Honeywell

SERIES 600/6000

SOFTWARE

INTEGRATED DATA STORE REFERENCE MANUAL

Honeywell

SERIES 600/6000

INTEGRATED DATA STORE REFERENCE MANUAL

SUBJECT:

General Description, Data Organization, Source Language, Programming Information, Operational Characteristics, and Capabilities of the Integrated Data Store (I-D-S) System.

SPECIAL INSTRUCTIONS:

This manual, order number BR69, Rev. 0, is a reprint of CPB-1565B, dated January 15, 1971. The new order number is assigned to be consistent with the overall Honeywell publications numbering system. The contents of this reprinted manual are the same as for CPB-1565B. Both CPB-1565B and BR69, Rev. 0, completely supersede the previous edition (CPB-1565A) and incorporate the information published in TIB 1565A-1, 1565A-2, and 1565A-3. New features implemented in *Series 600 System Development Letter* 3.3 are also included. New information and changes since the last edition are indicated by change bars; deletions are indicated by asterisks.

INCLUDES UPDATING PAGES PUBLISHED AS ADDENDUM A IN AUGUST, 1971, WHICH INCLUDE NEW FEATURES IMPLEMENTED IN SERIES 600 SYSTEM DEVELOPMENT LETTER 4.0 AND SERIES 6000 SYSTEM DEVELOPMENT LETTER B.

DATE:

January, 1971

ORDER NUMBER: BR69 (Formerly CPB-1565)

Rev. 0

Preface

This manual describes the Integrated Data Store (I-D-S) system, which is an information-oriented method of integrating the operating function of a business. I-D-S reduces the system and programming cost associated with implementing some other types of integrated business systems. It uses direct-access storage as an extension of memory and provides an efficient data organization technique.

Since I-D-S is used to extend the functions of the COBOL language, the reader should have a working knowledge of COBOL before using this manual.

The I-D-S program is identified by catalog numbers CD600H5.100 and CD600H7.000 in the Program Library.

Contents

Page

Page

Page

Illustrations

Figure		Page

1. Introduction

Integrated Data Store (I-D-S) is an information-oriented method of integrating the operating functions of a business. Its use reduces the high systems and programming costs associated with implementing some other types of integrated business systems. As a general-purpose system, it uses mass random access storage as an extension of memory and provides an efficient data organization technique. In addition, a simple but effective language is used to operate the system.

Present procedural languages offer programming convenience in field and sequential record processing. However, they are inadequate for processing records in the random environment of mass storage. The I-D-S language provides a simplified means for record processing in the environment of mass random access storage.

Language statements such as those for read/write operations produce serial rather than random actions. Ordinarily, the burden of organizing data records and designing the logic involved in processing and maintaining these records is placed upon the programmer. Many of these processing and maintenance functions are performed automatically by the I-D-S software system.

2. Data Organization

Data organization refers to the interrecord relationships established within the I-D-S data-file. The record is the basic unit of data. Record association is achieved through chains which provide cross-reference linkages between records. These chains provide the integrating force which is implied in the name Integrated Data Store.

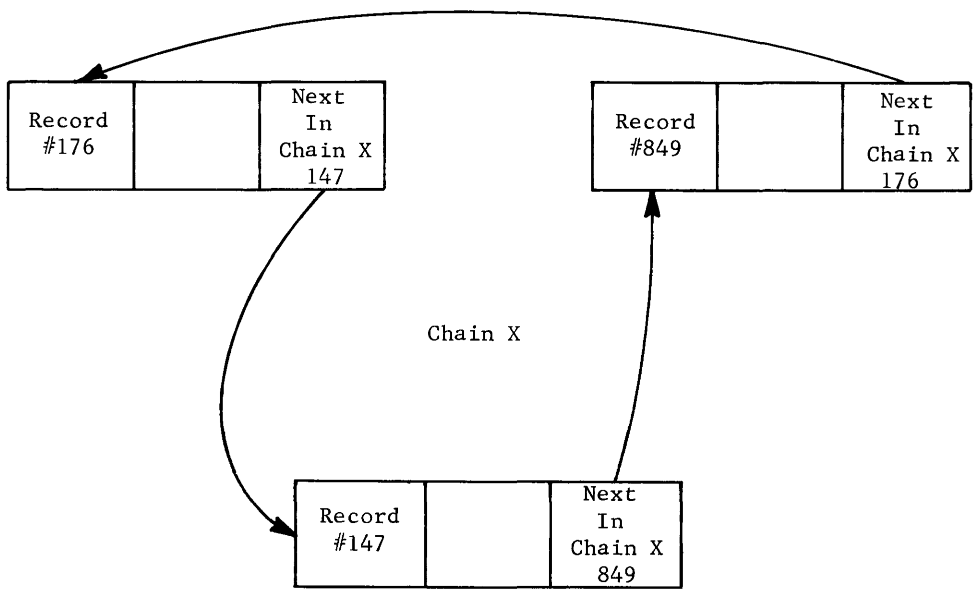


Figure 1. Chain Association

I-D-S records are stored only once. Conventional approaches to file organization often require records, or certain fields in the records, to be repeated in several files. With the ability to integrate records into any number of chains (as required by the system), the same data fields are available no matter which of its chains are processed. This technique has five important advantages:

1. Additional space required for duplicate records is eliminated, resulting in a reduction in the total storage capacity required.

2. The work of data maintenance is greatly reduced, as there is only one record to retrieve and modify.

3. The possibility that one copy of a record will not be properly modified is eliminated. Since there is only one copy, any incorrect information will be quickly recognized and corrected.

4. All reports drawn from the file will be consistent, since there is only one set of facts (records).

5. Due to the linking capability, the homogeneity of a file needs no longer to exist and records of different types may be intermixed to achieve a better utilization of the storage capacity.

I-D-S CHAINS

An I-D-S chain is illustrated in Figure 2.

All records in a chain are associated in a closed loop, with the last detail linked back to the master. Its characteristics are:

Contains one master record and any number of detail records
Links records together in an endless loop
Associates related records in meaningful sequences

Records are to be defined by the user as to their relationship within a chain--as master or detail records. These relationships are specified, when the chaining relationship is described, in the data description.

A chain is a set of records that are linked together to form a logical relationship between records.

A Master Record is the head of a set of records that make up a chain. There must be one and only one Master Record for each chain. Detail records are the other records that are members of the set or chain.

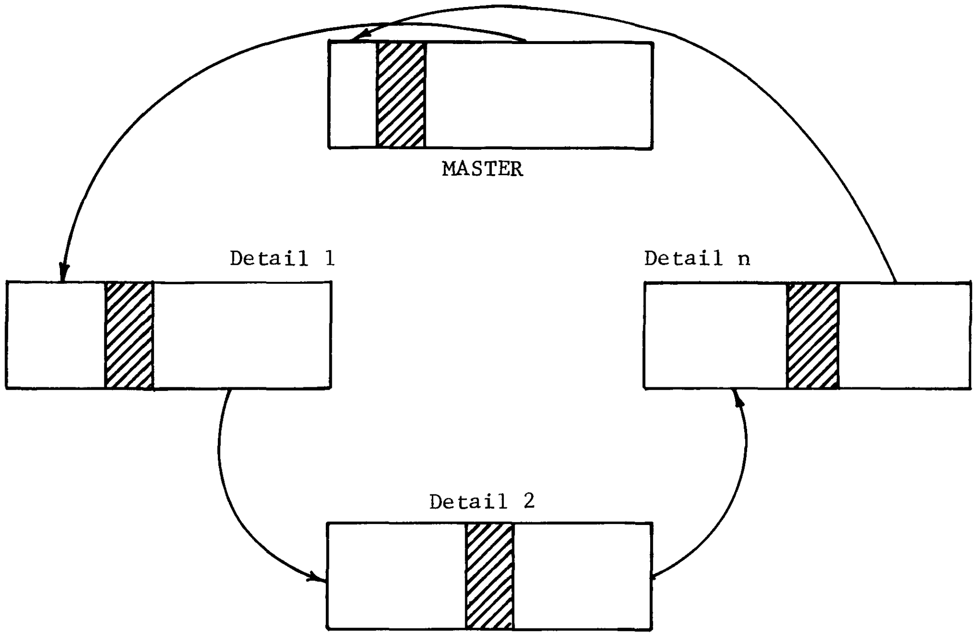


Figure 2. I-D-S Chain

Multiple Chains

Chains exist for two separate but closely related reasons. First, the source documentation or problem analysis shows that a portion of the information is often cross-referenced. An example is a personnel record with a variable number of deductions and work experiences.

This kind of information is easily structured by building a personnel master record type. Two chain types are created containing the personnel record as the master record. As many deduction records as necessary are linked into the deduction chain as details. Work experience for the employee involved is handled in a like manner. Both chains are now linked to the same master record, as shown in Figure 3.

The second case involves the logical association of several source documents. Relating all the purchase order information for a given vendor to the vendor information is an example. A purchase order chain associates all of the purchase order records with their vendor record.

I-D-S chains have several structural aspects which should be emphasized:

All similar chains are grouped by chain type.

A chain type is named with a symbolic name. There will be as many chains of the chain type as there are master records for that chain type.

Each chain can have only one master record. Its type is the same for all chains of the same type.

Any number of detail records may be in a chain. A chain may even contain more than one type of detail record.

Detail records cannot be stored unless their master record already exists in the file.

Whenever a master record is deleted, its entire chain is also removed.

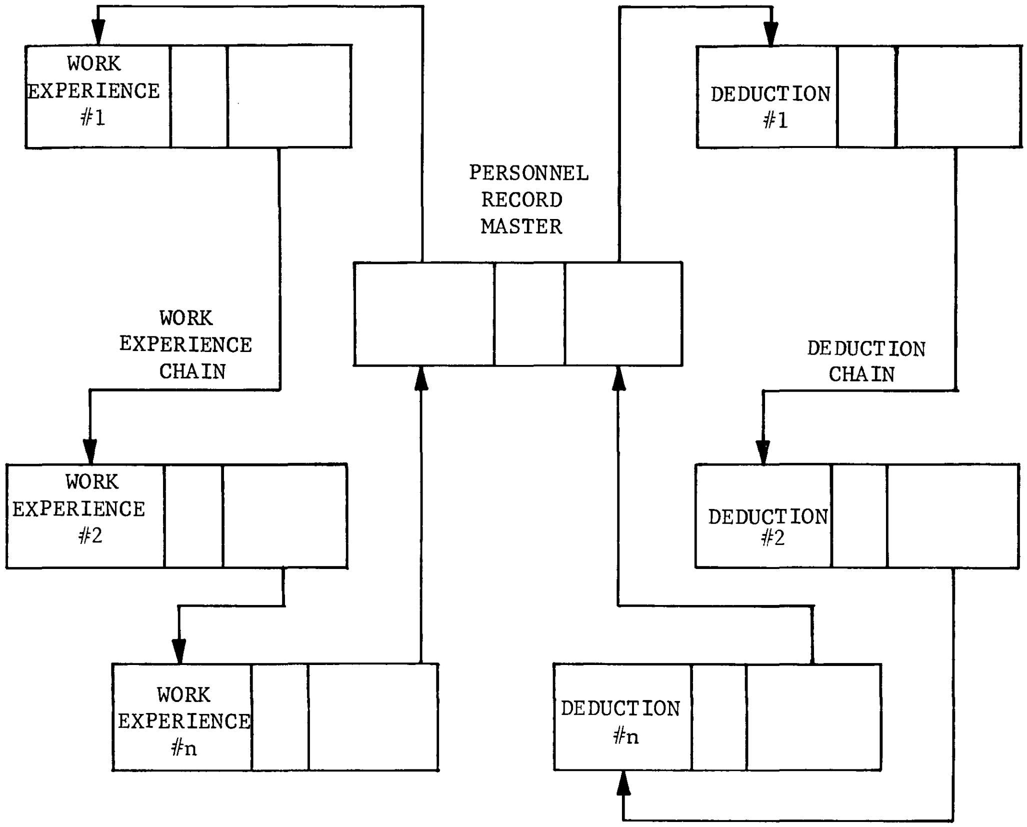


Figure 3. Master Record of Two Chains

The master record of the chain contains a code which references the first detail in the chain.

A record may be defined to be a member in as many chains as are required. It may be defined as master in one chain and detail in another.

A record cannot be defined as a detail to itself, directly or indirectly.

As records are stored in the system, they are automatically linked into their defined chains.

When a record is deleted, the chains in which it is a detail record are automatically modified to relink around the deleted record, which will eventually be physically deleted.

STRUCTURE REPRESENTATION

A special graphic technique called I-D-S shorthand has been developed to display records and their chaining relationships.

Its use is particularly important in developing an over-all view when planning a database structure. This technique (see Figure 4) uses a block shape to designate a record type--employee (record type 1) and deduction (record type 2)--and an arrow connecting two blocks to designate a chain type. The arrow points from the master to the detail, as shown in Figure 4.

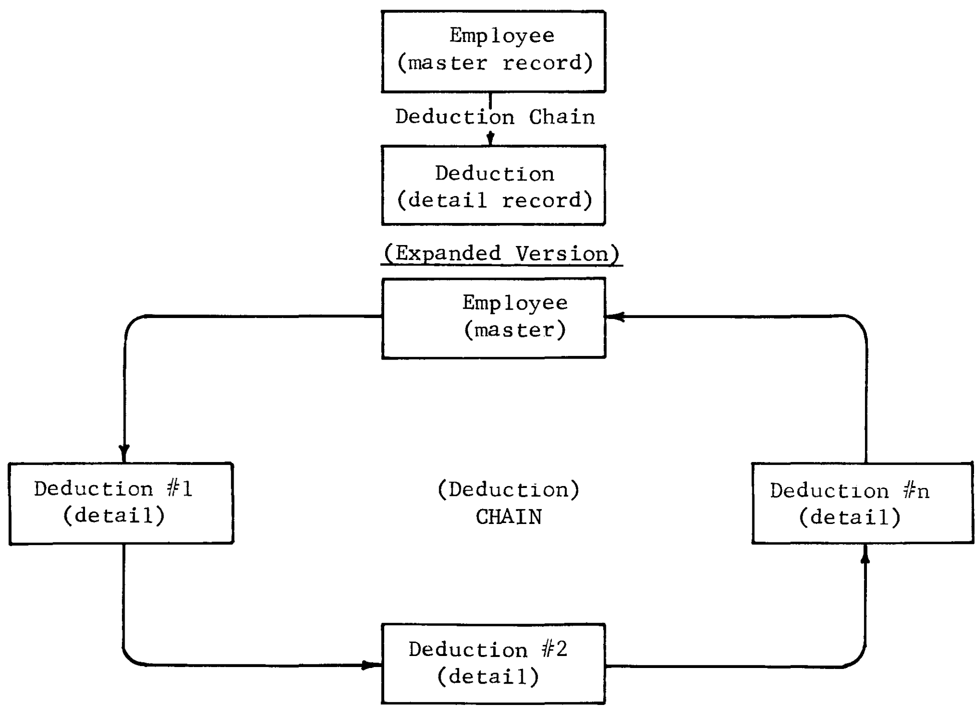


Figure 4. I-D-S Shorthand

In the foregoing example of I-D-S shorthand, the vertical block-arrow-block sequence carries the following message:

1. There are a number of records in the system of the master type (one for each employee).

2. Each of these records is the master of a chain of the specified type (deduction).

3. There are a number of records of the detail type (deductions 1, 2, 3, 4, etc.) in each such chain.

The purchase order data structure (Figure 5) shows how a vendor record and a particular order record from the example shown in Figure 6 is normally structured in the I-D-S system.

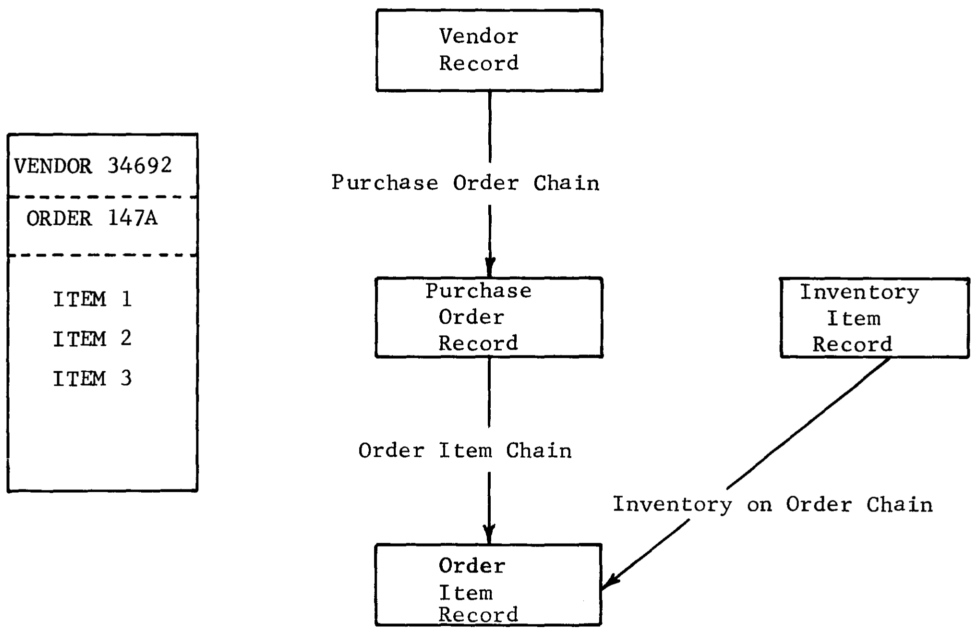


Figure 5. Purchase Order Data Structure

The purchase order contains four groups of information.

1. Information about the vendor--such as his name, address, and vendor code.

2. Information about the order--such as the order number, due date, mode of transportation, and dollar value.

3. Information about the order item--such as delivery date, quantity, unit price, and extended dollar value.

4. Information about the inventory item--such as its identification and description.

The data structure in Figure 5 shows all four groups and their chain associations with only four blocks and three arrows. To expand this structure, four different record types would be designed to carry the information contained in the four groups:

Vendor record--There would be a vendor record for every vendor with whom the business is concerned:

1. It would be the <u>master record</u> of a purchase <u>order chain</u>.

2. Thus, the vendor record is <u>only</u> a master.

Purchase order record--There would be an order record for each order currently stored in the system:

1. It would be a <u>detail</u> in the purchase <u>order chain</u>.

2. Each order, in turn, would be the <u>master</u> of an order <u>item chain</u>.

3. Thus, the purchase order record is <u>both</u> a master and a detail.

Order item record--There would be an order item record for each item on each order:

1. It would be a <u>detail</u> in the inventory on <u>order chain</u>.

2. It would be a <u>detail</u> in the order <u>item chain</u>.

3. Thus, the order item record is a <u>detail</u> in two chains.

Inventory item record--There would be an inventory record for each inventory item currently stored in the system:

It would be the master record of the inventory on order chain.

One expanded data structure concerning Vendor # 34692 for the above records is shown in Figure 6.

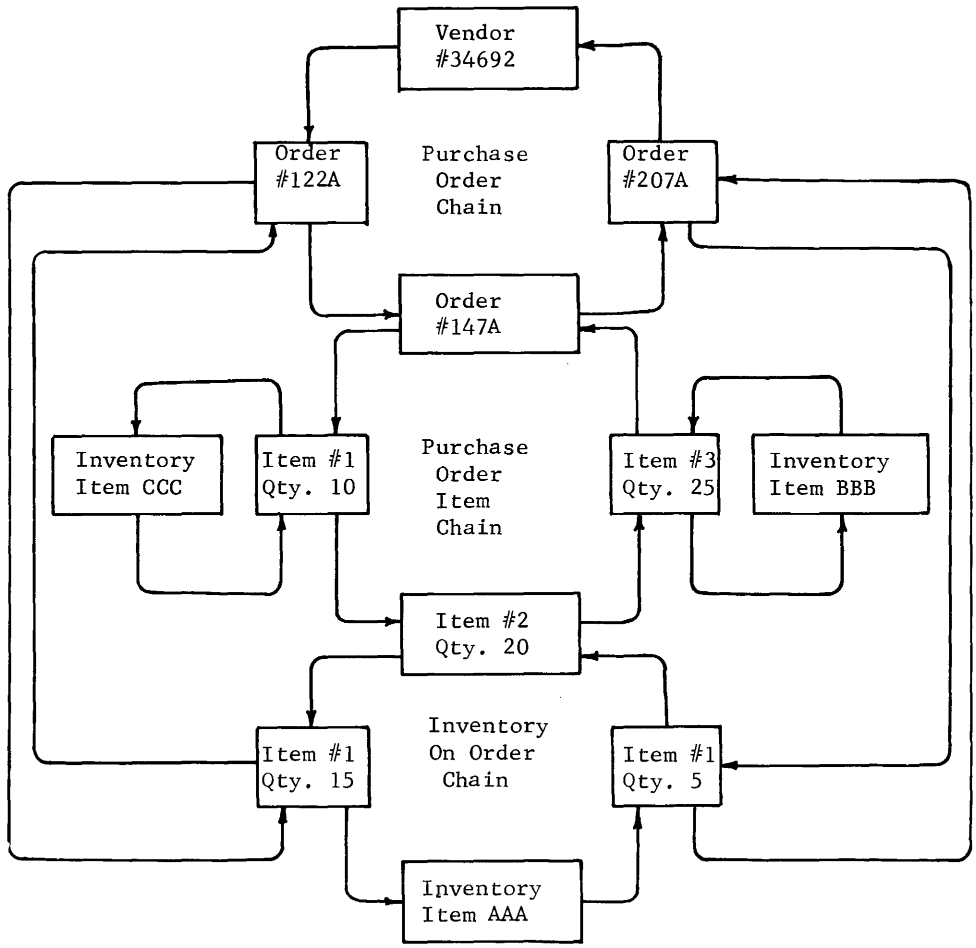


Figure 6. Chain Network

SUMMARY OF DATA STRUCTURES

By using I-D-S shorthand, very complex data structures may be presented in a condensed and understandable form. Figure 7 shows a quick summary of data structures which are legal and illegal within I-D-S. A circular definition (item 6), where the master becomes its own detail, is not allowed.

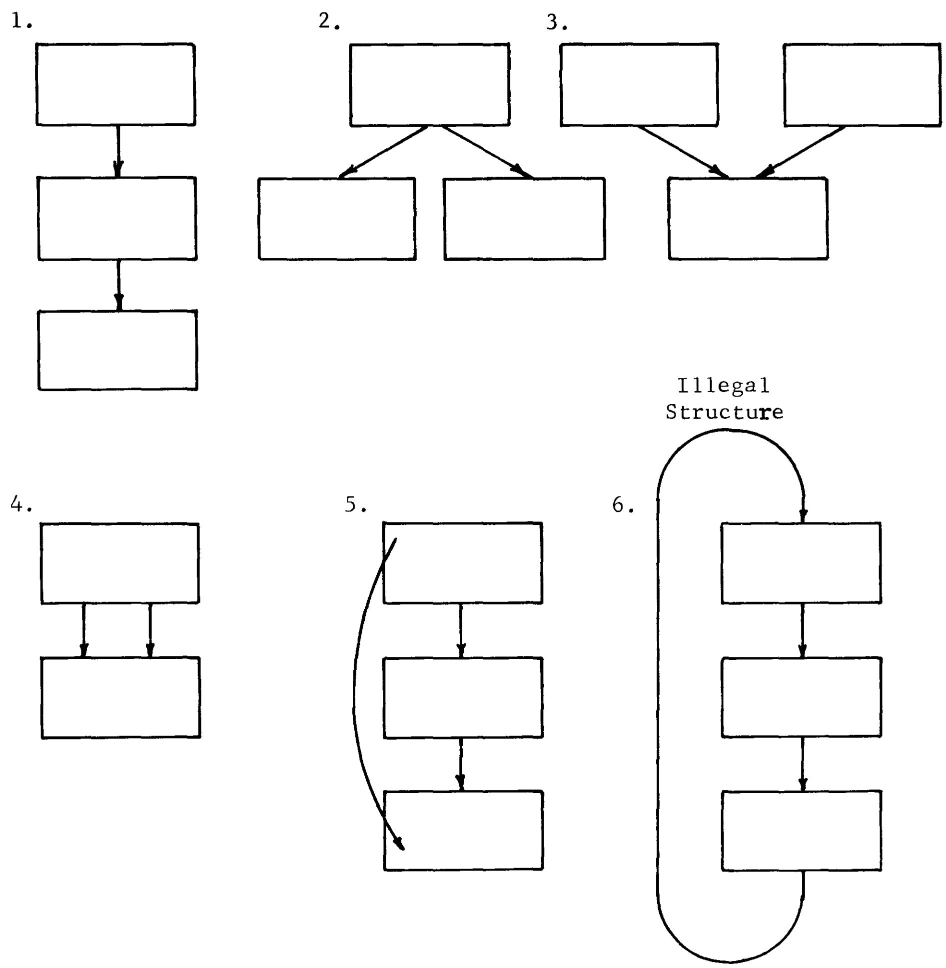


Figure 7. Legal and Illegal I-D-S Data Structures

Record Classes

I-D-S provides three distinct record classes. The designation of the data records as to class is the option of the systems designer and is based on the storage and retrieval requirements of these data records.

I-D-S record processing requires that there be some aspect of every record which makes it unique, or different from any other record. All records are unique by virtue of their reference code. Some records are also unique because they contain one or more data fields--such as a drawing number, order number, and pay number--where no duplicate vlues are allowed.

CALCULATED RECORDS. Any set of records within the system can be classified as a calculated record. Its storage and retrieval are based upon the contents of one or more data fields. The contents of these fields are externally specified values--such as employee numbers, part numbers, or order numbers. The contents of these fields are processed through a randomizing procedure which determines a page number for an initial storage location. The record is stored on this page. If space is not available on the calculated page, the record is stored on the next successive page with available space. The subsequent retrieval of this record follows this same basic procedure.

SECONDARY RECORDS. Secondary records are unique by their chain relationship to a specified type of master record. The item records associated with a purchase order (master) record are good examples of secondary order records. These records are stored and retrieved by first locating the purchase order record and then stepping through the order item chain to locate the item record by comparison of its item number field.

PRIMARY RECORDS. Records designated in the data description as primary are unique only as a result of their reference codes. Generally primary records are used in place of calculated records where the external assignment of identification fields, such as part number or order number, is not required. In these cases, an internally generated number (the reference code) is assigned and used as the key field for storage and retrieval.

I-D-S RECORDS

The I-D-S record contains a set of data fields which collectively describe the contents of the record. I-D-S augments these records with identification and chain fields (or chain pointers) as shown in Figure 8.

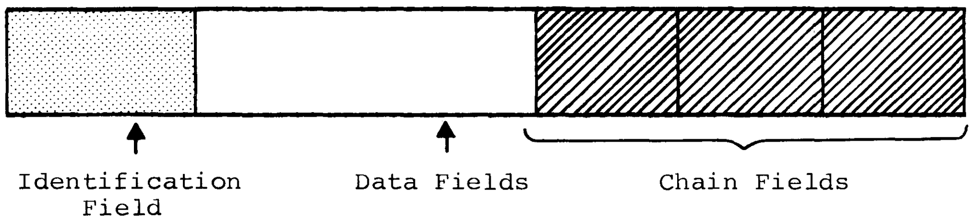


Figure 8. I-D-S Record

There is at least a chain field generated for each chain the record participates in.

The chain fields contain the reference codes of other I-D-S records. They point from one record to the next, creating a circular association of records (see Figure 1).

These associations are automatically processed according to the data descriptions and the procedural commands executed. The arrows in Figure 1 indicate the linking actually carried out through storing the reference code of one record in the body of the prior record.

A reference code is the relative logical (as opposed to physical) address of a data record. It consists of a page number and a line number. The reference code is used by I-D-S to develop and assign a unique address to each data record as it is stored on the mass storage device. Once a record is assigned a reference code, it maintains that reference code until it is physically deleted.

The reference code in its 24-bit binary form is available to the user in a communication area called DIRECT-REFERENCE immediately after the record is stored. For internal use, I-D-S uses a binary number of the form:

XXXXXXYY

where X is the octal page number and Y is the octal line number.

The page number is a sequential number permanently assigned to each page which defines where in the I-D-S environment the page is stored. It occupies three character positions and permits 262,143 pages per I-D-S file. At execution time page numbers are converted to actual mass storage device addresses by the I-D-S mapping routine.

The line number defines where a record is stored within a page. Line numbers are not sequential within the page because new records are always stored at the end of a page. Line numbers of deleted records are made available for use by new records; the first available line number (the first line number not in use) is assigned to a new record as it is stored into a page. The line number occupies one character and permits 63 line numbers per page.

For example, a reference code (as contained in DIRECT-REFERENCE) of 00010029 decimal becomes 00023455 when converted to octal. This then is page 234 (octal), line 55 (octal).

Linking Detail Records of a Chain

In order to insert a new detail record in a chain, three steps are required:

1. Physical storage space must be found.
2. The appropriate master and its chain must be selected.
3. The record must be inserted in that chain according to the chain ordering rules.

Selecting Master Record of a Chain

There are two rules under which the master record is selected for a new detail record. These are:

1. Select Unique Master--This rule uses the record retrieval criterion, established in the data definition for the master record, to retrieve the particular master record indicated by the data values currently stored in the match control field of Working-Storage.
2. Select Current Master--This rule uses the last record processed, of the master record type, as the master record of the new detail.

Chain Ordering

All chains in the Integrated Data Store system are ordered in one of six methods selected by the system designer with the CHAIN-ORDER clause in the I-D-S language.

The CHAIN-ORDER clause must be used in each Master Chain Definition entry.

The six options of the CHAIN-ORDER clause are:

1. Sorted--With this option all of the records of the chain are maintained in a single sequence regardless of the number of record types in the chain. With this option the same sorting-key(s) must be used to sort the various records.

2. Sorted Within Type--With this option the records of the chain are maintained in sequence within record type, independent of other types.

   NOTE: When either of the sorted options is specified, details are added to the chain based upon the contents of the defined sort control fields of the detail records.

3. First--This option causes the detail to be added as the first detail record in the chain relative to the master record.

4. Last--This option causes the detail to be added as the last detail record in the chain relative to the master record.

5. Before--This option causes the insertion of the detail record just before the current record in the chain. This option may be used only in conjunction with the Current Master selection rule.

6. After--This option causes the insertion of the detail record just after the current record of the chain. This option may be used only in conjunction with the Current Master selection rule.

Prime Chain

Access time in present disc-type random access memories varies greatly, since it depends on the position of the desired record relative to the record last accessed. The I-D-S organization of records acknowledges this factor of hardware design and stores new detail records as close as possible to the master record of the chain. When a detail record is specified as a detail in several chains, a prime chain may be chosen and defined by the systems designer preparing the data description. Selection of a prime chain should be based upon an estimate of the most active chain. Thereafter, when an I-D-S page is retrieved which contains the master record of a prime chain, it is highly probable that the detail records of that chain will also be contained in that page or a page closely associated with it. The prime chain is the chain used to retrieve a secondary record by the RETRIEVE command, unless specified otherwise by the data description.

Chain Processing

I-D-S offers complete flexibility in the retrieval of records within a chain by providing three methods of chain inter-linking.

Chain NEXT. The definition of a record as a memory of a chain automatically provides the record with a chain-next field. This is the manner in which all chains are constructed. Each record contains a chain-next field which contains the reference code of the next record in the chain.

Chain PRIOR (optional). I-D-S provides a chain-prior field for all records in a chain when the chain is specified by the system designer as prior processable. This field contains the reference code of the prior record in the chain. This permits the chain to be processed efficiently in a backward direction, as well as forward (through the automatic NEXT chain field).

Chain MASTER (optional). I-D-S provides a chain-master field for all detail records in a chain when specified in the data description. This field contains the reference code of the master record of the chain. Retrieval of the master record is much faster with this ability to address the master record directly from any detail in the chain. Processing need not access all the detail records in the process of seeking the master.

These methods are illustrated in Figure 9.

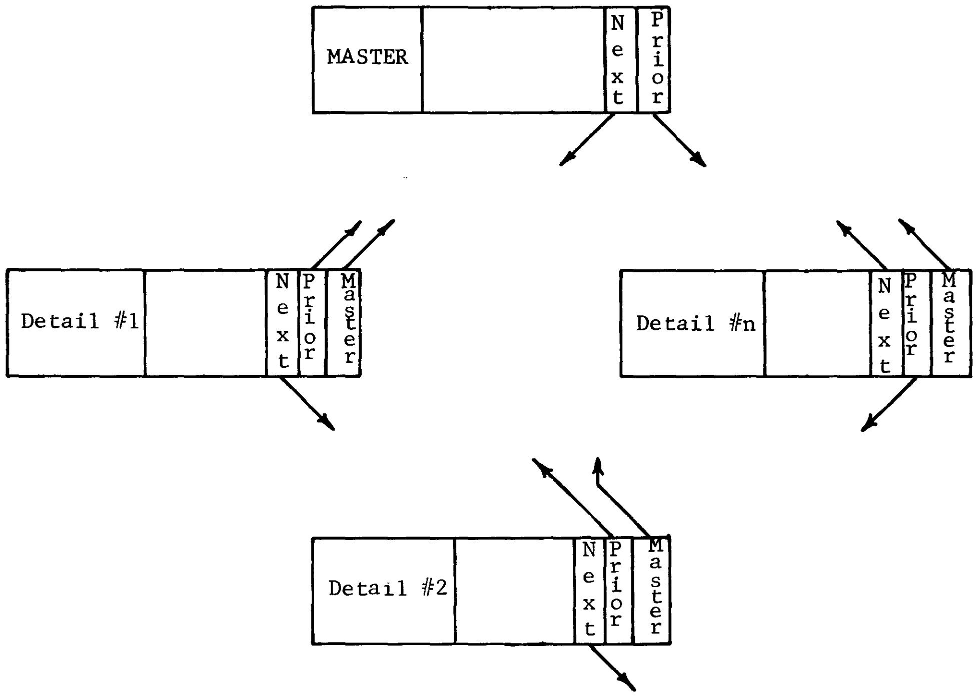


Figure 9. Chain Processing

Chain Tables

Chain tables are used internally by I-D-S subroutines. A chain table is built by I-D-S for each chain defined. The programmer can reference selective information in the chain table, and a knowledge of what they are and how I-D-S uses them can help in designing efficient chains.

A chain table comprises four entries: MASTER, PRIOR, CURRENT and NEXT. Refer to "Chain Processing" for a description of these entries. As I-D-S traverses a chain, the entries are updated with the reference codes of the data records that are being retrieved.

Figure 10 shows a chain using dummy reference codes.

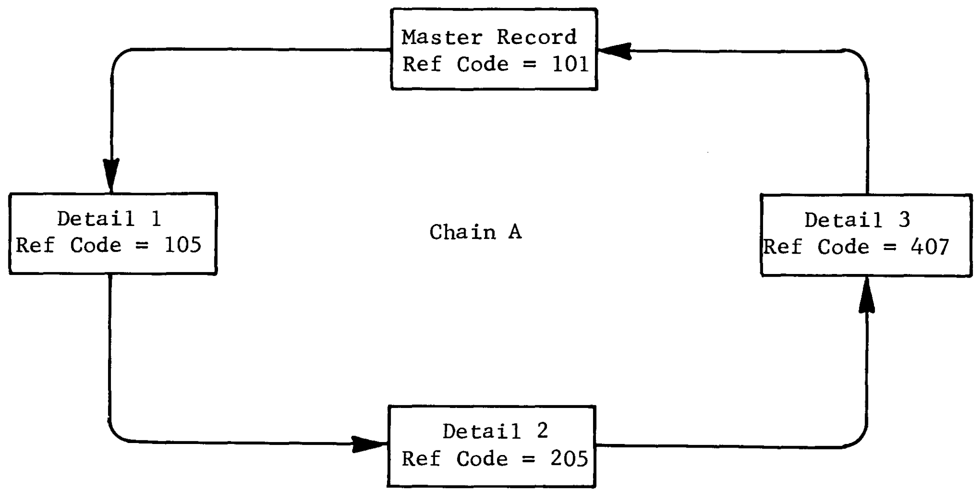


Figure 10. Chain with Dummy Reference Codes

To interpret the dummy reference codes: master record is 101, detail record 1 is 105, detail record 2 is 205, and detail record 3 is 407.

Assume only the reference code of the master record is known. When I-D-S is asked to get detail record 2 of chain A, I-D-S retrieves the master record of chain A and traverses the chain until the detail record 2 is found. While I-D-S is traversing the chain, it is updating the chain table. When detail record 2 is found, the chain table appears as shown in Figure 11.

Master	101
Prior	105
Current	205
Next	407

Figure 11. Chain Table After Retrieval of Detail 2

Although the chain is not PRIOR processable and is not LINKED TO MASTER, detail 1 is directly available with a RETRIEVE PRIOR OF CHAIN A command. Because of the PRIOR entry in the chain table (in Figure 11), I-D-S would not traverse the chain forward through detail 3, master record, etc., to locate detail 1 but would retrieve it directly at the location stored at the PRIOR entry in the chain table. However, after detail 1 is retrieved by "backing up," the record prior to detail 1 is not known. Therefore, the chain table would now appear as shown in Figure 12.

Master	101
Prior	000
Current	105
Next	205

Figure 12. Chain Table Backed Up to Detail 1

If a RETRIEVE PRIOR OF CHAIN A were executed at this point, I-D-S would have to traverse the chain until it found the PRIOR (in this case, the master) record.

If the chain had been defined as PRIOR processable, the chain table would be updated as shown in Figure 13.

Master	101
Prior	101
Current	105
Next	205

Figure 13. Chain Table for a PRIOR Processable Chain after Retrieval of Detail 1

If a chain were not PRIOR processable, I-D-S could back up one record as though it were PRIOR processable if the prior record in the chain had been passed.

Assume that the reference code of detail 2 is known and the chain is neither HEADED nor PRIOR processable. If a RETRIEVE DIRECT is executed, the chain table is updated as shown in Figure 14.

Master	000	(Unknown)
Prior	000	(Unknown)
Current	205	
Next	407	

Figure 14. Chain Table after Direct Retrieval of Detail 2

Since the chain is neither PRIOR processable nor HEADED (LINKED TO MASTER) and I-D-S did not pass the PRIOR record or the MASTER record in getting to the CURRENT record, I-D-S does not know the reference code of the MASTER or the PRIOR record in this chain. It knows where the NEXT record is because of the chain-next field in detail record 2.

If the chain has been defined as PRIOR processable and HEADED and I-D-S had retrieved detail 2 DIRECT, the chain table would appear as shown in Figure 15.

Master	101
Prior	105
Current	205
Next	407

Figure 15. Chain Table--PRIOR and HEADED

In this case, MASTER and PRIOR references were available from the chain fields in detail record 2.

3. I-D-S Programming Language

The source language of I-D-S is an extension of GE-600 Line COBOL; therefore, formats and language specifications of COBOL must be adhered to when preparing a source program.

IDENTIFICATION DIVISION

The purpose and usage of the Identification Division are identical with those defined for GE-600 Line COBOL, with no special function for I-D-S.

Fixed paragraph names are used as keys in the division. They identify the type of information contained in the paragraph. Paragraphs which may be included in the division are:

```
IDENTIFICATION DIVISION.
PROGRAM-ID.
AUTHOR.
DATE-WRITTEN.
DATE-COMPILED.
SECURITY.
REMARKS.
```

ENVIRONMENT DIVISION

All portions of the Environment Division are used as defined by GE-600 Line COBOL, in addition, I-D-S includes the IDS-SPECIAL-NAMES paragraph and the SELECT IDS sentence of the FILE-CONTROL paragraph.

The following illustration is an example of the Environment Division with the use of these two I-D-S functions.

```
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER.  GE-635.
      .
      .
      .
OBJECT-COMPUTER.  GE-635.
      .
      .
      .
IDS-SPECIAL-NAMES.
    IDS BLOCK...
         .
         .
         .
INPUT-OUTPUT SECTION.
FILE-CONTROL.
    SELECT IDS file-name ASSIGN TO file-code-1.
      .
      .
      .
I-O-CONTROL.
    APPLY...
      .
      .
```

Configuration Section, IDS-Special-Names Paragraph

Function: To indicate to the I-D-S translator which statements are to be selectively translated. To allow definition of a unique labeled common area for the generated structure of an I-D-S program. To allow RECORD, CHAIN and FIELD names to be included with the generated structure. To indicate to the COBOL compiler all translator generated sections and code are to be suppressed from the COBOL source listing.

Format Option 1:

IDS-SPECIAL-NAMES.

$$\underline{\text{PROCESS}} \left\{ \begin{array}{l} \underline{\text{ALL}} \\ \text{LEVEL alpha-1} \quad \underline{\text{THRU}} \text{ alpha-2} \end{array} \right\} \quad \underline{\text{DEBUG}} \text{ STATEMENTS}$$

Format Option 2:

IDS-SPECIAL-NAMES.

IDS BLOCK integer-1.

Format Option 3:

IDS-SPECIAL-NAMES.

INCLUDE STRUCTURE NAMES.

Format Option 4:

IDS-SPECIAL-NAMES.

APPLY LIST SUPPRESSION

Notes:

1. This paragraph may be omitted when its provisions are not used in the source-program.

2. The PROCESS DEBUG STATEMENTS option is a compiler directing clause that allows the programmer to specify that all or certain selected debugging statements in the source program are to be processed. Debugging statements can be identified by a single character (A-I) in column 7 of the coding form. When the programmer wants all the debugging statements in the source program processed, he specifies this by writing PROCESS ALL DEBUG STATEMENTS. When the programmer wants certain debugging statements processed, he specifies this by writing PROCESS LEVEL alpha-1 DEBUG STATEMENTS. When this is done, only those debugging statements with the specified character (alpha-1) appearing in column 7 are processed.

   In addition, the programmer can specify that a range of debugging statements are to be processed by writing PROCESS LEVEL alpha-1 THRU alpha-2 DEBUG STATEMENTS. When this is done, all the debugging statements in the range specified (alpha-1 THRU alpha-2) are processed. Note that when debugging statements identified by a single character in column 7 appear in the source program and the PROCESS DEBUG STATEMENTS option is not included in the source program, those statements with a character in column 7 are unconditionally bypassed (i.e., not processed). The PROCESS ALL DEBUG STATEMENTS option has no effect on statements that have nothing, a hyphen, or an asterisk in column 7.

3. In option 2 the value of integer-1 may be 01 through 99.

4. If Option 2 is used, the labeled common area in which the I-D-S generated structure is assembled will be identified by a symbol of the form "I(integer-1)". If Option 2 is not used, the default symbol for this area will be ".IDS..". (Refer to the GE-600 Line General Loader Reference Manual, CPB-1008, for a discussion of labeled common.)

   Example:

   IDS-SPECIAL-NAMES.

   IDS BLOCK 66.

   This would cause the symbol "I66" to be used for the I-D-S generated structure block.

5. Option 3 is used to cause the names of RECORDS, CHAINS and FIELDS to be assembled into the definition structure of the I-D-S-STRUCTURE SECTION.

6. Option 3 will have primary use for programs that use the TRACE and PRINT RECORD, DEBUG, and Utility Subroutine .QSTC, (Chapter 8).

7. Option 4 gives the user the ability to suppress printing of all translator generated coding from the COBOL source listing.

8. The following will be suppressed from the COBOL source listing:

   a. All I-D-S generated structure within the Working-Storage Section.

   b. All generated calls to the I-D-S subroutines within the Procedure Division.

   c. The generated Macro calls within the I-D-S Structure Section.

   d. All generated tables and constants.

   e. All generated Enter Definitions.

9. The statements in IDS-SPECIAL-NAMES paragraph may be in any desired order.

Input-Output Section, File-Control Paragraph

Function: To assign an I-D-S file name and to specify the logical device on which it resides.

Format:

FILE-CONTROL.

SELECT IDS file-name ASSIGN TO file-code-1.

Notes:

1. The SELECT IDS entry must be used only once to identify the I-D-S data file.

2. Other optional clauses of the SELECT entry as specified for COBOL should not be used with the SELECT IDS sentence.

3. File-code-1 must be a two-character word consisting of two letters (A,....,Z) or a letter and a digit (0,....,9). Each file code must be unique with respect to other file codes in the program.

DATA DIVISION

The description of the I-D-S data file is contained in a special section of the Data Division called the IDS Section. This section must physically follow the Working-Storage Section, if present, and precede the Constant Section.

The IDS Section contains a File Description, Record Description, and Chain Definition as required to describe the complete data file.

The following illustration shows the fixed sections of the Data Division in the order in which they must appear in the source program. A section may be omitted if it is not needed.

```
Data Division.
File Section.
Working-Storage Section.
IDS Section.
Constant Section.
Report Section.
```

File Description

The File Description entry provides information regarding the physical characteristics of the I-D-S data file. The entry is used only for documentation purposes and must appear only once in the I-D-S source program and must be the first entry in the IDS Section.

The entry consists of a level indicator, a file name, and a series of clauses which define the physical characteristics of the I-D-S file. The mnemonic level indicator MD is used to identify the start of the File Description entry and to distinguish it from the Record, Field and Chain Descriptions. The format for the complete I-D-S File Description entry follows.

Complete I-D-S File Description Entry

Function: To describe the physical structure of an I-D-S file.

Format:

MD file-name [;PAGE CONTAINS integer-1 CHARACTERS]
[;FILE CONTAINS integer-2 PAGES] .

Notes:

1. The file-name must be identical to the one used in the SELECT IDS sentence of the FILE-CONTROL paragraph of the Environment Division.

   Other optional phrases of the File Description entry as specified for COBOL do not apply to the IDS Section and must not be used.

2. The PAGE size (integer-1) specified may be any value up to a maximum of 4096 characters. However, the most efficient use of the storage capacity of the mass storage device involved should be considered when establishing the page size.

3. The FILE clause expresses the total physical storage requirements of the I-D-S file. This value must be equivalent to or less than the capacity which has been reserved for the file by the allocation procedure of GECOS. See the GE-600 Line Comprehensive Operating Supervisor (GECOS* III) Reference Manual, CPB-1518, for a discussion of the allocation of permanent random disc or drum files. The maximum number of pages possible within the I-D-S page numbering system is 262,143.

4. Page and file size is for documentation only; it is not used during execution.

*GECOS, Trademark

5. Page and file size clauses are not required.

6. The clauses may appear in any order within the entry. The entry must end with a period.

7. Example:

```
IDS SECTION.
MD DATA-BASE: PAGE CONTAINS 1920 CHARACTERS;
        FILE CONTAINS 100000 PAGES.
01 UNIT-MASTER-REC;
        TYPE IS 070;
        RETRIEVAL VIA MASTR FIELD;
   02 MASTR;SIZE 8 NUMERIC.
   98 UNIT CHAIN MASTER;
        CHAIN-ORDER IS SORTED.
```

Record Description

Record Description entries are used to:

1. Provide information to I-D-S regarding the external format of each logical record type as it will exist within a page on the external storage device.
2. Define internal Working-Storage areas which serve as communication interfaces between the user's routine and the I-D-S data file.
3. Provide parameters which control I-D-S processing. These parameters are defined at levels 01 and 98.

The external format of an I-D-S record consists of control fields and data fields. Records are stored as fixed-length records. Each record contains identification fields, a chain field for each chain association specified, and as many characters of data as required by the level 02 entries.

The level 02 entries are packed into the records, and records are packed into pages on a character-oriented basis. Computer word orientation is never used. When a record is retrieved from the storage device, the data fields of the record are available to the user only after they are moved to working storage. Before storing a record, the Working-Storage area must first have been initialized with the data fields of the record to be stored.

The I-D-S Translator creates an internal Working-Storage area for each level 02 entry. The area created may contain subfields which are defined by lower level entries and may be separately referenced by user COBOL procedure statements. However, I-D-S operates only on units of data defined by the level 02 entry. Therefore, any field that is to serve either as a control field or that is to be modified by I-D-S must be defined as a level 02 entry.

The Translator produces parameters from the clauses that are defined at levels 01 and 98. Lower level entries (03-49) may be used to define subfields of the level 02 entry. Any legal COBOL description clause may be used as long as it does not contradict the description provided for the level 02 entry. For a further clarification of the GE-600 Line COBOL Reference Manual, CPB-1652.

The parameters are described in detail on the following pages.

The level 02 data names may not be used for qualification. Qualification of lower level entries up to level 02 is permissible. If the same data name occurs as an 02 entry for different record types, the same Working-Storage area will be shared by the various records involved.

Standard COBOL record description clauses allowed at level 02 are REDEFINES and FILLER. They do not generate Working-Storage areas.

REDEFINES may be used for redefinition of an area previously defined. This enables COBOL procedural statements to reference the Working-Storage area by either of its definitions. The field-oriented functions of I-D-S (MOVE, MODIFY), however, respond only to the original definition of the field.

The use of FILLER as a data-name creates space in the external format only.

Although the PICTURE clause is the significant element of the level 02 description, any of the standard COBOL clauses may be used with the following exceptions:

OCCURS
RENAMES
Editing clauses
COPY

Complete I-D-S Record Description Entry

Function: To specify the parameters which define an I-D-S record.

Format:

```
01 record-name; TYPE IS integer-1

 ;RETRIEVAL VIA     { field-name           FIELD }
                    { {chain-name-1}       CHAIN }
                    { {CALC        }             }

[ ;PAGE-RANGE IS { integer-2 TO integer-3         } ]
[                { field-name-1 TO field-name-2   } ]

[;PLACE NEAR chain-name-2 CHAIN]

[;INTERVAL IS  integer-4 PAGES]

[;AUTHORITY IS integer-5]
```

Notes:

1. Each of the above clauses is applicable only at record level 01.
2. Record-name must be unique, since qualification by file name is not meaningful.
3. The clauses may appear in any order within the entry. The entry must end with a period.
4. All format considerations are as specified for COBOL.

TYPE

Type

Function: To define the Record Type code to be used for reference purposes for each record type within I-D-S.

Format: TYPE IS integer-1

Notes:

1. This clause is required for each level 01 entry.

2. Integer-1 may be any value from 1 to 999.

Retrieval Via

Function: To specify procedures for retrieving and storing a record.

Format:

```
;RETRIEVAL VIA   { field-name      FIELD }
                 { {chain-name-1}  CHAIN }
                 { {CALC        }        }
```

Notes:

1. This clause is required for each level 01 entry.

2. Records specified for RETRIEVAL VIA field-name FIELD are referred to as primary records.

   Field-name must be defined at level 02 in this record. It must be specified as:

   02 field-name PICTURE 9(8).

   The field is not stored in the record; it exists only in working storage. The field-name FIELD is called the prime retrieval field.

   If the user wishes to retrieve a primary record using the RETRIEVE record-name RECORD statement of the Procedure Division, he must first initialize the field-name with the reference code of the record to be retrieved.

   When a primary record is stored, its reference code is placed into DIRECT-REFERENCE. The user may specify the page where he wishes a primary record stored by placing its reference code in the DIRECT-REFERENCE. Zeros may also be stored in DIRECT-REFERENCE which causes the record to be stored on a page most convenient to I-D-S.

   Placement of primary records can be modified by the PAGE-RANGE, PLACE NEAR, and INTERVAL clauses.

3. Records specified for RETRIEVAL VIA chain-name-1 CHAIN are referred to as secondary records and are retrieved by their association in the named chain. The chain-name-1 CHAIN is the prime retrieval chain for the record.

   When the RETRIEVAL VIA chain-name-1 CHAIN clause is used, the record must be specified at level 98 as chain-name-1 CHAIN DETAIL.

   When the RETRIEVE record-name RECORD statement of the Procedure Division is used, the master record of the chain-name-1 CHAIN is first retrieved. Then the specific detail record is found by searching the chain.

   If the CHAIN-ORDER is FIRST or LAST, then the RETRIEVAL VIA chain-name CHAIN clause causes the record to be stored on the page of the master record of the chain named in the clause. Otherwise, the record is stored in the page of the current record of the chain named. When a secondary record is stored, I-D-S places its binary reference code into DIRECT-REFERENCE. Placement of secondary records can be modified by PAGE-RANGE, PLACE-NEAR, and INTERVAL clauses.

4. Records specified for RETRIEVAL VIA CALC CHAIN are referred to as calculated records.

   RETRIEVAL VIA CALC CHAIN operates the same as RETRIEVAL VIA chain-name-1 CHAIN, except that the master record of the chain is a Page Header record. The CALC CHAIN is called the prime retrieval chain for the record.

   When the RETRIEVAL VIA CALC CHAIN clause is used, the record must be specified at level 98 as a CALC CHAIN DETAIL.

   When the RETRIEVE record-name RECORD statement of the Procedure Division is used, the Page Header record is first retrieved. Then the specific detail record is found by searching the CALC chain. The Page Header record is found by randomizing the values in the control fields defined in the detail record to be retrieved. The number resulting from the randomization is mapped into the effective page range of the detail record to be retrieved, thereby yielding the page number of the Page Header record whose CALC chain is to be searched.

The RETRIEVAL VIA CALC CHAIN clause causes the record to be stored on the page calculated by randomizing on control fields and mapping into the effective page range of the record.

Placement of calculated records may be modified by the PAGE-RANGE clause. PLACE NEAR and INTERVAL clauses do not apply to calculated records.

5. These three RETRIEVAL procedures provide a basis for classification of each record as one of the following:

   Primary - Retrieved directly via reference code
   Secondary - Retrieved via its chain association
   Calculated - Randomized to the page containing the chain which leads to the record.

Subsequent discussions of I-D-S will refer to records using these terms.

PAGE-RANGE

Page-Range

Function: To provide a method for placing various record types within a designated segment of an I-D-S file.

Format:

$$\left[\ ;\underline{\text{PAGE-RANGE}}\ \text{IS}\ \left\{\begin{array}{l}\text{integer-2 TO integer-3}\\ \text{field-name-1 TO field-name-2}\end{array}\right\}\right]$$

Notes:

1. Integer-2 and integer-3 represent the first and last page numbers of a series of pages in which records of a particular type are stored. If integer-2 is greater than integer-3, the series of pages wraps around the end of the file and terminates at a lower page number.

   For example, if a 900-page file contained record types A, B, and C, each record type could be isolated in a segment of the file by specifying a page range of 1 to 300 for A, 301 to 600 for B, and 601 to 900 for C.

2. The page numbers must fall within the total number of pages specified for the file.

3. Different types of records may share the same page range.

4. The PAGE-RANGE clause delimits the action of the RETRIEVAL VIA, PLACE NEAR, and INTERVAL clauses.

5. The PAGE-RANGE clause may be used for calculated records.

6. If PAGE-RANGE is not specified, the range is assumed to be equal to the page range of the entire file.

7. If the field name option is used, field-name-1 and field-name-2 must be defined in Working-Storage.

   Example:

   ```
   77 field-name-1  PIC 9(6) COMP-1.
   77 field-name-2  PIC 9(6) COMP-1.
   ```

8. The page range values must be placed in field-name-1 and field-name-2 prior to STORE of the record or prior to RETRIEVE of the record.

9. Example:

```
IDS SECTION.
MD DATA-BASE; PAGE CONTAINS 1920 CHARACTERS;
        FILE CONTAINS 100000 PAGES.
01 UNIT-MASTER-REC;
        TYPE IS 070;
        RETRIEVAL VIA MASTR FIELD;
   02 MASTR PICTURE 9(8).
   98 UNIT CHAIN MASTER;
        CHAIN-ORDER IS SORTED.

01 UNIT-REC;
        TYPE IS 010;
        RETRIEVAL VIA CALC CHAIN;
        PAGE RANGE IS 1 TO 20000.
         .
         .
         .
01 QUAD4
        TYPE IS 004
        RETRIEVAL VIA CALC CHAIN
        PAGE-RANGE IS RNG-1 TO RNG-2.
```

PLACE
NEAR

Place Near

Function: To store a record physically near the master record of a specified chain.

