used with unsorted files since pushdown does not occur. However, the programmer must use judgment because too large a bucket may result in excessive search time.

UNSORTED RECORDS IN BUCKET



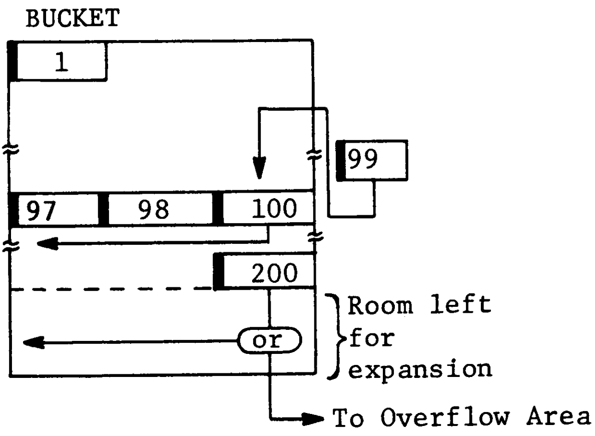
Records in the bucket are unsorted.

Buckets are generally in numeric order. For example, the first bucket contains records 000 - 100, the second bucket contains records 101 - 200, etc.

- Sorted Files

If the file is sorted (in numeric or alphabetic order), all insertions must be placed in their proper positions, e.g. account number 99 must be placed between 98 and 100. The INSERT instruction is used to place the record in its proper location. Assume that 99 was not assigned when the file was initially constructed and that no room exists for it. To accomodate 99, INSERT automatically pushes down all other accounts in the bucket following 98 and changes all the chaining labels of each block affected. (To avoid lengthy pushdown, small buckets should be used with sorted files that have relatively high insert activity.)

PUSHDOWN DUE TO INSERT



When 99 is inserted, all records starting with 100 are moved down by the INSERT instruction.

If room is left for expansion in the bucket, all the following records simply move down in the bucket.

If no room remains in the bucket, the following records are moved down, and the last record is moved to the overflow area.

In either of the above two cases (sorted or unsorted), the programmer may keep track of what is stored in each bucket by means of a table. During processing, the program looks in the table for the bucket location of the desired record. The RGET instruction is used to get the first record in that bucket, and the GET or SGET instruction to step further through the bucket. If necessary, GET or SGET follows the chaining into the overflow area. However, if the record is not found in the bucket or its associated overflow area, the SGET instruction transfers control to the user's routine referenced by the second operand of SGET. This routine could contain coding either to write the record into an exception file or to print it. The GET instruction does not transfer control but continues at the beginning of the next bucket.

A major advantage in using buckets is that it gives the program an indication that an error or exception exists without having to search through the entire file. A branch is taken if a record is not found by the time the end of a bucket is encountered since records are pushed down only in the bucket, not in an entire file or file section.

- Creating a Chained File with Buckets

A file with buckets may be created as a destination file or as a source-destination file.

- Creating a Chained File as a Destination File

When creating a chained file as a destination file, the programmer uses a 1-time conversion program that incorporates the PUT instruction. In filling out the file specification sheets for this 1-time program, he indicates the type of file (chained destination) and the desired bucket size.

Each time the PUT instruction fills a block, the software inserts the proper chaining characters, and the block is written into the file. If the programmer wishes to leave room for expansion in the bucket, he may do so by using the BLKOUT instruction to output null (empty) blocks.

It is the programmer's responsibility to count the number of blocks output to a bucket and thereby insure that only the desired records appear in each bucket. The BLKCHK instruction is used as an aid in this task. If desired, a file directory (table) can be built at this time to aid in the location of data in the file or bucket during another run.

When the file is closed, the software completes the chaining characters for all unused sectors in the file.

- Creating a Chained File as a Source-Destination File

When creating a chained file as a source-destination file, either a 1-time program or the normal update program may be used. In either case the programmer indicates on the file specification sheets the type of file (chained source-destination) and the desired bucket size.

When the program is run, the operator is informed through the console lights that the file does not exist. He indicates in turn that a new file is to be created. The new file is opened and the chaining characters are placed on each sector of the file, indicating null (empty) blocks. To place data in the file, the programmer uses the RGET, various SGET, and INSERT or REFILE instructions, depending on the file format.

If desired, a file directory (table) can be built using the MARK and RESET instructions at the same time the file is constructed to aid in the location of data in the file or bucket during another run.

- Selective Serial Processing

For a detailed explanation of selective serial processing, see the NEAT/3 REFERENCE MANUAL, FILES, "NCR Century File Concepts."

The programmer should use a chained file with an overflow area if records are inserted or deleted, or if the length of variable-length records changes during the update run. If records are not inserted or deleted, and the length of the record remains the same, a standard file could be used.

- Random Processing

Chained file organization is suited for the random processing of source-destination files. The bucket concept allows a logical break in the file for overflow. The SGET instruction transfers control to that portion of the program referenced by the second operand when the end of the bucket is reached. If overflow occurred in a previous run, SGET follows the chaining through those blocks in the overflow area associated with that bucket; then the branch occurs. Chained files may, however, be specified with or without an overflow area.

- Chained Files without an Overflow Area for Random Processing

Chained files without an overflow area can be specified. This is generally done when the bucket concept is desired and when the records and blocks are fixed length. The SGET instruction causes a branch when the end of a bucket is reached, as described previously under Unsorted Files. Records may be inserted or changed in length during the update run, if room is left in the bucket; however, this is not recommended when an overflow area is not present.

The master file is generally in sorted order (numeric or alphabetic), but the transaction records may be in sorted or unsorted order. The program examines the transaction record and, from the information contained in it, finds the address of the sector on which that master record is located. This address, which may point to the beginning of a bucket or some location within a bucket, can be determined by a table lookup, by computation on the account number, etc. The relative sector address is determined and placed in memory to be accessed by the RGET instruction.

COMPUTING SECTOR NUMBER FOR SELECTIVE SERIAL PROCESSING—CHAINED FILE ORGANIZATION

Transaction Record

Acct No. 0030	Name	Data

A transaction record is read.

Acct No. 0030

The account number is extracted.

ACCTABLE

Key	Rel. Sec.
0010	0000
0020	0005
0030	0010
0040	0015
0000	0020

The table is constructed so that a reference to any account number between 21 and 30 indicates relative sector 10. Relative sector 10 is the beginning of a bucket. RGET presents the first record in sector 10. SGET presents the remaining records in the sector and then accesses the next sector until the desired record is found. If the desired record is not found in the bucket, the end of bucket is reached. Control is then transferred to the routine referenced in the second operand of SGET.

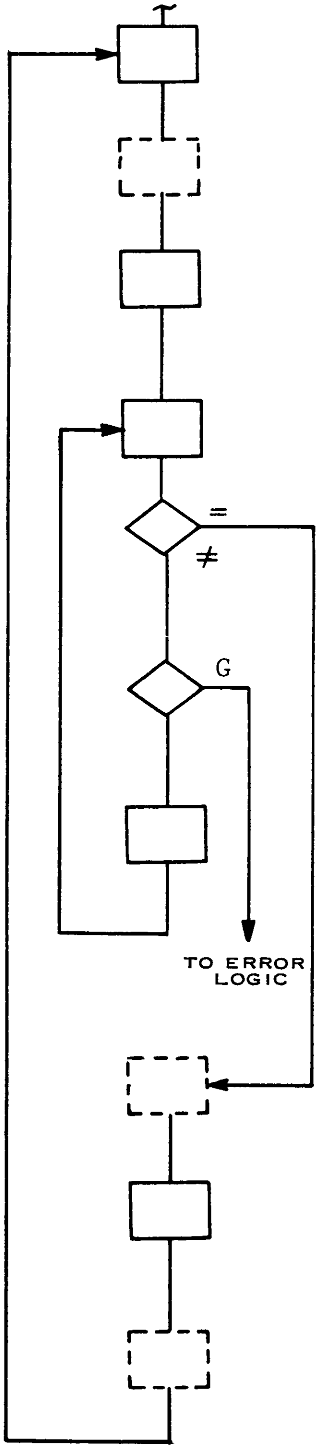
The RGET instruction presents the first record in the first block of the bucket. If, by comparison, it is found that this is not the desired record, the GET or SGET instruction is used to present the next record to the program. Succeeding executions of the SGET instruction step through that block and sequentially access new blocks until the desired record is found. If the end of the bucket is reached before the desired record is found, a branch is made to the user's routine referenced by the second operand of SGET.

Since reading from the disc does not destroy the information on the disc, only the updated blocks need be written back. The WRITSP instruction is used to write back only those blocks which have been updated. The programmer places WRITSP at the end of that portion of the program responsible for updating. Since the WRITSP instruction is then encountered only when a block has been updated, it guarantees that the updated block is automatically written back.

The updated record is placed over the old record in the block. Because of this, the record length and block length must remain the same. Since there is no overflow area, records should not be inserted or deleted during the update run. The entire file should be copied to insert or delete records as in father-son processing. The old master file is written over by updated records and becomes the new master file.

Since RGET uses relative addressing to randomly access records within a section, all sectors within one section of a randomly accessed file must be physically adjacent to each other. To keep the relative addressing consistent, the file must be in one piece within each section. Therefore, the question on the file specification sheets, "Are disjoint pieces acceptable if the number of sectors cannot be found in one piece," must be answered "NO."

USING WRITSP



GET Read in a transaction record.

Calculate the location of the master record and place the address in an area for accessing by RGET.

RGET Read in the block containing the master record or a block closely preceding the desired one. Present the first record in the block.

COMP Compare the presented master record to the transaction record.

BRE If they compare equal, branch to that leg of the program responsible for up-updating. If not, go to next instruction.

BRG If the master is greater than the transaction, go to error coding. Otherwise go to the next instruction.

SGET Present the next record in the block to the program. If no records are left in this block, read next block. If the end of a bucket is encountered, branch to the routine referenced in the second operand of the SGET instruction.

Update the master record.

WRITSP Set up to write the block containing the updated master record back into the file. (Writing takes place automatically.)

Set up for next transaction.

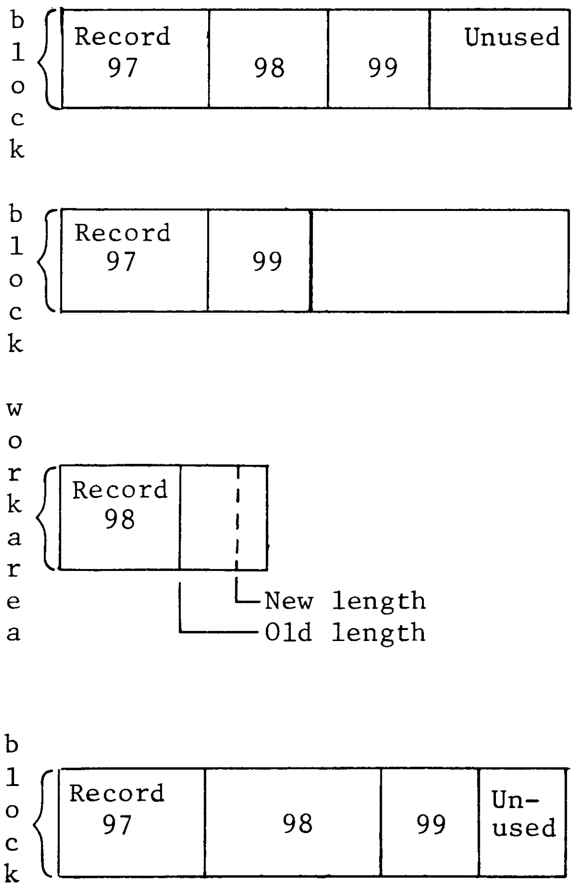
- Chained Files with an Overflow Area for Random Processing

The programmer may designate an overflow area to allow the size of a chained file to expand or contract. The size of this overflow area depends on the number of insertions expected, the size of the buckets, expected record growth, and the amount of padding left in each bucket. Each section of the file has its own overflow area.

This type of file is handled similarly to a chained file without an overflow area. The principal difference is that the presence of an overflow area permits the user to efficiently insert and delete records and to change the length of variable-length records.

If the length of a variable-length record is to be changed, it should be deleted from the file, specifying a move to a work area. The length of the record is then changed in the work area, and the record is placed back in the file by using the INSERT or RFILE instruction. (INSERT is used when the file is in sorted order within the bucket; RFILE when the file is in unsorted order within the bucket.)

ALTERING RECORD LENGTH — CHAINED FILE ORGANIZATION



Assume records are sorted within buckets.

RGET the proper bucket, and SGET the proper record.

DELETE the record to be changed in length, specifying a move to a work area.

Lengthen or shorten the record.

INSERT the record back in the file.

If records are unsorted within buckets, RFILE may be used to place the record in the first available location in the bucket.

SITUATING FILES ON THE DISC PACK

One-Zone Packs

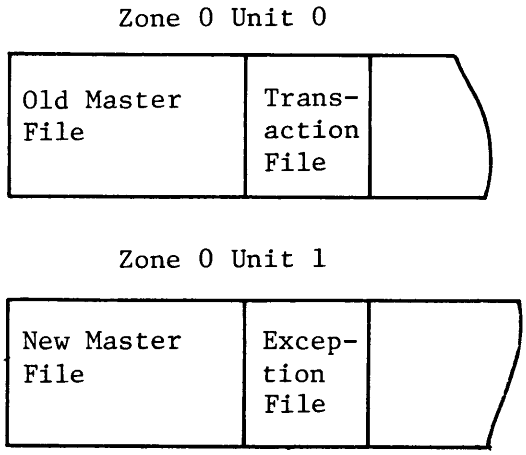
When a one-zone pack is used, files are automatically placed on the pack where room permits. The following are examples of situating files on packs for processing efficiency.

- Files for Father-Son Processing

  The old master file and the transaction file should be placed on the same pack on one unit, and the new master file should be placed on the pack on another unit. Multiple packs may be used for large files.

  In assigning room for the transaction file remember that the transaction record is generally much smaller than the master record. Consequently, many transaction records fit in a smaller area than the same number of master records.

FILES FOR FATHER-SON PROCESSING – SINGLE ZONE



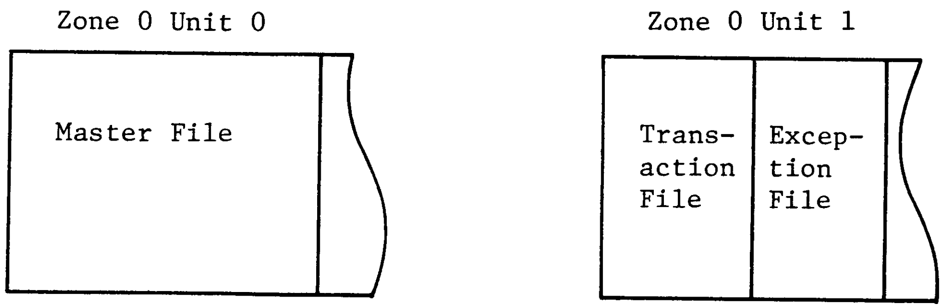
Place the old master file and transaction file on the same unit.

Place the new master file on the other unit.

** ● Files for Selective Serial Processing - Chained Files for Random Processing

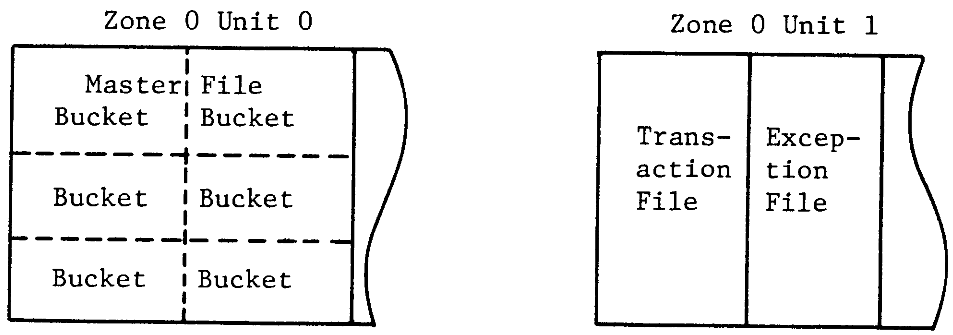
The transaction file should be placed on the pack on one spindle and the master file should be placed on the pack on another spindle.

FILES FOR SELECTIVE SERIAL PROCESSING — STANDARD FILE ORGANIZATION



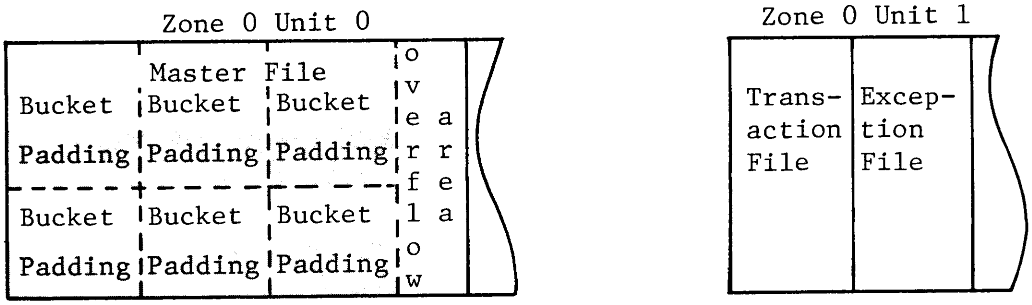
- All old master records should be placed on one unit.
    - The file may not contract or expand during the update run.
- All transaction records should be placed on the other unit.

FILES FOR SELECTIVE SERIAL OR RANDOM PROCESSING, NO OVERFLOW AREA — CHAINED FILE ORGANIZATION



- All old master records should be placed on one unit.
    - The file is divided into buckets.
    - The file should not contract or expand during the update run.
- All transaction records should be placed on the other unit.

FILES FOR SELECTIVE SERIAL OR RANDOM PROCESSING OVERFLOW AREA – CHAINED FILE ORGANIZATION



- All old master records should be placed on one unit.
  - The file is divided into buckets.
  - Each bucket has padding.
  - Each section has an overflow area.
  - Records may be inserted or deleted or changed in length since an overflow area is specified.
- All transaction records should be placed on the other unit.

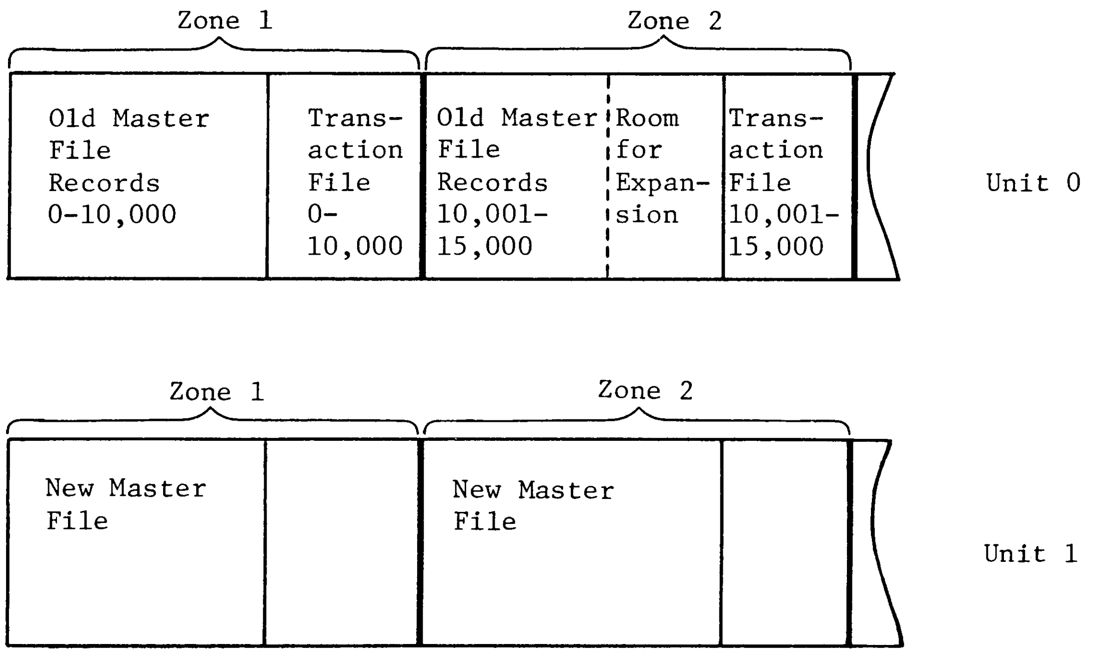
Four-Zone Packs

When a four-zone pack is used, files may be divided into sections and placed in a desired zone. The following are examples of situating files on packs for processing efficiency.

- Files for Father-Son Processing

The old master file and the transaction file should be placed in the same zone on the same pack. If the old master file and transaction file will not fit in one zone, more zones may be used.

FILES FOR FATHER—SON PROCESSING — MULTIPLE ZONES


Zone 1
Zone 2
Old Master File Records 0-10,000
Trans-action File 0-10,000
Old Master File Records 10,001-15,000
Room for Expan-sion
Trans-action File 10,001-15,000
Unit 0
Zone 1
Zone 2
New Master File
New Master File
Unit 1

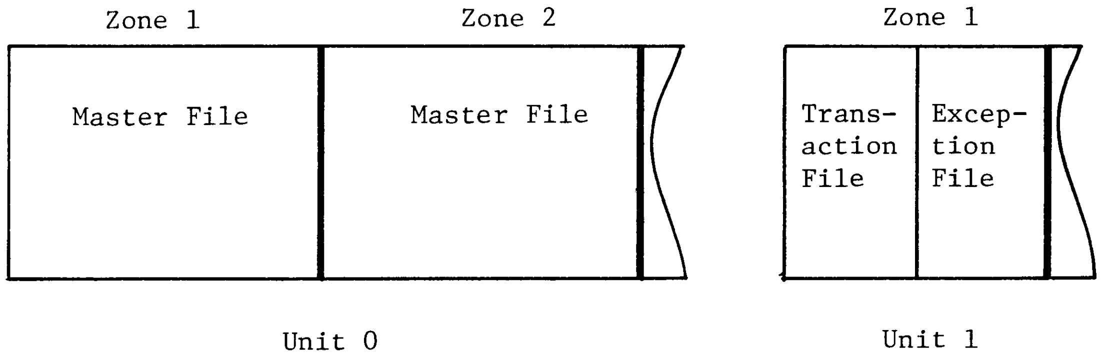
- Place related transaction records and old master records in the same zone.
- Place the new master file on the other unit.
- Leave room for expansion if new records are added.

** - Files for Selective Serial Processing - Chained Files for Random Processing

Transaction records may be on the same pack in the same zone as the old master records; or the transaction records may be on one unit and the old master records on the other unit. If both types of records are on the same pack, place related transaction and old master records in the same zone.

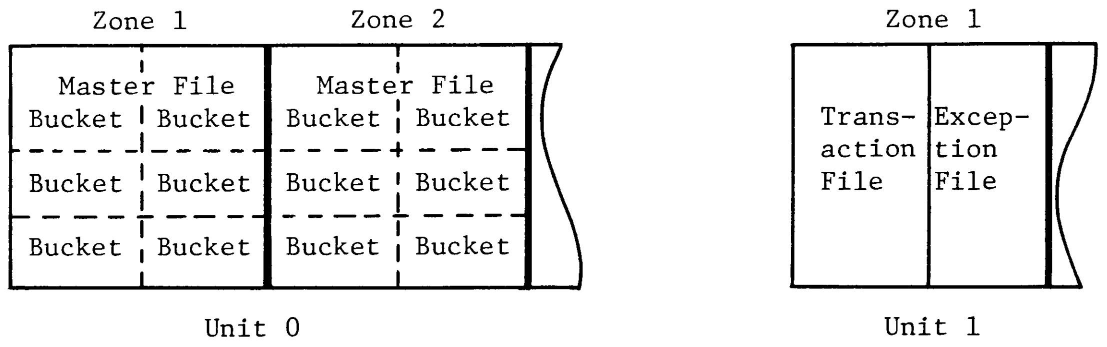
The run is more easily handled if the master file and transaction file are placed on separate disc units. With the old master records on one unit and transaction records on the other unit, no distribution run is necessary to place related transaction and master records in the same zone.

FILES FOR SELECTIVE SERIAL PROCESSING – STANDARD FILE ORGANIZATION



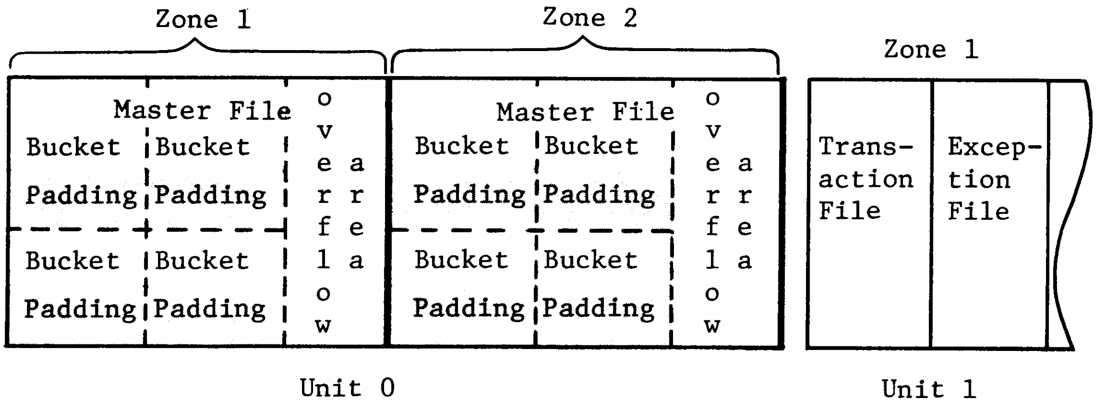
- All old master records may be placed on one unit in one or more zones.
  - The file may not contract or expand during the update run.
- All transaction records may be placed on the other unit in one or more zones.
  - No distribution run is necessary.

FILES FOR SELECTIVE SERIAL OR RANDOM PROCESSING, NO OVERFLOW AREA – CHAINED FILE ORGANIZATION



- All old master records may be placed on one unit in one or more zones.
  - The file is divided into buckets.
  - The file may not contract or expand during the update run.
- All transaction records may be placed on the other unit in one or more zones.
  - No distribution run is necessary.

FILES FOR SELECTIVE SERIAL OR RANDOM PROCESSING, OVERFLOW AREA – CHAINED FILE ORGANIZATION



- All old master records may be placed on one unit in one or more zones.
  - The file is divided into buckets.
  - Each bucket has padding.
  - Each section has an overflow area, thereby saving push down time.
  - Records may be inserted or deleted or changed in length since an overflow area is specified.
- All transaction records may be placed on the other unit in one or more zones.
  - No distribution run is necessary.

** OPENING FILES ON DISC

Normally, the programmer allows his disc files to be opened automatically; however, he may use an OPEN instruction. The OPEN instructions applicable to disc files, OPEN and OPENS, are covered in the NEAT/3 REFERENCE MANUAL, INSTRUCTIONS, tab 1, "Open Instructions."

- Opening Source Files

  When a source file on disc is opened, the software checks the Disc Directory to insure that the named file exists. Once the file is found, the data in the directory entry is checked to insure that the desired version is mounted.

- Opening Destination Files

  When a destination file on disc is opened, the disc pack is checked to insure that the desired space is available for the file. Once this is confirmed, the software makes entries in the Disc Directory. These entries contain information for future identification of the file, such as file name, file type, dates for protection, etc.

If the destination file is a chained file with buckets, certain software is included to allow the PUT instruction to build the chaining characters as the file is constructed.

- Opening Source-Destination Files

  When a source-destination file on disc is opened, the procedure is similar to opening a source file. However, if the file cannot be located, the operator can indicate that a new file is to be created. This allows the same program that updates a file to create a new file. If the new file that is created is a chained file, all the chaining characters for all the sectors are constructed. When this procedure is complete, processing can begin.

- Opening a Piggyback File

  When a piggyback file on disc is opened, the procedure is the same as for a source file. However, if the file cannot be located, the operator can indicate that a new file is to be created. This allows the same program that updates a file to create a new file.

AUTOMATIC FILE ALTERNATION

When a sequentially processed file reaches end-of-section, the software determines the location of the next section of the file. If the next section is on the same pack, no alternation of associated files is required. If the next section is on a different pack, any associated destination file is automatically alternated (the current section is closed and the new section on the new pack is opened) if both (or all) specify the same symbolic unit designator. It is not necessary to use the DEFALT instruction to achieve this alternation.

Randomly processed associated files and associated source or source-destination files are not automatically alternated; however, a source or source-destination file reaching end-of-section does initiate the automatic alternation of any associated destination file.

* CLOSING FILES ON DISC

Normally, the programmer allows his disc file to be closed automatically when the FINISH instruction is encountered; however, he may use a CLOSE instruction. The CLOSE instructions applicable to the disc, CLOSE and CLOSEO, are covered in the NEAT/3 REFERENCE MANUAL, INSTRUCTIONS, tab 1, "Close Instructions."

- Closing Source Files

  When a source file is closed, pertinent entries are made in the Disc Directory.

- Closing Destination Files

  When a destination file is closed, all the buffers for this file that are not yet empty are output, and the end-of-data sector number is placed in the Disc Directory. If this is a chained file with buckets, the software completes the chaining characters for the remaining unused sectors.

- Closing Source-Destination Files

  When a source-destination file is closed, any buffers for that file that are not yet empty are output, and certain entries are made in the Disc Directory.

- Closing Piggyback Files

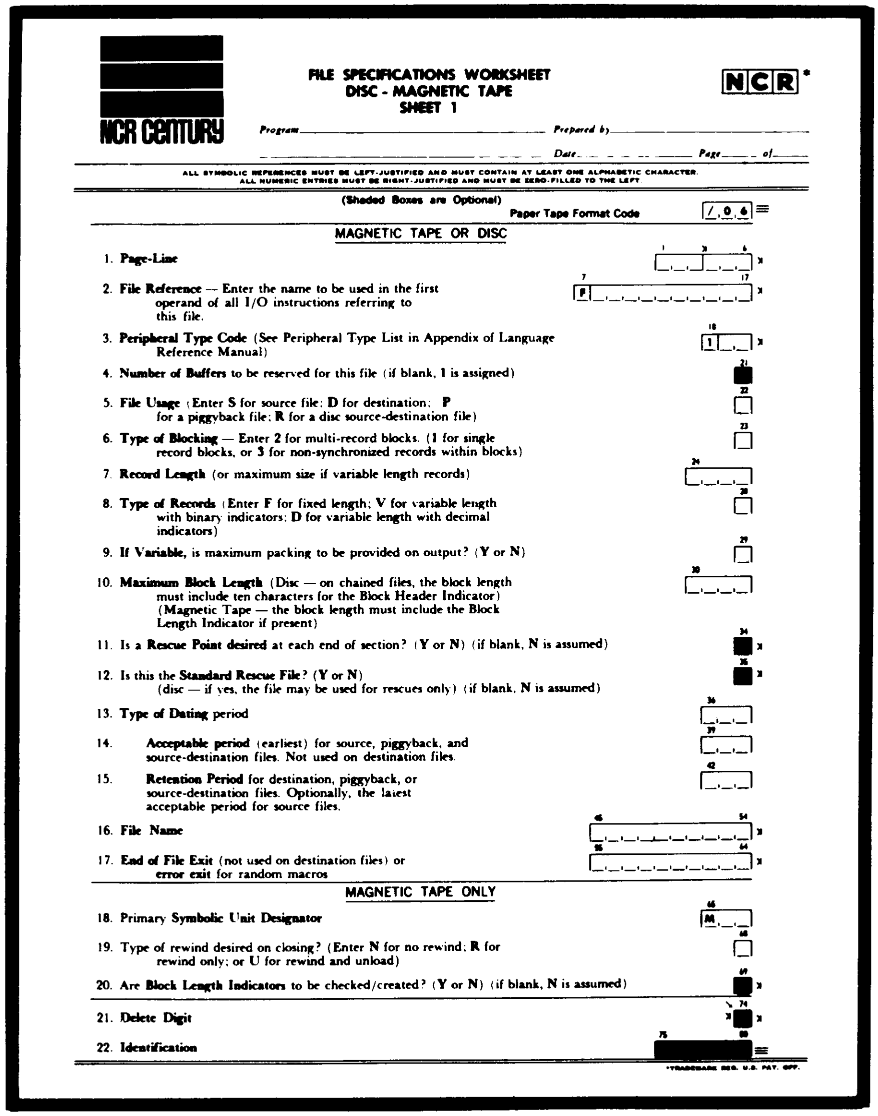
  When a piggyback file is closed, the procedure is similar to closing a destination file. All buffers not yet empty are output, and the end-of-data sector is placed in the Disc Directory.

* * * *

FILE SPECIFICATIONS WORKSHEET FOR DISC

A group of three file specifications sheets for disc files is shown in this section. Those entries applicable to magnetic tape are not completed; all other entries, where practical, are filled with a typical remark.

SHEET 1



NCR CENTURY

FILE SPECIFICATIONS WORKSHEET
DISC - MAGNETIC TAPE
SHEET 1

NCR*

Program ______ Prepared by ______

______ Date ______ Page ___ of ___

ALL SYMBOLIC REFERENCES MUST BE LEFT-JUSTIFIED AND MUST CONTAIN AT LEAST ONE ALPHABETIC CHARACTER.
ALL NUMERIC ENTRIES MUST BE RIGHT-JUSTIFIED AND MUST BE ZERO-FILLED TO THE LEFT.

(Shaded Boxes are Optional)

Paper Tape Format Code /06

MAGNETIC TAPE OR DISC

1. **Page-Line**
2. **File Reference** — Enter the name to be used in the first operand of all I/O instructions referring to this file. F
3. **Peripheral Type Code** (See Peripheral Type List in Appendix of Language Reference Manual) 1
4. **Number of Buffers** to be reserved for this file (if blank, 1 is assigned)
5. **File Usage** (Enter S for source file; D for destination; P for a piggyback file; R for a disc source-destination file)
6. **Type of Blocking** — Enter 2 for multi-record blocks. (1 for single record blocks, or 3 for non-synchronized records within blocks)
7. **Record Length** (or maximum size if variable length records)
8. **Type of Records** (Enter F for fixed length; V for variable length with binary indicators; D for variable length with decimal indicators)
9. **If Variable,** is maximum packing to be provided on output? (Y or N)
10. **Maximum Block Length** (Disc — on chained files, the block length must include ten characters for the Block Header Indicator) (Magnetic Tape — the block length must include the Block Length Indicator if present)
11. Is a **Rescue Point desired** at each end of section? (Y or N) (if blank, N is assumed)
12. Is this the **Standard Rescue File**? (Y or N) (disc — if yes, the file may be used for rescues only) (if blank, N is assumed)
13. **Type of Dating** period
14. **Acceptable period** (earliest) for source, piggyback, and source-destination files. Not used on destination files.
15. **Retention Period** for destination, piggyback, or source-destination files. Optionally, the latest acceptable period for source files.
16. **File Name**
17. **End of File Exit** (not used on destination files) or error exit for random macros

MAGNETIC TAPE ONLY

18. Primary **Symbolic Unit Designator** M
19. Type of rewind desired on closing? (Enter N for no rewind; R for rewind only; or U for rewind and unload)
20. Are **Block Length Indicators** to be checked/created? (Y or N) (if blank, N is assumed)
21. **Delete Digit**
22. **Identification**

*TRADEMARK REG. U.S. PAT. OFF.

The programmer should fill in the header, page-line number (Question 1), and the identification tag (positions 75-80) as defined in the NEAT/3 REFERENCE MANUAL, INTRODUCTION AND DATA, tab 3, "Programming Worksheets." The paper tape code must be punched if paper tape is used for input to the Compiler.

2. FILE REFERENCE - ENTER THE NAME TO BE USED IN THE FIRST OPERAND OF ALL I/O INSTRUCTIONS REFERRING TO THIS FILE. F|OMAGZFILE⍁

Enter the name used in the GET, PUT, and other I/O instructions that access this file in the program.

If this file is to be used in a Father-Son update run, the file reference name and the file name (Question 16, Sheet 1) must be different. (For further information, see the NEAT/3 REFERENCE MANUAL, FILES, "NCR Century File Concepts.")

The file reference name must begin in position 8, may be 10 characters long, and may consist of letters and numerals; however, at least 1 letter must be included.

The "F" in position 7 is preprinted and must be punched.

3. PERIPHERAL TYPE CODE (SEE PERIPHERAL TYPE LIST IN APPENDIX OF LANGUAGE REFERENCE MANUAL) 130

Enter the code that designates the type of disc being used. For a complete list of the code types, see the NEAT/3 REFERENCE MANUAL, APPENDIX, tab 1, "Peripheral Type Codes."

The "1" in position 18 is preprinted and must be punched.

4. NUMBER OF BUFFERS TO BE RESERVED FOR THIS FILE (IF BLANK, 1 IS ASSIGNED) □

Enter the number of buffers desired. Specifying two buffers provides read/compute or write/compute simultaneity in processing this file. However, since the disc is a high-speed peripheral, two buffers should be assigned to slower peripherals first, that is, the printer, punched card reader, punched tape reader, etc. Then, if memory space permits, multiple buffers should be assigned to the disc.

A maximum of two buffers is permitted when processing a source-destination file. However, more than two buffers may be assigned for files that are processed sequentially.

If this buffer is shared between two or more files, only 1 buffer may be requested. When this is done, there is no simultaneity in inputting or outputting to the files involved. (For further information on using one buffer for multiple files, see the discussion on using SAME in the location column in the NEAT/3 REFERENCE MANUAL, INTRODUCTION AND DATA, tab 3, "Data Layout Sheets."

"1" is assumed if no entry is made.

5. FILE USAGE (ENTER S FOR A SOURCE FILE; D FOR A DESTINATION FILE; P FOR A PIGGYBACK FILE; R FOR A DISC SOURCE-DESTINATION FILE) □

Enter "S" if this file is to be a source file in this run. In Father-Son processing the old master file and the transaction file are source files. In Selective Serial processing the transaction file is the source file.

Enter "D" if this file is to be a destination file. Most files are initially created as destination files. An exception is a chained file with buckets. In Father-Son processing the new master file is a destination file.

Enter "P" if this file is to be a piggyback file. If a file is created in segments (such as A through M one day, N through Z the next day), declare the file a piggyback file each time a segment is added. In this manner, a file may be extended without rewriting the entire file.

Enter "R" if this file is to be a source-destination file. In Selective Serial processing or random processing the master file is a source-destination file.

6. TYPE OF BLOCKING - ENTER 2 FOR MULTI-RECORD BLOCKS. (1 FOR SINGLE-RECORD BLOCKS.) $\boxed{2}$

Enter "1" if each block is to contain only one record.

Enter "2" if each block is to contain more than one record. This is the usual type of blocking.

The type of blocking desired is determined by the size of the record and by the number of characters that a block can accept. In the case of the integrated disc on the Century 100, each block can contain up to 512 characters. If the record length is 100 characters, a maximum of five records can be placed in a block.

7. RECORD LENGTH (OR MAXIMUM SIZE IF VARIABLE LENGTH RECORDS) $\boxed{0\,1\,0\,0}$

Enter the number of characters in each fixed-length record if all records are the same length. Enter the number of characters in the maximum-sized record, including the variable length indicator, if the record length is variable. The maximum-size record, when using the integrated disc, is 512 characters, one sector. When using chained files, 502 is the maximum.

If the record is 100 characters long, enter 0100.

8. TYPE OF RECORDS (ENTER F FOR FIXED LENGTH, V FOR VARIABLE LENGTH WITH BINARY INDICATORS, D FOR VARIABLE LENGTH WITH DECIMAL INDICATORS) □

Enter "F" if the records are fixed-length.

Enter "V" if variable-length records are used in this file.

- The VLI must be the first field in the record.
- The VLI field must be two characters long.
- The VLI field must be a binary field.

"D" is not applicable to the disc.

9. IF VARIABLE, IS MAXIMUM PACKING TO BE PROVIDED ON OUTPUT? (Y OR N) [Y]

Enter "Y" if maximum packing of variable-length records is desired. If maximum packing is specified, the space remaining in a block is compared to the actual length of the record to be output. The record is placed in this space if the space is capable of accepting it.

To best use disc space with variable-length records, maximum packing should be specified. However, such a selection requires more memory space for both additional coding and a work area. The work-area option must be used with the PUT instruction when maximum packing is specified. See the NEAT/3 REFERENCE MANUAL, INSTRUCTIONS, tab 1, "PUT Instructions."

Enter "N" if maximum packing of records is not desired. If maximum packing is not specified, the space remaining in a block is compared to the maximum indicated length of a variable-length record. If this space is found to be incapable of accepting a maximum-size record, the block is written out with that space unused.

Maximum packing may not be desired if disc space is not critical, but due to program length, memory space is critical. The work-area option may be excluded from the PUT instruction when "N" is entered here.

Enter "N" if single record blocking was specified in answer to Question 6 of this sheet.

10. MAXIMUM BLOCK LENGTH (DISC - ON CHAINED FILES, THE BLOCK LENGTH MUST INCLUDE TEN CHARACTERS FOR THE BLOCK HEADER INDICATOR) (MAGNETIC TAPE - THE BLOCK LENGTH MUST INCLUDE THE BLOCK LENGTH INDICATOR IF PRESENT) [0,5,0,0]

Enter the total number of characters in a block. On the integrated disc unit for the NCR Century 100, for example, a block may contain up to 512 characters. If each record were 100 characters long, the block could contain a maximum of 5 records or 500 characters; therefore, 0500 should be entered.

NOTE

When working with fixed-length records, the maximum block length must be an even multiple of the record size.

If this is a chained file, add ten to the total block length for the chaining characters. These characters are inserted automatically by the software; however, the programmer must define the block length to include them. Applying the above example to a chained file results in an entry of 510 characters (0510).

Optional common trunk disc units on the NCR Century 100 have a maximum block length of 512, 1024, or 2048 characters. All disc units on the NCR Century 200 have a maximum block length of 512, 1024, 2048, or 4096 characters. Caution must be used, since a specified block length of 520 characters would cause two entire sections to be assigned.

11. IS A RESCUE POINT DESIRED AT EACH END OF SECTION? (Y OR N) (IF BLANK, N IS ASSUMED) □

Enter "Y" if a rescue point is desired each time this file reaches the end of a section. If this is a destination file, the rescue point is placed at the beginning of the next section. If this is a one-section destination file, no rescue point is possible or necessary. If this is a source file, the rescue point is placed on the Standard Rescue File. (See next question for additional information on the Standard Rescue File.)

"N" is assumed if no entry is made.

Generally, if a program uses source and destination files, the programmer specifies that the rescue point be taken for the file that reaches end-of-section most often. Rescue points can be specified for a source file even if it is the only data file in a run, as in a trial-balance run. This assumes that a standard rescue file exists.

12. IS THIS THE STANDARD RESCUE FILE? (Y OR N) (DISC – IF YES, THE FILE MAY BE USED FOR RESCUES ONLY) (IF BLANK, N IS ASSUMED) □

Enter "Y" if this file is to be the Standard Rescue File. Since it will be written into, it must be a destination file. Only rescue dumps may be placed in a Standard Rescue File on disc.

If this is a Standard Rescue File, a set of file specification sheets must be filled out according to the following table, and a record description must be included on a data layout sheet.

Sheet 1

QUES.	ENTRY	
1, 2, 3		Standard entries.
4	1	1 buffer.
5	D	Destination file.
6	1	Single-record blocks.
8	F	Fixed-length records.
10	0001	Block length. The Rescue routine writes blocks of 512 characters. Since the rescue routine does not use a buffer area to accomplish this, an entry of 0001 saves memory space.
11	N	No rescue is possible on a rescue file.
12	Y	Standard rescue file.
13,14,15,16		Standard entries.

Sheet 2

This sheets, which contains optional entries, is not needed for a Standard Rescue File on disc.

Sheet 3

QUES.	ENTRY	
1		Standard entry.
2	X	Symbolic designator for the unit containing the file.
4	01	The rescue file is composed of 1 section.
5	X	The number of sectors must be large enough to contain at least one rescue dump. The minimum number of sectors is 80 + (2 x K). For a 16K NCR Century 100, 112 sectors are needed for each rescue dump saved [80 + (2 x 16) = 112].
9	E	The indicated number of sectors requested is exactly the number required.

For further information concerning rescue, restart, and Standard Rescue files see the Operating System Manual.

"N" is assumed if no entry is made.

All sets of file specification sheets must be followed by a record description. For a Standard Rescue File, fill out a record description with a length of 1 on a data layout worksheet.

x	REFERENCE	x CODE	LOCATION	x LENGTH	x DP	x TYPE	VALUE OR PICTURE
7	8 9 10 11 12 13 14 15 16 17	18	19 20 21 22 23	24 25 26 27	28 29	30	31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
D		R		0 0 0 1			

13. TYPE OF DATING PERIOD □□□

14. ACCEPTABLE PERIOD (EARLIEST) FOR SOURCE, PIGGYBACK, AND SOURCE-DESTINATION FILES. NOT USED ON DESTINATION FILES. □□□

15. RETENTION PERIOD/FOR DESTINATION, PIGGYBACK, OR SOURCE-DESTINATION FILES. OPTIONALLY, THE LATEST ACCEPTABLE PERIOD FOR SOURCE FILES. □□□

Questions 13, 14, and 15, which deal with dates for file protection, are interrelated and are therefore explained together.

The dates entered here instruct the software to accept only those files created within a certain period, thereby guaranteeing that the current version of a file is processed. The dates further define how long a file must be saved for backup. Files may not be written over until this backup date (retention period) has expired; however, files with an expired retention period are acceptable as source files.

Dates are specified as either work days (WD), work weeks (WW), or work months (WM), relative to the current date. These work periods are based on a five-day week (with holidays considered).

Generation numbers (GEN) may also be used. The programmer may specify that the latest generation or the oldest generation on a particular disc pack is to be used. He may also specify how many generations of a file are to be saved.

All generations do not have to be available for the software to write over an old generation of the file. The software need only know the number of generations to be kept and the generation being created. For example, if generation 7 is being created and three generations are to be saved, the software would write over generation 2 if this generation were present on the mounted disc pack.

The software must have the previous generation available in the form of a source, destination, or source-destination file to compute the generation number for the file to be created.

When working with generation numbers for source files, the programmer may use any of the following three entries to answer question 14.

1. NEW - The newest generation on the disc pack mounted is to be used. It is the operator's responsibility to mount the proper pack.

2. OLD - The oldest generation on the disc pack mounted is to be used. It is the operator's responsibility to mount the proper pack.

3. # - The actual generation number if known. This is generally used only when debugging a program or for input to a utility program.

- Example of Dates

To illustrate how dates might be used, assume the following:

- A destination file (file A) is being created today as a product of a Father-Son update run.
- File A is to be saved for two weeks as a backup file.
- File A is to be used as a source file in exactly one week when the program is run again.
- Today is Tuesday, January 3. (Use the following calendar in the example.)

JANUARY 19--

S	M	T	W	T	F	S
1	2	③	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

To comply with these assumptions, the dates on the file specification sheets for destination file A might be filled in as follows:

13. TYPE OF DATING PERIOD W W Ø

14. ACCEPTABLE PERIOD (EARLIEST) FOR SOURCE, PIGGYBACK, AND SOURCE-DESTINATION FILES. NOT USED ON DESTINATION FILES. Ø Ø Ø

15. RETENTION PERIOD FOR DESTINATION, PIGGYBACK, OR SOURCE-DESTINATION FILES. OPTIONALLY, THE LATEST ACCEPTABLE PERIOD FOR SOURCE FILES. 0 0 2

These entries indicate that the file is to be kept for two workweek periods beginning next Monday. (Work weeks begin on Monday and end on Friday.) Therefore, it may be written over after January 20.

When File A is used next week (January 10) as a source file, it must meet the programmer's qualifications for source files as specified on the file specification sheets.

The dates on the file specification sheets for source File A might be filled in as follows:

13. TYPE OF DATING PERIOD — WWØ

14. ACCEPTABLE PERIOD (EARLIEST) FOR SOURCE, PIGGYBACK, AND SOURCE-DESTINATION FILES. NOT USED ON DESTINATION FILES. — 001

15. RETENTION PERIOD FOR DESTINATION, PIGGYBACK, OR SOURCE-DESTINATION FILES. OPTIONALLY, THE LATEST ACCEPTABLE PERIOD FOR SOURCE FILES. — ØØØ

These entries indicate that a file is to be accepted as a source file if it was created during the previous work week. Therefore, a source file is acceptable on January 10 if it was created in the period between January 2 and January 6. File A was created on January 3 and is acceptable.

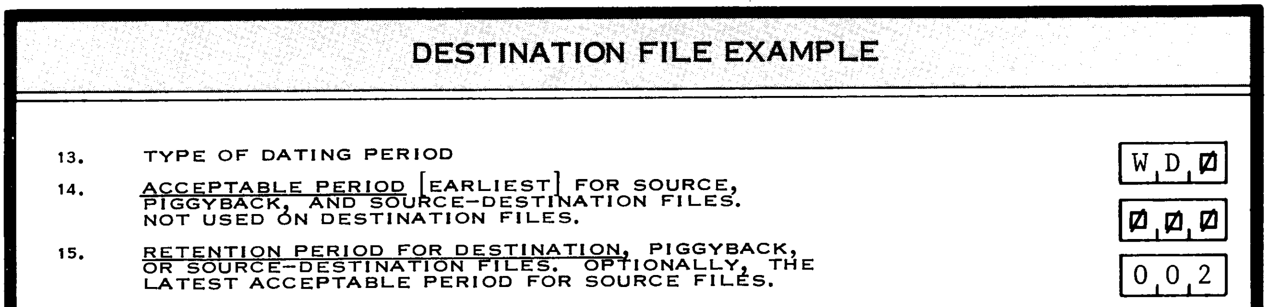
JANUARY 19--						
S	M	T	W	T	F	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	[✓ 24]	25	26	27	28
29	30	31				

Question 15 is optional for source files and is used to further define an acceptable file. For example, 002 might be entered for Question 14 and 001 for Question 15, with Question 13 remaining WWØ. According to these entries, a file is to be accepted if it was created in a period that started at the beginning of two work weeks back, and ended at the end of 1 work week back. If the program was run on January 24, a source file created between January 9 and January 20 would be acceptable.

If Question 15 was left blank, and 14 remained 002, a source file created between January 9 and January 13 only would be accepted. If more than one file, all with the correct name, was created in the specified period, the first one found is used.

The Disc Directory contains an entry that indicates when each file was created and when it expires. The dates in the directory are used to determine if a previously created file meets the requirements outlined on the file specification sheets.

