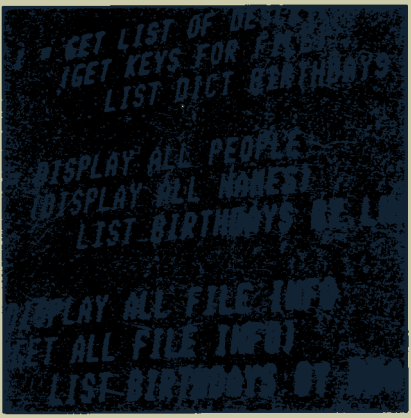
Prime Computer, Inc.

PRIME

INFORMATION Highlights



INFORMATION Companion

FDR5212

The INFORMATION Companion is a new series of pocket-size, quick reference guides to INFORMATION software.

Published by Prime Computer, Inc., INFORMATION Systems, 500 Old Connecticut Path, Framingham, MA 01701

Produced by Prime Computer Technical Publications Department, 500 Old Connecticut Path, MS 10B0701, Framingham, MA 01701

Copyright © 1981 by Prime Computer, Inc.

Printed in U.S.A. All rights reserved.

The following are trademarks of Prime Computer, Inc. PRIME, PRIMOS, PRIMENET, The Programmer's Companion, INFORMATION Companion and Office Automation Companion.

The information contained in this document reflects the software as of revision 4.5 of INFORMATION and is subject to change without notice. Prime Computer, Inc. assumes no responsibility for errors that may appear in this document.

First printing, June 1981

Credits

Research and copy
Laura Douros

Production
William I. Agush

Typesetting
JL Associates

Printing and binding
Regal Lithograph

TABLE OF CONTENTS

CONVENTIONS

This INFORMATION Companion uses the following typographic conventions.

Convention	Meaning
UPPERCASE	ALL INFORMATION verbs and keywords appear in uppercase to distinguish them from command line options and arguments. It is not necessary to type them in uppercase letters unless you want to. See Note, below.
lowercase	In command formats, items in lowercase are variables which represent arguments or options. The user must supply an appropriate numerical or text value for a variable.
[ ]	Items enclosed in brackets are optional and are not required on a command line.
{ }	Braces indicate that the user should choose one of the enclosed items. This convention is also used to indicate alternate versions of verb or command names.
(CR)	This symbol stands for a single carriage return. A carriage return is accomplished by hitting the RETURN key on most terminals. Anything you type at the keyboard is not transmitted to the system until you hit RETURN. This means that every time you communicate something to the system, you must terminate it with a carriage return.

Note

INFORMATION does distinguish between upper and lowercase, so if files are created with uppercase names, they can *only* be accessed by typing their names in uppercase letters. The same idea applies to files created with lowercase names.

OVERVIEW

The Prime INFORMATION system is a unique collection of file management tools that can be easily used by programmers and nonprogrammers. These tools do all the work of setting up and managing file structure and content, getting data from one place to another, formatting it for output, and so forth — all the tasks that programmers ordinarily have to do themselves. Because INFORMATION takes care of all this for you, it's not necessary to know anything about programming in order to use the system. All that's required is a basic understanding of some terms and concepts that have special meanings in INFORMATION. These terms and concepts are explained later.

Parts of INFORMATION

There are four parts to the INFORMATION system:

- **PERFORM** — the INFORMATION request monitor and command environment
- **INFORM** — the data manager, query language and report generator
- **INFO/BASIC** — the compiler for INFORMATION's special version of the BASIC language for writing user applications
- **EDITOR** — INFORMATION's own text editor for creating INFO/BASIC programs, editing data sets, etc.

PERFORM: acts like the system monitor. It picks up all user requests for action and decides what to do with them. Most often it passes the request on to another processor, like INFORM. The PERFORM colon prompt (:) appears whenever the system is waiting for you to give it some instruction. The colon prompt indicates that you are at PERFORM command level.

INFORM: is the INFORMATION system query language and report generator. Its primary function is to perform file searches and to format output. Another useful part of INFORM is the ENTRO processor which allows simple data entry and modification.

The INFO/BASIC Language: is a tool for programmers. It combines many features of PL/I and other high-level languages with the standard Dartmouth BASIC. With INFO/BASIC, complete applications packages can be written using the many available functions and sub-routines.

The INFORMATION EDITOR: is a line-oriented text editor. However, it does allow you to define "blocks" of lines within a file and move, copy, change, or delete them as a unit. With the EDITOR's "prestored command" feature you can create a file of EDITOR commands and execute them like a program or command file at any time, without need for user intervention. You can even recall and edit the commands you've already issued so they can be reused without being completely retyped.

Where to Find Help

Many concepts, terms, verbs, and so forth, are summarized in this companion, but you will probably need more details. Make sure you have all of these books:

- **The PERFORM Reference Guide**
- **The INFORM Reference Guide**
- **The INFO/BASIC Reference Guide**
- **The EDITOR Reference Guide**

More Help

A supplement to the documentation is the on-line system HELP facility, which is always kept up-to-date with each release of INFORMATION software. Simply type HELP and you'll be told what to do from there. Much of the information on commands and options in the INFORM and PERFORM books is briefly summarized in the HELP file for your convenience. The INFORMATION EDITOR commands are all summarized in the EDITOR HELP facility, which is available while you are in the EDITOR environment.

TERMS

There are many terms used in INFORMATION that require some introduction and explanation. Most of these terms are just familiar English words.

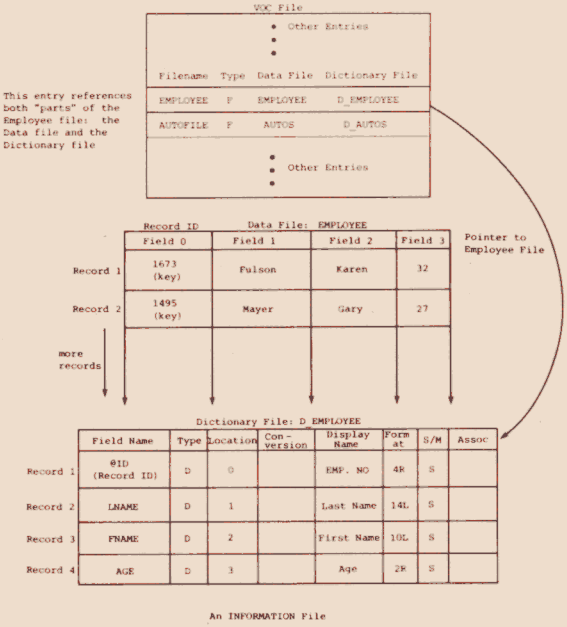
Files — A file is an organized collection of related information. For example, an EMPLOYEE file might contain information about each of a company's employees. See Figures 1 and 2 for a picture of the EMPLOYEE file.

Records — Files are made up of records. A record consists of related pieces of information that can be treated as a logical unit. In the EMPLOYEE file, there is one record for each employee.

