GA21-9194-3

IBM 3741
Models 3 and 4
Programmable Work Station
Programming Reference
Manual

Systems

IBM

Preface

This manual is primarily intended for marketing representatives, systems engineers, programmers, and operators who deal with the IBM 3741 Models 3 and 4 Programmable Work Station. The reader should be familiar with data processing concepts and the operation of the IBM 3741 Models 1 and 2 Data Station.

The manual is divided into the following chapters:

Chapter 1. Introduction provides general information about the overall structure and new functions of the work station and the ACL (application control language).

Chapter 2. Reference Material provides specific information about the control statements and instructions of ACL used in programming the work station.

Chapter 3. Design and Implementation Considerations provides detailed information and examples of ACL used in designing and implementing application programs.

Chapter 4. 3741 Operation provides information about operator procedures and interaction with the work station.

Appendixes provide detailed supplementary information.

Related Publications

- *IBM 3741 Models 3 and 4 Programmable Work Station, General Information,* GA21-9196.
- *IBM 3741 Data Station Operator's Guide,* GA21-9131.
- *IBM 3741 Data Station Reference Manual,* GA21-9183.
- *IBM 3741 Models 3 and 4 Programmable Work Station Reference Card,* GX21-9204.
- *The real measure . . . A Programmer's Design Guide to the IBM 3741 Models 3 and 4 Programmable Work Stations,* GA21-9229.
- *A Programmers Introduction to the Application Control Language,* GA21-9195.
- *Application Control Language Support Logic Manual,* SY21-9203.
- *IBM 3740 BTAM/TCAM Programmer's Guide,* GC21-5071.

Fourth Edition (October 1977)

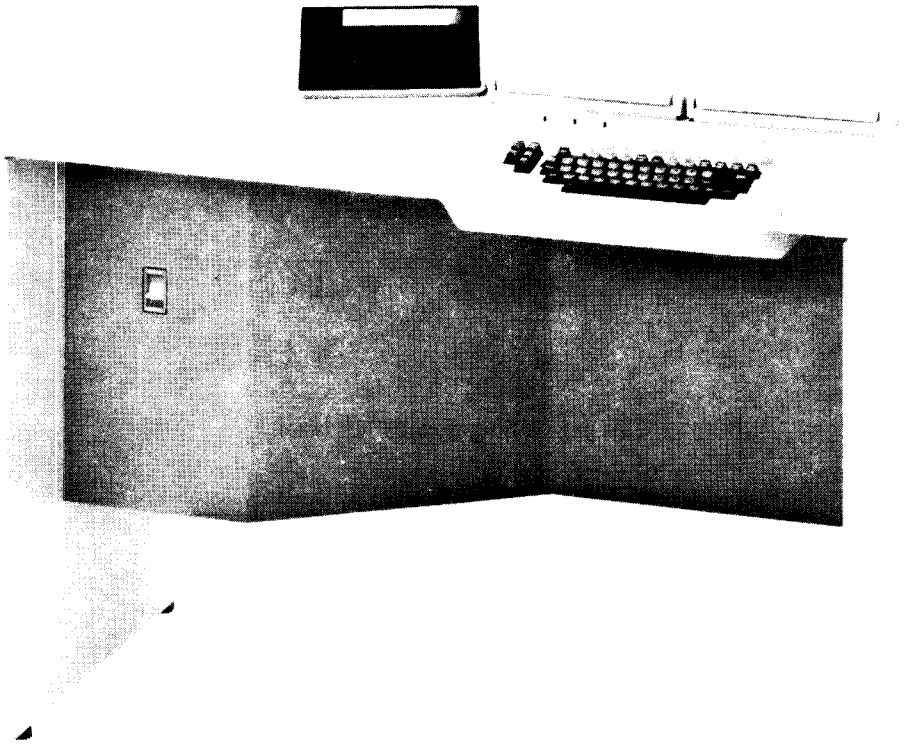
This is a major revision of, and obsoletes GA21-9194-2 and Technical Newsletter GN21-0244. Changes are indicated by a vertical line at the left of the change; new or extensively revised illustrations are denoted by a bullet (●) at the left of the figure caption.

Changes are periodically made to the information herein; any such changes will be reported in subsequent revisions or technical newsletters.

Requests for copies of IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

A Readers' Comment Form is at the back of this publication. If the form has been removed, address your comments to IBM Corporation, Publications, Department 245, Rochester, Minnesota 55901.

This page is intentionally left blank.



IBM 3741 Programmable Work Station

The IBM 3741 Models 3 and 4 Programmable Work Station substantially increase the data entry capabilities of the basic 3740 system. Significant additional functions beyond the scope of the IBM 3741 Models 1 and 2 Data Station are provided through a programming facility called ACL (application control language). The noncommunicating 3741 Model 3 provides functions beyond those of the corresponding 3741 Model 1, while the communicating 3741 Model 4 provides functions beyond those of the corresponding 3741 Model 2.

The work station provides programmable functions which can be adapted to new data entry applications, or be used to improve processing of existing jobs.

The work station provides the following new functions:

- *Expanded arithmetic* functions, including add, subtract, multiply, and divide operations, which enable additional field totals, zero balancing, and crossfooting operations
- *Data checking* operations, including range checking, limit checking, and table searching, which provide improvements in data accuracy
- *Data manipulation* which allows data entered from the keyboard to be reformatted according to instructions before being written onto a diskette, displayed, or printed
- *Additional keyboard and display* functions which allow messages to be displayed, including prompts for data entry, prompts for option selection, and prompted error correction
- *Additional diskette access* methods which allow processing of multiple data sets, reading and writing on two diskette drives, and creation and maintenance of data set indexes that provide fast access to online data
- *Overlapped printing* which allows the operator to key data while data within the system is being printed by any of the available printers attached

The 3741 Models 3 and 4 Programmable Work Station has all these functions in addition to the functions now available with the 3741 Models 1 and 2 Data Station. The work station is also compatible with all optional features currently available with the data station. When not operating under ACL program control, the work station functions exactly like the data station.

An optional ACL translator feature, which translates coded ACL source programs into machine-readable object code, is available. The ACL translator feature is only required on those work stations being used to generate object-level programs. These object-level programs can then be executed on work stations at remote locations (Figure 1). Both the printer and second disk features are prerequisites for the ACL translator feature. Work stations with the ACL translator feature should also be equipped with the optional 3741 record insert feature for easy source program maintenance.

Creating a Source Program

A source program consists of ACL control statements and instructions. Control statements define the program name, define and describe the data sets with which the program works, define the data set access method, provide printer control information, and define the prompting message and self-check algorithm. In addition, these statements specify data manipulation formats, constant information, and designate the end of the program.

The second type of program source is instructions. Instructions specify how operations are to be executed. Instructions can initiate I/O operations, internal data movement, branching within the program, and arithmetic operations. Instructions can either be preceded by step numbers (0-767 for a 4K machine, or 0-999 for an 8K machine) or symbolic labels in the first column on the coding sheet. Symbolic labels can be up to four characters long, but the first character must be alphabetic. The succeeding characters can be alphabetic, numeric, or symbol characters. When symbolic labels are used, the first phase of translation is label processing, during which labels are resolved and converted into step numbers.

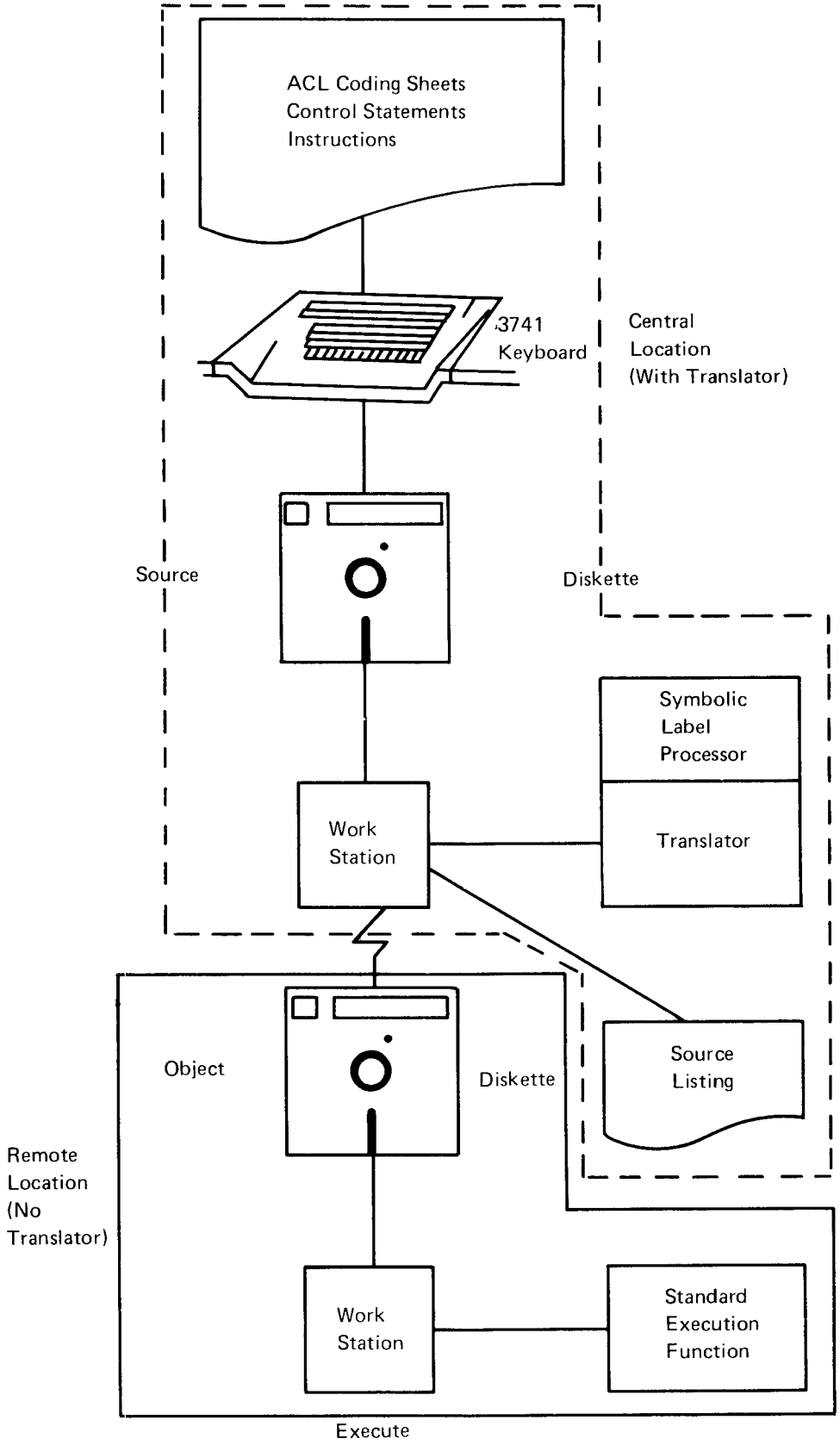


Figure 1. Structure of the 3741 Models 3 and 4

Comments can be included in the source program to clarify instructions. Control statements can be entered on *ACL Control Statements Coding Sheet One* and *Two,* GX21-9200 and GX21-9201. Instructions can be entered on *ACL Instructions Coding Sheet,* GX21-9199. These sheets can be ordered, by order number, in pads of fifty from your IBM marketing representative. For a detailed discussion of coding for control statements and instructions, see Chapter 2.

3741 Models 3 and 4 Architecture

The standard 3741 Model 3 or 4 is structured in a 4K-byte (K equals 1024) buffer/register concept (Figure 2). The first 1024 bytes of memory are allocated for system control. This block of memory also contains the 26 general purpose registers (A-Z). These 16-character registers are used to perform all arithmetic operations, and can also be specified to move data internally and to store data. The indicator table contains 255 indicators; indicators 1 through 99 can be assigned by the programmer.

The balance of read/write storage is allocated to 128-byte buffers (Figure 2). These buffers are referenced numerically (1-24) in program source statements. Although buffer 1 and buffer 2 are assigned to the display, the remaining buffers can be used to store operator prompting messages, keyboard control information, tables and data set indexes, and online disk data sets and printer output. (See *Storage Allocation and Requirements* in Chapter 3.)

An optional 8K storage feature provides an additional 32 general purpose buffers (128 bytes each). Throughout this manual, any reference to buffers refers to buffers 1-24 for 4K or 1-56 for 8K. The 8K feature allows up to 999 executable instructions with an instruction length of 4 bytes. All other areas of work station architecture (Figure 2) are unchanged by the feature, except that data formats are loaded starting in buffer 56.

A basic 4K-byte program can be translated and executed on a work station with the 8K feature. However, an 8K-byte program cannot be translated or executed on a 4K work station.

After you have completed the coding sheets, and created a source program on the diskette via the 3741 keyboard, the ACL translator feature creates object code from your coded entries. Translation occurs in two distinct phases of operation; the label processor phase, and the translator phase. During the label processor phase, the symbolic labels that precede instructions are converted into step numbers for internal processing. The source file listing of the program can also be printed during the label processor phase. During the translator phase of operation, the instructions you enter are converted into object code, and a listing of error messages is always printed out. (See *Initiating Translation With the Label Processor* in Chapter 4.) Note that, if step numbers are used in a source program, the label processor phase of translation must be bypassed.

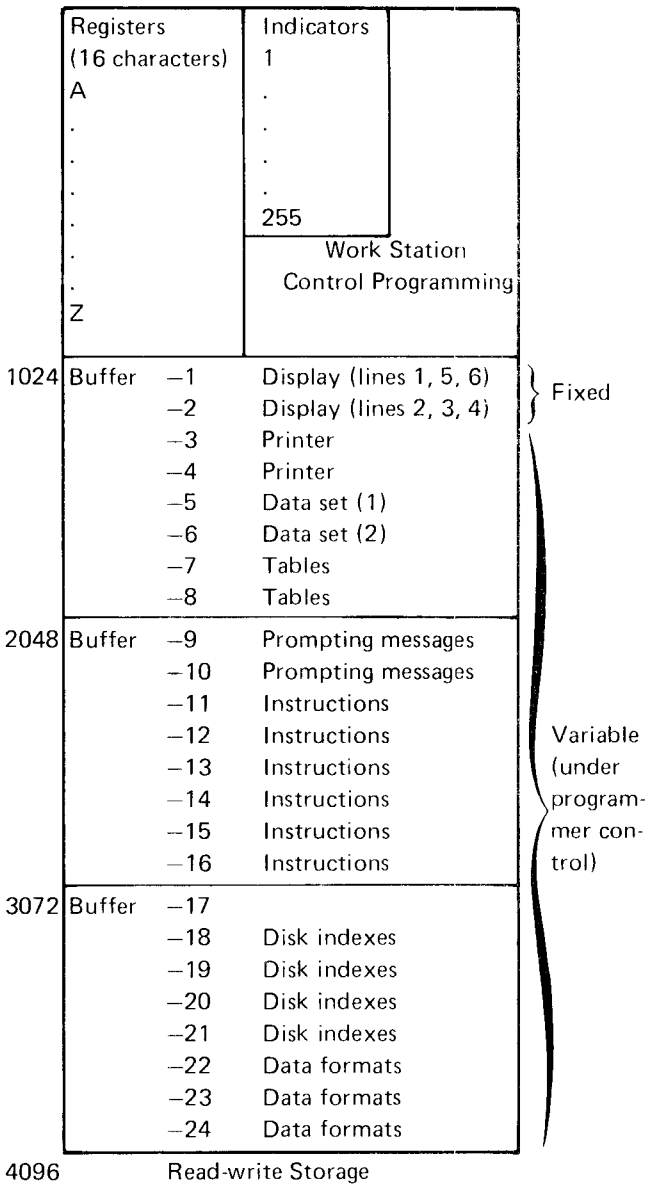


Figure 2. Typical ACL Program Structure (4K)

ACL PROGRAM OPERATION

As indicated previously, the work station combines the capabilities of the 3741 Models 1 and 2 Data Station with new capabilities provided by ACL. The following operations are available through appropriate implementation of the application control language.

Branching

Unconditional and conditional branching instructions alter the sequential execution of the ACL program. All executable instructions in the program can be identified with a step number (000-767), or a four-position symbolic label preceding the instruction. Only those instructions branched to must have a preceding step number or label; all other instructions do not require step numbers or labels. The step numbers or labels are used in branching instructions to indicate the instruction to be branched to if specified conditions are met. Branching instructions are described in detail under *Branching Operations* in Chapter 2. These instructions can also be used to aid in program debugging (see *Program Debugging* in Chapter 4).

Data Set Access Methods

The work station and the data station use the same data set label. On the work station, the data sets are used to store both ACL programs and data sets. During execution of an ACL program, four data sets can be online to the program at one time.

The ACL programming facility allows you to access data sets by three methods:

1. Sequential or consecutive
2. Random by relative record number
3. Key indexed

The *sequential or consecutive* access method requires that the records (or 128-position sectors) be processed sequentially or consecutively, based on the physical disk address. Given one record, the location of the next record is determined by the next sequential disk address in the data set. This method allows:

- Writing records into a new data set.
- Writing or adding records at the end of an existing data set.
- Reading records from an existing data set.
- Reading and updating records in an existing data set.
- Reading and updating records in an existing data set, and adding new records at the end of an existing data set.

The *random by relative record number* access method performs the same read and update operations as the sequential access method. The difference is that record accessing is done in a direct manner, thereby reducing or minimizing the diskette search time. A relative record number, specified during a read/write operation, designates the record to be processed. For example, an instruction to read the fifty-eighth record in a data set can be issued.

The *key indexed* access method requires that the records in the data set be arranged in ascending sequence, according to the search argument or control field. The search argument is a control field or a data element, up to 16 positions in length, within each record of a data set. The work station control program automatically builds an index table in storage for the data set to be accessed when the key indexed method is specified.

Table Operations

A table is a group of contiguous fields of the same length (for example, a table of seven-digit customer numbers). The maximum length of a table is not restricted (except by available storage). Tables are read into one or more general working buffers. Each buffer may contain from 1 to 16 tables. The following table operations may be specified:

- Search for an equal entry in a nonsequential table
- Search for an equal entry or the next higher entry in a table sequenced in ascending order
- Read a specific entry from a table
- Write a specific entry into a table

Self-Checking

The ACL program provides a broad range of self-checking capabilities in addition to the standard modulus 10 and 11 available on the base data station. A specific control statement contains the self-check parameters. See *Control Statements* in Chapter 2. The contents of any register can be self-checked during program execution. Self-check numbers can also be generated.

Display and Keyboard

The display (Figure 3) provides station status information, program information, data, and operator prompting on the display screen. The work station uses six 40-character lines of the display screen. Line 1 is used for station status halts and error messages (positions 5-8 are reserved for error messages). Positions 1-4 of line 1 can be used for user-posted halts. Lines 2, 3, and 4 display up to 120 character positions of data or program information. The last 30 positions of line 5 and all positions of line 6 are used to display prompting information (operator guidance) and the operator's response to the prompt.

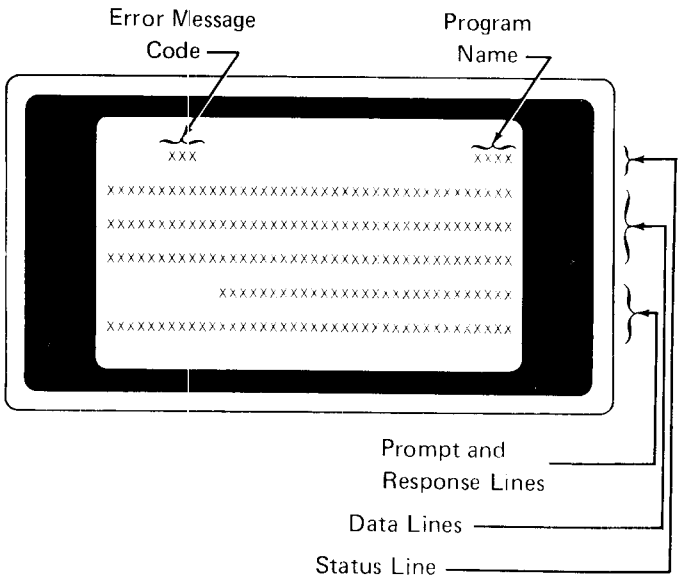


Figure 3. Display Unit

Figure 4 shows an example of a message which is prompting the operator to enter a name and address. Prompting begins in position 11 of line 5. A blank is inserted after the prompting message, followed by a series of periods or dashes which indicate the length and type of data that the operator should enter. Periods indicate that alphabetic data should be entered, and dashes indicate that the operator should enter numeric data. The prompting message and response can total a maximum of 69 characters. All data keyed by the operator is displayed in the area defined by dashes or periods. As each character is keyed, it replaces the next dash or period. The work station display unit does *not* utilize a cursor.

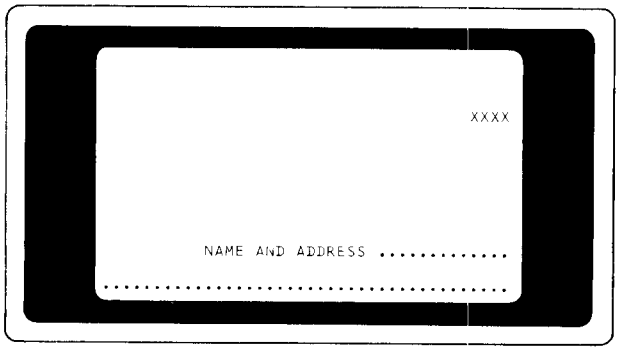


Figure 4. Example of a Prompt

As indicated, a prompting message can be displayed for each field to be entered by the operator. The prompting message and keyboard control information are stored in the general working buffers. A specific control statement contains keyboard control information (see *Control Statements* in Chapter 2). Fields may be chained together to allow more than one field to be entered with a single keyboard input instruction.

Reformatting and Editing

Data can be reformatted for display on the operator's display screen, for writing onto a diskette, or for printed output on a printer. This reformatting is accomplished by data format statements within the ACL program. Up to 254 data formats can be specified via the appropriate control statement and an associated format field record. The following options are also available for editing during printing or other input/output operations, including diskette and display operations:

- Comma and decimal point insertion
- Blank insertion
- Floating or fixed currency sign
- Asterisk protect or asterisk fill
- Zero suppression or zero fill
- Minus sign control

Chapter 2. Reference Material

Application programs for the work station can be defined in the same way that programs for the host computer system are defined. A detailed flowchart of the program, print report layouts, and diskette record layouts can provide the necessary documentation for the coding of your program(s). Programs for the work station are coded in ACL using both control statements and instructions. Note that all entries coded in your ACL program must be left-justified on the coding sheets.

CONTROL STATEMENTS

Control statements must begin with a period in the first column of the coding sheet. The name of the control function immediately follows the period. The control statements are listed below in the order they should be entered. Each control statement is described in this chapter.

Statement	Defines
.NAME	Program name, origin, and edit control
.DATASET	Data set name, data set characteristics, and data set access methods
.PRINTER	Type of attached printer and buffer containing data to be printed, and characteristics of the form
.SELF-CHECK	Modulus/algorithm used for self-checking
.REGISTER	Register name and initial content
.BUFFER	Buffer number and initial content
.FIELD	Prompting message length, content, and disposition of data after it is entered
.FORMAT	Format number and definition for reformatting of data for input/output operations
.END	End of source statements and instructions

The first five control statements should be entered on *ACL Control Statements Coding Sheet One,* GX21-9200, in the order listed. The .BUFFER, .FIELD, and .FORMAT control statements can be entered anywhere in the source program, but must precede instructions. Comments can be coded in the .FIELD, .FORMAT, .BUFFER, and .REGISTER control statements *after* the last defined field. *ACL Control Statements Sheet Two,* GX21-9201, is designed to accommodate these statements. Each control statement is described in detail below. Note that (R) identifies a required entry and (O) identifies an optional entry.

.NAME

The .NAME control statement must be the first statement in your ACL program. Figure 5 shows the coding placement for a .NAME statement.

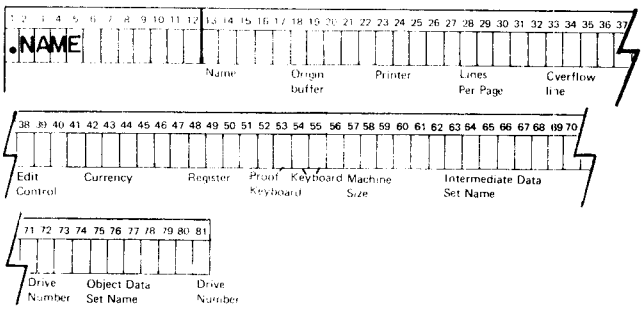


Figure 5. .NAME Coding Placement

Columns 1-5 Control Statement Name (R)

You must enter .NAME in columns 1-5.

Columns 13-16 Program Name (O)

The four-character program name is translated into the object program and can later be referenced by the operator. Blank program name is allowed.

Columns 18-19 Program Origin Buffer (R)

This entry identifies the starting *odd-numbered* buffer in which the ACL instructions are to be stored. This buffer number should be greater than the buffer numbers used for .FIELD and .BUFFER statements.

Column 23 Printer Type (R)

This entry identifies the printer attached to the work station to print the program listing *during translation.* Valid entries are:

1 = 3713 Printer
2 = 3717 Printer
3 = 3715 Printer (single direction)
4 = 3715 Printer (bidirectional with floating right margin)
5 = 3715 Printer (bidirectional with fixed right margin)

An incorrect entry may cause a translation error.

Columns 28-30 Print Form Size (O)

This entry indicates the number of lines (7-127) to be printed on a page of printed output during translation. Default is 66.

Columns 33-35 Printer Overflow Line Number (O)

This entry indicates the line number (7-126) at which skipping to the next page is to begin when the source listing is printed during translation. This parameter must be less than or equal to the print form size specified in columns 28-30. Default is 60. Note that there is a 6-line margin at the top of every printed page.

Columns 38-40 Decimal and Comma Edit Control (O)

Column 38 Edit Control Symbol (O)

This entry identifies an edit control symbol to produce a specific type of punctuated printer output. Default is a period (.) if you do not specify a substitute symbol.

Column 39 Edit Control Symbol (O)

This entry identifies an edit control symbol that produces a specific type of punctuated printer output. Default is a comma (,) if you do not specify a substitute symbol.

Column 40 Symbol Interval Count (O)

This entry specifies an interval (1-9) at which the symbol specified in column 39 is to be inserted. For example, an entry of 5 would cause the symbol specified in column 39 to be inserted in every fifth position of the printed data. Default is 3.

Columns 43-44 Edit Currency Characters (O)

This entry specifies the currency sign to be used when processing monetary data. Default is ø$. This parameter facilitates easy processing of different currencies. For example, the entry FF identifies amounts being processed as French francs. Any valid characters can be entered to represent other currencies and override the default.

Column 48 Prompting Register (O)

This entry specifies the register (A-Z) that receives data entered via the keyboard in response to a prompting message (see *Column 28 Data Disposition (R)* under *.FIELD* in this chapter). Default is register K.

Column 53 Proof Keyboard (O)

You can enter 1 in column 53 to indicate that your work station has the proof keyboard feature. Default is the standard key-entry configuration.

Columns 54-55 Keyboard Designation (O)

This entry identifies the keyboard on your work station. Valid entries are:

00 = United States, United Kingdom, France (Querty), and Italy
01 = Norway
02 = Sweden/Finland
03 = Denmark
04 = Germany
05 = Spain
06 = Belgium and France (Azerty)
07 = Portugal
08 = Japan — Katakana
09 = Brazil

Default is 00.

Column 58 Machine Size (O)

This entry identifies the size of your work station storage. You can enter 4 (4K) or 8 (8K). Default is 4K.

Columns 63-70 Intermediate Data Set Name (O)

This entry specifies the label processor output data set name. Default is TRANSLET. See *Label Processor Input Data Set* in Chapter 4.

Column 71 Drive Number (O)

This entry identifies the drive that contains the label processor output data set. Default is drive 2.

Columns 73-80 Object Data Set Name (O)

This entry identifies the object data set name of the translator output. If this name is not entered, translation cannot occur directly from the label processor.

Column 81 Drive Number (O)

This entry identifies the drive that contains the label processor object data set. Default is drive 1.

.DATASET

The .DATASET control statement should follow the .NAME statement in the ACL program. Because the work station can access up to four data sets online during execution, each data set to be accessed must have a corresponding. .DATASET control statement. In the .DATASET statement, the attributes of each data set are described for the execution program. These data set attributes include:

- Data set name
- Characteristics of the data set
- Data set access method
- Exits for special input/output routines

Figure 6 shows the coding placement for a .DATASET control statement.

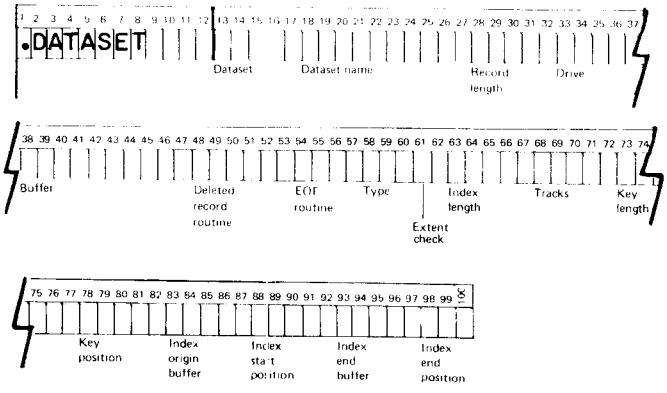


Figure 6. .DATASET Coding Placement

The required and optional parameters of the .DATASET control statement are listed in the following paragraphs.

Columns 1-8 Control Statement Name (R)

You must enter .DATASET in columns 1-8.

Column 13 Data Set Number (R)

This entry identifies the number of the data set (1-4) that the program uses. Note that data sets are opened in descending sequence (4-1) by the system. Thus, if four data sets are defined and there is an error condition in number 4, data sets 1-3 may not have been opened correctly.

Columns 18-25 Data Set Name (R)

This entry indicates the data set name as it appears on the HDR1 label (found on the index track of the diskette). (See *Index or Label Access Method,* under *Data Set Access Methods* in Chapter 3.) Note that an asterisk in column 18 inhibits system open, but allows the data set name to be specified during a programmed open (see *OPEN Data Set (OPEN)* and *Close Instruction (CLOZ)* under *Diskette Operations* in this chapter). A data set with the name specified in columns 18-25 can also be dynamically closed and reopened with another data set name during execution.

Columns 28-30 Record Length (O)

This entry identifies the logical record length (1-128 bytes) of the data set specified in columns 18-25. Default is the length from the data set label. If your data set access method is label update (see *Columns 58-60 Type (R)* under *.DATASET* in this chapter) default is 80 the first time the file is opened. Afterwards, when the file is opened, it opens to the size it was when last opened (if the last open specified a record length).

Note: Standard 3741 data set labels are 80 bytes long. Therefore, if you are creating or maintaining standard 3741 labels in the label update access method, you should specify 80 in columns 28-30.

Column 33 Drive Number (R)

This entry identifies the physical disk drive address (1 or 2) on which this data set is to reside. Drive 1 is on the right, and drive 2 is on the left.

Columns 38-39 Data Set Input/Output Buffer (R)

This entry identifies the buffer that is to be used for your disk file input/output area. When the system opens this data set, this buffer is used for reading labels, but data remains in the buffer. Thus, the buffer is not blank if no data set operations have occurred.

Columns 48-51 Deleted Record Routine (O)

This entry specifies the step number or label in a subroutine that is to be executed if a deleted record is read in this data set. An entry of 0 is invalid, and no entry causes deleted records to be bypassed.

A record is considered deleted if all of the following conditions are met:

1. A CRC (cyclic redundancy check) is successfully made at the end of the record.
2. A special address indicator is detected.
3. The first position of the record contains a D.
4. No other error indicators are on.

Note: A record with a special address indicator, but without a D in position 1, flags a read error. Under *data* station functions, the 3741 Models 1 and 2 do not provide a CRC for deleted records.

Columns 53-56 End-of-File Routine (O)

This entry specifies the step number or label in a subroutine that is to be executed on end of file. End of file occurs whenever an attempt is made to read at or beyond the EOD (end-of-data) disk address, or whenever an attempt is made to write above the EOE (end-of-extent) address. If you do not make an entry, all data sets will be closed, and the job will abort on end of file. An entry of 0 is invalid.

Deleted Record/End-of-File Routines

Because deleted records or end-of-file conditions can result from any of several disk operations in a program, a single subroutine can be programmed for all such conditions for the same file (all deleted records or end-of-file conditions). The step numbers or labels specified in the .DATASET control statement cause the following to occur during program execution:

1. When the specified disk condition is detected, (deleted record or end of file), program execution leaves the normal step number stream. This may or may not be at a disk operation step.
2. The next sequential step number or label in the normal instruction stream (that would have been executed) is placed at the step number or label specified in the .DATASET control statement. A dummy unconditional GOTO branch instruction should be coded at this location to accept the return step number or label (see *Branching Operations* in this chapter).
3. Program execution transfers to the step/label following the step number or label specified in the .DATASET statement.
4. Program execution continues in this subroutine stream as directed by the subroutine. Typically, the instruction number/label specified in the .DATASET control statement (which should be coded as a dummy GOTO) is branched to, so that program execution can return to the point in the program where execution was interrupted.

   Note: A dummy GOTO should be coded with the label assigned to that GOTO instruction to prevent label translator errors. Figure 7 is an example of end-of-file condition and explains the dummy GOTO coding.

In Figure 7 assume the LABL instruction is specified in positions 53-56 of the .DATASET control statement for data set number 1, and the following instructions are being executed when an end-of-file condition is detected during an attempted READ command:

```
STEP/
LABEL
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
RD1       READ  1
RD2       MOVE  2                 4
```

Instruction RD2 is not executed. Assume also that the following instructions are also in the program:

```
STEP/
LABEL
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
LABL      GOTO                                                    LABL
DAT1      ENTR  9             8
DAT2      GOTO                                                    LABL
```

Return to program stream

Post end-of-file message

Dummy GOTO for subroutine. For file exits, a dummy GOTO must be coded at the *beginning* of a subroutine.

DAT1 is executed next and execution returns to the LABL instruction where the return label (RD2) is placed. Execution then returns to RD2 via the GOTO at LABL.

CAUTION
I/O operations should not be executed during these routines.

Figure 7. End-of-File Condition During an Attempted READ Command

Columns 58-60 Type (R)

This entry specifies the data set access method and operation. Data set access methods are: (1) sequential, (2) key indexed, and (3) label update.

Sequential

Records are processed sequentially based on disk address order. Given one record, the location of the next is determined by the next sequential disk address in the data set. Sequential data sets are defined to be accessed as:

Entry in Col 58-60	Explanation
SR	Sequential read – records can be read sequentially, or directly by relative record number, or with the minus (-) specification (access the previous sequential record). Write commands are invalid for this data set type. A read command to this data set type will cause a physical read of the record, followed by any necessary formatting of the data from a buffer into registers.
SU	Sequential update – records can be read sequentially, or directly by relative record number, or with the minus (-) specification (access the previous sequential record). This access method allows update (write) and adding of records to the end of a data set. A write command (WRT, WRTS) to this data set type writes to the disk address of the record previously read. To update a record, the record must first be read, then written. To add records to the data set, a WRTE command (see *Instructions* in this chapter) must be issued. This increments the EOD disk address, after which the record is written at the EOD-1 disk address. If the data set is write protected, a 5XA error is posted (X = data set number).
SW	Sequential write – data sets can be created, but *not read.* The EOD is reset to BOE (beginning of extent) before any write operations begin, unless the data set has been write protected (P in position 43 of the data set label) which posts a 5XA error (Chapter 4). The data set is accessed starting at the first sequential record (BOE).
SWE	Sequential write extend – records can be written (added to an existing data set), but not read. If the data set is write protected, a 5XA error is posted (X = data set number). This data set is accessed starting at the next available record position (EOD).

Key Indexed

The key indexed access method requires that the records in the data set be arranged in ascending sequence according to a data key (search argument or control field). The data key can be up to 16 contiguous characters located in the same positions of each record of a data set. When the data set is opened, the work station control program can automatically build an index table in storage for the data set to be accessed when key indexed organization is used. All or part of the data key can be the index value. If the index is part of the key, the most significant characters (leftmost) of the key are the index value.

The index table contains an entry for the BOE address and another entry for the EOD-1 address. Between these addresses, entries are made using the index found at specified track offsets from the BOE address. Thus, the size of the index table can be controlled by specifying both the length and the number of entries in the table. The table can be built automatically at open time, or read from a separate user-maintained data set. When read from the data set, the location of the index area must be specified in the .DATASET control statement, because the work station does not check to see if the user supplied the data set or specified a dynamic open of KRN or KUN files.

Key indexed data sets are defined to be accessed as:

Entry in Col 58-60	Explanation
KR	Key indexed read only – records can be read from a data set, but not updated.
KRN	Key indexed read only – no index build – records can be read, but not updated. However, the automatic build of the index table is suppressed when the data set is opened.
KU	Key indexed update – records can be read and updated. The data set is accessed via the index table when a read command is issued. A write (WRT, WRTS) command to this data set type updates the record just read. Note that the key portion of the record must maintain its correct position in the sequence.
KUN	Key indexed update – no index build – records can be read and updated, as in the KU organization, but the automatic build of the index table is suppressed when the data set is opened.

The key indexed index is used as follows:

1. Index keys need *not* be unique; however, performance is best if they are.
2. The hardware ascends through the index table until the search index becomes $\leq$ the table index.
3. The hardware steps back to the preceding index entry.
4. The address of the preceding index entry is calculated.
5. The hardware begins searching the data set at the calculated address. This search continues until (a) a match is found, or (b) the search key is $>$ the record key. (The hardware searches as many tracks as necessary to get the $>$ condition.)
6. If a match is found, that record is put in the I/O buffer.
7. If a match is not found, the record with the next higher record key is put in the I/O buffer and indicator 225-228 is turned on.

If the index table is automatically built when the data set is opened, the following sequence takes place:

1. The buffer(s) for the index table is specified by the program loader or in columns 83-84 of the .DATASET control statement. If the buffer has not been specified, the system supplies a default buffer, which is generally between the buffer containing the last program instruction and the data formats.

2. The index (data key) at the BOE record is read from disk and inserted into the table.

3. The next index is read from the BOE plus the number of tracks specified for each index. Indexes are read in this manner until EOD is reached or exceeded. Note that, during operation, the system issues a warning message (5X8) (X = the data set number) if the index exceeds available storage space (Chapter 4). The operator must press the RESET key to continue operation. If a deleted track is found when building the index, or if the index cannot be built in sequence, another message (5X9) (X = the data set number) is issued, and the job is terminated.

4. The EOD-1 index is read.

5. The index table is normally terminated with a hexadecimal FF delimiter. However, if the space allocated by the system or in the .DATASET statement cannot hold all the index entries, (there must be room for at least 2 index entries) the EOD-1 entry is extracted and a resettable warning error is posted. Note that once a data set has been opened with a key indexed data set type, and the index table has been built, the starting and ending address of the table are recorded and used for any subsequent opening of this file.

6. The starting and ending table addresses are recorded in the work station control storage area.

The following steps are taken during execution for data set access of key indexed type:

1. A READ instruction specifies the general register containing the key of the record. (The key is right-justified in the register.)

2. The significant (leftmost) characters of the key are compared to the indexes in the table. The search stops when the key is equal to or less than the table entry. If the table is exceeded, or the key is less than the first index entry, two indicators (225-232) are set on (Appendix A).

3. The BOE, plus the number of tracks specified by the table index, is sought.

4. A fast scan is started on the track specified for the desired record. The entire data key is checked for a match. The scan continues on the next track until a match or a higher record is found. On a match, no indicators are set, and necessary formatting takes place. If no record is found, an indicator (225-228) is set on, and no formatting takes place, but the next higher record is contained in the data set buffer.

Entry in Col 58-60	Explanation
I	Label update — records can be read and updated. The diskette index track is accessed starting at 00001. Operation is the same as on a sequential update file (except for WRITE).

Column 61 Extent Check (O)

The extent boundaries (BOE and EOE) of the file are checked against all other valid labels on the disk at open time, if column 61 is blank. No overlap extent checking is done when the labels are modified using the data station functions, as opposed to the work station functions. Bypass this check by entering an A in column 61.

The remainder of parameters in the .DATASET control statement apply only to the key indexed access method, but can be coded for other access methods and used when a key indexed data set is opened in subsequent OPEN instructions.

Note: Null data sets received in BSCA mode appear to have overlapped extents, thus causing the extent check to fail (5X7 error is displayed).

Columns 63-64 Index Length (O)

This entry specifies the number of characters (1-16) in an index entry. The index is taken from the leftmost significant characters of the key. Default is 4.

Columns 68-69 Tracks/Index (O)

This entry specifies the number of tracks (1-74) associated with each index entry. Default is 1.

Columns 73-74 Key Length (O)

This entry specifies the number of characters (1-16) in the key field within the record. This number must be greater than or equal to the index entry length (columns 63-64). Default is 4.

Columns 78-80 Key Position (O)

This entry identifies the location of the index key within the record. The entry (1-128) specifies the record position that contains the left end position of the data key. Default is 1.

Columns 83-100

The entries in columns 83-100 are required for data set types KRN and KUN.

Columns 83-84 Index Origin Buffer (O)

This entry specifies the number (1-24) of the buffer in which the index table is to start. This buffer is used with following .DATASET parameters at open time for the index table. If you do not make an entry, the system supplies a default buffer.

Columns 88-90 Index Start Position (O)

This entry specifies the starting position (1-128) of the index table within the buffer just specified (columns 83-84). This entry allows the index table to be built at any position within the buffer, and is required if the index origin buffer (columns 83-84) is specified.

Columns 93-94 Index End Buffer (O)

If you entered an index table origin buffer in columns 83-84, you must enter the number of the buffer that is to contain the end of the index table in columns 93-94. This entry must be greater than or equal to the origin buffer number.

Columns 98-100 Index End Position (O)

This entry specifies the ending position (1-128) of the index table within the buffer specified in columns 93-94). This entry must be one greater than the last position of the index table.

.PRINTER

The .PRINTER control statement should follow the .DATASET statement in the ACL program. This control statement specifies general information about the attached 3713, 3715, or 3717 Printer. Printer output is detailed according to your instructions and edit formats. Figure 8 shows the coding placement for a .PRINTER control statement (see *Printer Operations* under *Programming Restrictions* in Chapter 3).

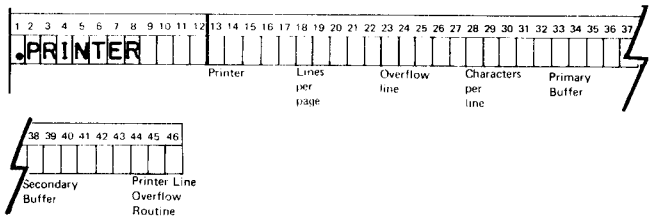


Figure 8. .PRINTER Coding Placement

Columns 1-8 Control Statement Name (R)

.PRINTER must be entered in columns 1-8.

Column 13 Printer Type (R)

Because five different types of printer can be attached to the work station, the specific printer attachment must be identified. Specify one of the following (zero is invalid):

Entry	Explanation
1	3713 Printer
2	3717 Printer
3	3715 Printer
4	3715 Printer
5	3715 Printer

The 3713 Printer is a single direction character printer with a printing speed of 40 characters per second.

The 3717 Printer is a line printer with a printing speed of 100-120 lines per minute.

The 3715 Printer is available in two models (40 or 80 characters per second). Each model can be programmed to print in three different ways:

- Specify a 3 for single-direction printing with a floating right margin. The print head must travel to the right as far as required by each print line. Only one printer output buffer is required.
- Specify a 4 for bidirectional printing with a floating right margin. Printing occurs two lines at a time, thus requiring two printer output buffers. Printing occurs after every other print command, that is, after the primary and secondary buffers are loaded. The machine scans the primary buffer to determine how far to the right to print, and scans the secondary buffer to determine if it must continue to the right in order to print the next line in the opposite direction. Data to be printed is always specified in the primary buffer while the secondary buffer is used only by the machine to control printing of the second line.
- Specify a 5 for bidirectional printing with a fixed right margin. The line length is specified in the program and the printer always travels to the fixed right margin. Because of this, only one printer output buffer is required.

Columns 18-20 Lines Per Page (O)

This entry specifies the maximum number of lines to be printed per page, up to six lines per inch. Default is 66.

Columns 23-25 Overflow Line (O)

This entry specifies the line number beyond which no printing should occur. The overflow condition occurs after printing the specified line. This entry should be less than or equal to the entry in columns 18-20. Default is 60.

Columns 28-30 Characters Per Line (R)

This entry specifies the number of characters (1-132) printed on a horizontal line. If more than 128 characters are specified, the buffer (columns 33-34) must be odd-numbered; the excess over 128 is contained in the next sequential buffer.

Columns 33-34 Primary Buffer (R)

This entry specifies the primary output buffer for use by the printer.

Columns 38-39 Secondary Buffer (O)

Because the 3715 Printer prints bidirectionally, a buffer must be specified for each direction. This entry is required only if 4 has been entered in column 13.

Columns 43-46 Printer Overflow Routine (O)

This entry specifies the step number or label of the first instruction in an overflow subroutine, when the printer reaches the overflow line number specified in columns 23-25.

Note: A dummy GOTO instruction should be coded at the step number/label specified.

The step number/label of the subroutine may be entered. An entry of step number 0 is invalid. If you do not make an entry, overflow detection requires testing of indicator 148.

.SELF-CHECK

The .SELF-CHECK control statement should follow the .PRINTER statement in the ACL program. The .SELF-CHECK statement defines the internal algorithm to be used during execution. All arithmetic is done to the base of the modulus. Figure 9 shows the coding placement for a .SELF-CHECK control statement.

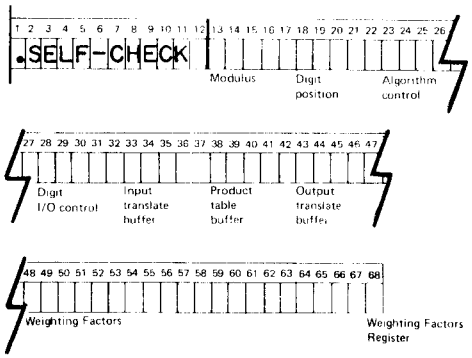


Figure 9. .SELF-CHECK Coding Placement

Columns 1-11 Control Statement Name (R)

.SELF-CHECK must be entered in columns 1-11.

Columns 13-15 Modulus (R)

This entry specifies the modulus (2-127) for all mathematical operations. If self-checking is with standard modulus 10 or 11, enter S10 or S11, respectively, and ignore the remaining parameters.

Columns 18-19 Digit Position (O)

This entry specifies the displacement (0-16) of the rightmost self-check digit within the register. Displacement is the position in the register of the rightmost self-check digit, as measured by the number of positions (0-16) from the left of the register. Blank defaults to zero. If the displacement is zero, the result of self-check computation must also be zero in order to pass the IF/CHK test (see *Branching Operations* in this chapter). If the displacement is zero, the result of the GSCK computation is not stored (see *Instructions* in this chapter). If the displacement is 1 and two self-check digits are specified, then the leftmost digit of the self-check computation must be zero for the IF/CHK to be true (the leftmost result of the GSCK computation is not stored).

Note: Specification of S10 or S11 requires that the low-order position of the register specified in the GSCK instruction be blank.

Columns 23-25 Algorithm Control

Column 23 Summing of Products (O)

Valid entries for column 23 are:

Entry	Explanation
Blank	Multiply weights times digits and sum whole numbers.
D	Multiply weights times digits and sum all the digits of the products.
U	Multiply weights times digits and sum the units digits of the products.
E	Translate digits to products and sum all the digits. Product table repeats every third digit (Figure 10).
F	Translate digits to products and sum all the digits. Product table repeats every fourth digit (Figure 10).

Product Table (Figure 10): The low-order four bits of each digit in the self-check register and the position of the digit in the self-check field are used to determine the buffer location of the product to be used. If two self-check digits are to be generated, the second product is displaced 64 positions from the first.

Buffer Position (00 thru 0F in Hex)	Digit Position in Register	
	With an E in Col 23	With an F in Col 23
1 thru 16	16, 13, 10, 7, 4, 1	16, 12, 8, 4
17 thru 32	15, 12, 9, 6, 3	15, 11, 7, 3
33 thru 48	14, 11, 8, 5, 2	14, 10, 6, 2
49 thru 64	Not used	13, 9, 5, 1
	Second Product if Used	
65 thru 80	16, 13, 10, 7, 4, 1	16, 12, 8, 4
81 thru 96	15, 12, 9, 6, 3	15, 11, 7, 3
97 thru 112	14, 11, 8, 5, 2	14, 10, 6, 2
113 thru 128	Not used	13, 9, 5, 1
Note: If second product is not used, positions 65-128 should be hex 00.		

Figure 10. Product Table

The result is that two numbers are referenced as NL (leftmost self-check number) and NR (rightmost self-check number). The NL number is forced to 0 if U is specified or column 23 is blank. See *Self-Check Examples* in this chapter.

Column 24 Sum Manipulation (O)

Valid entries for column 24 are:

Entry	Explanation
Blank	Divide the sum of NL and NR by the modulus.
D	Divide the sum of the digits of the sum of NL and NR by the modulus.
K	Special cross add of the digits of the sum of NL and NR. (The hundreds digit plus units digit equals the NR. The tens digit plus the carry from NR equals the NL).
E	Special modulus 8 and 3. The units position of the self-check number is stored modulus 8 and the tens position is stored modulus 3. Column 23 cannot be blank.