Format:

[;PLACE NEAR chain-name-2 CHAIN]

Notes:

1. Chain-name-2 must be a defined chain name. The record to be placed must be specified at level 98 as a detail of the chain named in the PLACE NEAR clause.

2. The PLACE NEAR clause may only be used with primary and secondary records.

3. If the CHAIN-ORDER is SORTED, SORTED WITHIN TYPE, FIRST or LAST, the record is stored on the page of the master record of the chain named in the PLACE-NEAR clause. Otherwise, the record is stored in the page of the current record of the chain.

4. If a current record of the type named exists, the INTERVAL clause supersedes this clause.

5. The PAGE-RANGE clause supersedes this clause when a conflict occurs.

6. Records stored using this clause are subject to the overflow rule.

Interval

Function: To enable uniform distribution of records of a given type across the I-D-S file.

Format:

[;<u>INTERVAL</u> IS integer-4 PAGES]

Notes:

1. Integer-4 represents the number of pages which will be skipped when a record is stored.

2. The INTERVAL clause may only be used with primary and secondary records.

3. Normally, primary records are stored physically according to a reference code furnished by the user. Secondary records are stored physically near the master record of the chain specified in the RETRIEVAL VIA chain-name-1 CHAIN clause or according to a PLACE NEAR clause. When INTERVAL is used, the above criteria apply only to the first record of the stored type. That is, if I-D-S has not processed a record of this type, the CURRENT record value is zero and INTERVAL is not in effect. Subsequent records are stored integer-4 pages away from the current record of the specified type. The current record is either the last record of the type stored or the last record of the type retrieved.

   For example, if the last record of type A is stored on page 5 and interval is 3, the next record of type A would be stored on page 8.

4. The INTERVAL clause is used normally for initial file loading of primary master records. By specifying an interval, the user can ensure sufficient space between the master records to store the detail records in their chains.

5. When INTERVAL reaches the end of the page-range or end of the file, it reverts either to the beginning of the page-range or to the beginning of the file.

6. Records stored using this clause are subject to the overflow rule.

7. Application of INTERVAL by I-D-S is not continuous between computer runs. If it is to continue from day to day, it must be reinitialized by retrieving the last record of the type processed in the previous run which makes it current in this run. Storage may continue from this point.

Authority

Function: To safeguard data in a record against unauthorized reference or modification.

Format:

[;AUTHORITY IS integer-5]

NOTE: Integer-5 may be any value not exceeding 4095(10). The value supplied is used as a lock for data in any record of this type. When this record is referred to during execution, a key must have been supplied that matches the lock. The key is supplied by the OPEN statement which is defined in the Procedure Division.

Chain Definition

A record belongs to at least one, and possibly many chains. A Chain Definition entry must exist for each chain in which the record is included. All Chain Definition entries for a given record must immediately follow the Record Description entries for that record.

The Chain Definition entry consists of a level 98 indicator which names the chain that a level 01 record is either a detail or master in, a chain name, and a series of clauses which define the characteristics of the chain. The complete Chain Definition entry skeleton and a detailed description of the clauses follow.

COMPLETE CHAIN
DEFINITION ENTRY

Complete Chain Definition Entry

Function: To name and describe the interrecord relationship between a master and detail record and to direct the placement of a record into the I-D-S file.

Format Option 1 (Master):

```
98 chain-name-1 CHAIN MASTER

                         ⎧SORTED WITHIN TYPE⎫
                         ⎪SORTED            ⎪
;CHAIN-ORDER IS          ⎨FIRST             ⎬
                         ⎪LAST              ⎪
                         ⎪BEFORE            ⎪
                         ⎩AFTER             ⎭

  [;LINKED TO PRIOR]
```

Format Option 2 (Detail):

```
98  {chain-name-2}      CHAIN DETAIL
    {CALC        }

   [;RANDOMIZE ON field-name-1   [;RANDOMIZE...]]

   [                    {ARE FIRST   }]
   [;DUPLICATES         {ARE LAST    }]
   [                    {NOT ALLOWED }]

   [  {ASCENDING }                      [   {ASCENDING... }]]
   [; {          }KEY IS field-name-2   [;  {             }]]
   [  {DESCENDING}                      [   {DESCENDING...}]]

   [;ASCENDING RANGE KEY IS field-name-3]

   [;SELECT         {UNIQUE }       MASTER]
   [                {CURRENT}             ]

[;MATCH-KEY IS field-name-4   [MATCH-KEY...]]

[;MATCH-KEY IS field-name-5 {SYNONYM} field-name-4   [MATCH-KEY...]]
[                           {SYN    }                              ]

[;LINKED to MASTER]
```

Master/Detail

Function: To describe a record as either a detail or master of a chain.

Format Option 1:

98 chain-name-1 CHAIN MASTER

Format Option 2:

98 $\left\{\begin{array}{l}\text{chain-name-2}\\ \underline{\text{CALC}}\end{array}\right\}$ CHAIN DETAIL

Notes:

1. This entry must be a level 98.

2. Option 1 defines a record as the master record of a chain structure. One option 1 entry is required for each chain structure for which the record is the master. A single chain structure can have only one master record but a single record can be the master of more than one chain structure.

   In the example below UNIT-REC is a master record in the SUB-UNIT CHAIN, the ASSIGNMENT CHAIN and the COMPLEMENT CHAIN.

3. Option 2 defines a record as a detail record in a chain structure. One option 2 entry is required for each chain structure in which the record is a detail. A record may be a detail in more than one chain structure. A single chain structure can be made up of any number of detail record types.

   In the example below UNIT-REC is a detail record in the CALC CHAIN and the UNIT CHAIN.

4. The record may be a master in one chain structure and a detail in another. In this case, both options are required for that record. A record may not be defined as both MASTER and DETAIL in the same chain.

   In the example below, UNIT-REC is a master record in the SUB-UNIT CHAIN, the ASSIGNMENT CHAIN and the COMPLEMENT CHAIN. It is also a detail record in the CALC CHAIN and UNIT CHAIN.

5. If the RETRIEVAL VIA chain-name-1 CHAIN or RETRIEVAL VIA CALC CHAIN clause is used in the level 01 Record Description entry, an option 2 entry must name the appropriate chain structure.

The following statements illustrate this rule:

```
01 SUB-UNIT REC;
        TYPE IS 030;
        RETRIEVAL VIA SUB-UNIT CHAIN;
        PAGE-RANGE IS 1 TO 20000.
    02 SUB-UNIT-CODE; SIZE 4 NUMERIC.
    98 SUB-UNIT CHAIN DETAIL;
                .
                .
                .
```

6. Example:

```
01 UNIT-REC;
        TYPE IS 010;
        RETRIEVAL VIA CALC CHAIN;
        PAGE RANGE IS 1 TO 20000.
    02 UNIT-CODE: SIZE 4 NUMERIC.
        03 DIVISION-CODE: SIZE 1 NUMERIC.
        03 DEPARTMENT-CODE: SIZE 1 NUMERIC.
        03 GROUP-CODE: SIZE 1 NUMERIC.
        03 SECTION-CODE: SIZE 1 NUMERIC.
    02 REPORTING-UNIT: SIZE 4 NUMERIC.
    02 ORG-NAME: SIZE 20 ALPHANUMERIC.
    02 TOTAL-BUDGET: SIZE 7 NUMERIC.
    98 CALC CHAIN DETAIL;
        RANDOMIZE UNIT-CODE.
    98 SUB-UNIT CHAIN MASTER;
        CHAIN-ORDER IS SORTED.
    98 ASSIGNMENT CHAIN MASTER;
        CHAIN-ORDER IS FIRST.
    98 COMPLEMENT CHAIN MASTER;
        CHAIN-ORDER IS SORTED.
    98 UNIT CHAIN DETAIL;
        SELECT CURRENT MASTER;
        ASCENDING KEY IS UNIT-CODE;
        DUPLICATES NOT ALLOWED.
```

Chain-Order

Function: To specify the criteria for sequencing detail records within a chain.

Format:

;CHAIN-ORDER IS { SORTED WITHIN TYPE | SORTED | FIRST | LAST | BEFORE | AFTER }

Notes:

1. This clause must be used in each Master Chain Definition entry (option 1).

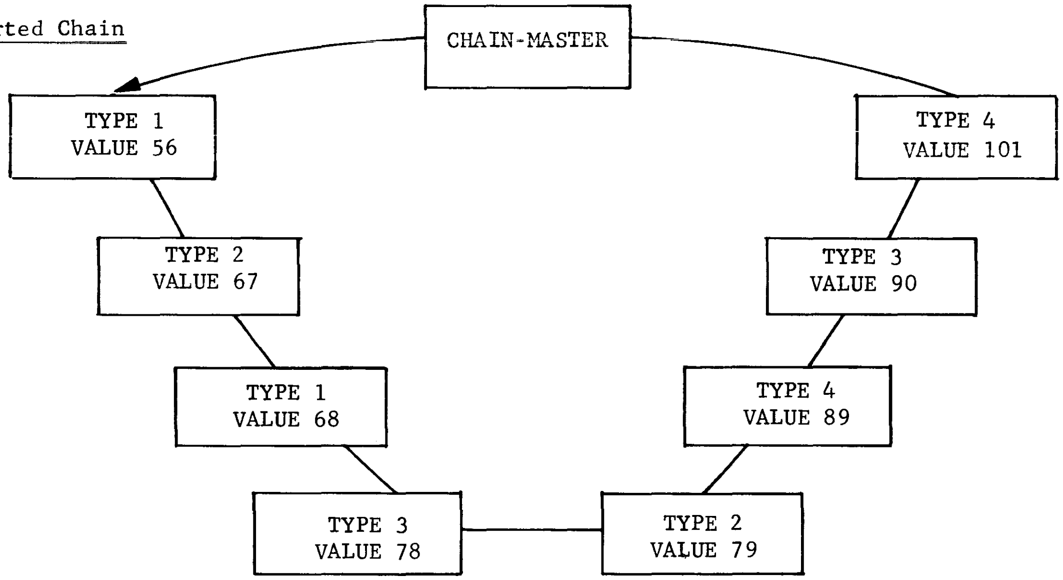
2. If either SORTED or SORTED WITHIN TYPE is used, detail records are positioned in the chain according to the value of their sort control fields.

   If SORTED is used, the various records of the chain are maintained in a single sequence regardless of the number of record types in the chain. The size and class of sort control fields of the various records must be identical.

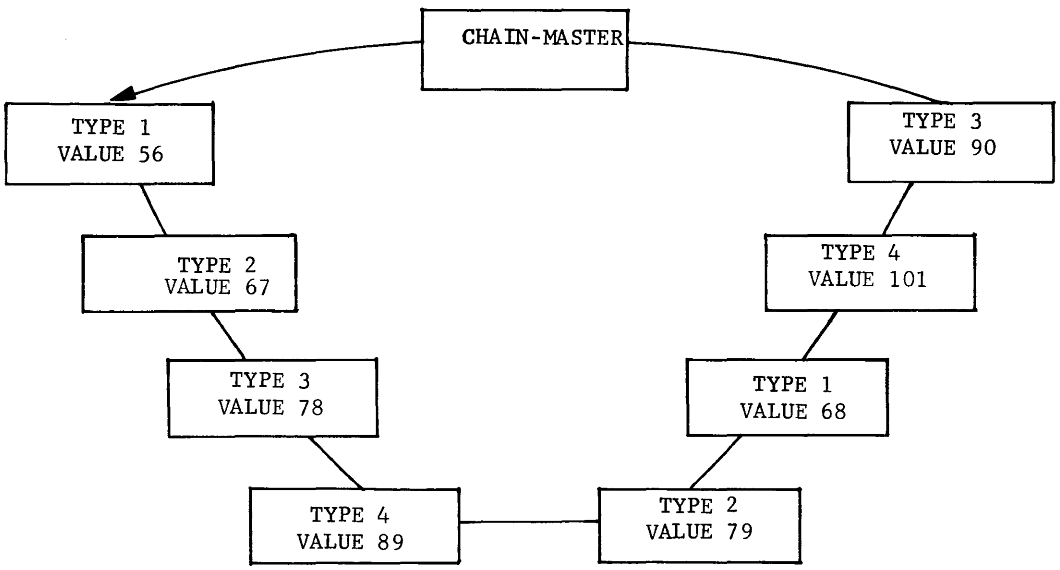
   If SORTED WITHIN TYPE is used, records of the chain are maintained in sequence within a record type, independent of other types. This does not mean that there is an implied major sort by record type code. It means only that when a given type of record is considered, it is in sequence by its own sort key.

   An example of a SORTED and SORTED WITHIN TYPE chain follows.

Sorted Chain



Sorted Within Type Chain



3. The last four forms, FIRST, LAST, BEFORE, and AFTER, of this clause cause a detail record to be inserted in the chain relative to some other record in the chain. These options are:

   <u>FIRST</u> Insert detail record in chain immediately following the master record.

   <u>LAST</u> Insert detail record in chain immediately preceding the master record.

   <u>BEFORE</u> Insert detail record in chain immediately preceding the current record of chain.

   <u>AFTER</u> Insert detail record in chain immediately following the current record of chain.

   The current record of a chain will always be the master record if SELECT UNIQUE MASTER has been specified.

   The selection of the BEFORE and LAST Options causes I-D-S to create an extra chain field which contains the reference code of the immediately preceding record of the chain.

   BEFORE causes the creation of this field in all record types of the chain. LAST introduces this field in the master record type only.

   The BEFORE and AFTER forms are compatible only with the SELECT CURRENT MASTER clause.

   If the chain has been defined as LINKED TO PRIOR and the CHAIN-ORDER IS BEFORE clause is used, the records in the chain are assigned only one chain field PRIOR; there is no duplication of chain fields.

4. When a record is defined as a detail of a calculated chain, no order is maintained because calculated chains have no defined sequence control.

Linked Prior

Function: To provide an additional chain field in each record of a chain which contains the reference code of the immediately preceding record in the chain. This field allows a chain to be traversed in either direction.

Format:

[;LINKED TO PRIOR]

Notes:

1. This clause is used only in the Master Chain Definition entry (option 1). It provides a prior chain field in each record of the chain so that the chain may be traversed in either direction. This feature is especially serviceable when using either the RETRIEVE PRIOR or MODIFY verbs. It also enables the immediate removal of a deleted record which would otherwise stay linked in this chain until the chain was traversed again.

2. Chain PRIOR fields have two disadvantages. First, the record size is increased to provide space for the additional field. Second, the linking process is slower because the chain PRIOR field of the next record must be adjusted when a new record is inserted.

3. When the CHAIN-ORDER IS BEFORE clause is specified, I-D-S automatically provides a chain PRIOR field for all record types defined for that chain.

   When the CHAIN-ORDER IS LAST is specified, I-D-S automatically provides a chain PRIOR field for the master record only.

Randomize

Function: To specify those fields of a calculated record used to generate the page number for record placement and retrieval.

Format:

[;RANDOMIZE ON field-name-1 [;RANDOMIZE...]]

Notes:

1. RANDOMIZE must be used for each calculated record.
2. Field-name-1 must be a level 02 field contained in the record being stored or retrieved.
3. The randomizing routine of I-D-S uses as many fields as are specified.
4. The word RANDOMIZE must precede each control field specified.
5. The fields designated as RANDOMIZE fields are compared at record storage time. An attempt to store a record with identical RANDOMIZE field values will be rejected as an error.
6. This clause may only be used when RETRIEVAL VIA CALC CHAIN is specified at level 01.

Duplicates

Function: To specify whether records with identical sort key values may exist in a chain and, if permitted, what ordering action should be taken.

Format:

```
[                 {ARE FIRST  }]
[ ;DUPLICATES     {ARE LAST   }]
[                 {NOT ALLOWED}]
```

Notes:

1. This clause must be used and only used when the chain has been defined as a sorted chain by the CHAIN-ORDER clause.

2. When duplicates are allowed, the new detail may be positioned as the FIRST or LAST of the string of records with identical sort key values.

3. If duplicates are not allowed and an attempt is made to link records with identical sort key values (STORE or MODIFY), an error code is placed in the ERROR-REFERENCE communication area and the duplicate record is rejected.

   It is the user's responsibility to examine this communication area.

4. Duplicates are not allowed in a CALC chain; however, it is not necessary to write the DUPLICATES NOT ALLOWED clause. Since CALC chains have no sequence, I-D-S ensures that there are no duplicates by searching the entire CALC chain before attempting to store a new CALC record.

Ascending/Descending

Function: To specify those data fields which control the sequence of detail records in a chain.

Format Option 1:

[; { ASCENDING | DESCENDING } KEY IS field-name-2 [; { ASCENDING... | DESCENDING }]]

Format Option 2:

[; ASCENDING RANGE KEY IS field-name-3]

Notes:

1. This clause must be used when a chain has been defined as a SORTED or SORTED WITHIN TYPE chain.

   For example:
```
01 UNIT-MASTER-REC;
        TYPE IS 070;
        RETRIEVAL VIA MASTR FIELD.
    02 MASTR; SIZE 8 NUMERIC.
    98 UNIT CHAIN MASTER;
        CHAIN-ORDER IS SORTED.

01 UNIT-REC;
        TYPE IS 010;
        RETRIEVAL VIA CALC CHAIN;
        PAGE RANGE IS 1 TO 20000.
    02 UNIT-CODE; SIZE 4 NUMERIC.
    02 REPORTING-UNIT; SIZE 4 NUMERIC.
    02 ORG-NAME; SIZE 20 ALPHANUMERIC.
    02 TOTAL-BUDGET; SIZE 7 NUMERIC.
    98 UNIT CHAIN DETAIL;
        ASCENDING KEY IS UNIT-CODE;
        DUPLICATES NOT ALLOWED;
        SELECT UNIQUE MASTER;
        MATCH-KEY IS MASTR.
```

2. Field-name-2 must be a level 02 field entry within the record being defined. In the above example, UNIT-CODE meets this requirement. However, field-name-2 may not be a level 02 field entry which has been specified at level 01 as a RETRIEVAL VIA field-name FIELD. In the above example MASTR cannot be a KEY.

3. When multiple sort control keys are required to define a chain sequence, the various field-names must be presented in sequence from major control field to minor, thus establishing the sort level of each field. Each sort control key must be independently defined as either ASCENDING or DESCENDING.

When ASCENDING is used, the sorted sequence will be from lowest value of key to highest value.

When DESCENDING is used, the sorted sequence will be from highest value of key to lowest value.

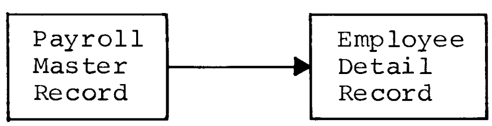
4. Option 2, ASCENDING RANGE KEY is used when the record is to serve as a range master. A range master is a detail record in a sorted chain. In addition, it is the master of a chain which includes detail records falling within the range of the range master. The value contained in field-name-3 controls the ascending sequence of the range masters. It also defines the upper range limit of details referenced by the range master.

Range masters are used primarily to segment long sorted chains. The purpose is to reduce access time in reaching the detail records.

The ASCENDING RANGE KEY clause modifies the search method of the RETRIEVE record-name RECORD and STORE record-name RECORD statements by searching the chain until the sort key value of the retrieved record is equal to <u>or greater</u> than the working-storage value of the record to be retrieved or stored.

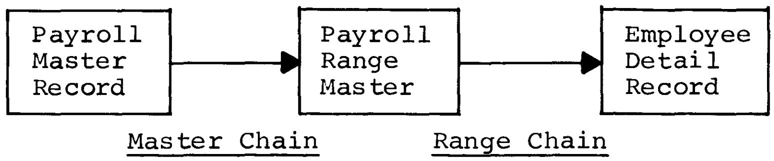
If the RANGE option is not specified, the chain is searched until the sort key value of the retrieved record is equal to the working-storage value of the record to be retrieved.

A payroll master chain structure of employee detail records is illustrated below:



<u>Master Chain</u>

By introducing range masters into the structure, the one long chain could be divided into several smaller ones. The structure would look like this:



The steps used to create this structure include:

1. Define Payroll Master record.

    a. Designate it master record of Payroll-Master chain.

    b. Designate CHAIN-ORDER as SORTED or SORTED WITHIN TYPE.

2. Define Payroll Range Master record.

    a. Name within it a field RANGE-NO.

    b. Designate it as a detail record in the Payroll-Master chain.

    c. Name RANGE-NO field as an ASCENDING RANGE KEY.

    d. Designate it master record of Payroll-Range chain.

3. Define Employee Detail record.

    a. Name within it a field EMPL-NO.

    b. Designate it as a detail record in Payroll-Range chain.

    c. Name RANGE-NO as MATCH-KEY for the Payroll-Range chain.

    d. Name EMPL-NO as a sort key or match-key for this record.

At execute time the user would identify a range master by placing an employee number into RANGE-NO in working-storage. I-D-S selects the first range master in sequence whose value in RANGE-NO is equal to or greater than the value placed in RANGE-NO in working-storage. Once the range master is found, the detail record can be stored or retrieved along its chain by using EMPL-NO as control.

SELECT

Select

Function: To specify the rule for selecting a specific master record from all master records of a given type when a detail record is being stored or retrieved by the RETRIEVE record-name RECORD statement or STORE record-name RECORD statement.

Format:

```
[;SELECT {UNIQUE } MASTER]
         {CURRENT}
```

Notes:

1. One of the two forms of the SELECT clause must be used in each Chain Description entry which specifies a level 98 chain detail. The SELECT clause does not apply to a CALC CHAIN DETAIL because the Page Header record (specified by the output of the randomizing procedure) is the unique master to be selected.

2. When UNIQUE is specified, the master is selected by matching the data field values in a master record with those initialized by the user in working storage. The fields to be initialized are those specified as MATCH-KEY fields in the level 98 entry.

3. When CURRENT is specified, the master of a chain relevant to current detail record of the named chain is selected. If the current record of the named chain is already the master, then it is selected. The responsibility for establishing the current master of the chain-name is left to the user.

Match-Key

Function: To specify those fields which must be initialized by the user in working-storage to allow unique identification of the master record of a chain.

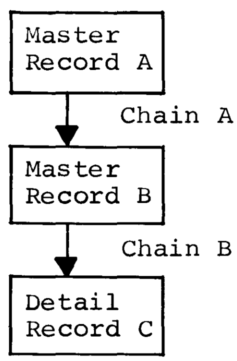
Format:

[MATCH-KEY is field-name-4 [;MATCH-KEY...]]

Notes:

1. This clause applies only to option 2 of the Chain Definition Entry. It must be used in conjunction with the SELECT UNIQUE MASTER clause.

2. Only those fields necessary to uniquely select the appropriate master need be specified. If the master is a detail record in a higher level chain structure, match-key fields for selection of its master are named with it, but need not be named with this record. For example:



When Master Record B is defined as a detail in Chain A, match-key fields are named for Master Record A. When Detail Record C is defined as a detail in Chain B, match-key fields are named for Master Record B, not for Master Record A.

3. The fields named in MATCH-KEY clauses depend upon the RETRIEVAL clauses specified for each of the higher-level master records defining the hierarchical structure which includes this record as a detail.

   The following rules should be used in naming the appropriate master record fields with MATCH-KEY clauses in this record.

   If the master record is defined as a primary record by the RETRIEVAL VIA field-name FIELD clause, the field-name must be named as a MATCH-KEY field-name for the detail record.

   If the master record is defined as a secondary record by the RETRIEVAL VIA chain-name CHAIN clause, each of the data fields which control the retrieval of the master record must be named as MATCH-KEY field names in this detail record. Thus, it is necessary that the master record be either in a sorted chain (sort keys) or a calculated chain (randomize keys).

   If the master record is defined as a calculated record by the RETRIEVAL VIA CALC CHAIN clause, the RANDOMIZE fields for that master must be named as MATCH-KEY fields.

4. All applicable MATCH-KEY fields must be initialized in working storage with the desired values before storing the record or before retrieving it using the RETRIEVE record-name RECORD verb. This includes the match-key fields for all higher level master records involved in the chaining structure even though the fields were not named with this record.

Synonym

Function: To specify an alternate name for a field defined as a MATCH-KEY field.

Format:

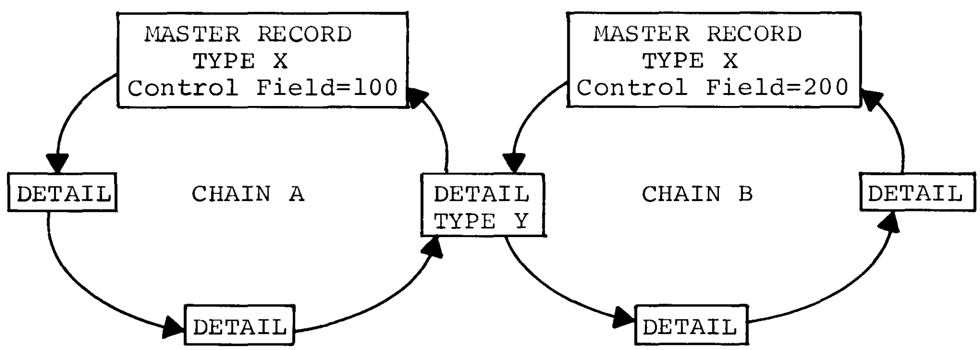
$$\left[;\underline{\text{MATCH-KEY}}\ \text{IS}\ \left[\text{field-name-5}\ \left\{\begin{array}{l}\underline{\text{SYNONYM}}\\ \underline{\text{SYN}}\end{array}\right\}\right]\ \text{field-name-4}\right]$$

Notes:

1. The use of the SYNONYM option within the MATCH-KEY clause defines an alternate name (field-name-5) for the MATCH-KEY field (field-name-4).

2. The alternate name (field-name-5) <u>must</u> have been previously defined in the Working-Storage Section in exactly the same format as the MATCH-KEY field for which it is an alternate.

3. Example:



Detail record type Y is defined in chain structures A and B. Chains A and B have the same record type (X) as their master records. Therefore, each of the two different master records of type X must be uniquely identified when the type Y detail record is stored.

I-D-S stores the detail record into Chain A with one store operation. The master record control field is named with a MATCH-KEY clause when detail Y is defined in both chains. In addition, for Chain B, an alternate working-storage area is named using the SYNONYM clause. Before storing the record, the user must initialize field-name-4 for the master record control field to 100 and the SYNONYM field-name-5 with 200.

LINKED-MASTER

Linked-Master

Function: To provide an extra chain field for each detail record of the chain which points to the master record of the chain.

Format:

[;LINKED TO MASTER]

Note:

This optional clause can improve the operation of the system by providing a direct path from each detail to the master of the chain, thus eliminating the need for processing all of the intervening detail records serially.

PROCEDURE DIVISION

Execution of I-D-S procedural statements will STORE, RETRIEVE, MOVE TO WORKING-STORAGE, MODIFY and DELETE records. In addition, these statements will maintain the structure of the data file created by the defined chain relationships.

The communication interface between I-D-S procedural statements and the balance of the COBOL Procedure Division is the working-storage areas which are established for each level 02 field defined in the field description entries of the I-D-S Section. All COBOL references to data from the I-D-S file are to these working-storage areas.

The procedural statements of I-D-S may appear anywhere in the context of the COBOL Procedure Division. An I-D-S sentence must be preceded by ENTER IDS and terminated by a period. The sentence may contain any number of I-D-S statements. A paragraph name or section name may be assigned to an I-D-S sentence in a manner consistent with normal COBOL format.

The following pages describe these various statement and verb formats.

I-D-S Imperative Statements

The imperative statements included in this section are provided as a part of the I-D-S language to extend the function of the basic STORE and RETRIEVE verbs. The DELETE, HEAD, MODIFY and MOVE statements apply only to the RETRIEVE verb; the DEBUG and GO statements may be used with either verb. OPEN must be used prior to any other I-D-S statements; CLOSE is self-explanatory.

When these statements are used, they must occur in the order in which they are to be executed. They may be contained within the sentence beginning with the basic verb and ending with a period, or they may be used as separate sentences preceded by ENTER IDS.

The specific formats of these statements and detailed discussions of the restrictions and limitations associated with each appear on the following pages.

CLOSE

Close

Function: To transfer all modified I-D-S pages currently residing in the core buffers to the mass storage unit.

Format OPTION 1:

<u>CLOSE</u>

Format OPTION 2:

<u>CLOSE</u> WITH <u>LOCK</u>

Notes:

1. This statement must be executed before any COBOL STOP RUN statement. No automatic closing takes place.
2. OPTION 2 will insure that the data base cannot be opened again during the execution of the run unit.
3. See (Chapter 6, Accessing an I-D-S File) for Sample Deck set up.

Debug

Function: To permit the selective dumping of pages, records, current data of program chain tables, or records of chain. The output produced will appear on the system execution report.

Format:

```
                    ┌ ┌BUFFER┐   ┌   ┌BUFFER   ┐┐┐
DEBUG CURRENT       │ <RECORD>   │ ; <RECORD...>││
                    │ └CCBLOC┘   └   └CCBLOC   ┘┘│
                    └                            ┘
                    [;chain-name-2 CHAIN]
           and
                    [;TRACE chain-name-3 CHAIN]
```

Notes:

1. Chain-name-2 and chain-name-3 must be names of chains as defined by level 98 entries in the IDS Section of the Data Division.

2. The BUFFER option will result in an octal/BCD printout of the current page of the I-D-S data file.

3. The RECORD option will result in an octal/BCD printout of the logical record last accessed by a successful STORE or RETRIEVE verb.

4. The CCBLOC option will result in a printout of the following format:

DIRECT REFERENCE	Ref. Code in octal	Ref. Code in BCD
FIRST REFERENCE	Ref. Code in octal	Ref. Code in BCD
LAST REFERENCE	Ref. Code in octal	Ref. Code in BCD
RECORD TYPE	Rec. Type in octal	Rec. Type in BCD
ERROR REFERENCE		Error Code in BCD

5. The chain-name-2 CHAIN clause will result in an octal/BCD printout of the reference codes of the named chain as follows:

CHAIN TABLE HEAD	Ref. Code in octal	Ref. Code in BCD
CHAIN TABLE PRIOR	Ref. Code in octal	Ref. Code in BCD
CHAIN TABLE CURRENT	Ref. Code in octal	Ref. Code in BCD
CHAIN TABLE NEXT	Ref. Code in octal	Ref. Code in BCD

6. The TRACE chain-name-3 CHAIN clause will result in a side-by-side octal/BCD printout of all of the records contained within the specified chain.

Delete

Function: To delete the current record of the program and remove it from all chains in which it is a detail to make the record unavailable for processing and, optionally, to perform certain functions when specified detail record types are accessed during the deletion process.

Format:

```
;DELETE [ CURRENT record-name-1 RECORD [ON record-name-2 DETAIL

        [MOVE TO WORKING-STORAGE]

        [HEAD chain-name-1 CHAIN  [HEAD...]]

        [PERFORM procedure-name-1]

        [GO TO procedure-name-2]]]

       [ {OTHERWISE} ON record-name-3 DETAIL... ]
       [ {ELSE     }                            ]
```

Notes:

1. The record deleted by the DELETE statement is the record last retrieved (CURRENT) by the RETRIEVE verb.

2. The deletion process deletes a record only when there are no dependent details in its chains. When details are present, the system first attempts to delete the dependent detail records. Since the hierarchical data structure of I-D-S may involve many levels of detail records, this statement should be used with care.

3. The execution of a DELETE statement makes the record retrieved unavailable for any further processing, and an attempt to reference such a record results in an error condition.

4. The conditional statement <u>ON</u> record-name-2 DETAIL is used only when it is necessary to interrupt the deletion process when a dependent detail of the type named by record-name-2 is encountered. When the statement is used, various imperative statements immediately following are executed prior to the actual deletion of the detail record. After the execution of these statements, the deletion process is continued unless one of the statements was a GO TO statement. In this case, control is not returned to the deletion process. When the record encountered is not the type named by record-name-2 it is compared with the type named by record-name-3. The reserved words OTHERWISE or ELSE separate the tests for different record types that may be encountered. A record encountered which does not match any of the specified record types is deleted in the normal manner.

5. As a record is deleted it is <u>not</u> implicitly moved to working storage.

6. The CURRENT record-name-1 RECORD option causes the record type of the record named to be compared with the record type in the current record definition. If they are not equal, an error code (R10) is returned to the user and no deletion takes place.

GO

Go

Function: To depart from the normal in-line sequence of procedures.

Format:

;<u>GO</u> <u>TO</u> procedure-name-1

Notes:

1. Procedure-name-1 may be any COBOL or I-D-S procedural paragraph in the Procedure Division.

2. When this statement is encountered within the I-D-S sentence, all subsequent statements are bypassed and control is transferred to the procedure named.

3. GO TO may be used with:

   If ERROR...

   If record-name...

   ON record-name DETAIL...

4. GO TO must be used with:

   RETRIEVE EACH AT END...

Head

Function: To retrieve the master record of the chain specified and to move its data fields to working storage making it available for processing.

Format:

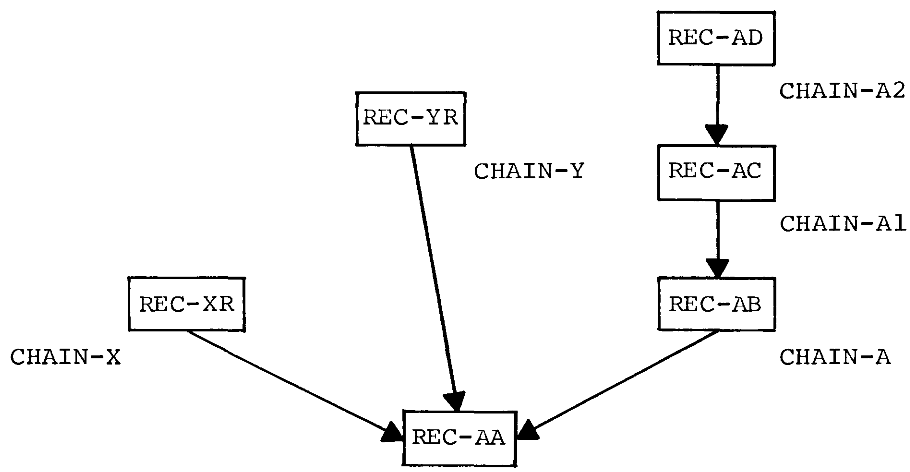
;HEAD chain-name-1 CHAIN [;HEAD...]

Notes:

1. The chain-name-1 must be a chain defined by a level 98 entry.

2. If no records of this named chain have been processed, or if the last record has been deleted, an error condition is noted.

3. A data structure in I-D-S shorthand is shown below.



In this case, assume that REC-AA was the record initially retrieved by the RETRIEVE verb. At this point, three chains include REC-AA, therefore, three possible master records may be referenced by the HEAD statement. Notice, however, that once HEAD has been used to reference CHAIN-A, the next higher level CHAIN-A1 can be referenced.

4. This statement includes an implied move of the record retrieved to working storage.

5. After execution of this statement, the master records retrieved are the CURRENT records of their respective types. They become the CURRENT records in each chain in which they are defined as details. However, they are not the CURRENT records in chains in which they are defined as master records. In those chains, the detail record which leads to the master is the CURRENT record.

6. Note that the function of the statement is very similar to that of the RETRIEVE MASTER RECORD statement, except for the manner in which CURRENT of chain is maintained (Note 5).

7. Example:

   Assume chains X, Y, A, and A1 are not PRIOR processable or HEADED (linked to MASTER). The chain tables show REC-AA after it has been retrieved via chain-X and before execution of the HEAD CHAIN-A CHAIN statement. Note that there is a chain table for each chain in which REC-AA is a detail record.

REC-AA
Chain-X

MASTER	REC-XR
PRIOR	REC-AA-1
CURRENT	REC-AA
NEXT	REC-AA+1

REC-AA
Chain-Y

MASTER	Unknown
PRIOR	Unknown
CURRENT	REC-AA
NEXT	REC-AA+1

REC-AA
Chain-A

MASTER	Unknown
PRIOR	Unknown
CURRENT	REC-AA
NEXT	REC-AA+1

After execution of the HEAD CHAIN-A CHAIN statement, the chain tables appear as shown below. Note that the chain tables for chains X and Y remain unchanged. The only change to the chain-A table is that the chain table's MASTER position has been updated with the reference code of the master record. Thus, if a RETRIEVE NEXT or PRIOR of chain X, Y, or A is issued, REC-AA is the CURRENT record from which I-D-S moves to the NEXT or PRIOR data record of chain X, Y, or A.

REC-AA
Chain-X

MASTER	REC-XR
PRIOR	REC-AA-1
CURRENT	REC-AA
NEXT	REC-AA+1

REC-AA
Chain-Y

MASTER	Unknown
PRIOR	Unknown
CURRENT	REC-AA
NEXT	REC-AA+1

REC-AA
Chain-A

MASTER	REC-AB
PRIOR	Unknown
CURRENT	REC-AA
NEXT	REC-AA+1

After the HEAD CHAIN-A CHAIN statement is executed, the chain A1 table is updated as shown below.

REC-AB
Chain-A1

MASTER	Unknown
PRIOR	Unknown
CURRENT	REC-AB
NEXT	REC-AB+1

Modify

Function: To modify the contents of all or selected fields of the current record and/or to relink any chain which may be affected by the modification of a control field.

Format Option 1:

;MODIFY field-name-1 [,field-name-2...]

Format Option 2:

;MODIFY CURRENT record-name [field-name-1 [,field-name-2...]]

Notes Option 1:

1. The fields to be modified must be level 02 entries. The contents of working storage are moved to the equivalent field of the current record which is in a data page buffer.

2. Field-name-1, field-name-2, may be control fields for the record. Modifying these fields can result in the record being logically repositioned within the I-D-S environment. Depending on the type of control field involved, I-D-S will take the following actions:

   Modifying a sort key field. The record is relinked into its chain according to the new value of the sort field. The sort field in the record is then modified.

   Modifying a randomize field. The record is relinked into a new CALC chain according to the new value of the randomize field. The randomize field in the record is then modified.

3. In relinking a record in a chain, I-D-S uses all the control fields in working storage defined in the record for that chain. Therefore, the user must not only initialize the control field to be modified, but the others as well. Depending upon the control fields involved, I-D-S will take the following action:

   Modifying a match-key field named to uniquely identify a master record. The record is relinked to the new master uniquely identified by the new value in the match-key field. Since the field is not in the detail record, no actual field modify occurs.

   Modifying field-name-5 of a MATCH-KEY IS field-name-5 SYNONYM field-name-4 clause. The record is relinked to a new master record along the chain for which the clause was named. The new master was uniquely identified by the new value in field-name-5. In this case, field-name-5 may or may not be a field in the record on disc. If it is, it is modified. If it is not, no further action is taken.

4. In no case is a record ever physically moved from one page to another in the I-D-S environment. Therefore, an attempt to modify the prime retrieval field of a primary record results in an error condition. Such a modify could result in a record needing to be moved from one page to another.

5. If the successful execution of the MODIFY statement would create DUPLICATE records in chains where they are not allowed, the modification will not be executed and an error occurs.

Notes Option 2:

1. Notes for option 1 also apply to option 2.

2. The record type of the record named is compared with the record type in the current record definition. If they are not equal, an R11 error code is returned to the user and no modification takes place.

3. If the field name option is not specified, all fields in the record are modified.

Move

Function: To move all or selected fields of the current record (record last processed) to working storage, or to move the contents of a chain table to working storage.

Format Option 1:

```
;MOVE TO WORKING-STORAGE [field-name-1 [,field-name-2...]]
```

Format Option 2:

```
                        ⎧CHAIN TABLE⎫
                        ⎪MASTER     ⎪
MOVE chain-name-1       ⎨PRIOR      ⎬      TO field-name-3
                        ⎪CURRENT    ⎪
                        ⎩NEXT       ⎭
```

Notes:

1. The implied source of an option 1 MOVE is the current record (last RETRIEVE or STORE).

2. Option 1 must be used before any reference can be made to the data in the record.

3. When the statement includes the list of fields identified by field-name-1, field-name-2, etc., only those fields are moved to working storage. Otherwise, all fields are moved.

4. When CHAIN TABLE is used in option 2, the master, prior, current, and next chain fields of the named chain are moved to four contiguous subfields specified by field-name-3. Field-name-3 should be equivalent to the form:

```
01 field-name-3
   02 Master-chain       PICTURE 9(6) COMP-1
   02 Prior-chain        PICTURE 9(6) COMP-1
   02 Current-chain      PICTURE 9(6) COMP-1
   02 Next-chain         PICTURE 9(6) COMP-1
```

5. When MASTER, PRIOR, CURRENT, or NEXT is used in option 2, the specified chain-table entry is moved to field-name-3. Field-name-3 should be equivalent to the form:

```
02 field-name-3         PICTURE 9(6) COMP-1
```

Open

Function: To initialize the processing of an I-D-S data file.

Format:

OPEN [FOR {RETRIEVAL | UPDATE}]

[WITH AUTHORITY-KEY integer-1]

Notes:

1. This statement must be executed before any other I-D-S verb is executed.

2. When the I-D-S file is opened for RETRIEVAL, the STORE, DELETE, and MODIFY statements of I-D-S are not operative. An attempt to use these statements under these conditions results in an error condition during program execution. Logically deleted records will not be physically deleted. If FOR RETRIEVAL or UPDATE is not specified, UPDATE is assumed by I-D-S.

3. The AUTHORITY-KEY clause enables access to various record types which may be protected by a defined AUTHORITY code. (See Data Division, Record Description.) The value of integer-1 may not exceed 4095(10).

   When this clause is used, each reference to a record of the I-D-S file involves a match of the AUTHORITY value defined for the record with the AUTHORITY-KEY supplied. When a valid match occurs, the I-D-S verb is allowed to function normally. Otherwise, the function of the verb is aborted and an error condition is returned to the user's program.

   The exact details of the matching process may be modified with each installation to suit individual requirements.

RETRIEVE

Retrieve

Function: To retrieve a record and make it available for processing.

Format Option 1:

$$\underline{\text{RETRIEVE}} \left\{ \begin{array}{l} \left\{\begin{array}{l}\text{record-name-1}\\ \underline{\text{CURRENT}}\ \text{record-name-1}\end{array}\right\} \text{RECORD} \\ \left\{\begin{array}{l}\underline{\text{NEXT}}\\ \underline{\text{PRIOR}}\\ \underline{\text{MASTER}}\end{array}\right\} \text{RECORD OF chain-name-2 CHAIN} \\ \underline{\text{EACH}}\ \text{AT}\ \underline{\text{END}}\ \underline{\text{GO}}\ \underline{\text{TO}}\ \text{procedure-name-1} \\ \underline{\text{DIRECT}} \end{array} \right\}$$

Format Option 2:

$$\underline{\text{RETRIEVE}}\quad \underline{\text{NEXT}}\quad \text{RECORD OF}\ \underline{\text{CALC}}\ \text{CHAIN}$$

Notes:

1. Record-name-1 must be the name of the record level 01 entry defined in the IDS Section of the Data Division.

2. Chain-name-2 must be the name of a chain defined by a level 98 entry in the IDS Section of the Data Division.

3. Regardless of the option used, this verb causes the record referenced to be retrieved and made available in the memory buffer. This action may or may not require that a page be transmitted from the mass storage device, since the record may already be in memory. No other action, such as moving the record to working storage takes place.

   The reference code of the record retrieved is accessible in the communication cell named DIRECT-REFERENCE after the retrieval process is completed.

4. Of the seven options available with the RETRIEVE verb, two may be classified as absolute. This means that only one record will satisfy the retrieval specification when one of the following options is used.

RETRIEVE record-name-1 RECORD

The record retrieval action is predicated upon the RETRIEVAL VIA clause defined in the level 01 entry in the IDS Section of the Data Division. The record retrieved depends on the values contained in the control fields of working storage which uniquely identify the record.

If the record is retrieved VIA field-name FIELD, the contents of the named field (the reference code of the record to be retrieved) are used.

If the record is retrieved VIA CALC CHAIN, the contents of the RANDOMIZE fields are used.

If the record is retrieved VIA chain-name CHAIN, the contents of its MATCH-KEY and ASCENDING and DESCENDING sort key fields are used.

RETRIEVE DIRECT

The record to be retrieved is identified by the reference code stored in a communication cell named DIRECT-REFERENCE. The user is responsible for initializing the communication cell prior to the execution of this command.

The other five options may be classified as context dependent, since the actual record retrieved is dependent upon previous record processing.

RETRIEVE CURRENT record-name-1 RECORD

The record retrieved will be the current record of record-name-1 specified. If no record of this name has been processed, or if the last record processed has been deleted, an error condition is noted.

$$\underline{\text{RETRIEVE}} \left\{ \begin{array}{l} \underline{\text{NEXT}} \\ \underline{\text{PRIOR}} \\ \underline{\text{MASTER}} \end{array} \right\} \text{RECORD OF chain-name-2 CHAIN}$$

Record retrieval depends upon the current record of the chain named. If NEXT or PRIOR is used, the appropriate record is retrieved regardless of the record type. If MASTER is specified, the master record of the chain named is retrieved. If no records of the chain have been processed, or if the last record has been deleted, such that no records exist in the chain, an error condition is noted.

RETRIEVE EACH AT END GO TO procedure-name-1

This option facilitates a reference code ascending sequence serial search of the I-D-S data file. This statement will retrieve the first record, in ascending reference code sequence, that has a reference code value equal to or greater than the reference code value stored in the FIRST-REFERENCE communication cell named. However, if the reference code value of the retrieved record is equal to or greater than the value stored in the communication cell named LAST-REFERENCE, control is transferred to procedure-name-1.

When a record is retrieved, the sum of its reference code value plus one will be stored in FIRST-REFERENCE, which initializes it for a subsequent execution of RETRIEVE EACH.

5. An option 2 entry record retrieval depends on the CURRENT record within the chain specified. If NEXT is used, the appropriate record is retrieved regardless of the record type. These record specifiers can be used only if some record has already been processed which is a member of the CALC chain.

6. If a record cannot be retrieved according to the specifications of the retrieval statement, an error condition is noted.

7. The record retrieved is recorded as the CURRENT record of its type and the CURRENT record in each chain in which it is a master or detail.

8. Example:

   The following statements will retrieve the master and detail records of the calc chain. The master of the calc chain is the Page Header record.

```
        COMPUTE DIRECT-REFERENCE = page-number * 64.
    ENTER IDS.
        RETRIEVE DIRECT   (master of calc chain)
             IF ERROR ...
        RETRIEVE NEXT of CALC chain.
             IF RECORD-TYPE = 1000 GO TO end-chain.
```

Return

Function: To relink the selected records of a specific chain into the order as returned by the sort. To return the data fields of the I-D-S record to Working-Storage.

Format:

RETURN chain-name-1 CHAIN

AT END GO TO procedure-name-1

Notes:

1. RETURN can only be used within an OUTPUT PROCEDURE associated with a SORT statement for sort-file-1. Any other use of a RETURN statement will lead to unpredictable results at object execution time.

2. The execution of the RETURN statement causes the next record in sorted order (according to the keys listed in the SORT statement) to control the retrieval of the corresponding I-D-S record in chain-name-1. The I-D-S record is then relinked into its ordered position in chain-name-1 as though the CHAIN-ORDER is described as AFTER. The chain will appear as: MASTER, 1st record from sort, 2nd record from sort, etc.

3. The data fields of the sorted selected I-D-S record will be moved to the I-D-S working-storage fields. The record returned from sort will not be available for processing in the record area associated with sort-file-1.

4. The I-D-S record will be current of program, current of type, and current of chain-name-1. The record will not be current in any other chains in which it participates.

5. Example:

```
FILE SECTION.
SD ST-FILE.
    DATA RECORD IS SORTR.
    01  SORTR.
        02  PRIOR-REF       PIC 9(6) COMP-1.
        02  CUR-REF         PIC 9(6) COMP-1.
        02  KEY-1  PIC      9999.
        02  KEY-2  PIC      999999.
    .
    .
    .
SORT-CALL SECTION.
SORT ST-FILE ON ASCENDING KEY KEY-1, KEY-2.
    INPUT PROCEDURE IS PHASE-1.
    OUTPUT PROCEDURE IS PHASE-2.
    .
    .
    .
ENTER IDS.
    RETURN TST-CHAIN CHAIN
        AT END GO TO PHASE-2X.
    .
    .
    .
```

Sort

Function: To sort the selected records of a specific chain into the specified order.

Format:

```
                         ┌ASCENDING ┐
SORT sort-file-1 ON     <            > KEY field-name-1
                         └DESCENDING┘

  [,field-name-2...] [;ON {ASCENDING | DESCENDING} KEY ...]

  {INPUT PROCEDURE IS section-name-1 [THRU section-name-2]}
  {  USING file-name-2                                    }

  GIVING chain-name-1  CHAIN
```

Notes:

1 The COBOL SORT is used to accomplish the sort of the selected I-D-S records.

2. All rules of COBOL SORT must be observed. The I-D-S exceptions are discussed in the following notes.

3. The sort-file-1 Record Description must be equivalent to the form:

```
01 SORT-IDS-REC.

   02 Prior-ref   PIC 9(6) COMP-1.
   02 Current-ref PIC 9(6) COMP-1.
   02 Sort-key-1.
        .
        .
        .
```

The prior-ref field must be the first entry in the sort record.

The current-ref field must be the second entry in the sort record.

4. The INPUT PROCEDURE must:

   RETRIEVE the I-D-S records from the specific chain.

   MOVE the PRIOR reference or zero to the prior-ref field.

   MOVE the CURRENT reference to the current-ref field.

   MOVE the data fields into the sort KEYS. (Other data may be placed in the sort record; however, I-D-S will not make use of the data.)

   RELEASE the sort record.

5. The GIVING chain-name-1 CHAIN clause means that all sorted records in sort-file-1 are used during the relink process to control the retrieval of the corresponding I-D-S record in chain-name-1. The I-D-S records are relinked into chain-name-1 as though the CHAIN-ORDER is described as AFTER. The chain will appear as MASTER, 1st record from sort, 2nd record from sort, etc.

6. If the prior-ref field is set to zero the execution of the relink function may be inefficient.

7. The chain may contain multiple record types. If only one type of record is selected for sorting, the selected sorted records will appear in order following the master record. The remaining record types will retain their relative order in the chain after all of the selected sorted records.

8. The USING file-name-2 option requires file-name-2 to be of the described format. The records must be equivalent to records which would result by using the INPUT PROCEDURE option. All records must be present in the selected chain.

9. At the completion of SORT the last record in the sort sequence will be current of program, current of type, current of chain-name-1, and its data fields will be moved to the I-D-S working-storage fields. It will not be current in any other chains in which it participates.

10. Example:

```
FILE SECTION.
SD  ST-FILE.
    DATA RECORD IS SORTR.
    01  SORTR.
        02  PRIOR-REF       PIC 9(6) COMP-1
        02  CUR-REF         PIC 9(6) COMP-1.
        02  KEY-1  PIC      9999.
        02  KEY-2  PIC      999999.
    .
    .
    .
SORT-CALL SECTION.
ENTER IDS.
SORT ST-FILE ON ASCENDING KEY KEY-1,  KEY-2
    INPUT PROCEDURE IS PHASE-1
    GIVING TST-CHAIN CHAIN.
    .
    .
    .
ENTER IDS.
    RETRIEVE MSTR.
LOOPA.
ENTER IDS.
    RETRIEVE NEXT TST-CHAIN CHAIN.
ENTER IDS.
    IF MSTR RECORD GO TO P1LAST.
ENTER IDS.
    IF DET-2 RECORD GO TO LOOPA.
ENTER IDS.
    MOVE.
    MOVE FIELDA1 TO KEY-2.
    MOVE FIELDB1 TO KEY-1.
ENTER IDS.
    MOVE TST-CHAIN PRIOR TO PRIOR-REF.
ENTER IDS.
    MOVE TST-CHAIN CURRENT TO CUR-REF.
    .
    .
    .
```

STORE

Store

Function: To place a record into the I-D-S data file, to establish any chain fields which have been defined, and to make the record available for processing.

Format:

STORE record-name-1 RECORD

Notes:

1. Record-name-1 must be defined as a level 01 entry in the IDS Section of the Data Division.

2. When this verb is used, the following is assumed:

   Working-Storage for this record has been initialized with the data contents for the record.

   Any other control fields required to provide unique identification of the master records of the defined chains which include record-name-1 have been initialized in their respective working-storage areas.

3. The record is placed into the file as defined by the PLACE NEAR or RETRIEVAL VIA clauses of the Record Description entry.

4. The reference code assigned to the record is left in the communication cell DIRECT-REFERENCE after the storage process is complete.

5. The record is recorded as the CURRENT record of its type and the CURRENT record in each chain in which it is a master or detail.

6. If the storage process creates a duplicate record in violation to any DUPLICATES NOT ALLOWED clause, or if the unique or range master selected cannot be retrieved, the storage process is terminated with all linkages restored as before and an error condition is noted.

7. When a primary record is stored, its reference code is moved to the working-storage field named by the RETRIEVAL VIA field-name FIELD clause.

8. Placement of records by I-D-S is influenced by the RETRIEVAL VIA, PAGE-RANGE, PLACE NEAR, and INTERVAL Clauses. The following summaries show priority of record storage criteria. If PAGE-RANGE is specified and the resultant page number falls outside the page range, the page number is always scaled down to fall within the page range.

9. Primary records are stored as follows:

   a. If INTERVAL is specified and the current page is not zero, on the page calculated by INTERVAL plus page of current record of the type.

   b. If not as a, above, on the page specified in DIRECT-REFERENCE, if it is not zero.

   c. If not as a or b, above, and if PLACE NEAR is specified and the CHAIN-ORDER is SORTED, SORTED WITHIN TYPE, FIRST, or LAST, on the page of the master record of the chain named in the PLACE clause.

   d. If not a, b, or c, above, on the page of the current record of the chain-name.

   e. If none of the above, on a page most convenient to I-D-S.

10. Secondary records are stored as follows:

   a. If INTERVAL is specified and the current page is not zero, on the page calculated by INTERVAL plus page of current record of the type.

   b. If not as a, above, and if PLACE NEAR is specified and the CHAIN-ORDER is SORTED, SORTED WITHIN TYPE, FIRST, or LAST, on the page of the master record of the chain named in the PLACE clause.

   c. If not as a or b, above, on the page of the current record of the chain-name.

   d. If not as a, b, or c, above, and if the CHAIN-ORDER of the RETRIEVAL VIA chain is SORTED, SORTED WITHIN TYPE, FIRST or LAST, on the page of the master record of the chain named in the RETRIEVAL VIA chain-name CHAIN clause.

   e. If none of the above, on the PAGE of the current record of the RETRIEVAL VIA chain.

11. Calculated records are stored as follows:

On the page calculated by randomizing the contents of fields named in the RANDOMIZE ON field-name clause.

12. Record storage is subject to the following Overflow rule:

If space is not available in the specified page, the record is placed on the first page in the direction of ascending page numbers in which there is available space as determined by search of the inventory records. Pages which do not have inventory records are bypassed until all pages controlled by inventory are searched. If space is not found by the inventory search, then all pages not controlled by inventory are searched. The boundaries specified by the use of a PAGE-RANGE clause are observed in this process.

I-D-S Conditional Statements

The conditional statements of I-D-S are logical extension of the basic STORE and RETRIEVE verbs. Generally, they involve the key word IF, followed by the condition to be tested, followed by the imperative statements to be performed.

I-D-S conditional statements are of two general forms; either form may appear in the string of statements following a basic verb.