Fields — Each file record is made up of one or more fields. Each field corresponds to a unique piece of information. Every record in a file consists of the same fields, but each record usually has different values for each of these fields. For example, in the EMPLOYEE file, each record contains information like last name, first name, age and so forth. While each record contains the same kinds of statistics, the actual data values stored in each field usually varies from one record to another.

INFORMATION FILE — An INFORMATION file is a collection of data, stored in the form of records, just like most files. However, each INFORMATION file has two parts: a **data** file, which houses all the information to be stored, and a **dictionary** file, which describes what's in the data file. The association between these two files enables fast and easy searches for data without requiring you to know how it is stored or organized. (For more details, see CONCEPTS.)

Descriptors — A descriptor names and describes a field in an INFORMATION file record. Every field in a record has a descriptor associated with it. In the EMPLOYEE file, the field describing last names is referenced by a descriptor called LNAME. The first name field is called FNAME, and so on.



An INFORMATION File

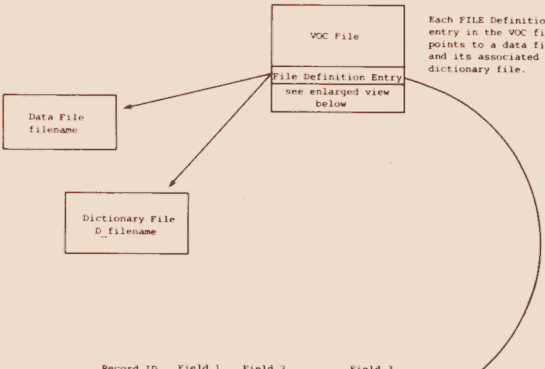
Figure 1

Data Descriptors — Data descriptors define actual fields in the records of the data file. Data descriptors, called "D" descriptors, define fields which can have one or more values. The LNAME, FNAME and AGE descriptors are all D descriptors.

I-Descriptors — I-descriptors, unlike data descriptors, describe fields which do not actually exist in the data file record. Instead, values for these fields are calculated from other fields according to some expression which the user must define. I-descriptors are handy for defining values that must be calculated on an ad hoc basis. For instance, if the EMPLOYEE file had a yearly salary field, a bonus field, and a tax exemption field, an I-descriptor could be defined for weekly salary and the value for it could be calculated using the data in all these fields.

Record IDs (Keys) — INFORMATION bases file access on a system of unique keys, or record identifiers. The terms "key" and "record identifier" are synonymous. In the INFORMATION system, they are also called "record IDs." Keys are used to distinguish one record from another. Since each record in an INFORMATION file has a unique key (record ID) value associated with it, each record can be located quickly on the basis of this value. Record ID characteristics are used to determine the method by which a file's records will be organized for efficient access. (See **File Types** below.) In the EMPLOYEE file, the record ID is the employee number field.

Hash-encoding — Hash-encoding is a method of treating record ID (key) values to arrive at some unique record location in a file. Generally, some portion of the key value is manipulated using a hashing algorithm to yield a unique number that represents a location in a file. Hash-encoded files can be accessed rapidly because of their structure. ALL INFORMATION files are hash-encoded except for INFO/BASIC programs and some miscellaneous housekeeping files. See the discussion on **File Types**, below.



Sample FILE Definition Record
(From VOC File)

Figure 2

A FILE Definition Entry

File Types — There are 13 types of INFORMATION files. All but one, the Type 1 file, are hash-encoded address files with unique record identifiers for each record. File types are determined by the physical characteristics of the record ID chosen for the file. Taking the EMPLOYEE file as an example, since the employee number field is numeric and is more likely to be unique in the rightmost digits, the best file type for this file is Type 2. (See **Chart of File Types**).

Modulus — A file modulus is a user-determined number of divisions or groups that file records can be stored in. Generally, the modulus can be determined by estimating the total size of a proposed data file and dividing this number by 1900. See **The PERFORM Reference Guide** for details.

Groups — Records in hash-encoded files are distributed in groups by means of a hashing algorithm based on record ID values. This enables rapid access to any record in the file.

Verbs — Directions to the system are given in the form of verbs. Most INFORMATION verbs do exactly what their names imply. Many verbs take arguments or options that tell the verb which file or record to perform an action on. A verb, accompanied by all the arguments necessary to complete an operation, forms an executable command or **sentence.** All the important INFORMATION verbs are summarized under **Summary of INFORMATION Verbs**, below.

Keywords — Words that modify or refine verb actions are called keywords. They are generally optional on command lines and force things like doublespacing, field-name suppression and so forth.

Sentences — Complete command lines can be saved for future use in the form of sentences. A sentence consists of a verb and whatever arguments (keywords, filenames, etc.) you want. Stored sentences are executed simply by typing the name associated with that sentence. See **The VOC File**, below.

Phrases — Phrases are incomplete sentences that do not contain verbs. They are usually command line arguments, like fieldnames, options, etc. They can be stored in the same manner as sentences and can also be stored in dictionary files.

Paragraphs — Many sentences can be stored for later execution in the form of a paragraph. This is the same idea as a single stored sentence: it's just a way of saving more than one sentence under a single name.

Select Lists — A select list is a group of file records or even filenames on which you want to perform the same operation. Typically you would create a select list as the result of a file search and then pass that select list to some other verb or sentence, enabling some action to be performed on this list of records or files.

Menus — The INFORMATION system has a menu processor that allows you to create menus. A menu is just a list of actions which can be performed by simply typing in the number that corresponds to that action. Often an entire INFORMATION application can be handled using a system of interwoven menus.

Accounts — Every INFORMATION user must have a valid account, or user ID, to gain access to the INFORMATION system. Your System Administrator should provide you with an account. All your personal files will be kept in this account (unless you choose to put them elsewhere).

Phantoms — A phantom process is one that runs independently of a user terminal and requires no user/terminal interaction. You simply submit a pre-stored series of commands or a program to the system to be executed as a phantom. The phantom will then run quietly on its own leaving the terminal free for other uses.

COMO Files — Command output (COMO) files are a complete transcript of everything you type at the terminal and everything that the system echoes back at you. COMO files are useful for preserving a complete account of the events occurring during a particular terminal session.

CONCEPTS

Below is a summary of some of the more important INFORMATION concepts. They include:

- The VOC file
- The ISYS account
- Data file
- Dictionary file
- File types

The VOC File

The VOC file is the most important file in the system from the user's point of view. It defines all the verbs, stored sentences, paragraphs, keywords, and files which are available to you while in a particular account. When a new account is created, a copy of the system VOC file is put in it. All the verbs which are part of the INFORM and PERFORM processors, plus all the INFORM keywords, are defined in the VOC file. In addition, there are some stored system sentences included for user convenience.