This parameter is used to manipulate the NL and NR. If D is specified or if column 24 is blank, NL is forced to 0. See *Self-Check Examples* in this chapter.

If E is specified, C cannot be specified in column 25 for a modulus less than 8.

Column 25 Complement (O)

Valid entries for column 25 are:

Entry	Explanation
C	NL and NR complemented to modulus
Blank	NL and NR unchanged

Columns 28-30 Digit I/O Control

Column 28 Number of Self-Check Digits (O)

Valid entries for column 28 are:

Entry	Explanation
Blank or 1	One digit generated or checked With an output translate table, and a K in column 24, the NL and NR are summed before translation.
2	Two digits generated or checked If E is entered in column 24, the NL is multiplied by 8 and added to the NR. That sum is then translated.

Column 29 Decimalize Self-Check Number (O)

Valid entries for column 29 are:

Entry	Explanation
D	The NR is used to produce a two-digit decimal number. The units digit is converted to the DR (displayable rightmost self-check digit), and the tens digit is converted to DL (displayable leftmost self-check digit).
Blank	An F-zone is ORed to NL and NR to produce DL and DR.

If the result of this operation exceeds 99, the units digit output is correct, and the second digit has an F-zone and a digit portion between A and C.

Column 30 U.K. Special Algorithm 1 (O)

Valid entries for column 30 are:

Entry	Explanation
F	Each byte in the input translate table (Figure 11) is interpreted as two hex digits. The low-order hex digit (four bits) becomes the input translate character. The high-order hex digit (four bits) becomes the shift left count. The position being translated, and all higher positions in the register, are shifted left (with zero fill) the number of positions in the shift count, when the shifted register contains 16 bytes. (All unused high-order bytes of the original register are bypassed.)
Blank	All eight bits of any input translate byte are used for the input translate number.

Columns 33-34 Input Translate Table Buffer Number (O)

Note that columns 33-34 below must contain 1-24 if you specify the U.K. special algorithm 1.

Valid entries for columns 33-34 are:

Entry	Explanation
Blank	Zone is ignored Use the low-order four bits of each source between 0-9. Use 0 for all numeric portions of A-F.
1-24	Buffer number (Figure 11)

Input Translate Table (Figure 11)

All data must be entered in hex. All positions left blank will translate the corresponding character to a decimal 64 or hex 40. Before any self-check operation is performed, all 64 graphic characters can be translated to some other hex character by specifying an input translation (columns 33-34 of the .SELF-CHECK control statement). Translation occurs on the low-order seven bits of each character in the self-check register.

When a character in the self-check register matches the character in the rightmost column, the hex character at the buffer position listed in the leftmost column is used for the self-check computation.

If no input translate table is specified, the low-order four bits of each byte are used for numerals 0-9 and all other EBCDIC characters with low-order values of 0-9. Zero is used for digits A-F.

Buffer Position	Character to be Replaced
65	Blank
66-74	A thru I
75	¢
76	.
77	<
78	(
79	+
80	|
81	&
82-90	J thru R
91	!
92	$
93	*
94	)
95	;
96	¬
97	– (dash)
98	/
99-106	S thru Z
107	\
108	,
109	%
110	_ (underscore)
111	>
112	?
113-122	0 thru 9
123	:
124	#
125	@
126	'
127	=
128	"

Figure 11. Input Translate Table

Columns 38-39 Product Table Buffer Number (O)

Valid entries for columns 38-39 are:

Entry	Explanation
Blank	No product table used
1-24	Buffer number (Figure 10) Column 23 must contain E or F.

Columns 43-44 Output Translate Table Buffer Number (O)

Valid entries for columns 43-44 are:

Entry	Explanation
Blank	No output table used
1-24	Buffer number (Figure 12) .SELF-CHECK column 29 must be blank.

Output Translate Table

This table provides a capability similar to input translation. The value of the self-check digit is used to determine the position of the buffer containing the character to be inserted in the self-check register.

If two digits and an output translate table are coded in the statement, the same table is used to translate both digits.

Buffer Position	Self-Check Digit
1	0
2	1
3	2
⋮	⋮
35	34
36	35
⋮	⋮
128	127

Note: Normally (modulus 63 or less), the input and output translate tables can be in the same buffer at the same time.

Figure 12. Output Translate Table

If an output translate table is used and one digit is to be generated, the two values are added and the sum is translated when column 24 is K. If column 24 is E, the NL is multiplied by 8, added to the NR, and the sum is translated.

Columns 48-63 Weighting Factors (O)

All 16 bytes represented in columns 48-63 must be entered in hex (Figure 13) and must be less than the modulus (Figure 14). A weight of hex 00 must be entered in the positions of the self-check digits and any other positions to be bypassed. A weight of hex 01 should be entered in all positions, except those to be bypassed, when the product table is used (Figure 10).

Graphic Character	EBCDIC 8-Bit Code Bit Positions 0123	4567	Hex Equivalent
Blank	0100	0000	40
¢	0100	1010	4A
.	0100	1011	4B
<	0100	1100	4C
(	0100	1101	4D
+	0100	1110	4E
|	0100	1111	4F
&	0101	0000	50
!	0101	1010	5A
$	0101	1011	5B
*	0101	1100	5C
)	0101	1101	5D
;	0101	1110	5E
¬	0101	1111	5F
-	0110	0000	60
/	0110	0001	61
,	0110	1011	6B
%	0110	1100	6C
_	0110	1101	6D
>	0110	1110	6E
?	0110	1111	6F
:	0111	1010	7A
#	0111	1011	7B
@	0111	1100	7C
'	0111	1101	7D
=	0111	1110	7E
"	0111	1111	7F
A	1100	0001	C1
B	1100	0010	C2
C	1100	0011	C3
D	1100	0100	C4
E	1100	0101	C5
F	1100	0110	C6
G	1100	0111	C7
H	1100	1000	C8
I	1100	1001	C9
J	1101	0001	D1
K	1101	0010	D2
L	1101	0011	D3
M	1101	0100	D4
N	1101	0101	D5
O	1101	0110	D6
P	1101	0111	D7
Q	1101	1000	D8
R	1101	1001	D9
\	1110	0000	E0
S	1110	0010	E2
T	1110	0011	E3
U	1110	0100	E4
V	1110	0101	E5
W	1110	0110	E6
X	1110	0111	E7
Y	1110	1000	E8
Z	1110	1001	E9
0	1111	0000	F0
1	1111	0001	F1
2	1111	0010	F2
3	1111	0011	F3
4	1111	0100	F4
5	1111	0101	F5
6	1111	0110	F6
7	1111	0111	F7
8	1111	1000	F8
9	1111	1001	F9

Note: Lowercase letters are not used by the 3741 Models 3 and 4.

Figure 13. EBCDIC Code for Graphic Characters

The following example shows how to compute weights for a divide by 7.

```
7 | 1000000
    0
    --
    10
     7
    --
    30
    28
    --
     20
     14
    --
     60
     56
    --
      40
      35
    --
       50
       49
    --
Repeats 1
```

The remainder for each step is the weight for that position.

Position in .SELF-CHECK Statement	Modulus 7
48	02
49	03
50	01
51	05
52	04
53	06
54	02
55	03
56	01
57	05
58	04
59	06
60	02
61	03
62	01
63	00

Note: Weights for the self-check digit positions must be hex 00.

Figure 14. Weight Table Examples

Column 68 Weighting Factors Register (O)

This entry specifies a register (A-Z) that contains the weighting factors for the self-checking algorithm. If a register is specified, columns 48-63 are ignored.

Self-Check Examples

Figure 15 shows the coding necessary for self-checking with the standard modulus 10 or modulus 11, which are also available on the 3741 Models 1 and 2. Note that, other than the entries in columns 1-15, none of the various parameters must be entered in the statement.

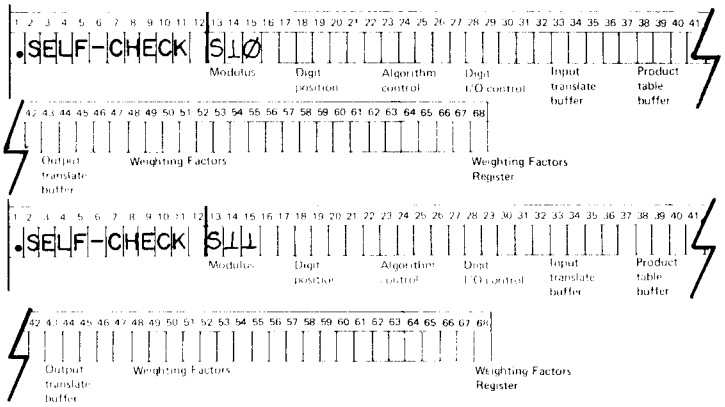


Figure 15. .SELF-CHECK Coding for Standard Modulus 10 and Modulus 11

Modulus 7

Figure 16 shows the coding for modulus 7.

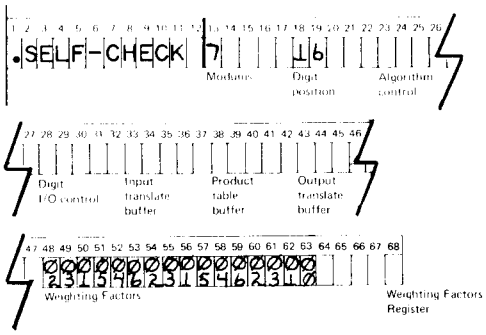


Figure 16. .SELF-CHECK Coding for Modulus 7

The result of the coding shown in Figure 16 is:

1. Multiply each digit of the check register times the weight for that position. (Weights are 2, 3, 1, 5, 4, 6, 2, 3, 1, 5, 4, 6, 2, 3, 1, 0).
2. Add the products.
3. Divide this sum by 7.

The remainder is the self-check digit.

.REGISTER

The .REGISTER control statement should follow the .SELF-CHECK statement in the ACL program. The .REGISTER statement initializes any of the 16-byte registers (A-Z) to a starting value for ACL program execution. All registers not specified in a .REGISTER control statement are initialized to blank. The capability to identify the initial contents of a register allows you to preinitialize registers for subsequent execution. Figure 17 shows the coding placement for a .REGISTER control statement.

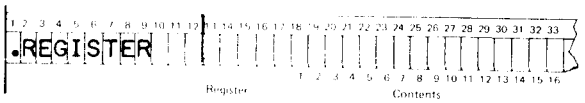


Figure 17. .REGISTER Coding Placement

Columns 1-9 Control Statement Name (R)

.REGISTER must be entered in columns 1-9.

Column 13 Register Name (R)

This entry specifies a register (A-Z) to be used for a particular application during program execution.

Columns 18-33 Register Contents (R)

This entry comprises the initial contents of the register specified in column 13 and may contain any message or constant desired. For example, you can fill all 16 positions of the register with a message, or fill only one position with a constant. The register is initialized exactly as specified (data is not right- or left-adjusted). Position 16 is the low-order (rightmost) position.

.BUFFER

The .BUFFER control statement is used to initialize any of the buffers (1-24 or 1-56 for 8K) in the work station (for example to create tables). .BUFFER statements can be entered in any order, by buffer number. After the two parameters for the .BUFFER statement are entered, the initial load for the buffer should be designated on the line immediately following the .BUFFER statement. The coding placement of a .BUFFER control statement is shown in Figure 18.

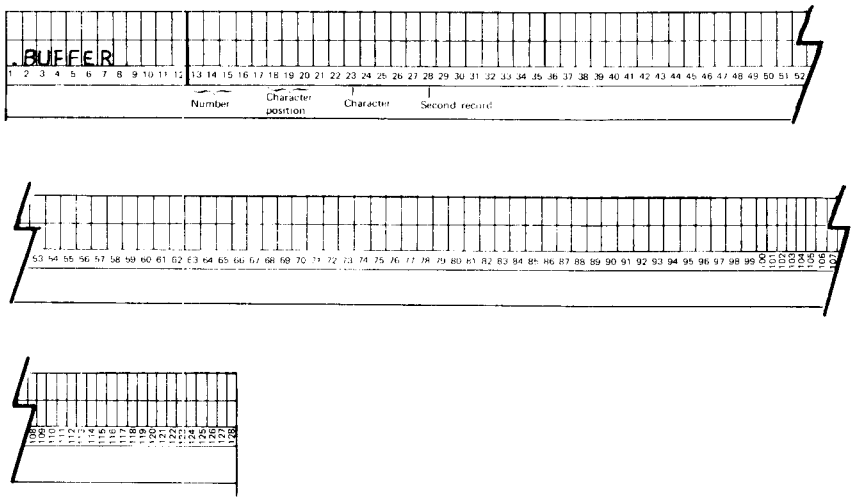


Figure 18. .BUFFER Coding Placement

Columns 1-7 Control Statement Name (R)

You must enter .BUFFER in columns 1-7.

Columns 13-15 Buffer Number (R)

This entry is used to assign the number of the buffers you wish to initialize.

Designated Buffer Load (R)

The information entered on the line following the .BUFFER statement is the content (1-128 positions) of the specified buffer. Because the system does not right-justify or left-justify this information, it must be positioned exactly as you want it.

.FORMAT

Several instructions contain an operand in which a format is specified (254 different formats are available). Each format has associated control information relating a register address to the location and length of data in the buffer. This information is defined to the system by the .FORMAT control statement and an associated format field record(s), which defines the image of the buffer and associated registers used in moving data to and from the buffer. .FORMATS must precede the use of the specified format.

The record immediately following a .FORMAT control statement describes the format fields. This record defines the location and length of the field, and the register affected, by a contiguous string of alphabetic characters. The character indicates the register and the number of characters indicates the field length. Thus, each data element is described according to its placement in a buffer and its register disposition. For example, a format record shown below defines three fields.

```
COLUMN 1       8   11
ENTRY  CCCCbbbbAAbEEEEEEEEEEEEEEEE
```

Field 1 is four characters long, starting in position 1 of the buffer. Data is moved to or from register C. Field 2 is two characters long, starting in position 8 of the buffer. Data is moved to or from register A. Field 3 is 16 characters long, starting in position 11 of the buffer. Data is moved to or from register E.

Coding placement for a .FORMAT control statement is shown in Figure 19.

Columns 1-7 Control Statement Name (R)

.FORMAT must be entered in columns 1-7.

Columns 13-15 Number (R)

This entry specifies the number (1-254) of the format to be referenced during read and write operations.

Columns 18-20 Character Position (O)

This entry specifies the position in subsequent input records that will cause the format number (columns 13-15) to be used, if the character specified in column 23 appears in this position.

Column 23 Character (O)

Any character may be entered to initiate the use of the format defined on the line(s) following the .FORMAT control statement. Whenever this character is found in the position specified in column 18 of this statement, the format defined in the succeeding line on the coding sheet is used.

Column 28 Second Record (O)

This entry is the number of format field records that follow the .FORMAT control statement. A 2 indicates that two format field records follow; a blank indicates one following format record. (Note that use of the 132-print position printer requires two format field records.)

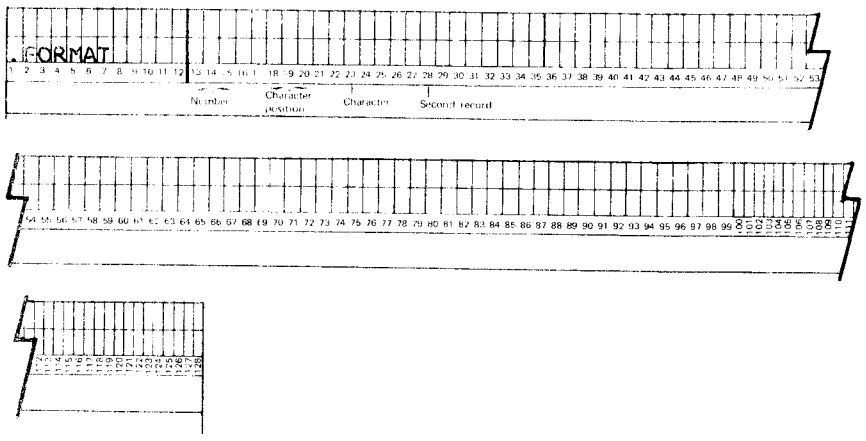


Figure 19. .FORMAT Coding Placement

Data Directed Formatting

Input instructions may be coded to provide automatic or data-directed format selection. Data-directed format selection results in the following sequence of operation:

1. Formats with entries in columns 18 and 23 are scanned in the sequential order in which they are defined in the source program.

2. The character of the input record which is located at the specified position is compared with the specified select character.

3. If a match is found, the format corresponding to the detected select character is used to format the input record into registers.

4. If a match is not found, the format indicator is zero, and formatting is suppressed.

An example of this operation is:

Column	1	13	18	23	28
Entry	.FORMAT	2	24	$	

This .FORMAT control statement specifies that if the character $ is detected in position 24 of the input record during scanning, format 2 should be used to format the input record into registers. Work station control programming internally stores the last format to be used to allow that format to be tested by an IF format instruction (see *Instructions* in this chapter).

Formatting Blocked Records

When short records (64 characters or less) are used, they may be compacted, or blocked, within records contained on disk. Thus, disk space can be used more efficiently. Physical records on disk cannot be more than 128 characters in length. Thus, each disk record can contain two or more logical records when using blocked records. When creating formats for records blocked in this manner, only the format for the first logical record is necessary. Formatting for the second and succeeding actual records in the buffer is based upon the format defined for the first logical record. Upon issuing a blocked reformat instruction (RBLK, WBLK), formatting begins in the buffer at the position specified by the register contained in the instruction, using the format defined for the first record. See *Blocking and Deblocking of Logical Records* in Chapter 3.

Formatting Records Greater Than 128 Characters

If it is necessary to format an output record that is greater than 128 characters, two format records can be specified with one .FORMAT control statement. By entering a 2 in column 28 of the .FORMAT statement, the next two coding lines can be used for specifying up to 256 bytes of formatting.

Note that the maximum record length to disk is 128 characters and to printer is 132 characters.

Editing

Format fields used in an output (register to buffer) instruction can produce punctuated output data (displayed, printed, or recorded on diskette). Characters coded directly into the format field are used to control editing. Edited formatting is ignored on input operations. Note that any editing forces an F-zone on all bytes of the register. Edit characters are as follows. Note that these characters do not take register positions.

Edit Control Character	Function
0 (zero)	This character is used for zero fill. The character must be placed in the leftmost position of the field (replacing a register designation character) where zero fill is to begin. Leading blanks in source data are forced to zero.
/ (slash)	This character is used for blank fill (zero suppress). The character must be placed in the leftmost position of the field where blank fill is to begin. Leading zeros in the source data are forced to blanks.
* (asterisk)	This character is used for asterisk fill. The character must be placed in the leftmost position of the field where asterisk fill is to begin. Leading blanks and zeros in the source data are forced to asterisks.
$ (dollar sign fixed)	This character must be used with one of the above fill control characters. The character must be placed immediately to the left of the fill character (leftmost register designation). The dollar sign is inserted into the associated position of the output. Any two characters can be substituted for the dollar sign in columns 43-44 of the .NAME control statement (see *Control Statements* in this chapter). If two characters are substituted for the dollar sign, the position to the left of the $ in the format must be blank.

Edit Control Character	Function
$ (dollar sign floating)	The floating dollar sign can be obtained by placing the dollar sign immediately to the left of a register designated for output from the field. The dollar sign is then inserted to the left of the most significant digit in the output data. Any two characters can be substituted for this character in the .NAME control statement. If two characters are substituted for the dollar sign, the position to the left of the $ in the format must be blank.
- (minus)	This character must be placed immediately to the right of a register field. If the field is negative, as indicated by a D-zone in the units position of the register, a minus sign is edited into the position indicated by the minus edit character. The units position is then output with a hexadecimal F-zone.
& (ampersand)	This character causes a space (hex 40) to be forced into the associated position of the field. The character can be used only once in a field and is mutually exclusive with a decimal point.
. (decimal point)	A decimal point is forced into the associated position of the field. Decimal point placement is not affected by the significance of the digits to the right or left. Any character can be substituted in column 38 of the .NAME control statement.
, (comma)	A comma is forced into the output field at every third position to the left of the position indicated by the decimal point or blank (ϕ) insert. If a decimal point or blank insert is not specified, the comma is inserted in every third position of the field. The comma is repeated automatically and may appear in the field as often as desired. Spacing is controlled by the entry in the .NAME statement. Comma insertion is suppressed during the fill operation. A substitute character can be specified in column 39 of the .NAME control statement. A substitute insertion interval (for example, every fourth position) can be specified in column 40 of the .NAME control statement. The ACL programmer must allow an adequate number of positions in the output field to provide for the maximum number of commas to be inserted. Note the comma control count (.NAME, column 40) controls the number of commas to be inserted.

Fill characters will stop with a significant digit within the field, a decimal point or space caused by an ampersand within the field, or with the units digit of the field. A minus field with no fill character specified defaults to blank fill. The output of a zero register with minus editing will be a blank field.

For additional information on editing requirements, see *Insert Character in Buffer (ICBF)* in this chapter.

Examples of Editing

Source Data	Field Specification	Resultant Output
0000000000012345	$AAA.AA	$123.45
0000000000000123	$BBB.BB	$1.23
000000000000012L	$*BB.BB–	$**1.23–
0000000000000012	/CCCCC	12
0000000000123456	D,DDD.DD	1,234.56
0000000000000000	/HHH.HH	.00
0000000000000000	$HHH.HH	$.00
0000000000000001	MMMMM–	1
0000000001234567	E,EEE,EEE	1,234,567

As shown in the last example above, you should allow for *all* the edit characters in the output. The .NAME statement can override certain edit options as follows:

Column 38 (decimal position character) – blank defaults to a decimal point.

Column 39 (comma position character) – blank defaults to a comma.

Column 40 (comma control count) – blank defaults to a value of 3; range is 1 to 9.

If the .NAME statement contained a comma in column 38, a decimal point in 39, and a 2 in 40, an example of the override is:

Source Data	Field Specification	Resultant Output
1234567	FF,FFF.FF	1.23.45,67

Data Movement

Formats are processed, field by field, from left to right. Consequently, on input, if fields are specified with duplicate register addresses, the rightmost field is the end result of the contents of the register. On output, duplicate fields may be specified, and the data in the specified register is output many times.

On input, data is moved from the right to the left into the register from the buffer. Input data is right-justified in the register with high-order blank fill, and edit control is ignored. On output, data is moved from the right to the left from the register to the buffer, starting from the byte position (17-n) in the register (n=the number of bytes to be moved to the leftmost position in the buffer). The buffer field is edited while data is being moved from the register.

Note: If the first positions of a buffer do not specify a large enough output field, the data is inserted in the last position of the next lower buffer.

.FIELD

The .FIELD control statement contains the parameters that are implemented by the ENTR instruction (see *Instructions* in this chapter). The .FIELD statement defines the prompting message displayed to the operator, allows data to be entered via the keyboard, and directs the disposition of data entered. Figure 20 shows the coding placement for the .FIELD control statement.

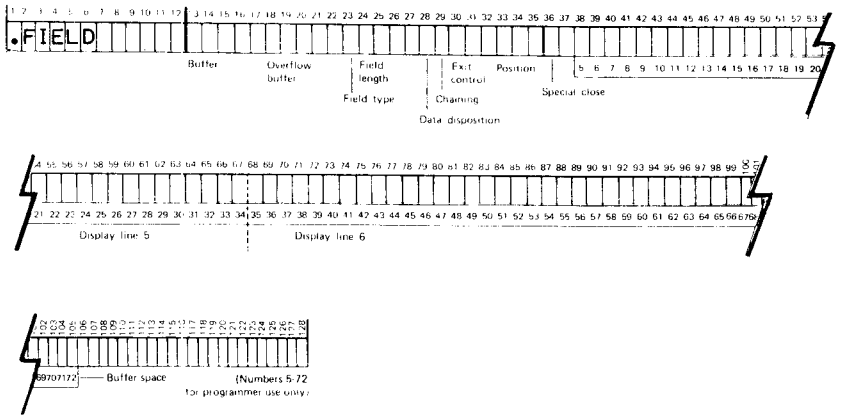


Figure 20. .FIELD Coding Placement

Columns 1-6 Control Statement Name (R)

.FIELD must be entered in columns 1-6.

Columns 13-14 Buffer (R)

This entry specifies the number of the buffer that is to hold the prompting message. This buffer holds the start of the prompting message. If two or less positions are left in this buffer, a 76 warning error message is posted by the translator, even if an overflow buffer is also specified.

Columns 18-19 Overflow Buffer (O)

This entry specifies the number of the buffer that is to hold the overflow from the prompting message, if the message is too large to be contained in the buffer specified in columns 13-14.

When chaining .FIELD messages across buffer boundaries, an overflow buffer is required. For a description of buffer usage, see *Using Operator Messages* in Chapter 3.

Column 23 Field Type (R)

This entry specifies the type of data to be entered via the keyboard in response to a prompting message. If D or U is entered, dashes are displayed on the display screen. If an A is entered, periods are displayed. Default is periods. Column 23 entries are:

Entry	Explanation
Blank or A	Indicates alphabetic or numeric data with the keyboard shift set for alphabetic (lower), and a manual shift required for numerics and special characters (upper).
U	Indicates alphabetic or numeric data with the keyboard shift set for numeric and special characters (upper), and a manual shift required for alphabetic.
D	Indicates digits 0-9 only (no override possible).

Columns 24-25 Field Length (R)

This entry specifies the number of characters to be keyed in response to a prompting message. This number must be 1-16 if data is moved to a register or 1-64 if data is moved to a buffer (according to the specification in column 28). Field length plus message length cannot exceed 68.

Column 28 Data Disposition (R)

Entry	Explanation
Blank or B	Data entered is moved to the display screen buffer (buffer 2) starting at the position specified in columns 33-35.
R	Data entered is moved to the register specified in columns 33-35.
M	Data entered and the prompting message displayed are moved to the display screen buffer (buffer 2) starting at the position specified in columns 33-35.
D	Data entered is moved to the display screen buffer (buffer 2) starting at the position specified in columns 33-35. The data entered is also moved to the register specified in column 48 of the .NAME control statement. Default is register K if you did not specify a register in the .NAME statement.

Column 29 Field Chaining (O)

Entry	Explanation
Blank	Indicates the last field, terminate chain, and that no field exit key (RIGHT ADJ, SKIP or dash) is required unless column 30 contains a J or Z.
C	Indicates a chain to the next field, and that no field exit key is required (unless column 30 contains J or Z).
J	Indicates a chain to the next field, and that a field exit key is required.
L	Indicates the last field, and that a field exit key is required.

This entry specifies the field exit option. If this column is blank, the system exits from the ENTR instruction (see *Instructions* in this chapter) when the entry field is full or a field exit key is pressed (unless column 30 contains J or Z). If C is entered, the system chains to the next field when the entry field is full or a field exit key is pressed. If J is entered, the system exits the field and chains to the next field only when a field exit key is pressed. If L is entered, the system exits the ENTR instruction when a field exit key is pressed.

Field Exit Keys

The following function keys are used for field exit. The operation resulting depends upon whether column 30 of the .FIELD control statement specifies the field to be right-adjusted or not right-adjusted.

Key	Operation
RIGHT ADJUST and SKIP	*Non-right-adjust field:* Inserts blanks into the remaining rightmost positions of the field, sets the corresponding field exit indicator, resets the other field exit indicators, and exits the field. *Right adjust field:* Aligns data keyed to the right boundary of the field and inserts blanks or zeros to the left of the first data character, sets the corresponding field exit indicator, resets all other field exit indicators, and exits the field.

Key	Operation
Dash (—)	*Non-right-adjust field or alpha right-adjust field:* Inserts dash (—) data character. *Right-adjust numeric field:* Inserts a D-zone on the last character keyed in the field, sets the corresponding field indicators, performs the right-adjust function and exits the field. The ALPHA SHIFT and dash (—) keys insert the dash character.

The CHAR BKSP and FIELD BKSP keys are active while keying into a prompted message field.

The action of these keys is:

Key	Operation
CHAR BKSP	Replaces the last data character keyed in the current field with a dash (for numeric) or period (for alphabetic), depending on the field shift. Backspacing is limited to the current field.
FIELD BKSP	Resets the current field to its status before any data characters were keyed. Backspacing is limited to the current field. Field backspacing from the first position of the field produces a keying error unless the special keyboard close option (a T in column 36 of the .FIELD control statement) has been selected.

If the special keyboard close option is selected, and one of the following keys is pressed while executing an ENTR instruction, the respective key indicator and the special keyboard master indicator (200) are turned on. The ENTR instruction is terminated and the keyboard is closed. None of the special close keys are accepted when not in an ENTR instruction, even when the special keyboard close option is selected. No data is transferred from the current field.

DUP
FIELD ADV
REC BKSP
REC ADV
SEL PROG
FIELD BKSP (The above conditions apply only when this key is pressed at the first position of the field; otherwise, the key must be pressed twice.)

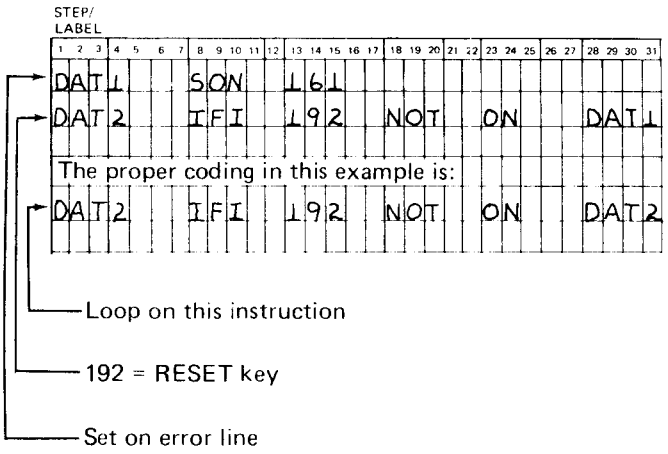
The special keyboard master indicator is turned off upon execution of the next ENTR instruction.

If the keyboard close option is not selected (no T in column 36 of the .FIELD control statement), and any of these keys is pressed, a 90 error results with no indicators set (FIELD BKSP must be pressed twice for these conditions to apply, unless at the first position of the field).

The RESET key is active at all times and performs the following functions:

- Blanks positions 1-8 of line 1 on the display screen. Positions 5, 6, 7, and 8 are used for posting system error messages. User errors may be posted in positions 1-4 of line 1, and the error line indicator (161) set on. In this case, the RESET key may have to be pressed more than once to clear the error line and the posted error.

  Note: This condition can be caused by improper program coding, such as:



- Clears error line (display screen flashing)
- Sets indicator 192 on and resets some of the indicators, but not the special keyboard master indicator

  Note: The RESET key resets indicators 161, 187, 188, 189, 191, and 201 through 224. (See Appendix A.)

Column 30 Exit Control (O)

Valid entries for column 30 are:

Entry	Explanation
Blank	Data is not justified on exit.
J	Data is rightjustified on exit with blank fill. Field exit key required to exit field.
Z	Data is rightjustified on exit with zero fill. Field exit key required.

Columns 33-35 Data Position (R)

Valid entries for columns 33-35 are:

Entry	Explanation
A-Z	The register can be specified if data disposition is to a register only.
1-128	You can specify the location (for the beginning of the prompting message and/or the data) in the display screen buffer (buffer 2) if data disposition is to the buffer. (Data over 120 positions is moved into positions 41-48 of buffer 1.)

Column 36 Special Keyboard Close (O)

This entry specifies that the following keys (if a T is coded) are enabled to terminate a chain, close the keyboard, and set appropriate indicators and special keyboard master indicator 200. No data is moved to buffer 2 or the specified register if one of these keys is pressed:

FIELD BKSP
DUP
FIELD ADV
REC BKSP
REC ADV
SEL PROG

Columns 38-105

Entry	Explanation
1-68 characters	Prompting message

The prompting message can be entered in columns 38-105. The prompting message plus the field length must not exceed 68 characters. The prompting message must be terminated by an asterisk (*). Comments can be added after the asterisk. The extra set of numbers (5-72) on the coding form indicate the amount of buffer space taken by each prompting message. The numbers begin at 5 because each message is stored with a four-position control block (which is inserted in the .FIELD buffer when an overflow buffer is specified). The vertical dotted line at message position 35 indicates the end of the message on line 5 of the display screen. Thus, you can ensure that words in the prompting message are not split between lines 5 and 6 when displayed to the operator. Note that the message string cannot contain hexadecimal 00, hexadecimal FF, or *.

.END

The .END control statement must be the last statement in the ACL program. This statement indicates the end of source statements. You can also use this statement to call any work station operating mode, thus providing continuous operation.

Columns 1-4 Control Statement Name (R)

.END must be entered in columns 1-4.

Column 13 Operating Mode (O)

Valid entries for column 13 are:

Entry	Explanation
Blank	Calls normal translator termination to end the program.
A	Calls translator mode. This entry bypasses normal translator mode operator setup procedures for the next translation. (See Chapter 4.)
E	Calls execution mode. This entry bypasses normal execution mode operator setup procedures and executes the next label processor run, the ACL program that has just been through the label processor and the translator, or any ACL object program. (See Chapter 4.)

Columns 18-25 Input/Output Data Set Name (O)

If an A is entered in column 13, the input (source) data set name must be entered as it appears on the data set label. This data set contains the coded ACL statements and instructions.

If E is entered in column 13, the output (object) data set name must be entered as it appears on the data set label. This data set contains the object code following translation. (This entry is the same as the entry required in positions 1-8 of line 2 of the display for manual operator setup).

Column 26 Drive Number (O)

The diskette containing the source data set or object data set can be mounted on either disk drive 1 or 2. If an A is entered in column 13, the number of the disk drive containing the input (source) data set must be entered. If E is entered in column 13, the number of the disk drive containing the output (object) data set must be entered. No entry or an entry of 1 indicates drive number 1. An entry of 2 indicates drive number 2.

Columns 28-35 Output Data Set or Program Name (O)

If an A is entered in column 13, the output (object) data set name must be entered as it appears on the data set label. This data set contains the object code after translation.

If E is entered in column 13, the program name must be entered as it appears in columns 13-16 of the .NAME control statement. Note that this entry is four positions long. (This entry is the same as the entry required in positions 11-18 of line 2 of the display for manual operator setup.)

Column 36 Drive Number

The diskette containing the object data set can be mounted on either disk drive. You can specify that the object data set is mounted on drive 1 by leaving column 36 blank or entering 1. Drive number 2 is indicated by an entry of 2.

The A entry in column 13 allows you to translate multiple programs by chaining from one to the next. Figure 21 illustrates this use of the .END control statement.

Column	1	13	18	26	28	36
Entry	.END	A	Source Name	1	Object	2

Figure 21. Program Chaining with .END Statement

The E entry in column 13 allows you to execute a program immediately after that program is translated, without the normal operator setup procedure. Figure 22 illustrates this use of the .END control statement.

Column	1	13	18	26	28	36
Entry	.END	E	Object Name	1	Program Name	

Figure 22. Program Execution with .END Statement

INSTRUCTIONS

Your instructions to the work station can be entered on the preprinted *ACL Instructions Coding Sheet,* GX21-9199. You instruct the system by the coding of control information within instructions. All coded entries must be left-justified. Comments can be added (beginning in column 33) to clarify instructions for the operator. (Note that any record containing only a comment must be preceded by an asterisk in the first column on the coding sheet.) Instructions can either be preceded by step numbers (000-767 for 4K or 000-999 for 8K) in ascending order in positions 1-3 on the coding sheet, or by symbolic labels (up to four characters beginning with an alphabetic character). Instructions are of the following types:

- Arithmetic operations
- Branching operations
- Display and keyboard operations
- Diskette operations
- Printer operations
- Table operations
- Internal data movement operations
- Miscellaneous instructions

Arithmetic Operations

All arithmetic operations are performed on the contents of registers, with an F-zone added to each position in the register. Each register (A-Z) is 16 bytes long, and is negative if the low-order (units) position contains a D-zone.

All registers containing results of arithmetic operations are filled with zeros in the high-order positions. If the arithmetic result is zero, the entire register contains zeros. The general format of an arithmetic instruction is:

Column	1	8	13	18	23	28
Entry	Step/Label	Result	=	Factor 1	Oper	Factor 2

Column(s)	Contents
1	*Step/label* — is the number or symbolic label assigned to this instruction.
8	*Result* — is the register that contains the result of the arithmetic operation. This register is, in general, the only register changed in arithmetic operations.
18 and 28	*Factor 1 and factor 2* — these can be registers or a single-digit constant (except in the divide operation). The contents of these registers are generally unchanged, unless the registers are also specified as the result register.
23	*Operator* -- identifies the type of operation to be performed, such as add or subtract.

Add Instruction

The following is a typical add instruction.

Column	1	8	13	18	23	28
Entry	Step/Label	A	=	B	+	C

In this example, factor 1 (register B) and factor 2 (register C) are algebraically added and the sum is stored in the result register (register A). This example concerns addition of the contents of specified registers. However, the following example illustrates the use of a single-digit constant in the add instruction.

Column	1	8	13	18	23	28
Entry	Step/Label	A	=	B	+	5

In this example, factor 2 is a constant (5). Thus, the operation adds 5 to the contents of register B and stores the sum in register A. The contents of register B are unchanged, unless, for example, the operation is:

Column	1	8	13	18	23	28
Entry	Step/Label	A	=	A	+	5

In this case, the contents of register A would increase by 5. Indicators 159 and 160 are set on if a carry results out of the high-order position of the register. These indicators must be reset by your program. Failure to reset the indicators will increase processing time.

Subtract Instruction

The subtract instruction is structured much like the add instruction. An example of the subtract instruction is:

Column	1	8	13	18	23	28
Entry	Step/Label	A	=	B	–	C

In this example, the contents of register C are subtracted from the contents of register B and the result is stored in register A. The same parameters apply to both the add and subtract operations.

Results of the add/subtract operations may be invalid if any of the numeric halves of the bytes are other than 0-9. The zone halves of the bytes are not used, except for sign control. The zones in the result field are hex F, except for the sign which, if negative, is a D-zone for the rightmost (position 16) digit only.

Multiply Instruction

The multiply instruction, like the add and subtract, can employ either the contents of specified registers and/or single-digit constants. A sample multiplication operation is:

Column	1	8	13	18	23	28
Entry	Step/Label	A	=	B	*	C

In this example, the contents of register B are multiplied by the contents of register C and the result is stored in register A. Many variations using single-digit constants are also possible. Such a variation is:

Column	1	8	13	18	23	28
Entry	Step/Label	A	=	3	*	8

In this example, the contents of register A are changed to the constant 24.

The multiplicand and multiplier registers cannot be the result register. If a carry results from the high-order position, indicators 158 and 160 are set on. These indicators must be reset by your program. Failure to reset the indicators will increase processing time. If the numeric halves of the bytes are something other than 0-9, results of the multiply/divide operations may be invalid. The zone halves of the bytes are not used, except for sign control. The zones in the result field are hex F, except for the sign which, if negative, is a D-zone on the units digit.

Divide Instruction

The divide operation causes factor 1 to be divided by factor 2, with the result stored in the result register. A sample divide instruction is:

Column	1	8	13	18	23	28
Entry	Step/Label	A	=	B	/	C

In this example, the contents of register B are divided by the contents of register C, with the result being stored in register A. The remainder is stored in register B, while register C is unchanged. The result register (A) and the dividend register (B) cannot be the same. A single-digit constant can be entered for the divisor register (C), but not for the dividend (B). Both dividend and divisor can be signed quantities. If both have the same sign, the result is positive; if the signs are different, the result is negative. The remainder keeps the sign of the dividend. Factor C may be a constant.

No checking is made to determine if the data in the registers is alphabetic or numeric. If the divisor is zero, indicators 157 and 160 are set on. These indicators must be reset by the program.

Shift Left Logical Instruction

The shift left instruction is designated by an L in column 23 on the coding sheet, with column 24 blank. This instruction shifts the contents of factor 1 (column 18) to the left, according to the number of bytes entered for factor 2 (column 28). A sample shift left instruction is:

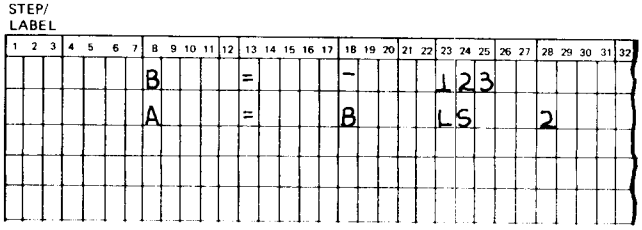
Column	1	8	13	18	23	28
Entry	Step/Label	A	=	B	L	2

In this example, the contents of register B (factor 1) are shifted two positions (factor 2) to the left, with the result stored in register A (result register). Factor 2 can be a number (1-15) or a register. If factor 2 is a register, the shift count is determined from the numeric position of the units byte in the register (0-F). The shifted result is placed in the specified result register. The low-order bytes of the result register are replaced by blanks. Data shifted out of the high-order end of the result register is lost. The previous contents of the result register are also lost. Factor 1 does not change unless it is also specified as the result register. Factor 2 does not change.

Shift Left Signed Instruction

This instruction is designated by an L in column 23, and an S in column 24 on the coding sheet. The execution of the instruction is similar to the execution of the shift left logical instruction, except that the sign of the result is the same as the sign of the register specified by factor 1 (column 18). The low-order bytes of the register are filled with zeros.

For example, the following instructions would cause register A to contain – 12300.



Note that register contents are shown in decimal, although actual register contents are in hexadecimal, with a D-zone sign in the low-order position.

Shift Right Logical Instruction

The shift right instruction is designated by an R in column 23 on the coding sheet, with column 24 blank. This instruction shifts the contents of factor 1 (column 18) to the right, according to the number of bytes entered for factor 2 (column 28). A sample shift right instruction is:

Column	1	8	13	18	23	28
Entry	Step/Label	A	=	B	R	2

In this example, the contents of register B (factor 1) are shifted two positions (factor 2) to the right, with the result stored in register A (result register). Factor 2 can be a number (1-15) or a register. If factor 2 is a register, the shift count is determined from the numeric portion of the units byte in the register (0-F). The shifted result is placed in the specified result register. The high-order bytes of the result register are replaced by blanks. Data shifted out of the low-order end of the result register is lost. The previous contents of the result register are also lost. Factor 1 does not change unless it is also specified as the result register. Factor 2 does not change.

The shift right logical instruction may be used to blank a register by using a constant (0-9) as factor 1. A sample instruction blanking a register is:

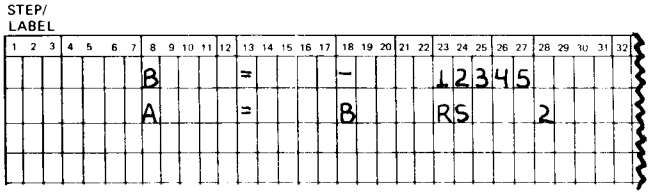
Column	1	8	13	18	23	28
Entry	Step/Label	A	=	1	R	1

In this example, register A is blanked.

Shift Right Signed Instruction

This instruction is designated by an R in column 23, and an S in column 24 on the coding sheet. The execution of the instruction is identical to the execution of the shift right logical instruction, except that the sign of the result is the same as the sign of the register specified by factor 1 (column 18). The high-order bytes of the register are filled with zeros.

For example, the following instructions would cause register A to contain – (minus) 0000000000000123 (register contents are shown in decimal):



Shift Right Round Instruction

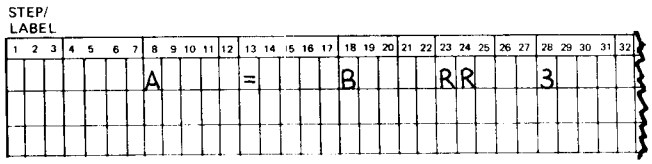
This instruction is designated by an R in column 23 and an R in column 24 on the coding sheet. Execution is similar to execution of the shift right signed instruction, except that the sign from the factor 1 register is maintained and the units position of the result register is incremented by one if the last character shifted out of the register is 5 or greater. The high-order bytes of the register are filled with zeros.

For example, assume that register B contains:

–(MINUS)0000000001234567

The following instruction would cause register A to contain:

–(MINUS)0000000000001235



Note: Register contents are shown in zone decimal format.

Move Register to Register Instruction

The move register instruction involves only a result register and factor 1. In this operation, the contents of the factor 1 register replace the contents of the result register. Factor 1, which is not changed, may be a single-digit constant (0-9). Sample move register instructions are:

Column	1	8	13	18	23	28
Entry	Step/Label	A	=	B		

In this example, the contents of register B are stored in register A. Register B is unchanged. Register contents are stored with blank fill if you specify a constant (0-9).

Column	1	8	13	18	23	28
Entry	Step/Label	A	=	1		

In this example, the constant 1 is stored with blank fill as the contents of register A.

Assigning a Constant Value to a Register

The instruction shown below stores a constant (up to 65535) into a specified register.

Column	1	8	13	18	23
Entry	Step/Label	Result	=	Sign	Value

Column	Contents
1	*Step/label* – is the number or symbolic label assigned to this instruction.
8	*Result* – is the register that will contain the assigned constant value.
18	*Sign* – you can also specify whether the constant is positive or negative. If the value is to be positive, enter a plus (+) sign or leave column 18 blank. If the value is to be negative, a minus (-) sign must be entered. High-order register positions are filled with zeros.
23	*Value* – is the constant being assigned to the result (+/-) register. This value can be five digits up to a maximum of 65535.

A sample load constant instruction is:

Column	1	8	13	18	23
Entry	Step/Label	A	=	+	2

In this example, register A would contain the positive value 2. Assigned values are stored with zero fill.

Branching Operations

The sequence of instruction execution in the work station program can be altered by unconditional and conditional branching instructions. Conditional branches must be within the same 256 instruction block.

The following is a detailed description of each branching instruction.

Unconditional Branch (GOTO)

This instruction orders an unconditional branch within the ACL program. The format of a GOTO instruction is:

Column	1	8	13	18	23	28
Entry	Step/Label	GOTO				Step/Label

Column	Contents
1	*Step/label* – is the number or symbolic label assigned to this instruction.
8	*GOTO* – identifies the operation to be performed (branch to the specified instruction).
28	*Step/label* – is the number or label of the instruction to which the branch should be taken.

Upon execution of this instruction, the instruction specified in column 28 is the next instruction executed by the program. A sample GOTO instruction is:

Column	1	8	28
Entry	DAT3	GOTO	DAT1

The next instruction executed is at label DAT1.

Indexed GOTO Unconditional Branch

This instruction branches within the ACL program by simply changing the contents of a register. The format of an indexed GOTO instruction is:

Column	1	8	13	28
Entry	Step/Label	GOTO	Index Reg	Step Number or Label

Column	Contents
1	*Step/label* – is the number or symbolic label assigned to this instruction.
8	*GOTO* – identifies the operation to be performed (branch to the resulting instruction).
13	*Index register* – The value of the low-order three bytes of this register is algebraically added to the number specified in column 28. The branch is taken to the resulting step number. The index is negative if the units position of the index register contains a D-zone.
28	*Step number or label* – this number and the value of the low-order three bytes of the index register are added to arrive at the number or label of the instruction to which the branch should be taken.

A sample indexed GOTO instruction is:

Column	1	8	13	28
Entry	DAT9	GOTO	A	DBT4

Assume that register A contains a minus 20. In this case, the next instruction to be executed is at label DBT4 minus 20 instructions.

Return Transfer [Subroutine Call] (RGO)

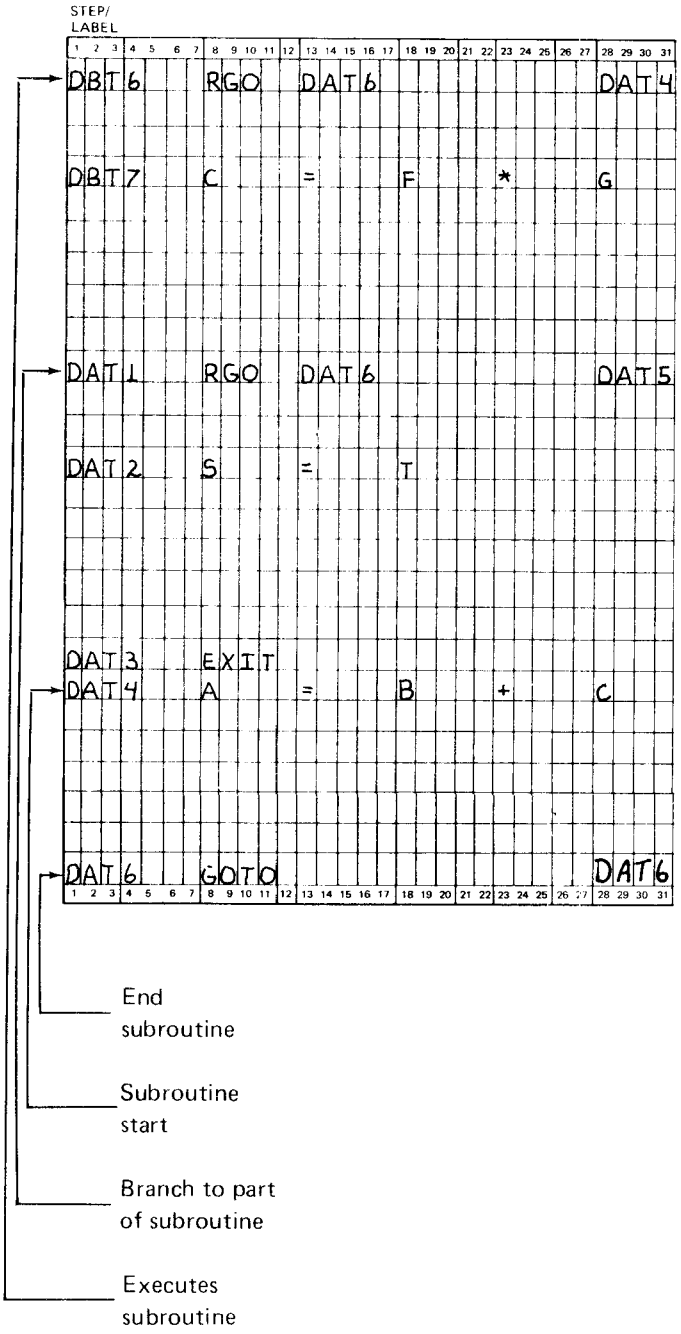
This instruction branches to and executes a subroutine, then returns to the normal sequence of program execution. A sample RGO instruction is:

Column	1	8	13	28
Entry	Step/Label	RGO	End Step Number or Label	Branch Step/Label

Column	Contents
1	*Step/label* – is the number or symbolic label assigned to this instruction.
8	*RGO* – identifies the operation to be performed (execute specified branch instruction and return).
13	*End step number or label* – is the number or label of the instruction that ends the subroutine and returns the program to the step number or label following the RGO instruction.
28	*Branch step number or label* – is the number or label of the instruction to which the branch is made. Upon execution of the RGO instruction, this step or label is branched to and executed. Succeeding instructions are then executed until the end step/label is reached.