The specific formats of these statements and a discussion of their restrictions and limitations follow.

Following the explanation of the IF-clause formats, PERFORM and USE, which also are conditional, are discussed.

IF

If

Function: To conditionally transfer control to an alternate procedure.

Format Option 1:

```
;IF record-name-1 RECORD   statement-1 [;statement-2...]

[{OTHERWISE}   statement-3  [;statement-4...]]
 {ELSE     }
```

Format Option 2:

```
;IF ERROR statement-1 [{OTHERWISE} statement-2 [;statement-3...]]
                       {ELSE     }
```

Notes Option 1:

1. The IF record-name-1 RECORD clause is specifically designed to support those retrieval statements where the type of record to be retrieved is unknown until after the retrieval is complete. Specifically, the IF record-name clause may only be used in conjunction with RETRIEVE DIRECT, RETRIEVE EACH, RETRIEVE NEXT and RETRIEVE PRIOR.

2. Statement-1, 2, 3, 4 may be any one of the following statements: MOVE TO WORKING-STORAGE, MODIFY, DELETE, HEAD, PERFORM, or GO TO. In addition, statement-3 may be another IF record-name clause. This allows multiple test-branch logic based on record type.

3. The record type field in the record just retrieved is compared with the record type named by record-name-1. If the record types are the same, statement-1 and subsequent statement-2's are executed in sequence and then control is transferred to the next sentence in the program. A GO TO procedure-name statement may be used as either statement-1 or statement-2 to cause a transfer to some alternate sentence in the program.

   If the record retrieved is not the type specified, then control is transferred around statement-1 and subsequent statement-2's to statement-3, or to the next sentence in the absence of an OTHERWISE or ELSE phrase.

Notes Option 2:

1. This form may only follow a STORE or RETRIEVE verb or a MODIFY, DELETE, HEAD, or MOVE imperative statement.

2. Statement-1 may only be a GO TO or a PERFORM imperative. Statement-2, statement-3, etc., may be any imperative statement appropriate to the basic verb, or a conditional of form 1, if appropriate.

3. The IF ERROR clause tests the occurrence of any logical error as a result of the last I-D-S statement. The specific errors which may occur are a function of the statement executed. The user program may determine the type of error by referring to the ERROR-REFERENCE communication cell.

4. If an error occurs because of hardware, data description, or improper use of an I-D-S function, the program is brought to an orderly halt, the file closed and the program aborted and memory dumped, if requested, with the appropriate error message.

5. If a data-dependent error is detected by I-D-S, an error code will be stored in ERROR-REFERENCE and control will pass to the IF ERROR STATEMENT.

6. The execution of a subsequent I-D-S statement will reset the error code stored in ERROR-REFERENCE.

PERFORM

Perform

Function: To depart from the normal in-line sequence of procedures in order to execute a specific procedure and then return to the normal sequence.

Format:

;PERFORM procedure-name-1 [THRU procedure-name-2]

Notes:

1. Procedure-name-1 may be any COBOL procedural paragraph in the Procedure Division.

2. For other details concerning the PERFORM statement see the GE-600 Line COBOL Reference Manual, CPB-1652. Only the simple PERFORM (option 1) is recognized within an I-D-S sentence.

3. PERFORM may be used with:

   IF ERROR...

   IF record-name...

   ON record-name DETAIL...

4. If PERFORM is used with ON record-name DETAIL, the procedure performed may not contain any I-D-S functions. The THRU procedure-name-2 may not be used.

Use

Function: To specify procedures to be executed for I-D-S error conditions which are in addition to the standard procedures supplied by I-D-S.

Format:

USE procedure-name-1 [THRU procedure-name-2]

[WITH TRACE]

ON {error-code-1 [, error-code-2...] . | ANY ABORT}

Notes:

1. The USE clause may appear anywhere within the Procedure Division.

2. The procedures specified will be executed by COBOL PERFORM.

3. The procedures may not contain I-D-S statements. The activity will be aborted if any I-D-S statements are executed while the USE procedures are being performed.

4. The I-D-S error codes used as error-code-1 and error-code-2, etc., are defined in Appendix B.

5. This clause may be used as many times as necessary to define appropriate procedures for specified error conditions.

6. Not all error codes need be specified. Selected error codes may appear in only one USE statement.

7. The ANY ABORT option may be used only once, and no other option may be used with it.

8. When a trace is made, a plain language statement defining the error and, when possible, the records or chains involved appears on the execution report. (See Appendix B.) All fatal I-D-S error conditions are traced prior to aborting.

The trace prints (1) the name of the subroutine called, (2) the name of the subroutine that called it, and (3) the alter number from which (1) was called. The trace continues to the point at which the main program is the calling routine. An example is shown below:

```
            IDS ERROR
RETRIEVE NEXT IN CHAIN NO CURRENT EXISTS MT0010-DT0020
TRACE OF ABOVE ERROR FOLLOWS -----
QUIT CALLED BY .QFWD   AT ALTER  000149
:QFWD  CALLED BY .QCHN   AT ALTER  000131
:QCHN  CALLED BY C.LDIN  AT ALTER  000054
            TRACE END
```

9. Example:

```
    .
    .
     PROCEDURE DIVISION.
     START-PARA.
           .
           .
           ENTER IDS.
           USE ERROR-PARA-1 ON D01.
              .
              .
     ERROR-PARA-1.
           DISPLAY "DUPLICATE RECORD FOUND".
              .
              .
           ENTER IDS.
           USE ABORT-PARA-1 THRU ABORT-END
               ON  15,31.
               .
               .
     ABORT-PARA-1.
           DISPLAY "RECOVERY REQUIRED - DELETE REPORTS".
              .
              .
     ABORT-END.
           CLOSE IN-FILE, OUT-FILE.
```

4. Translator Processing

The I-D-S Translator is a system program which is called from system storage by the $ IDS control card.

At the time of allocation for the I-D-S Translator, sufficient resources (memory and peripheral devices) are allocated to provide for COBOL. When the Translator has completed its function, it passes control to COBOL using the GECOS entry point GECALL. Figure 16 is a flow diagram of the compilation process of an I-D-S program.

PAGE EJECT AND COMDK LABELING

Page Eject in the Listing

The user can indicate that he desires a page eject in the listing by including a *EJECT card at the appropriate point. A *EJECT (starting in column 7) is treated as comments by the translator and causes a page eject in the listing before the printing of the *EJECT card. The *EJECT is passed to COBOL and causes a subsequent print and page eject in the COBOL portion of the listing.

COMDK Labeling

The translator uses the contents of columns 73-80 of the first source card encountered and includes it in columns 73-80 of any compressed deck created by the translator. Labeling and sequencing conform to the specifications of IOEDIT (see <u>GE-600 Line File and Record Control</u>, CPB-1003).

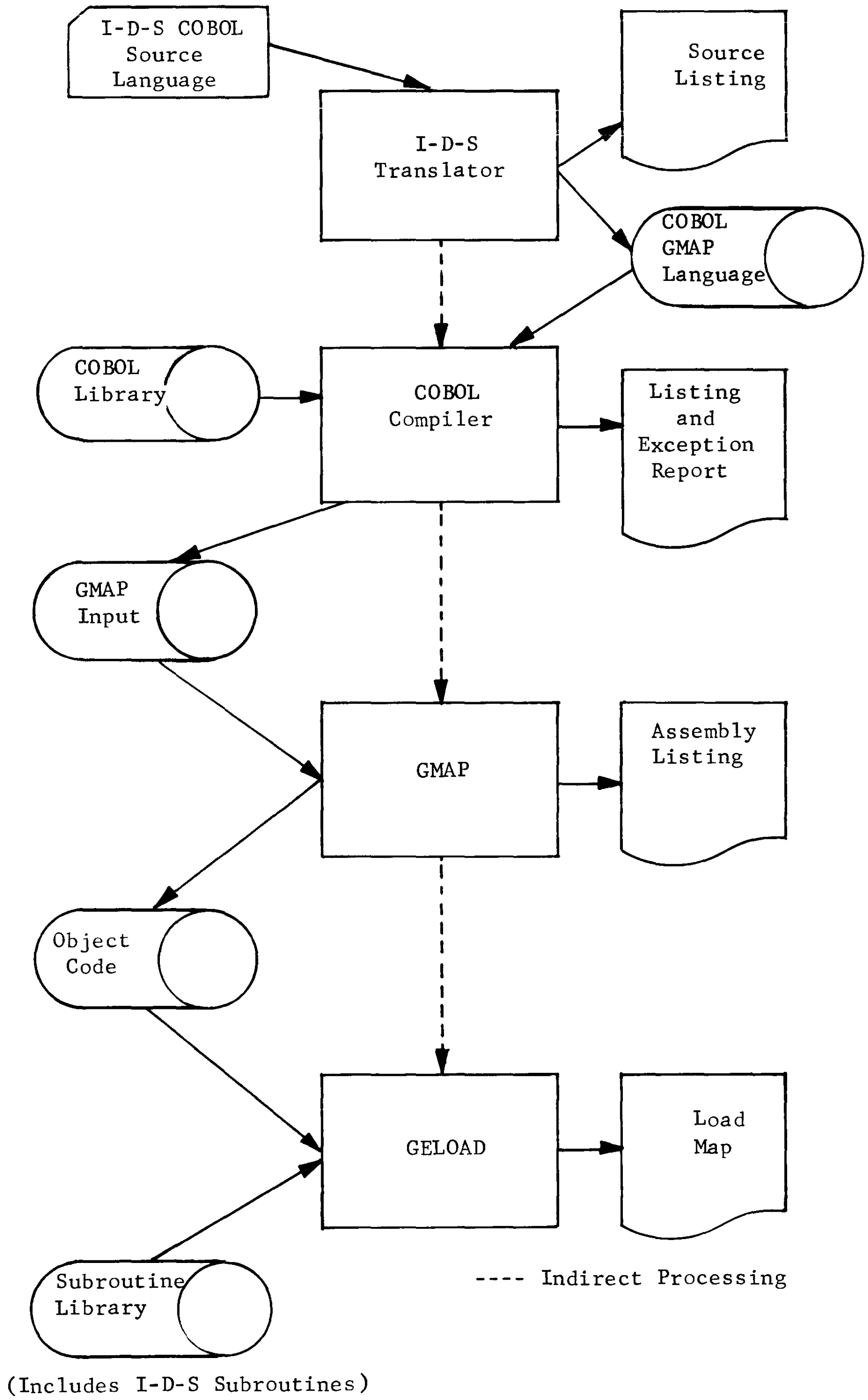


Figure 16. I-D-S Compilation and Execution Process

$ IDS CONTROL CARD DESCRIPTION

The $ IDS control card is used to call the I-D-S Translator. The operand field is used to specify the system options.

Example:

```
1      8       16
$      IDS     Options
```

Options available with I-D-S/COBOL are listed below; standard options are underlined.

<u>LSTIN</u> - An I-D-S listing and COBOL input listing will be prepared

NLSTIN - No I-D-S listing of input will be prepared. Option is reset to LSTIN prior to calling COBOL

LSTOU - A listing of assembled object program output will be prepared by GMAP

<u>NLSTOU</u> - No listing of output will be prepared

NDECK - No binary object program deck will be prepared

<u>DECK</u> - A binary object program deck will be prepared as output by GMAP

COMDK - A compressed deck of the source program will be prepared during translation

<u>NCOMDK</u> - No compressed deck of the source program will be prepared

DUMP - Slave core dump will be produced if activity terminates abnormally

<u>NDUMP</u> - Only a panel dump of program registers will be produced if activity terminates abnormally

ON6 - COBOL will generate a REF ON so that GMAP will build a Symbol Reference Table

COPY - A COBOL copy prepass is required (see rule 4)

<u>NCOPY</u> - No COBOL copy prepass is required

SYMTAB - GMAP will prepare a listing of the Symbol Reference Table (if one has been built) even though NLSTOU is also specified

Rules:

1. The $ IDS control card must precede the source cards of each program or subprogram to be processed and must precede any other control card associated with that activity.

2. The options can be listed in any order in the operand field.

3. If no options are specified in the operand field, GECOS uses the standard options (underlined).

4. All source decks which use the COPY clause or the RENAMING file option (see GE-600 Line COBOL, CPB-1652) must use the COPY option.

SAMPLE OUTPUT PRODUCED BY THE I-D-S TRANSLATOR

```
                                                                 PAGE      1
                                                              :SDL-12 CHG02
54975 02   09-26-68   10,291   GE600 INTEGRATED STORE TRANSLATOR

IDS ALTER NOS,

00001          IDENTIFICATION DIVISION .
00002          PROGRAM-ID,  5IDS .
00003 000030 AUTHOR, VANDERBUR ,
00004          DATE-WRITTEN ,   ,
00005 000050 ENVIRONMENT DIVISION,
00006 000060 CONFIGURATION SECTION,
00007 000070 SOURCE-COMPUTER, GE-635,
00008 000080 OBJECT-COMPUTER, GE-635,
00009 000090 INPUT-OUTPUT SECTION,
00010 000100 FILE-CONTROL.
00011 000110     SELECT IDS TEST-FILE ASSIGN TO TF,
00012 000130 I-O-CONTROL,
00013 000150 DATA DIVISION,
00014          FILE SECTION,
00015          WORKING-STORAGE SECTION ,
00016          77  PAGER  PICTURE 999999 COMPUTATIONAL-1 ,
00017          77  COUNT  PICTURE 9(6) COMPUTATIONAL-1 ,
00018          77  LIMIT-IS  PICTURE 999999 COMPUTATIONAL-1 ,
00019          77  CTLR  PICTURE 9(6) ,
00020          01  LOOPER ,
00021          05  LIMITER  PICTURE 9(6) ,
00022          05  FILLER  SIZE 74 ,
00023          01  GONOGO   ,
00024          05  TSTIT  PICTURE XXXXXX ,
00025          88  UGOON  VALUE "GO     " ,
00026          05  FILLER  SIZE 74 .
00027 000390 IDS SECTION,
                   01  CCBLOXK .
                   02 DIRECT-REFERENCE SIZE IS 6 USAGE IS COMPUTATIONAL-1
                   SYNCHRONIZED RIGHT,
                   02 FIRST-REFERENCE SIZE IS 8 USAGE IS COMPUTATIONAL-1
                   SYNCHRONIZED RIGHT,
                   02 LAST-REFERENCE SIZE IS 8 USAGE IS COMPUTATIONAL-1
                   SYNCHRONIZED RIGHT,
                   02 RECORD-TYPE SIZE IS 4 USAGE IS COMPUTATIONAL-1
                   SYNCHRONIZED RIGHT,
                   02 REC-FILE SIZE IS 6 CLASS IS ALPHANUMERIC
                   VALUE IS   "COCCTF",
                   02 ERROR-REFERENCE SIZE IS 3 CLASS IS ALPHANUMERIC
                   SYNCHRONIZED RIGHT,
00028 000400 MD    TEST-FILE
00029 000410       PAGE CONTAINS 1920 CHARACTERS
00030 000420       FILE CONTAINS 480 PAGES,
00031          01  PRIME-ER TYPE 555 RETRIEVAL VIA PRIME FIELD ,
00032              02  PRIME  PICTURE 99999999 .
00033              02  ABCDEF  PICTURE 999 .
00034          01  THE-MASTER
00035                  TYPE IS 990
00036              RETRIEVAL VIA CALC CHAIN
00037              PAGE-RANGE 121 TO 121
```

IDS ALTER NOS,

```
        02  MASTER-FIELD  PICTURE 999999 ,
        02  MASTER-DATA   PICTURE X(12) ,
        98  CALC CHAIN DETAIL
                 RANDOMIZE ON MASTER-FIELD ,
        98  THE-CHAIN CHAIN MASTER CHAIN-ORDER
            IS SORTED WITHIN TYPE,
        98  PAGE-TABLE CHAIN MASTER  CHAIN-ORDER IS SORTED .
        01  QUAD1
            TYPE IS 001
            RETRIEVAL VIA CALC CHAIN
            .
        02  QUAD1-NUM  PICTURE 999999 ,
        02  QUAD1-FIELD SIZE 24 .
        98  CALC CHAIN DETAIL
            RANDOMIZE ON QUAD1-NUM ,
        98  THE-CHAIN CHAIN DETAIL SELECT CURRENT
            ASCENDING KEY IS QUAD1-NUM.
        01  QUAD2
            TYPE IS 002
            RETRIEVAL VIA CALC CHAIN
            .
        02  QUAD2-NUM  PICTURE 999999 .
        02  QUAD2-FIELD SIZE 24 .
        98  CALC CHAIN DETAIL
            RANDOMIZE ON QUAD2-NUM ,
        98  THE-CHAIN CHAIN DETAIL SELECT CURRENT
            ASCENDING KEY IS QUAD2-NUM.
        01  QUAD3
            TYPE IS 003
            RETRIEVAL VIA CALC CHAIN
            .
        02  QUAD3-NUM  PICTURE 999999 .
        02  QUAD3-FIELD SIZE 24 .
        98  CALC CHAIN DETAIL
            RANDOMIZE ON QUAD3-NUM .
        98  THE-CHAIN CHAIN DETAIL SELECT CURRENT
            ASCENDING KEY IS QUAD3-NUM.
        01  QUAD4
            TYPE IS 004
            RETRIEVAL VIA CALC CHAIN
            .
        02  QUAD4-NUM  PICTURE 999999 ,
        02  QUAD4-FIELD SIZE 24 .
        98  CALC CHAIN DETAIL
            RANDOMIZE ON QUAD4-NUM ,
        98  THE-CHAIN CHAIN DETAIL SELECT CURRENT
            ASCENDING KEY IS QUAD4-NUM.
        01  PAGE-LIST
            TYPE IS 050
```

DECK SETUPS

The following deck setups show the most common uses of the I-D-S Translator: (1) translate and compile; and (2) translate, compile, and execute.

Translate and Compile

```
1      8       16
$      IDENT
$      IDS
         .
         .        Source Program
         .
$      ENDJOB
***EOF
```

With the above deck setup, the I-D-S Translator is called, and the source program is translated into a form acceptable to COBOL. COBOL is called to compile the translated program. Since no options are specified in the $ IDS card, standard I-D-S/COBOL options are used.

Translate, Compile, and Execute

```
1      8         16
$      IDENT
$      PROGRAM    QUTU
$      DATA       .Q
         .
         .        Directives
         .
$      "file"     A1,Options
$      DATA       I*
         .
         .        Directives
         .
$      IDS        Options
         .
         .        Source program
         .
$      EXECUTE    Options
$      "file"     fc,Options
                         fc is the same file code
                         specified in the File Code
                         Section of the user's program
$      DATA       .Q
         .
         .        Directives
         .
$      ENDJOB
***EOF
```

The deck setup above assumes a temporary I-D-S data base where "file" can reference any mass storage device such as PRMFL, DISC, MASS, etc.

OBJECT PROGRAM EXECUTION

The I-D-S object program consists of a modular set of subroutines which interpretively execute the I-D-S commands. GELOAD loads these subroutines as a result of the calls generated by the compilation process.

Because of the interpretive mode of execution, the complete data description of the I-D-S data file, also generated by the Translator, must be available to these routines.

Example:

Deck setup for execution using an I-D-S permanent data file.

```
1      8         16
$      IDENT     IDS00,DATABASEMGR,
$      USERID    IDSFOURYQUAD$DATABASE
$      OBJECT
         .
         .
         .
$      DKEND
$      EXECUTE
$      LIMITS
$      PRMFL     A1,R/W,R,IDSFOURYQUAD/QUAD01
$      PRMFL     A2,R/W,R,IDSFOURYQUAD/QUAD02
$      PRMFL     A3,R/W,R,IDSFOURYQUAD/QUAD03
$      PRMFL     A4,R/W,R,IDSFOURYQUAD/QUAD04
         .
         .
         .
```

5. I-D-S Data File Structure Descriptions

DEFINITION STRUCTURE

The Definition Structure required by I-D-S is a list structure which reflects the description of the records of the I-D-S data file. It defines the master/detail relationships between records, chain claracter6stics, and the physical and control characteristics of every field of every record type in the I-D-S data file.

The organization of a Definition Structure using the I-D-S shorthand technique is shown in Figure 17.

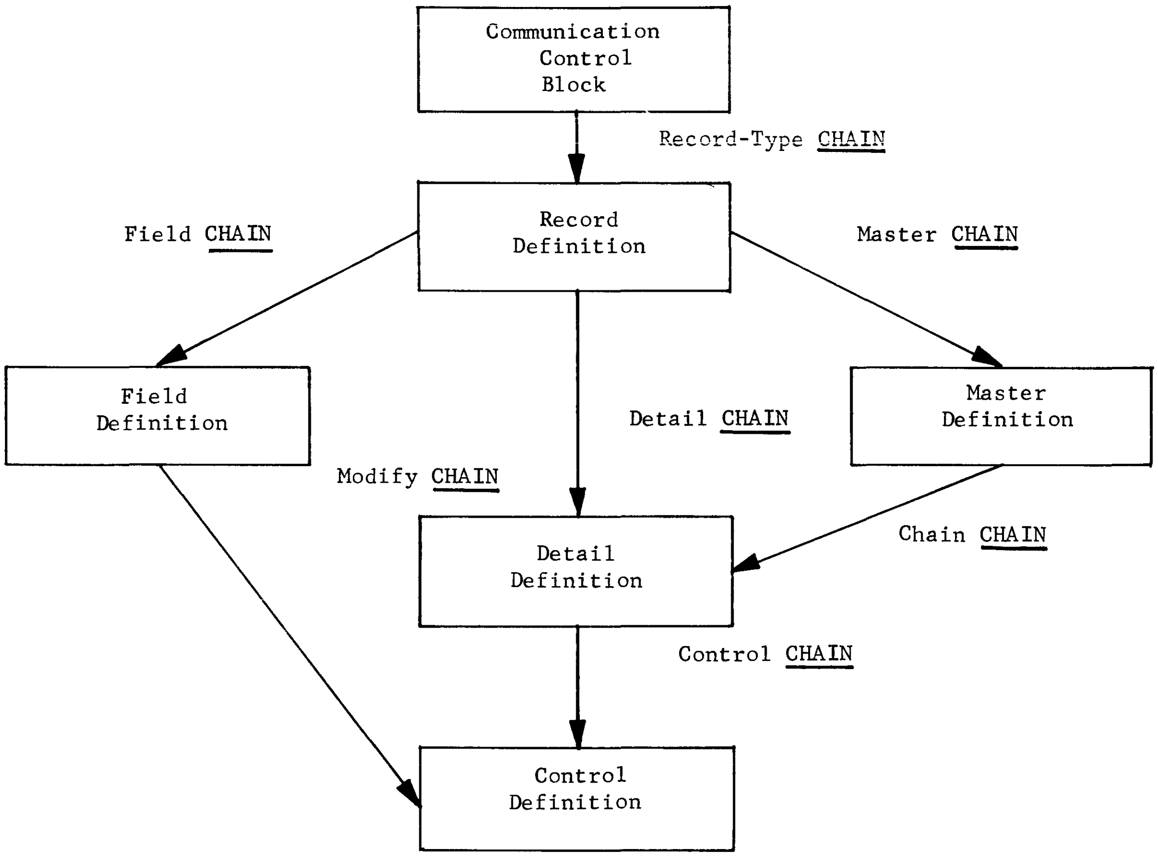


Figure 17. I-D-S Definition Structure

The Definition Structure is described in the following sections.

Communication Control Block

The Communication Control Block entry must be supplied as the master of the Record-Type Chain. It serves as the communication area for data which must be passed between the user's program and the I-D-S subroutines. The machine format of the entry is shown in Figure 18.

Bits

0 5 11 17 23 29 35

<table>
<tr><td>LOC-CCB</td><td>0 0</td><td>MBZ</td><td colspan="4">DIRECT-REFERENCE</td></tr>
<tr><td>+1</td><td colspan="2">MBZ</td><td colspan="4">FIRST-REFERENCE</td></tr>
<tr><td>+2</td><td colspan="2">MBZ</td><td colspan="4">LAST-REFERENCE</td></tr>
<tr><td>+3</td><td colspan="2">MBZ</td><td colspan="4">Record Type</td></tr>
<tr><td>+4</td><td colspan="3">Record Type Chain Next</td><td>MBZ</td><td colspan="2">File Code</td></tr>
<tr><td>+5</td><td colspan="3">MBZ</td><td colspan="3">ERROR-REFERENCE</td></tr>
<tr><td>+6</td><td>MBZ</td><td colspan="2">AUTHORITY</td><td colspan="2">MBZ</td><td>OPEN Mode</td></tr>
</table>

Figure 18. Format of Communication Control Block Entry

The bit structure of the format shown in Figure 18 serves the following purposes:

LOC-CCB Symbolic location of Communication Control Block.

0-5 Definition Type--an octal code of 00.

6-11 Must be zero.

12-35 DIRECT-REFERENCE--a reference code of the record last processed by any STORE or RETRIEVE.

LOC-CCB+1

0-11 Must be zero.

12-35 FIRST-REFERENCE--reference code of the first record to be retrieved by the RETRIEVE EACH verb. The value is supplied by the user's program. After each retrieval, the I-D-S subroutines modify the value to the next reference code.

LOC-CCB+2

0-11 Must be zero.

12-35 LAST-REFERENCE--a value supplied by the user's program. When this reference code is reached, the RETRIEVE EACH verb will execute the AT END procedure.

LOC-CCB+3

0-11 Must be zero.

12-35 Record type of the last record processed by any STORE or RETRIEVE. The value is supplied by an I-D-S subroutine.

LOC-CCB+4

0-17 Record Type Chain Next--the assigned symbol of the first Record Definition Structure.

18-23 Must be zero.

24-35 File Code--the user-supplied, two-character file code assigned to the I-D-S data file.

LOC-CCB+5

0-17 Must be zero.

18-35 ERROR-REFERENCE--a BCD code for an error condition encountered during execution. If there is no error, the value supplied by I-D-S will be zero.

LOC-CCB+6

0-5 Must be zero.

6-17 AUTHORITY--a value supplied by the user.

18-29 Must be zero.

30-35 OPEN Mode--a processing mode supplied via the OPEN routine.

Record Definition Entry

A Record Definition entry must be supplied for each data record type in the I-D-S data file. In addition, one such entry must be supplied for the Page Header record. The Record Definition entry is the master of the Master Chain, Detail Chain, and the Field Chain; it is a detail of the Record-Type Chain. The format is shown in Figure 19.

	0–5	6–7	8–17	18–29	30–32	33	34	35
LOC-SYM	0 1	MBZ	Record Type	Record Size	MBZ	S	P	R
+1	Page Interval			Master Chain Next				
+2	Field Chain Next			Detail Chain Next				
+3	Authority	Current Record Reference Code						
+4	Record Type Chain Next			MBZ				
+5	Minimum Page Range			Maximum Page Range				

Figure 19. Format of Record Definition Entry

The bit structure of the format shown in Figure 19 serves the following purposes:

LOC-SYM Symbol equivalent to the record name.

0-5 Definition Type--an octal code of 01.

6-7 Must be zero.

8-17 Record Type--a number from 1 to 999 assigned to each data record; 1000(10) is assigned to the Page Header record; for a Pagette Header record, the number is 1003(10).

18-29 Record Size--number of characters in the record including all control and chain fields.

30-32 Must be zero.

33 S--Storage Classification Indicator

0--Record is stored relative to the chain defined as the Retrieval Chain for this record.
1--Record is stored relative to the chain defined as the Storage Chain, which is not the same as the Retrieval Chain.

34 P--Page Range Indicator

0--Absolute Page Range not specified for this record type (see LOC-SYM+5).
1--Absolute Page Range is specified (see LOC-SYM+5).

35 R--Retrieval Classification Indicator

0--Secondary or calculated record
1--Primary record

LOC-SYM+1

0-17 Page Interval--Number of pages to be skipped relative to the page in which the last record of this type was stored. This only applies to primary or secondary records.

18-35 Master Chain Next--the assigned symbol of the first Master Definition for this record. If this record is not the master of any chain, this is the assigned symbol of the Record Definition.

LOC-SYM+2

0-17 Field Chain Next--assigned symbol of the first Field Definition for this record. If there are no data fields, then this is the assigned symbol of the Record Definition.

18-35 Detail Chain Next--assigned symbol of the first Detail Definition for this record. If this record is not the detail in any chain, then this is the assigned symbol of the Record Definition.

LOC-SYM+3

0-11 AUTHORITY--A value supplied by the user not to exceed 4095(10) which serves as a lock for the data contained in the record. Reference to this record during program execution is allowed only when a matching key is specified by the .QOPEN calling sequence.

12-35 Current Record Reference Code--reference code of the last record stored or retrieved of this record type. This is supplied by I-D-S during execution.

LOC-SYM+4

0-17 Record Type Chain Next--assigned symbol of the next Record Definition of the Definition Structure. Uf this is the last Record Definition entry, this field contains the symbolic location of the Communication Control Block.

18-35 Must be zero.

LOC-SYM+5

(For P, see LOC-SYM, bit 34.)

If P = 1, then:

0-17	Minimum Page Range--the first page number of a range of pages into which all records of this type are to be stored.
18-35	Maximum Page Range--the last page number of a range of pages into which all records of this type are to be stored.

If P = 0 and LOC-SYM+5, bits 0-35 = 0, then:

No Page Range is specified for this record type.

If P = 0 and LOC-SYM+5, bits 0-35 ≠ 0, then:

0-17	Minimum Page Range Pointer--points to a word in which bit positions 18-35 contain the first page number of a range of pages into which all records of this type are to be stored.
18-35	Maximum Page Range Pointer--points to a word in which bit positions 18-35 contain the last page number of a range of pages into which all records of this type are to be stored.

Detail Definition

A Detail Definition entry must be supplied each time a record is a detail in some chain. If a record is a detail in three different chains, three Detail Definition entries must be supplied. The Detail Definition entry is a detail of the Chain Chain and of the Detail Chain. It is also the master of the Control Chain. The machine format of this entry is shown in Figure 20.

<table>
<tr><td>bits</td><td>0</td><td>5</td><td>8</td><td>11</td><td>17</td><td></td><td>25</td><td>29</td><td>31</td><td></td><td></td><td>35</td></tr>
<tr><td>LOC-SYM</td><td>0 4</td><td>MBZ</td><td colspan="2">Data Record Type</td><td colspan="2">MBZ</td><td>Order</td><td>DUP</td><td>C</td><td>U</td><td>S</td><td>R</td></tr>
<tr><td>+1</td><td colspan="4">Chain Chain Next</td><td colspan="8">Control Chain Next</td></tr>
<tr><td>+2</td><td colspan="4">Chain Chain Head</td><td colspan="8">Detail Chain Next</td></tr>
<tr><td>+3</td><td colspan="3">Next Position</td><td>MBZ</td><td colspan="8">Detail Chain Head</td></tr>
<tr><td>+4</td><td colspan="3">Prior Position</td><td colspan="2">Head Position</td><td colspan="7">MBZ</td></tr>
</table>

Figure 20. Machine Format for Detail Definition Entry

The areas in the format shown in Figure 20 serve the following purposes:

LOC-SYM Symbol assigned to this entry.

0-5 Definition Type--an octal code of 04.

6-7 Must be zero.

8-17 Data Record Type--same as that specified by the Record Definition entry for this record.

18-25 Must be zero.

26-29 Order--a code to represent the chain-order of the various details of this chain. Note that when several different record types are defined as details of the same chain, the chain-order must be the same for all records. The chain-order for a CALC chain must be 11(8) for after current.

Octal Code	Chain-order
06	Sorted Within Type
04	Sorted
10	First in Chain
00	Last in Chain
01	Before Current
11	After Current

30-31 DUP--Duplicate Records Indicator

00--Not allowed
01--Allowed First
11--Allowed Last

32 C--CALC Chain Detail Indicator

0--Not a CALC Chain
1--CALC Chain

33 U--Chain Master Indicator

0--The master of this chain is a unique master retrievable via the MATCH-KEY fields defined for this chain.
1--The master of this chain is the current master record of its type.

34 S--Storage Chain Indicator

0--Record is not stored relative to this chain.
1--Record is stored relative to its logical position in this chain.

35 R--Retrieval Chain Indicator

0--Associative retrieval of this record not possible via this chain.
1--Associative retrieval of this record must be via this chain.

LOC-SYM+1

0-17 Chain Chain Next--assigned symbol of the next Detail Definition of this chain if there is more than one detail record type in the chain. If there is only one Detail Definition or if this is the last of several, then this is the assigned symbol of the Master Definition for this chain.

18-35 Control Chain Next--assigned symbol of the first Control Definition for this chain or, if none, the symbol assigned to this Detail Definition.

LOC-SYM+2

0-17 Chain Chain Head--assigned symbol of the Master Definition of this chain.

18-35 Detail Chain Next--assigned symbol of the next Detail Definition for this record if the record is a detail in more than one chain. If there is only one Detail Definition or if this is the last of several, then this is the assigned symbol of the Record Definition for this record.

LOC-SYM+3

0-11 Next Position--the character position, relative to the first character of the record, in which the first character of the chain next pointer is found.

If this is a CALC chain detail, the NEXT chain field must be the first field following the Record Size Field of the record; that is, it must be defined as beginning in character position 5.

12-17 Must be zero.

18-35 Detail Chain Head--assigned symbol of the Record Definition for this record.

LOC-SYM+4

0-11 Prior Position--the character position, relative to the first character of the record, in which the first character of the chain prior pointer is found. If the chain is not prior processable, this value is zero.

When a detail record of a given chain contains a prior pointer, all records of the chain must contain a prior pointer.

12-23 Head Position--the character position, relative to the first character of the record, in which the first character of the chain head pointer is found. If the chain is not a headed chain, this character is zero.

24-35 Must be zero.

Master Definition

A Master Definition entry must be supplied each time a record is defined as the master of some chain. The Master Definition is a detail of the Master Chain and the master of the Chain Chain. The machine format of this entry is shown in Figure 21.

Bits	0 5	7	11 17	35
LOC-SYM	0 2	MBZ	Data Record Type	Master Chain Head
+1	Chain Chain Next			Master Chain Next
+2	MBZ		Reference Code of Chain Master	
+3	Next Position		Reference Code of Chain Prior	
+4	Prior Position		Reference Code of Chain Current	
+5	MBZ		Reference Code of Chain Next	
+6	MBZ		Reference Code of Key Record	

Figure 21. Machine Format for Master Definition Entry

The areas in the format shown in Figure 21 serve the following purposes:

LOC-SYM Symbol equivalent to chain name.

0-5 Definition Type--an octal code of 02

6-7 Must be zero.

8-17 Data Record Type--same as that specified for the Record Definition entry for this record.

18-35 Master Chain Head--assigned symbol of the Record Definition entry for this record.

LOC-SYM+1

0-17 Chain Chain Next--assigned symbol of the first Detail Definition for this chain. If the chain has no detail records defined, then this is the symbol of this Master Definition.

18-35 Master Chain Next--assigned symbol of the next Master Definition if this record is the master of more than one chain. If the record is the master of only one chain or the master of the last of several chains, then this coding is the symbol of the Record Definition for this record.

LOC-SYM+2

0-11 Must be zero.

12-35 Reference Code of Chain Master--reference code of the Master Record of the chain defined by this Master Definition. This value is supplied by I-D-S during execution.

LOC-SYM+3

0-11 Next Position--the character position, relative to the first character of the record, in which the first character of the chain next pointer is found.

12-35 Reference Code of Chain Prior--reference code of the prior record of the chain defined by this Master Definition. This is supplied by I-D-S during execution.

LOC-SYM+4

0-11 Prior Position--the character position, relative to the first character of the record, in which the first character of the chain prior pointer is found. If the master record is not prior processable, this value is zero.

12-35 Reference Code of Chain Current--reference code of the current record of the chain defined by this Master Definition. This value is supplied by I-D-S during execution.

LOC-SYM+5

0-11 Must be zero.

12-35 Reference Code of Chain Next--reference code of the next record of the chain defined by this Master Definition. This is supplied by I-D-S during execution.

LOC-SYM+6

0-11 Must be zero.

12-35 Reference Code of Key Record--reference code of the record to which a record will be relinked if there is an error in modification. This code is supplied by I-D-S during execution.

Field Definition

A Field Definition entry must be supplied for each data field contained in the record. (Note that Field Definitions are not supplied for the In addition, if the record is defined as a secondary record, a Field Definition must be supplied for all MATCH-KEY fields defined. If the record is defined as a primary record, a Field Definition must be supplied for the field which is equivalent to the reference code. The Field Definition entry is a detail in the Field Chain and is the master of the Modify Chain. The machine format of the entry is shown in Figure 22.

	0	5	17				24 33 35
LOC-SYM	1 0	MBZ	C	AF	U	MBZ	Field Increment
+1	Location of Working Storage		Field Size			MBZ	First Char.
+2	Field Chain Next		Modify Chain Next				

Figure 22. Machine Format for Field Definition Entry

The areas in the format shown in Figure 22 serve the following purposes:

LOC-SYM Symbol assigned to this entry.

0-5 Definition Type--an octal code of 10.

6-17 Must be zero.

18 C--Computational Mode Indicator (*)

0--Noncomputational field recorded in BCD.
1--Computational field recorded in binary. (The implied size is 6 or 12 characters.)

19-20 AF--Arithmetic Form (*)

If bit 18=1 then:

00--Single Precision, Fixed Point
01--Single Precision, Floating Point
10--Double Precision, Fixed Point
11--Double Precision, Floating Point,

If bit 18=0 then:

00--Alphanumeric
01--Alphabetic
10--Numeric
11--Signed numeric (sign indicated by zone bits of low-order character of the field).

21 U--Unique Field Indicator

0--Field is not a unique or control field
1--Field is unique and required for identification of the record

When this record is a primary record its unique field is, by definition, the reference code. Since a Field Definition entry is not supplied for the reference code, a separate entry must be supplied to define the working-storage location for the field which is equivalent to the reference code. This entry must not include the Field Definition specifications indicated in this section by (*), since the field is not actually contained in the data record. I-D-S assumes that the format of this field in working storage is eight characters, BCD numeric.

22-23 Must be zero.

24-35 Field Increment (*)--character position of the first character of a field; increment zero is the first character of the record.

LOC-SYM+1

0-17 Location of Working Storage--assigned symbol of the leftmost word of working storage defined for this field. The symbol is equivalent to the field name.

18-29 Field Size--the number of characters in the field as it exists in the record or in working storage.

30-32 Must be zero.

33-35 First Character--position of the first character of the field within the first word of working storage.

LOC-SYM+2

0-17 Field Chain Next--assigned symbol of the next Field Definition of this record, if there is more than one field in the record. If there is only one field or if this is the last of several, then this value is the assigned symbol of the Record Definition for the record.

18-35 Modify Chain Next--assigned symbol of the first Control Definition for this field or, if the field is not a control field, the symbol of this Field Definition.

Control Definition

A Control Definition entry must be supplied each time a field is defined as a control field of some chain. A control field is defined as a sort field, MATCH-KEY field, or a RANDOMIZE field. The Control Definition entry is a detail of the Modify Chain and of the Control Chain. The machine format of this entry is shown in Figure 23.

	0	5	14	17	35
LOC-SYM	2 0	MBZ	R	CNTL	Control Chain Head
+1	Location of MATCH-KEY Field Definition				Control Chain Next
+2	Modify Chain Head				Modify Chain Next

Figure 23. Machine Format for Control Definition Entry

The areas in the format shown in Figure 23 serve the following purposes:

LOC-SYM Symbol assigned to this entry.

0-5 Definition Type--an octal code of 20

6-13 Must be zero.

14 R--Match Control Indicator

0--Equal match required
1--Match equal or greater (Range Record)

15-17 CNTL--Control field type

001--RANDOMIZE control field
010--Sort Control ascending sequence
011--Sort Control descending sequence
100--MATCH-KEY control field

18-35 Control Chain Head--assigned symbol of the Detail Definition of the chain controlled by this Control Definition.

LOC-SYM+1

0-17 Location of MATCH-KEY Field Definition--assigned symbol of the MATCH-KEY Field Definition associated with this SYNONYM Field. If there is no SYNONYM, this symbol is zero.

18-35 Control Chain Next--assigned symbol of the next Control Definition for the chain. If this is the last or only Control Definition, then the code is the symbol of the Detail Definition.

When several sort control fields are defined for a given chain, they must occur in sequence from major sort control to minor sort control.

LOC-SYM+2

0-17 Modify Chain Head--assigned symbol of the Field Definition for this control field.

18-35 Modify Chain Next--assigned symbol of the next Control Definition if this field is a control field in some other chain. If this is the last or only Control Definition for this field, then the code is the symbol of the Field Definition.

A definition structure produced by the I-D-S Translator and a definition structure as expanded by GMAP appear on the following pages.

```
42447 09  08-06-69   17,473   GE600 INTEGRATED STORE TRANSLATOR   ISDL-2 CHG00
IDS ALTER NCS,
                 ETC       CALC
         RD2435 ,QFD       0,0,000015,0024,FC5697,RD2434,
                 ETC       RD2435,RD5697,
                 ETC       QUAD3-FIELD
         RD2434 ,QFD       0,0,000009,0006,FC5953,RD2433,
                 ETC       RD2437,RD5953,
                 ETC       QUAD3-NUM
         RD2437 ,QCD       0,1,RD2436,RD2436,RD2434,RD2434,
                 ETC       0
         RD7233 ,QRD       002,000043,0,0,0,000000,
                 ETC       RD7233,RD7238,RD7235,0000,RD4737,
                 ETC       000000,000000,QUAD2
         RD7238 ,QDD       002,10,0,0,1,0,
                 ETC       0,RD7236,RD7233,RD4742,RD4097,RD7238,
                 ETC       0039,0000,0000,
                 ETC       THE-CHAIN
         RD7236 ,QDD       002,11,0,1,0,1,
                 ETC       1,RD7233,RD7233,RD4740,RD8129,RD7237,
                 ETC       0005,0000,0000,
                 ETC       CALC
         RD7235 ,QFD       0,0,000015,0024,FC4353,RD7234,
                 ETC       RD7235,RD4353,
                 ETC       QUAD2-FIELD
         RD7234 ,QFD       0,0,000009,0006,FC1025,RD7233,
                 ETC       RD7237,RD1025,
                 ETC       QUAD2-NUM
         RD7237 ,QCD       0,1,RD7236,RD7236,RD7234,RD7234,
                 ETC       0
         RD4737 ,QRD       001,000043,0,0,0,000000,
                 ETC       RD4737,RD4742,RD4739,0000,RD4481,
                 ETC       000000,000000,QUAD1
         RD4742 ,QDD       001,10,0,0,1,0,
                 ETC       0,RD4740,RD4737,RD4097,RD4097,RD4742,
                 ETC       0039,0000,0000,
                 ETC       THE-CHAIN
         RD4740 ,QDD       001,11,0,1,0,1,
                 ETC       1,RD4737,RD4737,RD4484,RD8129,RD4741,
                 ETC       0005,0000,0000,
                 ETC       CALC
         RD4739 ,QFD       0,0,000015,0024,FC1537,RD4738,
                 ETC       RD4739,RD1537,
                 ETC       QUAD1-FIELD
         RD4738 ,QFD       0,0,000009,0006,FC8065,RD4737,
                 ETC       RD4741,RD8065,
                 ETC       QUAD1-NUM
         RD4741 ,QCD       0,1,RD4740,RD4740,RD4738,RD4738,
                 ETC       0
         RD4481 ,QRD       990,000035,0,1,0,000000,
                 ETC       RD7809,RD4484,RD4483,0000,RD0513,
                 ETC       000121,000121,THE-MASTER
```

```
IDS ALTER NCS,

              RD7809 ,QMD     990,RD4097,RD4481,RD0196,0031,0000,
                     ETC      PAGE-TABLE
              RD4097 ,QMD     990,RD4481,RD4481,RD5062,0027,0000,
                     ETC      THE-CHAIN
              RD4484 ,QDD     990,11,0,1,0,1,
                     ETC      1,RD4481,RD4481,RD8129,RD8129,RD4485,
                     ETC      0005,0000,0000,
                     ETC      CALC
              RD4483 ,QFD     0,0,000015,0012,FC0321,RD4482,
                     ETC      RD4483,RD0321,
                     ETC      MASTER-DATA
              RD4482 ,QFD     0,0,000009,0006,FC1089,RD4481,
                     ETC      RD4485,RD1089,
                     ETC      MASTER-FIELD
              RD4485 ,QCD     0,1,RD4484,RD4484,RD4482,RD4482,
                     ETC      0
              RD0513 ,QRD     1000,000022,0,0,1,000000,
                     ETC      RD8129,RD0513,RD0513,0000,CCBLOC,
                     ETC      000000,000000,XPAGE-HEADXX
              RD8129 ,QMD     1000,RD0513,RD0513,RD5060,0005,0000,
                     ETC      CALC
                     USE
                     TRA      ,,IDS,
              LS9000 ZERO     000000,000002
              EE0001 VFD      26/0,10/990
              DE0001 ZERO     0,03
              ,,IDS, NULL
                     ENTER    COBOL,
              ENTER DEFINITIONS ,
                   SYMBOL RD3137 EQUALS
                   LINE-NO
                   SYMBOL FC3137 EQUALS INITIAL CHARACTER OF
                   LINE-NO
                   IDS SIZE    000002 EQUALS
                   LINE-NO
                   SYMBOL RD1473 EQUALS
                   PAGE-NO
                   SYMBOL FC1473 EQUALS INITIAL CHARACTER OF
                   PAGE-NO
                   IDS SIZE    000006 EQUALS
                   PAGE-NO
                   SYMBOL RD1217 EQUALS
                   QUAD4-FIELD
                   SYMBOL FC1217 EQUALS INITIAL CHARACTER OF
                   QUAD4-FIELD
                   IDS SIZE    000024 EQUALS
                   QUAD4-FIELD
                   SYMBOL RD7873 EQUALS
                   QUAD4-NUM
                   SYMBOL FC7873 EQUALS INITIAL CHARACTER OF
```

```
                                                          RD8129     CHN CHN NXT, RD4485    CON NXT,
     000216  004240 004170    033              ZERO      RD8129,RD4481
                                                          RD8129     CHN CHN HD, RD4481      DET NXT,
  END OF BINARY CARD 5IDS0032
     000217  000500004170     003              VFD       12/0005,6/0,18/RD4481
                                                          0005     POS NXT,  RD4481     DET CHN HD
     000220  000000000000     000              VFD       12/0000,12/0000,12/0
                                                          0000     PRIOR,  0000     HEAD
                                                          DETAIL OF CALC CHAIN
                       000221        522 RD4483 ,QFD     0,0,000015,0012,FC0321,RD4482,
                       000221        523        ETC      RD4483,RD0321,
                       000221        524        ETC      MASTER-DATA
     000221  100000000017     000              VFD       06/10,12/0,3/0,1/0,2/0,12/000015
                                                          0    CAF, 0    U, 000015 FLD INCR,
     000222  000755001400     010              VFD       18/RD0321,12/0012,3/0,3/FC0321
                                                          RD0321      WS, 0012    FLD SZ, FC0321  FST CH,
     000223  004224 004221    033              ZERO      RD4482,RD4483
                                                          RD4482     FLD CHN NXT, RD4483   MOD CHN NXT
                                                          MASTER-DATA  ****FIELD-NAME****
                       000224        525 RD4482 ,QFD     0,0,000009,0006,FC1089,RD4481,
                       000224        526        ETC      RD4485,RD1089,
                       000224        527        ETC      MASTER-FIELD
     000224  100000000011     000              VFD       06/10,12/0,3/0,1/0,2/0,12/000009
                                                          0    CAF, 0    U, 000009 FLD INCR,
     000225  000754000600     010              VFD       18/RD1089,12/0006,3/0,3/FC1089
                                                          RD1089      WS, 0006    FLD SZ, FC1089  FST CH,
     000226  004170 004227    033              ZERO      RD4481,RD4485
                                                          RD4481     FLD CHN NXT, RD4485   MOD CHN NXT
                                                          MASTER-FIELD  ****FIELD-NAME****
                       000227        528 RD4485 ,QCD     0,1,RD4484,RD4484,RD4482,RD4482,
                       000227        529        ETC      0
     000227  200001004214     003              VFD       06/20,8/0,1/0,3/1,18/RD4484
                                                          0    R, 1    CNTL, RD4484 CON CHN HD,
     000230  000000 004214    003              ZERO      0,RD4484
                                                          0  LOC SYN W.S., RD4484 CON, CHAIN NEXT
     000231  004224 004224    033              ZERO      RD4482,RD4482
                                                          RD4482  MOD CH HEAD, RD4482  MOD CHN NXT
                       000232        530 RD0513 ,QRD     1000,000022,0,0,1,000000,
                       000232        531        ETC      RD8129,RD0513,RD0513,0000,CCBLOC,
                       000232        532        ETC      000000,000000,XPAGE-HEADXX
                                                VFD       06/1,2/0,10/1000,12/000022,3/0,
     000232  011750002601     000              ETC       1/0,1/0,1/1
                                                          1000     RECORD TYPE, 000022     REC SIZE,
                                                          0      S, 0      P, 1      R,
     000233  000000 004240    003              ZERO      000000,RD8129
                                                          000000     PG INT, RD8129     MST CHN NEXT,
     000234  004232 004232    033              ZERO      RD0513,RD0513
                                                          RD0513     FLD CHN NXT, RD0513     DET CHN NXT,
     000235  000000000000     000              VFD       12/0000,24/0
                                                          0000     AUTHORITY,
     000236  000744 000000    010              ZERO      CCBLOC,0
                                                          CCBLOC     REC TYPE CHN NXT
```

```
     000237  000000 000000   000                  ZERO     000000,000000
                                                           000000  PAGE R MIN, 000000 PAGE R MAX
                                                           XPAGE-HEADXX  ****RECORD=NAME****
                     000240      533 RD8129 .QMD     1000,RD0513,RD0513,RD5060,0005,0000,
                     000240      534            ETC      CALC
     000240  021750004232    000                  VFD      06/2,2/0,10/1000,18/RD0513
                                                           1000     REC TYPE%  RD0513     MST CHN HD,
     000241  004037 004232   033                  ZERO     RD5060,RD0513
                                                           RD5060     CHN CHN NXT, RD0513    MST CHN NXT,
  END OF BINARY CARD 5IDS0053
     000242  000000000000    000                  VFD      12/0,24/0
     000243  000500000000    000                  VFD      12/0005,24/0
                                                           0005     POS NEXT,
     000244  000000000000    000                  VFD      12/0000,24/0
                                                           0000     POS PRIOR
                     000245                       BSS      2
                                                           MASTER OF CALC CHAIN
                     000610      535              USE
     000610  000614 7100 00  010 536              TRA      ..IDS.
     000611  000000 000002   000 537 LS9000 ZERO     000000,000002
     000612  000000001736    000 538 EB0001 VFD      26/0,10/990
     000613  000000 000003   000 539 DE0001 ZERO     0,03
                     000614      540 ..IDS. NULL
                                 541 *      ENTER DEFINITIONS ,
     000614  060000 7010 00  030 542 E00002 TSX1     .CMPET                                 000238
                                 543            EDITP    ON                                     000238
```

.QRD - RECORD DEFINITION

(See Figure 19. Format of Record Definition Entry.)

Line	Item	
1	1	RECORD TYPE
1	2	RECORD SIZE
1	3	S - STORAGE CLASSIFICATION INDICATOR
1	4	P - PAGE RANGE INDICATOR
1	5	R - RETRIEVAL CLASSIFICATION INDICATOR
1	6	PAGE INTERVAL
2	7	MASTER CHAIN NEXT
2	8	DETAIL CHAIN NEXT
2	9	FIELD CHAIN NEXT
2	10	AUTHORITY
2	11	RECORD TYPE NEXT
3	12	MINIMUM PAGE RANGE
3	13	MAXIMUM PAGE RANGE
3	14	RECORD NAME

FORMAT

```
                  1      2     3 4 5    6
RDxxxx   .QRD    xxx,xxxxxx,x,x,x,xxxxxx,             Line 1
                    7       8       9     10     11
         ETC     RDxxxx,RDxxxx,RDxxxx,xxxx,RDxxxx,    Line 2
                   12      13      14
         ETC     xxxxxx,xxxxxx,x(30)                  Line 3
```

TRANSLATOR OUTPUT (see preceding Definition Structure sample)

```
RD7233   .QRD    002,000043,0,0,0,000000,
         ETC     RD7233,RD7238,RD7235,0000,RD4737
         ETC     000000,000000,QUAD2
```

.QDD - DETAIL DEFINITION

(See Figure 20. Machine Format for Detail Definition Entry.)

Line	Item	
1	1	RECORD TYPE
1	2	CHAIN ORDER
1	3	DUPLICATE RECORD INDICATOR
1	4	CALC CHAIN DETAIL INDICATOR
1	5	U - CHAIN MASTER INDICATOR
1	6	S - STORAGE CHAIN INDICATOR
2	7	R - RETRIEVAL CHAIN INDICATOR
2	8	DETAIL CHAIN NEXT
2	9	DETAIL CHAIN HEAD
2	10	CHAIN CHAIN NEXT
2	11	CHAIN CHAIN HEAD
2	12	CONTROL CHAIN NEXT
3	13	NEXT POSITION
3	14	PRIOR POSITION
3	15	HEAD POSITION
4	16	CHAIN NAME SPECIFIED BY 98 LEVEL

FORMAT

```
                  1   2 3 4 5 6
RDxxxx   .QDD    xxx,xx,x,x,x,x,                       Line 1

                 7    8       9      10      11      12
         ETC     x,RDxxxx,RDxxxx,RDxxxx,RDxxxx,RDxxxx,    Line 2

                  13   14   15
         ETC     xxxx,xxxx,xxxx,                        Line 3

                  16
         ETC     x(30)                                  Line 4
```

TRANSLATOR OUTPUT (see preceding Definition Structure sample)

```
RD7238   .QDD    022,10,0,0,1,0
         ETC     0,RD7236,RD7233,RD4742,RD4097,RD7238,
         ETC     0039,0000,0000,
         ETC     THE-CHAIN
```

.QMD - MASTER DEFINITION

(See Figure 21. Machine Format for Master Definition Entry.)

Line	Item	
1	1	RECORD TYPE
1	2	MASTER CHAIN NEXT
1	3	MASTER CHAIN HEAD
1	4	CHAIN CHAIN NEXT
1	5	NEXT POSITION
1	6	PRIOR POSITION
2	7	CHAIN NAME SPECIFIED BY 98 LEVEL

FORMAT

```
                  1      2        3        4       5     6
RDxxxx    .QMD    xxx,RDxxxx,RDxxxx,RDxxxx,xxxx,xxxx,  Line 1
                   7
          ETC     x(30)                                Line 2
```

TRANSLATOR OUTPUT (see preceding Definition Structure sample)

```
RD7809    .QMD    990,RD4097,RD4481,RD0196,0031,0000,
          ETC     PAGE-TABLE
```

.QFD - FIELD DEFINITION

(See Figure 22. Machine Format for Field Definition Entry.)