Whenever you create a new INFORMATION file, a File Definition Entry is added to the VOC file, indicating the name of the file and the name of its associated dictionary. Any sentence, phrase or paragraph that you want to keep for future use is stored in the VOC file. Everything you type in response to the PERFORM colon prompt is looked up in the VOC file to see if there is a match. If whatever you typed isn't found in the VOC file, PERFORM tells you about it. The VOC file, like most other INFORMATION files, has an associated dictionary, and both parts of the file can be edited.

The ISYS Account

ISYS is the master INFORMATION account. It contains all the INFORMATION verbs, the NEWACC file (the master file used to create the VOC file in each new account), the MENU file, the SYSHELP file (contains the system HELP text), and others.

The Data File

The data file part of an INFORMATION file is just a receptacle for the data being stored. Records in the data file are organized into groups by the system according to the file's type. The dictionary file associated with a data file tells INFORM how to read or interpret what's actually stored in the data file. Although INFO/BASIC and the EDITOR do not need the dictionary to access a given data file, INFORM must have a dictionary in order to read the data file. See the preceding figure for a picture of a sample data file.

Dictionary Files

A dictionary file is the logical definition of a data file. It describes the fields in the data file record, giving

important information about the field's location in the record, its output format, and so forth. The relationship of the dictionary and data files is expressed in Figure 1. The dictionary file is automatically named D_ filename, where **filename** is the name of the file you are creating.

You can override this default by giving a dictionary file any name you choose. Furthermore, a single dictionary can be associated with many data files and one data file can have many dictionaries associated with it. This is done by creating multiple File Definition entries in the VOC file showing all the relationships among the various files. Each relationship should have a unique name to eliminate confusion.

Chart of File Types

There are 13 legal file types. Types 2 through 13 are all hash-encoded and are differentiated by the hashing algorithm used to distribute and locate records in the file. The hashing algorithm used depends on the type of key (record ID) you define for a file. The chart below describes the important characteristics of each file type.

File Types

FILE TYPE	DESCRIPTION	SAMPLE KEYS
1	Type 1 files are implemented as PRIMOS-level sub-UFDs. Used for INFO/BASIC programs and files like &SAVEDLISTS&.	N/A
2	Type 2 files are designed for general use. Ideal for sequential numeric keys. Rightmost characters of keys should be more unique than leftmost characters.	00011 00012 00013
3	Type 3 files are designed for use with principally numeric keys that have separators like * / or #. Rightmost characters of keys are most unique.	131-47-8905 131-56-6788

4	Type 4 files are designed for use with strictly alphabetic keys taken from the 64 character ASCII subset. Rightmost characters are most unique.	LJDOU LMGAR MJMCC
5	Type 5 files are designed for use with keys that consist of characters taken from the full ASCII set. Rightmost characters of keys are most unique.	AD*103 AD*140
6	Like Type 2 files (numeric) except the keys are more unique in the leftmost character positions.	1000 2000 1200
7	Like Type 3 files (numeric with separators) except that keys are more unique in leftmost character positions.	01/23/81 12/24/81 09/01/81
8	Like Type 4 files (alphabetic) except that keys are more unique in leftmost character positions.	DOUGLASJM HARVEYJT
9	Like Type 5 files (full range of ASCII characters) except that keys are more unique in leftmost character positions.	#8801/IS #9965/IS #7657/OS
10	Like Types 2 and 6 (numeric) but keys are most unique in middle.	102300 102400
11	Like Types 3 and 7 (numeric with separators) but keys are more unique in middle character positions.	5330-97-2400 5330-82-2400
12	Like Types 4 and 8 (alphabetic) but keys are more unique in middle character positions.	METCSPHYS METMGMTPHYS
13	Like Types 5 and 10 (full range of ASCII characters) but keys are more unique in middle character positions.	MET-CS103 METMGMT103

GETTING ONTO THE SYSTEM

Getting started in INFORMATION is easy once you get past the formalities of accessing the system. These are summarized briefly below. A more complete explanation of how to use Prime computers can be found in **The Prime User's Guide.**

Correcting Typos

Because some of us make mistakes while typing, it helps to know that the default PRIMOS erase and kill characters are the " (double-quote) and the ? (question mark) respectively. The " character erases one character at a time. The line kill character (?) deletes an entire line up to the current cursor position. (The cursor is the blinking light that leads you across the terminal screen and indicates your current position on a line.) As long as you correct errors before hitting the RETURN key, you won't get besieged by strange error messages. Just keep trying until you get it right.

Logging In

Gaining access to the system is called "logging in." It's a way of identifying each user to the system. To get onto the system, follow these steps:

1) Obtain a user id or account from your System Administrator. An ID, also called a 'login ID", identifies you to the system so it knows who you are. Only people with authorized login IDs are allowed access to the system. Your login ID will probably be the same as your INFORMATION account.

2) Turn on your terminal and make sure it's **on-line**. (Hit the RETURN key once or twice until the "LOGIN PLEASE" prompt appears. This means the connection between the system and your terminal is active.)

3) Type:

   LOGIN your-id [your-password] [-on sysname]

You only need a password if your System Administrator gave you one or if you created one for yourself. The *-on*

sysname argument is used only when there are several systems networked together and your account is on a "remote" system. Remote means the terminal you are using is not physically connected to the system your account happens to be on. The name of the remote system should be supplied in the **sysname** argument.

4) If you used the **-on sysname** option in Step 3, you will then be asked to enter a validation code. Obtain this from your System Administrator. If you aren't logging in to a remote system, ignore this step.

5) If your login ID is a valid INFORMATION account, all you have to do to access INFORMATION is type: INFO. If not, INFORMATION asks if you want to make your ID a valid account. Say "yes" and you will be admitted to the INFORMATION system.

Note

On some systems, you will automatically enter INFORMATION as soon as you successfully log in. You won't have to type "INFO" to get there.

INFORMATION Erase and Kill Characters

Unlike PRIMOS, the INFORMATION erase character is the BACKSPACE key. On some terminals this key has a ← character on it. Every time you hit this key, one character of the line you just typed is erased. The line kill character is CTRL-X, which is a nonprinting character. Depress the control key (usually labeled CTRL) and the X key simultaneously.

If you type an entire command line and then decide you don't want to execute it but you'd rather not type the whole thing over, simply type a ? character at the end of the line before hitting the RETURN key. This will preserve the line without executing it. You can then use a special set of edit commands to make the appropriate changes to it or save it for future use. This feature is described under **The PERFORM Stack**, below.