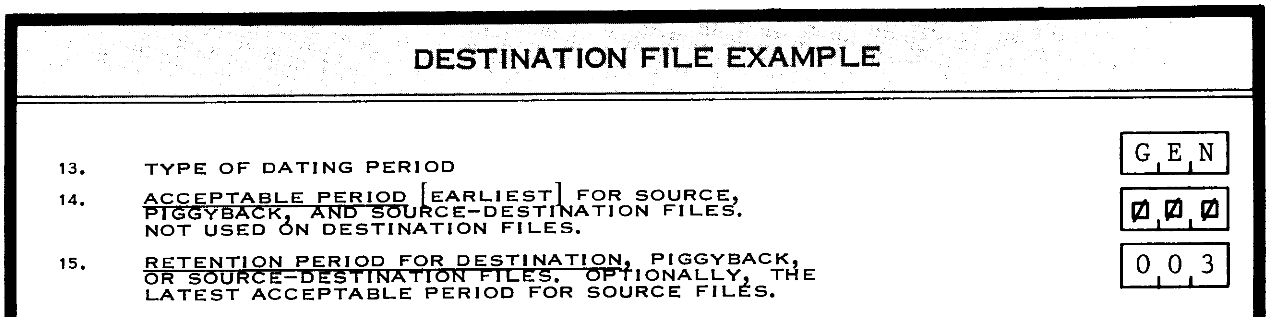
- Further Examples of Using Dates for File Protection



DESTINATION FILE EXAMPLE

13.	TYPE OF DATING PERIOD	W D Ø
14.	ACCEPTABLE PERIOD [EARLIEST] FOR SOURCE, PIGGYBACK, AND SOURCE-DESTINATION FILES. NOT USED ON DESTINATION FILES.	Ø Ø Ø
15.	RETENTION PERIOD FOR DESTINATION, PIGGYBACK, OR SOURCE-DESTINATION FILES. OPTIONALLY, THE LATEST ACCEPTABLE PERIOD FOR SOURCE FILES.	0 0 2

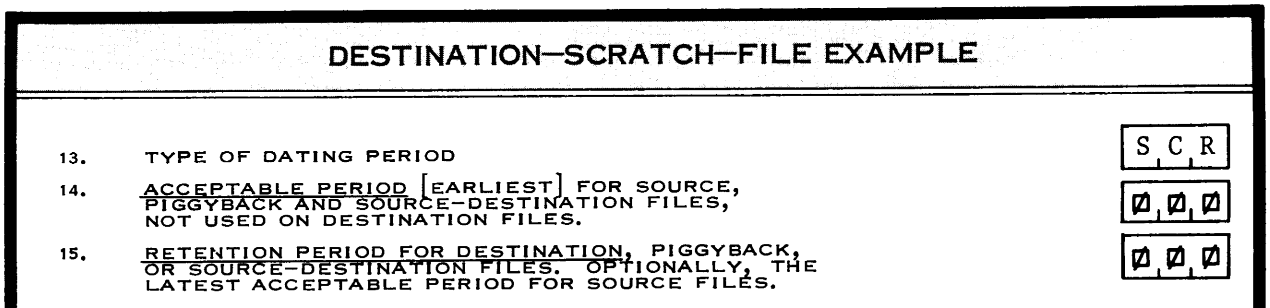
The created file is protected for two additional work days; that is, it may not be written over (destroyed) until the third work day after its creation.



DESTINATION FILE EXAMPLE

13.	TYPE OF DATING PERIOD	G E N
14.	ACCEPTABLE PERIOD [EARLIEST] FOR SOURCE, PIGGYBACK, AND SOURCE-DESTINATION FILES. NOT USED ON DESTINATION FILES.	Ø Ø Ø
15.	RETENTION PERIOD FOR DESTINATION, PIGGYBACK, OR SOURCE-DESTINATION FILES. OPTIONALLY, THE LATEST ACCEPTABLE PERIOD FOR SOURCE FILES.	0 0 3

The created file is protected for three generations. It expires when another file is created whose generation number is greater than this file's generation number plus three. This example can be interpreted as follows: keep this generation and three generations back.



DESTINATION—SCRATCH—FILE EXAMPLE

13.	TYPE OF DATING PERIOD	S C R
14.	ACCEPTABLE PERIOD [EARLIEST] FOR SOURCE, PIGGYBACK AND SOURCE-DESTINATION FILES, NOT USED ON DESTINATION FILES.	Ø Ø Ø
15.	RETENTION PERIOD FOR DESTINATION, PIGGYBACK, OR SOURCE-DESTINATION FILES. OPTIONALLY, THE LATEST ACCEPTABLE PERIOD FOR SOURCE FILES.	Ø Ø Ø

This indicates the file is a scratch file which will expire at the end of the day on which it was created; that is, it may be written over the next day. If the programmer does not desire to keep the file for the complete day (i.e., the disc space used by this file is needed during another run), he may use the CLOSEO instruction to obsolete the file.

SOURCE—SCRATCH—FILE EXAMPLE

13. TYPE OF DATING PERIOD — S C R

14. ACCEPTABLE PERIOD [EARLIEST] FOR SOURCE, PIGGYBACK, AND SOURCE-DESTINATION FILES. NOT USED ON DESTINATION FILES. — Ø Ø Ø

15. RETENTION PERIOD FOR DESTINATION, PIGGYBACK, OR SOURCE-DESTINATION FILES. OPTIONALLY, THE LATEST ACCEPTABLE PERIOD FOR SOURCE FILES. — Ø Ø Ø

This indicates that the program accepts a scratch file created the day the program is run.

SOURCE FILE EXAMPLE

13. TYPE OF DATING PERIOD — W D Ø

14. ACCEPTABLE PERIOD [EARLIEST] FOR SOURCE, PIGGYBACK, AND SOURCE-DESTINATION FILES. NOT USED ON DESTINATION FILES. — 0 0 5

15. RETENTION PERIOD FOR DESTINATION, PIGGYBACK, OR SOURCE-DESTINATION FILES. OPTIONALLY, THE LATEST ACCEPTABLE PERIOD FOR SOURCE FILES. — Ø Ø Ø

This indicates that the program accepts a source file if it was created five work days back. Only a file created on that day is accepted.

SOURCE FILE EXAMPLE

13. TYPE OF DATING PERIOD — G E N

14. ACCEPTABLE PERIOD [EARLIEST] FOR SOURCE, PIGGYBACK, AND SOURCE-DESTINATION FILES. NOT USED ON DESTINATION FILES. — N E W

15. RETENTION PERIOD FOR DESTINATION, PIGGYBACK, OR SOURCE-DESTINATION FILES. OPTIONALLY, THE LATEST ACCEPTABLE PERIOD FOR SOURCE FILES. — Ø Ø Ø

This indicates that the latest generation on the disc pack presently mounted is to be used. It is the operator's responsibility to mount the proper pack.

PIGGYBACK FILE EXAMPLE

13.	TYPE OF DATING PERIOD	W W Ø
14.	ACCEPTABLE PERIOD [EARLIEST] FOR SOURCE, PIGGYBACK, AND SOURCE-DESTINATION FILES. NOT USED ON DESTINATION FILES.	0 0 0
15.	RETENTION PERIOD FOR DESTINATION, PIGGYBACK, OR SOURCE-DESTINATION FILES. OPTIONALLY, THE LATEST ACCEPTABLE PERIOD FOR SOURCE FILES.	0 0 2

This indicates that a file created the same week is to be extended. All additional segments to the file will have the same creation and expiration date as the original segment.

If the file cannot be located, the operator is given the option of creating a new one. If a new file is created, the answer to Question 15 is used to establish the expiration date for it. In this way, the same program may be used to create a piggyback file and extend it.

SOURCE—DESTINATION FILE EXAMPLE

13.	TYPE OF DATING PERIOD	W W Ø
14.	ACCEPTABLE PERIOD [EARLIEST] FOR SOURCE, PIGGYBACK, AND SOURCE-DESTINATION FILES. NOT USED ON DESTINATION FILES.	0 0 1
15.	RETENTION PERIOD FOR DESTINATION, PIGGYBACK, OR SOURCE-DESTINATION FILES. OPTIONALLY, THE LATEST ACCEPTABLE PERIOD FOR SOURCE FILES.	0 0 3

This indicates that a file created one week ago is to be used.

If the file cannot be located, the operator is given the option of creating a new one. If a new file is created, the answer to Question 15 is used to establish the expiration date for it. In this way, the same program may be used to create a source-destination file and update it.

- Virtual Date and Actual Date

  The date the file was created is obtained from an entry made to the system at the start of each workday. This entry contains two dates: the actual date and the virtual date.

  - Actual date is today's date.
  - Virtual date is the date the program is scheduled to be run.

  Generally, the actual date and the virtual date are the same. However, an unavoidable interruption could interfere with a program being run on a scheduled date. In this case, the program could be run the next day, using the previous day's date as the virtual date.

- Example of Virtual and Actual Date

Phase 1: The program is to write File A and save it for one additional day. This program is normally run on January 4; but since the processing could not be done on that day, the program is run on January 5.

The actual and virtual dates are entered into the system at the beginning of the day as follows:

Actual date - January 5 Virtual date - January 4

When File A is created, the Disc Directory will contain the following creation and expiration dates:

Date created - January 4 Date expires - January 6

The files written on January 5 may now be used in Phase 2.

Phase 2: The program is to use File A one day after it is created. This program is usually run on January 5.

New actual and virtual dates are entered after completion of the programs usually run on January 4.

Actual date - January 5 Virtual date - January 5

File A is acceptable since it was written using the virtual date January 4, and this is actually January 5. Any files written in Phase 2 contain January 5 as the date created.

Programs may be run over a period of days using virtual dates until the work is back on schedule. When the system is back on schedule, all files contain valid dates that are acceptable to their associated programs.

16. FILE NAME

M	A	G	Z	F	I	L	E	Ø	Ø

Enter the file name as it is to appear in the Disc Directory. For Father-Son processing this name should be different than the file reference name. If the file is a source-destination file, this entry mav be the same as the file reference name. (For further detail, see the NEAT/3 REFERENCE MANUAL, FILES, "NCR Century File Concepts.")

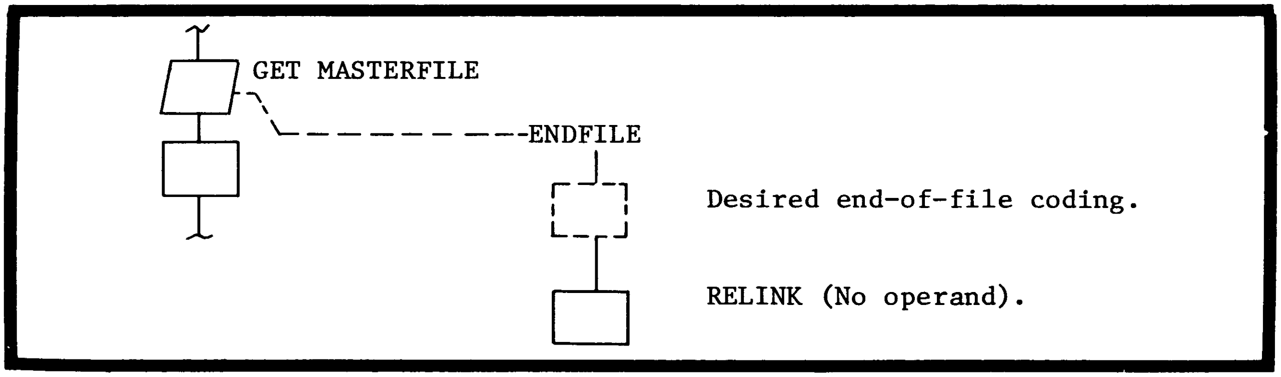
17. END OF FILE EXIT (NOT USED ON DESTINATION FILES) OR ERROR EXIT FOR RANDOM PROCESSING INSTRUCTIONS.

ENDFILEØØØ

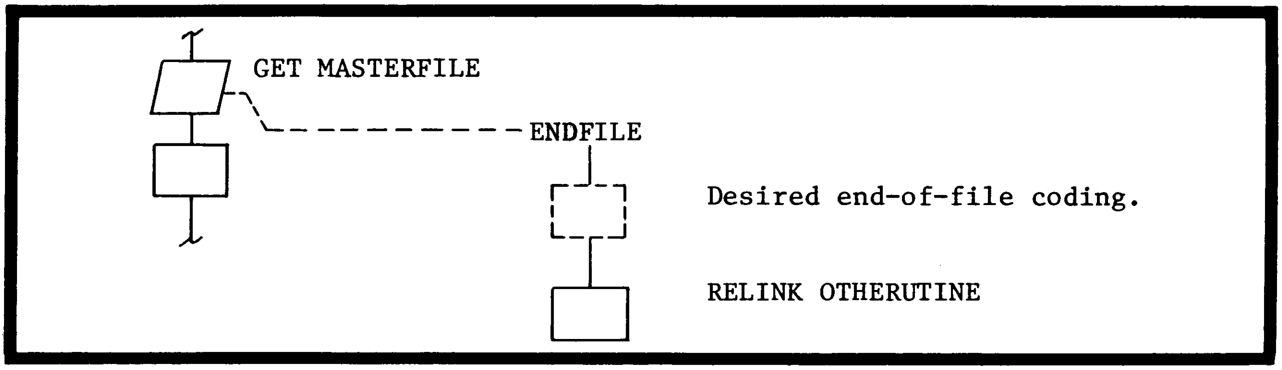
END-OF-FILE EXIT

Enter the name of the routine to be given control when end-of-file is reached. The software links to this routine when the end of a sequentially processed source or source-destination file is reached.

If the programmer wishes to return control to the instruction following the one that caused the link to the end-of-file routine, he may do so by using a RELINK instruction without an operand.



If the programmer wishes to return control to a different instruction or routine, he may do so by using a RELINK instruction that names the desired routine as an operand. The LINK or BRANCH instruction may not be used for this purpose.



For further details concerning RELINK and RELINK with an operand, see the NEAT/3 REFERENCE MANUAL, INSTRUCTIONS, tab 2, "Link and Relink Instructions."

NOTE

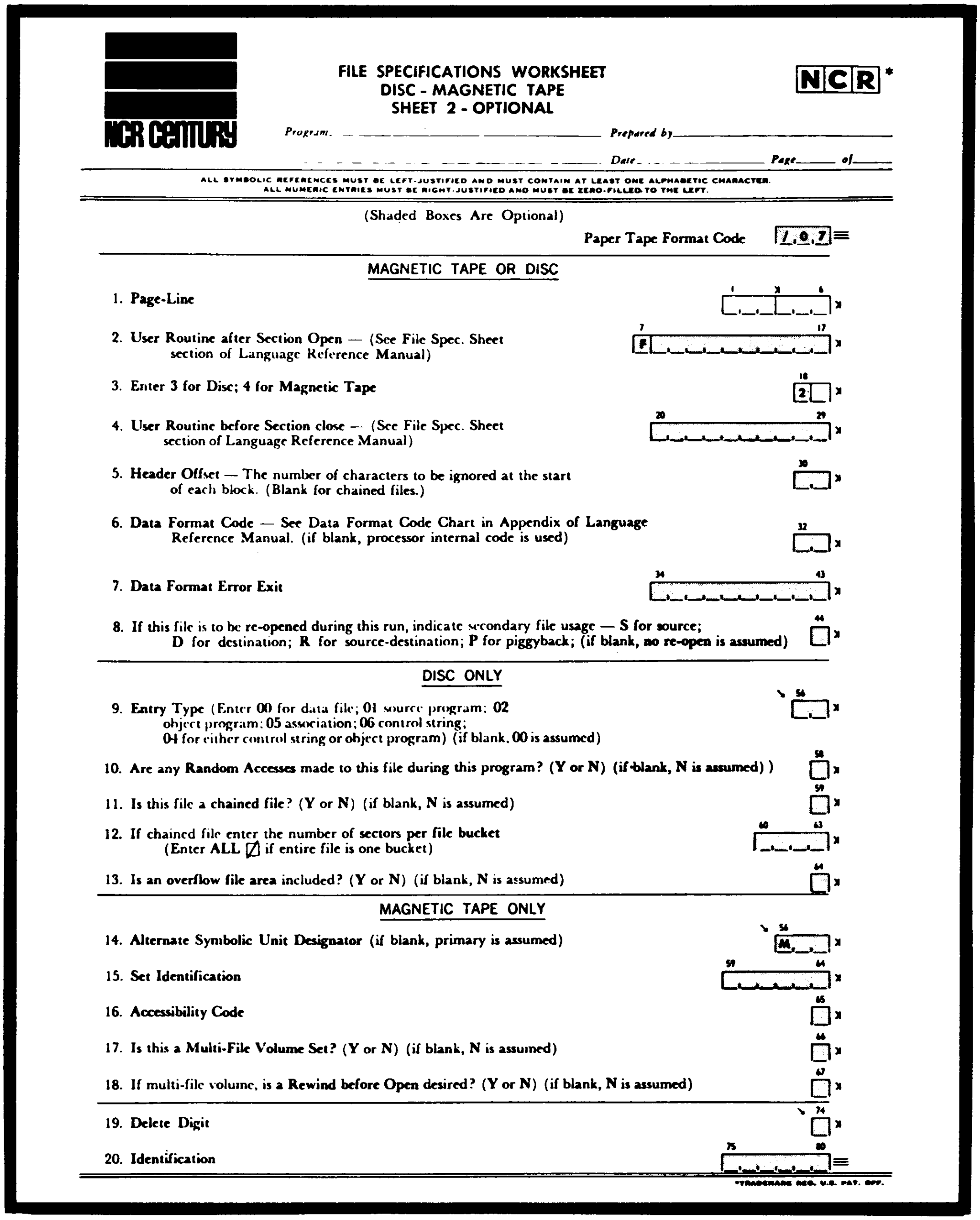
Once an end-of-file exit is taken, the last record in the input buffer is no longer accessible to the program.

ERROR EXIT FOR RANDOM PROCESSING INSTRUCTIONS

Enter the name of the routine to be given control if a random access is used specifying an illegal section or illegal relative sector number. The routine that is given control must end with a RELINK instruction. This entry is used when randomly processing a source-destination file.

SHEET 2

The information on this sheet is optional. If none of the options are desired, the entire sheet may be eliminated.



NCR CENTURY

FILE SPECIFICATIONS WORKSHEET
DISC - MAGNETIC TAPE
SHEET 2 - OPTIONAL

NCR*

Program ______ Prepared by ______
______ Date ______ Page ____ of ____

ALL SYMBOLIC REFERENCES MUST BE LEFT-JUSTIFIED AND MUST CONTAIN AT LEAST ONE ALPHABETIC CHARACTER.
ALL NUMERIC ENTRIES MUST BE RIGHT-JUSTIFIED AND MUST BE ZERO-FILLED TO THE LEFT.

(Shaded Boxes Are Optional)

Paper Tape Format Code 1 0 7

MAGNETIC TAPE OR DISC

1. Page-Line 1 3 6
2. User Routine after Section Open — (See File Spec. Sheet section of Language Reference Manual) 7 F 17
3. Enter 3 for Disc; 4 for Magnetic Tape 18 2
4. User Routine before Section close — (See File Spec. Sheet section of Language Reference Manual) 20 29
5. Header Offset — The number of characters to be ignored at the start of each block. (Blank for chained files.) 30
6. Data Format Code — See Data Format Code Chart in Appendix of Language Reference Manual. (if blank, processor internal code is used) 32
7. Data Format Error Exit 34 43
8. If this file is to be re-opened during this run, indicate secondary file usage — S for source; D for destination; R for source-destination; P for piggyback; (if blank, no re-open is assumed) 44

DISC ONLY

9. Entry Type (Enter 00 for data file; 01 source program; 02 object program; 05 association; 06 control string; 04 for either control string or object program) (if blank, 00 is assumed) 56
10. Are any Random Accesses made to this file during this program? (Y or N) (if blank, N is assumed)) 58
11. Is this file a chained file? (Y or N) (if blank, N is assumed) 59
12. If chained file enter the number of sectors per file bucket (Enter ALL ⍁ if entire file is one bucket) 60 63
13. Is an overflow file area included? (Y or N) (if blank, N is assumed) 64

MAGNETIC TAPE ONLY

14. Alternate Symbolic Unit Designator (if blank, primary is assumed) 56 M
15. Set Identification 59 64
16. Accessibility Code 65
17. Is this a Multi-File Volume Set? (Y or N) (if blank, N is assumed) 66
18. If multi-file volume, is a Rewind before Open desired? (Y or N) (if blank, N is assumed) 67
19. Delete Digit 74
20. Identification 75 80

*TRADEMARK REG. U.S. PAT. OFF.

The programmer should fill in the header, page-and-line number (Question 1), and the identification tag (position 75-80) as defined in the NEAT/3 REFERENCE MANUAL, INTRODUCTION AND DATA, tab 3, "Programming Worksheets." The paper tape format code must be punched if paper tape is used for input to the Compiler.

2. USER ROUTINE AFTER SECTION OPEN - [SEE FILE SPEC. SHEET SECTION OF LANGUAGE REFERENCE MANUAL]

F									

This optional entry contains the reference name of the routine to be given control immediately after the second and each subsequent section of the file is opened. Control is also transferred here after the first section is opened if an OPEN instruction is used for this file.

GET, PUT, OPEN, CLOSE, DEFALT, and other I/O instructions may not be executed in the user routine. RELINK must be the final instruction in the routine to return control to the software.

"F" is preprinted in position 7 and must be punched.

3. ENTER 3 FOR DISC, 4 FOR MAGNETIC TAPE

2	3

Enter "3" for disc.

"2" is preprinted in position 18 and must be punched.

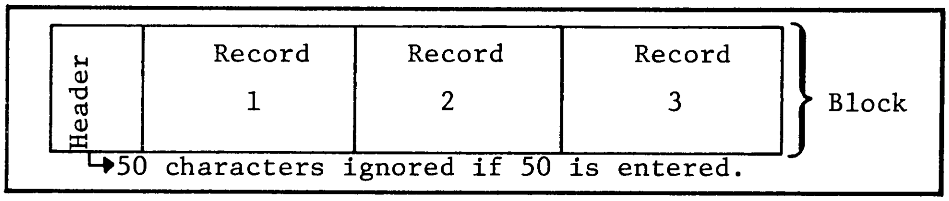
4. USER ROUTINE BEFORE SECTION CLOSE - [SEE FILE SPEC. SHEET SECTION OF LANGUAGE REFERENCE MANUAL]

This optional entry contains the reference name of the routine to be given control before each section is closed - with the exception of the last section. Control is also transferred here before the last section is closed if a CLOSE instruction is used for this file.

GET, PUT, OPEN, CLOSE, DEFALT, and other I/O instructions may not be executed in the user routine. RELINK must be the final instruction in the routine, to return control to the software.

5. HEADER OFFSET - THE NUMBER OF CHARACTERS TO BE IGNORED AT THE START OF EACH BLOCK. [BLANK FOR CHAINED FILES.]

This optional entry contains the number of characters that are to be ignored by the GET or RGET instruction when the first record in the block is requested. If 50 was entered, the GET or RGET instruction that affected the first record in a block would present a record beginning with the 51st character. This option is for standard files only; it does not refer to the 10 characters used for chaining in chained files.



Zero is assumed if no entry is made.

6. DATA FORMAT CODE - SEE DATA FORMAT CODE CHART IN APPENDIX OF LANGUAGE REFERENCE MANUAL. (IF BLANK, PROCESSOR INTERNAL CODE IS USED) □□

This optional entry contains a number that designates the format code for this file on the disc. This entry should be left blank since Century internal code is the only disc format code acceptable at this time.

7. DATA FORMAT ERROR EXIT □□□□□□□□□□

This optional entry is used only if a translation was asked for in Question 6. If an illegal character is detected during decode, control is given to the routine referenced here. (Since at present the disc uses only NCR Century interal code, no decode occurs.)

8. IF THIS FILE IS TO BE RE-OPENED DURING THIS RUN, INDICATE SECONDARY FILE USAGE - S FOR SOURCE, D FOR DESTINATION, R FOR SOURCE-DESTINATION, P FOR PIGGYBACK (IF BLANK, NO RE-OPEN IS ASSUMED) □

If this file is re-opened during the run, enter the letter that indicates the type of file desired upon re-opening. Multiple re-opens are permitted but must be restricted to the two types stated; however, the second re-open must declare the file type as being the same as at initial open (Question 5, Sheet 1).

Enter "S" if the file is first re-opened as a source file.

Enter "D" if the file is first re-opened as a destination file.

Enter "R" if the file is first re-opened as a source-destination file.

Enter "P" if the file is first re-opened as a piggyback file.

9. ENTRY TYPE (ENTER 00 FOR DATA FILE, 01 SOURCE PROGRAM, 02 OBJECT PROGRAM, 05 ASSOCIATION, 06 CONTROL STRING, 04 FOR EITHER CONTROL STRING OR OBJECT PROGRAM (IF BLANK, 00 IS ASSUMED) □□

This optional entry is primarily for utility routines, and the proper entry is indicated in the utility routine writeup.

"00" is assumed if no entry is made.

10. ARE ANY RANDOM ACCESSES MADE TO THIS FILE DURING THIS PROGRAM? (Y OR N) (IF BLANK, N IS ASSUMED) □

This optional entry is used if random processing occurs with this file.

Enter "Y" if random accesses are made to the file in this program.

If this entry is left blank no random accesses to this file are permitted in this program and the file must be processed sequentially.

11. IS THIS FILE A CHAINED FILE? (Y OR N) (IF BLANK, N IS ASSUMED) □

This optional entry is used with chained files only.

Enter "Y" if this is a chained file.

If this is a new source-destination file and "Y" is entered, the software constructs chaining characters for each sector and places them in the 10 characters reserved for chaining in that block (sector). End-of-bucket marks are also constructed and placed in the 10 chaining characters of the last block (sector) in each bucket.

Leave blank if this is a standard file. ("N" is assumed.)

12. IF CHAINED FILE, ENTER THE NUMBER OF SECTORS PER FILE BUCKET (ENTER ALLØ IF ENTIRE FILE IS ONE BUCKET) □□□□

This optional entry must be used if this is a chained file.

Enter the number of sectors to be placed in each bucket. Special software needed when buckets are used is included in the object program. Enter ALLØ if the file is to be contained in one bucket.

Leave blank if the file is a standard file. ("N" is assumed.)

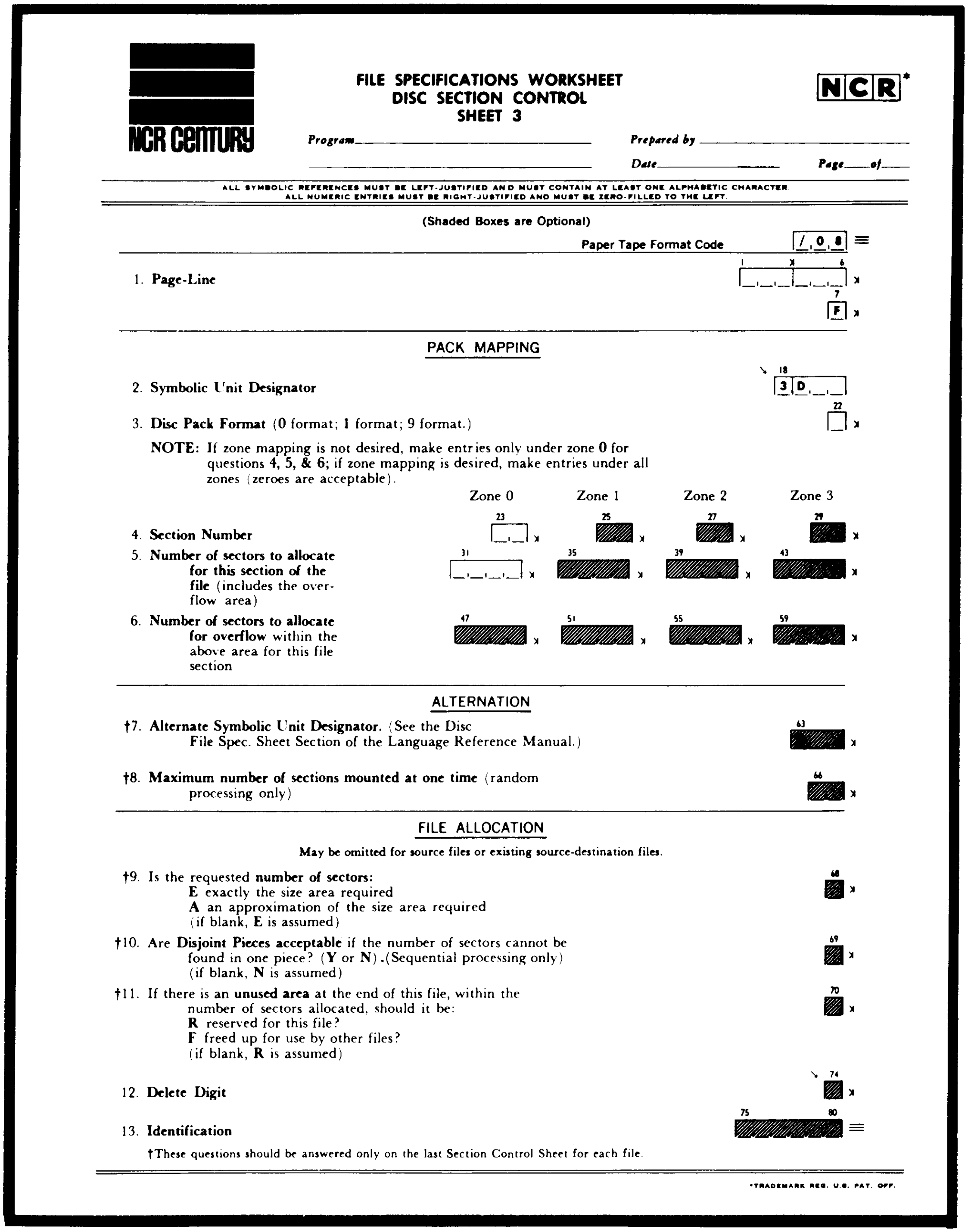
13. IS AN OVERFLOW FILE AREA INCLUDED? (Y OR N) (IF BLANK, N IS ASSUMED) □

This optional entry pertains to chained files only.

Enter "Y" if an overflow area is desired on a new file, or if an overflow area exists on a previously constructed file. Special software is included when an overflow area is constructed or is present. Use of an overflow area is strongly recommended when the INSERT instruction is used.

Leave blank if an overflow area is not desired or if an overflow area does not exist. ("N" is assumed.)

SHEET 3



NCR CENTURY

FILE SPECIFICATIONS WORKSHEET
DISC SECTION CONTROL
SHEET 3

NCR*

Program__________ Prepared by __________

Date__________ Page____of____

ALL SYMBOLIC REFERENCES MUST BE LEFT-JUSTIFIED AND MUST CONTAIN AT LEAST ONE ALPHABETIC CHARACTER
ALL NUMERIC ENTRIES MUST BE RIGHT-JUSTIFIED AND MUST BE ZERO-FILLED TO THE LEFT

(Shaded Boxes are Optional)

Paper Tape Format Code /08 ≡

1. Page-Line 1 □□□ 4 □□□ 6 ⌁ 7 F ⌁

PACK MAPPING

2. Symbolic Unit Designator 18 3 D □□

3. Disc Pack Format (0 format; 1 format; 9 format.) 22 □ ⌁

NOTE: If zone mapping is not desired, make entries only under zone 0 for questions 4, 5, & 6; if zone mapping is desired, make entries under all zones (zeroes are acceptable).

	Zone 0	Zone 1	Zone 2	Zone 3
4. Section Number	23 □□ ⌁	25 ▨ ⌁	27 ▨ ⌁	29 ▨ ⌁
5. Number of sectors to allocate for this section of the file (includes the overflow area)	31 □□□□ ⌁	35 ▨ ⌁	39 ▨ ⌁	43 ▨ ⌁
6. Number of sectors to allocate for overflow within the above area for this file section	47 ▨ ⌁	51 ▨ ⌁	55 ▨ ⌁	59 ▨ ⌁

ALTERNATION

†7. Alternate Symbolic Unit Designator. (See the Disc File Spec. Sheet Section of the Language Reference Manual.) 63 ▨ ⌁

†8. Maximum number of sections mounted at one time (random processing only) 66 ▨ ⌁

FILE ALLOCATION

May be omitted for source files or existing source-destination files.

†9. Is the requested number of sectors:
E exactly the size area required
A an approximation of the size area required
(if blank, E is assumed) 68 ▨ ⌁

†10. Are Disjoint Pieces acceptable if the number of sectors cannot be found in one piece? (Y or N).(Sequential processing only)
(if blank, N is assumed) 69 ▨ ⌁

†11. If there is an unused area at the end of this file, within the number of sectors allocated, should it be:
R reserved for this file?
F freed up for use by other files?
(if blank, R is assumed) 70 ▨ ⌁

12. Delete Digit 74 ▨ ⌁

13. Identification 75 ▨ 80 ≡

†These questions should be answered only on the last Section Control Sheet for each file.

*TRADEMARK REG. U.S. PAT. OFF.

The programmer should fill in the header, page-and-line number (Question 1), and the identification tag (position 75-80) as defined in the NEAT/3 REFERENCE MANUAL, INTRODUCTION AND DATA, tab 3, "Programming Worksheets." The paper tape format code must be punched if paper tape is used for input to the Compiler.

"F" is preprinted in position 7 and must be punched.

2. SYMBOLIC UNIT DESIGNATOR 3 D

3. DISC PACK FORMAT (0 FORMAT; 1 FORMAT; 9 FORMAT)

NOTE: IF ZONE MAPPING IS NOT DESIRED, MAKE ENTRIES ONLY UNDER ZONE 0 FOR QUESTIONS 4, 5, & 6; IF ZONE MAPPING IS DESIRED, MAKE ENTRIES UNDER ALL ZONES (ZEROES ARE ACCEPTABLE) .

	ZONE 0	ZONE 1	ZONE 2	ZONE 3
4. SECTION NUMBER	□	□	□	□
5. NUMBER OF SECTORS TO ALLOCATE FOR THIS SECTION OF THE FILE (INCLUDES THE OVERFLOW AREA)	□	□	□	□
6. NUMBER OF SECTORS TO ALLOCATE FOR OVERFLOW WITHIN THE ABOVE AREA FOR THIS FILE SECTION	□	□	□	□

Questions 2, 3, 4, 5, and 6, which deal with pack mapping, are interrelated and are therefore explained together.

Questions 3, 4, 5, and 6 initially pertain to the disc pack residing on the unit named in Question 2. On an NCR Century System with one dual disc unit, the two possible Symbolic Unit Designators for Question 2 are D01 and D02.

Question 3 is used to indicate in which format the disc pack was initialized. Enter "1" if the pack was initialized in Format 1 or enter "9" if the pack was initialized in Format 9. (Format 0 is reserved for future implementation.)

Questions 4, 5, and 6 are used to map the files on the pack. If the file currently exists, only question 4 need be answered. If a new file is to be created, all three questions are pertinent. The manner in which the files are mapped indicates whether the pack is a one-zone pack or a four-zone pack.

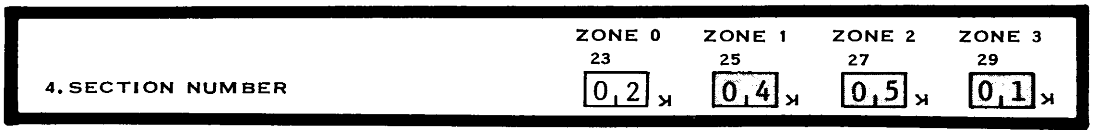
If a one-zone pack is desired, Questions 4, 5, and 6 must be answered under zone 0 only. Entries under zones 1, 2, and 3 must be left blank.

If a four-zone pack is desired, Questions 4, 5, and 6 must be answered under all zones being used. Zeros must be entered for Question 4 under those zones that are not used for this file.

The following conventions must be observed when assigning section numbers (question 4):

- All sections of a file must be numbered consecutively, beginning with 01. The maximum number that can be used is 99. For example, the first four sections of a file would be assigned section numbers 01, 02, 03 and 04.

- Although section numbers must be consecutive, their <u>zone</u> assignment need not be in ascending order. For example, file section numbers 02, 04, 03 and 01 could be presented for zones 0, 1, 2 and 3 respectively. Consider the following illustration:

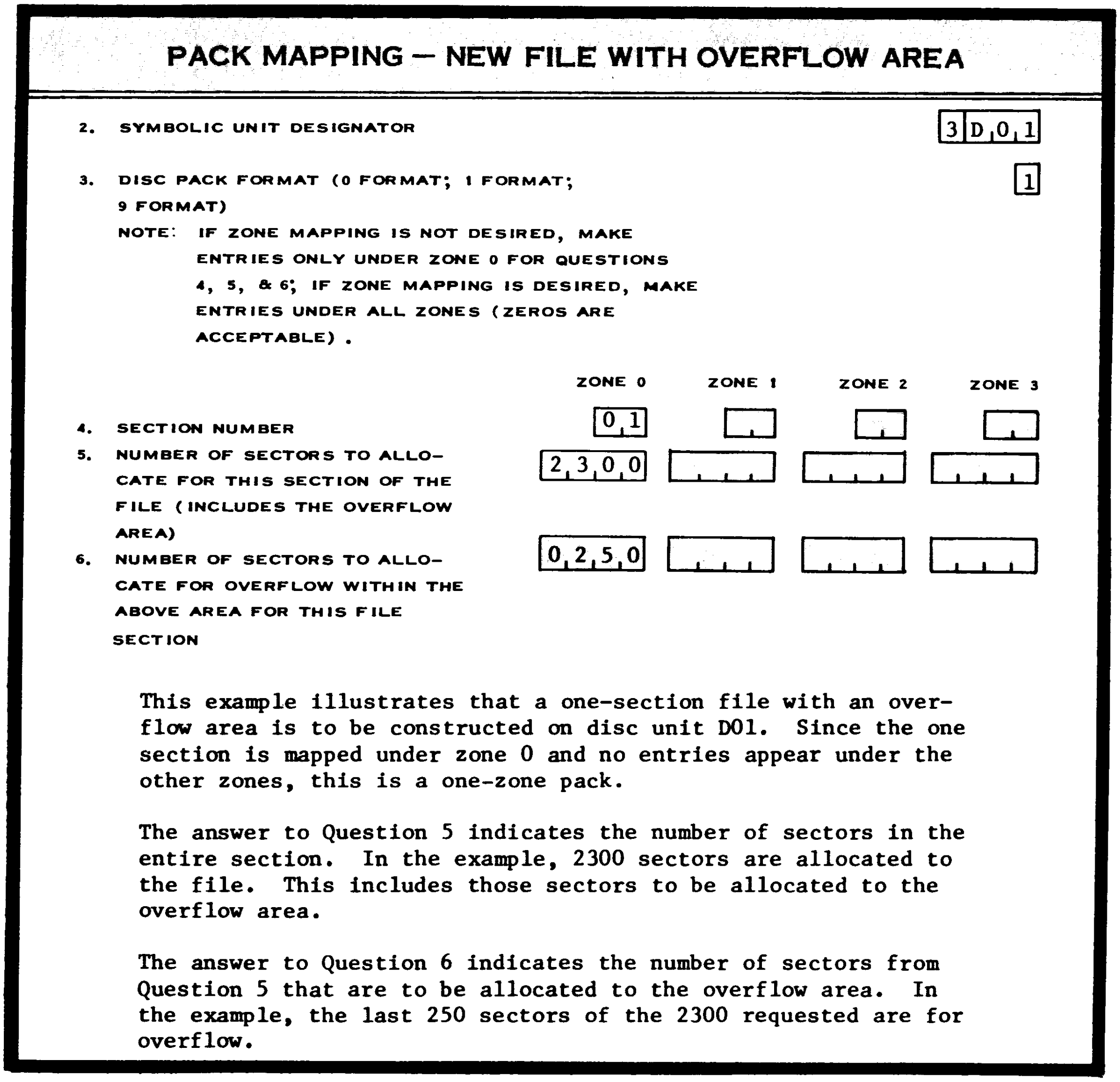


	ZONE 0 23	ZONE 1 25	ZONE 2 27	ZONE 3 29
4. SECTION NUMBER	0,2	0,4	0,5	0,1

- If more than one disc pack mapping worksheet (sheet 3) is used, the lowest section number on any following sheet must be higher than the highest section number presented on the preceding sheet.

- <u>Examples of Pack Mapping - One Zone</u>

Following are three examples of pack mapping on a one-zone pack.



PACK MAPPING — NEW FILE WITH OVERFLOW AREA

2. SYMBOLIC UNIT DESIGNATOR 3 D,0,1

3. DISC PACK FORMAT (0 FORMAT; 1 FORMAT; 9 FORMAT) 1

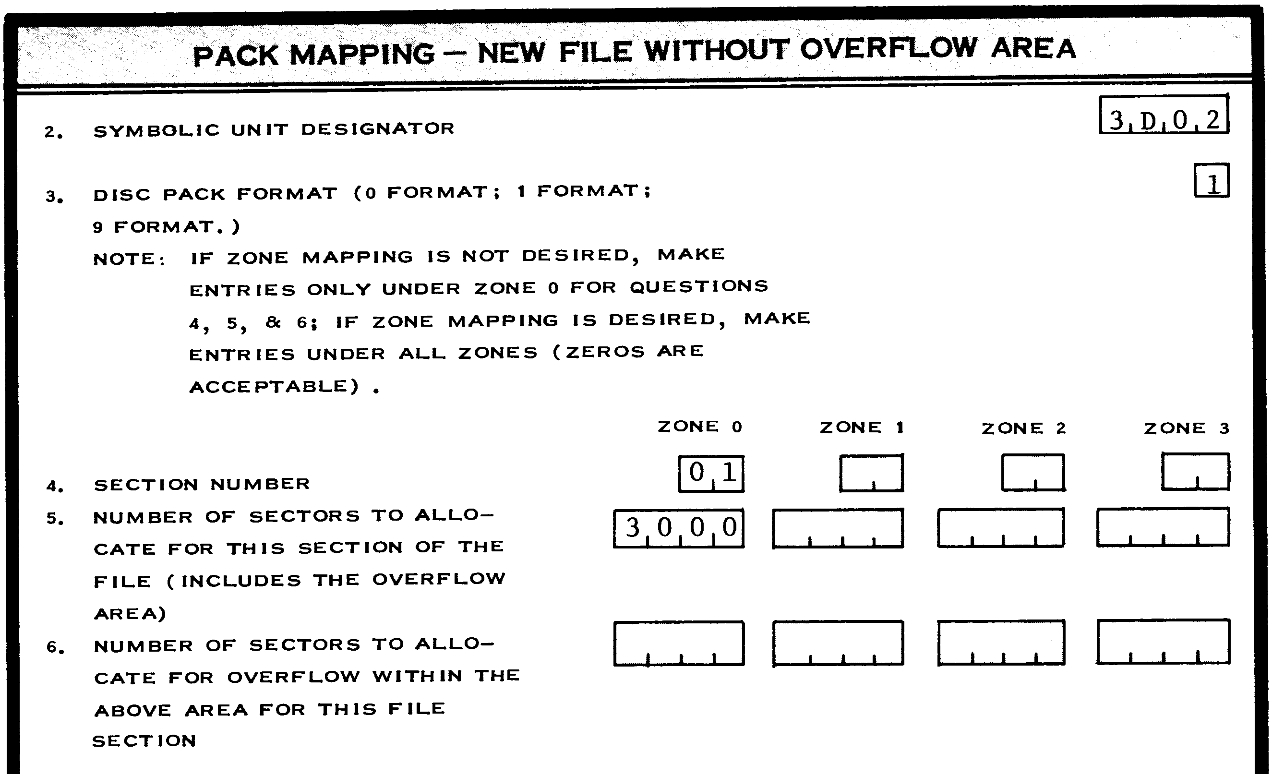
NOTE: IF ZONE MAPPING IS NOT DESIRED, MAKE ENTRIES ONLY UNDER ZONE 0 FOR QUESTIONS 4, 5, & 6; IF ZONE MAPPING IS DESIRED, MAKE ENTRIES UNDER ALL ZONES (ZEROS ARE ACCEPTABLE).

	ZONE 0	ZONE 1	ZONE 2	ZONE 3
4. SECTION NUMBER	0,1			
5. NUMBER OF SECTORS TO ALLOCATE FOR THIS SECTION OF THE FILE (INCLUDES THE OVERFLOW AREA)	2,3,0,0			
6. NUMBER OF SECTORS TO ALLOCATE FOR OVERFLOW WITHIN THE ABOVE AREA FOR THIS FILE SECTION	0,2,5,0			

This example illustrates that a one-section file with an overflow area is to be constructed on disc unit D01. Since the one section is mapped under zone 0 and no entries appear under the other zones, this is a one-zone pack.

The answer to Question 5 indicates the number of sectors in the entire section. In the example, 2300 sectors are allocated to the file. This includes those sectors to be allocated to the overflow area.

The answer to Question 6 indicates the number of sectors from Question 5 that are to be allocated to the overflow area. In the example, the last 250 sectors of the 2300 requested are for overflow.



PACK MAPPING — NEW FILE WITHOUT OVERFLOW AREA

2. SYMBOLIC UNIT DESIGNATOR 3D02

3. DISC PACK FORMAT (0 FORMAT; 1 FORMAT; 9 FORMAT.) 1

NOTE: IF ZONE MAPPING IS NOT DESIRED, MAKE ENTRIES ONLY UNDER ZONE 0 FOR QUESTIONS 4, 5, & 6; IF ZONE MAPPING IS DESIRED, MAKE ENTRIES UNDER ALL ZONES (ZEROS ARE ACCEPTABLE) .

	ZONE 0	ZONE 1	ZONE 2	ZONE 3
4. SECTION NUMBER	01			
5. NUMBER OF SECTORS TO ALLOCATE FOR THIS SECTION OF THE FILE (INCLUDES THE OVERFLOW AREA)	3000			
6. NUMBER OF SECTORS TO ALLOCATE FOR OVERFLOW WITHIN THE ABOVE AREA FOR THIS FILE SECTION				

This example illustrates that a one-section file without an overflow area is to be constructed on disc unit D02. Since the one section is mapped under zone 0 and no entries appear under the other zones, this is a one-zone pack.

The answer to Question 5 indicates the number of sectors in the entire section. In the example, 3000 sectors are allocated to the file, in zone 0.

PACK MAPPING — EXISTENT FILE WITH OR WITHOUT OVERFLOW

2. SYMBOLIC UNIT DESIGNATOR 3D01

3. DISC PACK FORMAT (0 FORMAT; 1 FORMAT; 9 FORMAT) 1

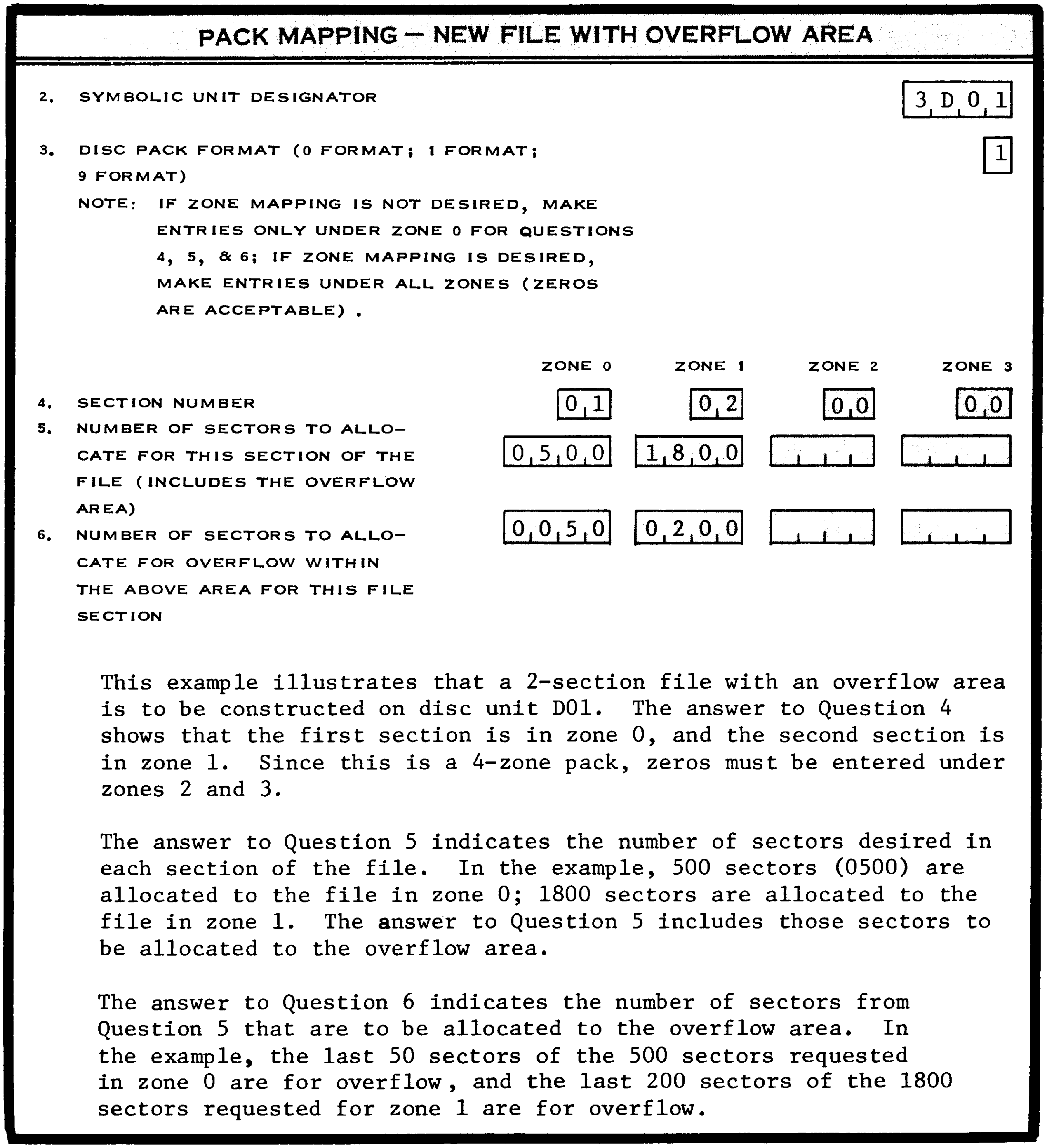
NOTE: IF ZONE MAPPING IS NOT DESIRED, MAKE ENTRIES ONLY UNDER ZONE 0 FOR QUESTIONS 4, 5, & 6; IF ZONE MAPPING IS DESIRED, MAKE ENTRIES UNDER ALL ZONES (ZEROS ARE ACCEPTABLE).

	ZONE 0	ZONE 1	ZONE 2	ZONE 3
4. SECTION NUMBER	01			
5. NUMBER OF SECTORS TO ALLOCATE FOR THIS SECTION OF THE FILE (INCLUDES THE OVERFLOW AREA)				
6. NUMBER OF SECTORS TO ALLOCATE FOR OVERFLOW WITHIN THE ABOVE AREA FOR THIS FILE SECTION				

This example illustrates that a one-section file exists on disc unit D01. The programmer need only indicate under zone 0 that this is a one-section-per-pack file.

- Examples of Pack Mapping - Four Zones

Following are three examples of pack mapping on a four-zone pack.