Line	Item	
1	1	COMPUTATION MODE AND ARITHMETIC FORM 0 = ALPHANUMERIC BCD FIELD 1 = ALPHABETIC BCD FIELD 2 = NUMERIC BCD FIELD 3 = SIGNED NUMERIC BCD FIELD 4 = SINGLE PRECISION FIXED POINT BINARY FIELD 5 = SINGLE PRECISION FLOATING POINT BINARY FIELD 6 = DOUBLE PRECISION FIXED POINT BINARY FIELD 7 = DOUBLE PRECISION FLOATING POINT BINARY FIELD
1	2	U - UNIQUE FIELD INDICATOR
1	3	FIELD INCREMENT
1	4	FIELD SIZE
1	5	FIRST CHARACTER
1	6	FIELD CHAIN NEXT
2	7	MODIFY CHAIN NEXT
2	8	LOCATION OF WORKING STORAGE
3	9	FIELD NAME

FORMAT

```
                    1 2    3       4      5       6
RDxxxx    .QFD    x,x,xxxxxx,xxxx,FCxxxx,RDxxxx,       Line 1

                    7      8
          ETC     RDxxxx,RDxxxx,                       Line 2

                    9
          ETC     x(30)
```

TRANSLATOR OUTPUT (see preceding Definition Structure sample)

```
RD2345    .QFD    0,0,000015,0024,FC5697,RD2434
          ETC     RD2435,RD5697,
          ETC     QUAD3-FIELD
```

.QCD - CONTROL DEFINITION

(See Figure 23. Machine Format for Control Definition Entry.)

Line	Item	
1	1	R-MATCH CONTROL INDICATOR
1	2	CNTL - CONTROL FIELD TYPE
1	3	CONTROL CHAIN HEAD
1	4	CONTROL CHAIN NEXT
1	5	MODIFY CHAIN HEAD
1	6	MODIFY CHAIN NEXT
2	7	LOCATION OF MATCH-KEY FIELD DEFINITION

FORMAT

```
                    1 2    3       4       5       6
RDxxxx    .QCD      x,x,RDxxxx,RDxxxx,RDxxxx,RDxxxx,     Line 1
                       7
          ETC       RDxxxx                               Line 2
```

TRANSLATOR OUTPUT (see preceding Definition Structure sample))

```
RD2437    .QCD      0,1,RD2438,RD2436,RD2434,RD2434,
          ETC       0
```

6. Operational Characteristics

I-D-S provides the following capabilities:

A controlled, concurrent access to a common I-D-S structured data file which is created by the File System Activity;

A common journal file for the automatic collection of journal records from each of multiple I-D-S activities in execution;

An integrated set of utility routines to enable recovery and/or restart following a condition which requires restoration of the data file.

Concurrent access to a common I-D-S data file is provided through the concept of subfile definition and allocation. A subfile is defined as a set of pages that fall within the total I-D-S data file. This range may be either the complete I-D-S data file or a portion. The File System Activity ($ FILSYS) procedures allow the creation, modification, and deletion of subfiles within an I-D-S file.

At execution time, the I-D-S user specifies the subfiles which must be allocated to his activity. Each subfile requested is given an associated access mode.

I-D-S DATA FILE INITIALIZATION

Prior to the operation of any I-D-S program, the mass storage device must have been initialized with a Page Header record as the first record of each page in the I-D-S data file.

The I-D-S utility program QUTI accomplishes this I-D-S data file initialization.

CREATING AN I-D-S DATA FILE

An I-D-S data file may be created on one or many mass storage devices with different hardware characteristics. It can be permanent, temporary, or a combination of the two. In creating this file, the number and location of pages must be considered.

The various directives necessary for creating an I-D-S data file are described below. Only the I-D-S options are included. Refer to the GE-600 Line GECOS III File System Reference Manual, CPB-1513, for a detailed description of the GECOS III File System.

Creating a Permanent I-D-S Data File

A permanent I-D-S data file is created by using the file system FCREAT/IDS/ directive. The options used with FCREAT/IDS/ are:

BASESIZE/n/ — Base size is required; /n/ defines the maximum size of the complete I-D-S data file; /n/ must be greater than or equal to 1 and less than or equal to 262143. If multiple files are created to form the complete I-D-S data file, the value of /n/ must be identical on all directives.

RNG/r1,r2/ — The page-range is required to define the pages contained in the file; r1 and r2 are the beginning and ending page numbers respectively; r1 must be less than or equal to r2; the values of r1 and r2 must be greater than or equal to 1 and less than or equal to 262143. This range may be either the complete I-D-S data file or a portion.

PAGESIZE/n/ — The page size is optional. If it is omitted, a size of 320-words is assumed. When it is present,/n/ must be greater than or equal to 40 and less than or equal to 640. This allows a different page size in each subfile within the complete I-D-S data file.

When /n/ is present, the actual page size used will be adjusted, if necessary, to reflect a multiple of sector size of the hardware device for this file. (For a DSU200 Magnetic Disc Subsystem, page size will be 40 x 2^n, where n is an integer and $1 \leq n \leq 4$. For a DSU270 or a DSU167 or for an MDS200 Magnetic Drum Subsystem, page size will be 64 x n, where n is an integer and $1 \leq n \leq 10$.)

LINESPERPAGE/n/ Lines per page is optional. If it is omitted or greater than 63, 63 lines per page is assumed. When /n/ is present and less than 63, multiple copies of data pages are created to satisfy all 63 line flags.

INVENTORY/n/ Inventory is optional. If it is omitted, a value of 75 is assumed. When /n/ is present it defines the percentage of page fill, which controls inventory update; /n/ may contain the word "NO" to allow exclusion of inventory pages and processing.

A sample deck setup to create a permanent I-D-S data file follows. It consists of 480 pages in the complete I-D-S data file but it is created as four files, each with 120 pages.

```
1      8          16
$      SNUMB
$      IDENT
$      FILSYS
$      PRIVITY

CRMAST        IDSFOURYQUAD/IDSFOURYQUAD,PASSWORD/DATABASE/,
              SIZE/100/
CCREAT        IDSFOURYQUAD,PASSWORD/DATABASE/
USERID        IDSFOURYQUAD$DATABASE
CPOS          IDSFOURYQUAD
FCREAT/IDS/   QUAD01,BASESIZE/480/,RNG/1,120/,
              PAGESIZE/160/,LINESPERPAGE/32/,
              INVENTORY/25/,SIZE/13/,MODE/RAND/,
              DEVICE/DS3/
FCREAT/IDS/   QUAD02,BASESIZE/480/,RNG/121,240/,
              PAGESIZE/320/,LINESPERPAGE/63/,
              INVENTORY/75/,SIZE/11/,MODE/RAND/,
              DEVICE/ST1/
FCREAT/IDS/   QUAD03,BASESIZE/480/,RNG/241,360/,
              PAGESIZE/320/,LINESPERPAGE/63/,
              INVENTORY/75/,SIZE/11/,MODE/RAND/,
              DEVICE/DS2/
FCREAT/IDS/   QUAD04,BASESIZE/480/,RNG/361,480/,
              PAGESIZE/320/,LINESPERPAGE/63/,
              INVENTORY/75/,SIZE/11/,MODE/RAND/,
              DEVICE/DS2/

$      ENDJOB
***EOF
```

The above control cards will create an I-D-S data file structure as shown in Figure 24.

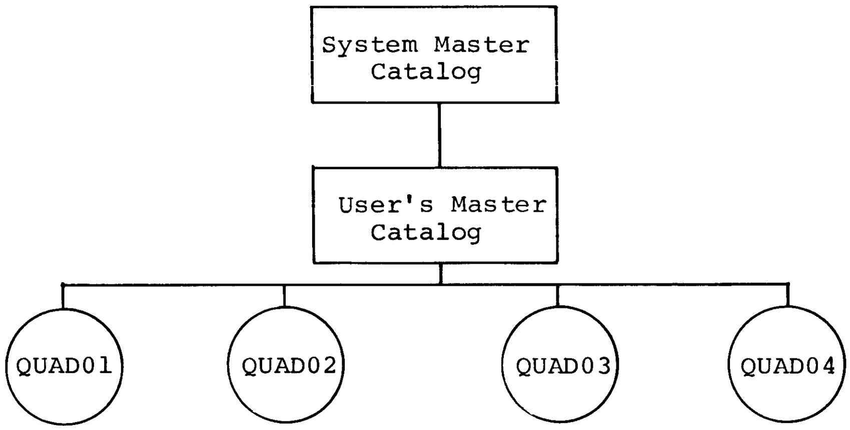


Figure 24. I-D-S Data File Structure

The name in the System Master Catalog is the USERID assigned by the CRMAST directive. This is the name I-D-S will use as the I-D-S data file name.

To have access to this I-D-S data file, the user must supply a $ USERID control card in the execution deck setup. The I-D-S journal records will contain this name, which will be used by the I-D-S utility routine when restart and recovery is required.

Creating a Temporary I-D-S Data File

A temporary I-D-S data file is created by including IDS Create directives with the I-D-S execution activity. These directives are contained in the .Q data file.

The directive format is:

```
1      8         16
IDS    CREATE    attributes
```

The attributes are separated by commas.

The attribute names may be the complete name or the abbreviation.

FILECODE(FC)/fc/ — File code is used to associate the attributes on this directive with the file code on the $ "File" card such as:

```
$    DISC  fc,lud,#random links
```

BASESIZE(BSSZ)/n/ — Base size is required on at least one directive card submitted for an I-D-S execute. If multiple directives are submitted, the value of /n/ must be identical; /n/ defines the maximum size of the complete I-D-S data file; /n/ must be greater than or equal to 1 and less than or equal to 262143.

RANGE(RNG)/r1,r2/ — Page-range is required to define the pages contained in a file; r1 and r2 are the beginning and ending page numbers respectively; r1 must be less than or equal to r2 and the value of r1 and r2 must be greater than or equal to 1 and less than or equal to 262143. This range may be either the complete I-D-S data file or it may be a portion.

PAGESIZE(PGSZ)/n/ — Page size is optional. If it is omitted, a size of 320 words is assumed. When it is present, /n/ must be greater than or equal to 40 and less than or equal to 640. This allows a different page size in each subfile within the complete I-D-S data file.

When /n/ is present the actual page size used will be adjusted, if required, to reflect a multiple of sector size of the hardware device for this file. (For a DSU200 Magnetic Disc Subsystem, page size will be 40 x 2^n, where n is an integer and $1 \leq n \leq 4$. For a DSU270 or a DSU167 or for an MDS200 Magnetic Drum Subsystem, page size will be 64 x n, where n is an integer and $1 \leq n \leq 10$.)

LINESPERPAGE(LPP)/n/

Lines per page is optional. If it is omitted or greater than 63, 63 lines per page is assumed. When /n/ is present and less than 63, multiple copies of data pages are created to satisfy all 63 line flags.

INVENTORY(INV)/n/

Inventory is optional. If it is omitted, a value of 75 is assumed. When /n/ is present it defines the percentage of page fill, which controls inventory update; /n/ may contain the word "NO" to allow exclusion of inventory pages and processing.

The deck setup below will create a temporary I-D-S data file to be used by the I-D-S activity.

```
1      8          16
$      SNUMB
$      IDENT
$      OBJECT
         .
         .
$      DKEND
$      EXECUTE    DUMP
$      LIMITS
$      DISC       A1,A1S,13R
$      DISC       A2,A2S,11R
$      DISC       A3,A3S,11R
$      DISC       A4,A4S,11R
$      DATA       .Q
IDS    CREATE     FC/A1/,BSSZ/480/,RNG/1,120/,PGSZ/160/,
                  LPP/32/,INV/25/
IDS    CREATE     FC/A2/,BSSZ/480/,RNG/121,240/,PGSZ/320/,
                  LPP/63/,INV/75/
IDS    CREATE     FC/A3/,RNG/241,360/
IDS    CREATE     FC/A4/,RNG/361,480/
$      ENDJOB
***EOF
```

Mixing Temporary and Permanent Files

An I-D-S data file is subordinate to the GECOS-III file system. The I-D-S data file may be created on one or many mass storage devices with different hardware characteristics. This facility allows selected I-D-S record types to be given page-ranges, which may then be directed to a specific hardware device when the file is created. I-D-S utility routines provide for selective file dump and reload. It is possible that an application may require that pages residing on one type of hardware be dumped and then reloaded on another type of hardware.

Two hypothetical cases where the user may want to mix permanent and temporary files follow:

A user may want to establish a page range for records that are only used weekly or monthly. For this application, the page range would not be created as a permanent file. Instead the page range would be created as a temporary file, the data stored, and the file dumped to tape.

When the records are to be used, the temporary file is established, the file is reloaded from the dumped tape, and the program is executed using this file in conjunction with the permanent file. Again, the file is dumped to tape and saved for the next weekly or monthly run.

Another example of mixed permanent and temporary files is using a temporary file for the work area of execute activities. In this usage, a permanent file would not be required for the delete process, since this temporary area would be purged at the end of the activity.

ACCESSING AN I-D-S FILE

Subfile Allocation

Each subfile requested must have been created previously as an I-D-S data file. A $ PRMFL control card is required for each subfile. Refer to CPB-1518 for a complete discussion of options used. The I-D-S options are discussed below:

```
1      8       16
$      PRMFL   fc,Permit,Mode,File String
$      PRMFL   fc,/LUD,Permit,Mode,File String
```

PERMIT is an option describing the I-D-S usage. Multiple access modes may be used. If used, they are separated by slashes (/). The valid options are:

WRITE - The user requests the subfile for updating records.

READ - The user requests the subfile for retrieving records.

RECOVERY - The user requests access to an aborted subfile to reestablish the integrity of the subfile.

Examples:

```
1      8       16
$      PRMFL   A1,READ,R,IDSFOURYQUAD/QUAD01

$      PRMFL   A2,READ/WRITE,R,IDSFOURYQUAD/QUAD02

$      PRMFL   A3,RECOVERY/READ/WRITE,R,IDSFOURYQUAD/QUAD03
```

Sample deck set up of LUD Option, used with the "CLOSE WITH LOCK" statement for dynamic release of I-D-S file.

```
1      8       16
$      IDENT
$      USERID
         .
         .
         .            (First Activity)
$      PRMFL   A1/D1S,R/W,R,FILE STRING
$      PRMFL   A2/D2S,R/W,R,FILE STRING
         .
         .
         .            (Last Activity)
$      FILE    TF,D1R,1R
$      FILE    TG,D2R,1R
```

Table A shows the action taken when the LUD option is used.

DISPOSITION CODE	PERMANENT FILE	TEMPORARY FILE
R	File is made unavailable to the run unit. File space is available to the system for allocation.	File is made unavailable to the run unit. File is available for allocation to other jobs.
S	File is made unavailable to the run unit. File space is held for allocation to other activities in this job.	File is made unavailable to the run unit. File is NOT available for allocation to other jobs. File is held for allocation to other activities in this job.

Table A.

An I-D-S activity which includes a request for subfiles is not allocated until all requested subfiles are allocated. The subfile allocation criteria are shown in Figure 25.

SUBFILE ALLOCATION CONDITION	ACCESS REQUESTED											
	READ				WRITE				RECOVERY			
FILE IN ABORT STATE	X				X				X			
FILE BUSY WRITE (UPDATE)		X				X				X		
FILE BUSY READ (RETRIEVE)			X				X				X	
FILE NOT BUSY				X				X				X
ACTION												
DENY ALLOCATION		X				X	X					
DELETE ALLOCATION REASON CODE	15				15					16	16	16
PERMIT ALLOCATION			X	X				X	X			

Figure 25. I-D-S Data File Allocation

Since a READ access mode does not alter the contents of a subfile, several I-D-S activities can share a subfile in READ mode. If a subfile is allocated to an activity in the READ mode, it can also be allocated to any other I-D-S activity which wishes to use it in the READ mode. Allocation of the subfile would be denied, however, to any activity requesting WRITE usage for a subfile which is already allocated for READ usage.

While there can be concurrent users of a subfile in READ mode, there can be only one active user for a subfile in the WRITE access mode. All other allocation requests for the subfile would be denied until the activity which is doing the UPDATE has terminated.

Subfiles allocated in the WRITE access mode are marked in ABORT status if the activity aborts. A subfile in ABORT status will be allocated by requesting RECOVERY access mode in addition to READ and WRITE.

The individual responsible for maintaining the I-D-S data file must prepare the necessary input for a RECOVERY run. The utility routines which aid in this preparation are discussed later.

The abort indicator is turned off for an aborted subfile after a successful RECOVERY run is made on that subfile. It is then available for normal allocation.

Subfile Deallocation

I-D-S data files are deallocated at activity termination. Figure 26 shows the deallocation activity and the action taken.

ACTIVITY TERMINATION CONDITION	FILE BUSY ACCESS MODE					
	READ		WRITE		RECOVERY	
NORMAL	X		X		X	
ABNORMAL		X		X		X
ACTION						
SET FILE ABORT ON				X		X
SET FILE NORMAL	X	X	X		X	
SET FILE ABORT OFF					X	

Figure 26. I-D-S Data File Deallocation

I-D-S JOURNAL FILE

A journal file is a recording of all I-D-S data file page transactions. Journal information is collected on the accounting file tape from each of multiple I-D-S activities in execution, thus providing a single source file that is used to reestablish a data file to some previously known status in the event that the file should lose its integrity. A journal tape is labeled and is a single file. Multiple reel output may be produced depending on the journalization required.

When an end of reel is reached or an activity with write permission aborts, a reel swap or unit switch occurs. Two operator inputs permit the accounting file to be closed for I-D-S purposes:

IDSEJ	Close the accounting file with an EOF trailer label when all I-D-S jobs known to the system are complete.
IDSER	Close the accounting file with an EOR trailer label at the time of the request.

I-D-S Journal File Configuration

The I-D-S Journal file is configured on the system accounting file tape at system startup time by adding the I-D-S options to the Startup $ ACCOUNT control card. Refer to the GE-600 Line GECOS III Startup Software Maintenance Document, CPB-1489.

The I-D-S options are:

IDS	This option indicates that the I-D-S journal records are to be included on the system accounting file as record type 13(8).
BUFSIZ/n	This option sets the size of the collecting buffers for I-D-S journal records and the accounting records. If omitted, then /n is assumed to be 320. The value /n must be set to at least 12 words larger than the maximum page size that may be placed on the journal file. If a journal record is encountered which is greater in size than the collecting buffer, the slave program will be terminated with a D2 abort code.
RETENTION/n	This option allows the retention period in days required for label checking/writing to be established for the I-D-S journal file.

Journal Record Format

Journal records are produced as record type 13(8) on the system Error and Accounting file which must be configured at system startup time and must be assigned to magnetic tape. Override options are discussed later.

With the exception of block size, records are written in standard system format as described in the GE-600 Line File and Record Control Reference Manual, CPB-1003. The block size is as large as the buffer size defined on the startup $ ACCOUNT control card.

The various formats for record type 13(8) that can be recorded on the journal tape appear below followed by definitions of terms common to all types.

Slave Begin Sync, Record Type 03. This record is written at the beginning of each I-D-S slave activity.

Word	Contents
1	Record control word for journalizing
2	Checksum
3	SNUMB
4	Start date (MMDDYY)
5	Start time (HH.TTT)
6	Indicators (bits 0-11) Activity number (bits 27-35)
7	Not used
8, 9	I-D-S data file name

Subroutine .QOPEN generates this record and stores it in the slave program prefix as follows. (See also "I-D-S Data Pages" in Chapter 7 for special conditions that apply when using disc sort.)

Location in Prefix (decimal)	Word Contents		
54	000010 (Size)		000013 (Type)
55	Checksum		
56	SNUMB		
57	MMDDYY		
58	HH.TTT		
59	030		Activity #
60	0		
61-62	I-D-S data file name		

Page Image Record, Record Types 05 and 06. There are two types of Page Image records (BEFORE and AFTER) written to the journal tape. The indicator word defines the type. A BEFORE page image is written before a page is modified. An AFTER page image is written after the modification.

Word	Contents
1	Record control word for journalizing
2	Checksum
3	Job number
4	Start date (MMDDYY)
5	Start time (HH.TTT)
6	Indicators (bits 0-11) Activity number (bits 27-35)
7	Lines per page (bits 0-17) Sequence number (bits 18-35)
8-9	I-D-S data file name
10-n	Activity page image

Slave End Sync, Record Type 04. This record is written when an I-D-S slave program terminates. The termination code is stored in the record.

Word	Contents
1	Record control word for journalizing
2	Checksum
3	Job number
4	Start date (MMDDYY)
5	Start time (HH.TTT)
6	Indicators (bits 0-11) Activity number (bits 27-35)
7	Termination code
8-9	I-D-S data file name

Journal Record, Record Type 09. This record is written when subroutine .QSTB is used to gather type B subroutine execution information. (See QUTR Program writeup in Chapter 8.)

Word	Contents
1	Record control word for journalizing
2	Checksum
3	Job number
4	Start date (MMDDYY)
5	Start time (HH.TTT)
6	Indicators (bits 0-11) Activity number (bits 27-35)
7	Alter number of call to subroutine
8-9	I-D-S data file name
10	Control word (see following explanation)
11	Number of reads (for any given control word (word 10))
12	Number of writes (for any given control word (word 10))

The control word format (word 10) is as follows:

0 5	6 17	18 23	24 35
Code	Type - 1	MBZ	Type - 2

where:

Code is one of the following function values:

1 - Store record type
2 - Retrieve record type
3 - Retrieve current record type
4 - Retrieve direct
5 - Retrieve each
6 - Retrieve next of chain
7 - Retrieve prior of chain
8 - Retrieve master of chain
9 - Head of chain
10 - Modify record type
11 - Delete record type
12 - Debug

Type - 1 is the record type for the preceding function or the record type of the master of a chain.

Type - 2 is the record type of a detail of a chain.

Definition of Terms

Checksum	The checksum of all words (other than the checksum word) in the record.
Date	A 6-character field indicating month, day, and year the record was written. For slave End Sync records, it is the date the corresponding Slave Begin Sync record was written.
Time	Time the activity was started expressed in hours, decimal point, and thousandths of an hour in BCD format (HH.TTT). For Slave End Sync or Page Image records, it is the time in the corresponding Slave Begin Sync record.
Indicators	A 1-word indicator which defines the record type and contains the activity number.
Record Type	A 1-character BCD field that appears in bits 6-11 of the indicator word. The record type indicators are shown below:

TYPE 3	Slave Begin Sync	(SLVBGN)
TYPE 4	Slave End Sync	(SLVEND)
TYPE 5	Before Page Image	(BEFORE)
TYPE 6	After Page Image	(AFTER)
TYPE 9	Statistics	

Lines per Page	The lines per page for the Before/After Page Image.
Termination Code	A 2-character code in bits 27-35 of the Slave End Sync record. Termination codes are:

00	Normal activity termination
00	Normal job termination
cc	Abnormal termination; cc is a 2-character alphanumeric abort code.

I-D-S Date File-Name	A 12-character name, left justified. This name is taken from the $ USERID card.

Job Number	A 5-character SNUMB for the job, left justified and followed by an ignore character.
Activity Number	A 9-bit binary job activity number.
Sequence Number	A binary sequence number carried in the Page Image records. BEFORE records are sequenced by 1 in ascending order starting with 1. AFTER records are sequenced in descending order starting with all binary 1's in bits 18-35.
Record control word for journalizing	A control word that contains the number of words in the record in bits 0-17 and defines it as record type 13(8), right-justified, in bits 18-35.

Closing Journal Files

The system-configured journal tape collects the journal data as one long file. From an operational point of view, it is necessary to periodically "close" one journal file and start another. This closing, followed by an opportunity to dismount and replace the journal tape, is done automatically when there is a master mode abort.

The operator may periodically request that a journal file be closed and another file started. He does this by requesting control and using the IDSEJ typein. The system response to this input is shown in the following table.

CONDITION	ACTION
An I-D-S activity is in execution.	IDSEJ DELAY message is typed out. The I-D-S journal file will be closed when there is no I-D-S activity in execution.
No I-D-S activity is in execution.	An end-of-file is recorded on the journal tape and a dismount message is issued.

Journal Override

Journal records are automatically written to the system-configured Error and Accounting tape; however, there are two activity override options available. Option 1 permits the user to request his own tape; option 2 suppresses all journalization.

The control card format for option 1 is:

```
1      8       16
$      TAPE    JX,X1D,,,,IDS-JOURNAL
```

If a tape file JX is assigned for an activity, all journal record types -- the Slave Begin Sync, Slave End Sync, and all BEFORE and AFTER records and all statistics records -- are written to this file.

The control card format for option 2 is:

```
1      8       16
$      EXECUTE DEBUG
```

DEBUG in the variable field of the $ EXECUTE control card causes bit 11 of the Program Switch Word to be set ON which prevents any journal records from being generated.

Examples:

1. The Slave Begin Sync, Slave End Sync, BEFORE and AFTER records are written to the user-supplied file JX.

```
1      8       16
$      IDENT
$      OBJECT
         .
         .
         .
$      DKEND
$      EXECUTE Options
$      TAPE    JX,X1D,,,,IDS-JOURNAL
```

2. No journalization takes place.

```
1      8          16
$      IDENT
$      OBJECT
        .
        .
        .
$      DKEND
$      EXECUTE    DEBUG
        .
        .
        .
```

Journal File Map

A map of all Sync records contained on the I-D-S Journal file may be produced by executing the .QUTJ I-D-S utility routine (1) when a journal file has been made available after the abnormal termination of an I-D-S activity or, (2) the operator requests an end-of-file condition.

A sample journal file map follows.

```
QJNL  01  09-27-68   11,341          IDS JOURNAL TAPE REPORT
           IDS UTILITY ROUTINE - ,QUTJ - VERSION      080168,

    9  SLVBGN   1-QUTI   09-27-68  11,199  030        0  IDSFOURYQUAD
   10  SLVEND   1-QUTI   09-27-68  11,199  040       00  IDSFOURYQUAD
   15  SLVBGN   1-TST03  09-27-68  11,210  030        0  IDSFOURYQUAD
   38  SLVBGN   1-TST3C  09-27-68  11,211  030        0  IDSFOURYQUAD
  117  SLVEND   1-TST03  09-27-68  11,210  040       00  IDSFOURYQUAD
  133  SLVEND   1-TST3C  09-27-68  11,211  040       00  IDSFOURYQUAD
  138  SLVBGN   1-QUTDL  09-27-68  11,222  030        0  IDSFOURYQUAD
  140  SLVEND   1-QUTDL  09-27-68  11,222  040       00  IDSFOURYQUAD
  142  SLVBGN   2-QUTDL  09-27-68  11,227  030        0  IDSFOURYQUAD
  143  SLVEND   2-QUTDL  09-27-68  11,227  040       00  IDSFOURYQUAD
  149  SLVBGN   1-TST4A  09-27-68  11,237  030        0  IDSFOURYQUAD
  150  SLVBGN   1-TST4B  09-27-68  11,238  030        0  IDSFOURYQUAD
  151  SLVEND   1-TST4A  09-27-68  11,237  040       00  IDSFOURYQUAD
  153  SLVEND   1-TST4B  09-27-68  11,238  040       00  IDSFOURYQUAD
  155  SLVBGN   1-TST4C  09-27-68  11,240  030        0  IDSFOURYQUAD
  156  SLVEND   1-TST4C  09-27-68  11,240  040       00  IDSFOURYQUAD
  161  SLVBGN   1-QUTD   09-27-68  11,247  030        0  IDSFOURYQUAD
  162  SLVEND   1-QUTD   09-27-68  11,247  040       00  IDSFOURYQUAD
```

RECOVERING AN I-D-S DATA FILE

All I-D-S slave programs interface with GECOS-III through the MME GEIDSE incorporated in the I-D-S object-time subroutines. The MME enables the subroutines to record page images on a system configured journal tape. BEFORE page images are written to the journal tape prior to the modification of a page; AFTER page images are written to the journal tape following modification of the page. When recovery of the data file is desired, the journal tapes containing the required pages are processed as illustrated in Figure 27A. Figure 27B illustrates an alternate method.

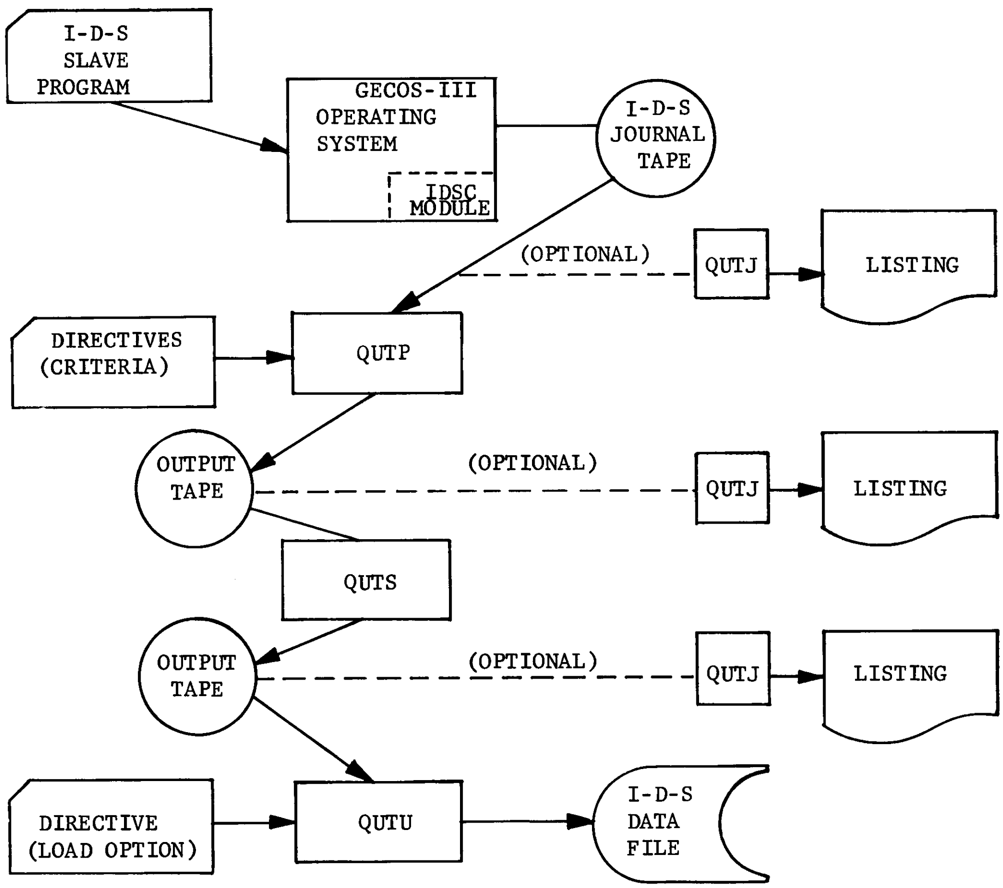


Figure 27A. Operational Sequence to Re-establish an I-D-S Data File

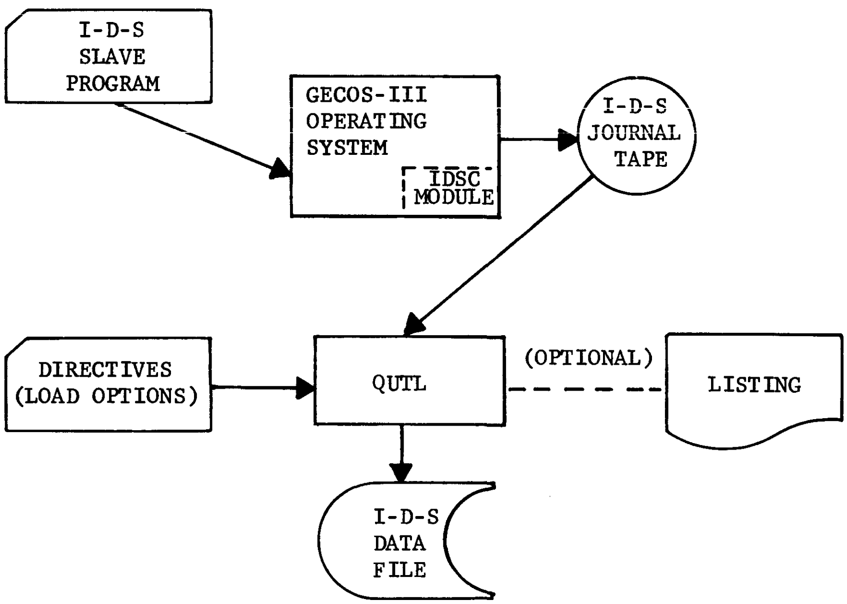


Figure 27B. Alternate Operation to Re-establish an I-D-S Data File

The individuals responsible for maintaining the data base establish the selection criteria for obtaining the appropriate pages from the journal tape. This is done using the information from the Journal Tape Map or from a complete journal dump created by the QUTJ utility routine. The QUTP utility routine selects pages from the journal tape. The QUTS utility routine then sorts the selected page image records and purges multiple page images having the same page number. The sorted output consists of the first BEFORE or the last AFTER image for a given page number as required for the data file reload. The QUTU utility routine reloads the output to the appropriate portions of the data file.

Since rollback does not reestablish the data file to a previous condition, the MME GECHEK and MME GEROLL should not be used by an I-D-S program.

I-D-S EXECUTION REPORT

I-D-S appends information about the data base to the execution report. This information includes (1) the attributes of the data base (2) total input/output performed on the data base, and (3) input/output performed on the data base as a function of each I-D-S subroutine. Formats of the three types of information are shown in the following examples and are explained by the notes corresponding to the circled callouts.

Example 1: Data Base Attributes

1 Files Allocated -- the number of permanent and/or temporary IDS files allocated to the activity

2 Range -- the smallest and largest page number present in the files

3 Basesize -- the value to be used in the randomize routine

4 Buffers -- the number of page buffers present

An entry appears under each of the following heads for each file or subfile:

5 Filecode -- the file code referenced by the program

6 Range -- the range for this file or subfile

(7) Pagesize -- the page size for this file or subfile

(8) Pages/Page -- the number of pages per page for the file or subfile

(9) Lines/Page -- the number of lines per page for the file or subfile

(10) Links Aloc -- the number of links allocated to the file or subfile

(11) Links Nec -- the number of links necessary to contain the pages defined for the file or subfile

(12) Access Mode -- the mode in which the file or subfile is being accessed

(13) Inventory -- the percentage value at which inventory will be updated on the file or subfile

```
      (1)               (2)              (3)            (4)
1 FILES ALLOCATED,  RANGE   1 - 100   BASESIZE    100 BUFFERS  29

FILECODE  RANGE  PAGESIZE  PAGES/PAGE LINES/PAGE  LINKS ALOC  LINKS NEC ACCESS MODE INVENTORY

  A1       1- 100   320       1         63           20          9       WRITE        75

  (5)       (6)     (7)      (8)        (9)          (10)        (11)     (12)         (13)
```

Example 2: Total I/O Performed on Data Base

The following are shown for each file or subfile:

1. File Code -- the file code referenced by the program
2. # of Reads -- the total number of reads that occurred on the file or subfile
3. # of Writes -- the total number of writes that occurred on the file or subfile
4. Inventory Reads -- the number of inventory reads that occurred on the file or subfile
5. Inventory Writes -- the number of inventory writes that occurred on the file or subfile

I-D-S UTILIZATION REPORT

FILE CODE	# OF READS	# OF WRITERS	INVENTORY READS	INVENTORY WRITES
TF	258	2883	1	1
1	2	3	4	5

Example 3: I/O Performed on Data Base as a Function of Each I-D-S Subroutine

This report is produced by the I-D-S close subroutine. Counts are accumulated for each primary entry subroutine -- that is, each subroutine called by the object program. These are known as type A (.QSTA) subroutine execution statistics. (An additional, more detailed (type B) report can also be produced as a separate output at the user's option. For this report the .QSTB subroutine is used to accumulate the statistics on the journal file, and the QUTR program produces the report. See the QUTR writeup in Chapter 8 for details.)

The type A report contains the following information:

1. Primary entry subroutine name
2. Total number of times subroutine was called
3. Total number of reads for execution of the subroutine
4. Total number of writes for execution of the subroutine

SUBROUTINE STATISTICS

NAME	NO. TIMES CALLED	NO. READS	NO. WRITES
.QSTOR	18	6	10
.QGET	18	0	2
.QCHN	88	0	6
.QMDFY	18	0	24
1	2	3	4

7. Memory Management

ASSIGNMENT OF I-D-S BUFFERS AND WORK AREAS

The I-D-S subroutines require data page buffer areas and working areas. The user defines the size of these areas by employing one of the two following procedures.

With a $ USE Card

A Labeled Common area (.QAREA) may be specified by the GELOAD control card shown below:

```
1      8       16
$      USE     .QMAX/1/,.QAREA/n/,.QMIN/1/
```

The $ USE control card must be inserted before the $ EXECUTE card in the object deck so that GELOAD will encounter it prior to loading the I-D-S subroutines from the library. Refer to the GE-600 Line General Loader Reference Manual, CPB-1008.

The value supplied for /n/ must be large enough to contain the working area plus at least three page buffers. The following formula may be used to determine the total space required.

(NF*10) + 10 + ((MP + 21)*NB) + NO +(I + 3)

where NF is the number of files allocated
MP is maximum page size allocated in words
*NB is number of page buffers
I is maximum sector size for files containing inventory. (For DSU200, I = 40; for all other mass storage devices I = 64.)
*NO is number of page buffers which overlay .QOPEN.

*The total number of buffers (TB) must be at least three. TB = NB+NO, where NO is determined by the following formula:

NO = 816/(MP+20)

When a sort is included as part of an I-D-S activity, a $ USE card must be used to constrain the work area of one of the systems. If this is not done, both systems will compete for the area not assigned to other program segments.

A sample deck setup for an I-D-S sort using disc sort and temporary I-D-S files follows. With this setup, the sort work area will be the core storage remaining from the $ LIMITS card after subtracting the user program size and the I-D-S page buffer size (.QAREA).

```
1      8       16
$      IDENT
$      USE     .QMAX/1/,.QAREA/5000/,.QMIN/1/
$      OBJECT  USERPROGRAM
$      EXECUTE
$      LIMITS  10, 32K
$      DISC    TF,D1S,10R
$      DISC    S1,X1R,5R
$      DATA    .Q
IDS    CREATE  FC/TF/,BSSZ/100/,RNG/1,100/
$      ENDJOB
***EOF
```

Without a $ USE Card

When the $ USE control card procedure is not used, the .QOPEN I-D-S subroutine attempts to use the area in memory not assigned to other program segments. The size of this available area is inserted in word 37(8) of the slave program prefix by GELOAD during the loading process. As in the procedure above, the available area is divided into a work area and some number of buffers, depending on the size of the area. A minimum of three buffers must be established or the slave program will be terminated. The .QOPEN subroutine modifies the content of word 37(8) to reflect the usage of this area.

When the file is opened, the size of .QAREA is determined and then used in the following manner (see Figure 28):

1. Slave I-D-S Control Table - this table consists of 10 control words plus 10 words for each I-D-S subfile (temporary or permanent) assigned to the activity.

2. Inventory Record Buffer - this area is equal to three words more than the largest inventory sector allocated.

3. Page Buffer Activity Table - this table contains one word for each page buffer.

4. Data Page Buffers - these buffers are equal to the page size of the largest page allocated plus 20 decimal words.

5. The first inventory buffer exists as defined in Figure 28. The other inventory buffers and their headers are generated by .QOPEN and overlay the code in .QOPEN that may be executed only once. As many buffers exist as will fit in the overlay area.

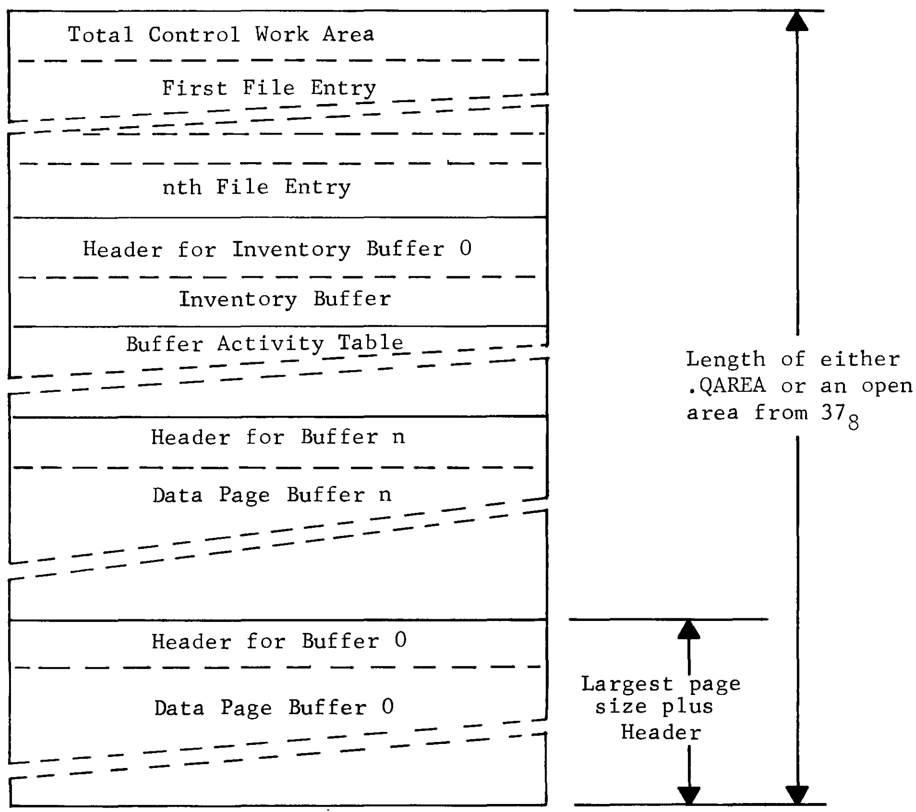


Figure 28. Labeled Common .QAREA

SLAVE I-D-S CONTROL TABLE

Figure 29 shows a Slave I-D-S Control Table used by the I-D-S subroutines to honor the attributes of an I-D-S data file. Each subfile may be different, such as page size and percent of page fill for inventory. The I-D-S subroutines use a common GEFRC file control block to do all I-D-S data page and inventory page I/O on the mass storage. To accomplish this, the file control block control information is kept in the SICT Table for each unique file. It is then placed into the file control block when an I/O request for a page is needed. The total length of the table is dependent on the number of files allocated.

Word	Bits 0–17	Bits 18–35	
0	Pointer to Current Entry	MBZ	Total Control Entry
1	Maximum Page Size	Base Size	
2	Lowest Page Number	Highest Page Number	
3	Maximum Inventory Sector	Page Buffer Size	
4–8	MBZ	MBZ	
9	MBZ	Count of Entries	
0	RANGE R1	RANGE R2	File Control
1	Inventory Write Counter	Page Size	
2	Pages/Page No.	Lines Per Page	
3	RBA of Current Inventory	Inventory Percent Fill	
4	Inventory Read Counter	RBA Current Page	
5	Sectors/Page	Sector Size	
6	Gross Write Counter	Gross Read Counter	
7	Base RBA of Inventory	FILCB+0 [18-35]	
8	FILCB-5 [18-35]	FILCB-1 [18-35]	
9	Access Mode	FILCB-4 [24-35]	

Figure 29. Slave I-D-S Control Table

The description of the Slave I-D-S Control Table (SICT) follows.

Total Control Entry

Word 0 bits 0-17	Pointer to current entry - the address of the SICT table entry which contains the relative block address of the page number last requested via the I-D-S mapping subroutine.
18-35	Must be zero.
Word 1 bits 0-17	Maximum page size - the value in words of the largest page size allocated.
18-35	Base size - the total number of pages in the I-D-S data file.
Word 2 bits 0-17	Lowest page number - the lowest page number allocated.
18-35	Highest page number - the highest page number allocated. Must be less than or equal to the value in the base size.
Word 3 bits 0-17	Maximum inventory sector - the size in words of the largest inventory sector allocated.
18-35	Page buffer size - the maximum page size plus 20 decimal to include the page header area.
Word 4 through Word 8	Must be zero.
Word 9 bits 0-17	Must be zero.
18-35	Count of entries - the number of subfiles allocated to form this I-D-S data file.

Individual File Entries

Word 0
bits

0-17	RANGE R1 - the lowest page number assigned to the subfile.
18-35	RANGE R2 - the highest page number assigned to the subfile. R2 must be greater than or equal to R1.

Word 1
bits

0-17	Inventory write counter - a counter for the number of times an inventory record has been written to the file.
18-35	Page size - the page size in words defined for the file. The page size must be greater than or equal to 40 and less than or equal to 640.

Word 2
bits

0-17	Pages/page No. - the number of pages as developed by dividing 63 by the number of lines per page.
18-35	Lines per page - the number of lines that may be used in any page or pagette.

Word 3
bits

0-17	RBA of current inventory - the Relative Block Address (RBA) of the current inventory record. Inventory records are physically stored beginning in the first sector, following the last data page of the file.
18-35	Inventory percent fill - the number of characters that may be placed in a page of this file before the inventory adjustment routines are called. If the value is negative (bit 18=1), there are no inventory records, therefore, there is no inventory processing.

Word 4
bits

0-17	Inventory read counter - a counter for the number of times an inventory record has been read from this file.
18-35	RBA current page - the Relative Block Address of the last page number accessed in this subfile.

Word 5 bits 0-17	Sectors/Page - the number of sectors within a page. The size is calculated by dividing sector size of the mass storage device into the page size.
18-35	Sector size - the sector size of the hardware device of this file.
Word 6 bits 0-17	Gross write counter - a counter for the number of times data pages or inventory records have been written to the file.
18-35	Gross read counter - counter for the number of times data pages or inventory records have been read from the file.
Word 7 bits 0-17	Base RBA of inventory - relative block address of the beginning of inventory for the file.
18-35	FILCB+0 - contents of the GEFRC file control block.
Word 8 bits 0-17	FILCB-5 (18-35) - contents of the GEFRC file control block.
18-35	FILCB-1 (18-35) - contents of the GEFRC file control block.
Word 9 bits 0-17	Access mode - the access permissions requested from the $ PRMFL card for this file or the permissions granted for the $ DISC or the $ MASS control card for this file. Bits 0 READ (RETRIEVE) 1 WRITE (UPDATE) 2 Not used by I-D-S 3 RECOVERY 4-17 Not used by I-D-S
18-35	FILCB-4 (24-35) - file code for the file.

I-D-S INVENTORY RECORDS

To minimize mass storage seek and transfer time, a number of inventory records are maintained in numerous buffers in memory.

Buffer Format

The I-D-S inventory record buffer format is shown in Figure 30.

	Word	Bits 0–11	12–17	18		35
Inventory Header Work Area	0	Pointer to Next Buffer		Buffer Number		
	1	MBZ	Beginning Reference Code			
	2	MBZ	Ending Reference Code			
Inventory Record Area Through	3	Beginning Page No.		A	Record Type	Begin Line No.
	n	MBZ	Ending Reference Code			
		Space Available				

Figure 30. Inventory Record Buffer

A description of the Inventory Record buffer follows:

INVENTORY RECORD WORD AREA

Word 0
bits

0-17 Address of the next Inventory buffer header (this list is circular).

18-35 The number of this buffer (starting at 0).

Word 1
bits

0-11 Must be zero.

12-35 The beginning reference code of the Inventory record in the buffer.

Bits 12 - 29 Page number
30 - 35 Line number

Word 2
bits

0-11	Must be zero.
12-35	The ending reference code of the Inventory record in the buffer.

Bits 12 - 29 Page number
30 - 35 Line number

Word 3
through
word n Inventory record area.

Buffer Strategy for Inventory Buffer

If the inventory is needed for a page and the inventory record is not in memory, it is read into the inventory buffer defined as empty; and words 1 and 2 of the buffer header are updated.

The next inventory buffer as defined by word 0 of the header is then established as the empty buffer. Its contents are written back to the data file if the contents have been altered.

Record Description

Inventory records are physically stored at the end of the file for the page-range specified. They are record type 1002(10). The Inventory record size is equal to the sector size of the device on which it is stored. Thus the number of pages covered by one Inventory record is variable; it is equal to 3 x (sector size-2). On a DSU204 one link holds inventory for 10,944 pages; on a DSU270 or a DSU167 one link holds inventory for 11,160 pages.

The initial inventory of space available will be the page size (in characters) less the space occupied by the Page Header record (22 characters).

The Inventory record format is shown in Figure 31.

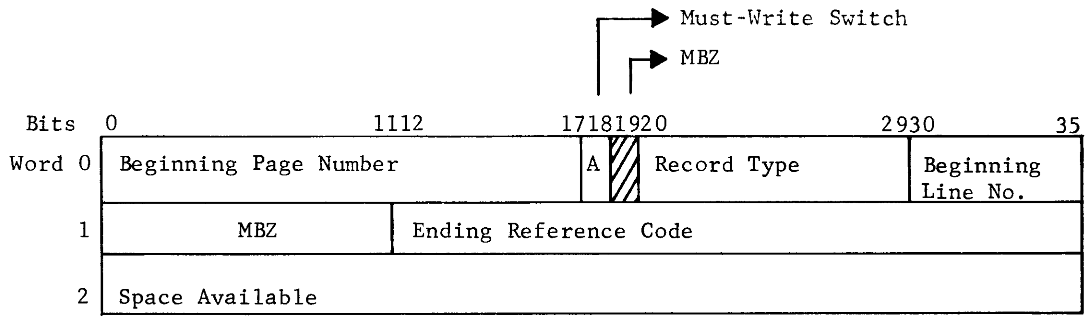


Figure 31. Inventory Record

The bit configuration for an Inventory record follows:

Word 0
bits

0-17	Beginning page number that is contained in the Inventory record.
18	Must-Write switch - an indicator used by I-D-S subroutines to determine if this record has been modified since retrieval.
19	Must be zero.
20-29	Record type - a value of 1002 (10) assigned to each Inventory record.
30-35	Beginning line number of the beginning page for this Inventory record.

Word 1
bits

0-11	Must be zero.
12-35	Ending Reference Code that is contained in this Inventory record. Bits 12 - 29 Page number Bits 30 - 35 Line number

Word 2
bits

0-11	Space available in characters for the Reference Code contained in word 0, bits 0-17 and 30-35.
12-23	Space available for the next ascending page (this may be a pagette).
24-35	Space available for the next page.
	Word 2 is repeated for consecutive pages until bits 24-35 of word n is the space, in characters, available in the page defined by the ending reference code (bits 12-35 of word 1).

I-D-S DATA PAGES

To minimize mass storage seek and transfer time, a number of data pages are maintained in numerous buffers in memory. The number of buffers depends on the amount of space available in .QAREA after loading the program.

The greater the number of data pages kept in memory, the greater the possibility that the one needed next will already be there. To further improve the possibility of finding the page desired in memory, the I-D-S subroutines keep track of page utilization (record activity) and hold the most recently active pages in memory. Pages infrequently accessed are retired from memory as others are called in. The I-D-S subroutines note which pages have been modified and only the modified pages are written back to mass storage.

Buffer Format

The I-D-S page buffer format is shown in Figure 32.

The description of the Data Page Buffer follows.

PAGE HEADER WORK AREA

Word 0 bits 0-17	Pointer to the next buffer. This will be zero in the last buffer.
18-35	Buffer number - the number of the buffer beginning with zero.
Word 1 bits 0-11	Must be zero.
12-35	The beginning reference code of the I-D-S data page in the buffer. Bits 12 - 29 Page number 30 - 35 Line number
Word 2 bits 0-11	Must be zero.
12-35	The ending reference code of the I-D-S data page in the buffer. Bits 12 - 29 Page number 30 - 35 Line number
Word 3 bits 0-23	Must be zero.
24-35	Character space available in the I-D-S data page when read from the mass storage device.

Bits 0 5 6 11 12 17 18 19 20 23 24 26 27 29 30 35

Area	Word	Contents
Page Header Work Area	0	Pointer to Next Buffer | Buffer Number
	1	MBZ | Beginning Reference Code of I-D-S Data Page
	2	MBZ | Ending Reference Code of I-D-S Data Page
	3	MBZ | Character Space Available
	4	Available Line Flag Indicator
	5–7	MBZ
Journal Tape Header	8–10	Work Area for GEFRC I/O Control
	11	Size | Accounting Record Type
	12	Checksum
	13	Job Number
	14	Start Date
	15	Start Time
	16	MBZ | Journal Record Type | MBZ | Activity Number
	17	Lines/pg. for B/A Page Image | Sequence Number
	18–19	I-D-S Data File Name
I-D-S Data Page	20 Through Word n	Page Number | A | B | Record Type
		CALC Chain NEXT | Space Available
		Available Line Number Flags
		MBZ

Figure 32. Data Page Buffer

Word 4
bits
0-35 Available line flag indicator of the I-D-S data page when read from the mass storage device.

= 0 line flags available
≠ 0 line flags not available

Word 5
through
Word 7 Must be zero.

JOURNAL TAPE CONTROL AREA

Word 8
through
Word 10 Work area for GEFRC I/O control - contains an I/O control word, Block Serial number and Record Control word.

Word 11
bits
0-17 Contains the number of words in the record when written to the journal tape.

18-35 Contains the value 13(8) to define the accounting record type.

Word 12
bits
0-35 Checksum - all words (other than the checksum word) in the record.

Word 13
bits
0-35 Job number - the five character SNUMB for the job, left justified and followed by an ignore character.

Word 14
bits
0-35 Start date - month, day, year the activity started, in BCD format (MMDDYY).

Word 15
bits
0-35 Start time - time the activity started expressed in hours, decimal point, and thousandths of an hour in BCD format (HH.TTT).

Word 16 bits 0-5	Must be zero.
6-11	A 1-character BCD field that defines the journal record type. Type 5 Before Page Image (BEFORE) Type 6 After Page Image (AFTER)
12-26	Must be zero.
27-35	Activity number - a 9-bit binary job activity number.
Word 17 bits 0-17	Lines per page for the Before/After Page Image.
18-35	Sequence number - a binary sequence number. BEFORE records are incremented by 1, starting with 1. AFTER records are decremented by 1, starting with -1.
Word 18 through Word 19 bits 0-35	I-D-S Data File Name - a 12-character name left justified.

I-D-S DATA PAGE AREA

Word 20 through Word n	The area which contains the I-D-S data page when read from the mass storage device.

Buffer Strategy for Page Buffers

Each time a page is brought into memory its buffer number is placed at the head of a buffer table. If a page already in memory is used again, its buffer number moves to the head of the table. Thus, the most frequently used pages are at the top of the table and the pages with little or no recent use are at the bottom of the table. Buffer space is always available for reading a data page. To make a buffer available, the page at the bottom of the list is written back to the mass storage device, provided there has been activity updating that page. This buffer is called the EMPTY buffer; it is the buffer with lowest activity.

The order of the chain is defined in an Page Buffer Activity Table (Figure 33) which contains one word for each page buffer in .QAREA. The activity chain shown in Figure 34 is a closed circular loop of buffer numbers.

There is always an EMPTY buffer whose NEXT is the buffer of highest activity.

The PRIOR of the buffer of highest activity is the EMPTY buffer.

The other buffers in the Page Buffer Activity Table have, in the PRIOR column (bits 0-17), the buffer number of the next higher (more recent) activity. The NEXT column (bits 18-35) contains the buffer number of the next lower (less recent) activity.

For example, if buffer 5 is the EMPTY buffer, then buffer 4 is the most active buffer.

Buffer Number	0 PRIOR 17	18 NEXT 35
0	4	2
1	3	5
2	0	3
3	2	1
4	5	0
5	1	4

Figure 33. Page Buffer Activity Table

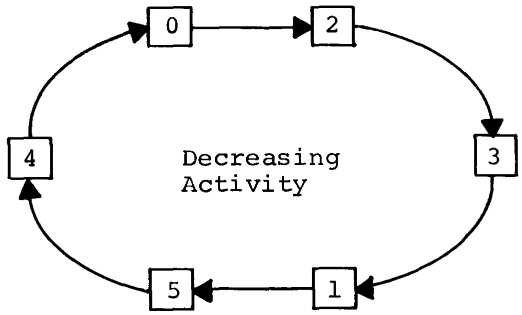


Figure 34. Chain Concept of Buffer Activity

Page Description

There are two types of I-D-S data pages:

Base Page
I-D-S Pagette

The I-D-S data page consists of a fixed size which is assigned when the I-D-S file is created. It may contain any combination of logical record types linked into their respective chains. Each type has its own specific length. Related record types are associated and linked according to their data content and may be stored within the same page. Space is fully utilized by packing these records within the page.