CREATING A FILE

The steps involved in creating a file are shown below. The CREATE.FILE verb is fully described in **The PERFORM Reference Guide.**

Steps in File Creation	
STEP	**ACTION**
1	Decide what information you want to store in the file. Divide this information into logical pieces, or fields. Choose unique names for each field so you can easily identify each piece of information.
2	Choose a unique key or record id for this file. It can be one of the pieces of information you've already identified as a field, but it must be able to have a unique value for each record in this file.
3	Based on the characteristics of the key or record id you chose in Step 2, (has the key all numeric, all aphabetic characters, a mixture, etc.) choose a file type from the file type chart. Types 2 - 13 are legal file types.
4	Obtain the average record size by determining the average size of each field (in characters), including the key. Add all the field sizes together, multiply by 1.11 and add 4 to get the average record size.
5	Obtain the correct **modulus** for this file by multiplying the average record size obtained in Step 4 by an estimate of the number of records you want in the file. This gives you the approximate file size. Divide this number by 1900 to get the modulus. Use a modulus of 1 if file size 1900.

6	Use the CREATE.FILE verb to allocate space for the data file and dictionary file. Two files will be created in your dictionary with the names **filename** and **D_filename** using the filename you specify. This process also creates a VOC file entry for this file.
7	Using the ENTRO processor, define the contents of the dictionary file. You should create a descriptor for each field in the data file record that you want to access with INFORM verbs. Use the field names you chose in Step 1 as descriptor names.
8	Once descriptors have been defined in the dictionary file, you can add records to the data file. Again, use ENTRO to do this. ENTRO will prompt for field values using the descriptor names (field names) you defined in the dictionary file in Step 7, above.

USING ENTRO

The ENTRO processor lets you define the contents of both dictionary and data files. In both cases, prestored prompting sequences make it easy to set up an INFORMATION file.

Defining Dictionary Contents

ENTRO's prestored dictionary prompting sequence asks you to name each field in the data file, describe its location in the data file record, define the type of descriptor it is (D or I), define its output format, assign an optional display name, and provide a legal conversion expression for numeric data. You can also create phrases in the dictionary file using PH as the type, instead of D or I. The second field of the phrase record should name all the fields you want associated with that phrase. There are some special phrases that ENTRO looks for in a file dictionary: more on this under **Special Phrases** below.

Default Record ID: When a dictionary file is first created, INFORM automatically puts a record in the file called @ID. This is the default record ID descriptor. It is a D descriptor and defines field 0 of the data file record. The default @ID format is always 10L. You can change the default output format or the display name of the record ID using ENTRO.

Invoking ENTRO: To invoke ENTRO to create or modify the contents of a dictionary, type:

ENTRO DICT filename

where **filename** is the name you assigned to the INFORMATION file with the CREATE.FILE verb.

ENTRO Dictionary File Prompts: These are the prompts ENTRO paints on your screen, one at a time, when defining dictionary contents. Instructions are indicated in parentheses.

FIELD=	(Enter field name)
New record	
TYPE+EXP=	(Enter D, I or PH, plus optional explanation)
LOCATION=	(Enter location, expression or phrase items)
CONV=	(Optional: enter INFO/BASIC conversion formula)
DISPLAY NAME=	(Optional: enter display name for this field)
FORMAT=	(Enter output format, e.g., 10T, 7R$###.##)
S/M=	(Enter S for single-value fields, M for multi-value)
ASSOC=	(Optional: for multi-value fields only)

TYPE+EXP: Indicate whether this record will be a data descriptor (D), an I-descriptor (I) or a phrase (PH). If you enter a description, it will be displayed during data entry when the user types a ? to get help. If you don't want to define any more descriptors or phrases, simply hit RETURN or type QUIT in response to this prompt.

LOCATION: If it's a D descriptor, indicate where this field will reside in the record. Use sequential numbers, beginning with 1. This is just a way of logically ordering the contents of the data record. If you're defining an I-descriptor, this field contains the expression used to determine the I-descriptor value. When defining a phrase record, this field specifies the contents of the phrase.

CONV: This is an optional field governing the conversion of data using INFO/BASIC functions. See **The INFO/BASIC Reference Guide** for details.

DISPLAY NAME: If no DISPLAY NAME is desired, the default name used for this field will be the field name you defined in response to the FIELD= prompt.

FORMAT: Every field must have an output format. A legal format must contain one of the following indicators:

Symbol	Means
L	Left-justify field
R	Right-justify field
T	Text field

In addition, the format should include field size (in characters), and can include a conversion indicator and a mask field (like $ # - or /). For example, the format field:

11R$,

will print out an 11-character field, right-justified, beginning with a dollar sign, using commas between every three digits.

S/M: Most fields will be single-valued fields, that is, they will have one value in each file record. Multi-value fields are like tables and allow more than one value to exist in the field. For example, if you were keeping track of student course grades, you might have one field for STUDENT-NAME, which would be a single-value field (since each student has one name) and another field for GRADES, which would be a multi-value field since most students would have more than one course grade.

ASSOC: Multi-value fields can be associated or "related" by creating a phrase (PH) in the dictionary that lists all the multi-valued field names you want associated. These names go in field 2 of the phrase record. Each multi-value descriptor in this association must contain the name of this phrase record in the ASSOC field. Associated fields

are added, modified and deleted as a unit. See **The INFORM Reference Guide** for details.

Special Phrases

INFORM recognizes three special phrases in the dictionary file. The @ENTRO phrase, which is created automatically when ENTRO is first invoked to add data to the data file. It names all the D descriptors in the file. The other two phrases have special meaning to INFORM verbs. These are the "@" and the "@LPTR" phrases. Such phrases should be created in the dictionary of every INFORMATION file you set up.

The @ Phrase: The "@" phrase is the default terminal display list. It defines the fields in a file to be displayed on the terminal whenever the ENTRO, LIST and SORT verbs are used without fieldname arguments. The @ phrase names all the fields you want listed by default whenever these commands are used with only a filename argument. If no @ phrase exists, only the record ID values from the file will be listed and no other field values will be displayed.

The @LPTR Phrase: The @LPTR phase defines the default print list for a file. This phrase should contain the names of all the fields you want printed whenever you use the LPTR option with the LIST, SORT and ENTRO verbs. (The LPTR option (keyword) is discussed in **The INFORM Reference Guide**, along with the rest of the INFORM keywords.) If no @LPTR phrase is found in the file dictionary, these verbs then look for the @ phrase. If neither exists, only the record ID values will be printed.

Need Help?

If you're not sure what to enter as a response to any ENTRO prompt, type a question mark (?). ENTRO will then display a brief explanation of what it's looking for. The prompt will reappear following the explanation.

Mistakes?

As each descriptor or phrase is defined in the dictionary, ENTRO gives you a chance to change what you just entered. The prompts are as follows:

```
1 FIELD        user-entered-value
2 TYPE                 "
3 LOC                  "
4 CONV                 "
5 NAME                 "
6 FORMAT               "
7 S/M                  "
8 ASSOC                "
CHANGE= (user enters one of above numbers) (or
        simply hits carriage return)
```

If everything looks okay, just hit (CR). Otherwise, enter the number that corresponds to the field you want changed. The field will then be displayed and you can:

- Reenter it entirely
- Change it with the C/str1/str2/ command (see below)
- Append something to it with the "A str" command
- Delete the record entirely by typing: DELETE or FD
- Change nothing at all: just hit (CR)

Adding Data File Records

To add records to the data file, type:

ENTRO filename

where **filename** is the name of the data file.