PACK MAPPING — NEW FILE WITH OVERFLOW AREA

2. SYMBOLIC UNIT DESIGNATOR 3D01

3. DISC PACK FORMAT (0 FORMAT; 1 FORMAT; 9 FORMAT) 1

NOTE: IF ZONE MAPPING IS NOT DESIRED, MAKE ENTRIES ONLY UNDER ZONE 0 FOR QUESTIONS 4, 5, & 6; IF ZONE MAPPING IS DESIRED, MAKE ENTRIES UNDER ALL ZONES (ZEROS ARE ACCEPTABLE).

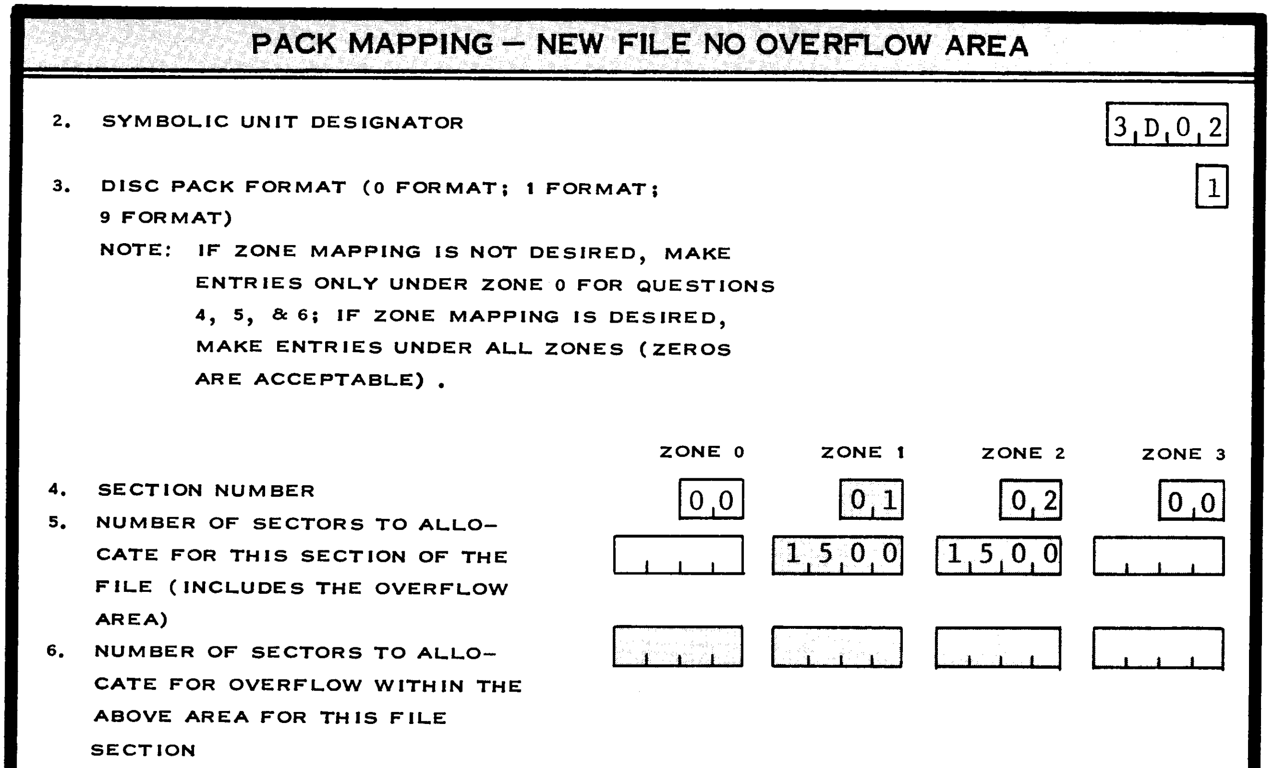
	ZONE 0	ZONE 1	ZONE 2	ZONE 3
4. SECTION NUMBER	01	02	00	00
5. NUMBER OF SECTORS TO ALLOCATE FOR THIS SECTION OF THE FILE (INCLUDES THE OVERFLOW AREA)	0500	1800		
6. NUMBER OF SECTORS TO ALLOCATE FOR OVERFLOW WITHIN THE ABOVE AREA FOR THIS FILE SECTION	0050	0200		

This example illustrates that a 2-section file with an overflow area is to be constructed on disc unit D01. The answer to Question 4 shows that the first section is in zone 0, and the second section is in zone 1. Since this is a 4-zone pack, zeros must be entered under zones 2 and 3.

The answer to Question 5 indicates the number of sectors desired in each section of the file. In the example, 500 sectors (0500) are allocated to the file in zone 0; 1800 sectors are allocated to the file in zone 1. The answer to Question 5 includes those sectors to be allocated to the overflow area.

The answer to Question 6 indicates the number of sectors from Question 5 that are to be allocated to the overflow area. In the example, the last 50 sectors of the 500 sectors requested in zone 0 are for overflow, and the last 200 sectors of the 1800 sectors requested for zone 1 are for overflow.

PACK MAPPING — NEW FILE NO OVERFLOW AREA



2. SYMBOLIC UNIT DESIGNATOR 3D02

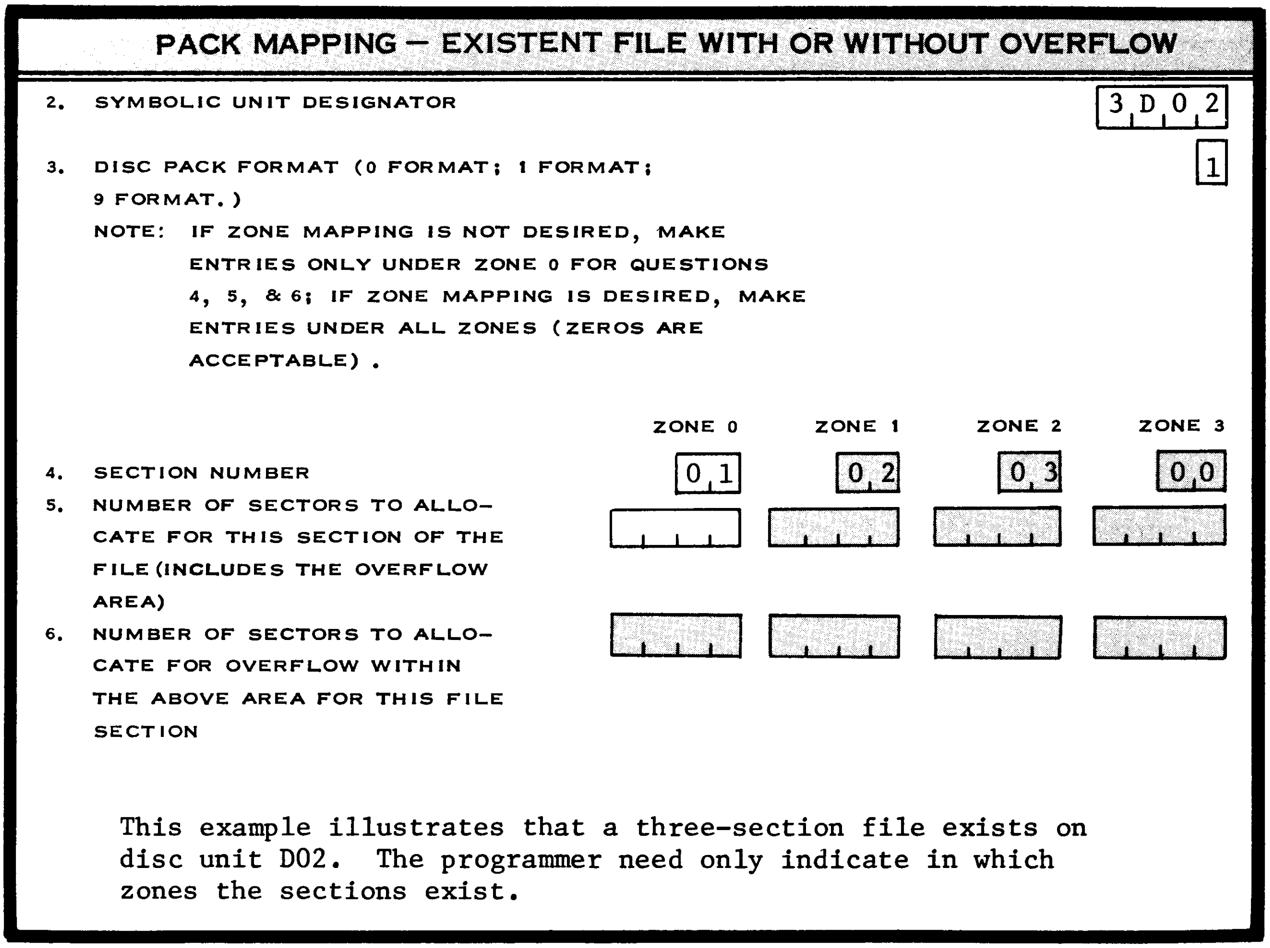
3. DISC PACK FORMAT (0 FORMAT; 1 FORMAT; 9 FORMAT) 1

NOTE: IF ZONE MAPPING IS NOT DESIRED, MAKE ENTRIES ONLY UNDER ZONE 0 FOR QUESTIONS 4, 5, & 6; IF ZONE MAPPING IS DESIRED, MAKE ENTRIES UNDER ALL ZONES (ZEROS ARE ACCEPTABLE).

	ZONE 0	ZONE 1	ZONE 2	ZONE 3
4. SECTION NUMBER	00	01	02	00
5. NUMBER OF SECTORS TO ALLOCATE FOR THIS SECTION OF THE FILE (INCLUDES THE OVERFLOW AREA)		1500	1500	
6. NUMBER OF SECTORS TO ALLOCATE FOR OVERFLOW WITHIN THE ABOVE AREA FOR THIS FILE SECTION				

This example illustrates that a 2-section file without an overflow area is to be constructed on disc unit D02. The answer to Question 4 shows that the first section is in zone 1, and the second section is in zone 2. Since this is a 4-zone pack, zeros must be entered under zone 0 and zone 3.

The answer to Question 5 indicates the number of sectors desired in each section of the file. In the example, 1500 sectors in zone 1 and 1500 sectors in zone 2 are allocated to the file.



PACK MAPPING — EXISTENT FILE WITH OR WITHOUT OVERFLOW

2. SYMBOLIC UNIT DESIGNATOR 3D02

3. DISC PACK FORMAT (0 FORMAT; 1 FORMAT; 9 FORMAT.) 1

NOTE: IF ZONE MAPPING IS NOT DESIRED, MAKE ENTRIES ONLY UNDER ZONE 0 FOR QUESTIONS 4, 5, & 6; IF ZONE MAPPING IS DESIRED, MAKE ENTRIES UNDER ALL ZONES (ZEROS ARE ACCEPTABLE) .

	ZONE 0	ZONE 1	ZONE 2	ZONE 3
4. SECTION NUMBER	01	02	03	00
5. NUMBER OF SECTORS TO ALLOCATE FOR THIS SECTION OF THE FILE (INCLUDES THE OVERFLOW AREA)				
6. NUMBER OF SECTORS TO ALLOCATE FOR OVERFLOW WITHIN THE ABOVE AREA FOR THIS FILE SECTION				

This example illustrates that a three-section file exists on disc unit D02. The programmer need only indicate in which zones the sections exist.

* * * *

7. ALTERNATE SYMBOLIC UNIT DESIGNATORS. (SEE THE DISC FILE SPEC. SHEET SECTION OF THE LANGUAGE REFERENCE MANUAL.)

D

This entry, which is for sequentially-processed files only, is always left blank for random processing. For random processing, a separate Section Control Sheet must be included which names each disc spindle (Question 2) and indicates the pack mapping for the pack on the named spindle.

If this entry is left blank and the file is sequentially processed in this run, the primary spindle (Question 2) and the alternate spindle are assumed to be the same.

If an alternate spindle is specified here, the pack mapping indicated in Questions 3, 4, 5 and 6 is applicable to the pack on the specified alternate unit. The interaction between the primary and alternate units during processing is covered in the table that follows.

NOTE

> When a file reaches end-of-section and the next section is on a different pack, that file and any associated destination files are automatically alternated (current sections closed and new sections on the new pack opened) if all specify the same alternate symbolic unit designator. It is not necessary to use the DEFALT instruction to achieve this alternation.

More than one Section Control Sheet must be used any time different mapping than that specified on the first sheet is desired, or any time the file occupies three or more packs, and more than two spindles are involved. The interaction between the primary and alternate spindles during processing is covered in the table that follows.

If multiple Section Control Sheets are used, Question 7 is answered on any one sheet. If this entry is left blank on all sheets, all further packs of the file are processed on the spindle named in Question 2 of that last sheet. If a spindle is named in Question 7 on any control sheet, further packs are processed alternately between the primary spindle (Question 2) and the alternate spindle.

Following is a table showing the relationship of Question 2 and 7 on the pack mapping of sequentially-processed files. In using this table assume the system uses three discs, the integrated discs (D01 and D02) and one spindle of a freestanding disc unit (D03).

Any time the mapping specified for a disc pack has been completed, the operator receives a message informing him of this fact. In response, he may enter 21, which indicates that the processed pack is to be changed before processing continues to the alternate pack or to the primary pack specified on the next control sheet; or he may enter EE, which indicates that the processed pack is not to be changed before processing continues to the alternate pack or to the primary pack specified on the next control sheet. The programmer must provide the operator with the appropriate instructions on his run sheet.

CONTROL SHEET	QUESTION 2	QUESTION 7	RESULT
Only One	D01	D01 (or blank)	All processing is done on the disc unit spindle designated as D01. The operator is informed when the mapping specified on the control sheet has been completed. If he enters EE, the same mapping is continued on the same pack. If he enters 21, he may change packs on D01 and then continue with the same mapping on the new pack. D01→D01→D01
First Second	D01 D01	 D01 (or blank)	All processing is done on spindle D01. The operator is informed when the mapping specified on both sheets has been completed. If he enters EE, the mapping specified on the second sheet is continued on the same pack. If he enters 21, he may change packs on D01 and then continue on the new pack with the mapping specified on the second sheet. D01→D01→D01
First Second	D01 D01	 D02	Processing begins on spindle D01. The operator is informed when the mapping specified on both sheets has been completed. If he enters EE, the mapping specified on the second sheet is immediately continued on D02 (no pack change on D01.) If he enters 21, processing continues on D02 (after the appropriate pack change on D01) with the mapping specified on the second sheet. The operator is informed when this mapping has been completed; he may enter EE or 21 (as explained previously) and continue processing on D01, with the mapping specified on the second control sheet. D01→D02→D01→Etc.

CONTROL SHEET	QUESTION 2	QUESTION 7	RESULT
Only One	D01	D02	Processing begins on spindle D01. The operator is informed when the specified mapping has been completed. If he enters EE, the same mapping is immediately continued on D02 (no pack change on D01.) If he enters 21, processing continues on D02 (after the appropriate pack change on D01) with the same mapping. The operator is informed when this mapping has been completed; he may enter EE or 21 and continue processing on D01, with the mapping specified on the second control sheet. D01→D02→D01→Etc.
First Second	D01 D02	D02	Processing begins on spindle D01. The operator is informed when the mapping specified on the first sheet has been completed. If he enters EE, no pack change is made and mapping continues. If he enters 21, processing continues on D02, after the appropriate pack change on D01, with the mapping specified on the second sheet. The operator is informed when this mapping has been completed on D02. He may then elect either to continue on the same pack or to change packs on D02. D01→D02→D02→Etc.
First Second	D01 D02	D01	Processing begins on spindle D01. The operator is informed when the mapping specified on the first sheet has been completed. If he enters EE, no pack change is made and mapping continues on D02. If he enters 21, processing continues on D02, after the appropriate pack change on D01, with the mapping specified on the second sheet. The operator is informed when this mapping has been completed. He may then remove the pack on D02 or continue on to the alternate spindle (D01) without removing the pack on D02. If more packs are associated with this file, the software looks back to D02. D01→D02→D01→Etc.

CONTROL SHEET	QUESTION 2	QUESTION 7	RESULT
First Second Third	D01 D02 D03	 D03 (or blank)	Processing begins on spindle D01. The operator is informed when the mapping specified on the first sheet has been completed. If he enters EE, no pack change is made and mapping continues on D02. If he enters 21, processing continues on D02, after the appropriate pack change on D01, with the mapping specified on the second sheet. The operator is informed when this mapping has been completed. If he enters EE, no pack change is made and mapping continues on D03. If he enters 21, processing continues on D03 after the appropriate pack change on D02. The operator is informed when this mapping has been completed. He may then elect to continue on the same pack with the same mapping, or to change packs on D03 and continue with the same mapping. D01→D02→D03→D03→Etc.
First Second Third	D01 D02 D03	 D01 or D02	This is the same as the preceding example until the operator is informed that the mapping specified on the third sheet has been completed on D03. If the operator enters 21 at this point, processing is continued on the specified alternate spindle, after the appropriate pack change on D03, with the mapping specified on the third sheet. D01→D02→D03→D02→D03→Etc.

8. MAXIMUM NUMBER OF SECTIONS MOUNTED AT ONE TIME (RANDOM PROCESSING ONLY) □□

This entry need be completed only if all sections of a randomly processed file are not mounted at the same time. For example, if a six section file is randomly processed on a system with two dual-disc units (four spindles) three sections may be mounted at one time. (Assume the fourth spindle is used for transactions.) In this case "3" must be entered in answer to this question. When the RGET instruction attempts to access section four (all transactions for the first three sections processed) the software informs the operator to mount the next three packs.

FILE ALLOCATION

MAY BE OMITTED FOR SOURCE FILES OR EXISTING SOURCE-DESTINATION FILES.

9. IS THE REQUESTED NUMBER OF SECTORS: □
   E EXACTLY THE SIZE AREA REQUIRED?
   A AN APPROXIMATION OF THE SIZE AREA REQUIRED?
   (IF BLANK, E IS ASSUMED)

Enter "E" if the number of sectors requested in answer to Question 4 is exactly the number required. Some types of processing, such as random processing, may require exact section sizes.

Enter "A" if the number of sectors requested in answer to Question 4 is approximately the number required. When "A" is entered, the software allocates the number of sectors requested ± 25%. For sequential files (as in Father-Son processing), indicating the approximate number of sectors is sufficient since further sections may be allocated as needed.

"E" is assumed if no entry is made.

10. ARE DISJOINT PIECES ACCEPTABLE IF THE NUMBER OF SECTORS CANNOT BE FOUND IN ONE PIECE? (Y OR N) (SEQUENTIAL ONLY) (IF BLANK, N IS ASSUMED) □

Enter "Y" if smaller disjointed pieces may be used should the number of sectors requested in answer to Question 4 not be available in one piece. In many instances, especially during sequential processing (Father-Son processing), the file area does not have to be one large group of sectors.

Enter "N" if the number of sectors requested in Question 4 must be in one group as in random processing.

"N" is assumed if no entry is made.

11. IF THERE IS AN UNUSED AREA AT THE END OF THIS FILE, WITHIN THE NUMBER OF SECTORS ALLOCATED, SHOULD IT BE: □

R RESERVED FOR THIS FILE?

F FREED UP FOR USE BY OTHER FILES?

(IF BLANK, R IS ASSUMED)

Enter "F" if unused sectors at the end of the file should be made available for other use.

Enter "R" if unused sectors at the end of the file should be reserved for this file. "R" should be entered for random processing.

"R" is assumed if no entry is made.

PRINTER FILES

INTRODUCTION

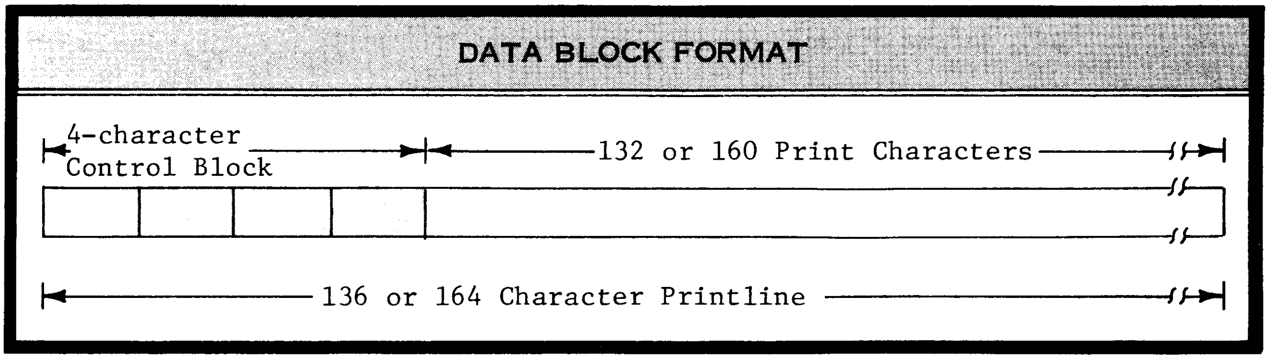
Several models of printers are available with NCR Century Systems. The printers vary in the number of print columns, and character sets in the type/line, as well as in speed of printing. The user may choose the printer or printers best suited to his needs. (Multiple printers may be used with a single system if desired.)

The I/O software treats the printer as a file device, thereby easing the programmer's coding task. After naming and defining the file on a printer file specification sheet and the record on data layout sheets, the programmer formats the printline in the output buffer or work area, moves a 4-character printer control block into the printline, and uses the PUT instruction to output the printline to the printer.

FILE DESCRIPTION

Data Blocks

Each data block consists of one complete record, which is composed of a printer control block (4 characters) and the actual data (132 or 160 characters, depending upon the printer model); therefore, all blocks are either 136 or 164 characters long.

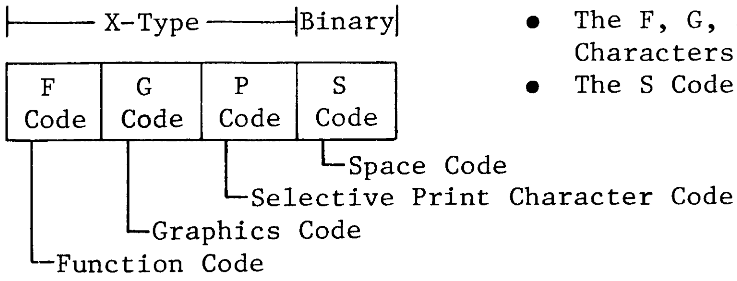

DATA BLOCK FORMAT

4-character Control Block
132 or 160 Print Characters
136 or 164 Character Printline

- Printer Control Block

  The portion of the data block used to control the operation of the printer is called the printer control block. It consists of four characters: three X-type characters and one binary type character.

CONTROL BLOCK FORMAT



- The F, G, and P Codes are X-Type Characters.
- The S Code is a binary character.

The control block is defined on data layout sheets as an area and the control block characters are input to the compiler as constants. The programmer decides which control blocks are needed, codes them, and then uses MOVE instructions to place the proper block in each printline as that line is built in the output buffer or work area. Therefore each control block must be given a unique reference name.

- F Code - Function Code

F	G	P	S
Code	Code	Code	Code

The F Code position of the printer control block informs the printer whether printing, or spacing, or both is desired. One of the following characters must be entered in the F Code position:

- P - Print the data portion of the record after spacing the paper the number of lines specified in the S Code.
- L - Print the data portion of the record after spacing the paper to the line number specified in the S Code.
- N - Do not print, but space the paper the number of lines specified in the S Code. When F is N, S may not be 0.
- E - Eject the paper to the top of the next page. The S Code must be zero.

- G Code - Graphics Code

F	G	P	S
Code	Code	Code	Code

The G Code defines the set of characters needed to print the desired report. In choosing a character set, the programmer should consider that a smaller set allows greater printing speed. He should not use special or extra characters unless absolutely needed. Different G Codes may be used in different control blocks that concern specific portions of a report. For example, a set of numeric characters might be selected in those control blocks that are used in the body of a report, and an alphanumeric set might be selected for the header lines in the same report. One of the following X-type characters must be entered in the G code position.

N	B				A				E				
NUMERIC	BASIC ALPHANUMERIC				ALPHANUMERIC				EXTENDED ALPHANUMERIC				
0	0	A	N	$	0	A	N	$	0	A	N	$	"
1	1	B	O	£	1	B	O	£	1	B	O	£	%
2	2	C	P	*	2	C	P	*	2	C	P	*	&
3	3	D	Q		3	D	Q	/	3	D	Q	/	;
4	4	E	R		4	E	R	+	4	E	R	+	?
5	5	F	S		5	F	S	:	5	F	S	:	@
6	6	G	T		6	G	T	=	6	G	T	=	[
7	7	H	U		7	H	U	'	7	H	U	'	\
8	8	I	V		8	I	V	(	8	I	V	(	]
9	9	J	W		9	J	W	)	9	J	W	)	←
.	.	K	X		.	K	X	<	.	K	X	<	↑
,	,	L	Y		,	L	Y	>	,	L	Y	>	#
-	-	M	Z		-	M	Z		-	M	Z	!	

- N - Numeric set of 13 characters.
- B - Basic alphanumeric set of 42 characters.
- A - Alphanumeric set of 51 characters.
- E - Extended alphanumeric set of 64 characters.
- U - Upper/lower character set of 90 characters.

NOTE

Normally, the # symbol (available only in the E set) is printed by outputting a binary zero (00000000), and the £ symbol (available in the B, A and E sets) is printed by outputting a binary 35 (01000011). By use of a special hardware code disc, however, these can be reversed so that the # symbol is printed by outputting a binary 35 (01000011), and the £ symbol is printed by outputting a binary zero (00000000).

The space character is not included in the character set count because the printer automatically leaves spaces in the printline where they are indicated by the program.

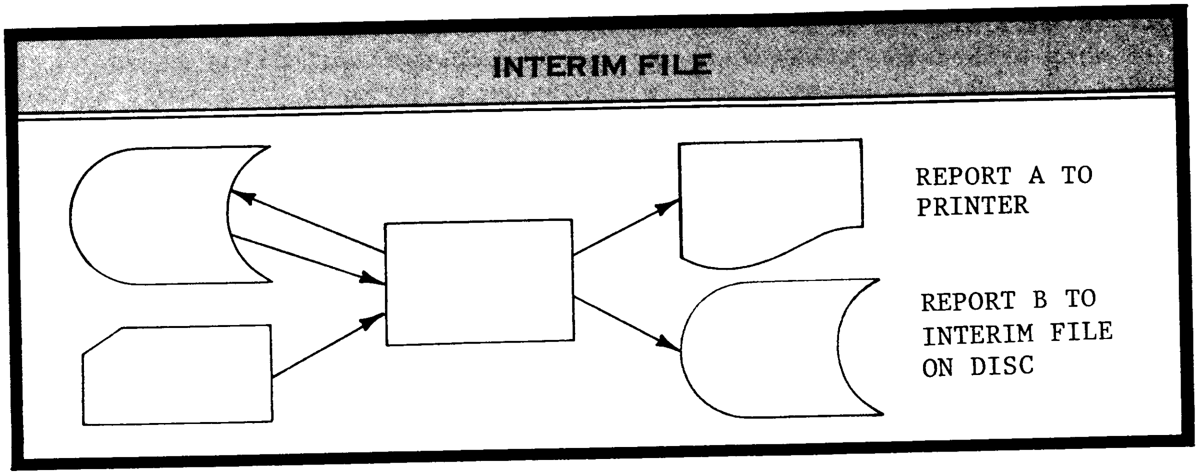
The 640-102 printer is available with a double numeric character set for greater speed. This printer does not have the E set available when equipped with double numerics.

The G code is operative only on the NCR Century 100 integrated printer. All other printers print all characters contained in a printline. Therefore, the G code must not be used to block unwanted characters out of a printline in an NCR Century 100 program which may at a later date be run on a larger system.

- P Code - Selective Print Character Code

F Code	G Code	P Code	S Code

The selective print character distinguishes this report from any other if this report is output to an interim file on magnetic disc or magnetic tape. Interim files are used as temporary storage for one or more reports when multiple reports are generated in the same run and the system has only one printer. Interim files are discussed in detail later in this section. Any X-type character may be used as the P code.



- S Code - Space Code

F Code	G Code	P Code	S Code

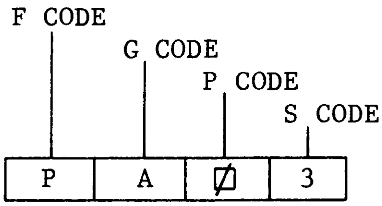
The S Code is a binary character that specifies either the number of lines to space or the actual line number upon which printing is to occur.

- S Code - number of lines to space if F Code is P or N.
- S Code - actual line number if F Code is L.
- S Code - must be zero if F Code is E.

- Printer Control Block Definition

The programmer defines, on data layout sheets, the necessary printer control blocks as constants. He assigns a unique reference name to each printer control block so that it may be referenced in a MOVE instruction. Below are some examples:

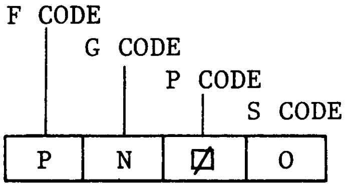
SPACE AND PRINT



Space three lines and print, using the alphanumeric set. This control block has the reference SPACEPRINT for use with the MOVE instruction.

	REFERENCE	CODE	LOCATION	LENGTH	DP	TYPE	VALUE OR PICTURE
7	8 9 10 11 12 13 14 15 16 17	18	19 20 21 22 23	24 25 26 27	28 29	30	31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
D	SPACEPRINT	A		4			
D	FGPCODES	F	0	3		X	PAØ
D	SCODE	F	3	1		B	3
D							

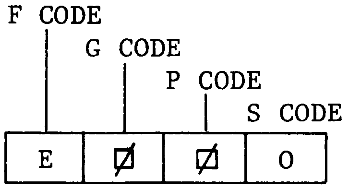
PRINT WITHOUT SPACING



Space no lines and print, using the numeric set. This control block has the reference OVERPRINT for use with the MOVE instruction.

	REFERENCE	CODE	LOCATION	LENGTH	DP	TYPE	VALUE OR PICTURE
7	8 9 10 11 12 13 14 15 16 17	18	19 20 21 22 23	24 25 26 27	28 29	30	31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
D	OVERPRINT	A		4			
D	FGPCODES	F	0	3		X	PNØ
D	SCODE	F	3	1		B	0

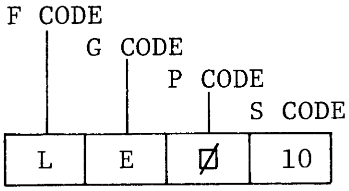
EJECT A PAGE



Eject a page. This exact format must be used to move the paper to the top of the next page. This control block has the reference EJECTPAGE for use with the MOVE instruction.

	REFERENCE	CODE	LOCATION	LENGTH	DP	TYPE	VALUE OR PICTURE
7	8 9 10 11 12 13 14 15 16 17	18	19 20 21 22 23	24 25 26 27	28 29	30	31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
D	EJECTPAGE	A		4			
D		F	0	3		X	EØØ
D		F	3	1		B	0

PRINT ON A SPECIFIC LINE



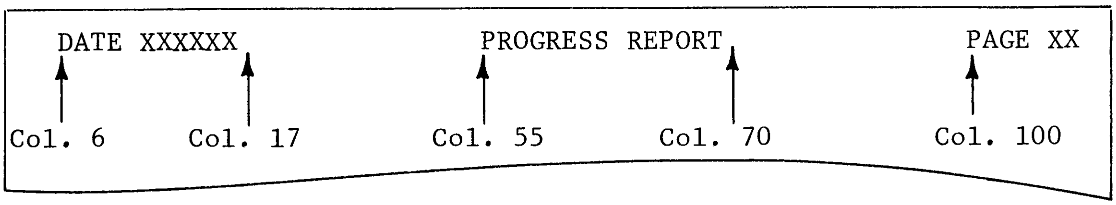
Space to line 10 and print, using the extended alphanumeric set. The line number specified in the S Code must be greater than the current line number or printing will not occur on the desired line. Thus, the programmer may not specify a line on the next page. This block has the reference TOPPAGE for use with the MOVE instruction.

	REFERENCE	CODE	LOCATION	LENGTH	DP	TYPE	VALUE OR PICTURE
7	8 9 10 11 12 13 14 15 16 17	18	19 20 21 22 23	24 25 26 27	28 29	30	31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
D	TOPPAGE	A		4			
D		F	0	3		X	LEØ
D		F	3	1		B	10

- Header Line Definition

The programmer generally desires to print header lines on the top of each page to identify the report and columns in the report. These header lines are normally stored as constants in memory; they must, therefore, be defined on data layout sheets.

HEADER LINE DEFINITION



	REFERENCE	CODE	LOCATION	LENGTH	DP	TYPE	VALUE OR PICTURE
7	8 9 10 11 12 13 14 15 16 17	18	19 20 21 22 23	24 25 26 27	28 29	30	31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
D	HEADER1	A		136			
D	CNTBLKFGP	F	0	3		X	PAØ
D	SCODE	F	3	1		B	3
D		F	4	5		S	
D	WORDDATE	F	9	5		X	DATEØ
D	INSRTDATE	F	14	6		X	
D		F	20	38		S	
D	TITLE	F	58	15		X	PROGRESSØREPORT
D		F	73	30		S	
D	WORDPAGE	F	103	5		X	PAGEØ
D	INSERTPAGE	F	108	2		Z	
D		F	110	26		S	

- Printline Definition

Normally, multiple buffers are assigned to a print file and printlines are constructed in a workarea. However, the programmer may assign single or multiple buffers and construct the printline directly in the buffer.

When multiple buffers are used and printlines are constructed directly in the buffer, the buffer must be cleared immediately before printline construction begins. If this is not done, unwanted characters from a previously constructed printline could be printed. The programmer may use a literal MOVE instruction to clear a buffer (MOVE '⍁',PRNTRECORD).

Normally printlines are constructed in a workarea and then moved to the buffer. In this case, upon moving the contents of a workarea, the buffer is filled completely thereby eliminating all unwanted characters. Building printlines in a workarea affords the programmer more flexibility because, unlike the buffer (when multiple buffers are used), the workarea need not be cleared before each printline is constructed. Fields from a previous printline, common to the new printline, are not destroyed and, therefore, need not be reconstructed for the new printline. The programmer only clears the workarea before changes in printline format, for example, before constructing total lines after detail lines and again before building additional detail lines. The workarea may be cleared with a MOVE instruction, in the same manner a buffer (MOVE '⍁',PRNTAREA).

When printlines with more than one format are needed for a single report, the programmer normally uses one workarea and redefines that area (by using SAME in the location column) for each format. However, he may use a different workarea for each separate format. For a complete description of using SAME in the location column of a data layout sheet to redefine an area, see the NEAT/3 REFERENCE MANUAL, INTRODUCTION AND DATA, tab 3, "Data Layout Sheets."

The programmer may desire to use a single buffer and no workarea if relatively few printlines are output or if the program is large and memory space is critical. The rules for using a single buffer and for using a work area are the same.

X	REFERENCE	X CODE	LOCATION	X LENGTH	X DP	X TYPE	VALUE OR PICTURE
7	8 9 10 11 12 13 14 15 16 17	18	19 20 21 22 23	24 25 26 27	28 29	30	31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
D	PRNTLINE1	R		136			
D	PRCONTROL	F		4		X	
D	PRNTFIELD1	F	4	14		X	
D							
D							

- The printer control block must be defined as the first field and must be X-type.

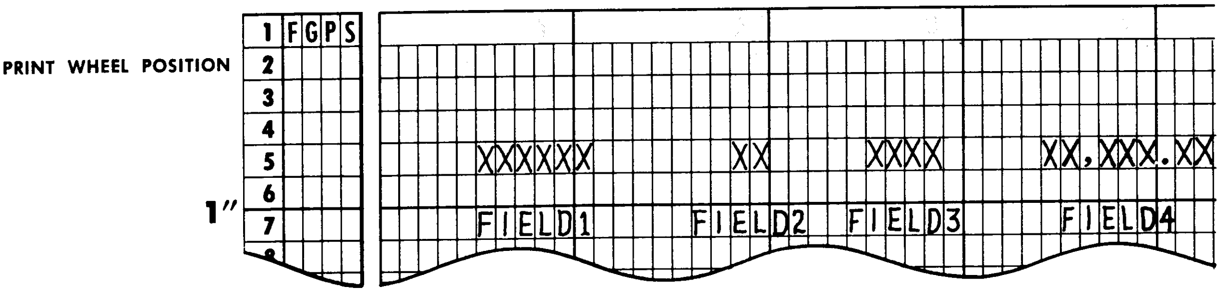
- The first actual print position is relative position 4.

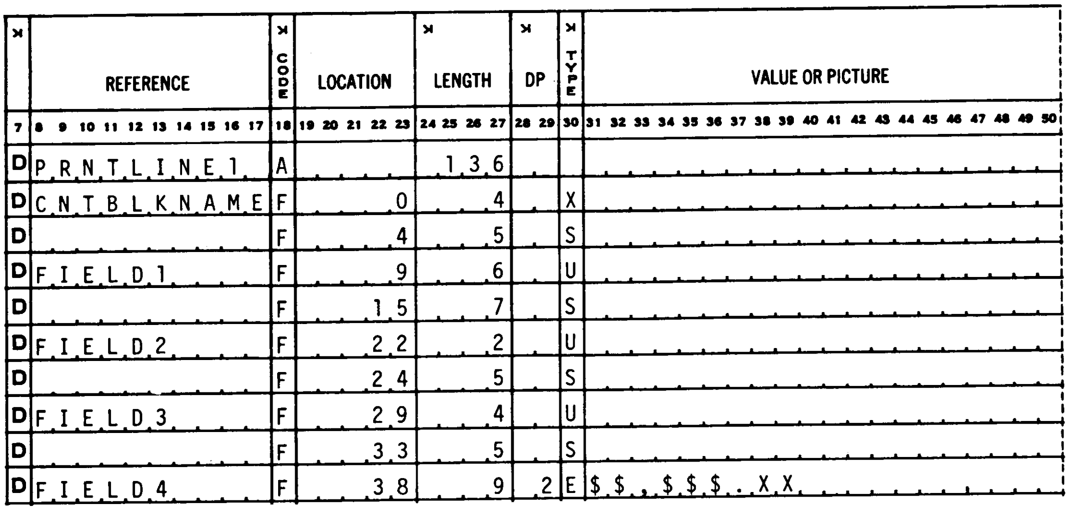
In any case, the portion of the record reserved for the printer control block must be the first field in the record and must be X-type. Therefore, the first data field must not start before relative position four, which corresponds to the first print position on the typeline.

A form design worksheet aids the programmer in formatting the layout of his report. Each print-wheel position on this sheet corresponds to a print column on the printer. Since column 1 on both the printer and sheet is equal to relative position 4 (positions 0-3 are for the control block), the programmer can easily convert from print-wheel positions to relative positions by adding 3 to the print-wheel positions (relative position in record definition = print-wheel position + 3).

USING THE FORM DESIGN WORKSHEET

FORM DESIGN WORKSHEET





X	REFERENCE	CODE	LOCATION	LENGTH	DP	TYPE	VALUE OR PICTURE
7	8 9 10 11 12 13 14 15 16 17	18	19 20 21 22 23	24 25 26 27	28 29	30	31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
D	PRNTLINE1	A		136			
D	CNTBLKNAME	F	0	4		X	
D		F	4	5		S	
D	FIELD1	F	9	6		U	
D		F	15	7		S	
D	FIELD2	F	22	2		U	
D		F	24	5		S	
D	FIELD3	F	29	4		U	
D		F	33	5		S	
D	FIELD4	F	38	9	2	E	$$,$$$.XX

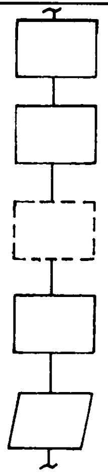
- Each field is defined on the data layout sheet as it was positioned on the form design worksheet.

Printline Construction

Once the workarea and buffer have been defined, the programmer constructs printlines by using MOVE instructions to place the desired control block and data in the desired fields within the printline. If the printline is constructed directly in the buffer, it may then be output with a simple PUT instruction (PUT PRNTFILE).

If the printline is constructed in a workarea, the contents of the area may be placed in the buffer either by using the MOVE instruction (MOVE PRNTAREA, PRNTRECORD) followed by a PUT instruction (PUT PRNTFILE) or by using the PUT instruction with the work area option (PUT PRNTFILE,PRNTAREA).

CONSTRUCTING A PRINT RECORD



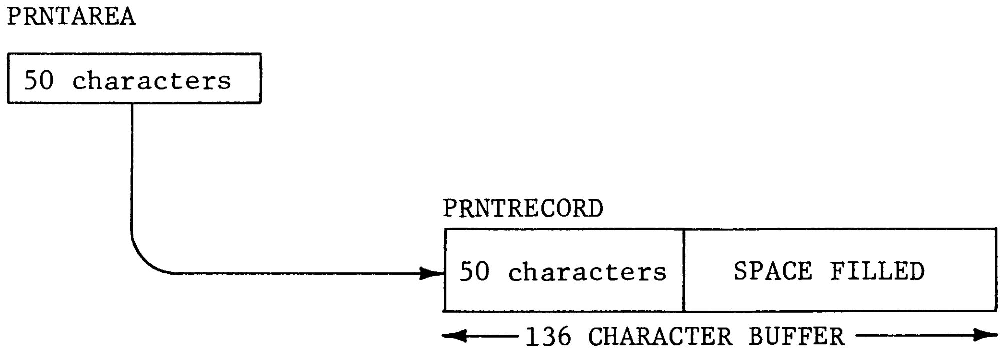
MOVE '⍁',PRNTAREA	Space out the workarea.
MOVE CNTBLK1,PRCONTROL	Place the desired control block into the workarea.
MOVE (data fields), (prntarea fields)	Place the desired data into the workarea.
MOVE PRNTAREA,PRNTRECORD	Place the contents of the workarea into the printer buffer.
PUT PRNTFILE	Print the record currently in the buffer.

- Using the MOVE Instruction

  If the MOVE instruction is used to place the contents of a workarea into the buffer, it will left-justify the data in the buffer and space fill to the right any unused portion of the buffer. This is extremely useful when the workarea is shorter than the buffer, as it would be if a narrow form were used.

  The MOVE instruction may also be used to place previously defined header lines into the buffer (MOVE HEADER1,PRNTRECORD). MOVE should always be used when the area or header to be printed is shorter in length than the buffer.

USING MOVE



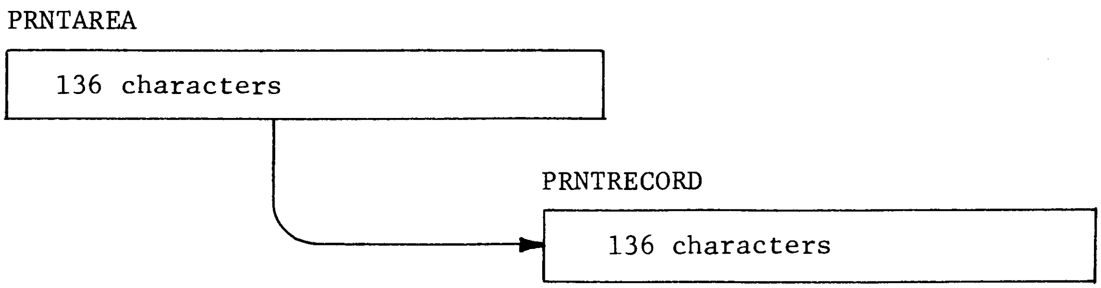
```
MOVE  PRNTAREA, PRNTRECORD
```

The MOVE instruction left-justifies the contents of a 50-character area called PRNTAREA in the 136-character buffer (PRNTRECORD) and space fills 86 characters to the right.

- Using the PUT Instruction with Workarea Option

  If the PUT instruction with a workarea option is used, it will place the contents of the workarea in the buffer, but it will not space fill to the right any unused portion of the buffer. Therefore, the workarea must be the same size as the buffer.

USING PUT WITH WORKAREA



```
PUT  PRNTFILE, PRNTAREA
```

The PUT instruction with workarea option inserts, left-justified,the contents of 136-character area called PRNTAREA in the 136-character buffer (PRNTRECORD).

The PUT instruction with the workarea option may also be used to place previously defined header lines into the buffer when the header line areas are equal in length to the buffer (PUT PRINTFILE,HEADER1).

Vertical Format Control

Normally, the programmer specifies on the printer file specification sheet that vertical format control is desired. Vertical format control is a software routine that converts the printer control block into a format that is acceptable to the printer. It also keeps track of the line number on which printing occurs and causes a branch to the programmer's end-of-page routine if desired.

Vertical format control is not specified when a utility routine is used to print an interim file.

End-of-Page Routine

At his option, the programmer may enter the reference name of an end-of-page routine on the printer file specification sheet. He must enter a last-data line number, that is, a line number to signal termination of printing on a page.

When a printline is output that causes the paper to space either to the actual last data line or to cross the last data line, the software checks the F code of the control block.

- If the printline contains a control block with an F code of P or N, the software prints the line, stores a link and transfers control to the programmer's end-of-page routine.

- If the printline contains a control block with an F code of L or E, the software does not branch to the end-of-page routine.

This arrangement permits the programmer to pass the last data line for special printing on certain pages if desired.

CONTROL BLOCK EFFECT ON END—OF—PAGE ROUTINE

CONTROL BLOCK	RESULT OF ENCOUNTERING LAST DATA LINE
P A Ø 2 N A Ø 2	A link is stored and control is given to the programmer's end-of-page routine.
L A Ø 62 E Ø Ø 0	A link is not stored and control is not given to the programmer's end-of-page routine.

The programmer normally uses the end-of-page routine to accomplish the following:

- Print totals at the bottom of the page.

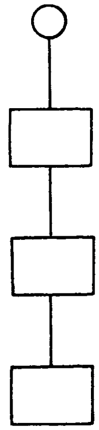
- Eject the paper to the top of the next page and then position it to a desired line.

- Print headers at the top of the new page.

Since headers must be printed on the top of the first page, and totals may be desired at bottom of a partially full last page, and both of these are normally desired in the end-of-page routine, the following procedure is suggested for handling the end-of-page routine.

END-OF-PAGE ROUTINE

ENDOFPAGE



Enter here if end-of-page and proper control block.

LINK BOTTOMPAGE - Go to a routine called BOTTOMPAGE to print totals and eject the page; then, RELINK here.

LINK TOPPAGE - Go to a routine called TOPPAGE to properly space paper and print headers; then, RELINK here.

RELINK - Go back to main program. Software stored a link before linking to end-of-page.

The above technique allows the programmer complete flexibility. For example, he may link to TOPPAGE at the beginning of his program (initiate time) to space the paper properly and to print headers on the first page. Since TOPPAGE ends with a RELINK instruction, control will be returned to the initiate portion of the program when the TOPPAGE routine is complete.

The use of a control block with an F code of L and the use of a LINK instruction to BOTTOMPAGE allows the programmer to space the paper past the last data line (in his end-of-program routine) and print totals at the bottom of the last page. Since the BOTTOMPAGE routine ends in a RELINK, control will return to the portion of the program responsible for end-of-program procedures.

If an end-of-page routine is not specified, the paper is automatically ejected to the top of the next page (by a software end-of-page routine) when the last data line is encountered and the rules governing F Codes are followed.

* * * *

SAMPLE PRINT PROGRAM

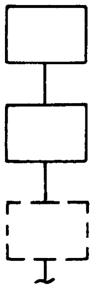
The following illustrates a sample program to print a report. Only that portion of the program needed to convey the overall picture is illustrated. Examples of the report format, flowchart, printer file specification sheet, and data definitions are included.

Report Format

line 1		
line 3	Title Line One	(Alpha)
line 5	Title Line Two	(Alpha)
line 8	Detail Line (first)	
line 10	Detail Line	
.	.	
.	.	
.	.	
line 56	Detail Line (last)	
line 60	Totals	(Numeric)

Flowchart

INITIATE
ROUTINE



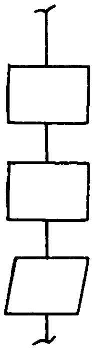
MOVE VIRTUAL DATE TO HEADER1.

LINK TO TOPPAGE.

OTHER INITIATE LOGIC.

NOTE: The actual or virtual date may be moved from its software-reserved location in memory into header lines by using the MOVE instruction and the proper reserved reference name (MOVE >EXEC.ACDATE, fieldname for the actual date or, MOVE >EXEC.VRDATE, fieldname for the virtual date).

GENERAL
PRINT
ROUTINE

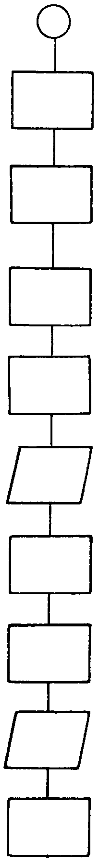


MOVE THE FIELDS TO BE PRINTED INTO PRNTAREA.

MOVE THE CONTENTS OF PRNTAREA INTO PRNTRECORD.

PUT PRNTFILE

BOTTOMPAGE



SPACE OUT PRNTAREA.

MOVE THE PRINTER CONTROL BLOCK FOR PRINTING TOTALS ON LINE 60 $\boxed{L}\boxed{N}\boxed{\not b}\boxed{60}$ INTO TOTALINE. (TOTALINE IS PRNTAREA REDEFINED.)

MOVE TOTALS TO TOTALINE.

MOVE THE CONTENTS OF TOTALINE INTO PRNTRECORD.

PUT PRNTFILE.

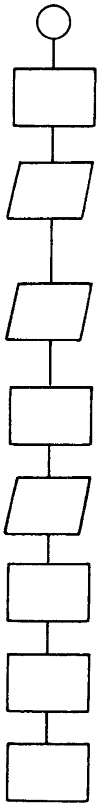
CLEAR TOTALS.

MOVE CONTROL BLOCK TO EJECT THE PAGE $\boxed{E}\boxed{\not b}\boxed{\not b}\boxed{0}$ TO PRNTRECORD.

PUT PRNTFILE.

RELINK

TOPPAGE



SET UP PAGE NUMBER IN HEADER1.

PUT PRNTFILE, HEADER1. PRINT THE FIRST HEADER, WHICH CONTAINS CONTROL BLOCK $\boxed{L}\boxed{B}\boxed{\not b}\boxed{3}$ ON LINE 3.

PUT PRNTFILE, HEADER2. PRINT THE SECOND HEADER, WHICH CONTAINS CONTROL BLOCK $\boxed{P}\boxed{B}\boxed{\not b}\boxed{2}$ ON LINE 5.

MOVE CONTROL BLOCK FOR SPACING THE PAPER 1 LINE $\boxed{N}\boxed{\not b}\boxed{\not b}\boxed{1}$ TO PRNTRECORD.