Every page begins with a unique Page Header record. This record contains several control fields used by the I-D-S subroutines, as follows:

1. Reference address of the page (page number).
2. Space available for additional records.
3. I/O control indicating whether the page has been altered since retrieval.
4. Chain field indicating reference code of the first record of a chain of calculated records, all of which randomize to this page.
5. Line numbers available for assignment within the page.

Base Page. The format of the Base Page Header record is shown in Figure 35.

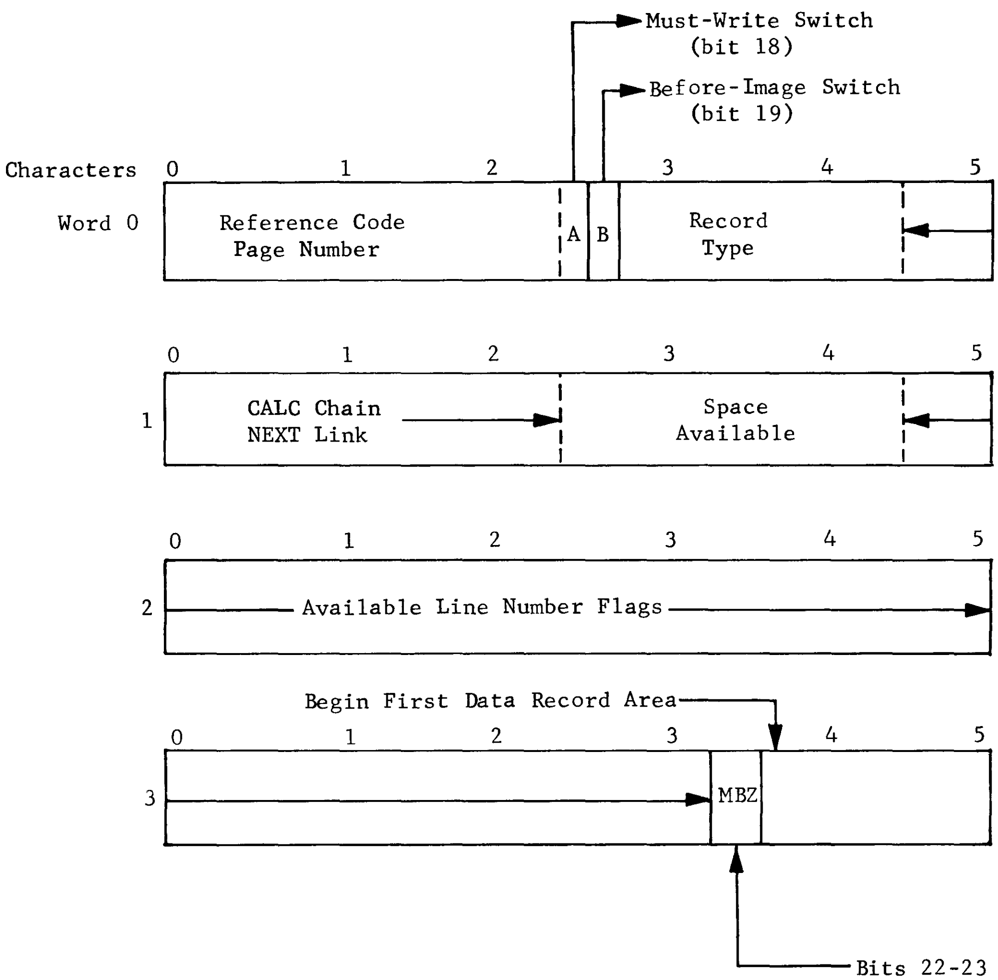


Figure 35. Base Page Header Record

The bit configuration for the Base Page Header record follows:

Word 0

bits	
0-17	Reference code page number - a number from 1 through 262,143. During file initialization, each page requested by the user is assigned a unique number within this range.
18	Must-Write switch - an indicator used by I-D-S to determine if a page has been altered since retrieval.
19	Before-Image switch - an indicator used by I-D-S to indicate that a page to be modified has been written to the journal tape prior to the modification.
20-29	Record type - a code of 1000(10) assigned to each Page Header record.
30-35	First character of the CALC chain NEXT Link - a pointer to the first CALC record contained in the chain. If no CALC records are present, it points to itself. (The Page Header record is the defined master of the CALC chain.)

Word 1

bits	
0-17	CALC Chain NEXT Link
18-29	Space available - current status of available space for storing records within a page.
30-35	Available line number flags (0-5) - an indicator used by I-D-S to determine line numbers available for assignment within a page: 0 = line number available 1 = line number not available There are 64 line number flags. They are numbered left to right starting with zero. Line number 0 is always used; it is line number of the Page Header record. A maximum of 63 data records can be stored to a page.

Word 2

bits	
0-35	Available line number flags (6-41)

Word 3

bits 0-21	Available line number flags (42-63).
22-23	Must be zero.
	End of Page Header record. The length of the record is 22 characters.
24-35 through word n	Bits 24-35 of word 3 through bit 35 of word n contain data records.

Pagette. A pagette is introduced by setting the value of lines per page to less than 63. By dividing the lines per page into 63, the number of pages required to hold 63 line flags is developed. The first of these pages is called the BASE page. It contains a Page Header record (type 1000 decimal) which is the master record of the CALC chain for this page number. The remaining pages are called PAGETTES. They contain a Pagette Header record (type 1003 decimal). They are not the master of any chain.

The available line number flags begin in the base page. For example, if the lines per page equal 21, this would require (63/21=3) pages to hold the 63 line number flags. Pages will have the line number flags set off for line numbers not allowed in the page. Thus, a base page will have line number flags as follows:

1 - 21	Available
22 - 63	Not available

Pagette number 1 will have:

1 - 21	Not available
22 - 42	Available
43 - 63	Not available

Pagette number 2 will have:

1 - 42	Not available
43 - 63	Available

The pagette allows users to increase the number of reference codes in their I-D-S data file. This facility is probably most useful on some portion of the total file that has been filled by large records.

For example, let record size be equal to the available space in a page such that one logical record fills a page. This record may be a dictionary record. When this record is stored the user eliminates 62 available reference codes. Thus, if several records are stored, several hundred potential reference codes are eliminated.

The user may choose to increase the page size such that several large records may fit into a single page, but practical limits on page size must be observed. Thus, the next approach may be to limit the lines per page, making available all 63 lines (reference codes) for each page.

The Pagette Header record format is shown in Figure 36.

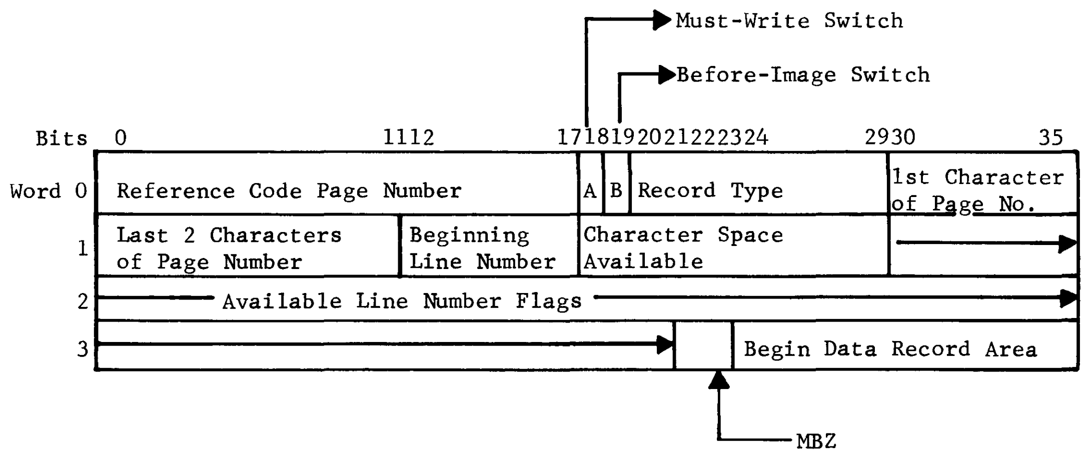


Figure 36. Pagette Header Record

The bit configuration for the Pagette Header record follows:

Word 0
bits

0-17	Reference code pagette number - a number from 1 through 262,143. During initialization, each pagette is assigned a unique number within this range.
18	Must-Write switch - an indicator used by the I-D-S subroutines to determine if a pagette has been modified since retrieval.

19 Before-Image switch - an indicator used by the I-D-S subroutines to indicate that a pagette has been written to the journal tape prior to modification.

20-29 Record type - a value of 1003 (10) assigned to each Pagette Header record.

30-35 First character of the pagette number, which forms the beginning reference code.

Word 1
bits
0-11 Last two characters of the pagette number, which forms the beginning reference code.

12-17 Beginning line number of pagette - first available line number that may be placed in the pagette.

18-29 Space available - characters of available space for storing records within the pagette.

30-35 Available line number flags (0 through 5) - an indicator used by I-D-S to determine which line numbers are available for assignment within a pagette:

0 = line number available
1 = line number not available

There are 64 line number flags. They are numbered left to right starting with 0. Line number 0 is always used; it is the number of the Pagette Header record.

Word 2
bits
0-35 Available line number flags (6-41)

Word 3
bits
0-21 Available line number flags (42-63)

22-23 Must be zero.

End of Pagette Header record. The length of the record is 22 characters.

24-35
through
word n Bits 24-35 of word 3 through bit 35 of word n contain data records.

I-D-S DATA RECORDS

Data records of I-D-S are fixed-format, fixed-length; that is, the length and format of a specific type of record, such as payroll or inventory, are fixed by the specifications of the systems designer. Records of many different types, each with its own length and format, may be used in the system. To maintain control, each record must have the same identification fields at the beginning. These fields are (1) line number portion of the reference code, (2) record type and (3) record length. The rest of the record consists of data and chain fields to suit the application requirements.

Records may have any number of data fields, each defined as some number of decimal, alphabetic or alphanumeric characters. Fields may vary in size from one character to many characters, as for a drawing or part number or an employee's name. These fields must be specified by the systems designer.

The format of the data record is shown in Figure 37.

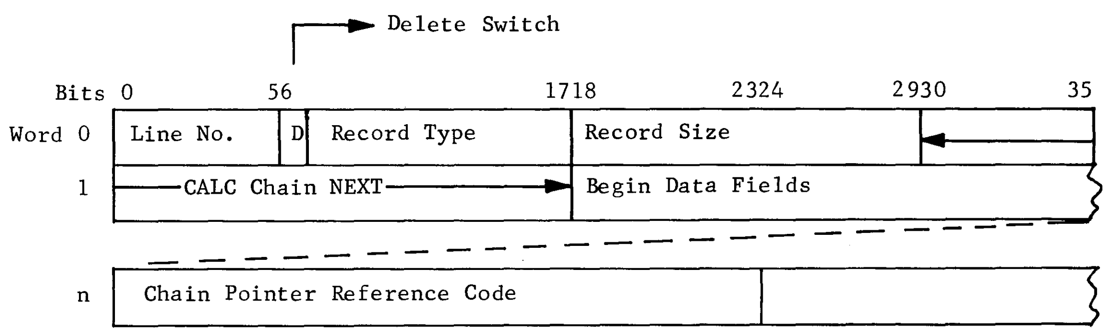


Figure 37. Data Record

The bit configuration for a data record follows:

Word 0 bits 0-5	Line number - a number from 1 to 63. A unique number is assigned to each data record as it is stored in a page. This number combined with the page number from the Page Header record completes the reference code.
6	Delete switch - an indicator used by the I-D-S subroutines to recognize a record that is logically but not physically deleted. When all chain pointers in a record are equal to zero, the record will then be physically deleted and its line number will be made available for use in the page.
7-17	Record type - a unique number from 1 to 999 used to identify different kinds of data records. The numbers 1000 and greater are reserved for use by I-D-S.
18-29	Record size - the number of characters in the record including all control fields, data fields and chain pointers. The line number is character 1 of a record.
30-35 and Word 1 bits 0-17	CALC chain NEXT - the reference code of the NEXT record in the CALC chain. If this is the last record in the CALC chain, it will contain a reference code of the Page Header record which is the master of this CALC chain. The chain pointer defined as detail of CALC chains. All other records do not contain this pointer and the data begins in this area. Bits 30-35 and 0-11 Page number 12-17 Line number
18-35 through word n	Beginning of available space for data characters. The data may be n characters in length.
Word n bits 0-23	The Chain pointers begin in the character position immediately following the last data character. The chain pointer reference code is 24 bits in length. Bits 0-17 Page number 18-23 Line number

There may not be any chain pointers in the record if it is not a member of any chain, such as a Primary record; or the only pointer may be the CALC chain NEXT. The presence of chain pointers is dependent on the description of the I-D-S record. The type of chain pointer, NEXT, PRIOR, HEAD, and the chain it is pointer for is described in the definition structure associated with this record type.

8. I-D-S Utility Programs and Subroutines

The following I-D-S utility programs and utility subroutines are described in this chapter:

Programs:

Randomizing Analyzer/Calc Pre-Load Sort Utility Program (QUTC)
Storage Tape Dump/Print Routine (QUTD)
Page Initialize Utility Routine (QUTI)
Journal Tape Dump (QUTJ)
Data Base Load/Print Utility Routine (QUTL)
Journal Record Selector (QUTP)
Execution Information Report (QUTR)
Selected Record Sort (QUTS)
File Utility (QUTU)

Subroutines:

Directive Processor (.QDIR)
Trace and Print Record (.QSTC)
Verify and Print (.QUTF)

I-D-S execution activities may require that permanent, temporary, or a mixture of an I-D-S data file be used. The following examples of deck setups may be applied to all I-D-S utility programs and subroutines and user execute activities.

PERMANENT I-D-S DATA FILE

The following deck setup is for a permanent I-D-S data file.

```
1        8          16
$        IDENT      IDS00,DATABASEMGR, PERM IDS FILE
$        USERID     IDSFOURYQUAD$DATABASE
           .
           .
$        PRMFL      A1,R/W,R,IDSFOURYQUAD/QUAD01
$        PRMFL      A2,R/W,R,IDSFOURYQUAD/QUAD02
$        PRMFL      A3,R/W,R,IDSFOURYQUAD/QUAD03
$        PRMFL      A4,R/W,R,IDSFOURYQUAD/QUAD04
           .
           .
$        ENDJOB
***EOF
```

TEMPORARY I-D-S DATA FILE

The following deck setup is for a temporary I-D-S data file.

```
1        8          16
$        IDENT      IDS00,DATABASEMGR, TEMP IDS FILE
           .
           .
$        MASS       A1,X1S,13R
$        DISC       A2,X2S,11R
$        DRUM       A3,X3S,11R
$        MASS       A4,X4S,11R
$        DATA       .Q
IDS      CREATE     FC/A1/,BASESIZE/480/,RANGE/1,120/,
                      PAGESIZE/160/,LINESPERPAGE/32/,
                      INVENTORY/25/
IDS      CREATE     FC/A2/,BSSZ/480/,RNG/121,240/,PGSZ/320/
IDS      CREATE     FC/A3/,RNG/241,360/
IDS      CREATE     FC/A4/,RNG/361,480/,PAGESIZE/64/,LPP/15/
$        DATA       I*
           .
           .
$        ENDJOB
***EOF
```

TEMPORARY AND PERMANENT I-D-S DATA FILE

The following deck setup is for a mixed temporary and permanent I-D-S data file.

The permanent I-D-S data file attributes were supplied when the file was created; the temporary attributes must agree with the permanent attributes.

```
1      8       16
$      IDENT   IDS00,DATABASEMGR, MIXED IDS FILE
$      USERID  IDSFOURYQUAD$DATABASE
         .
         .
$      PRMFL   A1,R/W,R,IDSFOURYQUAD/QUAD01
$      MASS    A2,X2S,11R
$      DISC    A3,X3S,11R
$      PRMFL   A4,R/W,R,IDSFOURYQUAD/QUAD04
$      DATA    .Q
IDS    CREATE  FC/A2/,BSSZ/480/,RNG/121,240/,PGSZ/320/
IDS    CREATE  FC/A3/,RNG/241,360/
$      DATA    I*
         .
$      ENDJOB
***EOF
```

UTILITY PROGRAM AND SUBROUTINE DESCRIPTIONS

Descriptions of the I-D-S utility programs and utility subroutines are presented on the following pages.

QUTC

Randomizing Analyzer/CALC Pre-Load Sort Utility Program (QUTC)

The QUTC utility program performs two distinct functions depending upon the directive option chosen.

1. The ANAL option utilizes the user supplied directive cards to generate numbers which are randomized to produce base page numbers and the total number of times each base page number is returned by the CALC routine. This information is printed on the Base Page Report. In addition, should the page number occur more than 63 (maximum lines per page) times, or any smaller number supplied by the user, this and the page number of the first page having space available will be indicated on the Overflow Report.

2. The RAND or RANDA option is used to sort CALC records into base page sequence prior to loading the data base. The user's input file must contain only one record type and must be in system standard format. A minimum of one control field for randomizing must be specified and a maximum of three control fields may be specified. These control fields must appear on the directive card in the order in which randomization is to be performed.

The total number of records randomizing to each page is accumulated and printed on the Base Page Report. In addition, a control for overflow is maintained which forces all overflow records to be sorted to the end of the file. The page which has overflow and the first page with space available is indicated on the Overflow Report.

Printing of these two reports may be suppressed by use of the RAND option.

Directive

General Format:

1	8	16
IDS	OPTION	GENERATE/,OPTION/,
	ETC	CONTROL/OPTION/,
	ETC	CONTROL/OPTION/,
		...

ANAL OPTION

This option generates page numbers based on control parameters explained below.

INPUT:
Directive cards

OUTPUT:
IDS BASE PAGE REPORT
IDS OVERFLOW REPORT

Control for ANAL option on directive card:

RNG/P1,P2/	Specifies the page range to be analyzed.
MAX/nn...n/	Specifies the largest number to be randomized.
INCR/nn...n/	Specifies the increment to be added for each iteration.
FILL/nn/	Specifies the point at which the page will be considered full.

Example:

```
1       8          16
IDS     OPTION     GENERATE/,ANAL/,
        ETC        RNG/1,100000/,MAX/50000/,
        ETC        INCR/2/,FILL/32/
```

RANDA OPTION

This option takes a user's file of CALC records, randomizes on specified control fields, producing a base page number, sorts the file into page number sequence with all overflow records sorted to the end of the file and produces two reports.

INPUT:
USER's CALC file
Directive cards

OUTPUT:
Sorted CALC file
IDS BASE PAGE REPORT
IDS OVERFLOW REPORT

Control for RANDA option on directive card:

RNG/P1,P2/	Specifies the page range into which records are to be stored.

CF/C1,L1/ Specifies fields to be used for randomizing.

C1 reflects the beginning character of the field relative to one.

L1 reflects the lengths in characters of the control field.

A minimum of one control field must be specified and a maximum of three may be given.

FILL/nn/ Specifies the point at which a page will be considered full. If this parameter is not supplied 63 records per page is assumed.

Example:

```
1      8          16
IDS    OPTION     GENERATE/,RANDA/,RNG/500,1000/,
       ETC        CF/2,6/,CF/20,5/,CF/10,2/,
       ETC        FILL/63/
```

RAND OPTION

This option has the same effect as the RANDA option with the exception that no reports are produced.

Example:

```
1      8          16
IDS    OPTION     GENERATE/,RAND/,RNG/10,200/,
       ETC        CF/8,10/,FILL/8/
```

Directive Restrictions

1. Directives are examined to ensure that columns 1-3 contain IDS, that columns 8-13 contain OPTION and that the first parameter in the variable field is GENERATE.

2. All control parameters are required with the exception of FILL. This is assumed to be 63 when not specified.

Operation

1. Deck Setup for RANDA Option:

```
     1       8         16
    |$      |IDENT    |VTA00,YOUNGMAN,K72
    |       |         |
    |$      |PROGRAM  |QUTI                     Activity 1.
  * |$      |MASS     |A1,D1S,10R
    |$      |DATA     |.Q
*** |IDS    |CREATE   |FC/A1/,BSSZ/100/,RNG/1,100/
    |$      |DATA     |I*
    |IDS    |INITIAL  |1,100
    |       |         |
    |$      |PROGRAM  |QUTC                     Activity 2.
    |$      |LIMITS   |10,26K,,
 ** |$      |MASS     |A1,A1R,25L               (Work file)
 ** |$      |MASS     |B1,B1R,25L               (Work file)
 ** |$      |TAPE     |T1,T1D,,1234,,USER-IN    (User's input file)
 ** |$      |TAPE     |C1,C1D,,,,USER-SORTED    (User's output file)
 ** |$      |MASS     |S1,S1R,10R               (Sort work file)
  * |$      |MASS     |D1,D1R,10R               (Work IDS file)
    |$      |SYSOUT   |P1                       (Report file)
    |$      |DATA     |.Q
*** |IDS    |CREATE   |FC/D1/,BSSZ/100/,RNG/1,100/
    |$      |DATA     |I*
*** |IDS    |OPTION   |GENERATE/,RANDA/,RNG/1,30000/,
    |IDS    |ETC      |CF/2,6/,CF/20,5/,CF/10,2/,FILL/63/
    |$      |ENDJOB   |
    |***EOF |         |
```

* The required file codes are A1 and D1 respectively. The file code A1 on LUD D1S in activity 1 is used as file code D1 on LUD D1R in activity 2. This file must be mass storage.

** The required file codes are as defined in the example. Tape or mass storage are acceptable as file types.

*** The BASESIZE and RANGE for the work I-D-S file may be computed in the following manner using the RANGE from the OPTION directive:

((maximum range - minimum range)+1)/300=BSSZ

((30000 -1)+1)/300=100

The PAGESIZE for this file must be 320 words and the LINES per page must be 63.

2. Deck Setup for ANAL Option:

```
       1      8         16
       $      IDENT     VTA00,YOUNGMAN,K72

       $      PROGRAM   QUTI                        Activity 1.
   *   $      MASS      A1,D1S,10R
       $      DATA      .Q
 ***   IDS    CREATE    FC/A1/,BSSZ/100/,RNG/1,100/
       $      DATA      I*
       IDS    INITIAL   1,100

       $      PROGRAM   QUTC                        Activity 2.
       $      LIMITS    10,26K
  **   $      MASS      A1,A1R,25L                  (Work file)
  **   $      MASS      B1,B1R,25L                  (Work file)
   *   $      MASS      S1,S1R,10R                  (Sort work file)
  **   $      MASS      D1,D1R,10R                  (Work I-D-S file)
       $      SYSOUT    P1                          (Report file)
       $      DATA      .Q
 ***   IDS    CREATE    FC/D1/,BSSZ/100/,RNG/1,100/
       $      DATA      I*
 ***   IDS    OPTION    GENERATE/,ANAL/,RNG/1,30000/,
       IDS    ETC       INCR/1/,FILL/63/,MAX/100000/
       $      ENDJOB
       ***EOF
```

* Same as for RANDA Option.

** Same as for RANDA Option.

*** Same as for RANDA Option.

3. Subroutines Called:

.QOPEN - opens mass storage device files and builds tables to describe them.

.QDIR - reads directives.

.QMEX - writes messages on the execution report.

.QSFD - advances subfields of the variable field of directive for processing.

.QCALC - computes a base page number.

IDS BASE PAGE REPORT

PAGE 001

PAGE NUMBER	NR OF RECORDS	PAGE NUMBER	NR OF RECORDS	PAGE NUMBER	NR OF RECORDS
1	6	2	2	3	6
4	4	5	4	6	4
7	2	8	12	9	5
10	3	11	6	12	5
13	3	14	5	15	1
16	6	17	3	18	3
19	5	20	6	21	4
22	4	23	5	24	10
25	4	26	3	27	6
28	6	29	8	30	3
31	2	32	6	33	4
34	3	35	5	36	12
37	2	38	4	39	1
40	6	41	2	42	5
43	4	44	2	45	4
46	5	47	8	48	12
49	5	50	3	51	4
52	5	53	4	54	5
55	7	56	7	57	11
58	7	59	5	60	5
61	4	62	4	63	6
64	7	65	5	66	5
67	3	68	7	69	8
70	5	71	3	72	5
73	5	74	3	75	3
76	5	77	10	78	3
79	2	80	12	81	7
82	4	83	4	84	7
85	3	86	2	87	4
88	8	89	1	90	4
91	5	92	6	93	5
94	1	95	6	96	6
97	6	98	6	99	3
100	8				

IDS OVERFLOW REPORT

RANDOMIZED TO	STORED ON	RANDOMIZED TO	STORED ON	RANDOMIZED TO	STORED ON
8	9	8	9	8	9
8	10	24	25	24	25
36	37	36	37	36	37
36	37	48	49	48	49
48	49	48	50	57	58
57	59	57	59	77	78
77	78	80	81	80	82

Storage Tape Dump/Print Utility Routine (QUTD)

QUTD dumps to tape and/or prints all or selected portions of the appropriate storage devices allocated to the I-D-S data file. The portions of the file to be processed and the output media are specified by input data cards (directives).

Directives

Directive fields begin in column 16 and are separated by commas. One or more ETC cards may be used to continue the fields if they run beyond column 72. Each card to be continued must end with a complete field, followed by a comma.

There are four directives recognized by QUTD.

1	8	16
IDS	DUMP	RNG/P1,P2/ null

The DUMP directive causes pages P1 through P2 to be written on magnetic tape. If the variable field is null, all pages of the file are written on magnetic tape. The file code for the magnetic tape is OT. RNG/P1,P2/ is the only option valid for this directive.

1	8	16
IDS	PRINT	RNG/P1,P2/,..., Print option

The PRINT directive causes P1 through P2 to be written in print format and directed to SYSOUT via file code P*. If RNG is not specified, all pages of the file are written.

Print Options

NULL — Prints nonempty pages and indicates empty pages.

EMPTY — Prints nonempty pages and the page header for each empty page rather than indicating a succession of empty pages only by a first page entry and a last page entry.

TYPES/A,B,C,.../ Prints only the record types specified by A,B,C, etc. (to a maximum of 10 types).

DELETE Produces a file containing reference code, size and record type of all records deleted but still present on the file.

```
1      8          16
IDS    DPRINT     RNG/P1,P2/,.../
       PDUMP
```

The DPRINT/PDUMP directive (either form is acceptable) causes pages P1 through P2 to be written on magnetic tape and to be sent to SYSOUT in print format, via file code P*. Either directive is a combination of the DUMP and PRINT directives. A null variable field causes all pages of the file to be written. All print options listed above are acceptable with either of these directives.

```
1      8          16
IDS    EOR        (not examined)
```

The EOR directive forces an end-of-reel condition on the magnetic tape file.

```
PAGE:  XXXXX XX   ACTIVE PAGE SIZE:   XXXX CH.

WD:   LN:    TYPE:
XXX   XX     XXXX    [OCTAL] [OCTAL] [OCTAL] [OCTAL] [BCD]
                     [OCTAL] [OCTAL] [OCTAL] [OCTAL] [BCD]
                     [OCTAL] [OCTAL] [OCTAL] [OCTAL] [BCD]
XXX   XX     XXXX    [OCTAL] [OCTAL] [OCTAL] [OCTAL] [BCD]

PAGE:  XXXXX XX   ACTIVE PAGE SIZE:   XXXX CH.   PAGE EMPTY
       AND ALL INTERVENING PAGES

PAGE   XXXXX XX   ACTIVE PAGE SIZE:   XXXX CH.   PAGE EMPTY
```

Tape Format

The data sent to the output tape file is written as variable length, logical records using the GEFRC subroutine PUT. The file is in standard system format with the exception of block size which is 1602 words. The Page Image record format is:

Word	Contents
0	Accounting Record Header. The number of data words in the record is specified in bits 0-17. The record type, octal 000013, is contained in bits 18-35.
1	Checksum.
2	SNUMB in bits 0-29. Ignore character (octal 17) in bits 30-35.
3	Date as MMDDYY.
4	Start time in hours and thousandths of hours as HH.TTT.
5	Record type in bits 0-11 as 10. Bits 12-35 are presently unused and are zero.
6	This word is presently unused and is zero.
7	First six characters of user identification.
8	Second six characters of user identification.
9-n	Active page image.

Execution Report

An execution report is produced as part of the user output. It describes, in chronological order, the functions performed as specified in the directives. In addition, error conditions are included to advise the user of exception conditions.

Operation

The following deck setups can be used to execute QUTD from the software library.

1. Example for temporary files.

```
1      8       16
$      IDENT
$      PROGRAM QUTD
$      LIMITS  OPTIONS
$      MASS    A1,X1R,15R              (required file code)
$      TAPE    OT,X2S,,,,DUMP-FILE     (required file code)
$      TAPE    DE,X3S,,,,DELETE-FILE   (required file code)
$      DATA    .Q
IDS    CREATE  FC/A1/,BSSZ/480/,RNG/1,120/,LPP/63/
$      DATA    I*
IDS    PDUMP   DELETE/
$      ENDJOB
***EOF
```

a. Pages 1 through 120 will be written to tape (file code OT).

b. All nonempty pages will be printed on P* and all empty pages will be indicated with a beginning and ending page number.

c. All records logically but not physically deleted from the file will be written to tape (file code DE) and flagged on the printed report.

2. Example for permanent files.

```
1      8       16
$      IDENT
$      PROGRAM QUTD
$      LIMITS  OPTIONS
$      USERID  IDSFOURYQUAD$DBASE
$      PRMFL   TF,R/W,R,IDSFOURYQUAD$DBASE/QUAD01
$      PRMFL   TG,R/W,R,IDSFOURYQUAD$DBASE/QUAD02
$      TAPE    QT,X2S,,,,DUMP-FILE
$      DATA    I*
IDS    DUMP    RNG/1,120/
IDS    DPRINT  EMPTY/,TYPES/100,101,102/
$      ENDJOB
***EOF
```

This deck setup will result in the following:

a. Pages 1 through 120 (file code TF) will be written to tape (file code OT).

b. File code TG, in its entirety, will be written to tape.

c. All page headers and all record types 100, 101 and 102 on file code TG will be printed.

Page Initialize Utility Routine (QUTI)

QUTI initializes all or selected portions of the appropriate storage devices allocated to the I-D-S data file with the page headers and creates or updates the inventory records. The portions of the file to be processed are specified by an input data card (directive). The attributes of the file are acknowledged during the initialization process.

Directives

There are two directives recognized by .QUTI.

```
1      8       16
IDS    INITIAL P1,P2
```

The initial directive causes pages P1 through P2 to be initialized with their page headers and the inventory records to be created as required.

```
1      8       16
IDS    HEADER  P1,P2
```

The header directive causes pages P1 through P2 to be initialized with their page headers and the inventory records to be updated as required. This requires that the portion of the file must have been previously initialized by the initial directive. This directive allows a portion of the file to be purged and the inventory to be reset.

Directive Restrictions

1. The argument P2 must be greater than or equal to the argument P1.

2. Directives are examined to ensure that columns 8-13 contain a legal directive code as described. Directives in error are written on the execution report followed by appropriate comments.

Execution Report

An execution report is produced as part of the user output. It describes, in chronological order, the functions performed as specified in the directives. In addition, error conditions are included to advise the user of exception conditions.

Operation

1. Deck setup.

   The following deck setup will initialize a permanent I-D-S data file.

```
1      8       16
$      SNUMB   QUTI
$      IDENT   IDS00,DATABASEMGR
$      USERID  IDSFOURYQUAD$DATABASE
$      PROGRAM QUTI
$      PRMFL   A1,R/W,R,IDSFOURYQUAD/QUAD01
IDS    INITIAL 1,120
$      ENDJOB
***EOF
```

   The following deck setup can be used to initialize a temporary I-D-S data file.

```
1      8       16
$      SNUMB   QUTI
$      IDENT   IDS00,DATABASEMGR,TEMP FILE
$      PROGRAM QUTI
$      "file"  or A1,X1S,13R   (A1 is the required file code)
$      DATA    .Q
IDS    CREATE  FC/A1/,BSSZ/480/,RNG/1,120/,PGSZ/192/,
                 LPP/32/,INV/25/
$      DATA    I*
IDS    INITIAL 1,120
$      ENDJOB
***EOF
```

2. Subroutines called.

   .QDIR - reads directives.

   .QMEX - writes messages on the execution report.

.QMWD - moves blocks of words from one location in memory to another.

.QPSP - supplies a tallied I/O list for pages to be read from the mass storage device(s).

.QSFD - advances subfields of the variable field of directive for processing.

.QDIRC - closes the directive file.

.QDIRP - establishes the file code (I*) for directives.

.QOPEN - opens the mass storage device file(s) and builds the tables that describe them.

.QRTAB - verifies that the requested pages are allocated and builds the tables, by device, for the required page ranges.

.QSICT - points indirectly to the mass storage device file descriptions.

.QTAB1 - contains table of FROM page ranges.

.QTAB2 - contains table of TO page ranges.

.QTAB3 - contains the number of entries, minus one, in .QTAB1/ :QTAB2. This count is in bits 0-17. Bits 18-35 are not examined.

.QWAIT - insures that all outstanding I/O on the mass storage device is completed.

.QBCD - converts binary to BCD and replaces leading zeros with blanks.

.QMCH - moves blocks of characters from one location in memory to another.

.QINV1 - updates inventory.

.QWRIT - performs buffered writing to the mass storage device.

.QPHI - generates the page headers.

.QCLOS - closes the files and generates the I/O statistic report.

.QMAP1 - calculates the relative sector.

Journal Tape Dump Utility Program (QUTJ)

QUTJ dumps selected portions of tapes in the standard I-D-S journal format. This includes tapes created by master mode or slave mode journalization, and tapes produced by either QUTU, QUTP, or QUTS.

Directive

One directive is recognized by the QUTJ program.

```
1      8      16
IDS    SYSTEM
```

The SYSTEM directive causes only record types 3 and 4 (SLVBGN and SLVEND) to be printed on the report. All other record types are ignored.

Printer Format

The record types recognized by QUTJ are printed in the format shown below.

Record Type	Column 1	Column 2	Column 3	Column 4	Column 5	Column 6	Column 7	Column 8	Column 9	Column 10
Slave Begin	Logical Record Number		SLVBGN	AA-SSSSS	Date	Time	Record Type	Sequence Word	12 Charac-ter user ID	Blank
Slave End	Logical Record Number		SLVEND	AA-SSSSS	Date	Time	Record Type		12 charac-ter user ID	Blank
BEFORE Image	Logical Record Number		BEFORE	AA-SSSSS	Date	Time	Record Type	Sequence Word	12 charac-ter user ID	Page Number
AFTER Image	Logical Record Number		AFTER	AA-SSSSS	Date	Time	Record Type	Sequence Word	12 charac-ter user	Page Number
QUTU Image	Logical Record Number		QUTU	AA-SSSSS	Date	Time	Record Type	Sequence Word	12 charac-ter user ID	Page Number

Column Description

Column 2 - Two asterisks appear in this column if the checksum of this record is in error.

Column 5 - DATE is displayed in the form of MM-DD-YY.

Column 6 - TIME is displayed as HH.TTT, hours and thousandths of hours.

Column 8 - This word is the abort code for SLVEND records. A code of 00 is used for end of activity and end of job.

The SEQUENCE word is zero for all record types except BEFORE and AFTER. BEFORE page image records are incremented by 1, starting with 1 in bits 18-35. AFTER page image records are decremented by 1, starting with -1.

Column 10 - The PAGE NUMBER is the first 18 bits of the first word of the page image followed by the line number.

Execution Report

QUTJ writes the printed output on the execution report via SYSOUT. The input tape label is the first line of the report.

Operation

The following deck setup can be used to execute QUTJ.

```
1      8       16
$      IDENT
$      PROGRAM QUTJ
$      LIMITS  Options
$      TAPE    IN,Options   (IN is required file code)
$      ENDJOB
***EOF
```

Sample Output

The following page illustrates the output format produced by QUTJ; the input tape was a master mode journal tape.

QJNL1 01 10-01-68 11,092 IDS JOURNAL TAPE REPORT

IDS UTILITY ROUTINE - ,QUTJ - VERSION 080168,

3	SLVBGN	1-TST03	10-01-68	11,037	030	0	IDSFOURYQUAD		
4	BEFORE	1-TST03	10-01-68	11,037	050	1	IDSFOURYQUAD	22	0
5	BEFORE	1-TST03	10-01-68	11,037	050	2	IDSFOURYQUAD	382	0
6	BEFORE	1-TST03	10-01-68	11,037	050	3	IDSFOURYQUAD	20	0
7	BEFORE	1-TST03	10-01-68	11,037	050	4	IDSFOURYQUAD	380	0
8	BEFORE	1-TST03	10-01-68	11,037	050	5	IDSFOURYQUAD	112	0
9	BEFORE	1-TST03	10-01-68	11,037	050	6	IDSFOURYQUAD	472	0
10	BEFORE	1-TST03	10-01-68	11,037	050	7	IDSFOURYQUAD	55	0
11	BEFORE	1-TST03	10-01-68	11,037	050	8	IDSFOURYQUAD	415	0
12	BEFORE	1-TST03	10-01-68	11,037	050	9	IDSFOURYQUAD	65	0
13	BEFORE	1-TST03	10-01-68	11,037	050	10	IDSFOURYQUAD	425	0
14	BEFORE	1-TST03	10-01-68	11,037	050	11	IDSFOURYQUAD	33	0
15	BEFORE	1-TST03	10-01-68	11,037	050	12	IDSFOURYQUAD	393	0
16	BEFORE	1-TST03	10-01-68	11,037	050	13	IDSFOURYQUAD	86	0
17	BEFORE	1-TST03	10-01-68	11,037	050	14	IDSFOURYQUAD	446	0
18	BEFORE	1-TST03	10-01-68	11,037	050	15	IDSFOURYQUAD	115	0
19	BEFORE	1-TST03	10-01-68	11,037	050	16	IDSFOURYQUAD	475	0
20	AFTER	1-TST03	10-01-68	11,037	060	262142	IDSFOURYQUAD	22	0
21	BEFORE	1-TST03	10-01-68	11,037	050	17	IDSFOURYQUAD	70	0
22	AFTER	1-TST03	10-01-68	11,037	060	262141	IDSFOURYQUAD	382	0
23	BEFORE	1-TST03	10-01-68	11,037	050	18	IDSFOURYQUAD	430	0
24	AFTER	1-TST03	10-01-68	11,037	060	262140	IDSFOURYQUAD	20	0
25	BEFORE	1-TST03	10-01-68	11,037	050	19	IDSFOURYQUAD	68	0
26	AFTER	1-TST03	10-01-68	11,037	060	262139	IDSFOURYQUAD	380	0
27	BEFORE	1-TST03	10-01-68	11,037	050	20	IDSFOURYQUAD	428	0
28	AFTER	1-TST03	10-01-68	11,037	060	262138	IDSFOURYQUAD	112	0
29	BEFORE	1-TST03	10-01-68	11,037	050	21	IDSFOURYQUAD	40	0
30	AFTER	1-TST03	10-01-68	11,037	060	262137	IDSFOURYQUAD	472	0
31	BEFORE	1-TST03	10-01-68	11,037	050	22	IDSFOURYQUAD	400	0
32	AFTER	1-TST03	10-01-68	11,037	060	262136	IDSFOURYQUAD	55	0
33	BEFORE	1-TST03	10-01-68	11,037	050	23	IDSFOURYQUAD	16	0
34	AFTER	1-TST03	10-01-68	11,037	060	262135	IDSFOURYQUAD	415	0
35	BEFORE	1-TST03	10-01-68	11,037	050	24	IDSFOURYQUAD	376	0
36	AFTER	1-TST03	10-01-68	11,037	060	262134	IDSFOURYQUAD	65	0
37	BEFORE	1-TST03	10-01-68	11,037	050	25	IDSFOURYQUAD	111	0
38	AFTER	1-TST03	10-01-68	11,037	060	262133	IDSFOURYQUAD	425	0
39	BEFORE	1-TST03	10-01-68	11,037	050	26	IDSFOURYQUAD	471	0
40	AFTER	1-TST03	10-01-68	11,037	060	262132	IDSFOURYQUAD	33	0
41	BEFORE	1-TST03	10-01-68	11,037	050	27	IDSFOURYQUAD	1	0
42	AFTER	1-TST03	10-01-68	11,037	060	262131	IDSFOURYQUAD	393	0
43	BEFORE	1-TST03	10-01-68	11,037	050	28	IDSFOURYQUAD	361	0
44	AFTER	1-TST03	10-01-68	11,037	060	262130	IDSFOURYQUAD	86	0
45	BEFORE	1-TST03	10-01-68	11,037	050	29	IDSFOURYQUAD	81	0
46	SLVBGN	1-TST3C	10-01-68	11,038	030	0	IDSFOURYQUAD		
47	AFTER	1-TST03	10-01-68	11,037	060	262129	IDSFOURYQUAD	446	0
48	BEFORE	1-TST03	10-01-68	11,037	050	30	IDSFOURYQUAD	441	0
49	BEFORE	1-TST3C	10-01-68	11,038	050	1	IDSFOURYQUAD	142	0
50	AFTER	1-TST03	10-01-68	11,037	060	262128	IDSFOURYQUAD	115	0
51	BEFORE	1-TST03	10-01-68	11,037	050	31	IDSFOURYQUAD	22	0
52	BEFORE	1-TST3C	10-01-68	11,038	050	2	IDSFOURYQUAD	262	0

Data Base Load/Print Utility Routine (QUTL)

QUTL loads and/or prints all or selected I-D-S pages from an input file. The input file may be:

- Dump File created by QUTD
- Selected File created by QUTS
- System Statistical Collection File or User Journal File (JX)

DIRECTIVES

The QUTL utility is controlled through the following directive:

1	8	16
IDS	OPTION	Function/Input Descriptor/,
	ETC	Descriptor options/,
	ETC	PRINT OPTIONS/

Directive fields being in column 16, they are terminated by a slash (/) and are separated by commas. One or more ETC cards may be used to continue the fields. A directive card to be continued must end with a complete field, followed by a comma.

Operation

The operation of the utility varies depending upon the type of input file. The utility is written in a modular (overlay) manner such that only the coding needed to accomplish the desired function is engaged.

There are three INPUT DESCRIPTOR options recognized by the utility:

DTAPE
STAPE
JTAPE

The directive options applicable for each type of INPUT DESCRIPTOR and resulting operation are described as though three unique utilities actually exist.

The Printer Format and Tape Formats are common to the three modes of operation.

DTAPE - Input Descriptor

The use of DTAPE as thc Input Descriptor indicates that the input file contains data produced by the I-D-S utility QUTD.

DIRECTIVE OPTIONS

FUNCTION

LOAD	Causes specified pages to be written on the mass storage device.
PRINT	Causes specified pages to be written in print format and directed to SYSOUT via file code P*.
LPRINT PLOAD	These options (either form is acceptable) cause the specified pages to be written on the mass storage device and to be sent to SYSOUT in print format via file code P*. Either directive is a combination of the PRINT and LOAD functions.
EOR	Forces a unit switch on the input magnetic tape file.
RNG/P1,P2/	Specifies the page range to be reloaded and/or printed. If no range is present the entire range of all subfiles allocated is assumed. The argument P2 must be greater than or equal to P1.

PRINT OPTIONS

EMPTY	Prints non-empty pages and the page header for each empty page rather than indicating a succession of empty pages only by a first page entry and last page entry.
TYPES/A,B,C,../	Prints only the record types specified by A,B,C,...(to a maximum of 10 types).
DELETE	Produces a file containing reference code, size and record type of all records logically but not physically deleted from the file. The required file code is DE.

Directive Examples:

```
1      8       16
IDS    OPTION  DTAPE/,LOAD/
```

This requests reloading of all pages for all files allocated to the activity.

```
1      8       16
IDS    OPTION  DTAPE/,PLOAD/
```

This requests reloading of all pages for all files allocated to the activity and printing of all nonempty pages with all empty pages being indicated with a first page and last page entry.

```
1      8       16
IDS    OPTION  DTAPE/,PLOAD/,EMPTY/,
       ETC     TYPES/100,200/,RNG/18500,25000/,
       ETC     DELETE/
```

This requests reloading of all pages for the specified range, printing of all page headers and any records of the specified types. A file of all deleted records will also be produced.

Execution Report

An execution report is produced as a part of the output. It describes in chronological order, the functions performed as specified in the directive. In addition, error conditions are included to advise the user of exception conditions.

The input and output files are double buffered to obtain maximum throughput. The input file must contain consecutive pages for the files allocated or the PAGE-RANGE specified on the directive card. If nonconsecutive pages are encountered during execution, and error comment is written on the execution report and the program is aborted with a D2 reason code.

Inventory records will be created for the file or PAGE-RANGE re-loaded if applicable.

The minimum core requirement for this activity is 16K.

Operation

The following deck setup can be used to execute QUTL from the software library.

1. Example for temporary files.

```
1       8           16
$       IDENT       (options)
$       PROGRAM     QUTL
$       LIMITS      (options)      (minimum 16K)
$       MASS        A1,X1R,15R
$       TAPE        IN,X2S,,1234,,DUMP-FILE (Required File code)
$       TAPE        DE,X3S,,,,DELETE-FILE (Required File code)
$       DATA        .Q
IDS     CREATE      FC/A1/,BSSZ/480/,RNG/1,120/
$       DATA        I*
IDS     OPTION      DTAPE/,PLOAD/,RNG/1,120/,DELETE/
$       ENDJOB
```

This deck setup will result in the following:

a. Pages 1 through 120 will be written to the mass storage device.

b. All non-empty pages will be printed on P* and all empty pages will be indicated with a beginning and ending page number.

c. All records logically but not physically deleted from the data base will be written to tape (file code DE) and flagged on the printed report.

2. Example for permanent files:

```
1       8           16
$       IDENT       (options)
$       PROGRAM     QUTL
$       LIMITS      (options)
$       USERID      IDSFOURYQUAD$DBASE
$       PRMFL       TF,R/W,R,IDSFOURYQUAD$DBASE/QUAD01
$       PRMFL       TG,R/W,R,IDSFOURYQUAD$DBASE/QUAD02
$       PRMFL       TH,R/Q,R,IDSFOURYQUAD$DBASE/QUAD03
$       PRMFL       TI,R/Q,R,IDSFOURYQUAD$DBASE/QUAD04
$       TAPE        IN,X2S,,1234,,DUMP-TAPE  (Required File code)
$       DATA        I*
IDS     OPTION      DTAPE/,PLOAD/,EMPTY/,RNG/121,240/
```

This deck setup will result in the following:

a. Pages 121 through 240 will be written on the mass storage device.

b. All non-empty pages and page headers for all empty pages will be printed on P*.

STAPE - Input Descriptor

The use of STAPE as the Input Descriptor indicates that the input file contains data as produced by the I-D-S utility QUTS.

Input File

The input file is standard system formats with the exception of block size, which is 1602 words. The data on the file must have been written as output by the I-D-S utility QUTS; therefore it must consist of either the first BEFORE or last AFTER for each page supplied as input and only one image for each page will be present.

DIRECTIVE OPTIONS

FUNCTION

LOAD	Causes specified pages to be written on the mass storage device.
PRINT	Causes specified pages to be written in print format and directed to SYSOUT via file code P*.
LPRINT PLOAD	These options (either form is acceptable) cause specified pages to be written on the mass storage device and to be sent to SYSOUT in print format via file code P*. Either directive is a combination of the PRINT and LOAD functions.
EOR	Forces a unit switch on the input magnetic tape file.

DESCRIPTOR OPTIONS

RNG/P1,P2/	Specifies the page range to be reloaded and/or printed. If no range is present, the entire range of all subfiles allocated is assumed. The argument P2 must be greater than or equal to P1.

PRINT OPTIONS

TYPES/A,B,C,.../	Prints only the record types specifies by A,B,C,... (to a maximum of 10 types).
DELETE	Produces a file containing reference code, size and record type of all records logically but not physically deleted from the file. The required file code is DE.

DIRECTIVE EXAMPLES

```
1      8       16
IDS    OPTION  STAPE/,LOAD/
```

This requests reloading of all pages found on the input tape for all files allocated.

```
1      8       16
IDS    OPTION  STAPE/,PLOAD/,RNG/27500,35000/
```

This requests reloading and printing of all pages found for the specified range.

Execution

An execution report is produced as a part of the output. It describes in chronological order, the functions performed as specified in the directive. In addition error messages are included to advise the user of exception conditions.

Since pages are non-consecutive on this type load and each page must be processed based on the page number found in the record, the input is double buffered and the output is accomplished from the input buffer.

Minimum core requirement for this type load is 14K.

Inventory records will be updated for each page reloaded if applicable.

Operation

The following deck setup can be used to execute QUTL from the software library.

1. Example for temporary files.

```
1      8         16
$      IDENT     (options)
$      PROGRAM   QUTL
$      LIMITS    (options)
$      MASS      A1,X1R,15R    (required File code)
$      TAPE      IN,X2S,,1234,,DUMP-FILE  (Required File code)
$      DATA      .Q
IDS    CREATE    FC/A1/,BSSZ/480/,RNG/1,120/
$      DATA      I*
IDS    OPTION    STAPE/,LOAD/
$      ENDJOB
```

This causes reloading all pages found on the input tape for all files allocated.

2. Example for permanent files:

```
1      8         16
$      IDENT     (options)
$      PROGRAM   QUTL
$      LIMITS    (options)    (minimum 14K)
$      USERID    IDSFOURYQUAD$DBASE
$      PRMFL     TF,R/W,R,IDSFOURYQUAD$DBASE/QUAD01
$      PRMFL     TG,R/W,R,IDSFOURYQUAD$DBASE/QUAD02
$      PRMFL     TH,R/Q,R,IDSFOURYQUAD$DBASE/QUAD03
$      PRMFL     TI,R/W,R,IDSFOURYQUAD$DBASE/QUAD04
$      TAPE      IN,X2S,,1234,,SELECT-FILE
$      DATA      I*
IDS    OPTION    STAPE/,PLOAD/,RNG/100,200/
$      ENDJOB
```

This requests reloading and printing of all pages found for the specified range.

JTAPE - Input Descriptor

The use of JTAPE as the Input Descriptor indicates that the input file contains data of the System Statistical Collection Tape or a User Journal File (JX

Input File

The input file may be one or more reels of the master mode System Statistical Collection tape or User Journal File. The file must be in system standard format with the exception of block size, which is 1602 words.

DIRECTIVE OPTIONS

FUNCTION

LOAD	Causes specified pages to be written on the mass storage device.
PRINT	Causes specified pages to be written in print format and directed to SYSOUT via file code P*.
LPRINT PLOAD	These options (either form is acceptable) cause specified pages to be written on the mass storage device and to be sent to SYSOUT in print format via file code P*. Either directive is a combination of the PRINT and LOAD functions.
EOR	Forces a unit switch on the input magnetic tape file.
NORWD	Suppresses rewinding of the input tape at the end of each directive.

DESCRIPTOR OPTIONS

RNG/P1,P2/	Specifies the page range to be loaded and/or printed If no range is present, the entire range of all subfiles allocated is assumed. The argument P2 must be greater than or equal to P1.
SNUMB/XXXXX/,ACT/A1,A2/	This selects page images for the specified SNUMB starting with activity A1 through activity A2. If only activity A1 is specified, that is the only activity to be selected. If no activity is specified, all page images for the SNUMB are looked at.
FILE/FILENAME/	This selects page images associated with the specified filename. If this option is used, it must be the only directive to be processed.
PAGE/1,2,3.../	This option provides selection of specific pages. When this option is used, a FILE/FILENAME/ must be specified. A maximum of 10 pages may be specified on one directive.
DATE/YYMMDD/	This option, in conjunction with FILENAME or SNUMB, accomplishes selection of records with a date equal to or greater than the one specifed.
DATE/YYMMDD/,TIME/HHTTT/	This provides selection of records with a date and time equal to or greater than that specified.
AFTER BEFORE	Specifies either BEFORE or AFTER images are to be loaded. If neither option is specifed, BEFORE is assumed.

PRINT OPTIONS

TYPES/A,B,C,.../	Prints only the record types specified by A,B,C,... (to a maximum of 10 types).
DELETE	Produces a file containing reference code, size and record type of all records logically but not physically deleted from the file. The required file code is DE.

DIRECTIVE EXAMPLES

```
1      8             16
IDS    OPTION        JTAPE/,PLOAD/,SNUMB/12345/,
       ETC           ACT/05/
```

This loads and prints the first BEFORE images found on the journal tape for SNUMB 12345, activity 5. All nonempty pages and page headers for all empty pages will be printed on P*.

```
1      8             16
IDS    OPTION        JTAPE/,PLOAD/,FILE/IDSFOURYQUAD/,
       ETC           DATE/700608/,TIME/13.058/,AFTER/
```

This loads all AFTER page images found on the journal tape for SNUMB 12345, activity 5. All nonempty pages and page headers for all empty pages will be printed on P*.

```
1      8             16
IDS    OPTION        JTAPE/,PLOAD/,FILE/IDSFOURYQUAD/,
       ETC           DATE/700608/,TIME/13.058/,AFTER/
```

This loads all AFTER page images found on the journal tape tape for file IDSFOURYQUAD with a date and time equal to or greater than the one specified. All nonempty pages will be printed and all empty pages will be indicated.

Execution

An execution report is produced as part of the user output. It describes, in chronological order, the functions performed as specified in the directives. In addition, error conditions are included to advise the user of execution conditions.

Considerations

This type load utilizes a tape which will probably contain multiple before and after images for each page; therefore, when loading before images a control must be maintained to ensure that only the first before image of each page is written to the data base. This is accomplished through utilization of a page-flag "bit-buffer".

In order to allow dynamic construction of the bit-buffer at execution time, the amount of core required for the bit buffer is based on the accumulated ranges of all subfiles allocated to the job and the accumulated ranges specified on the directive cards. It is the user's responsibility to provide enough core to accommodate this requirement.

The minimum core requirement for this version of QUTL (excluding the bit-buffer) is 15K. A formula for calculating the bit buffer size per subfile or range is described below:

```
(MAX. RANGE - MIN. RANGE +1)*PAGES-PER-PAGE/36
              = Number of words of core required per subfile or range
```

Total core required would be the sum of all subfile or range computations plus 15K.

EXAMPLE

A program aborts leaving two subfiles to be recovered. One subfile has a page-range of 1 - 10000 while the second subfile contains pages 10001 - 20000 for a total of 20,000 pages. If no range is specified on the QUTL directives, enough core (556 words) must be allocated to construct a bit buffer large enough to map 2000 pages. However, suppose the determination can be made, based on knowledge of the aborted program, that only pages 9000 - 12000 were affected, this range may then be specified on the directive and only enough core (84 words) to map 3000 pages would be required.

Multiple directives may be processed with one execution of QUTL; however, they will be processed in the order in which they appear in the job stream. Consider the following example:

1	8	16
IDS	OPTION	JTAPE/,LOAD/,SNUMB/12345/,ACT/2/
IDS	OPTION	JTAPE/,LOAD/,SNUMB/23456/,ACT/2,5/

All before images for SNUMB/12345/,ACT/2/ will be looked at on the first pass of the accounting tape. The bit buffer will be checked to ensure only the first before image of each page is written to the data base.