ENTRO will then prompt you for input for each field described by a data (D) descriptor in the dictionary file. This is done using a default ENTRO-create phrase called @ENTRO, which merely lists all the D descriptors defined in the file dictionary. The @ENTRO phrase is automatically created the first time you type:

ENTRO filename

This assumes that you haven't already created your own @ENTRO phrase in the file dictionary.

ENTRO's Data File Prompts: ENTRO uses the field names defined in the dictionary as prompts. As each field name is displayed, ENTRO waits for you to add an appropriate value or a null line (a single carriage return). This continues until you've defined values for an entire record. ENTRO then gives you a chance to change any of the fields you just added in case you made an error or you changed your mind. This prompting sequence looks something like a menu in that you enter the number of the field you want to change. You then reenter the field value the way you want it, or you can use some simple "edit" commands to change the value.

ENTRO Screens

There are two types of screens that ENTRO uses for prompting. The Screen 1 format is for single-value fields, and the second screen format is designed for multi-value fields. ENTRO automatically creates a screen format for each multi-value field and numbers them consecutively, beginning with Screen 2.

Getting Help In ENTRO

If you want a quick summary of all the possible responses you can make to an ENTRO "CHANGE=" prompt, simply type two question marks (??). All the legal ENTRO conventions are listed in **The INFORM Reference Guide.**

SUMMARY OF INFORMATION VERBS

All the INFORMATION verbs you are likely to need are summarized in the following table. For a complete description of verb syntax and usage, see the indicated reference guide. For convenience, the verbs are divided into categories by function, so all the file handling verbs are together, all the system status verbs are together, and so forth. Also included in this list are the 12 stored system sentences that appear in everyone's VOC file.

Note

The system HELP facility will give you a brief description of anything that's listed here if you type:

HELP verb-name

where **verb-name** is any name that appears in the list below.

Getting Around

Command Name	Meaning	Reference
DISPLAY	Displays a specified message on your terminal: used in paragraphs	PERFORM Guide
HELP	Invokes HELP facility	PERFORM Guide
IAM	Tells PERFORM which account name should be used by commands which require one	PERFORM Guide
LO	An abbreviation for LOGOUT — exits you from INFORMATION and logs you off the system	PERFORM Guide
LOGOUT	Exits you from INFORMATION and logs you off the system	PERFORM Guide
LOGTO	Lets you "move" to another INFORMATION account	PERFORM Guide
MAIL	Invokes mail-sending/-reading routine	PERFORM Guide
MESSAGE	Sends message to system console	PERFORM Guide
PASSWD	Specifies new owner and non-owner passwords for an account	PERFORM Guide
QUIT	Gets you out of INFORMATION and leaves you at PRIMOS level	PERFORM Guide
SLEEP	Suspends execution for a specified period of time	PERFORM Guide

System Status and Maintenance Commands

Command Name	Meaning	Reference
AVAIL	Displays number of physical disk records available for use	PERFORM Guide
CHAP	Raises or lowers your task execution priority on system	PERFORM Guide
CLEAN.ACCOUNT	Performs routine maintenance and checks integrity of files in your account	PERFORM Guide
CLEAR.LOCKS	Clears a specified lock, or all system locks set by your account name	PEFORM Guide

DATE	Displays system date	PERFORM Guide
LIST.LOCKS	Displays status of the 64 synchronization locks	PERFORM Guide
LISTME	Returns status of all users with your LOGIN name	VOC File
LISTU	Displays names of all users currently on the INFORMATION system	VOC File
LOCK	Locks one of 64 task synchronization semaphores	PERFORM Guide
MAKE.MAP.FILE	Loads catalogued program map records into &MAP& file	PERFORM Guide
MAP	Displays a map showing contents of system catalog	PERFORM Guide
MASTER	Releases locks, controls breaks, and enables forced logouts: for System Administrator only	PERFORM Guide
PTIME	Invokes the PRIMOS TIME command	PERFORM Guide
STATUS	Invokes PRIMOS STATUS command	PERFORM Guide
TIME	Displays current system time and date	PERFORM Guide
USERS	Displays number of users currently on system	PERFORM Guide
WHO	Displays your port number and account name	PERFORM Guide

File and Record Related Commands

Command Name	Meaning	Reference
CREATE.FILE	Allocates space for and names data and/or dictionary files	PERFORM Guide
CLEAR.FILE	Deletes all records in an INFORMATION file	PERFORM Guide
CNAME	Changes name of a file or changes names of records withing a file	PERFORM Guide
COPY	Copies records from one file to another, or from a file to itself: i.e., duplicates records	PERFORM Guide
COUNT	Counts number of records in file	INFORM Guide

DELETE	Deletes records from a file	PERFORM Guide
DELETE.FILE	Deletes an INFORMATION file	PERFORM Guide
GROUP.STAT	Displays file statistics in histogram form	PERFORM Guide
GROUP.STAT.DETAIL	Produces detailed record distribution summary for a file	PERFORM Guide
HASH.HELP	Recommends file type and modulo for an existing file, based on record id and file size	PERFORM Guide
HASH.HELP.DETAIL	Same as HASH.HELP but displays details on key size and record size (in bytes)	PERFORM Guide
HASH.TEST	Displays record distribution histogram for a user-suggested file type or modulus	PERFORM Guide
HASH.TEST.DETAIL	Same as HASH.TEST but displays number of bytes per record, bytes per group, etc.	PERFORM Guide
LIST	Invokes INFORMATION query and report generator	INFORM Guide
LIST.READU	Lists records locked by INFO/BASIC READU, READVU or MATREADU commands	PERFORM Guide
RECORD	Determines which group a given record would belong in	PERFORM Guide
RELEASE	Releases a specified record lock, all locks in a file, or all locks in your account	PERFORM Guide
RELEASE.ITEMS	Releases all locks in account	PERFORM Guide
RESIZE	Reorganizes a file with a new file type/modulus	PERFORM Guide
SETFILE	Lets user create synonym for an existing file	PERFORM Guide
SORT	Sorts a file by record id	INFORM Guide

Printer Commands

Command Name	Meaning	Reference
BLOCK.PRINT	Prints specified words in big block letters on line printer	PERFORM Guide
P.ATT	Requests exclusive control of some line printer	PERFORM Guide
P.DET	Releases a previously assigned printer (assigned with P.ATT)	PERFORM Guide
PROP	Invokes PRIMOS spooler status command	PERFORM Guide
RESET.PRINTER	Restores PERFORM environment during execution of an INFO/BASIC program or a paragraph	PERFORM Guide
SETPTR	Sets characteristics for any of up to 255 logical printers	PERFORM Guide
SP.KILL	Cancels spooled files	PERFORM Guide
SPOOL	Puts named file on printer queue	PERFORM Guide
SP.STATUS	Displays status of files on spool queue waiting to be printed	PERFORM Guide