Upon execution of the RGO instruction, the program branches to and executes the instruction specified in column 28. The program then continues to execute instructions following that instruction until the end step/label is reached. This instruction must be a dummy GOTO. A dummy GOTO should reference the step number/label on the GOTO as shown in the following example. The actual step number is supplied by the system (when the RGO is executed) with the step number or label of the instruction immediately following the RGO instruction. Thus, the end step/label causes the program to branch back to one step past the RGO.

A sample RGO instruction is:



In this example, when the RGO instruction at label DBT6 is executed, the returning label (DBT7) is stored at the label address of the dummy GOTO instruction (DAT6). The subroutine starting at label DAT4 is then executed and, when label DAT6 is executed, the GOTO branches to label DBT7. When the program reaches label DAT1, the dummy GOTO (DAT6) contains DAT2. If an EXIT or GOTO is not at label DAT3, the program branches back to DAT2.

Conditional Branching Instructions

The following conditional branching instructions change the sequence of program execution if specified conditions are met. If these conditions are not met, the next sequential instruction is executed.

If Register Zero or Blank: The format of this instruction is:

Column	1	8	13	18	23	28
Entry	Step/Label	IF	Reg. Tested	IS/NOT	0	Branch Step/Label

Column **Contents**

1 *Step/label* – is the number or symbolic label assigned to this instruction.

8 *IF* – identifies the conditional branching operation.

13 *Register tested* – the register (A-Z) that is being tested for the specified condition.

18 *IS/NOT* – identifies the condition (with column 23) that controls branching.

23 *0* – identifies the other condition that controls branching.

28 *Branch step/label* – indicates the step or label to which the program branches if the conditions (columns 18 and 23) are met.

This instruction causes a branch to the specified step or label if the tested register contains a zero or blank value.

By entering NOT in column 18, you could cause the program to branch to the specified step number or label if the tested register does not contain a zero or blank value.

If Register Is/Not Negative: The format of this instruction is:

Column	1	8	13	18	23	28
Entry	Step/Label	IF	Reg. Tested	IS/NOT	–	Branch Step/Label

Column **Contents**

1 *Step/label* – is the number or symbolic label assigned to this instruction.

8 *IF* – identifies the conditional branching operation.

13 *Register tested* – the register that is being tested for the specified conditions.

18 *IS/NOT* – identifies the condition (with column 23) that controls branching.

23 *- (minus)* – identifies the other condition that controls branching (with column 18).

28 *Branch step/label* – indicates the step/label to which the program branches if the specified conditions (columns 18 and 23) are met.

This instruction causes the program to branch to the specified step/label if the units position (low-order) of the tested register has a hex D-zone (negative value) and IS is entered in column 18. By entering NOT in column 18, you can cause the program to branch to the specified step/label if the units position of the tested register does not have a D-zone.

If Registers Equal: The format of this instruction is:

Column	1	8	13	18	23	28
Entry	Step/Label	IFD or IF	R1	=	R2	Branch Step/Label

Column	Contents
1	*Step/label* -- is the number or symbolic label assigned to this instruction.
8	*IFD* -- identifies the conditional branching operation.
13	*R1* -- one of the registers to be compared.*
18	= -- identifies this operation as a compare of two registers, with branching to occur if the registers are found to be equal.
23	*R2* -- the second register to be compared or a constant (+0-9).
28	*Branch step/label* -- indicates the step/label to which the program branches if the specified conditions are met. This must be in the same 256-step instruction block (a contiguous string of instructions starting at 0-255, 256-511, 512-767). (See Error Message 119 in Appendix B.) *Note:* To get from one instruction block into another, you can issue a branching instruction to a GOTO instruction in the same block. The GOTO instruction could then be coded to transfer the program to an instruction within another instruction block.

Upon execution of this instruction, the program branches to the specified step/label if the two registers specified are equal to each other (bit by bit). If a D is entered in column 10, the branch to the specified step/label is taken if the digit portions of the registers compare, regardless of zone portion (blanks or zeros), except for the sign zone in the low-order positions. All indicators, registers, and buffers are unchanged. The register digit postion compare is algebraic.

*If R1 contains leading zeros and R2 is a constant, a branch is not made. However, if R1 contains leading blanks and R2 is a constant, the branch is made. Thus, IFD should be used for numeric compares, and IF should be used for character compare of register contents (R1 and R2).

If Registers Greater/Less: The format of this instruction is:

Column	1	8	13	18	23	28
Entry	Step/Label	IFD or IF	R1	</>	R2	Branch Step/Label

Column	Contents
18	< -- indicates that the branch should be taken if the contents of the R1 register compare less than the contents of the R2 register.
18	> -- indicates that the branch should be taken if the contents of the R1 register compare greater than the contents of the R2 register.

All other operands and the execution of these instructions are identical to the operands and execution of the if registers equal instruction. The D entry in column 10 specifies that the algebraic compare be done only on the digit portions of the registers, regardless of zone characters (blanks or zeros), except for sign zone. All indicators, registers, and buffers are unchanged.

If Indicator: This instruction controls branching on the condition of a specified indicator. The format of this instruction is:

Column	1	8	13	18	23	28
Entry	Step/Label	IFI	Ind	IS/NOT	ON	Branch Step/Label

Column	Contents
1	*Step/label* – is the number or symbolic label assigned to this instruction.
8	*IFI* – identifies this branching operation as based on the condition of an indicator.
13	*Indicator* – identifies the indicator (1-255) being tested:
18	*IS/NOT* – identifies the condition controlling this branching operation.
23	*ON* – indicates the status of the indicator being tested.
28	*Branch step/label* – indicates the step or label to which the program will branch, based on the conditions specified.

If the entry in column 18 is IS, the program branches to the specified step number if the tested indicator is on. The indicator is not reset. If the entry in column 18 is NOT, the program branches to the specified step/label if the tested indicator is not on. The indicator is not reset.

In order to reset the specified indicator after this operation, you can enter IFIR in column 8 on the coding sheet. All other operands for the IFIR IS/NOT instructions are the same as shown above. If NOT is specified in column 18 of an IFIR instruction, and the tested indicator is on, the branch is not taken, but the indicator is still reset. Indicators for the three switches (AUTO REC ADV, PROG NUM SHIFT, AUTO DUP/SKIP) and the error line (keyboard lock/display flash) are updated according to their status when the instruction is executed. No registers or buffers are changed by these instructions.

If Format: This instruction controls branching according to the use of a specified format (1-254). When used in conjunction with a data-directed read operation (see *Read,* Column 18, under *Diskette Operations* in this chapter), this instruction must follow the READ instruction. If format instructions can be used to compare the last format used during any operation using a format, then branch to a specific routine designed to handle that type of record. Note that, on end of file, a format being used is not completed. Thus, the if format branch cannot be taken on this format. This instruction is shown below:

Column	1	8	13	18	23	28
Entry	Step/Label	IF	FMT	IS/NOT	F	Branch Step/Label

Column	Contents
1	*Step/label* – is the number or symbolic label assigned to this instruction.
8	*IF* – identifies this as a conditional branching operation.
13	*FMT* – indicates that a format number is being tested.
18	*IS/NOT* – identifies the condition that controls branching.
23	*F* – is the format number tested.
28	*Branch step/label* – indicates the step to which the program branches if the specified conditions are met.

Upon execution of this instruction, the program branches to the specified step number or label if column 18 contains IS, and column 23 contains the number of the last format level used in any instruction with a format parameter coded. The branch is also taken if column 18 contains NOT and column 23 does not contain the number of the last format level used. If the branch condition is not met, the next sequential instruction is executed. These instructions do not set or reset any indicators or change any buffers or registers.

Skip If Character Is/Not Equal: This instruction skips the next sequential instruction according to the presence of a specified character. The format of this instruction is:

Column	1	8	13	18	23
Entry	Step/Label	SCE/SCN	Buff/Reg	Position	Character

Column	Contents
1	*Step/label* – is the number or symbolic label assigned to this instruction.
8	*SCE* – indicates that the next sequential step should be skipped if the tested character matches the entry in column 23.
8	*SCN* – indicates that the next sequential step should be skipped if the tested character does not match the entry in column 23.
13	*Buffer* – the number of the buffer containing the character to be tested.
13	*Register* – if the character to be tested is not in a buffer, the register (A-Z) containing the character is specified in column 13.
18	*Position* – the location of the character to be tested from the left of the buffer or from the leftmost position of the register containing the character.
23	*Character* – this is the character of immediate data being tested.

Upon execution of this instruction, the next sequential instruction is skipped if SCE is entered in column 8, and the character located at the specified position matches the character specified in column 23. If SCN is entered in column 8, and the character at the position does not match the character specified in column 23, the next instruction is also skipped. These instructions do not change any indicators, registers, or buffers.

If Register Is/Not Absolute Numeric: This instruction branches to a specified step number or label according to the contents of a register. The format of the instruction is:

Column	1	8	13	18	23	28
Entry	Step/Label	IF	Reg	IS/NOT	AN	Branch Step/Label

Column	Contents
1	*Step/label* – is the number or symbolic label assigned to this instruction.
8	*IF* – identifies this as a conditional branching operation.
13	*Register* – is the register (A-Z) being tested.
18	*IS* – indicates that the program should branch to the step number specified in column 28 if the register tested has hexadecimal F-zones in all nonblank positions. The program branches if the register contains all blanks. The register may not contain intermixed or trailing blanks.
18	*NOT* – indicates that the program should branch to the step/label specified in column 28 if the register tested has one or more nonblank characters with a hexadecimal zone other than F, or has intermixed or trailing blanks.
23	*AN* – identifies the condition for which the register is being tested as absolute numeric.
28	*Branch step/label* – is the step or label to which the program branches if the specified conditions are met.

If Register Is/Not Signed Numeric: This instruction branches to a specified step/label according to the contents of a register. The instruction format is:

Column	1	8	13	18	23	28
Entry	Step/Label	IF	Register	IS/NOT	SN	Branch Step/Label

Column	Contents
1	*Step/label* – is the number or symbolic label assigned to this instruction.
8	*IF* – identifies this as a conditional branching operation.
13	*Register* – is the register (A-Z) being tested.
18	*IS* – indicates that the program should branch to the specified step/label if the register tested has hexadecimal F-zones in all nonblank positions except the units position (which may have either an F or a D-zone). The numeric characters must be contiguous from the rightmost position with no intermixed blanks. A branch is taken if the tested register contains all blanks. The register may not contain intermixed or trailing blanks.
18	*NOT* – indicates that the program should branch to the specified step number if the register tested does not have hexadecimal F-zones in all nonblank positions (except the units position). The program does not branch if the tested register contains all blanks.
23	*SN* – identifies the condition for which the register is being tested as signed numeric.
28	*Branch step/label* – is the step or label to which the program branches if the specified conditions are met.

If Register Is/Not Self-Check: The work station computes a self-check digit based on the algorithm in the .SELF-CHECK statement, from the contents of a register, then uses that digit to control branching to a specified step/label. The format of the instruction is:

Column	1	8	13	18	23	28
Entry	Step/Label	IF	Register	IS/NOT	CHK	Branch Step/Label

Column	Contents
1	*Step/label* – is the number or symbolic label assigned to this instruction.
8	*IF* – identifies this as a conditional branching operation.
13	*Register* – specifies a register (A-Z) to be tested.
18	*IS* – indicates that a self-check digit should be computed, according to specifications contained in the .SELF-CHECK control statement, from the data contained in the register specified in column 13. The computed self-check number is then compared to the check number contained in the register (as specified in the .SELF-CHECK control statement). If the numbers compare, the program branches to the specified step/label (if IS is entered in column 18).
18	*NOT* – indicates that, after arriving at the computed self-check digit as described above, if the numbers do not compare, the program branches to the specified step/label.
23	*CHK* – identifies the condition for which the register is being tested.
28	*Branch step/label* – the step or label to which the program branches if the specified conditions are met.

If Printer Is/Not Busy: This instruction branches the program to a particular instruction depending on the status of the printer. The format of the IF PRT instruction is:

Column	1	8	13	18	23	28
Entry	Step/Label	IF	PRT	IS/NOT	BSY	Branch Step/Label

Upon execution of this instruction, the program branches to the step/label specified in column 28 if the status of the printer matches the status specified in column 18. For example, if IS is entered in column 18, and the printer is busy, the branch is taken. If the printer does not match the status specified in column 18, the program continues to the next sequential instruction. If an error is pending from the last print instruction, the printer error is posted, the print retry is initiated, and the program control goes to the IF PRT instruction.

If CRD Is/Not Busy: This instruction branches the program depending on the status of the attached card I/O device (129 or 5496). The format of the instruction is identical to the if-printer-is/not-busy instruction, except that CRD is required in columns 13-16. The format of the IF CRD instruction is:

Column	1	8	13	18	23	28
Entry	Step/Label	IF	CRD	IS/NOT	BSY	Branch Step/Label

Display and Keyboard Operations

The ENTR instruction is the means by which the parameters in a .FIELD control statement are implemented. This instruction allows prompting to be displayed to the operator, defines the type of data to be entered, accepts data entered via the keyboard, and defines the disposition of that data.

The format of an ENTR instruction is:

Column	1	8	13	18	23
Entry	Step/ Label	ENTR	Buffer/ Register	Message Number/ Register	Overlap

Column	Contents
1	*Step/label* – is the number or symbolic label assigned to this instruction.
8	*ENTR* – indicates that the keyboard should be opened to accept input.
13	*Buffer/register* – identifies which buffer contains the desired prompting message. This buffer is also specified in columns 13-14 of the .FIELD control statement. A register (A-Z) containing the buffer number can also be specified in column 13 of the ENTR instruction.
18	*Message number/register* – identifies which message (1-99) within the buffer is to be displayed. An entry of 1 indicates that the message starts in the first position of the buffer. If the message number is greater than 1, the specified buffer is scanned beginning at the first position. A register containing the message number can also be specified in column 18. The first message in an overflow buffer is message number 2 (see *.FIELD, Columns 18-19*).
23	*Overlap* – this entry determines if operation is to be overlapped or nonoverlapped. Overlapped operation (enter an X in column 23) means that processing of subsequent instructions continues concurrently with the data being entered. Nonoverlapped operation (leave column 23 blank) means that processing of subsequent instructions does not continue until the ENTR instruction is exited (waits for data to be entered). Note that the ENTR operations cannot be overlapped with the OPEN and CLOZ instructions.

The first prompting message defined by the ENTR instruction is displayed, along with fill characters defining the field to be entered, in lines 5 and 6 of the display screen. The field to be entered is identified by a series of either dashes (numeric) or periods (alphabetic). Note that the system inserts a blank between the prompting message and the field to be entered.

Diskette Operations

Reading and writing of records on a diskette are controlled by disk instructions and by the .DATASET control statement. The results of execution of a disk instruction depend on the information contained in the .DATASET control statement (see *.DATASET* in this chapter). Disk instructions are described in detail in the following paragraphs.

Read (READ)

A read instruction causes the system to read or search and read a data record from a data set on a diskette. The format of a read instruction is:

Column	1	8	13	18	23
Entry	Step/ Label	READ	Data Set	Fmt No.	Register, Key, or Minus

Column	Contents
1	*Step/label* — is the number or symbolic label assigned to this instruction.
8	*READ* — identifies the operation to be performed (read a data record).
13	*Data set* — specifies the number assigned to the data set in the .DATASET control statement. The buffer specified in the .DATASET control statement (columns 38-39) contains the record after the read. The valid range for this required entry is 1-4.
18	*Format no.* — specifies the number of the format to be used after completion of the physical read. Formatting is from the I/O buffer into the registers specified in the format record. The formatting is data-directed if a D is entered into column 18 of the READ instruction instead of a format number (see *Data Directed Formatting* in this chapter). If no entry is made in column 18, no formatting takes place, but the I/O buffer (specified in the .DATASET control statement) contains the record just read.
23	*Register* — indicates a register address, key, or can be blank or minus. If data set access is sequential (except write or write extend), the specified register contains the relative record number to be read. This number is relative to BOE. The register value is incremented by one after each READ instruction. For key indexed access method, the entry in column 23 on the coding sheet indicates the register containing the key of the record to be read. If a matching record is not found, indicators 225-228 are set on to indicate data sets 1-4, respectively, as specified in columns 13-15. Formatting does not take place, and the next higher record is contained in the I/O buffer. If the index value is not within the index table range, indicators 229-232 are set on to indicate the respective data set specified in columns 13-15, and no disk operation takes place. If no key register address is entered, the next sequential record is read, regardless of data set access method, except for sequential write or sequential write extend (READ instruction invalid). If a minus sign is entered, the preceding nondeleted sequential record is read unless the deleted record exit procedure has been coded in the .DATASET statement (except at BOE), regardless of data set access method, except for sequential write or sequential write extend (READ instruction invalid). If a deleted record is read, and the deleted record exit has been specified in the .DATASET control statement, the exit routine is taken. If the exit is not specified in the .DATASET statement, sequential reads are issued until a nondeleted record or EOD (end of date) is encountered. If deleted records are skipped, the relative number is incremented for each deleted record read.

A sample READ instruction is:

Column	1	8	13	18	23
Entry	Step/Label	READ	1	15	A

Sequential or Index Update Access Methods: If register A contains 53 and BOE equals 10001, this instruction reads record number 12001 (track 12, sector 1). Format level 15 is used to fill registers after the physical read. Register A contains 54 at the completion of the instruction. The file disk address is 12001 assuming that there are no deleted records within the file.

Key Indexed Access Method: Assuming that register A contains the key *Jones*, this instruction reads the first record with the matching key of *Jones* and uses format level 15 for formatting. The file disk address is the address of the record containing *Jones*. If no match is found, indicator 225 is turned on, no formatting takes place, and the file disk address is the next higher record above the position where the match would have been. The I/O buffer for the data set also contains the next higher record.

Write Disk Record (WRT)

The work station will write a record on the diskette and allow for overlapped machine and operator-machine functions, when this instruction is used.

The format of a WRT instruction is:

Column	1	8	13	18	23
Entry	Step/ Label	WRT	Data Set	Format	Buffer/Register

Column	Contents
8	*WRT* — identifies the operation to be performed (write record).
13	*Data set* — the number assigned to the data set in the .DATASET control statement. This required operand has a valid range of 1-4.
18	*Format* — identifies the number of the format to be used before the physical write. Formatting is from registers to the output buffer. The output buffer specified in the .DATASET control statement (columns 38-39) contains the record to be written. If no entry is made, formatting does not take place. (Note that, if both the format and buffer numbers are omitted, a blank record is written.)
23	*Buffer* — identifies the number of the buffer to be moved into the data set output buffer, if any. This is the first operation of the instruction execution. The contents of the specified buffer are not changed. If no entry is made, the I/O buffer is blanked before formatting. If both the buffer number of the data set and this entry are the same, the contents of the I/O buffer are not changed.
23	*Register* — identifies the register (A-Z) which contains the relative record number within the data set where the write takes place. This number is relative to BOE. When a register is coded, no buffer movement or blanking takes place at execution. This type of write is only valid to a sequential update (SU) or label update (I) data set. A write invalid error (7XC) is posted if SU or I type is not specified. A relative record number of 1 accesses the BOE record. A record number of zero, or beyond BOE, posts a 7XD error. A record number greater than or equal to end of file posts a 7XC error. If the 7XC or 7XD errors are detected, any specified formatting is performed. Under this condition, the write operation is suppressed. The register value is incremented by 1 after completion of this instruction.

If a read with relative record number is issued prior to the write with relative record number, the record written will be one greater than expected. For example:

Assume register A = 10:

Column	1	8	13	18	23
Entry	Step/Label	READ	1		A

Register A is incremented by one after the read:

Column	1	8	13	18	23
Entry	Step/Label	WRT	1		A

Register A contains a value of 11, not 10. Therefore, use the following:

Column	1	8	13	18	23
Entry	Step/Label	WRT	1		I/O buffer number

A sample WRT instruction is:

Column	1	8	13	18	23
Entry	Step/Label	WRT	2	5	6

In this example, the instruction moves the contents of buffer 6 into the output buffer specified, then formats data according to format 5. The current file disk address and EOD are incremented, if necessary, and the record from the specified buffer is written at the current file disk address (data set number 2). The contents of buffer 6 are unchanged.

Extend Data Set and Write Disk Record (WRTE)

The work station will write a record on the diskette and allow for overlapped machine and operator-machine functions, when this instruction is used.

The format of a WRTE instruction is:

Column	1	8	13	18	23
Entry	Step/Label	WRTE	Data Set	Format	Buffer

Column	Contents
1	*Step/label* – is the number or symbolic label assigned to this instruction.
8	*WRTE* – identifies the operation to be performed (extend data set and write disk record).
13-24	All other operands in this instruction are defined as for the WRT instruction.

Execution of the WRTE instruction is identical to execution of the WRT instruction except that it is valid only for SU data sets so that the disk is automatically positioned at EOD prior to writing.

Delete a Disk Record (WRTS)

The format of a WRTS instruction is:

Column	1	8	13	18	23
Entry	Step/Label	WRTS	Data Set	Format	Buffer/ Register

Column	Contents
8	*WRTS* – identifies the operation to be performed (delete a disk record).
13-24	All other operands are defined as described for the WRT instruction. Execution of a WRTS instruction is identical to execution of a WRT instruction, except that a special data address mark is written to the disk outside of the 128-position record. If the format and buffer numbers are not specified, the character D is forced into the first position of the I/O buffer. Thus, if a buffer or format is specified, a D must be forced into the first position. Therefore, an instruction with only a data set number entered writes a deleted record, as defined by basic disk interchange.

To ensure successful execution of this instruction, the following checks are made:

- The first character of the written record is saved.
- The record is read after the write attempt.

A record is considered deleted if the following conditions are met:

- A cyclic redundancy check (CRC) is successfully made at the end of the record.
- A special address indicator is detected.
- The first position of the record contains a D.
- No other error indicators are on.

Note: A deleted record is written by the Models 1 and 2 data station with a D in the first position of the record, the CRC and a special address indicator. However, no check is made to see that the D or CRC was written on disk. Only the special address indicator is checked when a deleted record is read by the Models 1 and 2 or the Models 3 and 4 in data station mode.

A sample WRTS instruction is:

Column	1	8	13	18	23
Entry	Step/Label	WRTS	1		

In this example, a deleted record is written to data set 1. The current file disk address is changed, if necessary.

Wait I/O (WAIT)

This instruction ensures that any input/output operation is completed before executing the next instruction. This instruction waits until all outstanding I/O is complete including keyboard, diskette, and printer, and detects any pending errors. The instruction will not, however, wait on card I/O. Note that the IF PRT BSY instruction can be used to wait on the printer with errors detected. If any errors are detected on overlapped files, the system posts a 700 series error message and the operator must abort the job. The instruction is coded by the entry WAIT in columns 8-11, following the step number or label. The IF CRD instruction must be used to wait on card I/O.

Open Data Set (OPEN)

This instruction controls dynamic opening of data sets during program execution. However, if this instruction is not used, data sets are opened by the system prior to execution. The re-opening of an open data set resets the EOD. The format of an OPEN instruction is:

Column	1	8	13	18	23
Entry	Step/Label	OPEN	Data Set	Format	Register

Column	Contents
1	*Step/label* – is the number or symbolic label assigned to this instruction.
8	*OPEN* – identifies the operation to be performed (open data set).
13	*Data set* – the number assigned to the data set in the .DATASET control statement. This required operand has a valid range of 1-4.
18	*Format* – identifies the format to be used when reading desired information from the data set label into registers. If the data set cannot be opened, no formatting occurs. If nothing is entered in column 18, no formatting occurs. Zero is invalid.
23	*Register* – identifies a register containing information controlling the definition of certain data set characteristics (such as name, drive number). All pending operations to the data set are completed before changing data set attributes.

The specified register in column 23 may contain (left-justified) the information in the following specified bytes:

Bytes	Contents
1-8	Data set name
9	Drive number (drive one = 1, drive two = 2)
10	Data set access method (See *Columns 58-60 Type (R)* under *.DATASET* in this chapter.)
11	Data set access method (See *Columns 58-60 Type (R)* under *.DATASET* in this chapter.)
12	Data set access method (See *Columns 58-60 Type (R)* under *.DATASET* in this chapter.)
13	Suppress extent checking (Enter character A). *Note:* Suppressing extent checking improves performance. Also, the extent check will fail for null data sets received in BSCA mode.
14	Redefine additional parameters in next sequential register (Enter character X).
15-16	Not used

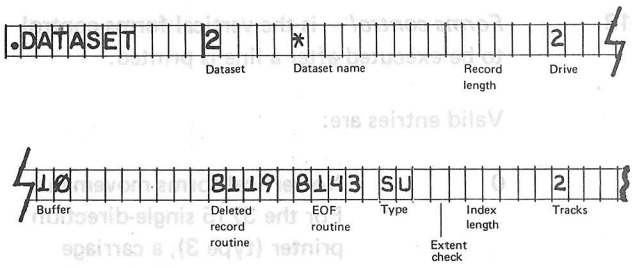
If an X is entered in position 14 of the register, additional attributes are redefined in the next sequential register (for example, A and B). (Bytes 1-12 apply only to key indexed data set access methods.) These parameters must be right-justified with leading zeros or blanks.

Bytes	Contents
1-3	Number of bytes per index entry (See *Columns 63-64 Index Length (O)* under *.DATASET* in this chapter.)
4-6	Number of tracks per index entry (See *Columns 68-69 Tracks/Index (O)* under *.DATASET* in this chapter.)
7-9	Number of bytes per key (See *Columns 73-74 Key Length (O)* under *.DATASET* in this chapter.)
10-12	Position of key (starting position) in the record (See *Columns 78-80 Key Position (O)* under *.DATASET* in this chapter.)
13-15	Record length (See *Columns 28-30 Record Length (O)* under *.DATASET* in this chapter.)
16	Not used

If these parameters are to be bypassed (use the original .DATASET specification), blanks must be inserted in the appropriate positions of the register (except for extent checking). If no register is specified in column 23 of the OPEN instruction, the existing data set is opened (as defined in the .DATASET control statement). A sample OPEN instruction is:

Column	1	8	13	18	23
Entry	Step/Label	OPEN	2	3	S

Register S = Positions: 1 6 10 16
Register S Contents: PROBLEM SU
Relative .DATASET control statement:



In this example, data set 2 is opened, formatting is done according to format 3, and the data set is defined as an SU. The data set name is PROBLEM and extent checking is specified. The disk drive number is that defined in the .DATASET control statement for this data set, and register T is not used (no X in position 14 of register S).

Open Data Set Errors: An attempt to open a data set that is open already opens the requested data set and does not update the original file extents. If a data set cannot be opened, a 500 series error, along with the data set number, is posted. This error can be reset by:

- The RESET key which retries the open sequence.
- ALPHA or NUMERIC SHIFT and RESET which closes the data set and posts a 100 error (job complete).
- ALPHA and NUMERIC SHIFT and RESET which returns the work station to index mode.

End of Job (EXIT)

All valid ACL programs for the work station must include an EXIT instruction for a normal end of job (unless an EXEC instruction is specified). This instruction closes all data sets and waits for all printer operations to complete (system close), and then posts a 100 halt message on the status line (screen flashes). RESET must be pressed to return to data station mode. You can indicate an EXIT instruction by simply entering EXIT beginning in column 8 on the instruction coding sheet. A sample EXIT instruction is:

Column	1	8
Entry	Step/Label	EXIT

Close Instruction (CLOZ)

Although the system automatically closes data sets when the EXIT instruction is used, you can control the dynamic closing of data sets during execution with the CLOZ instruction. The format of a CLOZ instruction is:

Column	1	8	13
Entry	Step/Label	CLOZ	Data set Number

Column	Contents
8	*CLOZ* — identifies the operation to be performed (close data set).
13	*Data set number* — identifies the number of the data set to be closed.

Closing Data Sets: The following steps occur during the closing of a data set:

1. All physical input/output to the data set is completed. A close instruction for a data set already closed is ignored and does not cause an error.
2. The data set label is read (SU, SW, SWE organization only). If this read develops an I/O error, the CLOZ instruction posts a terminal error message in the display.
3. The EOD address on the label is updated.
4. The label is written on the index track. This write does not occur if an error is detected during the read of the label or if the data set type is read only.
5. The data set is marked closed within the system.

Closing Data Errors: If a data set cannot be closed, a 600 series error, along with the data set number, is posted. The error can be reset by the following:

- The RESET key, which retries the close sequence (except for a drive not ready error – 6X0).
- ALPHA SHIFT and NUM SHIFT and the RESET key, which aborts the job and posts a 100 error (job complete).

Note that the file EOD is always posted on the display on a drive 1 or 2 error, if the drive is found in a not-ready state, or a disk error occurs.

Printer Operations

Form size, type of printer attached, and printer output buffer are specified in the .PRINTER control statement. Editing is controlled by specifications in the .FORMAT control statement. The primary printer control instruction is the PRNT instruction.

Each time a new program is initiated, the paper must be manually set in the printer to the top of the page because the internal line counter is set to one at the start of each program.

Print a Line (PRNT)

This instruction specifies requirements for printer output. The instruction format is:

Column	1	8	13	18	23	28
Entry	Step/ Label	PRNT	Forms Control	Format	Buffer	Overlap

Column	Contents
1	*Step/label* – is the number (0-767) or symbolic label (four-position) assigned to this instruction.
8	*PRNT* – is the print instruction name.
13	*Forms control* – is the vertical forms control to be executed after a line is printed. Valid entries are: 0 – No vertical forms movement. For the 3715 single-direction printer (type 3), a carriage return command is issued. S or blank – Single space D – Double space T – Triple space 1-127 – Skipping to a specific line number
18	*Format* – is the format number for formatting and editing data out of registers to the buffer assigned in the .PRINTER control statement.
23	*Buffer* – is the buffer which is to have its contents moved to the output buffer assigned to the printer.
28	*Overlap* – if X is entered, successive instructions are executed concurrently with previously issued printer output. If column 28 is blank, further execution of any ACL instruction does not proceed until the printer cycle is completed.

The following operations occur when the PRNT instruction is executed:

1. The contents of the buffer specified in column 23 are moved to the buffer assigned to the printer. If column 23 is blank, the printer buffer is also blanked (hex 40). If the buffer in column 23 is the same as the buffer specified in the .PRINTER statement, its contents are not changed.

2. Data is moved from registers to the printer buffer as specified by the format number in column 18. If column 18 is blank, no formatting occurs.

3. The contents of the printer buffer are output to the printer. If column 28 of the PRNT instruction contains X, printer operations are overlapped with all other processing.

   If line length exceeds 128 characters, both the primary (.PRINTER columns 33-34) and secondary (.PRINTER columns 38-39) buffers must be odd. The characters beyond 128 are then output from the next even buffer. Because the 3715 (type 4) printer prints two lines at a time, the physical print occurs after every other PRNT instruction. This allows formatting of lines printed in both directions in order to look ahead for the longest line.

   If a 3715 application requires immediate output, use a printer type 3 or 5.

   If a printer type 4 is used, a PRNT instruction containing the desired immediate output, followed by a dummy PRNT, (PRNT with format parameter and buffer number blank) is required.

   Normal program termination closes the printer, although pending lines are printed before the system close message (100) is posted.

4. The vertical forms control (column 13) is executed after the last character is printed.

   Note: Normal nonprintable characters print (on the 3713 and 3715) as graphic characters (for example, hexadecimal FF prints as '). This differs from printing in standard data station mode on the 3741 Models 1 and 2.

If the forms control stops at or between the end of forms line and overflow line, as defined in the .PRINTER control statement, indicator 148 is set on. Also, in this instance, a branch is made to the page overflow processing step number/label (if specified in columns 43-46 of the .PRINTER control statement).

Skip to Line Number or Space (PCTL)

This instruction executed vertical forms control independent of actual printing, but always overlaps with other executable instructions. The PCTL instruction format is:

Column	1	8	13
Entry	Step/Label	PCTL	Forms Control

Column	Contents
1	*Step/label* – is the number or symbolic label (four-position) assigned to this instruction.
8	*PCTL* – is the print instruction name.
13	*Forms control* – is the vertical forms control with the same valid entries as the PRNT instruction.

This instruction does not output data to the printer, and does not change any buffers or registers. Indicator 148 and page overflow are processed as defined in the PRNT instruction.

Table Operations

A table is a group of successive fields of the same length. The content of any one field is called the argument, and the placement or location of the field in the group of fields is called the index.

For example, a list of employee numbers can be processed as a table. The table argument refers to an employee number in the list and the table index refers to a location in the list. The argument length is fixed at the length of the employee number.

When each table argument is greater (from a collating sequence standpoint) than the argument with the next lower index, it is said to be an ordered table.

The total length of a table is limited by available work station storage and each table must end in a hexadecimal FF. Tables can be loaded or read from disk into buffers or they can be created using .BUFFER control statements. There may be more than one table in a buffer as shown in the following example:

Buffer 6:

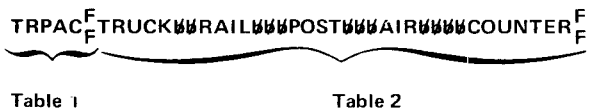


Table instructions are as follows.

Search Table for Equal Entry (TBFX)

The format of a TBFX instruction is:

Column	1	8	13	18	23	28
Entry	Step/ Label	TBFX	Buffer	Table	Register	Length

Column	Contents
1	*Step/label* — is the number or symbolic label assigned to this instruction.
8	*TBFX* — is the table instruction (find equal table entry).
13	*Buffer* — is the number of the buffer in which the table starts.
18	*Table* — is the number of the table in the buffer specified in column 13. The buffer specified and all succeeding buffers are scanned for a hexadecimal FF delimiter until the correct table number is found.
23	*Register* — specifies a pair of sequential registers (such as A, B; W, X; X,Y). The register specified in this operand is always the first of the pair and will contain the table index. The second (implied) register contains the table argument. Registers I, R, and Z cannot be specified in this operand.
28	*Length* — specifies the argument length. This instruction is used when the table argument is known and you want to find the table index of that argument (Figure 23). The instruction can be used on ordered or non-ordered tables because the entire table is scanned for an exact compare. If an equal entry is not found, the program can notify the operator by posting an invalid number message on the display screen.

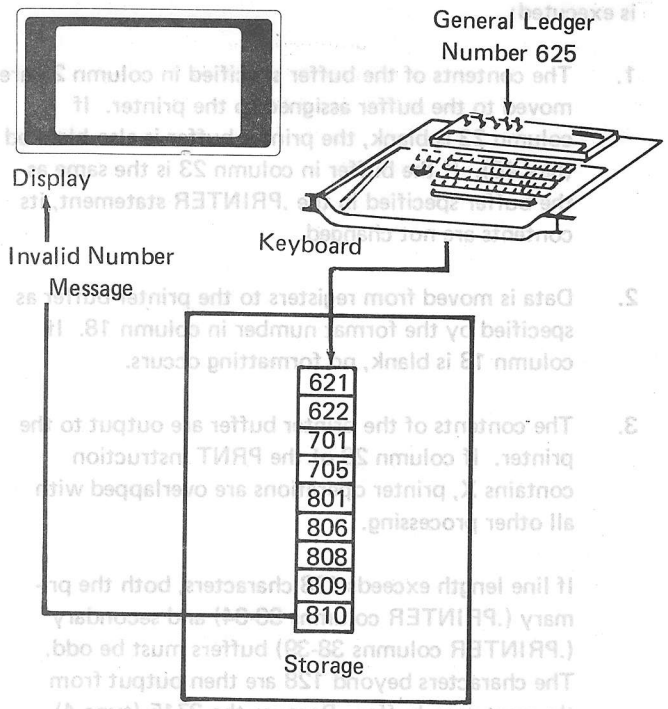


Figure 23. Table Search for Equal Entry

The number of bytes specified by the length is taken from the low-order bytes of the argument register, and the table is scanned looking for an equal entry. If found, it sets the four low-order positions of the index register to the relative number of the table entry (first entry in the table is number 1). If no equal entry is found, the index register is set to zero. Only the index register is changed.

Search Table for Equal/High Entry (TBFN)

The format of a TBFN instruction is:

Column	1	8	13	18	23	28
Entry	Step/Label	TBFN	Buffer	Table	Register	Length

Column	Contents
8	*TBFN* – is the table instruction (find equal/high table entry).
13	*Buffer* – is the number of the buffer in which the table starts.
18	*Table* – is the number of the table in the buffer specified in column 13. The buffer specified and all succeeding buffers are scanned for a hexadecimal FF delimiter until the correct table number is found.
23	*Register* – specifies a pair of sequential registers (such as A, B; W, X; X, Y). The register specified in this operand is always the first of the pair and will contain the table index. The second (implied) register contains the table argument. Registers I, R, and Z cannot be specified in this operand.
28	*Length* – specifies the argument length.

This instruction is used when a table argument is known and an equal or high index entry is to be found. The table, which must be in ascending sequence, is scanned sequentially looking for an equal or high entry. If the entry found is high, indicator 163 is set on (and must be reset by the program), and the index entry number of that higher argument is put in the index register. If the argument is higher than the last entry in the table, indicator 163 is not set on, and the index register is set to zero.

If an equal entry is found, this instruction sets the four low-order positions of the index register to the entry number of the equal entry. The normal EBCDIC collating sequence must be used. Indicator 163 is not set on and the index register is not set to zero.

Read Table Entry (TBRD)

The format of a TBRD instruction is:

Column	1	8	13	18	23	28
Entry	Step/Label	TBRD	Buffer	Table	Register	Length

Column	Contents
1	*Step/label* – is the number or symbolic label assigned to this instruction.
8	*TBRD* – is the table instruction (read table entry).
13	*Buffer* – is the number of the buffer in which the table starts.
18	*Table* – is the number of the table in the buffer specified in column 13. The buffer specified and all succeeding buffers are scanned for a hexadecimal FF delimiter until the correct table number is found.
23	*Register* – specifies a pair of sequential registers (such as A, B; W, X; X, Y). The register specified in this operand is always the first of the pair and contains the table index. The sign, if any, is ignored. An index of zero does not change the argument register and sets on indicators 156 and 160. The second (implied) register contains the table argument at the end of the operation. Registers I, R, and Z cannot be specified in this operand.
28	*Length* – specifies the argument length.

This instruction is used when the table entry is known and the table argument is to be put into a register. The instruction can be used on ordered or nonordered tables.

Using the low-order four bytes of the index register, the instruction takes the argument at that index number and puts it into the low-order positions of the argument register. The unused high-order positions are set to blanks. Indicators 156 and 160 are set on if the table read goes beyond the table end. These indicators must be reset by the program.

Write Table Entry (TBWT)

The format of a TBWT instruction is:

Column	1	8	13	18	23	28
Entry	Step/Label	TBWT	Buffer	Table	Register	Length

Column	Contents
1	*Step/label* – is the number or symbolic label assigned to this instruction.
8	*TBWT* – is the table instruction (write table entry).
13	*Buffer* – is the number of the buffer in which the table starts.
18	*Table* – is the number of the table in the buffer specified in column 13. The buffer specified and all succeeding buffers are scanned for a hexadecimal FF delimiter until the correct table number is found.
23	*Register* – specifies a pair of sequential registers (such as A, B; W, X; Y, Z). The register specified in this operand is always the first of the pair and contains the table index. The sign, if any, is ignored. An index of zero does not change the argument register and sets on indicators 156 and 160. The second (implied) register contains the table argument to be written. Registers I, R, and Z cannot be specified in this operand.
28	*Length* – specifies the argument length.

This instruction is used when the index and the argument are known and the argument is to be written at the index. The argument must be in the low-order bytes of the register. No registers are changed. When using this instruction in an ordered table, be careful that the existing ascending sequence is not destroyed. Indicators 156 and 160 are set on if a write operation goes beyond the end of the table. These indicators must be reset by the program.

Move Data from Buffer to Register (GETB)

This is a table-type instruction which indexes through a buffer and extracts specific entries. The format of a GETB instruction is:

Column	1	8	13	18	23	28
Entry	Step/Label	GETB	Buffer	Table	Register	Length

Column	Contents
1	*Step/label* – is the number or symbolic label assigned to this instruction.
8	*GETB* – is the table instruction (move data from buffer to register).
13	*Buffer* – is the number of the buffer in which the table starts.
18	*Table* – must be a table number (1-16).
23	*Register* – specifies a pair of sequential registers (such as A, B; W, X; X, Y). The register specified in this operand is always the first of the pair and contains the byte position from the leftmost byte in the table. The second register contains the data moved from the buffer. Registers I, R, and Z cannot be specified in this operand.
28	*Length* – specifies the number of bytes to be moved.

This instruction is a special form of the read table (TBRD) instruction and is used to read a specific field from buffer into the argument register. The table index register is used, not as a field position, but as a byte position from the leftmost byte of the character string to be moved. The number of bytes moved is specified by the length operand. The unused high-order bytes of the argument register are set to blanks. Only the argument register is changed.

This instruction does not set any indicators, and ignores the hexadecimal FF delimiter once the correct table has been found.

Move Data from Register to Buffer (PUTB)

The format of a PUTB instruction is:

Column	1	8	13	18	23	28
Entry	Step/Label	PUTB	Buffer	Table	Register	Length

Column	Contents
1	*Step/label* – is the number or symbolic label assigned to this instruction.
8	*PUTB* – is the table instruction (move data from register to buffer).
13	*Buffer* – is the number of the buffer in which the table starts.
18	*Table* – must be a table number (1-16).
23	*Register* – specifies a pair of sequential registers (such as A, B; W, X; X, Y). The register specified in this operand is always the first of the pair and contains the byte position from the leftmost byte in the buffer. The second register contains the data to be moved to the buffer. Registers I, R, and Z cannot be specified in this operand.
28	*Length* – specifies the number of bytes to be moved.

This instruction is a special form of the write table (TBWT) instruction. It is used to write a specific field into buffer. The last four bytes of the table index register are used, not as a field position, but as a byte position from the leftmost byte of the start buffer. This byte position must point to the leftmost byte where the bytes are to be moved. Data movement is from left to right into the low-order positions of the argument register. The number of bytes moved is specified by the length operand. No registers are changed.

This instruction sets no indicators, and ignores the hexadecimal FF delimiter once the correct table has been found.

Internal Data Movement Operations

Internal data movement instructions move and exchange the contents of registers and buffers to other registers and buffers, read from buffer, and write to buffer.

Read from Buffer (REFM)

The REFM instruction internally reads from buffer and formats into registers. This instruction is typically used to read data from the display screen (buffer 2). The format of an REFM instruction is:

Column	1	8	13	18
Entry	Step/Label	REFM	Buff	Format

Column	Contents
1	*Step/label* – is the number or symbolic label (four-position) assigned to this instruction.
8	*REFM* – identifies the operation to be performed (read from buffer).
13	*Buffer* – identifies the buffer containing the data to be read and formatted.
18	*Format* – indicates the format to be used. Formatting is from buffer to registers. This instruction causes no physical I/O (except for the display screen). If a D is specified for this required operand, formatting is data-directed (see *Data Directed Formatting* in this chapter).

A sample REFM instruction is:

Column	1	8	13	18
Entry	Step/Label	REFM	2	9

In this example, the REFM instruction uses format number 9 to format data into registers from buffer 2 (display screen lines 2, 3, and 4).

Read Blocked Record from Buffer (RBLK)

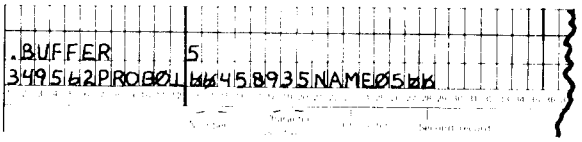
This instruction accesses individual records within a block of records which contains records of less than 128 bytes. The format of an RBLK instruction is:

Column	1	8	13	18	23
Entry	Step/Label	RBLK	Buffer	Format	Register

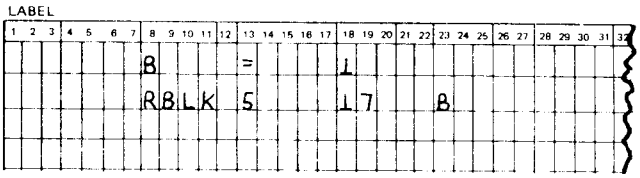
Column	Contents
1	*Step/label* – is the number or symbolic label assigned to this instruction.
8	*RBLK* – identifies the operation to be performed (read blocked record from buffer).
13	*Buffer* – identifies the number of the buffer containing the data to be formatted. This entry must be coded.
18	*Format* – identifies the number (1-254) of the format to be used. This is a required entry. Formatting is from buffer to registers. Only the data starting at the position specified in the register entry (column 23) is formatted. Format definition, however, is taken from the start of the format record. The format must define formatting for the first logical record in the buffer. This instruction causes no physical disk reading. Formatting is data-directed if D is entered in column 18. (See *Data Directed Formatting* in this chapter.)
23	*Register* – identifies the register containing the buffer position (first from the left) at which formatting is to start within the buffer. This is a required entry.

Some sample RBLK instructions are:

Assume the following control statement entries:



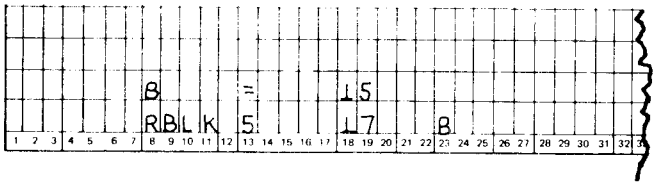
Also assume the following instructions:



Assuming these conditions, the result would be as follows.

Register A would contain – 349562
Register C would contain – PROB
Register D would contain – 01

Then, assuming the following instructions are executed:



The result would be:

Register A contains – 458935
Register C contains – NAME
Register D contains – 05

Write to Buffer (WRFM)

This instruction writes and formats data into buffer. The format of a WRFM instruction is:

Column	1	8	13	18	23
Entry	Step/Label	WRFM	Buffer One	Format No.	Buffer Two

Column	Contents
1	*Step/label* — is the number or symbolic label (four-position) assigned to this instruction.
8	*WRFM* — identifies the operation to be performed (write to buffer).
13	*Buffer one* — identifies the buffer that receives the contents of the buffer specified in column 23 of this instruction. The buffer does not have to be associated with a .DATASET control statement.
18	*Format no.* — identifies the number of the format to be used. Formatting is from registers to buffer. If this entry is not coded, no formatting occurs.
23	*Buffer two* — the contents of this buffer are moved into the buffer specified in column 13 of this instruction before formatting. The contents are unchanged. If this entry is not coded, the buffer specified in column 13 is blanked before formatting. If the same buffer is specified in columns 13 and 23, the contents of buffer one (column 13) are not changed.

A sample WRFM instruction is:

Column	1	8	13	18	23
Entry	Step/Label	WRFM	2	7	4

In this example, the contents of buffer 4 are moved into buffer 2 (lines 2, 3, and 4 of the display). Format 7 is then used to format register data into buffer 2. The contents of buffer 4 are not changed.

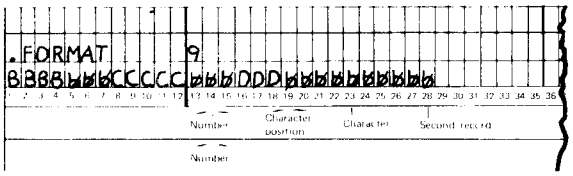
Write Blocked Record to Buffer (WBLK)

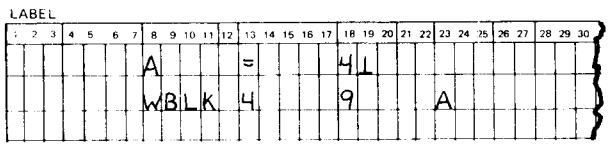
This instruction writes and formats blocked records into buffer. The format of a WBLK instruction is:

Column	1	8	13	18	23
Entry	Step/Label	WBLK	Buffer	Format	Register

Column	Contents
1	*Step/label* — is the number or symbolic label (four-position) assigned to this instruction.
8	*WBLK* — identifies the operation to be performed (write blocked record to buffer).
13	*Buffer* — identifies the buffer that will be partially formatted each time this instruction is executed. This operand must be coded, although no physical disk writing takes place.
18	*Format* — identifies the number of the format to be used when formatting from registers to buffer. If this operand is not coded, an error is flagged by the translator.
23	*Register* — identifies the register containing the location (offset) from the first (leftmost) position of the buffer. This is a required entry. Execution is similar to that for a RBLK instruction, except that data movement is from registers to buffer.