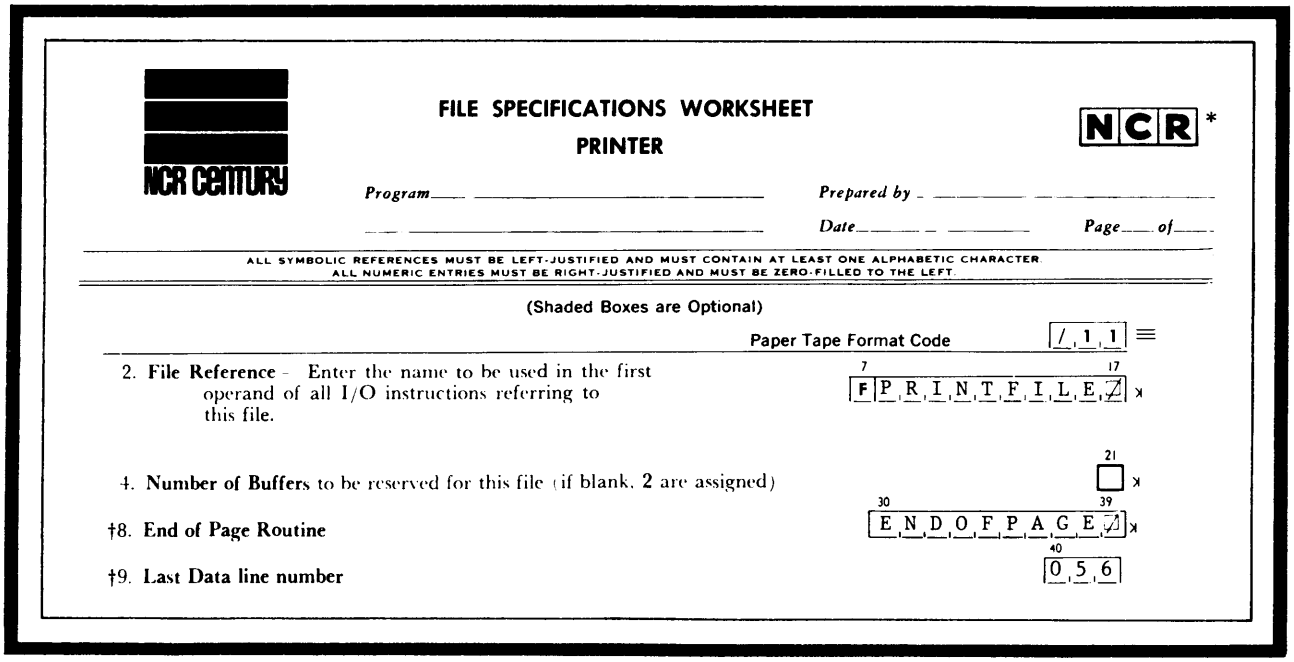
PUT PRNTFILE. (SPACE THE PAPER BUT DO NOT PRINT.)

SPACE OUT PRNTAREA.

MOVE CONTROL BLOCK FOR DOUBLE SPACING DETAIL LINES TO PRNTAREA $\boxed{P}\boxed{N}\boxed{\not b}\boxed{2}$.

RELINK.

Printer File Specification Sheet



NCR CENTURY

FILE SPECIFICATIONS WORKSHEET
PRINTER

NCR*

Program ______________________ *Prepared by* ______________________
______________________ *Date* __________ *Page* ___ *of* ___

ALL SYMBOLIC REFERENCES MUST BE LEFT-JUSTIFIED AND MUST CONTAIN AT LEAST ONE ALPHABETIC CHARACTER.
ALL NUMERIC ENTRIES MUST BE RIGHT-JUSTIFIED AND MUST BE ZERO-FILLED TO THE LEFT.

(Shaded Boxes are Optional)

Paper Tape Format Code [/ 1 1] ≡

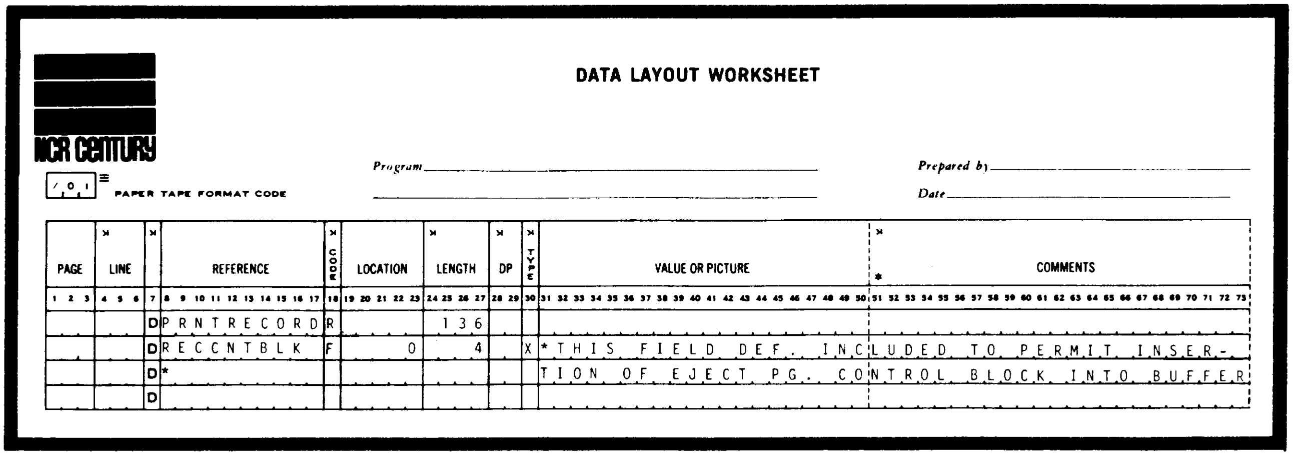
2. **File Reference** – Enter the name to be used in the first operand of all I/O instructions referring to this file. — 7 [F][P R I N T F I L E ⍁] 17 ⋊

4. **Number of Buffers** to be reserved for this file (if blank, 2 are assigned) — 21 □ ⋊

†8. **End of Page Routine** — 30 [E N D O F P A G E ⍁] 39 ⋊

†9. **Last Data line number** — 40 [0 5 6]

Record Definitions



NCR CENTURY

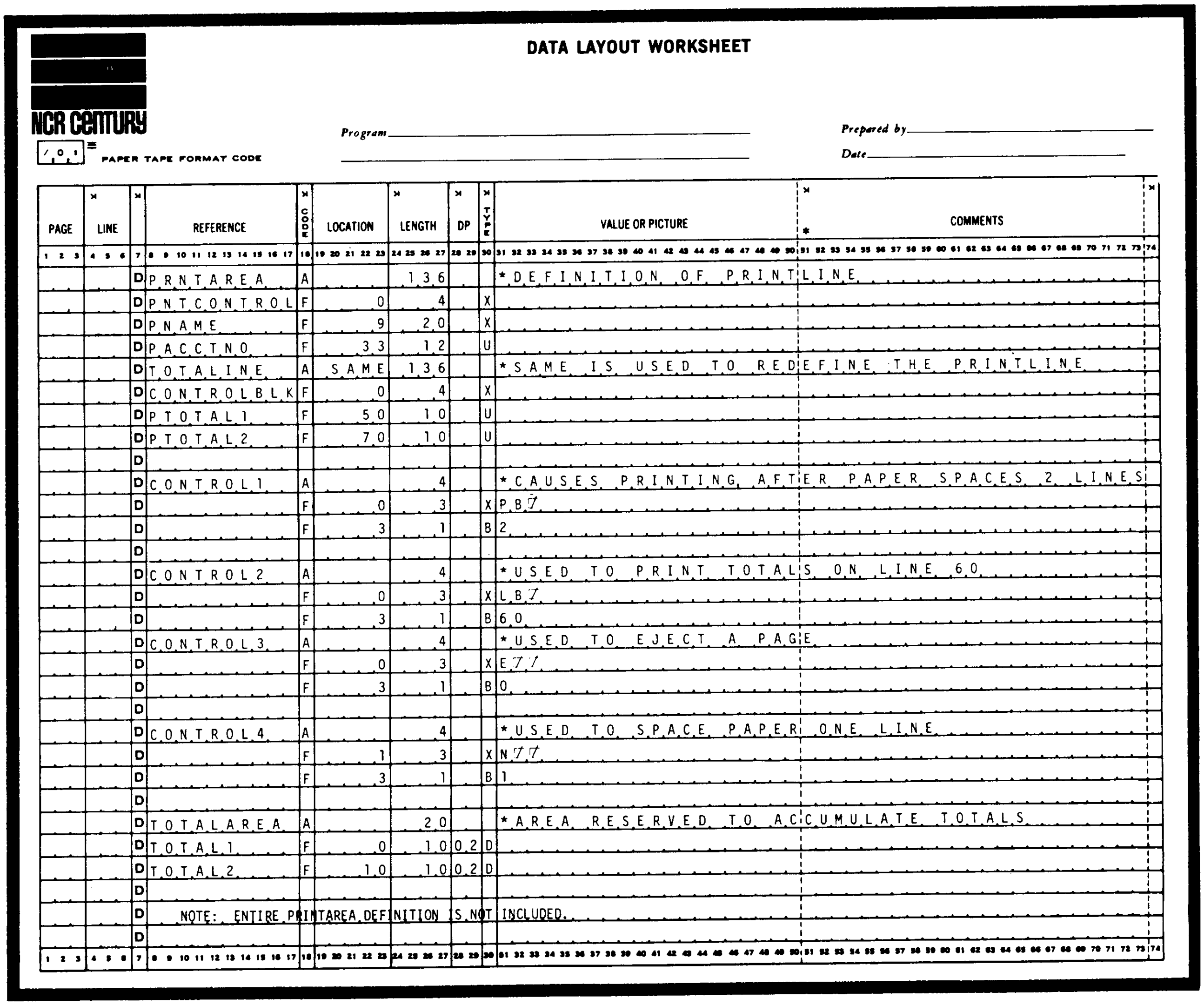
DATA LAYOUT WORKSHEET

Program ______________________ *Prepared by* ______________________
______________________ *Date* ______________________

[/ 0 1] ≡ PAPER TAPE FORMAT CODE

PAGE	⋊ LINE	⋊	REFERENCE	⋊ CODE	LOCATION	⋊ LENGTH	⋊ DP	⋊ TYPE	VALUE OR PICTURE	⋊ * COMMENTS
1 2 3	4 5 6	7	8 9 10 11 12 13 14 15 16 17	18	19 20 21 22 23	24 25 26 27	28 29	30	31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50	51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73
		D	PRNTRECORD	R		136				
		D	RECCNTBLK	F	0	4		X	*THIS FIELD DEF. INC	LUDED TO PERMIT INSER-
		D	*						TION OF EJECT PG. CO	NTROL BLOCK INTO BUFFER
		D								

Area Definitions



NCR CENTURY

/ 0 1 PAPER TAPE FORMAT CODE

DATA LAYOUT WORKSHEET

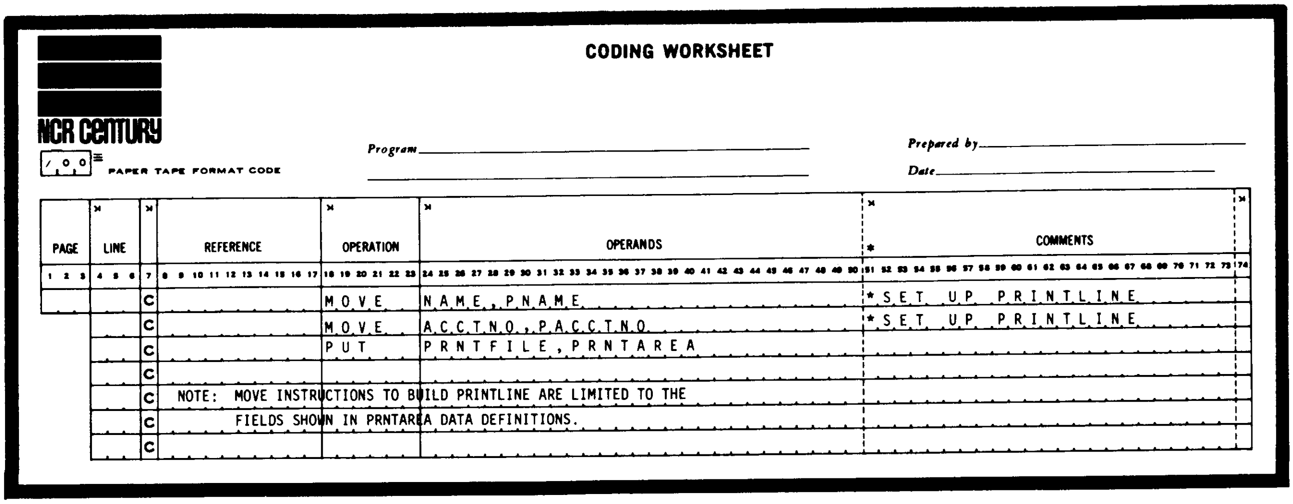
Program ______________________

Prepared by ______________________

Date ______________________

PAGE	LINE		REFERENCE	CODE	LOCATION	LENGTH	DP	TYPE	VALUE OR PICTURE	* COMMENTS
1 2 3	4 5 6	7	8 9 10 11 12 13 14 15 16 17	18	19 20 21 22 23	24 25 26 27	28 29	30	31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50	51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74
		D	PRNTAREA	A		136			*DEFINITION OF PRINT	LINE
		D	PNTCONTROL	F	0	4		X		
		D	PNAME	F	9	20		X		
		D	PACCTNO	F	33	12		U		
		D	TOTALINE	A	SAME	136			*SAME IS USED TO RED	EFINE THE PRINTLINE
		D	CONTROLBLK	F	0	4		X		
		D	PTOTAL1	F	50	10		U		
		D	PTOTAL2	F	70	10		U		
		D								
		D	CONTROL1	A		4			*CAUSES PRINTING AFT	ER PAPER SPACES 2 LINES
		D		F	0	3		X	PB7	
		D		F	3	1		B	2	
		D								
		D	CONTROL2	A		4			*USED TO PRINT TOTAL	S ON LINE 60
		D		F	0	3		X	LB7	
		D		F	3	1		B	60	
		D	CONTROL3	A		4			*USED TO EJECT A PAG	E
		D		F	0	3		X	E77	
		D		F	3	1		B	0	
		D								
		D	CONTROL4	A		4			*USED TO SPACE PAPER	ONE LINE
		D		F	1	3		X	N77	
		D		F	3	1		B	1	
		D								
		D	TOTALAREA	A		20			*AREA RESERVED TO AC	CUMULATE TOTALS
		D	TOTAL1	F	0	10	02	D		
		D	TOTAL2	F	10	10	02	D		
		D								
		D	NOTE: ENTIRE PRINTAREA DEFINITION IS NOT INCLUDED.							
		D								
1 2 3	4 5 6	7	8 9 10 11 12 13 14 15 16 17	18	19 20 21 22 23	24 25 26 27	28 29	30	31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50	51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74

General Print Routine Sample Coding



NCR CENTURY

/ 0 0 PAPER TAPE FORMAT CODE

CODING WORKSHEET

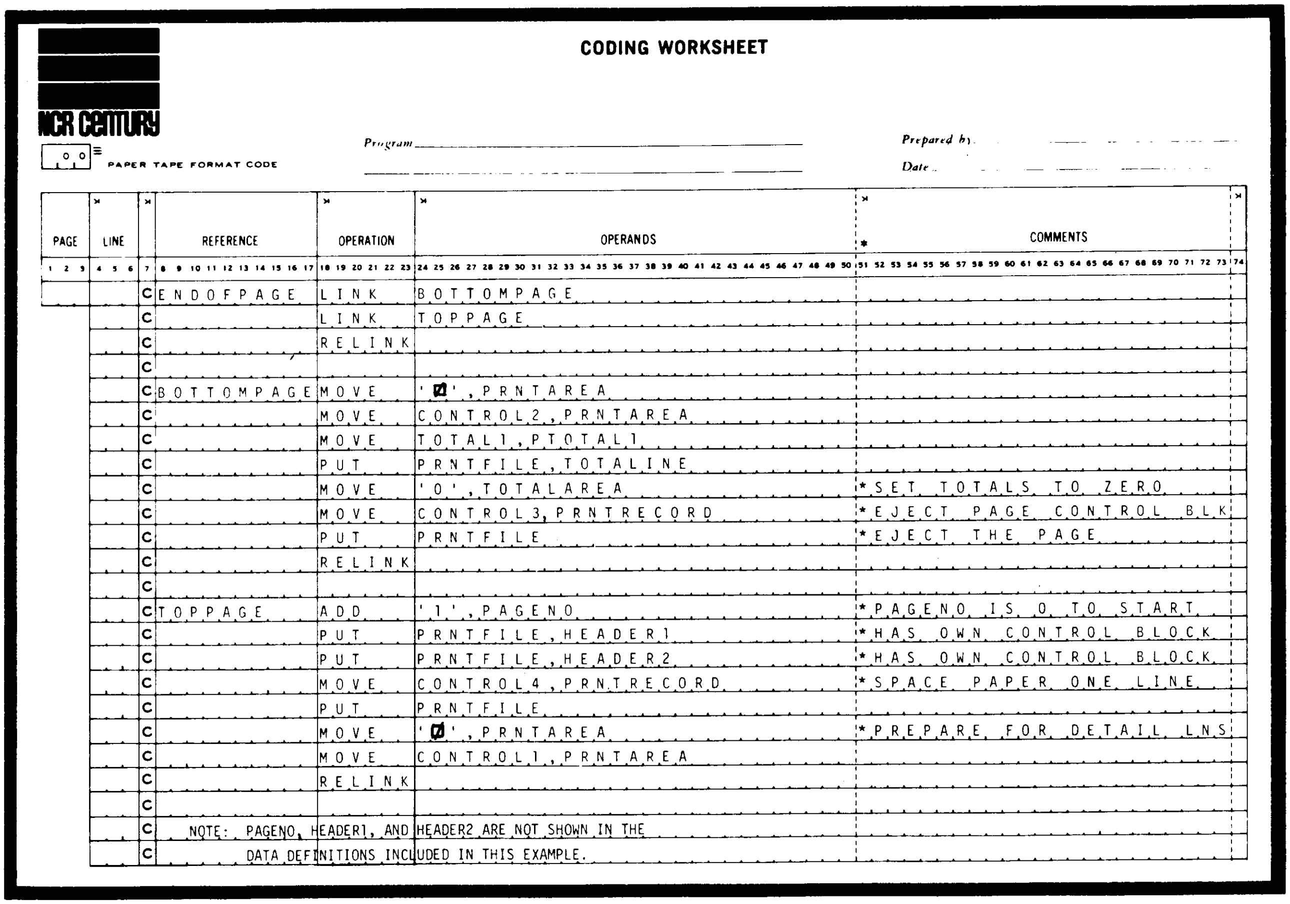
Program ______________________

Prepared by ______________________

Date ______________________

PAGE	LINE		REFERENCE	OPERATION	OPERANDS	* COMMENTS
1 2 3	4 5 6	7	8 9 10 11 12 13 14 15 16 17	18 19 20 21 22 23	24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50	51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74
		C		MOVE	NAME,PNAME	*SET UP PRINTLINE
		C		MOVE	ACCTNO,PACCTNO	*SET UP PRINTLINE
		C		PUT	PRNTFILE,PRNTAREA	
		C				
		C	NOTE: MOVE INSTRUCTIONS TO BUILD PRINTLINE ARE LIMITED TO THE			
		C	FIELDS SHOWN IN PRNTAREA DATA DEFINITIONS.			
		C				

End-Of-Page Routine Sample Coding



NCR CENTURY

CODING WORKSHEET

PAPER TAPE FORMAT CODE

Program ______________________ Prepared by ______________________

Date ______________________

PAGE	LINE		REFERENCE	OPERATION	OPERANDS	COMMENTS
1 2 3	4 5 6	7	8 9 10 11 12 13 14 15 16 17	18 19 20 21 22 23	24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50	51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74
		C	ENDOFPAGE	LINK	BOTTOMPAGE	
		C		LINK	TOPPAGE	
		C		RELINK		
		C				
		C	BOTTOMPAGE	MOVE	'Ø',PRNTAREA	
		C		MOVE	CONTROL2,PRNTAREA	
		C		MOVE	TOTAL1,PTOTAL1	
		C		PUT	PRNTFILE,TOTALINE	
		C		MOVE	'0',TOTALAREA	*SET TOTALS TO ZERO
		C		MOVE	CONTROL3,PRNTRECORD	*EJECT PAGE CONTROL BLK
		C		PUT	PRNTFILE	*EJECT THE PAGE
		C		RELINK		
		C				
		C	TOPPAGE	ADD	'1',PAGENO	*PAGENO IS 0 TO START
		C		PUT	PRNTFILE,HEADER1	*HAS OWN CONTROL BLOCK
		C		PUT	PRNTFILE,HEADER2	*HAS OWN CONTROL BLOCK
		C		MOVE	CONTROL4,PRNTRECORD	*SPACE PAPER ONE LINE
		C		PUT	PRNTFILE	
		C		MOVE	'Ø',PRNTAREA	*PREPARE FOR DETAIL LNS
		C		MOVE	CONTROL1,PRNTAREA	
		C		RELINK		
		C				
		C	NOTE: PAGENO, HEADER1, AND HEADER2 ARE NOT SHOWN IN THE			
		C	DATA DEFINITIONS INCLUDED IN THIS EXAMPLE.			

SUMMARY

In summary, the programmer should consider the following when producing a report:

- A printer file specification sheet must be filled out for each printer file. Compiler rules require that file specification sheets appear immediately before their associated record definitions and prior to any coding or any area definitions. The printer file specification sheet contains all those parameters that remain constant throughout a run.

- The first four character positions of the printer record are referred to as the printer control block. Each printline to be output needs its own control block.

- The PUT instruction must be used to output a printline. PUT has two formats: PUT File Reference and PUT File Reference,Workarea.

- The final item that is always required is an end-of-page routine. The software provides a standard end-of-page, but the programmer normally specifies his own.

- When the programmer specifies his own end-of-page routine, he should consider these five steps:

  1. Print end-of-page totals (optional).

  2. Eject page (required).

  3. Print heading (optional).

  4. Space the paper vertically (optional).

  5. Execute a RELINK (required).

  NOTE: A page must be ejected and a RELINK instruction must be used in the user's end-of-page routine.

PRINTER FILE OPENING

Printer files may be opened either automatically when the program is loaded into memory or by an OPEN instruction within the program. Any file having an OPEN instruction associated with it will not be automatically opened when the program is first loaded.

When the file is opened, two full pages of test pattern containing all the print characters in the 51-character alphanumeric set may be output on an optional basis. The programmer requests this option on the file specification sheet if output is not to magnetic media. The operator uses the test pattern to properly align the paper in the printer. After the test pattern is completed, the paper is ejected to the top of the next page.

As a further option, the programmer may specify the number of the form to be placed in the printer for the run. The form number is output before the test pattern is printed on an I/O writer, if available, or on the system printer.

PRINTER FILE CLOSING

Printer files may be closed at the end of a run when the FINISH instruction is encountered or during the run when a CLOSE instruction is encountered. Any file which has a CLOSE instruction associated with it and which is not closed before the FINISH instruction is encountered will be closed automatically.

When the file is closed, all buffers not yet empty are output and one full page is ejected. If a special form number was output when the file was opened, it will again be output when the file is closed.

** INTERIM FILES

Introduction

An interim file is used to temporarily store one or more printer reports when multiple reports are generated in the same run but the system has only one printer. The interim file is composed of printlines fully formatted with a control block. Each control block for a single report must contain the same selective print character. When multiple reports are output to the same interim file, unique selective print characters must be used for each report. (Any numeric or alphabetic character is acceptable.)

The programmer specifies the selective print character on a printer file specification sheet and omits the printer file reference name. The selective print character becomes the identifier for each report output to the interim file.

An interim file may also be created with the major function, Line Reporter. In this case, all the necessary data, including the selective print character, is acquired from the major function parameter sheets; therefore, the printer file specification sheet is eliminated.

The programmer uses the PUT instruction to output the formatted printlines to the interim file. He names the interim file as the file reference operand in the PUT instruction, e.g. PUT INTRMFLNAM. The resultant file may be printed in a later run by using a utility routine and specifying the selective print character of the desired report.

Block Size and Record Size

The size of the block output to the magnetic media depends on the actual peripheral used and on the system; however, each block can contain more than one record. All the records output to an interim file must be the same size. If multiple reports are output and the records for each report are constructed concurrently in memory, a workarea must be assigned to each report. If multiple reports are output and the records for each report are constructed consecutively, both records may be constructed in the output buffer by using SAME to redefine the record. For a complete description of using SAME to redefine a record, see the NEAT/3 REFERENCE MANUAL, INTRODUCTION AND DATA, tab 3, "Data Layout Sheets."

Interim File Definition

The programmer must complete file specification sheets for the printer and magnetic media files. The record must be defined on data layout sheets. This data is input to the compiler in the following order:

1. One set of magnetic media file specification sheets - the software uses these sheets for definition of the interim file. The set must be completed in the same manner as any other magnetic media destination file.

2. One printer file specification sheet for each report output - the software uses this sheet to obtain such information as the last data line number, the end-of-page routine reference name, and the selective print character for each report. (This sheet not needed with Line Reporter.)

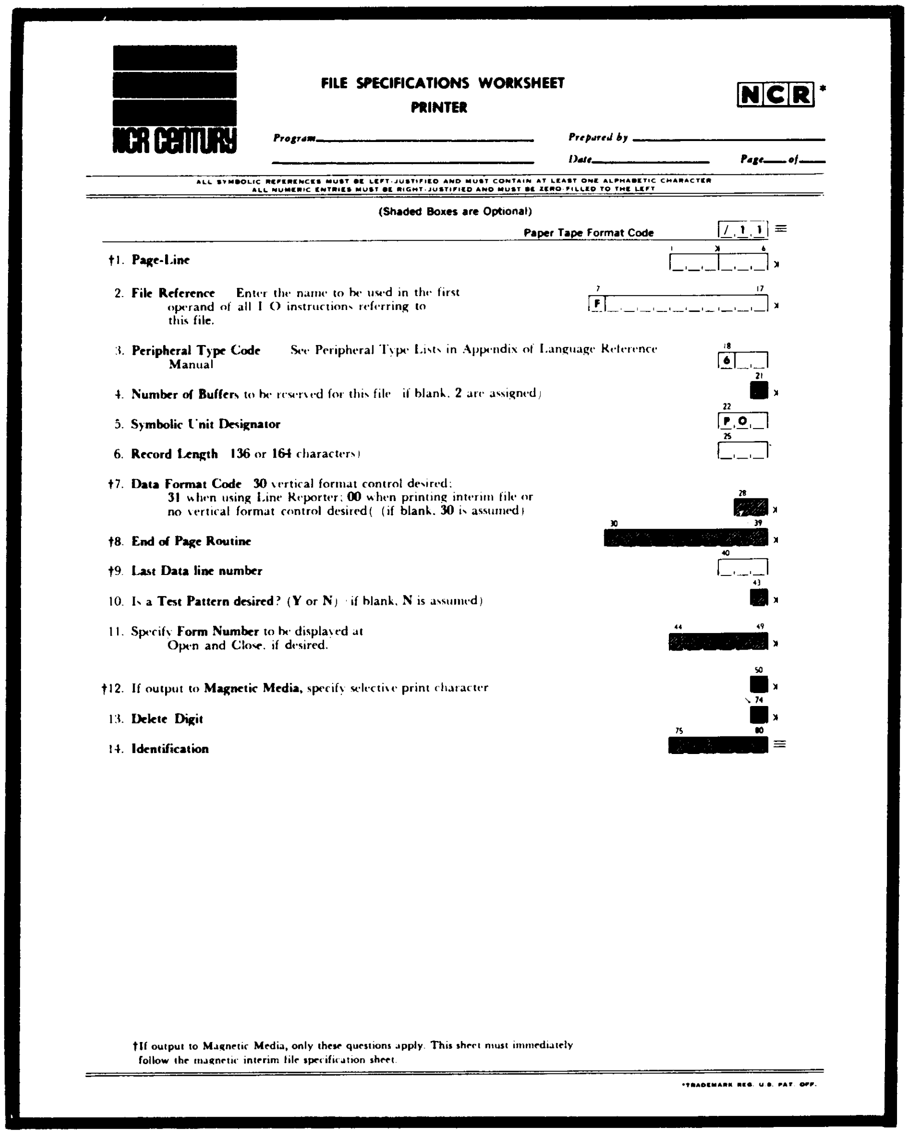
3. Data layout sheets - the software uses these sheets to define the record in the magnetic media output buffer.

* * * *

FILE SPECIFICATION SHEET FOR PRINTER

The printer file specification sheet describes file characteristics and file options for a printer file. One printer file specification sheet must be filled out for each printer file or each report output to an interim file.

Sheet 1



NCR CENTURY

FILE SPECIFICATIONS WORKSHEET
PRINTER

NCR*

Program__________ Prepared by __________
__________ Date__________ Page___of___

ALL SYMBOLIC REFERENCES MUST BE LEFT-JUSTIFIED AND MUST CONTAIN AT LEAST ONE ALPHABETIC CHARACTER
ALL NUMERIC ENTRIES MUST BE RIGHT-JUSTIFIED AND MUST BE ZERO-FILLED TO THE LEFT

(Shaded Boxes are Optional)

Paper Tape Format Code /11 ≡

†1. Page-Line	1 – 6	□□□□□□ ☒
2. File Reference Enter the name to be used in the first operand of all I O instructions referring to this file.	7 – 17	F□□□□□□□□□□ ☒
3. Peripheral Type Code See Peripheral Type Lists in Appendix of Language Reference Manual	18	6□□
4. Number of Buffers to be reserved for this file if blank, 2 are assigned)	21	■ ☒
5. Symbolic Unit Designator	22	PO□
6. Record Length 136 or 164 characters)	25	□□□
†7. Data Format Code 30 vertical format control desired; 31 when using Line Reporter; 00 when printing interim file or no vertical format control desired((if blank, 30 is assumed)	28	■ ☒
†8. End of Page Routine	30 – 39	■ ☒
†9. Last Data line number	40	□□□
10. Is a Test Pattern desired? (Y or N) if blank, N is assumed)	43	■ ☒
11. Specify Form Number to be displayed at Open and Close, if desired.	44 – 49	■ ☒
†12. If output to Magnetic Media, specify selective print character	50	■ ☒
13. Delete Digit	74	■ ☒
14. Identification	75 – 80	■ ≡

†If output to Magnetic Media, only these questions apply. This sheet must immediately follow the magnetic interim file specification sheet.

*TRADEMARK REG. U.S. PAT. OFF.

The programmer should fill in the header, the page-and-line number (question 1), and the identification tag (positions 75-80) as defined in the NEAT/3 REFERENCE MANUAL, INTRODUCTION AND DATA, tab 3, "Programming Worksheets". The paper tape format code is preprinted on this sheet and must be punched if paper tape is used for input to the compiler. Wherever practical on the following pages, the entries are filled with a typical remark.

2. FILE REFERENCE - ENTER THE NAME TO BE USED IN THE FIRST OPERAND OF ALL INSTRUCTIONS REFERRING TO THIS FILE.

F PRINTFILEØ

Enter the name used in the PUT instructions that access this file in the program.

If output is to an interim file on magnetic media, this entry is left blank. Printlines are output to an interim file using the PUT instructions and the file reference name of that interim file.

The "F" in position 7 is preprinted and must be punched.

3. PERIPHERAL TYPE CODE - (SEE PERIPHERAL TYPE LISTS IN APPENDIX OF LANGUAGE REFERENCE MANUAL)

6 _ _

Enter the code for the type of printer being used. For the proper code, see the peripheral type list in the appendix of this manual.

The 6 in position 18 is preprinted and must be punched.

4. NUMBER OF BUFFERS TO BE RESERVED FOR THIS FILE (IF BLANK, 2 ARE ASSIGNED)

□

Enter the number of buffers desired. Specifying two or more buffers provides write/compute simultaneity in processing this file. Since the printer is a low-speed peripheral, two buffers should be assigned to it before assigning multiple buffers to higher speed peripherals, such as the disc. Then, if memory space permits, multiple buffers can be assigned to the higher speed peripherals.

Two buffers are assigned if this entry is left blank.

5. SYMBOLIC UNIT DESIGNATOR

P01

Enter the symbolic unit designator for the system printer. The printers use codes P01-P09. The letter P and the numeral 0 are preprinted and must be punched.

6. RECORD LENGTH (136 OR 164 CHARACTERS)

136

The record length specified in this entry must include the 4-character control block. This length must be either 136 or 164 characters, depending upon the type of printer used.

† 7. DATA FORMAT CODE (30 WHEN VERTICAL FORMAT CONTROL IS DESIRED; 31 WHEN USING LINE REPORTER; 00 WHEN PRINTING INTERIM FILE OR NO VERTICAL FORMAT CONTROL DESIRED) (IF BLANK, 30 IS ASSUMED)

Enter 30 for vertical format control. This is the normal entry when a major function is not used to create the file. It indicates that the software is to keep track of the line count, convert the printer control block to a hardware acceptable format, and provide an end-of-page exit if desired, whether the report generated is output to a printer or an interim file.

Enter 31 if this sheet is used with a report generated by the major function Line Reporter and is output to the printer.

Do not use a printer file specification sheet with a report generated by the major function Line Reporter and output to an interim file.

Enter 00 if this sheet is used in conjunction with a utility routine printout of an interim file.

† 8. END OF PAGE ROUTINE ENDOFPAGEØ

Enter the name of an end-of-page routine if desired. Control is transferred to the routine referenced here when the last data line (Question 9) is encountered and the control block contains an F Code of P or N.

The programmer may use the routine referenced for any of the following reasons:

- Print and/or clear end-of-page totals.
- Eject a page.
- Print headers.

If an end-of-page routine is specified, the programmer must provide coding to eject a page and must use RELINK as a last instruction.

If this entry is left blank and vertical format control was requested in answer to the previous question, the page is ejected when the last data line (Question 9) is encountered.

This entry must be left blank if 31 or 00 was specified in answer to Question 7.

† IF OUTPUT TO MAGNETIC MEDIA, ONLY THESE QUESTIONS APPLY; HOWEVER, THE PREPRINTED "F" IN POSITION 7, AND "6" IN POSITION 18 MUST BE PUNCHED. THIS SHEET MUST IMMEDIATELY FOLLOW THE MAGNETIC INTERIM FILE SPECIFICATION SHEET.

† 9. LAST DATA LINE NUMBER | 0 5 6 |

Enter the number of the last line on which printing is to occur. If a printline which causes the paper to space either to the actual last data line or to cross the last data line is output, the software checks the F code of the control block. If the control block contains an F code of P or N, the software prints the line, stores a link, and transfers control to the programmer's end-of-page routine.

Enter the last data line if vertical format control is not specified in answer to Question 7 but a test pattern (Question 10) is desired, such as when a utility routine is used to print an interim file and a test pattern is desired before the report is printed.

10. IS A TEST PATTERN DESIRED? (Y OR N) (IF BLANK, N IS ASSUMED) | Y |

Enter Y for Question 10 if a test pattern is desired. If y is specified for Question 10 and if a last line number is specified, the software prints two pages of all characters in the 51-character alphanumeric character set, and ejects a page. The test pattern is printed until the last data line is encountered. If the STOP button is depressed while the test pattern is being printed, printing halts. During this halt, the paper may be adjusted. When the START button is pressed, one more page is ejected and two pages of test pattern are printed. The test pattern is printed immediately after OPEN and out-of-paper.

Leave blank if test pattern is not desired; N is assumed.

11. SPECIFY FORM NUMBER TO BE DISPLAYED AT OPEN AND CLOSE, IF DESIRED. | | | | | | |

The programmer may enter a number that specifies a preprinted or standard form in answer to this question. The form number is displayed at OPEN (prior to the printing of a test pattern) and again after closing the file on an I/O writer, if available, or on the system printer.

† 12. IF OUTPUT TO MAGNETIC MEDIA, SPECIFY SELECTIVE PRINT CHARACTER □

Enter a selective print character only if the file is to be output to a magnetic file device. Any X-type character is acceptable.

The selective print character entered must be the same as the third character in the printer control blocks for lines that pertain to this report.

* * * *

† IF OUTPUT TO MAGNETIC MEDIA, ONLY THESE QUESTIONS APPLY; HOWEVER, THE PREPRINTED "F" IN POSITION 7, AND "6" IN POSITION 18 MUST BE PUNCHED. THIS SHEET MUST IMMEDIATELY FOLLOW THE MAGNETIC INTERIM FILE SPECIFICATION SHEET.

PUNCH CARD FILES

INTRODUCTION

The NCR Century Series offers the option to process punch card files contained on 80-column punch cards.

The reading rate of the integrated Class 682-100 Punch Card Reader is 300 cards per minute. The Class 686 Card Reader/Punch of the Century Series offers three optional peripheral units.

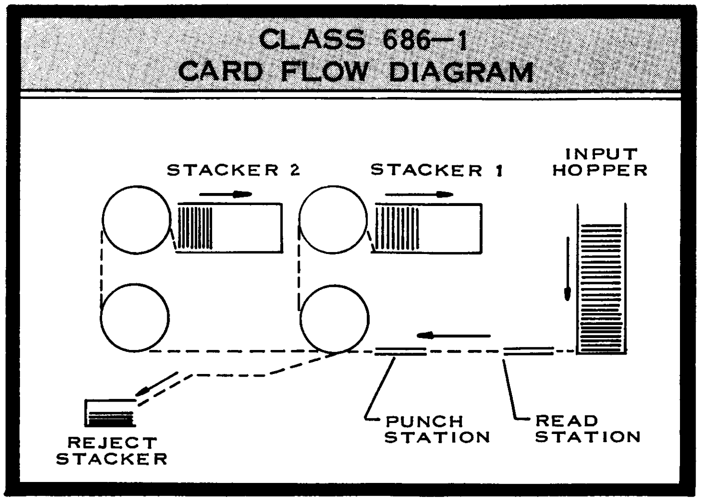
- The Class 686-101 is a combination card reader/punch that permits the use of cards as source or destination file media. The Class 686-101 also permits the use of cards for the processing of source-destination files. In this case the card reader/punch reads source file data at the read station and, after moving the card to the punch station, punches destination file data into reserved fields in the same card.

  The Class 686-101 reads at a rate of 750 cards per minute. The punching speed is 100 cards per minute.

- The Class 686-201 is a card reader that reads punch cards at a rate of 750 cards per minute.

- The Class 686-301 is a card punch that punches cards at a rate of 100 cards per minute.

All of the above Class 686 units have three output stackers, including a special reject stacker. The programmer may specify the output stacker into which each card is to be stacked after reading or punching.

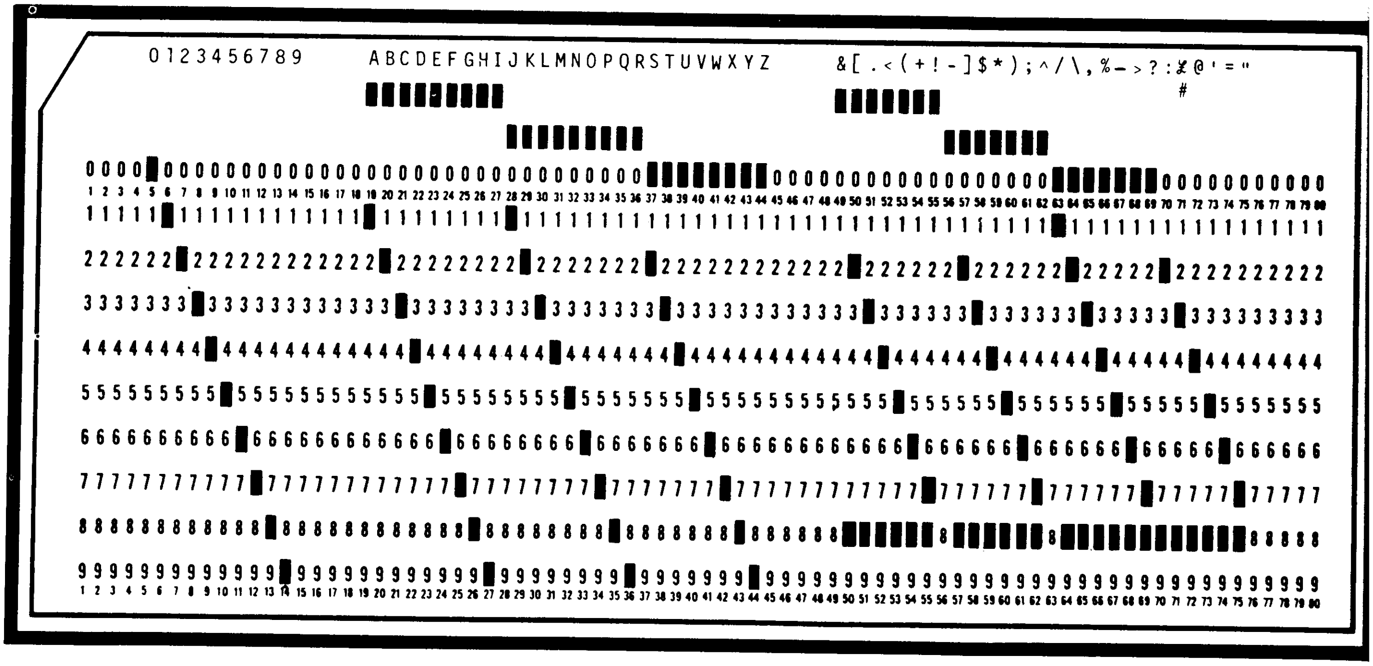


PUNCH CARD CODES

The NCR Century Series software offers the option of reading or punching cards in the Hollerith Extended "A" Set, the Hollerith Extended "H" Set, the 315 Hollerith code, and in the binary mode.

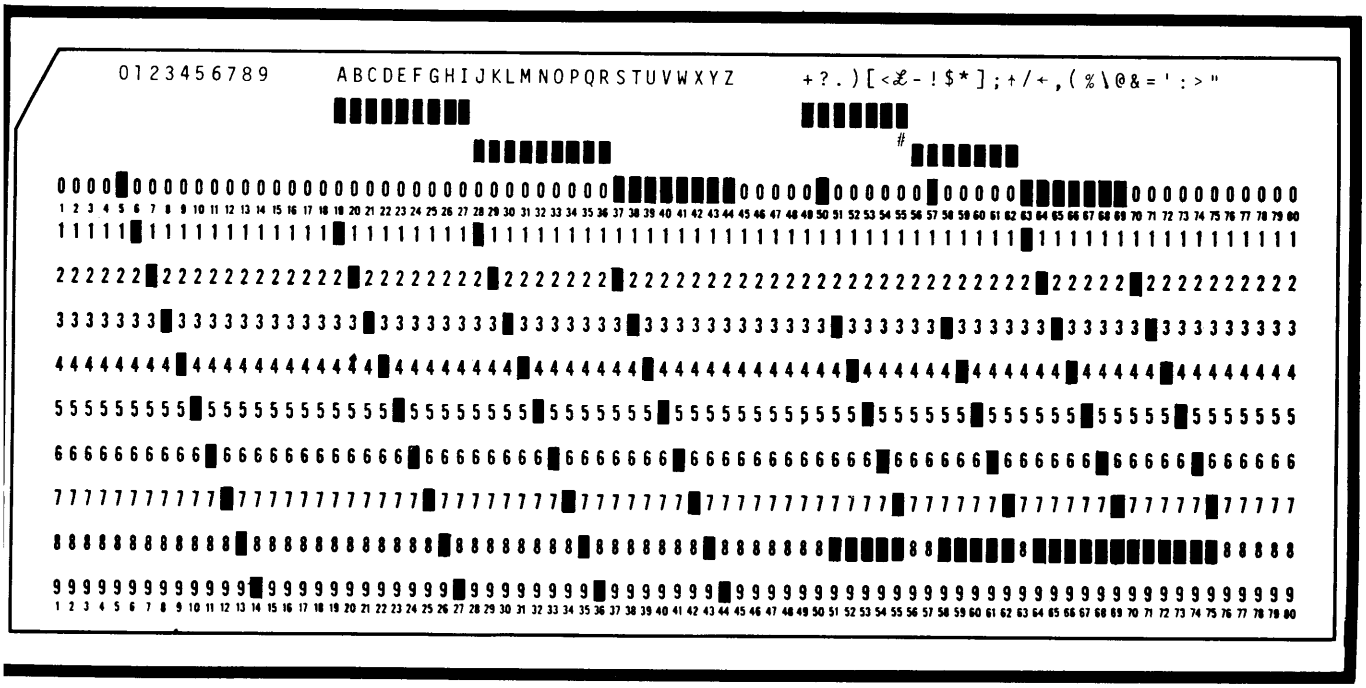
- Hollerith Extended "A" Set

  The NCR Century Series standard Hollerith Extended "A" Set consists of 64 characters. These characters and the corresponding punch configurations are shown in the punched card below.



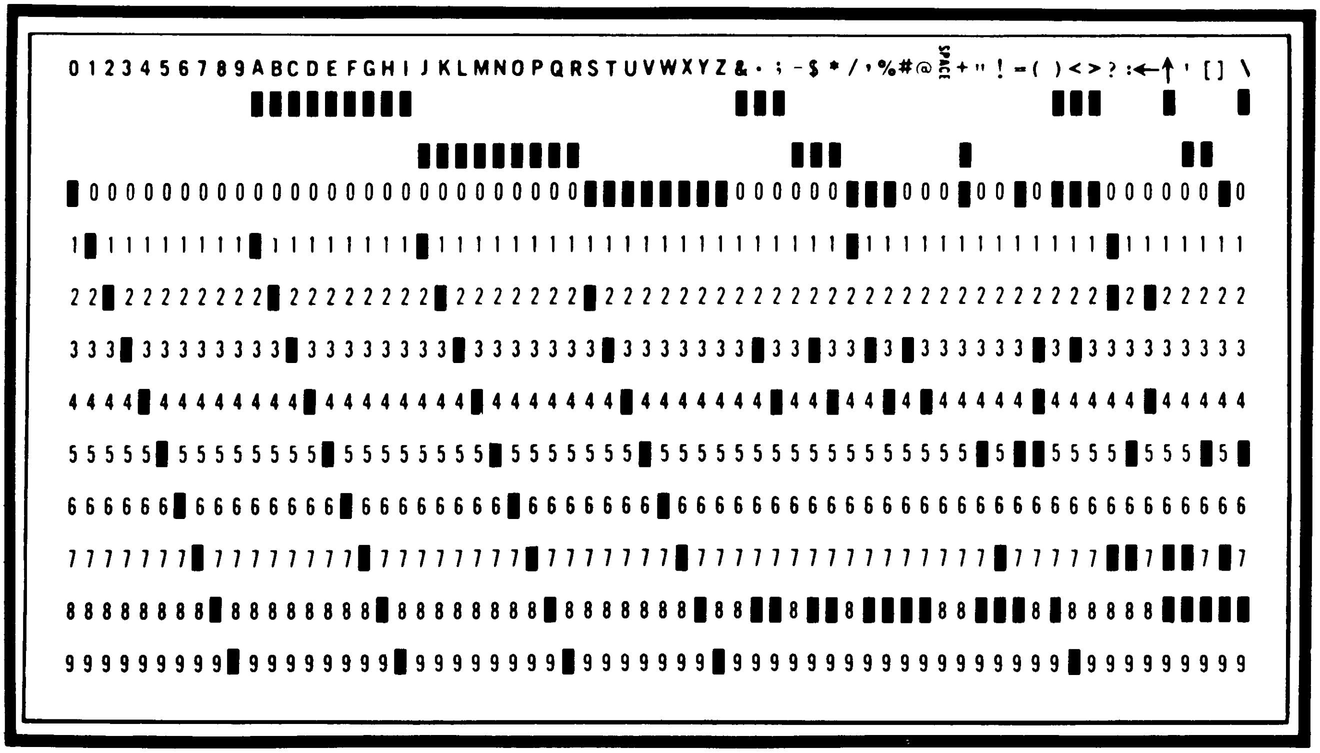
- Hollerith Extended "H" Set

  This code has the same character set as the Hollerith Extended "A" Set, but the punch configurations for the special symbols (+,?,%,:,*,etc.) are different.


0123456789 ABCDEFGHIJKLMNOPQRSTUVWXYZ +?.)[<£-!$*];+/←,(%\@&=':>"
#

- NCR 315 Hollerith Set

The characters and the corresponding punch configurations of the NCR 315 Hollerith Set are shown below.


0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ&.;-$*/,%#@ SPACE +"!=()<>?:←↑'[]\

- Binary Mode

  For reading and punching punched cards in the binary mode, see the PRODUCT INFORMATION MANUAL, PUNCHED MEDIA PERIPHERALS, Pub. No. 3.

FILE ORGANIZATION

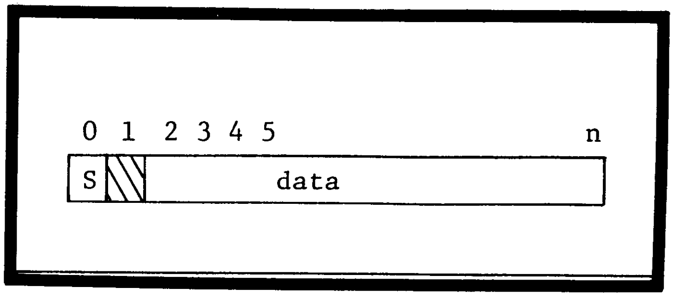
The programmer normally uses the major function Card Regiment to read and the PUT instruction to punch one 80-column card at a time. The major function permits the use of up to 99 different card formats in one file. See NEAT/3 REFERENCE MANUAL, MAJOR FUNCTIONS, tab 1, "Card Regiment."

NOTE

> For processing punched cards without using Card Regiment, see the OPERATING SYSTEMS MANUAL, I/O EXECUTIVE, Peripheral Dependent tab, "Punched Cards."

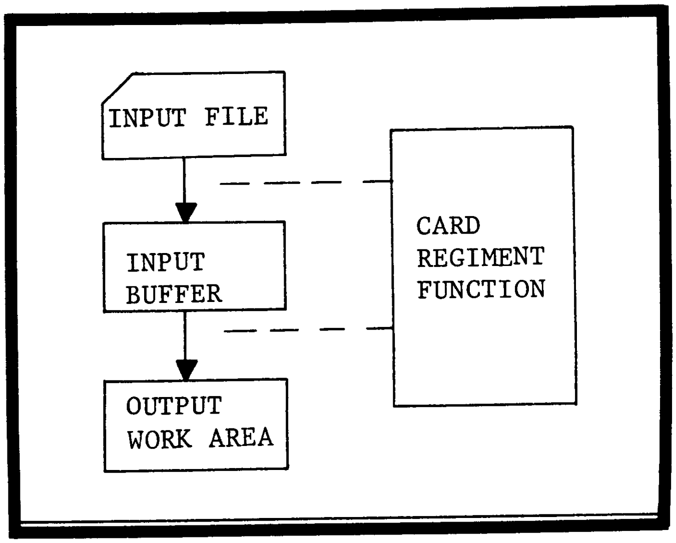
The programmer defines the number and length of buffers for card input or output. The maximum length of input and output buffers is 82 characters: 2 characters for a stacking code and 80 characters for punched card data.

To selectively stack cards in a Class 686 Card Reader/Punch, the programmer must place the stacking code in the first character position of the input or output buffer. The second character position of this buffer is reserved for future use. "S" represents the stacking code in the following illustration.



The programmer must define the field for the stacking code on the data layout sheets for the input or output buffers.

When reading punched cards, the program must access the input records in the output work area specified on the parameter sheets for the Card Regiment function.