Terminal Specific Commands

Command Name	Meaning	Reference
BLOCK.TERM	Prints text in big block letters on terminal screen	PERFORM Guide
CLR	Erases a video display: same as CS (designed for PT25 terminal)	PERFORM Guide
CS	Same as CLR	PERFORM Guide
DATE.FORMAT	Sets default display of current date to international format	PERFORM Guide
DELAY	Sets timing characteristics for hard copy terminals	PERFORM Guide
HUSH	Turns "hush mode" on or off: nothing echoes at terminal while hush mode is on	PERFORM Guide
PTERM	Sets terminal characteristics (PRIMOS command)	PERFORM Guide
TERM	Sets/displays terminal and/or line printer characteristics	PERFORM Guide

Tape Command

Command Name	Meaning	Reference
ASSIGN	Assigns a physical device to you	PERFORM Guide
SP.TAPE	Reads a SPOOL tape created with SETPTR mode 4 and prints tape contents directly on printer	PERFORM Guide
T.ATT	Dedicates a physical tape drive to your user id	PERFORM Guide
T.BCK	Backspaces tape a specified number of blocks	INFORM Guide
T.DET	Unassigns a physical tape drive	PERFORM Guide
T.DUMP	Dumps records from INFORMATION file to tape	INFORM Guide
T.FWD	Advances tape a specified number of blocks	INFORM Guide
T.LOAD	Loads INFORMATION files from tapes created by T.DUMP	INFORM Guide
T.READ	Reads magnetic tape previously created by T.DUMP program and prints it on CRT or printer	INFORM Guide
T.REW	Rewinds tape on assigned drive	INFORM Guide
T.WEOF	Puts an EOF mark on magnetic tape	INFORM Guide
UNASSIGN	Unassigns a physical device currently held by you	PERFORM Guide

Stored System Sentences

Command Name	Meaning	Reference
LISTF	Lists all files defined in your VOC file	PERFORM Guide
LISTFL	Lists all files which are local to this account	PERFORM Guide
LISTFR	Lists all files defined in VOC file which are not local to your account	PERFORM Guide
LISTK	Lists all keywords defined in your VOC file	PERFORM Guide
LISTM	Lists menu selector records in VOC file	PERFORM Guide
LISTO	Lists all entries in VOC file which are not one of the seven reserved types	PERFORM Guide

LISTPA	Lists stored paragraphs in your VOC file	PERFORM Guide
LISTPH	Displays all phrases in your VOC file	PERFORM Guide
LISTR	Displays all remote references contained in VOC file	PERFORM Guide
LISTS	Lists all sentences in VOC file	PERFORM Guide
LISTV	Lists all verbs in VOC file	PERFORM Guide
VVOC	Compares contents of VOC file to those of NEWACC file, reporting any differences	PERFORM Guide

Select List Commands

Command Name	Meaning	Reference
CLEARSELECT	Terminates an active select list	PERFORM Guide
COPY.LIST	Copies a select list from &SAVEDLISTS& file to another file	PERFORM Guide
DELETE.LIST	Deletes a select list from the &SAVEDLISTS& file	INFORM Guide
EDIT.LIST	Edits a select list saved in the &SAVEDLISTS& file	PERFORM Guide
FORM.LIST	Activates a user-built (with an INFO/BASIC program) select list	PERFORM Guide
GET.LIST	Reactivates a stored select list	INFORM Guide
SAVE.LIST	Saves a select list in the &SAVEDLISTS& file	PERFORM Guide
SELECT	Creates a list of records which meet specified selection criteria	INFORM Guide
SELECTFL	Makes a select list of local files in your account	PERFORM Guide
SSELECT	Creates a sorted list of records meeting some selection criteria	INFORM Guide

Dictionary Commands

Command Name	Meaning	Reference
CD	Compiles the I-descriptors in an INFORMATION dictionary file	PERFORM Guide
COMPILE.DICT	Compiles all I-descriptors in a dictionary file (same as CD)	PERFORM Guide
LIST.DICT	Displays contents of dictionary file sequenced by record type	PERFORM Guide
PRINT.DICT	Same as LIST.DICT except the listing is sent to line printer	PERFORM Guide

INFO/BASIC Commands

Command Name	Meaning	Reference
BASIC	Invokes INFO/BASIC compiler	PERFORM Guide
CATALOG	Moves compiled INFO/BASIC object code to system catalog	PERFORM Guide
DELETE.CATALOG	Removes named executable program from system catalog	PERFORM Guide
FORMAT	Formats an INFO/BASIC source file into logical block structure	PERFORM Guide
RUN	Loads and executes a named INFO/BASIC program	PERFORM Guide
VCATALOG	Compares a program stored in the system catalog to object code stored in another file	PERFORM Guide

Miscellaneous Commands

Command Name	Meaning	Reference
COMO	Opens a file to record all subsequent terminal output	PERFORM Guide
ED	Invokes the INFORMATION Editor	EDITOR Guide
ENTER	Invokes cursor control-independent processor for entering/modifying dictionary and/or data file records	INFORM Guide
ENTRO	Same as ENTER, above	INFORM Guide

ENTROC	Invokes cursor control-dependent processor for entering/modifying dictionary and/or data file records	INFORM Guide
PHANTOM	Starts up a pre-defined process like a pre-stored paragraph, sentence or PERFORM command that requires no terminal input	PERFORM Guide
RADIX	Converts from one number base to another: includes dec. to bin., dec. to hex., dec. to octal, and vice-versa	PERFORM Guide

DOING THINGS

Most users want to do a few basic things on a routine basis. The table below describes some of these common activities and briefly explains how to accomplish them. You won't find everything you'd want to do here, but it will get you started.

Desired Action	How It's Done
Listing Verbs in your Voc file	Type: **LISTV**
Listing all files defined in your VOC file	Type: **LISTF**
Listing sentences stored in your VOC file	Type: **LISTS**
Listing phrases stored in your VOC file	Type: **LISTPH**
Listing paragraphs stored in your VOC file	Type: **LISTPA**
Listing keywords stored in your VOC file	Type: **LISTK**
Listing all remote items in your VOC file	Type: **LISTR**
Listing all record ID values (in data file)	Type: **LIST DICT filename** Uses @ phrase, if it exists, otherwise lists record IDs.