Assume that buffer 4 contains three blocked, logical records, each record being 40 bytes long. Also assume that format number 9 specifies data to be moved from registers to the logical records. The following sequence will format the registers specified in format 9 into the second logical record contained in buffer 4.





Exchange Buffer Contents (EXCH)

This instruction exchanges the contents of two separate buffers. The number of the buffers to be exchanged must be entered. A sample EXCH instruction is:

Column	1	8	13	23
Entry	Step/Label	EXCH	2	8

In this example, the contents of buffer 2 will be the previous contents of buffer 8, and vice versa.

Move Data from Buffer to Buffer (MOVE)

This instruction moves the contents of one buffer into another buffer. This is done by simply entering both buffer numbers. The first buffer specified receives the contents of the second buffer specified. A sample MOVE instruction is:

Column	1	8	13	23
Entry	Step/Label	MOVE	1	4

In this example, the contents of buffer 4 are moved into buffer 1. The contents of buffer 4 are not changed.

Move Partial Content from Register to Register (MVER)

This instruction moves part of the contents of a register into corresponding positions in another register. The format of a MVER instruction is:

Column	1	8	13	18	23	28
Entry	Step/ Label	MVER	To Register	From Register	Byte Positions	Length

Column	Contents
1	*Step/label* – is the number or symbolic label assigned to this instruction.
8	*MVER* – identifies the operation to be performed (move specified register contents to register indicated).
13	*To register* – identifies the register to which data should be moved.
18	*From register* – identifies the register from which data is to be moved.
23	*Byte position* – specifies the leftmost byte position of both the to and the from registers.
28	*Length* – indicates the number (1-16) of characters to be moved.

Upon execution of this instruction, characters are moved from the register specified in column 18 to the register specified in column 13. Character positions not referenced are unchanged. Data is moved from left to right starting at the position specified in column 23. The number of characters moved is specified in column 28. Thus, data moved from the column 18 register is placed in equivalent positions in the column 13 register. No indicators are set by this instruction. A sample MVER instruction is:

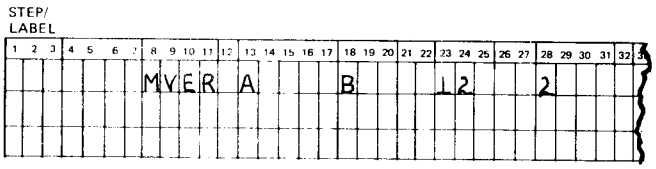
Register A contents:

F F F
1 1 1
1 14 15 16

Register B contents:

F F F F F
6 5 4 3 2
1 12 13 14 15 16

The following MVER instruction produces:



Register A contents:

F F F F F
6 5 1 1 1
1 12 13 14 15 16

Register B contents: Unchanged

Move Partial Content to Register with Offset (MOFF)

This instruction moves part of the contents of a register into specific locations in another register. The format of a MOFF instruction is:

Column	1	8	13	18	23	28
Entry	Step/ Label	MOFF	To Register	From Register	Byte Position	Length

Column	Contents
1	*Step/label* – is the number or symbolic label assigned to this instruction.
8	*MOFF* – identifies the operation to be performed (move specified data into the offset position of the register indicated).
13	*To register* – identifies the register to which data should be moved.
18	*From register* – identifies the register from which data should be moved.
23	*Byte position* – specifies the leftmost byte position of the register (column 13) into which the characters are to be moved.
28	*Length* – specifies the number of characters to be moved.

Upon execution of this instruction, the number of characters (column 28) counting from the low-order (rightmost) end of the from register (column 18) are moved to the to register (column 13). The characters start in the receiving register at the byte position (column 23), and continue for (column 28) positions to the right. Positions not referenced in the (column 13) register are unchanged. A sample MOFF instruction is:

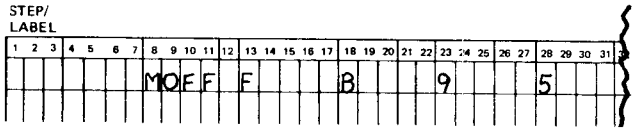
Register F contents:

```
                F  F  F
                3  2  1
1 ---------------14 15 16
```

Register B contents:

```
                F  F  F  F  F
                8  7  6  5  4
1 ---------------12 13 14 15 16
```

This MOFF instruction produces the following results:



Register F contents:

```
                F  F  F  F  F  F  F  F
                8  7  6  5  4  3  2  1
1 ---------- 9 10 11 12 13 14 15 16
```

Register B contents: Unchanged

Load Data Buffer to Register (LOAD)

This instruction is used to access a specific field in buffer. The buffer position of the field is specified in the instruction rather than through a register. The format of a LOAD instruction is:

Column	1	8	13	18	23	28
Entry	Step/Label	LOAD	Buffer	Buffer Offset	Register	Length

Column	Contents
1	*Step/label* – is the number or symbolic label assigned to this instruction.
8	*LOAD* – identifies the operation to be performed (load specified register with indicated data).
13	*Buffer* – identifies the number of the buffer containing the data.
18	*Buffer offset* – identifies the position in the buffer of the leftmost byte to be accessed.
23	*Register* – identifies the register to which data is to be moved.
28	*Length* – specifies the number of characters to be moved.

Upon execution of this instruction, the specified register is loaded with the number of bytes indicated by length from the address computed by the buffer and buffer offset entries (columns 13 and 18). The buffer and buffer offset address indicates the byte that loads into the byte position of the register specified by subtracting bytes (column 28) from 17. Data is loaded so that the units (rightmost) position of the register is loaded last. Only the specified register is changed. The high-order bytes in the register are set to blank (hex 40). A sample LOAD instruction is:

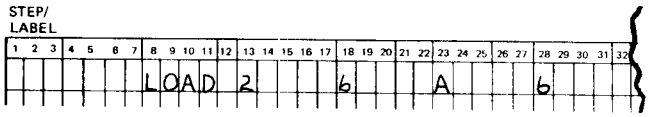
Buffer 2 contents:

```
44444CCCC F F
000005678 7 8
1_____11.....
```

Register A contents:

```
FFFFFFFFF F F F F F F F F
999999999 9 9 9 9 9 9 9 9
1_____16
```

This load instruction produces the following results:



Register A contents:

```
                C C C C F F
ØØØØØØØØØØ Ø 5 6 7 8 7 8
1_____11 12 13 14 15 16
```

Store Data Register to Buffer (STOR)

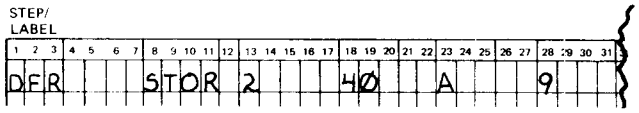
Execution of the STOR instruction is opposite to that of the LOAD instruction. The rightmost contents of the specified register are written to the specified buffer area. Register contents are unchanged.

The format of a STOR instruction is:

Column	1	8	13	18	23	28
Entry	Step Number or Label	STOR	Buffer	Buffer Offset	Register	**Length**

Column	Contents
1	*Step/label* – is the number or symbolic label assigned to this instruction.
8	*STOR* – identifies the operation to be performed (store specified buffer with indicated data).
13	*Buffer* – identifies the number of the buffer in which data is to be stored.
18	*Buffer offset* – identifies the position within the buffer that stored data should start.
23	*Register* – identifies the register from which data should be taken.
28	*Length* – identifies the number of byte positions to be moved from the register.

The following instruction would cause the nine low-order positions of register A to be stored in buffer 2 (display) starting at position 40.



Zone Bytes in Register (ZONE)

This instruction assigns zones to specified bytes in a register. The format of a ZONE instruction is:

Column	1	8	13	18	23	28
Entry	Step/ label	ZONE	Register	Zone	Byte Position	Length

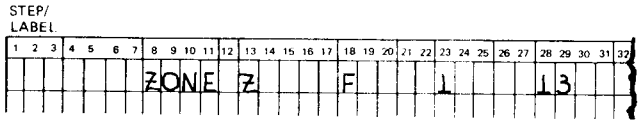
Column	Contents
1	*Step/label* — is the number or symbolic label assigned to this instruction.
8	*ZONE* — identifies the operation to be performed (zone bytes in a register).
13	*Register* — identifies the register address containing data to be zoned.
18	*Zone* — specifies immediate data indicating the zone to be forced (0-9, A-F).
23	*Byte position* — indicates the position of the bytes from the leftmost position of the register.
28	*Length* — indicates the number of bytes to be zoned.

Upon execution of this instruction, the bytes in the register specified in column 13, starting at the entry in column 23, and going to the right for the number of bytes specified in column 28, receive zones as specified by column 18. Only the register being zoned is changed.

Assume that register Z contains the following:

```
444444444 4 4 4 4 F F F
000000000 0 0 0 0 3 2 1
1234 56 789 10 11 12 13 14 15 16
```

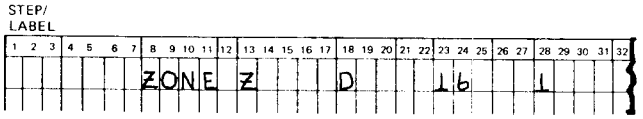
Sample ZONE instruction 1:



Resulting register Z contents:

```
FFFFFFFFF F F F F F F F
000000000 0 0 0 0 3 2 1
1234 56 789 10 11 12 13 14 15 16
```

Sample ZONE instruction 2:



Resulting register Z contents:

```
FFFFFFFFF F F F F F F D
000000000 0 0 0 0 3 2 1
1234 56 789 10 11 12 13 14 15 16
```

Miscellaneous Instructions

The following instructions have important uses in the ACL program, as shown below.

Set Indicators On (SON)

This instruction selects and turns on certain indicators. The format of a SON instruction is:

Column	1	8	13	18	23
Entry	Step/Label	SON	NNN	NNN	NNN

where NNN = an indicator (1-255) or blank for no indicator.

Upon execution of this instruction, the selected indicators are turned on. No registers or buffers are changed. Note that indicator 161 turns on the flashing display screen while indicator 162 gives a short audible keyboard buzz.

The ACL instruction, SON 161, turns on indicator 161 and turns off indicator 192 (RESET key indicator). The instruction, SON 161 192, turns on both indicators 161 and 192. The instruction, SON 192 161, turns on only indicator 161. (After turning on indicator 192, indicator 161 turns on, which then turns off indicator 192.)

Set Indicators Off (SOFF)

This instruction turns off selected indicators. The instruction has the same format as the SON instruction. Upon execution of this instruction, however, the selected indicators are turned off. All indicators, except for the three switches (AUTO REC ADV, PGM NUM SHIFT, and AUTO DUP/SKIP), can be turned off with this instruction. Note that indicator 161 can be set off with this instruction to turn off the display error line (flashing screen). This indicator can be reset by pressing the RESET key. Avoid turning certain system indicators off or on (Appendix A). The keyboard indicator (197) can only be tested. This indicator does not lock or unlock the keyboard.

Checkpoint Statement (CKPT)

The CKPT statement, which interrupts an ACL program and provides a reentry point, has the following format:

Column	1	8	13	18
Entry	Step/Label	CKPT	Data set Number	Register

Column	Contents
1	*Step/label* – is the number or symbolic label assigned to this instruction. The CKPT cannot be at step number 255, 511, or 767.
8	*CKPT* – identifies the operation to be performed.
13	*Data set number* – is the number of the data set (with a .DATASET statement) that contains the executing ACL program as it exists at the interrupted point, or checkpoint. This data set must meet the requirements of an object data set.
18	*Register* – is the register that contains information about the checkpoint data set. The register must contain the following information:

Register Position	Contents
1-8	Data set name
9	Drive number (1 or 2)
10-12	Data set access method (SW or SWE)
13	Restart point (C or E)
15-16	Checkpoint ID no. (any EBCDIC character)

Data set name – is the name of the data set to be used to store the checkpoint records. This is a required field.

Drive number – is the number of the disk drive containing the checkpoint data set. This entry must be 1 or 2. This is a required field.

Data set type – must be SW or SWE.

Restart point – a C indicates the program is to continue from the next sequential statement after completion of the request. Note that for the SWE access method, the current checkpoint record does not overlay the previous checkpoint records. An E or any other character other than a C indicates that the program is to be terminated and a system close be attempted after the checkpoint (see *Program Restart* in Chapter 4).

Checkpoint ID – a two-character field used to identify the checkpoint. These two characters overlay the last two characters of the program name on the display screen, and can be used to identify the current checkpoint.

The CKPT statement saves the status of the machine at the execution time of the statement. This allows a restart from this point in the program.

The checkpoint data set must not be open when the CKPT instruction is executed. If it is open, a 9X1 error is posted. The CKPT instruction opens the checkpoint data set, and closes it after the checkpoint is completed (see *Program Restart* in Chapter 4).

Insert Character in Buffer (ICBF)

This instruction generates a character of code (hexadecimal or keyboard character) and inserts that character in a specific position in a buffer or register. This can be used for editing a print line (inserting slashes in a date, for example) with characters not otherwise available. The format of an ICBF instruction is:

Column	1	8	13	18	23
Entry	Step/Label	ICBF	Buffer/ Register	Position	Character

Column	Contents
8	*ICBF* – identifies the operation to be performed.
13	*Buffer/register* – identifies the number of the buffer/register that is to receive the character.
18	*Position* – indicates the position within the buffer at which the character should be entered.
23	*Character* – identifies the character of immediate data that is inserted in the buffer/register.

Generate Self-Check Number (GSCK)

This instruction creates a self-check number when, for example, a master file of customer numbers is created. This eliminates calculation of a self-check number. The format of a GSCK instruction is:

Column	1	8	13	18	23
Entry	Step/Label	GSCK	Register		

Column	Contents
8	*GSCK* – identifies the operation to be performed.
13	*Register* – identifies the register (A-Z) containing the data which is used to compute the self-check number. The self-check number is computed from this data as specified by the .SELF-CHECK control statement. The .SELF-CHECK control statement also specifies the placement of the self-check number in the register.

No Operation (NOP)

The NOP instruction reserves a program step/label for later use. The instruction causes no operation and does not change any indicators or registers. The instruction is initiated by entering NOP in column 8 on the coding sheet.

Execute Program Chain (EXEC)

This instruction closes all data sets and waits for all printer operations to complete (system close), then chains directly from the end of one job to translation or execution of the next program. The format of the EXEC instruction is:

Column	1	8	13
Entry	Step/Label	EXEC	A or E

Column	Contents
1	*Step/label* – is the number or symbolic label assigned to this instruction.
8	*EXEC* – is the name of this instruction.
13	Type of execution to be performed. An entry of A indicates that the next program should be translated. An entry of E indicates that the next program should be executed.

If an A is entered in column 13, this instruction can be preceded by a MOVE instruction, which moves the input (source) data set name (as it appears on the data set label) and the output (object) data set name into buffer 2. (See Chapter 4 for information about the Translator feature.) This information can also be read from a data set stored in buffer 2. The EXEC instruction then reads this data from the display screen and initiates translation.

If an E is entered in column 13, this instruction can be preceded by a MOVE instruction, which moves the output (object) data set name and the program name into buffer 2. This information could also be read from a data set stored in buffer 2. The EXEC instruction reads this data from the display screen and initiates execution of the program. This option can also be used to execute label processing.

Assigning a Step Number (ORG)

You can assign any valid step number to the next sequential instruction by using the ORG function. The format of this function is:

Column	1	8	13-15
Entry		ORG	Step Number Assigned

When the ORG is executed, the instruction following the ORG is assigned the step number specified in columns 13-15. If these columns are blank, step number 000 is assigned. In the object data set, space created by the ORG function is filled with hexadecimal FF. The space is considered unused. Because the ORG is used, then deleted by the label processor, it is ignored by the translator. ORG may not be used if the label processor step is bypassed (the translator will post an invalid instruction error). The ORG function is useful in moving the object code in storage, or in program overlays. Note that the step number entry must be right-justified.

Read a Card (CRDR)

The CRDR instruction reads and transfers data from cards in the IBM 5496 Data Recorder or the IBM 129 Card Data Recorder. The transferred data can be loaded into a specified buffer, and then formatted. The CRDR instruction is:

Column	1	8	13	18	28
Entry	Step/Label	CRDR	Buffer	Format	Operation

Column	Contents
1	*Step/label* – is the number or symbolic label assigned to this instruction.
8	*CRDR* – identifies the operation to be performed (read a card).
13	*Buffer* – identifies the buffer into which the data read from the card is to be loaded. This is a required entry.
18	*Format* – identifies the number of the format which is to control formatting of the data after it is loaded into the buffer. A blank defaults to data-directed formatting.
28	*Operation* – identifies the sequence of the operation. Valid entries are:

- S – specifies that the read be started, but no data be transferred to the work station.
- X – specifies that data read by the 5496 or 129 be transferred to the specified buffer.
- Blank – specifies that data be read by the 5496 or 129 and concurrently loaded into the specified buffer.

Upon execution of this instruction, data is read from cards by the 129 or 5496, transferred into the specified buffer, and then form .ed according to the format specified in column 18. The S in column 28 allows data to be read from the card, but not transferred until the entire read is completed. Data is then transferred by use of the X entry in column 28. The S parameter can be used in conjunction with the IF CRD IS/NOT BSY instruction to loop on this operation until cards are completely read. Because the reading is fully overlapped with the attached printer, this eliminates the printer speed from being limited by reading time.

Punch a Card (CRDP)

This instruction transfers data from the work station to the IBM 129 Card Data Recorder or the IBM 5496 Data Recorder. The receiving machine then punches that data onto 80 or 96-column cards. The format of the CRDP instruction is:

Column	1	8	13	18	23
Entry	Step/ Label	CRDP	Buffer	Format	Buffer

Column	Content
1	*Step/label* – is the number or symbolic label assigned to this instruction.
8	*CRDP* – identifies the operation to be performed (punch a card).
13	*Buffer* – identifies the buffer from which data is transferred to the 129 or 5496. This is a required entry.
18	*Format* – identifies the number of the format which controls formatting of data into the buffer specified in columns 13-14.
23	*Buffer* – identifies the buffer which can contain additional data to be transferred to the 129 or 5496. This data is first loaded into the required buffer (columns 13-14), formatted according to the format in columns 18-20, then transferred to the 129 or 5496. If columns 23-24 are blank, the buffer specified in columns 13-14 is first blanked, then formatted.

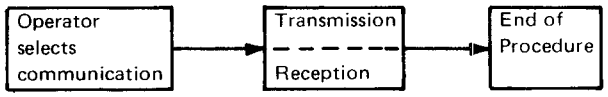
COMMUNICATIONS

The 3741 Model 4 Programmable Work Station has all of the functions available through ACL (application control language), plus the communications facility currently available on the 3741 Model 2.

Communication capabilities of the Model 2 and Model 4 are described in Chapter 10 of the *IBM 3741 Data Station Reference Manual,* GA21-9183. The access method facilities and macro instructions required to write an application program that defines, activates, and controls the 3741 Models 2 and 4 are described in the *IBM 3740 BTAM/TCAM Programmer's Guide,* GC21-5071.

Binary Synchronous Communication

The binary synchronous communications adapter allows the 3741 to function as a point-to-point terminal. The basic BSCA function allows data transmission using EBCDIC directly as the communication line code. Operation is half-duplex over a private line, a common carrier leased line, or a common carrier switched network. The mode of transmission is synchronous, serial-by-bit, and serial-by-character.



The operator selects BSCA communications from the index and update modes only. The following procedure establishes a data link and communication begins:

1. Load the diskette.
2. Position the diskette to the data set label or record where communication is to start.
3. Press FUNCT SEL upper and COMM.
4. Press the appropriate mode key.
5. Press DATA on the modem.

The 3741 disconnects from the line when communication, whether transmission or reception, is complete.

Expanded Communications Feature

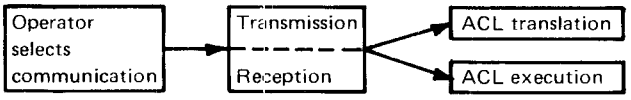
The Expanded Communications feature provides the following additional functions for the 3741 Models 2 and 4:

- Expanded buffer (512 bytes)
- Transmit selected field
- Transmit selected records
- Receive and insert constants and blanks
- Unattended printing after completion of communication
- Unattended ACL program mode after completion of communication (Model 4 only)

Unattended ACL program mode is explained here because it relates directly to the 3741 Model 4.

Unattended ACL Program Mode after Communications

The information required to translate or execute an ACL program is put in track 0, sector 3 of disk 1. When communication is complete, the information in sector 3 is shifted one position to the left and displayed.



Disk 1, with the load parameters properly recorded, must always be mounted before establishing the mode. The object data set, containing the Model 4 program, must be labeled and mounted on the drive as specified by the parameters recorded in track 0, sector 3 of the disk in drive 1.

The information required for translation is:

Position	Required Information
2-9	Source data set name.
10	1 or 2, depending on the disk drive the source data set is mounted on. Default is 1.
12-19	Object data set name.
20	1 or 2, depending on the disk drive the object data set is mounted on. Default is 1.
21	The character A.

The information required for execution is:

Position	Required Information
2-9	Object data set name.
10	1 or 2, depending on the disk drive the object data set is mounted on. Default is 1.
12-15	ACL program name.
21	The character E.

The operator can set up the unattended ACL program mode by keying:

1. FUNCT SEL upper and COMM
2. The character E
3. The appropriate mode key

Communication Mode from an ACL Program

Operator intervention can be reduced further when the Communication Link (RPQ) feature is installed on a 3741 with the Expanded Communications feature.

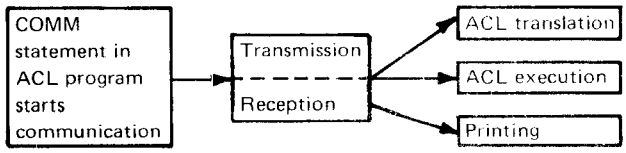
The purpose of the Communication Link (RPQ) feature is to set up the communication mode parameters under control of an ACL program, rather than having an operator key the setup sequence manually.

The communication mode and additional parameters are entered in ACL registers A and B by the program. To transmit or receive blocked records or to print under format control requires that proper masks or formats be set up in ACL buffers 8, 9, and 10.

The linkage to communications begins when the COMM instruction is encountered in an ACL program.

Column	1	8	13	18	23
Entry	Step/ Label	COMM			

The 3741 does extended validity checking of input parameters. The communication function takes control at the conclusion of setup checking by transferring registers A and B to Model 2 registers and the ACL buffers 8, 9, and 10 to Model 2 program levels 8, 9, and A. Then, communications takes place in the same manner as with a Model 2 with the Expanded Communications feature.



The object data set must precede the received data sets during receive mode to avoid overwriting of the object data set.

Register A must contain the following information:

Register Position	Contents
1	The character M, if the call is placed at the 3741. Any other character defaults to auto-answer mode. (This position is not used on nonswitched lines.)
2	The character P, if upon completion of communications, the first data set received is to be printed. The character E, if upon completion of communications, an ACL program is to be loaded and executed. Any other character causes the 3741 to go to communications complete mode (upon the completion of communications) requiring RESET to return to index (X) mode.
3	The communications mode character (T, J, P, R, B, K, or D).
4	Format numbers 2 through 9. This position is used only if position 2 contains a P and buffer 10, position 1 contains X'7A' (:).
5	If a read or a write to file 1 has been performed and the character N is coded in this position, no multivolume indicator (continuation) check will be made and EBSCA will assume that the entire file resides on disk 1. If a read or a write has not been performed or if a blank is coded in this position, the multivolume indicator (continuation) check will be made.
6-16	Not used.

Register B must contain the following information if the terminal ID feature is installed:

Register Position	Contents
1-15	Remote ID.
16	Remote ID length (hex) even if the length is zero (no remote ID being sent).

Data set 1 must have the following characteristics:

- Open and loaded on disk drive 1
- SW, SWE, or SU type file for receive (R) and inquiry (I) modes
- KU, KR, or SR type file for transmit (T, B, P, D, J, K) modes

Data sets 2, 3, and 4 *must* be closed when using the communications linkage function.

The mode selected determines whether the first record is written to disk or transmitted.

The first record is written to disk in receive mode as follows:

- At the EOD address if data set 1 is an SWE type file.
- At the data set 1 disk address plus one if data set 1 is an SU or SW type file. If no activity was generated against the file, the first record is written at the BOE address.

When a read or write instruction is executed before the COMM instruction, the received records must have the same record length as the existing records in the file. The first record is transmitted in transmit mode as follows:

- Null record (STX ETX) if data set 1 is a null data set (BOE = EOD).
- BOE record if no activity was generated against data set 1.
- Record at the data set 1 disk address.

CONSIDERATIONS FOR EFFICIENT KEY ENTRY PROGRAMS

The work station coupled with ACL can be an efficient data entry tool. However, the efficiency of the data entry job for the key entry operator is dependent on the programmer. Paramount in all key entry programs is adequate error correction facilities for the operator. A discussion and coded examples of error correction routines are provided later in this chapter. Additional guidelines for writing efficient key entry programs are as follows.

Alpha/Numeric Fields

Because it is easier to manually shift the keyboard for numeric characters than for alpha characters, mixed alpha/numeric fields should be defined as A in position 23 of the .FIELD; unless there is only a very occasional alpha character in a field, then U in position 23 should be used.

Error Correction

Operators detect approximately 80 percent of their keying errors, so immediate operator correction of previously entered fields should be constantly provided using the T in column 36 of the .FIELD statement (special keyboard close).

Prompting Messages

Prompting messages issued to the operator should be spelled exactly like the captions on the source document. Prompting messages should also appear in the same sequence as listed on the source document to facilitate the ease of keying. ACL allows you to reformat the fields to meet the requirements of the output record format. After fields are entered, they should be displayed on lines 2, 3, and 4 of the display screen, and should be separated by spaces rather than displayed in packed diskette output record format, which is difficult to read.

Restart

Because operators are frequently interrupted during their work, the ACL program should indicate to the operator the point at which processing was stopped. You should consider using the checkpoint/restart capabilities of ACL. The CKPT instruction is discussed in Chapter 2, and the program restart procedure is discussed in Chapter 4. Output data sets are automatically positioned at the interrupted point upon restart. See *.DATASET* in Chapter 2, and note the SW and SWE access methods.

Guidance Techniques

In many applications the operator keys directly from a source document; that is, the operator performs a data transcription function. The operator's eyes are on the documentation and not following the display screen. The ACL provides two tools to guide the operator in this environment.

The first of these tools is the display screen flash. The flashing screen is turned on by setting on indicator 161 in the ACL program. This feature is normally used when the ACL program determines an error condition (for example, keyed data is outside a range check). Besides flashing the screen, this indicator locks out the keyboard and provides an auditory signal (no key click), so the operator can recognize the error even if the operator does not look at the screen.

The second tool is the buzzer. Keying throughput can be increased by setting on indicator 162 (buzzer) to give a ready buzz when the operator must wait for the machine (for example, searching a file to validate an account number). On the other hand, if the operator should be looking at the screen for visual verification, then the ready buzz should not be used (for example, searching a file on customer number for the customer name and address and wanting the operator to visually verify that the address is correct).

Keying Pattern

The keying pattern of the job should be thoroughly studied. If there is a fixed-length field in a logical series of variable length fields, an exit key such as RIGHT ADJ should be required (position 29 in .FIELD control statement) for all of the fields so the operator does not have to learn the exceptional field.

Variable-length fields which are seldom filled should require an exit key so the operator does not have to be concerned if the field is full and erroneously exit the next field.

Do not control branching in the ACL program with extra keystrokes by calling for extraneous yes- and no-type messages rather than controlling branch key blank field on the (SKIP, RIGHT ADJ) exit keys.

The keying pattern for the job should have a logical flow. The capability of the work station to allow fields to be entered in a sequence convenient for the operator and output on the diskette or printer in a different arrangement to meet other data processing requirements should be utilized.

Function Keys

The work station installation may be operated in both data station mode (Models 1 and 2) and ACL program mode (Models 3 and 4). Thus, the ACL programs should simulate the data station function keys (for example, REC ADV). This also provides consistency among the various ACL programs to be run by the same operator.

Key	Recommended Function
FIELD BKSP	Go back to the previously entered field.
REC BKSP	Go back to the beginning of the previously entered group of fields. The beginning depends on the format of the source document and not necessarily on the record written to the diskette (the beginning of a line on the source document).
FIELD ADV	Go to the next field to be entered. In some applications, you may want to inhibit FIELD ADV if data was not previously entered for the field.
REC ADV	Go to the next group of fields to be entered or reviewed. The group of fields depends on the source document. Some applications may require that data is mandatory in all fields. Thus, you may want to inhibit REC ADV if this data was not previously entered.

Displaying Data

Displaying data on the display screen should be thoroughly planned so it is most convenient and readable for the operator. For example, data is often recorded on the disk in a compact form, but the capability of the work station to display this data with separations between fields for easier reading should be utilized.

STORAGE ALLOCATION AND REQUIREMENTS

The standard work station has 4096 positions of programmable storage. This storage is used to accommodate the ACL programs during execution. Figure 24 shows how a typical ACL program appears in storage. Note that the first 1024 positions of storage contain 26 general purpose registers (A-Z), an indicator table, and work station control programming. This area of storage cannot be altered by the ACL programmer, except for the contents of the registers and the status of certain indicators.

The remaining 3072 positions of storage are divided into 128-position buffers (1-24). Buffers 1 and 2 are permanently assigned to the display screen. Any data written to buffer 1 or 2 is immediately displayed (first 120 bytes) on the screen.

The remaining 22 buffers are used to store:

- Data set and printer input/output buffers.
- Tables.
- Operator prompting messages.
- Program instructions.
- Disk indexes (key indexed access method).
- Data formats.

Buffers 3-24 are assigned by the ACL control statements, except for the data formats, which are always loaded beginning in buffer 24. Disk indexes used by the key indexed access method can be loaded automatically or assigned in the .DATASET control statement. If the automatic option is selected, the system locates an unused area in storage to build the index.

As you code an ACL program, buffer assignments start with buffer 3. The first buffers assigned are usually the input/output buffers for the data sets to be accessed and for the printer, if required. Tables and messages can be defined by using the .BUFFER control statement. You control the point where program instructions are loaded by identifying the starting buffer number. This must be an odd-numbered buffer and must follow buffers assigned for data sets, the printer, prompting messages, tables, and constants.

The 8K storage feature provides 32 additional buffers, for a total of 56.

Translator Storage Assignments

The following rules are used by the translator for placing instructions in storage. They can aid you in coding or patching a source program. (For the rules governing the label processor, see Chapter 4.)

1. Instructions preceded by step numbers are placed in storage at the position corresponding to the step number. (The actual physical location of the instruction depends on the starting buffer number of the program, as specified in columns 18-19 of the .NAME control statement.) If the indicated storage position already contains an instruction, a translator error is posted.

2. Instructions not preceded by a step number (columns 1-3 blank) are placed at the next available storage location. If the first instruction in the program has no step number, it is placed at the location for step number 000. Note that instructions do not have to be coded in the source program in the exact order they appear in the object program, unless columns 1-3 are blank.

3. When the translator has assigned all instructions to storage locations, it checks to see that any unused storage locations are preceded by a GOTO instruction. If any other instruction is followed by an unused storage location, a translator error is indicated. Note that comment records should be coded to identify buffer and indicator usage within the program.

Storage Requirements

To determine the storage requirements for your ACL programs, use the ACL storage estimator form shown in Figure 25.

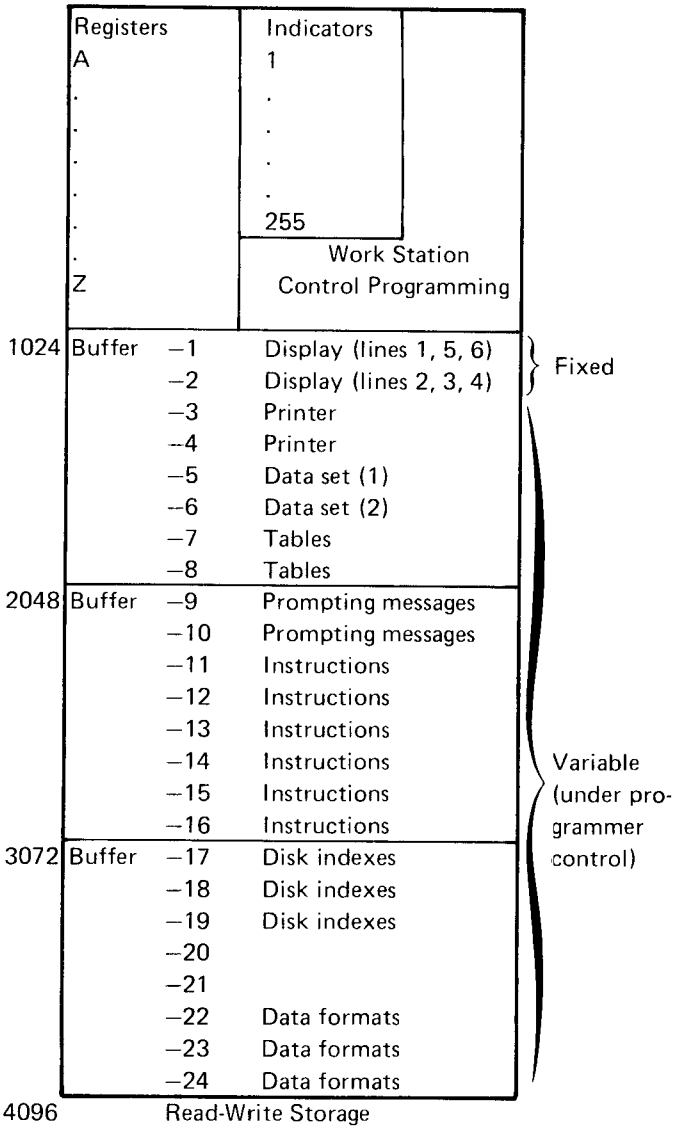


Figure 24. Typical ACL Program Structure (4K)

1	Registers, indicators, and work station control storage (FIXED)	(1024)
2	Display screen buffers (FIXED)	(256)
3	Operator prompting message*	
	Number of messages ( ) x 4 = ( )	
	Total number of characters in messages + ( )	
	TOTAL	( )
4	Data sets	
	Number of data sets ( ) x 128	( )
5	Printer buffers	
	128 print positions ( ) x 128 132 print positions ( ) x 256**	
	TOTAL	( )
6	Tables*	
	Number of entries per table ( )	
	Entry length x ( ) Plus 1 ( ) TOTAL ( )	( )

Figure 25 (Part 1 of 2). ACL Storage Estimator Form

7	Instructions*	
	Number of instructions ( ) x 4	( )
8	Data formats*	
	Number of formats plus 1 ( )	
	Total number of fields in formats + ( ) TOTAL ( ) x 4	( )
9	Index tables (key indexed access method) per index	
	Tracks in data set ( ) / Tracks per entry ( ) + 2 ( )	
	Entry length x ( )	
	TOTAL ( )	( )
TOTAL STORAGE REQUIREMENT		( )

* Because these parts of your program start in a buffer, that entire buffer may be allocated for only that part, unless you specify another use for the remaining buffer area.

** Only positions 1-4 are actually used in the next sequential buffer.

Figure 25 (Part 2 of 2). ACL Storage Estimator Form

EFFICIENT USE OF WORK STATION STORAGE

The following sections provide a number of programming techniques which allow you to conserve and fully utilize storage space in the work station.

Using Operator Messages

A key area in conserving storage is the definition and use of operator prompting messages to be displayed. If overused, these messages can not only slow down the operator but significantly increase your program storage requirements. The following are general rules for writing the ACL program:

- Specify M in column 28 of your .FIELD control statements, when applicable.
- Do not use the .BUFFER control statement to define captions displayed on lines 2, 3, and 4 of the screen.
- Reuse prompting messages defined by your .FIELD control statements, when applicable.
- Utilize abbreviations in your operator prompting messages.

In many cases, it can be advantageous to retain keyed data by moving it to lines 2, 3, and 4 of the display screen. You may also want to display a caption with the data. By specifying M in column 28 of your .FIELD control statements, the prompting message and related data is moved to the specific position on lines 2, 3, and 4, after the data is keyed. You also specify the starting display position in columns 33-35 of the .FIELD control statement. Do not use the .BUFFER control statement to define additional captions to be displayed on lines 2, 3, and 4. Each .BUFFER control statement uses 128 positions of additional storage.

It is possible to use the same messages as defined in .FIELD statements for print, diskette write, or other display operations. Each prompting message defined by a .FIELD statement is stored contiguously (followed by a 4-position control block) in the buffer specified in the .FIELD control statement. If the prompting message overflows to the overflow buffer, an additional three bytes are used at the end of the primary buffer.

Total buffer space can be utilized by combining .FIELD prompting messages, initialized .BUFFER data, and table data in the same buffer. .FIELD information, however, must start in the first position of the buffer. For example, a buffer can be initialized (with a .BUFFER statement) or table data can be loaded to fill the remaining positions of a buffer containing a .FIELD prompting message. (See *Columns 13-14 Buffer (R)* and *Columns 18-19 Overflow Buffer (O)* under *.FIELD* in Chapter 2.)

You can load prompting messages into general registers (A-Z) by issuing a REFM instruction to the buffer containing the messages. Once in the registers, the message can be used in any one of the operations mentioned.

Because the operator needs only enough information to understand what action must be taken, you can minimize storage requirements by using abbreviations in your operator prompting messages. This may, however, depend on the individual operator and the complexity of the abbreviation, although the operator will probably rely less on the prompting messages as experience is gained.

Using Tables

ACL allows you to search, read, and write tables in storage. You may occasionally find that too much storage area is required to maintain an entire table. It is possible, however, to segment a long table into subtables, and store the segments in a diskette data set. The subtables can be retrieved by using the key indexed or relative record number access methods. The key indexed method assumes an ascending sequenced table. The relative record number method assumes a table with no sequence. For both methods, let us assume that table entries are 10 positions in length, and that 10 entries are stored in each 128-byte sector, with a hexadecimal FF in position 121. The hexadecimal FF indicates the end of a table to the work station. Both access methods search for an equal entry, as specified in the TBFX instruction.

In Figure 26, the key indexed access method starts with a read or search of the table file based upon the search argument contained in register B. Note that the key is specified in the .DATASET statement as starting in position 111 with a length of 10 positions. The search of the file is terminated when an equal key or a higher key is found. The subtable segment that should contain an equal entry is in buffer 3, the input buffer for the table file. A table search is then initiated by the TBFX instruction. If no equal entry is found, register A will be zero. Register A is then tested with an IF instruction, causing a branch to an error routine called ERR1.

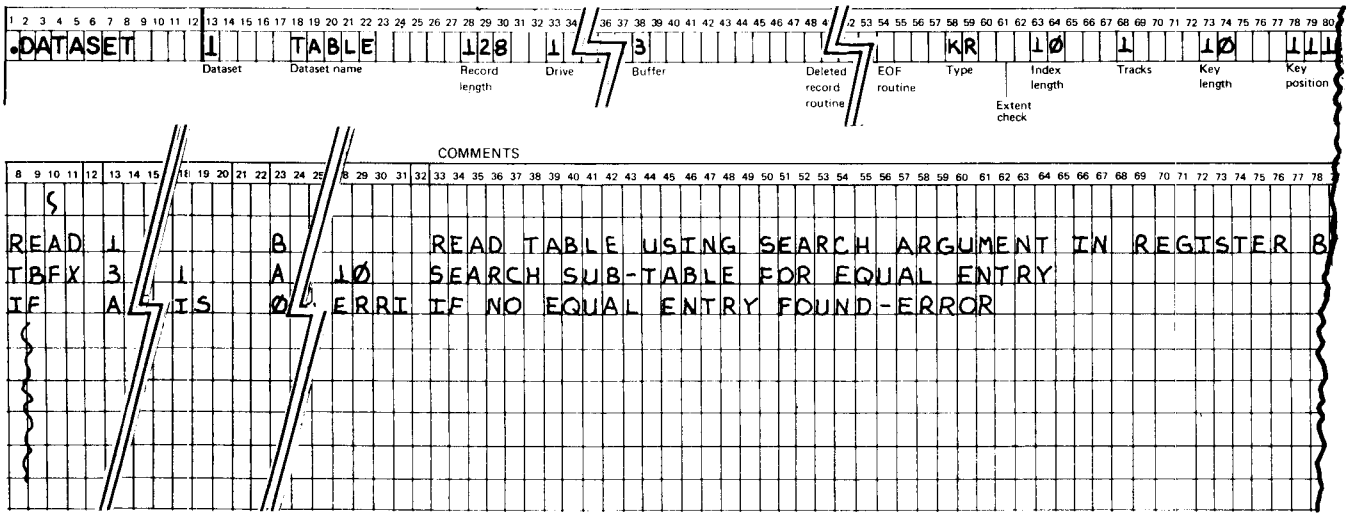


Figure 26. Key Indexed Method Searching a Table of Ascending Sequence

The relative record number access method (Figure 27), starts by initializing the relative record number in the table to 1. A read of the first record in the table file is done, followed by a table search of those table entries contained in the input buffer. If no match is found, the program branches back to the read operation and the next record in the table is read. When the end of the table file is reached and no matching entry has been found, the ACL program branches to an error routine called ERR1.

Note that the end-of-file routine is specified in columns 53-56 of the .DATASET statement. If the table is in ascending sequence, the reading of subtables can be terminated when a high entry is found by using the TBFN instruction and testing indicator 163 (table high, no equal entry found).

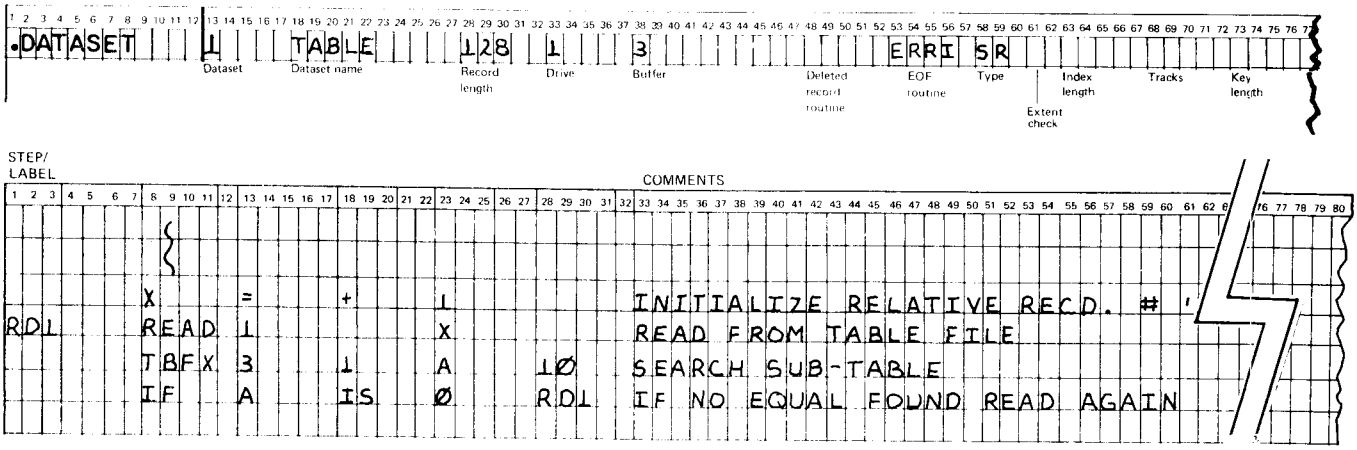


Figure 27. Relative Record Number Method Searching a Sequential Table

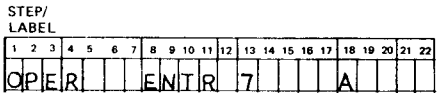
Using the Key Indexed Access Method

The key indexed access method minimizes the time required to search data sets arranged in ascending sequence by using the index, which can be built in storage automatically. The storage required for the index may exceed available storage space, depending on the size of the data set. You can vary the size of the index to be built with two entries in the .DATASET control statement. In columns 63-64 of the statement, you can specify an index length smaller than the key length specified in columns 73-74 of the .DATASET statement. This reduces the size of the total index because the index contains the high-order positions of the key. The significance of the key should not be lost. In columns 68-69 of the .DATASET control statement, the number of tracks represented by each index entry may be specified. The more tracks represented by an index entry, the less storage required for the total index. Note, however, that access time increases in proportion to the increase in tracks represented. Thus, the size of the index should be determined by both the storage required and the access time desired.

Providing Operator Error Correction

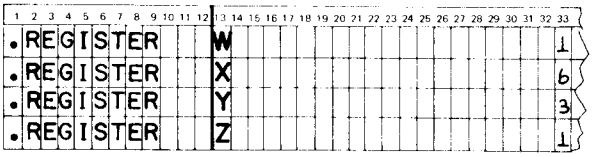
Operator error correction techniques may be provided in the ACL program, although the storage space required must be a consideration. The special keyboard close (column 36 of the .FIELD control statement) and a register in place of the actual buffer and message number (columns 13 and 18 of the ENTR instruction) may be specified to minimize the number of instructions required to provide operator error correction techniques. Figures 28 and 29 show the use of these options. Note that 14 instructions are needed in Figure 28 to provide for entered data and operator error correction. Figure 29 shows another method of providing operator error correction requiring 24 instructions.

Programming for operator error correction can be enhanced by use of .FIELD overflow buffers. For example, a corresponding ENTR instruction is:



Where register A contains the current message number, and 7 is the number of the buffer containing the start of .FIELD messages. If the overflow buffer is specified in the .FIELD control statement (columns 18-19), the buffer number does not have to be incremented or range-checked. When crossing buffer boundaries, the first complete message in the next buffer has a message number one greater than the last message in the preceding buffer.

Remember that each instruction, regardless of type, occupies four positions of storage.