To output punched cards using the PUT instruction, see OPERATING SYSTEM MANUAL, I/O EXECUTIVE, Peripheral Dependent tab, "Punched Cards."

Source Files

The programmer uses the Card Regiment function to pass one complete punch card through the card reader and to store the data from the card in the output work area specified on the Card Regiment parameter sheets. (The programmer does not use a GET instruction for inputting punch card records.)

The programmer must specify on the file specifications sheet the code set for the data in the source file cards (Hollerith Extended "A" or "H" Code Set).

- Rescue Cards

  The I/O software offers the option to use rescue cards for dividing large source files into segments. The use of this option saves rerun time in case of an interruption of the card reading run.

  If the programmer specifies the use of rescue points on the file specifications sheet and a rescue card is encountered in a source deck, the software initiates a memory dump and writes the entire memory into a magnetic rescue file. If an interruption of the card reading run occurs, the operator can restart the run from the rescue point. He then needs to reread only the cards in the file segment which was being processed at the time the interruption occurred.

  If the programmer does not use the rescue option and an interruption of the card reading run occurs, the operator has to reread the entire file.

  The format of the rescue card is RES$ in columns 1 through 6 of the card.

NOTE

Rescue cards initiate a rescue dump only if the programmer specifies on the file specifications sheet the rescue option and no selective stacking. If no rescue option or selective stacking is specified, the software accepts rescue cards as data.

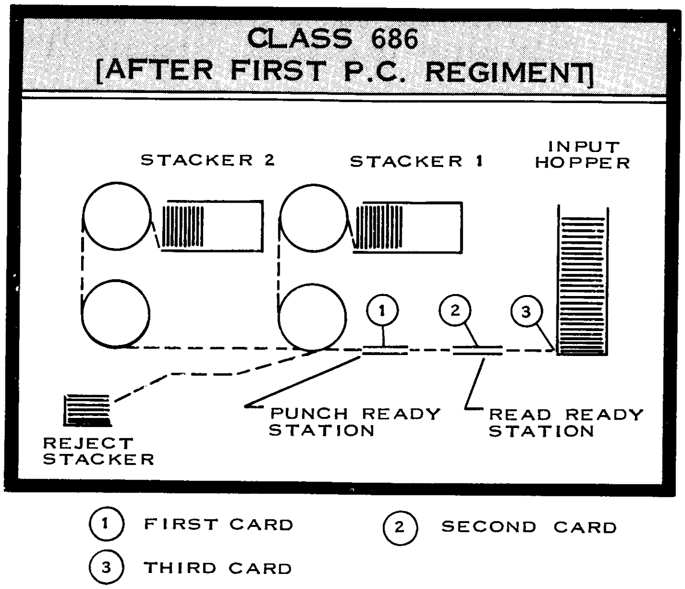
- Selective Stacking (Source Files)

If the user's program calls for selective stacking of cards, the input buffer must include two additional character positions for the stacking code. These two character positions are relative positions 0 and 1 of the input buffer. The programmer must include these two character positions in the data definition for the input buffer.

The Class 686 Card Reader/Punch can stack cards selectively; the integrated card reader does not have this capability.

The Class 686 Card Reader/Punch stacks cards selectively in the following manner. After reading a card in the read station and processing the input record, the user's program must place the stacking code in the input buffer for the record just processed. The next execution of the Card Regiment function for the same file then selects an output stacker according to the stacking code which was placed in the input buffer. As the next card passes through the read station, the previous card moves into the selected stacker.

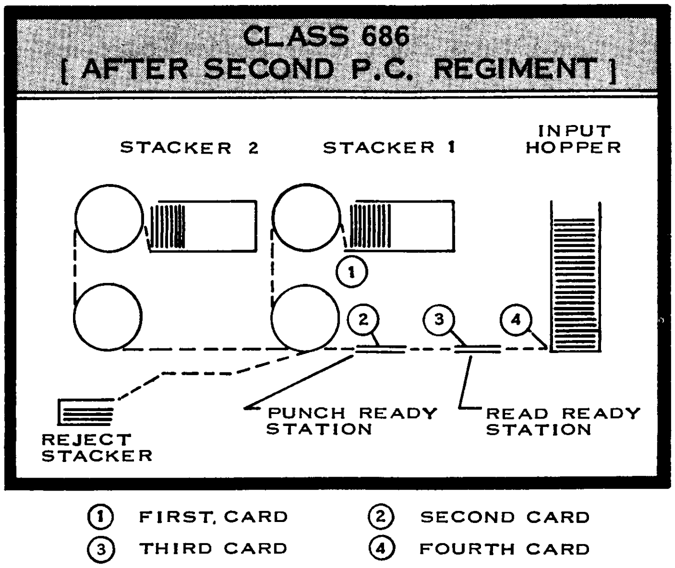
Example:



The first execution of the Card Regiment function moves the first card through the read station and makes the data from the first card available to the program. The first card stops in the punch ready station.

The user's program must insert in the input buffer the stacking code for the first card. In this example, S=1 to select stacker 1.

Example (continued):



The second execution of the Card Regiment function reads the second card and makes the data from this card available to the program.

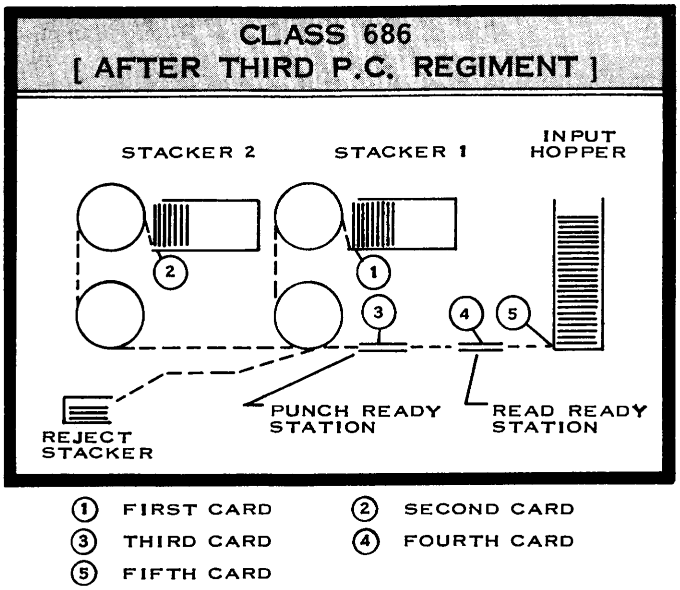
The first card moves into the previously designated stacker (1).

The second card moves into the punch ready station.

The third card moves into the read ready station.

The programmer must insert in the input buffer the stacking code for the second card. In this example, S=2 to select stacker 2.

Example (continued):



The third execution of the Card Regiment function reads the third card and makes the data from this card available to the program.

The second card moves into the previously designated stacker (2).

The third card moves into the punch ready station.

The fourth card moves into the read ready station.

The programmer must insert in the input buffer the stacking code for the third card.

The stacking code "S" corresponds to the number of the output stacker.

"S" Code	Function
1	Send previous card to stacker 1
2	Send previous card to stacker 2
3	Send previous card to stacker 3*
* Reject Stacker - has limited capacity. Primarily for use by software	

NOTE

If the user's program does not place a stacking code in the input buffer, the I/O software automatically places the card in stacker 1.

The software logs any "S" code other than 1, 2, or 3 as a programming error and permits continuation of the program.

- Multiple Buffers

  Multiple input buffers provide more efficient input/compute simultaneity. However, if the program calls for selective stacking or if the source file is part of a source-destination file, the compiler automatically assigns one buffer only.

Destination Files

Each execution of the PUT instruction punches the contents of a specified work area into a single card. The programmer assembles each output record in the specified work area.

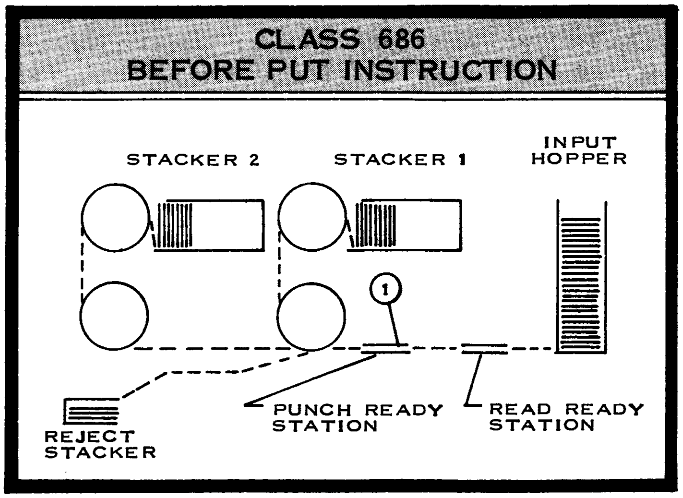
The programmer must specify on the file specifications worksheet the code set in which the cards are to be punched.

- Selective Stacking (Destination Files)

If a program calls for selective stacking of cards, the output buffer must include two extra character positions in relative positions 0 and 1. Relative character position 0 is for the stacking code, and relative character position 1 is reserved for future use. The programmer must include these two character positions in the data definition for the output buffer.

Before executing a PUT instruction the user's program must place the stacking code in the output buffer for the card to be punched. The stacking code in the output buffer then selects an output stacker to receive the card after punching.

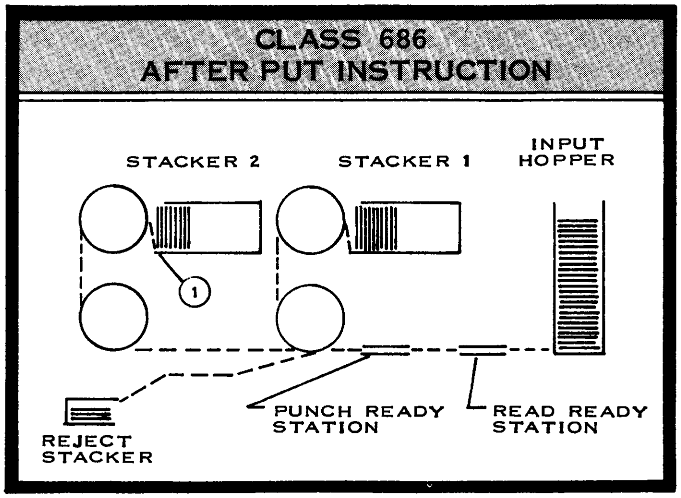
Example:



① CARD BEFORE EXECUTION

The program processes a record to be output and inserts a stacking code in the output buffer for this record. In this example, S=2 to select stacker 2.

An execution of the PUT instruction for this file then punches a card and stacks the card according to the stacking code in the output buffer.



① CARD AFTER EXECUTION

The stacking code "S" corresponds to the number of the output stacker.

"S" Code	Function
1	Send present card to stacker 1
2	Send present card to stacker 2
3	Send present card to stacker 3*

* Reject Stacker - has limited capacity.

Primarily for use by software.

NOTE

> If the user's program does not place a stacking code in the output buffer, the I/O software automatically places the card being punched in stacker 1.
>
> The software logs any "S" code other than 1, 2, or 3 as a programming error and permits continuation of the program.

- Punch Error Rejection

  The programmer may specify on the file specification sheet the option to repunch all cards in which a punch error occurs. The software then rejects all punch error cards into the reject stacker and repunches the same record into the next card.

- Multiple Buffers

  Multiple output buffers provide more efficient output/compute simultaneity. However, if the program calls for selective stacking or if the destination file is part of a source-destination file, the compiler automatically assigns one buffer only.

Source-Destination Combination Files

The user may design punch cards so that each card contains a source file record and a destination file record. The Class 686-101 Card Reader/Punch processes a source-destination file as independent source and destination files, each with its own file specification sheet.

NOTE

> For a source-destination file, the compiler automatically assigns one input buffer for the source portion of the file and one output buffer for the destination portion of the file.
>
> The automatic punch error rejection option is not available when processing source-destination files.

NOTE

A program should not successively execute the PUT instruction for a source-destination file. Two successive executions of a PUT instruction move the next card through the read station, and the opportunity to read the first card is lost.

Similarly, two successive executions of a Card Regiment function for a source-destination file move the first card through the punch station, and the opportunity to punch this card is lost.

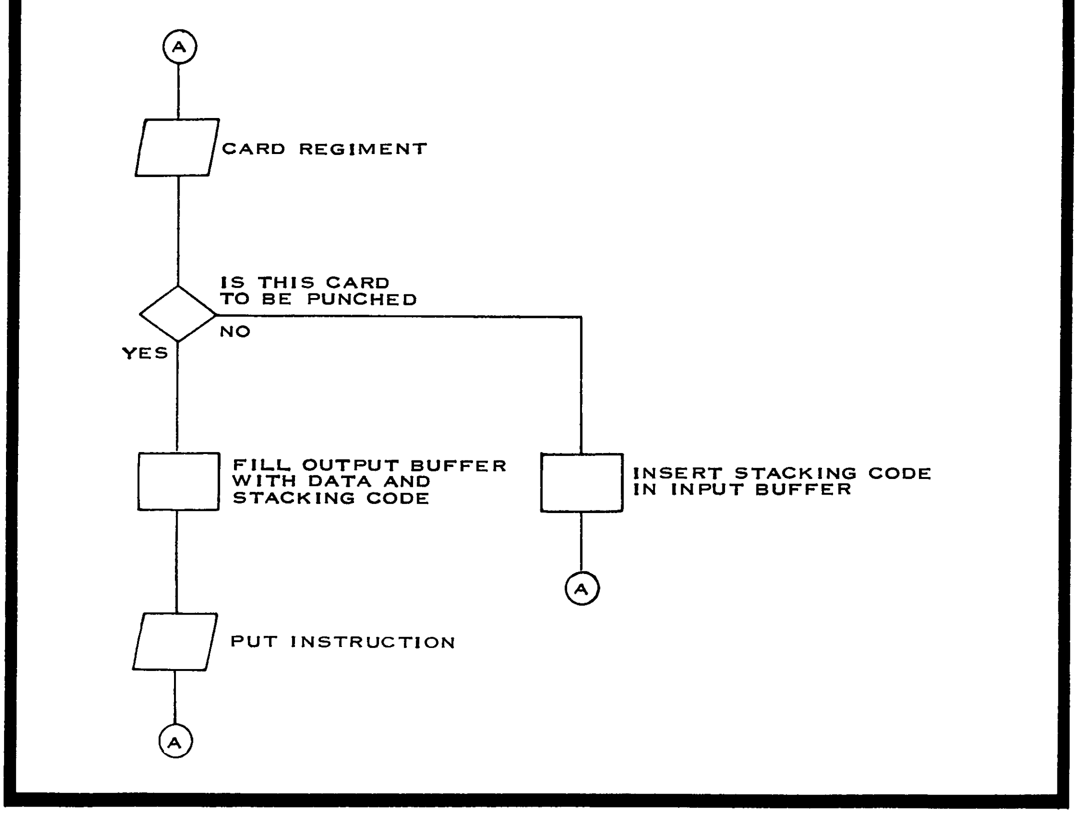
- Buffer Assignment

The software automatically assigns a single buffer to a source or destination file that is described on the file specifications sheet as being part of a source-destination file.

- Selective Stacking (Source-Destination Files)

The programmer may use stacking codes in processing either the source portion or the destination portion of the file. The same rules as for stacking cards in individual source files or destination files apply.

SELECTIVE STACKING OF CARDS
SOURCE-DESTINATION FILE PROCESSING


A
CARD REGIMENT
IS THIS CARD TO BE PUNCHED
NO
YES
FILL OUTPUT BUFFER WITH DATA AND STACKING CODE
INSERT STACKING CODE IN INPUT BUFFER
A
PUT INSTRUCTION
A

OPENING OF CARD FILES

Before the user's program can read or punch cards, the respective source or destination files must be opened. This is accomplished either automatically at the beginning of the program or, at the programmer's option, at any point in the program through the use of an OPEN instruction. The OPEN instruction is explained in the NEAT/3 REFERENCE MANUAL, INSTRUCTIONS, tab 1, "Open Instruction."

- Opening Card Source Files

  The automatic open or the OPEN instruction for a source file performs the following functions:

  - Checks card reader for being ready.

  - Checks card reader for being a Class 686 if the file specifications sheet requests selective stacking.

  - Reads cards and fills N-1 buffers. (N is the number of input buffers specified on the file specifications sheet.)

  - Inserts a "1" in the stacking code position of the input buffer if the file specifications sheet specifies selective stacking.

- Opening Card Destination Files

  The automatic open or the OPEN instruction for a destination file performs the following functions:

  - Checks card punch for being ready.
  - Checks card punch for being Class 686 if the file specifications sheet requests selective stacking.
  - Inserts a "1" in the stacking code position of the output buffer if the file specifications sheet specifies selective stacking.

CLOSING OF CARD FILES

Before a program terminates, all the files used by the program must be closed. This is accomplished either automatically at the end of the program or, at the programmer's option, at any point in the program through the use of a CLOSE instruction. The CLOSE instruction is explained under Instructions, Tab 1, "Close Instruction."

- Closing of Card Source Files

  The automatic close or the CLOSE instruction for a source file performs only internal functions except under the following condition. If a FINISH or a CLOSE instruction closes a source file before the end card (END$) has been read, the software continues reading cards until the end card is found. On a Class 686 Card Reader, the software sends all cards read during the closing of a file to stacker 1.

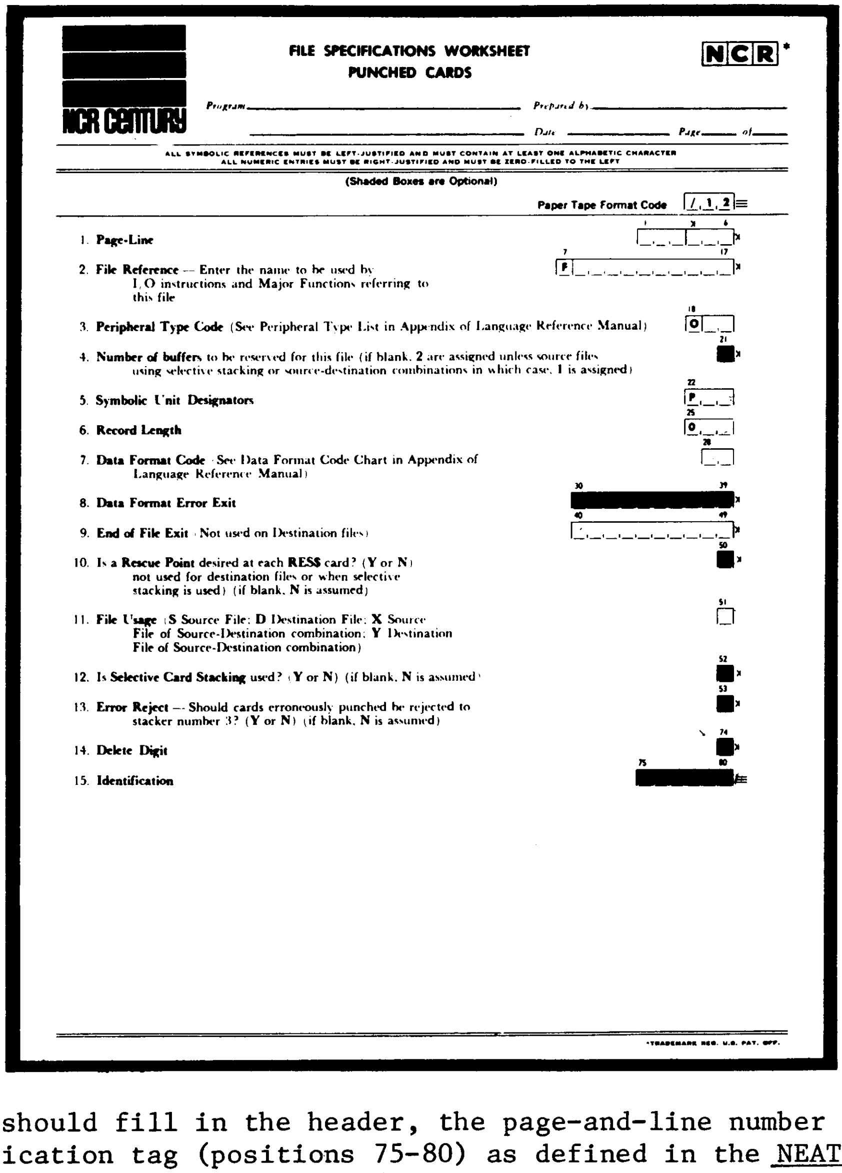
- Closing of Card Destination Files

The automatic close or the CLOSE instruction for a destination file performs the following functions:

- Completes already initiated card output operations.
- Punches an end card with END$ in columns 1-4 and stacks this card in stacker 1.

FILE SPECIFICATIONS WORKSHEET FOR PUNCHED CARDS

The programmer must prepare a file specifications worksheet for each file used in a program. The file specifications worksheets define the file characteristics, the file options, and the type of processing desired. The entries on file specifications worksheets become part of a source program and are input to the NEAT/3 Compiler.



NCR CENTURY

FILE SPECIFICATIONS WORKSHEET
PUNCHED CARDS

NCR*

Program ____________ Prepared by ____________

____________ Date ________ Page ____ of ____

ALL SYMBOLIC REFERENCES MUST BE LEFT-JUSTIFIED AND MUST CONTAIN AT LEAST ONE ALPHABETIC CHARACTER
ALL NUMERIC ENTRIES MUST BE RIGHT-JUSTIFIED AND MUST BE ZERO-FILLED TO THE LEFT

(Shaded Boxes are Optional)

Paper Tape Format Code /12≡

1. Page-Line (1–6)
2. File Reference -- Enter the name to be used by I/O instructions and Major Functions referring to this file (7–17) F
3. Peripheral Type Code (See Peripheral Type List in Appendix of Language Reference Manual) (18) O
4. Number of buffers to be reserved for this file (if blank, 2 are assigned unless source files using selective stacking or source-destination combinations in which case, 1 is assigned) (21)
5. Symbolic Unit Designators (22) P
6. Record Length (25) O
7. Data Format Code (See Data Format Code Chart in Appendix of Language Reference Manual) (28)
8. Data Format Error Exit (30–39)
9. End of File Exit (Not used on Destination files) (40–49)
10. Is a Rescue Point desired at each RES$ card? (Y or N) not used for destination files or when selective stacking is used) (if blank, N is assumed) (50)
11. File Usage (S Source File; D Destination File; X Source File of Source-Destination combination; Y Destination File of Source-Destination combination) (51)
12. Is Selective Card Stacking used? (Y or N) (if blank, N is assumed) (52)
13. Error Reject -- Should cards erroneously punched be rejected to stacker number 3? (Y or N) (if blank, N is assumed) (53)
14. Delete Digit (74)
15. Identification (75–80) ≡

*TRADEMARK REG. U.S. PAT. OFF.

The programmer should fill in the header, the page-and-line number (question 1), and the identification tag (positions 75-80) as defined in the NEAT/3 REFERENCE MANUAL, INTRODUCTION AND DATA, tab 3, "Programming Worksheets." The paper tape format code is preprinted on this sheet and must be punched if paper tape is used for input to the compiler.

7

F

2. FILE REFERENCE - ENTER THE NAME TO BE USED BY I/O INSTRUCTIONS AND MAJOR FUNCTIONS REFERRING TO THIS FILE

The letter F in position 7 is preprinted and must be punched.

Enter in positions 8 through 17 the name of the card file being described for this program. Throughout the program, Card Regiment may refer to this punch card file by using the file reference name.

The file reference name must start in position 8, may be up to ten characters long, and may consist of letters and numerals. Include at least one letter in the file reference name.

Each file, including those which are part of a source-destination file, must have its unique name within the program.

18

O

3. PERIPHERAL TYPE CODE (SEE PERIPHERAL TYPE LIST IN APPENDIX OF LANGUAGE REFERENCE MANUAL)

The 0 in position 18 is preprinted and must be punched.

Enter in positions 19 and 20 a 2-numeral code to identify the type of card reader or card punch used. The compiler uses this code to generate the proper object program for handling the cards on a particular type of card reader or card punch.

See the NEAT/3 REFERENCE MANUAL, APPENDIX, tab 1, "Peripheral Type Codes."

21

▨

4. NUMBER OF BUFFERS TO BE RESERVED FOR THIS FILE (IF BLANK, 2 ARE ASSIGNED UNLESS SOURCE FILES USING SELECTIVE STACKING OR SOURCE-DESTINATION COMBINATIONS IN WHICH CASE, 1 IS ASSIGNED)

Enter in position 21 the number of input or output buffers to be used by this file. If column 21 is a space, two buffers are assigned automatically. Two buffers normally provide the greatest efficiency.

If selective stacking is specified, or if this file is part of a source-destination file, position 21 is ignored. The software assigns one input or output buffer.

22

P

5. SYMBOLIC UNIT DESIGNATORS

The letter P in position 22 indicates a peripheral unit for paper media. This letter is preprinted on the file specifications worksheet and must be punched.

Enter in positions 23 and 24 a number to complete the symbolic unit designator.

The symbolic unit designator for any freestanding card reader may be an entry from P11 to P19. (If a system has an integrated card reader, P11 is reserved for that unit.) If a card source file is to be processed on the integrated card reader, the symbolic unit designator must be P11.

The symbolic unit designator for a card punch may be any entry from P21 to P29.

25
O

6. RECORD LENGTH

The number 0 in position 25 is preprinted and must be punched.

Enter in positions 26 and 27 the fixed length of the punched card records (from the first column in the card to the last column actually used). If selective stacking is specified, the maximum record length is 82 characters: 80 data characters (80 card columns) plus two stacking code characters. The record length entry must agree with the record length indicated on the data layout sheet following this file specification worksheet, except in the case of either binary data format code. With binary cards, the entry in answer to this question is a maximum of 80 or 82; however, the entry on the data layout sheet must be a maximum of 160 or 162 because two passes must be made to read a binary card.

28

7. DATA FORMAT CODE

Enter in positions 28 and 29 the data format code of the character set used.

DATA FORMAT CODES

	With Stacking	Without Stacking
No Translation	10	00
Standard Century H Set	11	01
Standard Century A Set	12	02
315 Hollerith	13	03
Binary	14	04

30 39

8. DATA FORMAT ERROR EXIT

Leave this entry blank if Card Regiment is used for this file. Otherwise, the name of a routine to handle records containing illegal characters may be entered here. By accessing File-Reference.BADCHAR, the programmer can determine the number of illegal characters in the current record. This exit is taken after the entire record has been translated.

When a source file is processed, punch configurations outside of the specified code set are considered invalid. The software translates these configurations as replacement characters and substitutes a hexadecimal 3C, which is interpreted as the > character.

When a destination file is processed, the software encodes the characters in the output buffer before punching. Characters outside of the specified code set are considered invalid and are translated by the software as spaces (no punch).

40 49

9. END OF FILE EXIT (NOT USED ON DESTINATION FILES)

Leave this entry blank if Card Regiment is used for this file. Otherwise, enter the reference of the routine to which control is to be transferred when the end of the file is detected.

NOTE

Once an end-of-file exit is taken, the last record in the input buffer is no longer accessible to the program.

50

10. IS A RESCUE POINT DESIRED AT EACH RES$ CARD? (Y OR N) (NOT USED FOR DESTINATION FILES OR WHEN SELECTIVE STACKING IS USED) (IF BLANK, N IS ASSUMED)

Position 50 is an optional entry and applies only to source files.

Enter "Y" in position 50 if rescue points are desired. When requested, the software initiates a rescue dump each time it reads a rescue card (RES$ in columns 1-4) in a card source file. The rescue dump is made to the magnetic destination file designated as the standard rescue file.

Enter "N" in position 50, or leave this position blank, if no rescue points are wanted. Any rescue cards read are then considered data.

If the file being described is a destination file (see entry 11) or if selective stacking is specified (see entry 12) the software ignores the entry in position 50 and treats rescue cards as data.

51 □

11. FILE USAGE (S SOURCE FILE, D DESTINATION FILE, X SOURCE FILE OF SOURCE-DESTINATION COMBINATION, Y DESTINATION FILE OF SOURCE-DESTINATION COMBINATION)

This one-character entry in position 51 identifies the type of file.

Enter "S" to specify a source file.

Enter "D" to specify a destination file.

Enter "X" to specify the source file of a source-destination combination file.

Enter "Y" to specify the destination file of a source-destination combination file.

52 ▨

12. IS SELECTIVE CARD STACKING USED? (Y OR N) (IF BLANK, N IS ASSUMED)

This entry is ignored if the entry in positions 19 and 20 specifies the integrated card reader.

Enter a "Y" in position 52 to specify selective stacking of cards on a Class 686 Card Reader/Punch. The software looks for the stacking code in the first character position of the input or output buffers for this file and ignores the second character position.

Enter an "N" in position 52 if no selective stacking is wanted. The input or output buffers are considered to contain data only.

53 ▨

13. ERROR REJECT - SHOULD CARDS ERRONEOUSLY PUNCHED BE REJECTED TO STACKER NUMBER 3? (Y OR N) (IF BLANK, N IS ASSUMED)

This entry is optional and is used only for destination files which are not part of a source-destination file.

Enter "Y" in position 53 to reject any card on which a punch error occurs. The rejected card is stacked in the reject stacker (3).

Enter "N" in position 53 if the reject feature is not to be used. If position 53 is left blank, "N" is assumed.

The error reject feature and the automatic retry routine in the software permit the automatic handling of random punch errors and allow the continuous processing with a minimum need for operator intervention.

14. DELETE DIGIT

This entry is explained in the NEAT/3 REFERENCE MANUAL, INSTRUCTIONS, tab 3, "Compiler Control Worksheet."

PUNCH PAPER TAPE FILES

INTRODUCTION

The NCR Century Series offers the optional use of paper tape media for data input and output. The following standard paper tape widths are acceptable.

Width	Channels
1"	8
7/8"	7
11/16"	5

Paper Tape Readers

The integrated paper tape reader permits the use of source files on punched paper tape media. The reading rate is 1000 characters per second. The paper tape may be in rolls or in strips.

An optional peripheral unit in the NCR Century Series is the Class 660-101 Paper Tape Reader. The reading rate of this unit is 1500 characters per second. The use of supply and take-up reels permits tape to be rewound at the end of a reel either manually, by pressing a rewind switch, or automatically, under software control. Rolls and strips of punched paper tape may be processed on the Class 660-101 Paper Tape Reader.

Paper Tape Punch

An optional peripheral unit in the NCR Century Series is the Class 665-101 Paper Tape Punch. This unit punches paper tape at a rate of 200 characters per second.

PAPER TAPE FORMAT

The NCR Century Series software offers the option of reading or punching paper tape in any code. The programmer specifies on the file specifications worksheet for paper tape files one of the following two codes: the NCR Century Series paper tape code (USASI) or the user-defined paper tape code.

NCR Century Series Standard Code Set

The USASI paper-tape code is the standard code for the NCR Century Series. The use of this code permits information interchange between different systems and the operation of remote communications equipment.

The NCR Century software contains a translation table for translation between NCR Century Series internal code and the NCR Century Series standard paper-tape code.

The NCR Century Series standard paper-tape code and corresponding identification are shown on the following page. This code requires 8-channel paper tape, and the hole configuration for each character has even parity (even number of holes).

The NCR Century software recognizes certain control characters which are footnoted on the following page. (The characters not footnoted are considered as data characters by the software.)

NCR Century Series (USASI) Standard Paper Tape Code

CHARACTER I/D	8	7	6	5	4	·	3	2	1	
NUL						·				*1
SOH	●					·			●	
STX	●					·		●		
ETX						·		●	●	
EOT	●					·	●			
ENQ						·	●		●	
ACK						·	●	●		
BEL	●					·	●	●	●	
BS	●				●	·				
HT					●	·			●	
LF					●	·		●		
VT	●				●	·		●	●	
FF					●	·	●			
CR	●				●	·	●		●	
SO	●				●	·	●	●		*2
SI					●	·	●	●	●	*3
DLE	●			●		·				
DC1				●		·			●	
DC2				●		·		●		
DC3	●			●		·		●	●	
DC4				●		·	●			
NAK	●			●		·	●		●	
SYN	●			●		·	●	●		*4
ETB				●		·	●	●	●	
CAN				●	●	·				*5
EM	●			●	●	·			●	*6
SS	●			●	●	·		●		
ESC				●	●	·		●	●	*7
FS	●			●	●	·	●			*8
GS				●	●	·	●		●	
RS				●	●	·	●	●		*9
US	●			●	●	·	●	●	●	*10

CHARACTER I/D	8	7	6	5	4	·	3	2	1	
@	●	●				·				
A		●				·			●	
B		●				·		●		
C	●	●				·		●	●	
D		●				·	●			
E	●	●				·	●		●	
F	●	●				·	●	●		
G		●				·	●	●	●	
H		●			●	·				
I	●	●			●	·			●	
J	●	●			●	·		●		
K		●			●	·		●	●	
L	●	●			●	·	●			
M		●			●	·	●		●	
N		●			●	·	●	●		
O	●	●			●	·	●	●	●	
P		●		●		·				
Q	●	●		●		·			●	
R	●	●		●		·		●		
S		●		●		·		●	●	
T	●	●		●		·	●			
U		●		●		·	●		●	
V		●		●		·	●	●		
W	●	●		●		·	●	●	●	
X	●	●		●	●	·				
Y		●		●	●	·			●	
Z		●		●	●	·		●		
[	●	●		●	●	·		●	●	
\		●		●	●	·	●			
]	●	●		●	●	·	●		●	
∧	●	●		●	●	·	●	●		
_		●		●	●	·	●	●	●	

SP (¢)	●		●			·			
!			●			·			●
"			●			·		●	
# (£)	●		●			·		●	●
$			●			·	●		
%	●		●			·	●		●
&	●		●			·	●	●	
'			●			·	●	●	●
(			●		●	·			
)	●		●		●	·			●
*	●		●		●	·		●	
+			●		●	·		●	●
,	●		●		●	·	●		
-			●		●	·	●		●
.			●		●	·	●	●	
/	●		●		●	·	●	●	●
0			●	●		·			
1	●		●	●		·			●
2	●		●	●		·		●	
3			●	●		·		●	●
4	●		●	●		·	●		
5			●	●		·	●		●
6			●	●		·	●	●	
7	●		●	●		·	●	●	●
8	●		●	●	●	·			
9			●	●	●	·			●
:			●	●	●	·		●	
;	●		●	●	●	·		●	●
<			●	●	●	·	●		
=	●		●	●	●	·	●		●
>	●		●	●	●	·	●	●	
?			●	●	●	·	●	●	●

`		●	●			·				
a	●	●	●			·			●	
b	●	●	●			·		●		
c		●	●			·		●	●	
d	●	●	●			·	●			
e		●	●			·	●		●	
f		●	●			·	●	●		
g	●	●	●			·	●	●	●	
h	●	●	●		●	·				
i		●	●		●	·			●	
j		●	●		●	·		●		
k	●	●	●		●	·		●	●	
l		●	●		●	·	●			
m	●	●	●		●	·	●		●	
n	●	●	●		●	·	●	●		
o		●	●		●	·	●	●	●	
p	●	●	●	●		·				
q		●	●	●		·			●	
r		●	●	●		·		●		
s	●	●	●	●		·		●	●	
t		●	●	●		·	●			
u	●	●	●	●		·	●		●	
v	●	●	●	●		·	●	●		
w		●	●	●		·	●	●	●	
x		●	●	●	●	·				
y	●	●	●	●	●	·			●	
z	●	●	●	●	●	·		●		
{		●	●	●	●	·		●	●	
|	●	●	●	●	●	·	●			
}		●	●	●	●	·	●		●	
¬		●	●	●	●	·	●	●		
DEL	●	●	●	●	●	·	●	●	●	*11

*MEANING TO 615 SOFTWARE

1 - NULL
2 - DOWN SHIFT
3 - UP SHIFT
4 - SYNC CODE
5 - VOID DATA
6 - END OF MEDIA
7 - ESCAPE CHARACTER
8 - END OF FILE
9 - END OF RECORD
10 - END OF FIELD
11 - IGNORE (DELETE)

LEGEND FOR CONTROL CHARACTERS IN USASI CODE SET

NUL	NULL
SOH	START OF HEADING
STX	START OF TEXT
ETX	END OF TEXT
EOT	END OF TRANSMISSION
ENQ	ENQUIRY
ACK	ACKNOWLEDGE
BEL	BELL [AUDIBLE OR ATTENTION SIGNAL]
BS	BACKSPACE
HT	HORIZONTAL TABULATION [PUNCHED CARD SKIP]
LF	LINE FEED
VT	VERTICAL TABULATION
FF	FORM FEED
CR	CARRIAGE RETURN
SO	SHIFT OUT
SI	SHIFT IN
DLE	DATA LINK ESCAPE
DC1	DEVICE CONTROL 1
DC2	DEVICE CONTROL 2
DC3	DEVICE CONTROL 3
DC4	DEVICE CONTROL 4 [STOP]
NAK	NEGATIVE ACKNOWLEDGE
SYN	SYNCHRONOUS IDLE [SYNC CODE]
ETB	END OF TRANSMISSION BLOCK
CAN	CANCEL [VOID DATA]
EM	END OF MEDIA
SUB	SUBSTITUTE
ESC	ESCAPE
FS	FILE SEPARATOR [END OF FILE]
GS	GROUP SEPARATOR
RS	RECORD SEPARATOR [END OF RECORD]
US	UNIT SEPARATOR [END OF FIELD]
DEL	DELETE

In the NCR Century standard (USASI) paper tape code, the leader (run-in) and trailer punch configuration are always the NUL character (sprocket hole only).

User-Defined Paper Tape Code

To input or output paper tape in any code other than the NCR Century Series standard code set, the programmer must completely define the nonstandard code set on supplemementary file specifications worksheets. The programmer must assign a value to every punch configuration he expects to use. The software considers any undefined character as invalid.

If the programmer defines NUL in his code set, the software assigns NUL as the leader (run-in) and trailer punch configuration. If the programmer does not define NUL and defines DELETE, the software assigns DELETE as the leader and trailer punch configuration. If the programmer defines neither NUL nor DELETE, the software assigns a blank punch configuration (sprocket hole only) for the leader and the trailer.

The worksheets for nonstandard code set definition are explained following the mandatory file specifications worksheets.

FILE ORGANIZATION

The philosophy and organization of punched paper tape files are fully explained in the NEAT/3 REFERENCE MANUAL, MAJOR FUNCTIONS, tab 1, "Paper Tape Regiment."

PROGRAM CODING FOR PAPER TAPE INPUT AND OUTPUT

To read data input from paper tape, the programmer normally uses the Paper Tape Regiment major function. The Paper Tape Regiment function facilitates the input of paper tape files. When the programmer needs to access an item from punched paper tape, he transfers control to the Paper Tape Regiment function without using a GET instruction.

However, the programmer may code his own GET instruction to input data from paper tape. See OPERATING SYSTEM MANUAL, I/O EXECUTIVE, peripheral dependent tab, "Punched Paper Tape," The Decode Routine.

To output data to the paper tape punch, the programmer must use the PUT instruction. See OPERATING SYSTEM MANUAL, I/O EXECUTIVE, peripheral dependent tab, "Punched Paper Tape," The Encode Routine.

OPENING OF PAPER TAPE FILES

Before the user's program (Major Function) can read or punch paper tape, the respective source or destination files must be opened. Paper tape files can be opened automatically either at the beginning of a program or, at the programmer's option, at any point in the program by the use of the OPEN instruction. The OPEN instruction is explained in the NEAT/3 REFERENCE MANUAL, INSTRUCTIONS, tab 1, "File Instructions."

Opening Paper Tape Source Files

The software reads paper tape to fill N-1 input buffers. (N is the number of input buffers specified on the file specifications worksheet.)

Opening Paper Tape Destination Files

The software initializes output buffers and punches a leader (300 run-in control characters) in the paper tape.

CLOSING OF PAPER TAPE FILES

Before a program terminates, all the files used in the program must be closed. This is accomplished either automatically at the end of the program or, at the programmer's option, at any point in the program by the use of the CLOSE instruction. The CLOSE instruction is explained in the NEAT/3 REFERENCE MANUAL, INSTRUCTIONS, tab 1, "File Instructions."

Closing Paper Tape Source Files

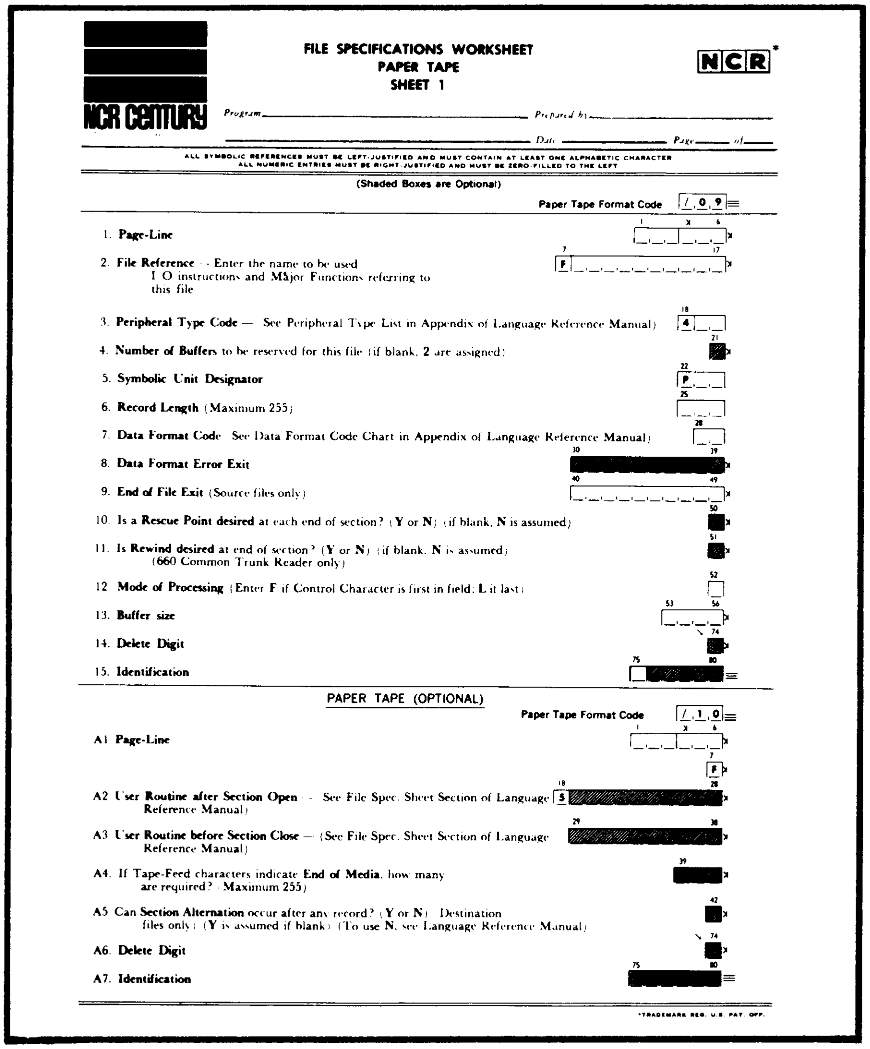
The software performs only internal functions.

Closing Paper Tape Destination Files

The software completes punching of data in the output buffers, punches an End-of-File character (if defined), and punches a trailer (300 run-in control characters) in the paper tape.

FILE SPECIFICATIONS WORKSHEETS FOR PAPER TAPE FILES

The programmer must prepare one or more file specifications worksheets (depending upon the options used) for each file used in a program. The file specifications worksheets define the file characteristics, the file options, and the type of processing desired. The entries on the file specifications worksheets become part of a source program and are input to the NEAT/3 Compiler.



NCR CENTURY

FILE SPECIFICATIONS WORKSHEET
PAPER TAPE
SHEET 1

NCR*

Program ______ Prepared by ______

______ Date ______ Page ____ of ____

ALL SYMBOLIC REFERENCES MUST BE LEFT-JUSTIFIED AND MUST CONTAIN AT LEAST ONE ALPHABETIC CHARACTER
ALL NUMERIC ENTRIES MUST BE RIGHT-JUSTIFIED AND MUST BE ZERO-FILLED TO THE LEFT

(Shaded Boxes are Optional)

Paper Tape Format Code /09

1. **Page-Line** (1–6)
2. **File Reference** -- Enter the name to be used I O instructions and Major Functions referring to this file (7–17) F
3. **Peripheral Type Code** — See Peripheral Type List in Appendix of Language Reference Manual) (18) 4
4. **Number of Buffers** to be reserved for this file (if blank, 2 are assigned) (21)
5. **Symbolic Unit Designator** (22) P
6. **Record Length** (Maximum 255) (25)
7. **Data Format Code** See Data Format Code Chart in Appendix of Language Reference Manual) (28)
8. **Data Format Error Exit** (30–39)
9. **End of File Exit** (Source files only) (40–49)
10. Is a **Rescue Point desired** at each end of section? (**Y** or **N**) (if blank, **N** is assumed) (50)
11. Is **Rewind desired** at end of section? (**Y** or **N**) (if blank, **N** is assumed) (660 Common Trunk Reader only) (51)
12. **Mode of Processing** (Enter **F** if Control Character is first in field; **L** it last) (52)
13. **Buffer size** (53–56)
14. **Delete Digit** (74)
15. **Identification** (75–80)

PAPER TAPE (OPTIONAL)

Paper Tape Format Code /10

A1 **Page-Line** (1–6)

(7) F

A2 **User Routine after Section Open** -- See File Spec. Sheet Section of Language Reference Manual) (18–28) S

A3 **User Routine before Section Close** — (See File Spec. Sheet Section of Language Reference Manual) (29–38)

A4. If Tape-Feed characters indicate **End of Media**, how many are required? (Maximum 255) (39)

A5 Can **Section Alternation** occur after any record? (**Y** or **N**) Destination files only) (**Y** is assumed if blank) (To use **N**, see Language Reference Manual) (42)

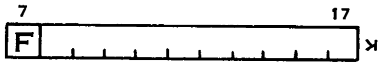
A6. **Delete Digit** (74)

A7. **Identification** (75–80)

*TRADEMARK REG. U.S. PAT. OFF.

The programmer should fill in the header, the page-and-line number (question 1), and the identification tag (positions 75-80) as defined in the NEAT/3 REFERENCE MANUAL, INTRODUCTION AND DATA, tab 3, "Programming Worksheets." The paper tape format code is preprinted on this sheet and must be punched if paper tape is used for input to the compiler.

2. FILE REFERENCE – ENTER THE NAME TO BE USED BY ALL I/O INSTRUCTIONS AND MAJOR FUNCTIONS REFERRING TO THIS FILE



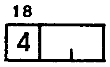
The letter F in position 7 is preprinted on the file specifications worksheet and must be punched.

Enter in positions 8 through 17 the name of the paper tape file being described for this program. Throughout the program, Paper Tape Regiment may refer to this paper tape file by using the file reference name.

The file reference must start in position 8, may be up to ten characters long, and may consist of letters or numerals. Include at least one letter in the file reference name.

Each file in a program must have its unique reference.

3. PERIPHERAL TYPE CODE – (SEE PERIPHERAL TYPE LIST IN APPENDIX OF LANGUAGE REFERENCE MANUAL)

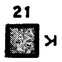


The number 4 in position 18 is preprinted and must be punched.

In positions 19 and 20, enter a 2-numeral code to identify the type of paper tape reader or paper tape punch used. The compiler uses this code to generate the proper object program for handling the paper tape on a particular type of paper tape reader or paper tape punch.

See the NEAT/3 REFERENCE MANUAL, APPENDIX, tab 1, "Peripheral Type Codes."

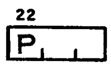
4. NUMBER OF BUFFERS TO BE RESERVED FOR THIS FILE (IF BLANK, 2 ARE ASSIGNED)



Enter in position 21 the number of input buffers or output buffers to be used by this file. This number may be any digit from 1 through 9. Two buffers normally provide the greatest efficiency.

If column 21 is left blank (space), the compiler automatically assigns two input or output buffers.

5. SYMBOLIC UNIT DESIGNATOR



The letter P in position 22 indicates a peripheral for paper media. This letter is preprinted on the file specifications worksheet and must be punched.

Enter in positions 23 and 24 a number to complete the symbolic unit designator. A 3 in position 23 specifies a paper tape.

reader; a 4 in position 23 specifies a paper tape punch. The entry in position 24 identifies individual paper tape readers or paper tape punches. If there is only one reader (or punch) in the system, this number must be 1. For example, if a system contains three paper tape readers and one paper tape punch, the symbolic unit designators for these four units are P31, P32, P33, and P41.

6. RECORD LENGTH (MAXIMUM 255) — 25

Enter in positions 25 through 27 the number of characters expected in the longest record in the file. The maximum record length is 255 characters, except when Paper Tape Regiment is used to input paper tape, in which case the record length is limited to 50 characters.

7. DATA FORMAT CODE (SEE DATA FORMAT CODE CHART IN APPENDIX OF LANGUAGE REFERENCE MANUAL) — 28

Enter in positions 28 and 29 the paper tape format of the file. Enter 20 to specify translation to or from the Century standard paper tape code, or enter 2D to specify translation to or from a user-defined code set.

8. DATA FORMAT ERROR EXIT — 30 39 X

Leave this optional entry blank (spaces) when using Paper Tape Regiment. Otherwise, a user routine to which control is transferred when data format errors are detected during translation may be entered here.

9. END OF FILE EXIT (SOURCE FILES ONLY) — 40 49 X

Leave this entry blank (spaces) if this file specifications worksheet describes a destination file, or if Paper Tape Regiment (major function) is used to read a source file. Otherwise, enter the reference name of the user routine to which control is transferred when end-of-data is reached.

NOTE

Once an end-of-file exit is taken, the last record in the input buffer is no longer accessible to the program.

10. IS A RESCUE POINT DESIRED AT EACH END OF SECTION? (Y OR N) (IF BLANK, N IS ASSUMED) — 50

Enter Y in position 50 to specify a rescue dump when this file reaches an end-of-section. Enter N, or leave position 50 blank (space) if no rescue dump is to be initiated by this file.

11. IS REWIND DESIRED AT END OF SECTION? (Y OR N) (IF BLANK, N IS ASSUMED) (660 COMMON TRUNK READER ONLY)

51 □

The entry in position 51 is only meaningful if a source file is being processed on a 660 Paper Tape Reader. Enter a Y in position 51 if an automatic rewind is desired at the end of a file section (end of reel). Enter N or leave this entry blank if no automatic rewind is required.

12. MODE OF PROCESSING (ENTER F IF CONTROL CHARACTER IS FIRST IN FIELD, L IF LAST)

52 □

Enter F in position 52 if the control characters for a data field precede the data characters in that field; enter L in position 52 if the control characters follow the data characters.

13. BUFFER SIZE

53 □□□□ 56

Enter in positions 53 through 56 the desired length for input or output buffers. This length must be at least 0020 characters. Two 255-character buffers usually provide maximum operating efficiency when processing paper tape through the use of Paper Tape Regiment or Paper Tape Out.