Listing a particular file record	Type: **LIST filename record-ID** is the key value of the sought record.
List all data file records	Type: **LIST filename @ENTRO** (Uses default @ENTRO phrase)
Changing data file values	Type: **ENTRO filename** ENTRO prompts for record ID value then prompts for changes. or Type: **ED filename record-ID** Find the line to be changed. Use EDITOR commands to make changes. Save changes with FILE command.
Changing records in a dictionary file	Type: **ENTRO DICT file-name** ENTRO asks for field name with "FIELD=" prompt. It then prompts for changes. or Type: **ED DICT filename [field-name]** The EDITOR will prompt for a field-name if you didn't specify one. Make changes with EDITOR commands and save changes with FILE command.
Changing record ID values	Type: **ED filename record-ID** Then type: **SAVE new-record-ID** Then type: **FD** (to delete old record) Say "YES" when EDITOR asks if it's OK to delete old record. or Type: **CNAME fname old-ID new-ID** where **old-ID** is the record ID value that you want changed to **new-ID.**
Deleting data file records	Type: **ENTRO filename** ENTRO asks for record ID value. Type **DELETE** as response to ENTRO change prompt. or Type: **ED filename [record-ID]** Then type: **DELETE** in response to the EDITOR prompt (----:).

Deleting a dictionary file record

Type: **ED DICT filename**
EDITOR asks for record ID value.
Type: **DELETE** in response to CHANGE= prompt.

or

Type: **ENTRO DICT filename**
ENTRO prompts for a field-name.
Type: **DELETE** in response to "CHANGE=" prompt.

Checking to see if file has proper type and modulus

Type: **HASH.HELP filename**

Creating a menu

Type: **MENUS**
A menu appears offering a choice of 17 options. After you define your menu, using options 1 or 2, you must create a menu pointer in your VOC file that points to the menu you just defined. Use option 5 to do this.

Deleting files

Type: **DELETE.FILE** { DATA / DICT } filename

Printing INFO/BASIC programs, COMO files

Type: **SPOOL filename [record-IDs]**
If no record IDs are given, SPOOL looks for an active select list; if none exists, no action is taken. Used mainly to print Type 1 files.

Generating printed reports

Use LPTR option with LIST, HASH.HELP, GROUP.STAT, et al.

Storing a sentence in your VOC file

Type: **.S [sent-no] name**
sent-no is the number of the sentence in your PERFORM stack that you want to save. **name** is the name under which the sentence will be stored. See **The PERFORM Stack,** below.

Saving paragraphs

Same as saving a sentence except you indicate the line numbers of consecutive stack sentences that you want saved in the paragraph.

PERFORM STACK COMMANDS

All the commands you type at PERFORM level are automatically stored in what is called the PERFORM stack. The stack is a list of the last 99 commands you typed and are available for you to change, save and re-execute. Each stack item is numbered consecutively, beginning with the most recently typed item, which is stack item 1 (also known as sentence 1). The special commands available to operate on stack items are summarized below.

You can put an item on the stack without actually executing it by typing a question mark (?) before hitting the RETURN key. Use the stack edit commands to change and/or save it for future use.

PERFORM STACK COMMANDS

COMMAND	FORMAT	WHAT IT DOES
.?	.?	Displays list of all stack commands along with their formats and a brief explanation of what each one does.
.A	.A[nn] string	Appends string to end of stack sentence number **nn**. If nn is not specified, string is appended to sentence 1 in that stack.
.C	.C[nn]/str1/str2	Changes first string, **str1**, to second string, **str2**, in sentence nn. If **nn** is not specified, change is made in sentence 1.

.D	.D[nn]	Deletes sentence number **nn** from the stack. Default is delete sentence 1 from stack.
	.D name	The second format deletes the named sentence or paragraph from your VOC file.
.I	.I[nn] new-sent	Inserts the sentence indicated by **new-sent** into your stack just prior to sentence **nn**. The new sentence will then become sentence number nn. If no nn specified, it become sentence 1.
.L	.L[nn]	Lists **nn** lines from your stack. The default is 20 lines if no nn is given.
	.L name	The second format lists the contents of the sentence or paragraphs indicated by **name** as stored in your VOC file.
.R	.R[nn]	Recalls indicated sentence from your stack and makes it current sentence. If no **nn** given, recalls sentence 1.
	.R name	Take sentence or paragraph from your VOC file and loads it into your stack. It becomes sentence 1.
.S	.S name n1 n2	Saves lines **n1** through **n2** of stack under **name** and stores it in your VOC file. Does not remove lines from stack.
.X	.X[nn]	Executes sentence number **nn** of your stack. Sentence 1 is executed if nn is not specified.

THE EDITOR

Aside from its primary function of creating INFO/BASIC programs, the EDITOR is useful for adding phrases, sentences, etc., to the VOC file, for changing record ID values and for modifying multiple records with a select list. The EDITOR operates on one record of a file at a time on a line-by-line basis. An imaginary "line pointer" is kept by the EDITOR to mark its current position within a record. Thus each line in a record has a unique number by which it can be identified. Although the EDITOR is line-oriented, it can operate on a block of lines. A block consists of one or more consecutive lines in a record. Blocks can be moved, copied, deleted or changed as a unit. All commands applied to a block affect all the lines in that block.

The EDITOR Stack: In addition to the regular edit commands, there are special "stack" commands which allow you to edit commands you've previously issued. A stack of the last 99 commands you've entered is continuously maintained by PERFORM and is available throughout an editing session. You can recall, modify and re-execute any command in this stack with the special stack commands shown below.

External Command Execution: With the XEQ feature, you can execute any legal PERFORM command from within the EDITOR without leaving the EDITOR environment. See the EDITOR's HELP file for details.

HELP In The EDITOR: A complete, up-to-date EDITOR HELP facility is always available from within the Editor environment. Type: HELP to obtain the formats and meanings of all EDITOR commands and stack commands. It also summarizes the pre-stored command feature which involves storing edit commands in a special "Edit file" that can be executed any time, much like a program or command file.

Chart of EDITOR Commands

EDITOR Commands

Command	Format	Meaning
A (APPEND)	**A[str]**	Appends given string, **str** to current line. If **str** is not supplied, the last APPEND command is executed again.
B (BREAK)	**B[str]**	Breaks current line into two lines. Current line includes everything up to **str** and the rest of the line becomes the next line in the record.
B (BOTTOM)	**B**	Puts line pointer to bottom of record.
C (CHANGE)	**C[/str1/str2/] [nn] [G]**	Changes **str1** to **str2** in current line. Ending delimiter (/) is not required. If no arguments specified, the last C command will be repeated. **G** means change all occurrences of **str1** to **str2**. **nn** is the number of lines in which to execute the change. See **Block Commands.**
CAT (CATENATE)	**CAT[str]**	Joins current line with next line in record, optionally joining them by **str.**
COL (COLUMN)	**COL**	Displays relative column positions on terminal screen.
D or DE (DELETE LINE)	**{D / DE} [nn]**	Deletes indicated number of lines (**nn**) from current record, beginning with current line.
DELETE (DELETE) RECORD)	**DELETE**	Deletes current record. EDITOR displays message giving user a chance to cancel this command.
DUP (DUPLICATE)	**DUP [nn]**	Duplicates the current line one or more times, as indicated by **nn.**
EX (EXIT)	**EX**	Exits you from EDITOR without writing current record back to file. Synonym is QUIT (Q).
FD (DELETE (RECORD)	**FD**	Deletes entire record from file. Same as DELETE command, above.
F (FIND)	**F [str]**	Finds first line in record which begins with **str.** If **str** not specified, the last FIND command is executed.