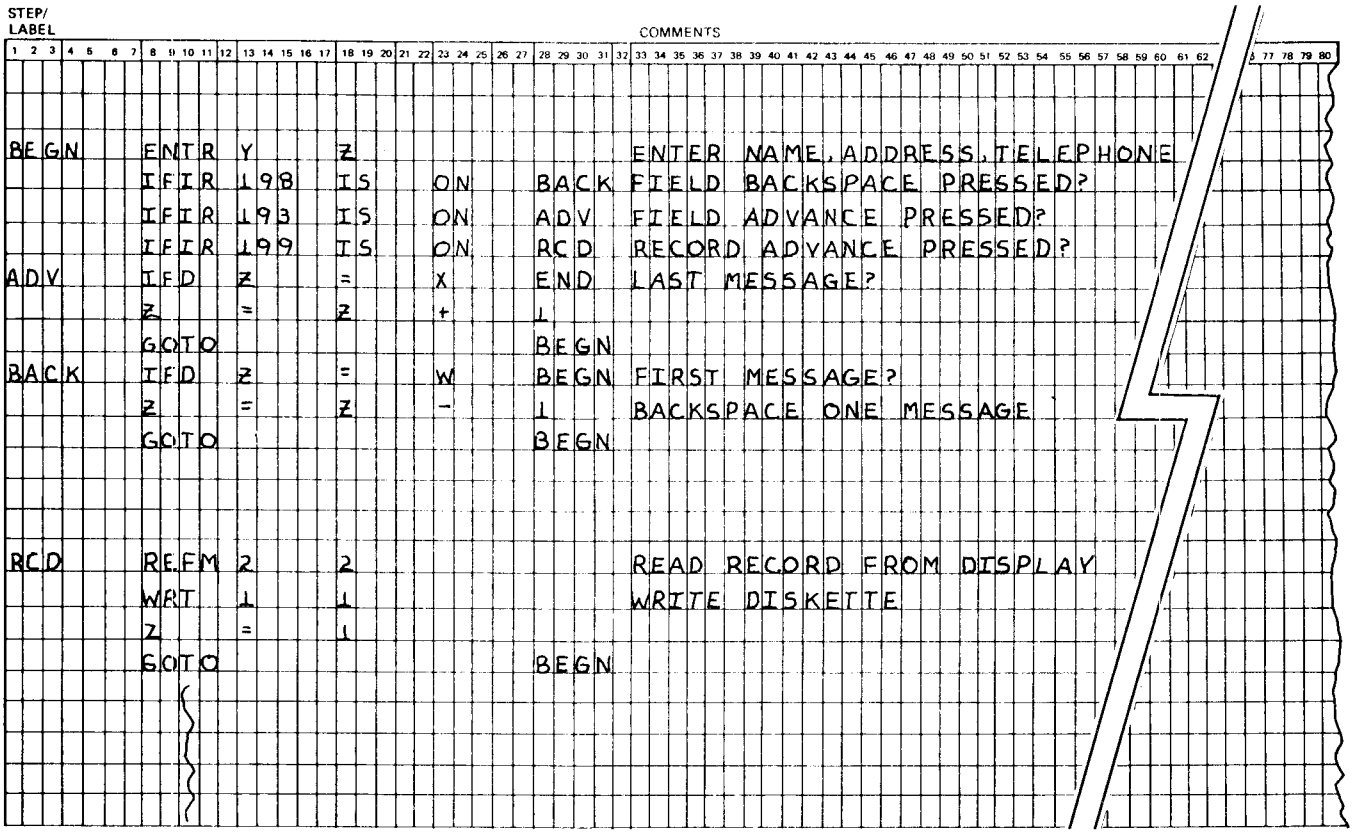


Figure 28. Technique for Providing Operator Error Correction

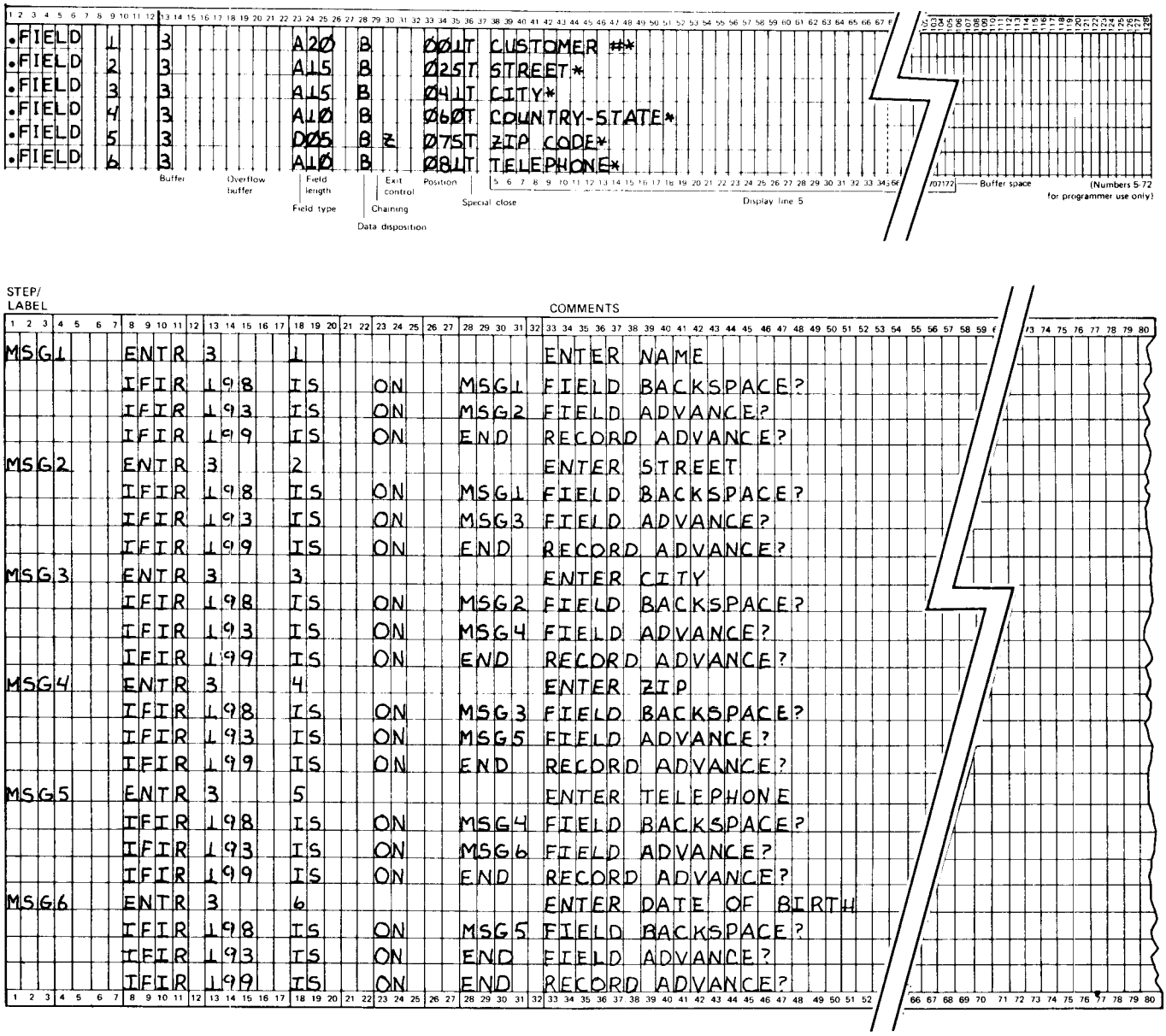


Figure 29. Technique for Providing Operator Error Correction

OPERATOR DOCUMENTATION, TRAINING, AND TESTING

Since operating procedures for the work station depend on the programming for each application, documentation and training for the operator for each job must be provided. Clear and complete documentation is essential for the following reasons:

- Work stations and operators may be at remote locations where assistance from a programmer is not readily available.
- The incidence of new operator training exists due to normal operator turnover.

Operator Documentation

To aid the operator in running your job, a job run sheet and step-by-step instructions should be provided. The job run sheet should include:

- Job name and number.
- Name, location, and telephone number of the programmer.
- Printer setup instructions (if used) specifying form number and description, print head vertical alignment, spacing desired, and number of parts per form.
- Diskette data set label specifications (if scratch diskettes are used). Note that data set labels should be preestablished for the operator and controlled by the program.
- Instructions for starting the program, including diskettes and respective disk drives, and the keying sequence for starting a job. Note that there are three basic techniques to start a job depending on (1) if the program identification is written on sector 8 of the index track, (2) if the program identification is written on a reserved sector of the index track, or (3) if the program identification is keyed by the operator.
- Instructions for ending the job.
- Instructions for aborting or continuing the job in the event of a system error.
- A summary of special keys and functions, such as FIELD BKSP, REC BKSP, REC ADV, FIELD ADV, DUP, and SEL PGM, when used to add or delete records, or for special branching.

A sample job run sheet is shown in Figure 30.

Step-by-step instructions should be provided to supplement prompting messages to the operator. Although the program contains prompting messages, these messages only tell the operator the location in the job flow. Particularly for the new operator, these prompting messages require the additional interpretation provided by step-by-step instructions. Note that these instructions should include a review of the work station error list and any necessary special instructions for some of the errors.

IBM 3741 Models 3 and 4 Job Run Sheet

Items with a box ☐ are optional and can be ignored unless checked ☒.

Job Name _____ Job Number _____

Programmer _____ Location _____ Telephone _____

Printer Setup

Form Number/Description _____

Single Double Space 1 2 3 4 5 6 Part Form

Printer Alignment _____

Starting Program

Drive 1 Diskette _____

☐ Drive 2 Diskette _____

Program Identification ☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐

Program Identification: ☐☐☐☐ will be displayed when drive 1 is ready.

☐☐☐☐ is written on the diskette. Press REC BKSP____ times until it is displayed.

☐☐☐☐ must be keyed. Press FUNCT SEL lower and DELETE REC to clear the display, then key the program identification.

After the program identification is displayed, press FUNCT SEL upper then E.

Ending the Job

_____

_____

Aborting the Job

_____

To continue after aborting

_____

Summary of Special Keys and Functions

_____

_____

_____

_____

_____

_____

_____

_____

_____

Figure 30. Sample Job Run Sheet

Operator Training

The most efficient method of operator training is for the programmer to demonstrate the program using the job run sheet and step-by-step instructions for reference. During this demonstration, the operator should execute each step while the programmer explains the meaning of prompting messages and special keys and switches. The demonstration should include operator correction of errors and unbalanced hash totals. New operators must be shown how to key minus (or credit) right-adjust fields. Note that the dash key inserts the minus sign (D zone) in the units position of numeric fields, then performs the right-adjust function. Once trained, operators can instruct new operators in the use of the job run sheet, step-by-step instructions, and work station functions.

Application Debugging

When debugging the program, use an operator who typically uses the program. This tests both the program and supporting documentation. The debugging also allows the operator to identify difficult keying sequences and make suggestions for increased efficiency.

Additional Documentation

The programmer should also provide the operator with:

- A listing of the source program.
- The files and data set labels for each diskette.

Although this documentation is not always needed by the operator, it could be useful to the IBM service personnel if a difficulty arises.

DATA SET ACCESS METHODS

The following section is provided to aid in selecting the access method most suited to your particular application. The format of the diskette for the work station is the same as for the data station. The disk unit reads and writes on only one side of the diskette. The diskette is divided into an index or label track and 73 data tracks. Each track is divided into 26 sectors. Each sector can contain up to 128 positions of data. All ACL read and write operations access one sector at a time using one of the following access methods.

- Sequential
- Random by relative record number
- Key indexed

The index or label track, track 0, can contain up to 19 different data set labels. These data set labels are used to define the data sets contained on the diskette. Data set labels are created and modified using the data station functions or by an ACL program using the label update access method. For detailed information concerning the format of the label or index layout, and the data set labels, see the *IBM 3741 Data Station Reference Manual,* GA21-9183. Data set access methods available within ACL are discussed in the following section.

Sequential Access Method

The sequential access method reads or writes sequentially based upon the physical disk address, one record or sector at a time. This same method is used by the 3741 Models 1 and 2 Data Station. The sequential method:

- Writes records into a new data set.
- Writes records at the end of an existing data set.
- Reads records from an existing data set.
- Reads and updates records in an existing data set.
- Reads and updates records in an existing data set and writes new records at the end of an existing data set.

To use the sequential method, specify an S in column 58 of the .DATASET control statement and the appropriate operation in column 59 (R for read, W for write, or U for update). In addition, to extend or write additional records in an existing data set, enter SWE in columns 58-60 of the .DATASET statement (Figure 31). To perform the actual read and write operations, the following instructions are used:

Instruction	Function
READ	Read a data record
WRT	Write a data record
WRTS	Delete a data record
WRTE	Add a data record at the end of a data set
READ and WRT	Update a data record

You can code two operands with the READ instructions. The first operand in column 13 defines the file number to be accessed, and the second operand in column 18 defines the format number to be used with the operation (Figure 32). During the read and write operations, data is moved to or from the input or output buffer assigned to that data set in column 38 of the .DATASET control statement. During a read operation, the data from the sector read is transferred to the assigned buffer; and, if a format number is specified, the appropriate data is also transferred to the general purpose registers used in the .FORMAT statement specified.

The WRT instruction has three operands. The first operand in column 13 defines the file or data set number to be accessed; the second, in column 18, identifies the .FORMAT statement to be used with the operation; and the third, in column 23, defines the buffer number from which data is transferred in order to be written on the diskette.

The WRTE instruction can contain the same operands but always writes the record at the end of data in the data set or at the current end of data address.

The WRTS instruction can be used to write a record and insert the special address mark used for deletion of a sector or record. If a format or buffer is specified in the WRTS instruction, the sector is updated to reflect values contained in the registers or the buffer. If no format or buffer is specified, a D is placed in position 1 of the record and remaining positions 2-128 are blanked.

To perform an update on an existing record within a data set, first specify the update operation by coding a U in column 59 of the .DATASET control statement.

A read to the record to be updated must be issued followed by a write to the same data set in order to update selected fields or the entire record (Figure 33).

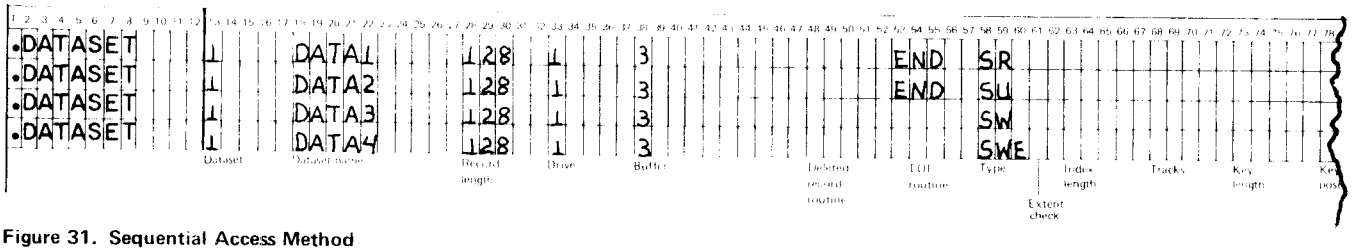


Figure 31. Sequential Access Method

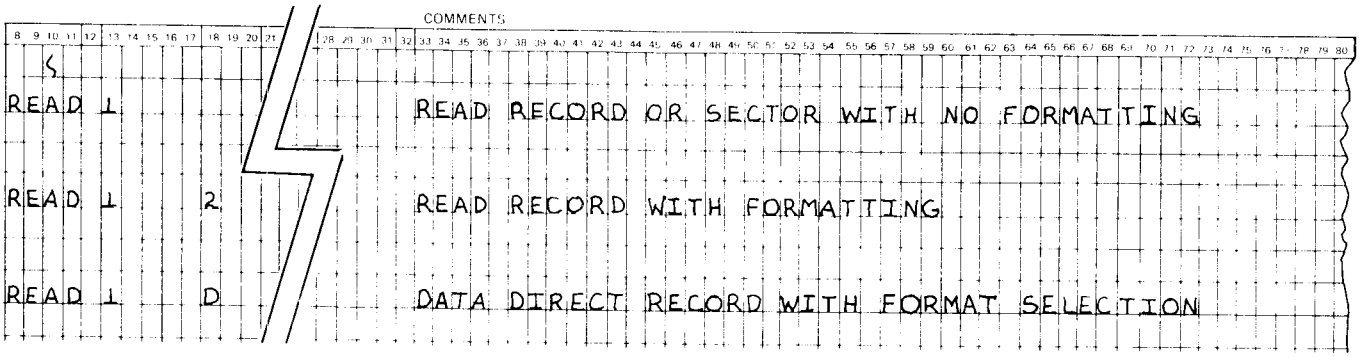


Figure 32. Sequential Read Instructions

Contained within the .DATASET control statement are two programmable exits which can be used to exercise program control when the specific conditions occur during processing of a data set:

- Deleted Record — columns 48-51
- End of File — columns 53-56

These exits are discussed under *.DATASET* in Chapter 2.

Relative Record Number Access Method

The relative record number access method reads and updates records within a data set based upon the relative record number of the record from the start of the data set. With the READ instruction, you identify the specific relative record number desired. The system then searches to that location and reads that record or sector. The relative record number access method:

- Reads records from an existing data set.
- Reads and updates records in an existing data set.

To use the relative record number access method, specify S in column 58 and R or U in column 59 of the .DATASET control statement. These specifications are the same as for the sequential access method. The difference between the two methods is in the coding of the READ operation. A third operand is required for the relative record method. The general purpose register (A-Z) that contains the relative record number to be read (Figure 34) must be identified in column 23. When the READ instruction is executed, the work station control programming calculates the physical disk address of the relative record specified. A direct seek is then issued to that record, the record is read and transferred to the input/output buffer assigned, and the data is transferred to the appropriate registers if a .FORMAT number has been assigned.

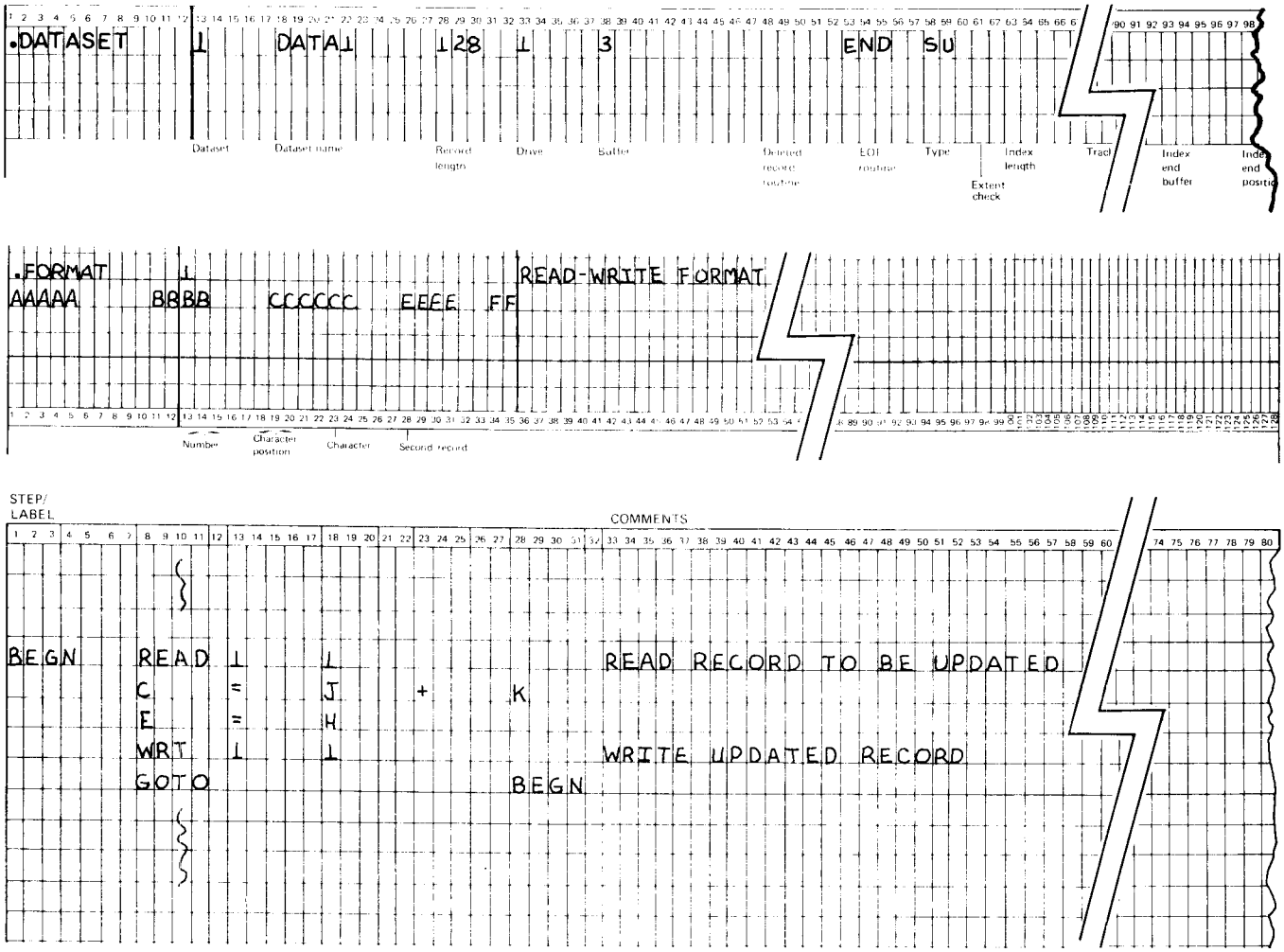
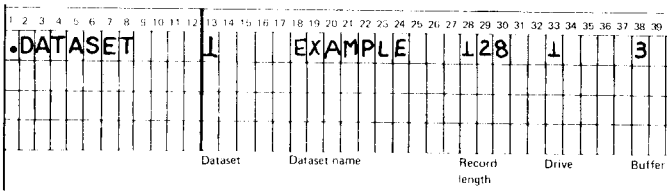
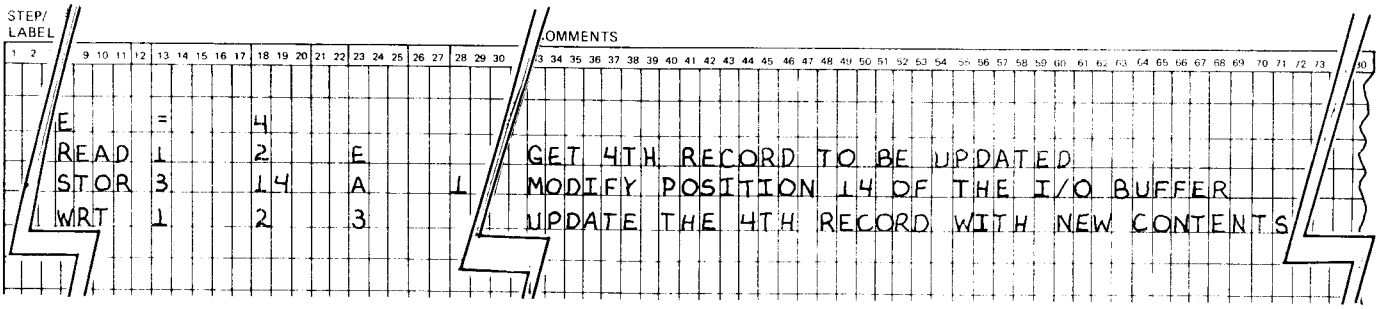


Figure 33. Coding Required for Sequential Update Access Method

To read and update a record with relative record number accessing, the record is first read into the I/O buffer. The program then modifies either the I/O buffer or the content of the register used for formatting the I/O buffer. A write command then places the updated I/O buffer back on the disk. Although the relative record is not referenced by the write command, the updating is done to the last record read. The content of the register containing the record number is incremented after the read operation. The following example illustrates this procedure.

The relative record number and sequential access methods are combined so that both methods can be used to access the same data set. After the read operation is completed, when a register has been specified in column 23, the value in the register is incremented by the work station control program to reflect the next sequential record to be read.





Note that register E is 5 after the read is completed, although the update is to the fourth record.

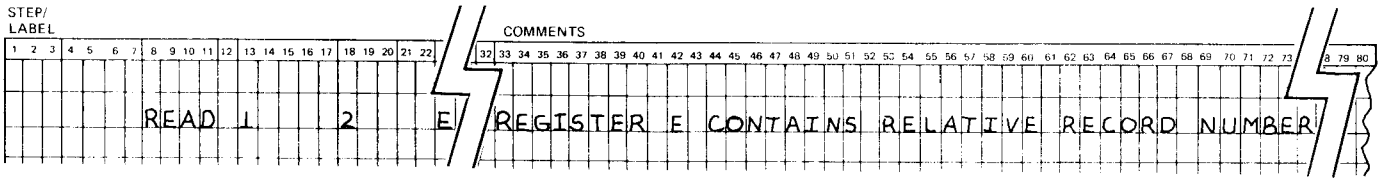


Figure 34. Instruction for Read with Relative Record Number

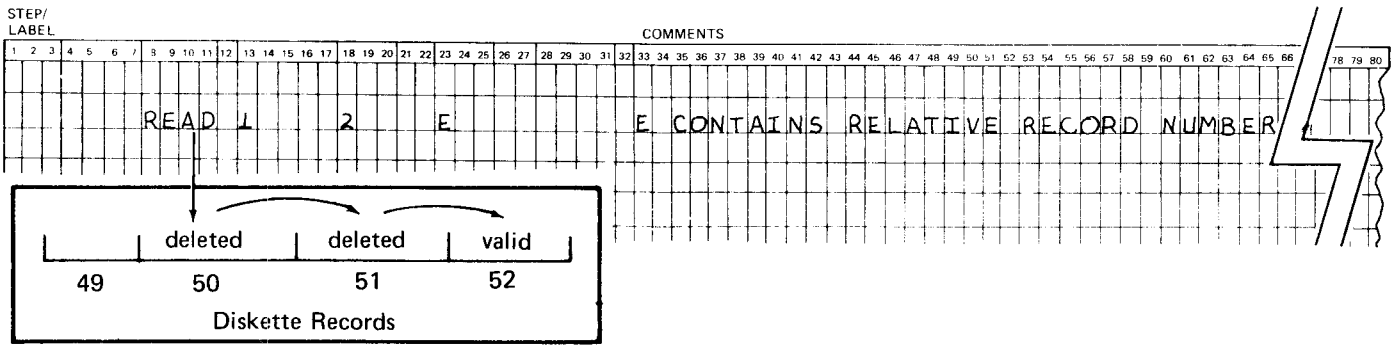


Figure 35. Relative Record Read to a Deleted Record

Deleted records should not be allowed in a data set accessed with the relative record number access method. If deleted records are found in a data set being accessed with this method, they are bypassed until the first valid record is found and processed. In addition, the value in the register specified is incremented to reflect the deleted records bypassed.

In Figure 35, a read is issued to relative record number 50. Record numbers 50 and 51 are deleted. Record 52 is a valid record. At the completion of the read operation, record 52 has been processed and the register has been incremented to 53.

To process the deleted records rather than let the work station control programming bypass them, an exit must be defined in columns 48-51 of the .DATASET control statement and instructions must be provided in the ACL program to handle the processing of the deleted records. Note that formatting is not performed if the deleted record exit is taken.

Key Indexed Access Method

The key indexed access method searches an existing data set randomly using a search argument up to 16 positions long. The search argument can be a control field or a data element within each record of a data set. The key indexed access method requires that the records in the data set be arranged in ascending sequence, according to the search argument or control field. When using the key indexed access method, the work station control program automatically builds an index table in storage for the data set to be accessed. When a search argument is specified by the ACL program, a search is first made in the index table. The index table points to the appropriate track or tracks in the data set where the record should be found. If the matching record is not present, a record-not-found condition is posted by setting indicator 225, 226, 227, or 228 (for data set 1, 2, 3, or 4, respectively) for appropriate action by the ACL program.

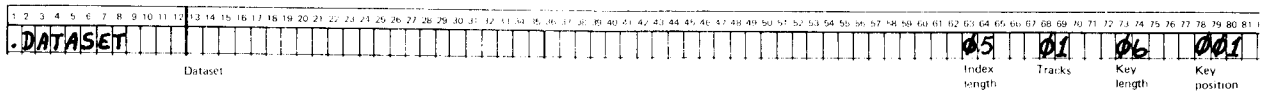
The key indexed access method provides the ability to:

- Randomly read records from an existing data set.
- Randomly read records and update in an existing data set.

To use the key indexed method, specify K in column 58 and R or U in column 59 of the .DATASET control statement. In addition, the characteristics of the index to be built by the work station control program and the size and position of the key within the records of the data set must be defined. In columns 63-64 specify the size of each index entry and in column 68-69 the number of tracks each index entry represents. Columns 73-74 are used to define the length of the key field within the data records, and columns 78-80 define the beginning position within the data record where the key is located. At the time the data set is opened during ACL program execution, the index table is built by scanning the data set and extracting the appropriate keys based upon the above parameters.

Sector No.	01	02	03	04		26
Track No.						
Data Tracks 1	KEY101	KEY102	KEY103	KEY104	KEY10	KEY126
2	KEY201	KEY202	KEY203	KEY204	KEY20	KEY226
3	KEY301	KEY302	KEY303	KEY304	KEY30	KEY326
4	KEY401	KEY402	KEY403	KEY404	KEY40	KEY426
5	KEY501	KEY502	KEY503	KEY504	KEY50	KEY526
6	KEY601	KEY602	KEY603	KEY604	KEY60	KEY626

In Core Index	KEY626	KEY601	KEY501	KEY401	KEY301	KEY201	KEY101	F
Index No.	1	2	3	4	5	6	7	8
Record Location	BOE	BOE+ 1 Track	BOE+ 2 Tracks	BOE+ 3 Tracks	BOE+ 4 Tracks	BOE+ 5 Tracks	EOD-1	End of Index Mark



In Core Index	KEY62	KEY60	KEY50	KEY40	KEY30	KEY20	KEY10	F
Index No.	1	2	3	4	5	6	7	8
Record Location	BOE	BOE+1 Track	BOE+2 Tracks	BOE+3 Tracks	BOE+4 Tracks	BOE+5 Tracks	EOD-1	End of Index Mark

To perform the search and read operation using the key indexed method, define a third operand in column 23 of the READ operation. Column 23 must contain the name of a general purpose register (A-Z) which contains the search argument.

The contents of the register are compared to the index to locate the appropriate track or tracks. A scan is then performed on the appropriate track or tracks to locate the matching record. If the record is found, the data is transferred to the input/output buffer assigned; and, if a .FORMAT number is specified, the data is moved to the assigned registers.

When the index is built by the work station control program, the program automatically locates vacant storage (between the last instruction and the data formats) for the index table. If there is not enough storage available to build the entire index, the work station control program builds a partial index and issues a halt message to the operator indicating that there is not enough space. The operator can reset the halt message and continue at a degraded performance level. The work station control program checks the sequence of index entries. If the index is found to be out of sequence, a halt is issued to the operator indicating that the data set is out of sequence, and the job is terminated.

The following parameters in the .DATASET control statement define where the index is to be loaded in storage:

Columns	Function
83-84	The buffer number where the index table is to start.
88-90	Start address within the start buffer where the index table is to start.
93-94	The buffer number where the index table is to end.
98-100	Position (plus one) in the end buffer where the index is to end.

These parameters build and load the index into specific buffers automatically. Once in the buffer, the index can be written onto a data set. The automatic index build is bypassed if N is entered in column 60 of the .DATASET control statement.

If the data set has no changes or very few changes, the index can be built, stored on a diskette data set, and, loaded at program execution. This can be done by defining the data set containing the index entries, reading the entries into storage, and moving the entries to buffers defined in columns 83-84, 88-90, 93-94, and 98-100. This approach to loading the index table may take less time than building the index every time the data set is opened.

Index or Label Access Method

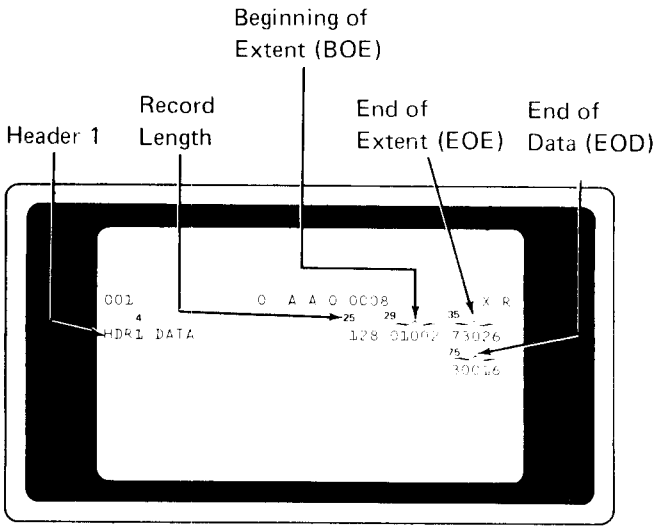
Data set labels for a given diskette are always located in sector 8 through 26 of the index track, track 00. The format and content of each label are shown in Figure 36. Figure 37 illustrates how the data set labels are displayed on the display screen to the work station operator. Labels can be created and maintained by the work station operator using the standard data station functions. See the *IBM 3741 Data Station Operator's Guide,* GA21-9131. An ACL program can also be written to create and maintain the data labels by using the index or label access method. Selection of the method used for maintaining data set labels should consider the qualifications of the operator. Labels 1-26 can be accessed.

To gain access to the data set labels via an ACL program, define the index track as a data set with a .DATASET control statement. Starting in column 18, define the name as LABEL. Be sure to specify the record length as 80 in column 28. Specify the access method in column 58 of the .DATASET statement as I. No entry in column 59 is required. Code an A in column 61 to suppress extent checking. Access the individual labels by identifying the relative record number of the label on track 0. The relative record number is placed in the general purpose register (A-Z) coded in the third operand of the READ instruction, column 23. To update or modify the label, the READ instruction must be followed by a WRT instruction to the same file. Note that, when using this access method, you should avoid creating a data set label with extents that overlap with an existing data set.

Field Name	Position	Purpose
Header 1	1-4	Label identifier; must be HDR1
	5	Reserved
Data set name	6-13	Descriptive name for data set
	14-22	Reserved
Record length	23-27	Logical record length
	28	Reserved
Beginning of extent (BOE)	29-33	Identifies the address of the first sector of the data set. Positions 29 and 30 contain the track number, position 31 must be 0, positions 32 and 33 contain the sector number.
	34	Reserved
End of extent (EOE)	35-39	Identifies the address of the last sector reserved for this data set.
	40	Reserved
Bypass data set	41	The IBM 3747 Data Converter and the 3741 communication feature require that this field contain a B or a blank. If a B is not present, the data set is processed. The coding allows the user to store programs and data on the same disk. In communications mode on the 3741 Models 2 and 4, a B in column 41 indicates that the data set is to be bypassed during transmit mode and used during receive mode.
Accessibility	42	This field must contain a blank in order for processing to take place.
Write protect	43	If this field contains a P, the disk can be read only; otherwise this field must be blank, in which case both reading and writing are permitted.
Interchange type indicator	44	Must be blank. A blank indicates the data set can be used for data interchange.
Multivolume indicator	45	A blank in this field indicates a data set contained on one diskette; a C indicates a data set is continued on another diskette*; an L indicates the last diskette on which a continued data set resides.
Volume sequence number	46-47	Volume sequence number specifies the sequence of volumes in a multivolume data set. The sequence must be consecutive, beginning with 01 (to a maximum of 99), if used.
Creation date	48-53	Can be used to record the date the data set was created. The format of the creation date is YYMMDD, where YY is the year, MM is the month, and DD is the day.
	54-66	Reserved
Expiration date	67-72	Can be used to contain the date that the data set expires. The format of the expiration date is YYMMDD, where YY is the year, MM is the month, and DD is the day.
Verify mark	73	This field must contain a V or a blank. V indicates the data set was verified.
	74	Reserved
End of data (EOD)	75-79	Identifies the address of the next available sector.
	80	Reserved

*The work station program loader does not recognize continued data sets (BSC) when loading the ACL program. The same is true for the translator source data set, and for data sets used by the ACL program. (See *Multiple Diskette Data Sets* in this chapter.)

● **Figure 36. Data Set Label Format**



Note the positions in which these fields are located.

Figure 37. Data Set Label Fields on the Display Screen

BLOCKING AND DEBLOCKING OF LOGICAL RECORDS

All read and write operations to diskette data sets are performed on a sector basis. A READ instruction always transfers one sector from the diskette to the work station storage. A write instruction (WRT, WRTE, WRTS) always transfers one sector to the diskette from the work station storage. In some cases, storing more than one logical record in a 128-byte sector may be advantageous, because of file size or access performance. The ACL programmer must perform the blocking and deblocking of logical records within his program.

To assist in the blocking and deblocking operations, two instructions are provided within the application control language:

Instruction	Function
RBLK	Read blocked record with offset
WBLK	Write blocked record with offset

The RBLK instruction rereads a portion of a blocked record or sector after it has been transferred by a READ instruction (without a .FORMAT defined from the diskette) into the input/output buffer in work station storage. A .FORMAT statement that defines the image of the first logical record within the sector must be defined.

Follow the READ instruction with a series of RBLK instructions to reread each logical record, and the appropriate instruction to process the data. The format of the RBLK instruction is:

Columns	Specification
8-11	RBLK operation code
13	Data set input/output buffer defined in .DATASET statement
18	.FORMAT statement number
23	General purpose register (A-Z) containing logical record offset

Assume that there are three logical records, 40 characters long, stored in a sector. The first logical record begins in position 1 and the last record ends in position 120. A .FORMAT statement is defined for record 1 as shown in Figure 38. Once the sector is read without a .FORMAT defined from the diskette data set, the RBLK instruction issues first with the offset of 1, second with an offset of 41, and, finally, with an offset of 81 stored in register A.

The offset is always equal to the first position in the logical record. This same technique can be used in the blocking of logical records using the WBLK instruction. See Figure 39.

Using the RBLK and WBLK instructions in conjunction with the key indexed access method requires special processing techniques with the ACL program. The key indexed access method allows you to specify one key field per sector. Multiple logical records within a sector mean multiple key fields. To handle multiple records, specify the key field within the last logical record of the sector in the .DATASET control statement. When the READ instruction for the key indexed data set is issued, the sector transferred to input/output buffer is the sector containing an equal key or the next higher key. Test for the record-not-found condition, indicators 225-229, depending upon the data set number. If the appropriate indicator is not on, the last logical record in the sector is equal. If the indicator is on, reread the other logical records in sector and compare the key field of each to the search argument to find the equal record.

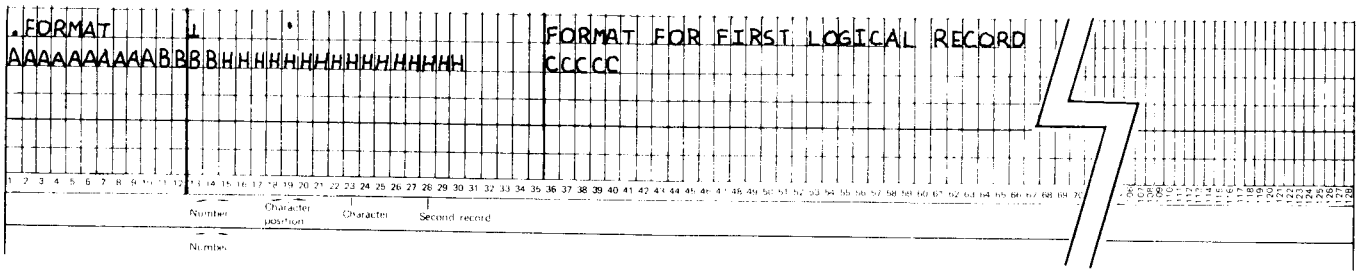


Figure 38. Sample .FORMAT Control Statement

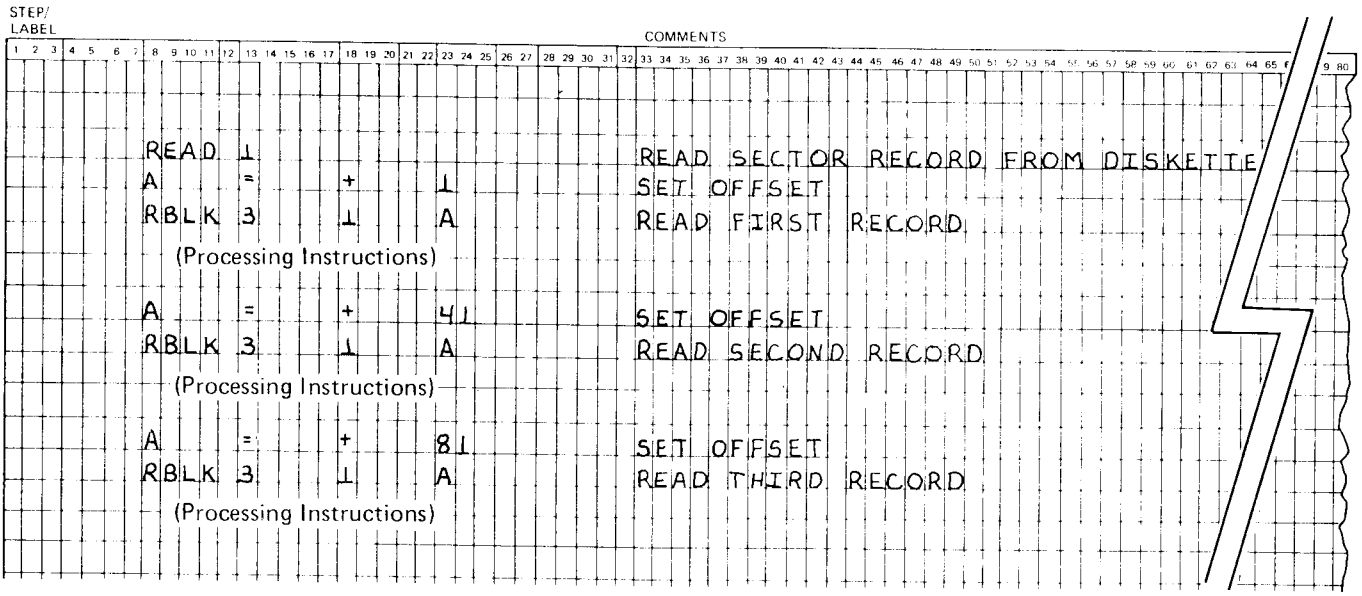


Figure 39. Instructions for Reading Blocked Records with Format Offset

MULTIPLE DISKETTE DATA SETS

The work station control program opens all data sets defined with a .DATASET control statement when the operator loads an ACL program. At the conclusion of a job, when an EXIT or EXEC instruction is executed, all open data sets are closed and the normal end of job halt (100) is posted for the operator. These procedures are followed by the work station control program, unless the OPEN and CLOZ instructions are specified within the ACL program.

The OPEN and CLOZ instructions allow you to dynamically open and close data sets during the execution of the ACL program. This ability provides flexibility in creating your work station application. For example, a data set may be larger than one diskette. If so, mount one volume of the data set, open, process, and close the volume. In turn, the second volume can be mounted, opened, processed, and closed.

The OPEN instruction can specify:

- The data set to be opened.
- A .FORMAT statement to be used to read the label.
- A register or a pair of registers containing information which overrides certain parameters within the .DATASET control statement.

The first operand identifies the data set number to be opened.

The second operand defines a .FORMAT statement to be used in reading selected fields within the data set label such as the BOE, EOD, or EOE.

The third operand overrides selected parameters within the .DATASET statement. The parameters included are data set name, drive number, and access method. (See Chapter 2 for the discussion of OPEN and the related parameters.)

This operand makes it possible to open more than four data sets during an ACL program execution. A dummy .DATASET control statement can be defined with asterisks for the data set name. This specification indicates to the work station control programming that the data set is to be opened during program execution. The .DATASET statement can be updated with the actual data set name and its related attributes.

CURRENT FILE DISK ADDRESS (CFDA)

The CFDA is an internal address consisting of disk sector and cylinder (track) numbers. The programmer will find it useful, in some cases, to be able to manipulate this disk address. The CFDA cannot be changed directly, but its value can be altered by issuing read or write commands.

When the CFDA is changed by the system, it is altered after a disk instruction (read, write, for example) is initiated, but before any physical disk execution is attempted. Figure 40 summarizes how the work station alters the CFDA as a function of the data set access method and as a function of the type of disk instruction issued. Note that:

1. The BOE is the sector-cylinder address of the first record in the data set.

2. The EOD is the sector-cylinder address of the first available record space after the last record in the data set. EOD is continually updated as records are added to the end of a data set (extending).

The value of the CFDA is altered by the system as shown in Figure 40. The system uses the CFDA, plus the change shown in Figure 40, prior to attempting the operation shown in the left column.

	Data Set Organization Type						
	(SR) Sequential Read	(SW) Sequential Write	(SWE) Sequential Write Extend	(SU) Sequential Update	(KR,KRN) Key Indexed Read	(KU,KUN) Key Indexed Update	(I) Index Update
Value after file open	BOE-1	BOE-1	EOD-1	BOE-1	BOE-1	BOE-1	0000
Value change during write WRT	Invalid	+1	+1	None	Invalid	None	None
WRT with relative record number	Invalid	Invalid	Invalid	+1	Invalid	Invalid	+1
Value change during write WRTE	Invalid	Invalid	+1	+1	Invalid	Invalid	Invalid
Value change during read-offset not specified	+1	Invalid	Invalid	+1	+1	+1	+1
Value change during read-offset specified	Value in register	Invalid	Invalid	Value in register	Value in register	Value in register	Value in register
Value change during read minus (-)	-1	Invalid	Invalid	-1	-1	-1	-1
Value change during read with sequential key specified-match found	Value in register	Invalid	Invalid	Value in register	CFDA placed at matched record address	CDFA placed at matched record address	Value in register
Same as above but match not found	Value in register	Invalid	Invalid	Value in register	CFDA placed at next higher record address	CFDA placed at next higher record address	Value in register

Figure 40. Alterations to Current File Disk Address

PROGRAMMING HINTS

The following section contains a collection of hints for consideration when designing your ACL program.

Control Program

A control or master program can be created to control program execution sequence and selection and to minimize the operator interface in program or job selection. For example, you can specify that the invoicing program automatically follow the order entry job. Also, the execution of a specific job may be inhibited until all prerequisite jobs have been completed. The major benefit of this approach, however, is easier usage by the operator. Program or job selection can then be done interactively to minimize operator confusion and resulting errors. See *Execute Program Chain (EXEC)* in Chapter 2 for a discussion of implementing this procedure.

.FIELD Control Statement

When .FIELD control statements are used, consider the following:

1. To share a buffer by using .FIELD statements in the low-numbered positions (positions 1-80) and data for constants in the high-numbered positions (positions 81-128), the .BUFFER control statement must precede the .FIELD statement.

2. The use of chained .FIELD statements is not recommended, because it eliminates the ability to identify the current message. This is especially true when using the special keyboard close indicator. In this case, when the special keyboard close option is specified (T in column 36 of the .FIELD control statement) and chaining is also specified, it is impossible to identify the current message if the special keyboard close indicator is on when the ENTR instruction is completed.

Keyboard Indicator

1. Indicator 197 is used to indicate the status of the keyboard. It is only significant when overlapped ENTR instructions are used. When it is on, indicator 197 indicates that the keyboard is open for data entry. This means that the program has opened the keyboard with an overlapped ENTR instruction (an X in column 21) and the operator has not yet exited the field.

2. The keyboard *cannot* be opened by setting on indicator 197 (with a SON 197 instruction). Setting on indicator 197 results in no operation. Most non-data keys and the three switches (AUTO REC ADV, PGM NUM SHIFT, AUTO DUP/SKIP) are active (they set their respective indicators) even though the keyboard is not open.

3. Setting off indicator 197 does not close the keyboard.

PROGRAMMING RESTRICTIONS

The following are general restrictions to consider when designing the ACL program.

Tables

1. No more than 16 tables can start in one buffer.
2. Registers I, R, and Z cannot be specified as the table index.
3. Table entries are limited to 16 positions.

Program Origin Buffer

The program origin buffer (which identifies where instructions are to begin) must be specified in the .NAME control statement as an odd-numbered buffer.

Sequence of ACL Source Programs

1. The .NAME control statement must be the first statement in the source data set.
2. The .DATASET, .PRINTER, .SELF-CHECK, and .REGISTER control statements may follow the .NAME statement in any sequence.
3. The remaining control statements, .FORMAT, .BUFFER, and .FIELD, can follow in any sequence, but must precede instructions.
4. The .END control statement must be the last statement in the source data set.

Display Unit

Lines 1, 5, and 6 are referenced by reading or writing to buffer 1. Lines 2, 3, and 4 are referenced through buffer 2.

1. The first 40 positions of buffer 1 contain line 1 of the display, which is used by the work station control programming. Positions 1-8 are used to post error messages and halts. Positions 9-36 are used by the trace functions (Chapter 4) during program execution. Positions 37-40 are used to post the ACL program name.
2. The programmer should leave positions 49 and 50 of buffer 1 (line 5, positions 9 and 10) blank for aesthetic purposes; however, position 49 can be used if needed. Position 50 is blanked by the work station each time an ENTR is processed.
3. Positions 51-120 of buffer 1 contain prompting messages and fill characters that display on lines 5 and 6.
4. Positions 121-128 of either buffer 1 or 2 are not displayed or maintained by the work station control programming. However, when the above positions are specified in columns 33-35 of the .FIELD statement, the data is moved into positions 121-128 of buffer 2 and positions 41-48 of buffer 1. Positions 41-48 of buffer 1 are not cleared when buffer 2 is cleared.
5. Positions 1-120 of buffer 2 are displayed on lines 2, 3, and 4.
6. To display all 128 positions of a diskette record, the first 120 positions can be displayed on lines 2, 3, and 4, and positions 121-128 can be displayed in positions 41-48 of buffer 1, or on line 5 of the display unit.
7. The maximum allowable number of positions for a prompting message and related fill characters is 68.
8. Position 8 of line 1 must be blank the first time card I/O (129 or 5496) is called. If position 8 is not blank, the error line (indicator 161) is turned on. This condition only applies to the first call to card I/O (subsequent calls do not affect the error line).
9. If an overflow buffer is specified in columns 18-19 of the .FIELD control statement, and if a message overflows to that buffer, then the first message in the overflow buffer is referenced as message number 2.

Printer Operations

1. A maximum form length of 127 lines can be specified in the .PRINTER control statement.

2. When a 132-position print line is specified, two consecutively numbered buffers are required, and the first buffer must be an odd-numbered buffer.

3. Positions 129-132 are not maintained by the system. If 132 print positions are specified, ensure that the first four positions of the buffer following the print buffer are cleared, as required.

4. Take care in attempting to print records containing certain characters. In some cases, nongraphics print as graphic characters.

5. The 3713 Printer allows for suppression of spacing during a print operation. Time must be allowed, however, for the print head to return to the left margin. This is done automatically by the ACL control program.

6. .PRINTER control statements are required if trace or dump to printer is to be used. The .PRINTER statement should be removed or verification must be made to ensure that the type printer attached and the .PRINTER statement are in agreement during program execution. If the .PRINTER statement is not correct, errors can result.

Arithmetic Operations

The basic format of arithmetic instructions is as follows:

$R_1 = R_2$ operator R_3

1. R_1 must always be a general purpose register. R_2 can be a register or a single digit constant (0-9) except in the divide operation where it can only be a register. R_3 can be a register or a single digit constant.

2. The move immediate instruction shown below is limited to a constant of plus or minus 65535.

   R = ±65535

   R equals general purpose registers A through Z.

3. In the multiply and divide instructions, the R_2 and R_3 registers cannot be the result register (R_1).

Branching Operations

1. The Indexed GOTO provides a step number from 0 to 767, or 0-999.

2. The following instructions are limited to branching within the same 256-instruction block.

   IF A = B
   IF A > B
   IF A < B
   IFD A = B
   IFD A > B
   IFD A < B

Disk Access Methods

1. The maximum number of data sets that can be online at one time during program execution is four.

2. In the key indexed method, the maximum size for a search argument or key is 16 positions. The key indexed access method provides for read and update operations.

Internal Data Movement

Take care when processing data directly to and from overlapped input and output buffers with the following instructions:

ICBF
TBWT
PUTB
STOR
ENTR (keyboard overlap)
WRT and WRTE (diskette operations which are always overlapped)
PRNT (printer overlap)

To ensure the integrity of these I/O buffers, issue a WAIT instruction before or after the I/O instruction or test the I/O busy indicator(s) before issuing one of the above instructions.

Restricted Areas

Modification of storage used for ACL control can result in invalid execution of programs if IBM changes the location or definition of this area.

PROGRAM PERFORMANCE

This section presents various techniques and procedures to improve the performance of the ACL program. Because the work station is an operator-oriented device, preserve this orientation by assuring that the data processing performed by the machine is not apparent to the operator. In other words, the operator should not have to wait for the machine. For this reason, the following suggestions are made to improve the performance of the program.