PAPER TAPE [OPTIONAL]

A1 PAGE-LINE

1 □□□□□□ 6

7 F

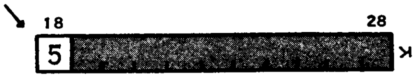
Complete the optional portion of the file specifications worksheet only if one or more of the options listed in this portion are required.

If the optional portion is used, the programmer must fill in the page-line number (positions 1-6), the delete digit (position 74), and the identification (positions 75 through 80) as defined in the NEAT/3 REFERENCE MANUAL, INTRODUCTION AND DATA, tab 3, "Programming Worksheets."

The page-line number for this optional portion must sequentially follow the page-line number for the mandatory portion of this file specifications worksheet.

If punched paper tape is used for input to the compiler, the paper tape format code preprinted on the optional portion of this file specifications worksheet must be punched before the page-line number.

The letter F in position 7 is preprinted on the file specifications worksheet and must be punched.



A2 USER ROUTINE AFTER SECTION OPEN – (SEE FILE SPEC. SHEET SECTION OF LANGUAGE REFERENCE MANUAL)

The number 5 in position 18 is preprinted on the worksheet and must be punched.

Enter in positions 19 through 28 the reference name of the user's routine to be given control immediately after the second and each subsequent section of the file is opened. Control is also transferred to this user's routine after the first section is opened only if an OPEN instruction is used for this file.

Code a RELINK instruction as the last instruction in the user's routine to return control to the software.

Leave this entry blank if no user's intervention is required after opening file sections.

A3 USER ROUTINE BEFORE SECTION CLOSE – (SEE FILE SPEC. SHEET SECTION OF LANGUAGE REFERENCE MANUAL)

Enter in positions 29 through 38 the reference name of the user's routine to be given control before each section is closed - with the exception of the last section. Control is also transferred to this user's routine before the last section is closed if a CLOSE instruction is used for this file.

Code a RELINK instruction as the last instruction in the user's routine to return control to the software.

Leave this entry blank if no user's intervention is required before closing file sections.

A4 IF TAPE-FEED CHARACTERS INDICATE END OF MEDIA, HOW MANY ARE REQUIRED? (MAXIMUM 255)

The user may wish to indicate end-of-media by a number of adjacent tape feed (trailer) characters. To exercise this option, enter in positions 39 through 41 the number of adjacent tape feed characters which are to indicate end-of-media. This number may range from 001 through 255, inclusive.

The punch configuration for the tape feed character (leader and trailer) is explained in the NEAT/3 REFERENCE MANUAL, FILES, tab 1, "Punch Paper Tape File," under Paper Tape Format.

A5 CAN SECTION ALTERNATION OCCUR AFTER ANY RECORD? (Y OR N) (DESTINATION FILES ONLY) (Y IS ASSUMED IF BLANK) (TO USE N, SEE LANGUAGE REFERENCE MANUAL)

42
▨ K

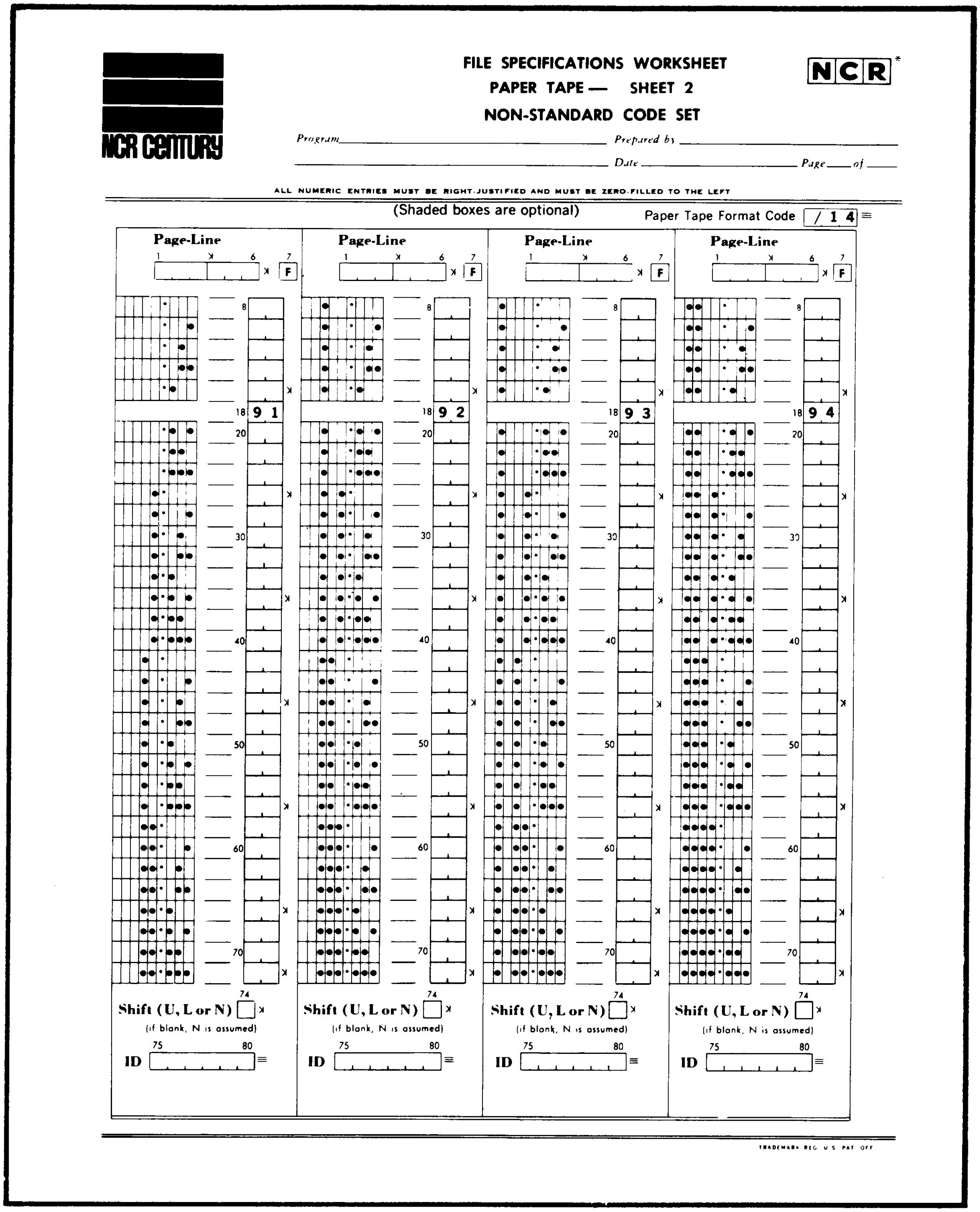
The optional entry in position 42 is used for destination files only.

A "Y" entry causes section alternation to occur after detection of the approaching end-of-media signal from the punch. Alternation can occur after any record.

An "N" should be entered if the programmer prefers to delay section alternation until the end of a logical group of records. This causes the $FILSTATUS flag to be set to a binary 1. The programmer must then test this flag (accessing it as FileReference.$FILSTATUS) before outputting the last record of the logical group. If the flag is set, he must change its value to a binary 2 to permit section alternation following the last record of the group.

File Specifications Worksheet -- Nonstandard Code Set

This form consists of two pages and must be filled in to specify code sets other than the NCR Century standard paper tape code set. Only the characters used in the code set need to be specified. The two pages cover every possible hole configuration in an 8-channel paper tape and allow for upshift and downshift characters. Each worksheet contains four columns, each with its own page-line number.



NCR CENTURY

FILE SPECIFICATIONS WORKSHEET
PAPER TAPE — SHEET 2
NON-STANDARD CODE SET

NCR*

Program ____________ Prepared by ____________
____________ Date ____________ Page ____ of ____

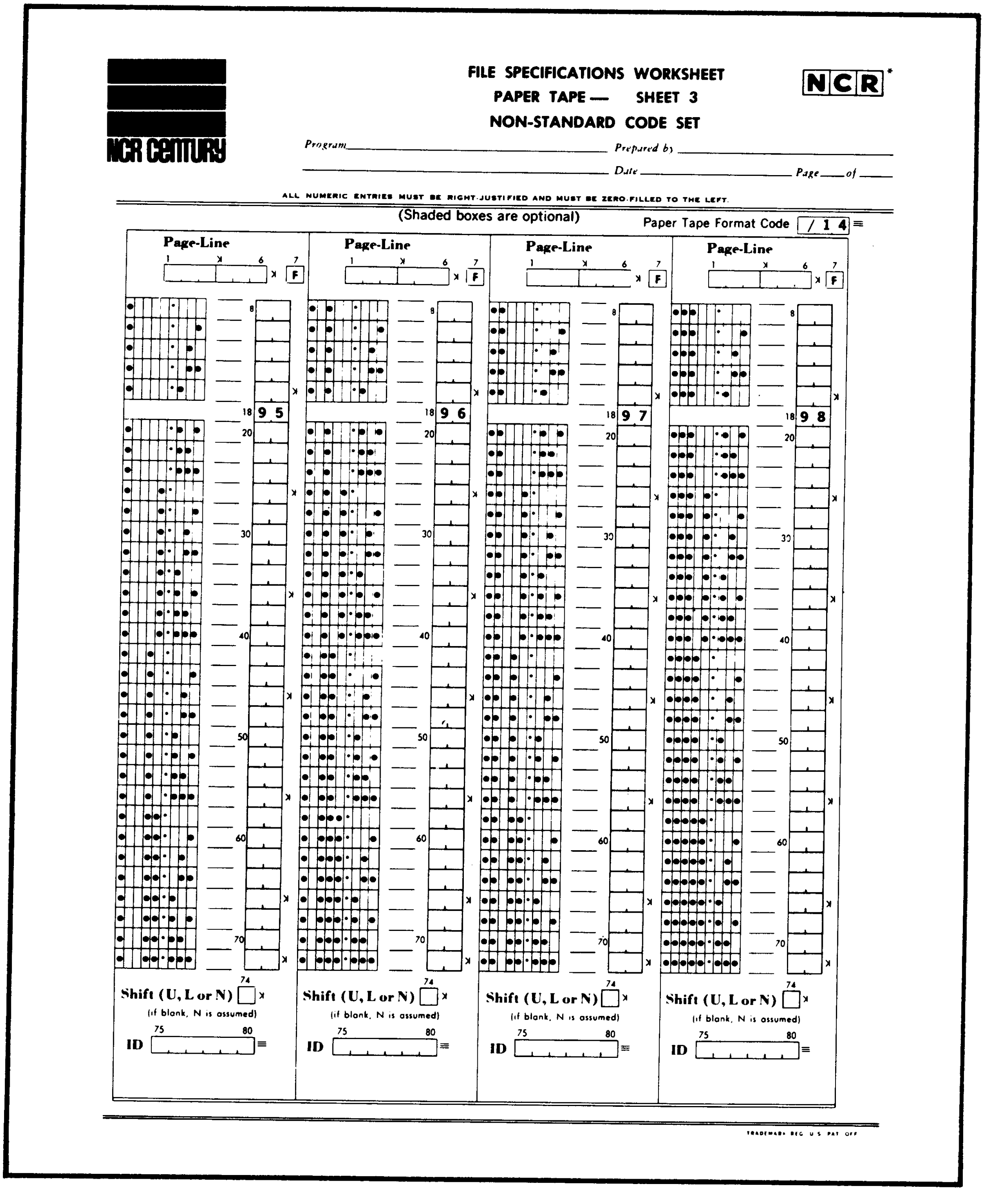
ALL NUMERIC ENTRIES MUST BE RIGHT-JUSTIFIED AND MUST BE ZERO-FILLED TO THE LEFT

(Shaded boxes are optional)

Paper Tape Format Code / 1 4

Page-Line 1–6 □ 7 F	Page-Line 1–6 □ 7 F	Page-Line 1–6 □ 7 F	Page-Line 1–6 □ 7 F
18 9 1	18 9 2	18 9 3	18 9 4
Shift (U, L or N) 74 □ (if blank, N is assumed)	Shift (U, L or N) 74 □ (if blank, N is assumed)	Shift (U, L or N) 74 □ (if blank, N is assumed)	Shift (U, L or N) 74 □ (if blank, N is assumed)
ID 75–80 □	ID 75–80 □	ID 75–80 □	ID 75–80 □

TRADEMARK REG U S PAT OFF



NCR CENTURY

FILE SPECIFICATIONS WORKSHEET
PAPER TAPE — SHEET 3
NON-STANDARD CODE SET

NCR*

Program ______________ Prepared by ______________
______________ Date ______________ Page ____ of ____

ALL NUMERIC ENTRIES MUST BE RIGHT-JUSTIFIED AND MUST BE ZERO-FILLED TO THE LEFT.

(Shaded boxes are optional)

Paper Tape Format Code / 1 4

Page-Line 1 5 6 7 F — 18 9 5 — Shift (U, L or N) □ (if blank, N is assumed) — ID

Page-Line 1 5 6 7 F — 18 9 6 — Shift (U, L or N) □ (if blank, N is assumed) — ID

Page-Line 1 5 6 7 F — 18 9 7 — Shift (U, L or N) □ (if blank, N is assumed) — ID

Page-Line 1 5 6 7 F — 18 9 8 — Shift (U, L or N) □ (if blank, N is assumed) — ID

TRADEMARK REG U S PAT OFF

The programmer specifies a nonstandard code set by relating individual paper tape hole configurations to specific internal NCR Century data characters or control characters. To permit the preparation (punching) of the nonstandard code set specifications for input to the compiler, each data or control character must be represented on the worksheets as two hexadecimal characters.

Following is an explanation of the representation and the meaning of data characters and control characters.

- Hexadecimal Representation of NCR Century Data Characters

HEXADECIMAL REPRESENTATION OF CENTURY DATA CHARACTERS

B4–B1 → B8–B5 ↓		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0000	0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
0001	1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
0010	2	SP	!	"	#	$	%	&	'	(	)	*	+	,	–	.	/
0011	3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
0100	4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
0101	5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	Λ	_
0110	6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
0111	7	p	q	r	s	t	u	v	w	x	y	z	{	¦	}	~	DEL

L.H. CHARACTER

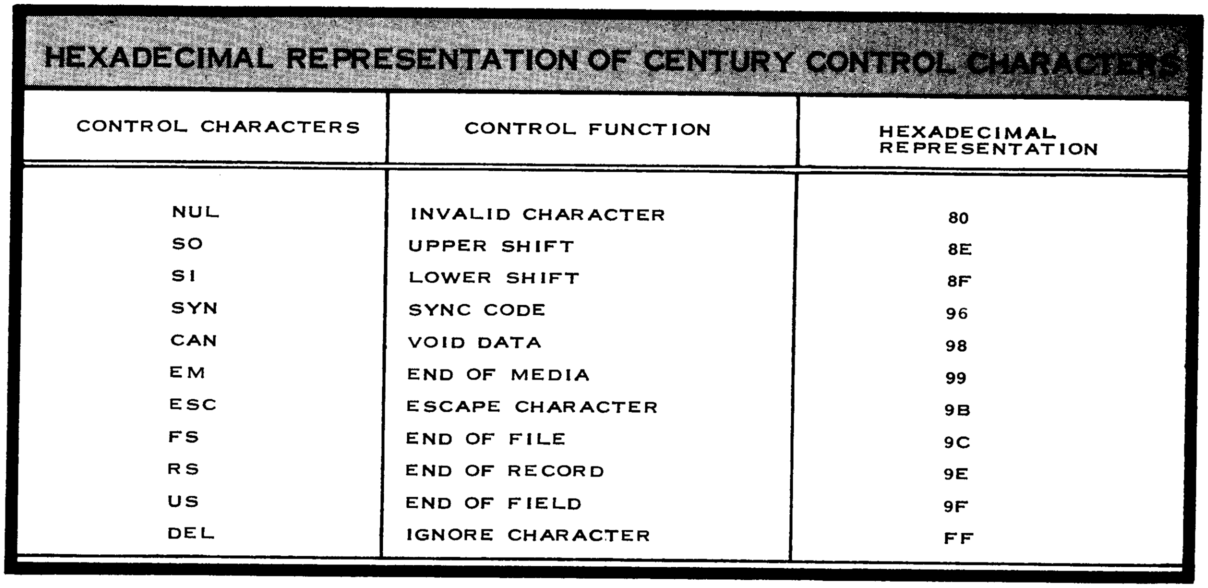
R.H. CHARACTER

To find the two proper hexadecimal characters for representing an NCR Century data character, locate the desired character in its non-shaded square on the above chart. The shaded box to the left of the data character (on same line) contains the left-hand hexadecimal character, and the shaded box above the data character (same column) contains the right-hand hexadecimal character.

Following are some examples of NCR Century data characters and their corresponding hexadecimal representation.

NUL	=	00	+	=	2B	R	=	52
SYN	=	16	2	=	32	Z	=	5A
&	=	26	7	=	37	m	=	6D

- Hexadecimal Representation of Control Characters



HEXADECIMAL REPRESENTATION OF CENTURY CONTROL CHARACTERS

CONTROL CHARACTERS	CONTROL FUNCTION	HEXADECIMAL REPRESENTATION
NUL	INVALID CHARACTER	80
SO	UPPER SHIFT	8E
SI	LOWER SHIFT	8F
SYN	SYNC CODE	96
CAN	VOID DATA	98
EM	END OF MEDIA	99
ESC	ESCAPE CHARACTER	9B
FS	END OF FILE	9C
RS	END OF RECORD	9E
US	END OF FIELD	9F
DEL	IGNORE CHARACTER	FF

The hexadecimal representations of control characters (which perform the control functions above) are recognized by the software. The control characters have the same name as some of the data characters. (Whether a character is considered as a control character or a data character depends on its hexadecimal representation in the nonstandard code set specifications.) The fixed 2-character hexadecimal representation for the control characters is shown in the above chart. (The hexadecimal representation of data characters is shown in the chart on page 9 of this publication.)

The software considers a control character (except NUL) preceded by an ESC (Escape) control character as a data character and not as a control character. The control character (except NUL) preceded by the ESC character becomes part of the data being read. (A NUL character may be punched in paper tape, but can never be input as a data character.)

Worksheet Entries - Nonstandard Code Set

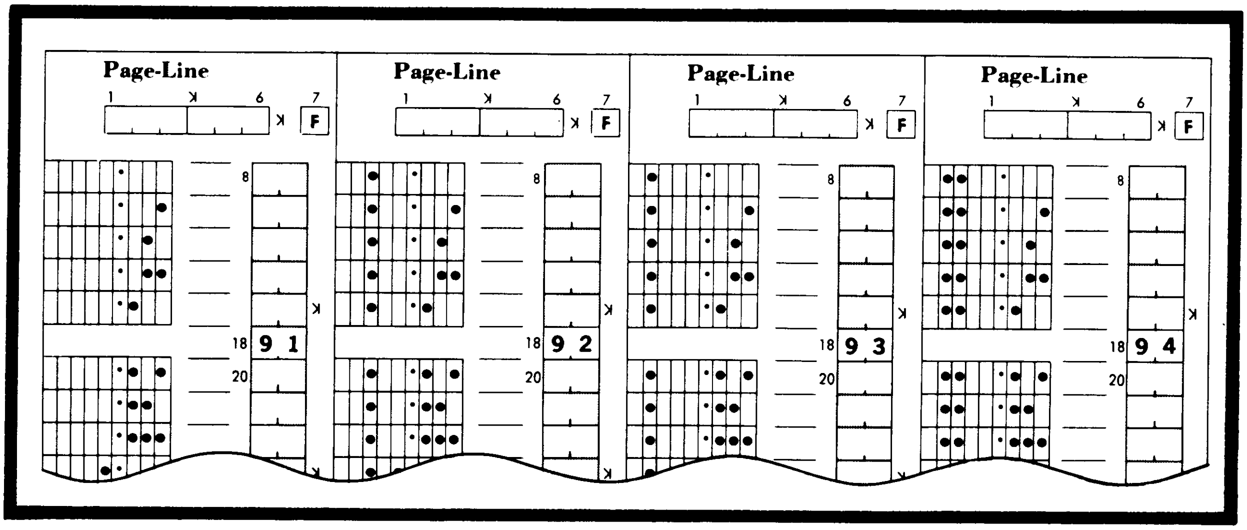
The following is a brief description of the entries on the file specifications worksheets - sheet 2 and sheet 3. Each of these worksheets has four columns that represent individual source lines with unique page-line numbers. If the nonstandard code set uses less than eight paper tape channels, only one worksheet (sheet 2) is required.

If the programmer includes upper and lower shift characters in a nonstandard code set, he must fill in two sets of worksheets. Two different data characters may be represented by the same hole configuration, with the shift mode being the only differentiating factor. Control characters must be defined in both shift modes, and each control character must have the same hole configuration in the upper shift mode as in the lower shift mode.

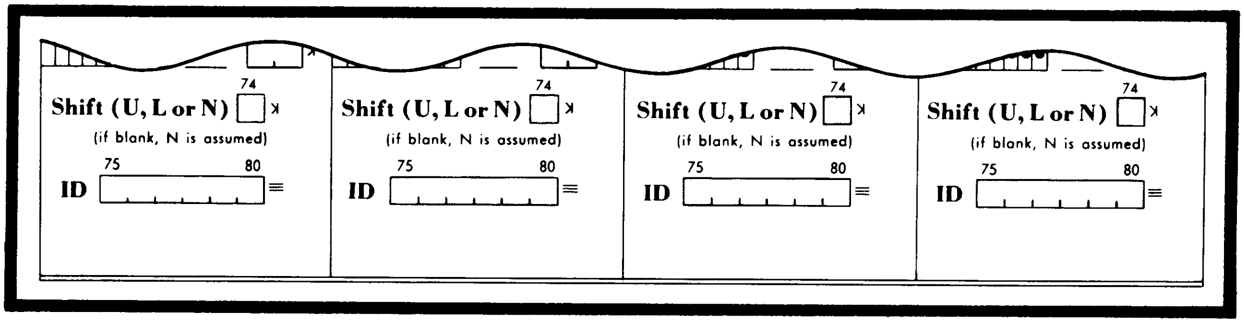
The programmer must fill in the header, the page-line numbers (positions 1 through 6) and the identification tags (positions 75 through 80) of the columns being used, as defined in INTRODUCTION AND DATA, tab 3.

The page-line numbers on the nonstandard code set worksheets must immediately follow the page-line number of the associated file specifications worksheet (or of the optional portion of the file specifications worksheet, if used).

If punched paper tape is used for input to the compiler, the paper tape format code preprinted on the file specifications worksheet must be punched before the first page-line number of the columns that are used for defining the code set.



Every column on the worksheets contains the preprinted character F in position 7 and a unique identification number (91 through 98) in positions 18 and 19. These preprinted entries must be punched in the source lines for the columns that are used for the nonstandard code set definition.



To define a one-shift nonstandard code set, enter the character N in position 74 of the columns being used, or leave position 74 blank.

To define a code set containing lower shift and upper shift characters, enter L or U in position 74. An L indicates that the paper tape configurations defined in the column are to be interpreted as lower shift characters; a U indicates that the paper tape configurations defined in the column are to be interpreted as upper shift characters.

Example of Nonstandard Code Set

A set of paper tape characters and their intended meaning are charted on the following page. The file specifications worksheets defining this nonstandard code set are shown on the subsequent pages. (When considering this example, refer to Hexadecimal Representation of Control Characters and Hexadecimal Representation of NCR Century Data Characters in this publication.

NONSTANDARD CODE SET — EXAMPLE

LOWER SHIFT CHARACTERS

CHANNEL 8	7	6	5	4	•	3	2	1	MEANING
					•				NUL
					•			●	LOWER SHIFT
					•		●		UPPER SHIFT
					•	●			END OF FIELD
					•	●		●	END OF RECORD
					•	●	●		END OF MEDIA
					•	●	●	●	END OF FILE
				●	•	●	●	●	SYNC CODE
	●	●	●	●	•	●	●	●	DELETE
			●		•				Ø (SPACE)
			●		•			●	A
			●		•		●		B
			●		•		●	●	C
			●		•	●			D
			●		•	●		●	E
			●		•	●	●		F
			●		•	●	●	●	G
			●	●	•				H
			●	●	•			●	I
	●				•			●	J
	●				•		●		K
	●				•		●	●	L
	●				•	●			M
	●				•	●		●	N
	●				•	●	●		O
	●				•	●	●	●	P
	●			●	•				Q
	●			●	•			●	R
		●			•		●		S
		●			•		●	●	T
		●			•	●			U
		●			•	●		●	V
		●			•	●	●		W
		●			•	●	●	●	X
		●		●	•				Y
		●		●	•			●	Z

UPPER SHIFT CHARACTERS

CHANNEL 8	7	6	5	4	•	3	2	1	MEANING
					•				NUL
					•			●	LOWER SHIFT
					•		●		UPPER SHIFT
					•	●			END OF FIELD
					•	●		●	END OF RECORD
					•	●	●		END OF MEDIA
					•	●	●	●	END OF FILE
				●	•	●	●	●	SYNC CODE
	●	●	●	●	•	●	●	●	DELETE
			●		•				0
			●		•			●	1
			●		•		●		2
			●		•		●	●	3
			●		•	●			4
			●		•	●		●	5
			●		•	●	●		6
			●		•	●	●	●	7
			●	●	•				8
			●	●	•			●	9

NCR CENTURY

FILE SPECIFICATIONS WORKSHEET
PAPER TAPE — SHEET 2
NON-STANDARD CODE SET

NCR*

*Program*____________ *Prepared by*____________

____________ *Date*________ *Page*____ *of*____

ALL NUMERIC ENTRIES MUST BE RIGHT-JUSTIFIED AND MUST BE ZERO-FILLED TO THE LEFT.

(Shaded boxes are optional)

Paper Tape Format Code / 1 4 ≡

Page-Line 1 5 6 7: 0 1 8 0 0 0 F

8	NUL	8 0
	SI	8 F
	SO	8 E
	US	9 F
18		9 1
20	RS	9 E
	EM	9 9
	FS	9 C
30		
40	SYN	9 6
	Ø	2 0
	A	4 1
	B	4 2
	C	4 3
50	D	4 4
	E	4 5
	F	4 6
	G	4 7
	H	4 8
60	I	4 9
70		

Shift (U, L or N) 74 L
(if blank, N is assumed)

ID 75 80 ≡

Page-Line 1 5 6 7: 0 1 8 0 1 0 F

8		
	S	5 3
	T	5 4
	U	5 5
18		9 2
20	V	5 6
	W	5 7
	X	5 8
	Y	5 9
	Z	5 A
30		
40		
50		
60		
70		

Shift (U, L or N) 74 L
(if blank, N is assumed)

ID 75 80 ≡

Page-Line 1 5 6 7: 0 1 8 0 2 0 F

8		
	J	4 A
	K	4 B
	L	4 C
	M	4 D
18		9 3
20	N	4 E
	O	4 F
	P	5 0
	Q	5 1
	R	5 2
30		
40		
50		
60		
70		

Shift (U, L or N) 74 L
(if blank, N is assumed)

ID 75 80 ≡

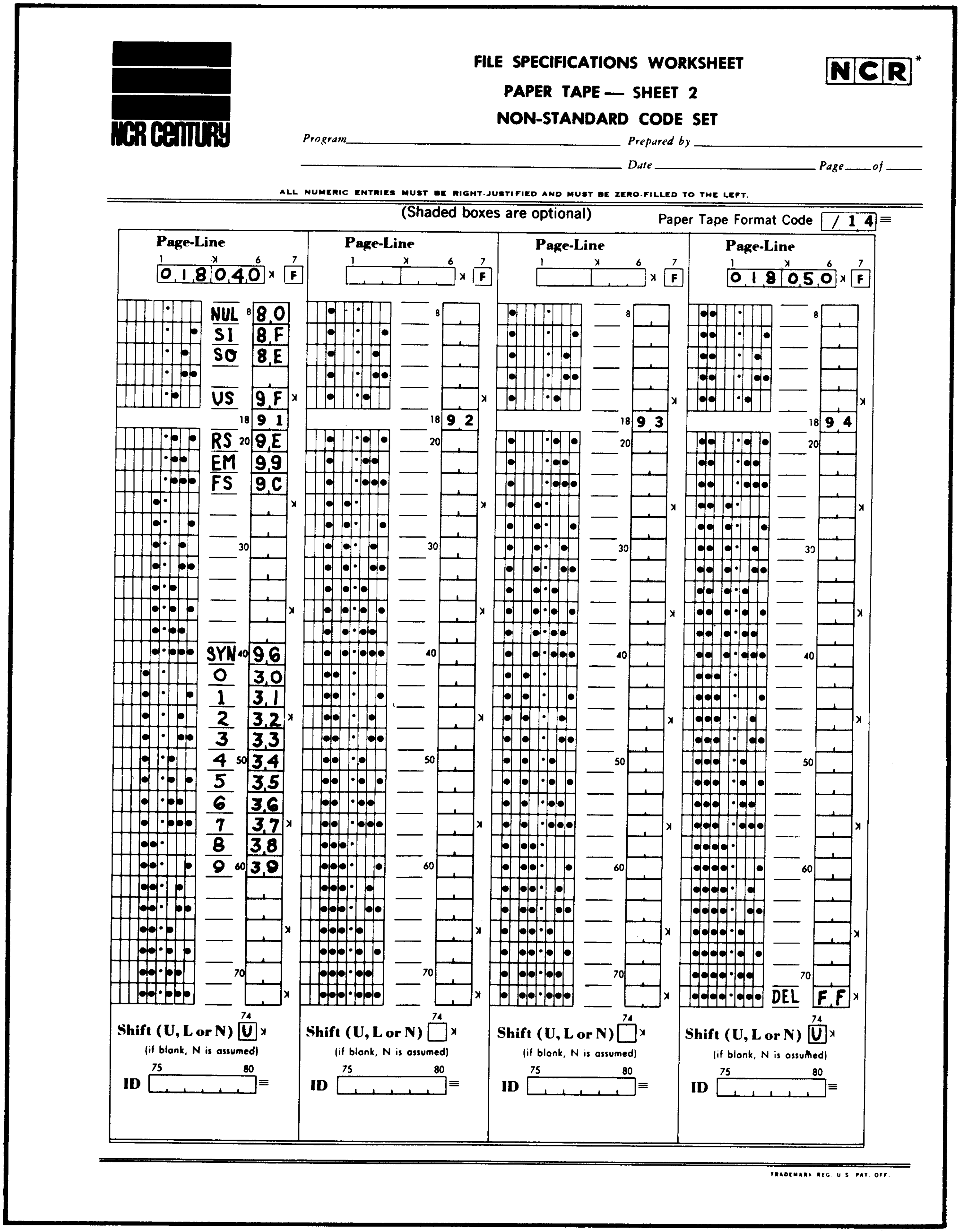
Page-Line 1 5 6 7: 0 1 8 0 3 0 F

8		
18		9 4
20		
30		
40		
50		
60		
70		
	DEL	F F

Shift (U, L or N) 74 L
(if blank, N is assumed)

ID 75 80 ≡

TRADEMARK REG. U.S. PAT. OFF.



NCR CENTURY

FILE SPECIFICATIONS WORKSHEET
PAPER TAPE — SHEET 2
NON-STANDARD CODE SET

NCR*

Program ______________________ *Prepared by* ______________________

______________________ *Date* ______________________ *Page* ____ *of* ____

ALL NUMERIC ENTRIES MUST BE RIGHT-JUSTIFIED AND MUST BE ZERO-FILLED TO THE LEFT.

(Shaded boxes are optional)

Paper Tape Format Code /14

	Page-Line	Page-Line	Page-Line	Page-Line
1–6	018040			018050
7	F	F	F	F

Line	Sheet 1 character	Sheet 1 code	Sheet 2 code	Sheet 3 code	Sheet 4 character	Sheet 4 code
8	NUL	8,0				
	SI	8,F				
	SO	8,E				
	US	9,F				
18		9 1	9 2	9 3		9 4
20	RS	9,E				
	EM	9,9				
	FS	9,C				
40	SYN	9,6				
	0	3,0				
	1	3,1				
	2	3,2				
	3	3,3				
50	4	3,4				
	5	3,5				
	6	3,6				
	7	3,7				
	8	3,8				
60	9	3,9				
70					DEL	F,F

	Sheet 1	Sheet 2	Sheet 3	Sheet 4
Shift (U, L or N) (if blank, N is assumed)	U			U
ID (75–80)				

TRADEMARK REG. U.S. PAT. OFF.

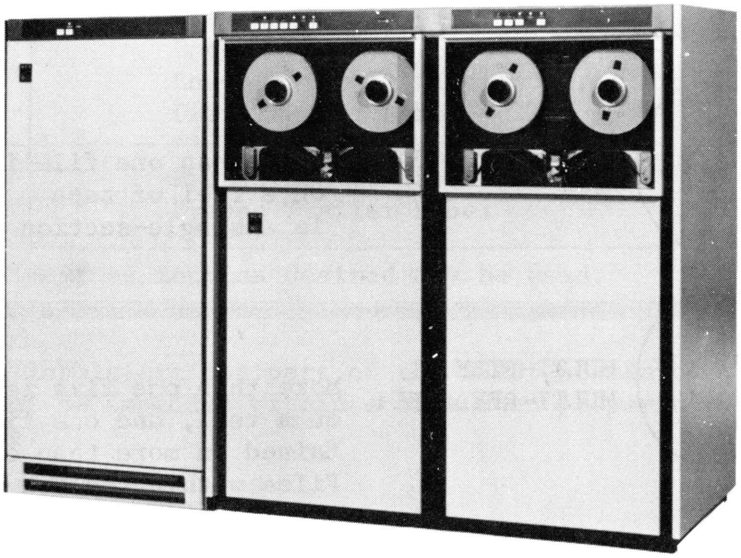
MAGNETIC TAPE FILES

A wide range of magnetic tape handlers featuring various recording densities and transfer rates are available for use with NCR Century Systems. Within this range are handlers capable of reading and recording on 7-channel or 9-channel tape, using either the non-return-to-zero (NRZ) or the phase-modulated technique of recording.

NCR provides such a wide range of tape handlers to permit the user to base his peripheral choice not only on system performance-to-cost ratios, but also on the desire to interchange tapes with other equipment manufacturers. The NCR Century software is designed to create and read tapes that conform with the adopted standard of the United States of America Standards Institute (USASI). Further, the GET and PUT instructions are capable of translating data input from, or output to, tapes in any of the following external code sets:

- NCR 315
- IBM Binary-Coded-Decimal (BCD)
- IBM Extended-Binary-Coded-Decimal Interchange Code (EBCDIC)

The software is also designed to automatically handle tapes recorded in NCR 315 format, that is, tapes with 315 labels. The format of tapes in external code sets other than NCR 315 must comply with USASI standards.



FILE DESCRIPTION

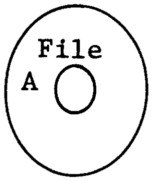
Reel-File Relationship

One or more files may be placed on a reel of tape, or one file may occupy many reels of tape. When one file occupies one reel of tape, the file is considered to be a single-section file. In this instance, a file section equates to a reel (section 1 is on reel 1). When one file extends over two or more reels of tape, the file is considered a multi-section file; here again, a file section equates to a reel (section 1 is on reel 1, section 2 is on reel 2, etc.).

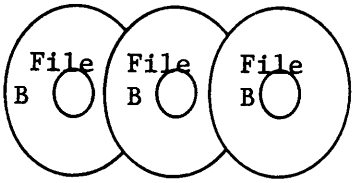
When more than one file appears on the same reel, each file is considered a single-section file and the reel is considered a multi-file reel. A file section no longer equates to a reel, since more than one file appears on the same reel (file 1 and file 2 on reel 1).

When more than one file appears on the same reel, and one file extends over two or more reels, the group of reels containing those files is considered a multi-file, multi-reel set.

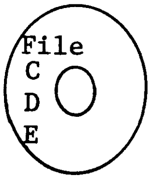
REEL—FILE RELATIONSHIP



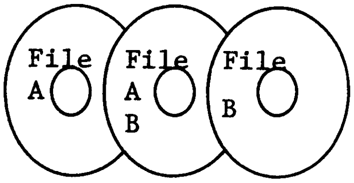
SINGLE-REEL FILE: One file is contained on one reel of tape. This is a single-section file.



MULTI-REEL FILE: One file is contained on two or more reels of tape. This is a multi-section file.



MULTI-FILE REEL: More than one file is contained on a reel of tape. Each file is a single-section file.



MULTI-FILE, MULTI-REEL SET: More than one file is contained on a reel, and one file is contained on more than one reel. Files within the set may be single- or multi-section files.

When a multi-file, multi-reel set is used, the programmer can indicate on the file specification sheets a six-character identification code to identify the set. On a destination set, the software insures that no files with a different set identification are placed on a reel within this set. The software uses the set identification code on a source set to insure that all reels mounted during processing are part of the desired set.

Labels

The I/O software is designed to create and process labels that conform to the USASI standard. These labels are special 80-character blocks that provide the software with a means of identifying reels, sets, files, and sections of files stored on magnetic tape. Labels also provide the software with a means of embedding special information (such as rescue dumps) within a data file. This capability allows a normal data file to also be used as the standard rescue file in a run, and allows the software to bypass rescue information when presenting data from that file to a program.

All labels are identified by their first four characters. The first three characters indicate the label type and the fourth character indicates the number of the label within that type; e.g., HDR2 = Second File Header Label. The following is a list of the label identifiers, their meanings, and the number that may be used per reel or file according to the USASI standard:

LABEL TYPES

IDENTIFIER	MEANING	MAXIMUM NUMBER
VOL	Volumn Header Label	1 per reel
UVL	User Volume Header Label	9 per reel
HDR	File Header Label	9 per section
UHL	User Header Label	* per section
EOV	End-of-Volume Trailer Label	1 per reel
EOF	End-of-File Trailer Label	1 per file
UTL	User Trailer Label	* per section
BYH	Bypass Header Label	1 per rescue
BYT	Bypass Trailer Label	1 per rescue

* - No set maximum; as many as desired may be used.

The BYH and BYT labels are not part of the USASI standard. Therefore, rescue dumps may not be embedded within a file when preparing a USASI standard interchange tape.

- VOL - Volume Header Label

Each reel of magnetic tape is considered a volume and must contain a Volume Header Label to identify it. VOL labels, placed as the first data on tape by the magnetic tape initializer utility routine, contain the following information in the indicated format:

VOL LABEL FORMAT

FIELD NAME	REL. POS.	LENGTH	DESCRIPTION
Label identifier	0	3	Contains "VOL" to identify this as a Volume Header Label.
Label number	3	1	Contains decimal "1" to identify this as the first and only Volume Header Label.
Volume serial number	4	6	Contains information supplied by the programmer to uniquely identify this reel. Any "a" character from the following code chart below may be used.
Accessibility code	10	1	Contains the accessibility code supplied by the programmer. Any "a" character may be used. At present, software makes no use of this field.
Reserved	11	26	Contains space characters. This field, designated as optional by USASI, is unused by the NCR Century software.
Owner's identification	37	14	Contains data supplied by the programmer to identify the owner of the volume. Any "a" characters may be used.
Reserved	51	28	Contains space characters. This field is reserved for future standardization.
Label standard level	79	1	Contains either decimal "1" if this tape follows the the USASI standard, or a space character if this tape does not follow the USASI standard. This field is set to "1" by the initializer.

NCR CENTURY CODE CHART

B_4-B_1 →		0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
B_8-B_5 ↓		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0000	0	NUL	SOH	STX	ETX	EOT	ENQ	ACK	BEL	BS	HT	LF	VT	FF	CR	SO	SI
0001	1	DLE	DC1	DC2	DC3	DC4	NAK	SYN	ETB	CAN	EM	SUB	ESC	FS	GS	RS	US
0010	2	SP	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
0011	3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
0100	4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
0101	5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	Λ	_
0110	6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
0111	7	p	q	r	s	t	u	v	w	x	y	z	{	¦	}	~	DEL

"a" characters are any of the 58 characters occupying the unscreened center four rows of the above chart.

- UVL - User Volume Header Label

User Volume Header Labels are optional. The NCR Century software will not present them to the program, but will index past them during processing. Up to nine UVL labels may be placed on the tape at the time it is initialized, if desired for interchange purposes. The following format must be followed in creating these labels:

UVL LABEL FORMAT

FIELD NAME	REL. POS.	LENGTH	DESCRIPTION
Label identifier	0	3	Contains "UVL" to identify this as a User Volume Header Label.
Label number	3	1	Contains decimal "1" for the first UVL label, decimal "2" for the second, etc.
Data	4	76	Contains any information desired.

- HDR - File Header Label

The required number of File Header Labels, as defined by the USASI standard, is one; however, up to nine may be used. The NCR Century software uses two which are automatically created by the software when a new file or file section is opened, and are automatically read each time an existing file or file section is opened. If an interchange tape with more than two HDR labels is input, the software indexes past all but the first two. The information contained in the first two labels is used by the software to locate a desired file, protect a current file, and identify file sections. The two HDR labels (HDR1 and HDR2), which have the following formats, are available for examination by the program as described later in this publication under the heading "Creating and Reading Labels".

HDR1 LABEL FORMAT

FIELD NAME	REL. POS.	LENGTH	DESCRIPTION
Label identifier	0	3	Contains "HDR" to identify this as a File Header Label.
Label number	3	1	Contains "1" for the first File Header Label.
File name	4	17	Contains the name of the file as indicated on the file specification sheets. The first 10 characters contain the file name and the last 7 characters contain spaces.
Set identification	21	6	Contains the set identification code as specified on the file specification sheets. Any "a" characters may be used.
File section number	27	4	Contains the volume (section) number of the volume within a file. "0001" for the first section, "0002" for the second, etc.
File sequence number	31	4	Contains the sequence number of a file within a volume set. "0001" for the first file, "0002" for the second, etc.
Generation number	35	4	Contains "0001". This field is not used by the NCR Century software.
Generation version	39	2	Contains spaces. This field is not used by the NCR Century software.

CONTINUED

HDR1 LABEL FORMAT (CONTD.)

FIELD NAME	REL. POS.	LENGTH	DESCRIPTION
Creation date	41	6	Contains the virtual date specified when the file was created. The date is in the following format - a space character followed by two numeric characters that represent the year, followed by three numeric characters that represent the sequence day within the year. (⍁68001 = Jan. 1, 1968)
Expiration date	47	6	Contains the date the file expires. This date is in the same format as the creation date.
Accessibility code	53	1	Contains the accessibility code specified on the file specification sheets. At present, software makes no use of this field.
Block count	54	6	Contains zeros. (Note: The block count does not appear in this label; however, the software does keep track of the block count and stores that count in the E-O-V or E-O-F label when a file is created.
System Code	60	13	Contains spaces. This field is not used by the NCR Century software.
Reserved	73	7	Contains spaces.

HDR2 LABEL FORMAT

FIELD NAME	REL. POS.	LENGTH	DESCRIPTION
Label identifier	0	3	Contains "HDR".
Label number	3	1	Contains decimal "2" for the second File Header Label.
Record format	4	1	Contains a character to indicate the type of records in the file: F = Fixed-length; V = Variable-length with record in binary; D = Variable-length with record size in decimal; U = Undefined.
Block length	5	5	Contains a decimal number that indicates the maximum number of characters in a block.
Record length	10	5	Contains a decimal number that indicates the number of characters in a fixed-length record or the maximum number of characters in a variable-length record.
Reserved	15	35	Contains spaces.
Buffer offset	50	2	Contains a decimal number that indicates the number of characters to be ignored at the beginning of a block. The software presents records beginning with the character following the offset.
Reserved	52	28	Contains spaces

- UHL - User Header Label

Optional labels (called User Header Labels) may be placed on the tape immediately following the File Header Labels, as described later in this publication under the heading "Creating and Reading Labels". The USASI standard permits the user to create as many UHL labels as desired; however, they must conform to the following format:

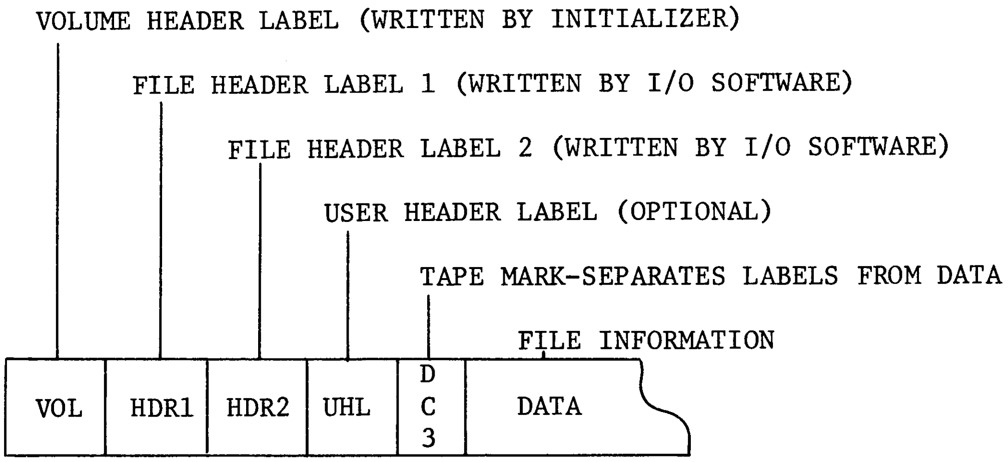
UHL LABEL FORMAT

FIELD NAME	REL. POS.	LENGTH	DESCRIPTION
Label identifier	0	3	Contains "UHL" to identify this as a User Header Label.
Label number	3	1	Contains a character to indicate the label sequence. Any "a" character is acceptable.
Data	4	76	Contains any information desired.

- Tape Marks

The USASI Character "DC3" (BINARY 00010011) is used as a tape mark. The software writes one tape mark to separate labels from data, one or two tape marks to separate groups of labels, and two tape marks to indicate the end of a file or the end of a reel. Since tape marks are not placed in the input buffer when they are read, the programmer need never concern himself with them.

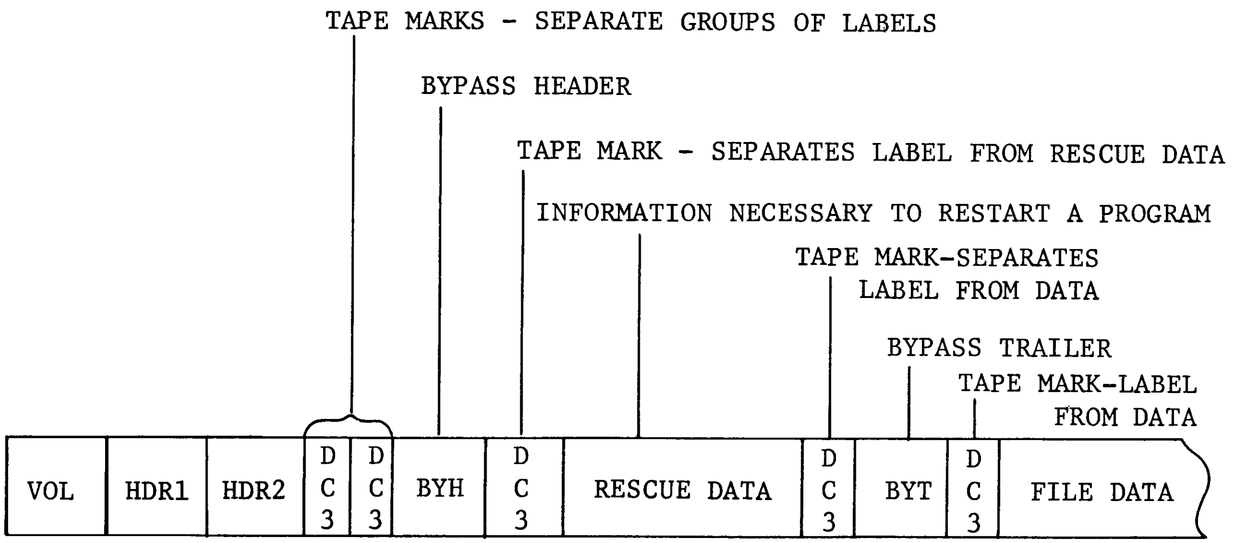
TYPICAL LABEL CONFIGURATION


VOLUME HEADER LABEL (WRITTEN BY INITIALIZER)
FILE HEADER LABEL 1 (WRITTEN BY I/O SOFTWARE)
FILE HEADER LABEL 2 (WRITTEN BY I/O SOFTWARE)
USER HEADER LABEL (OPTIONAL)
TAPE MARK-SEPARATES LABELS FROM DATA
FILE INFORMATION
VOL
HDR1
HDR2
UHL
DC3
DATA

- BYH - Bypass Header Label and BYT - Bypass Trailer Label

The Bypass Header Label precedes, and the Bypass Trailer Label follows, rescue dumps on tape. These labels, which contain the information needed to properly identify a rescue dump, are never placed in the input buffer by the software; the programmer need not concern himself about BYH or BYT Labels. However, since these labels are not part of the USASI standard, they may not appear on a USASI interchange tape. For further information on rescue dumps, see the NEAT/3 REFERENCE MANUAL, FILES, "NCR Century File Concepts".

TYPICAL LABEL CONFIGURATION SHOWING A RESCUE DUMP


TAPE MARKS - SEPARATE GROUPS OF LABELS
BYPASS HEADER
TAPE MARK - SEPARATES LABEL FROM RESCUE DATA
INFORMATION NECESSARY TO RESTART A PROGRAM
TAPE MARK-SEPARATES LABEL FROM DATA
BYPASS TRAILER
TAPE MARK-LABEL FROM DATA
VOL
HDR1
HDR2
DC3
DC3
BYH
DC3
RESCUE DATA
DC3
BYT
DC3
FILE DATA

- EOF - End-of-File Trailer Label

The software places an End-of-File Trailer Label on the tape each time a destination file or a piggyback file is closed. When encountered on a source file, it indicates that all data in the file has been processed (end-of-file).

The software then compares the data block count contained in this label with the number of data blocks input during processing. For a multi-section file, the count in this label reflects the number of data blocks in the last section only. Since the block count is for data blocks only, it does not include tape marks, label blocks (software or user), or bypass blocks. EOF labels have the same format as HDR2 labels, with the following exceptions:

EOF LABEL FORMAT

FIELD NAME	REL. POS.	LENGTH	DESCRIPTION
Label identifier	0	3	Contains "EOF".
Label number	3	1	Contains "1".
Block count	5	5	Contains the decimal number that indicates the number of blocks in the last section of a file (or the entire file if the file is only one section long).

- EOV - End-of-Volume Trailer Label

The software places an End-of-Volume Trailer Label at the end of a reel of tape when the last file on the tape extends to another reel. When encountered on a source file, it indicates that all the data in a file section has been processed (end-of-section). The software then compares the data block count contained in the label with the number of data blocks input while processing this section of the file. The count is for data blocks only and does not include tape marks, label blocks (software or user), or bypass blocks. EOV labels have the same format as HDR2 labels, with the following exceptions:

EOV LABEL FORMAT

FIELD NAME	REL. POS.	LENGTH	DESCRIPTION
Label identifier	0	3	Contains "EOV".
Label number	3	1	Contains "1".
Block count	5	5	Contains the number of data blocks in this section of the file.