The Journal tape will be rewound and the second directive, SNUMB/23456/ activities two through five will be processed. The page flag bits set by the first directive will be checked while processing the second directive, so that should each job have changes the same pages in the data base, only the first before image written by the first directive is restored. At completion of the QUTL activity, the data base would be restored to a point prior to any changes made by either job.

The user may suppress rewinding of the accounting tape between directives by specifying NORWD on the directive cards. This option should be used when the jobs to be recovered were run in sequence rather than concurrently.

Operation

The following deck setup can be used to execute QUTL from the software library:

Example for permanent files:

```
1      8          16
$      IDENT      (options)
$      PROGRAM     QUTL
$      LIMITS     (options)  (minimum 15K + bit buffer)
$      USERID     IDSFOURYQUAD$DBASE
$      PRMFL      TF,R/W,R,IDSFOURYQUAD$DBASE/QUAD01
$      PRMFL      TG,RECOVERY/R/W,R,IDSFOURYQUAD$DBASE/QUAD02
$      TAPE       IN,X2S,,1234,,JOURNAL-TAPE (Required File Code)
$      DATA       I*
IDS    OPTION     JTAPE/,LOAD/,SNUMB/56789/,ACT/01,05/,NORWD/
IDS    OPTION     JTAPE/,LOAD/SNUMB/567890/,ACT/02/
$      ENDJOB
```

This deck setup will result in the following:

1. The first before image of each page associated with SNUMB 56789 activity 1 through activity 5 found on the journal tape will be written to the I-D-S DATA BASE IDSFOURYQUAD.

2. The NORWD directive prevents rewinding the journal tape prior to processing the next directive.

3. The first before image of each page associated with SNUMB 567890 activity 2, not reloaded when processing the first directive, will be reloaded on file IDSFOURYQUAD.

Printer Format

The format of pages selected for printer output is shown below:

```
PAGE:    XXXXX XX ACTIVE PAGE SIZE: XXXX CH.

WD:    LN:    TYPE:
XXX    XX     XXXX   [OCTAL]   [OCTAL]   [OCTAL]   [OCTAL]   [BCD]
                     [OCTAL]   [OCTAL]   [OCTAL]   [OCTAL]   [BCD]
                     [OCTAL]   [OCTAL]   [OCTAL]   [OCTAL]   [BCD]
XX     XX     XXXX   [OCTAL]   [OCTAL]   [OCTAL]   [OCTAL]   [BCD]

PAGE:    XXXXX XX ACTIVE PAGE SIZE:  XXXX CH.         PAGE EMPTY
         AND ALL INTERVENING PAGES

PAGE     XXXXX XX ACTIVE PAGE SIZE:  XXXX CH.       PAGE EMPTY
```

Input File Format

The data read from the input tape file consists of variable length, logical records. The file is in standard system format with the exception of block size which is 1602 words. The record format is:

<u>Word</u>	<u>Contents</u>
0	Accounting Record Header. The number of data words in the record is specified in bits 0-17. The record type, octal 000013, is contained in bits 18-35.
1	Checksum.
2	SNUMB in bits 0-29. Ignore character (octal 17) in bits 30-35.
3	Date as MMDDYY.
4	Start time in hours and thousanths of hours as HH.TTT.
5	Record type in bits 0-11 as 10. Bits 12-35 are presently unused and are zero.
6	This word is presently unused and is zero.
7	First six characters of user identification.

Word	Contents
8	Second six characters of user identification.
9-n	Active page image.

Delete File FORMAT

The optional output file of records logically but not physically deleted from the data base is in standard system format. The record format is:

Word	Contents
0	Reference code. Page number in bits 12 - 29. Line number in bits 30 - 35.
1	Record type.
2	Record size in characters.

QUTP

Journal Record Selector Utility Program (QUTP)

QUTP selects records from an I-D-S journal tape according to user-supplied criteria and writes them on an output tape.

Directives

Two types of control cards are recognized by QUTP: SELECT and ETC. The first card must be a SELECT; the second is optional.

```
1      8         16
IDS    SELECT    f1,f2,f3
```

where f1 must be AA/SSSSS. This field specifies the Activity and SNUMB of the corresponding Slave Begin record which must be found to initiate interrogation of this criterion. The AA/SSSSS format must be one or two digits for the activity number, slash, and five digits for the SNUMB.

f2 must be AA/SSSSS. This field specifies the Activity and SNUMB (which must be the same SNUMB as in f1) of the corresponding Slave End record which must be found to terminate interrogation of this criterion.

f3 is either B, A, or null. This field specifies the type of record to be selected for interrogation. If this field is null, B is implied. If ETC cards are present, this field is ignored. B stands for Before and A stands for After.

Two SELECT card examples are:

```
1      8         16
IDS    SELECT    1/53607,1/53607,B
IDS    SELECT    12/88802,13/88802,A
```

The ETC card is optional. It is used to specify that only BEFORE or AFTER records for a given page range are to be selected for output. The format of this directive is:

```
1      8         16
       ETC       f1/f2/f3,f1/f2/f3,  ...etc...
```

where f1 is B or A meaning BEFORE or AFTER.

f2 is the lower limit of a page range.

f3 is the upper limit of a page range.

NOTE: $1 \leq f2 \leq f3 \leq 262,143$

Several page range specifications may be placed on one ETC card, but each triplet must be separated from the next one by a comma. A slash must separate each element of a triplet. If several page range specifications are placed on one ETC card, the last data character must be followed by a blank, and the blank must appear prior to or in column 72. Several ETC cards may follow a SELECT card as long as the maximum of 8 triplets per SELECT is not exceeded.

If ETC card(s) follow a SELECT, then field f3 of the SELECT card is ignored since this option is specific for each page range.

Two ETC card examples are:

```
1      8      16
       ETC    B/129/352,A/26243/53409
       ETC    A/1/100,A/10/20
```

Directive Restrictions

A maximum of 50 directives is allowed. Following each SELECT directive, there may be a maximum of eight page range specifications.

Tape Format

The input data for this program can be one or more reels of master mode journal tape information. The file must be in standard system format with the exception of block size, which is 1602 words.

Records are written on the output file in standard system format with the exception of block size. Two types of records may appear on the output tape: BEFORE and AFTER. Their format is:

Word	Contents
0	Accounting Record Header. The number of data words in the record is specified in bits 0-17. The record type, octal 000013, is contained in bits 18-35.
1	Checksum.
2	SNUMB in bits 0-29. Ignore character (octal 17) in bits 30-35.
3	Date as MMDDYY.
4	Start time in hours and thousandths of hours as HH.TTT.
5	Record type in bits 0-11 as 10. Bits 12-35 are presently unused and are zero.
6	Lines per page for this page image (bits 1-17).
7	First six characters of user identification.
8	Second six characters of user identification.
9-n	Active page image.

Execution Report

A detailed execution report is printed by QUTP. The report is divided into two parts. Part 1 is a listing of the directives and part two is the summary report.

Operation

1. Deck setup.

   The following deck setup can be used to execute QUTP.

```
1      8        16
$      IDENT
$      PROGRAM  QUTP
$      LIMITS   Options
$      TAPE     IN,Options   (IN is required file
                          code for the input tape)
$      TAPE     OT,Options   (OT is required file
                          code for the output tape)
            Directives
$      ENDJOB
***EOF
```

2. QUTP performs three distinct functions to select the specified records.

PROCESSING DIRECTIVES. The directives are read as data from the input file I*. Each card is checked for errors in both content and format. If errors are present, an error comment is written with the card image on the execution report; and a switch is set so that a D2 abort occurs when all directives have been scanned, but before processing of the input tape is initiated. When scanning is complete and no error has occurred, a sequence number is assigned to the directive and printed on the execution report. The criterion is then stored in memory. Since all criteria are resident in core, the user need not order them.

RECORD INTERROGATION. As each input record is read from tape, its record type is examined to determine how it should be handled.

The SLVBGN and SLVEND records are used to initiate and terminate testing on a criterion. For example, if a criterion specifies all BEFORE records within a specific SLVBGN-SLVEND, the criterion is turned on when the matching SLVBGN is encountered to interrogate BEFORE records and output those that match. Correspondingly, the matching SLVEND turns off the criterion. This technique allows inactive criteria to be quickly recognized and bypassed.

The BEFORE and AFTER records are matched against specific criteria. If the tests are successful, the records are written.

SUMMARY REPORT. After all criteria are satisfied or an end-of-file is reached on the input file, a summary report is produced on the execution report. Specific criteria of each directive and the number of output records for the directive are shown.

Execution Information Report Program (QUTR)

QUTR selects type B information records from an I-D-S journal file, sorts the records, and produces an execution information report.

Input Tape Format

The input file is standard system format except for a maximum block size of 1602 words.

Operation

1. Subroutine .QSTB.

   For each SNUMB-activity that engages subroutine .QSTB, type B statistics are collected on the I-D-S journal file as a type 09 record. These are the records used as input by QUTR. Thus, to provide this input, the following loader control card must be included in the job stack for the activity:

1	8	16
$	USE	.QSTB

   Type B information is then accumulated by .QSTB for each primary entry subroutine (that is, each subroutine called by the object program).

2. Deck Setup.

The following deck setup shows the appropriate control cards for (1) collecting type B statistics on the journal file and (2) executing QUTR.

```
       1      8         16
 (1)   $      IDENT
       $      USERID
 (2)   $      USE       .QSTB
       $      OBJECT
               .
 (3)           .
               .
       $      DKEND
       $      EXECUTE
       $      PRMFL
 (4)   $      TAPE      JX,X1S,,,,I-D-S JOURNAL
               .
 (5)           .
               .
 (6)   $      PROGRAM   QUTR
 (7)   $      SYSOUT    P1
 (8)   $      TAPE      A1,X1R
 (9)   $      TAPE      B1,X2R,,99999
(10)   $      NTAPE     S1,T,2
       $      ENDJOB
       ***EOF
```

Notes:

(1) Beginning of activity.

(2) Provides for collecting type B information in type 09 journal records.

(3) Object deck.

(4) User-created journal file.

(5) Other user files (and end of activity).

(6) Beginning of second activity (for producing a type B statistics report).

(7) P1 is required output file code.

(8) A1 is required input journal tape file code. (Note that this is the journal file created in first activity.)

(9) B1 is required file code for sort work file (scratch tape).

(10) S1 is required file code for the first of two collation tapes needed by GE-600 Line Sort/Merge.

Sample Output

A sample output for QUTR is shown on the following page. The circled callouts are keyed to the following notes:

1. Alter number (from GMAP codes) of the call to the subroutine
2. Function (similar to I-D-S statement)
3. Record type
4. Record type of chain master followed by record type of a detail
5. Number of times the call was executed
6. Number of times the subroutine was executed without requiring I/O
7. Total number of reads for execution of the subroutine
8. Total number of writes for execution of the subroutine

ALTER	FUNCTION		CALLS	ZERO I/O	READS	WRITES
135	STORE RECORD TYPE	990	1	0	1	0
166	RETRIEVE RECORD TYPE	990	1	1	0	0
169	RETRIEVE NEXT OF CHAIN	990 4	21	3	17	18
179	DELETE RECORD TYPE	4	20	2	0	18
184	RETRIEVE NEXT OF CHAIN	990 50	20	19	0	6
215	RETRIEVE DIRECT		19	0	19	19
217	RETRIEVE NEXT OF CHAIN	1000 4	19	19	0	0
282	STORE RECORD TYPE	1	5	0	6	4
291	STORE RECORD TYPE	2	5	0	6	5
300	STORE RECORD TYPE	3	5	1	4	4
309	STORE RECORD TYPE	4	5	0	6	5
410	STORE RECORD TYPE	50	20	20	0	0
1	2	3	5	6	7	8

Selected Record Sort Utility Program (QUTS)

QUTS sorts and merges records selected from an I-D-S journal tape. The sorted and merged records may be used to reload the user data base when recovery to a previous file status is desired.

Input Tape Format

The input file is standard system format with the exception of block size, which is 1602 words. The data on the input file must have been written as output by the I-D-S Journal Record Selector (QUTP); therefore, it must consist of BEFORE and AFTER record types only.

Output Tape Format

The output files are standard system format with the exception of block size, which is 1602 words. The data on the output files consists of the first BEFORE or last AFTER record for each page supplied as input.

Execution Report

QUTS produces an execution report as part of the user output. This report describes in chronological order the functions performed during the execution. In addition, error messages are included to advise the user of exception conditions.

Operation

1. Deck setup.

   The following deck setup describes the appropriate control cards for executing QUTS using tapes. Disc sort may also be used instead of tapes.

1	8	16
$	IDENT	
$	PROGRAM	QUTS
$	LIMITS	10,24K
$	TAPE	IN,Options
$	TAPE	OT,Options
$	TAPE	OU,Options
$	NTAPE	S1,Options,3
$	ENDJOB	
***EOF		

A limit card is required. The minimum is 19K, however for sort to run with greater efficiency a limit of at least 24K is suggested.

IN is the required input file code.

OT is the required file code for the first output file.

OU is the required file code for the second output file. (This file need not be present if the input to QUTS consists only of records from a single file; that is, one file name.)

S1 is the required file code for the first of three collation tapes required by GE-600 Line Sort/Merge. A minimum of three collation tapes is required.

2. QUTS consists of input coding and output coding elements coupled to the standard GE-600 Line Sort/Merge. The individual functions performed are described below.

INPUT CODING. The input coding element reads and preprocesses all input records from the input file:

A sequence number is placed in bits 0-17 of the seventh word of all BEFORE and AFTER records. For each BEFORE record, the sequence number is ascending and ranges in value from 1 to 777777(8). For each AFTER record, the sequence number is descending and ranges in value from 777777(8) to 1. This sequence number preserves the chronological order of the input records in cases where start times may be identical for two different activities.

Each input record is tested to ensure that only record types 05 and 06 comprise the input. Invalid records are dumped in octal format on the execution report accompanied by an appropriate error comment. An indicator is set when invalid records are encountered so that the program terminates with a D2 code after all input records are processed.

SORT CODING. The standard GE-600 Line Sort/Merge is used to arrange input records in the desired order for output. The fields used for sorting and their sequence are:

Sequence	Field Size	Field Description
1st (major key)	2 words	File-name
2nd	18 bits	Page number
3rd	10 bits	Page
4th	2 characters	Record type
5th	1 word	Sequence number
6th	24 bits	CALC chain next

OUTPUT CODING. Two files are available for output in this coding element. A control break on file-name results in closing the first output file and opening the second output file. The specific functions performed in the output coding element are:

a. The sequence number in bits 0-17 of the sixth word is set to zero.

b. The page number of the current record is compared to the page number of the previous record and, if they are the same, the current record is not written to the output file.

QUTT

QUTT Not Available

QUTT Tape Conversion Utility Program is no longer available.

File Utility Program (QUTU)

QUTU performs the following I-D-S utility functions, depending upon the directives chosen:

- File initialize (INIT directive): establishes page headers and initializes inventory.
- File print/graph (PRINT directive): prints requested pages, record types, and inventory; graphs space and lines used for requested pages; and prints a record type usage report.
- File movement (WRITE directive): moves requested pages from one file to another. (This is a DUMP/LOAD facility.)
- File reformat (WRITE directive): changes page size and/or lines per page of requested pages while performing file movement to a tape or random file.

Directives

Directive fields begin in column 16 and are separated by commas. One or more ETC cards may be used to continue the fields if they run beyond column 72. Each card to be continued must end with a complete field, followed by a comma. A directive card followed by one ETC card is shown below.

```
1      8        16
IDS    INIT     FC/XX/,RNG/A,B/,RNG/C,D/,
       ETC      RNG/E,F/,RNG/G,H/,...
```

Formats for the program input directives are shown below, arranged by program function. Directive restrictions are listed at the conclusion of the format explanations.

Function 1: File Initialize (random files only; if file is tape, directive is ignored)

```
1      8        16
IDS    INIT     FC/XX/,RNG/A,B/,RNG/C,D/,...
```

where FC/XX/ is the file to be initialized. This field must be present.

For permanent random files: XX is as defined on the $PRMFL card.

For temporary random files: XX is A1 for the first file, A2 for the second, etc.

RNG/A,B/ is a page range to be initialized.
If no range field is present, the entire range of the file is initialized. A must be less than or equal to B, and B must be less than or equal to 262,144.

RNG/C,D/, if present, is a second page range to be initialized. A maximum of 8 ranges will be considered on one directive.

Example for permanent files:

```
1      8       16
$      SNUMB
$      IDENT
$      PROGRAM QUTU
$      LIMITS  10,24K
$      USERID  IDSFOURYQUAD$DBASE
$      PRMFL   TF,R/W,R,IDSFOURYQUAD/QUAD01
$      PRMFL   TG,R/W,R,IDSFOURYQUAD/QUAD02
IDS    INIT    FC/TF/,RNG/1,120/
IDS    INIT    FC/TG/,RNG/121,240/
```

Example for temporary files:

```
1      8       16
$      SNUMB
$      IDENT
$      PROGRAM QUTU
$      LIMITS  10,24K
$      MASS    A1,X1S,11R
$      MASS    A2,X2S,22R
$      DATA    .Q
IDS    CREATE  FC/A1/,BSSZ/480/,RNG/1,120/,LPP/63/
IDS    CREATE  FC/A2/,BSSZ/480/,RNG/121,240/,LPP/32/
$      DATA    I*
IDS    INIT    FC/A1/
IDS    INIT    FC/A2/
```

Function 2: File Print/Graph

```
1      8       16
IDS    PRINT   FC/XX/,RNG/A,B/,RNG/C,D/,...,
                  print option
```

where FC/XX/ is the file to be printed; and
RNG/A,B/ RNG/C,D/ are the ranges of that file to be printed.

Example for permanent files:

```
1      8       16
$      SNUMB
$      IDENT
$      PROGRAM QUTU
$      LIMITS  10,24K
$      USERID  IDSFOURYQUAD$DBASE
$      PRMFL   TF,R/W,R,IDSFOURYQUAD/QUAD01
$      PRMFL   TG,R/W,R,IDSFOURYQUAD/QUAD02
$      DATA    I*
IDS    PRINT   FC/TF/,RNG/1,10/,PAGES
IDS    PRINT   FC/TG/,EMPTY
```

Example for temporary files:

```
1      8       16
$      SNUMB
$      IDENT
$      PROGRAM QUTU
$      LIMITS  10,24K
$      MASS    A1,X1R,11R
$      MASS    A2,X2R,22R
$      DATA    .Q
IDS    CREATE  FC/A1/,BSSZ/480/,RNG/1,120/,LPP/63/
IDS    CREATE  FC/A2/,BSSZ/480/,RNG/121,240/,LPP/32/
$      DATA    I*
IDS    PRINT   FC/A1/
IDS    PRINT   FC/A2/,RNG/121,150/,GRAPH
```

The print options and their resulting actions are as follows (each option generates a different report code to prevent report "shuffling" on SYSOUT; only one option is allowed per PRINT directive but a maximum of 8 PRINT directives is allowed):

NULL — Results in the same action as PAGES (see below).

EMPTY — Prints nonempty pages and prints a line for each empty page rather than indicating a succession of empty pages only by a first-page entry and a last-page entry. An inventory printout is included.

GRAPH	Prints a graph showing, for each page, the percent of space used and number of lines used in the page.
GRAPH/N/	Prints a graph showing, for each N pages and/or pagettes, the average percent of space used and average number of lines used per page. Note: Use caution in interpreting averages that include two different page sizes.
INV	Prints inventory only.
PAGES	Prints nonempty pages, indicates empty pages, and prints inventory.
RECORD	Prints a report of record types usage within each of the specified ranges. Because of the large buffer space required, a record type usage report cannot be generated for both the input and output file over the same range. If two reports are requested, the second request is ignored.
TYPES/A,B,C,.../	Prints only the record types specified by A,B,C, etc. (to a maximum of 8 types).

Function 3: File Movement/Reformat

```
1      8         16
IDS    WRITE     FC/XX/,RNG/A,B/,RNG/C,D/,...,
                     ONFC/YY/,PAGE/SZ,LPP/
```

where	FC/XX/	is the file to be read; and
	RNG/A,B/ RNG/C,D/	are the ranges of that file.
	ONFC/YY/	is the file to be written. For temporary random files, YY must be A1,A2, etc.
	PAGE/SZ,LPP/	indicates the reformatting parameters for a tape output file. (This field is not used if the output file is random. Page format on random output files is defined by the file attributes.) SZ is page size in words. LPP is lines per page. If PAGE is present, both parameters must be present. If the output file is tape and the PAGE field is not present, the output format will be the same as the input format.

Example for permanent files:

```
1      8       16
$      SNUMB
$      IDENT
$      USERID  IDSFOURYQUAD$DBASE
$      PROGRAM QUTU
$      LIMITS  10,24K
$      PRMFL   TF,R/W,R,IDSFOURYQUAD/QUAD01
$      PRMFL   TG,R/W,R,IDSFOURYQUAD/QUAD02
$      TAPE    DT,X1S
$      DATA    I*
IDS    WRITE   FC/TF/,ONFC/DT/
IDS    WRITE   FC/TG/,ONFC/DT/

               And the reloading of the files from
                  the dump tape
$      PROGRAM QUTU
$      LIMITS  10,24K
$      PRMFL   TF,R/W,R,IDSFOURYQUAD/QUAD01
$      PRMFL   TG,R/W,R,IDSFOURYQUAD/QUAD02
$      TAPE    DT,X1D
$      DATA    I*
IDS    WRITE   FC/DT/,ONFC/TF/
IDS    WRITE   FC/DT/,ONFC/TG/
```

Example of initialize, execute, dump, and reload for temporary files:

```
1      8        16
$      SNUMB
$      IDENT
$      PROGRAM  QUTU
$      LIMITS   10,24K
$      MASS     A1,X1S,11R
$      MASS     A2,X2S,22R
$      DATA     .Q
IDS    CREATE   FC/A1/,BSSZ/480/,RNG/1,120/,LPP/63/
IDS    CREATE   FC/A2/,BSSZ/480/,RNG/121,240/,LPP/32/
$      DATA     I*
IDS    INIT     FC/A1/
IDS    INIT     FC/A2/
```

User's Program to be executed with its required control cards

```
   $      PROGRAM  QUTU
   $      LIMITS   10,24K
   $      TAPE     DT,X6S
** $      MASS     A1,X2S,22R
   $      DATA     .Q
** IDS    CREATE   FC/A1/,BSSZ/480/,RNG/121,240/,LPP/32/
   $      DATA     I*
** IDS    WRITE    FC/A1/,RNG/121,240/,ONFC/DT/
```

And the reloading of that file from the dump tape

```
   $      PROGRAM  QUTU
   $      LIMITS   10,24K
** $      MASS     A1,X2R,22R
   $      TAPE     DT,X6R
   $      DATA     .Q
** IDS    CREATE   FC/A1/,BSSZ/480/,RNG/121,240/,LPP/32/
   $      DATA     I*
** IDS    WRITE    FC/DT/,RNG/121,240/,ONFC/A1/
```

**NOTE: FC/A1/ is used to reference the file which in the first activity was created and defined as FC/A2/.

Directive Restrictions

Besides the restrictions included in the discussions of the various directives, the following apply:

1. A maximum of 8 INIT, 8 PRINT, and 8 WRITE directives will be processed.

2. A maximum of 8 RNG fields will be considered on any one directive.

3. A maximum of 8 record types will be considered on any one PRINT directive TYPES field.

4. When a range is defined on a directive, the input file must contain that entire range with pages in sequence.

5. If no RNG field is present on a PRINT or WRITE directive, any pages found on the input file that can be written to SYSOUT (with PRINT) or the output file (with WRITE) will be handled.

6. Only SDL-1 (and later) dump format tapes will be read and written.

7. 24,000 words of core storage are necessary for program execution.

8. Any output tape files must contain ranges which do not require writing on one tape, then on another and then on the first again. For example, the following is legal:

Ranges	Tapes
1 - 100	1
200 - 1000	
1001 - 1100	2
1700 - 1800	

The following is illegal:

Ranges	Tapes
1 - 100	1
200 - 1000	
101 - 150	2

9. Since subroutine OPEN is used, all rules defined by OPEN for overlapping ranges, etc., hold for this utility, if a random file is involved.

10. Only SDL-2 (and later) sorted journal tapes may be processed.

Printer Format

The printer output formats for the PRINT options are described below and illustrated in the "Sample Outputs" section. The circled numerals refer to the corresponding callouts on the sample outputs (Figures 38-41).

PAGES option (see Figure 38):

1 PAGE xxxxxx xxxxxx: page number in octal, then decimal.

2 xx LINES xx USED: total number of lines existing on this page, number of these used; both in decimal.

3 SIZE, CHAR; USED xxxx, AVAIL xxxx: number of characters used, number still available; both in decimal. (Sum of these is size in number of characters.)

4 BEGINNING LINE NUMBER xx: beginning line number of page.

5 CALC CHAIN NEXT xxxxxxxx: contents of the CALC chain NEXT field (octal).

6 LN xx xx: line number in octal, then in decimal.

7 TP xxxx: data record type in decimal.

8 SZ xxxx: record size in characters (decimal).

9 W+xxx: number of words from beginning of the page.
Cxx : beginning character in the word W+xxx.

10 xxxxxxxxxx: octal control word, equivalent to 6, 7, and 8.

11 Contents of line defined in 6, 7, 8, and 10.

12 Octal data.

13 BCD data, equivalent to octal data on same printed line.

14 Same information as in 6, 7, 8, 9 and 10 for next line.

Notes:

1. If a page is empty, only the following appears:

   PAGE xxxxxx xxxxxx PAGE EMPTY

   If two or more succeeding pages are empty, the following appears after the line shown above:

   THRU

   PAGE xxxxxx xxxxxx PAGE EMPTY

2. Selecting the PAGES option also causes an inventory for the range (as shown for the INV option) to be printed by SYSOUT.

EMPTY option:

The output for this option is the same as for PAGES (including an inventory), except that for each empty page - that is, succeeding pages as well as single ones - the following appears:

PAGE xxxxxx xxxxxx PAGE EMPTY

TYPES option:

The output for this option is the same as for PAGES, except for the following:

1. No inventory is included.
2. Only the selected record types are printed.
3. If a page contains records but none are of the requested types, the following appears:

   PAGE xxxxxx xxxxxx NO REQUESTED RECORD TYPES

INV option (see Figure 39):

1 PAGE

xxxxxx xxxxxx : Page number in octal, then decimal

2 LINE

xx xx : Line number in octal, then decimal

3 #AVAIL

xxx : Percent of space available, shown in either of two ways:

a. When the space used in the page is less than the percentage specified in the user's inventory update request, this condition is indicated by a #AVAIL of xx (where xx is 100# minus the inventory update request). Thus, this indication shows only that the inventory update request percentage has not been exceeded.

b. When the space used in the page is greater than the percentage specified in the user's inventory update request, this condition is indicated by a #AVAIL xx (where xx is the actual percentage of total space that is still available).

4 THRU : Indicates that, for the pages from the page and line preceding this word through the page and line following, the AVAIL is the same.

RECORD option (see Figure 40):

1 RECORD TYPE

xxx : I-D-S record type.

2 SIZE

xxx : Record size in characters. (The size is flagged by an asterisk if it is inconsistent.)

3 NUMBER

xxxx : Number of occurrences of record type within specified range.

4 NUMBER DELETED

xxxx : Number of this record type logically deleted within the specified range.

5 LOW PAGE

xxxx : Page number of first occurrence of record type within the specified range.

6 HIGH PAGE

xxxx : Page number of last occurrence of record type within the specified range.

7 RANGE

xxxx - xxxx : Specified range for report.

GRAPH and GRAPH/N/ options (see Figure 41):

1 Page numbers.

2 Scale for percent of space used (0 - 100).

3 Scale for number of lines used (0 - 63).

4 # character, showing percent of space used.

5 # character, showing number of lines used.

6 X character, used when # and # values coincide.

Notes:

1. For GRAPH/N/, the numbers of the pages at interval N appear in the column at 1 . The symbols opposite these numbers represent averages for the percent of space used and number of lines used within the interval.

2. Multiple entries for the same page number can occur if GRAPH (rather than GRAPH/N/) is specified when pagettes are included.

Tape Format

The data sent to the output tape file is written as variable length, logical records using the GEFRC subroutine PUT. The file is in standard system format with the exception of block size which is 1602 words. The Page Image record format is:

Word	Contents
0	Accounting Record Header. The number of data words in the record is specified in bits 0-17. The record type, octal 000013, is contained in bits 18-35.
1	Checksum.
2	SNUMB in bits 0-29. Ignore character (octal 17) in bits 30-35.
3	Date as MMDDYY.
4	Start time in hours and thousandths of hours as HH.TTT.
5	Record type in bits 0-11 as 10. Bits 12-35 are presently unused and are zero.
6	UTL in bits 0-17 to indicate utility tape rather than journal tape. Bits 18-35 unused.
7	First six characters of user identification.
8	Second six characters of user identification.
9-n	Active page image.

Execution Report

An execution report is produced as part of the user output. It includes open and close reports for any random files used (see examples 1 and 2, respectively, in Chapter 6, "I-D-S Execution Report") and a list of the directives used in order of execution. (See Figure 42 in "Sample Outputs" section for example of directive list.)

The report may also include any of the following error messages (all but no. 9 describe conditions causing a program abort):

1. FILE CODE XX RANGE REQUESTED NOT IN FILE

   A range has been defined by a directive for a random file (XX) which is inconsistent with the range defined in the file attributes.

2. FILE CODE XX CANNOT HANDLE REDUNDANT RANGE

   Range had been defined by a directive for this file (XX) with an intervening range requested by another file (see "Directive Restrictions," no. 9).

3. CANNOT HANDLE MORE THAN 8 FILES

   More than 8 directives of any one type (INIT, PRINT, WRITE) have been input.

4. PREVIOUS CARD FATAL ERROR

   The preceding card contains an error -- no file code, missing comma, or missing slash.

5. PREVIOUS CARD TOO MANY RANGES

   More than eight ranges are defined on the preceding directive.

6. PREVIOUS CARD TOO MANY RECORD TYPES

   More than eight record types are defined on the preceding directive.

7. FILE CODE XX INPUT IS NOT SEQUENTIAL

   The file (XX) does not contain all of the pages defined by a following range field.

8. FCXX ON FCXX PAGE TOO SMALL

   In reformatting, the output page size is not large enough to contain the lines to be written in the page.

9. FILE CODE XX CHECKSUM ERROR

   A checksum error has been found on file XX. This is noted but the program does not abort. (See also "Directive Restrictions," no. 7.)

10. FILE CODE XX DATA READ NOT PAGE OR PAGETTE

    The file XX does not contain page images. This is the result of a bad tape or bad random file.

Operation

The following deck setup can be used to execute QUTU.

```
1      8       16
$      IDENT
$      USERID
$      PROGRAM QUTU
$      LIMITS  10,24000
$      PRMFL
$      DISC
$      DRUM or Options
$      MASS
$      TAPE    Options
         .
         .
         .  Directives
         .
         .
         .
$      ENDJOB
***EOF
```

Sample Outputs

Figures 38-42 on the following pages show the various sample outputs mentioned in the "Printer Format" and "Execution Report" sections.

```
     (1)                   (2)                 (3)                       (4)                       (5)
PAGE    000003 000003, 32 LINES 32 USED  SIZE,CHAR; USED 1415,AVAIL  505  BEGINNING LINE NUMBER  1  CALC CHAIN NEXT 00000301
LN 01 01,TP 0001,SZ 0203,W+  3,C 4    0100010313  000003100000 000006061103 020031244325 511462202631   00380000669320IDLER@S FI
                                                  254324202020 202020202020 202020010745 031103060400   ELD             17N393640
  (6)      (7)     (8)     (9)          (10)      450100020302 030366000503 244500040105 004500014545   N1023233W053DN0415ON01NN
                                                  212145450005 030000000000 060527516227 433121622120   AANN05300000065GRSGLIASA
                                                  202045010607 642020202045 200304076420 202020452045     N167U    N 347U     N N   (11)
                                                  204520204520 202020000001 254520202020 202020202020    N   N     001EN
                                                  202020202020 202020202020 202020202020 204520202020                        N
                                                  202020202020 202020204520 202020202020 202020202020                N
                                                  202100000302                                           A0032
LN 02 02,TP 0026,SZ 0014,W+ 37,C 3    0200320016  000500050300 000303                                   050530033
                                                                        (12)                                  (13)
```

Figure 38. Sample PAGES Option Output

```
    PAGE         LINE   %AVAIL          PAGE         LINE   %AVAIL           PAGE         LINE   %AVAIL

000001 000001   01 01     0        000001 000001   41 33   >75 THRU     000003 000003   00 00   >75
000003 000003   01 01     0        000003 000003   41 33   >75          000004 000004   01 01     0
000004 000004   41 33   >75 THRU   000007 000007   00 00   >75          000007 000007   01 01     0
000010 000008   00 00     0        000010 000008   01 01   >75 THRU     000015 000013   00 00   >75
000015 000013   01 01     0        000015 000013   41 33   >75 THRU     000017 000015   00 00   >75
000017 000015   01 01     0        000017 000015   41 33   >75          000020 000016   01 01     0
000021 000017   40 32    02        000021 000017   41 33   >75 THRU     000025 000021   00 00   >75

     (1)          (2)    (3b)                                (3a) (4)
```

Figure 39. Sample INV Option Output

```
USAGE    RECORD TYPE    SIZE    NUMBER    NUMBER DELETED    LOW PAGE  HIGH PAGE  RANGE        1 -    50

              2          139      377           0               1        50
             (1)         (2)      (3)          (4)             (5)       (6)            (7)
```

Figure 40. Sample RECORD Option Output

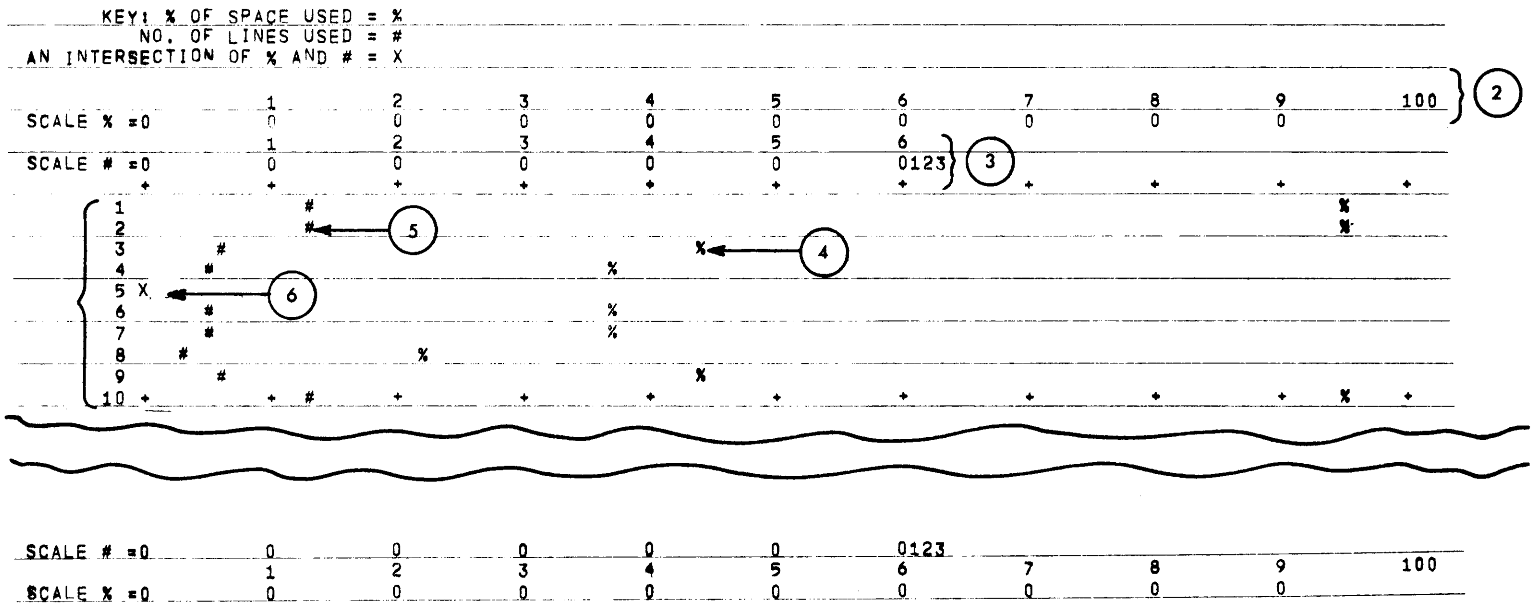


Figure 41. Sample GRAPH Option Output

```
41632 01  08-12-69   14.778

DIRECTIVE:          PRINT   FC/T2/,RNG/1,50/
DIRECTIVE:          PRINT   FC/T2/,RNG/1,50/,GRAPH
DIRECTIVE:          PRINT   FC/T2/,RNG/1,50/,RECORD
```

Figure 42. Sample QUTU Execution Report Directive List

Directive Processor and Service Subroutine (.QDIR)

.QDIR is a collection of ten different subroutines designed to provide common functions for I-D-S utility programs and subroutines. Each different function is defined by its SYMDEF name:

.QDIRF

This symbol identifies word -4 of the file control block for the data file. Bits 24-35 of this word contain the file code for the data file. If the user wishes to use his own file code, then he must initialize these bits prior to any call to .QDIR or .QSFD. The assumed directive file code is I*.

.QDIR

This subroutine opens the file for directives and reads the directive into memory. Columns 8-13 are left justified and stored in a cell pointed to by the user in the calling sequence. This value is also returned to the user in the A-register.

As each directive is read from the data file, columns 1 through 84 (14 words) are moved to a working buffer. The literal words DIRECTIVE: precede this buffer. After the move is completed, the .QMEX subroutine is called to print the literal and the card image on the execution report. A slew to the next line is given with each line of printing.

In addition, a tally word is initialized to point to column 16 of the directive for scanning the variable field through calls to the .QSFD entry point.

ETC cards are also read by this subroutine.

The calling sequence is:

```
1      8       16
       CALL    .QDIR(ARG1)ALT1
```

where:

ARG1 = The location for the contents of columns 8-13 of the directive.

ALT1 = The location for an end of file exit.

.QSFD

This entry point is called to scan the variable field of a directive starting in column 16. Each call to this entry point will scan a maximum of 12 characters, if a delimiter is not encountered. The valid delimiters are comma, blank, and slash.

The n characters are returned left justified with trailing blanks in the AQ-register as well as being returned to the three cells pointed to by the user in the calling sequence. The delimiter character is not returned.

The third word pointed to by the calling sequence will contain three values:

Bits 0-17: The number of characters in the subfield.

Bits 18-23: The delimiter character found.

Bits 24-35: The value required for a right shift of the AQ-register in order to right justify the subfield.

It should be noted that if the value in 0-17 is zero, then the value in bits 24-35 will be 72.

If more than 12 characters are present in the subfield then only the first 12 characters are returned to the user. The tally word for the scan is advanced through the next delimiter. The character count in bits 0-17 of the user's argument will contain the total number of characters in the subfield.

The calling sequence is:

1	8	16
	CALL	.QSFD(ARG1)

where:

ARG1 = The address of three consecutive cells for return information. The first two cells will contain the subfield, left justified, with trailing blanks. The third word will contain the three values described above.

.QDIRC

This subroutine closes the directive file. If only one file code is used with .QDIR, the user need not call this subroutine.

.QBCD

This subroutine converts a number from binary to BCD and replaces leading zeros with blanks. The number to be converted may not be larger than 999,999(10). If the binary number is zero, it will be converted to five blanks and a zero.

The calling sequence is:

```
1      8          16
       LDA        BINARY
       CALL       .QBCD
```

(value returned in the Q-register)

.QCLR

This subroutine clears n words to a preset value. The argument list specifies the number of words to be cleared, the address of the area to be cleared, and a pointer to the value to be stored in the area.

The calling sequence is:

```
1      8          16
       CALL       .QCLR(15,BUF,=6Hbbbbbb)
```

.QCSM

This subroutine calculates the checksum of a specified number of words starting at a given location. If the starting location is given as A, then the word at A+1 will be skipped (not added into the checksum).

The calculated checksum is returned to the user in the A-register.

The calling sequence is:

```
1      8          16
       CALL       .QCSM(ARG1,ARG2)
```

where:

ARG1 = The number of words to be checksummed.

ARG2 = Address of word 0 of data to be checksummed.

Note: If the number of words to be checksummed is 0, 1, or 2, then the first word of data is returned to the user as the checksum.

.QMCH

This subroutine moves n characters from address A, starting character position A1, to address B, starting character position B1.

The calling sequence is:

```
1      8          16
       CALL       .QMCH(15,BUFA,3,BFRB,0)
```

Starting character positions must be from 0 through 5.

.QMEX

This subroutine is called to write messages on the execution report. Messages must be less than or equal to 22 words in length. If a length of zero is given, a line of blanks will be written and the specified slew code will be appended to the end of the line. Messages greater than 22 words in length will be truncated to 22 words.

The calling sequence is:

```
1      8          16
       CALL       .QMEX(ARG1,MSG,SLEW)
```

where:

ARG1 = The number of words in the message

MSG = The address of the message

SLEW = The number of lines to be slewed after printing. (See the GEFRC routines, IOEDIT and PRINT, for slew code rules.)

If a fourth argument is present (the value in the argument has no bearing) in the call, then a 'Top of Page' will be issued prior to printing the line requested by the caller. After the top of page is issued, a heading line is printed with the SNUMB, activity number and date followed by a double space. Then the caller's line is printed. If the caller wants just a top of page without any information printed, he should write the call as:

```
1      8          16
       CALL       .QMEX(0,0,SLEW,0)
```

The zero word count in the above printed call will cause a line of blanks to be printed with the slew code specified.

.QMWD

This subroutine moves n words from address A to address B.

The calling sequence is:

```
1      8          16
       CALL       .QMWD(15,BUF1,BUF2)
```

.QPBK

This subroutine is called to journalize a page. The page will be sent to the user's journal file if it is present. If no user's file is present, then the page will be journalized to the I-D-S system journal.

The calling sequence is:

```
1      8          16
       CALL       .QPBK(FCB,PTR)
```

where:

FCB = The LOCSYM of the file control block for the journal file.

PTR = The address of a word which points to the origin of the data to be journalized. This origin is the location of the accounting header word which precedes the page.

The record size for journalization is obtained from bits 0-17 of the accounting header word.

It is the user's responsibility to checksum the record and store the checksum in the record prior to calling .QPBK. The file control block defined by the user for the journal file must specify only one buffer.

Trace and Print Record, Debug, and Utility Subroutine (.QSTC)

The .QSTC subroutine generates a trace entry for all calls to I-D-S primary subroutines (except for .QOPEN and .QCLOSE). In addition, each time a call is issued to one of the following primary subroutines, .QSTC prints the current record:

.QSTOR	.QCHN
.QGET	.QHEAD
.QGETC	.QMDFY
.QGETD	.QMOVE
.QGETE	.QDELETE

The trace data and the record to be printed are directed to P* unless otherwise specified by the user. The user can direct this output to his own file, if desired.

Trace data and print record entries are generated on P* or a users' file for all I-D-S record types, for each I-D-S primary subroutine (those listed above), and for the entire page range of the I-D-S file unless otherwise specified by the user. The user has the option of selecting:

1. Which primary subroutine(s) should be traced.
2. Which record(s) should be printed.
3. Up to five different page ranges within the I-D-S file.
4. Which record types (up to a maximum of 50) should be traced or printed.

The .QSTC subroutine is controlled through the following I-D-S Directive.

```
1      8        16
IDS    OPTION   DEBUG OPTIONS, FILE OPTION,
       ETC      DIRECTIVE/OPTION/,
       ETC      DIRECTIVE/OPTION/,
       ETC      ...
```

DEBUG OPTIONS

- PRTREC   This Debug Option causes the contents of the current record to be printed after the completion of an I-D-S call. Sample output is shown in Figure 43.
- TRACE   This Debug Option causes a trace data line to be generated each time one of the previously listed I-D-S primary subroutines is called. Sample output is shown in Figure 43.

Note: Either PRTREC or TRACE or both PRTREC and TRACE must be specified.

FILE OPTION

- ONFC/xx/ The inclusion of ONFC/xx/ causes the trace data and/or the output generated as a result of PRTREC to be directed to the users' file with the file code xx. If ONFC/xx/ is not included, the output is directed to P*.

DIRECTIVES

- NULL Provides the full capabilities of the option specified.
- ALL Provides the full capabilities of the option specified.
- DO Only the specified options will be performed.
- DONTDO Processing of the specified options is inhibited.

OPTIONS

- TYPES/nnn,...,nnn/

  Depending on the specified directive, this option allows or inhibits the tracing and/or printing of specified record types. A maximum of 50 different record types can be specified.

- VERBS/xxx,...,xxx/

  Depending on the specified directive, this option allows or inhibits the tracing and/or printing of the current record as a result of a call to an I-D-S function. The allowable verbs are:

  RETRIEVE
  RETRIEVEEACH (or EACH)
  RETRIEVENEXT (or NEXT)
  RETRIEVECURRENT (or CURRENT)
  RETRIEVEDIRECT (or DIRECT)
  HEAD
  STORE
  MODIFY
  MOVE
  DELETE

- RNG/1B,1E,...,5B,5E/

  Depending on the specified directive, this option allows or inhibits the tracing and/or printing of current I-D-S records that are within a specified page range. 1B...5B specify beginning page numbers; 1E...5E specify ending page numbers. A maximum of five different page ranges may be specified.

RESTRICTIONS

- For the same option, specification of a DONTDO directive overrides the specification of a DO directive.
- Both the DONTDO and the DO directives apply to both the TRACE and PRTREC functions.
- Imbedded blanks cause the processing of an OPTION card to be terminated.

EXAMPLES

```
1      8       16
IDS    OPTION  TRACE
```

This causes a trace data line to be generated on P* each time one of the previously listed I-D-S primary subroutines is called.

```
1      8       16
IDS    OPTION  PRTREC,ALL
```

This causes a print record entry on P* for all I-D-S record types and for each I-D-S primary subroutine (those previously listed) for the entire range of the I-D-S file.

```
1      8       16
IDS    OPTION  TRACE,DO/TYPES/001,942/,
       ETC     DONTDO/VERBS/MOVE,STORE/
```

This causes the tracing of only the record types 001 and 942 for the entire range of the I-D-S file and inhibits tracing of the I-D-S verbs MOVE and STORE for those record types. The output is directed to P*.

```
1      8       16
IDS    OPTION  TRACE,PRTREC,ONFC/AB/,
       ETC     DO/RNG/001,005,009,010/,
       ETC     ALL/TYPES/,DO/VERBS/,
       ETC     RETRIEVE,MODIFY,DELETE/
```

This causes the tracing and printing of all record types referred to by the verbs RETRIEVE, MODIFY, and DELETE that are within the page ranges 001 to 005 and 009 to 010. The trace data and print record are directed to the user's file with the code AB.

```
1      8        16
$      DATA     .Q
IDS    OPTION   PRTREC,DO/TYPES/941/,
       ETC      DO/VERBS/RETRIEVE/,
       ETC      DO/RNG/016,900/
```

This example causes all record type 941 (name and address record) to be printed each time a RETRIEVE verb accesses a record type 941 between pages 016 and 900. The output is directed to P*.

DECK SETUPS

The following deck setup may be used to execute on an I-D-S PRMFL.

```
1      8        16
$      IDENT    IDSTST,PAT
$      USERID   IDSFOURYQUAD$DATABASE
       OBJECT PROGRAM
$      USE      .QSTC
$      EXECUTE
$      PRMFL    A1,R/W,R,IDSFOURYQUAD/QUAD1
$      DATA     .Q
IDS    OPTION   PRTREC,TRACE,ALL
$      END JOB
***EOF
```

The following deck setup may be used to execute a program using a temporary I-D-S file.

```
1      8        16
$      IDENT    IDSTST,PAT
       OBJECT PROGRAM
$      USE      .QSTC
$      EXECUTE
$      DISC     A1,X1R,9R
$      TAPE     B1,X2D
$      DATA     .Q
IDS    CREATE   FC/A1/,BSSZ/100/,RNG/1,100/
IDS    OPTION   TRACE,ONFC/B1/,DONTDO/RNG/1,50/
$      ENDJOB
***EOF
```

Note: To provide the TRACE and PRTREC options the following LOADER control card must be included in the job stack for the activity.

```
1      8        16
$      USE      .QSTC
```

USER ENTRY POINT

A users' entry point has been provided which enables the printing of the current I-D-S record. This entry is available to the user regardless of whether PRTREC or TRACE has been specified.

```
1      8        16
       SYMREF   QSTA4
       CALL     QSTA4(ARG);
```

where ARG is a one word working storage location to be used as a line count.

STANDARD ERROR OPTION

If TRACE is specified by the user and an I-D-S error occurs, an error message is generated to the output file code specified by the user. Refer to Figure 43 for example.

SUBROUTINE RESTRICTIONS

1. If the user entry point (QSTA4) is called, all output generated is directed to P*.

2. If any field within an I-D-S record exceeds 84 characters, only the first 84 will be printed by the PRTREC module. If a field is modified, the PRTREC module shows the result of the entire field.

3. If the modify verb is called to modify a record with more than 100 fields, modify flags appear on the first 100 fields modified; all others are not flagged.

4. If this subroutine is used with a user program which has not been compiled using the .QNAMS macro, the field and record name areas of all output will contain unpredictable data. This condition will also cause faulty printing in some cases.

   To include this macro, the user must include the following code within the Procedure Division after the first ENTER IDS. statement.

   ```
   14       22
   ENTER    GMAP .
   .QNAMS
   ENTER    COBOL .
   ```

5. If ONFC/XX/ file option is used and the specified file is not defined as having variable-length records or if file is not assigned as a printed file, the results are unpredictable.

6. If ONFC/XX/ file option is used and the specified file is not opened before the first I-D-S statement, all output is directed to P*.

OUTPUT DESCRIPTION

Figure 43 shows typical TRACE and PRTREC output. A description of all generated data fields follows:

1 Complete trace entry

2 Complete PRTREC entry showing fields names and field content

3 TRACE heading

4 GMAP alter number within program where I-D-S call was issued

5 Current type of I-D-S operation

6 Current record type

7 Page and line number of current I-D-S record

8 PRTREC header shows type of I-D-S operation, record name, and page and line number of current I-D-S record

9 Field-name of record

10 Field contents

11 Control field - this field shows field usage, allowable contents are:

RDM - Randomize field key
STA - Sorted ascending field key
STD - Sorted descending field key
MAT - Match key or synonym field

12 Data Type:

AN - Alphanumeric
A - Alpha
N - Numeric
SN - Signed Numeric
SFX - Single precision, fixed point
SFP - Single precision, floating point
DFX - Double precision, fixed point
DFP - Double precision, floating point

Note: All field contents that are not BCD will be printed in OCTAL.

13 I-D-S error entry with error code

14 An (*) will appear by each field name which had its contents changed by the user calling the I-D-S modify routine

```
66536  05  09-29-69      21.173     4              5                      6                 7

*****  IDS-TRACE * ALTER NO.-       293     CTYPE-STOR        RTYPE-     993       PG/LN=     120/ 2 }  1
            3                       DATREC       MAT CALDAT              STA ACTDAT                      CODDAT
STOR DATDET                120/ 2   0992              9998                  999999                       4
*****  IDS-TRACE * ALTER NO.-       293     CTYPE-STOR        RTYPE-     993       PG/LN-     120/3
                8                   DATREC       MAT CALDAT  9           STA ACTDAT                      CODDAT }  2
STOR DATDET                120/ 3   0992       10  9999                     999999                       5      }
***** IDS-TRACE * ALTER NO.-        263     CTYPE-STOR        RTYPE-     993       PG/LN-     120/ 3
XXXXX AN IDS ERROR HAS OCCURED, ERROR CODE - D01 }  13
                                                                            11
                                    DATREC       MAT CALDAT              STA ACTDAT                      CODDAT
STOR DATDET                120/ 3   0992              9999                  999999                       5
***** IDS-TRACE * ALTER NO.-        282     CTYPE-GET         RTYPE-     992       PG/LN-     120/ 1
                                    DATREC       RDM X1                   X3                             WKGWK          YYY
GET DATMAT                 120/ 1   0992              0000               00000006             00005                     0068
                                    BASE2             YYYMON                PERMAX        12             FSTMON
                                    0001              0001                  07                SFX        0001
***** IDS-TRACE * ALTER NO.-        303     CTYPE-CHN         RTYPE-     993       PG/LN-     120/ 2
                                    DATREC       MAT CALDAT              STA ACTDAT                      CODDAT
CHN  DATDET                120/ 2   0992              9999                  999999                       5
***** IDS-TRACE * ALTER NO.-        319     CTYPE-MDFY        RTYPE-     993       PG/LN-     120/ 2                14
                                    DATREC       MAT CALDAT              STA ACTDAT                      CODDAT      *
MDFY DATDET                120/ 2   0992              9998                  999999                       4
```

Figure 43. Sample .QSTC Output

Verify and Print Utility Subroutine (.QUTF)

QUTF verifies the integrity of a page and formats and prints I-D-S data base information received from QUTU, QUTL or QUTD.

The calling sequences are:

```
1      8          16
       CALL       .QUTF1(ARG1,ARG2)
```

This entry point must be called first to initialize .QUTF. ARG1 is the name of the file control block to which the dump output is sent. Normally, this is the file control block for SYSOUT, P*.

ARG2 is the symbolic location of four words that are included in the title line of the dump output. Normally, it is name and version of QUTU, which produces the output from QUTD, QUTL, or QUTU.

```
1      8          16
       CALL       .QVFY(ARG1,ARG2,ARG3)ARG4
```

This entry point is called for each page that is to be verified.

ARG1 is the location of a word which contains the 24 bit page reference code, right justified.

ARG2 is the address of the first word of the page to be verified.

ARG3 is the location of a word which specifies whether the page will be dumped on the printer via SYSOUT. If the word is zero, no printing will be performed. If the word is nonzero, the page will be printed. In addition, if this word is nonzero it must be preceded by and followed by two words of zeroes.

ARG4 is the location of the user's alternate exit which is taken whenever a page cannot be verified.

The following checks are performed to verify the integrity of a page:

- The page number supplied by the caller (ARG1) equals the page number in the input record.
- Every line present in the page has its line flag properly set.
- The sum of record sizes equals the active page size.
- Line flags are not set for lines that are not contained in the page.

If any one of these tests fails, an appropriate error message is written on the execution report followed by a snapshot of the page in error. The registers shown in the panel portion of the snapshot dump display the following information:

X0 The page number supplied by the caller in argument 1.

X1 The current word address within the page where processing was being performed when the error occurred.

X2 The current character position in the word described by index register 1.

X3 The usable page size expressed in characters.

X4 The available space expressed in characters.

X5 The active page size expressed in characters.

X6 The number of characters in the page which have already been processed.

X7 The number of the current line being processed.

AR/QR The current status of the available line flags, left justified. These flags are taken from working storage and some bits may not be present for those lines already processed.

As each page is verified, secondary entry points of .QUTF are called to format and print the page, if required. These entry points are described below:

1	8	16
	CALL	.QUTF2(ARG1,ARG2)

This entry point is called at the beginning of each page. It supplies information to .QUTF concerning the page number and the active page size.

ARG1 is the location of the page number, in binary, right justified.

ARG2 is the location of the active page size, in binary, right justified.

```
1      8       16
       CALL    .QUTF3(ARG1,ARG2,ARG3)
```

The third entry point is called to print each line. ARG1 is the location of the word number within the page for this line. The value is binary, right justified.

ARG2 is the location of a tally word containing the address and starting character position of the line.