General Considerations

ACL application programs can perform some processing during data entry; however, processing should be kept at a level that will not impact the data entry operators efficiency. Delaying and overlapping processing steps to utilize the time it takes an operator to turn pages or skip to different areas on a data entry form is called effective overlapping. Combining effective overlapping with the overlapping capabilities of instructions ENTR, PRNT, WRT, and WRTE (actual overlapping) will provide the most efficient programming performance. Processing may be delayed until a batch editing run is made after data entry is complete. This completely minimizes operator delay time.

Another method of improving program performance is to reinitialize a diskette to an alternate sequence. A diskette initialized to sequence 01 may require a longer program run time than a diskette initialized to alternate sequence 02, 03, . . . 13. Refer to *Disk Record Sequences* in the section *Disk Initialization* in the *IBM 3741 Data Station Reference Manual*, GA21-9183.

Overlapped I/O-Printer

Under normal operation of the PRNT instruction, the machine waits until the printer has completed its action (print/return, skip is always overlapped) before proceeding to the next instruction. This prevents the inadvertent modification of the printer buffer by subsequent instructions before the printer uses the buffer. To improve performance, enter an X in column 28 of the PRNT instruction to cause the machine to begin execution of the next instruction as soon as the print cycle has been initiated. However, it then becomes your responsibility to preserve the contents of the printer buffer and ensure that the printer has completed its cycle before modifying that buffer. Use the IF PRT BSY to determine if the printer is busy, or use the WAIT instruction. Figure 41 shows the coding for overlapped printing. If the printer buffer is not protected, intermittent print problems can occur, and not be detected during trace operations. The overlapped print instruction can significantly improve performance, but should be used only after it is thoroughly understood. Note that overlapped printing does not occur during the trace operations (Chapter 4).

In Figure 41, the X in column 28 of the PRNT instruction indicates that the following READ instruction can be executed while the print cycle is being completed. The IF PRT BSY instruction indicates that the program should loop on this instruction until the printer has completed its cycle. The contents of buffer 4 (containing data set 1) are then moved into buffer 5 (printer output buffer).

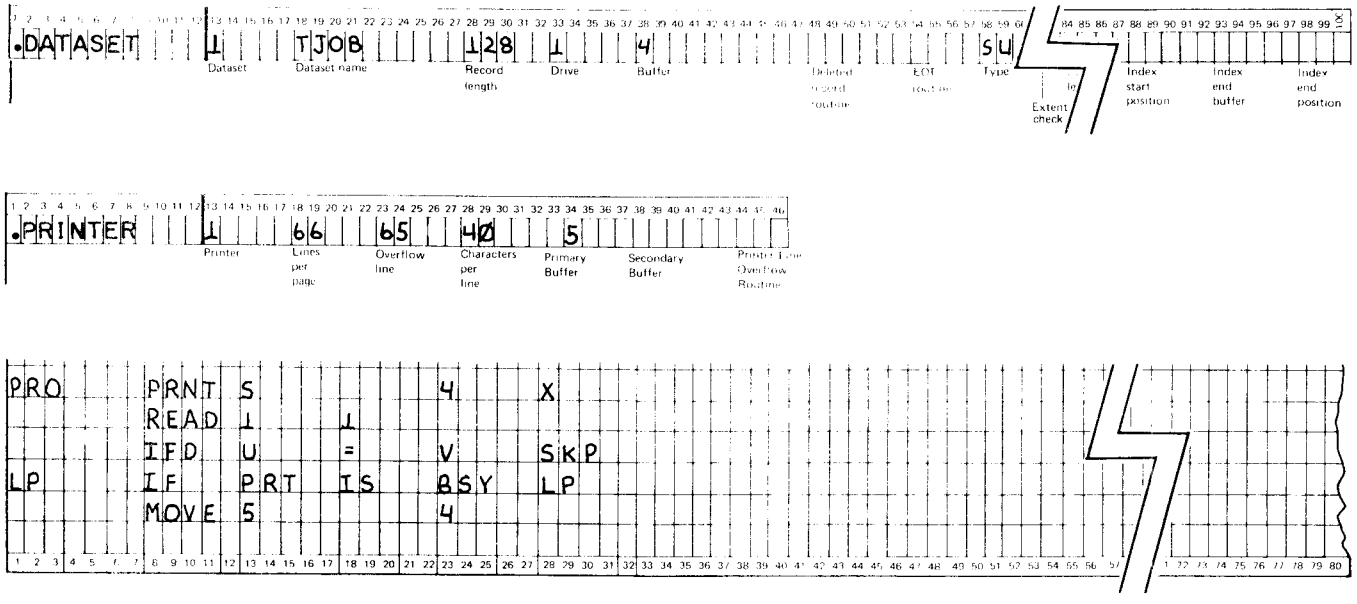


Figure 41. Sample Coding for Overlapped Printing

Overlapped I/O-Keyboard

Like the PRNT instruction, the normal operation of the ENTR instruction is nonoverlapped. If processing can be done before the data from the keyboard is available, use the overlapped ENTR instruction. By entering X in column 23 of the ENTR instruction, you allow the machine to proceed to the next sequential instruction as soon as the prompting message is displayed to the operator. Indicator 197 is set on by the ENTR instruction and set off as soon as the last ENTR field is exited. Indicator 197 can be tested to determine if the required data is available (Figure 42).

In Figure 42, the X in column 23 of the ENTR instruction indicates that subsequent instructions can be executed as soon as the prompting message (ENTER QUANTITY) is display to the operator. The IFI 197 instruction indicates that the program should loop on this instruction until the operator has entered required data and exited the field (indicator 197 off). The IFI 200 instruction indicates that the program should branch to instruction SP (not shown) if the operator exited the field via a special keyboard close key (FIELD, BKSP, DUP, FIELD ADV, REC BKSP, REC ADV, or SEL PGM). The special keyboard close option is indicated by the T in column 36 of the .FIELD statement.

Disk/Data Set Procedures

You can enter an A in column 61 of the .DATASET control statement to suppress the overlap extent checking which is part of the process of opening a data set. This decreases the amount of time required to open a data set. By taking this option, however, you lose some of the protection provided by the system to ensure that multiple data set labels do not address the same disk space (that is, the extents do not overlap).

Program Load of Index Table for Key Indexed Data Set

By using the options specified for program load of the index table in the .DATASET control statement, the time required to open a key indexed data set is significantly reduced. See *Data Set Access Methods* in Chapter 3, for a discussion of implementing this option.

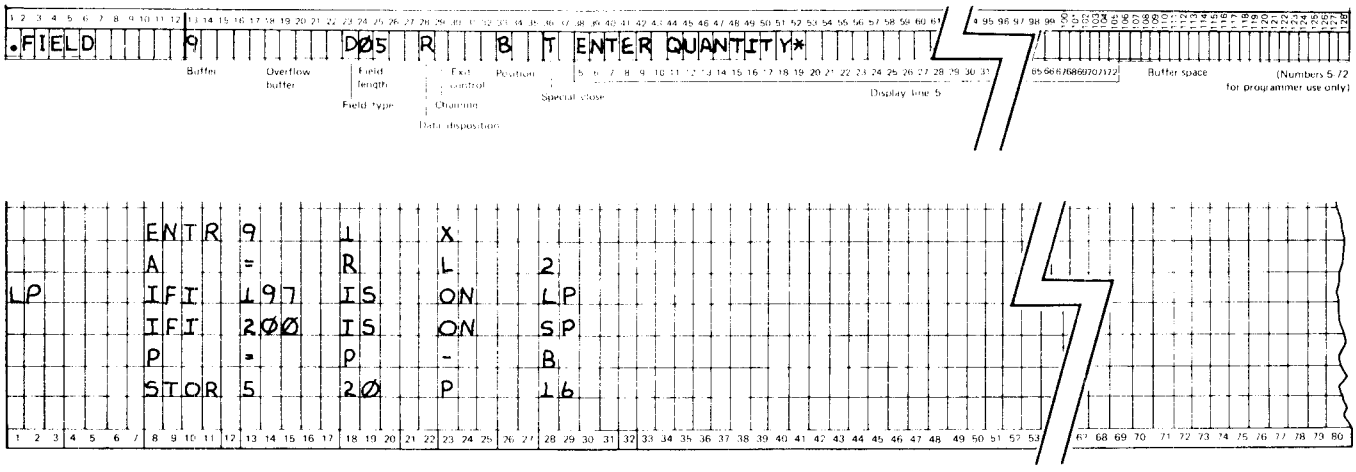


Figure 42. Sample Coding for Overlapped ENTR Instruction

Record Access

Review the ACL access methods to ensure that the method selected provides the greatest possible advantage for a specific application. The data sets used in that application should be positioned to minimize disk seek times. To do this, place only one active data set on each disk drive. If this is not practical, ensure that the most active data sets are on separate disk drives. Analyze the expected usage of the data sets to determine if special placement on the diskette provides any benefits. For example, if the majority of activity on a given data set is expected in the first portion of that data set, place it as the last data set on a diskette. It should be placed last in terms of extents, not necessarily in terms of data set labels. This reduces the average time required to access a different data set on the same diskette.

It is often possible to improve the performance of a program by controlling the overlap of machine and operator-machine functions. For example, there is a normal delay in operator data entry which occurs as operator attention shifts from one line to the next on a source document. This delay increases as the operator finishes a document and starts the next. These intervals can be used to perform I/O functions that might otherwise affect operator throughput. Certain I/O functions can also be overlapped (such as ENTR, PRNT, and disk WRT) to minimize the impact of disk record accessing. For example, allow the operator to begin keying the first field of the next document before writing the first record to the diskette. This makes the record access time transparent to the operator and improves program performance.

Execution Timing

The execution timings for certain machine operations in the following list are for use in estimating program execution time, and to aid in selection of alternatives for resolving specific problems. Since many operation times depend on the data being manipulated, it is not practical to attempt a calculation of exact execution times. However, ranges are provided to allow reasonable estimates.

Operation	Execution Time (ms)
Shift Right	4.4
Shift Right and Round	6.6
Shift Left	4.4
Shift Left with Sign	6.6
Add	12-22
Subtract	12-22
Multiply	45-148
Divide	10-130
Equate (A = +XXX)	11
Replace (A=B)	4.8
LOAD	3.6-4.8
STOR	2.4-4.8
GOTO	1.8
GOTO (indexed)	2.4
IF (Logicals)	2.4-6.0
NOP	2.5
ICBF	1.7
EXCH	38.0
MOVE	26.0
SON	2.2 (1 indicator)
	2.8 (2 indicators)
	3.6 (3 indicators)
SOFF	2.1 (1 indicator)
	2.7 (2 indicators)
	3.6 (3 indicators)
RGO	2.0
SCE – Register	1.6
– Buffer	1.9
SCN – Register	1.6
– Buffer	1.9
Table Operations	See Figures 43-46
Disk Operations	See Figures 43-46

The following execution timings have assumed operations with no overlapped I/O or formatting. T represents time in milliseconds.

Operation		Range
MVER	TMVER = 2.6 + 0.14L	2.74-4.84
MOFF	TMOFF = 2.5 + 0.2L	2.7-5.7

L = The value in the length field of the source instruction (column 28).

Operation

GETB TGETB = 4.6 + 0.09(N-1) + 0.09C + 0.096L
PUTB TPUTB = 3.6 + 0.09(N-1) + 0.09C + 0.16L

N = The value for table number (column 18).
C = The number of characters bypassed, including table separator X'FF', until the specified table is reached. The value is 0 for table 1 (N=1).
L = The value in the length field of the source instruction (column 28).

Operation

REFM $T_{REFM} = 9.7 + 0.336H + 1.83_{int}\left(\frac{H}{8}\right) + 0.064F + 2.43 + 0.9(D+1)_{mod\ 2} + 0.93E$

WRFM $T_{WRFM} = 10.6 + \begin{Bmatrix} A \\ B \\ C \end{Bmatrix} + 0.336H + 1.83_{int}\left(\frac{H}{8}\right) + 0.064F + 1.78D + 0.9(D+1)_{mod\ 2} + 0.144E$

A = 16.4 if source column 23 = column 18 is not blank.
B = 0.0 if column 23 = column 18.
C = 5.8 if column 23 is blank.
H = The number of formats defined before the specified format.
F = The total number of fields defined in all of the formats defined prior to the specified format.
D = The number of fields defined within the specified format.
E = The total number of character positions defined within the specified format.
int = The int means take the integer value of. The term in which int appears assigns an additional 1.83 milliseconds for every eight formats defined before the specified format. Take the next lowest integer value.
mod 2 = The mod 2 means modulus 2 which adds 0.9 million when D is even.

Operation

RBLK TRBLK = TREFM + .7 milliseconds
WBLK TWBLK = TWRFM

The timings for REFM with data directed formatting (D in column 18) and WRFM with editing will be slightly higher than the timings for REFM and WRFM which have been defined.

Refer to the preceding list and note the following items.

1. The variation in the execution times of the SON and SOFF instructions is caused by the varying number of indicators being set on or off.

2. The execution time of the WRFM instruction depends on the number of fields in the referenced format, the length of those fields, and whether or not a buffer is specified (Figure 43).

3. The execution time of the REFM instruction depends on the number of fields in the referenced format, and the length of those fields (Figure 44).

4. The variation in execution time of the IF (logicals) is caused by left-to-right scan of the specified registers. The instruction is exited as soon as the required mismatch or unequal entry is found.

5. The execution times of the table operations depend on the physical characteristics of the table, and the index within the table of the element being accessed.

TBFX/TBFN

The following characteristics of TBFX and TBFN instructions have significant impact on the execution time of these instructions:

- The location of the specified element in the table. Since the table is searched from the low-ordered end to the high-ordered end (first of first buffer through last of last buffer), it takes less time to find an element of a table if that element is toward the front end of the table.

- The location within each element of the significant characters. The search argument is compared against the individual elements of a table on a character-by-character basis from the left. The first mismatch within an element causes the machine to start comparison on the next element in the table.

The effect of this can be seen in Figure 45, where the only difference between the execution time for table operations on tables with different element lengths (that is, number of characters per element) is in the potential location of the significant character. (For example, if an argument of 000123 is compared with an element of 002345, the third character would be the first significant character).

TBRD/TBWT

The execution time of these instructions is a function of the length of elements within the table and the relative location (index) of the element being accessed. Sample execution times are shown in Figure 46.

Diskette Operations

The execution times for the disk operations (READ, WRT, WRTS, WRTE) are a function of the access method of the data set, and the location of the access mechanism of the affected disk drive when the instruction is issued. The most significant factor is the distance the read head (access mechanism) must travel to the record being accessed. Performance is significantly improved if this distance is minimized by sequential accessing of data sets, and by separating active data sets onto separate drives. Sample timings for disk read operations are:

Read with relative record number	.06-4.8 seconds/record
Read with key	.5-6.0 seconds*/record
Sequential	5 seconds/track

* Assumes that one index entry is assigned for each track.

Number of Fields	Size of Fields	Use Buffer?	Execution Time (ms)
0	--	Yes	26.1
1	4	Yes	27.6
1	16	Yes	28.8
2	16	Yes	31.8
8	8	Yes	52.2
8	8	No	42.0

Figure 43. Execution Time for WRFM Instruction

Number of Fields	Size of Fields	Execution Time (ms)
1	4	10.8
8	4	30.6
8	8	38.4
32	4	99.0
64	2	179.4
128	1	364.8

Figure 44. Execution Time for REFM Instruction

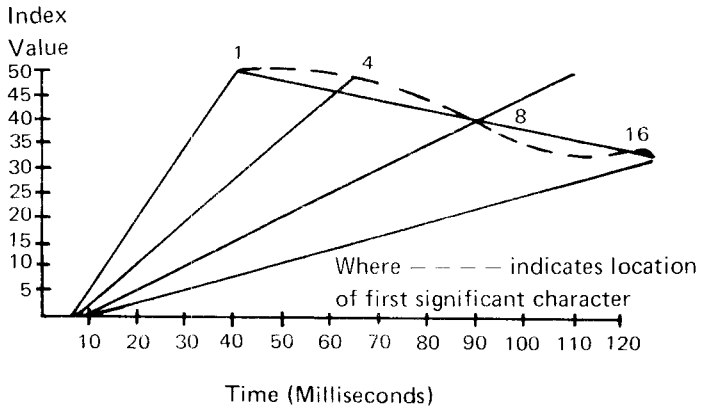


Figure 45. Execution Time Chart for TBFX/TBFN Instructions

Figure 45 can be used for estimating execution times of TBFX and TBFN instructions. The chart shows the times for the TBFX instruction and should be increased by 0.5 milliseconds for TBFN. The only other difference between the TBFX and TBFN is in the situation where the search argument is not in the table. In that case, the TBFX execution time is determined by the size of the table, and the TBFN is determined by the location, within the table, of the element which is higher in sequence than the search argument (if this element exists).

This chart is the result of timings of table operations on specific tables. To use it for estimating the execution time for a TBFX instruction for your table, first estimate the average index which the TBFX instruction will return. This gives the range for your particular table. For example, if your table has 50 entries uniformly distributed, then the average value of the index is 25. From the chart, this gives a range of 22 to 98 milliseconds for finding an entry in the table if the elements are 16 characters each, or a range of 22 to 37.5 milliseconds, if the elements of the table are four characters long.

The range for 16-character elements if the search argument is not found is 37 to 180 milliseconds.

For a more accurate estimate, it is necessary to determine the average location of the first mismatch (first significant character) during a compare. For example, a table made up of the numbers from 1101 to 1150 (four character elements) would have the mismatch on the third comparison, with 3.2 as the average. You can use this information to estimate the average execution time for a TBFX instruction for this table to be 35 milliseconds.

Index Number	Instruction Type	Character/ Element	Execution Time (ms)
1	TBWT	1	3.6
1	TBWT	16	6.3
25	TBWT	1	8.4
25	TBWT	16	10.8
50	TBWT	1	13.2
50	TBWT	16	15.6
1	TBRD	1	4.8
1	TBRD	16	6.3
25	TBRD	1	9.3
25	TBRD	16	10.8
50	TBRD	1	14.1
50	TBRD	16	15.6

Figure 46. Execution Time for TBWT/TBRD Instructions

The performance numbers in this section should only be used for estimating program execution times. Because of all the variables involved, a 10% variation from the projected timings could be the case. The timings presented assume no overlapped I/O is in process, and that the machine check indicator (160) is not on. Either of these conditions can cause longer execution times for some instructions, but the machine check indicator increases the execution of all instructions by 4%.

Chapter 4. 3741 Operation

The 3741 Models 3 and 4 Programmable Work Station has the same data station functions as the 3741 Models 1 and 2 Data Station unless the work station is under control of an ACL program. The 3741 Model 3 has the same standard functions and available features as the 3741 Model 1. The 3741 Model 4 has the same standard functions and available features as the 3741 Model 2. For a detailed description of the 3741 standard mode of operation, see the *IBM 3741 Data Station Operator's Guide,* GA21-9131. For a detailed description of the 3741 standard functions and available features, see the *IBM 3741 Data Station Reference Manual,* GA21-9183.

ACL programs for the work station can only be loaded from disk. The translator feature is available to translate ACL source programs into object programs. To gain access to the translator feature or the ACL program execution mode, the work station must first be in index (X) mode. It is convenient for the operator to key a function select delete sequence in the index (X) mode to blank the screen before keying the setup parameters.

The following sections contain details on translation, execution, and program debugging.

INITIATING TRANSLATION WITH THE LABEL PROCESSOR

The label processor disk (included with work stations that have the translator feature) must be used with ACL programs that have labels preceding source statements. The label processor associates the labels with step numbers for internal processing. After the labels have been associated with step numbers, the label processor sends the ACL program into translation. Label processor operation occurs in three passes made possible by an overlay structure. The program code for the base pass and pass 1 is loaded at program load time. After pass 1 is complete, the object code for pass 2 is transferred from disk to overlay the code used for pass 1. The same is true for pass 3 code, which is transferred from disk to overlay pass 2 code after pass 2 is complete.

Base Pass

The base pass of the label processor controls the execution of the three passes of label processor operation. The base pass also loads the object code for passes 2 and 3.

Pass 1

The first pass of label processor operation reads the ACL label processor input file and generates a cross-reference work file for later use in sorting and printing the cross-reference listing. Pass 1 of the label processor then checks the four character executable instruction labels, and tables all defined labels for subsequent association of labels with step numbers.

Pass 2

The second pass of the label processor reads the ACL label processor input file and writes the translator input file. During the one-for-one transfer of records, any ORG functions are written as deleted records. Label processor pass 2 then prints the ACL label processor input file (when the AUTO DUP/SKIP switch is on). This pass of operation then replaces labels with corresponding step numbers and prints all invalid, unresolved, or duplicate label error messages.

Pass 3

The third pass of the label processor sorts the defined labels, links labels with corresponding references, prints the cross-reference listing, and links to the translator.

Label Processor Input Data Set

The first record of the ACL label processor input data set must be a .NAME control statement, and cannot be a deleted record. *The ACL label processor input data set (source) name cannot be blank.* Columns 63-70 of the .NAME statement contain the label processor output data set name (translator input data set). Default is TRANSLAT. Column 71 of the .NAME statement contains the drive number for the label processor output data set. Default is drive 2. Columns 73-80 of the .NAME statement contain the translator output data set name (object data set). Column 81 contains the drive number for this data set. Default is drive 1. If columns 73-80 are blank, the ACL translator is not selected upon completion of the label processor.

The label processor disk contains the object program (four tracks), a temporary work data set (four tracks) called TEMPDATA, and output data sets to be defined by the programmer (65 tracks). The output data sets are also input for the translator. Note that two data set names are reserved and cannot be used by the programmer. These are the TEMPDATA (temporary work data set) and SYMBOLIC (which contains the object code for passes 2 and 3). Any attempt to use these data set names may result in an error, and label processor results are invalid.

The label processor assigns files in the following manner:

- File 1 contains the ACL label processor input data set, which is required through pass 2.
- File 2 contains the TEMPDATA temporary work file, which is required for passes 1 and 3. This file must be allocated four tracks.
- File 3 contains the label processor output data set, which is required for passes 2 and 3. This data set is defined in columns 63-70 of the .NAME control statement.
- File 4 contains the SYMBOLIC data set, which is required for all three passes. This data set contains the object code for passes 2 and 3. The ACL label processor configurator program is used to merge the object code for the three passes into the SYMBOLIC data set. All printed headings and error messages are included immediately after the object code.

  A recommended data set allocation is shown below.

- Drive 1 contains the ACL label processor input data set and the translator output data set. These can be on the same disk or on separate disks. The ACL label processor input data set is required through pass 2.
- Drive 2 contains the ACL label processor object code (four tracks), the temporary work data set (TEMPDATA – four tracks), and output data sets (65 tracks).

After an ACL program with labeled statements has been through the label processor, it may be changed without going through the label processor again. This is done by changing the contents in the data sets used for the label processor output. After the label processor output data set has been changed, translation is initiated the same way as translation without using the label processor.

The following steps are required when using the label processor disk:

1. While the label processor is in operation, the printed output can be controlled by setting the three switches to the following positions:

   AUTO DUP/SKIP Switch

   ON – prints source statements and cross-reference (error messages are always printed). Printing only occurs when this switch is on.

   OFF – suppresses printing of source statements and cross-reference.

   This switch may be set on and off during printing.

   PROG NUM SHIFT Switch

   ON – suppresses generation of cross-reference work file.

   OFF – generates cross-reference work file.

   This switch must be set at the start of the program.

   REC ADV Switch

   This switch should be off. It has no effect on the label processor, but controls the translator listing.

2. Load the disk with the ACL program into disk drive 1. The object program may or may not be on the same disk. If the object program is not on the same disk, an error code (511) is displayed when the label processor is finished. The operator should then insert the disk with the object program data set and press RESET to continue. The label processor runs faster if the label processor input and output data sets are on different drives.
3. Load label processor disk into disk drive 2.
4. Blank display screen by pressing FUNCT SEL lower and DELETE REC.

5. Key in the following parameters.

Columns	Entry
1-8	SYMBOLIC
9	2
11-14	LABL

6. Press FUNCT SEL upper and E.

7. Display message tells the operator to key in the data, then press RIGHT ADJ.

8. Display message tells the operator to key in the source data set name, then press RIGHT ADJ.

The following information will be displayed as the label processor goes through its three passes.

PASS 1:

- Line 1 contains ACL LABEL TRANSLATOR.
- Lines 2, 3, and 4 contain the first 120 characters of the records as they are read from the input data set.
- Line 5 contains REQUEST INPUT DATA SET AND DISK DRIVE NUMBER.
- The bottom line contains PASS-1.

PASS 2:

- Lines 2, 3, and 4 contain the first 120 characters of the records as they are read from the input data set.
- The bottom line contains PASS-2.

PASS 3:

- Lines 2, 3, and 4 contain the cross-reference listing as it is being printed, followed by any labels which are in error.
- The bottom line contains PASS-3.

Label Processor Output

Label processor printer output includes:

1. Source listing of statements and instructions.
2. Cross reference.
3. Error messages (duplicate labels, undefined and/or invalid labels, and omitted labels).

Source Listing

The first page of the source listing contains the heading (ACL SYMBOLIC LABEL PROCESSOR 3741 MODELS 3 AND 4), the file name, the date, and the page number. Comment statements and control statements are printed beginning in position 19, with the last 18 positions of the input record truncated. The disk address of the source input record is printed in positions 1-5. The record sequence number within the input data set is printed in positions 8-11. Definition records for .FORMAT and .BUFFER are printed intact beginning in position 1. Executable instructions are printed beginning in position 19, with the last 18 positions of the input record truncated. The disk address of the source input record is printed in positions 1-5. The record sequence number in the source data set is printed in positions 8-11. The generated instruction number is printed in positions 14-16. Figure 47 shows a sample source listing printed for *Sample Program 2-Mailing List Inquiry* in Appendix D.

Cross Reference

At the start of the cross-reference listing, the heading CROSS-REFERENCE is printed. The symbolic label is printed in positions 1-4. The record sequence number (where the label is defined) is printed in positions 7-10, followed by the record sequence numbers of where the label is used. If the number of references exceeds a printed line, another line is started beginning under the referenced column. Control statements with label fields are also referenced. Figure 48 shows a sample cross reference printed for the *Sample Program 2-Mailing List Inquiry* in Appendix D.

```
A C L    SYMBOLIC LABEL TRANSLATOR        3741 MODELS 2 AND 4    RF        DEMO              DATE: 10-21-74                PAGE-01-

13002  0001          .NAME          KEY   2    4                                                        DEMOBJCT
13003  0002          ***
13004  0003          .DATASET       1     MAILIST   L28  1    5                     KU    10    1    10    1
13005  0004          ***
13006  0005          .FORMAT        1                        READ AND WRITE DISKETTE FORMAT
HHHHHHHHHHIIIIIIIIIIIIIIIIIIJJJJJJJJJJJLLLLFFFFFFBBBBBBBBBBBBBBBBBCCCCCCCCCCCCCCCDDDDDDDDDDDDDDDEEEEE
13008  0007          .FORMAT        2                        READ AND WRITE DISPLAY FORMAT
IIIIIIIIIIIIIIIIIIJJJJJJJJJJJJ       FFFFFFBBBBBBBBBBBBBBBBB                       CCCCCCCCCCCCCCCCCDDDDDDDDDDDDDDD      EEEEE
13010  0009          ***
13011  0010          .FIELD  1    3        D10   RE2   A   1 CUSTOMER NO.*
13012  0011          .FIELD  2    3        A16    L    041T STREET*
13013  0012          .FIELD  3    3        A16    I    081T CITY*
13014  0013          .FIELD  4    3        A15    I    097T COUNTRY OR STATE*
13015  0014          .FIELD  5    3        D5     LZ   116T ZIP CODE*
13016  0015          .FIELD  6    3        D07    I    034T TELEPHONE*
13017  0016          .FIELD  7    3        A01   R     G    CUSTOMER NOT FOUND R=REKEY*
13018  0017          ***
13019  0018  000   BEGN    WRFM 2                      CLEAR LINES 2,3, AND 4 OF DISPLAY
13020  0019  001           Y    =    +    3            SET BUFFER NUMBER = 3
13021  0020  002           Z    =    +    1            SET MESSAGE NUMBER = 1
13022  0021  003           ENTR Y    Z                 ENTER CUSTOMER NUMBER
13023  0022  004           IFIR 212  IS   ON    END    IF INDEX GO TO END OF JOB
13024  0023  005           READ 1    1    A            SEARCH DATASET FOR CUSTOMER RECORD
13025  0024  006           IFIR 225  IS   ON    ERR    IF NO RECORDS FOUND - ERROR MESSAGE
13026  0025  007           WRFM 2    2                 DISPLAY RECORD
14001  0026          ***
14002  0027  008   ADD     Z    =    Z    +     1      ADD ONE TO MESSAGE NUMBER
14003  0028          ***
14004  0029  009   NEXT    ENTR Y    Z                 ENTER STREET, CITY, STATE, ZIP, AND TELEPHONE
14005  0030  010           IFIR 198  IS   ON    BACK FIELD BACKSPACE
14006  0031  011           IFIR 199  IS   ON    SKIP RECORD ADVANCE
14007  0032  012           IFIR 185  IS   ON    NEXT SELECT PROGRAM? RESTORE SAME MESSAGE
14008  0033  013           IFIR 186  IS   ON    NEXT DUP? RESTORE SAME MESSAGE
14009  0034  014           IFIR 190  IS   ON    NEXT RECORD BACKSPACE? RESTORE SAME MESSAGE
14010  0035          ***
14011  0036  015           IFD  Z    =    6     SKIP LAST MESSAGE? UPDATE DISKETTE RECORD
14012  0037  016           GOTO             ADD
14013  0038          ***
14014  0039  017   BACK    Z    =    Z    -     1
14015  0040  018           IFD  Z    =    1     BEGN FIRST MESSAGE?
14016  0041  019           GOTO             NEXT
14017  0042          ***
14018  0043  020   SKIP    REFM 2    2                 READ DISPLAY FOR UPDATE RECORD
14019  0044  021           WRT  1    1                 WRITE UPDATED DISK RECORD
14020  0045  022           GOTO                  BEGN
14021  0046          ***
14022  0047  023   ERR     ENTR 3    7                 INVALID CUST MESSAGE
14023  0048  024           GOTO                  BEGN
14024  0049          ***
14025  0050  025   END     EXIT
14026  0051          .END
**************** NUMBER OF ERRORS =
```

Figure 47. Source Listing for Sample Program 2

```
                    C R O S S  -  R E F E R E N C E

ADD     0027  0037
BACK    0039  0030
BEGN    0018  0040  0045  0048
END     0050  0022
ERR     0047  0024
NEXT    0029  0032  0033  0034  0041
SKIP    0043  0031  0036
```

Figure 48. Cross Reference Printing for Sample Program 2

Label Processor Error Messages

- An error message is printed prior to any source records if more than 159 (4K) or 319 (8K) labels are defined, or more than 1584 references are made to these labels.
- An error listing of the invalid, unresolved, or duplicate labels is printed. The error is listed before the statement in error, or at the beginning of the error listing.
- An error message is displayed if the first record of the ACL source program is not a .NAME control statement. The operator can press the RESET key to exit the label processor.

During pass 1 of the label processor, the printed error listing contains the record in error, followed by the error message (five asterisks preceding the relative record number of the record in error), and one of the following headings:

- *STATEMENT NUMBER IN ORG STATEMENT NON-NUMERIC* – this heading indicates an ORG instruction in error (ORG is ignored). The instruction number in an ORG is nonnumeric.
- *NUMBER OF LABEL REFERENCES EXCEEDS SYMBOL TABLE SIZE* – this heading indicates reference symbol table overflow. The reference is not included in the cross reference or replaced with an instruction number.
- *NUMBER OF DEFINED LABELS EXCEEDS SYMBOL TABLE SIZE* – this heading indicates label table overflow. The label is undefined to the program.
- *INSTRUCTION NOT DEFINED* – this heading indicates that the label has an instruction mnemonic (PRT, for example) that is not defined for ACL.

During pass 2, the following errors are detected. If the AUTO DUP/SKIP switch is off, the record in error is not printed before the error message. Errors are identified by the following headings:

- *DUPLICATE LABEL* – this heading indicates a label defined more than once in a program. All of the multiple definitions of the label are printed after the heading.
- *INVALID LABEL* – this heading indicates that labels are undefined and/or invalid because they violate syntax rules. All such labels and their references are listed after the heading.
- *OMITTED OR INVALID REFERENCES* – this heading indicates that an instruction requiring a label has a blank label field.
- *NUMBER OF ERRORS = NNNN* – this heading is printed at the end of pass 2 to indicate the total number of errors detected.

Note that, because the label processor phase of translation is a program, execution errors could also occur during translation (Appendix C).

ACL Label Processor Configurator

In order to tailor the ACL label processor to your particular system configuration, a disk is provided at installation time to record the following parameters:

- Printer type
- Printer forms control
  - Number of characters per line
  - Number of lines per page
  - Number of printed lines per page
- Keyboard
  - Standard 3741 or proof keyboard
  - Language group

The procedure below must be followed at system installation time in order to build a label processor disk.

1. Insert the translator diagnostic diskette in drive 1.

2. Insert a blank disk in drive 2. (A blank disk has been initialized, its label has been recorded at sector 08, it has extents defining the entire disk, it is a null data set and sectors 09-26 on the index track are deleted.)

3. Press FUNC SEL upper and E. A message is displayed to verify that the configurator program is loaded and the disk in drive 2 is being checked for the following:

   Sector 08
   BOE = 01001
   EOD = 01001
   EOE = 73026
   The data set is not write-protected or secure.

   Sector 07 is a VOL1.

   The buzzer also sounds, and the system waits for the RIGHT ADJ key to be pressed before continuing.

   The remaining sectors (09-26) on the index track do not define a data set with extents overlapping those of sector 08.

4. If any of these conditions are violated, the data set label defined by sector 08 is displayed on lines 2, 3, and 4 of the display screen. The operator can press the RIGHT ADJ key to continue, or press RESET to go to job completion system code 100.

5. If these checks are completed, sectors 08-26 of the index track are written with the following:

   Sector 08

HDR1	TRANSLAT	128	09001	73026
				09001

   Sector 09

HDR1	TEMPDATA	128	05001	08026
				05001

   Sector 10

HDR1	SYMBOLIC	128	01001	04026
P				01001

   Sectors 11-26 (these sectors are written deleted)

DDR1	DATAXX	128	74001	73026
				74001

   (XX = corresponding sector number.)

After the index track is created, messages are displayed requesting system configuration parameters. These parameters are:

- Keyboard = STANDARD (RIGHT ADJ) or PROOF (P).
- Machine size = 4 (4K) or 8 (8K)
- Keyboard language group
  - 0 = United States, United Kingdom, France (Querty), Italy, Germany (U.S. graphics), and Japan (English nomenclature).
  - 1 = Norway
  - 2 = Sweden
  - 3 = Denmark
  - 4 = Germany (German graphics)
  - 5 = Spain/Latin America
  - 6 = Belgium and France (Azerty)
  - 7 = Portuguese
  - 8 = Katakana
  - 9 = Brazil
- Printer type
  - 1 = 3713 Printer
  - 2 = 3717 Printer
  - 3 = 3715 (bidirectional floating margin)
- Printer forms control (Right-adjusted, unless overridden by entry of C). Entry is a three-digit number with leading zeros, if required).
  - Number of characters per printed line (4-128)
  - Number of printed lines per page (7-127)
  - Number of lines per page (7-127)

The object code from three data sets is combined into one data set (SYMBOLIC) to allow execution of the ACL label processor. Sector 26 of track 1 identifies the level of the ACL label processor plus the selected system configuration parameters. Sector 26 of track 1 is:

Position	Description
1-8	Revision date-MM-DD-YY (where MM = month, DD = day, and YY = year)
9	Printer type
10-12	Maximum number of characters per line minus one
13-15	Maximum number of printed lines per page
16-18	Maximum number of lines per page
19	0 = Standard keyboard
20-21	Keyboard language group
22	Machine size (4 or 8K)
23-128	Blank-Reserved

After the ACL label processor disk is built, a message is displayed requesting one of two responses. RIGHT ADJ ends the job normally (100 is posted on the display), or D creates a link to program execution, which invokes the ACL label processor.

INITIATING TRANSLATION WITHOUT LABEL PROCESSOR

The translator feature converts program source statements into object code. Object programs must be contained in data sets with a logical record length of 128 characters; source programs must be in data sets with a record length of 80 characters or more. The object data set must be at least two tracks long and start on a track boundary. The source and object data sets may be on either the same or separate disks.

To initiate the translator, the operator may place the disk(s) in either disk drive 1 or 2. With the work station in the index (X) mode, the operator should:

1. Set the AUTO REC ADV switch on to print the translator input data set.

2. Blank the display screen by pressing FUNCT SEL lower and DELETE REC.

3. Key in the following parameters.

Columns	Entry
1-8	Source data set name. (If the operator keys the wrong data set name, the translator must be aborted and the job restarted.)
9	1 or 2 depending on the disk drive the source file is mounted on. Default is disk drive 1.
11-18	Object data set name.
19	1 or 2 depending on the disk drive the object file is mounted on. Default is disk drive 1.

Note: The source or object data set name may be left blank when initiating translation, but both cannot be blank. The first data set, on the disk, with a blank name is then used for the data set. If both source and object data set names are left blank, an error code (10) is posted on the display. This error code indicates that the source and object data sets are the same data set. The operator must press ALPHA and NUM SHIFT with RESET to go to index (X) mode in this situation.

4. Press FUNCT SEL upper and A to initiate translation. The object file is checked for a duplicate program name as specified in the .NAME control statement. If a duplicate name is found, a warning message (19) is displayed. To write the new program over the old object program, press RESET. If the new object program is not to replace the old object program, press ALPHA and NUM SHIFT with RESET to return to index (X) mode.

PROGRAM EXECUTION

To execute an ACL object program the operator may place the disk in either disk drive 1 or 2. With the work station in the index (X) mode the operator should:

1. Blank the display screen by pressing FUNCT SEL lower and DELETE REC.

2. Key in the following parameters.

Column	Entry
1-8	Object code data set name
9	1 or 2 depending on which drive contains the disk Default is disk drive 1
11-14	The program name as it appears in the .NAME control statement of the program

3. Press FUNCT SEL upper and E to initiate execution of the ACL program.

After the program has been completed, a job completion-system code (100) is displayed. The operator must then press RESET to place the 3741 in index (X) mode.

Communications

See *Communications* in Chapter 2.

PROGRAM DEBUGGING

To aid program debugging, select from the following program trace capabilities:

- Step trace
- Register trace
- Step and register trace
- Step stop
- Single step trace

Note that using traces causes the printer to revert to non-overlap mode.

Step Trace

Step trace provides the ability to follow the flow of the program while it is being executed. This is accomplished by displaying to the programmer the step number every time the sequence of execution is altered from sequential.

After the execution of a GOTO, RGO or any IF instruction that takes the nonsequential path, or any special exits taken from I/O control operations, the step number of the instruction where control is to be transferred is displayed. Then, after display is completed, the step number that has just been displayed is executed.

Register Trace

Register trace provides the capability to display the contents of a register each time a step that changes a register (except for I/O type steps and GSCK) is executed.

The register trace outputs the contents of the changed register following execution of: Add, subtract, multiply, divide, immediate, shift right, shift left, TBRD, GETB, LOAD, ENTR (to a register), MVER, ZONE, and MOFF. The result register is displayed after execution of the step is completed, and before the next sequential step is started.

Step Stop

Step stop provides a means to stop at a specified step number or instruction. After initiating the step stop, a + is displayed in position 10 of line 1 of the display screen followed by the step number and the program executes until this step number is reached. When the specified step number is reached the + changes to a - and program execution stops before that instruction is executed. To continue, press NUM SHIFT and RESET.

Single Step Trace

Single step trace provides the capability to step through the program one step at a time. This trace is used in conjunction with the step and register trace to monitor the machine while it steps through the program. Output must be to the display. Single step trace to the display requires the NUM SHIFT and RESET keys to execute the next instruction.

Trace Output

The step, register, step and register, or single step trace can be output to the display or the printer. A .PRINTER statement is required to output the trace to the printer. Step stop is output to the display as described previously.

The output from trace is displayed on the screen beginning in position 10 of line 1. If the output is to the display only, execution is halted until the NUM SHIFT and RESET keys are pressed. Pressing any other key advances the step and also performs the function of the key. If the output is to the printer, execution is halted until all the characters are printed. Execution is then resumed automatically.

The output forms of the step trace and register trace are:

- Snnn
- Rnnn aaaaaaaaaaaaaaa
- Rnnn

The S indicates a step trace, and nnn indicates the next step number to be executed. The R indicates that the output is a register trace, the nnn is the associated step number, and the a's are the contents of the register being traced. Rnnn is output in single step mode for steps not included in register trace.

Selecting Trace

The trace can be turned on or off from the keyboard any time during program execution. It cannot be started until after the program is loaded. No change is required in the source program or translation to activate the trace, except a .PRINTER statement is required to output the trace to the printer (see *Printer Operations* under *Programming Restrictions* in Chapter 3).

To Turn On Trace:

1. Hold down ALPHA and NUM SHIFT and press FUNCT SEL lower.

2. For step stop trace, key nnn, where nnn is a step number, and n is any number 0-9 (a numeric keyed in the first position specifies step stop). If NUM SHIFT and RESET are pressed after the program is halted at the desired step, the program continues until the specified step number is encountered again.

3. For the remainder of the trace functions (single step, step and/or register), key Tnm, where:
   - n = 0 for output to the display screen only.
   - n = 4 for output to printer and display screen.
   - m = 1 for trace of transfer steps only.
   - m = 2 for trace register changes only.
   - m = 3 for trace steps and registers (1 and 2 above).
   - m = 4 for trace all steps not included in 1 and 2 above.
   - m = 5 for trace combining 1 and 4 above.
   - m = 6 for trace combining 2 and 4 above.
   - m = 7 for trace all steps in step stop mode (1, 2, and 4 above).

   Note: The NUM SHIFT key must be pressed while keying n and m. If NUM SHIFT is not pressed, the process must be repeated from step 1.

To Turn Off Trace:

1. Hold down ALPHA and NUM SHIFT keys and press FUNCT SEL lower.

2. Key T10.

Program Restart

It is possible to intermix work station program operation and base 3741 operations, if the necessary coding has been introduced in the program. To interrupt the ACL program, use the checkpoint (CKPT) instruction (see *Instructions* in Chapter 2).

To restart the program the operator must:

1. Blank the display screen by pressing FUNCT SEL lower and DELETE REC.

2. Key in the following parameters.

Columns	Entry
1-8	Checkpoint program data set name
9	1 or 2 depending on which disk drive contains the disk Default is disk drive 1
11-14	Checkpoint program name

3. Press FUNCT SEL upper and E.

The restart program loads the checkpoint program and opens all data sets which were open at the time of the checkpoint statement execution. All data sets are positioned at the point they were when the CKPT was initiated. The checkpoint program name is displayed in positions 37-40 of the status line. Execution is returned to the sequential step following the CKPT instruction.

A 5XA (X = the data set number) error is displayed if a CKPT instruction is issued on a write protected file.

Customer Diagnostic Diskette

Each work station is provided with a diagnostic diskette which can be used to isolate the cause of an error to either the program or the execution hardware. A second diskette is provided for work stations with the translator feature. This second diskette is updated by the label processor configurator to conform to the particular system. The diagnostic diskettes contain programs that should be run before you call for IBM service assistance. These programs exercise the work station hardware. At the completion of successful execution of these programs, a 100 message (job completed) is posted on the display. This indicates that the hardware is functioning properly and that the cause of the error may be in the program. Both diagnostic program diskettes must be configured (only once) to the type of printer, keyboard, and storage attached to the work station. The procedure for configuration is:

1. Insert one of the diagnostic diskettes in drive 1.

2. Advance to data set label 10.

3. Press FUNCT SEL upper and E.

4. Respond to prompted data entry requests.

When this procedure is completed, data set label 7 is updated as a summary of the diagnostic configuration, and may be used as a quick reference to proper configuration. Note that configuration is necessary only once.

To process the execution hardware diagnostic:

1. Insert the execution diagnostic diskette in drive 1.
2. Check data set label 7 for proper configuration.
3. If the proper configuration is displayed, backspace to data set label 6.
4. Press FUNC SEL upper and E.

With a 3717 Printer attached, this program runs for approximately seven minutes. With a 3713 or 3715 Printer attached, this program runs for approximately nine minutes. With no printer attached, program run time is approximately three minutes. The 100 message is displayed after a successful program run.

To execute the translator diagnostic:

1. Insert the translator diagnostic diskette in drive 1.
2. Check data set label 7 for proper configuration.
3. If the proper configuration is displayed, record backspace to data set label 6.
4. Press FUNC SEL upper and A.

With a 3717 Printer attached, this diagnostic runs for approximately five minutes. With a 3713 or 3715 Printer, the program runs for approximately seven minutes. The 100 message is displayed after a successful program run.

STORAGE DUMPS

All or part of the 4K/8K storage may be output to the display, the printer, or to disk. If terminal program errors are encountered, select a dump option, or abort the job (all data sets are closed and the 100 message is posted). After completion of the selected dump option, control returns to index mode, no files are closed, and the 100 message is not posted. If you select a dump option during normal execution, control returns to the interrupted point in the program after completion of the dump. The four storage dumps are:

- Unformatted display dump.
- Formatted display dump.
- Printer dump.
- Disk dump.

To initiate a storage dump any time during program execution:

- Press FUNC SEL lower with ALPHA and NUMERIC SHIFT.
- Key Dkn, where:

  D = the dump function.

  k = 0, 1, 2, 3, (4K) or up to 7 (8K) indicating the starting 1K buffer address of the dump.

  n = 0 to specify the unformatted display dump.

  n = 1, 2, 3, 4 (4K) or up to 8 (8K) indicating the number of 1K increments to dump to the printer.

  n = 9 to specify the formatted display dump.

  kn = 99 to specify the disk dump.

- The NUM SHIFT key must be pressed while keying k and n. If it is not pressed, the process must be repeated from FUNC SEL lower with ALPHA and NUM SHIFT.

Note the functions of the following keys:

- Any key advances the display dump.
- The ALPHA SHIFT key and any key backspaces the display dump.
- The ALPHA and NUMERIC SHIFT keys and any key terminate dump operations.
- The ALPHA and NUMERIC SHIFT and the RESET key (twice) aborts the job.

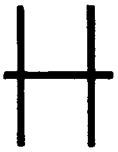
All display dumps will wrap around storage contents if you advance beyond the last buffer (the first buffer is displayed) or backspace past the first buffer (the last buffer is displayed).

Unformatted Display Dump

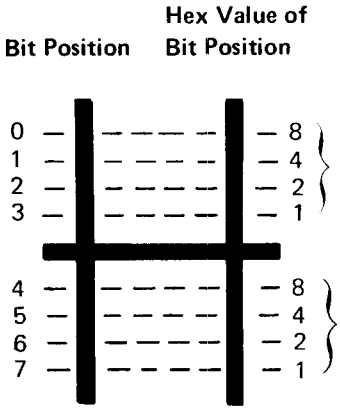
The unformatted display dump sets the display address to the specified starting address. The first 120 characters of the 128-byte buffer are displayed (in hexadecimal) in lines 2, 3, and 4 of the display. The last eight bytes of the buffer are displayed in hexadecimal starting in position 10 of line 1. By pressing any key, the display is advanced by 128 bytes. By pressing FUNC SEL upper and ALPHA and NUMERIC SHIFT, the dump mode is exited and the program is returned to program execution (at the point where the dump is initiated), leaving the display at the address last displayed.

Hexadecimal Display

When hex data is keyed or displayed, not all of the data represents displayable characters such as an A B C or 0 1 2 3. When the hex data keyed is not a displayable character, a hexadecimal display is provided that represents the eight bit code for the data. The basic display (no bits on) looks like this:



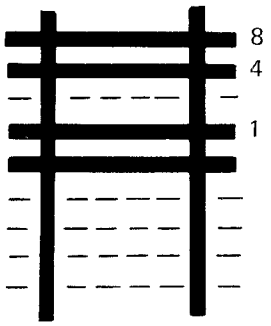
Other lines are added to the display for each bit that is on in the EBCDIC code.



Add these values together to get the first hex digit. Only add those values together for the lines displayed.

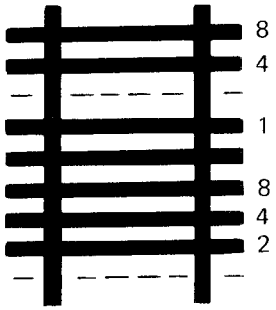
Add these values together to get the second hex digit. Only add those values together for the lines displayed.

Example: Assume the display looks like this:



The first hex digit is 8 + 4 + 1 = D and the second hex digit is a 0. The hexadecimal value is hex D0.

Assume another display looks like this:



The first hex digit is 8 + 4 + 1 = D and the second hex digit is 8 + 4 + 2 = E. The hexadecimal value is hex DE.

This procedure allows you to observe the dynamic state of any buffer. The eight bytes displayed on line 1, however, are not dynamically updated. By pressing any key (except FUNC SEL upper) with ALPHA and NUMERIC SHIFT, the dump mode is exited and the display address is restored to buffer 2. In this case, the eight bytes in line 1 are blanked. By pressing ALPHA SHIFT with any key, the display is backspaced 128 bytes.