- UTL - User Trailer Label

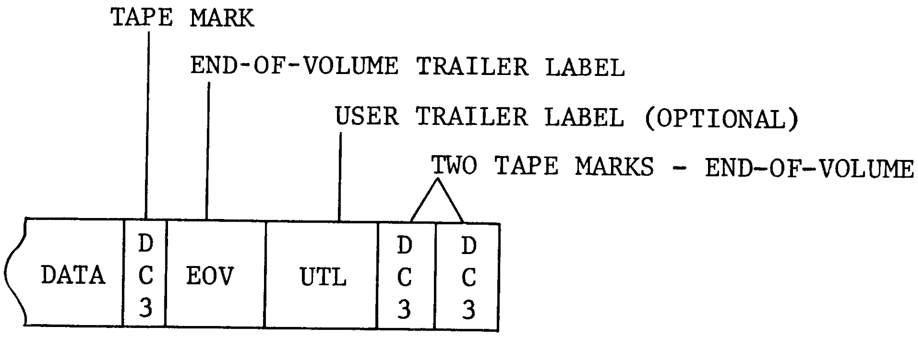
Optional labels (called User Trailer Labels) may be placed on tape immediately following EOV or EOF labels, as described later in this publication under the heading"Creating and Reading Labels". The USASI standard permits the user to create as many UTL labels as desired; however, if they are written after an EOV label, limited room exists for them since the physical end of tape is near. User Trailer Labels must conform to the following format:

UTL LABEL FORMAT

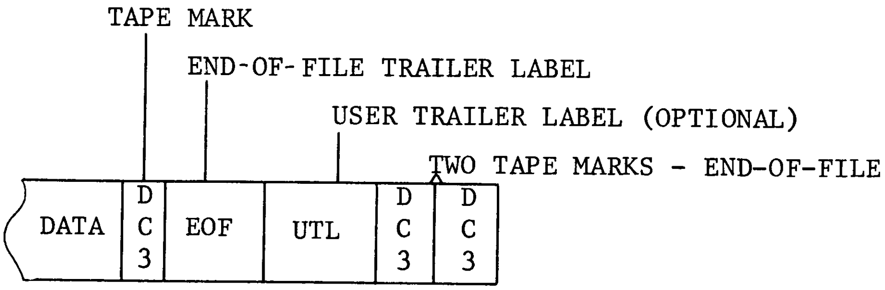
FIELD NAME	REL. POS.	LENGTH	DESCRIPTION
Label identifier	0	3	Contains "UTL" to identify this as a User Trailer Label.
Label number	3	1	Contains a character to indicate the label sequence. Any "a" character may be used.
Data	4	76	Contains any information desired.

TYPICAL LABEL CONFIGURATIONS SHOWING EOV, EOF, AND UTL

MULTI-VOLUME FILE


TAPE MARK
END-OF-VOLUME TRAILER LABEL
USER TRAILER LABEL (OPTIONAL)
TWO TAPE MARKS - END-OF-VOLUME
DATA
DC3
EOV
UTL
DC3
DC3

LAST VOLUME OF A FILE


TAPE MARK
END-OF-FILE TRAILER LABEL
USER TRAILER LABEL (OPTIONAL)
TWO TAPE MARKS - END-OF-FILE
DATA
DC3
EOF
UTL
DC3
DC3

- Beginning-of-Tape and End-of-Media Markers

Two reflective tabs are pasted on the non-recording side of the tape. One tab, the Beginning-of-Tape marker (BOT), is approximately 10 feet from the start of the reel. The other tab, the End-of-Media marker (EOM) is approximately 18 feet from the end of the tape. These marks are sensed by a photoelectric system.

The BOT, which indicates the physical position at which the recording or reading is to begin, allows approximately a ten-foot leader for loading the tape on the handler.

EOM indicates that the physical end of tape is near. When the EOM marker is encountered on a destination file, the software places an End-of-Volume (EOV) Label on the tape. If a rescue point was requested at the end-of-section, the rescue point data is placed at the beginning of the next volume (reel).

Exercise caution in writing User Trailer Labels (UTL) on a destination file when End-of-Volume (EOV) occurs, as limited space remains for these labels.

Creating and Reading Labels

All labels other than optional user labels are created automatically by the software. If the programmer wishes to create or read optional UHL or UTL labels, or examine HDR, EOV, or EOF labels, he must do so in a user routine after open or in a user routine before close. The software links to these routines after each section of the file (except the first) is opened and before each section of the file (except the last) is closed. If the programmer also desires this option for the first section of the file, he must use an OPEN instruction; if he desires this option for the last section, he must use a CLOSE instruction with the file. (For further information on OPEN and CLOSE instructions, see the NEAT/3 REFERENCE MANUAL, INSTRUCTIONS, tab 1, "OPEN Instructions" and "CLOSE Instructions".

After the software links to the user routine the program must execute the LPUT instructions to output labels to a destination or piggyback file, or the LGET instruction to input labels from a source file. For further information concerning these instructions, see the NEAT/3 REFERENCE MANUAL, INSTRUCTIONS, tab 1, "GET Instructions" and "PUT Instructions".

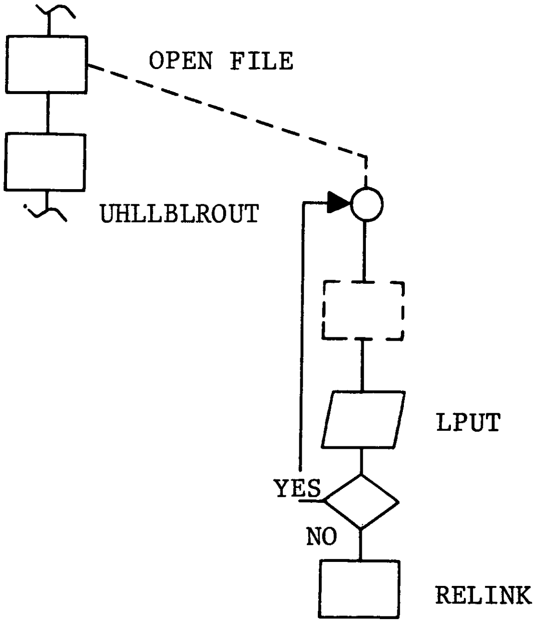
Assume for the following discussion on UHL and UTL labels that (1) the file involved is opened by an OPEN instruction; (2) that the names of both a user routine after open and a user routine before close are named on the file specification sheets; and (3) that the file involved is closed by a CLOSE instruction.

- Creating UHL Labels

  When a new file is opened using an OPEN instruction, the software writes the HDR1 and HDR2 labels on tape and then links to the user routine after open. The programmer uses the LPUT instruction within this routine to create as many UHL labels as desired. After the labels are output, a RELINK instruction (without an operand) must be executed to return control to the software. The software then writes the proper number of tape marks and returns control to the main program.

  Each time a new section is opened, the software again writes HDR1 and HDR2 labels and links to the user routine to permit construction of user labels for that section.

CREATING UHL LABELS



The software links when the first section of a file is opened by an OPEN instruction and each time a new section of the file is opened.

CREATE LABEL.

OUTPUT LABEL.

MORE LABELS?

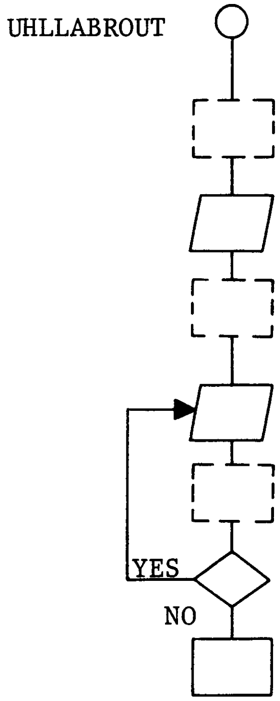
RETURN TO SOFTWARE.

- Inputting UHL Labels

When an existing file is opened using an OPEN instruction, the software reads the HDR1 label into the input buffer and then links to the user routine after open. The program may examine the contents of the HDR1 label, or it may ignore it by executing an LGET instruction to input the HDR2 label. Again, the program may examine the contents of the label or ignore it by executing another LGET instruction to input the first UHL label.

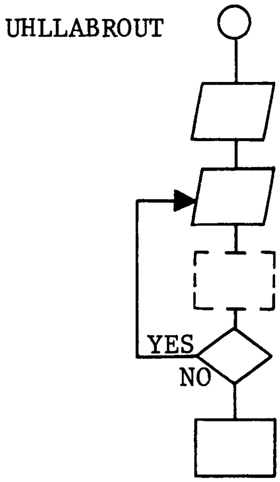
The programmer may code his routine to input all UHL labels or a portion of them. If all the labels are not input, a RELINK instruction (with no operand) must be used to return control to the software.

USER ROUTINE THAT EXAMINES HDR AND UHL LABELS



	EXAMINE HDR1 LABEL
LGET	INPUT HDR2 LABEL
	EXAMINE HDR2 LABEL
LGET	INPUT UHL LABEL OVER PREVIOUS LABEL
	EXAMINE UHL LABEL
	MORE LABELS?
RELINK	RELINK TO SOFTWARE

USER ROUTINE THAT EXAMINES UHL LABELS ONLY



LGET	INPUT HDR2 OVER HDR1
LGET	INPUT UHL LABEL OVER PREVIOUS LABEL
	EXAMINE UHL LABEL
	MORE LABELS?
RELINK	RELINK TO SOFTWARE

If all the labels (plus one) are input, a RELINK instruction is not needed. When an LGET instruction reads a tape mark (end of a group of labels), it causes an automatic relink to the software.

- Creating UTL Labels

Before each section of a source file is closed (including the last section, since a CLOSE instruction is used), the software writes an EOV or EOF label and links to the user routine before close. The programmer uses the LPUT instruction within this routine to create as many UTL Labels as desired. After the labels are output, a RELINK instruction (without an operand) must be executed to return control to the software. The software then writes the proper number of tape marks and returns control to the main program.

Caution should be used when UTL labels are written following the EOV label; since EOV is triggered by the end-of-media marker, limited space remains for UTL labels.

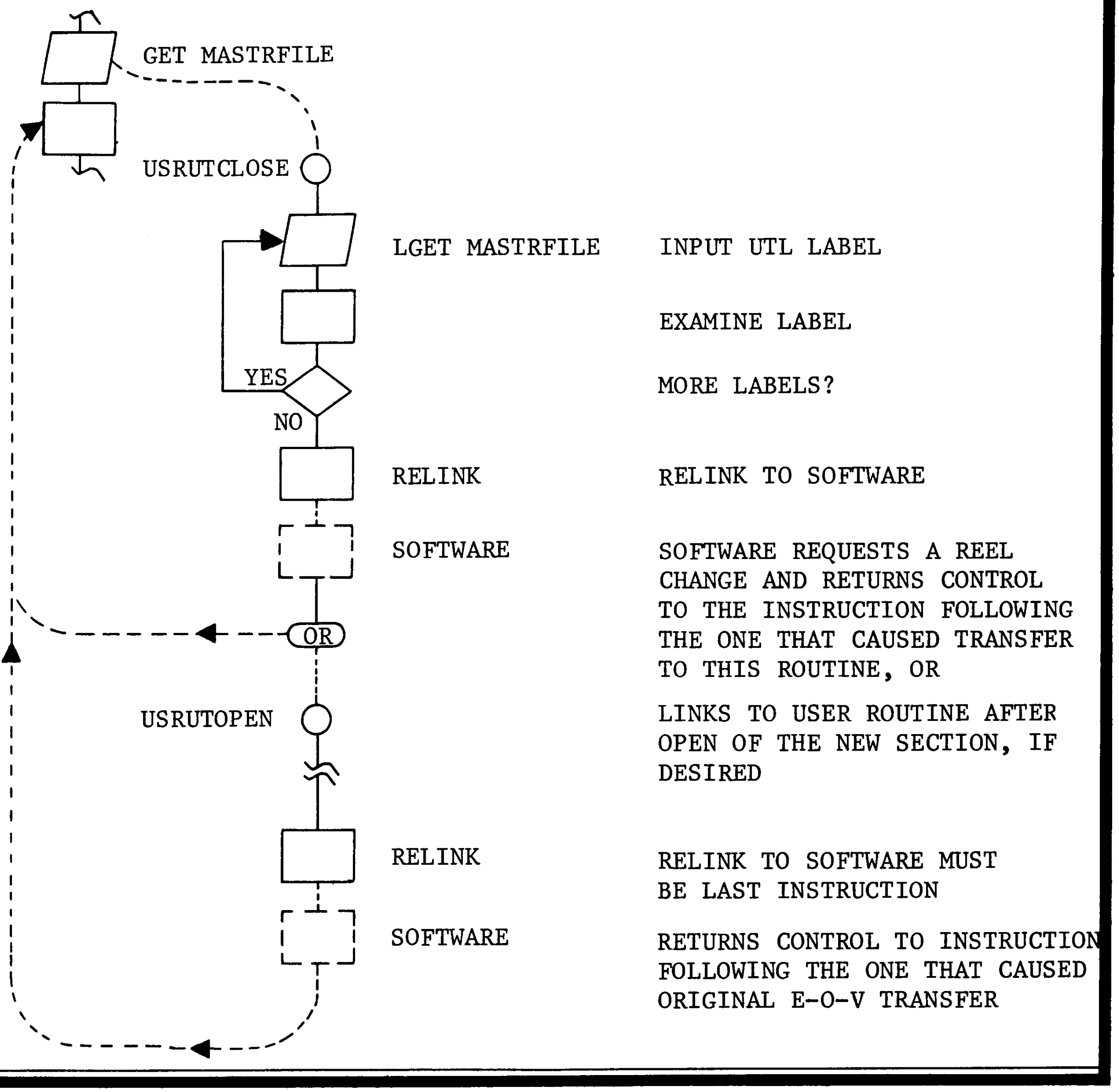
- Inputting UTL Labels

When an EOV is read from a source file, the software places that label into the input buffer and links to the user routine before close. The programmer may examine the contents of the label, or he may ignore it by executing an LGET instruction to input the first UTL label.

The programmer may code his routine to input all UTL labels or a portion of them. If all labels are not input, a RELINK instruction (with no operand) must be used to return control to the software. If all labels (plus one) are input, a RELINK instruction is not needed. When an LGET instruction reads a tape mark (end of a group of labels), it causes an automatic relink to the software.

When the software receives control again, it requests the operator to change the reel. Once the reel is changed, the software opens the new file section and transfers control to the user routine after open if desired (see inputting UHL labels). Upon receiving control back from that routine (or if that routine was not specified), the software reads the first block of records from the new section and returns control to the instruction following the GET instruction that initially caused the link to the user routine before close.

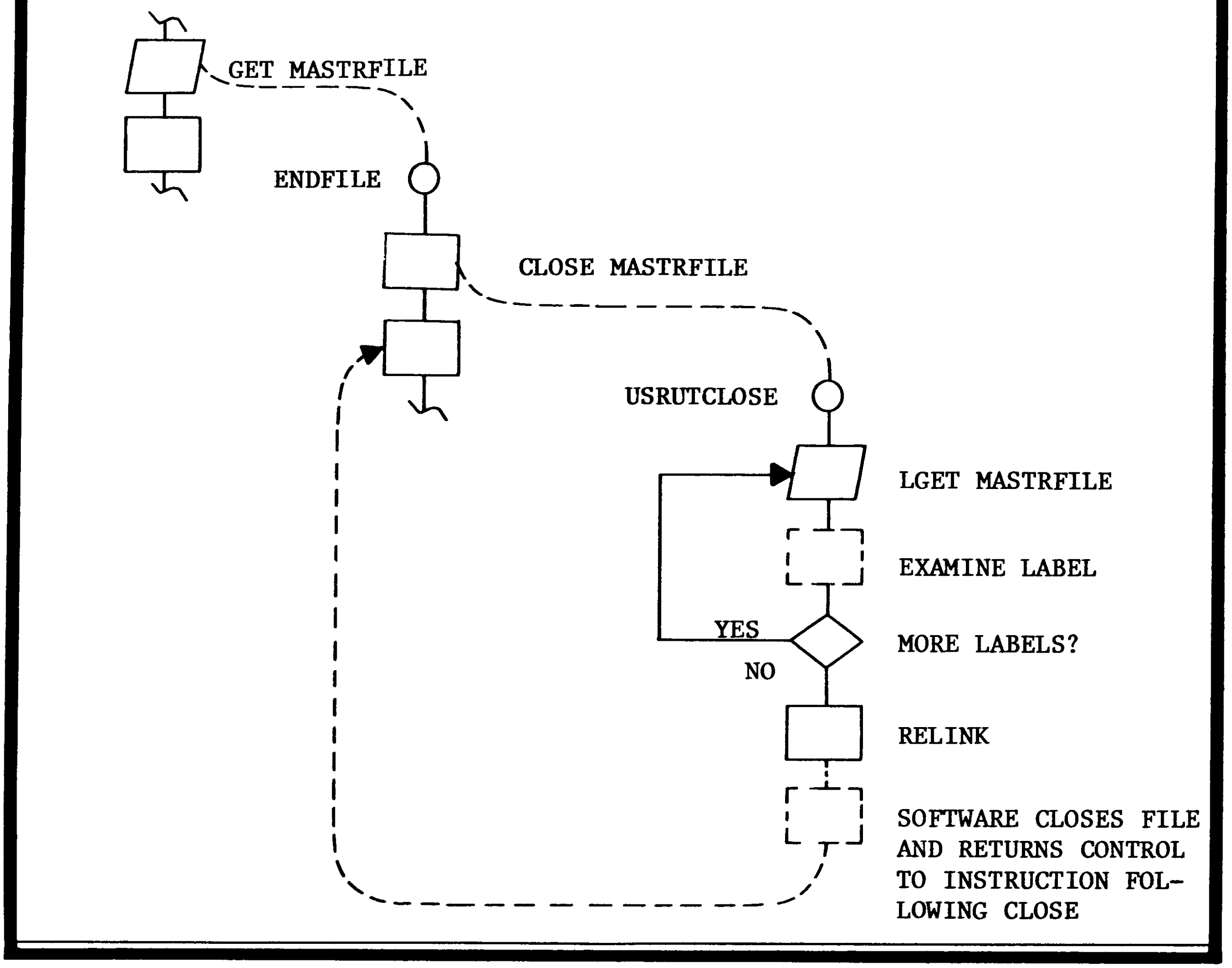
INPUTTING UTL LABELS AT END-OF-VOLUME


GET MASTRFILE
USRUTCLOSE
LGET MASTRFILE
INPUT UTL LABEL
EXAMINE LABEL
YES
MORE LABELS?
NO
RELINK
RELINK TO SOFTWARE
SOFTWARE
SOFTWARE REQUESTS A REEL CHANGE AND RETURNS CONTROL TO THE INSTRUCTION FOLLOWING THE ONE THAT CAUSED TRANSFER TO THIS ROUTINE, OR
OR
USRUTOPEN
LINKS TO USER ROUTINE AFTER OPEN OF THE NEW SECTION, IF DESIRED
RELINK
RELINK TO SOFTWARE MUST BE LAST INSTRUCTION
SOFTWARE
RETURNS CONTROL TO INSTRUCTION FOLLOWING THE ONE THAT CAUSED ORIGINAL E-O-V TRANSFER

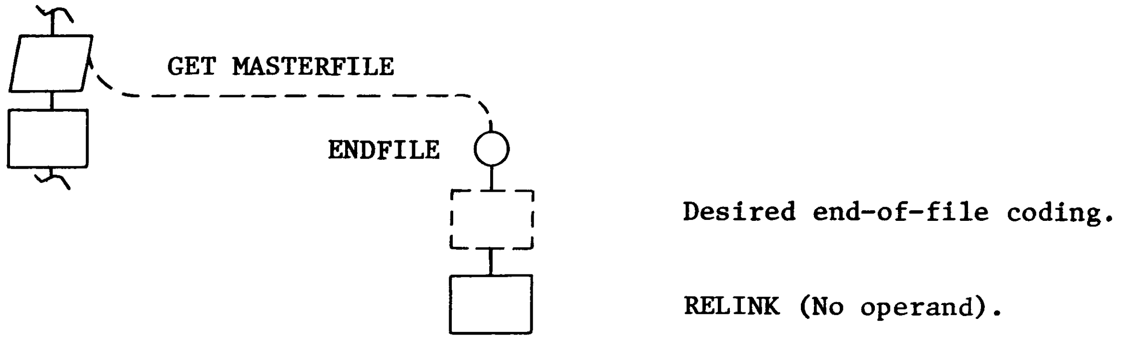
When an EOF label is read from a source file, the software places that label in the input buffer and links to the programmer's end-of-file routine. (The programmer may not use the LGET instruction within this routine to input UTL labels since execution of LGET and LPUT is restricted to the user routine before close or after open.) A CLOSE instruction must be used to enter the user routine before close for the last section; the programmer may code that CLOSE instruction within his end-of-file routine.

When the CLOSE instruction is executed, the software links to the user routine before close in the same manner previously described for an EOV label situation. At the end of the user routine before close, the RELINK instruction returns control to the software, which in turn closes the file and returns control to the instruction following the CLOSE instruction in the end-of-file routine.

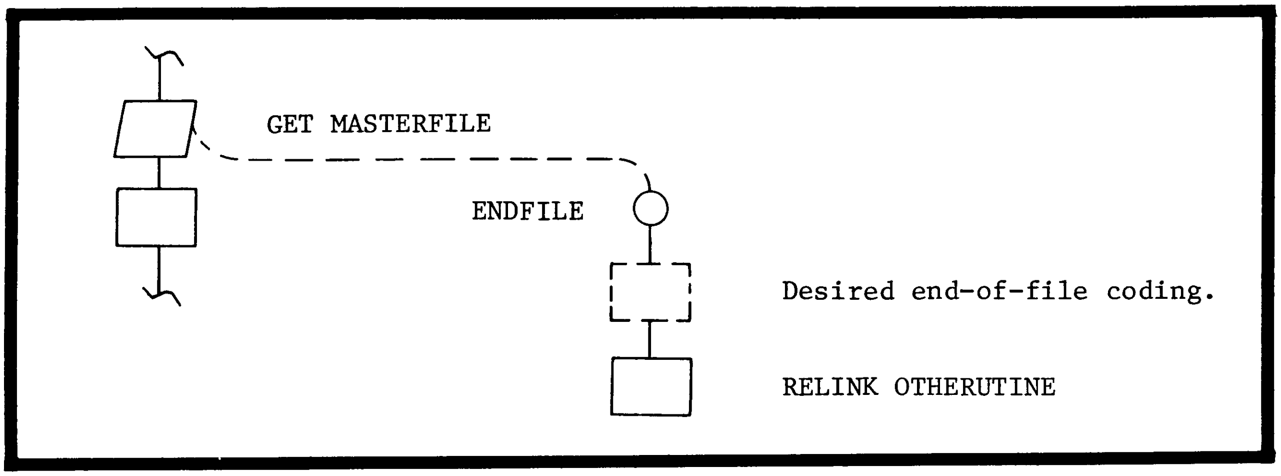
INPUTTING UTL LABELS AT END-OF-FILE



Once control is returned to the end-of-file routine, the programmer may wish to return control to the instruction following the GET which caused the branch to the end-of-file routine; he may do so by using a RELINK instruction without an operand. (A LINK or BRANCH instruction may not be used for this purpose.)

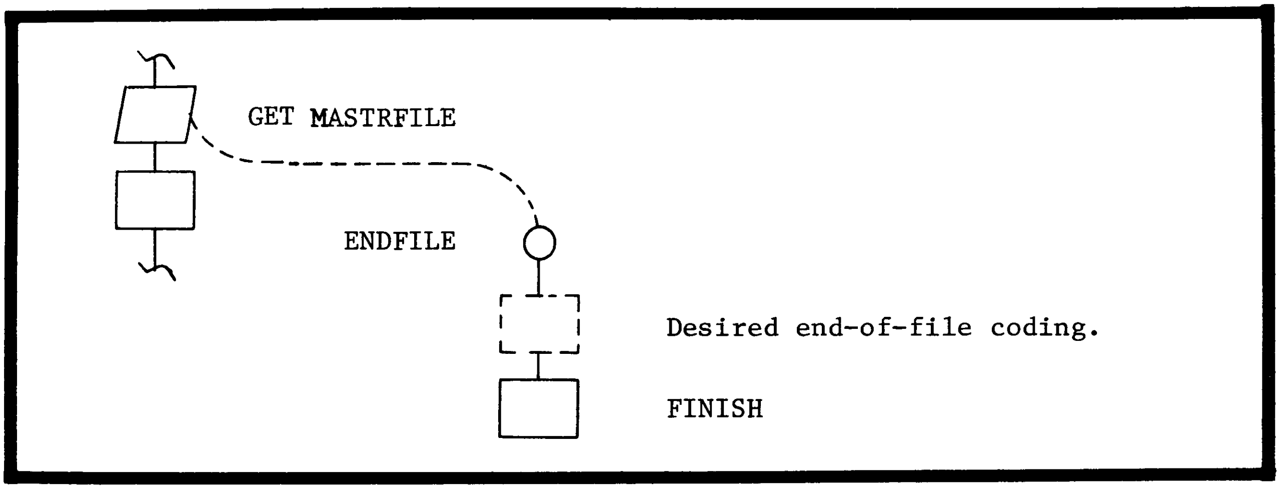


If the programmer wishes to return control to a different instruction or routine, he may do so by using a RELINK instruction that names the desired routine as an operand. The LINK or BRANCH instruction may not be used for this purpose.

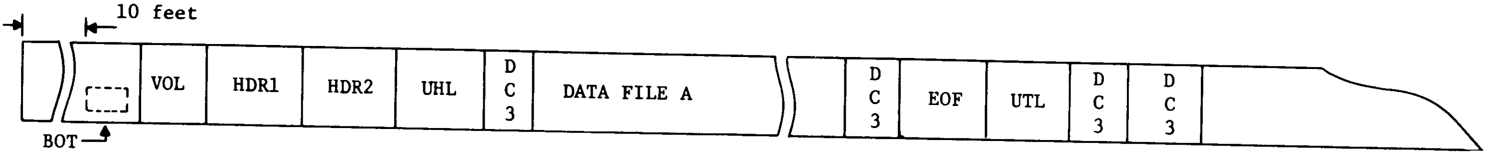


For further details concerning RELINK and RELINK with an operand, see the NEAT/3 REFERENCE MANUAL, INSTRUCTIONS, tab 2, "Link and Relink Instructions."

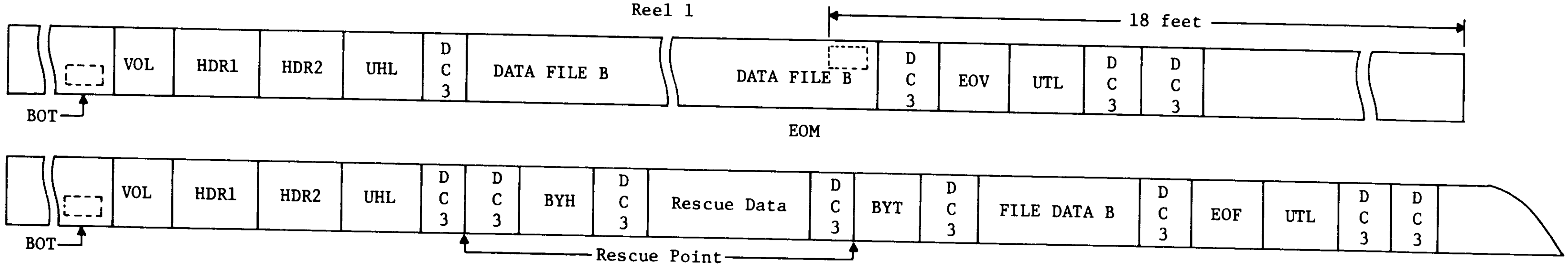
The programmer need not relink from his end-of-file routine but may continue his coding within the routine and complete the program with a FINISH instruction.



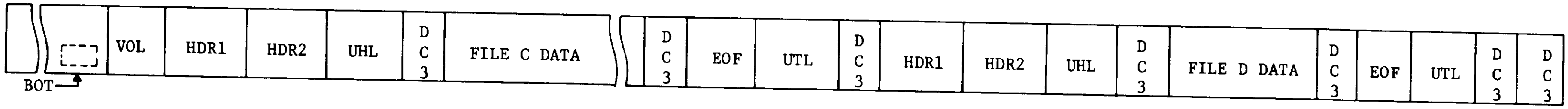
Single File - Single Volume



Single File - Multi Volume



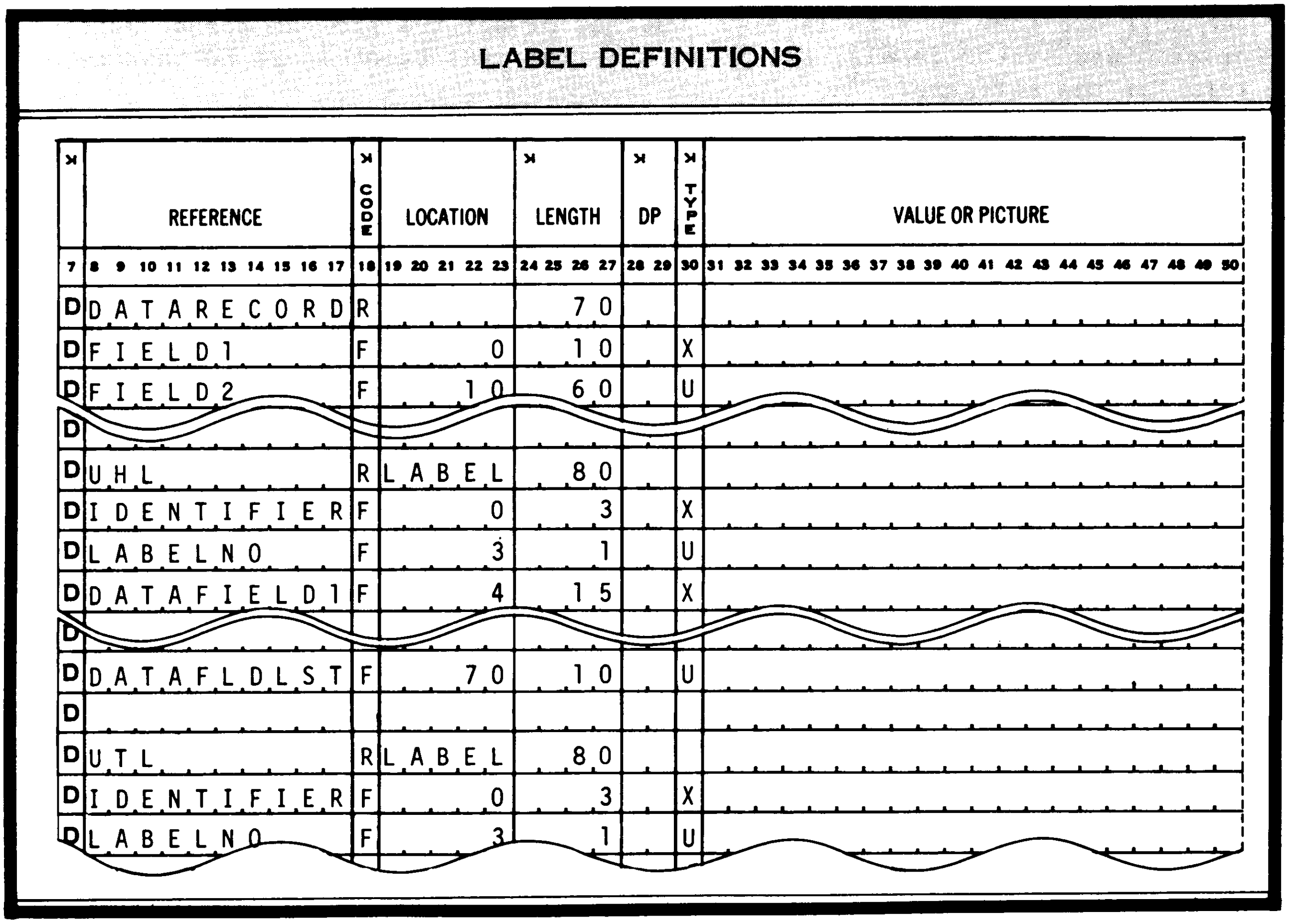
Multi File - Single Volume



- VOL Placed on the tape by Initializer Utility Routine.
- HDR1 Placed on tape by I/O software when file is created.
- HDR2 Placed on tape by I/O software when file is created.
- UHL Optional label placed on tape by using the LPUT instruction; read by using the LGET instruction.
- DC3 Tape mark placed on tape by software.
- BYH Placed in front of a rescue dump by the software.
- BYT Placed at the end of a rescue dump by the software.
- EOV Placed at end of a volume (section) by software when file is created.
- EOF Placed at end of a file by software when file is closed.
- UTL Optional label placed on tape by using the LPUT instruction; read by using the LGET instruction.

Defining the Buffer for Inputting or Outputting Labels

Since labels are input to and output from the buffer, a magnetic tape buffer may not be less than 80 characters in length. A definition of each type label affected (written, read, or examined) by the program must be included with the record definitions for each file. The programmer accomplishes this by inserting the word "LABEL" in the location column of the label definition. Labels must be defined last regardless of the size of the data record for the file.



LABEL DEFINITIONS

	REFERENCE	CODE	LOCATION	LENGTH	DP	TYPE	VALUE OR PICTURE
7	8 9 10 11 12 13 14 15 16 17	18	19 20 21 22 23	24 25 26 27	28 29	30	31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
D	DATARECORD	R		70			
D	FIELD1	F	0	10		X	
D	FIELD2	F	10	60		U	
D							
D	UHL	R	LABEL	80			
D	IDENTIFIER	F	0	3		X	
D	LABELNO	F	3	1		U	
D	DATAFIELD1	F	4	15		X	
D							
D	DATAFLDLST	F	70	10		U	
D							
D	UTL	R	LABEL	80			
D	IDENTIFIER	F	0	3		X	
D	LABELNO	F	3	1		U	

The programmer may eliminate all label definitions if his program does not affect labels. The software will index past all labels before the first GET instruction is encountered.

Data Blocks

Data blocks may consist of one or more records, depending upon the record size. Records within these blocks may be fixed or variable in length.

Minimum and maximum block-lengths are dependent upon several factors. Two of these factors are the software and the machine hardware; another factor is the possible use of the tape for information interchange with other systems.

- Software and Hardware Limitations

  The hardware allows a one-character block; however, in the normal NCR Century mode of processing, buffers must be a minimum of 80 characters to allow the software room to read labels. Whether or not the program reads, writes, or examines system labels, the minimum buffer must be at least 80 characters for the software.

  The NCR Century 100 hardware and software allow a maximum block length of 2048 characters. The NCR Century 200 hardware and software allow a maximum block length of 9,999 characters.

- USASI Standards for Interchange

  USASI standards allow a minimum block length of 18 characters and a maximum block length of 2048 characters for interchange tapes. Since all NCR Century Systems can create blocks between 80 and 2048 characters and can read blocks between 3 and 2048 characters, system limitations fall well within the standard.

- Block Length Indicators (BLI)

  Optionally, the software is capable of creating or reading Block Length Indicators (BLI) which specify the number of characters in each block. These indicators are not a part of the USASI standard and must not appear in a tape that must conform to the standard. They may be used on tapes interchanged between users who agree to deviate from the standard or by users not interested in interchange.

  By answering a question on the file specification sheets, the programmer requests that the software create or check BLI's. On output, the software computes the number of characters in a block and places this information at the beginning of a block; the programmer must add two or four characters to his block length to accomodate the type of BLI desired.

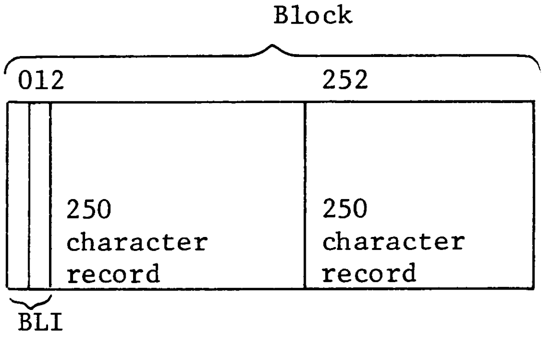
  On input, the software compares the number in the BLI with the actual number of characters read. If the two are not equal, the software re-reads the block in an attempt to correct the condition.

  The BLI may be two or four characters long; however, the first two characters always contain the block length in binary.

  - 2-character BLI

    If the 2-character BLI is used, the programmer must add two characters to his total block length; a header offset of two is assumed by the compiler. This offset is in addition to that specified on the file specification sheets; for example, if the header offset specified on the file specification sheet is zero and a two character BLI is requested, records are presented to the program beginning at relative position two (the third character).

2—CHARACTER BLI



- The BLI is a two-character binary field containing 502.

00000001	11110110

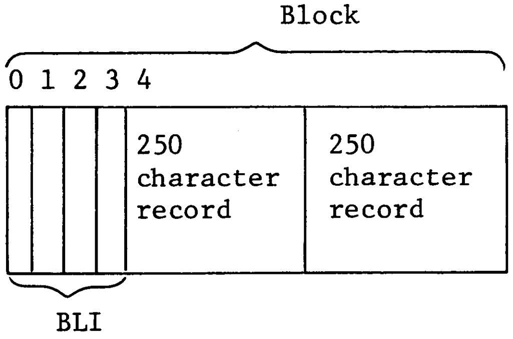
- A Header offset of 2 is assumed by the compiler.
- GET presents records beginning at relative position 2.

- 4-character BLI

The BLI may be a 4-character field in which the first two characters are the character count in binary and the second two characters are blank. The programmer must indicate a header-offset of 2 on the file specification sheets; this indicated header-offset, plus the header-offset of 2 assumed by the compiler, allows a header-offset of 4.

The first record presented to the program from each block begins at relative position four (the fifth character).

4—CHARACTER BLI



- The BLI is a four-character field containing 504 in the first two characters.

00000001	11111000	00000000	00000000

- A Header offset of 2 is assumed by the compiler. An additional header offset of 2 must be specified by the programmer for a total header offset of 4.
- GET presents records beginning at relative position 4.

Records in Block

Records within magnetic tape blocks may be fixed- or variable-length. On the NCR Century 100, record sizes may range from a minimum of 1 character to a maximum of 2048 characters.

On the NCR Century 200, record sizes may range from a minimum of 1 character to a maximum of 9,999 characters. If interchange is desired, the maximum record size must be limited to the maximum block size allowed by the USASI standard (2048 characters).

The GET and PUT instructions are capable of handling fixed- or variable-length records.

- Fixed-Length Records

  If the records are fixed-length, the programmer assigns a block length that is a multiple of his record length. For example, if each record is 100 characters long, the block length may be 100, 200, 300, or any other multiple of 100 up to the maximum block length for the system used.

- Variable-Length Records

  When variable-length records are used, a Variable-Length Indicator (VLI) must be defined as the first field in each record. This indicator, which contains a number that specifies the number of characters in a record (including the VLI itself), is used by the software to present complete records to the program. The programmer must include in his program the coding necessary to alter the contents of the VLI when the record length changes, unless tables are used as the variable-length portion of the records. For a complete description of tables in records, see the NEAT/3 REFERENCE MANUAL, INSTRUCTIONS, tab 4, "Table Concepts."

  The VLI is never translated. If an interchange tape is read or written in a code other than NCR Century internal code, the VLI must be either in binary or in NCR Century internal code on the tape.

- VLI - No Interchange

  If interchange is not desired, the VLI on tape may be either two characters long with the VLI in binary, or four characters long with the VLI in decimal. Binary indicators are used most often. Decimal indicators are generally reserved for instances when data is to be transmitted via communication equipment.

  If decimal VLI's are used, certain restrictions are necessary:

  - Since the GET routine does not translate the VLI, the VLI must be on tape in NCR Century internal code.

  - On input, the VLI is changed from decimal to binary by the GET routine and is then placed in the first two characters of the field. Once the VLI is in memory, the programmer must manipulate it in binary.

BINARY VLI - NO INTERCHANGE

DATA LAYOUT SHEET

x	REFERENCE	x CODE	LOCATION	x LENGTH	x DP	x TYPE	VALUE OR PICTURE
7	8 9 10 11 12 13 14 15 16 17	18	19 20 21 22 23	24 25 26 27	28 29	30	31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
D	MAGZREC	R		502			
D	VLIFLDNAME	F	0	2		B	
D	ACCTNO	F	2	10		X	

- Record Length — The programmer specifies the maximum length of the variable-length records as indicated on the file specification sheets.
- VLI Field — The programmer specifies a 2-character binary field.

- On output, the PUT routine converts the binary VLI in memory to decimal before the record is output. The decimal VLI is always output in NCR Century internal code (USASCII standard) and is not translated if the tape is written using an external code.

- VLI - Interchange

When interchange is desired, the VLI field must be four characters in length. On tape, the data within this field may be contained either in binary in the first two characters or in decimal in all four characters. If the VLI is in decimal on the tape, it is converted on input to binary in memory as described previously under VLI - No Interchange.

DECIMAL VLI FOR INTERCHANGE OR FOR NO INTERCHANGE
BINARY VLI FOR INTERCHANGE

X	REFERENCE	X CODE	LOCATION	X LENGTH	X DP	X TYPE	VALUE OR PICTURE
7	8 9 10 11 12 13 14 15 16 17	18	19 20 21 22 23	24 25 26 27	28 29	30	31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
D	MAGZREC	R		504			
D	VLIFIELD	F	0	4		X	
D	BINARYVLI	F	0	2		B	

- Record Length — The programmer specifies the maximum length of the variable-length record as indicated on the file specifications sheets.

- VLI Field — The programmer specifies a four-character field.

- Binary VLI — The programmer redefines the first two characters of VLIFIELD to permit internal manipulation of the VLI when the record length changes during processing.

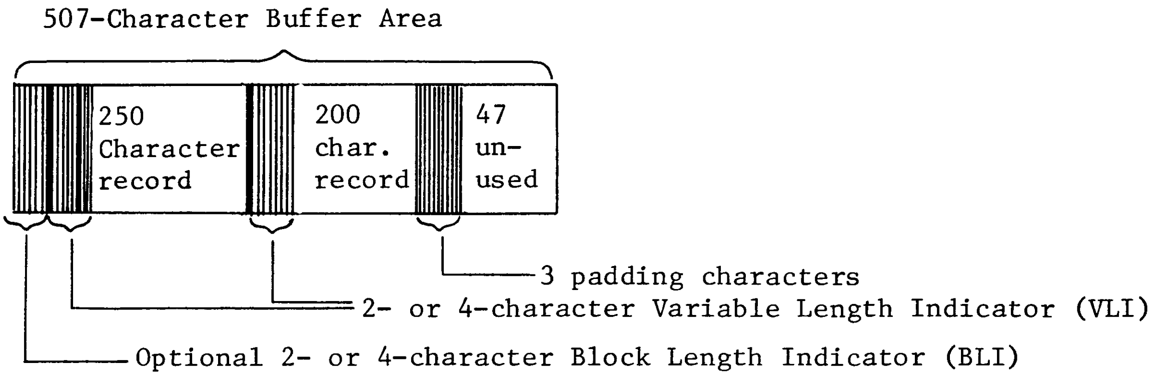
  - If decimal VLI's are used, the GET routine changes the decimal VLI on tape to a binary VLI in memory and places it in the first two characters of the field. The PUT routine changes the binary VLI in memory to a decimal VLI on tape.

When writing variable-length magnetic tape records with the NCR Century 100, three padding characters are automatically placed after the last record in the buffer area and a trailer offset of 3 is assumed. These three characters must be considered when calculating the maximum length of a block for variable-length records.

When the combination of records placed in the buffer area of an NCR Century 100 does not completely fill that area, the entire buffer is still written out. The block written contains the records, padding characters, and unused portion of the buffer. On input, the GET instruction recognizes the padding characters and does not present the padding characters or the unused portion of the block to the program.

On the NCR Century 200, only that portion of the buffer that contains usable data is output; the padding characters are not used.

DETERMING MAXIMUM BLOCK LENGTH ON NCR CENTURY 100


507-Character Buffer Area
250 Character record
200 char. record
47 un-used
3 padding characters
2- or 4-character Variable Length Indicator (VLI)
Optional 2- or 4-character Block Length Indicator (BLI)

- Assume that a 4-character BLI is used with each block.
- Assume that the maximum-length record, including a 2-character VLI, is 250 characters.
- Assume that the maximum-size block may contain two full records.
  - Maximum Block Length: BLI (4 characters) + records in block (500 characters) + padding characters (3 characters) = total (507 characters).
  - Buffer Area = Maximum block-size (507 characters)
  - Entire buffer, including padding characters and any unused portion of buffer, is always written. Therefore, each block written = total buffer length (507 characters).

SUMMARY OF USASI STANDARDS FOR AN INTERCHANGE TAPE

- The tape must be recorded in the USASI standard code set for information interchange (USASCII). The NCR Century Series uses this code.
- The tape must have the following 80-character labels:

LABEL TYPE	NUMBER	DESCRIPTION
VOL	1 per reel	Volume Header Label
HDR	1 per section	File Header Label 1
*EOV	1 per reel	End-of-Volume Trailer Label
EOF	1 per file	End-of-File Trailer Label

* Multi-section files only.

- The tape may additionally have the following 80-character labels:

LABEL TYPE	NUMBER	DESCRIPTION
UVL	9 per reel	User Volume Header Label
HDR	8 additional	File Header Label 2-9
UHL	* per section	User Header Label
UTL	* per section	User Trailer Label
* No set maximum; as many as desired.		

- The tape may not have the following labels:

LABEL TYPE	DESCRIPTION
BYH	Bypass Header Label
BYT	Bypass Trailer Label

- One tape mark must be used to separate labels from data and data from labels.

- Two tape marks must be used following:

  - The last EOF label on a tape and
  - Each EOV label.

- The tape mark is a series of characters with the binary configuration 00010011; this is equivalent to the USASCII character, "DC3".

- Records may be a maximum of 2048 characters long (including the VLI if variable length records are used).

- The VLI must be four characters in length with:

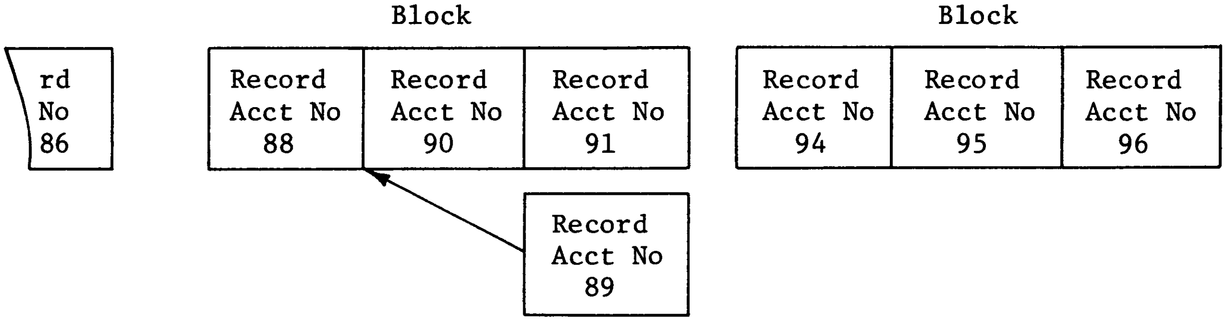
  - The data within the field in binary in the first two character positions, or
  - The data in decimal (USASCII) in all four character positions.

- Blocks of records may be a maximum of 2048 characters long and may not contain block length indicators (BLI).
- Three padding characters may be used within a short block to signal end-of-data. These padding characters, which have the binary configuration 01011110, are the USASCII character, "∧".
- The tape must be recorded in odd parity.

FILE ORGANIZATION

Standard File Organization is used for magnetic tape files. The records in the file, sorted or unsorted, are always presented to the program sequentially. No random accesses are possible. When a record is to be inserted or deleted, the entire volume (reel) must be copied. A record may not be read in, updated, and then written back in the same location. Father-Son is the normal type of processing for magnetic tape. For a complete description of Father-Son processing, see the NEAT/3 REFERENCE MANUAL, FILES, "NCR Century File Concepts".

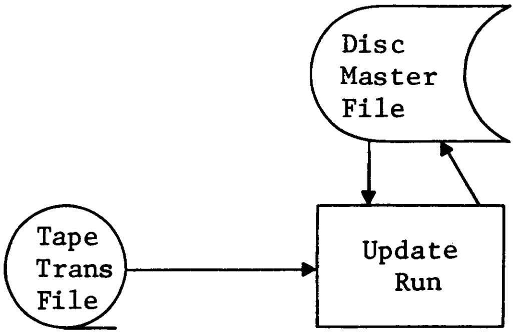
INSERTING A NEW RECORD



- To place the record for Acct. No. 89 between record for Acct. No. 88 and Acct. No. 90, the entire volume or file must be copied.

Tape files may be used to input the transaction records in a disc file update run.

INPUTTING TRANSACTION RECORDS



- Transactions are read from the tape.
- Master records are read from the disc.
- Updated master records are written back on disc.

FILE OPTIONS

To provide further ease for the programmer using interchange tapes in a nonstandard code set, translation may be requested by entering a code on the file specification sheets. This code instructs the compiler to include a translation table in the program for use by the GET and PUT instructions in reading interchange tapes. GET then translates data from the specified code set on the tape into NCR Century internal code in memory. The PUT instruction translates from NCR Century internal code in memory into the specified external code set desired on the tape.

In this way, a tape that conforms to the USASI interchange standards for labels and block lengths (but is not recorded in the USASI standard code set) may be easily decoded from (or encoded to) any of the following three code sets:

- NCR 315
- IBM Binary-Coded-Decimal (BCD)
- IBM Extended-Binary-Coded-Decimal-Interchange-Code (EBCDIC)

NCR 315 Format Tapes

The software is also capable of processing and creating tapes in the NCR 315 labeling format and code set. Information concerning this capability will be supplied at a later date.

OPENING FILES ON MAGNETIC TAPE

Normally, the programmer allows his magnetic tape files to be opened automatically at the time his program is loaded into memory; however, he may use an OPEN instruction. One use of the OPEN instruction is to place multiple files on a reel during one run. The OPEN instructions, OPEN and OPENS, are covered in the NEAT/3 REFERENCE MANUAL, INSTRUCTIONS, tab 1, "OPEN Instructions."

A magnetic tape file may be reopened during a program by using one of the ROPEN instructions and indicating on the file specification sheets that ROPEN is desired. This allows the programmer to create a file as a destination file, for example, and then reopen it as a source file in the same run. For further information on ROPEN instruction, see the NEAT/3 REFERENCE MANUAL, INSTRUCTIONS, tab 1, "ROPEN Instructions."

Each time a file is opened, a message is recorded in the log.

Opening Source Files

Before a source file is opened on a single-file reel or a multi-reel file, the tape is in a rewound state with the VOL label in a position to be read. When the OPEN is executed, the software reads the VOL label and the HDR1 label and compares the information in HDR1 with the information specified on the file specification sheets. If this is the desired file, the software opens the file, leaves the HDR1 label in the input buffer, and transfers control to the user routine after open (if such a routine is specified and an OPEN instruction is used).