ARG3 is the location of an indicator. If ARG3 is 0, the Page Header (line 0) is sent for printing. If ARG3 is ≠0 and negative, a normal line is sent for printing.

Printer Format

The format of pages selected for printer output is shown below:

```
PAGE:  XXXXX XX    ACTIVE PAGE SIZE:  XXXX  CH.

WD:  LN:  TYPE:

XXX  XX   XXXX   [OCTAL]  [OCTAL]  [OCTAL]  [OCTAL]  [BCD]
                 [OCTAL]  [OCTAL]  [OCTAL]  [OCTAL]  [BCD]
                 [OCTAL]  [OCTAL]  [OCTAL]  [OCTAL]  [BCD]
XXX  XX   XXXX   [OCTAL]  [OCTAL]  [OCTAL]  [OCTAL]  [BCD]
```

Execution Report

The output produced by the .QUTF subroutine is written on the file provided by the user in his call to .QUTF1. The report code is 25(10). Output is produced by calling the PRINT and EPRINT subroutines of GEFRC. The GEFRC subroutine IOEDIT is used for page numbering and format control.

Operation

QUTF is used by QUTD and QUTL to print data base pages in the desired format. It is made available to the utility routines from the subroutine library through the use of the SYMREF feature of GMAP. QUTF is not freestanding and cannot be called except through the user's own program.

Sample Output

An I-D-S Selective Tape Dump report using .QUTF is shown on the following page.

100	34	50	4200620021	000000000003	000100017125		K0S0A0000030101ZE
103	35	50	430062002100	000000060700	0100017206		L0S0A0000670101←6
105	36	50	44	006200210000	000103010001	00017177	M0S0A0001310101Z!
108	37	50	4500	620021000000	010402000100	017227	N0S0A0001420101←G
111	38	50	460062	002100000001	040000010001	7145	O0S0A0001400101ZN
114	39	50	47006200	210000000101	020001000172	25	P0S0A0001120101←E
117	40	50	5000620021	000000000505	000100017215		Q0S0A0000550101←!
120	41	50	510062002100	000004020500	0100017156		R0S0A0004250101Z;
122	42	50	52	006200210000	000207030001	00017175	-0S0A0002730101Z=
125	43	50	5300	620021000000	030206000100	017232	$0S0A0003260101←&
128	44	50	540062	002100000003	050500010001	7162	*0S0A0003550101ZS
131	45	50	55006200	210000000403	000001000172	31	)0S0A0004300101←!
134	46	50	5600620021	000000040210	000100017155		(0S0A0004280101Z)
137	47	50	570062002100	000001060000	0100017201		'0S0A0001600101←1
139	48	50	60	006200210000	000307060001	00017221	+0S0A0003760101←A
142	49	50	6100	620021000000	040701000100	017222	/0S0A0004710101←B
145	50	50	620062	002100000003	060100010001	7133	S0S0A0003610101Z.
148	51	50	63006200	210000000206	020001000172	41	T0S0A0002620101←J
151	52	50	6400620021	000000021101	000100017170		U0S0A0002910101ZY
154	53	50	650062002100	000002030600	0100017104		V0S0A0002360101Z4
156	54	50	66	006200210000	000301020001	00017225	W0S0A0003120101←F
159	55	50	6700	620021000000	030403000100	017237	X0S0A0003430101←\
162	56	50	700062	002100000002	110700010001	7106	Y0S0A0002970101Z6
165	57	50	71006200	210000000111	100001000171	30	Z0S0A0001980101ZH
168	58	50	7200620021	000000020207	000100017165		←0S0A0002270101ZV
171	59	50	730062002100	000000040400	0100017103		,0S0A0000440101Z3
173	60	50	74	006200210000	000204100001	00017113	%0S0A0002480101Z#
176	61	50	7500	620021000000	020711000100	017164	=0S0A0002790101ZU
179	62	50	760062	002100000001	061100010001	7132	"0S0A0001690101Z&
182	63	50	77006200	210000000103	040001000171	27	!0S0A0001340101ZG

PAGE: 122- 1 ACTIVE PAGE SIZE: 617 CHARACTERS

WD:	LN:	TYPE:					
0	0	1000	000172775000	017200242777	777777777700	00000000	01←!Q01←0DG!!!!!!00000
3	1	50	0100	620021000000	010603000100	017176	10S0A0001630101Z"
6	2	50	020062	002100000001	100400010001	7242	20S0A0001840101←K
9	3	50	03006200	210000000201	050001000171	72	30S0A0002150101Z←
12	4	50	0400620021	000000010005	000100017147		40S0A0001050101ZP
15	5	50	050062002100	000002011100	0100017203		50S0A0002090101←3
17	6	50	06	006200210000	000007000001	00017125	60S0A0000700101ZF
20	7	50	0700	620021000000	001111000100	017140	70S0A0000990101Z'
23	8	50	100062	002100000001	020000010001	7144	80S0A0001200101ZM
26	9	50	11006200	210000000105	010001000171	57	90S0A0001510101Z'
29	10	50	1200620021	000000000401	000100017173		[0S0A0000410101Z,
32	11	50	130062002100	000000010700	0100017243		#0S0A0000170101←L
34	12	50	14	006200210000	000003050001	00017137	@0S0A0000350101Z\
37	13	50	1500	620021000000	000506000100	017112	;0S0A0000560101Z[
40	14	50	160062	002100000000	100700010001	7207	>0S0A0000870101←7
43	15	50	17006200	210000000405	070001000171	31	?0S0A0004570101Z!
46	16	50	2000620021	000000030005	000100017124		0S0A0003050101ZD

Appendix A. Reserved Words

I-D-S RESERVED WORDS

I-D-S uses all the reserved words specified for COBOL. In addition, it employs the reserved words listed below. The user must avoid using words on both these lists for data-names.

ABORT	EACH	PROCESS
ALLOWED	ERROR-REFERENCE	RANDOMIZE
ANY	FIELD	REC-FILE
AUTHORITY	FIRST-REFERENCE	RECORD-TYPE
AUTHORITY-KEY	HEAD	REPLACE
BUFFER	IDS	RETRIEVAL
CALC	IDS-SPECIAL-NAMES	RETRIEVE
CCBLOC	INTERVAL	SORTED
CCBLOXK	LAST-REFERENCE	STORE
CHAIN	LINKED	SYN
CHAIN-ORDER	MASTER	SYNONYM
CURRENT	MATCH-KEY	TABLE
DEBUG	MD	TRACE
DELETE	MODIFY	UNIQUE
DIRECT	NEAR	UPDATE
DIRECT-REFERENCE	PAGE-RANGE	VIA
DUPLICATES	PRIOR	WITHIN
		WORKING

I-D-S GENERATED GMAP SYMBOLS

GMAP symbols defined in the location field must not conflict with reserved system symbols. (See GE-600 Line Programming Reference Manual, CPB-1004.) Symbols in the form LLNNNN, where L is any letter and N is a number, must not be defined in the location field of GMAP statements.

Appendix B. I-D-S Error Conditions

Two types of error conditions may occur during I-D-S program execution. The code, I-D-S source, and description for error conditions of both types are shown in the following sections.

DATA-DEPENDENT ERROR CONDITIONS

Testing for data-dependent error conditions must be incorporated in the procedural logic of the user program. Codes for this type of error are stored in the communication cell ERROR-REFERENCE for reference by the user program. The various codes are listed in the following table. With each code is shown the I-D-S source of the error condition and its description as printed by the TRACE option of the USE statement. This description will be printed if the TRACE option is selected. (See the USE description in Chapter 3 for an example of TRACE output.)

The key to abbreviations in the descriptions is shown below:

RT - record type	MT - master record type
REF - reference code	DT - detail record type
	XXXX - variable inserted by TRACE

Error Code	Source	Description from Trace
R01	QASC	No current record reference code record type XXXX
R02	QASC	Record retrieved logically deleted RTXXXX REFXXXXXXXX
R03	QASC	Request retrieval of record RTXXXX got RTXXXX
R04	QASC	No record on chain MTXXXX-DTXXXX or structure error for record type XXXX
R05	QGTC	Retrieve current, current equals zero rec-type XXXX

R06	QGTD	Retrieve direct and direct reference equals zero
R07	QGTD	Retrieve direct and record is logically deleted
R08	QMRA	Line number not on specified page ref code XXXXXXXX
R09	QBIC QSMT	Page requested is not allocated reference code XXXXXXXX
R10	QDLT	Illegal delete request of RTXXXX want RTXXXX
R11	QMDF	Illegal modify request of RTXXXX want RTXXXX
R12	QMNO QCAL	Working storage for page range zero record type XXXX
D01	QTLN	Store of unallowed duplicate record type XXXX
S01	QMNO	No space available for record type XXXX

ERROR CONDITIONS CAUSING ABORT

Improper use of I-D-S functions, invalid data file definition, and unrecoverable hardware malfunctions cause an automatic trace and abort of the user program. In addition, a memory dump occurs.

Whenever an I-D-S program aborts, the I-D-S data file is first CLOSED, with the appropriate pages restored to the mass storage device.

If the trace cannot acquire a link on mass storage for an overlay, the following error comment may occur:

CANNOT TRACE ERROR, INADEQUATE SPACE

The various abort reason codes are listed in the following table. With each code is shown the I-D-S source of the error condition and its trace description.

Note that while they are included in this table, codes 65 through 88 are not associated with an abort condition, but have been added solely to permit the printing of an appropriate error message while TRACE-ing the non-fatal errors discussed above. These codes may be encountered in a memory dump, or among the inner workings of the I-D-S subroutines, but will otherwise be invisible to the user.

The key to abbreviations in the descriptions is shown below:

RT - record type	MT - master record type
REF - reference code	DT - detail record type
CC - communication control	XXXX - variable inserted by TRACE

Reason Code	Source	Description from Trace
04	QAUT	Authority key does not match record type XXXX
05	QSMT	No records returned from sort
06	QRLN	Read error - check error reference in CC block
07	QRLN	Record retrieved logically deleted RTXXXX REFXXXXXXXX
08	QRLN	No position prior pointer chain MTXXXX - DTXXXX
09	QRLN	No detail definions for this chain MTXXXX
10	QASC	Retrieval via missing for record type XXXX
11	QASC QDLT QSTO	Detail in too many chains record type XXXX or/master of too many chains record type XXXX
12	QASC	No unique field for primary record - record type XXXX
13	QGTD QRLN	No record definition has been established
14	QHED	Chain next equal zero chain - MTXXXX-DTXXXX

*

15	QDLT QMDF QSTO	Processing mode not up-date
16	QMDF QMOV	Field of modify/move not in record type XXXX
17	QDLT	No current record of program on delete
18	QFWD	Retrieve next in chain no current exists MTXXXX-DTXXXX
19	QGDE	Invalid control definition record type XXXX
20	QGDE	Control field error, equals zero for record type XXXX
24	QSTO	No unique field on store for record type XXXX
25	QSTOR	No storage chain specified for record type XXXX
26	QTYP QRLN	Record retrieved not specified for chain MTXXXX-DTXXXX
27	QDLT	Delete action list is invalid
29	QUDC QRLN	No position next pointer chain MTXXXX-DTXXXX
30	QMRA	Record size conflict for record type XXXX
31	QSBF QSMT	Attempt to write not update, reference code XXXXXXXX
32	QBIC QSMT	Invalid page size for reference code XXXXXXXX
33	QIV3 QIV4 QFWD	Page requested is not allocated reference code XXXXXXXX
34	QIOS	Read/write error
35	QSMT QBIC	No empty buffer for REND

36	QRDN	Attempted update while in READ only mode
52	QTLN	Record cannot be linked chain MTXXXX-DTXXXX
53	QTLN	Error trying to retrieve prior chain MTXXXX-DTXXXX
54	QTLN	Error trying to retrieve next chain MTXXXX-DTXXXX
55	QTLN	Error trying to retrieve new chain MTXXXX-DTXXXX
56	QBIC	Page read is not page requested, reference code XXXXXXXX
57	QDLN	Next of chain is equal to zero chain MTXXXX-DTXXXX
58	QTLN	Attempt to link, next equals zero chain MTXXXX-DTXXXX
59	QIV3	Inventory read not one requested
60	QTLN QRLN	Next in chain not retrievable chain MTXXXX-DTXXXX
61	QOPE	Error in file definition at open time
65		(See Error Code "R01")
66		(See Error Code "R02")
67		(See Error Code "R03")
68		(See Error Code "R04")
69		(See Error Code "R05")
70		(See Error Code "R06")
71		(See Error Code "R07")
72		(See Error Code "R08")
73		(See Error Code "R09")
74		(See Error Code "R10")

75		(See Error Code "R11")
76		(See Error Code "R12")
80		(See Error Code "D01")
88		(See Error Code "S01")
129	QCHN QDBG QDLT QGET QGTC QGTD QGTE QHED QMDF QMOV QRLN QSTO	File unopened but access requested
130	QCHN QDBG QDLT QGET QGTC QGTD QGTE QHED QMDF QMOV QRLN QSTO	Primary subroutine entry during error processing
All others		Error code undefined

Rev. August 1971

Appendix C. GE-600 COBOL/I-D-S/FORTRAN Communication and Overlaying

This appendix explains the procedures and techniques to follow when overlaying a COBOL program, using the Integrated Data Store (I-D-S) software and mixing FORTRAN programs with COBOL or the COBOL/I-D-S software on a GE-600 system.

OVERLAYING A COBOL PROGRAM

Basis for Overlaying

Most programs should be segmented and overlayed when they become large. The memory allocated to a program will vary among sites. That is, some sites will have a billing formula to compute the cost of a computer run. If a particular computer run requires, for example, more than 40k of memory, the user's cost will have a very drastic increase after this limit has been reached.

In a multiprogramming system, the more memory required for a particular program decreases the effectiveness of the overall system. So, there is a justification for increasing the charge for a program when a set memory limit has been exceeded.

Many programs can be overlayed to reduce their memory requirements. These programs may have sections that are utilized only once or just a few times. These sections definitely do not have to reside in memory for the entire duration of a computer run. Other sections which do not have direct references to one another can be swapped in and out of memory, also under user control.

Segmentation

To accomplish overlaying, the program must be divided into subroutines, subprograms, or segments, whichever term you wish to choose. The term subprogram is used in this appendix. This program, when divided, will consist of numerous subprograms, each compiled separately or each appearing to be an entity or program.

Thus, each subprogram will be a separate COBOL compilation, each with an Identification Division, Environment Division, Data Division, and Procedure Division. Each program will have a uniqueness to depict that they are subprograms. These features, imbedded in the programs, are various transfers, entry points, exits, and common data storage areas.

Communication Between Subprograms

Once a subprogram exists, the means of communicating with the other subprograms (and also examining constant or variable data used in different subprograms) must be accomplished.

First, the method of passing constant or variable information between the subprograms. In COBOL, use the labeled common area method. These areas are defined in each subprogram that use any of the constant or variable information. The following example will show how to set up the labeled common areas so that the different subprograms can examine the same data.

<u>Subprogram MAIN</u>

```
000010 IDENTIFICATION DIVISION.
000020 PROGRAM-ID. MAIN.
              .
              .
000140 ENVIRONMENT DIVISION.
              .
              .
000180 SPECIAL-NAMES.
000190    BLOCK 31 IS ENTRY-REC THRU LAST-REC.
              .
              .
000320 DATA DIVISION.
000330 FILE SECTION.
              .
              .
000500 WORKING-STORAGE SECTION.
000510 01  ENTRY-REC.
000520     02  OTHER-LEVELS SIZE IS 48 NUMERIC.
000530 01  LAST-REC.
000540     02  MORE-LEVELS SIZE IS 42 NUMERIC.
```

Subprogram NEXTPG

```
000010 IDENTIFICATION DIVISION.
000020 PROGRAM-ID. NEXTPG.
              .
              .
000120 ENVIRONMENT DIVISION.
              .
              .
000150 SPECIAL-NAMES.
000160    BLOCK 31 IS REC-ENTRY THRU REC-LAST.
              .
              .
000400 DATA DIVISION.
000410 FILE SECTION.
              .
              .
000550 WORKING-STORAGE SECTION.
000560 01  REC-ENTRY.
000570     02  DATA-HERE SIZE IS 48 NUMERIC.
000580 01  REC-LAST.
000590     02  MORE-DATA SIZE IS 42 NUMERIC.
              .
              .
```

The preceding example shows the entries necessary for communication in the Environment Division and Data Division of two subprograms. The labeled common area is the same in both since Block 31 was mentioned, and the size of the 01 records is consistent.

At load time, one labeled common area called C31 (COBOL always prefixes the integer with the Character C) will be generated. The total size will be 90 characters. In subprogram MAIN, references to the common area (C31) will be by the name OTHER-LEVELS and MORE-LEVELS; whereas in subprogram NEXTPG, references to this same common area (C31) will be by the name DATA-HERE and MORE-DATA.

Since subprograms MAIN and NEXTPG are compiled separately, the names can be the same or different. The important concepts to remember from this example are that only one labeled common area (C31) will be generated when subprogram MAIN is loaded, and any subsequent subprogram referring to the identical area (C31) will have its references adjusted to this area.

When a COBOL program is divided into subprograms, transferring control during execution from one subprogram to another is done by using the CALL statement. If a return to the calling subprogram is desired, then the EXIT statement is used.

The following example shows the basic method of using the CALL and EXIT statements.

Subprogram SNOOPY

```
00001 010010 IDENTIFICATION DIVISION.
00002 010020 PROGRAM-ID. SNOOPY.
             .
             .
00149 040010 PROCEDURE DIVISION.
             .
             .
00191 041080    ENTER LINKAGE MODE.
00193 041100       CALL CHKSEG
             .
             .
```

Transfer is to the PROGRAM-ID whose location is the first executable statement in the PROCEDURE DIVISION of subprogram CHKSEG.

Subprogram CHKSEG

```
00001 010010 IDENTIFICATION DIVISION.
00002 010020 PROGRAM-ID. CHKSEG.
             .
             .
00092 040010 PROCEDURE DIVISION.
             .
             .
00104 043120 200-CALL-CK-END.
00105 043150     EXIT.
             .
             .
```

When EXIT is reached, execution returns to the next statement after CALL CHKSEG in subprogram SNOOPY.

NOTE: This example consists of excerpts taken from the program included with this appendix.

Transferring control to entries other than the PROGRAM-ID is accomplished by defining ENTRY POINTS in the program referred to. By using the ENTRY POINT statement, a SYMDEF is generated making entry possible at that particular point from any other subprogram.

When ENTRY POINT is written in a subprogram to return to the calling subprogram, the EXIT statement with the name of this ENTRY POINT is written.

The following illustrates the proper usage of the ENTRY POINT and EXIT.

```
$    FORTRAN
    COMMON/C20/ITABLE(20)
         .
         .
    CALL SNOOPY
         .
         .
    CALL ENTABC
         .
         .
```

The CALLs above are from a FORTRAN subprogram. They could have been from a COBOL subprogram which had ENTER LINKAGE MODE preceding each CALL.

- -

```
$    COBOL
00001 010010 IDENTIFICATION DIVISION.
00002 010020 PROGRAM-ID. SNOOPY.
                  .
                  .
00150 040022 PROCEDURE DIVISION.
                  .
                  .
00240 044014   ENTER LINKAGE MODE.
00241 044015       ENTRY POINT ENTABC.
00242 044016   ENTER COBOL.
                  .
                  .
00349 045261 309-PROGRAM-EXIT.
00350 045262     EXIT ENTABC.
00351 045263 310-SNOOPY-EXIT.
00352 045280     EXIT PROGRAM.
```

NOTE: This example consists of excerpts taken from the program included with this appendix.

Overlaying Procedure

The program is now chopped into many subprograms; each contains the necessary statements to refer to other subprograms.

To overlay, a few more statements have to be inserted into the subprograms. These statements are the CALL's to load specific subprograms from the H* file. In an overlay job, the overlays are not retained in memory but are stored on a peripheral to be loaded only when requested by the user. (See GE-600 Line General Loader, CPB-1008 for a complete explanation of the general overlaying method.)

There are two subroutines in the subroutine library (L*) to load the overlays. They are LINK and LLINK. When overlaying a program, the CALL LLINK loads the overlay and returns control to the statement following the CALL. The CALL LINK loads the overlay and returns control to the overlay. It is not possible to return to the statement following the CALL LINK after executing the overlay. Use the CALL LLINK so that you can retain control in a situation where a main subprogram will control transfer to an overlay brought into memory.

The following example illustrates the procedure to follow when overlaying a COBOL program. In the example, the subprogram SNOOPY resides in memory the duration of the execution, and subprograms CHKSEG, SAVSEG, and LOASEG are loaded into memory by the CALL LLINK statements located in SNOOPY.

```
$     SNUMB     12345
$     IDENT     HA963,ERICKSON
$     COBOL
010010 IDENTIFICATION DIVISION.
010020 PROGRAM-ID. SNOOPY.
                 .
                 .
020010 DATA DIVISION.
                 .
                 .
020300 WORKING-STORAGE SECTION.
020310 77    SEG-1  PICTURE X(6) VALUE IS "LINKAA".
020320 77    SEG-2  PICTURE X(6) VALUE IS "LINKBB".
020330 77    SEG-3  PICTURE X(6) VALUE IS "LINKCC".
                 .
                 .
040010 PROCEDURE DIVISION.
                 .
                 .
041080     ENTER LINKAGE MODE.
041090        CALL LLINK USING SEG-1
041100        CALL CHKSEG
                 .
                 .
041160     ENTER LINKAGE MODE.
041170        CALL LLINK USING SEG-3
041180        CALL LOASEG
                 .
                 .
042030     ENTER LINKAGE MODE.
042040        CALL LLINK USING SEG-2
042050        CALL SAVSEG
                 .
                 .
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

$     LINK     LINKAA
$     COBOL
010010 IDENTIFICATION DIVISION.
010020 PROGRAM-ID. CHKSEG.
                 .
                 .
040010 PROCEDURE DIVISION.
                 .
                 .
043120 200-CALL-CK-END.
043150     EXIT.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

```
$     LINK     LINKBB,LINKAA
$     COBOL
00001 010010 IDENTIFICATION DIVISION.
00002 010020 PROGRAM-ID. SAVSEG.
                      .
                      .
00091 040010 PROCEDURE DIVISION.
                      .
                      .
00103 043200 200-CALL-SA-END.
00104 043230     EXIT.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -


$     LINK     LINKCC,LINKBB
$     COBOL
00001 010010 IDENTIFICATION DIVISION.
00002 010020 PROGRAM-ID. LOASEG.
                      .
                      .
00091 040010 PROCEDURE DIVISION.
                      .
                      .
00103 043160 200-CALL-LO-END.
00104 043190     EXIT.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
```

NOTE: This example consists of excerpts taken from the program included with this appendix.

USING I-D-S WITH A COBOL OVERLAYED PROGRAM

Since the I-D-S statements are coded within the COBOL subprograms, there are certain procedures that must be considered.

A Communications Control Block (CCBLOC) must be established in a labeled common area. Normally, the CCBLOC is located in the COBOL program and the remaining structure is located in a labeled common area (.IDS...). When a program is divided into subprograms, each subprogram must be able to examine the CCBLOC. If it is isolated in the first subprogram loaded, then the remaining subprograms loaded will not be able to communicate with the CCBLOC.

To establish the CCBLOC in a labeled common area, write the following coding:

```
00014 010060 ENVIRONMENT DIVISION.

00018 010091 SPECIAL-NAME.
00023 010096    BLOCK nn is CCBLOXK.
                (nn is a 1 or 2 digit integer).
```

This is essentially the most important feature to realize when overlaying a COBOL/I-D-S program.

Another method to consider is placing the structure in a different labeled common area other than the .IDS.. area. Since the program is segmented, it is now possible to execute more than one I-D-S file. In this situation the first file must be closed before the second can be opened and executed. In other words, only one file can be in the open mode. The reason for this is that the page buffers for a file must be flushed before executing another file.

FORTRAN - INTERFACING WITH COBOL AND I-D-S

A FORTRAN program can easily communicate with the COBOL/I-D-S software. The knowledge that a FORTRAN user needs of COBOL is minimal, and if a FORTRAN user would like to utilize the I-D-S features, again the COBOL coding required and understanding can be minimal.

How to Communicate Between Compilers

Reiterating what was mentioned regarding COBOL segmentation--

1. Labeled common areas generated by the COBOL compiler are a one or two integer number always prefixed by the Letter C.

2. Entries into the Procedure Division can be made by referring to the PROGRAM-ID or ENTRY POINT name.

3. Return to the calling program is via the terminal EXIT statement of the EXIT name statement.

With these facts about the COBOL compiler, a FORTRAN user can create a program using these two together.

The FORTRAN subprogram contains labeled common areas corresponding to the COBOL areas. Variables and/or constants stored in these areas should have the same classification. That is, if a variable has been defined as floating point in one subprogram, it is defined as floating point in the other. Illustration of the above statements is depicted in the following examples with the addition of I-D-S.

```
$     SNUMB     24788
$     IDENT     HA963,ERICKSON
$     OPTION    FORTRAN
$     FORTRAN
    SUBROUTINE GENO
    COMMON/C35/IDATA,FLTNUM,IADD, -----
              .
              .
    CALL MAINPG
              .
              .
    CALL SECPRG
              .
              .
    END
$     ENTRY     GENO
$     USE       .QMAX/1/,.QAREA/2000/,.QMIN/1/
$     IDS
000010 IDENTIFICATION DIVISION.
000020 PROGRAM-ID. MAINPG.
              .
              .
020010 ENVIRONMENT DIVISION.
020020 SPECIAL-NAMES.
020030    BLOCK 20 IS CCBLOXK.
020040    BLOCK 35 IS ENTRY-REC THRU LAST-REC.
              .
              .
030100 IDS SECTION
030101 MD  IDS-PORTION PAGE CONTAINS 1920 CHARACTERS
030102     FILE CONTAINS 1000 PAGES.
030103 01  ENTRY-REC -----------
030104     02  DATA-HERE SIZE IS 9(6) COMPUTATIONAL-3.
030105     02  MORE-DATA SIZE IS 9(8) COMPUTATIONAL-2.
030106     02  ADD-MORE-DATA SIZE IS 9(6) COMPUTATIONAL-3.
              .
              .
030115 01  INNER-RECORDS ------------
              .
              .
030130 01  LAST-REC ------------
           02  THATS-ALL SIZE IS 9(7) COMPUTATIONAL-2.
              .
              .
              .
```

```
                .
                .
040010 PROCEDURE DIVISION.
                .
040200       GO TO MAIN-EXIT.
040201     ENTER LINKAGE MODE.
040202         ENTRY POINT SECPRG.
040203     ENTER IDS.
040204         RETRIEVE ENTRY-REC.
040205 BACK-TO-FORT.
040206     EXIT SECPRG.
040207 MAIN-EXIT.
040208     EXIT.
040209     END PROGRAM.
$      EXECUTE
$      DISC    DF,X2R,2R
$      DATA    .Q
IDS    CREATE  FILECODE/DF/,BASESIZE/1000/,RANGE/1,1000/,
       ETC     PAGESIZE/320/
$      ENDJOB
```

The contents of the labeled common area, C35, actually contain the I-D-S working storage area for all 02 levels beginning at record ENTRY-REC through record LAST-REC. Thus the equivalent values in C35 are the following:

FORTRAN	COBOL	TYPE
IDATA	DATA-HERE	Fixed Point Integer
FLTNUM	MORE-DATE	Floating Point
IADD	ADD-MORE-DATE	Fixed Point Integer

Consideration When Mixing Software

The 01 levels are always begun at an even memory location. Problems could occur when trying to pass information between COBOL and FORTRAN in the labeled common areas. To avoid this, check the labeled common size generated from the COBOL compiler and adjust the FORTRAN subprogram so that the correct data will be examined.

FORTRAN programs, at execution time, have file control blocks and buffers generated by the loader in the unused portion of slave memory. I-D-S checks word 37 (octal) of the slave prefix area to determine the size of unused memory and establishes as many page buffers as possible in this area. So now there is a major conflict of interest. Solution: At load time, create a labeled common area for the page buffers and other I-D-S control tables by inserting a $ USE control card. Now the FORTRAN I/O routines can use unused slave memory without conflict.

```
$$     86226 ENTERED 519609 AT 80,024 FROM CD RDR

0001  $      SNUMB    86226
0002  $      IDENT    HA963,RUDOLPH
0003  $      PROGRAM  QUTU             INITIALIZE TEMPORARY DATA BASE
0004  $      LIMITS   ,24K             CORE REQUIREMENTS OF QUTU
0005  $      DISC     A1,X1S,9R        TEMPORARY MASS STORAGE FILE
0006  $      DATA     ,Q               TEMPORARY I-D-S DATA FILE FOR DIRECTIVES
0007  $      DATA     I*               DATA STORAGE FILE
0008  $      OPTION   FORTRAN
0009  $      USE      ,QMAX/1/,,QAREA/1577/,,QMIN/1/
0010  $      OBJECT                                                F10;66102057000000000
0012  $      OBJECT   SDL-2-CHG00                                  I10;598020570SNOOPY00
0014  $      LINK     LINKAA
0015  $      OBJECT   SDL-2-CHG00                                  I14;805020570CHKSEG00
0017  $      LINK     LINKBB,LINKAA
0018  $      OBJECT   SDL-2-CHG00                                  I10;611020570SAVSEG00
0020  $      LINK     LINKCC,LINKBB
0021  $      OBJECT   SDL-2-CHG00                                  I10;619020570LOASEG00
0023  $      EXECUTE
0024  $      LIMITS   ,25K             INCREASE STORAGE SIZE
0025  $      DISC     H*,X2S,8R
0026  $      DISC     TF,X1S,9R        TEMPORARY MASS STORAGE FILE
0027  $      DATA     ,Q               TEMPORARY I-D-S DATA FILE FOR DIRECTIVES
0028  $      SYSOUT   PR               ASSIGN PRINTER TO OUTPUT MEDIA CONVERSION
0029  $      DATA     CR               TEMPORARY FILE FOR CARD INPUT
0030  $      ENDJOB
  TOTAL CARD COUNT THIS JOB = 000160


 * BEGIN ACTIVITY -01-  QUTU     02/06/70   SW=010000000000
  * NORMAL TERMINATION      AT 003710  INDICATORS  5020

      START  0,026     LINES     20     PROC  0,0001      I/O   0:000    IU 13   MEMORY    26K
      STOP   0,027     LIMIT   5000     LIMIT 0,0500     LIMIT            CU 13   M*T        53

      LAPSE  0,001    FC D TYPE      BUSY    IP/AT     FP/RT     IS/#C FS/#E    ADDRESS L#/T#

                      ,Q R MASS *     15       0         0         1     1     0-01-01  902
                      I* R MASS *     57       0         0         1     1     0-01-01  902
                      A1 S MASS      701       0         0         9     9R    0-01-01  904

       LIST    20  LINES


 * BEGIN ACTIVITY -02-  GELOAD   02/06/70   SW=000000000000
  * NORMAL TERMINATION      AT 046770  INDICATORS  1000

      START  0,027     LINES    543     PROC  0,0019      I/O   0:003    IU 13   MEMORY    27K
      STOP   0,033     LIMIT   5000     LIMIT 0,0500     LIMIT            CU 13   M*T       761

      LAPSE  0,006    FC D TYPE      BUSY    IP/AT     FP/RT     IS/#C FS/#E    ADDRESS L#/T#

                      TF D MASS      119       0         0         9     9R    0-01-01  904
                      ,Q R MASS *     38       0         0         1     1     0-01-01  903
                      CR R MASS *     40       0         0         1     1     0-01-01  903
                      R* R MASS *    380       0         0        12    12     0-01-01  902
                      H* D MASS     2694       0         0         8     8R    0-01-01  913
                      L* R MASS     4731       0         0        25    25R    0-01-01  325
```

```
85091 01  02-05-70   14.810

          COMMON/C20/ITABLE(24)/C21/ICK,ILO,ISA
      *
          CALL SNOOPY
      *
          CALL ENTABC                                                        2
      *
          ITOT=ICK+ILO+ISA                                                   3
          PRINT 11, ICK                                                      4
       11 FORMAT(42H NUMBER OF CHECKING ACCOUNT RECORDS READ *,I6)           7
          PRINT 12, ILO                                                      7
       12 FORMAT(30H NUMBER OF LOAN RECORDS READ *,I6)                      10
          PRINT 13, ISA                                                     10
       13 FORMAT(40H NUMBER OF SAVING ACCOUNT RECORDS READ *,I6)            13
          PRINT 15, ITOT                                                    13
       15 FORMAT(31H TOTAL NUMBER OF RECORDS READ *,I6)                     16
          STOP                                                              16
          END                                                               17

  23260 WORDS OF MEMORY USED BY THIS COMPILATION
```

NOTE: This is a FORTRAN program that was compiled illustrating just a labeled common area (C20) and two CALL statements which reference COBOL programs.

```
85091 01  02-05-70   14.811
        PREFACE

     PROGRAM BREAK   132
     COMMON LENGTH     0
     V COUNT BITS      5

   PRIMARY SYMDEF  ENTRY

      ......           0

 SECONDARY SYMDEF  ENTRY


      BLOCK       LENGTH

   1  C20             30
   2  C21              3

      SYMREF

   3  ENTABC
   4  .FCNV.
   5  .FEXIT
   6  .FFIL.
   7  .FPRN.
  10  SNOOPY
  END OF BINARY CARD 00000006
   132 IS THE NEXT AVAILABLE LOCATION. GMAP AID 051169
 THERE WERE    NO  WARNING FLAGS IN THE ABOVE ASSEMBLY
**  18715 WORDS OF MEMORY WERE USED BY GMAP FOR THIS ASSEMBLY.
```

IDS ALTER NOS.

```
010010 IDENTIFICATION DIVISION.
010020 PROGRAM-ID. SNOOPY.
010030 AUTHOR. GEORGE A RUDOLPH.
010040 DATE-WRITTEN. MAY 1969.
010050 INSTALLATION. G E - PHOENIX.
010051 REMARKS.  THIS PROGRAM LOADS DATA FROM THE CARD READER
010052          ONTO A TEMPORARY DISC FILE
010053             DEPENDING ON THE ACCOUNT TYPE ROUTINES ARE
010054             CALLED TO STORE THE DATA
010055                 WHEN ALL OF THE RECORDS HAVE BEEN
010056                 STORED ON THE DATA BASE THEY ARE
010057                 RETRIEVED AND PRINTED ON A CONTROL
010058                 REPORT.
010060 ENVIRONMENT DIVISION.
010070 CONFIGURATION SECTION.
010080 SOURCE-COMPUTER. GE-635.
010090 OBJECT-COMPUTER. GE-635.
010091 SPECIAL-NAMES.
010092     GETIME IS TODAYS-DATE.
010093*    DEFINES A LABLED COMMON AREA FOR THE IDS COMMUNICATION
010094*    CONTROL BLOCK AND RECORD DEFINATIONS FOR SEGMENTATION
           BLOCK 21 IS NUCK THRU NUSA.
010095     BLOCK 10 IS CCBLOXK.
010096     BLOCK 20 IS ENTRY-REC THRU LOAN-REC.
010200 INPUT-OUTPUT SECTION.
010210 FILE-CONTROL.
010215     SELECT PRINT-UNIT ASSIGN TO PR FOR LISTING.
010220     SELECT CARD-READER ASSIGN TO CR FOR CARDS.
010225*    ASSIGN IDS FILE NAME AND DEVICE
010230     SELECT IDS TEST-FILE ASSIGN TO TF.
010240 I-O-CONTROL.
010245     APPLY SYSTEM STANDARD FORMAT ON PRINT-UNIT.
010250     APPLY SYSTEM STANDARD FORMAT ON CARD-READER.
020010 DATA DIVISION.
020020 FILE SECTION.
020021 FD  PRINT-UNIT
020022     LABEL RECORDS ARE STANDARD
020023     DATA RECORD IS PRINT-LINE.
020024 01  PRINT-LINE                   PICTURE X(132).
020030 FD  CARD-READER
020040     LABEL RECORDS ARE STANDARD
020050     DATA RECORD IS CARD-IN.
020060 01  CARD-IN.
020070     02  ACCT-TYPE                PICTURE XX.
020080         88  LOAN-ACCT        VALUE "LO".
020090         88  SAVE-ACCT        VALUE "SA".
020200         88  CHECK-ACCT       VALUE "CK".
020210     02  CUST-NO-IN               PICTURE 9(6).
020220     02  ACCT-NO-IN               PICTURE 9(6).
020230     02  CUST-NAME-IN             PICTURE X(26).
```

```
86222 01  02-05-70   14.799   GE600 INTEGRATED STORE TRANSLATOR   ISDL-2 CHG00

IDS ALTER NOS.

020240     02  AMOUNT-IN                   PICTURE 9(10)V99.
020250     02  FILLER                      PICTURE X(31).
020300 WORKING-STORAGE SECTION.
020305*    CODING TO DEFINE THE LINKAGE FOR SEGMENTATION
020310 77  SEG-1   PICTURE X(6)    VALUE IS "LINKAA".
020320 77  SEG-2   PICTURE X(6)    VALUE IS "LINKBB".
020330 77  SEG-3   PICTURE X(6)    VALUE IS "LINKCC".
       77  NUCK PICTURE 9(6) COMP-1.
       77  NULO PICTURE 9(6) COMP-1.
       77  NUSA PICTURE 9(6) COMP-1.
020340 01  WORK-LINE.
020350     02  HEAD-ONE.
020360         03  FILLER                  PICTURE X(41).
020370         03  TITLE-1                 PICTURE X(49).
020380         03  FILLER                  PICTURE X(42).
020390     02  HEAD-TWO REDEFINES HEAD-ONE.
020400         03  FILLER                  PICTURE X(62).
020410         03  MONTH-P                 PICTURE Z9.
020420         03  DASH-1                  PICTURE X.
020430         03  DAY-P                   PICTURE Z9.
020440         03  DASH-2                  PICTURE X.
020450         03  YEAR-P                  PICTURE 99.
020460         03  FILLER                  PICTURE X(62).
020470     02  HEAD-THREE REDEFINES HEAD-TWO.
020480         03  FILLER                  PICTURE X(24).
020490         03  TITLE-10                PICTURE X(15).
020500         03  FILLER                  PICTURE X(8).
020510         03  TITLE-20                PICTURE X(15).
020520         03  FILLER                  PICTURE X(8).
020530         03  TITLE-30                PICTURE X(10).
020540         03  FILLER                  PICTURE X(14).
020550         03  TITLE-40                PICTURE X(6).
020560         03  FILLER                  PICTURE X(32).
020570     02  DETAIL-LINE REDEFINES HEAD-THREE.
020580         03  FILLER                  PICTURE X(28).
020590         03  CUST-NO-P               PICTURE 9(6).
020600         03  FILLER                  PICTURE X(16).
020610         03  TYPE-P                  PICTURE X(8).
020620         03  FILLER                  PICTURE X(14).
020630         03  ACCT-NO-P               PICTURE 9(6).
020640         03  FILLER                  PICTURE X(9).
020650         03  AMOUNT-P                PICTURE Z,ZZZ,ZZZ,ZZZ,99-.
020660         03  FILLER                  PICTURE X(28).
020670 01  DATE-AND-TIME.
020680     02  MONTH                       PICTURE 99.
020690     02  DAY                         PICTURE 99.
020700     02  YEAR                        PICTURE 99.
020710     02  TIME                        PICTURE 9(8)
020720                 USAGE IS COMPUTATIONAL-1.
020730 01  DISPLAY-FIELD             PICTURE 9(8).
```

```
86222 01  02-05-70   14,799   GE600 INTEGRATED STORE TRANSLATOR   ISDL-2 CHG00

IDS ALTER NOS.

00101 030010 IDS SECTION.
                 01  CCBLOXK .
                 02 DIRECT-REFERENCE SIZE IS 8 USAGE IS COMPUTATIONAL-1
                 SYNCHRONIZED RIGHT.
                 02 FIRST-REFERENCE SIZE IS 8 USAGE IS COMPUTATIONAL-1
                 SYNCHRONIZED RIGHT.
                 02 LAST-REFERENCE SIZE IS 8 USAGE IS COMPUTATIONAL-1
                 SYNCHRONIZED RIGHT.
                 02 RECORD-TYPE SIZE IS 4 USAGE IS COMPUTATIONAL-1
                 SYNCHRONIZED RIGHT.
                 02 REC-FILE SIZE IS 6 CLASS IS ALPHANUMERIC
                 VALUE IS   "0000TF".
                 02 ERROR-REFERENCE SIZE IS 3 CLASS IS ALPHANUMERIC
                 SYNCHRONIZED RIGHT.
00102 030020 MD  TEST-FILE
00103 030030     PAGE CONTAINS 1920 CHARACTERS
00104 030040     FILE CONTAINS 100 PAGES.
00105 030050 01  ENTRY-REC
00106 030060     TYPE IS 010
00107 030070     RETRIEVAL VIA CALC CHAIN
00108 030080     PAGE-RANGE IS 1 TO 1.
00109 030090     02  ENTRY-FIELD             PICTURE 9(6).
00110 030200     98  CALC CHAIN DETAIL
00111 030210         RANDOMIZE ON ENTRY-FIELD.
00112 030230     98  CUST-NO-CHN CHAIN MASTER
00113 030240         CHAIN-ORDER IS SORTED.
00114 030250 01  CUST-NO-REC
00115 030260     TYPE IS 020
00116 030270     RETRIEVAL VIA CUST-NO-CHN CHAIN.
00117 030290     02  CUST-NO-DSU             PICTURE 9(6).
00118 030300     02  CUST-NAME-DSU           PICTURE X(26).
00119 031010     98  CUST-NO-CHN CHAIN DETAIL
00120 031020         DUPLICATES NOT ALLOWED
00121 031030         ASCENDING KEY IS CUST-NO-DSU
00122 031040         SELECT CURRENT MASTER.
00123 031050     98  CHECK-CHN CHAIN MASTER
00124 031060         CHAIN-ORDER IS FIRST.
00125 031070     98  SAVE-CHN CHAIN MASTER
00126 031080         CHAIN-ORDER IS FIRST.
00127 031090     98  LOAN-CHN CHAIN MASTER
00128 031200         CHAIN-ORDER IS FIRST.
00129 031210 01  CHECK-REC
00130 031220     TYPE IS 021
00131 031230     RETRIEVAL VIA CHECK-CHN CHAIN.
00132 031250     02  CUST-NO-CK              PICTURE 9(6).
00133 031260     02  ACCT-NO-CK              PICTURE 9(6).
00134 031270     02  AMOUNT-CK               PICTURE S9(10)V99.
00135 031280     98  CHECK-CHN CHAIN DETAIL
00136 031290         SELECT CURRENT MASTER.
00137 032010 01  SAVE-REC
```

IDS ALTER NOS,

```
032020      TYPE IS 022
032030      RETRIEVAL VIA SAVE-CHN CHAIN.
032050      02  CUST-NO-SA              PICTURE 9(6).
032060      02  ACCT-NO-SA              PICTURE 9(6).
032070      02  AMOUNT-SA               PICTURE S9(10)V99.
032080      98  SAVE-CHN CHAIN DETAIL
032210          SELECT CURRENT MASTER.
032220 01   LOAN-REC
032230      TYPE IS 023
032240      RETRIEVAL VIA LOAN-CHN CHAIN.
032260      02  CUST-NO-LO              PICTURE 9(6).
032270      02  ACCT-NO-LO              PICTURE 9(6).
032280      02  AMOUNT-LO               PICTURE S9(10)V99.
032290      98  LOAN-CHN CHAIN DETAIL
032320          SELECT CURRENT MASTER.
040010 PROCEDURE DIVISION.
040022 010-START.
040023      ACCEPT DATE-AND-TIME FROM TODAYS-DATE.
040024      OPEN INPUT CARD-READER
040025           OUTPUT PRINT-UNIT.
040030*     OPEN IDS DATA BASE
040040      ENTER IDS.
040050          OPEN.
040055      MOVE 000001 TO ENTRY-FIELD.
040057*     CREATE ENTRY RECORD
040060      ENTER IDS.
040070          STORE ENTRY-REC
040080          IF ERROR GO TO 100-RET-MST-ENTRY-ERR.
040081      ENTER IDS.
040082      DEBUG CURRENT BUFFER
040083                    RECORD
040084                    CCBLOC.
040090 020-READ-CARDS.
040100      READ CARD-READER AT END GO TO 310-SNOOPY-EXIT.
040110      IF LOAN-ACCT OR SAVE-ACCT OR CHECK-ACCT
040120          GO TO 030-PROCESS-CARD.
040130      DISPLAY "INVALID CARD CODE".
040140      DISPLAY CARD-IN.
040150      GO TO 020-READ-CARDS.
040160 030-PROCESS-CARD.
040170      ENTER IDS.
040180          RETRIEVE ENTRY-REC RECORD
040190          IF ERROR GO TO 100-RET-MST-ENTRY-ERR.
040200 040-RET-CUST-REC.
040210      ENTER IDS.
040220          RETRIEVE NEXT RECORD OF CUST-NO-CHN CHAIN
040230          IF ERROR GO TO 110-RET-MST-ERR ELSE
040240          IF ENTRY-REC RECORD GO TO 060-STORE-MST-REC
040250          ELSE MOVE.
041010      IF CUST-NO-IN IS EQUAL TO CUST-NO-DSU
```

IDS ALTER NOS.

```
041020            GO TO 050-STORE-DETAIL.
041030        GO TO 040-RET-CUST-REC.
041040    050-STORE-DETAIL.
041050        IF LOAN-ACCT GO TO 060-STORE-LOAN.
041060        IF SAVE-ACCT GO TO 070-STORE-SAVE.
              ADD 1 TO NUCK.
041065*       CALL CHECKING SEGMENT
041070*           CREATES AND STORES CHECK-REC RECORD
041080        ENTER LINKAGE MODE.
041090            CALL LLINK USING SEG-1
041100            CALL CHKSEG
041110        ENTER COBOL.
041130        GO TO 020-READ-CARDS.
041140    060-STORE-LOAN.
              ADD 1 TO NULO.
040045*       CALL LOAN SEGMENT
040050*           CREATES AND STORES LOAN-REC RECORD
041160        ENTER LINKAGE MODE.
041170            CALL LLINK USING SEG-3
041180            CALL LOASEG
041190        ENTER COBOL.
041210        GO TO 020-READ-CARDS.
042010    070-STORE-SAVE.
              ADD 1 TO NUSA.
042015*       CALL SAVING SEGMENT
042020*           CREATES AND STORES SAV-REC RECORD
042030        ENTER LINKAGE MODE.
042040            CALL LLINK USING SEG-2
042050            CALL SAVSEG
042060        ENTER COBOL.
042080        GO TO 020-READ-CARDS.
042090    080-STORE-MST-REC.
042091*       CREATE AND STORE CUSTOMER NUMBER RECORD
042100        MOVE CUST-NO-IN TO CUST-NO-DSU.
042110        MOVE CUST-NAME-IN TO CUST-NAME-DSU.
042120        ENTER IDS.
042130            STORE CUST-NO-REC
042140            IF ERROR GO TO 150-STORE-CUST-REC-ERR.
042141        ENTER IDS.
042142            DEBUG CURRENT BUFFER
042143                          RECORD
042143                          CCBLOC.
042150        GO TO 050-STORE-DETAIL.
043010    100-RET-MST-ENTRY-ERR.
043020        DISPLAY "RETRIEVE ERROR".
043030        DISPLAY "FILE ENTRY RECORD".
043040        GO TO 300-WRAP-UP.
043050    110-RET-MST-ERR.
043060        DISPLAY "RETRIEVE ERROR".
043070        DISPLAY "CUSTOMER RECORD".
```

IDS ALTER NOS.

```
043080      GO TO 040-RET-CUST-REC.
043210 150-STORE-CUST-REC-ERR.
043220      DISPLAY "STORE ERROR".
043230      DISPLAY "CUSTOMER RECORD".
043240      GO TO 020-READ-CARDS.
043241*
043242*     THIS IS THE ENTRY PROINT ROUTINE THAT
043243*     IS CALLED BY THE FORTRAN PROGRAM
043244*
044014      ENTER LINKAGE MODE.
044015          ENTRY POINT ENTABC.
044016      ENTER COBOL.
044020      MOVE SPACES TO WORK-LINE.
044030      WRITE PRINT-LINE FROM WORK-LINE
044040          BEFORE ADVANCING TO TOP OF PAGE.
044045*     SET UP AND PRINT REPORT HEADINGS
044050      MOVE "CUSTOMER NUMBERS AND ACCOUNTS STORED ON DATA FILE"
044060          TO TITLE-1.
044070      WRITE PRINT-LINE FROM WORK-LINE BEFORE ADVANCING 2 LINES.
044080      MOVE SPACES TO WORK-LINE.
044090      MOVE MONTH TO MONTH-P.
044100      MOVE DAY TO DAY-P.
044110      MOVE YEAR TO YEAR-P.
044130      MOVE "-" TO DASH-1, DASH-2.
044140      WRITE PRINT-LINE FROM WORK-LINE BEFORE ADVANCING 2 LINES.
044150      MOVE SPACES TO WORK-LINE.
044160      MOVE "CUSTOMER NUMBER" TO TITLE-10.
044170      MOVE "TYPE OF ACCOUNT" TO TITLE-20.
044180      MOVE "ACCOUNT NO" TO  TITLE-30.
044190      MOVE "AMOUNT" TO TITLE-40.
044200      WRITE PRINT-LINE FROM WORK-LINE BEFORE ADVANCING 2 LINES.
044210      MOVE SPACES TO WORK-LINE.
044220      MOVE 000001 TO ENTRY-FIELD.
044230      ENTER IDS.
044240          RETRIEVE ENTRY-REC RECORD
044250          IF ERROR GO TO 209-RET-ENT-ERR.
044260 201-GET-CUST-CHN.
044270      WRITE PRINT-LINE FROM WORK-LINE BEFORE ADVANCING 1 LINES.
044275*     RETRIEVE AND PRINT CUSTOMER NUMBER RECORD*
044280      ENTER IDS.
044290          RETRIEVE NEXT RECORD OF CUST-NO-CHN CHAIN
044300          IF ERROR GO TO 210-RET-CUST-ERR ELSE
044310          IF ENTRY-REC RECORD GO TO 206-CREAT-ERROR
044320          ELSE MOVE.
044330      MOVE CUST-NO-DSU TO CUST-NO-P.
044340 203-GET-CHECK-REC.
044345*     RETRIEVE AND PRINT DETAIL RECORDS OF CHECK-CHN CHAIN
044350      ENTER IDS.
044360          RETRIEVE NEXT RECORD OF CHECK-CHN CHAIN
044370          IF ERROR GO TO 211-RET-CK-ERR ELSE
```

IDS ALTER NOS.

```
044380            IF CUST-NO-REC RECORD GO TO 204-GET-SAVE-REC
044390            ELSE MOVE.
044400        MOVE "CHECKING" TO TYPE-P.
044410        MOVE ACCT-NO-CK TO ACCT-NO-P.
044420        MOVE AMOUNT-CK TO AMOUNT-P.
044430        WRITE PRINT-LINE FROM WORK-LINE BEFORE ADVANCING 1 LINES.
044440        MOVE SPACES TO WORK-LINE.
044450        GO TO 203-GET-CHECK-REC.
044460    204-GET-SAVE-REC.
044465*       RETRIEVE AND PRINT DETAIL RECORDS OF SAVE-CHN CHAIN
044470        ENTER IDS.
044480            RETRIEVE NEXT RECORD OF SAVE-CHN CHAIN
044490            IF ERROR GO TO 212-RET-SA-ERR ELSE
044500            IF CUST-NO-REC RECORD GO TO 205-GET-LOAN-REC
044510            ELSE MOVE.
044520        MOVE "SAVINGS " TO TYPE-P.
044530        MOVE ACCT-NO-SA TO ACCT-NO-P.
044540        MOVE AMOUNT-SA TO AMOUNT-P.
044550        WRITE PRINT-LINE FROM WORK-LINE BEFORE ADVANCING 1 LINES.
044560        MOVE SPACES TO WORK-LINE.
044570        GO TO 204-GET-SAVE-REC.
044580    205-GET-LOAN-REC.
044585*       RETRIEVE AND PRINT DETAIL RECORDS OF LOAN-CHN CHAIN
044590        ENTER IDS.
044600            RETRIEVE NEXT RECORD OF LOAN-CHN CHAIN
044610            IF ERROR GO TO 213-RET-LO-ERR ELSE
044620            IF CUST-NO-REC RECORD GO TO 201-GET-CUST-CHN
044630            ELSE MOVE.
044640        MOVE "LOAN    " TO TYPE-P.
044660        MOVE ACCT-NO-LO TO ACCT-NO-P.
044670        MOVE AMOUNT-LO TO AMOUNT-P.
044680        WRITE PRINT-LINE FROM WORK-LINE BEFORE ADVANCING 1 LINES.
044690        MOVE SPACES TO WORK-LINE.
044700        GO TO 205-GET-LOAN-REC.
044710    206-CREAT-ERROR.
044720        WRITE PRINT-LINE FROM WORK-LINE BEFORE
044730            ADVANCING TO TOP OF PAGE.
044820        ENTER IDS.
044830            DEBUG CURRENT BUFFER
044840                          RECORD
044850                          CCBLOC
044855            TRACE CUST-NO-CHN CHAIN.
044870        GO TO 300-WRAP-UP.
044920    209-RET-ENT-ERR.
044930        DISPLAY "RETRIEVE ERROR".
044940        DISPLAY "FILE ENTRY RECORD".
044950        GO TO 300-WRAP-UP.
044960    210-RET-CUST-ERR.
044970        DISPLAY "RETRIEVE ERROR".
044980        DISPLAY "CUSTOMER RECORD".
```

IDS ALTER NOS.

```
00338 044990     GO TO 201-GET-CUST-CHN.
00339 045100 211-RET-CK-ERR.
00340 045110     DISPLAY "RETRIEVE ERROR".
00341 045120     DISPLAY "CHECK RECORD".
00342 045130     GO TO 203-GET-CHECK-REC.
00343 045140 212-RET-SA-ERR.
00344 045150     DISPLAY "RETRIEVE ERROR".
00345 045160     DISPLAY "SAVING RECORD".
00346 045170     GO TO 204-GET-SAVE-REC.
00347 045180 213-RET-LO-ERR.
00348 045190     DISPLAY "RETRIEVE ERROR".
00349 045200     DISPLAY "LOAN RECORD".
00350 045210     GO TO 205-GET-LOAN-REC.
00351 045220 300-WRAP-UP.
00352 045240     CLOSE CARD-READER, PRINT-UNIT.
00353 045245*    CLOSE IDS DATA BASE
00354 045250     ENTER IDS.
00355 045260         CLOSE.
00356 045261 309-PROGRAM-EXIT.
00357 045262     EXIT ENTABC.
00358 045270 310-SNOOPY-EXIT.
00359 045280     EXIT PROGRAM.
                  IDS-STRUCTURE SECTION.
                 ENTER GMAP .
                   PMC      ON
                   BLOCK    ,IDS,,
            RD0641 ,QRD     023,000033,0,0,0,000000,
                   ETC      RD0641,RD0645,RD0643,0000,RD6273,
                   ETC      000000,000000,LOAN-REC
            RD0645 ,QDD     023,10,0,0,1,1,
                   ETC      1,RD0641,RD0641,RD4609,RD4609,RD0645,
                   ETC      0029,0000,0000,
                   ETC      LOAN-CHN
            RD0643 ,QFD     0,0,000011,0006,FC4610,RD0644,
                   ETC      RD0643,RD4610,
                   ETC      ACCT-NO-LO
            RD0644 ,QFD     3,0,000017,0012,FC6209,RD0642,
                   ETC      RD0644,RD6209,
                   ETC      AMOUNT-LO
            RD0642 ,QFD     0,0,000005,0006,FC1153,RD0641,
                   ETC      RD0642,RD1153,
                   ETC      CUST-NO-LO
            RD6273 ,QRD     022,000033,0,0,0,000000,
                   ETC      RD6273,RD6277,RD6275,0000,RD1665,
                   ETC      000000,000000,SAVE-REC
            RD6277 ,QDD     022,10,0,0,1,1,
                   ETC      1,RD6273,RD6273,RD5889,RD5889,RD6277,
                   ETC      0029,0000,0000,
                   ETC      SAVE-CHN
            RD6275 ,QFD     0,0,000011,0006,FC7745,RD6276,
```

86226 01 02-05-70   14.791   GE600 INTEGRATED STORE TRANSLATOR   ISDL-2 CHG00

IDS ALTER NOS.