During any storage dump to the display, a specific register or indicator may be addressed. Figure 49 shows the actual storage address of each register and the object code equivalent for each register.

Indicators are located in storage in the address range of 0120 to 013F. This 32-byte area of storage contains 256 bits, each bit corresponding to an indicator. Each byte, thus contains eight indicators. For example, in order to address indicator 72, display byte 0128.

Source Reference	Object Code	Actual Address	Source Reference	Object Code	Actual Address
A	70	0070	N	B1	01B0
B	80	0080	O	C1	01C0
C	90	0090	P	D1	01D0
D	A0	00A0	Q	E1	01E0
E	B0	00B0	R	F1	01F0
F	C0	00C0	S	82	0280
G	D0	00D0	T	92	0290
H	E0	00E0	U	A2	02A0
I	F0	00F0	V	B2	02B0
J	71	0170	W	C2	02C0
K	81	0180	X	D2	02D0
L	91	0190	Y	E2	02E0
M	A1	01A0	Z	F2	02F0

Figure 49. Register Addresses in Storage

Formatted Display Dump

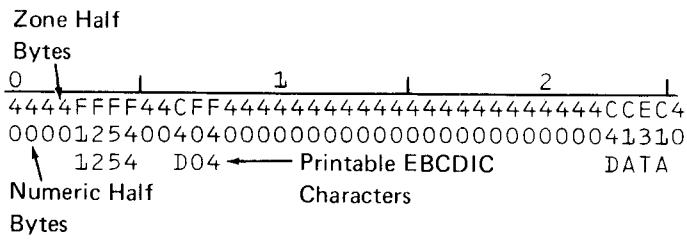
The formatted display dump displays 32 characters as follows:

- Line 2 – Zone half bytes
- Line 3 – Numeric half bytes
- Line 4 – Displayable EBCDIC character or byte display

The first display shows the first 32 bytes of the specified starting location. Press any key to advance the display by 32 bytes. Press ALPHA SHIFT with any key to backspace the display 32 bytes. A counter on line 1 of the display shows the number of increments keyed. Press ALPHA and NUMERIC SHIFT and any key to exit the dump mode. Note that the display buffer (buffer 2) is not restored to its original contents.

Printer Dump

The printer dump formats 128 bytes across the page in three lines as follows:



The print line is terminated when remaining bytes of the 128-byte segment are the same as the last printed byte. Printline length is assumed to be 128. When a printer dump is to be initiated, the executing program must contain a .PRINTER control statement (see *Printer Operations* under *Programming Restrictions* in Chapter 3). If a .PRINTER control statement is not included, this function defaults to a formatted display dump.

When the printer dump is completed, the work station returns to its state at the time the dump was initiated. Printed register and indicator addresses are shown in Figure 49.

Disk Dump

The disk dump function provides IBM service personnel with the ability to write to disk the entire contents of the work station storage. This data can then be printed or displayed for analysis. Although you need not be concerned with the analysis of the stoage dump to disk, debugging time can be saved by initiating the disk dump function before IBM service personnel arrive.

The disk dump function can be initiated at any time during program execution by the following procedure:

1. Press ALPHA and NUM SHIFT, followed by the FUNC SEL lower key. This sequence interrupts the current program for subsequent restart.

2. Press D99. This sequence specifies the disk dump. The system saves the volume name of the diskette in drive 1 (disk dumps are output only to drive 1), posts a DS message, and waits for the drive to go not-ready.

3. When drive 1 goes not-ready, a MS message is posted and the system waits for the customer diagnostic diskette to be inserted.

4. The customer diagnostic diskette has a volume name of %DMP66. When dumping to another scratch diskette, it must have a volume name of %DMPnn (where nn is a number from 1-71). If the volume name is incorrect, a DS message is posted, and steps 2 and 3 must be repeated. The numerics in the volume name indicate the diskette track at which the dump is to start. The diagnostic diskette has nine tracks (66-74) reserved for three dumps. If the initial seek to the dump area fails, a DC3 error is posted.

5. The image of the buffers in storage is written into three tracks of the %DMPnn diskette beginning at the nn track. The system internally records the number of dumps taken in the job. If less than three tracks are available for the next dump, the internal count is reset and the next dump overlays the first dump at track nn. The internal count is reset at each start of job, and is saved on checkpoint/restart. The VOL label is never updated. Buffer contents (128 characters) are written to disk beginning in sector 1 of the track and continuing for 32 or 64 consecutive tracks.

   A ready message (RS) is posted during the dump. If disk write errors occur during the dump, an RS5 message is posted, the error line is turned on, and the dump continues.

6. When the dump is completed, a DC message is posted and the system waits for the drive to go not-ready (indicating removal of the dump diskette). If the drive goes not-ready during the dump, no errors are posted, the internal dump count is not incremented, and the dump is considered complete.

7. If the drive is ready when the dump is initiated (step 2), an MC message is posted. Insert the original program diskette. The system checks the volume name against the name stored (step 2). If the names match, the message area is blanked and execution continues from the interrupted point (step 1). If the names do not match, a DC message is again posted.

Appendix A. Indicators

INDICATOR	DEFINITION	SET ON BY	SET OFF BY
1-99	User-specified	User program	User program
100-146	Reserved		
147	Printer error	Any printer error	User program
148	Printer page overflow	Printer reaches overflow line specified in columns 23-25 of .PRINTER	Next PRNT instruction
149	Card I/O end of data set	When ?/ terminating card is read	User program
150	Card I/O end of job	When ?* terminating card is read	User program
151-153	Reserved		
154	Invalid GSCK result in modulus 11	GSCK result is 10	User program
155	RBLK/WBLK overflow	Low-order three bytes of the register are 000	User program
156	Table index error	Attempt to read or write beyond the end of the table	User program
157	Division error	Attempt to divide by zero	User program
158	Multiply overflow	Carry results from multiply operation	User program

INDICATOR	DEFINITION	SET ON BY	SET OFF BY
159	Add/subtract overflow	Carry results out of high-order positions of factor 1 register	User program
160	Machine check	Arithmetic overflow indicators (155-159)	User program
161	Error line	All system errors causing keyboard lock or display screen flash	User program or RESET key
162	Short keyboard buzz	User program	User program
163	Table high entry found	Equal entry not found, but a higher entry is found.	User program or by the machine hardware if an equal entry is found.
164	Printer busy	Execution of any print command.	Completion of any print command or end of printing due to a print error
165	Disk drive busy	Disk busy	Disk not busy
166	AUTO REC ADV switch	Switch on	Switch off
167	PROG NUM SHIFT switch	Switch on	Switch off
168	AUTO DUP/SKIP switch	Switch on	Switch off
169-184	Reserved		

Keyboard Indicators

The keyboard is set to a locked status (indicator 197 off) until an ENTR instruction opens the keyboard for data entry, and sets indicator 197 on. After execution of the ENTR instruction, the keyboard is again locked and indicator 197 is turned off. When the keyboard is locked, only the keys which set keyboard indicators, special function keys, and the RESET key are active.

The following are keyboard indicators and their respective keys.

INDICATOR	DEFINITION	SET ON BY	SET OFF BY
185	SEL PGM	Pressing key when keyboard open (at ENTR instruction)	User program or next ENTR instruction
186	DUP	Pressing key when keyboard open (at ENTR instruction)	User program or next ENTR instruction
187	FIELD COR*	Pressing key	User program or pressing any asterisked key
188	NEW LINE*	Pressing key	User program or pressing any asterisked key
189	TAB*	Pressing key	User program or pressing any asterisked key
190	REC BKSP	Pressing key when keyboard open (at ENTR instruction)	User program or next ENTR instruction
191	CHAR ADV*	Pressing key	User program or pressing any asterisked key
192	RESET*	Pressing key	User program or pressing any asterisked key
193	FIELD ADV	Pressing key when keyboard open (at ENTR instruction)	User program or next ENTR instruction
194	SKIP	Pressing key to exit field	User program or next ENTR instruction
195	RIGHT ADJ	Pressing key to exit field	User program or next ENTR instruction
196	Negative right adj, dash (—)	Pressing key to exit field	User program or next ENTR instruction
197	Keyboard open/closed	Initiating ENTR instruction	Completion of ENTR instruction
198	FIELD BKSP	Pressing key when keyboard open (at ENTR instruction)	User program or next ENTR instruction
199	REC ADV	Pressing key when keyboard open (at ENTR instruction)	User program or next ENTR instruction
200	Special keyboard indicator	Indicators 185, 186, 190, 193, 198, and 199	User program or next ENTR instruction
240	Continued checkpoint taken	Completion of continued (C) CKPT and return to normal program execution	User program or by start of a CKPT operation

*Program control keys that are used when the programmer desires to receive communication from the operator without using an ENTR instruction. Only one of these indicators can be on at a time.

Indicators Within a Function-Selected Sequence

The following indicators are set on when the corresponding key is pressed in a FUNC SEL sequence. Each of these indicators can be set off by the RESET key, or by the next keyboard indicator (187-189 or 191), or by the next FUNC SEL sequence (only one of these indicators can be on at a time), or by the user program.

INDICATOR	FUNC SEL key
201	Lower CHAR ADV
202	Lower DUP
203	Lower FIELD COR
204	Lower less than (<)
205	Lower asterisk (*)
206	Lower percent (%)
207	Lower slash (/)
208	Lower HEX
209	Upper percent (%)
210	Upper slash (/)
211	Upper HEX
212	Lower dash (–)
213	Lower REC ADV
214	Lower at sign (@)
215	Lower SEL PROG
216	Lower FIELD ADV
217	Upper at sign (@)
218	Upper SEL PROG
219	Upper FIELD ADV
220	Upper CHAR ADV
221	Upper DUP
222	Upper FIELD COR
223	Upper less than (<)
224	Upper asterisk (*)

Indicators Set by Data Movement

The following indicators are set by data management during operation. Each of these must be set off by the user program.

225*	No record found–data set 1	These indicators are set on if the desired record is not found on the specified track.
226*	No record found–data set 2	
227*	No record found–data set 3	
228*	No record found–data set 4	
229*	Key not in table–data set 1	These indicators are set on if the index table is exceeded, or the key is less than the first index entry. Indicators 225-228 are also set on.
230*	Key not in table–data set 2	
231*	Key not in table–data set 3	
232*	Key not in table–data set 4	

*See *Columns 58-60 Type (R)* under *.DATASET* in Chapter 2.

Indicators 233-255 are reserved.

Appendix B. Translator Error Messages

TRANSLATOR ERROR FORMATS

Errors detected during the translation phase are listed on both the display and the printer. An object program with errors should not be used until all errors are corrected. Action on warning messages is at the users discretion. Disk errors include the disk address (cylinder and sector) of the error in the source or object file (from the label processor).

The display lists a maximum of seven messages, while the printer lists all error messages. If the source code is being printed, the error message is printed *before* the line found in error. Error codes have the following format:

XX0YY ZZZ	where XX = track, YY = sector, and ZZZ = error message.
XXX** ZZZ	where XXX = step number and ZZZ = error message.
***** ZZZ	where ZZZ = warning message.

Figure 50 shows printed error codes. Appendix B contains a detailed listing of translator error messages.

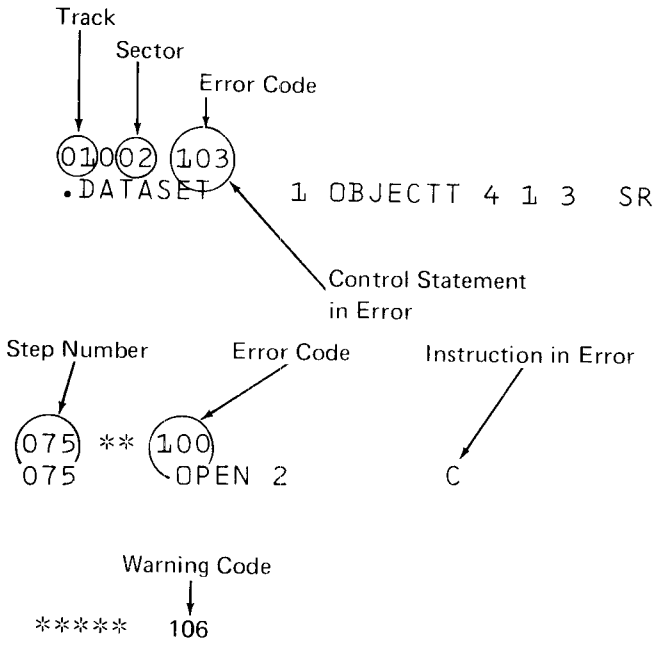


Figure 50. Error and Warning Messages During Translation

The following is a list of the error messages displayed to the operator. The corresponding meaning of each message is also listed. The first group of messages are those that will stop translation. To leave translation, the operator should press the ALPHA and NUM SHIFT keys and the RESET key. Messages are preceded by an S (reading from the source file), or an O (reading from the object file).

MESSAGES THAT STOP TRANSLATION APPEAR ON LINE 1

Message	Meaning	
0	Early disk removal error	Messages also display disk address
3	Disk seek error	
4	Disk read error	
5	Disk write check error	
8	Disk write error	
9	Disk no record found error (no ID match)	
10	Source and object on the same data set	
11	Invalid logical record length for source (must be greater than or equal to 80)	
12	Empty source file error (beginning of extent = end of data)	
14	Invalid logical record length for object (must be equal to 128)	
15	Object BOE not at track boundary	
16	No space available for object storage	
17	Invalid entry in .NAME control statement, or the .NAME is not the first source statement (positions 123-128 of .NAME are reserved)	
18	Printer not ready	

MESSAGES THAT DO NOT STOP TRANSLATION APPEAR ON LINE 6

Message	Meaning
19	Program name already in object data set. To continue translation, press RESET. The object stored under this name will be replaced. To abort translation, press ALPHA and NUMERIC SHIFT and RESET. Note that this message appears on line 1.

Control Statement Messages

Message	Meaning
20	Unrecognizable control statement. Control statement must have a period in column 1, followed by the statement name and parameters in the appropriate columns.
21	Control statements in invalid order. (.NAME, .DATASET, .PRINTER, .SELF-CHECK, and .REGISTER must appear first.)

.NAME

Message	Meaning
30	Multiple .NAME statements in the source data set
31	Invalid program start address specified (entry must be an odd-numbered buffer, and may not exceed available buffer storage)
32	Invalid machine size specified or attempt to translate 8K program on a 4K machine

.DATASET

Message	Meaning
40	Invalid logical record length specified (the entry in columns 28-30 must be 1-128 bytes)
41	Invalid data set access method specified (the entry in columns 58-60 must be SR, SU, SW, SWE, KR, KRN, KU, KUN, or I)
42	Invalid tracks/index (the entry in columns 68-69 must be 1-74)
43	Invalid bytes/key (the entry in columns 73-74 must be 1-16 and must be greater than or equal to the index entry length in columns 63-64).
44	Invalid index length (the entry in columns 63-64 must be 1-16 bytes)
45	Invalid key position (the entry in columns 78-80 must be 1-128 bytes)
46	Origin buffer must be specified for an indexed data set (KUN or KRN type). The entry in columns 83-84 must be 1-24.
47	Invalid physical drive address (the entry in column 33 must be 1 or 2)
48	Invalid index start or end position within buffer (the entry in columns 88-90 and 98-100 must be 1-128 bytes)
49	Multiple .DATASET control statements with the same data set number (column 13 entry must be 1-4)
50	Key length (columns 73-74) must be greater than or equal to the index length (columns 63-64)
51	Record length +1 (columns 28-30) must be greater than or equal to key length (columns 73-74) plus the key position (columns 78-80) in the record. If the record length is not specified, the key length plus key position cannot exceed 129.
52	The number of the index end buffer (columns 93-94) is less than the number of the index origin buffer (columns 83-84).
53	The index end position (columns 98-100) is less than or equal to the index start position (columns 88-90) for an index table to be built within the same buffer.
54	The character A must be entered in column 61 in order to bypass extent checking.

.PRINTER

Message	Meaning
60	Printer type not specified (column 13 must be 1 for 3713, 2 for 3717, 3 for 3715 single direction, 4 for 3715 bidirectional floating margin, or 5 for 3715 bidirectional fixed margin). Zero and blank are invalid. This error also applies to the printer specification in the .NAME statement (column 23).
61	Overflow line (columns 23-25) exceeds lines per page (columns 18-20). Overflow defaults to 60, and lines per page to 66.
62	Buffer (columns 33-34) not specified. Odd-numbered buffer must be specified if the characters to print exceeds 128.
63	Invalid characters to print (entry in columns 28-30 must be 4-132)
64	3715 bidirectional printer specified, but no secondary buffer (columns 38-39) is identified, or print line exceeds 128 characters and secondary buffer is not odd-numbered.
65	Invalid lines per page (columns 18-20 must contain 1-127) or overflow line number (columns 23-25 must contain a number less than or equal to the lines per page entry). This entry also applies to the .NAME control statement (columns 28-30 and 33-35), where lines per page must be equal to or greater than 7.
66	Multiple .PRINTER control statements
67	Primary and secondary print buffers cannot have the same buffer number.

.FIELD

Message	Meaning
70	Invalid field type character (column 23 must contain A for alpha, U for numeric, or D for digits).
71	Data disposition error (column 28 must contain B or blank, R, M, or D)
72	Field chaining error (column 29 must contain blank, C, J, or L or column 36 is not blank or T)
73	Exit control error (column 30 must contain blank, J, or Z)
74	Delimiter error (asterisk missing after message) or prompting message plus field length exceeds 69.
75	Field length error (columns 24-25 must contain 1-16 if data is moved to a register, or 1-64 if data is moved to a buffer) or the field length plus the buffer displacement (columns 33-35) exceeds 129, or the field length plus the message length is greater than 69.
76	Buffer full error, or 2 or fewer positions remain in a .FIELD buffer.
77	Invalid message character (hex 00 or hex FF are invalid)

.FORMAT

Message	Meaning
80	*, /, 0 following an invalid field (indicates an undefined field with no register definition character)
81	Blank following an invalid field (no register definition character)
82	$ following an invalid field (no register definition character)
83	Minus sign following an invalid field (no register definition character)
84	& or period following an invalid field (no register definition character)
85	Formats have overlaid instruction storage area.
86	Blank record warning error (no format following .FORMAT)
87	Field break warning error (indicates that more than 16 register designation characters have been entered for the same register in a format).

.SELF-CHECK

Message	Meaning
90	A GSCK or IFCHK instruction has been issued without a corresponding .SELF-CHECK statement.
91	Self-check format error (columns 23-25 and 28-30)
92	Multiple .SELF-CHECK statements error
93	.SELF-CHECK modulus error
94	.SELF-CHECK displacement error
95	.SELF-CHECK table error
96	.SELF-CHECK weight error
97	.SELF-CHECK format combination error

General Error Messages

Message	Meaning
100	Data set number specified has no .DATASET control statement.
101	Unrecognizable statement
102	Invalid register specification (must be A-Z) Registers I, R, and Z cannot be used in table instructions.
103	Buffer number specified must be 1-24 for 4K program or 1-56 for 8K program. It may exceed available buffer storage.
104	Out of range arithmetic constant (must be 0-9), or constant coded in sign position when assigning a 2-5 digit constant
105	Out of range number error (greater than 256)
106	Invalid character in a numeric field
107	Invalid data set number (must be 1-4)
108	Unrecognizable arithmetic statement (column 23 must contain +, –, *, /, or blank)
109	Invalid displacement or number of characters to be moved or zoned in MVER, MOFF, or ZONE instruction
110	Invalid indicator number specified (must be 1-255)
111	Invalid format number (must be 1-254)
112	Invalid print control code specified (must be T, D, S, or 0-127)
113	Invalid length or displacement specified in a LOAD or STOR instruction. (Length must be 1-16 and displacement must be 1-256, but length plus displacement cannot exceed 257. For displacements greater than 128, an odd-numbered buffer must be specified.)
114	Invalid table number (must be 1-16)
115	Invalid table entry length field (must be 1-16)
116	Save step number or label missing in RGO instruction

Message	Meaning
117	Message number in ENTR instruction (column 18) must be entered (valid range is 1-99)
118	Duplicate instruction number or label (program must not extend into format storage area, and each control statement must have a buffer specification within the program storage area)
119	Branching step number or label is not in the same 256-instruction block. The following branching instructions must have a GOTO address within the same 256-instruction block: IF A=, IF A >, IF A <, IFD A=, IFD A >, IFD A <.
120	Invalid step number/label specified in columns 1-4 or 28-31 of a branching instruction.
121	.PRINTER control statement missing
122	Invalid replace (move register to register) instruction (columns 23 and 28 must be blank)
123	Register name cannot be the same as the result register in multiply and divide instructions.
124	Invalid constant in shift instruction (column 28 must be 1-15)
125	Program storage overflow (step number exceeds available buffer storage)
126	Factor 1 (column 18) and factor 2 (column 28) cannot be blank in arithmetic instructions.
127	Invalid buffer displacement (must be 1-128)
128	Invalid register displacement (must be 1-16)
129	Duplicate format number error
130	Entry appears in field that should be blank (check columns 5-7, 17, 21-22, and 26-27)
131	Constant is out of range in assigning a constant instruction (columns 23-27 must be 1-65535)
132	Invalid zone specified (must be 0-9 or A-F)
133	Invalid overlap specified in ENTR, PRNT, or CRDR instructions (column 23 or 28 must be X for overlapped or blank for nonoverlapped operation)
134	Unrecognizable shift instruction
135	Unrecognizable IF instruction
137	Invalid assigning constant instruction (column 28 contains an entry)
138	Invalid option specified in EXEC instruction
139	Zero control code for the printer type specified

Warning Error Messages

Message	Meaning
140	Format number not defined by .FORMAT statement
141	Column 23 cannot be blank in IF or IFD branching instructions
150	Dummy GOTO missing at return address in RGO instruction
151	Branch to undefined step number/label error
152	Step number/label missing from sequence. Also ensure that control statements, (.BUFFER and .FIELD) do not extend into the source instructions program area.
153	Branching address exceeds available buffer storage
154	EXIT instructions missing (can occur if the EXIT instruction is within the instructions causing program overflow)
155	Reserved
156	Reserved
157	Reserved
158	Reserved
159	Data set 1 deleted record procedure step number/label invalid (must be dummy GOTO)
160	Data set 2 deleted record procedure step number/label invalid (must be dummy GOTO)
161	Data set 3 deleted record procedure step number/label invalid (must be dummy GOTO)
162	Data set 4 deleted record procedure step number/label invalid (must be dummy GOTO)
163	Data set 1 end of file error step number/label invalid (instruction must be a dummy GOTO)
164	Data set 2 end of file error step number/label invalid (instruction must be a dummy GOTO)
165	Data set 3 end of file error step number/label invalid (instruction must be a dummy GOTO)
166	Data set 4 end of file error step number/label invalid (instruction must be a dummy GOTO)
167	Printer overflow routine step number/label invalid (instruction must be a dummy GOTO)
172	.END control statement missing in program

Note that second pass errors (greater than 150) may occur if buffers are assigned above the program origin buffer for data storage.

Appendix C. Execution Error Codes, Meanings, and Operator Responses

Errors during execution are displayed in positions 5, 6, 7, and 8 of display line 1. Printer errors and disk errors overlay keyboard errors.

Printer Errors

If a printer error is encountered the operator should first check for the obvious problems, such as the cover open or no forms. The operator must choose one of the following responses:

Display Position Contents (5 6 7 8)	Error Meaning	Operator Response
2-16 P	P errors preceded by a numeric are posted if the 3717 Printer is specified in the .PRINTER statement (Note 2).	Press NUM SHIFT and RESET to bypass the print instruction.
P 1	Printer is not attached.	Press ALPHA SHIFT and RESET to bypass any remaining print instruction.
P 2	Print error during forms movement (Reposition forms if necessary.)	Press RESET to retry the dump from one last buffer (P3 only).
P 3	Print error during dump (Reposition forms if necessary.)	Press ALPHA and NUM SHIFT with RESET to return to index (X) mode.
P 4	Print error while printing or during trace to printer.	Press RESET to retry the print instruction (P3 and P4 only).

Keyboard indicators are not turned off when correcting printer errors.

Notes:

1. If a P2 or P3 error occurs and the operator selects to bypass the print instruction or retry the dump, the operator must manually check vertical forms alignment to ensure that it is correct for the next print instruction.
2. For a work station with the 3717 Printer, the numbers 2-16 are displayed in positions 5 and 6 to indicate the following errors:
   - 2 Print belt synchronization check
   - 3 Not used
   - 4 Thermal overload check
   - 5 Hammer check (1-22)
   - 6 Hammer check (23-44)
   - 7 Hammer check (45-66)
   - 8 Carriage synchronization check
   - 9 Forms jam
   - 10 Busy-too-long check
   - 11 Cover interlock open
   - 12 Throat interlock open
   - 13 Hammer echo check
   - 14 Print belt speed check
   - 15 Printer not ready
   - 16 End of form

Card I/O Errors

The following errors apply to the attached card I/O device (5496 or 129) and have the same operator response options as the P4 printer errors. Note that resetting read errors before clearing the 5496 can jam two cards in the read station.

Display Position Contents (5 6 7 8)	Error Meaning
A 1	The attached device is offline, or switches are set improperly, or card jam in the transport.
A 2	Hopper empty; stacker full, or misfeed.
A 3	Byte transferred between work station and I/O device does not compare.
A 4	Program sequence invalid (successive start read instructions — S in column 23 of CRDR instruction or punch following start read)

Invalid Key Errors

Display Position Contents 5 6 7 8	Error Meaning	Operator Response
9 0	Invalid key was pressed in an ENTR statement	Press RESET to continue program execution. Note that this turns off keyboard indicators.
9 1	Key is pressed when not in an ENTR statement or entry field is full (requiring exit key) and nonexit key is pressed	
9 2	Keyboard overrun — keying rate exceeds keyboard capability	

Job Completion System Halt

Display Position Contents 5 6 7 8	Error Meaning	Operator Response
1 0 0	Job complete - not an error condition	Press RESET to go to index (X) mode.

.FIELD Control Statement Errors

If an error is encountered, the .FIELD control statement must be corrected before the program can be executed.

Display Position Contents 5 6 7 8	Error Meaning	Operator Response
1 5 0	Offset and length is greater than 128 or buffer does not contain prompting messages	Press ALPHA and NUM SHIFT with RESET to return to index (X) mode.
1 5 1	Data sent to register and field length greater than 16	
1 5 2	Register specification is zero	
1 5 3	Message not found in buffer (message number greater than 1 can't be found in buffer in ENTR)	
1 5 4	Prompting message and field greater than 69	
1 5 5	More than eight continuation fields in one ENTR	

Miscellaneous Errors

If an error is encountered, the operator must choose one of the responses associated with the error code.

Display Position Contents 5 6 7 8	Error Meaning	Operator Response
2 0 0	Disk error on program load	Press ALPHA and NUM SHIFT with RESET to return to index (X) mode.
2 0 1	Object data set length is zero	Press ALPHA and NUM SHIFT with RESET to return to index (X) mode.
2 0 2	Object data set is less than two tracks (4K) or 4 tracks (8K)	Press ALPHA and NUM SHIFT with RESET to return to index (X) mode.
2 0 3	Attempt to load an 8K program on a 4K machine	
2 0 4	Program name not found or program name is not on a correct track boundary	Press ALPHA and NUM SHIFT with RESET to return to index (X) mode.
2 0 5	Invalid drive number (position 9 or 19) is keyed when selecting operation. This error is possible for both translation and execution	Press ALPHA and NUM SHIFT with RESET to return to index (X) mode. If the data set is a continued data set, try another disk to see if the program name is there.
2 0 6	Object data set has been altered	Press RESET to initiate execution with altered data set. Press ALPHA and NUM SHIFT with RESET to return to index (X) mode. New translation required.
2 0 7	Feature not available	Press ALPHA and NUM SHIFT with RESET to return to index (X) mode.
2 5 3	Work station storage failure	Press ALPHA and NUM SHIFT with RESET to return to index (X) mode.
2 5 5	Hardware parity error	Press ALPHA and NUM SHIFT with RESET to return to index (X) mode.

Disk Open Errors

If an error is encountered, the operator must choose one of the responses associated with the error code(s). Note that disk open failed in each case.

Display Position Contents 5 6 7 8	Error Meaning	Operator Response
5 X 0	Drive not ready	Press RESET to retry open. Press NUM SHIFT and RESET to go to job completion-system code 100 or to return to index mode for 5X1 during program load only.
5 X 1	Data set name not found in user program, or object data set not found during program load or data set(s) from .NAME statement missing	
5 X 2	Record length on index track does not match record length specified in .DATASET	
5 X 3	Invalid label extent	
5 X 4	Read error on index track*	
5 X 5	No space available to build index table, or disk error while building index, for key indexed data set	Press RESET to go to job completion-system code 100.
5 X 6	Register information invalid for dynamic open (column 23 of OPEN instruction)	

Display Position Contents 5 6 7 8	Error Meaning	Operator Response
5 X 7	Extents overlap with some other data sets on this disk	Press RESET to retry open. Press NUM SHIFT and RESET to go to job completion-system code 100.
5 X 8	Not enough space to build a complete index table for a key indexed data set	This is a warning message. Press RESET to continue program execution. Press NUM SHIFT with RESET to go to job completion-system code 100.
5 X 9	Key indexed data set index entries are not in sequence	Press RESET to go to job completion-system code 100.
5 X A	Secured/protected file (position 11 of sector 7 is not blank, position 42 of the label is S, position 43 of the label is P, and the file is opened for a write operation, or positions 1-4 of sector 7 do not contain 'VOL1')**	Press RESET to retry open. Press NUM SHIFT with RESET to go to job completion-system code 100.
5 X ?	Write gate check (hardware error)	Press RESET to retry operation. Press NUM SHIFT and RESET to go to job completion-system code 100.

Note: X = data set number (1 through 4).

* If a record length is shortened on a data set already written, a length error (1) occurs when reading these records in Model 1 (Data Station) mode. If this data set is specified as key indexed (KR) with index table build, a 544 length error occurs while reading records into the index table during an open operation in Model 3-4 mode. If the data set is specified as sequential, a 7X4 (read) error occurs on the first read to the data set, and not during the open operation.

** These errors make the disk unusable in work station mode, but usable in data station mode.

Disk Close Errors

If an error is encountered, the operator should manually record the EOD value in positions 12 through 16 of buffer 1. The operator must then choose one of the following responses. Disk close failed in each case.

Display Position Contents 5 6 7 8	Error Meaning	Operator Response
6 X 0	Drive not ready*	Press RESET to retry any error except 6X0. Press NUM SHIFT and RESET to go to job completion-system code 100.
6 X 2	Read ID error (can't find label on index track)	
6 X 3	Seek command error (can't return to index track)	
6 X 4	Read command error (found label, but can't read it)	
6 X 5	Write check error (located and read label, but can't write it)	
6 X 8	Write command error (read label but can't write it)	
6 X 9	Seek read ID error	

Note: X = data set number (1 through 4).

* If the drive is opened and closed after a data set has been opened, a 7X0 error occurs when an attempt is made to access the data set. The EOD of the data set cannot be updated. If the opened data set is SU, SW, or SWE, however, a 6X0 error is posted. Press NUM SHIFT and RESET to bypass the data set, then the remaining data sets can be closed.

Disk Errors

Display Position Contents 5 6 7 8	Error Meaning	Operator Response
7 X 0	Data set not opened or drive not ready*	Press RESET to go to job completion system code 100.
7 X 2	Read ID error	
7 X 3	Seek error	
7 X 4	Read error (See 5X4)	
7 X 5	Write check error	
7 X 7	Write control address mark error	
7 X 8	Write error	
7 X 9	Seek read ID error	
7 X B	Read to write only data set error	
7 X C	Write to read only, WRTE to nonextended data set, or WRT with relative record number with KU, SW, SWE, or KUN data set organization	
7 X D	Attempt to read below BOE	
7 X E	End of file is encountered, but no EOF exit is specified.	

Note: X = data set number (1 through 4).

* If the drive is opened and closed after a data set has been opened, a 7X0 error occurs when an attempt is made to access the data set. The EOD of the data set cannot be updated. If the opened data set is SU, SW, or SWE, however, a 6X0 error is posted. Press NUM SHIFT and RESET to bypass the data set, then the remaining data sets can be closed.

Checkpoint Errors

Display Position Contents 5 6 7 8	Error Meaning	Operator Response
9 X 0	Drive not ready	Press RESET to go to job completion-system code 100.
9 X 1	Data set open error. Checkpoint data set already open at time of checkpoint. *Note:* CKPT instruction cannot be at instruction step number 255, 511, or 767.	
9 X 2	Syntax error in register or data set length not 128	
9 X 3	Data set BOE not on sector one	
9 X 4	EOE reached before checkpoint complete	
9 X 5	Invalid data set specified	
9 X 6	Write error	

Note: X = data set number (1 through 4).

Communication Link Errors

Display Position Contents 6 7 8 9	Error Meaning	Operator Response
C 0 1	File 1 is an I-type file	Press RESET to return to index (x) mode.
C 0 2	File 1 is not on drive 1	Press RESET to return to index (x) mode.
C 0 3	File 1 is not open	Press RESET to return to index (x) mode.
C 0 4	File is an SU-, SW-, or SWE-type file, but the communications mode selected is not R or I	Press RESET to return to index (x) mode.
C 0 5	File 1 is a KU-, KR-, or SR-type file, but the communications mode selected is not T, P, B, D, J, or K	Press RESET to return to index (x) mode.
C 0 6	Unattended print is selected, but the AUTO REC ADV switch is not on	Press RESET to return to index (x) mode or set the AUTO REC ADV switch on and press RESET to continue.
C 0 7	Keylock is locked	Press RESET to return to index (x) mode or unlock key and press RESET to continue.
C 0 8	Unattended print and transmit mode are selected	Press RESET to return to index (x) mode.
C 0 9	Unattended ACL program execution and inquiry mode are selected	Press RESET to return to index (x) mode.
C 1 0	Inquiry mode is selected, but file 1 record length does not equal 128	Press RESET to return to index (x) mode
C 1 1	Inquiry or receive mode is selected and the disk is positioned beyond sector 73026	Press RESET to return to index (x) mode.
C 1 3	Transmit mode is selected with the disk at EOD	Press RESET to return to index (x) mode.
C 1 4	There is a disk write error	Press RESET to return to index (x) mode
C 1 5	There is a disk read error	Press RESET to return to index (x) mode.
C 1 6	No record is found	Press RESET to return to index (x) mode.
C 1 7	Volume label on disk 2 is secure	Remove disk 2 and return to the index (x) mode.
C 1 9	Receive data and insert constants program is not equal to the file 1 record length	Press RESET to return to index (x) mode.
C 2 1	Disk drive 1 is not ready when the COMM instruction is executed	Press RESET to return to index (x) mode.
C 2 2	Unattended print is selected and byte 4 of register A does not equal 2 through 9	Press RESET to return to index (x) mode.
C 2 3	There is a disk seek error	Press RESET to return to index (x) mode.
C 2 4	Disk drive is not ready during ACL communications linkage execution	Press RESET to return to index (x) mode.
C 2 8	File 2 is not closed	Press RESET to return to index (x) mode.
C 3 8	File 3 is not closed	Press RESET to return to index (x) mode.

Display Position Contents 6 7 8 9	Error Meaning	Operator Response
C 4 8	File 4 is not closed	Press RESET to return to index (x) mode.
C 9 9	Communications feature is not installed.	Press RESET to return to index (x) mode.
C ? ?	There is a write gate error on disk drive 2	Remove disk 1 and return to the index (x) mode, or remove disk 2 and press RESET to continue, or press RESET to continue. (Any writing to disk 2 can be unpredictable.)

Operation Code Error

Display Position Contents 5 6 7 8	Error Meaning	Operator Response
1 Y Y Y	Attempt to execute an invalid operation	Press ALPHA and NUM SHIFT and RESET to return to index mode.

Note: YYY is for diagnostic purposes only.

SAMPLE PROGRAM 1-ORDER ENTRY

Sample program 1 illustrates the coding of an order entry application which operates on the 3741 Models 3 and 4. Figure 51 is the order form to be used. Note that the fields to be entered include:

- Customer number
- Order number
- Date
- Salesman number
- Purchase order number
- Ship instructions
- Product code
- Quantity
- Price

Figure 52 shows the data displayed after entry, and the format of the diskette record written for the loader-type information. Figure 53 shows how the data is displayed and written for each item on the order.

During the data entry function, a shipping code is used to search a table to find the corresponding shipping instruction. If an invalid shipping code is keyed, an error message (Figure 54) is displayed to the operator.

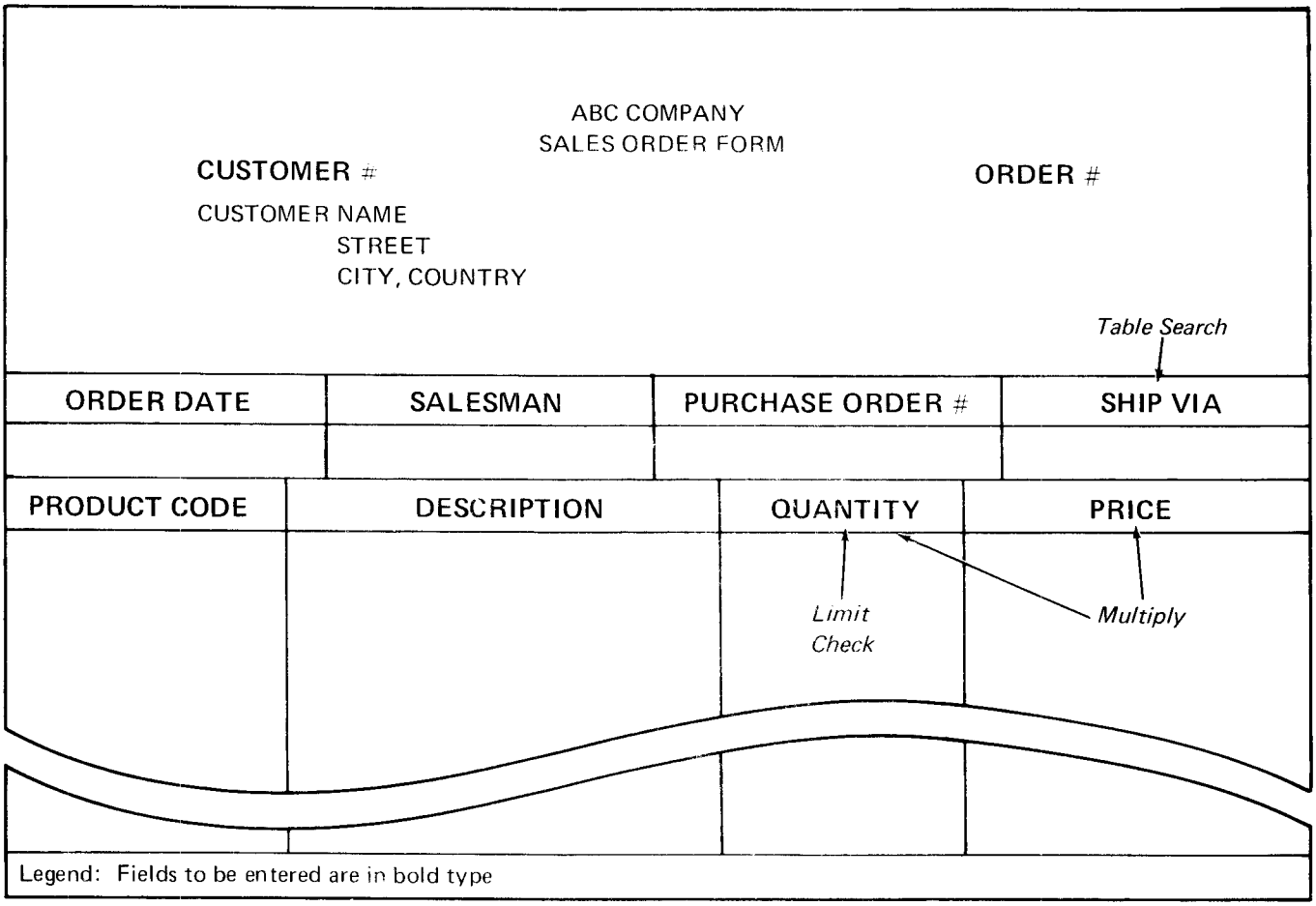
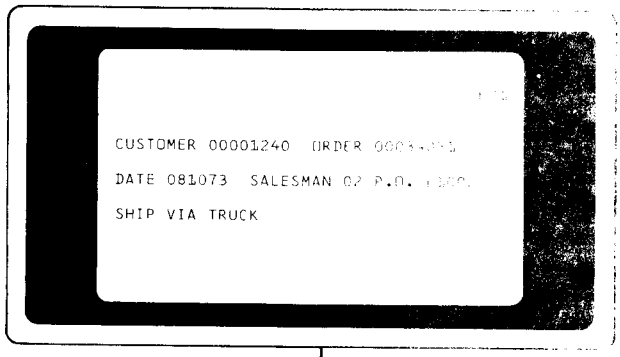


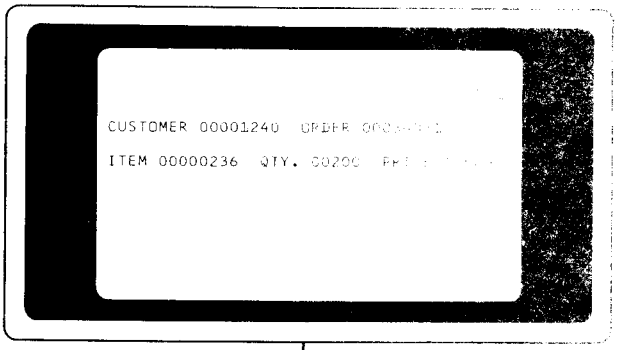
Figure 51. Order Form for Sample Order Entry Program



CUSTOMER NUMBER	ORDER NUMBER	DATE	SALES-MAN	PURCHASE ORDER ORDER NUMBER	SHIPPING INSTRUCTIONS
1	9	17	23	25	35

DISKETTE RECORD

Figure 52. Display of Entered Data and Format of the Diskette Record



CUSTOMER NUMBER	ORDER NUMBER	ITEM NUMBER	QTY	PRICE
1	9	17	25	30

DISKETTE RECORD

Figure 53. Display and Diskette Writing for Each Item on the Order

After the quantity is entered, a limit check is made to see if the quantity is greater than 1000. If it is, the operator is notified by an error message, shown in Figure 55.