Before a source file is opened on a multi-file reel (or on a multi-file, multi-reel set) the tape may or may not be in a rewound state. The tape will be in a rewound state if (1) this reel was newly mounted and this is the first file opened on that reel or (2) "rewind before open" was requested on the file specification sheets. The tape will not be in a rewound state if a file was previously opened and "rewind before open" was not requested on the file specification sheets. When the open is executed, the software scans the tape (beginning at its current location) for the HDR1 label of the desired file. When the file is found, the open procedure is the same as for a single-file reel.

If the software encounters an EOV label while searching for the file, it requests the operator to change the reel. If an EOF label is encountered for the last file on the reel or in the set, it signifies that the file cannot be found, and the condition is displayed to the operator.

Opening a Destination File

The procedure involved in opening a destination file depends on the type of reel/file relationship involved.

- Single-Reel and Multi-Reel File

  Before a single-reel or multi-reel file is opened, the tape is in a rewound state with the VOL label in a position to be read. When the open is executed, the software reads the VOL and HDR1 label of the file currently on the tape to assure that the file has expired. If it has expired, the software backs up the tape and writes HDR1 and HDR2 labels for the new file. If an OPEN instruction was used for the file, control is transferred to the user routine after open (if such a routine was specified).

  If the file currently on tape is not out of date, the condition is displayed to the operator.

- Multi-File Reel

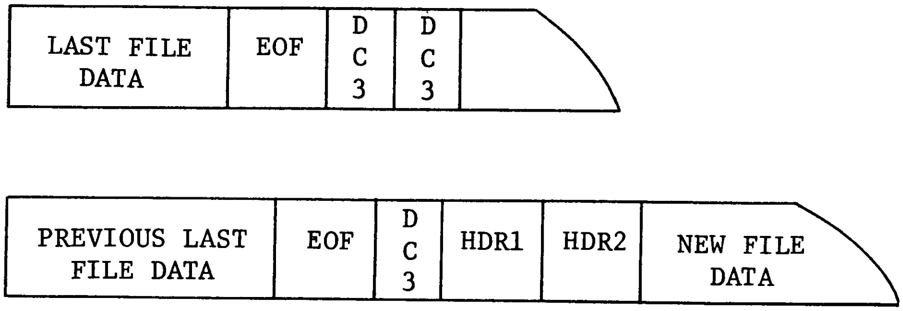
  When a multi-file reel is used, the procedure depends on whether (1) a file is to be added to the existing files or (2) all old files are to be destroyed and a new file created at the beginning of the reel.

  - Adding a File to Existing Files

    If a file is added to a reel of existing files, the reel may or may not be in a rewound state before the new file is opened. It will be in a rewound state if the reel was newly mounted, if "rewind before open" was requested on the file specification sheets for the new file, or if rewind after close was requested on the file specification sheets for a previously accessed file.

    With the tape positioned at the beginning of the reel, as in any of the above cases, the software checks the date in the first file which determines the expiration date of all files on the reel or in the set. If the first file has not expired, the software indexes down the tape until it encounters the EOF label for the last file on the reel (multi-file reel) or in the set. If the software encounters an EOV label during the search , it requests the operator to mount the next reel. When the EOF label is found, the software repositions the tape to remove the second tape mark after that label, and writes the HDR1 and HDR2 labels for the new file.

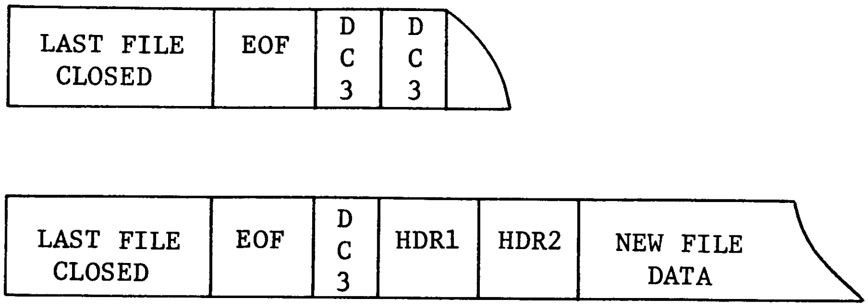
ADDING A FILE



The software finds the EOF label for the last file, removes the second tape mark and writes the HDR1 and HDR2 labels for the new file.

The tape will not be in a rewound state if a previous file was accessed and then closed without rewind, or a new file open is executed without specifying "rewind before open". In this case, the software places the new file in a position immediately following the previously closed file. Using this option saves the programmer the tape rewind time as well as the search time necessary to relocate the position for the new file. However, the programmer must use this option with care, since the software does not rewind the tape to date-check the new file against the first file on the reel or in the set.

ADDING A FILE - NO REWIND



The software removes the second tape mark following the EOF for the last file and writes the HDR1 and HDR2 label for the new file at the current tape location.

- Creating a New File at the Beginning of the Reel or Set

  If a file is created at the beginning of a multi-file reel or set, the old files within the entire reel or set are destroyed. This procedure is used each time a reel or set is to contain a new generation of files.

  Before the file is opened, the reels must be in a rewound state. They will be in a rewound state if the reels are newly mounted, rewind was requested on the file specifications sheets for the file last closed, or "rewind before open" was requested on the file specification sheets for the new file. The software checks the expiration date in the label and, finding the file has expired or the tape is a scratch tape, writes the HDR1 and HDR2 labels for the new file over the old labels. Once this is done, the only file that can be accessed by a program is the first one on tape.

  If another new file is added, it is placed as the second file on the reel or in the set.

Opening Piggyback Files

When a piggyback file is opened, the operator normally mounts the reel that contains the current last section of the file; the software indexes down the tape to locate the EOF label. If the last section was not mounted, the software encounters an EOV label and informs the operator to change the reel.

When the EOF label is located, the software checks to insure that this is the last file on the reel or in the set. If it is not, new data cannot be added to the file without destroying the next file; therefore, an error message is displayed to the operator. If this is the last file, control is given to the programmer's end-of-file routine if one is specified on the file specification sheets. The programmer may use this routine to read any existing user trailer labels. He must code a RELINK instruction as the last instruction in the routine to return control to the software.

When the relink from end-of-file occurs or if no end-of-file routine was specified, the tape is repositioned immediately beyond the last data block and a rescue dump is taken. This rescue dump destroys the previous trailer labels and the tape marks before control is returned to main program.

EXTENDING A PIGGYBACK FILE

PREVIOUS LAST DATA FOR FILE	EOF	UTL	D C 3	D C 3	NO OTHER FILES

PREVIOUS LAST DATA FOR FILE	D C 3	BYH	D C 3	RESCUE DATA	D C 3	BYT	D C 3	NEW DATA FOR FILE

The software checks to insure this is the last file on the reel, gives control to the programmer's end-of-data routine to read optional UTL labels if desired, repositions the tape over the EOF label, and initiates a rescue dump. New file data is placed following the rescue dump on the tape.

When a section of a piggyback file cannot be located on the mounted reel, this condition is displayed to the operator. In turn, the operator may change the reel or indicate that the first section of a new piggyback file is to be created; a normal destination file open is then performed.

CLOSING FILES ON MAGNETIC TAPE

Normally, the programmer allows his magnetic tape files to be closed automatically at the time the FINISH instruction is encountered; however, a CLOSE instruction may be used. One use of the CLOSE instruction is to place multiple files on a reel during a run. The CLOSE instructions, CLOSE and CLOSEO, are covered in the NEAT/3 REFERENCE MANUAL, INSTRUCTIONS, tab 1, "CLOSE Instructions.'

When a file is closed, a message is placed in the log only if an error occurred while processing that file.

Closing a Source File

When a close is executed for a source file, the software transfers control to the user routine before close (if such a routine was specified and a CLOSE instruction was used). The programmer may use this routine to input user trailer labels (UTL) from the tape. He must use a RELINK instruction as the last instruction in the routine to return control to the software.

Once the software completes the close procedure, it rewinds the tape to an unload condition, or rewinds the tape to the beginning of the reel, or leaves the tape at its current position, depending upon the programmer's request on the file specification sheets.

Closing Destination or Piggyback Files

When a close is executed for a destination file or piggyback file, all buffers not yet empty are output. The software then writes the EOF label and transfers control to the user routine before close (if such a routine was specified and a CLOSE instruction was used). The programmer may use this routine to output UTL labels. He must use a RELINK instruction as the last instruction in the routine to return control to the software.

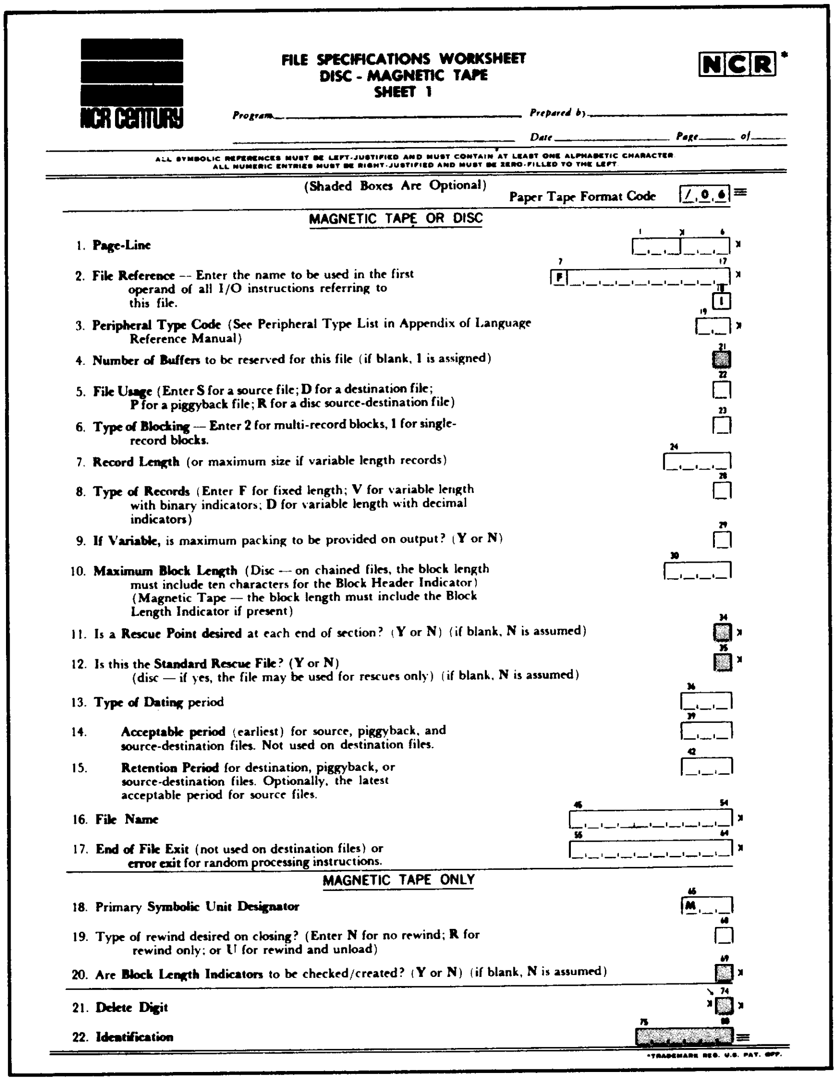
Once the software receives control from the user routine or if no routine was specified, the software writes two tape marks to indicate the end of the last file on the tape.

When the close procedure is complete, the software rewinds the tape to an unload condition, or rewinds the tape to the beginning of the reel, or leaves the tape at its current position, depending upon the programmer's request on the file specification sheets.

FILE SPECIFICATION SHEETS FOR MAGNETIC TAPE

Two File Specification Sheets for a typical magnetic tape file are shown. Many of the entries pertain to disc as well as tape; however, disc will not be considered in the examples given here. All entries, where practical, are filled with a typical remark.

SHEET 1



NCR CENTURY

FILE SPECIFICATIONS WORKSHEET
DISC - MAGNETIC TAPE
SHEET 1

NCR*

Program__________ Prepared by__________

Date________ Page____ of____

ALL SYMBOLIC REFERENCES MUST BE LEFT-JUSTIFIED AND MUST CONTAIN AT LEAST ONE ALPHABETIC CHARACTER
ALL NUMERIC ENTRIES MUST BE RIGHT-JUSTIFIED AND MUST BE ZERO-FILLED TO THE LEFT

(Shaded Boxes Are Optional)

Paper Tape Format Code /06 ≡

MAGNETIC TAPE OR DISC

1. **Page-Line** (1, 6) x
2. **File Reference** -- Enter the name to be used in the first operand of all I/O instructions referring to this file. (7 F, 17) x (18 I)
3. **Peripheral Type Code** (See Peripheral Type List in Appendix of Language Reference Manual) (19) x
4. **Number of Buffers** to be reserved for this file (if blank, 1 is assigned) (21)
5. **File Usage** (Enter **S** for a source file; **D** for a destination file; **P** for a piggyback file; **R** for a disc source-destination file) (22)
6. **Type of Blocking** — Enter **2** for multi-record blocks, **1** for single-record blocks. (23)
7. **Record Length** (or maximum size if variable length records) (24)
8. **Type of Records** (Enter **F** for fixed length; **V** for variable length with binary indicators; **D** for variable length with decimal indicators) (28)
9. **If Variable,** is maximum packing to be provided on output? (**Y** or **N**) (29)
10. **Maximum Block Length** (Disc — on chained files, the block length must include ten characters for the Block Header Indicator) (Magnetic Tape — the block length must include the Block Length Indicator if present) (30)
11. Is a **Rescue Point desired** at each end of section? (**Y** or **N**) (if blank, **N** is assumed) (34) x
12. Is this the **Standard Rescue File**? (**Y** or **N**) (disc — if yes, the file may be used for rescues only) (if blank, **N** is assumed) (35) x
13. **Type of Dating** period (36)
14. **Acceptable period** (earliest) for source, piggyback, and source-destination files. Not used on destination files. (39)
15. **Retention Period** for destination, piggyback, or source-destination files. Optionally, the latest acceptable period for source files. (42)
16. **File Name** (45, 54) x
17. **End of File Exit** (not used on destination files) or **error exit** for random processing instructions. (55, 64) x

MAGNETIC TAPE ONLY

18. Primary **Symbolic Unit Designator** (65 M)
19. Type of rewind desired on closing? (Enter **N** for no rewind; **R** for rewind only; or **U** for rewind and unload) (68)
20. Are **Block Length Indicators** to be checked/created? (**Y** or **N**) (if blank, **N** is assumed) (69) x
21. **Delete Digit** (74) x
22. **Identification** (75, 80) ≡

*TRADEMARK REG. U.S. PAT. OFF.

The programmer should fill in the header, page-and-line number (Question 1), and the identification tag (positions 75-80) as defined in the NEAT/3 REFERENCE MANUAL, INTRODUCTION AND DATA, tab 3, "Programming Worksheets." The paper tape format code is preprinted on this sheet and must be·punched if paper tape is used for input to the compiler.

2. FILE REFERENCE – ENTER THE NAME TO BE USED IN THE FIRST OPERAND OF ALL I/O INSTRUCTIONS REFERRING TO THIS FILE. F O M A G Z F I L E Ø

Enter the name used in the PUT, GET, and other I/O instructions that access this file in the program.

If this file is to be used in a Father-Son update run, the file reference name and the file name (Question 16 on Sheet 1) must be different. (For further information, see the NEAT/3 REFERENCE MANUAL, FILES, "NCR Century File Concepts.")

The "F" in position 7 is preprinted and must be punched.

3. PERIPHERAL TYPE CODE [SEE PERIPHERAL TYPE LIST IN APPENDIX OF LANGUAGE REFERENCE MANUAL] 1

Enter the code that designates the type of magnetic tape handler being used. For a complete list of the code types, see the NEAT/3 REFERENCE MANUAL, APPENDIX, tab 1, "Peripheral Type Codes."

The "1" in position 18 is preprinted and must be punched.

4. NUMBER OF BUFFERS TO BE RESERVED FOR THIS FILE [IF BLANK, 1 IS ASSIGNED] □

Enter the number of buffers desired. Specifying 2 buffers provides read/compute or write/compute simultaneity in processing this file. However, 2 buffers should be assigned to slower peripherals first; that is, the printer, punched card reader, punched tape reader, etc. Then if memory space permits, assign multiple buffers to the magnetic tape file.

More than 2 buffers may be assigned to provide greater read/compute or write/compute simultaneity.

If this buffer is shared between two or more files, only 1 buffer may be requested. When this is done, there is no simultaneity in inputting or outputting to the files involved. (For further information on using one buffer for multiple files, see the discussion on using SAME in the location column in the NEAT/3 REFERENCE MANUAL, INTRODUCTION AND DATA, tab 3, "Data Layout Sheets."

"1" is assumed if no entry is made.

5. FILE USAGE [ENTER S FOR SOURCE FILE; D FOR DESTINATION; P FOR PIGGYBACK FILE; R FOR A DISC SOURCE-DESTINATION FILE]. □

Enter "S" if the file is to be a source file in the run. In Father-Son processing the old master file and the transaction file are source files.

Enter "D" if this file is to be a destination file. Most files are initially created as destination files. In Father-Son processing, the new master file is a destination file.

Enter "P" if this file is to be a piggyback file. If a file is created in segments (such as A through M one day, N through Z the next day), declare this file a piggyback file each time a segment is added. In this manner a file may be extended without rewriting the entire file.

"R" is not used since source-destination files are not applicable to magnetic tape.

6. TYPE OF BLOCKING – ENTER 2 FOR MULTI-RECORD BLOCKS, 1 FOR SINGLE RECORD BLOCKS. [2]

Enter "1" if each block is to contain only one record.

Enter "2" if each block may contain more than one record. This is the usual type of blocking.

The type of blocking desired is determined by size of the record and by the number of characters that a block can accept. For example, if the record length is 100 characters, up to of 20 records can be placed in a block on the NCR Century 100.

7. RECORD LENGTH [OR MAXIMUM SIZE IF VARIABLE LENGTH RECORDS]. [0,2,5,0]

Enter the number of characters in each fixed-length record if all records are the same length. If the record length is variable, enter the number of characters in the maximum-size record, including the variable length indicator and the 3 characters for padding (on NCR Century 100 only). The maximum-size record, when using tape on the NCR Century 100 system, is 2048 characters. The maximum on the NCR Century 200 is 9,999 characters if interchange is not desired or 2048 characters if interchange is desired.

If the record length was 250 characters long, 0250 would be entered.

8. TYPE OF RECORDS [ENTER F FOR FIXED LENGTH; V FOR VARIABLE LENGTH WITH BINARY INDICATORS; D FOR VARIABLE LENGTH WITH DECIMAL INDICATORS]. □

Enter "F" if the records are fixed-length.

Enter "V" if the records are variable-length and if the variable length indicator is in binary on the tape.

Enter "D" if the records are variable-length and the variable length indicator is in decimal on the tape.

If variable-length records are used, a variable length indicator (VLI) must be defined on the data layout sheets as the first field in the record. The VLI field may be two or four characters long. This indicator contains a number that specifies the number of characters in a record (including the VLI itself). It is used by the software to present complete records to the program.

If interchange is desired, the following rules apply:

- The VLI field must be the first field in the record.
- The VLI field must be four characters long.
    - On tape, or in memory, the first two characters contain the VLI in binary, with the next two characters reserved or,
    - On tape, all four characters contain the VLI in decimal. In memory the first two characters of the field contain the VLI in binary.

If interchange is not desired, the VLI field may be two or four characters long. (The two character VLI is normally used.) The following rules apply for a 2-character VLI:

- The VLI field must be the first field in the record.
- The VLI field must be two characters long with the VLI in binary on tape and in memory.

The rules that apply for a 4-character VLI are the same as for interchange tapes.

9. IF VARIABLE, IS MAXIMUM PACKING TO BE PROVIDED ON OUTPUT? [Y OR N] 	Y

Enter "Y" if maximum packing of records is desired. If maximum packing is specified, the space remaining in a block is compared to the actual length of the record to be inserted. The record is inserted if this space is capable of accepting it.

To reduce the number of input or output operations, and to best use tape space with variable-length records, maximum packing should be specified. However, such a selection requires more memory space for both additional coding and a work area. The work area option must be used with the PUT instruction when maximum packing is indicated. See the NEAT/3 REFERENCE MANUAL, INSTRUCTIONS, tab 1, "PUT Instruction."

Enter "N" if maximum packing of records is not desired. If maximum packing is not specified, the space remaining in a block is compared to the maximum indicated length of a variable-length record. If this space is incapable of accepting a maximum-size record, the block is written out with that space unused.

Maximum packing may not be desired if memory space is critical because of program length. The work area option may be excluded from the PUT instruction when "N" is entered here.

Enter "N" if single-record blocking was specified in answer to Question 6 of this sheet.

10. MAXIMUM BLOCK LENGTH [DISC – ON CHAINED FILES, THE BLOCK LENGTH MUST INCLUDE TEN CHARACTERS FOR THE BLOCK HEADER INDICATOR] [MAGNETIC TAPE – THE BLOCK LENGTH MUST INCLUDE THE BLOCK LENGTH INDICATOR IF PRESENT]. 0,5,0,7

Enter the total number of characters in a block. On the NCR Century 100 and on the NCR Century 200, the minimum block length allowed is 80 characters, the length of a label.

The maximum block length permitted on the NCR Century 100 is 2048 characters; the maximum block length permitted on the NCR Century 200 is 9,999 characters (2048 for interchange). These limits include all header and trailer offsets. When computing the block length, be sure to include the padding characters (if variable-length records are used on NCR Century 100) and BLI characters (if used).

11. IS A RESCUE POINT DESIRED AT EACH END OF SECTION? [Y OR N] [IF BLANK, N IS ASSUMED] □

Enter "Y" if a rescue point is desired each time this file reaches the end of a section. If this is a destination file, the rescue point is placed at the beginning of the next section. If this is a one-section destination file, no rescue point is possible or necessary. If this is a source file, the rescue point is placed on the Standard Rescue File. (See next entry for additional information on the Standard Rescue File.)

"N" is assumed if no entry is made.

Generally, if a program uses source and destination files, the programmer specifies that the rescue point be taken for the file that reaches end-of-section most often. Rescue points can be specified for a source file even if it is the only data file in a run, as in a trial-balance run. This assumes that a standard rescue file exists.

12. IS THIS THE STANDARD RESCUE FILE? [Y OR N] □
[DISC – IF YES, THE FILE MAY BE USED FOR RESCUES
ONLY] [IF BLANK, N IS ASSUMED]

Enter "Y" if this file is to be the Standard Rescue File. On tape, a normal data file may simultaneously be used as the standard rescue file. When a data file is used as the standard rescue file, both data and rescue dumps are placed in the same file. The file specification sheets are filled in the normal manner and "Y" is entered here.

If this file is to be a Standard Rescue File only, a set of file specification sheets must be filled out according to the following table:

Sheet 1

QUES.	ENTRY	
1, 2, 3		Standard entries.
4	1	1 buffer.
5	D	Destination file.
6	1	Single-record blocks.
8	F	Fixed-length records.
10	0080	A block length of 80 characters is needed to read and write labels on opening the file. The rescue routine writes portions of memory in 512 character segments.
11	N	No rescue is possible on a rescue file.
12	Y	Standard rescue file.
13, 14, 15		Standard entries.
18, 19		Standard entries.
20	N	No block length indicators are desired.

Sheet 2

This sheet may be omitted when a magnetic tape file is used as a standard rescue file only.

Sheet 3

Sheet 3 is not used for magnetic tape files.

For further information concerning rescue, restart, and Standard Rescue files, see the Operating System Manual.

"N" is assumed if no entry is made.

13. TYPE OF DATING PERIOD. □

14. ACCEPTABLE PERIOD [EARLIEST] FOR SOURCE, PIGGYBACK, AND SOURCE-DESTINATION FILES. NOT USED ON DESTINATION FILES. □

15. RETENTION PERIOD FOR DESTINATION, PIGGYBACK, OR SOURCE-DESTINATION FILES. OPTIONALLY, THE LATEST ACCEPTABLE PERIOD FOR SOURCE FILES. □

Questions 13, 14, and 15, which deal with dates for file protection, are interrelated and are therefore explained together.

The dates entered here instruct the software to accept only those files created within a certain period, thereby guaranteeing that the current version of a file is processed. The dates further define how long a file must be saved for backup. Files may not be written over until this backup date (retention period) has expired; however, files with an expired retention period are acceptable as source files.

Dates are specified as either work days (WD), work weeks (WW), or work months (WM), relative to the current date. These work periods are based on a five-day week (with holidays considered).

Generation numbers (GEN) may not be used with magnetic tape files.

- Example of Dates

To illustrate how dates might be used, assume the following:

- A destination file (File A) is being created today as a product of a Father-Son update run.
- File A is to be saved for two weeks as a backup file.
- File A is to be used as a source file in exactly one week when the program is run again.
- Today is Tuesday, January 3. (Use the following calendar in the example.)

JANUARY 196—

S	M	T	W	T	F	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

To comply with these assumptions, the dates on the file specification sheets for destination file A might be filled in as follows:

13. TYPE OF DATING PERIOD WW

14. ACCEPTABLE PERIOD [EARLIEST] FOR SOURCE, PIGGY-BACK, AND SOURCE-DESTINATION FILES. NOT USED ON DESTINATION FILES.

15. RETENTION PERIOD FOR DESTINATION, PIGGYBACK, OR SOURCE-DESTINATION FILES. OPTIONALLY, THE LATEST ACCEPTABLE PERIOD FOR SOURCE FILES. 002

These entries indicate that the file is to be kept for two work-week periods beginning next Monday. (Work weeks begin on Monday and end on Friday.) Therefore, it may be written over after January 20.

When File A is used next week (January 10) as a source file, it must meet the programmer's qualifications for source files as specified on the file specification sheets.

The dates on the file specification sheets for source File A might be filled in as follows:

13. TYPE OF DATING PERIOD. WW

14. ACCEPTABLE PERIOD [EARLIEST] FOR SOURCE, PIGGY-BACK, AND SOURCE-DESTINATION FILES. NOT USED ON DESTINATION FILES. 001

15. RETENTION PERIOD FOR DESTINATION, PIGGYBACK, OR SOURCE-DESTINATION FILES. OPTIONALLY, THE LATEST ACCEPTABLE PERIOD FOR SOURCE FILES.

These entries indicate that a file is to be accepted as a source file if it was created during the previous work week. Therefore, a source file is acceptable on January 10 if it was created in the period, between January 2 and January 6. File A was created on January 3 and is acceptable.

JANUARY 196–

S	M	T	W	T	F	S
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

Question 15 is optional for source files and is used to further define an acceptable file. For example, 002 might be entered for Question 14 and 001 for Question 15, with Question 13 remaining WWØ. According to these entries, a file is to be accepted if it was created in a period that started at the beginning of two work weeks back, and ended at the end of 1 work week back. If the program was run on January 24, a source file created between January 9 and January 20 would be acceptable.

If Question 15 was left blank, and 14 remained 002, a source file created between January 9 and January 13 only would be accepted.

- Further Examples of Using Dates for File Protection

DESTINATION FILE EXAMPLE

13. TYPE OF DATING PERIOD. W,D,

14. ACCEPTABLE PERIOD [EARLIEST] FOR SOURCE, PIGGYBACK, AND SOURCE-DESTINATION FILES. NOT USED ON DESTINATION FILES.

15. RETENTION PERIOD FOR DESTINATION, PIGGYBACK, OR SOURCE-DESTINATION FILES. OPTIONALLY, THE LATEST ACCEPTABLE PERIOD FOR SOURCE FILES. 0,0,2

The created file is protected for two additional work days; that is, it may not be written over (destroyed) until the third work day after its creation.

DESTINATION-SCRATCH-FILE EXAMPLE

13. TYPE OF DATING PERIOD. S,C,R

14. ACCEPTABLE PERIOD [EARLIEST] FOR SOURCE,PIGGYBACK,AND SOURCE-DESTINATION FILES, NOT USED ON DESTINATION FILES.

15. RETENTION PERIOD FOR DESTINATION, PIGGYBACK, OR SOURCE-DESTINATION FILES. OPTIONALLY, THE LATEST ACCEPTABLE PERIOD FOR SOURCE FILES.

This indicates that the file created is a scratch file. In effect, it is obsolete when created and may be written over immediately. Although obsolete, the file could be used as a source file in another run.

SOURCE–SCRATCH–FILE EXAMPLE

13. TYPE OF DATING PERIOD. SCR

14. ACCEPTABLE PERIOD [EARLIEST] FOR SOURCE, PIGGYBACK, AND SOURCE–DESTINATION FILES. NOT USED ON DESTINATION FILES. □

15. RETENTION PERIOD FOR DESTINATION, PIGGYBACK, OR SOURCE–DESTINATION FILES. OPTIONALLY, THE LATEST ACCEPTABLE PERIOD FOR SOURCE FILES. □

This indicates that the program should accept a scratch file.

SOURCE–FILE EXAMPLE

13. TYPE OF DATING PERIOD. WD

14. ACCEPTABLE PERIOD [EARLIEST] FOR SOURCE, PIGGYBACK, AND SOURCE–DESTINATION FILES, NOT USED ON DESTINATION FILES. 005

15. RETENTION PERIOD FOR DESTINATION, PIGGYBACK, OR SOURCE–DESTINATION FILES. OPTIONALLY, THE LATEST ACCEPTABLE PERIOD FOR SOURCE FILES. □

This indicates that the program accepts a source file if it was created five work days back. Only a file created on that day is accepted.

PIGGYBACK—FILE EXAMPLE

13. TYPE OF DATING PERIOD. W,W,

14. ACCEPTABLE PERIOD [EARLIEST] FOR SOURCE, PIGGYBACK, AND SOURCE-DESTINATION FILES. NOT USED ON DESTINATION FILES. 0,0,0

15. RETENTION PERIOD FOR DESTINATION, PIGGYBACK, OR SOURCE-DESTINATION FILES. OPTIONALLY, THE LATEST ACCEPTABLE PERIOD FOR SOURCE FILES. 0,0,2

This indicates that a file created the same week is to be extended. All additional segments to the file will have the same creation and expiration date as the original segment.

If the file cannot be located, the operator is given the option of creating a new one. If a new file is created, the answer to Question 15 is used to establish the expiration date for it. In this way, the same program may be used to create a piggyback file and extend it.

- Virtual Date and Actual Date

The date the file was created is obtained from an entry made to the system at the start of each workday. This entry contains two dates: the actual date and the virtual date.

- Actual date is today's date.
- Virtual date is the date the program is scheduled to be run.

Generally, the actual date and the virtual date are the same. However, an unavoidable interruption could interfere with a program being run on a scheduled date. In this case, the program could be run the next day using the previous day's date as the virtual date.

- Example of Virtual and Actual Date

Phase 1: The program is to write File A and save it for one additional day. This program is normally run on January 4; but since the processing could not be done on that day, the program is run on January 5.

The actual and virtual dates are entered into the system at the beginning of the day as follows:

Actual date - January 5   Virtual date - January 4

When File A is created, the HDR1 label will contain the following creation and expiration dates:

Date created - January 4   Date expires - January 6

The files written on January 5 may now be used in Phase 2.

Phase 2: The program is to use File A one day after it is created. This program is usually run on January 5.

New actual and virtual dates are entered after completion of the programs usually run on January 4.

Actual date - January 5   Virtual date - January 5

File A is acceptable since it was written using the virtual date January 4, and this is actually January 5. Any files written in Phase 2 contain January 5 as the date created.

Programs may be run over a period of days using virtual dates until the work is back on schedule. When the system is back on schedule, all files contain valid dates that are acceptable to their associated programs.

16. FILE NAME   MAGZFILE

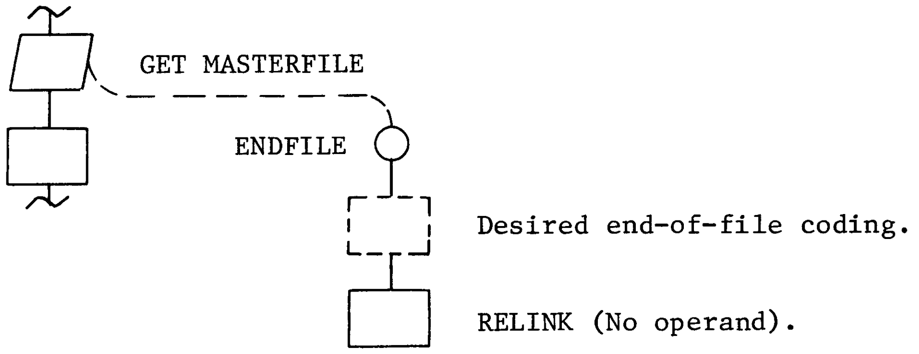
Enter the file name as it is to appear in the HDR1 label. For Father-Son processing this name should be different than the file reference name. (For further details, see the NEAT/3 REFERENCE MANUAL, FILES, "NCR Century File Concepts.")

17. END OF FILE EXIT [NOT USED ON DESTINATION FILES] OR ERROR EXIT FOR RANDOM MACROS.   ENDFILE

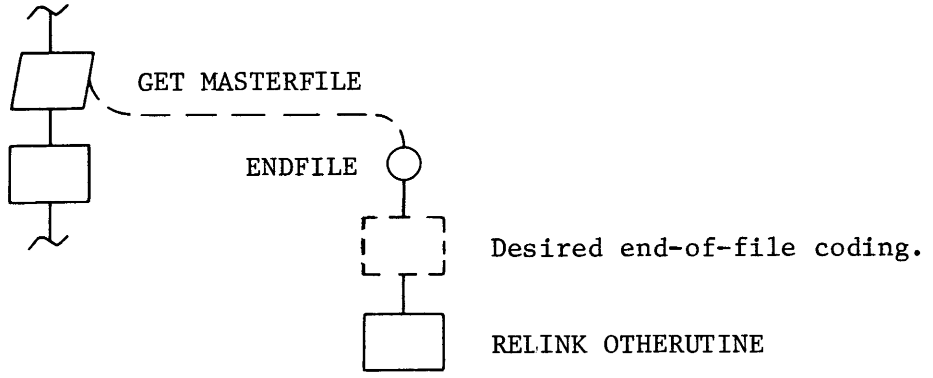
END-OF-FILE EXIT

Enter the name of the routine to be given control when end of a source file is reached.

If the programmer wishes to return control to the instruction following the one that caused the link to the end-of-file routine, he may do so by using a RELINK instruction without an operand. (A LINK or BRANCH instruction may not be used for this purpose.)

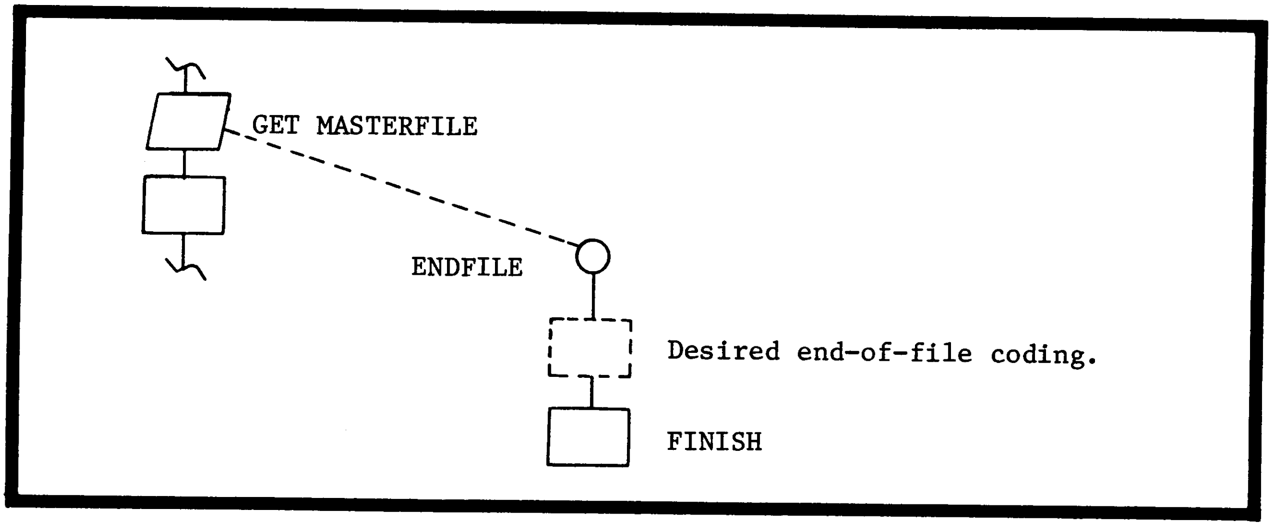


If the programmer wishes to return control to a different instruction or routine, he may do so by using a RELINK instruction that names the desired routine as an operand. The LINK or BRANCH instruction may not be used for this purpose.



For further details concerning LINK and RELINK with an operand, see the NEAT/3 REFERENCE MANUAL, INSTRUCTIONS, tab 2, "Link and Relink Instructions."

The programmer need not relink from his end-of-file routine but may continue his coding within the routine and complete the program with a FINISH instruction.



If this is a piggyback file, the routine referenced here allows the program to read existing user trailer labels. The file is opened, tape is advanced to the end of the existing portion of the file, and control is transferred to the routine referenced here.

Once the labels have been read, the programmer must RELINK to the software. If a tape mark is encountered while reading these labels, an automatic RELINK to the software takes place. (Tape marks indicate the end of a group of labels.)

The software automatically repositions the tape for proper extension of the file. The existing EOF labels, UTL labels, and their associated tape marks are written over when the file is extended.

If this is an NCR 315 source file, control is transferred to the routine referenced here each time a tape mark is encountered.

NOTE

Once an end-of-file exit is taken, the last record in the input buffer is no longer accessible to the program.

18. PRIMARY SYMBOLIC UNIT DESIGNATOR.

M

Enter the Symbolic Unit Designator of the unit containing the first section of the file.

19. TYPE OF REWIND DESIRED ON CLOSING? [ENTER N FOR NO REWIND; R FOR REWIND ONLY; OR U FOR REWIND AND UNLOAD] □

Enter "N" if the tape is not to be rewound when the end of this file is reached. The programmer may enter "N" if a multiple-file reel is being processed and if the next desired file is located beyond the end of the current file.

Enter "R" if the tape is to be rewound when the end of this file is reached. The tape is rewound to the beginning of the tape (reel).

Enter "U" if the tape is to be rewound and unloaded when the end of this file is reached. The tape will be rewound and the unit will be placed in a non-ready state.

20. ARE BLOCK LENGTH INDICATORS TO BE CHECKED/CREATED? [Y OR N] [IF BLANK, N IS ASSUMED] □

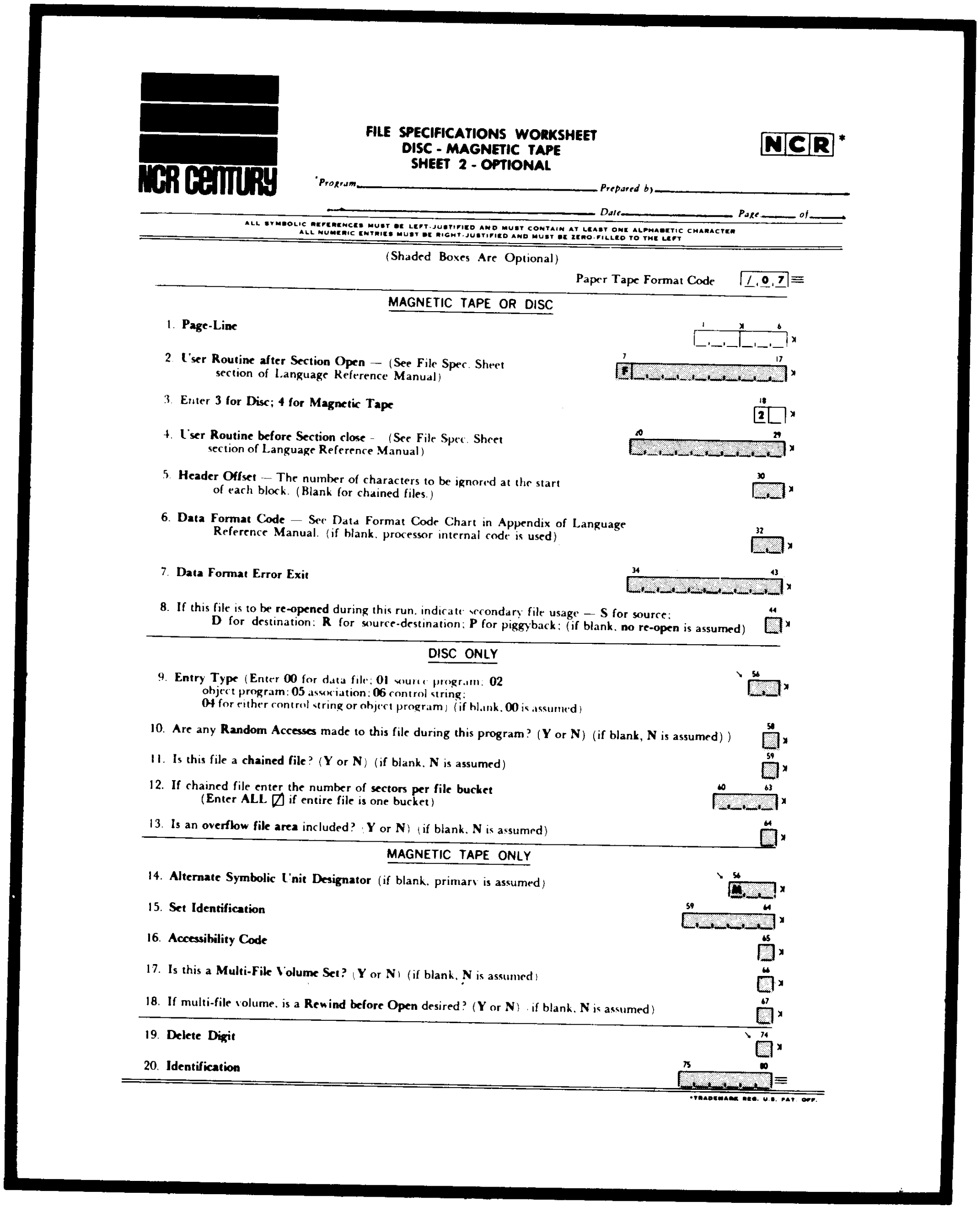
Enter "Y" if Block Length Indicators (BLI) are to be written or checked automatically. The compiler will automatically set up a header offset of two characters. If the BLI's are in a four-character field, enter "Y" here and specify a header offset (Sheet 2, Question 5) of two.

Enter "N" if BLI's are not to be written or checked automatically. If this is a source file and if BLI's exist but are not to be checked, enter "N" here and specify a header offset equal to the length of the BLI field (2 or 4 characters).

"N" is assumed if no entry is made.

SHEET 2

The information on this sheet is optional. If none of the options are desired, the entire sheet may be eliminated.



NCR CENTURY

FILE SPECIFICATIONS WORKSHEET
DISC - MAGNETIC TAPE
SHEET 2 - OPTIONAL

NCR*

*Program__________ Prepared by__________

Date__________ Page____ of____

ALL SYMBOLIC REFERENCES MUST BE LEFT-JUSTIFIED AND MUST CONTAIN AT LEAST ONE ALPHABETIC CHARACTER
ALL NUMERIC ENTRIES MUST BE RIGHT-JUSTIFIED AND MUST BE ZERO-FILLED TO THE LEFT

(Shaded Boxes Are Optional)

Paper Tape Format Code /07

MAGNETIC TAPE OR DISC

1. **Page-Line** (1–6)
2. **User Routine after Section Open** — (See File Spec. Sheet section of Language Reference Manual) (7–17) F
3. Enter **3 for Disc; 4 for Magnetic Tape** (18) 2
4. **User Routine before Section close** - (See File Spec. Sheet section of Language Reference Manual) (20–29)
5. **Header Offset** — The number of characters to be ignored at the start of each block. (Blank for chained files.) (30)
6. **Data Format Code** — See Data Format Code Chart in Appendix of Language Reference Manual. (if blank, processor internal code is used) (32)
7. **Data Format Error Exit** (34–43)
8. If this file is to be **re-opened** during this run, indicate secondary file usage — **S** for source; **D** for destination; **R** for source-destination; **P** for piggyback; (if blank, **no re-open** is assumed) (44)

DISC ONLY

9. **Entry Type** (Enter **00** for data file; **01** source program; **02** object program; **05** association; **06** control string; **04** for either control string or object program) (if blank, **00** is assumed) (56)
10. Are any **Random Accesses** made to this file during this program? (**Y** or **N**) (if blank, **N** is assumed)) (58)
11. Is this file a **chained file**? (**Y** or **N**) (if blank, **N** is assumed) (59)
12. If chained file enter the number of **sectors per file bucket** (Enter ALL ☐ if entire file is one bucket) (60–63)
13. Is an **overflow file area** included? (**Y** or **N**) (if blank, **N** is assumed) (64)

MAGNETIC TAPE ONLY

14. **Alternate Symbolic Unit Designator** (if blank, primary is assumed) (56) M
15. **Set Identification** (59–64)
16. **Accessibility Code** (65)
17. Is this a **Multi-File Volume Set**? (**Y** or **N**) (if blank, **N** is assumed) (66)
18. If multi-file volume, is a **Rewind before Open** desired? (**Y** or **N**) (if blank, **N** is assumed) (67)
19. **Delete Digit** (74)
20. **Identification** (75–80)

*TRADEMARK REG. U.S. PAT. OFF.

The programmer should fill in the header, page-and-line number (Question 1), and the identification tag (positions 75-80) as defined in the NEAT/3 REFERENCE MANUAL, INTRODUCTION AND DATA, tab 3, "Programming Worksheets." The paper tape format code is preprinted on this sheet and must be punched if paper tape is used for input to the compiler.

2. USER ROUTINE AFTER SECTION OPEN – [SEE FILE SPEC. SHEET SECTION OF LANGUAGE REFERENCE MANUAL]

F U H L L A B E L

This optional entry contains the reference name of the routine to be given control immediately after the second and subsequent sections of the file are opened. Control is also transferred here after the first section is opened if an OPEN instruction is used for this file. This routine may contain coding to write or read User Header Labels (UHL). The last instruction in the routine must be RELINK without an operand. For further details on labels, see the NEAT/3 REFERENCE MANUAL, FILES, tab 2, "Magnetic Tape Files."

LPUT and LGET are the only I/O instructions that may be executed in this routine.

3. ENTER 3 FOR DISC; 4 FOR MAGNETIC TAPE

2	4

Enter "4" for magnetic tape.

The number "2" is preprinted in position 18 and must be punched.

4. USER ROUTINE BEFORE CLOSE – [SEE FILE SPEC. SHEET SECTION OF LANGUAGE REFERENCE MANUAL]

U T L L A B E L

This optional entry contains the reference name of the routine to be given control before each section of the file is closed -- with the exception of the last section. Control is also transferred here before the last section is closed if a CLOSE instruction is used for this file. This routine may contain coding to write or read User Trailer Labels (UTL). The last instruction in the routine must be RELINK without an operand. For further information on labels, see the NEAT/3 REFERENCE MANUAL, FILES, tab 2, "Magnetic Tape Files."

LPUT and LGET are the only I/O instructions that may be executed in this routine.

5. HEADER OFFSET – THE NUMBER OF CHARACTERS TO BE IGNORED AT THE START OF EACH BLOCK. [BLANK FOR CHAINED FILES.]

▨▨

This optional entry contains the number of characters to be ignored at the beginning of each block. These characters will not be presented to the program by the GET instruction.

When Block Length Indicators (BLI) are used, a header offset of 2 is automatically assumed. If a 4-character BLI is used, 2 must be indicated here. The indicated header offset (2) plus the assumed header offset of 2 equals a total header offset of 4.

6. DATA FORMAT CODE – SEE DATA FORMAT CODE CHART IN APPENDIX OF LANGUAGE REFERENCE MANUAL. [IF BLANK, PROCESSOR INTERNAL CODE IS USED] ▨

This optional entry contains a code that signifies the code set of the data on tape. If left blank, NCR Century internal code is assumed. This is the usual code for magnetic tape files.

If NCR Century internal code is not desired, enter a code as indicated on the data format code chart. See the NEAT/3 REFERENCE MANUAL, APPENDIX, tab 1, "Data Format Codes."

7. DATA FORMAT ERROR EXIT ▨

This optional entry is used only if translation was asked for in answer to Question 6. If an illegal character is detected during translation, control is given to the routine referenced here.

8. IF THIS FILE IS TO BE RE-OPENED DURING THIS RUN, INDICATE SECONDARY FILE USAGE – S FOR SOURCE; D FOR DESTINATION; R FOR SOURCE-DESTINATION; P FOR PIGGYBACK; [IF BLANK, NO RE-OPEN IS ASSUMED] □

If this file is reopened during the run, enter the letter that indicates the type of file desired on reopening. Multiple reopens are permitted; however, the second reopen must declare the file type as being the same as at initial open (Question 5, Sheet 1).

Enter "S" if the file is first reopened as a source file.
Enter "D" if the file is first reopened as a destination file.
Enter "P" if the file is first reopened as a piggyback file.

"R" may not be entered since source-destination files are not applicable to magnetic tape.

Entries 9-13 are for disc only.

14. ALTERNATE SYMBOLIC UNIT DESIGNATOR [IF BLANK, PRIMARY IS ASSUMED] M

This optional entry contains the Symbolic Unit Designator of an alternate unit. The software will look to the unit named here for a continuation of the file when the section on the unit specified in Question 18 of Sheet 1 is exhausted. When the section of the file on the alternate unit (specified here) is exhausted, the software will look back to the first unit for a continuation of the file.

If this entry is left blank, the section of the file initially mounted on the unit specified in answer to Question 18 of Sheet 1 is processed. Then, the operator is signaled to change the reel on that unit.

15. SET IDENTIFICATION □□□□□□

Enter the 6-character name to be given to the volume set if this is a destination file on a multi-file reel (or in a multi-file, multi-reel set).

If this is a source file on a multi-file reel (or in a multi-file, multi-reel set), enter the name of the set.

If left blank, the volume serial number in the Volume Header Label is used.

16. ACCESSIBILITY CODE □

This optional entry contains a one-character code that is placed in the File Header Label when the file is created.

Presently, the software makes no use of the accessibility code.

17. IS THIS A MULTI-FILE VOLUME SET? [Y OR N] [IF BLANK, N IS ASSUMED] □

18. IF MULTI-FILE VOLUME, IS A REWIND BEFORE OPEN DESIRED? [Y OR N] [IF BLANK, N IS ASSUMED] □

These two optional entries are interrelated and will be explained together.

Enter a "Y" in answer to Question 17 if the reel mounted contains (or is to contain) more than one file, or if the reel is part of a multi-file, multi-reel set.

Leave blank if the reel mounted contains (or is to contain) only one file.

If Question 17 was answered "Y", Question 18 is applicable.

- Enter "N" in answer to Question 18 if the desired file is beyond (or is to be created beyond) the current position of the tape.
- Enter "Y" answer to Question 18 if the position of the desired file is not known; then, the software will rewind the tape before searching for the desired file (or file location).