```
010010 IDENTIFICATION DIVISION.
010020 PROGRAM-ID. CHKSEQ.
010030 AUTHOR. GEORGE A RUDOLPH.
010040 DATE-WRITTEN. MAY 1969.
010050 INSTALLATION. G-E - PHOENIX.
010051 REMARKS. THIS IS THE CHECKING SEGMENT WHICH IS CALLED BY
010052     THE MAIN PROGRAM SNOOPY TO CREATE AND STORE
010053     CHECK-REC RECORDS.
010060 ENVIRONMENT DIVISION.
010070 CONFIGURATION SECTION.
010080 SOURCE-COMPUTER. GE-635.
010090 OBJECT-COMPUTER. GE-635.
010091 SPECIAL-NAMES.
010092     GETIME IS TODAYS-DATE.
010095     BLOCK 10 IS CCBLOXK.
010096     BLOCK 20 IS ENTRY-REC THRU LOAN-REC.
010200 INPUT-OUTPUT SECTION.
010210 FILE-CONTROL.
010220     SELECT CARD-READER ASSIGN TO CR FOR CARDS.
010230     SELECT IDS TEST-FILE ASSIGN TO TF.
010240 I-O-CONTROL.
010250     APPLY SYSTEM STANDARD FORMAT ON CARD-READER.
020010 DATA DIVISION.
020020 FILE SECTION.
020030 FD  CARD-READER
020040     LABEL RECORDS ARE STANDARD
020050     DATA RECORD IS CARD-IN.
020060 01  CARD-IN.
020070     02  ACCT-TYPE                PICTURE XX.
020080         88  LOAN-ACCT       VALUE "LO".
020090         88  SAVE-ACCT       VALUE "SA".
020200         88  CHECK-ACCT      VALUE "CK".
020210     02  CUST-NO-IN               PICTURE 9(6).
020220     02  ACCT-NO-IN               PICTURE 9(6).
020230     02  CUST-NAME-IN             PICTURE X(26).
020240     02  AMOUNT-IN                PICTURE 9(10)V99.
020250     02  FILLER                   PICTURE X(31).
020300 WORKING-STORAGE SECTION.
030010 IDS SECTION.
           01  CCBLOXK .
           02 DIRECT-REFERENCE SIZE IS 8 USAGE IS COMPUTATIONAL-1
           SYNCHRONIZED RIGHT.
           02 FIRST-REFERENCE SIZE IS 8 USAGE IS COMPUTATIONAL-1
           SYNCHRONIZED RIGHT.
           02 LAST-REFERENCE SIZE IS 8 USAGE IS COMPUTATIONAL-1
           SYNCHRONIZED RIGHT.
           02 RECORD-TYPE SIZE IS 4 USAGE IS COMPUTATIONAL-1
           SYNCHRONIZED RIGHT.
           02 REC-FILE SIZE IS 6 CLASS IS ALPHANUMERIC
           VALUE IS   "0000TF".
```

```
86226 01  02-05-70   14,791   GE600 INTEGRATED STORE TRANSLATOR   ISDL-2 CHG00

IDS ALTER NOS,

                 02 ERROR-REFERENCE SIZE IS 3 CLASS IS ALPHANUMERIC
                 SYNCHRONIZED RIGHT.
030020 MD        TEST-FILE
030030           PAGE CONTAINS 1920 CHARACTERS
030040           FILE CONTAINS 100 PAGES.
030050 01        ENTRY-REC
030060           TYPE IS 010
030070           RETRIEVAL VIA CALC CHAIN
030080           PAGE-RANGE IS 1 TO 1.
030090           02  ENTRY-FIELD                PICTURE 9(6).
030200           98  CALC CHAIN DETAIL
030210               RANDOMIZE ON ENTRY-FIELD.
030230           98  CUST-NO-CHN CHAIN MASTER
030240               CHAIN-ORDER IS SORTED.
030250 01        CUST-NO-REC
030260           TYPE IS 020
030270           RETRIEVAL VIA CUST-NO-CHN CHAIN.
030290           02  CUST-NO-DSU                PICTURE 9(6).
030300           02  CUST-NAME-DSU              PICTURE X(26).
031010           98  CUST-NO-CHN CHAIN DETAIL
031020               DUPLICATES NOT ALLOWED
031030               ASCENDING KEY IS CUST-NO-DSU
031040               SELECT CURRENT MASTER.
031050           98  CHECK-CHN CHAIN MASTER
031060               CHAIN-ORDER IS FIRST.
031070           98  SAVE-CHN CHAIN MASTER
031080               CHAIN-ORDER IS FIRST.
031090           98  LOAN-CHN CHAIN MASTER
031200               CHAIN-ORDER IS FIRST.
031210 01        CHECK-REC
031220           TYPE IS 021
031230           RETRIEVAL VIA CHECK-CHN CHAIN.
031250           02  CUST-NO-CK                 PICTURE 9(6).
031260           02  ACCT-NO-CK                 PICTURE 9(6).
031270           02  AMOUNT-CK                  PICTURE S9(10)V99.
031280           98  CHECK-CHN CHAIN DETAIL
031290               SELECT CURRENT MASTER.
032010 01        SAVE-REC
032020           TYPE IS 022
032030           RETRIEVAL VIA SAVE-CHN CHAIN.
032050           02  CUST-NO-SA                 PICTURE 9(6).
032060           02  ACCT-NO-SA                 PICTURE 9(6).
032070           02  AMOUNT-SA                  PICTURE S9(10)V99.
032080           98  SAVE-CHN CHAIN DETAIL
032210               SELECT CURRENT MASTER.
032220 01        LOAN-REC
032230           TYPE IS 023
032240           RETRIEVAL VIA LOAN-CHN CHAIN.
032260           02  CUST-NO-LO                 PICTURE 9(6).
032270           02  ACCT-NO-LO                 PICTURE 9(6).
```

```
86224 01  02-05-70   14.836   GE600 INTEGRATED STORE TRANSLATOR   ISDL-2 CHG00

IDS ALTER NOS,

032280     02  AMOUNT-LO                PICTURE S9(10)V99,
032290     98  LOAN-CHN CHAIN DETAIL
032320         SELECT CURRENT MASTER.
040010 PROCEDURE DIVISION.
042000 100-SAVE-PARA.
042020     MOVE CUST-NO-IN TO CUST-NO-SA,
042030     MOVE ACCT-NO-IN TO ACCT-NO-SA,
042040     MOVE AMOUNT-IN TO AMOUNT-SA.
042050     ENTER IDS.
042060         STORE SAVE-REC RECORD
042070         IF ERROR GO TO 140-STORE-SA-ERR.
042080     GO TO 200-CALL-SA-END.
043170 140-STORE-SA-ERR.
043180     DISPLAY "STORE ERROR".
043190     DISPLAY "SAVE RECORDS".
043200 200-CALL-SA-END.
043230     EXIT.
            IDS-STRUCTURE SECTION.
           ENTER GMAP .
               PMC      ON
               BLOCK    .IDS..
       RD0641 ,QRD      023,000033,0,0,0,000000,
               ETC      RD0641,RD0645,RD0643,0000,RD6273,
               ETC      000000,000000,LOAN-REC
       RD0645 ,QDD      023,10,0,0,1,1,
               ETC      1,RD0641,RD0641,RD4609,RD4609,RD0645,
               ETC      0029,0000,0000,
               ETC      LOAN-CHN
       RD0643 ,QFD      0,0,000011,0006,FC4610,RD0644,
               ETC      RD0643,RD4610,
               ETC      ACCT-NO-LO
       RD0644 ,QFD      3,0,000017,0012,FC6209,RD0642,
               ETC      RD0644,RD6209,
               ETC      AMOUNT-LO
       RD0642 ,QFD      0,0,000005,0006,FC1153,RD0641,
               ETC      RD0642,RD1153,
               ETC      CUST-NO-LO
       RD6273 ,QRD      022,000033,0,0,0,000000,
               ETC      RD6273,RD6277,RD6275,0000,RD166[illegible],
               ETC      000000,000000,SAVE-REC
       RD6277 ,QDD      022,10,0,0,1,1,
               ETC      1,RD6273,RD6273,RD5889,RD5889,RD6277,
               ETC      0029,0000,0000,
               ETC      SAVE-CHN
       RD6275 ,QFD      0,0,000011,0006,FC7745,RD6276,
               ETC      RD6275,RD7745,
               ETC      ACCT-NO-SA
       RD6276 ,QFD      3,0,000017,0012,FC3137,RD6274,
               ETC      RD6276,RD3137,
               ETC      AMOUNT-SA
```

```
86224 01  02-05-70   14.836   GE600 INTEGRATED STORE TRANSLATOR   ISDL-2 CHG00

IDS ALTER NOS,

010010 IDENTIFICATION DIVISION,
010020 PROGRAM-ID. SAVSEG.
010030 AUTHOR, GEORGE A RUDOLPH,
010040 DATE-WRITTEN, MAY 1969.
010050 INSTALLATION. G E - PHOENIX,
010051 REMARKS.  THIS IS THE SAVING SEGMENT WHICH IS CALLED BY
010052      THE MAIN PROGRAM SNOOPY TO CREATE AND STORE
010053      SAVE-REC RECORDS.
010060 ENVIRONMENT DIVISION.
010070 CONFIGURATION SECTION.
010080 SOURCE-COMPUTER. GE-635,
010090 OBJECT-COMPUTER. GE-635,
010091 SPECIAL-NAMES.
010092      GETIME IS TODAYS-DATE,
010095      BLOCK 10 IS CCBLOXK,
010096      BLOCK 20 IS ENTRY-REC THRU LOAN-REC.
010200 INPUT-OUTPUT SECTION.
010210 FILE-CONTROL.
010220      SELECT CARD-READER ASSIGN TO CR FOR CARDS.
010230      SELECT IDS TEST-FILE ASSIGN TO TF,
010240 I-O-CONTROL.
010250      APPLY SYSTEM STANDARD FORMAT ON CARD-READER.
020010 DATA DIVISION.
020020 FILE SECTION,
020030 FD  CARB-READER
020040      LABEL RECORDS ARE STANDARD
020050      DATA RECORD IS CARD-IN,
020060 01   CARD-IN.
020070      02  ACCT-TYPE               PICTURE XX.
020080          88  LOAN-ACCT       VALUE "LO".
020090          88  SAVE-ACCT       VALUE "SA".
020200          88  CHECK-ACCT      VALUE "CK".
020210      02  CUST-NO-IN              PICTURE 9(6).
020220      02  ACCT-NO-IN              PICTURE 9(6).
020230      02  CUST-NAME-IN            PICTURE X(25).
020240      02  AMOUNT-IN               PICTURE 9(10)V99.
020250      02  FILLER                  PICTURE X(31).
020300 WORKING-STORAGE SECTION,
030010 IDS SECTION,
            01  CCBLOXK .
            02 DIRECT-REFERENCE SIZE IS 8 USAGE IS COMPUTATIONAL-1
            SYNCHRONIZED RIGHT.
            02 FIRST-REFERENCE SIZE IS 8 USAGE IS COMPUTATIONAL-1
            SYNCHRONIZED RIGHT.
            02 LAST-REFERENCE SIZE IS 8 USAGE IS COMPUTATIONAL-1
            SYNCHRONIZED RIGHT.
            02 RECORD-TYPE SIZE IS 4 USAGE IS COMPUTATIONAL-1
            SYNCHRONIZED RIGHT.
            02 REC-FILE SIZE IS 6 CLASS IS ALPHANUMERIC
            VALUE IS   "0000TF",
```

IDS ALTER NOS.

```
                02 ERROR-REFERENCE SIZE IS 3 CLASS IS ALPHANUMERIC
                SYNCHRONIZED RIGHT.
030020 MD  TEST-FILE
030030     PAGE CONTAINS 1920 CHARACTERS
030040     FILE CONTAINS 100 PAGES.
030050 01  ENTRY-REC
030060     TYPE IS 010
030070     RETRIEVAL VIA CALC CHAIN
030080     PAGE-RANGE IS 1 TO 1.
030090     02  ENTRY-FIELD             PICTURE 9(6).
030200     98  CALC CHAIN DETAIL
               RANDOMIZE ON ENTRY-FIELD.
030210     98  CUST-NO-CHN CHAIN MASTER
030230         CHAIN-ORDER IS SORTED.
030240 01  CUST-NO-REC
030250     TYPE IS 020
030260     RETRIEVAL VIA CUST-NO-CHN CHAIN.
030270     02  CUST-NO-DSU             PICTURE 9(6).
030290     02  CUST-NAME-DSU           PICTURE X(26).
030300     98  CUST-NO-CHN CHAIN DETAIL
031010         DUPLICATES NOT ALLOWED
031020         ASCENDING KEY IS CUST-NO-DSU
031030         SELECT CURRENT MASTER.
031040     98  CHECK-CHN CHAIN MASTER
031050         CHAIN-ORDER IS FIRST.
031060     98  SAVE-CHN CHAIN MASTER
031070         CHAIN-ORDER IS FIRST.
031080     98  LOAN-CHN CHAIN MASTER
031090         CHAIN-ORDER IS FIRST.
031200 01  CHECK-REC
031210     TYPE IS 021
031220     RETRIEVAL VIA CHECK-CHN CHAIN.
031230     02  CUST-NO-CK              PICTURE 9(6).
031250     02  ACCT-NO-CK              PICTURE 9(6).
031260     02  AMOUNT-CK               PICTURE S9(10)V99.
031270     98  CHECK-CHN CHAIN DETAIL
031280         SELECT CURRENT MASTER.
031290 01  SAVE-REC
032010     TYPE IS 022
032020     RETRIEVAL VIA SAVE-CHN CHAIN.
032030     02  CUST-NO-SA              PICTURE 9(6).
032050     02  ACCT-NO-SA              PICTURE 9(6).
032060     02  AMOUNT-SA               PICTURE S9(10)V99.
032070     98  SAVE-CHN CHAIN DETAIL
032080         SELECT CURRENT MASTER.
032210 01  LOAN-REC
032220     TYPE IS 023
032230     RETRIEVAL VIA LOAN-CHN CHAIN.
032240     02  CUST-NO-LO              PICTURE 9(6).
032260     02  ACCT-NO-LO              PICTURE 9(6).
```

```
86226 01  02-05-70   14.791   GE600 INTEGRATED STORE TRANSLATOR   ISDL-2 CHG00

IDS ALTER NOS.

00088 032280      02  AMOUNT-LO                PICTURE S9(10)V99.
00089 032290      98  LOAN-CHN CHAIN DETAIL
00090 032320          SELECT CURRENT MASTER.
```

IDS ALTER NOS.

```
080000*EJECT
080010 PROCEDURE DIVISION.
081050 100-CHECK-PARA.
081070     MOVE CUST-NO-IN TO CUST-NO-CK.
081080     MOVE ACCT-NO-IN TO ACCT-NO-CK.
081090     MOVE AMOUNT-IN TO AMOUNT-CK.
081100     ENTER IDS.
081110        STORE CHECK-REC RECORD
081120        IF ERROR GO TO 120-STORE-CK-ERR.
081130     GO TO 200-CALL-CK-END.
083090 120-STORE-CK-ERR.
083100     DISPLAY "STORE ERROR".
083110     DISPLAY "CHECK RECORD".
083120 200-CALL-CK-END.
083150     EXIT.
            IDS-STRUCTURE SECTION.
           ENTER GMAP .
             PMC      ON
             BLOCK    .IDS..
      RD0641 ,QRD     023,000033,0,0,0,000000,
             ETC      RD0641,RD0645,RD0643,0000,RD6273,
             ETC      000000,000000,LOAN-REC
      RD0645 ,QDD     023,10,0,0,1,1,
             ETC      1,RD0641,RD0641,RD4609,RD4609,RD0645,
             ETC      0029,0000,0000,
             ETC      LOAN-CHN
      RD0643 ,QFD     0,0,000011,0006,FC4610,RD0644,
             ETC      RD0643,RD4610,
             ETC      ACCT-NO-LO
      RD0644 ,QFD     3,0,000017,0012,FC6209,RD0642,
             ETC      RD0644,RD6209,
             ETC      AMOUNT-LO
      RD0642 ,QFD     0,0,000005,0006,FC1153,RD0641,
             ETC      RD0642,RD1153,
             ETC      CUST-NO-LO
      RD6273 ,QRD     022,000033,0,0,0,000000,
             ETC      RD6273,RD6277,RD6275,0000,RD1665,
             ETC      000000,000000,SAVE-REC
      RD6277 ,QDD     022,10,0,0,1,1,
             ETC      1,RD6273,RD6273,RD5889,RD5889,RD6277,
             ETC      0029,0000,0000,
             ETC      SAVE-CHN
      RD6275 ,QFD     0,0,000011,0006,FC7745,RD6276,
             ETC      RD6275,RD7745,
             ETC      ACCT-NO-SA
      RD6276 ,QFD     3,0,000017,0012,FC3137,RD6274,
             ETC      RD6276,RD3137,
             ETC      AMOUNT-SA
      RD6274 ,QFD     0,0,000005,0006,FC1025,RD6273,
             ETC      RD6274,RD1025,
```

```
86225 01  02-05-70   14.809   GE600 INTEGRATED STORE TRANSLATOR   ISDL-2 CHG00

IDS ALTER NOS.

010010 IDENTIFICATION DIVISION.
010020 PROGRAM-ID. LOASEG.
010030 AUTHOR. GEORGE A RUDOLPH.
010040 DATE-WRITTEN. MAY 1969.
010050 INSTALLATION. G E - PHOENIX.
010051 REMARKS.  THIS IS THE LOAN SEGMENT WHICH IS CALLED BY
010052     THE MAIN PROGRAM SNOOPY TO CREATE AND STORE
010053     LOAN-REC RECORDS.
010060 ENVIRONMENT DIVISION.
010070 CONFIGURATION SECTION.
010080 SOURCE-COMPUTER. GE-635.
010090 OBJECT-COMPUTER. GE-635.
010091 SPECIAL-NAMES.
010092     GETIME IS TODAYS-DATE.
010095     BLOCK 10 IS CCBLOXK.
010096     BLOCK 20 IS ENTRY-REC THRU LOAN-REC.
010200 INPUT-OUTPUT SECTION.
010210 FILE-CONTROL.
010220     SELECT CARD-READER ASSIGN TO CR FOR CARDS.
010230     SELECT IDS TEST-FILE ASSIGN TO TF.
010240 I-O-CONTROL.
010250     APPLY SYSTEM STANDARD FORMAT ON CARD-READER.
020010 DATA DIVISION.
020020 FILE SECTION.
020030 FD  CARD-READER
020040     LABEL RECORDS ARE STANDARD
020050     DATA RECORD IS CARD-IN.
020060 01  CARD-IN.
020070     02  ACCT-TYPE                PICTURE XX.
020080         88  LOAN-ACCT        VALUE "LO".
020090         88  SAVE-ACCT        VALUE "SA".
020200         88  CHECK-ACCT       VALUE "CK".
020210     02  CUST-NO-IN               PICTURE 9(6).
020220     02  ACCT-NO-IN               PICTURE 9(6).
020230     02  CUST-NAME-IN             PICTURE X(25).
020240     02  AMOUNT-IN                PICTURE 9(10)V99.
020250     02  FILLER                   PICTURE X(31).
020300 WORKING-STORAGE SECTION.
030010 IDS SECTION.
           01  CCBLOXK .
           02 DIRECT-REFERENCE SIZE IS 8 USAGE IS COMPUTATIONAL-1
           SYNCHRONIZED RIGHT.
           02 FIRST-REFERENCE SIZE IS 8 USAGE IS COMPUTATIONAL-1
           SYNCHRONIZED RIGHT.
           02 LAST-REFERENCE SIZE IS 8 USAGE IS COMPUTATIONAL-1
           SYNCHRONIZED RIGHT.
           02 RECORD-TYPE SIZE IS 4 USAGE IS COMPUTATIONAL-1
           SYNCHRONIZED RIGHT.
           02 REC-FILE SIZE IS 6 CLASS IS ALPHANUMERIC
           VALUE IS   "0000TF".
```

IDS ALTER NOS,

```
                 02 ERROR-REFERENCE SIZE IS 3 CLASS IS ALPHANUMERIC
                 SYNCHRONIZED RIGHT.
030020 MD        TEST-FILE
030030           PAGE CONTAINS 1920 CHARACTERS
030040           FILE CONTAINS 100 PAGES.
030050 01        ENTRY-REC
030060           TYPE IS 010
030070           RETRIEVAL VIA CALC CHAIN
030080           PAGE-RANGE IS 1 TO 1.
030090           02  ENTRY-FIELD              PICTURE 9(6).
030200           98  CALC CHAIN DETAIL
030210               RANDOMIZE ON ENTRY-FIELD.
030230           98  CUST-NO-CHN CHAIN MASTER
030240               CHAIN-ORDER IS SORTED.
030250 01        CUST-NO-REC
030260           TYPE IS 020
030270           RETRIEVAL VIA CUST-NO-CHN CHAIN.
030290           02  CUST-NO-DSU              PICTURE 9(6).
030300           02  CUST-NAME-DSU            PICTURE X(26).
031010           98  CUST-NO-CHN CHAIN DETAIL
031020               DUPLICATES NOT ALLOWED
031030               ASCENDING KEY IS CUST-NO-DSU
031040               SELECT CURRENT MASTER.
031050           98  CHECK-CHN CHAIN MASTER
031060               CHAIN-ORDER IS FIRST.
031070           98  SAVE-CHN CHAIN MASTER
031080               CHAIN-ORDER IS FIRST.
031090           98  LOAN-CHN CHAIN MASTER
031200               CHAIN-ORDER IS FIRST.
031210 01        CHECK-REC
031220           TYPE IS 021
031230           RETRIEVAL VIA CHECK-CHN CHAIN.
031250           02  CUST-NO-CK               PICTURE 9(6).
031260           02  ACCT-NO-CK               PICTURE 9(6).
031270           02  AMOUNT-CK                PICTURE S9(10)V99.
031280           98  CHECK-CHN CHAIN DETAIL
031290               SELECT CURRENT MASTER.
032010 01        SAVE-REC
032020           TYPE IS 022
032030           RETRIEVAL VIA SAVE-CHN CHAIN.
032050           02  CUST-NO-SA               PICTURE 9(6).
032060           02  ACCT-NO-SA               PICTURE 9(6).
032070           02  AMOUNT-SA                PICTURE S9(10)V99.
032080           98  SAVE-CHN CHAIN DETAIL
032210               SELECT CURRENT MASTER.
032220 01        LOAN-REC
032230           TYPE IS 023
032240           RETRIEVAL VIA LOAN-CHN CHAIN.
032260           02  CUST-NO-LO               PICTURE 9(6).
032270           02  ACCT-NO-LO               PICTURE 9(6).
```

```
86225 01  02-05-70   14.809   GE600 INTEGRATED STORE TRANSLATOR   ISDL-2 CHG00

IDS ALTER NOS.

032280      02  AMOUNT-LO                  PICTURE S9(10)V99.
032290      98  LOAN-CHN CHAIN DETAIL
                SELECT CURRENT MASTER.
032320 PROCEDURE DIVISION.
040010 100-LOAN-PARA.
041130      MOVE CUST-NO-IN TO CUST-NO-LO.
041150      MOVE ACCT-NO-IN TO ACCT-NO-LO.
041160      MOVE AMOUNT-IN TO AMOUNT-LO.
041170      ENTER IDS.
041180          STORE LOAN-REC RECORD
041190          IF ERROR GO TO 130-STORE-LO-ERR.
041200      GO TO 200-CALL-LO-END.
041210 130-STORE-LO-ERR.
043130      DISPLAY "STORE ERROR".
043140      DISPLAY "LOAN RECORD".
043150 200-CALL-LO-END.
043160      EXIT.
043190
             IDS-STRUCTURE SECTION.
            ENTER GMAP .
              PMC      ON
              BLOCK    .IDS..
       RD0641 ,QRD     023,000033,0,0,0,000000,
              ETC      RD0641,RD0645,RD0643,0000,RD6271,
              ETC      000000,000000,LOAN-REC
       RD0645 ,QDD     023,10,0,0,1,1,
              ETC      1,RD0641,RD0641,RD4609,RD4609,RD0645,
              ETC      0029,0000,0000,
              ETC      LOAN-CHN
       RD0643 ,QFD     0,0,000011,0006,FC4610,RD0644,
              ETC      RD0643,RD4610,
              ETC      ACCT-NO-LO
       RD0644 ,QFD     3,0,000017,0012,FC6209,RD0642,
              ETC      RD0644,RD6209,
              ETC      AMOUNT-LO
       RD0642 ,QFD     0,0,000005,0006,FC1153,RD0641,
              ETC      RD0642,RD1153,
              ETC      CUST-NO-LO
       RD6273 ,QRD     022,000033,0,0,0,000000,
              ETC      RD6273,RD6277,RD6275,0000,RD1665,
              ETC      000000,000000,SAVE-REC
       RD6277 ,QDD     022,10,0,0,1,1,
              ETC      1,RD6273,RD6273,RD5889,RD5889,RD6277,
              ETC      0029,0000,0000,
              ETC      SAVE-CHN
       RD6275 ,QFD     0,0,000011,0006,FC7745,RD6276,
              ETC      RD6275,RD7745,
              ETC      ACCT-NO-SA
       RD6276 ,QFD     3,0,000017,0012,FC3137,RD6274,
              ETC      RD6276,RD3137,
              ETC      AMOUNT-SA
```

```
86226 02  02-06-70   00.02/                                                                                      PAGE      1

   ORIGIN   053069      ENTRY LOCATION    ENTRY LOCATION    ENTRY LOCATION    ENTRY LOCATION    ENTRY LOCATION


                                 SUBPROGRAMS INCLUDED IN DECK.

                        $      OPTION  FORTRAN
                        $      USE     ,QMAX/1/,,QAREA/1577/,,QMIN/1/
   056570   020570      ...... 056570
         BLOCK COMMON   C20    056540      C21    056534
   054560   020570      C.LDIN 054572      SNOOPY 054573    C.SNOO 056524    ENTABC 055232
         BLOCK COMMON   ,IDS.. 054346      C21    056534    C10    054336    C20    056540    PR     053060
                        CR     051614


                                 SUBPROGRAMS OBTAINED FROM SYSTEM  LIBRARY.

   051540   110268      ,SETU. 051545
   047322   102969      ,FRDD. 051024     ,FWRD. 050756    ,FPRN. 050756    ,FPUN. 050752    ,FCNV. 047647
                        ,FFIL. 051061     ,FRTN. 051060    ,FRCD. 051024    ,DBCNV 051073    ,BDCNV 051814
   047260   110268      ,FEOF. 047260
   047122   073069      ,FSLEW 047122
   046526   110268      ,FOPEN 046532     EXIT   046736    ,FEXIT 046736    ,FBAD. 047105    ,FBFTB 047106
                        ,FXOP. 046721     ,FGTFB 046526    ,FJOV, 046530
   045244   102969      FXEM   046047     ,FXEM. 045244    ,FXMC, 046032    ANYERR 046155    FXOPT  046066
                        FXDVCK 046127     FXALT  046143    FXDV   046440    FXFDV  046436    FXCODE 046046
                        ,FXSW1 046034     ,FXSW2 046036    ,FXSW3 046040    ERRLK  046163    ,FLTPR 046165
   045006   110268      ,F1D0. 045122
   044670   110268      LINK   044700     LLINK  044676    IDLINK 044721    LENTRY 044671
   043440   073069      ,QDBUG 043440     H5,138 044655
   041412   082869      ,QOPEN 041412     ,QTAB1 043344    ,QTAB2 043375    H5,122 043426
         BLOCK COMMON   ,QMAX  061776     ,QMIN  056722
   041322   073069      ,QCHN  041322     H5,106 041405
   041254   073069      ,QGET  041254     H5,112 041315
   040614   073069      ,QSTOR 040614     H5,127 041236
   040472   073069      ,QMOVE 040472     H5,120 040607
   037436   073069      ,QTLNK 037436     ,QTALY 037704    H5,129 040461
   037420   110268      ,QUCCB 037420     H5,131 037433
   037364   110268      ,QSDSW 037364     H5,126 037414
   036770   080769      ,QMNO  036770     H5,119 037355
   036410   073069      ,QASC  036410     H5,102 036763
   036234   073069      ,QCALC 036234     ,QCAL1 036372    H5,105 036404
   035772   073069      ,QGDET 035772     H5,111 036227
   035734   110268      ,QSYN  035734     H5,128 035766
   035654   110268      ,QUPDC 035654     H5,134 035731
   035630   110268      ,QAUTH 035630     H5,103 035651
   035362   073069      ,QFWD  035362     H5,110 035623
   035160   073069      ,QLNK1 035344     ,QLNK5 035345    ,QLSW  035353    ,QDLNK 035160    H5,108 035354
   034776   073069      ,QUDCH 034776     H5,133 035152
   034706   110268      ,QRUND 034706     H5,125 034770
   034614   110268      ,QIYPX 034614     H5,130 034702
   034500   073069      ,QADJU 034500     ,QINSW 034607    H5,101 034611
```

ORIGIN	053069	ENTRY	LOCATION	ENTRY	LOCATION	ENTRY	LOCATION	ENTRY	LOCATION	ENTRY	LOCATION
034262	073069	.QMRAC	034262	H5.121	034473						
034222	110268	.QPACK	034222	H5.123	034257						
034010	073069	.QUIT	034010	.QUITX	034203	.QUITY	034067	H5.132	034216		
033344	073069	.QCLOS	033344	H5.107	034004						
032172	073069	.QBIC	032172	.QFLSH	033130	.QSBEF	033222	.QSEMT	032676	H5.104	033136
031760	110268	.QINV2	031760	H5.160	032164						
031672	110268	.QLAR	031672	.QLOCK	031707	.QRLS	031733	H5.152	031755		
031572	073069	.QINV1	031572	H5.159	031666						
031266	073069	.QINV3	031266	H5.161	031564						
031210	073069	.QINV4	031210	H5.162	031262						
031036	073069	.QMAP1	031036	H5.163	031205						
030554	110268	.QRTAB	030554	.QTAB3	030762	H5.164	031032				
030344	073069	.QIOS	030367	.QWRIT	030344	.QREAD	030353	.QRDB	030362	.QWAIT	030365
		H5.165	030550								
027462	073069	.QWKA	027511	.QIRBA	027514	.QAVAL	027531	.QIREF	027511	.QINIT	027512
		.QIPOS	027513	.QFOP	027607	.QPMLV	027610	.QRELV	027611	.QERTB	027613
		OUTCB	027631	OUTFD	027620	.QIDCW	027515	.FINV	027537	.EINV	027540
		.IDSF	027472	.IJRNL	027504	.QSICT	027517	.QACT	027520	.QMBUF	027523
		.INVBF	027530	.QBWA	027516	.QLCCB	027522	.QVECT	027527	.QCBUF	027524
		.QEMTY	027525	.QDBGS	027521	.QCDSW	027526	.QCREC	027532	.QCURD	027533
		.QCURT	027534	.QDPSW	027535	.QNFN	027536	.QBARG	027541	.QICTR	027542
		.QSTST	027544	.QGAST	027550	.QGCST	027554	.QGDST	027560	.QGEST	027564
		.QCHST	027570	.QHDST	027574	.QMFST	027600	.QDLST	027604	.QECAC	027612
		H5.166	030340								
026264	110268	.QUTF	026264	.QUTF1	026264	.QUTF2	026303	.QUTF3	026400	.QVFY	026712
		.QVFY4	027121	.QVFY6	027442	.QVFYI	027451	H5.145	027453		
024712	073069	.QBCD	024712	.QCLR	024745	.QCSM	024765	.QDIR	025022	.QDIRC	025652
		.QDIRF	025116	.QMCH	026047	.QMEX	026103	.QMWD	026153	.QPBK	026177
		.QSFD	025672	.QDIR9	025130	H5.154	026256				
024702	073069	.QSTA	024702	.QSTA1	024702	.QSTA2	024704	H5.137	024706		
024440	102969	.CNTRY	024530	.CMXIT	024543	.CMSER	024456	.CMENT	024441	.CMRET	024475
		.CMPSH	024456	.CMPOP	024467	.CMUST	024505	.CMUET	024515	.CMUEX	024522
		..FICB	024675								
024426	102969	.CDATE	024434	.CTALY	024427	.CTMP0	024430	.CTMP1	024432	.C1020	024434
024340	110268	.CMSTK	024341	.CMSTE	024424						
023756	073069	.CGOPN	023760	.CTEOF	024247	.CIOER	024311	.CIOEI	024330		
023456	110268	.COSWR	023457								
023310	110268	.COSYS	023312								
023132	073069	.CICON	023135	.CITYP	023133	.COCBF	023266	.COSIZ	023244	.CXXXX	023136
023050	110268	.COBUF	023076	.CLINE	023071	.CIBUF	023052				
022402	110268	.CNFXA	022414	.CNFXB	022661	.CNFX1	022404				
022370	110268	.CED07	022371								
022362	110268	.CED05	022363								
022244	110268	.CERPL	022245								
022166	110268	.CESSN	022167								
022132	110268	.CEITL	022134	.CEGET	022140	.CEPVW	022163				
022012	110268	.CERSN	022013	.CETSS	022127	.CETSN	022044				
021732	110268	.CETLS	021734	.CESTL	021734	.CERTL	021735	.CEDLS	022000	.CECHR	022001
		.CEDEC	022005	.CBCMA	022006	.CESSW	021776	.CNNUM	021736	.CNQU1	021736
		.CEQU1	021736	.CNQU2	021737	.CEQU2	021737	.CEZST	021740	.CENOP	021741

```
86226 02  02-06-70   00.027

ORIGIN   053069     ENTRY LOCATION    ENTRY LOCATION    ENTRY LOCATION    ENTRY LOCATION    ENTRY LOCATION

                    .CESAV 021742     .CEOPS 021777
021726   110268     .CMEND 021726
021676   102969     .CCCCC 021717     X1060  021720     X1091  021721     X1095  021722     .CTBEG 021712
                    BEGIN  021712     .CTCOR 021713     CORECT 021713     .CTCMT 021714     XCMNT  021714
                    .CTDMP 021715     XDUMP  021715     .CTIOM 021716     X4000  021716     .CTEND 021677
                    XXEND  021714     .CTFIN 021677     XTMIII 021677     .CQUIT 021677
021604   110268     .GWRIT 021604     .GAWRI 021604     WRITE  021604
021510   110268     .GREAD 021510     .GAREA 021510     READ   021510
021426   110268     .GWAIT 021426     .GAWAI 021426     WAIT   021426
021332   110268     .GSTOT 021332     SETOUT 021332
021302   110268     .GSTIN 021302     SETIN  021302
021026   072569     .GEPRN 021026     EPRINT 021026
020540   110268     .GPRNT 020540     .GAPRN 020540     PRT004 020575     PRT024 020717     PRT031 020736
                    PRT032 020745     PRT035 020752     PRT051 020770     PRT002 020572     PRINT  020540
020454   110268     .GIOPG 020454     IOP021 020530     IOP024 020533
020404   073069     .GWTRC 020404     .GAWTR 020404     WTREC  020404
020270   110268     .GEDIT 020270     .GE062 020363     .GE063 020364     .GE064 020365     .GE065 020367
                    .GE066 020370     .GE067 020371     .GAEDI 020270     .GE068 020372     .GE069 020373
                    .GE071 020374     .GE072 020375     EDATE  020376     ETIME  020377     IOEDIT 020270
017562   073069     .GGTBK 017562     GETBK  017562     .GGET  017564     GET    017564     .GAGTB 017562
                    .GAGET 017564
017554   073069     .GOPNR 017554     .GCLSR 017554     .GGETR 017554     .GPUTR 017554
017040   073069     .GCOPY 017040     COPY   017040     .GPTBK 017043     PUTBK  017043     .GPUT  017046
                    PUT    017046     .GACOP 017040     .GAPTB 017043     .GAPUT 017046     .GFR67 017531
016742   110268     .GPTSZ 016742     .GAPTS 016742     PUTSZ  016742
016202   073069     .GOPEN 016202     .GAOPE 016202     OPEN   016202
015530   073069     .GCLSE 015530     .GACLS 015530     .GR185 015634     .GR186 015725     .GR178 015641
                    CLOSE  015530
015512   110268     .GBNRY 015512
015412   073069     .GRLSE 015412     .GARLS 015412     RELSE  015412
015220   110268     .GR200 015220
015172   110268     .GBCD  015172
015114   110268     .GR225 015114
015042   110268     .GR250 015042
014550   073069     .GR275 014550
014400   110268     .GR377 014434     .GR385 014373     .GR375 014400     .GR37X 014453     .GR390 014473
014274   110268     .GR980 014274     .GR979 014372     .GR99X 014300     .GR984 014336     .GR985 014372
                    .GR999 014304
014214   110268     .GR960 014214
013476   110268     .GINHD 013503     .GOUTH 013502     .GINTL 013501     .GOUTL 013500     .GUSWH 013477
                    .GOVRL 013504     .GLREA 013562     .GRCVY 013476
013474   110268     .GINID 013474
013442   110268     .GR990 013442     .GR991 013463     15AUG5 013470
                                            RANGE                SIZE
                    ALLOCATED CORE    000000 THRU 061777     062000
                    OBJECT PROGRAM
                       RELOCATABLE    013440 THRU 061777     046340

                    $      LINK    LINKAA
```

```
86226 02  02-06-70   00.027

   ORIGIN   053069      ENTRY LOCATION    ENTRY LOCATION    ENTRY LOCATION    ENTRY LOCATION    ENTRY LOCATION

                               SUBPROGRAMS INCLUDED IN DECK.

 *** NON FATAL ERROR * C.LDIN LOADED PREVIOUSLY
   013320   020570      CHKSEG 013333     C.CHKS 013427
         BLOCK COMMON   ,IDS., 054346     C10    054336     C20     056540     CR     051614

                               SUBPROGRAMS OBTAINED FROM SYSTEM  LIBRARY.

                                                   RANGE                SIZE
                        ALLOCATED CORE     000000 THRU 061777         062000
                        OBJECT PROGRAM
                           RELOCATABLE     013320 THRU 061777         046460
 *** NON FATAL ERROR * MISSING ROUTINE    ,CFICB
 *** NON FATAL ERROR * MISSING ROUTINE    LOASEG
 *** NON FATAL ERROR * MISSING ROUTINE    SAVSEG

                        $      LINK    LINKBB,LINKAA

                               SUBPROGRAMS INCLUDED IN DECK.

 *** NON FATAL ERROR * C.LDIN LOADED PREVIOUSLY
   013320   020570      SAVSEG 013333     C.SAVS 013427
         BLOCK COMMON   ,IDS., 054346     C10    054336     C20     056540     CR     051614

                               SUBPROGRAMS OBTAINED FROM SYSTEM  LIBRARY.

                                                   RANGE                SIZE
                        ALLOCATED CORE     000000 THRU 061777         062000
                        OBJECT PROGRAM
                           RELOCATABLE     013320 THRU 061777         046460
 *** NON FATAL ERROR * MISSING ROUTINE    ,CFICB
 *** NON FATAL ERROR * MISSING ROUTINE    LOASEG

                        $      LINK    LINKCC,LINKBB

                               SUBPROGRAMS INCLUDED IN DECK.

 *** NON FATAL ERROR * C.LDIN LOADED PREVIOUSLY
   013320   020570      LOASEG 013333     C.LOAS 013426
         BLOCK COMMON   ,IDS., 054346     C10    054336     C20     056540     CR     051614

                               SUBPROGRAMS OBTAINED FROM SYSTEM  LIBRARY.

                                                   RANGE                SIZE
                        ALLOCATED CORE     000000 THRU 061777         062000
```

```
86226 02  02-06-70   00.027

   ORIGIN   053069      ENTRY LOCATION    ENTRY LOCATION    ENTRY LOCATION    ENTRY LOCATION    ENTRY LOCATION

                        OBJECT PROGRAM
                           RELOCATABLE    013320 THRU 061777      044460
  *** NON FATAL ERROR * MISSING ROUTINE   .CFICB
                        $      DISC    H*,X2S,8R
                        $      DISC    TF,X1S,9R      TEMPORARY MASS STORAGE FILE
                        $      DATA    ,Q             TEMPORARY I-D-S DATA FILE FOR DIRECTIVES
                        $      SYSOUT  PR             ASSIGN PRINTER TO OUTPUT MEDIA CONVERSION
                        $      DATA    CR             TEMPORARY FILE FOR CARD INPUT

                        FCB AND BUFFER SPACE

                           AVAILABLE      000101 THRU 013315      013215
                           FILE CTRL BLKS 013166 THRU 013316      000131
                           MAXIMUM BUFFER SPACE REQUIRED          001200

                            21K, IS THE MINIMUM MEMORY NEEDED TO LOAD THIS ACTIVITY WITH ALL FILES OPEN

                        EXECUTION   PROGRAM ENTERED AT      056570

                        THERE WERE  000009 WARNING FLAGS IN THE ABOVE LOAD
```

CUSTOMER NUMBERS AND ACCOUNTS STORED ON DATA FILE

2-6-70

CUSTOMER NUMBER	TYPE OF ACCOUNT	ACCOUNT NO	AMOUNT
000123	CHECKING	003302	.74
	SAVINGS	000022	.00
	LOAN	002301	.10
000235	CHECKING	024501	145.71
001100	SAVINGS	000501	.09
	SAVINGS	002403	.01
004444	LOAN	000302	.00
055555	CHECKING	000904	1,987,654.32
123456	LOAN	000703	10.00
666111	CHECKING	005503	5.83

COBOL Program Output

```
   SNUMB = 86226, ACTIVITY # = 02, REPORT CODE = 52, RECORD COUNT = 00004

NUMBER OF CHECKING ACCOUNT RECORDS READ =    4
NUMBER OF LOAN RECORDS READ =      3
NUMBER OF SAVING ACCOUNT RECORDS READ =      3
TOTAL NUMBER OF RECORDS READ =      10
```

FORTRAN Program Output

Appendix D. Primary Subroutines

Primary subroutines are those subroutines which are called directly as a result of an I-D-S verb. The primary subroutine then calls other subroutines to perform the function. The following is a list of the I-D-S verbs and the corresponding primary subroutine which is called as a result of the verb.

I-D-S Verb	Primary Subroutines
CLOSE	.QCLOS
DELETE	.QDLTE
HEAD	.QHEAD
MODIFY	.QMDFY
MOVE	.QMOVE
OPEN	.QOPEN
RETRIEVE	.QGET
RETRIEVE CURRENT	.QGETC
RETRIEVE DIRECT	.QGETD
RETRIEVE EACH	.QGETE
RETRIEVE MASTER RETRIEVE NEXT RETRIEVE PRIOR	.QCHN
STORE	.QSTOR

Appendix E. Sample Deck Setups

COMPILE AND EXECUTE PERMFILES

The following Deck Setup will compile and execute an I-D-S program using a permanent I-D-S data file.

```
1      8       16
$      IDENT   IDSOO,PERMFILE
$      USERID  IDSFOURYQUAD$DATABASE
$      IDS
       I-D-S SOURCE DECK OR COMDK
$      EXECUTE
$      PRMFL   A1,R/W,R,IDSFOURYQUAD$DATABASE/QUAD01
$      PRMFL   A2,R/W,R,IDSFOURYQUAD$DATABASE/QUAD02
$      PRMFL   A3,R/W,R,IDSFOURYQUAD$DATABASE/QUAD03
$      PRMFL   A4,R/W,R,IDSFOURYQUAD$DATABASE/QUAD04
$      ENDJOB
***EOF
```

EXECUTE USING TEMPORARY FILES

The following Deck Setup will execute an I-D-S object program using temporary files. NOTE: The QUTU activity will initialize the database.

```
1      8       16
$      IDENT   IDSOO,TEMPFILE
$      PROGRAM QUTU
$      LIMITS  ,24k
$      MASS    A1,X1S,11R
$      DISC    A2,X2S,22R
$      DRUM    A3,X3S,11R
$      DATA    .Q
IDS    CREATE  FC/A1/,BSSZ/480/,RNG/1,120/
IDS    CREATE  FC/A2/,BSSZ/480/,RNG/121,240/,LPP/32/
IDS    CREATE  FC/A3/,BSSZ/480/,RNG/241,360/
$      DATA    I*
IDS    INIT    FC/A1/
IDS    INIT    FC/A2/
IDS    INIT    FC/A3/
$      OBJECT
       I-D-S   OBJECT DECK
$      DKEND
$      EXECUTE
$      MASS    T1,X1S,11R
$      DISC    T2,X2S,22R
$      DRUM    T3,X3S,11R
$      DATA    .Q
IDS    CREATE  FC/T1/,BSSZ/480/,RNG/1,120/
IDS    CREATE  FC/T2/,BSSZ/480/,RNG/121,240/,LPP/32/
IDS    CREATE  FC/T3/,BSSZ/480/,RNG/241,360/
$      ENDJOB
***EOF
```

COMPILE AND EXECUTE USING PERMANENT AND TEMPORARY FILES

The following Deck Setup will compile and execute an I-D-S program using permanent and temporary files. NOTE: The QUTU activity will reload the temporary file from tape.

```
1      8       16
$      IDENT   IDS00,MIXEDFILES
$      USERID  IDSFOURYQUAD$DATABASE
$      PROGRAM QUTU
$      LIMITS  ,24k
$      DISC    A1,X2S,22R
$      TAPE    DT,X6D
$      DATA    .Q
IDS    CREATE  FC/A1/,BSSZ/480/,RNG/121,240/,LPP/32/
$      DATA    I*
IDS    WRITE   FC/DT/,RNG/121,240/,ONFC/A1/
$      IDS
       I-D-S SOURCE DECK OR COMDK
$      EXECUTE
$      PRMFL   T1,R/W,R,IDSFOURYQUAD$DATABASE/QUAD01
$      DISC    T2,X2S,22R
$      PRMFL   T3,R/W,R,IDSFOURYQUAD$DATABASE/QUAD03
$      PRMFL   T4,R/W,R,IDSFOURYQUAD$DATABASE/QUAD04
$      DATA    .Q
IDS    CREATE  FC/T2/,BSSZ/480/,RNG/121,240/,LPP/32/
$      ENDJOB
***EOF
```

PRINT A PERMANENT FILE

The following Deck Setup is an example of a QUTU activity which prints a permanent file.

```
1      8       16
$      IDENT   IDS00,PRINT
$      USERID  IDSFOURYQUAD$DATABASE
$      PROGRAM QUTU
$      LIMITS  ,24k
$      PRMFL   TF,R/W,R,IDSFOURYQUAD$DATABASE/QUAD01
$      PRMFL   TG,R/W,R,IDSFOURYQUAD$DATABASE/QUAD02
$      DATA    I*
IDS    PRINT   FC/TF/,RNG/1,10/,PAGES
IDS    PRINT   FC/TG/,EMPTY
$      ENDJOB
***EOF
```

TRACE ENTRY

The following Deck Setup will compile and execute an I-D-S program using an I-D-S Permanent File and will generate a trace entry for all calls to the I-D-S primary subroutines.

```
1      8       16
$      IDENT   IDS00,TRCEDATA
$      USERID  IDSFOURYQUAD$DATABASE
$      IDS
       I-D-S SOURCE DECK OR COMDK
$      USE     .QSTC
$      EXECUTE
$      DATA    .Q
IDS    OPTION  TRACE
$      PRMFL   A1,R/W,R,IDSFOURYQUAD$DATABASE/QUAD01
$      PRMFL   A2,R/W,R,IDSFOURYQUAD$DATABASE/QUAD02
$      PRMFL   A3,R/W,R,IDSFOURYQUAD$DATABASE/QUAD03
$      PRMFL   A4,R/W,R,IDSFOURYQUAD$DATABASE/QUAD04
$      ENDJOB
***EOF
```

EXECUTE QUTJ

Deck Setup to execute QUTJ from the Software Library.

```
1      8       16
$      IDENT   IDS00,JOURNAL
$      PROGRAM QUTJ
$      LIMITS  OPTIONS
$      TAPE    IN,X1D,,1234,,JOURNAL-TAPE
$      DATA    I*
IDS    SYSTEM
$      ENDJOB
***EOF
```

EXECUTE QUTP

Deck Setup to execute QUTP from the Software Library.

```
1      8       16
$      IDENT   IDS00,PICKER
$      PROGRAM QUTP
$      LIMITS  OPTIONS
$      TAPE    IN,X1D,,1234,,JOURNAL-TAPE
$      TAPE    OT,X2S
$      DATA    I*
IDS    SELECT  1/53607,1/53607,B
IDS    SELECT  12/88802, 13/88802,B
$      ENDJOB
***EOF
```

EXECUTE QUTS

Deck Setup for executing QUTS from the Software Library.

```
1      8       16
$      IDENT   IDS00,SORT
$      PROGRAM QUTS
$      LIMITS  10,17k
$      TAPE    IN,X2D
$      TAPE    OT,X3S,,99999
$      TAPE    OU,X4S,,99999
$      NTAPE   S1,X5R,3
$      ENDJOB
***EOF
```

EXECUTE QUTI AND QUTC

Deck Setup for executing QUTI and QUTC from the Software Library.

```
1      8        16
$      IDENT    IDS00,CALC
$      PROGRAM  QUTI
$      MASS     A1,D1S,10R
$      DATA     .Q
IDS    CREATE   FC/A1/,BSSZ/100/,RNG/1,100/
$      DATA     I*
IDS    INITIAL  1,100
$      PROGRAM  QUTC
$      LIMITS   10,26k
$      TAPE     A1,A1R,,,,WORK1
$      TAPE     B1,B1R,,,,WORK2
$      TAPE     T1,T1D,,1234,,USER-IN
$      TAPE     C1,C1D,,,,USER-SORTED
$      NTAPE    S1,S1R, 3
$      MASS     D1,D1R,10R
$      SYSOUT   P1
$      DATA     .Q
IDS    CREATE   FC/D1/,BSSZ/100/,RNG/1,100/
$      DATA     I*
IDS    OPTION   GENERATE/,RANDA/,RNG/1, 30000/
$      ENDJOB
***EOF
```

EXECUTE QUTD

Deck Setup for executing QUTD from the Software Library.

```
1      8        16
$      IDENT    IDS00,DUMP
$      PROGRAM  QUTD
$      LIMITS   OPTIONS
$      USERID   IDSFOURYQUAD$DBASE
$      PRMFL    TF,R/W,R,IDSFOURYQUAD$DBASE/QUAD01
$      PRMFL    TG,R/W,R,IDSFOURYQUAD$DBASE/QUAD02
$      TAPE     OT,X2S,,,,DUMP-FILE
$      DATA     I*
IDS    DUMP
$      ENDJOB
***EOF
```

EXECUTE QUTL

Deck Setup for executing QUTL from the Software Library.

```
1       8       16
$       IDENT   IDS00,LOAD
$       PROGRAM QUTL
$       LIMITS  OPTIONS
$       MASS    A1,X1R,15R
$       TAPE    IN,X2S,,1234,,DUMP-FILE
$       TAPE    DE,X3S,,,,DELETE-FILE
$       DATA    .Q
IDS     CREATE  FC/A1/,BSSZ/480/,RNG/1,120/
$       DATA    I*
IDS     OPTION  PLOAD/,RNG/1,120/,DELETE/
$       ENDJOB
***EOF
```

COLLECTING TYPE B STATISTICS

Deck Setup for collecting type B statistics on the journal file and executing QUTR from the Software Library.

```
1       8       16
$       IDENT   IDS00,STATISTICS
$       USERID  IDSFOURYQUAD$DATABASE
$       USE     .QSTB
$       OBJECT
$       DKEND
$       EXECUTE
$       PRMFL   A1,R/W,R,IDSFOURYQUAD$DATABASE/QUAD01
$       TAPE    JX,X1S,,,,I-D-S-JOURNAL
$       PROGRAM QUTR
$       SYSOUT  P1
$       TAPE    A1,X1R,,,,I-D-S-JOURNAL
$       TAPE    B1,X2R,,99999
$       NTAPE   S1,T,2
$       ENDJOB
***EOF
```

Activity 1 is the execution of an IDS program which provides for collection of type B information on the user-created journal file (JX tape).

Activity 2 is the execution of QUTR.

Appendix F. Reference Code Manipulation

EXTRACT A PAGE NUMBER

Procedure Division statements similar to the following may be used to extract a page number from a reference code.

```
COMPUTE PAGE-NO = DIRECT-REFERENCE /64.
```

EXTRACT A LINE NUMBER

Procedure Division statements similar to the following may be used to extract a line number from a reference code.

a. Assume PAGE-NO was previously extracted.

```
COMPUTE LINE-NO = DIRECT-REFERENCE - (PAGE-NO * 64).
```

b. Assume PAGE-NO was not previously extracted.

```
COMPUTE LINE-NO = DIRECT-REFERENCE - ((DIRECT-REFERENCE/64)*64)
```

CREATE A REFERENCE CODE

Procedure Division statements similar to the following may be used to create a reference code.

a. Assume PAGE-NO has previously been initialized with the desired page number.

b. Assume LINE-NO has previously been initialized with the desired line number.

```
COMPUTE DIRECT-REFERENCE = (PAGE-NO *64) + LINE-NO.
               .
               .
               .
77 PAGE-NO PIC 9(6) COMP-1.
77 LINE-NO PIC 9(2) COMP-1.
               .
   .           .
   .           .
   .
IDS SECTION
   01  CCBLOXK.
   02  DIRECT-REFERENCE  PIC 9(8) COMP-1.
               .
               .
               .
```

Index

HONEYWELL INFORMATION SYSTEMS

Technical Publications Remarks Form*

TITLE:	SERIES 600/6000 INTEGRATED DATA STORE REFERENCE MANUAL	ORDER NO:	BR69, REV. 0
		DATED:	JANUARY, 1971

ERRORS IN PUBLICATION:

SUGGESTIONS FOR IMPROVEMENT TO PUBLICATION:

(Please Print)

FROM: NAME ______________________ DATE ______________

COMPANY ______________________

TITLE ______________________

ADDRESS ______________________

____

*Your comments will be promptly investigated by appropriate technical personnel, action will be taken as required, and you will receive a written reply. If you do not require a written reply, please check here. ☐

POSTAGE STAMP NOT NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL: FIRST CLASS PERMIT NO. 39531, NEWTON HIGHLANDS, MASS

POSTAGE WILL BE PAID BY:

HONEYWELL INFORMATION SYSTEMS

60 WALNUT STREET
WELLESLEY HILLS, MASS. 02181

ATTN: U.S. GROUP PUBLICATIONS, MS 050

Honeywell

The Other Computer Company:

Honeywell

HONEYWELL INFORMATION SYSTEMS

2628
51071
Printed in U.S.A.

BR69, Rev. 0