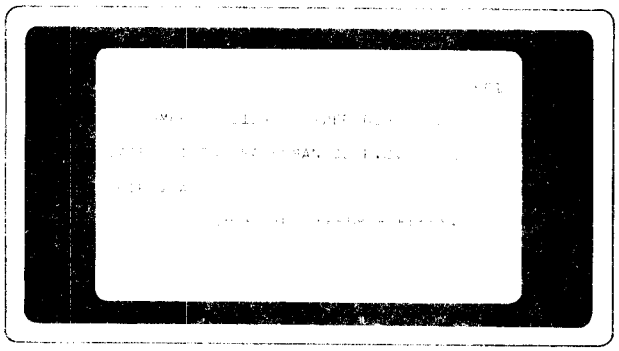


Figure 54. Table Search and Error Message for Shipping Code

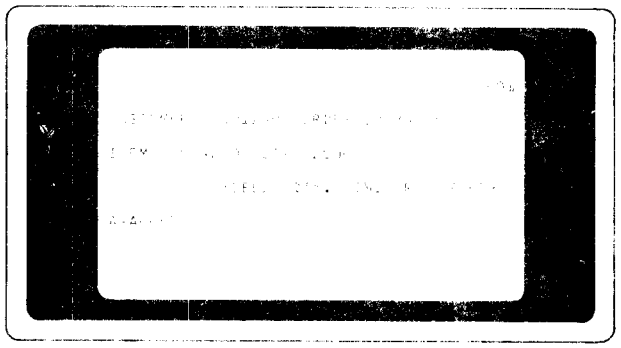


Figure 55. Limit Check and Error Message for Quantity Exeeds 1000

The coding necessary for this program is shown in Figure 56

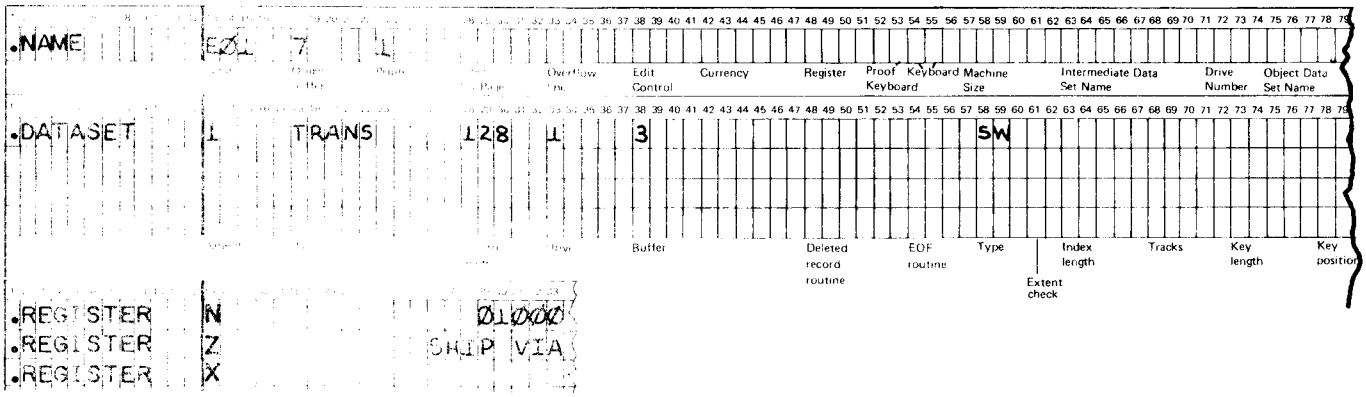


Figure 56 (Part 1 of 4). Coding for Sample Order Entry Program

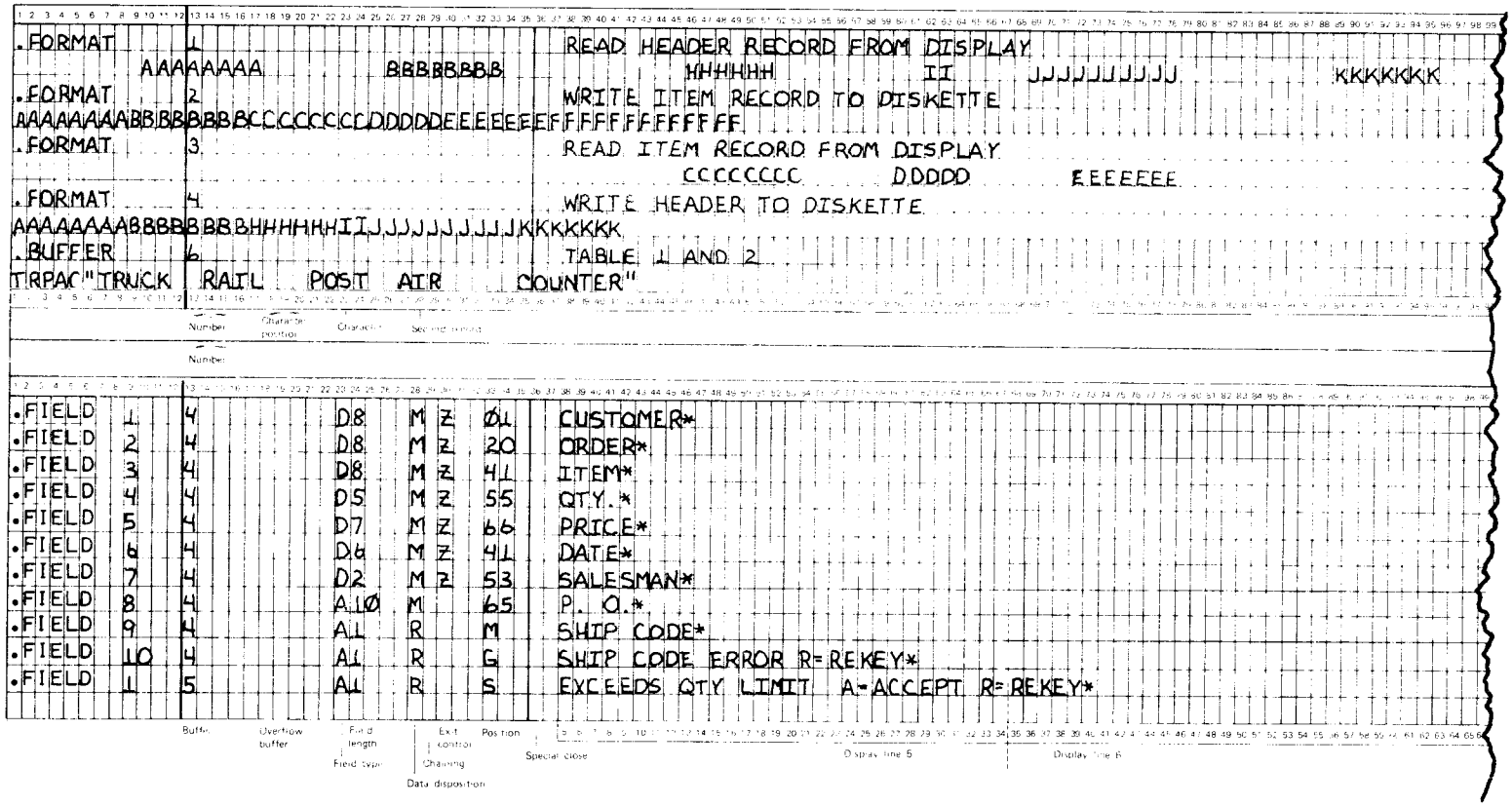
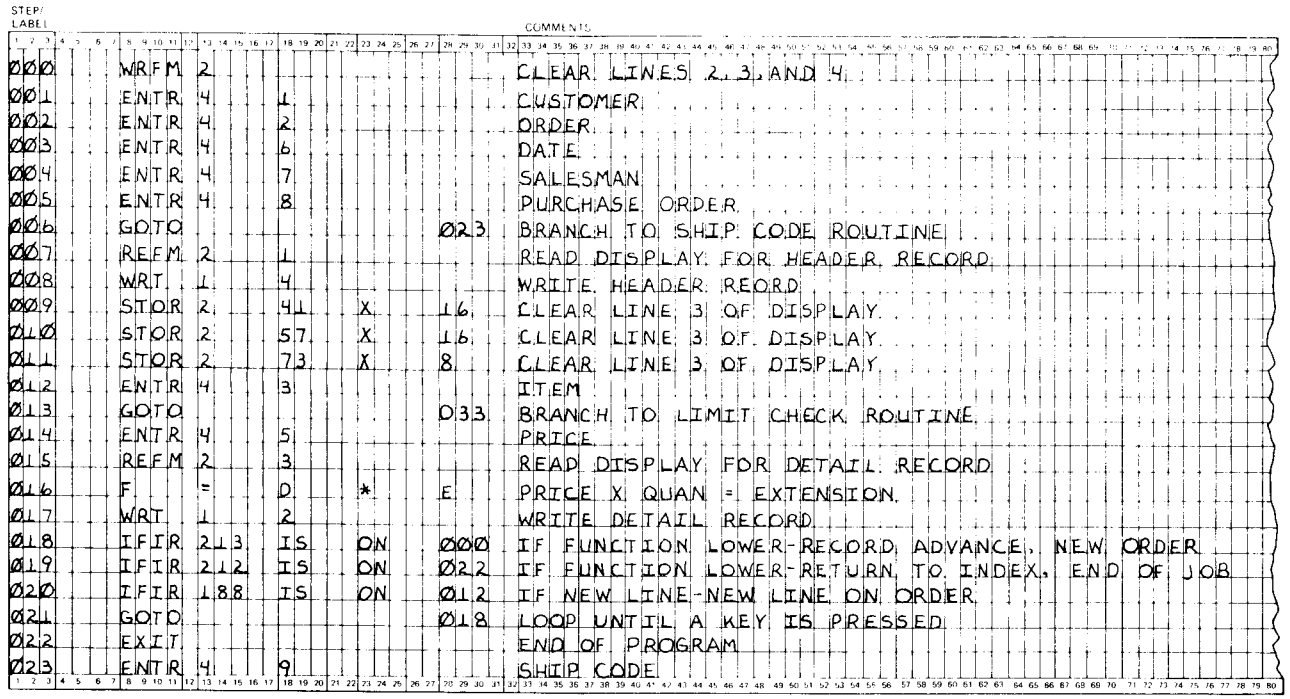


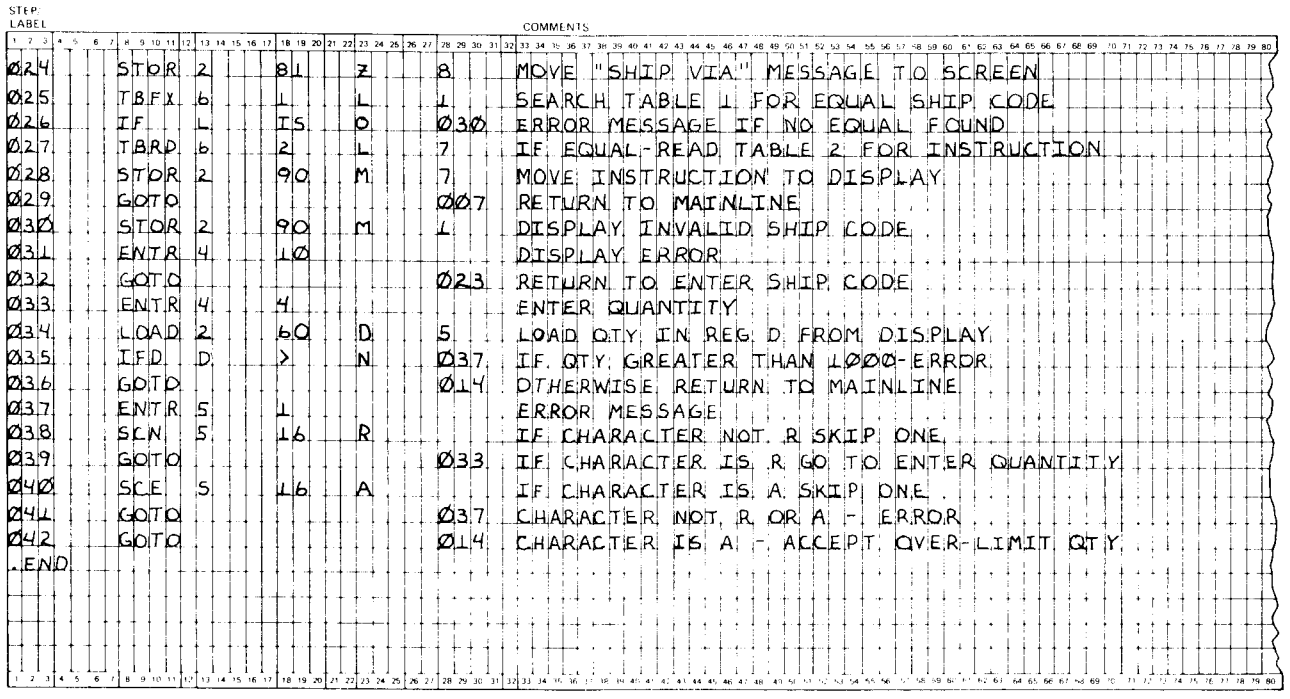
Figure 56 (Part 2 of 4). Coding for Sample Order Entry Program

~~~~ indicates continuation of the first and last character (register).
~~~~



```
STEP/
LABEL                                                                COMMENTS
000     WRFM  2                                                      CLEAR LINES 2, 3, AND 4
001     ENTR  4     1                                                CUSTOMER
002     ENTR  4     2                                                ORDER
003     ENTR  4     6                                                DATE
004     ENTR  4     7                                                SALESMAN
005     ENTR  4     8                                                PURCHASE ORDER
006     GOTO                                    023                  BRANCH TO SHIP CODE ROUTINE
007     REFM  2     1                                                READ DISPLAY FOR HEADER RECORD
008     WRT   1     4                                                WRITE HEADER RECORD
009     STOR  2     41          X       16                           CLEAR LINE 3 OF DISPLAY
010     STOR  2     57          X       16                           CLEAR LINE 3 OF DISPLAY
011     STOR  2     73          X       8                            CLEAR LINE 3 OF DISPLAY
012     ENTR  4     3                                                ITEM
013     GOTO                                    033                  BRANCH TO LIMIT CHECK ROUTINE
014     ENTR  4     5                                                PRICE
015     REFM  2     3                                                READ DISPLAY FOR DETAIL RECORD
016     F     =     D           *       E                            PRICE X QUAN = EXTENSION
017     WRT   1     2                                                WRITE DETAIL RECORD
018     IFIR  2 1 3 IS      ON          000                          IF FUNCTION LOWER-RECORD ADVANCE, NEW ORDER
019     IFIR  2 1 2 IS      ON          022                          IF FUNCTION LOWER-RETURN TO INDEX, END OF JOB
020     IFIR  1 8 8 IS      ON          012                          IF NEW LINE-NEW LINE ON ORDER
021     GOTO                            018                          LOOP UNTIL A KEY IS PRESSED
022     EXIT                                                         END OF PROGRAM
023     ENTR  4     9                                                SHIP CODE
```

Figure 56 (Part 3 of 4). Coding for Sample Order Entry Program



```
STEP/
LABEL                                                                COMMENTS
024     STOR  2     81          Z       8                            MOVE "SHIP VIA" MESSAGE TO SCREEN
025     TBFX  6     L           L       1                            SEARCH TABLE 1 FOR EQUAL SHIP CODE
026     IF    L     IS          O       030                          ERROR MESSAGE IF NO EQUAL FOUND
027     TBRD  6     2           L       7                            IF EQUAL-READ TABLE 2 FOR INSTRUCTION
028     STOR  2     90          M       7                            MOVE INSTRUCTION TO DISPLAY
029     GOTO                            007                          RETURN TO MAINLINE
030     STOR  2     90          M       1                            DISPLAY INVALID SHIP CODE
031     ENTR  4     10                                               DISPLAY ERROR
032     GOTO                            023                          RETURN TO ENTER SHIP CODE
033     ENTR  4     4                                                ENTER QUANTITY
034     LOAD  2     60          D       5                            LOAD QTY IN REG D FROM DISPLAY
035     IFD   D     >           N       037                          IF QTY GREATER THAN 1000-ERROR
036     GOTO                            014                          OTHERWISE RETURN TO MAINLINE
037     ENTR  5     1                                                ERROR MESSAGE
038     SCN   5     16          R                                    IF CHARACTER NOT R SKIP ONE
039     GOTO                            033                          IF CHARACTER IS R GO TO ENTER QUANTITY
040     SCE   5     16          A                                    IF CHARACTER IS A SKIP ONE
041     GOTO                            037                          CHARACTER NOT R OR A - ERROR
042     GOTO                            014                          CHARACTER IS A - ACCEPT OVER-LIMIT QTY
.END
```

Figure 56 (Part 4 of 4). Coding for Sample Order Entry Program

SAMPLE PROGRAM 2-MAILING LIST INQUIRY

Sample program 2 illustrates the use of two key functions within the ACL language:

- Using the key indexed access method
- Providing error correction using the special keyboard close function of the .FIELD control statement, and using registers to specify the buffer and message number in the ENTR instruction

The operator is first requested to enter the customer number. A search of the mailing list data set (Figure 57) is initiated. If the matching record is *not* found, the operator is requested to reenter the customer number. If the record is found, it is displayed to the operator in the format shown in Figure 58. The operator is prompted to enter a new street, city, state, zip code, and telephone number. If any of these fields are not to be udpated, the operator presses the FIELD ADV key to advance to the next field, or REC ADV key to advance to the end of the record and update the diskette. If an error is detected in a previously entered field, the operator can press the FIELD BKSP key to return the program to the previous field.

The ACL coding necessary for this sample program is shown in Figure 59.

CUSTOMER NUMBER	1-10
CUSTOMER LAST NAME	11-25
CUSTOMER FIRST NAME	26-36
TELEPHONE NUMBER	37-43
STREET	44-59
CITY	60-75
COUNTRY OR STATE	76-85
ZIP CODE	86-90

Figure 57. Format of Records in Mailing List Data Set

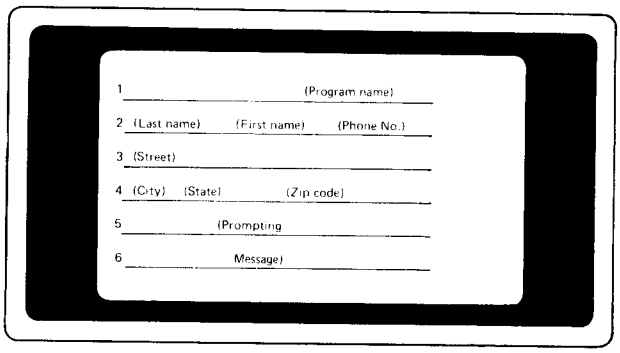


Figure 58. Format of Mailing List Record Displayed to the Operator

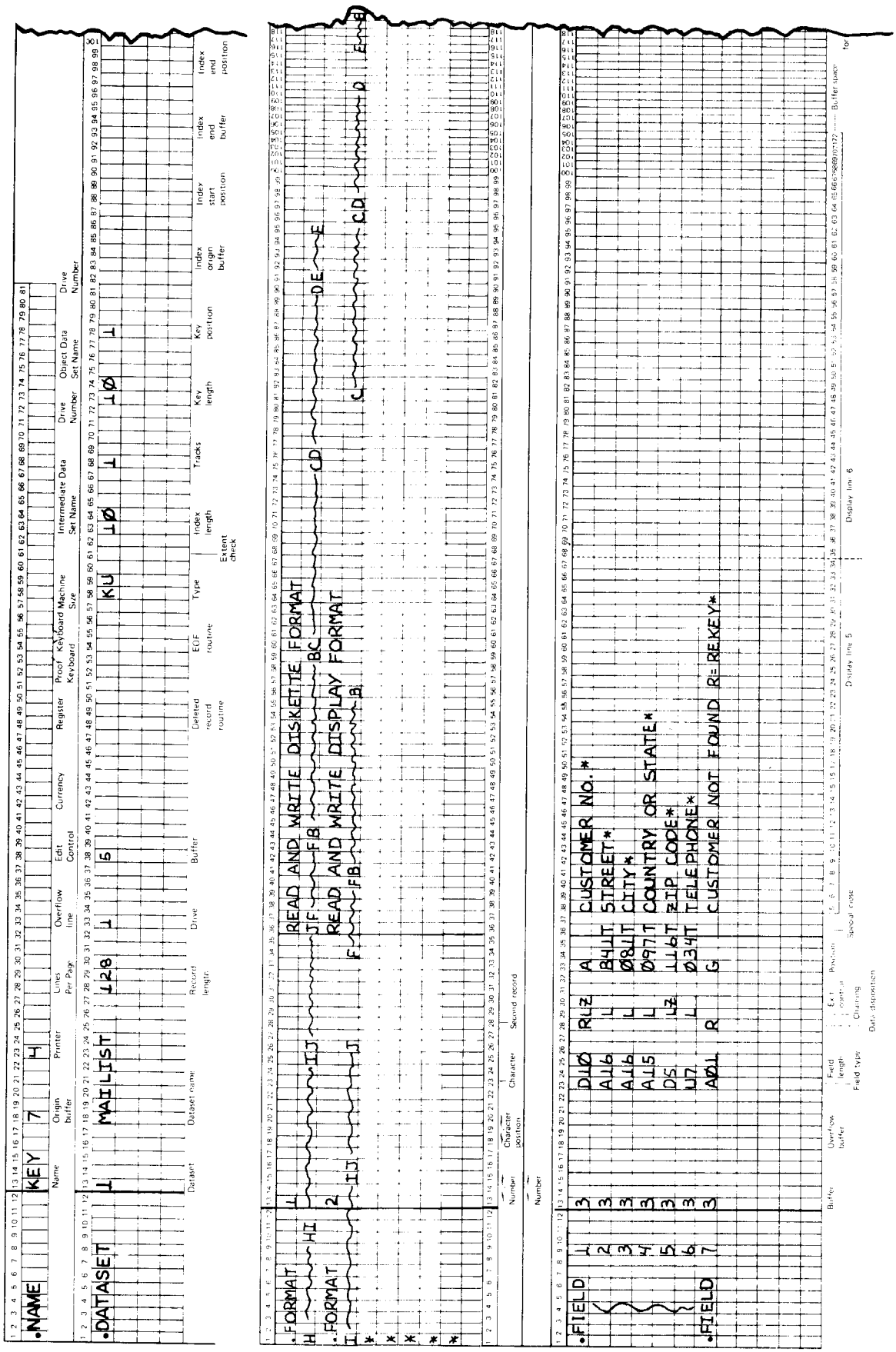
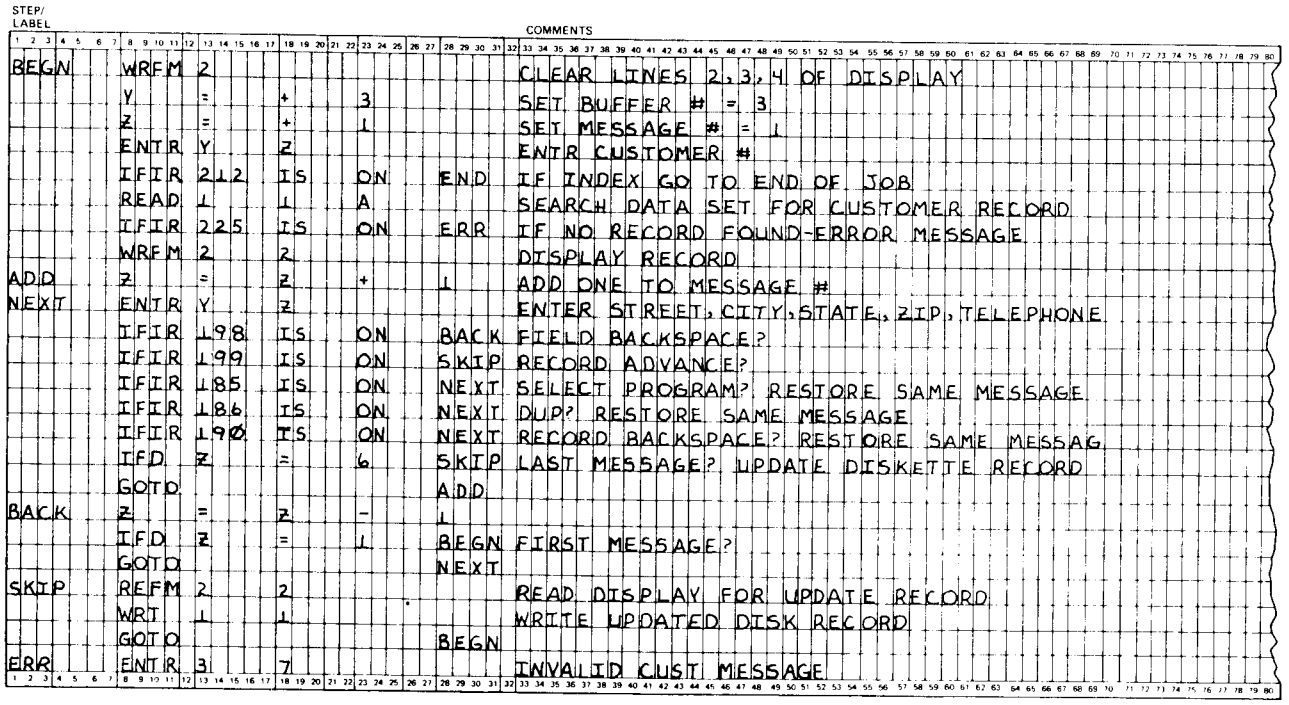


Figure 59 (Part 1 of 3). Coding for Sample Mailing List Inquiry Program

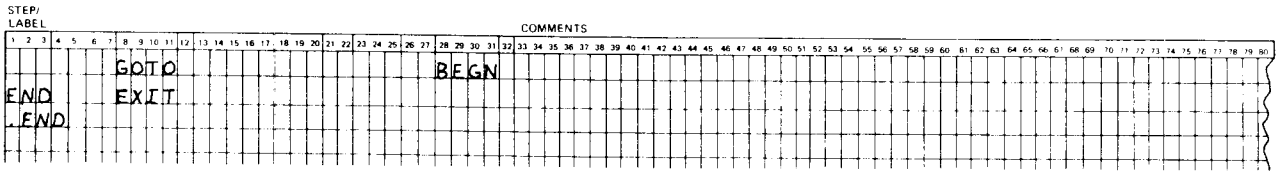
Register Usage

A – Customer number (keyed)
B – Street
C – City
D – Country or state
E – Zip Code
F – Telephone
G – Scratch
H – Customer number (read from file)
I – Customer last name
J – Customer first name



```
STEP/
LABEL                                                    COMMENTS
BEGN    WRFM  2
        Y     =     +     3
        Z     =     +     1
        ENTR  Y     Z
        IFIR  212   IS    ON    END   IF INDEX GO TO END OF JOB
        READ  1     1     A           SEARCH DATA SET FOR CUSTOMER RECORD
        IFIR  225   IS    ON    ERR   IF NO RECORD FOUND-ERROR MESSAGE
        WRFM  2     2                 DISPLAY RECORD
ADD     Z     =     Z     +     1     ADD ONE TO MESSAGE #
NEXT    ENTR  Y     Z                 ENTER STREET, CITY, STATE, ZIP, TELEPHONE
        IFIR  198   IS    ON    BACK  FIELD BACKSPACE?
        IFIR  199   IS    ON    SKIP  RECORD ADVANCE?
        IFIR  185   IS    ON    NEXT  SELECT PROGRAM? RESTORE SAME MESSAGE
        IFIR  186   IS    ON    NEXT  DUP? RESTORE SAME MESSAGE
        IFIR  190   IS    ON    NEXT  RECORD BACKSPACE? RESTORE SAME MESSAG
        IFD   Z     =     6     SKIP  LAST MESSAGE? UPDATE DISKETTE RECORD
        GOTO                    ADD
BACK    Z     =     Z     -     1
        IFD   Z     =     1     BEGN  FIRST MESSAGE?
        GOTO                    NEXT
SKIP    REFM  2     2                 READ DISPLAY FOR UPDATE RECORD
        WRT   1     1                 WRITE UPDATED DISK RECORD
        GOTO                    BEGN
ERR     ENTR  3     7                 INVALID CUST MESSAGE
```

Figure 59 (Part 2 of 3). Coding for Sample Mailing List Inquiry Program



```
STEP/
LABEL                                                    COMMENTS
        GOTO                    BEGN
END     EXIT
        END
```

Figure 59 (Part 3 of 3). Coding for Sample Mailing List Inquiry Program

SAMPLE PROGRAM 3-OVERLAY PROGRAM

Sample program 3 illustrates how an overlay program is coded. In the example, there is one mainline program and two overlay programs. Some important items to be considered when using overlays are:

- The .NAME statement must reference the same origin buffer in the mainline and overlay programs. (Buffer 9 in this example.)

- .FORMAT statements must be used in the mainline program, while dummy formats may be used in the overlay program. This is because formats are loaded in high storage by the work station at translation time and consist of pointers only.

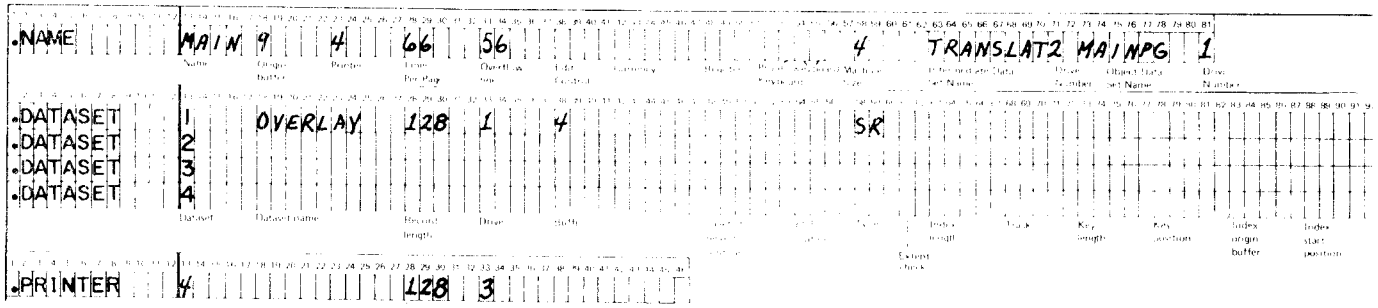
- The ORG instruction references the actual buffer where the overlay will be positioned in the mainline program. Since each instruction is 4 bytes in length, there are 32 instructions per buffer, and by assigning a step number of 128 with the ORG instruction, the next instruction would start four buffers away from the buffer referenced in the .NAME statement. A step number of 000 is in the .NAME referenced buffer.

- In the example, the overlay programs are executed once to create the overlay data set. The first overlay program references the overlay data set as an SW file and writes the first three records of the file. The second overlay program references the overlay data set as an SWE file and writes the next three records of the file.

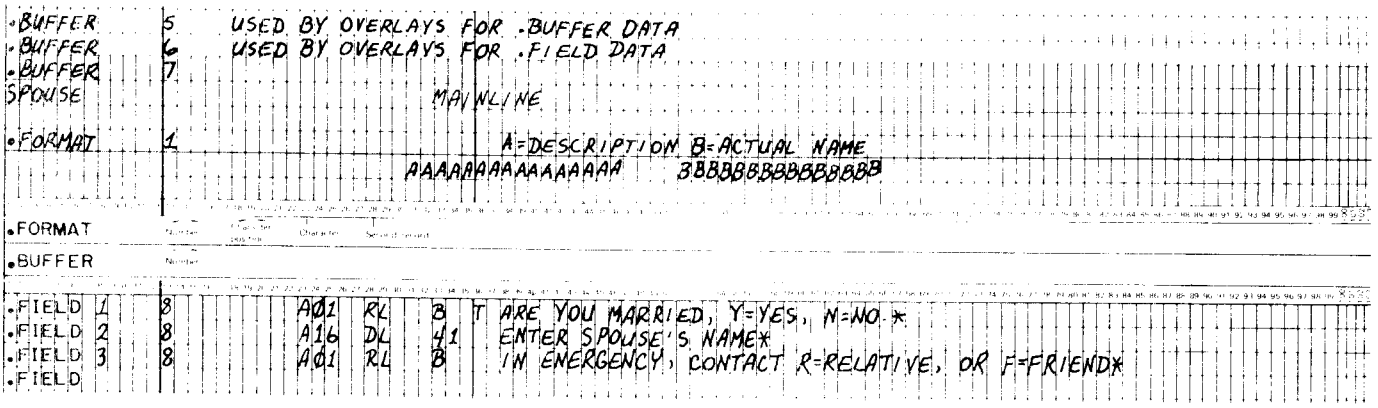
- The retrieving of the overlays from the data set is accomplished by using relative record reads.

- Overlays may degrade performance due to the reading of additional records.

The coding necessary for this program is shown in Figure 60.



```
.NAME     MAIN 9    4    66   56                  4   TRANSLAT2 MAINPG  1
.DATASET  1    OVERLAY  128  1    4                   SR
.DATASET  2
.DATASET  3
.DATASET  4
.PRINTER  4              128  3
```

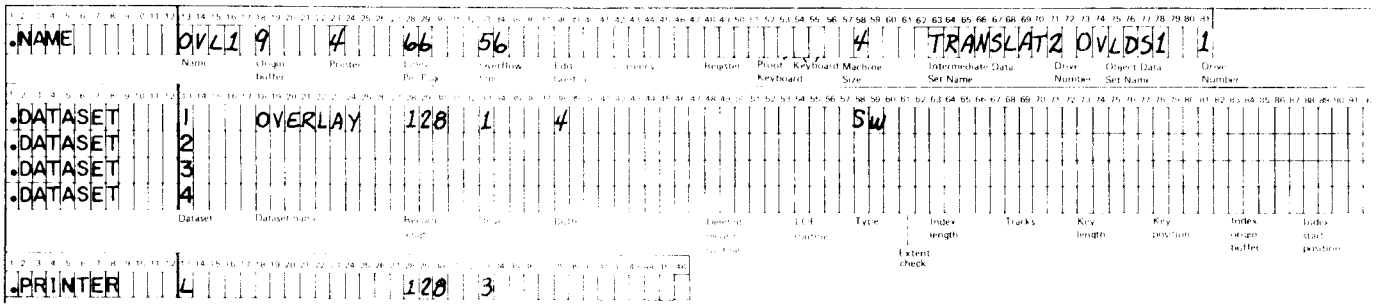


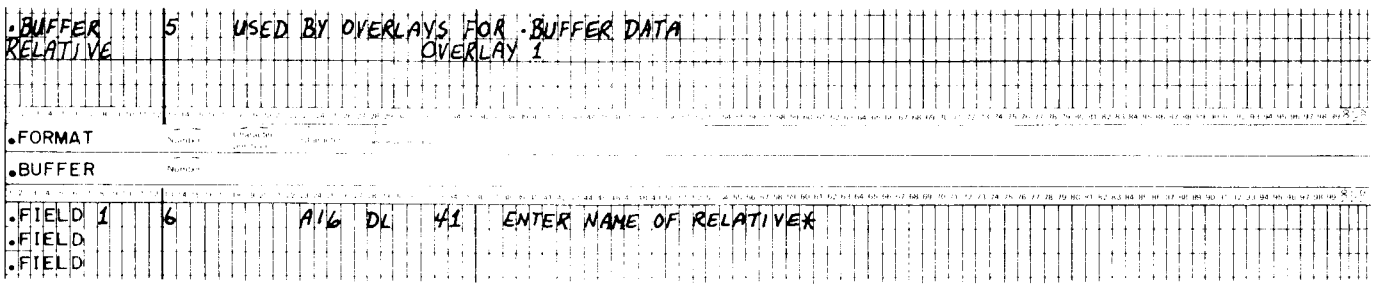
```
.BUFFER   5    USED BY OVERLAYS FOR .BUFFER DATA
.BUFFER   6    USED BY OVERLAYS FOR .FIELD DATA
.BUFFER   7
SPOUSE                MAINLINE
.FORMAT   1                    A=DESCRIPTION B=ACTUAL NAME
                      AAAAAAAAAAAAAAAA  BBBBBBBBBBBBBBBB
.FORMAT
.BUFFER
.FIELD 1  8           A01  RL  8  T ARE YOU MARRIED, Y=YES, N=NO *
.FIELD 2  8           A16  DL  41 ENTER SPOUSE'S NAME*
.FIELD 3  8           A01  RL  8  IN ENERGENCY, CONTACT R=RELATIVE, OR F=FRIEND*
.FIELD
```

```
***  MAINLINE (ROOT) PROGRAM
U03Ø      WRFM  2                          CLEAR CRT
          ENTR  8     1                    ASK IF MARRIED
          IFI   199   NOT   ON     DØ1Ø    IF RECORD ADV NOT PRESSED, GO ON
          EXIT                             END OF PROGRAM
DØ1Ø      SCE   B     16    Y              IF YES, SKIP
          GOTO                     DØ2Ø    GO ASK FOR EMERGENCY CONTACT (OVERLAY PROGRAM)
          MOVE  2           7              MOVE MAINLINE CAPTION TO CRT
          ENTR  8     2                    ASK FOR SPOUSE'S NAME
          LOAD  7     1     A      16      GET 'SPOUSE' FOR PRINT
PRNT      PRNT  D     1                    PRINT INFORMATION
          GOTO                     UØ3Ø    START OVER

*  GET THE PROPER OVERLAY
DØ2Ø      ENTR  8     3                    ASK WHO TO CONTACT IN EMERGENCY
          A     =     1                    SET INITIAL REC NO FOR 'RELATIVE'
          SCE   B     16    R              IF RELATIVE OVERLAY NEEDED
          A     =     4                    SET INITIAL REC NO FOR 'FRIEND'
          READ  1           A              READ .BUFFER OVERLAY
          MOVE  5           4              STORE .BUFFER OVERLAY
          READ  1           A              READ .FIELD OVERLAY
          MOVE  6           4              STORE .FIELD OVERLAY
          READ  1           A              READ INSTRUCTIONS OVERLAY
          MOVE  13          4              STORE INSTRUCTIONS OVERLAY
          GOTO                     OVRL    GO EXECUTE OVERLAY
          ORG   128                        SET SO NEXT INSTRUCTION IN BUFFER 13
OVRL      NOP                              TO SET UP OVERLAY BRANCH LABEL
          ORG   16Ø                        SET SO NEXT INSTRUCTION IN BUFFER 14
RETN      GOTO                     PRNT    GOTO PRINT INFORMATION
.END
```

Figure 60 (Part 1 of 3). Coding for Sample Overlay Program





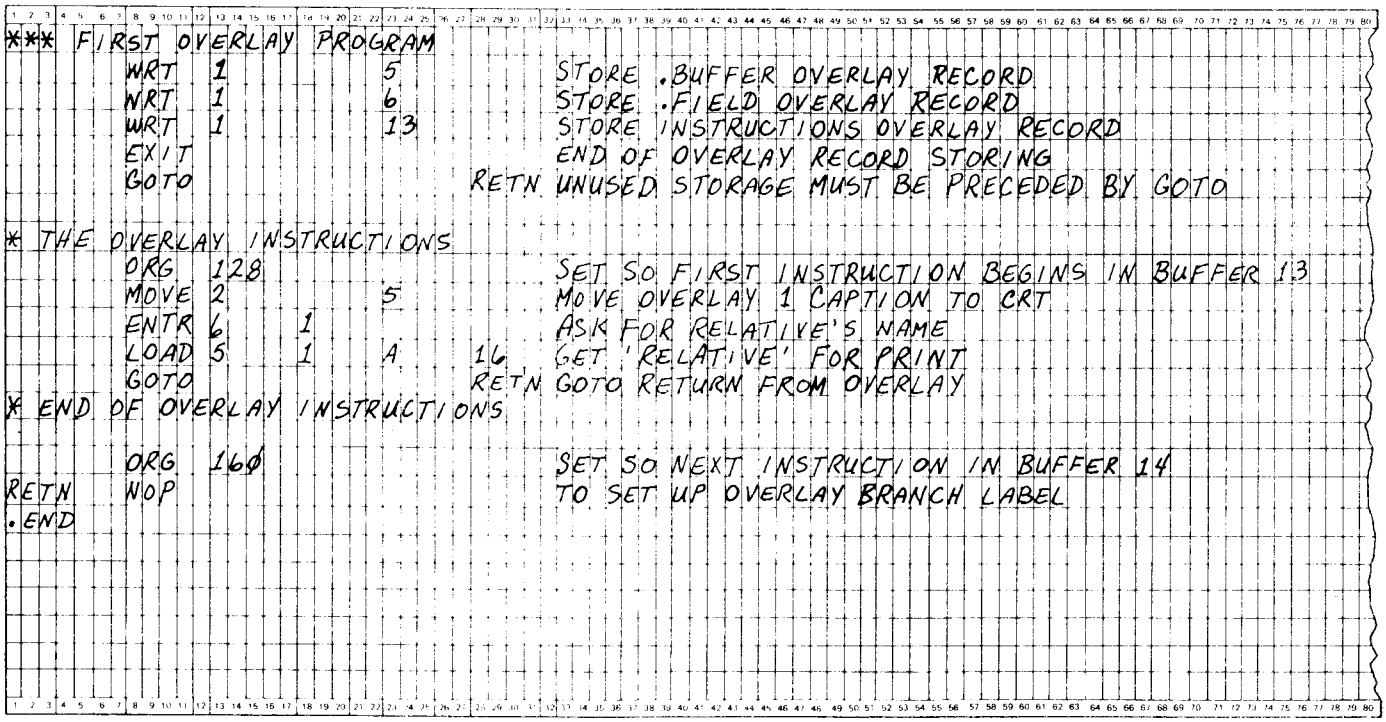
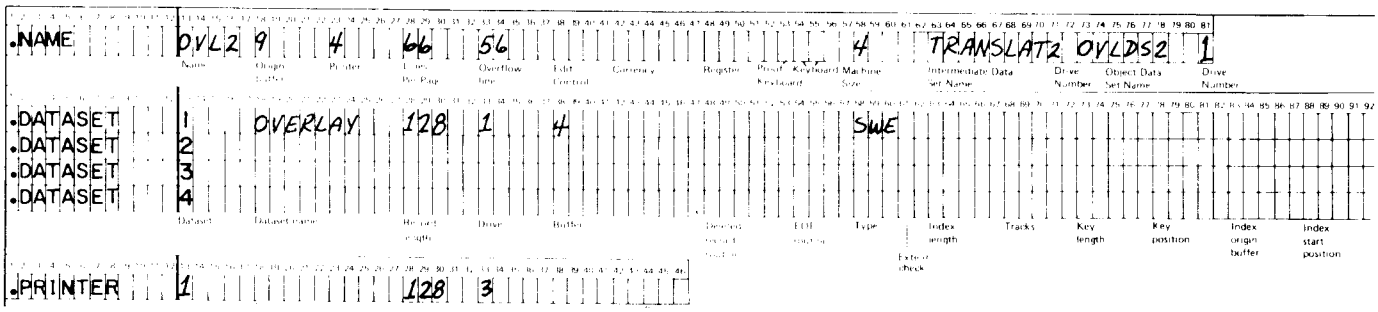
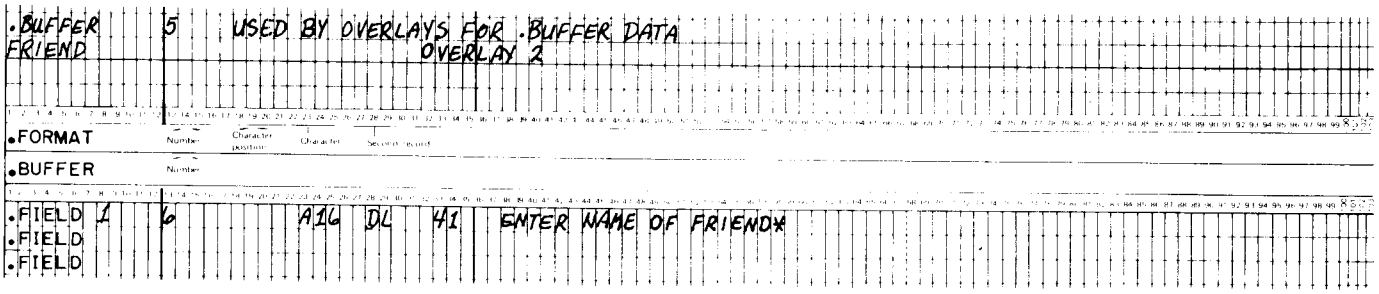


Figure 60 (Part 2 of 3). Coding for Sample Overlay Program





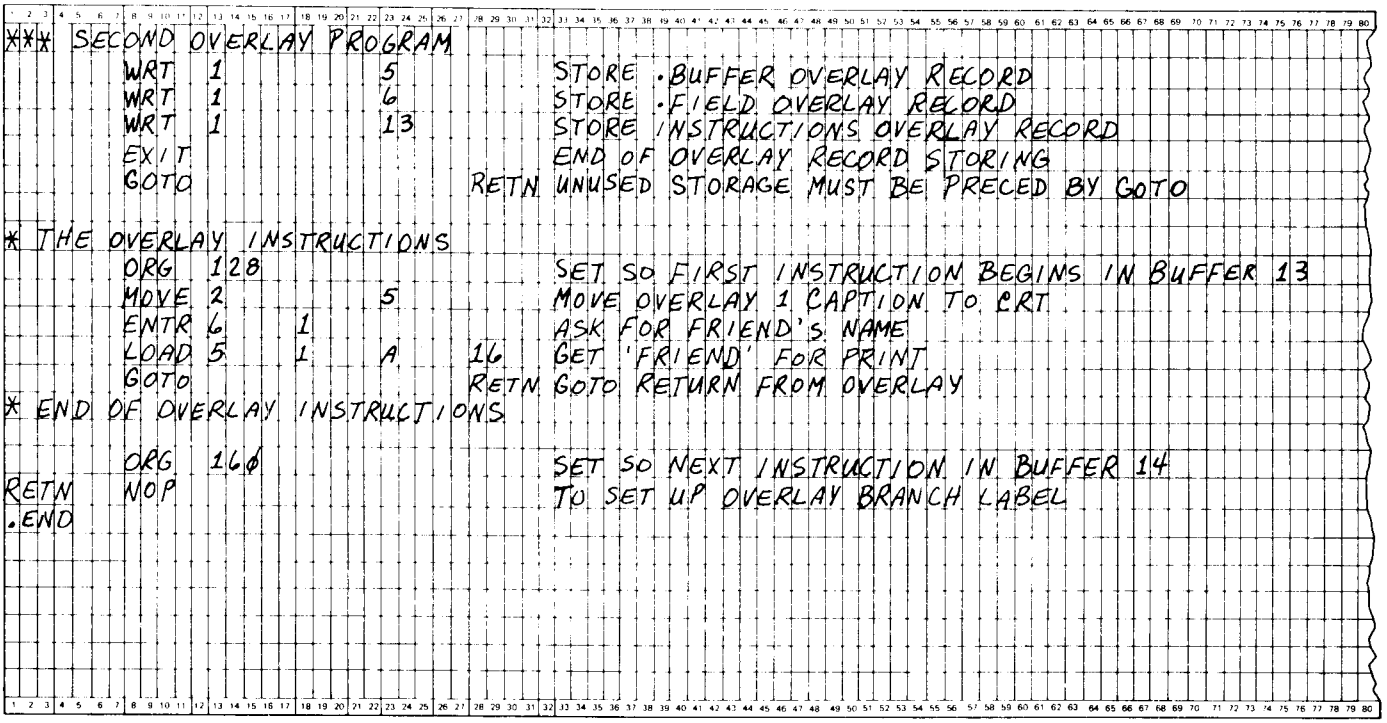


Figure 60 (Part 3 of 3). Coding for Sample Overlay Program

The Printer Link (RPQ) feature allows an ACL program to load printer control format programs into format buffers, then select PRINT TO EOD mode at the Model 2 level. At the conclusion of printing, a method is provided to return to ACL program mode at the Model 4 level.

Printer Link is applicable only to a 3741 Model 4 with the Communication Link (RPQ) feature, Expanded Communication feature, and either a 3715 or 3717 Printer attached.

Printer control format programs and the Expanded Communication feature are described in the *IBM 3741 Data Station Reference Manual,* GA21-9183. The Communication Link (RPQ) feature, the 3715 Printer, and the 3717 Printer are described in Chapter 2 of this manual.

In order to start the Model 2 PRINT TO EOD function from an ACL program you must:

- Close data sets 2, 3, and 4.
- Open data set 1 as an SR type file on drive 1.
- Load printer programs in ACL buffers 2-10 (buffer 1 cannot be used).
- Load ACL register A with print control information.
- Program the COMM instruction.
- Turn on the AUTO REC ADV switch.

The linkage to print mode begins when a COMM instruction is encountered in an ACL program. The format of this instruction is:

Column	1	8	13	18	23
Entry	Step/ Label	COMM			

The work station assumes that the input to the print mode linkage function is in register A, and that buffers 2 through 10 contain print control programs.

Register A must contain the following information before the COMM instruction is executed:

Position	Required Information
1	Not used.
2	The character D.
3	Not used.
4	Format numbers 2 through 9. Used only if buffer 10 contains a question mark.
5	The character N if the check for a continued data set is to be skipped.
6-16	Not used.

When printing is complete, the work station will return either to the index track or to ACL program mode to translate or execute a program. You must put one of the following print control characters in position 001 of buffer 10 to tell the work station which way to return.

Return Point after Print		Meaning
Index Track	**ACL Program Mode**	
Colon (:)	Question mark (?)	Printing is under control of the format program selected by byte 4 in register A.
Percent sign (%)	Exclamation point (!)	Printer format program is selected by the character in position 1 of each record.
Plus sign (+)	Left parenthesis: (	Printing is under control of data stream characters.

When the question mark, exclamation point, or left parenthesis is in position 1 of buffer 10 and either the COMM instruction is executed or PRINT TO EOD is manually selected, the work station completes PRINT TO EOD and returns to ACL program mode. The work station will either translate or execute a program, depending on the information you put in track 0, sector 3 of drive 1.

The information required to translate an ACL program is:

Position	Required Information
2-9	Step numbered source program name
10	Drive number where source data set is mounted
12-19	Object program data set name
20	Drive number where object data set is mounted
21	The character A

The information required to execute an ACL program is:

Position	Required Information
2-9	Object program name
10	Drive number where object data set is mounted
12-15	ACL program name
21	The character E

SAMPLE PROGRAM

This program illustrates sample coding required for register A, a format program in buffer 9, and the COMM instruction. It also shows the question mark in buffer 10, and sector 3 which returns the work station to ACL execute mode.

The following is a printout of the SOURCE data set:

```
.NAME        TEST 13   3
.DATASET     1    RECORD          1    11                    SP 0
.PRINTER     3                 128  04
.REGISTER    A      0 0N
.BUFFER      9
N128ACQ
.BUFFER      10
?
000     COMM
001     EXIT
.END
```

The following is a printout of sector 3:

Column 2

```
OBJECT  1 TEST     E
```

The first instruction in this program, the COMM instruction, uses the information in register A to go to Model 2 PRINT TO EOD mode. Printing is now under control of the programs in buffers 9 and 10. When printing is finished, the question mark in buffer 10 tells the work station to reload and execute the program in ACL mode by using the information in sector 3 of the index track.

This program is for a 3715 Printer, but can be used with a 3717 Printer by changing byte 23 of the .NAME statement to a 2 and byte 13 of the .PRINTER statement to a 2.

The following steps are required to use this program:

1. Label three unused diskette data sets with the following names (positions 6-13 of the index track); SOURCE (one track), OBJECT (two tracks starting on a track boundary), and RECORD.

2. Key the sample program into the SOURCE data set.

3. Enter the information to be printed into the RECORD data set.

4. Enter the following information into sector 3 of the index track:

Positions	Entry
2-7	OBJECT
10	1
12-15	TEST
20	E

5. Translate the sample program.

6. Turn on the AUTO REC ADV switch. The C09 error occurs if this switch is not on.

7. Execute the sample program.

   Execution error codes for the Printer Link (RPQ) feature are listed in Appendix C of this manual.

   Printing can be stopped by turning off the AUTO REC ADV switch. Printing could be restarted again by turning on the AUTO REC ADV switch and pressing REC ADV.

International Business Machines Corporation

General Systems Division
4111 Northside Parkway N.W.
P.O. Box 2150
Atlanta, Georgia 30301
(U.S.A. only)

General Business Group/International
44 South Broadway
White Plains, New York 10601
U.S.A.
(International)

IBM 3741 Models 3 and 4 Programmable Workstation Programming Reference Manual Printed in U.S.A. GA21-9194-3

GA21-9194-3

READER'S IMENT FORM

Please use this form only to identify publication errors or request changes to publications. Technical questions about IBM systems, changes in IBM programming support, requests for additional publications, etc, should be directed to your IBM representative or to the IBM branch office nearest your location.

Error in publication (typographical, illustration, and so on). **No reply.**

Page Number *Error*

Inaccurate or misleading information in this publication. Please tell us about it by using this postage-paid form. We will correct or clarify the publication, or tell you why a change is not being made, provided you include your name and address.

Page Number *Comment*

Note: All comments and suggestions become the property of IBM.

Name _____

Address _____

_____

● No postage necessary if mailed in the U.S.A.

IBM 3741 Models 3 and 4
Programmable Work Station
Programming Reference
Manual

GA21-9194-3

Cut Along Line

IBM 3741 Models 3 and 4 Programmable Work Station Programming Reference Manual Printed in U.S.A. GA21-9194-3

Fold

Fold

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE:

IBM Corporation
General Systems Division
Development Laboratory
Publications, Dept. 245
Rochester, Minnesota 55901

Fold

Fold

IBM®

International Business Machines Corporation

General Systems Division
4111 Northside Parkway N.W.
P.O. Box 2150
Atlanta, Georgia 30301
(U.S.A. only)

General Business Group/International
44 South Broadway
White Plains, New York 10601
U.S.A.
(International)

READER'S ,MENT FORM

Please use this form ».. ' to identify publication errors or request changes to publications. Technical questions about IBM systems, changes in IBM programming support, requests for a(ditional publications, etc, should be directed to your IBM representative or to the IBM branch office nearest your location.

Error in publication (typographical, illustration, and so on). **No reply.**

Page Number *Error*

Inaccurate or misleading information in this publication. Please tell us about it by using this postage-paid form. We will correct or clarify the publication, or tell you why a change is not being made, provided you include your name and address.

Page Number *Comment*

Note: All comments and suggestions become the property of IBM.

Name _____

Address _____

_____

● No postage necessary if mailed in the U.S.A.

IBM 3741 Models 3 and 4
Programmable Work Station
Programming Reference
Manual

GA21-9194-3

Cut Along Line

IBM 3741 Models 3 and 4 Programmable Work Station Programming Reference Manual Printed in U.S.A. GA21-9194-3

Fold

Fold

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE:

IBM Corporation
General Systems Division
Development Laboratory
Publications, Dept. 245
Rochester, Minnesota 55901

Fold

Fold

IBM®

International Business Machines Corporation

General Systems Division
4111 Northside Parkway N.W.
P.O. Box 2150
Atlanta, Georgia 30301
(U.S.A. only)

General Business Group/International
44 South Broadway
White Plains, New York 10601
U.S.A.
(International)

READEF OMMENT FORM

Please use this form only to identify publication errors or request changes to publications. Technical questions about IBM systems, changes in IBM programming support, requests for additional publications, etc, should be directed to your IBM representative or to the IBM branch office nearest your location.

Error in publication (typographical, illustration, and so on). **No reply.**

Page Number *Error*

Inaccurate or misleading information in this publication. Please tell us about it by using this postage-paid form. We will correct or clarify the publication, or tell you why a change is not being made, provided you include your name and address.

Page Number *Comment*

Note: All comments and suggestions become the property of IBM.

Name _____

Address _____

_____

● No postage necessary if mailed in the U.S.A.

IBM 3741 Models 3 and 4
Programmable Work Station
Programming Reference
Manual

GA21-9194-3

Cut Along Line

IBM 3741 Models 3 and 4 Programmable Work Station Programming Reference Manual Printed in U.S.A. GA21-9194-3

Fold

Fold

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 40 ARMONK, N.Y.

POSTAGE WILL BE PAID BY ADDRESSEE:

IBM Corporation
General Systems Division
Development Laboratory
Publications, Dept. 245
Rochester, Minnesota 55901

Fold

Fold

IBM

International Business Machines Corporation

General Systems Division
4111 Northside Parkway N.W.
P.O. Box 2150
Atlanta, Georgia 30301
(U.S.A. only)

General Business Group/International
44 South Broadway
White Plains, New York 10601
U.S.A.
(International)