FILE **(FILE)**	**{FI {FILE}[fname] [record]**	Files current record to specified file (**fname**) under specified record name, **record**. Default is to file current record under the current file under its present record name. If a select list is active or more than one record ID was specified with the EDIT command, the next record is brought up. Otherwise, user exits EDITOR.
FORMAT	**FORMAT**	Formats a program so FOR/NEXT loops and IF/THEN structures will be indented to show nesting levels. For use in editing INFO/BASIC programs.
G **(GO TO)**	**[G]nn**	Sets line pointer at indicated line number. "G" is optional.
HELP	**HELP{init-char {str**	Displays syntax of an EDITOR command at the terminal. **init-char** displays all commands beginning with this character. **str** causes all commands in which this string (**str**) appears to be displayed.
I **(INSERT)**	**I[line]**	Inserts indicated line into current record. If no **line** is indicated, the EDITOR will accept lines of input from the terminal until a null line (CR) is entered.
L **(LOCATE)**	**L[str]**	Locates next line which contains **str**. If no **str** is given, does last Locate again.
LIST **(LIST)**	**Lnn**	Lists **nn** number of lines beginning with current line.
LOAD	**LOAD [fname] recname**	Loads one or more lines from another record into current record. If **fname** is specified, EDITOR looks for indicated record in that file; otherwise, the current file is searched. In either case, the EDITOR prompts for line numbers to load.
M **(MATCH)**	**M[str]**	Searches for next line that matches pattern indicated by **str**. If no **str** is specified, the last Match command is executed.

N	**N**	Used only when select list is active. Moves line pointer to next selected record. The record is not updated (similar to "QUIT" command).
OOPS	**OOPS**	Restores record to condition prior to last modification. Must be issued before the next modification command, but can be used after any non-modifying command (like L or P) has been issued.
P	**P[nn]**	Prints **nn** number of lines starting with the current line. If no **nn** specified, current line is displayed. (No space is allowed between P and **nn**).
PO (POINT)	**[PO]nn**	Moves line pointer to indicated line (**nn**) in current record. "PO" is optional. (Spaces are allowed between PO and **nn**.)
QUIT	**Q[UIT]**	Exits the Editor without making changes to the current record. If current record is a new one it will not be written to disk.
R (RETYPE)	**R/str1/str2/**	Same as C (CHANGE) command, above.
RELEASE	**RELEASE**	Releases the update lock currently active for this record.
SAVE	**SAVE [fname] [recname]**	Saves current record to indicated file (**fname**) under specified name (**recname**). Default is to file record under its present name in the current file. User remains in the EDITOR.
SEQ (SEQUENCE)	**SEQ/str/nn/**[**##** / **B**]**[inc]**	Replaces the indicated string (**str**) with sequential numbers beginning with **nn** and continuing in increments of 1, unless **inc** is specified, for **##** number of lines. The default is to change current line only. Use **B** option to effect change throughout a block.
SIZE	**SIZE**	Displays size of the current record, including number of lines, fields and bytes.

SPOOL	**SPOOL**	Spools current record to line printer. **nn** specifies number of lines, beginning with current line, to print. Default is to print entire current record.
SPOOLHELP	**SPOOLHELP**	Spools the entire HELP file to printer.
UNLOAD	**UNLOAD [fname] recname**	Saves a portion of a record under indicated name **recname** in the same file or another file (**fname**). EDITOR then prompts for starting and ending line numbers to unload from record.
X **(EXIT)**	**X**	Exits user from the EDITOR and abandons an active select list. Can only be used when a select list is active.
? **(INQUIRE)**	**?**	Displays filename, record name, current line number, last CHANGE command executed, currently defined block and status of up-arrow mode.
+	**+nn**	Advances line pointer by **nn** lines.
-	**-nn**	Backs line pointer up by **nn** lines.
↑ **(up-arrow)**	**↑**	Turns on/off up-arrow mode to allow display of ASCII codes of non-printing characters.

The EDITOR's Block Commands

Several of the EDITOR's commands are meant for use with blocks only. A block is a series of lines within a record that can be copied, deleted, changed, or moved as a unit. Use the < and > symbols to define the boundary lines of the block. To set the beginning Line of a block, move the line pointer to the desired line by typing PO **nn** or **nn**, where **nn** stands for the line number in the record. Then type the < symbol. Similarly, define the last line of the block by setting the line pointer to the desired line and hitting the > symbol. Only one block can be active at a time.

The special block commands are listed below.

BLOCK COMMANDS

Command	Format	Meaning
<	<	Defines the first line of a block.
>	>	Defines the last line of a block.
C	C/str1/str2/B	Changes all lines in this block that contain the indicated string, **str1**.
COPY	COPY	Copies a pre-defined block. The source block is not altered. EDITOR asks user to verify the COPY.
DROP	DROP	Deletes all lines in a pre-defined block. EDITOR asks you to verify the DROP.
MOVE	MOVE	Moves a block from one position in the record to another. It esentially does a COPY and a DELETE. The current block will remain the current block after the MOVE: it will just be in a different place.
PB	PB	The PRINT BLOCK command displays the currently defined block.

EDITOR Stack Commands

Like PERFORM, the INFORMATION EDITOR also keeps track of everything you type at the terminal. The last 99 commands you enter are saved in an EDITOR command stack, making them available for you to change and re-execute. (Do not confuse the PERFORM and EDITOR command stacks — they are separate entities. All lines in the stack are numbered consecutively, beginning with line 1 which is the most recent command you've issued. There are eight stack commands which you can use to manipulate the stack items. They are shown below.

Command	Format	Meaning
.A	.A[nn] [str]	Adds indicated string, **str,** to indicated command line (**nn**). If **nn** is not specified, default is to append string to command 1.
.C	.C[nn]/str1/str2/	Changes string **str1** to **str2** in command line **nn.** Default is to change string in command line 1.
.D	.D[nn]	Deletes indicated command line. Default is to delete command line 1.
.I	.I[nn] [com]	Inserts indicated command line, **com** below line **nn.** If **com** isn't specified, EDITOR prompts for input until a null line is entered. If **nn** isn't specified, **com** or input lines are inserted before stack line 1.
.L	.L[nn]	Lists indicated number of lines from stack. Default is to list 21 lines.
.R	.R[nn]	Recalls indicated command line from stack. Default is to recall stack line 1.
.X	.X[nn]	Executes command line **nn** from stack. Default is to execute stack line 1.

Note

Line 1 of the stack is the most recently issued command.